Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА/GRADUATION THESIS

Разработка модуля сбора, трансформации и загрузки данных для платформы, поддерживающей хакатон соревнования

Автор/ Author
Панаиотиди Афанасий Фёдорович
Направленность (профиль) образовательной программы/Major
Программирование и интернет-технологии 2017
Квалификация/ Degree level
Бакалавр
Руководитель ВКР/ Thesis supervisor
Повышев Владислав Вячеславович, Университет ИТМО, факультет информационных технологий и программирования, ассистент (квалификационная категория "ассистент")
Группа/Group
M3406
Факультет/институт/кластер/ Faculty/Institute/Cluster
факультет информационных технологий и программирования
Направление подготовки/ Subject area
09.03.02 Информационные системы и технологии

Обучающийся/Student

Документ подписан
Панаиотиди Афанасий Фёдорович
08.06.2021

(эл. подпись/ signature)

Панаиотиди Афанасий Фёдорович
(Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Документ подписан
Повышев Владислав Вячеславович
08.06.2021

(эл. подпись/ signature)

Повышев Владислав Вячеславович
(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

АННОТАЦИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ / SUMMARY OF A GRADUATION THESIS

Обучающийся/ Student
Панаиотиди Афанасий Фёдорович
Наименование темы ВКР / Title of the thesis
Разработка модуля сбора, трансформации и загрузки данных для платформы, поддерживающей хакатон соревнования
Наименование организации, где выполнена ВКР/ Name of organization
Университет ИТМО

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ/ DESCRIPTION OF THE GRADUATION THESIS

1. Цель исследования / Research objective
Спроектировать и реализовать модуль статистики и триггерных рассылок для платформы с более ста тысячами активных пользователей
2. Задачи, решаемые в ВКР / Research tasks
Анализ архитектуры информационной системы, проектирование модуля, проектирования базы данных. Реализация изменений. Тестирование полученных результатов
3. Краткая характеристика полученных результатов / Short summary of results/conclusions
В ходе работы была проанализирована существующая система ITS и необходимые для разработки модули. Были сформированы функциональные требования, описаны необходимые программные доработки. В последствии описанные методы были реализованы и был получен модуль для сбора, трансформации и загрузки данных, а также возможность получения значений по временному отрезку и отправки уведомлений пользователям.
4. Наличие публикаций по теме выпускной работы/ Have you produced any publications on the topic of the thesis

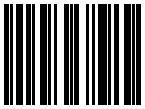
5. Наличие выступлений на конференциях по теме выпускной работы/ Have you produced any conference reports on the topic of the thesis

6. Полученные гранты, при выполнении работы/ Grants received while working on the thesis

7. Дополнительные сведения/ Additional information

Обучающийся/Student

Документ подписан
Панаиотиди Афанасий Фёдорович
08.06.2021



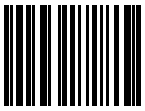
(эл. подпись/ signature)

Панаиотиди Афанасий Фёдорович

(Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Документ подписан
Повышев Владислав Вячеславович
08.06.2021



(эл. подпись/ signature)

Повышев Владислав Вячеславович

(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ / OBJECTIVES FOR A GRADUATION THESIS

Обучающийся / Student Панаиотиди Афанасий Фёдорович
Группа/Group M3406
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Квалификация/ Degree level Бакалавр
Направление подготовки/ Subject area 09.03.02 Информационные системы и технологии
Направленность (профиль) образовательной программы/Major Программирование и интернет-технологии 2017
Специализация/ Specialization
Тема ВКР/ Thesis topic Разработка модуля сбора, трансформации и загрузки данных для платформы, поддерживающей хакатон соревнования
Руководитель ВКР/ Thesis supervisor Повышев Владислав Вячеславович, Университет ИТМО, факультет информационных технологий и программирования, ассистент (квалификационная категория "ассистент")

Срок сдачи студентом законченной работы до / Deadline for submission of complete thesis 15.05.2021

Техническое задание и исходные данные к работе/ Requirements and premise for the thesis
Спроектировать и реализовать модуль сбора, трансформации, загрузки данных и триггерных рассылок для платформы с более ста тысячами активных пользователей

Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов)/ Content of the thesis (list of key issues)
Анализ архитектуры информационной системы.
Проектирование модуля.
Проектирование доработок базы данных.
Реализация модуля
Тестирование

Перечень графического материала (с указанием обязательного материала) / List of graphic materials (with a list of required material)

Исходные материалы и пособия / Source materials and publications
1. Документация фреймворка Flask [Электронный ресурс] – 2021. – URL: https://flask.palletsprojects.com/en/2.0.x/ (дата обращения 17.03.2021).

2. Документация базы данных postgresql [Электронный ресурс] – 2021. – URL: https://www.postgresql.org/docs/9.2/app-psql.html (дата обращения 17.03.2021).
3. Документация Python [Электронный ресурс] – 2021. – URL: https://docs.python.org/3/ (дата обращения 12.04.2021).
4. Святослав Куликов. Тестирование программного обеспечения. Базовый курс – 2021- С.117
5. E.Gamma Design Patterns. Elements of Reusable Object-Oriented Software– 2020 – С.32

Дата выдачи задания/ Objectives issued on 08.06.2021

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/ Thesis supervisor

Документ подписан
Повышев Владислав Вячеславович
08.06.2021

(эл. подпись)

Повышев Владислав Вячеславович

Задание принял к исполнению/ Objectives assumed by

Документ подписан
Панаиотиди Афанасий Фёдорович
08.06.2021

(эл. подпись)

Панаиотиди Афанасий Фёдорович

Руководитель ОП/ Head of educational program

Документ подписан
Зубок Дмитрий Александрович
08.06.2021

(эл. подпись)

Зубок Дмитрий Александрович

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ 5
ВВЕДЕНИЕ 6
ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 7
1.1. Описание прикладного процесса 7
1.2. Требования к модулю 10
1.2.1. Функциональные требования 10
1.2.2. Нефункциональные требования 11
ГЛАВА 2. АНАЛИЗ И ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ 13
2.1. Сбор, трансформация и загрузка данных 15
2.2. Рассылка оповещений 19
ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 25
3.1. Сбор, трансформация, загрузка данных 25
3.2. Рассылка оповещений 31
ЗАКЛЮЧЕНИЕ 36
СПИСОК ИСТОЧНИКОВ 37

ГЛОССАРИЙ

ИС – информационная система

HTTP (Hypertext Transfer Protocol) – протокол передачи данных

ITS – компания, занимающаяся разработкой платформы для проведения хакатон соревнований.

Хакатон – форум для разработчиков, во время которого специалисты решают какую-либо задачу за отведенное время

Снапшот – снимок состояния системы или какого-либо значения в определенный момент времени

Python – язык программирования

Flask – Фреймворк для создания веб-приложения на языке Python

ВВЕДЕНИЕ

Ежегодно проводится большое количество форумов для разработчиков, во время которых специалисты решают какие-либо проблемы за отведенное время, подобные мероприятия называются хакатон. Зачастую формат проведения подразумевает присутствие всех участников в одном месте.

Однако в связи с пандемией COVID-19, случившейся в 2020 году, появилась необходимость проводить соревнования в онлайн формате, т.к. собирать всех в одном месте стало невозможным.

