TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA

DISTRIBUTED SYSTEMS
Assignment 1

Request-Reply
Communication
2023-2024

Student: Alexandra Ilovan
Group: 30442
Lab Teacher: Andreea Oana Marin

I. Introduction

This project focuses on developing an Energy Management System comprising a user-friendly frontend and two microservices dedicated to managing users and smart energy metering devices. The system caters to two distinct user roles: administrators (managers) and clients. Upon successful login, users are directed to role-specific pages, ensuring a tailored and secure user experience.

Administrators are equipped with CRUD operations over user accounts (ID, name, role), smart energy metering devices (ID, description, address, maximum hourly energy consumption), and the mapping of users to devices. This mapping feature allows users to own multiple smart devices across various locations, enhancing flexibility and customization.

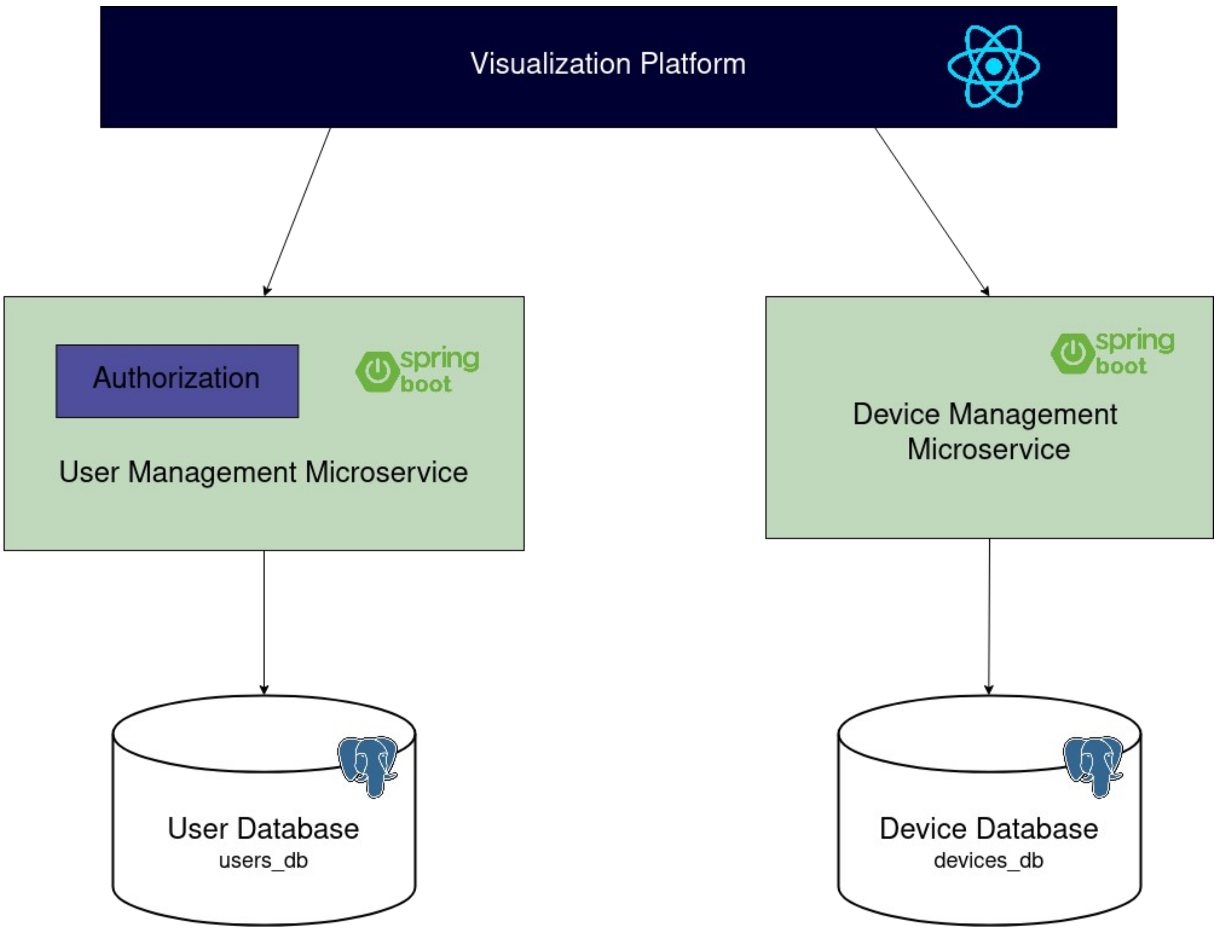
The implementation employs cutting-edge technologies, with Spring Boot chosen for the microservices and PostgreSQL as the database engine. The frontend is developed using React JS, providing a dynamic and responsive user interface for seamless interaction.

