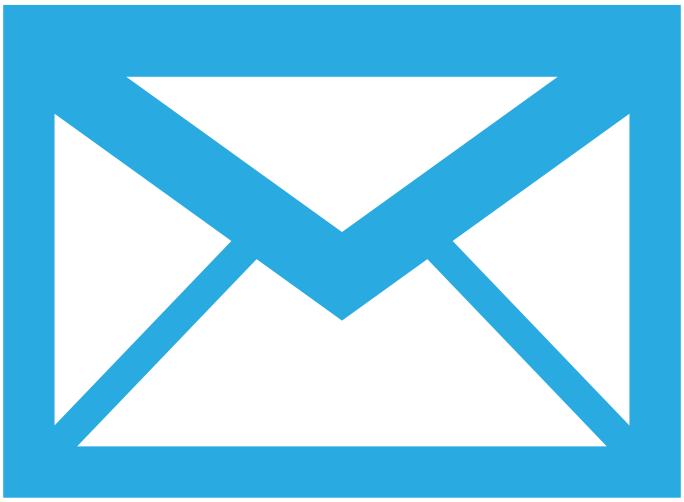
Email Sending Service

Тестовое задание для кандидатов-разработчиков



12.01.2020

Суть задания

Предлагаем вам сделать сервис отправки почтовых уведомлений **Email Sending Service**, который будет принимать сообщения с уведомлениями по протоколу HTTP, отправлять их на указанный в уведомлении адрес и сохранять отправленные уведомления в базу.

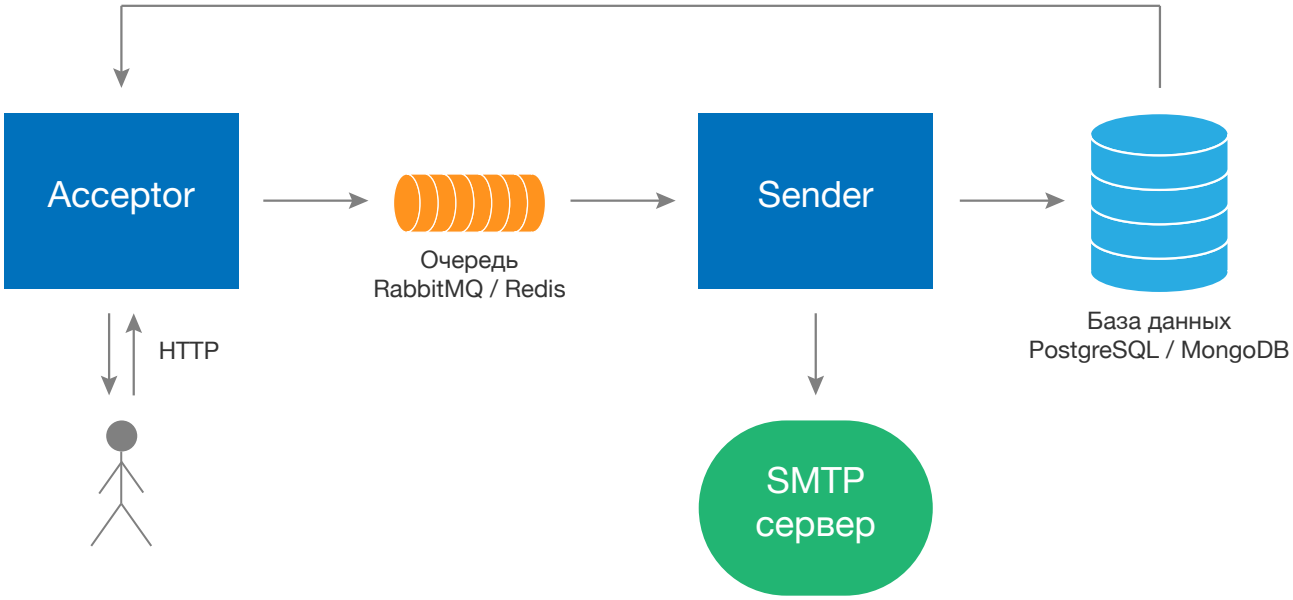
Необходимо, чтобы Email Sending Service работал по протоколу REST API, в котором будут три ручки: ручка для отправки уведомления, ручка для вывода списка отправленных уведомлений и ручка для вывода полной информации о конкретном уведомлении.

