МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ “ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

Разработка Телеграм-бота «Календарь»

Курсовая работа

09.03.04 *Программная инженерия*

Зав. кафедрой _____ *С.Д.Махортов, д.ф.-м.н., доцент __. __.20__*

Обучающийся _____ *О. А. Лукашев, 3 курс, д/о*

Руководитель _____ *П.С.Лысачев, ст. преподаватель*

Воронеж 2023

Содержание

Введение

В современном быстротечном мире, где каждая минута имеет особое значение и конкуренция становится все более ожесточенной, все больше людей стремятся найти способы организовать свою жизнь наиболее эффективно и продуктивно. И нет сомнения в том, что одним из ключевых аспектов этого сложного процесса является управление временем. Безусловно, мы все хотим научиться умело распреде лять свои дела и задачи, чтобы иметь возможность достигать своих высоких амбициозных целей и исполнять обязанности вовремя.

Для эффективного управления своим временем люди все чаще обращаются к различным инструментам и приложениям, которые предлагают современные технологии в области планирования и управления своим расписанием. В особенности, календари стали популярным выбором в этом многогранном процессе. Они предоставляют нам возможность вести учет и организацию наших дел и событий, а также позволяют нам управлять нашим временем более осознанно и систематично.

Актуальность данной темы неоспорима, поскольку в современном обществе все больше людей сталкиваются с проблемой неэффективного управления своим временем. Все испытывают давление современного образа жизни, и недостаточная организованность и планирование своих дел и задач могут привести к непредвиденным ситуациям и неудачам, которые, в свою очередь, могут оказать негативное влияние на все сферы нашей жизни, начиная с личной сферы и заканчивая профессиональной карьерой.

Проблема неэффективного управления временем имеет множество причин и проявлений. Одной из наиболее распространенных причин является ограниченное количество времени, которое мы можем выделить на планирование и организацию своих дел и задач. В современном быстром мире кажется, что время летит незаметно, и мы все более сжаты во временных

рамках наших ежедневных обязанностей. Кроме того, многие люди сталкиваются с проблемой отсутствия определенной системы управления своим временем, могут тратить бесценные минуты на бесполезные задачи, не имея ясного плана и структуры для достижения целей.

В данной курсовой работе описан процесс разработки и создания Телеграм-бота Календарь, который предоставляет пользователям не только средства эффективного управления своим временем, но и помогает им легко и удобно управлять своими событиями и обязанностями. Телеграм-бот Календарь является программой, разработанной на основе популярного мессенджера Телеграм, и предлагает уникальный функционал для создания событий и заметок на определенную дату. Кроме того, пользователи могут группировать свои события в удобные папки, что облегчает хранение и поиск необходимой информации.

Таким образом, разработка и создание Телеграм-бота Календарь представляют собой актуальную тему и имеют значительный потенциал для решения проблемы неэффективного управления временем. Этот бот предоставляет пользователям удобный и эффективный инструмент для планирования и организации своих дел и задач, что способствует повышению продуктивности и достижению желаемых результатов. Одновременно он способствует более осознанному использованию времени, что позволяет нам насладиться более сбалансированной жизнью и успешно совмещать личные и профессиональные аспекты нашей деятельности.

1 Постановка задачи

1.1 Постановка задачи

Целью данной курсовой работы является разработка и создание высокофункционального Телеграм-бота под названием "Календарь", который предназначен для эффективного управления временем, организации задач и событий, а также для повышения продуктивности пользователей. Основная цель заключается в создании инструмента, который позволит пользователям более эффективно планировать, структурировать и контролировать свои дела и задачи, что приведет к более эффективному использованию времени и достижению поставленных целей.

Для успешной реализации данной цели был поставлен ряд задач. Предоставить пользователю:

— Возможность делиться с другими пользователями событиями;

— Возможность просматривать доступные события;

— Возможность добавлять события или папки для событий;

— Неперегруженный и понятный интерфейс.

2 Анализ предметной области

2.1 Терминология предметной области

Для лучшего понимания и взаимодействия с предметной областью разработки Телеграм-бота "Календарь", необходимо ознакомиться с основными терминами, используемыми в данном контексте. Ниже приведены определения некоторых сложных слов и названий, которые будут использоваться в дальнейшем описании:

Телеграм-бот – программа, разработанная на основе мессенджера Телеграм, которая автоматизирует определенные задачи и предоставляет пользователю возможность взаимодействовать с ботом через интерфейс мессенджера.