The microservices architecture includes the User Management Microservice and the Device Management Microservice. These components enable the efficient execution of CRUD operations, ensuring the smooth functioning of the Energy Management System.

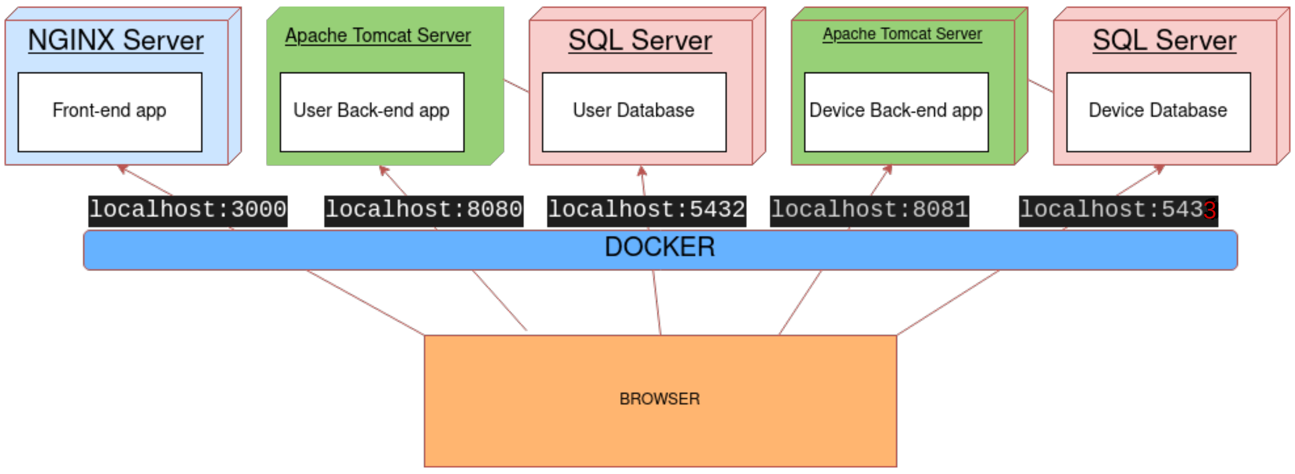
Security is a top priority, with authentication mechanisms implemented using sessions, and other tools. This approach guarantees that only authorized users can access administrator pages, safeguarding sensitive information and maintaining the overall integrity of the system.

II. System Architecture

2.1. Conceptual Architecture



2.2. UML Deployment Diagram



III. Interfaces

As mentioned above, the system is comprised of two main microservices, one in charge of managing the users and authorization part of the system, and the second one is used for device management. In the following lines, I will provide a short description about the functionalities provided by each microservice.

3.1. User Management Microservice

The User Management Component serves as the backbone for handling user-related functionalities within the Energy Management System. It is implemented as a microservice using Spring Boot, providing a reliable and scalable solution for managing user accounts.

Functionalities:

- **User CRUD Operations:** The User Management Component facilitates the creation, retrieval, updating, and deletion of user accounts. The User entity is defined through the following fields: UUID, name, email, password, and role (admin/client).

- **Authentication and Authorization:** Upon successful login, the component issues a JWT (JSON Web Token) to the user. This token serves as a secure and stateless mechanism for subsequent authorization. Admin users, identified by their role in the JWT, gain the privilege to perform CRUD operations on both users and devices.

- **Security Measures:**The User Management Component employs robust security measures, including secure password storage, JWT-based session management, and encryption protocols, ensuring the integrity and confidentiality of user data.

3.2. Device Management Microservice

The Device Management Component complements the system by overseeing the creation, modification, and deletion of smart energy metering devices. Like the User Management Component, it operates as a microservice developed using Spring Boot.

Functionalities:

- **Device CRUD Operations:** This component manages the lifecycle of smart energy metering devices, capturing information such as ID, description, address, and maximum hourly energy consumption.

- **User-Device Linkage in the Database:** The linkage between users and devices is established at the database level. The devices table includes a userId column, forming a direct association between each device and its owner.

- **Interactions with User Management:** The Device Management Component interacts with the User Management Component to validate user permissions and ensure that only authorized administrators, validated through JWT tokens, can perform device-related CRUD operations.