Цифровой прорыв - проект президентской платформы «Россия – страна возможностей». На форуме представлены реальные IT-задачи от бизнеса и государства, а экспертами являются топовые специалисты из индустрии. Платформа предоставляет возможность проведения хакатон соревнований полностью в онлайн формате.

Сбор информации о количестве зарегистрированных участников на мероприятие, количестве команд, отвечающих определенным критериям, местоположении зарегистрировавшихся команд и участников является неотъемлемой частью проведения каждого мероприятия.

Для удобного выполнения вышеописанных операций хорошим решением стал бы модуль, способный получить информацию по заданным критериям, сохранить ее, а также имеющий возможность предоставления нужных данных в момент времени.

Целью работы является разработка данного модуля для предоставления информации об участниках и мероприятиях пользователям веб-интерфейса администратора.

ГЛАВА 1. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной главе описан процесс работы с данными в разрабатываемом модуле. В ходе анализа процесса будут выделены функциональные и нефункциональные требования к модулю, которые в дальнейшем позволят оценить завершенность модуля и проанализировать результаты работы.

1.1. Описание прикладного процесса

В связи с пандемией, случившейся в середине 2020 года, возникла необходимость проведения хакатонов, которые организовывались в офлайн формате, в удаленном режиме. «Цифровой прорыв» стал одним из решений данной проблемы, позволяющей производить регистрацию участников, сбор команд, презентацию заданий и решений участников в удаленном формате.

Платформа «Цифровой прорыв» проводит форумы для разработчиков, на которых представлены задания от множества крупных компаний по различным направлениям.

Система, помимо личного кабинета участника, состоит из личного кабинета организатора и админ панели, с помощью которой можно просматривать и редактировать информацию об участниках, командах, организаторах, а также вносить информацию о новостях, отображаемых на главной странице, партнерах платформы и кейс, представленных на том или ином мероприятии.

Наиболее часто встречаемые варианты использования системы изображены на Рисунке 1

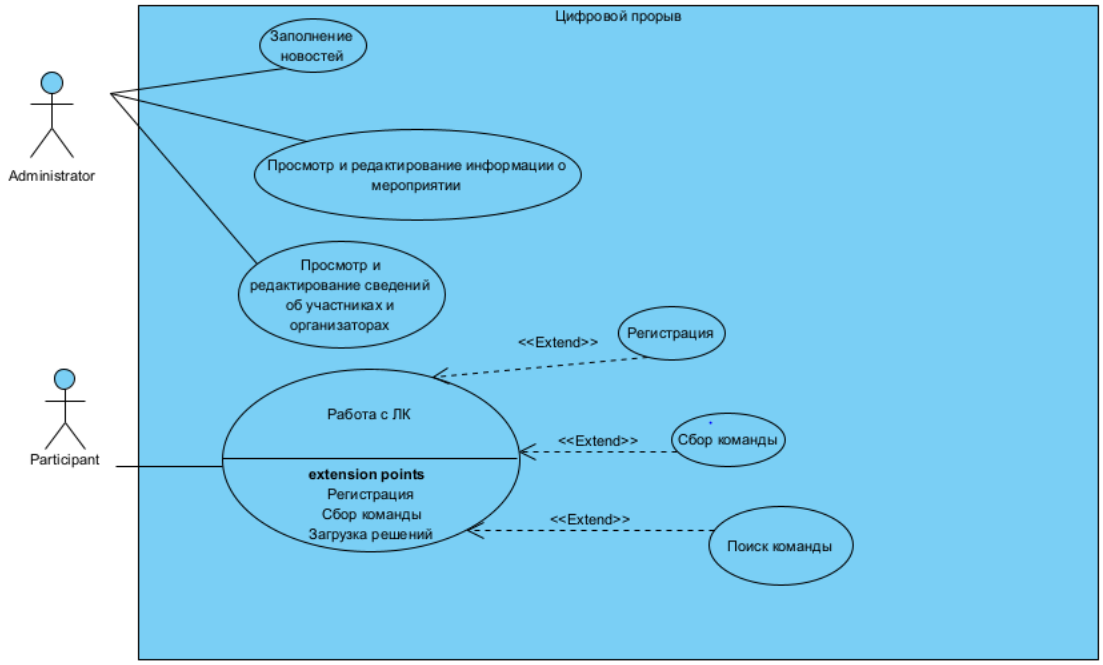


Рисунок 1 Диаграмма вариантов использования

Для участия необходимо зарегистрироваться на сайте, заполнить анкету и собрать команду. Команда допускается до участия в мероприятии, если на момент окончания подачи заявок в ней числится как минимум 3 подтвержденных участника. В противном случае команда дисквалифицируется и не допускается до участия.

Внешний вид страницы с анкетой изображен на Рисунке 2.

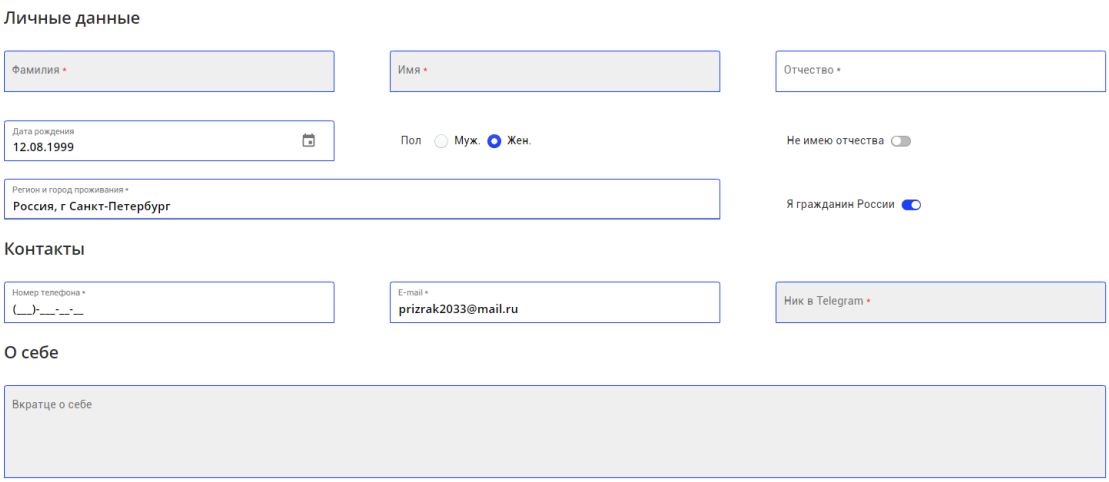


Рисунок 2 Анкета

Так как о платформе можно узнать из разных источников, то в проекте используются UTM метки для получения сведений о ресурсе, который привлек участника.

Администраторы проекта хотят иметь возможность отслеживать информацию о количестве регистраций участников на мероприятие, подтвержденных команд, об участниках, заполнивших анкету и о ресурсах, которые привлекли наибольшее число зарегистрированных пользователей в определенный момент времени.

Этот процесс нуждается в автоматизации, так как на данный момент нет доступных методов получения этой информации и каждый раз требуется делать прямые запросы к базе данных, для чего необходимо участие программиста в данном процессе.

Также при текущей реализации отсутствует возможность получения timeline series по различным значениям.

На данный момент в веб-интерфейсе администратора доступна лишь информация об общем числе регистраций на конкретное мероприятие. Пример изображен на Рисунке 3.

