Руководство администратора ПО «Система уведомлений пользователей посредством электронной почты»

1.Введение

1.1. Область применения

Настоящий документ предназначен для сотрудников эксплуатирующей организации и отражает основные функциональные возможности и порядок действий при выполнении операций, связанных с администрированием программного обеспечения «Система уведомлений пользователей посредством электронной почты» (далее - «Система»)

1.2. Перечень выполняемых функций администратора/оператора

В перечень выполняемых функций администратора Системы входят:

- Установка и настройка Системы
- Реализация планов устранения сбоев и нетиповых нештатных ситуаций
- Выполнение сбора и предоставление в вышестоящую линию технической поддержки информации для воспроизведения технических проблем и выработки решений по их разрешению
- Реализация рекомендаций по устранению нештатных ситуаций, полученных с вышестоящей линии поддержки
- Восстановление работоспособности Системы при сбоях в работе функциональных модулей
- Разработка решения по устранению технических проблем в работе функциональных модулей

1.3. Уровень подготовки администратора/оператора

Администратор/оператор (далее по тексту Администратор) Системы должен уметь пользоваться и настраивать среду функционирования контейнеров или систему оркестрации, используемую на предприятии.

Рекомендуемая численность персонала для эксплуатации Системы — 1 штатная единица.

Администраторы Системы должны пройти обязательную общую и специальную подготовку для работы с Системой.

Общая подготовка должна включать в себя получение знаний и навыков работы с Системой в качестве администратора.

Специальная подготовка должна включать в себя получение знаний и навыков в объеме, необходимом для выполнения своих должностных обязанностей

1.4. Перечень документации

В состав документации, с которой необходимо ознакомиться администратору Системы входят:

- описание функциональных характеристик Системы
- описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения.

2. Установка Системы

В данном разделе будет описана установка Системы на Debian Linux. Предполагается, что были предварительно установлены также Docker, Docker Compose, RabbitMQ, PostgreSQL.

2.1. Системные требования к ПО

Минимальные аппаратные требования:

- Операционная система, способная запускать контейнеры. Предпочтительно Linux.
- Система управления контейнерной виртуализацией. Предпочтительно Docker Swarm или Kubernetes.
- Подключение к серверу очередей RabbitMQ
- Подключение к СУБД Postgresql
- Количество логических ядер процессора: 4
- Семейство процессоров: x86
- Частота процессора: 3.0. ГГц
- Объем установленной памяти: 16 Гб

2.1.2. Минимальные требования к сторонним компонентам и/или системам, необходимым для установки и работы ПО

- Debian 11 (Открытая лицензия GNU)
- Docker 24.0.2 (open-source community edition)
- RabbitMQ (Открытая лицензия Mozilla Public License)
- Grafana Loki 2.6.1 (Открытая лицензия GNU)
- Grafna 9.2.2 (Открытая лицензия GNU)
- Postgresql 14 (Открытая лицензия PostgreSQL license)

2.1.3. Языки программирования

При разработке Системы был использован язык программирования Javascript (Node.js v20.11.0) (Открытая лицензия MIT)

2.2. Порядок установки

1. Создайте папку /home/app
2. Смонтируйте диск с дистрибутивом в папку /mnt
3. Скопируйте из дистрибутива исходники из папки /mnt в папку /home/app
4. Выполните скрипт install.sql, находящийся в папке /home/app, на вашем сервере PostgreSQL. Обратите внимание, что установка и настройка сервера PostgreSQL, а также создание базы данных находятся вне компетенции этого документа и не будут тут описаны.
5. Смените текущую папку на /home/app и выполните команды
   sudo chown 10001:10001 ./volumes/loki
   sudo chown 472:472 ./volumes/grafana
6. Отредактируйте файл docker-compose.yml, в соответствии с пунктами 3.2 и 3.3 данного документа
7. Создайте и отредактируйте файлы настроек для обоих модулей, в соответствии с пунктами 3.2.2, 3.2.3 и 3.3.2 данного документа
8. Смените текущую папку на /home/app и выполните в ней команду
   docker compose -up -d --build
9. Войдите браузером на ваш сервер на порт 3000 в систему мониторинга с пользователем admin и паролем admin. Измените пароль на безопасный.