3.3. User-Device Mapping

The association between users and devices is realized through meticulous design at both the database and frontend levels. The devices table's userId column establishes a direct link between each device and its respective owner. Additionally, the frontend, powered by React JS, orchestrates user-device mapping through well-defined API calls to the User and Device Management Components, presenting the JWT token for secure and streamlined authorization. This approach ensures a coherent and efficient management of the intricate relationships between users and their smart energy metering devices.

IV. Implementation and Testing

4.1. User Management Microservice

User Entity

The User entity represents the core structure of a user in the system. It utilizes Java's Persistence API (JPA) for database mapping and extends UserDetails for Spring Security integration. Key attributes include:

- userId: A universally unique identifier (UUID) generated using Hibernate's uuid2 strategy.
- name, email, password: Basic user information with email as the unique identifier.
- role: Represents the user's role in the system.

The entity also implements methods from UserDetails to support Spring Security, providing details about user authentication and authorization.

User DTOs

DTOs (Data Transfer Objects) are used to facilitate communication between the UserController and the UserService. These include UserDto and UpdateUserDto.

UserController

The UserController is responsible for handling HTTP requests related to user management. Key functionalities include:

- **User Retrieval:**
    - getUsers: Retrieve a list of all users.
    - getUsersByRole: Retrieve users by a specified role.
    - getUserById: Retrieve a user by their UUID.
    - getUserByEmail: Retrieve a user's UUID by their email.
- **User Modification:**
    - updateUser: Update user information based on the provided UpdateUserDto.
    - deleteUser: Delete a user by their UUID.

Security annotations such as @PreAuthorize ensure that only authorized users can access certain endpoints.

AuthService

The AuthService handles user registration and authentication. Key features include:

- **User Registration:**
    - register: Validates registration data, checks for existing users, encodes the password using Bcrypt, and generates a JWT token.

- **User Authentication:**
    - authenticate: Authenticates users based on provided credentials and generates a JWT token upon successful authentication.

Security Measures

Security measures include the use of Bcrypt for password encoding and JWT for authorization. The JWTAuthenticationFilter and JwtService work together to secure endpoints and generate JWT tokens.

Exceptions

Two custom exceptions, UserNotFoundException and UserAlreadyExistsException, handle scenarios where a user is not found or already exists during registration.

4.2. Device Management Microservice

Device Entity

The Device entity defines the core structure of a device in the system, utilizing JPA for database mapping. Key attributes include a UUID-generated deviceId, a description, address, and maximum hourly energy consumption. A nullable field, userId, associates the device with a user.

Device DTOs

DTOs, including CreateDeviceDto and DeviceDto, facilitate communication between the DeviceController and DeviceService.

DeviceController

The DeviceController handles HTTP requests related to device management:

- **Device Retrieval:**
    - getDevices: Retrieve a list of all devices.
    - getDeviceById: Retrieve a device by its UUID.
    - getDevicesByUser: Retrieve devices associated with a specific user.
- **Device Modification:**
    - addDevice: Create and add a new device based on the provided CreateDeviceDto.
    - updateDevice: Update device information based on the provided CreateDeviceDto.
    - deleteDevice: Delete a device by its UUID.

DeviceService

The DeviceService provides the business logic for device management, including creating, updating, and deleting devices.