Участники	Команды
1	68
2	7
3	52
4	37
5	47

Рисунок 3 Пример информации о командах

Однако очень часто присутствует необходимость в получении информации не только о числе зарегистрированных участников и команд на конкретное мероприятие, а в какой временной промежуток произошло максимальное число регистраций. Также важным показателем является число общих регистраций на платформе и количество команд, задействованных на всех мероприятиях. Вместе с этим необходимо знать, какое количество

участников заполнили необходимую о себе информацию, а также ссылка с какого ресурса привела их на платформу «Цифрового прорыва»

В связи с этим было принято решение разработать модуль, с помощью которого можно было бы собирать необходимую информацию в нужный момент времени (в данном случае – один раз в 12 часов), сохранять значения и иметь возможность доступа к ним для составления отчетов и графиков.

Разрабатываемый модуль предоставляет возможность сбора, обработки и загрузки информации. А также набор скриптов, вычисляющих желаемые значения по заданным параметрам.

Изначально создается ключ, хранящий в себе путь к методу, умеющему вычислять нужное значение, а также набор параметров, необходимых для этого. Все вычисления реализуются через отдельные python скрипты, которые инициализируются в системе и которые можно импортировать.

Далее по идентификатору ключа можно сделать снимок текущего состояния значения и сохранить в базу.

Модуль позволяет получить информацию о значениях величин в определенный момент времени. И далее эту информацию можно отразить на графиках в панели администратора для более удобного анализа.

После анализа полученной информации у администратора возникает надобность в организации массовой рассылки участникам по тем или иным критерием, следовательно, модуль должен предоставить такую возможность.

1.2. Требования к модулю

1.2.1. Функциональные требования

- Модуль должен получать все необходимые данные с сервера с использованием REST API
- Модуль должен делать снепшот по заданному ключу
- Реализованы методы для вычисления величины снапшота

- Реализовать возможность добавить новые ключи для снапшотов с использованием различных параметров
- Реализовать хранение значений снапшотов для дальнейшего использования
- Реализовать метод для получения time series по снапшотами
- Модуль должен предоставлять возможность рассылки оповещений участникам
- Должна быть возможность использования шаблонов html писем

При текущей реализации участник хакатон никаким образом не оповещается о событиях, произошедших с его командой, будь то входящая в нее заявка, создание или удаления команды.

Также у администраторов проекта нет возможности просматривать информацию о количестве подтвержденных участниках или собранных командах в тот или иной момент времени, т.к эта информация попросту не хранится в базе данных.

Для отслеживания значений необходимы методы, которые бы вычисляли эти значения, используя данные, полученные с сервера ITS.

Необходимо реализовать REST API методы для предоставления информации о снапшотах фронтенду системы. Т.к. каждый снапшот – это отдельное значение, необходим метод, который предоставлял бы множество значений выбранного набора значений по заданному временному промежутку.

1.2.2. Нефункциональные требования

- Использование языка программирование python для написания скриптов
- Документирование api запросов с помощью swagger
- Использование фреймворка Flask для реализации REST API
- Использование postgresql в качестве базы данных

Нефункциональные требования установлены компанией ITS и командой проекта. Уже реализованная система использует данный стек технологий, что является веской причиной для использования аналогичного подхода при реализации модуля

ГЛАВА 2. АНАЛИЗ И ПРОЕКТИРОВАНИЕ АРХИТЕКТУРЫ

Прежде чем анализировать и проектировать структуры модуля, следует обратить внимание на архитектуру всей системы в целом.

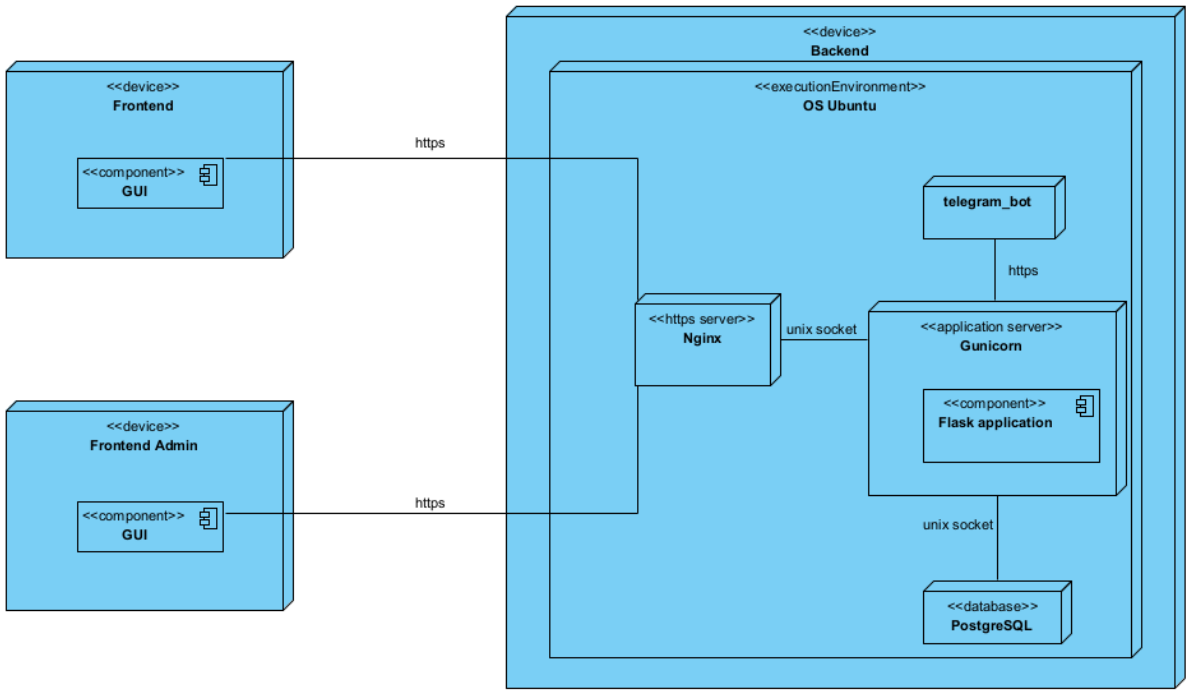


Рисунок 4 Структурная схема

На рисунке 4 представлена структурная схема программной системы, по ней видно, что помимо основной клиентской части предусмотрено веб-приложение для администраторов системы, которое взаимодействует с сервером системы через REST API. Проектируемый модуль для работы с данными должен предоставлять информацию именно для приложения администратора, а, следовательно, запросы должны быть защищены runet токеном, используемым для аналогичных запросов для этой часть веб приложения.

Для массовых и триггерных рассылок будет использоваться бот в социальной сети Телеграмм. Запросы к боту должны быть также защищены токеном.

Кроме структурной схемы, для понимания системы в целом очень важно знать и то, какие данные в ней хранятся. Это позволить более четко понимать,

какую информацию предоставляет та или иная сущность и какие сущности будет необходимо добавить для реализации модуля. Текущая схема модели данных представлена на рисунке 5. Это не все данные, хранящиеся в системе, но именно с этими данными будет производиться большинство взаимодействий в проектируемом модуле.

