|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ**

**НА ТЕМУ:**

**Разработка сервиса автоматизации регистрации участников на мероприятия.**

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О.Фамилия)

Москва, 2024 г.

Содержание

[**Введение** 3](#_Toc176783997)

[**1 Обзор предметной области** 4](#_Toc176783998)

[1.1 Обзор классических способов автоматизации регистрации на мероприятия 4](#_Toc176783999)

[1.2 Актуальность автоматизации через telegram бот 5](#_Toc176784000)

[1.3 Практическая реализация 6](#_Toc176784001)

[1.3.1 Предметная область и требования 6](#_Toc176784002)

[1.3.2 Модель «Сущность-связь» 6](#_Toc176784003)

[**2 Проектирование приложения** 11](#_Toc176784004)

[2.1 Предметная область и требования 11](#_Toc176784005)

[2.2 Обеспечение правил минимальной кардинальности. 19](#_Toc176784006)

[**3 Разработка приложения** 25](#_Toc176784007)

[3.1 Разработка архитектуры приложения 25](#_Toc176784008)

[3.2 Выбор технологических средств 26](#_Toc176784009)

[3.3 Ключевые типы данных и функции 27](#_Toc176784010)

[3.3.1 Сервис авторизации 27](#_Toc176784011)

[3.3.2 Сервис работы с ботами 28](#_Toc176784012)

[**4 Тестирование** 31](#_Toc176784013)

[4.1 Разработка тестов 31](#_Toc176784014)

[4.1.1 Генерация ботов 31](#_Toc176784015)

[4.1.2 Регистрация пользователя 33](#_Toc176784016)

[4.2 Тестирование 33](#_Toc176784017)

[**5 Заключение** 36](#_Toc176784018)

[**6 Список использованных источников** 37](#_Toc176784019)

# Введение

В условиях активной общественной жизни и большого числа мероприятий, проводимых в высших учебных заведениях, автоматизация процесса регистрации участников становится важной частью эффективного управления событиями. В частности, в МГТУ имени Н. Э. Баумана ежегодно проводится не менее пятисот различных мероприятий. Важно обеспечить оперативную и удобную регистрацию участников, чтобы каждое событие проходило на высоком организационном уровне.

Быстрая и эффективная регистрация участников на мероприятия позволяет существенно сократить временные затраты и повысить удобство для студентов, преподавателей и гостей университета. Возможность зарегистрироваться на мероприятие онлайн ускоряет процесс для участников, что особенно актуально в условиях большого количества проводимых мероприятий.

Для организаторов автоматизация процесса регистрации открывает дополнительные возможности для управления массовыми мероприятиями. При проведении более пятисот мероприятий в год крайне важно иметь инструменты, которые позволяют эффективно обрабатывать и систематизировать данные о зарегистрированных участниках.

Таким образом, в условиях высокой активности и большого количества проводимых мероприятий, как в МГТУ имени Баумана, автоматизация регистрации является необходимым инструментом для обеспечения успешной организации и проведения событий, а также для повышения уровня удовлетворенности участников и организаторов.

# **1 Обзор предметной области**

## Обзор классических способов автоматизации регистрации на мероприятия

Автоматизировать регистрацию участников на мероприятия можно тремя основными способами:

- Google или Yandex формы;

- Веб-сайт;

- Telegram бот;

В оценке качества различных сервисов регистрации на мероприятия метриками являются удобство для организаторов, трудоёмкость организации рассылок, время на создание регистрации, простота внесения изменений и удобство для конечного пользователя.

Бот в Telegram проявляет себя как удобное решение для организаторов, интегрируясь с системой мессенджера и предоставляя возможности адаптации для различных пользователей. Организации рассылок легко интегрируется на этапе разработки. Создание регистрации требует значительного времени, поскольку связано с разработкой программного продукта. Внесение изменений также трудоёмко из-за необходимости внесения изменений в исходный код программного продукта.

Google или Yandex Форма обеспечивает удобное взаимодействие и быструю настройку регистрации, требующую лишь списка вопросов. Форма не является адаптивной, что ограничивает ее в использовании разными группами участников. Организация рассылок представляет собой ручное написание сообщений каждому участнику. Внесение изменений в форму представляет собой не трудоёмкий процесс.

Веб-сайт предоставляет удобный интерфейс для регистрации и обладает возможностями адаптивности к разным группам участников, если это предусмотрено при разработке. Организация рассылок не представляет собой трудоёмкий процесс, если данный функционал интегрирован при разработке. Создание регистрации занимает много времени из-за трудоёмкого процесса разработки сайта. Внесение изменений также является сложным и требует обращения к разработчикам, что может потребовать дополнительных временных ресурсов.

Классические методы автоматизации регистрации на мероприятия (Google/Yandex формы, веб-сайты и Telegram-боты) обладают различными характеристиками, которые необходимо учитывать при выборе подходящего решения. Google и Yandex формы отличаются простотой настройки и минимальными временными затратами, однако ограничены в функциональности и адаптивности, что делает их менее универсальными для использования с разными группами участников. Веб-сайты и Telegram-боты, напротив, предоставляют широкие возможности для кастомизации и интеграции, однако их создание требует значительных ресурсов, как временных, так и технических, а внесение изменений в такие системы связано с высокой трудоёмкостью.

## Актуальность автоматизации через telegram бот

Автоматизация регистрации с использованием Telegram-бота обладает рядом преимуществ, что делает данный подход предпочтительным для организаторов мероприятий. Прежде всего, Telegram-боты обеспечивают интеграцию с популярной платформой обмена сообщениями, что значительно упрощает взаимодействие пользователей с системой. Это исключает необходимость установки дополнительного программного обеспечения или использования внешних веб-ресурсов, что повышает доступность и удобство регистрации для участников.

Telegram-боты обладают высокой степенью адаптивности, предоставляя организаторам возможность настроить сбор данных в соответствии с требованиями конкретного мероприятия. Инструмент позволяет автоматизировать процесс рассылок, отправку уведомлений и напоминаний, что снижает трудозатраты на коммуникации с участниками. Важным преимуществом является возможность интеграции бота с внешними системами, такими как базы данных или сторонних API, что облегчает хранение и обработку информации.