Device Not Found Exception

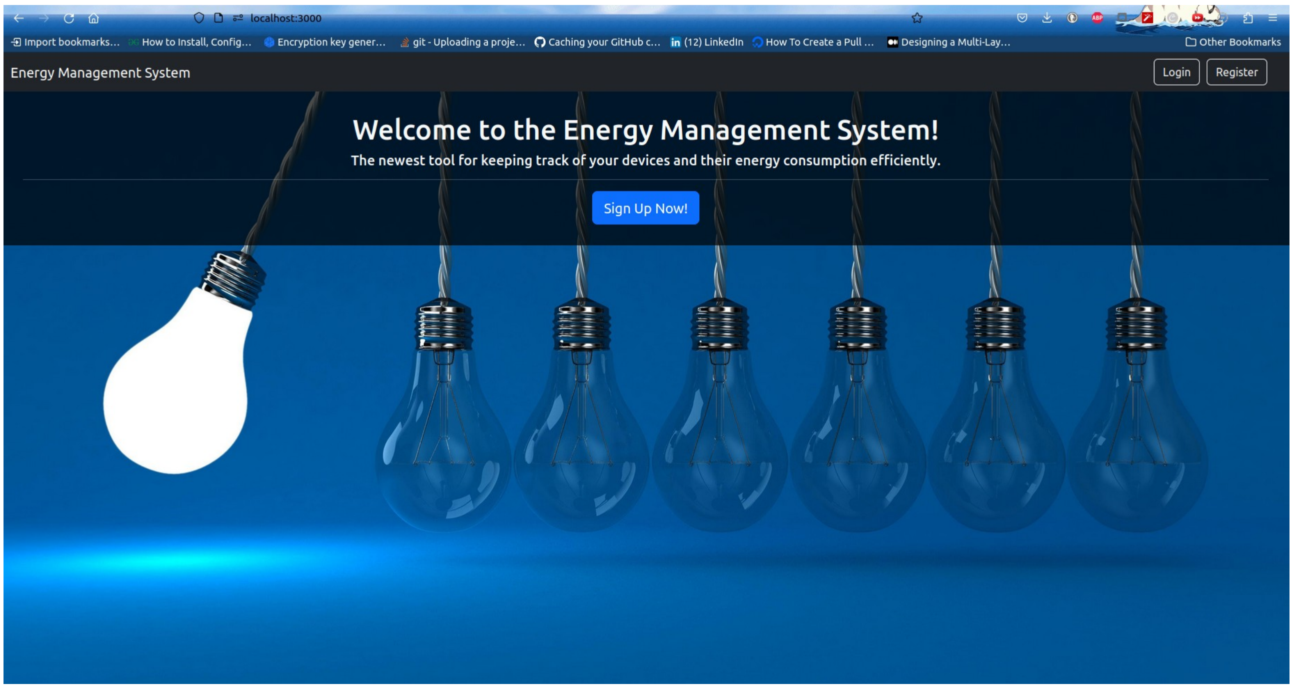
A custom exception, DeviceNotFoundException, handles scenarios where a device is not found during operations.

4.3. Testing

I used Postman for streamlined API testing and Swagger for clear and interactive API documentation for both microservices.

V. Results

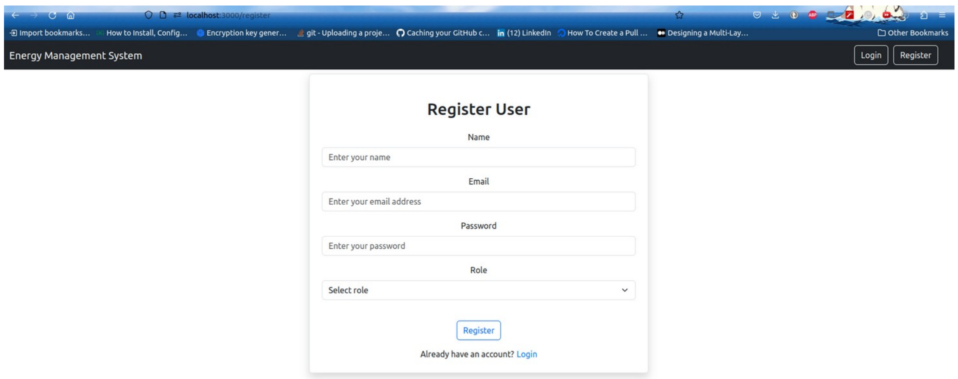
5.1. Dashboard page

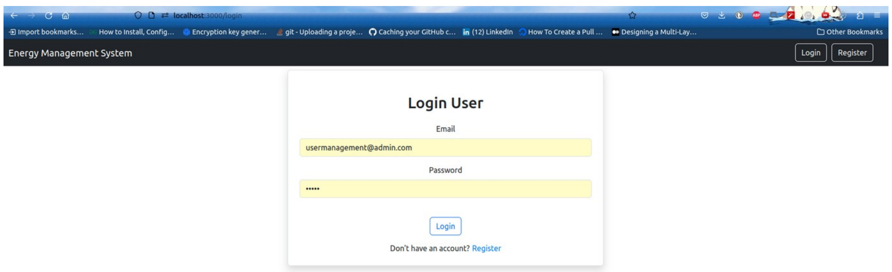


Welcome to the Energy Management System!

