# Краткое описание проекта:

Проект создан для того, чтобы автоматически заполнять атцарат бриют в определенные школы/кружки и т/д/.

В проекте используется RabbitMQ для передачи запросов между различными микро сервисам.

1. В данном проекте каждое утро в 7 часов 15 минут утра включается scheduler-service и проверяет – нужно ли сегодня заполнять какой-либо атцарат бриют.

Существует 2 варианта планирования для заполнения. Это регулярное заполнение (проверяется по дню недели – например – каждый вторник) и одноразовое заполнение (проверяется по текущей дате)

1. Если scheduler-service находит запись для заполнения, он отправляет в соответствующий сервис запрос на заполнение. Отправка запроса может проводиться в tsofim-service, school-service, gymnast-service)   
   При запросе scheduler-service отправляет в соответствующий сервис UuidChild (то есть идентификатор ребенка, для которого необходимо заполнить атцарат бриют)
2. Все сервисы, которые «получают» запросы на заполнение документа, работают по одинаковой схеме.  
   Сервис «слушает» RabbitMQ с определенными настройками, и когда «слышит» свой запрос:
   1. По UuidChild получает из user-service все необходимые ему данные про родителей/ответственное лицо ребенка (такие как имя, номер паспорта и т.д.) – при помощи HystrixApi
   2. По UuidChild получает из children-service необходимые данные про ребенка HystrixApi
   3. При необходимости из базы данных получает информацию про конкретного ребенка, необходимую для заполнения атцарат бриют.
   4. Поcле получения всех необходимых данных, сервис идет по указанному URL на нужную страницу. И заполняет там все поля, необходимые для заполнения документа (Используется Selenium + Firefox driver)
   5. Заполнение любого атцарат бриют заканчивается сообщением, что «документ успешно заполнен». Сервис делает скриншот этого сообщения
   6. Сервис отправляет в email-service все данные ребенка, его родителей и скриншот в формате Base64 (при помощи RabbitMQ)
3. Email-service «слушает» RabbitMQ и если слышит что нужно отправить сообщение – шлет письмо на указаный в сообщении email. Письмо содержит информацию о пользователе (фамилия/имя), скриншот заполненного документа, место, для которого заполнялся документ.
4. Если с письмом что то не так – то email-service отправляет письмо администратору проекта, в котором сообщает что именно произошло и почему письмо не было отправлено.

# Нюансы безопасности:

В проекте используется Spring Security. Аутентификация происходит при помощи токен.

Для получения токен пользователь должен зарегистрироваться в приложении. При регистрации пользователь получает на указанный электронный адрес письмо с временным токеном длят регистрации. Пользователь должен пройти по ссылке с токеном и подтвердить таким образом свой электронный адрес. Только после этого он сможет «войти на сайт» для настройки проекта (внесения всех необходимых данных)

Если пользователь забыл свой пароль -0 он может его заменить. Для этого он должен ввести свой электронный адрес, на который придет письмо с временным токен для изменения пароля. перейдя по ссылке в письме, пользователь сможет ввести новый пароль для своей учетной записи.