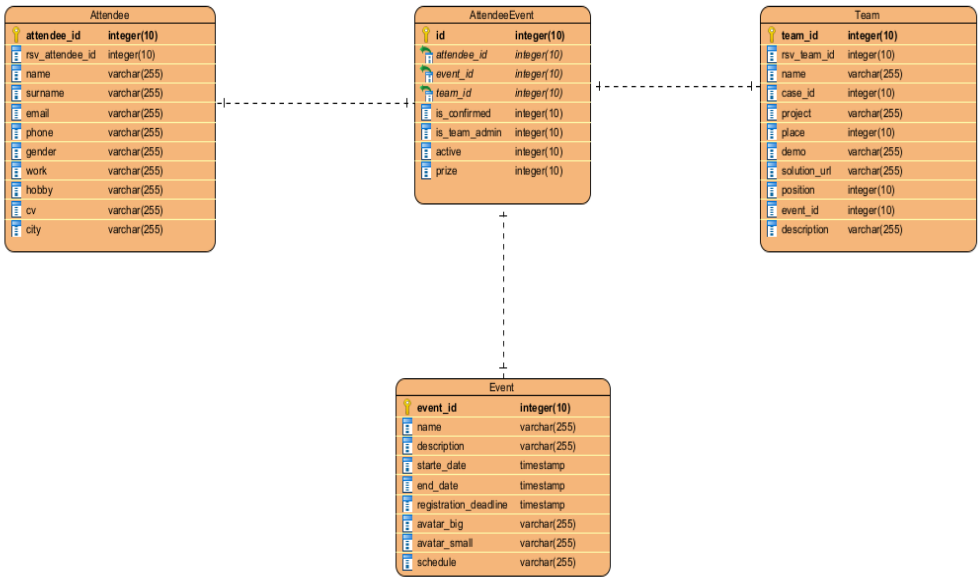


Рисунок 5 Схема данных

Коротко о каждой сущности:

- Attendee(участник) - информация об участнике. Его имя, фамилия, контактная информация, utm маркер.
- Event(мероприятие) - здесь хранится информация о конкретном мероприятии, проводимом на платформе. Название, описание, сроки проведения, дедлайн регистрации.
- Team(команда) - таблица, в которой содержатся данные о команде. Ее название, кейс, в котором команда принимает участие, ссылки на решение, презентацию и видео о работе команды во время хакатона, а также итоговое место команды в общем зачете.
- AttendeeEvent(участник-мероприятие) - таблица-связка, содержащая информация о том, на какое мероприятие зарегистрирован участник и к

какой команде он относится, а также, является ли данный участник капитаном команды

2.1. Сбор, трансформация и загрузка данных

Для получения нужных значений должны быть реализованы скрипты, обращающиеся к базе данных и взаимодействующие с описанными ранее сущностями. Данные скрипты должны быть импортируемы, чтобы была возможность вызвать их, используя REST API методы и подставить нужные параметры, полученные в виде json объекта.

Для вызова того или иного скрипта и снимке значения необходимо предварительно создать snapshot-key - ключ, который хранит в себе вызываемый метод, а также параметры вызова данного метода.

Набор скриптов, которые необходимо реализовать для первоначальной работы модуля сформирован исходя из желаемой для отслеживания информации для администраторов платформы.

Список скриптов и краткое описание для каждого из них:

- Количество участников, зарегистрированных на мероприятие. Для более удобного использование данную функцию нужно вызывать, передав event_id - уникальный идентификатор мероприятия в качестве параметра
- Количество команд, зарегистрированных на мероприятие. По аналогии с функцией выше уникальный идентификатор мероприятия - параметр, необходимый для вызова функции.
- Количество команд, зарегистрированных на мероприятие и допущенных к участию. Допущенной к участию считается команда, в которой есть как минимум 3 участника, подтвердивших свое участие на мероприятии

- Список utm маркеров и количество участников по каждому маркеру. Также необходим скрипт для автоматического отслеживания новых маркеров и создания snapshot-key для них

  После создания ключа с указанием параметров и необходимого метода для подсчета, появляется возможность сделать снимок текущего состояния этого значения, т.е подсчитать значение и сохранить в базу. Для этого необходимо реализовать REST API метод, вычисляющий значение по идентификатору ключа и сохраняющий его в базе данных системы.

  Поэтому необходима сущность, хранящая информацию о ключе снапшота, т.е. его название, вызываемый метод и параметры, а также сущность, которая будет использоваться для сохранения значений и времени, когда значение было получено.

  Для этого были спроектированы две таблицы, которые будут добавлены в базу данных и которые можно увидеть на рисунке 6.

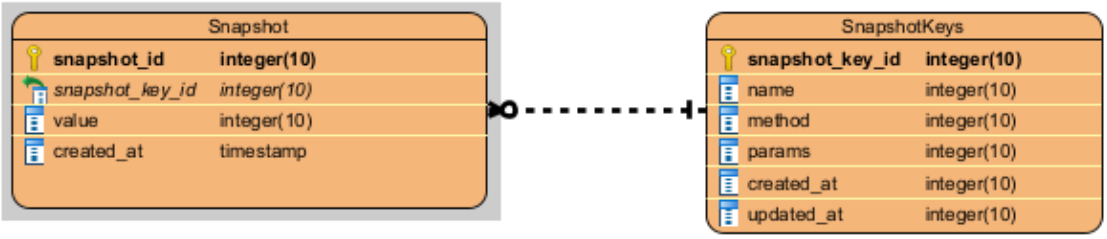


Рисунок 6 Модель данных снапшотов

1. SnapshotKeys - таблица, которая хранит в себе название, строковый путь к методу, параметры вызова, а также дату

создания и изменения ключа, необходимого для дальнейших операций со снапшотами

2. Snapshot - таблица, хранящая в себе информацию о конкретном снапшоте - его значение, идентификатор ключа и время, когда был сделан снимок

После того как сделан снимок, информацию о значении нужно уметь получить с помощью GET запроса, который и нужно реализовать. Помимо общей информации, требуется получать time series для конкретного значения. Для этого будет реализован запрос, принимающий ключ снапшота и временные рамки и возвращающий все значения, сделанные в заданном промежутке.

Опишем все запросы, необходимые для корректной работы:

Метод	Маршрут	Параметры запроса	Описание
GET	/snapshot_keys		Возвращает все созданные ключи для снапшотов
POST	/snapshot_keys	name - название method - строковый путь params - параметры в виде строки, представляющей json-объект	Создание ключа для снапшота

Метод	Маршрут	Параметры запроса	Описание
PUT	/snapshot_keys	key_id - идентификатор ключа снапшота name - название ключа снапшота method - строковый путь к методу для вычисления величины снапшота params - параметры для вычисления снапшота в виде строки,представляющей json-объект	Редактирование жключа для снапшота
POST	/snapshot_keys/utm_source_monitor		Создание ключей для снапшотов, если в системе появились новые utm маркеры

Метод	Маршрут	Параметры запроса	Описание
GET	/snapshot		Возвращает все значения снапшотов
POST	/snapshot	snapshot_key_ids - массив идентификаторов ключей снапшотов, для которых нужно сделать снимок значения	Для каждого snapshot_key_id делает снапшот и сохраняет значение
GET	/snapshot/timeseries	snapshot_key_ids - массив идентификаторов ключей снапшотов date_from, date_to - время создания снапшота	Возвращает все значения снапшотов, созданных по snapshot_key_id и удовлетворяющ их временному отрезку

