Backend Engineer Task for NinjaOne

Implement a REST API in Java with data persistency to fulfill the basic requirements of an RMM Platform.

Introduction

A Remote Monitoring and Management (RMM) platform helps IT professionals manage a fleet of Devices with Services associated with them. This Web Service will fulfill the most basic requirements of an RMM by keeping a simple inventory of Devices and Services to calculate their total costs.

Breakdown of Service costs:

- Each Device costs $4
- Antivirus costs $5 for each Windows, $7 for each Mac per Device
- Backup costs $3 per Device
- PSA costs $2 per Device
- Screen Share costs $1 per device

Devices have the following properties:

- Id
- System Name
- Type (Windows Workstation, Windows Server, Mac, etc.)

Feature Requirements

Implement endpoints and logic for the following.

a. Get, Add, Update or Delete Devices. Duplicate Devices should not be allowed.
b. Add, Delete available Services. Duplicate Services should not be allowed.
c. Calculate the total monthly cost of the services depending on Services used by a Device.
   Example:
   Customer with 2 Windows, 3 Mac with Antivirus, Backup, and Screen Share.

   Total Cost: $71

   Explanation:
   Devices cost: $20
   Antivirus cost: $31
   Backup: $15
   Screen Share: $5

Project Template

A Project Template hosted at https://github.com/NinjaRMM/backend-interview-project-app-template is provided to aid with this exercise. The template provides the following items already pre-configured and ready to consume:

- Spring Boot Web Service
- H2 Embedded Relational Database
- Unit Tests

Instructions

Using the provided **Project Template** or from scratch, implement the service as described in **Feature Requirements**

Items to Focus On

In addition to fulfilling the requirements of the service, special attention should be made to:

- Providing a flexible data model so that in the future, additional device and service types can be easily added
- Efficient and correct calculation of service costs

Bonus

- Implementation of thorough unit testing and/or integration testing
- Securing the web service
- Good documentation