# Список микро сервисов

|  |  |  |
| --- | --- | --- |
| Название микро сервиса | Описание микро сервиса | Порт |
| [config-service](#_config-service)  spring.application.name  =config-server | Сервис конфигурации. Хранит ссылку на git-hub. По ссылке хранятся все application-properties всех микро сервисов | 8012 |
| [eureka-service](#_eureka-service)  spring.application.name  =eureka-service | Сервис Эврики. Приложение, в котором хранится информация обо всех клиентских сервисных приложениях. Каждая служба Micro регистрируется на сервере Eureka, и сервер Eureka знает все клиентские приложения, работающие на каждом порту и IP-адресе. | 8761 |
| [zuul-service](#_zuul-service)  spring.application.name  =zuul-service | Приложение-шлюз, которое обрабатывает все запросы и выполняет динамическую маршрутизацию приложений микросервисов | 8011 |
| [user-service](#_user-service)  spring.application.name  = user-service | Сервис, который хранит информацию о пользователе или об ответственном лице для каждого конкретного ребенка.  Функционал – регистрация, логин, обновление пароля + CRUD пользователя | 8090 |
| [children-service](#_children-service)  spring.application.name  = children-service | Сервис, который хранит всю информацию по поводу ребенка. Его имя, фамилию, теудат зеут, линк на родителей или ответственное лицо | 8092 |
| [scheduler-service](#_scheduler-service)  spring.application.name  = scheduler-service | Сервис, который раз в день проверяет базу данных и смотрит, нужно ли сегодня заполнять атцарат бриют. И если нужно – то какой. После того, как scheduler-service выполняет проверку, он отправляет запрос на заполнение документа в необходимый сервис (school-service, tsofim-service или gymnast-service) | 8084 |
| [school-service](#_school-service)  spring.application.name  = school-service | Сервис, который парсит необходимую страничку, и заполняет атцарат бриют для школы для конкретного ученика. После того, как документ заполнен, сервис делает скриншот документа и отправляет его в e-mail сервис | 8085 |
| [tsofim-service](#_tsofim-service)  spring.application.name  = tsofim-service | Сервис, который парсит необходимую страничку, и заполняет атцарат бриют для скаутов для конкретного ученика. После того, как документ заполнен, сервис делает скриншот документа и отправляет его в e-mail сервис | 8086 |
| [gymnast-service](#_gymnast-service)  spring.application.name  = gymnast-service | Сервис, который парсит необходимую страничку, и заполняет атцарат бриют для гимнастики для конкретного ученика. После того, как документ заполнен, сервис делает скриншот документа и отправляет его в e-mail сервис | 8087 |
| email-service  spring.application.name  = email-service | Сервис, который постоянно «слушает» куда нужно отправить письмо, и если в очереди появляется письмо – сервис шлет его по указанному адресу. | 8093 |

# Порядок загрузки

1. Config-service
2. Eureka-service (берет данные для загрузки из config-service)
3. Zuul-service
4. Остальные сервисы

# config-service

хранит в себе ссылку на git репозиторий, в котором хранятся application-properites для все сервисов (https://github.com/smilyk/briut.git

)

# eureka-service

для подключения используется:

|  |  |
| --- | --- |
| spring.security.user.name | user |
| spring.security.user.password | password |

# zuul-service

## безопасность

Требует аутентификации. Без аутентификации работают методы

|  |  |
| --- | --- |
| Подтверждение электронной почты | /user-service/users/v1/email-verification |
| Регистрация пользователя | /user-service/users/v1 |
| логин | /user-service/users/login |
| обновление пароля пользователя | /user-service/users/v1/password-reset-request |
| новый пароль пользователя | /user-service/users/v1/password-reset |

Все остальные методы требуют аутентификации.

# user-service

## описание

сервис, который хранит все данные про пользователя (родителя ребенка и/или ответственное лицо – кто заполняет атцарат бриют)

## база данных

**a\_b\_user-service**

таблицы:

### users

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор пользователя | Автогенерируемый.  Используется как ключ |
| uuidUser | Уникальный идентификатор пользователя | Генерируется в процессе сохранения.  Обязательное поле |
| firstName | Имя пользователя | Обязательное поле |
| secondName | Фамилия пользователя |  |
| mainEmail | Главная электронная почта | Именно на эту электронную почту будет приходить письмо, для подтверждения регистрации и/или изменения пароля.  Обязательное поле |
| altEmail | Альтернативная электронная почта | Нужна, если пользователь хочет регистрироваться с одного адреса, а атцарат бриют получать на другой адрес(скриншоты) |
| password | Пароль | Обязательное поле.  В процессе сохранения в БД шифруется (bCryptPasswordEncoder  ) |
| tz | Номер теудат зэута | Для регистрации это поле не обязательно, но при планировании заполнения атцарат бриют приложение потребует заполнить это поле, так как атцарат бриют требует заполнение тз родителя/ответственного лица |
| confirmEmailToken | Токен подтверждения электронной почты | Токен создается в момент, когда пользователь регистрируется. Этот токен отправляется пользователю на электронный адрес, и при подтверждении почты токен удаляется. Нужен исключительно для подтверждения e-mail |
| deleted | Удаление пользователя | Данные пользователя не удаляются физически из БД. Ставится флаг deleted |
| confirmEmail | По умолчанию false. | Как только пользователь подтверждает свой e-mail поле становится true и пользователь может логинится в приложение. Для пользователей с неподтвержденным  e-mail логин запрещен. |