Таблица 1 Методы для снапшотов

Программную архитектуру модуля можно наблюдать на диаграмме классов

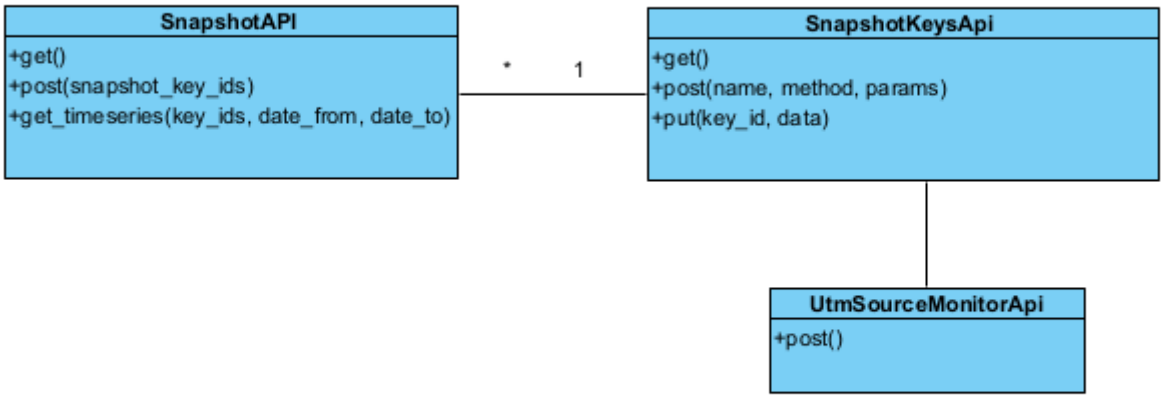


Рисунок 7 Диаграмма классов

2.2. Рассылка оповещений

Для рассылок используется Телеграмм бот, реализованный другой командой компании ITS. Чтобы начать взаимодействие с ботом, нужно связать пользователя системы(участника) с ботом, т.к. все запросы в личном кабинете участникам работают с использованием rsv-token, а для запросов к боту используется bot-token.

Помимо attendee_id для участников введем новое поле telegram_chat_id - уникальный идентификатор чата с ботом. Чтобы получить telegram_chat_id и внести его в базу, пользователь должен перейти по ссылке, сгенерированной на основе его attendee_id и ключа sha256.

sha256 используется для того, чтобы нельзя было получить алгоритм генерации ссылки и привязаться к чату бота другого пользователя.

После того как пользователь перешел по ссылку, ему присваивается telegram_chat_id и с этого момента у нас появляется возможность использовать api чат-бота и посылать пользователю уведомления.

Процесс привязки пользователя к боту можно наблюдать на Рисунке 8

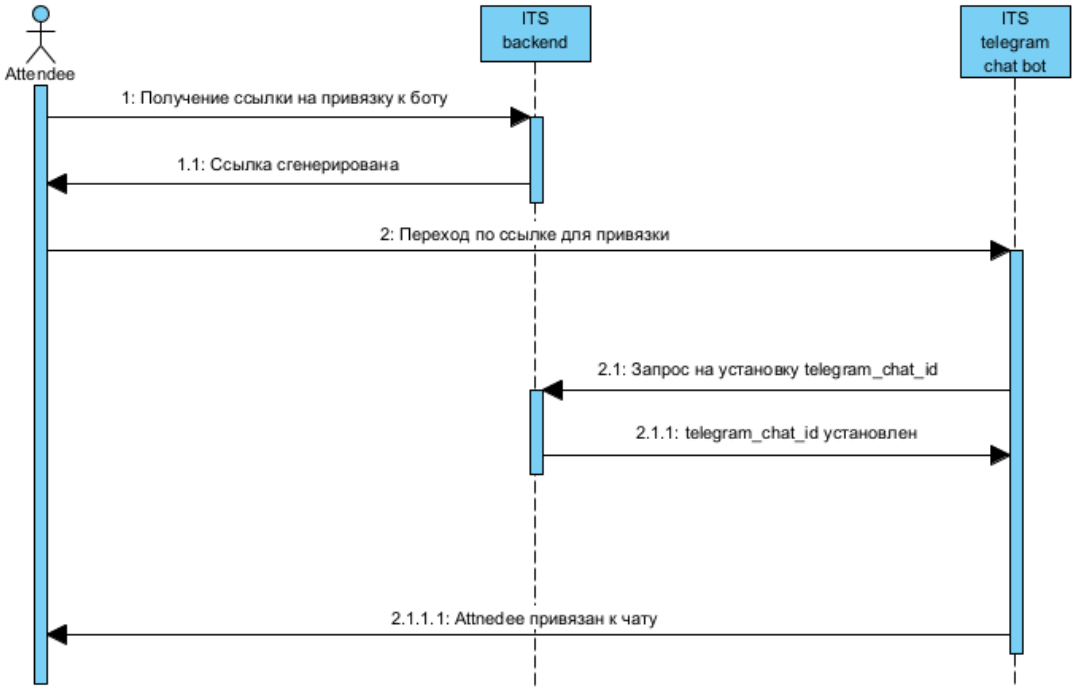


Рисунок 8 Диаграмма последовательности действий для привязки пользователя к чат-боту

Для этого необходимо реализовать метод, генерирующий ссылку для привязки к боту пользователю. Данный метод должен быть защищен rsv токеном и получать данные о пользователе, используя этот токен.

Ссылка будет генерироваться по следующему правилу:

{BOT_START_LINK}{CODE}, где CODE = {attendee_id}_{sha256(attendee_id+secret_key)}

Таким образом присутствует возможность узнать идентификатор пользователя в системе по CODE, однако невозможно подобрать ссылку для другого пользователя

Также необходим метод, с помощью которого бот сможет установить telegram_chat_id пользователя в системе ITS после прохождения по сгенерированной ссылке.

Список необходимых методов выглядит следующим образом

Метод	Маршрут	Параметры запроса	Описание
GET	/attendee/me/get_bot_link		Генерирует ссылку для авторизации в чат боте
PUT	/bot/attach	telegram_chat_id - идентификатор чата с ботом CODE tg - ник пользователя в Телеграм, если он не совпадает с ником, установленным в системе ITS	Устанавливает пользователю telegram_chat_id, связывая с конкретным чатом в боте

Таблица 2Методы для привязки участника к боту

Далее появляется возможность использовать api чат бота для отсылки уведомлений, для этого рассмотрим нужный запрос

Метод	Маршрут	Параметры запроса	Описание
POST	/notification/send/new	notifications - массив уведомлений: [telegram_chat_id - идентификатор чата пользователя attendee id - идентификатор пользователя в системе ITS text - текст уведомления]	Отправляет уведомление в telegram для каждого telegram_chat_id