Сервис можно реализовать либо на **Go,** либо на **Perl**. Вы также можете выбрать другой язык программирования, если были предварительные договорённости об этом на собеседовании.

Код сервиса должен быть выложен на Github.

Предполагаемая архитектура Email Sending Service

Разработка Email Sending Service предполагает, как минимум, два микросервиса: **Acceptor** и **Sender**. Acceptor принимает и обрабатывает HTTP-запросы в рамках API. Sender принимает от Acceptor-а уведомления, отправляет их через SMTP-сервер и сохраняет отправленные уведомления в базу. Acceptor и Sender взаимодействуют друг с другом через очередь. Когда Acceptor получает запрос на отправку уведомления, он пишет его в очередь. Sender вычитывает уведомление из очереди, устанавливает соединение с SMTP-сервером и отправляет полученное уведомление на указанный в нём адрес. После этого сохраняет отправленное уведомление в базу.



В качестве брокера очередей необходимо использовать либо **RabbitMQ**, либо **Redis** (Streams). В качестве СУБД необходимо использовать либо **PostgreSQL**, либо **MongoDB**.

Сервис должен уметь обрабатывать некорректные пользовательские запросы и осуществлять валидацию всех входящих параметров. Ошибка обработки одного запроса не должна приводить к падению всего сервиса.

Параметры подключения к брокеру очередей, базе данных и к SMTP-серверу должны быть конфигурируемыми через конфигурационные файлы (*.yml, *.json), либо через переменные окружения.

Email Sending Service API

В рамках тестового задания мы ожидаем, чтобы API сервиса содержало следующие ручки:

- ручка для отправки уведомления;
- ручка для вывода списка отправленных уведомлений;
- ручка для вывода полной информации о конкретном уведомлении.

Ниже приведён типовой шаблон описания API сервиса и его ручек, чтобы задать общее направление для разработки, вы можете расширить предлагаемое API любыми своими идеями.

Отправка уведомления

Отправка уведомления осуществляется POST-запросом. Параметры уведомления задаются в теле запроса в формате JSON.

```
POST /notifs/

{
  "sender": "sender_A",
  "to": ["bar@example.ru", "jar@example.ru"],
  "subject": "Тема уведомления",
  "message": "Текст уведомления"
}
```

Параметр	Тип	Обязательный	Описание
sender	`string`	Нет	Отправитель уведомления.
to	`string[]`	Да	Список email-адресов получателей уведомления.
subject	`string`	Нет	Тема почтового сообщения с уведомлением.
message	`string`	Да	Текст уведомления.

Если запрос был обработан без ошибок и уведомление было принято на отправку, то в ответ будет выглядеть следующим образом:

```
Status: 202 Accepted

{
  "id": "9b08620f-f7da-4164-897c-83a8c27d43de"
}
```

`id` — это идентификатор уведомления присвоенный сервисом.

Вывод списка отправленных уведомлений

Вывод списка отправленных уведомлений осуществляется GET-запросом. Список уведомлений возвращается в теле ответа в формате JSON и должен выводиться постранично. Максимальное количество уведомлений на одну страницу — 1000.

Параметры пагинации

Управление выводом постраничного списка осуществляется двумя параметрами `page` и `per_page`. Параметры должны передаваться в строке запроса.

Параметр	Тип	Обязательный	Описание
page	`int`	Нет	Номер страницы. По умолчанию 1.
per_page	`int`	Нет	Число уведомлений на страницу. По умолчанию выводится 20 уведомлений.

Заголовки пагинации

В ответе при выводе постраничного списка должны присутствовать заголовки пагинации, в которых должна содержаться информация об общем количестве элементов в списке, общее число страниц и другая вспомогательная информация.