3. Настройка Системы

3.1. Общие сведения

В данном документе приводятся примеры настройки Системы с использованием среды Docker Compose. Настройка операционной системы, СУБД, сервера очередей RabbitMQ, а также возможная настройка использования систем оркестрации, находятся вне компетенции этого документа и не будут тут описаны.

3.2. Модуль приема запросов

3.2.1. Конфигурируемые параметры

Для корректной работы модуля приема запросов, необходимо настроить для него следующие переменные окружения:

- HOST - адрес сервиса, который будет слушать модуль. На этот адрес следует пробросить внешний порт или настроить проксирующий сервер для поддержки протокола https.
- AMQP_HOST — имя хоста или IP сервера RabbitMQ, с указанием порта и учетной записи. Пример смотрите ниже.
- AMQP_PORT — порт сервера RabbitMQ
- AMQP_USER — пользователь сервера RabbitMQ
- AMQP_PASSWORD — пароль пользователя сервера RabbitMQ
- AMQP_EXCHANGE — точка обмена на сервере RabbitMQ
- DB_HOST — адрес сервера PostgreSQL
- DB_PORT— порт сервера PostgreSQL
- DB_USER — пользователь базы данных
- DB_PASSWORD — пароль пользователя базы данных
- DB_NAME — имя базы данных
- LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug

Пример настройки модуля:

```
 httpin:
   build:
     context: ./http-api/
   restart: always
   ports:
     - '80:80'
   environment:
     HOST: :80
     LOG_LEVEL: debug
     AMQP_HOST: host.docker.internal
     AMQP_PORT: 5672
     AMQP_USER: rabbit
     AMQP_PASSWORD: Q3ij6X4DZnb9uPT
     AMQP_EXCHANGE: out_test
     DB_HOST: host.docker.internal
     DB_PORT: 5432
     DB_USER: postgres
     DB_PASSWORD: LDP8brN7B8ZLzf5Q879QXuJYXCwm9nP
     DB_NAME: messages_db
```

```
  extra_hosts:
    - "host.docker.internal:host-gateway"
```

3.3. Модуль доставки сообщений на электронную почту

3.3.1. Конфигурируемые параметры

Для корректной работы модуля, необходимо настроить для него следующие переменные окружения:

- SMTP_HOST — Адрес SMTP сервера
- SMTP_PORT — порт SMTP сервера
- SMTP_USER — имя пользователя SMTP сервера
- SMTP_PASSWORD - пароль пользователя SMTP сервера
- SMTP_SECURE — протокол использует SSL или нет
- SMTP_REJECTUNAUTHORISED — автоматически сбрасывать соединение при проблемах с проверкой сертификата
- SMTP_FROM — адрес электронной почты, который будет фигурировать в поле «От»
- RABBIT_HOST — имя хоста или IP сервера RabbitMQ.
- RABBIT_PORT — порт сервера RabbitMQ.
- RABBIT_USER — пользователь сервера RabbitMQ.
- RABBIT_PASSWORD — пароль пользователя сервера RabbitMQ.
- RABBIT_EXCHANGE — точка обмена на сервере RabbitMQ
- TEMPLATE_PATH — путь к папке с шаблонами сообщений.
- CONSOLE_LOG_LEVEL - уровень логгирования. Поддерживаемые значения:
    - error
    - warn
    - info
    - debug