Хотя разработка Telegram-бота требует первоначальных затрат ресурсов, этот метод позволяет значительно снизить затраты времени на последующих этапах регистрации и обработки данных. Бот также может быть масштабируемым, обеспечивая регистрацию для мероприятий различного масштаба и адаптацию под особенности различных категорий пользователей

Таким образом, при решении проблем, связанных с разработкой и модификацией telegram-бота, будет получен универсальный сервис для автоматизации регистрации участников на мероприятия, сочетающий в себе гибкость, удобство и высокую степень автоматизации регистрации.

## Практическая реализация

### Предметная область и требования

В качестве реализации модели был создан сервис для автоматической генерации telegram ботов, для проведения регистрации на мероприятия, проводимых в МГТУ им. Н.Э. Баумана.

На основании существующих мероприятий можно выделить две основные сущности любого мероприятия:

- Конечный участник мероприятия;

- Организатор мероприятия.

C точки зрения участника сервис должен представлять классический telegram бот-анкету, содержащую конечный набор вопросов.

Организатору, в свою очередь, необходимо предоставить следующие механизмы для проведения регистрации:

- Возможность по заранее описанной схеме сгенерировать telegram бот;

- Возможность создавать несколько telegram ботов;

- Возможность получать доступ к ранее созданным telegram ботам;

- Возможность в произвольный момент времени включить или выключить ранее созданный telegram бот;

### 1.3.2 Модель «Сущность-связь»

В ER модель (рисунок 1) включено восемь сущностей:

1) bot\_owner - сущность, характеризующая пользователя, который создаёт бота, т.е. представителя студенческой организации, проводящей мероприятие. Идентификатор сущности «email», атрибуты сущности:

- password\_hash – пароль в хэшированном представлении;

- email – электронная почта;

- created\_at – дата создания аккаунта;

- updated\_at – дата обновления аккаунта.

2) bot - сущность, характеризующая генерируемого telegram бота с идентификатором «token» и атрибутами:

- name – название бота;

- status – состояние бота, представляет собой элемент перечисления: “Включен”, “Выключен”, “Ошибка запуска”.

- created\_at – дата создания;

- updated\_at – дата обновления.

3) block - сущность, представляющая вопрос регистрации. Атрибуты сущности:

- type – атрибут, определяющий тип элемента регистрации - элемент перечисления из трёх элементов: “Вопрос”, “Сообщение”, “Выбор”;

- next\_state – указатель на следующий элемент регистрации, не пустой в случае непустого списка кнопок у элемента регистрации;

- title – название элемента регистрации;

- text - текст сообщения, выводимого конечному пользователю в Telegram;

- state – порядковый номер блока;

4) option - сущность, представляющая собой кнопку для элемента регистрации, с атрибутами:

- value - значение, которое будет считаться ответом конечного пользователя после нажатия на кнопку;

- next - указатель на элемент регистрации, который следует после нажатия на данную кнопку;

5) answer - сущность, представляющая собой ответ конечного пользователя на вопрос регистрации с единственным атрибутом:

- value - текст ответа на вопрос;

6) participant – сущность, представляющая собой конечного участника мероприятия с идентификатором “user\_id”, представляющий собой уникальный идентификатор, предоставляемый telegram API и атрибутами:

- current\_state – номер вопроса, на который отвечает пользователь в данный момент;­­

7) entry\_points – сущность, представляющая собой точки входа в telegram бот - команды управления от участника мероприятия, например, “/start”. Атрибут сущности:

- key – текст пользовательской команды.

8) mailing - сущность, представляющая собой рассылку внутри telegram бота с атрибутом:

- text - текст рассылки.

Между описанными сущностями были построены связи, согласующиеся с особенностями предметной области, описанных в пункте 1.3.1.

Организатор мероприятия, представляемый сущностью “bot\_owner”, может создавать произвольное количество ботов для регистрации, следовательно, между сущностями “bot\_owner” и “bot” образуется связь “один ко многим”, а минимальные кардинальные числа равны 1 и 0 соответственно.

В свою очередь, каждый бот имеет произвольное количество участников, а участники также могут участвовать в произвольном количестве мероприятий, поэтому между сущностями “bot” и “participant” образуется связь “многие ко многим” с минимальными кардинальными числами 1 и 0 соответственно.

Количество рассылок внутри каждого бота также произвольно, поэтому между сущностями “bots” и “mailings” образуется связь “один ко многим” с минимальными кардинальными числами 1 и 0 соответственно.

Каждого бота однозначно можно представить в виде не пустого набора элементов регистрации, представляемого сущностью “blocks”, поэтому между сущностями “bots” и “blocks” образуется связь “один ко многим” с минимальными кардинальными числами 1 и 1 соответственно.

В каждом боте содержится произвольное количество ответов участников на вопросы регистрации, поэтому между сущностями “bots” и “answers” образуется связь “Один ко многим” с минимальными кардинальными числами 1 и 0 соответственно.

Каждый бот содержит произвольное количество точек входа, поэтому между сущностями “bots” и “entry\_points” образуется связь “один ко многим” c минимальными кардинальными числами 1 и 1 соответственно, так как у бота должна быть хотя бы одна точка входа.

Помимо этого, между сущностями “entry\_points” и “blocks” образуется связь “один к одному” с минимальными кардинальными числами 1 и 1, так как каждая точка входа указывает ровно на один элемент регистрации.

Каждый элемент регистрации может ссылаться на другие элементы регистрации посредством атрибута “default”, поэтому между сущностями “blocks” и “blocks” образуется рекурсивная связь типа “один к одному” с минимальными кардинальными числами 1 и 0, потому что элементы регистрации могут ссылаться друг на друга и посредством сущности “options”.

Каждый блок регистрации типа “выбор” может иметь конечный набор кнопок, который представляется сущностью “option”, поэтому между сущностями “blocks” и “option” образуется связь вида “один ко многим” с минимальными кардинальными числами 1 и 0 соответственно. В свою очередь каждой кнопке соответствует ровно один следующий элемент регистрации, поэтому между сущностями “blocks” и “options” образуется связь “один к одному” с минимальными кардинальными числами 1 и 1.

Каждый ответ должен ссылаться на один из существующих блоков регистрации для дальнейшей идентификации, поэтому между сущностями “blocks” и “answers” образуется связь “один ко многим” c минимальными кардинальными числами 1 и 0 соответственно.