Сервер – это компьютер или устройство, которое обеспечивает доступ и предоставляет ресурсы (например, данные, файлы, приложения) другим устройствам, называемым клиентами, в рамках сети. Он отвечает на запросы клиентов, обрабатывает их и передает необходимую информацию обратно. Сервер выполняет роль центра управления и обеспечивает взаимодействие между клиентами, обеспечивая надежность, безопасность и доступность ресурсов.

Бизнес-логика – определяет основные процессы и функциональность бота, которые непосредственно связаны с его предназначением и целями бизнеса. Это включает в себя обработку запросов пользователей, взаимодействие с базой данных, выполнение операций, обеспечение безопасности данных и принятие решений на основе заданных правил и алгоритмов.

API (Application Programming Interface) – это набор определенных правил и протоколов, которые позволяют программам взаимодействовать друг с другом.

Аватар – это изображение, обычно в виде фотографии или иконки, которое представляет пользователя или персонализирует его профиль в онлайн-сервисах, социальных сетях или мессенджерах.

Токен – это уникальная строка символов или код, которая используется для авторизации и идентификации пользователя или приложения в системе. В контексте разработки Телеграм-бота, токен является ключевым элементом, который позволяет боту взаимодействовать с Telegram API и получать доступ к необходимым функциям и возможностям платформы. Токен предоставляется разработчику бота при регистрации приложения и обеспечивает безопасное взаимодействие между ботом и платформой Telegram.

Юзернейм, также известный как пользовательское имя или никнейм, представляет собой уникальное идентификатор, который используется пользователем для идентификации и отображения своей личности в определенной системе или платформе.

Телеграм – это мессенджер и платформа обмена сообщениями, которая позволяет пользователям общаться с другими людьми посредством текстовых, голосовых и видеосообщений. Он предлагает шифрование данных для обеспечения безопасности и конфиденциальности, а также поддерживает функциональность ботов, которые могут автоматизировать определенные задачи и предоставлять различные сервисы через интерфейс мессенджера. Телеграм является популярным средством коммуникации, обмена информацией и взаимодействия в различных сферах жизни и бизнеса.

База данных – это организованная коллекция данных, которая структурирована, хранится и управляется с помощью компьютерной системы. Она представляет собой структурированное хранилище информации, доступ к которой осуществляется по определенным правилам и методам. База данных позволяет эффективно хранить, обрабатывать и извлекать данные, обеспечивая целостность и безопасность информации, а также обеспечивая

возможность выполнения различных операций, таких как добавление, обновление и удаление данных.

Интерфейс в контексте разработки Телеграм-бота "Календарь" представляет собой средство взаимодействия пользователя с ботом. Это пользовательский интерфейс, который позволяет пользователю вводить команды, просматривать информацию и получать ответы от бота.

2.2 Обзор аналогов

2.2.1 DP_TodoList (@DP_TodoList_bot)

DP_TodoList — это Телеграм-бот, специализирующийся на создании напоминаний и управлении событиями. Пользователи могут добавлять события, устанавливать даты и время, а также настраивать повторяющиеся напоминания. Бот предоставляет гибкую систему настройки напоминаний, позволяя пользователям быть в курсе своего расписания.

Преимущества:

— Простой и легкий в использовании;

— Позволяет устанавливать напоминания и таймеры;

— Не требует регистрации и авторизации.

Недостатки:

— Не позволяет создавать папки и делиться напоминаниями с другими пользователями;

— Ограниченный функционал.

2.2.2 Google Календарь(@GoogleCalendarBot)

Google Календарь также предоставляет Телеграм-бота, который позволяет пользователям получать уведомления о предстоящих событиях, добавлять и редактировать события, а также выполнять другие операции с календарем. Бот может синхронизироваться с вашим Google-аккаунтом, обеспечивая доступ к вашим событиям и задачам на любом устройстве.

Преимущества:

— Широкая интеграция с другими Google-сервисами, такими как Gmail и Google Документы;

— Гибкие настройки уведомлений и повторений событий.

Недостатки:

— Интерфейс может быть перегружен функциями для некоторых пользователей.

— Ограниченные возможности сотрудничества и совместной работы с другими пользователями.

