**Smart Home Automation API with Device Control**

**OVERVIEW:**

Create a Spring Boot REST API for smart home automation that allows users to control various IoT devices in their home. Implement a one-to-many relationship between users and devices, enabling users to manage multiple devices and perform actions such as turning them on/off or adjusting settings remotely.

**FUNCTIONAL REQUIREMENTS:**

* Develop an application for smart home automation.
* Create a folder named "Controller" inside the src directory.
* Inside the Controller folder, create a Java file named "HomeController.java" to handle API requests related to users and devices.
* Create another folder named "Models" within the src directory.
* Inside the Models folder, create Java files named "User.java" and "Device.java" to represent user and device entities, respectively.
* Define the attributes for the User entity:
* id: int (user ID)
* name: String (name of the user)
* email: String (email address of the user)
* Define the attributes for the Device entity:
* id: int (device ID)
* name: String (name of the device)
* type: String (type of the device, e.g., light, thermostat, security camera)
* status: boolean (status of the device, true for on, false for off)
* settings: Map<String, String> (settings/configuration for the device)
* Implement the one-to-many relationship between User and Device entities:
* Each user can have multiple devices.
* Each device belongs to only one user.
* Implement getter and setter methods for the attributes in the User and Device classes.
* Create a Repository folder within the src directory.
* Inside the Repository folder, create Java interfaces named "UserRepository.java" and "DeviceRepository.java" to interact with the database using Spring Data JPA.

**Implement the following API endpoints:**

**For User:**

* POST - "/users" --> Creates a new user.
* GET - "/users/{id}" --> Retrieves details of a specific user by ID.
* GET - "/users" --> Retrieves a list of all users.
* PUT - "/users/{id}" --> Updates details of a specific user by ID.
* DELETE - "/users/{id}" --> Deletes a specific user by ID.

**For Device:**

* POST - "/devices" --> Creates a new device for a specific user.
* GET - "/devices/{id}" --> Retrieves details of a specific device by ID.
* GET - "/devices" --> Retrieves a list of all devices.
* PUT - "/devices/{id}" --> Updates details of a specific device by ID.
* DELETE - "/devices/{id}" --> Deletes a specific device by ID.
* PUT - "/devices/{id}/toggle" --> Toggles the status of a specific device (on/off).
* PUT - "/devices/{id}/settings" --> Updates the settings of a specific device.
* GET - "/devices/user/{userId}" --> Retrieves all devices belonging to a specific user.

**Note**: Configure the database connection details in the application.properties file.

**API Endpoint:** localhost:8080

**Platform Guidelines:**

* Use Terminal to run the application.
* To start/run the application: 'mvn spring-boot:run'