Пример настройки модуля:

```
 emlout:
   build:
     context: ./email-out/
   restart: always
   environment:
     SMTP_HOST: "smtp.yandex.ru"
     SMTP_PORT: "587"
     SMTP_USER: "mail@ya.ru"
     SMTP_PASSWORD: "ccgvnsgdfgddsfggdf"
     SMTP_SECURE: "false"
     SMTP_REJECTUNAUTHORISED": "true"
     SMTP_FROM: "mail@ya.ru"
     RABBIT_HOST: host.docker.internal
     RABBIT_PORT: "5672"
     RABBIT_USER: "rabbit"
     RABBIT_PASSWORD: "Q3ij6X4DZnb9uPT"
```

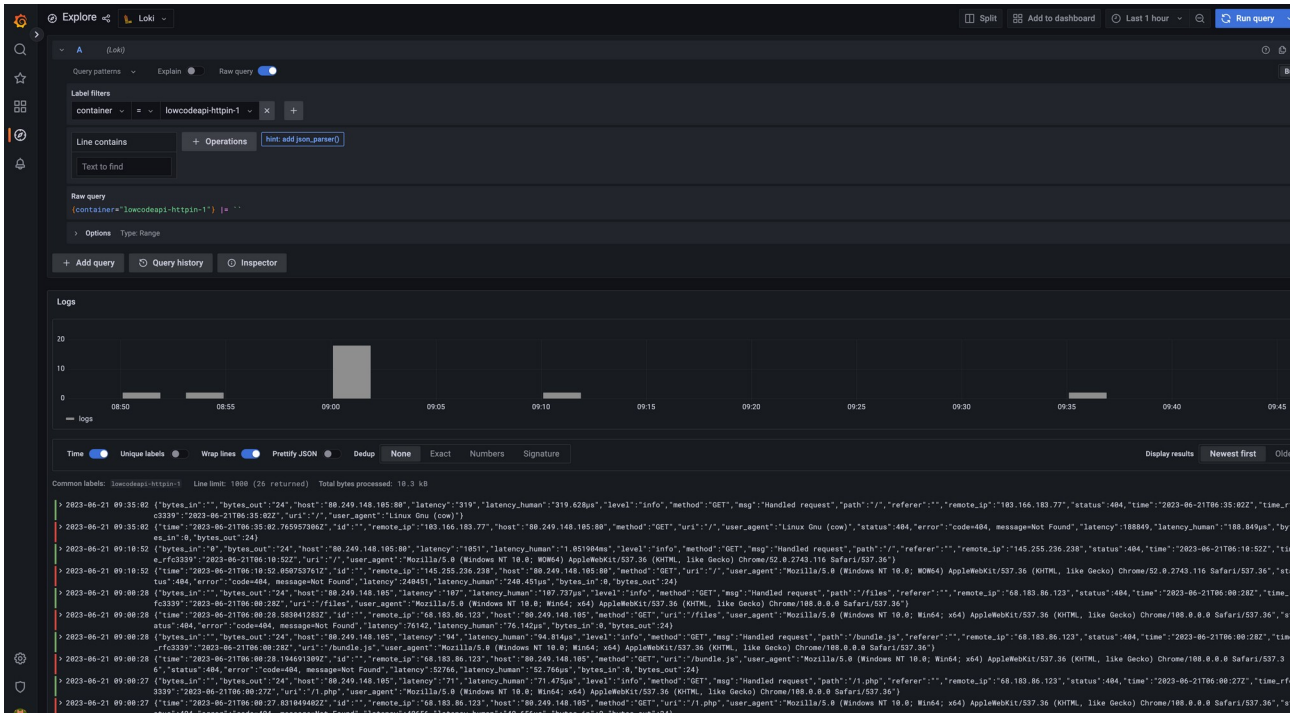
```
  RABBIT_EXCHANGE: "out_test"
  TEMPLATE_PATH: /files
volumes:
  - ./volumes/files:/files
extra_hosts:
  - "host.docker.internal:host-gateway"
```

