**Documentation of**

**Assignment 1**

**Request-Reply Communication**

Name: Onea Vlad

Group: 30442

1. **Conceptual architecture of the distributed system**

**Backend:** For the backend I created 2 microservices: EnergyManagementUser and EnergyManagementSystem for both of them I used .NET. The microservices are done using layered architecture so I divided the projects in more packages: Controllers, Data, Dtos. Migrations and Services. Controllers: handle incoming requests, interact with the application's business logic, and coordinate the flow of data between the user interface and the underlying services. **Data:** refers to the data access layer, which is responsible for interacting with the database. It often includes entities or models that represent the data structure and a repository pattern for performing CRUD (Create, Read, Update, Delete) operations. **DTOs (Data Transfer Objects):** DTOs are objects used to transfer data between layers of an application. They’re used to encapsulate data and send it between the client and server and between different layers of the application. **Migrations:** Migrations are files that define changes to the database schema over time**. Services:** Services typically contain the business logic of the application. They encapsulate the functionality that doesn't belong directly in the controllers or data access layer.

1.1 **User Management:** In this project, the focus was on managing the users. User class contains the attributes of interests for the user. UserService is responsible for CRUD operations on user and it integrates UserDataContext to interact with the database and IConfiguration interface for accessing secret key used for JWT token generation. AddUser() and AddManager() add a new user to the system, either with User or Admin role so they are responsible for CREATE. GetAllUsers() method retrieves a list of users from database and maps them to specific attributes from GetAllUsersResponseDto (id, username,email, role, phoneNumber) so this method is responsible for READ. UpdateUser() method finds a user in the database by id and updates properties based on UserDtoUpdate (username, phoneNumber, email,role) so this method is responsible for UPDATE. DeleteUser() method finds a user in the database by id and removes it from database so this method is responsible for DELETE. CreateToken() method generates a JWT (JSON Web Token) for a given user and creates a list of claims (user information), defines a security key, sets token parameters (subject, expiration, signing credentials), and uses a JwtSecurityTokenHandler to create and write the token. CreatePasswordHash() method takes a plain-text password, computes its SHA-256 hash, and returns the hashed password in a hexadecimal format. In Program.cs I added JWT Bearer Authentication as the default authentication scheme, I configured CORS policy. HttpClientService class is responsible for making post and delete request to user table associated in device database so it basically when a user is added or deleted it performs action on users table from devices too. In the AuthController I used authorization to restrict access to certain endpoints based on user role. In this class I implemented all the endpoints including that one when I send user information to device api. Here I used the methods presented in UserService and in HttpClientService.

1.2 **Device Management:** In this project, the focus was on managing the devices. Device class contains the attributes of interest for the device and User class contains an id and username.UserId from device table is foreign key for id from User table. DeviceService is responsible for CRUD operations on device, and it integrates DeviceDataContext to interact with the database. AddDevice() methods adds a new device to the database based on information provided in DeviceDto, it checks if a device has an user and retrieves the corresponding user from database and creates a new device object with provided details and sets the owner’s id so this method is responsible for CREATE. DeleteDevice() method checks if the device exists and deletes the device from the database so it is responsible for DELETE. GetAllDevices() method retrieves a list of all devixes from database along with some user information and returns a list of DTOs. GetUserDevices() method retrieves a list of devices owned by a specific user based on the user’s id so it is responsible for READ. The last method UpdateDevice() method updates the details of a device in the database based on the provided device ID and DeviceDto so it is responsible for UPDATE. In this microservice I also have DeviceController I implemented the endpoints with the help of DeviceService and UserController which manages user-related http requests

1.3 **Frontend**: I decided to write the frontend in Angular and divided it into more packages. All the requests to backend are done in user service and device service. In user service I have getCurrentUser, signup, login, getAllUsers, update and delete responsible for CRUD operations on user. In the device service I have add, update, getAllDevices, getUserDevice and delete, methods responsible for CRUD operations for devices. Another service which I used is snackbar service which is responsible for displaying the messages when a CRUD operation is done.

Home package which is the first page that will appear on my application. Here I added 2 buttons: one for login and one for signup which if someone presses them will appear a form. The form for signup is done in signup package and the user should introduce the username, password, email and phoneNumber, these fields are presented in models in the interface IUserRegisterRequest. When the user is ready to create the account, he should press signup button which is based on signup method from user service. The form for login is done in login package where I have 2 fields: username and password which should be introduced by the user. After the completion of the fields, the user should press login button, and this click will trigger the method login used in user service. This will redirect the user to userdashboard where the user can see all associated devices (getUserDevice method is called). Here the user can logout, the click of the button will open the confirmation component which will ask the user if you are sure that you want to perform the action. Another feature which will appear on this page is the security: only the admin can click manage user and manage devices. This is done in helpers package: manager.guard. If the user login as admin he will manage the users and devices. These are done in manage-device and manage-user component and when the admin decided to add or edit a device a form will be opened, action which is done in device component and will call the add or update method from device service. Almost the same is done in user component, but the difference is that the admin can only edit a user, the creation of the user was done when he registers.

Another package which I used is shared package where I have the GlobalConstants class where I stored some messages and some regexes for name, telephoneNumber, email and role.

1. **UML Deployment Diagram**

I used Docker for the deployment of the application. Web app, services and database are deployed on containers. Web app is using port 4200, databases are using port 5432, user microservice port 5000 and device microservice port 5050.

A diagram of a computer network

Description automatically generated