Таблица 3Метод для отправки уведомлений

Программную архитектуру можно наблюдать на рисунках 9, 10

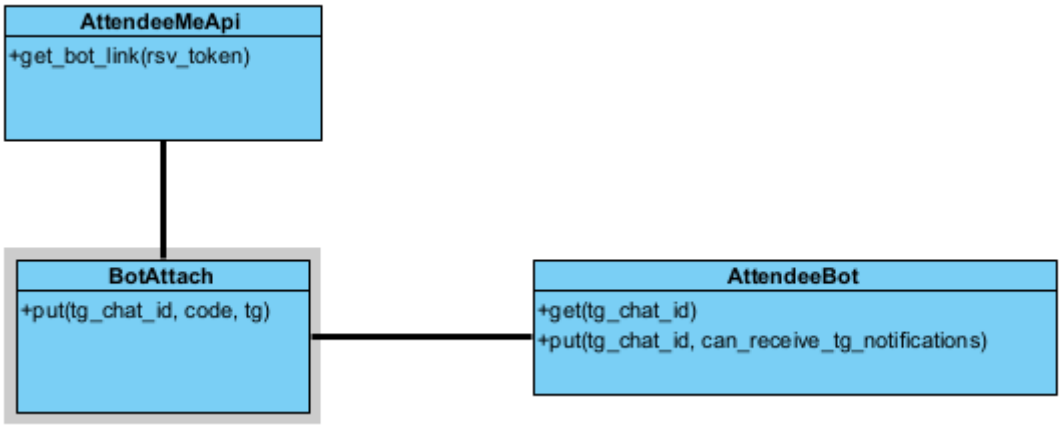


Рисунок9 Диаграмма классов BotApi

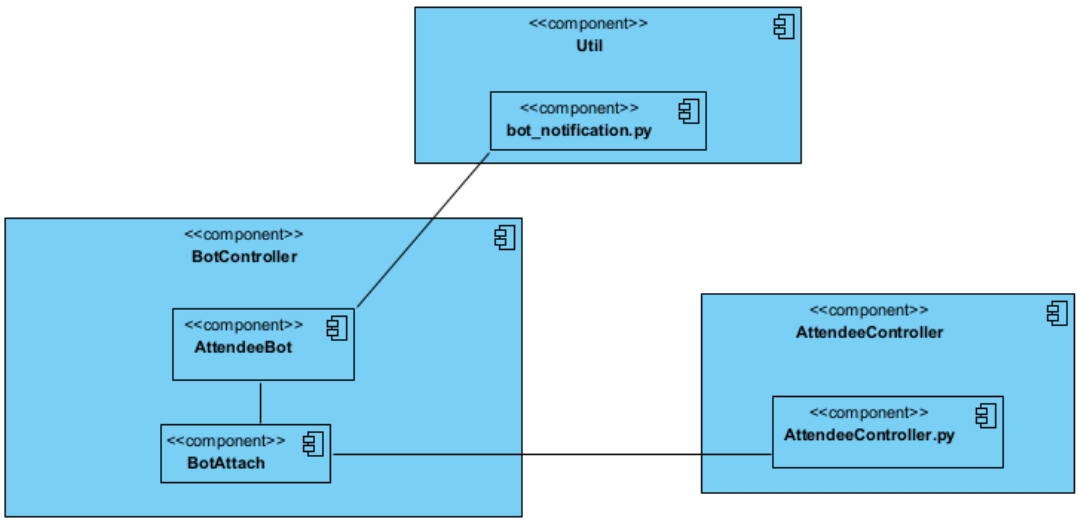


Рисунок10 Диаграмма компонентов модуля рассылки оповещений

Для того, чтобы реализовать триггерные рассылки понадобится аналогичная функция.

Рассмотрим ситуации, в которых необходимо автоматически отправлять уведомление пользователям:

1. При создании команды уведомление создателю о том, что действие произошло
2. При удалении команды оповещением всем ее участникам.

3. Если участник подает заявку на вступление в команду - оповещение админу этой команды о входящей заявке
4. Если команда приглашает участника вступить - оповещение участнику о пришедшей заявке
5. Заявка о вступлении в команду была принята/отклонена - оповещение участнику о результате заявки
6. Приглашение на вступление в команду было принято участником - оповещение всем участникам команды о новом товарище

Для каждой из них необходима функция, которая будет вызываться в момент выполнения описанного действия и отправлять уведомления пользователю.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1. Сбор, трансформация, загрузка данных

Для реализации данной части модуля использовался фреймворк flask, позволяющий создать REST API [1]

Первоначально необходимо описать используемые данные.

Таблица snapshot_keys должна хранить имя, используемый метод для вычисления снапшота, параметры вызова этого метода и время, когда ключ был создан.

Создание класса для таблицы snapshot_keys - представлено на Листинге 1

```
class SnapshotKeys(EntityBase):
    __tablename__ = 'snapshot_keys'

    snapshot_key_id = Column(Integer, seq, primary_key=True)
    name = Column(String)
    method = Column(String)
    params = Column(String)
    created_at = Column(DateTime, default=EntityBase.now)
    updated_at = Column(DateTime, default=EntityBase.now,
onupdate=EntityBase.now)
    is_active = Column(Boolean)
```

Листинг 1 Реализация модели snapshot_keys

Далее нам необходимы методы для создания объекта, получения объекта по его snapshot_key_id.

Одним из важных методов является получение всех доступных ключей снапшотов, хранимых в базе данных, а также получение всех имен ключей, чтобы можно было выяснить, какие из снапшотов можно сделать на текущий момент и сохранить в базе данных.

В листинге 2 представлен код, создающий описанные методы для класса SnapshotKeys

```
@classmethod
def get_snapshot_key_by_id(cls, snapshot_key_id):
    return cls.dict_item(cls.query.filter_by(snapshot_key_id=snapshot_key_id).first())
@classmethod
def get_snapshot_keys(cls):
    return [snapshot_key.to_dict() for snapshot_key in
cls.query.order_by(cls.snapshot_key_id).all()]

@classmethod
def get_snapshot_keys_names(cls):
    return [snapshot_key.to_dict()['name'] for snapshot_key in
cls.query.order_by(cls.snapshot_key_id).all()]

@classmethod
def create(cls, data):
    snapshot_key = cls(name=data['name'], method=data['method'], params=data['params'],
is_active=True)
    snapshot_key.add()
    snapshot_key_dict = snapshot_key.to_dict()
    return snapshot_key_dict
```

Листинг 2 Реализация методов класса SnapshotKeys

Так как в проекте отсутствует возможность автоматической миграции новых таблиц в базу данных, необходимо выполнить sql-запрос для создания таблицы snapshot_keys.

Данный запрос представлен на листинге 3

```
CREATE TABLE public.snapshot_keys
(
    snapshot_key_id integer,
    name character varying,
    method character varying,
    params character varying,
    created_at timestamp without time zone,
    updated_at timestamp without time zone,
    is_active boolean,
    PRIMARY KEY (snapshot_key_id)
);
```

Листинг 3 Создание таблицы snapshot_keys в БД

Затем аналогичным образом опишем таблицу Snapshot, в которой и будут храниться значения.

Данная таблица должна содержать идентификатор ключа, по которому был сделан снапшот, значение самого снапшота и время, в которое он был сделан,

чтобы в последствии иметь возможность отобразить значения на временном отрезке

Запрос для создания таблицы в базе данных представлен на Листинге 4.