4. Система мониторинга

В качестве системы мониторинга используется Grafana Loki — это набор компонентов для полноценной системы работы с логами. Loki-стек состоит из трёх компонентов: Promtail, Loki, Grafana. Promtail собирает логи, обрабатывает их и отправляет в Loki. Loki их хранит. А Grafana умеет запрашивать данные из Loki и показывать их. Loki можно использовать не только для хранения логов и поиска по ним. Весь стек даёт большие возможности по обработке и анализу поступающих данных

Чтобы открыть интерфейс системы мониторинга, перейдите в браузере на IP Вашего сервера и порт 3000. Если Вы входите туда в первый раз, используйте логин admin и пароль admin. После первого входа система попросит Вас изменить пароль на безопасный.

Интерфейс выглядит так:



Выберите в меню пункт «Explore» - Вы увидите страницу поиска логов.

Сам запрос состоит из двух частей: selector и filter. Selector — это поиск по индексированным метаданным (лейблам), которые присвоены логам, а filter — поисковая строка или регэксп, с помощью которого отфильтровываются записи, определённые селектором.

Выберите в разделе Label filters в ниспадающем списке Label значение container, а в

ниспадающем списке value выберите нужный контейнер. Выполните запрсо Run query и Вы увидите логи выбранного контейнера.

5. Описание API

5.1. Общие сведения

Модуль приема запросов отвечает за взаимодействие с Системой. Модуль принимает HTTP сообщения. В случае, если требуется взаимодействие по протоколу HTTPS, это настраивается внешними, по отношению к Системе, способами.

5.2. Управление типами сообщений

Система принимает, заранее сконфигурированные, типы сообщений. Для того, чтобы Система могла работать с сообщением, необходимо внести его в базу данных Системы и подготовить соответствующие шаблоны сообщений.

5.2.1. Шаблоны сообщений

Шаблоны сообщений могут быть написаны на нескольких языках. В случае, если отсутствует шаблон сообщения на необходимом языке, будет использован шаблон сообщения на русском языке. Если соответствующий шаблон сообщения не будет найден — сообщение не будет доставлено. Шаблоны сообщений хранятся в папке, которую использует Модуль доставки сообщений в Телеграм. Конкретное их расположение в файловой системе модуля регулируется переменной окружения TEMPLATES_PATH.

Внутри этой папки должны располагаться папки с кодами поддерживаемых языков. Например ru или en, внутри которых уже располагаются файлы шаблонов.

Шаблон представляет собой текстовый файл с любым именем, внутри которого находится сообщение, которое будет отправлено пользователю. Сообщение может содержать именованные параметры, окруженные двойными круглыми скобками. Например: {{test}}. При получении сообщения, Система будет заменять все параметры в шаблоне на пришедшие в сообщении параметры.

5.2.2. Создание типа сообщений

Для создания типа сообщений, надо отправить PUT запрос на url /subjects. Тело запроса должно содержать JSON-объект с параметрами:

- subject — тип сообщения или его тема. Уникальный идентификатор, по которому Система идентифицирует типы сообщений. Также будет использован в виде темы электронного письма.
- params — параметры типа сообщения. Объект, содержащий необходимые для отправки сообщения настройки.
    - email_template - имя файла шаблона сообщения.

Например: PUT http://localhost/subjects

```
{
   "subject": "mail_test",
   "params": {
      "email_template": "test.tpl",
   }
}
```

В качестве ответа, Система возвращает уникальный идентификатор созданного типа сообщений. Этот идентификатор, в дальнейшем, будет использоваться для управления этим типом сообщений. Например:

```
{
   "result": "241b5680-4696-4e70-a8b6-14adcd68971d",
}
```

5.2.3. Чтение типа сообщений

Для чтения сообщения необходимо отправить GET-запрос на url /subjects/<идентификтор типа сообщений>

Например: GET http://localhost/subjects/241b5680-4696-4e70-a8b6-14adcd68971d

В ответе Система возвращает свойства указанного типа сообщений. Например:

```
{
      "subject": "mail_test",
      "params": {
         "email_template": "test.tpl",
      }
}
```

5.2.4. Чтение списка типов сообщений в Системе

ля чтения сообщения необходимо отправить GET-запрос на url /subjects Вернется список всех имеющихся в системе типов сообщений.