2.2.3 MyCloudStore(@tenvsten_bot)

MyCloudStore — это Телеграм-бот с функциональностью календаря, который помогает пользователям управлять своим расписанием и задачами. Бот позволяет создавать события, задавать напоминания, добавлять задачи и делиться списками дел с другими пользователями.

Преимущества:

— Возможность делиться списками дел с другими пользователями, что упрощает совместную работу.

Недостатки:

— Некоторые расширенные функции могут быть доступны только в платной версии;

— Интерфейс может показаться сложным для некоторых пользователей из-за множества функций.

2.2.4 Вывод по обзору аналогов

Выделив общее, можно сделать вывод, что пользователю необходимо предоставить простой и легкий в использовании интерфейс, поддержку создания напоминаний и управления событиями, возможность делиться событиями с другими пользователями. Это позволит ему эффективно управлять своим расписанием и задачами с минимальными препятствиями и максимальной гибкостью.

2.3 Средства реализации

Для успешной реализации разработки Телеграм-бота был задействован целый ряд инструментов и ресурсов. В процессе создания данного бота использовались следующие технологии, программные средства и библиотеки, которые играли важную роль в достижении целей:

— Java;

— PostgreSQL;

— Telegrambots;

— Telegram API;

— Spring framework.

2.4 Обзор средств

Выбор данных средств и технологий для реализации разработки Телеграм-бота был обоснован рядом факторов и соображений. Вот некоторые из них:

— Java – является одним из самых популярных языков программирования, который обладает широким функциональным набором, поддержкой объектно-ориентированного программирования и развитой экосистемой инструментов и библиотек. Это делает его привлекательным выбором для разработки Телеграм-бота, обеспечивая надежность, масштабируемость и производительность.

— PostgreSQL – является мощной и открытой системой управления базами данных (СУБД), которая обладает расширенными

возможностями, надежностью и хорошей производительностью. Использование PostgreSQL позволяет эффективно хранить и управлять данными Телеграм-бота, обеспечивая консистентность и безопасность.

— Telegrambots и Telegram API: Telegrambots является библиотекой, которая предоставляет удобные инструменты для разработки Телеграм-ботов на языке Java. Она упрощает взаимодействие с Telegram API, предоставляя высокоуровневые абстракции и функциональность для работы с ботом. Использование Telegram API позволяет создавать и управлять ботом, обрабатывать входящие сообщения и выполнять другие операции через Telegram-платформу.

— Spring framework – является одним из ведущих фреймворков для разработки приложений на языке Java. Он обеспечивает модульность, инверсию управления, аспектно-ориентированное программирование и другие функциональности, которые делают процесс разработки более эффективным. Использование Spring framework позволяет создавать гибкие, масштабируемые и легко поддерживаемые Телеграм-боты.

Комбинация этих технологий и инструментов обеспечивает мощную основу для разработки Телеграм-бота, позволяя достичь поставленных целей в области функциональности, производительности и масштабируемости. Каждое из выбранных средств играет важную роль в реализации и успешной эксплуатации бота, обеспечивая необходимые функции, работу с данными и взаимодействие с Telegram-платформой.

2.5 Обзор альтернативных средств реализации

Помимо использованных технологий и программных средств, существуют альтернативные варианты для создания Телеграм-бота и управления базой данных. Рассмотрим некоторые из них:

— Язык программирования: вместо Java можно выбрать другой язык, такой как Python, Node.js, Ruby или C#. Каждый из них имеет свои преимущества и недостатки в зависимости от требований проекта и предпочтений разработчика. Java был выбран в данном случае из-за его широкой популярности, кросс-платформенности и богатой экосистемы инструментов.

— База данных: PostgreSQL используется в данном проекте, однако существуют и другие альтернативы, такие как MySQL, MongoDB, SQLite и другие. Выбор базы данных зависит от многих факторов, включая требования к масштабируемости, производительности, надежности и доступности данных. PostgreSQL, как реляционная база данных, обладает мощными функциями и широкой поддержкой сообщества, что может быть причиной такого выбора.

— Telegram API и Telegrambots: существуют различные библиотеки и фреймворки для работы с Telegram API на разных языках программирования. Вместо Telegrambots можно выбрать, например, python-telegram-bot, Telegraf, Telebot и другие аналогичные библиотеки. Выбор библиотеки зависит от удобства использования, документации и функциональности, которую она предоставляет.