Помимо этого, каждый ответ должен соотноситься с участником мероприятия, который ответил на него, поэтому между сущностями “participants” и “answers” образуется связь “один ко многим” с минимальными кардинальными числами 1 и 0 соответственно.

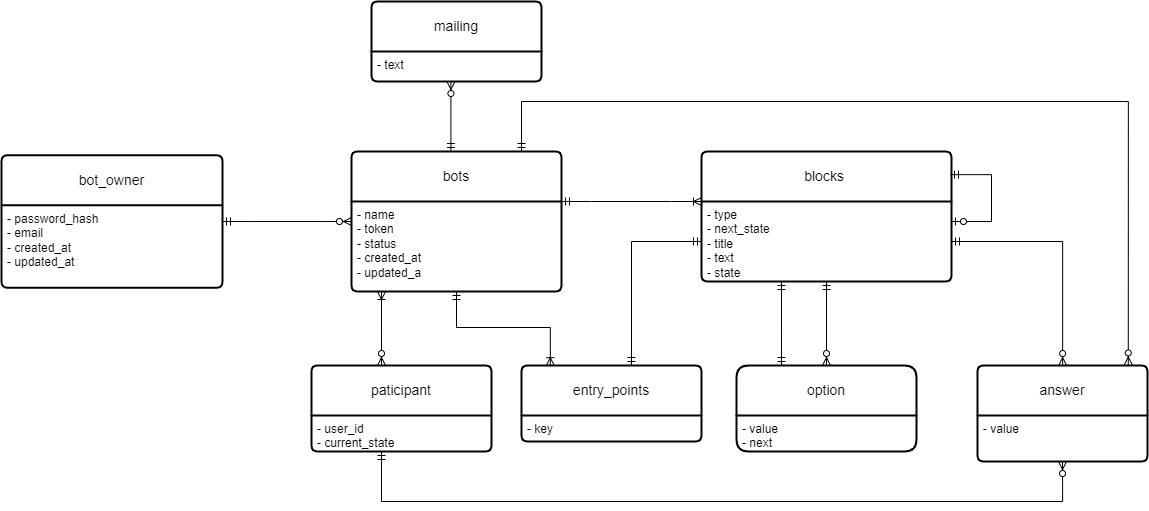


Рисунок 1. Модель «Сущность-связь»

# **2 Проектирование приложения**

## 2.1 Предметная область и требования

В соответствии с правилами преобразования, из созданной ранее модели «сущность-связь», представленной на рис. 1, была получена реляционная модель, представленная на рисунке 2.

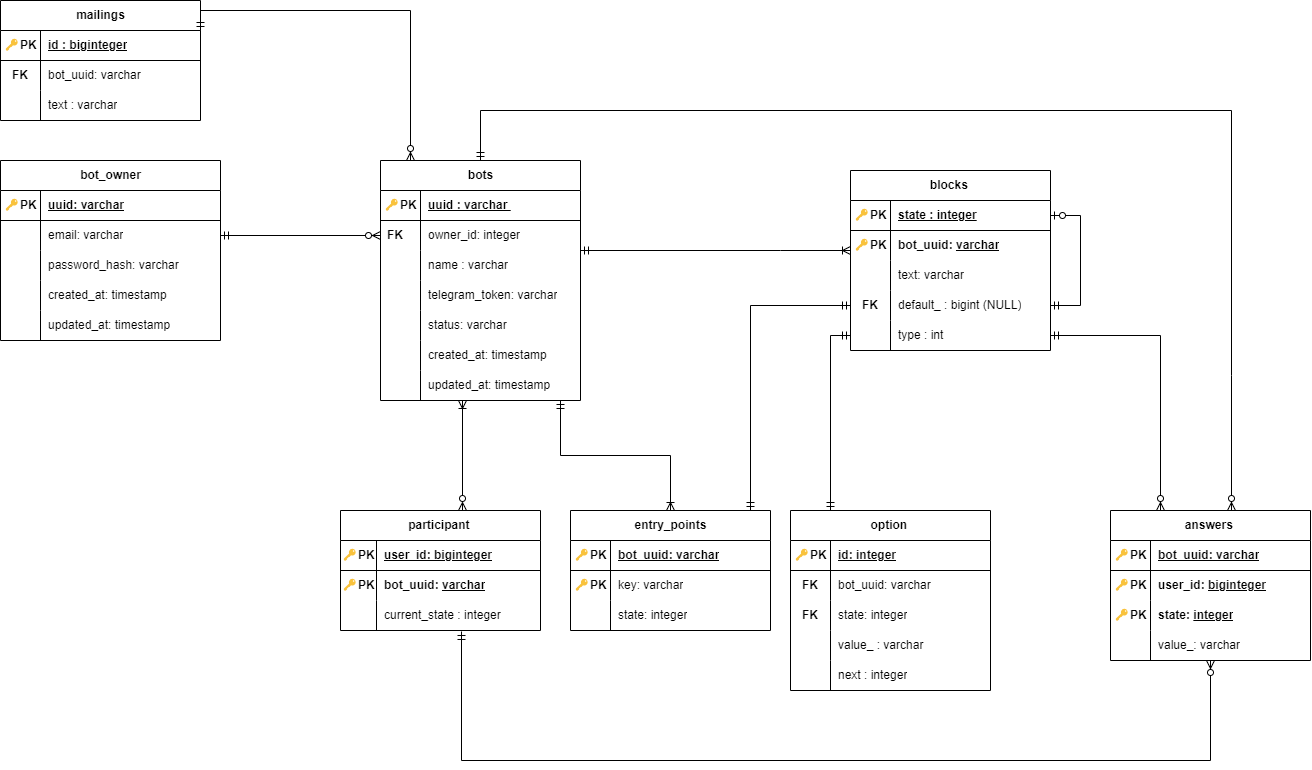


Рисунок 2. Реляционная модель

Были спроектированы таблицы, описывающие отношения реляционной модели:

Таблица 1. Отношение «bot\_owner»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| uuid | Varchar(36) | PK | Not null | Суррогатный строковый ключ |
| email | Varchar(256) | Candidate Key | Not null | Уникальный ключ (AK 1.1) |
| password\_hash | Varchar(256) | No | Not null | Пароль в хэшированном представлении |
| created\_at | TimeStamp | No | Not Null | Время создания аккаунта |
| updated\_at | TimeStamp | No | Not Null | Время обновления аккаунта, по умолчанию равняется времени создания. |