Например: GET http://localhost/subjects

```
{
   "result": [
      {
         "id": "fa791266-d945-406c-9be5-a74ff95f7e60",
         "subject": "mail_test",
         "params": {
            "email_template": "test.tpl"
```

```
        }
      },
      {
        "id": "241b5680-4696-4e70-a8b6-14adcd68971d",
        "subject": "testing",
        "params": {
          "email_template": "test.tpl"
        }
      }
   ]
}
```


5.2.5. Изменение типа сообщений

Для изменения надо отправить POST запрос на url /subjects В теле запроса должен быть JSON-объект с параметрами:

- id — идентификатор типа сообщения
- subject — тип сообщения или его «тема». Уникальный идентификатор, по которому Система идентифицирует типы сообщений.
- params — параметры типа сообщения. Объект, содержащий необходимые для отправки сообщения настройки.
    - email_template - имя файла шаблона сообщения.

Например: POST http://localhost/subjects


```
{
   "id": "241b5680-4696-4e70-a8b6-14adcd68971d",
   "subject": "test1",
   "params": {
      "email_template": "test1.tpl"
   }
}
```


5.2.6. Удаление типа сообщений

Для чтения сообщения необходимо отправить DELETE-запрос на url /subjects/<идентификтор типа сообщений>

Например: DELETE http://localhost/subjects/241b5680-4696-4e70-a8b6-14adcd68971d

5.3. Управление получателями сообщений

Система рассылает сообщения только, заранее настроенным, получателям. Для того, чтобы

Система могла доставить сообщение, получателя необходимо внести его в базу данных.

5.3.1. Создание получателя сообщений

Для создания типа сообщений, надо отправить PUT запрос на url /recipients . Тело запроса должно содержать JSON-объект с параметрами:

- caption — наименование получателя сообщений. Используется только для того, чтобы администратор мог легко отличать получателей сообщений между собой
- params — параметры получателя сообщений. Этот объект может содержать любые настройки как самого получателя сообщений, так и настройки типов или маршрутов сообщений. При отправке сообщения, формируется консолидированные json-объект, в котором настройки получателя сообщений имеют приоритет над настройками типа сообщений или маршрута. Это позволяет, например, задать персональный шаблон сообщений для конкретного пользователя.
    - lang — язык пользователя. Система будет искать шаблон сообщения на этом языке. Если шаблон с таким языком не будет найден — будет использован русский язык (ru)

Например: PUT http://localhost/recipients

```
{    "caption": "test",
   "params": {
      "lang": "ru"
   }
}
```

В качестве ответа, Система возвращает уникальный идентификатор созданного получателя сообщений. Этот идентификатор, в дальнейшем, будет использоваться для управления этим получателем сообщений, а также для его авторизации в Системе. Например:

```
{
   "result": "241b5680-4696-4e70-a8b6-14adcd68971d",
}
```

5.3.2. Чтение свойств получателя сообщений

Для чтения свойств получателя сообщения необходимо отправить GET-запрос на url /recipients/<идентификтор типа сообщений>

Например: GET http://localhost/recipients/241b5680-4696-4e70-a8b6-14adcd68971d

В ответе Система возвращает свойства указанного типа сообщений. Например:

```
{
      "caption": "test",
      "params": {
         "lang": "ru"
```

```
    }
}
```

5.3.3. Чтение списка получателей сообщений в Системе

Для чтения списка получателей сообщения необходимо отправить GET-запрос на url /recipients

Например: GET http://localhost/recipients

```
{
  "result": [
    {
      "id": "09d49abe-cc01-494e-b69e-99cb9aab7eb4",
      "caption": "user",
      "params": {
        "lang": "ru",
      }
    },
    {
      "id": "241b5680-4696-4e70-a8b6-14adcd68971d",
      "caption": "test",
      "params": {
        "test": "test"
      }
    }
  ]
}
```

5.3.4. Изменение получателя сообщений

Для изменения надо отправить POST запрос на url / recipients В теле запроса должен быть JSON-объект с параметрами:

- id — идентификатор получателя сообщения
- caption — наименование получателя сообщений. Используется только для того, чтобы администратор мог легко отличать получателей сообщений между собой
- params — параметры получателя сообщений.

