DESARROLLO DE APLICACIONES OPEN SOURCE (SI729)

Ejercicio 1
2024-1

Caso SmartHomeManagement

Contexto: SmartHomeManagement Inc. (https://www.smarthomemanagement.com) desea desarrollar una plataforma para gestionar dispositivos inteligentes en hogares modernos. El objetivo es proporcionar una solución que permita a los usuarios controlar y monitorear sus dispositivos de manera eficiente, utilizando un backend robusto y una aplicación web intuitiva.

Objetivo: El equipo de desarrollo debe crear un RESTful API que soporte las operaciones de SmartHomeManagement, incluyendo la gestión de dispositivos y el monitoreo de indicadores de rendimiento.

REQUISITOS:

1. Gestión de Dispositivos (Devices):

Cada dispositivo inteligente (Device) debe tener los siguientes atributos:

- id (Long, Primary Key, Autogenerado)
- serialNumber (String, Obligatorio, Único, máximo 30 caracteres)
- model (String, Obligatorio, máximo 50 caracteres)
- deviceType (DeviceType, Enum, Obligatorio) - Posibles valores: LIGHTING, HEATING, SECURITY
- installationDate (Date, Obligatorio) - Debe ser mayor o igual a la fecha actual del sistema
- status (String, Obligatorio) - Posibles valores: ACTIVE, INACTIVE, máximo 10 caracteres

Reglas de negocio:

- serialNumber debe ser único.
- deviceType debe ser uno de los valores enumerados.
- status debe ser uno de los valores válidos.
- installationDate no puede ser una fecha pasada.

2. Indicadores de Rendimiento (PerformanceIndicators):

Cada indicador de rendimiento debe tener los siguientes atributos:

- id (Long, Primary Key, Autogenerado)

- name (String, Obligatorio, máximo 40 caracteres)
- description (String, Opcional, máximo 200 caracteres)
- minValue (Double, Obligatorio)
- maxValue (Double, Obligatorio)
- deviceType (DeviceType, Enum, Obligatorio)

Reglas de negocio:

- Un PerformanceIndicator puede estar asociado a múltiples Devices.
- minValue debe ser menor que maxValue.
- No se puede asociar un PerformanceIndicator con un deviceType si ya existe otro con el mismo name para el mismo deviceType.

3. Población de Tabla Inicial:

Se debe tener una tabla pre-poblada device_types que contenga los posibles valores de DeviceType. Esta tabla debe ser verificada y poblada al inicio de la aplicación utilizando un evento de ApplicationReady.

Tabla DeviceTypes:

- id (Long, Primary Key, Autogenerado)
- type (String, Obligatorio, Único, máximo 20 caracteres)

Los valores iniciales deben ser:

- 1, "LIGHTING"
- 2, "HEATING"
- 3, "SECURITY"

Endpoints:

1. Devices Endpoint:

- **Agregar un Device (POST):** /api/v1/devices
  - Al agregar un nuevo dispositivo, se debe retornar el status HTTP 201 (Created) y el objeto creado incluyendo su id generado.
- **Actualizar un Device (PUT):** /api/v1/devices/{id}
  - Debe permitir actualizar un dispositivo existente. Se debe retornar el status HTTP 200 (OK) si la actualización es exitosa, y el objeto actualizado. Si el id no existe, retornar el status HTTP 404 (Not Found).

2. Performance Indicators Endpoint:

- **Agregar un Performance Indicator (POST):** /api/v1/performance-indicators
  - Al agregar un nuevo indicador de rendimiento, se debe retornar el status HTTP 201 (Created) y el objeto creado incluyendo su id generado.

Technical Constraints:

1. Elabore la solución con Java 22 y Spring Boot Framework 3.
2. La información debe ser persistente en una base de datos relacional (MySQL) en un esquema smarthome.
3. Los packages deben tener como nombre raíz com.smarthome.platform.upc
4. Incluya documentación de los Endpoints con OpenAPI.
5. Gestione las excepciones en la aplicación.

Bounded Context:

- Inventory: Gestiona los dispositivos inteligentes (Device).
- Analytics: Gestiona los indicadores de rendimiento (PerformanceIndicator).
- Shared: Contiene elementos comunes/reutilizables como DeviceType.

Consideraciones adicionales:

1. La tabla device_types debe ser verificada y poblada al inicio de la aplicación usando un evento ApplicationReady.
2. Utilice minúsculas para los nombres de URL y términos compuestos separados por guión medio (-) para todos los endpoints.
3. Utilice la biblioteca Lombok para el manejo de métodos constructores y de acceso en las clases POJO.
4. Utilice records en vez de clases para almacenamiento de valores inmutables.
5. Para Device, incluya atributos de auditoría createdAt y updatedAt con valores poblados de forma automática por Spring Boot al momento de la creación.
6. Utilice el patrón Assembler para el Object Mapping en la sección transform en la interface layer.
7. Documente su código con JavaDoc, colocando información de propósito para principales objetos de programación, así como propósito, parámetros y valor retornado en clases y métodos relevantes. Incluya como parte de la documentación sus nombres y apellidos como valor para @author.
8. Aplique buenas prácticas de Arquitectura de Software, enfoque de Domain-Driven Design, separación en bounded contexts, layered architecture (domain, application, interfaces, infrastructure), patrones de strategic y tactical Domain-Driven Design, patrón CQRS, principios y patrones de diseño de software orientado a objetos, convenciones de nomenclatura en inglés, así como buenas prácticas de nomenclatura en Java (entre ellas Upper-Camel-Case para Clases, Lower-Camel-Case para atributos y métodos) y buenas prácticas para nomenclatura de objetos de Base de Datos (entre ellas snake case, tablas en plural, sin mnemónicos).
9. Incluya documentación de Endpoints con OpenAPI.

No incluido en el alcance:

1. Soporte de CORS.
2. Seguridad.
3. Testing.