Таблица 2. Отношение «bots»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| uuid | Varchar(36) | PK | Not null | Суррогатный строковый ключ |
| owner\_uuid | Varchar(36) | FK | Not null | Внешний ключ, указывающий на создателя бота. |
| name | Varchar(256) | No | Not null | Название бота |
| token | Varchar(256) | Candidate key | Not Null | Уникальный ключ (AK1.1), связывает бота с telegram API |
| created\_at | TimeStamp | No | Not Null | Время создания аккаунта |
| updated\_at | TimeStamp | No | Not Null | Время обновления аккаунта, по умолчанию равняется времени создания. |

Таблица 3. Отношение «particpants»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| User\_id | BigInteger | PK | Not null | Идентификатор пользователя, предоставляемый telegram. Часть составного первичного ключа |
| bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ, указывающий на бот, с которым работает участник. Часть составного первичного ключа |
| Current\_state | integer | No | Not null | Указатель на текущий элемент регистрации пользователя. |

Таблица 4. Отношение «blocks»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| State | integer | PK | Not null | Порядковый номер блока в боте. Часть составного первичного ключа. |
| Bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ – ссылка на бот. Часть составного первичного ключа. |
| text | Text | No | Not null | Текстовое сообщение элемента регистрации |
| Title | VarChar(256) | No | Not null | Название элемента регистрации |
| Next\_state | integer | FK | Not null | Внешний ключ, образующий рекурсивную связь. Последовательность ссылок блоков друг на друга. По умолчанию равен 0. |
| type | integer | No | Not null | Элемент перечисления: «Вопрос», «Сообщение», «Выбор». |

Таблица 5. Отношение «mailings»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| id | integer | PK | Not null | Суррогатный ключ |
| Bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ, указывает на бота-владельца. |
| Text | Text | No | Not null | Текст рассылки |

Таблица 6. Отношение «entry\_points»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| Bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ, указывающий на бота-владельца. Часть составного первичного ключа. |
| key | Varchar(256) | PK | Not null | Текст команды для входа в бот. Часть составного первичного ключа. |
| state | integer | FK | Not null | Внешний ключ, ссылка на блок, в который ведёт команда входа. |

Таблица 7. Отношение «options»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| Option\_id | integer | PK | Not null | Суррогатный ключ. |
| Bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ, ссылка на бота владельца. |
| state | Integer | FK | Not null | Внешний ключ, ссылка на блок, к которому прикреплена данная кнопка. |
| text | VarChar(128) | No | Not null | Текст, указанный на данной кнопке. |
| next | integer | FK | Not null | Внешний ключ, указывает на блок, к которому необходимо перейти после нажатия на кнопку. |

Таблица 8. Отношение «answers»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column name** | **Type** | **Key** | **Null status** | **Remarks** |
| Bot\_uuid | VarChar(36) | FK | Not null | Внешний ключ, ссылка на бот, который содержит в себе ответ. Часть составного первичного ключа. |
| User\_id | BigInteger | FK | Not null | Внешний ключ, ссылка на участника мероприятия, который отвечает на вопрос. Часть составного первичного ключа. |
| state | Integer | FK | Not null | Внешний ключ, ссылка на элемент регистрации, на который пользователю необходимо дать ответ. Часть составного первичного ключа. |
| text | Text | No | Not null | Текстовое представление ответа пользователя. |

## 2.2 Обеспечение правил минимальной кардинальности.

Таблица 9. Обеспечение минимальной кардинальности связи отношений bot\_owner и bots

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bot\_owner) | Дочерняя таблица (bots) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 10. Обеспечение минимальной кардинальности связи отношений bots и mailings

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bots) | Дочерняя таблица (mailings) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 11. Обеспечение минимальной кардинальности связи отношений bots и entry\_points

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bots) | Дочерняя таблица (entry\_points) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 12. Обеспечение минимальной кардинальности связи отношений bots и participants

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bots) | Дочерняя таблица (participants) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 13. Обеспечение минимальной кардинальности связи отношений bots и blocks

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bots) | Дочерняя таблица (blocks) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 14. Обеспечение минимальной кардинальности связи отношений bots и answers

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (bots) | Дочерняя таблица (answers) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 15. Обеспечение минимальной кардинальности связи отношений participants и answers

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (participants) | Дочерняя таблица (answers) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 16. Обеспечение минимальной кардинальности связи отношений blocks и entry\_points

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (blocks) | Дочерняя таблица (entry\_points) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 17. Обеспечение минимальной кардинальности связи отношений blocks и options

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (blocks) | Дочерняя таблица (options) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Каскадное удаление | Без ограничения |

Таблица 18. Обеспечение минимальной кардинальности связи отношений blocks и blocks

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (blocks) | Дочерняя таблица (blocks) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Разрешено, если новое значение соответствует какому-либо другому блоку в рамках одного бота. |
| Удаление | Запрещено | Запрещено |

Таблица 19. Обеспечение минимальной кардинальности связи отношений blocks и answers

|  |  |  |
| --- | --- | --- |
| Операция | Родительская таблица (blocks) | Дочерняя таблица (answers) |
| Вставка | Без ограничения | Подбор родительской записи |
| Изменение первичного или внешнего ключа | Запрещено | Запрещено |
| Удаление | Без ограничения | Без ограничения |

# **3 Разработка приложения**

## 3.1 Разработка архитектуры приложения

В разрабатываемом сервисе можно выделить два основные блока, представленных на рис. 3:

1. Сервис авторизации;
2. Сервис работы с ботами.

Сервис авторизации имеет свою базу данных с единственной таблицей “bot\_owner”, описанной в разделе 1.3.2, и производит поиск по существующим организаторам или осуществляет регистрацию нового организатора мероприятия. После успешной авторизации организатору предоставляется JWT (JSON web token) токен, который предоставляет доступ к взаимодействию с сервисом работы с ботами.

Сервис работы с ботами представляет собой более сложную структуру.

