Тестовое задание на позицию стажер-java в СКБ Контур

Сокращатель ссылок

Необходимо разработать HTTP-сервис, который генерирует короткие ссылки.

Описание

В сервисе должна быть реализована следующая функциональность:

1. Генератор ссылок

Ресурс `/generate` должен обрабатывать POST-запрос, содержащий оригинальную ссылку, и генерировать короткую ссылку. Оригинальная ссылка передается в теле POST-запроса в поле “original” JSON-объекта. Сгенерированная короткая ссылка передается в теле ответа в поле “link” JSON-объекта. Сгенерированная короткая ссылка должна иметь формат `/l/{some-short-name}`, то есть не должна содержать адрес сервера, где some-short-name - идентификатор ссылки. Алгоритм генерации этого идентификатора и его формат остается на усмотрение автора.

Пример запроса:
```
POST /generate
```

Пример тела запроса:
```
{
  “original”: “https://some-server.com/some/url?some_param=1”
}
```

Пример ответа:
```
{
  “link”: “/l/some-short-name”
}
```

2. Редирект

Ресурс `/l/{some-short-name}` должен осуществлять редирект на оригинальный url.
Параметры пути запроса:

- some-short-name - идентификатор ссылки

Пример запроса:
```
GET /l/some-short-name
```

Предлагаем подумать, каким образом можно оптимизировать работу с ссылками, по которым происходит много переходов.

3. Статистика

3.1 Статистика по конкретным ссылкам

Ресурс `/stats/{some-short-name}` должен обрабатывать GET-запрос и возвращать статистику переходов по конкретной ссылке.

Параметры пути запроса:

- some-short-name - идентификатор ссылки

В ответе сервиса должен содержаться JSON-объект со следующими полями:

- link - сгенерированная короткая ссылка
- original - оригинальная ссылка
- rank - место ссылки в топе запросов
- count - число запросов по короткой ссылке

Пример запроса:

```
GET /stats/some-short-name
```

Пример ответа:

```
{
  “link”: “/l/some-short-name”,
  “original”: “http://some-server.com/some/url”
  “rank”: 1,
  “count”: 100500
}
```

3.2 Рейтинг ссылок

Ресурс `/stats` должен обрабатывать GET-запрос и возвращать статистику запросов с сортировкой по частоте запросов по убыванию и возможностью постраничного отображения.

Параметры строки запроса:

- page - номер страницы
- count - число записей, отображаемых на странице, максимальное возможное значение 100 (включительно)

В ответе сервиса должен содержаться массив из JSON-объектов, описанных в разделе 3.1 Статистика по конкретным ссылкам.

Пример запроса:

```
GET /stats?page=1&count=2
```

Пример ответа:

```
[
  {
    “link”: “/l/some-short-name”,
    “original”: “http://some-server.com/some/url”
    “rank”: 1,
    “count”: 100500
  },
  {
    “link”: “/l/some-another-short-name”,
    “original”: “http://another-server.com/some/url”
    “rank”: 2,
    “count”: 40000
  }
]
```

Требования к сервису

1. При создании сервиса можно воспользоваться любым из указанных фреймворков: Spring Framework, Spring Boot, Undertow, или написать приложение на чистой Java.
2. Сервис должен запускаться как standalone-приложение (*java -jar service.jar*).
3. Необходимо покрыть сервис Unit-тестами. Полнота покрытия остается на усмотрение автора решения.
4. Приложение должно собираться с использованием Apache Maven.
5. По возможности необходимо придерживаться REST-архитектуры.
6. Не использовать отдельно устанавливаемые базы данных, кэши и т.п. Можно использовать embedded базы данных (например, H2 в режиме встроенной БД).

Прочие требования

1. Язык программирования: Java 8.
2. Оформление кода должно соответствовать общепринятым нормам (например, https://google.github.io/styleguide/javaguide.html)
3. Исходники необходимо упаковать в ZIP-архив вместе с кратким описанием решения и инструкцией по сборке и запуску. Архив необходимо загрузить на файлообменник, и отправить ссылку на архив в заявке на стажировку
4. Проверка решения будет осуществляться в автоматизированном режиме (функциональные и нагрузочные тесты), поэтому важно соблюдение указанных в задании требований.
5. Всё, что явно не указано в условиях, остаётся на усмотрение автора решения.