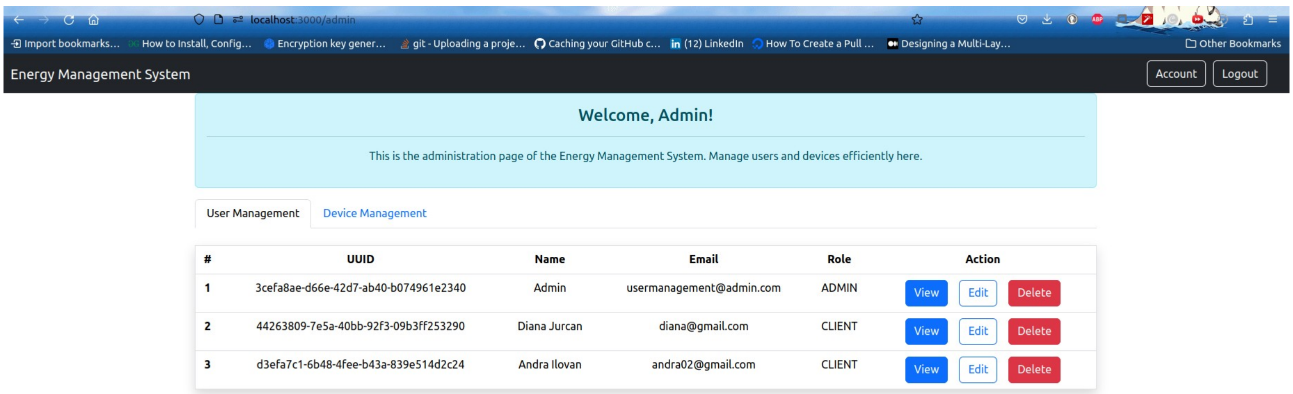
The newest tool for keeping track of your devices and their energy consumption efficiently.

5.2. Register and Login pages





5.3. Admin page



Energy Management System

Account Logout

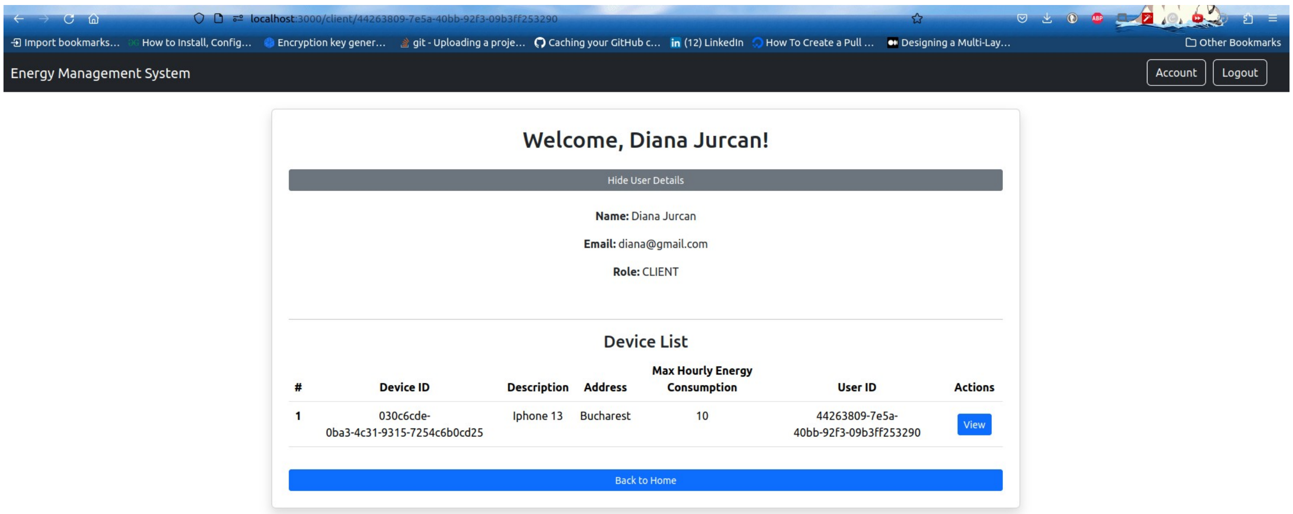
Welcome, Admin!

This is the administration page of the Energy Management System. Manage users and devices efficiently here.

User Management | Device Management

#	UUID	Name	Email	Role	Action
1	3cefa8ae-d66e-42d7-ab40-b074961e2340	Admin	usermanagement@admin.com	ADMIN	View Edit Delete
2	44263809-7e5a-40bb-92f3-09b3ff253290	Diana Jurcan	diana@gmail.com	CLIENT	View Edit Delete
3	d3efa7c1-6b48-4fee-b43a-839e514d2c24	Andra Ilovan	andra02@gmail.com	CLIENT	View Edit Delete

5.4. Client page



Energy Management System

Account Logout

Welcome, Diana Jurcan!

Hide User Details

Name: Diana Jurcan

Email: diana@gmail.com

Role: CLIENT

Device List

#	Device ID	Description	Address	Max Hourly Energy Consumption	User ID	Actions
1	030c6cde-0ba3-4c31-9315-7254c6b0cd25	Iphone 13	Bucharest	10	44263809-7e5a-40bb-92f3-09b3ff253290	View

Back to Home