С одной стороны, он принимает HTTP запросы от авторизированных по JWT токену организаторов, с другой самостоятельно отправляет HTTP запросы к открытому API, предоставляемому Telegram. Помимо этого, сервис работы с ботами осуществляет функционал «сервера» для запущенных telegram ботов

Для оптимизации и ускорения взаимодействия между пользовательскими запросами и запросами отправляемыми к Telegram, используется брокер сообщений «RabbitMQ». Брокер сообщений необходим для того, чтобы значительно ускорить отправку сообщений telegram конечным участника мероприятия, поскольку с его помощью клиенту не приходится ждать отправку сообщения в telegram, для перехода к следующему элементу регистрации, необходимо только дождать добавления в очередь, предоставляемую брокером сообщений. Такой подход также предоставляет асинхронность обработки запросов в случае одновременной работы с несколькими ботами, запущенными на сервисе.

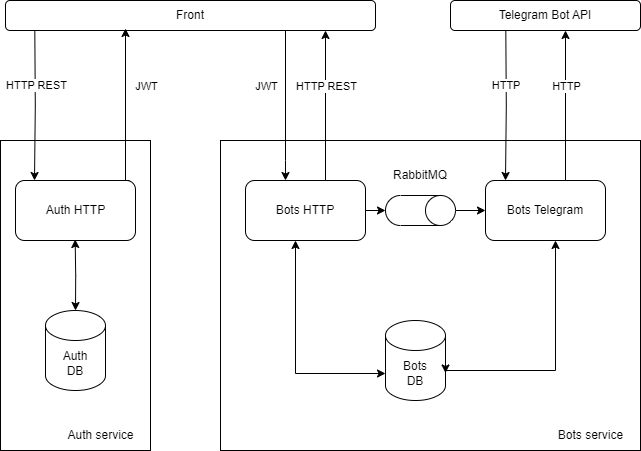


Рисунок 3 - Архитектура приложения.

## 3.2 Выбор технологических средств

Для разработки технического решения был выбран язык программирования «Go» версии 1.22 . Данный язык представляет собой распространённый инструмент для разработки высоконагруженных серверных приложений, обеспечивая высокий уровень производительности и предоставляя разработчикам широкий ряд возможностей и сторонних библиотек.

Для разработки хранилища данных использовалась СУБД с открытым исходным кодом PostgreSQL. Данная СУБД отличается высокой производительностью и гибкостью, что делает ее привлекательной для серверных приложений с высокой нагрузкой и требующими надежного управления данными.

## 3.3 Ключевые типы данных и функции

### 3.3.1 Сервис авторизации

Точкой входа в приложение является файл “http.go”, находящийся в директории “cmd/http” и запускающий сервер HTTP по указанному в переменных среды порту. Сервер представляет собой реализацию паттерна REST (Representational State Transfer) API, реализующее три HTTP метода:

* **Register User** – POST метод, осуществляющий регистрацию нового организатора. Входными данными метода является JSON-объект, представляющий организатора. В случае успешной регистрации возвращает код ответа 201 «Created», в случае ошибки возвращает код ответа 400 «Bad Request».
* **Login user** – POST метод, осуществляющий авторизацию организатора в сервисе. Входными данными является JSON-объект, содержащий аутентификационные данные пользователя: «email» и «password». В случае успешной авторизации, возвращает код ответа 200 «OK» и JSON-объект, содержащий JWT токен авторизации. В случае ошибки во входных данных, возвращает код ответа 401 «Unauthorized».
* **Get user** – GET метод для получения информации пользователя, метод принимает на вход один параметр заголовка – uuid организатора, В случае успешного поиска организатора, возвращает код ответа 200 «OK» и JSON-объект, содержащий не конфиденциальную информацию пользователя. В случае ошибки поиска, возвращает код ответа 400 «Not Found».

В рамках реализации сервиса авторизации были выделены следующие ключевые функции:

* **NewAccessToken –** создаёт новый JWT токен для авторизации пользователя, принимает на вход uuidпользователя для последующей авторизации и время истечения токена, после окончания которого пользователю необходимо будет снова авторизироваться.
* **ParseAccessToken –** функция проверяющая валидность введённого токена, на вход принимает JWT токен, в случае успешной авторизации возвращает декодированную информацию о пользовательском uuid.

### 3.3.2 Сервис работы с ботами

Сервис работы с ботами представляет собой одновременно и HTTP сервер для получения запросов от организаторов и клиент, работающий с Telegram Bot API по протоколу HTTP.

HTTP сервер реализует паттерн REST API, для доступа к которому в заголовке отправляемого запроса с тегом «Authorization» необходимо указать, полученный в сервисе авторизации JWT токен, в случае не указанного токена или некорректного токена, будет получен код ответа 401 «Unauthorized». Реализует шесть HTTP методов:

* **Create Bot –** POST метод, отправляющий запрос на генерацию нового бота. На вход принимает JSON-объект, содержащий полную, заранее описанную схему бота. В случае успешного создания бота, будет получен код ответа 201 «Created». В случае ошибки при описании схемы, будет получен код возврата 400 «Bad request».
* **Start Bot –** POST метод, запускающий ранее созданного бота, на вход принимает один параметр заголовка – uuid бота. В случае существования бота, по указанному uuid будет получен код ответа 200 «ОК», в противном случае, будет получен код ответа 404 «Not Found».
* **Stop Bot -** POST метод, выключающий ранее созданного бота, на вход принимает один параметр заголовка – uuid бота. В случае существования бота, по указанному uuid будет получен код ответа 200 «ОК», в противном случае, будет получен код ответа 404 «Not Found».
* **Get Bots –** GET метод, получающий список ботов конкретного организатора. Не принимает входных параметров, возвращает список из полных схем, всех созданных организатором ботов с кодом ответа 200 «ОК».
* **Get Bot –** GET метод, возвращающий одного бота, на вход принимает один параметр заголовка запроса – uuid бота. В случае успешного поиска бота, возвращает код ответа 200 «ОК» и JSON-объект, содержащий полную схему бота, в противном случае, возвращает код ответа 404 «Not Found».
* **Get Answers –** GET метод, возвращающий список ответов пользователей. На вход принимает единственный параметр заголовка запроса - uuid бота. В случае успешного поиска бота, возвращает код ответа 200 «OK» и ответы пользователей в формате «.csv», в противном случае, возвращает код ответа 404 «Not Found».

На рисунке 4 представлена UML (Unified Modeling Language) диаграмма сервиса работы с ботами. На ней представлены основные типы данных и их связи между собой.