Например: POST http://localhost/ recipients

```
{
   "id": "241b5680-4696-4e70-a8b6-14adcd68971d",
  "caption": "test1",
  "params": {
```

```
    "test1": "test1"
  }
}
```

5.3.5. Удаление получателя сообщений

Для удаления получателя сообщения необходимо отправить DELETE-запрос на url /recipients/<идентификтор типа сообщений>

Например: DELETE http://localhost/recipients/241b5680-4696-4e70-a8b6-14adcd68971d

5.4. Управление списками рассылки

Систему можно настроить таким образом, чтобы часть сообщений отправлялась одной группе пользователей, а часть другой. Это соответствие между типом сообщений и пользователем в Системе называется «маршрут».

5.4.1. Создание или модификация маршрута сообщений

Для создания маршрута сообщений, надо отправить PUT запрос на url /routes В теле запроса должен быть JSON-объект с параметрами:

- subject_id — идентификатор типа сообщений
- recipient_id — идентификатор получателя сообщений
- params — параметры маршрута сообщений. Этот объект может содержать любые настройки, которыми можно переопределить настройки типа сообщений. Это позволяет, например, задать персональный шаблон сообщений для конкретного пользователя.

Например: PUT http://localhost/routes

```
{
  "subject_id": "241b5680-4696-4e70-a8b6-14adcd68971d",
  "recipient_id": "09d49abe-cc01-494e-b69e-99cb9aab7eb4",
  "params": { }
}
```

5.4.2. Чтение свойств маршрута сообщений

Для чтения маршрута необходимо отправить GET-запрос на url /routes/<идентификтор типа сообщений>/<идентификатор получателя сообщений>

Например: GET http://localhost/routes/241b5680-4696-4e70-a8b6-14adcd68971d/09d49abe-cc01-494e-b69e-99cb9aab7eb4

В ответе Система возвращает свойства указанного маршрута сообщений. Например:

```
{
   "result": {},
}
```


5.4.3. Чтение списка маршрутов сообщений в Системе

Для чтения маршрута необходимо отправить GET-запрос на url /routes

Например: GET http://localhost/routes


```
{
   "result": [
      {
         "subject_id": "fa791266-d945-406c-9be5-a74ff95f7e60",
         "recipient_id": "09d49abe-cc01-494e-b69e-99cb9aab7eb4",
         "params": {}
      }
   ]
}
```


5.4.4. Удаление маршрута сообщений

Для удаления маршрута необходимо отправить DELETE-запрос на url /routes/<идентификтор типа сообщений>/<идентификатор получателя сообщений>

Например: DELETE http://localhost/routes/241b5680-4696-4e70-a8b6-14adcd68971d/09d49abe-cc01-494e-b69e-99cb9aab7eb4

5.5. Отправка сообщений

После того, как система была настроена, а пользователи авторизованы, можно приступать к отправке сообщений. Для этого надо отправить POST-запрос на url /message. В теле запроса должен быть JSON-объект с параметрами:

- subject — наименование типа сообщений.
- message_id — идентификатор сообщения
- payload — параметры сообщения. Внутри этого объекта должны содержаться параметры, которые будут использованы при формировании текста сообщения на основании шаблона сообщения.

Например: POST http://localhost/message


```
{
```

```
    "subject": "test",
    "message_id": "1",
    "payload": {
        "test": "test"
    }
}
```