### resp\_person

таблица, в которой хранятся данные ответственых лиц. Используется тогда, когда атцарат бриют заполняет не родитель ребенка, а другой человек, который является ответственным лицом. Соответственно все скриншоты будут приходить на электронный адрес этого ответственного лица

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор пользователя | Автогенерируемый.  Используется как ключ |
| uuidRespPerson | Уникальный идентификатор пользователя | Генерируется в процессе сохранения.  Обязательное поле |
| firstName | Имя пользователя | Обязательное поле |
| secondName | Фамилия пользователя |  |
| emailRespPerson | Главная электронная почта | Обязательное поле. |
| tz | Номер теудат зэута | Для регистрации это поле не обязательно, но при планировании заполнения атцарат бриют приложение потребует заполнить это поле, так как атцарат бриют требует заполнение тз родителя/ответственного лица |
| deleted | Удаление пользователя | Данные пользователя не удаляются физически из БД. Ставится флаг deleted |

### password\_reset\_tokens

таблица, в которой хранится токен, необходимый для изменения пароля пользователя

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор пользователя | Автогенерируемый.  Используется как ключ |
| Token | Токен | Необходим для изменения пароля пользователя. Отправляется на электронный адрес пользователя, указанный при регистрации (или добавления ответственного лица) |

## функциональность(API)

### Mapping - "/users/v1"

#### регистрация пользователя

Метод POST

API

При регистрации пользователя:

* Создается userUuid
* Шифруется пароль пользователя
* ConfirmEmailToken = создается !!!срок его действия – 10 дней
* Deleted = false
* Отправляется запрос для e-mail сервис для того, что бы отправить письмо пользователю, которое будет хранить в себе token для подтверждения электронного адреса.

При создании пользователя собирается EmailVerificationDto(

String tokenValue;  
String userName;  
String userLastName;  
String email;

) которое передается в e-mail-service для отправки сообщения.

#### передача данных в e-mail-service при регистрации пользователя

Для передачи данных в email-service используются очереди (RabbitMQ)

RabbitMQ настройки –

|  |  |
| --- | --- |
| email.key | сonfemail |
| email.queue | CONFEMAIL |
| email.exchange | Email |

После регистрации пользователь должен подтвердить свой электронный адрес.

#### подтверждение электронной почты

Для того, что бы подтвердить свою электронную почту, пользователь должен открыть письмо и перейти по ссылке, которая указана в письме.

Изображение выглядит как текст

Автоматически созданное описание

При переходе

Метод GET

API /email-verification

Response:

Изображение выглядит как текст

Автоматически созданное описание

Поле confirm-email становится true, confirm\_email\_token становится null

#### смена пароля шаг 1

Метод POST

API /password-reset-request

При этом генерируется запрос в email-service. Происходит отправка письма пользователю с ссылкой, на которую необходимо нажать, что бы изменить пароль.

#### В базе данных в таблице password\_reset\_tokens создается и сохраняется токен для изменения пароля (срок его действия – 1 день)

#### смена пароля шаг 1

Для того, что бы изменить пароль пользователь должен открыть письмо и перейти по указанной в нем ссылке.

Изображение выглядит как текст

Автоматически созданное описание

При переходе

Метод POST

API /email-verification

В базе данных users сохраняется новый пароль,

#### В базе данных в таблице password\_reset\_tokens токен удаляется.

#### Login user

Метод GET

API -> /{uuidUser}

Localhost:0000/users/login

В Headers получаем token

#### Получить пользователя по его Uuid

Метод GET

API -> /{uuidUser}

#### Обновить пользователя по его Uuid

Метод PUT

API -> /{uuidUser}

#### Удалить пользователя по его Uuid

Метод DELETE

API -> /{uuidUser}

/f5279f6b-7eef-4593-adce-ef19a7158e33

#### Получить всех пользователей

Метод GET

### Mapping - /resp\_pers/v1

Обычный CRUD

Методы – создать, получить по Uuid, удалить, обновить, получить список всех ответственных лиц