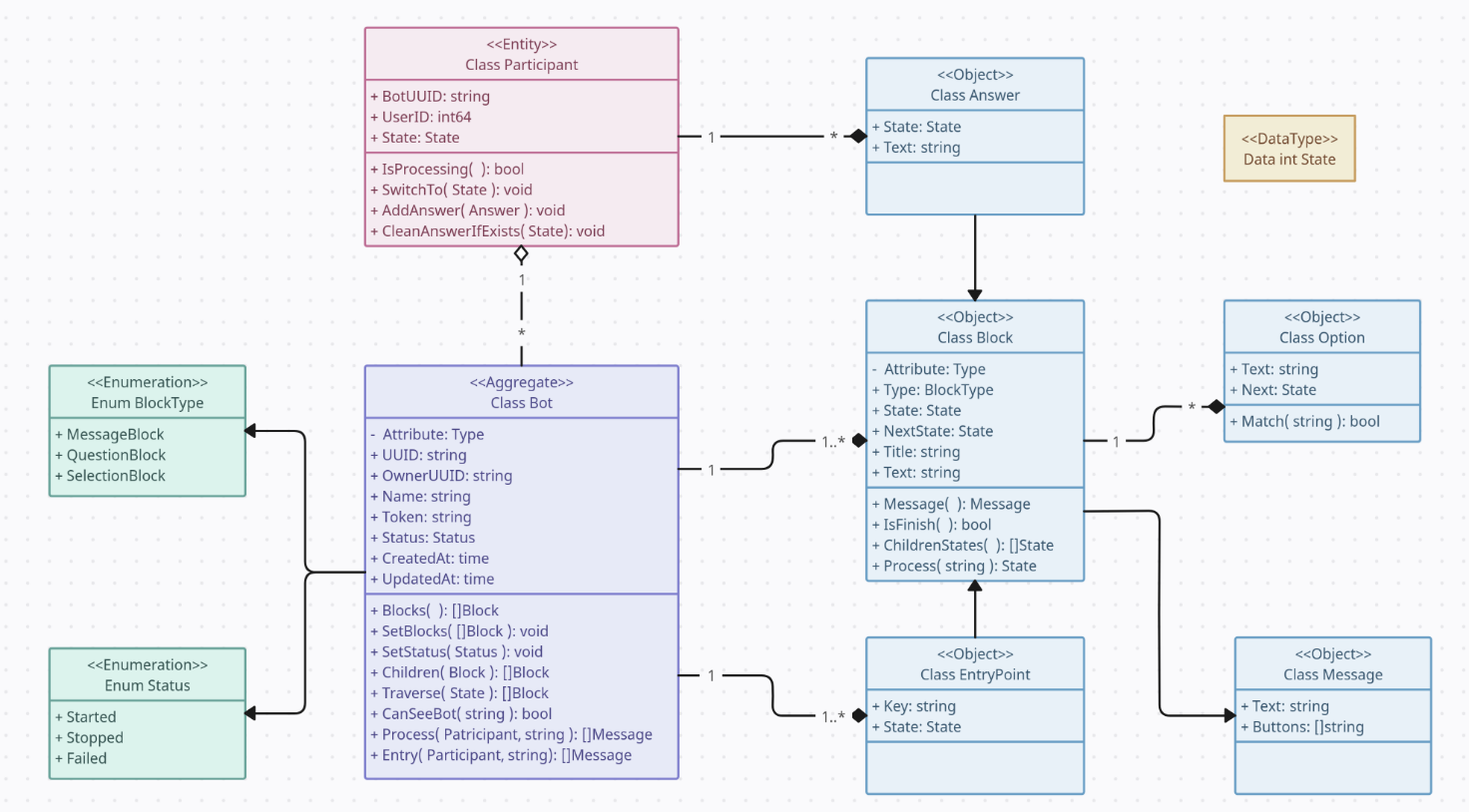


Рисунок 4. UML диаграмма сервиса работы с ботами

В рамках реализации сервиса авторизации были выделены следующие ключевые функции:

* **RunTelegramPort –** функция, отвечающая за обработку сообщений от конечных участников мероприятия, она работает с двумя потоками данных: «msgCh» и «runCh», отвечающими за получение и отправку сообщений соответственно.
* **newTelegramBot –** функция, отвечающая за соединение описанной ранее схемы бота и TelegramAPI, путём предоставления нового экземпляра клиента API.
* **Start –** функция, отвечающая непосредственно за «запуск» бота по предоставленному ранее клиенту API, запускает процесс получения обновлений с сервера Telegram.

# **4 Тестирование**

Целью проведения тестирования является проверка работоспособности сервиса автоматизации регистрации участников на мероприятия.

## 4.1 Разработка тестов

Для комплексного тестирования необходимо разработать ряд наборов данных, которые проверят работоспособность и выполнение заявленного функционала конечных точек API.

### 4.1.1 Генерация ботов

Таблица 20. Данные для генерации бота

|  |
| --- |
| {    "botUUID": "test-bot",    "name": "Example",    "token": "6277601007:AAECQkdMAKkIrkoDxXhG4Bg2CJymJddeqw8",    "entries": [      {        "key": "start",        "state": 1      }    ],    "blocks": [      {        "type": "message",        "state": 1,        "nextState": 2,        "title": "Greeting",        "text": "Hello, user!"      },      {        "type": "question",        "state": 2,        "nextState": 3,        "title": "Username",        "text": "What's your name?"      },      {        "type": "selection",        "state": 3,        "nextState": 4,        "title": "Pill color",        "text": "Choose a pill: red or blue",        "options": [          { "text": "red", "next": 5 },          { "text": "blue", "next": 6 }        ]      },      {        "type": "message",        "state": 4,        "nextState": 3,        "title": "Error",        "text": "Oops, choose red or blue pill!"      },      {        "type": "message",        "state": 5,        "nextState": 0,        "title": "Red",        "text": "Okay, you choose red pill"      },      {        "type": "message",        "state": 6,        "nextState": 0,        "title": "Blue",        "text": "Okay, you choose blue pill"      }    ]  } |

### 4.1.2 Регистрация пользователя

Таблица 21. Тестовые данные для регистрации нового пользователя

|  |
| --- |
| {    "uuid": "alex",    "email": "alexey@gmail.com",    "password": "1234"  } |

## 4.2 Тестирование

Тестирование проводилось в среде для тестирования API «Postman».

