Sukhrobbek Ilyosbekov

**Project Description**

Project “Logistics” is aimed to automate transportation management systems (TMS). TMS is a software application designed to manage and optimize inbound and/or outbound transportation operations. The client part of the project consists of an administrator web application, a management web application, and a driver mobile application. The backend consists of REST API and Identity Server applications. The project was designed as multi-tenant architecture. There are two types of databases: main and per-tenant databases. The main database stores user credentials, and tenant data (company name, subdomain name, database connection string, and billing periods). Each tenant (company or organization) has its own database.

Client frontend applications:

1. Administrator application – web application that manages tenant related operations such as creating a new tenant, managing tenant billings. Built using the Blazor frontend framework, source code located in the folder **src/Client/Logistics.AdminApp**
2. Office application – web application that manages company trucks, cargoes, employees, and reports, built using the Angular framework, source code located in the folder **src/Client/Logistics.OfficeApp**
3. Driver application – mobile application in which the driver can track their income and active deliveries, built in .NET MAUI; source code located in the folder **src/Client/Logistics.DriverApp**

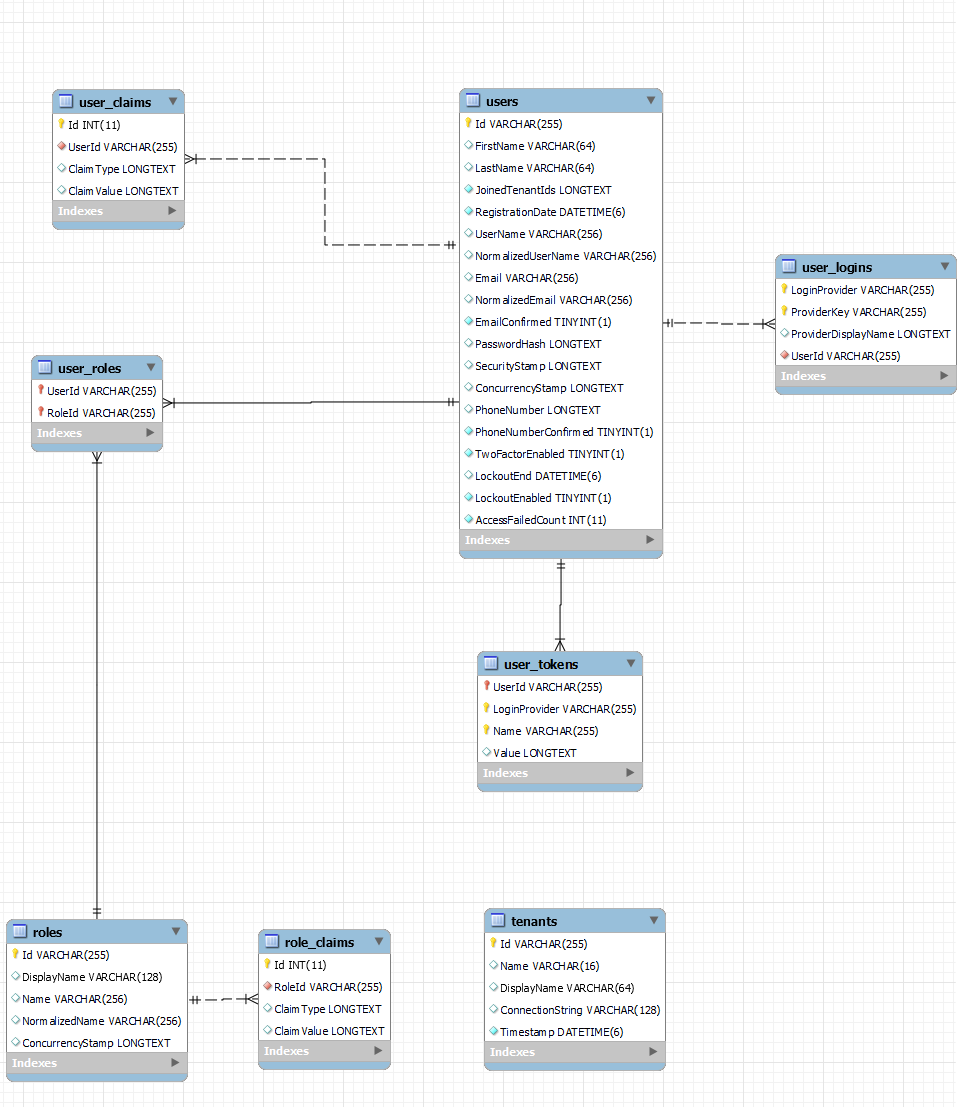
Backend applications:

1. REST API application – handles all HTTP requests from client applications, built using the ASP.NET Core, source code located in the folder **src/Server/Logistics.API**
2. Identity Server application – web application built on top of OpenID protocol, manages user authentication, authorization, and token issuing flows, built using the ASP.NET Core, source code located in the folder **src/Server/Logistics.IdentityServer**

Source code repository: <https://github.com/suxrobgm/logistics-app>

**ER Diagrams**

Main database:



Tenant database:

Diagram

Description automatically generated

**Database schemas**

Main database:

Users(Id, FirstName, LastName, JoinedTenantIds, RegistrationDate, UserName, NormalizedUserName, Email, EmailConfirmed, PasswordHash, SecurityStamp, ConcurrencyStamp, PhoneNumber, PhoneNumberConfirmed, TwoFactorEnabled, LockoutEnd, LockoutEnabled, AccessFailedCount)

UserClaims(Id, UserId, ClaimType, ClaimValue)

UserRoles(UserId, RoleId)

Roles(Id, DisplayName, Name, NormalizedName, ConcurrenyStamp)

RoleClaims(Id, RoleId, ClaimType, ClaimValue)

UserTokens(UserId, LoginProvider, Name, Value)

UserLogins(LoginProvider, ProviderKey, ProviderDisplayName, UserId)

Tenant database:

Loads(Id, RefId, Name, SourceAddress, DestinantionAddress, DeliveryCost, Distance, DispatchedDate, PickUpDate, DeliveryDate, Status, AssignedDispatcherId, AssignedDriverId, AssignedTruckId)

Trucks(Id, TruckNumber, DriverId)

Employees(Id, JoinedDate)

EmployeeRoles(EmployeeId, RoleId)

Roles(Id, DisplayName, Name, NormalizedName, ConcurrenyStamp)

RoleClaims(Id, RoleId, ClaimType, ClaimValue)

**Demo screenshots**

Office application:

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Graphical user interface, application

Description automatically generated

Chart

Description automatically generated

Chart

Description automatically generated

Graphical user interface, table

Description automatically generated

Admin application:

Graphical user interface

Description automatically generated with low confidence

Identity Server application:

Graphical user interface, application

Description automatically generated

How to open the demo application?

1. Open the URL: <https://office.jfleets.org>
2. Click the Login button
3. Enter admin credentials in the login page:  
   Email: suxrobgm@gmail  
   Password: Test12345#