### Mapping - /users/v1/admin"

Содержит один метод – создание суперадминистратора (нужен для того, что бы получить токен администратора, который необходимо для работы парсерам)

## Безопасность

Доступ к сервису производится посредством проверки token

Без авторизации доступны методы:

/\*\*create admin

/\*\*проверка e-mail

/\*\*изменение пароля

/\*\*вход по ссылке для изменения пароля из письма

# children-service

## описание

сервис, который хранит все данные про ребенка, для которого заполняется атцарат бриют

## база данных

#### a\_b\_user-service

таблицы:

### children

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор ребенка | Автогенерируемый.  Используется как ключ |
| uuidChild; | Уникальный идентификатор ребенка | Генерируется в процессе сохранения.  Обязательное поле |
| firstName | Имя ребенка | Обязательное поле |
| secondName | Фамилия ребенка |  |
| uuidParent | Uuid родителя ребенка |  |
| uuidRespPers | Uuid ответственного лица |  |
| tz | Номер теудат зэута ребенка | Для регистрации это поле не обязательно, но при планировании заполнения атцарат бриют приложение потребует заполнить это поле, так как атцарат бриют требует заполнение тз ребенка |
| deleted | Удаление пользователя | Данные ребенка не удаляются физически из БД. Ставится флаг deleted |
|  |  |  |

## функциональность (API)

Mapping - "children"

Обычный CRUD

Методы – создать, получить по Uuid, удалить, обновить, получить список всех детей

## Безопасность

Доступ к сервису производится посредством проверки token

# scheduler-service

## описание

сервис, который проверяет, нужно ли заполнять атцарат бриют или нет. Если нужно – шлет запрос на определенный сервис.

## база данных

#### a\_b\_user-service

таблицы:

#### plan

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор записи | Автогенерируемый.  Используется как ключ |
| uuidPlan; | Уникальный идентификатор записи | Генерируется в процессе сохранения.  Обязательное поле |
| uuidChild; | Уникальный идентификатор ребенка | Получен при сохранении записи планирования.  Обязательное поле |
| sendEmail | Отправлять ли письмо | Если стоит флаг true то письмо с сообщением о том, что будет заполнен атцарат бриют будет отправлено ответственному лицу/родителю (будет реализовано во второй версии) |
| service | Название сервиса. | Название сервиса, при помощи которого будет происходить заполнение атцарат бриют |
| date | Дата | Дата, когда необходимо заполнить атцарат бриют |
| Monday  Tuesday  Wednesday  Thursday  Friday  Saturday  Sunday | День недели | День недели – день, когда будет заполнен атцарат бриют при условии, что его нужно заполнять регулярно. Например – каждый вторник |
| deleted | Удаление записи | Записи не удаляются физически из БД. Ставится флаг deleted |
|  |  |  |

## функциональность (API)

Mapping - /plan/v1

Обычный CRUD

Методы – создать, получить по uuidChild, получить по uuidPlan, удалить, обновить, получить список всех запланированных записей

## отправка сообщений для микросервисов

### school-service

отправка сообщения на API …/{childUuid}

Отправка сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| school.key | sc |
| school.queue | SCHOOL |
| rabbitmq.exchange | atBriut |

### tsofim-service

отправка сообщения на API …/{childUuid}

Отправка сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| tsofim.key | ts |
| tsofim.queue | TSOFIM |
| rabbitmq.exchange | atBriut |

### gymnast-service

отправка сообщения на API …/{childUuid}

Отправка сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| gymnsat.key | gym |
| gymnsat.queue | GYMNAST |
| rabbitmq.exchange | atBriut |

# school-service

## описание и функциональность

Сервис ждет от scheduler-service сообщение о том, что необходимо заполнить атцарат бриют.