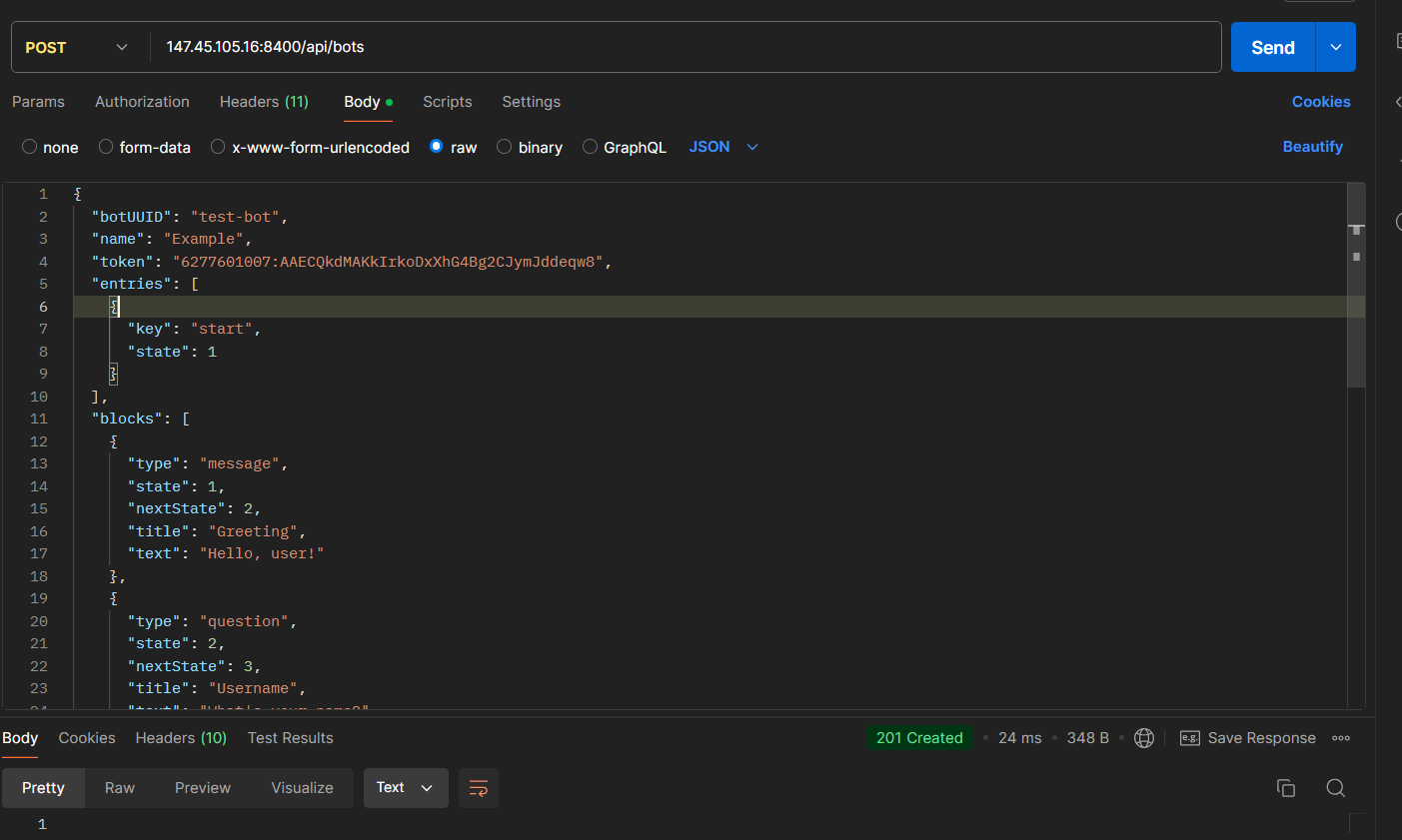
На рисунке 5 и 6 представлены результаты тестирования генерации telegram бота 

Рисунок 5. Отправка запроса на генерацию бота

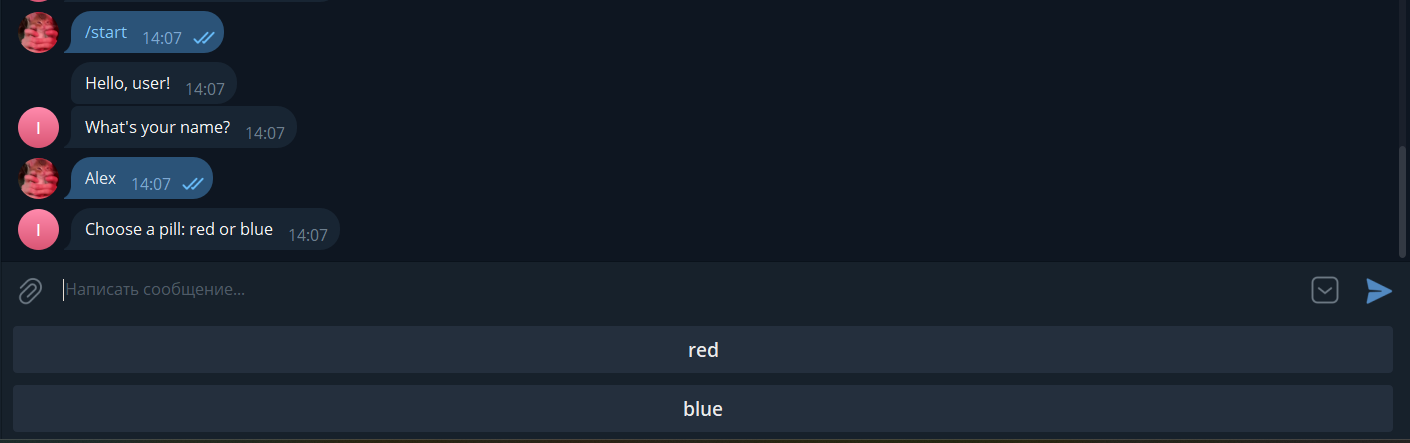


Рисунок 6. Созданный по схеме telegram бот

На рисунке 7 представлена таблица с ответами пользователей.

