Система розсилки електронних листів

Стек технологій: Java 11+, Spring Boot, H2 database, Gmail SMTP server.

Постановка задачі: розробити REST API що включає наступні можливості:

1. Управління користувачами.
2. Відправка листів користувачам.
3. Перегляд статистики відправлених листів(logs).

Управління користувачами

Структура запису користувача(user) в базі даних**:**

1. *id* - primary key, int, auto increment, генерується автоматично базою даних.
2. *username* - string, not null, not empty.
3. *email* - string, not null, not empty.
4. *createdOn* - datetime, not null, not empty, дата та час створення, генеруються автоматично.

Управління користувачами складається з:

1. Створення користувача за двома вхідними параметрами: username, email.
2. Редагування даних користувача: username, email.
3. Видалення користувача.
4. Перегляд списку користувачів:
    a. Можливість посторінкового перегляду.
    b. Результати мають бути відсортовані за датою створення користувача в системі - найновіші спочатку.
    c. Можливість пошуку користувача за іменем або поштою.

Відправка листів

Система для відправки листів відбувається двома способами:

1. За допомогою REST API call конкретному користувачу.
2. Автоматично всім користувачам, на основі наперед заданого часового правила(так звані cron jobs). Приклади правил: надсилати лист кожні 5 хвилин, кожен день о п'ятій ранку, щопонеділка о 8 вечора і тд. **Необхідна підтримка всіх можливих правил.**

Шаблон листа
Subject: Вітання!

To: {email}
Body:
Ім'я користувача: {username}
Дата та час створення: {createdOn}

Відправка листа за допомогою REST API
Необхідно розробити API який буде надсилати лист вказаному користувачу(по user Id).

Cron jobs
Структура запису cron в базі даних**:**

1. *id* - primary key, int, auto increment, генерується автоматично на рівні бази даних.
2. *expression* - string, not null, not empty, формат. Приклад: 0 0 * * * *, */10 * * * * *, 0 0 6,19 * * * і тд.
3. *createdOn* - datetime, not null, not empty, дата та час створення, генерується автоматично.

Cron job можна налаштовувати за допомогою REST API, тобто необхідна можливість:

1. Створення cron, де на вхід передається expression.
2. Редагування cron expression.
3. Видалення cron.
4. Посторінковий перегляд списку cron задач.

Статистика відправлених листів(logs)

Кожен факт відправленого листа має бути записаний в базу даних аби була можливість переглянути статистику.

Структура запису log в базі даних**:**

1. *id* - primary key, int, auto increment, генерується автоматично на рівні бази даних
2. *user_id* - foreign key, int, not null.
3. *type* - enum {REST, CRON}, not null. REST - для логів, що згенеровані відправкою листа через REST API. CRON - ті що генерувались cron задачею.
4. *createdOn* - datetime, not null, not empty, дата та час створення, генерується автоматично.

Необхідно розробити REST API яке буде віддавати наступну статистику по кожному користувачу:

1. Кількість відправлених листів з типом REST.
2. Кількість відправлених листів з типом CRON.
3. Дата та час першого листа.
4. Дата та час останнього листа.
5. Посторінковий перегляд.

6. Результати мають бути відсортовані за загальною кількістю відправлених листів - найбільша кількість на початку.

Приклад відповіді з серверу:

```
[
  {
    "username": "user1",
    "email": "user1@mail.com",
    "count": {
      "rest": 10,
      "cron": 20
    },
    "first": "2022-01-01 23:00:00",
    "last": "2022-01-02 23:00:00"
  },
  {
    "username": "user2",
    "email": "user2@mail.com",
    "count": {
      "rest": 12,
      "cron": 22
    },
    "first": "2022-01-02 23:00:00",
    "last": "2022-01-03 23:00:00"
  }
]
```

Додаткові вимоги:

1. API має бути стійким до помилкового використання, тобто має бути врахована валідація вхідних даних.
2. API має отримувати і відповідати даними у форматі JSON.
3. Результат має бути опублікований на Github.
4. Репозиторій має включати файл README з роз'ясненням як запускати сервер.