1. Получение сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| school.key | sc |
| school.queue | SCHOOL |
| rabbitmq.exchange | atBriut |

1. После того, как сервис заполняет атцарат бриют, он делает скриншот экрана, и отправляет данные о том, что нужно отправить письмо со скриншотом ответственному лицу/родителю ребенка

Данные отправляются в email-service при помощи RabbitMQ

RabbitMQ настройки –

|  |  |
| --- | --- |
| email.key | email |
| email.queue | EMAIL |
| email.exchange | Email |

В запросе email-service содержатся:

EmailDto

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| email | электронная поста, на которую нужно будет отправить письмо |
| picture | Скриншот экрана (Base64) |
| lastName | Фамилия родителя/ответственного лица |
| firstName | Имя родителя/ответственного лица |
| childFirstName | Имя ребенка |
| childSecondNam | Фамилия ребенка |
| service | School-service (название сервиса, от которого исходит запрос об отправке письма) |

1. Для того, что бы заполнить атцарат бриют сервису необходимо получить данные ребенка и его ответственного лиц/родителя.

Для этого используется HystrixApi:

|  |  |
| --- | --- |
| Название | Метод |
| Получение данных о ребенке – запрос в child-service | @GetMapping("/child/v1/{uuidChild}") @Headers("Authorization: {token}") public Response getChildByChildUuid(@PathVariable String uuidChild,  @RequestHeader("Authorization") String token); |
| Получение данных о ответственном лице – запрос в user-service | @GetMapping("/resp\_pers/v1/{uuidRespPerson}}")  @Headers("Authorization: {token}")  public Response getResponsePersonByUserUuid(@PathVariable String uuidRespPerson,  @RequestHeader("Authorization") String token); |
| Получение данных о родителе– запрос в user-service | @GetMapping("/users/v1/{uuidUser}") @Headers("Authorization: {token}") public Response getUserByUserUuid(@PathVariable String uuidUser, @RequestHeader("Authorization") String token); |

## База данных

**a\_b\_school-service**

### таблицы:

### school\_details

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор пользователя | Автогенерируемый.  Используется как ключ |
| uuidChildDetails | Уникальный идентификатор записи для ребенка | Обязательное поле |
| uuidChild | Уникальный идентификатор ребенка | Обязательное поле |
| schoolUserName | Логин для школы | Выдается в школе |
| schoolPassword | Пароль для школы | Выдается в школе |

## функциональность (API)

Mapping – “/school/v1”

Обычный CRUD

Методы – создать, получить по childUuid, получить по planUuid, удалить, обновить, получить список всех записей

## Безопасность

Доступ к сервису производится посредством проверки token

# tsofim-service

## описание и функциональность

Сервис ждет от scheduler-service сообщение о том, что необходимо заполнить атцарат бриют.

1. Получение сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| tsofim.key | ts |
| tsofim.queue | TSOFIM |
| rabbitmq.exchange | atBriut |

1. После того, как сервис заполняет атцарат бриют, он делает скриншот экрана, и отправляет данные о том, что нужно отправить письмо со скриншотом ответственному лицу/родителю ребенка

Данные отправляются в email-service при помощи RabbitMQ

RabbitMQ настройки –

|  |  |
| --- | --- |
| email.key | email |
| email.queue | EMAIL |
| email.exchange | Email |

В запросе email-service содержатся:

EmailDto

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| email | электронная поста, на которую нужно будет отправить письмо |
| picture | Скриншот экрана (Base64) |
| lastName | Фамилия родителя/ответственного лица |
| firstName | Имя родителя/ответственного лица |
| childFirstName | Имя ребенка |
| childSecondNam | Фамилия ребенка |
| service | sofim-service (название сервиса, от которого исходит запрос об отправке письма) |

1. Для того, чтобы заполнить атцарат бриют сервису необходимо получить данные ребенка и его ответственного лиц/родителя.

