Тестовое задание

Необходимо реализовать WEB сервер с базой данных и двумя REST запросами. Для выполнения задания можно использовать любые библиотеки и технологии.

Ссылку на исходники проекта кидайте на почту: yuriy.lyamin@ambiot.io

Дамп БД скидывать не нужно.

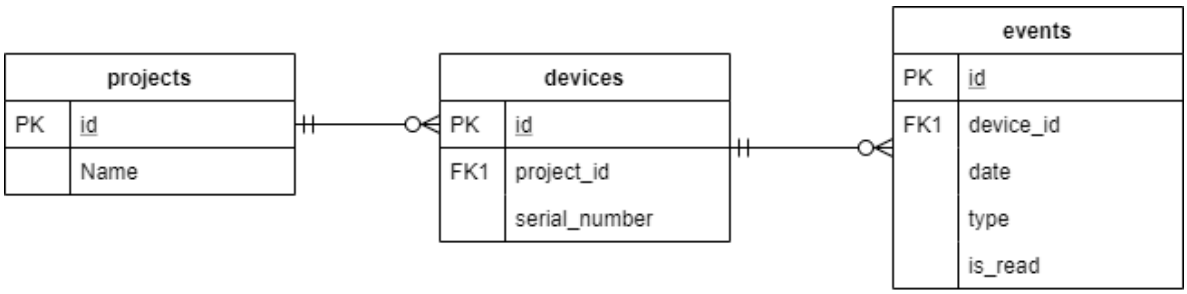
Язык реализации проекта: Java (11) или Kotlin.

СУБД: PostgreSQL или MySQL.

Задание можно сделать двумя способами:

- Если вы никогда не работали с БД ранее или испытываете трудности, то можете реализовать структуру БД локально. Создать по классу под каждую из таблиц, а затем заполнить локальными данными.
- Если же у вас есть опыт работы с БД, то необходимо сделать задание полностью как описано в документе далее.

1. Структура базы данных



База данных состоит из трех таблиц:

- `projects` -содержит информацию о проектах
  - `id` - уникальный идентификатор проекта
  - `name` - название проекта
- `devices` - содержит информацию об устройствах
  - `id` - уникальный идентификатор устройства
  - `project_id` - идентификатор проекта в таблице `projects`
  - `serial_number` - уникальный серийный номер устройства, например: A4BCC
- `events` - содержит информацию о истории событий и ошибках, которые возникали на устройствах
  - `id` - уникальный идентификатор события
  - `device_id` - идентификатор устройства в таблице `devices`
  - `date` - дата возникновения события (UTC)

- `type` - (enum) - тип события. Поле может принимать значения: `event`, `warning` или `error`
- `is_read` - (boolean) - прочитано событие или нет

Пример:

Есть проект "Омск", в нем находится 6000 устройств, для каждого устройства ведется история событий и ошибок.

2. Серверная часть

На стороне сервера необходимо реализовать REST интерфейс для получения данных о проектах и устройствах.

REST интерфейс должен реализовывать два запроса:

Первый запрос должен должен возвращать информацию о всех устройствах проекта.

Входные параметры: `id` проекта

Формат ответа:

```
{
    "4BCD": {
        "id": 45,
        "serialNumber": "4BCD",
        "projectId": 3,
        "hasErrors": false,
        "summaryInfo": {
            "eventCount": 78,
            "warningCount": 15,
            "errorCount": 0
        }
    },
    "4BCC": {
        "id": 46,
        "serialNumber": "4BCC",
        "projectId": 3,
        "hasErrors": true,
        "summaryInfo": {
            "eventCount": 0,
            "warningCount": 5,
            "errorCount": 19
        }
    }

}
```

`hasError` - (boolean) - флаг, которые показывает наличие ошибок на устройстве

SummaryInfo:

- `eventCount` - (int) - количество событий на устройстве

- `warningCount` - (int) - количество предупреждений на устройстве
- `errorCount` - (Int) - количество ошибок на устройстве

Второй запрос должен возвращать информацию о всех проектах.

Входных данных нет.

Формат ответа:

```
[
    {
        "id":3,
        "projectName": "Амстердам",
        "stats": {
            "deviceCount": 560,
            "deviceWithErrors": 60,
            "stableDevices":500
        },
        "devices": ["4BCD", "4BCE", "6BCF"]

    },
    {
        "id":4,
        "projectName": "Школа №259",
        "stats": {
            "deviceCount": 29,
            "deviceWithErrors": 5,
            "stableDevices":20
        },
        "devices": ["4BC6", "4BFF", "JORA"]

    }
]
```

Stats:

- `deviceCount` - (int) - общее количество устройств в проекте
- `deviceWithErrors` - (int) - количество устройств с ошибками за последние сутки. Устройством с ошибкой считается то устройство, для которого есть хотя бы одна запись за последние сутки в таблице `events` и поле `type = 'error' или 'warning'`.
- `stableDevices` - (int) - количество стабильно работающих устройств. Стабильно работающее устройство - это устройство для которого в таблице `events` есть только записи у которых поле `type = 'event'`

`devices` - (array) - массив с серийными номерами устройств проекта