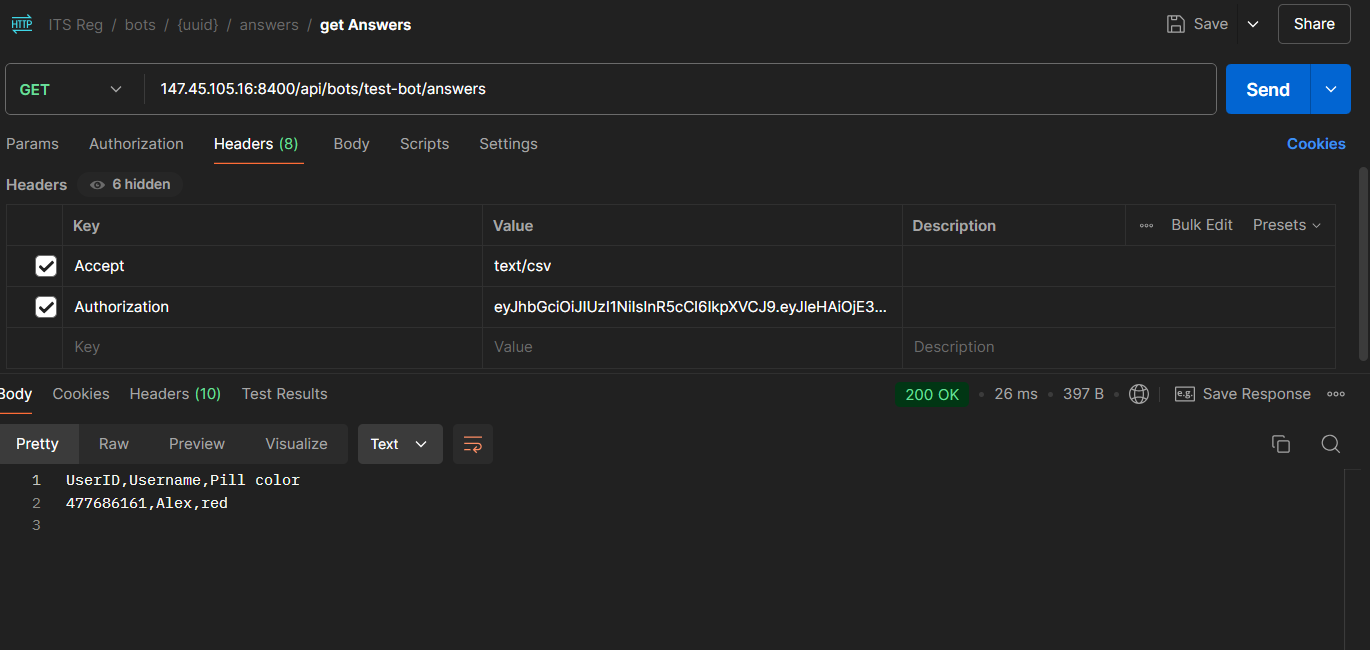


Рисунок 7. Таблица с ответами пользователей

На рисунке 8 представлены результаты тестирования авторизации организатора.

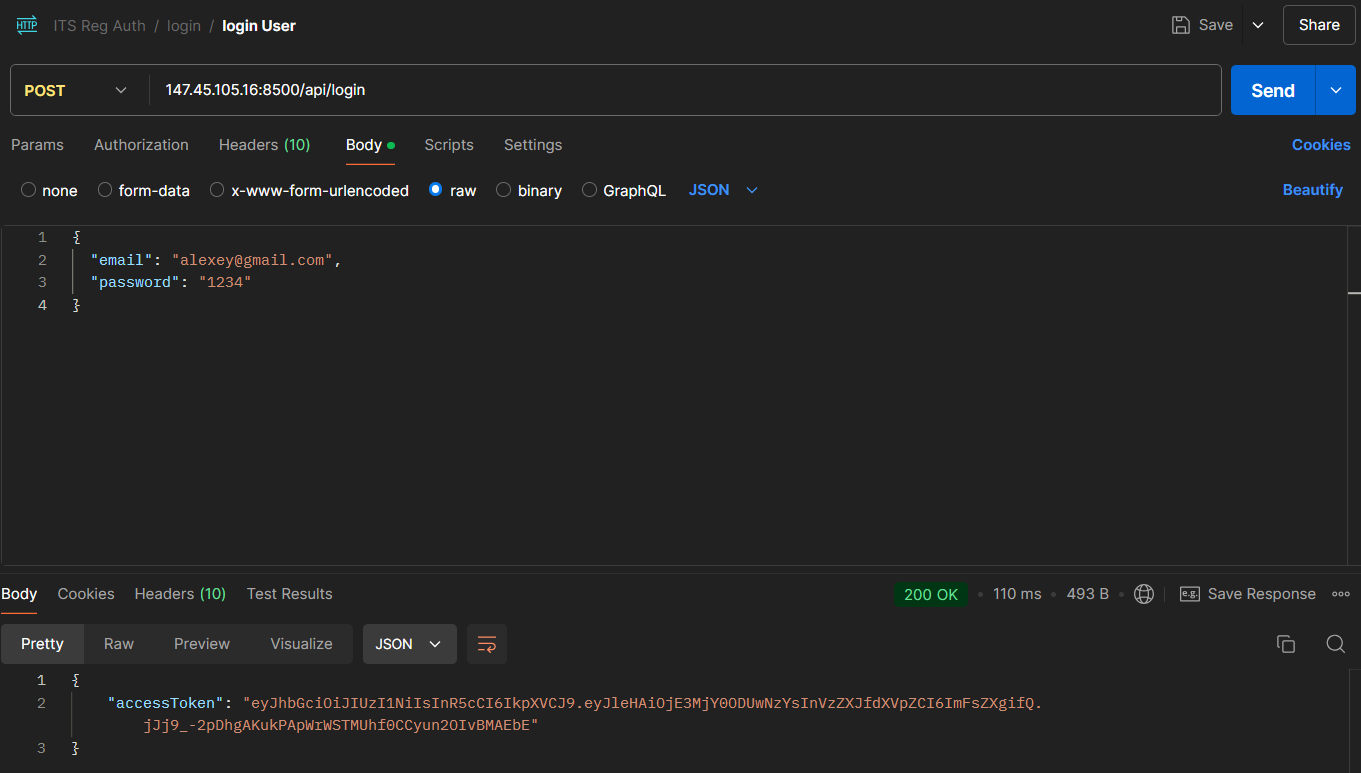


Рисунок 8. Авторизация пользователя

# **5 Заключение**

# **6 Список использованных источников**