# Backend Documentation for User Management and Device Service

# 1. User Management Service

Purpose:

Manages user data (registration, updates, and deletion). It also integrates with the Device Service to remove associated devices when a user is deleted.

Key Endpoints:

* POST /api/users: Create a new user.
* GET /api/users/{id}: Get a user by ID.
* GET /api/users: Get all users.
* PUT /api/users/{id}: Update user details.
* DELETE /api/users/{id}: Delete a user and all associated devices.

User Model:

O imagine care conține text, captură de ecran, Font

Descriere generată automat

Service Implementation:

The UserServiceImpl service provides the CRUD functionality, and integrates with a device microservice to delete associated devices when a user is deleted.

O imagine care conține text, captură de ecran, Font

Descriere generată automat2. Device Management Service

Purpose:

Manages devices related to users. Supports operations like creating, updating, fetching, and deleting devices. Devices can be associated with users via userId.

Key Endpoints:

POST /api/admin/devices: Create a new device.

GET /api/admin/devices: Get all devices.

GET /api/admin/devices/{id}: Get a device by ID.

PUT /api/admin/devices/{id}: Update a device.

DELETE /api/admin/devices/{id}: Delete a device.

GET /api/admin/devices/user/{userId}: Get all devices by user ID.

DELETE /api/admin/devices/user/{userId}: (Commented out) Delete all devices associated with a user.

Device Model:

@Entity

@Table(name = "devices")

public class Device {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private long userId;

private String description;

private String address;

private double maxHourlyConsumption;

}

Service Implementation:

The DeviceService handles CRUD operations for devices and interacts with a database using Spring Data JPA.

public class DeviceService {

// CRUD methods for devices

public List<Device> getDevicesByUserId(Long userId) {

// Fetch devices based on user ID

}

}

3. Docker Setup

Dockerfile:

Multi-stage Build: One stage for building the app using Maven (mvn clean package), and another for running the app using OpenJDK.

Expose Ports:

8081 for the User Service.

8080 for the Device Service.

3000 for the Frontend.

Docker Compose:

The Docker Compose file defines services for:

MySQL databases for user and device storage.

User Service and Device Service applications.

Frontend React application.

services:

mysql-db:

image: mysql:8.0

environment:

MYSQL\_DATABASE: devicespsd

MYSQL\_ROOT\_PASSWORD: 12345

ports:

- "3307:3306"

user-management-app:

build:

context: "./demo"

depends\_on:

- mysql-user-db

environment:

SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql-user-db:3306/userspsd

ports:

- "8081:8081"

device-service-app:

build:

context: "./demo(1)/"

depends\_on:

- mysql-db

environment:

SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql-db:3306/devicespsd

ports:

- "8080:8080"

4. Integration and Flow

User Service and Device Service are separate microservices that communicate over HTTP (through REST API calls).

The User Service handles CRUD operations for users.

The Device Service manages devices associated with users. When a user is deleted, the service ensures that the associated devices are also deleted.

The frontend connects to the backend via routes like /page2/devices/userId/{userId} to fetch devices associated with a specific user.

5. MySQL Configuration

Both the User Service and Device Service connect to their respective MySQL databases using the following environment variables:

MYSQL\_USER: user

MYSQL\_PASSWORD: 12345

MYSQL\_DATABASE: userspsd for User Service, devicespsd for Device Service.

6. Known Issues and Considerations

Device Deletion on User Deletion: The deletion of devices associated with a user requires the device-service to properly handle cascading deletions.

Security: Passwords are not hashed, which is not recommended for production. Ensure to hash passwords using tools like Spring Security in the final version.

Error Handling: Some services may need more robust error handling (e.g., invalid user ID or device not found).