Реализация класса Snapshot представлена на Листинге 5.

```
CREATE TABLE public.snapshot
(
    snapshot_id integer,
    snapshot_key_id integer,
    value double precision,
    created_at timestamp without time zone,
    PRIMARY KEY (snapshot_id)
);
```

Листинг 4 Создание таблицы snapshots в БД

```
class Snapshot(EntityBase):
    __tablename__ = 'snapshot'

    snapshot_id = Column(Integer, seq, primary_key=True)
    snapshot_key_id = Column(Integer)
    value = Column(Float)
    created_at = Column(DateTime, default=ceil_dt_5_minutes)
    serialize_items_list = ['snapshot_id', 'snapshot_key_id', 'value', 'created_at']

    @classmethod
    def get_snapshot_by_id(cls, snapshot_id):
        return cls.dict_item(cls.query.filter_by(snapshot_id=snapshot_id).first())
      @classmethod
def get_snapshots(cls):
    return [snapshot.to_dict() for snapshot in cls.query.order_by(cls.snapshot_id).all()]

@classmethod
def get_snapshots_by_filter(cls, data):
    query = cls.query.filter_by(snapshot_key_id=data['key_id'])
    if data['date_from'] is not None:
        query = query.filter(data['date_from'] <= cls.created_at)
    if data['date_to'] is not None:
        query = query.filter(data['date_to'] >= cls.created_at)
    key_ids_dicts = [snapshot.to_dict() for snapshot in query.all()]
    return [[key_ids_dict['created_at'], key_ids_dict['value']] for key_ids_dict in key_ids_dicts]

@classmethod
def create(cls, snapshot_key_id, value):
    snapshot = cls(snapshot_key_id=snapshot_key_id, value=value)
    snapshot.add()
    snapshot_dict = snapshot.to_dict()

    return snapshot_dict
```

Листинг 5 Реализация класса Snapshot

Далее необходимо реализовать набор скриптов, которые будут использоваться для подсчета значений.

На Листинге 6 представлен один из скриптов, считающих количество команд, имеющих 3 и более подтвержденных участника.

Чтобы реализовать это действие необходимо соединить две таблицы Team и AttendeeEvent, для которых идентификатор команды совпадает, отфильтровать по идентификатору мероприятия, а также посчитать количество участников, попавших в группировку. И выбрать только те команды, которые удовлетворяют условию наличия трех и более участников. Остальные скрипты реализованы аналогично.

```
def event_teams_count_with_size(data):
    teams = db.session.query(Team,
func.count()).outerjoin(AttendeeEvent, AttendeeEvent.team_id ==
Team.team_id)\

.filter_by(event_id=data['event_id']).group_by(Team.team_id).all()
    value = 0
    for team in teams:
        if team[1] >= 3:
            value += 1
    Snapshot.create(data['key_id'], value)
```

Листинг 6 Скрипт для подсчета команд с 3+ участниками

Таким образом все готово для написания запросов на создание и получение snapshot_keys. Для этого создадим контроллер в flask-приложении и укажем его в файле __init__.py , чтобы иметь возможность использовать api запросы данного контроллера при запуске приложения.
Создаем api для данного контроллера и указываем в файле.

app/util/dto.py

```
class SnapshotKeysDto:
    api = Namespace('snapshot_keys', description='Snapshot keys
operations')
```

snapshots_keys_controller.py

```
from ..util.dto import SnapshotKeysDto

api = SnapshotKeysDto.api
```

__init__.py

```
api.add_namespace(snapshot_keys_controller.api, path='/snapshot_keys')
```

Реализация методов GET, POST представлена на Листинге 7

```
@api.route('')
@api.doc(security='runet-token')
class SnapshotKeysApi(Resource):
    @api.doc('get_snapshot_keys')
    @runet_token_required
    @role_access_required('admin')
    def get(self):
        """ Return all snapshot keys"""
        return SnapshotKeys.get_snapshot_keys()

    @api.doc('create_snapshot_key')
    @api.expect(SnapshotKeysDto.snapshot_key_in)
    @api.response(200, 'snapshot keys monitored')
    @api.response(201, 'snapshot created', SnapshotKeysDto.snapshot_key_out)
    @runet_token_required
    @role_access_required('admin')
    def post(self):
        """ Create new snapshot key """
        data = api.payload
        if data['method'] == 'by_utm_source':
            unique_utm_sources = Attendee.get_attendee_unique_utm_sources()
            for unique_utm_source in unique_utm_sources:
                data['name'] = unique_utm_source
                SnapshotKeys.create(data)
            return 201
        else:
            snapshot_key = SnapshotKeys.create(data)
            if snapshot_key:
                return snapshot_key, 201
            api.abort(HTTPStatus.BAD_REQUEST, 'Wrong parameters')
```

Листинг 7Методы для создания и получения snapshot_keys

После создание необходимых методов протестируем, что все работает верно. Чтобы убедиться в правильности работы модуля, было принято решение произвести функциональные тестирование. Далее разобран один из тест-кейсов [4].

Для этого вызовем POST запрос /snapshot_keys со следующими параметрами:

name: "Teams with 3+ members",

method: "teams_count_with_size",

params: ""

Запрос успешно выполнен, о чем свидетельствует ответ, представленный на Рисунке 10.

Рисунок 10 Ответ на POST /snapshot_keys

Далее вызовем GET запрос на получение всех созданных ключей.

В ответе, представленном на Рисунке 11, видим созданный ранее ключ

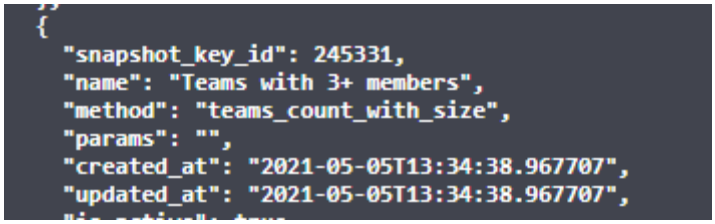


Рисунок 11 Ответ на GET /snapshot_keys

Методы для snapshot_controller реализуются аналогичным образом, этот процесс продемонстрирован в Листинге 8.

```python
@api.route('')
@api.doc(security='runet-token')
class SnapshotApi(Resource):
    @api.doc('get_snapshots')
    @runet_token_required
    @role_access_required('admin')
    def get(self):
        """ Return all snapshots """
        return Snapshot.get_snapshots()

    @api.doc('create_snapshots')
    @api.expect(SnapshotDto.snapshot_in)
    @runet_token_required
    @role_access_required('admin')
    def post(self):
        """ Create new snapshot """
        hooks = get_hooks()
        key_ids = api.payload['snapshot_key_ids']
        if key_ids == -1:
            key_ids = [key['snapshot_key_id'] for key in SnapshotKeys.get_snapshot_keys()]

        for key_id in key_ids:
            snapshot_key = SnapshotKeys.get_snapshot_key_by_id(key_id)
            if snapshot_key is not None:
                if snapshot_key['params'] != '':
                    params = json.loads(snapshot_key['params'])
                else:
                    params = {}
                params['key_id'] = key_id
                hooks[snapshot_key['method']](params)
        return 201
```

Листинг 8 snapshot_controller.py

Также необходимо реализовать метод GET /timeseries, который позволит получать значения по временной выборке. Реализация данного метода продемонстрирована в Листинге 9

3.2. Рассылка оповещений

Первым делом необходимо реализовать метод, генерирующий ссылку для привязки участника к Телеграммам боту.

Для этого сначала реализуем функцию sha256(attendee_id), которая будет использоваться в дальнейшем. Алгоритм получения шифра следующий:

составляется code - строка, содержащая в себе идентификатор пользователя и секретный ключ бота, далее от этой строки берется функция sha256. Получившийся результат и будет использоваться при генерации ссылки. Для того, чтобы нельзя было подобрать code для любого участника, узнав, что используется алгоритм sha256, в конец строки шифрования подставляется секретный ключ, который хранится в переменной окружения и доступ к которому нельзя получить через приложение.

