**SMART METERING DEVICES APPLICATION**

**ASSIGNMENT 1**

**TAT TEODORA**

**GROUP 30411**

**Project Overview:**

The microservices architecture encompasses two distinct and independently deployable services:

User Microservice:

* Orchestrates user data operations.
* Developed leveraging the Spring Boot framework with Java.

Device Microservice:

* Manages device-related functionalities.

- Also constructed utilizing the Spring Boot framework with Java.

**In the provided architectural overview, we have developed a robust microservices application focusing on user and device management, harmoniously integrated through synchronous RESTful APIs. Let's delve into the architectural components in more detail.**

**1. User Microservice:**

**Endpoints:**

* /users (GET, POST): Manages user information retrieval and creation.
* /users/{id} (GET, PUT, DELETE): Handles individual user details.

**Technology Stack:**

Spring Boot with Java: Ensures the efficiency and reliability of the User Microservice.

**Architecture Structure:**

**Entity:** Represents the User entity, a fundamental component mapped by DTOs for seamless communication.

**Repository**: Manages data access and persistence.

**Service**: Contains business logic and acts as a bridge between the repository and controller.

**Controller:** Orchestrates the interaction with external systems via RESTful APIs**.**

**2. Device Microservice:**

**Endpoints:**

* /devices (GET, POST): Oversees device information retrieval and creation.
* /devices/{id} (GET, PUT, DELETE): Governs individual device details.

**Technology Stack:**

Spring Boot with Java: Ensures the efficiency and reliability of the Device Microservice.

**Architecture Structure:**

**Entity**: Represents the Device entity, including a user association, facilitating a comprehensive device management system.

**Repository:** Manages data access and persistence**.**

**Service:** Contains business logic and acts as a bridge between the repository and controller**.**

**Controller:** Orchestrates the interaction with external systems via RESTful APIs.

**3. Frontend: React**

**Technology Stack:**

**React**: A powerful and flexible JavaScript library for building user interfaces.

**Architecture Structure:**

**UI Components:** Developed using React, providing a responsive, dynamic, and visually pleasing user experience.

**4. Inter-Service Communication:**

**Communication Protocol:**

**HTTP:** The Microservices communicate synchronously via RESTful APIs, ensuring efficient integration.

**5. Real-Time Interaction:**

**Features:**

**Real-time Communication**: Microservices orchestrate real-time communication, enabling instantaneous updates and seamless data exchange, using RESTful APIs.

**6. Development Tools:**

**IDEs:**

**IntelliJ IDEA**: Employed for Microservices development and React development, ensuring a streamlined and efficient development process.

**7. UML Deployment Diagram:**

A visual representation of the deployment architecture, illustrating the relationships and interactions between microservices, databases, and the frontend.

Conceptual architecture of the distributed system:

A screenshot of a computer

Description automatically generated

UML Deployment diagram:

A diagram of a computer program

Description automatically generated with medium confidence

**10. Conclusion:**

In conclusion, the Microservices Project for Users and Devices is an exemplary showcase of simplicity and functionality. This comprehensive documentation provides a clear roadmap for understanding and contributing meaningfully to the project. The choice of technologies, architecture, and development tools reflects a thoughtful approach to creating a dynamic and responsive application.