Для этого используется HystrixApi:

|  |  |
| --- | --- |
| Название | Метод |
| Получение данных о ребенке – запрос в child-service | @GetMapping("/child/v1/{uuidChild}") @Headers("Authorization: {token}") public Response getChildByChildUuid(@PathVariable String uuidChild,  @RequestHeader("Authorization") String token); |
| Получение данных о ответственном лице – запрос в user-service | @GetMapping("/resp\_pers/v1/{uuidRespPerson}}") @Headers("Authorization: {token}") public Response getResponsePersonByUserUuid(@PathVariable String uuidRespPerson,  @RequestHeader("Authorization") String token); |
| Получение данных о родителе– запрос в user-service | public interface UserServiceClient {  @GetMapping("/users/v1/{uuidUser}")  @Headers("Authorization: {token}")  public Response getUserByUserUuid(@PathVariable String uuidUser,  @RequestHeader("Authorization") String token); |

## База данных

### **a\_b\_tsofim-service**

### таблицы:

### tsofim\_details

|  |  |  |
| --- | --- | --- |
| **Поле** | **Описание** | **Примечания** |
| id | Идентификатор пользователя | Автогенерируемый.  Используется как ключ |
| uuidTsofimDetails | Уникальный идентификатор записи для ребенка | Обязательное поле |
| uuidChild | Уникальный идентификатор ребенка | Обязательное поле |
| place | Место, где дислоцируются скауты | Выдается в школе |
| groupTs | Название группы | Выдается в школе |
| school | Название школы |  |
| childClass | Класс, где учится ребенок |  |
| deleted | Удаление записи | Данные не удаляются физически из БД. Ставится флаг deleted |

## функциональность(API)

Mapping – “/tsofim/v1”

Обычный CRUD

Методы – создать, получить по childUuid, получить по planUuid, удалить, обновить, получить список всех записей

## Безопасность

Доступ к сервису производится посредством проверки token

# gymnast-service

## описание и функциональность

Сервис ждет от scheduler-service сообщение о том, что необходимо заполнить атцарат бриют.

1. Получение сообщения о необходимости заполнения происходит при помощи RabbitMQ.

RabbitMQ настройки –

|  |  |
| --- | --- |
| gymnsat.key | gym |
| gymnsat.queue | GYMNAST |
| rabbitmq.exchange | atBriut |

1. После того, как сервис заполняет атцарат бриют, он делает скриншот экрана, и отправляет данные о том, что нужно отправить письмо со скриншотом ответственному лицу/родителю ребенка

Данные отправляются в email-service при помощи RabbitMQ

RabbitMQ настройки –

|  |  |
| --- | --- |
| email.key | email |
| email.queue | EMAIL |
| email.exchange | Email |

В запросе email-service содержатся:

EmailDto

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| email | электронная поста, на которую нужно будет отправить письмо |
| picture | Скриншот экрана (Base64) |
| lastName | Фамилия родителя/ответственного лица |
| firstName | Имя родителя/ответственного лица |
| childFirstName | Имя ребенка |
| childSecondNam | Фамилия ребенка |
| service | gymnast-service (название сервиса, от которого исходит запрос об отправке письма) |

1. Для того, чтобы заполнить атцарат бриют сервису необходимо получить данные ребенка и его ответственного лиц/родителя.

Для этого используется HystrixApi:

|  |  |
| --- | --- |
| Название | Метод |
| Получение данных о ребенке – запрос в child-service | @GetMapping("/child/v1/{uuidChild}") @Headers("Authorization: {token}") public Response getChildByChildUuid(@PathVariable String uuidChild,  @RequestHeader("Authorization") String token); |
| Получение данных о ответственном лице – запрос в user-service | @GetMapping("/resp\_pers/v1/{uuidRespPerson}}") @Headers("Authorization: {token}") public Response getResponsePersonByUserUuid(@PathVariable String uuidRespPerson,  @RequestHeader("Authorization") String token); |
| Получение данных о родителе– запрос в user-service | public interface UserServiceClient {  @GetMapping("/users/v1/{uuidUser}")  @Headers("Authorization: {token}")  public Response getUserByUserUuid(@PathVariable String uuidUser,  @RequestHeader("Authorization") String token); |

## База данных

НЕТ

## Безопасность

Доступ к сервису производится посредством проверки token

# email-service

## описание и функциональность

Сервис ждет сообщения от других сервисов для того, чтобы отправить письма с необходимым содержанием.

Сервис «слушает» сообщения при помощи RabbitMQЕсли при отправке письма на указанный адрес происходит какой то сбой, то сервис отправляет письма администратору с сообщением о том, что произошёл сбой, и какой