Функцию sha256 можно увидеть в Листинге 10

```python
def sha_256(attendee_id):
    code = f"{attendee_id}_{os.environ['BOT_SECRET_KEY']}"
    return sha256(code.encode()).hexdigest()[:SHA_256_MAX_LEN]
```

Листинг 9 sha256

Реализация метода /attendee/me/get_bot_link представлена в Листинге 11

```python
@api.route('/me/get_bot_link')
class AttendeeBotLinkApi(Resource):
    @api.doc('get_attendee_bot_link', security='rsv-token')
    @api.response(200, 'Success')
    @api.response(401, 'Unauthorized')
    @api.response(404, 'Attendee not found')
    @rsv_token_required
    def get(self):
        """Return attendee bot link"""
        attendee = get_attendee_from_token(api)
        return {
            'bot_link':
f"{os.environ['BOT_START_LINK']}{attendee['attendee_id']}_{sha_256(attendee['attendee_id'])}"}
```

Листинг 10 /attendee/me/get_bot_link

Следующим этапом реализации является добавление функций для отсылки оповещений.

Первоначально необходимо реализовать функцию, принимающую список идентификаторов участников и текст для отправки и вызывающей api метод бота, передавая ему необходимый набор параметров.

Параметры передаются следующим образом: для каждого attendee_id проверяется, есть ли у данного участника привязка к боту. Если привязки нет, данный участник пропускается, иначе в итоговый массив параметров добавляется объект, содержащий в себе attendee_id, telegram_chat_id, text.

Реализация данной функции продемонстрирована на Листинге 12

```python
def bot_notify_now(args):
    data = []
    for _id in args['attendee_id']:
        attendee = Attendee.get_attendee_by_id(_id)
        if attendee is not None and attendee['tg_chat_id'] is not None:
            data.append({
                'attendee_id': attendee['attendee_id'],
                'chat_id': attendee['tg_chat_id'],
                'text': args['content']
            })
    requests.post(notify_now_url, json={'notifications': data}, headers=HEADERS)
```

Листинг 11 Функция для отправки уведомлений

Следующим шагом будет создание функций для каждой из ситуаций, когда необходимо отправить оповещение.

Пример одной из таких функций, использующейся для отправки уведомлений при оповещениях о вступлении нового участника продемонстрирован в Листинге 13

```python
def bot_notify_new_member(attendee, team, team_members, event_id):
    bot_notify_now({
        'attendee_id': [member['attendee_id'] for member in team_members],
        'content': f"Поприветствуй {attendee['name']} {attendee['surname']} --
{attendee['rsv_att_id']}"
                   f" теперь это участник твоей команды <b>{team['name']}</b>.
Дополнительная информация доступна в"
                   f" <a href='{cabinet_url}{event_id}'>Личном кабинете</a>"
    })
```

Листинг 12 Оповещение о новом участнике команды

Остальные функции реализованы аналогично. Далее нужно вызвать описанные функции в необходимых местах. Например, функцию bot_notify_new_member нужно вызывать в момент добавления участника, передав в нее команду, участников, которым необходимо разослать оповещений и идентификатор мероприятия, на которое зарегистрирована команда. Пример приведен в Листинге 14

```
team = get_item(api, args['team_id'], Team)
team_members = AttendeeEvent.get_relations_by_team(team['team_id'])

bot_notify_new_member(attendee, team, team_members, event_id)
```

Листинг 13 Пример вызова триггерной функции

Следующим шагом будет функциональное тестирование работы оповещений. Основным тестовым кейсом является успешная привязка и оповещение о создании команды, т.к. привязка к боту является основным шагом для работы модуля, без которого будет невозможно осуществить дальнейшие действия оповещения пользователя.

Функции оповещения реализованы похожим образом, поэтому ниже приведен сценарий тестирования оповещения, вызываемого при создании участником новой команды.

Для этого необходимо привязать тестовый аккаунт участника к боту, а затем сделать действия, вызывающие отправку уведомлений, чтобы проверить основные методы.

Получим ссылку для привязки к боту, вызвав метод attendee/me/get_bot_link

Результат данного запроса можно наблюдать на Рисунке 12

Рисунок 12 Получение ссылки на авторизацию

Далее перейдем по этой ссылке и увидим, что привязка к боту произошла успешна.

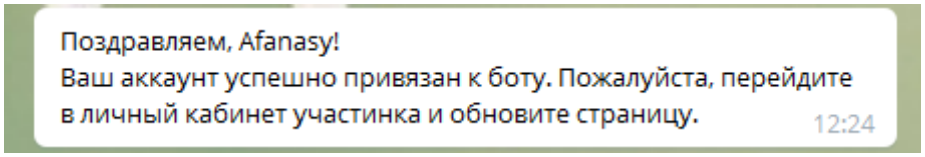


Рисунок 13 Сообщение об успешной привязке

Далее через личный кабинет пользователя создадим команду и увидим, что нам пришло уведомление об этом

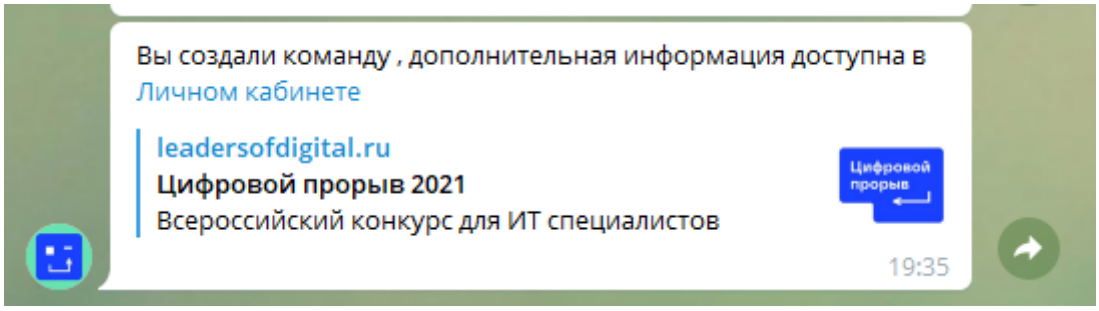


Рисунок 14 Уведомление о создании команды

Остальные функции проверяются аналогичным образом.

ЗАКЛЮЧЕНИЕ

В ходе работы была проанализирована существующая система ITS и необходимые для разработки модули. Были сформированы функциональные требования, описаны необходимые программные доработки. В последствии описанные методы были реализованы и был получен модуль для сбора, трансформации и загрузки данных, а также возможность получения значений по временному отрезку и отправки уведомлений пользователям.

СПИСОК ИСТОЧНИКОВ

1. Документация фреймворка Flask [Электронный ресурс] – 2021. – URL: https://flask.palletsprojects.com/en/2.0.x/ (дата обращения 17.03.2021).
2. Документация базы данных postgresql [Электронный ресурс] – 2021. – URL: https://www.postgresql.org/docs/9.2/app-psql.html (дата обращения 17.03.2021).
3. Документация Python [Электронный ресурс] – 2021. – URL: https://docs.python.org/3/ (дата обращения 12.04.2021).
4. Святослав Куликов. Тестирование программного обеспечения. Базовый курс – 2021- С.117
5. E.Gamma Design Patterns. Elements of Reusable Object-Oriented Software– 2020 – С.32