Заголовок	Обязательный	Описание
X-Total	Да	Общее число элементов.
X-Total-Pages	Да	Общее число страниц.
X-Per-Page	Да	Число элементов на страницу.
X-Page	Да	Номер страницы.
X-Next-Page	Нет	Номер следующей страницы.
X-Prev-Page	Нет	Номер предыдущей страницы.

```
GET /notifs/?page=1&per_page=10

Status: 200 OK
X-Total: 2
X-Total-Pages: 1
X-Per-Page: 20
X-Page: 1

[
  {
    "id": "9b08620f-f7da-4164-897c-83a8c27d43de",
    "sender": "sender_A",
    "to": ["bar@example.ru", "jar@example.ru"],
    "subject": "Тема уведомления",
    "message": "Текст уведомления",
    "sent_status": true,
    "created_at": "2020-01-10T12:27:12Z"
  },
  {
    "id": "72012884-d270-4347-8fe1-be3c03e255dd",
    "sender": "sender_B",
    "to": ["foo@example.ru", "jar@example.ru"],
    "subject": "Тема уведомления",
    "message": "Текст уведомления",
    "sent_status": true,
    "created_at": "2020-01-10T12:10:25Z"
  }
]
```

Параметр	Тип	Обязательный	Описание
id	`string`	Да	Идентификатор уведомления.
sender	`string`	Нет	Отправитель уведомления.
to	`string[]`	Да	Список email-адресов получателей уведомления.
subject	`string`	Нет	Тема почтового сообщения с уведомлением.
message	`string`	Да	Текст уведомления.
sent_status	`bool`	Да	Статус отправки уведомления: `true` — отправлено, `false` — не отправлено.
created_at	`string`	Да	Дата создания уведомления в формате RFC3339.

Вывод информации об отправленном уведомлении

Вывод полной информации об отправленном уведомлении осуществляется GET-запросом с указанием идентификатора уведомления. Информация об уведомлении возвращается в теле ответа в формате JSON.

```
GET /notifs/<id>

Status: 200 OK

{
  "id": "9b08620f-f7da-4164-897c-83a8c27d43de",
  "sender": "sender_A",
  "to": ["bar@example.ru", "jar@example.ru"],
  "subject": "Тема уведомления",
  "message": "Текст уведомления",
  "sent_status": true,
  "created_at": "2020-01-10T12:27:12Z"
}
```

Параметр	Тип	Обязательный	Описание
id	`string`	Да	Идентификатор уведомления.
sender	`string`	Нет	Отправитель уведомления.
to	`string[]`	Да	Список email-адресов получателей уведомления.
subject	`string`	Нет	Тема почтового сообщения с уведомлением.
message	`string`	Да	Текст уведомления.
sent_status	`bool`	Да	Статус отправки: `true` — отправлено, `false` — не отправлено.
created_at	`string`	Да	Дата создания уведомления в формате RFC3339.

Дополнения к заданию

Основная цель данного тестового задания это написать сервис отправки почтовых уведомлений, который будет выполнять свою функцию, но его можно сделать еще лучше. Для этого Вы можете выполнить ряд дополнительных задач, по вашему усмотрению, которые будут дополнительно оценены и позволят Вам еще шире продемонстрировать Ваши навыки.

Логирование

Реализовать информативное логирование внутри Acceptor-а и Sender-а, чтобы по логам можно было отследить ход обработки каждого запроса и уведомления. Также нужно реализовать логирование ошибок, которые могут возникнуть в ходе работы микросервисов. Логи должны выводиться на STDOUT или STDERR. Формат логов на Ваше усмотрение.

Re-connect

Часто сеть бывает ненадежной. Разрывы сетевых соединений между базой данных или брокером очередей не должны приводить к падению сервиса и потерям запросов и уведомлений. Если вдруг база или очередь не доступны, сервис должен попытаться восстановить соединение и повторить свой запрос.

OpenAPI

Описать API сервиса в виде OpenAPI (Swagger) спецификации и сделать, чтобы по адресу `/docs/` открывалась страница со Swagger UI, в котором была бы подгружена данная OpenAPI спецификация. Пример: https://petstore.swagger.io

Docker

Подготовить Acceptor и Sender для запуска внутри docker-контейнера. Для этого нужно для каждого микросервиса написать свой Dockerfile. Затем, используя их, собрать docker-образы и запустить docker-контейнеры. Acceptor и Sender должны работать в разных docker-контейнерах.