— Spring framework: вместо Spring framework можно использовать другие фреймворки, такие как Django, Flask, Express.js или ASP.NET. Каждый фреймворк имеет свои особенности и

предназначен для разных типов проектов. Spring framework был выбран из-за своей популярности, расширяемости и поддержки в разработке Java-приложений.

Выбор конкретных инструментов и технологий зависит от требований проекта, опыта разработчиков, доступных ресурсов и других факторов. В данном случае были выбраны Java, PostgreSQL, Telegrambots, Telegram API и Spring framework на основании их соответствия требованиям проекта, удобства использования и поддержки со стороны сообщества разработчиков.

3 Реализация

3.1 Диаграмма последовательностей

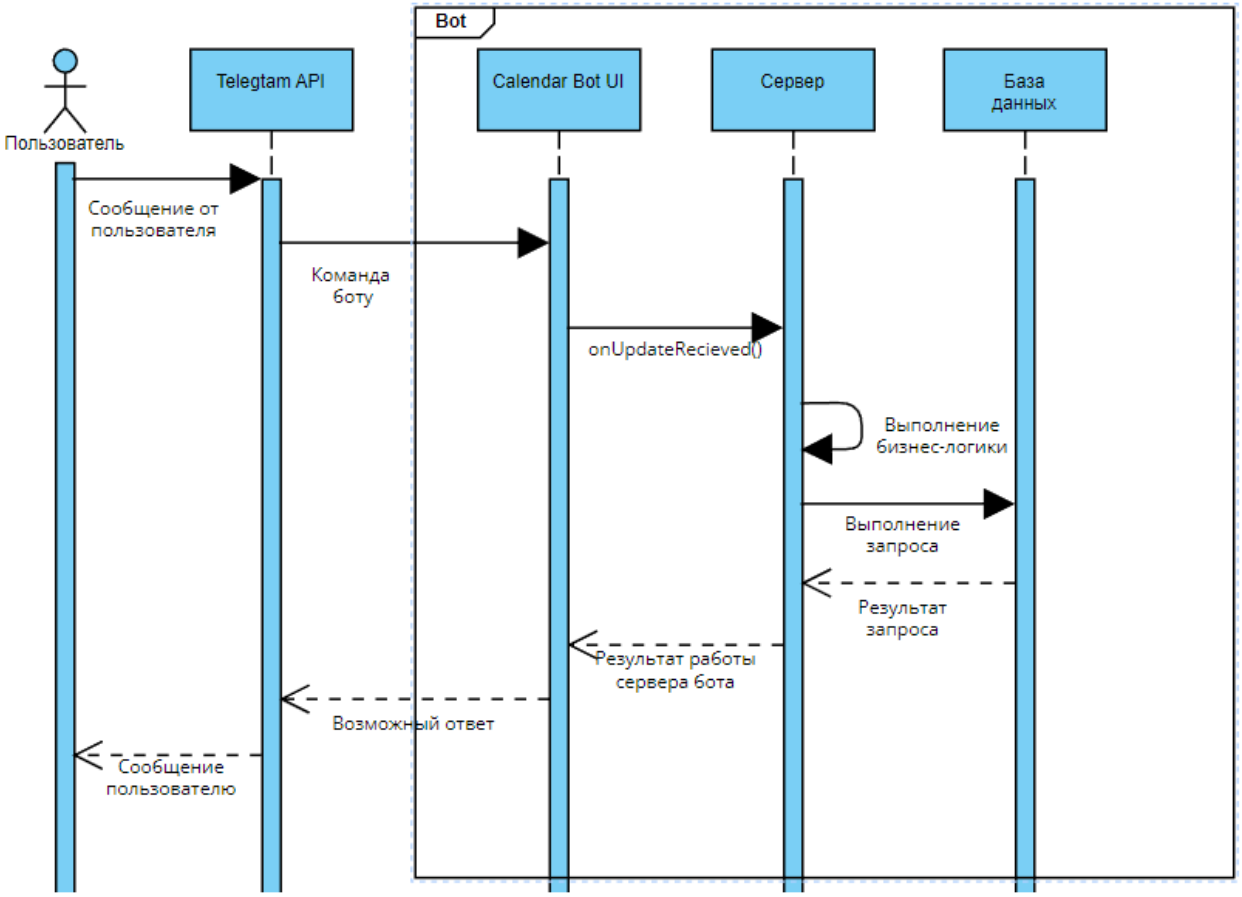

Рисунок 1 - Диаграмма последовательностей

Взаимодействие компонентов происходит следующим образом:

— Пользователь начинает взаимодействие, отправляя сообщение боту в Телеграм;

— Отправленное пользователем сообщение поступает в компонент Telegram API и перенаправляется боту;

— Бот получает обновление и вызывает метод onUpdateReceived() на сервере приложения;

— Сервер приложения принимает запрос и начинает выполнение бизнес-логики;

— В рамках бизнес-логики может возникнуть необходимость взаимодействия с компонентом базы данных (БД);

— Если требуется доступ к БД, сервер приложения отправляет запрос на получение или сохранение данных;

— Компонент БД обрабатывает запрос и выполняет соответствующие операции чтения или записи данных;

— Результат операции БД возвращается обратно на сервер приложения;

— Сервер приложения продолжает выполнение бизнес-логики, используя полученные данные из БД;

— После завершения бизнес-логики, сервер приложения формирует ответное сообщение;

— Ответное сообщение передается обратно в компонент Telegram API;

— Telegram API отправляет ответ пользователю;

— Пользователь получает ответ от бота и взаимодействие завершается.

Таким образом, весь процесс взаимодействия начинается с пользователя, проходит через Telegram API, передается на сервер приложения для выполнения бизнес-логики, может включать в себя доступ к компоненту базы данных, и возвращается обратно в Telegram API для отправки ответа пользователю. Этот поток позволяет эффективно обрабатывать запросы и обеспечивать взаимодействие с телеграм-ботом.

3.2 Диаграмма прецедентов

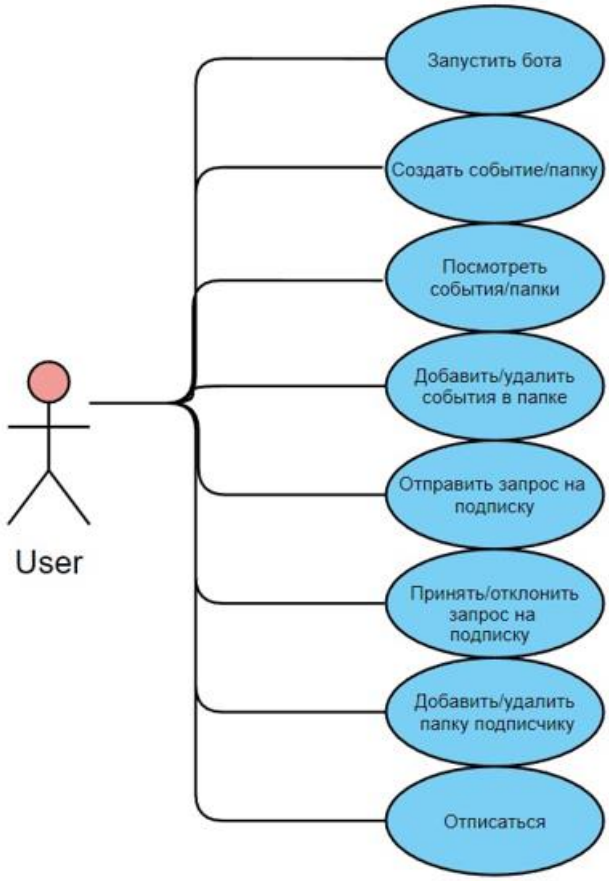

Рисунок 2 - Диаграмма прецедентов

На рисунке 2 продемонстрирован функционал, доступный пользователю.

Функционал роли Пользователь:

— Запустить бота;

— Создать событие;

— Создать папку;

— Добавить событие в папку;

— Посмотреть события в папке;

— Удалить событие из папки;

— Отправить запрос другому авторизованному пользователю на отслеживание событий в папках, доступных ему.

— Принять или отклонить запрос;

— Добавить папку подписчику;

— Удалить папку, доступную подписчику;

— Отписаться от другого пользователя;

— Отписать пользователя.

Так же создатель события и все, кто имеет доступ к нему, получают уведомление в виде сообщения от бота при наступлении даты, указанной при создании события.

3.3 Проектирование базы данных

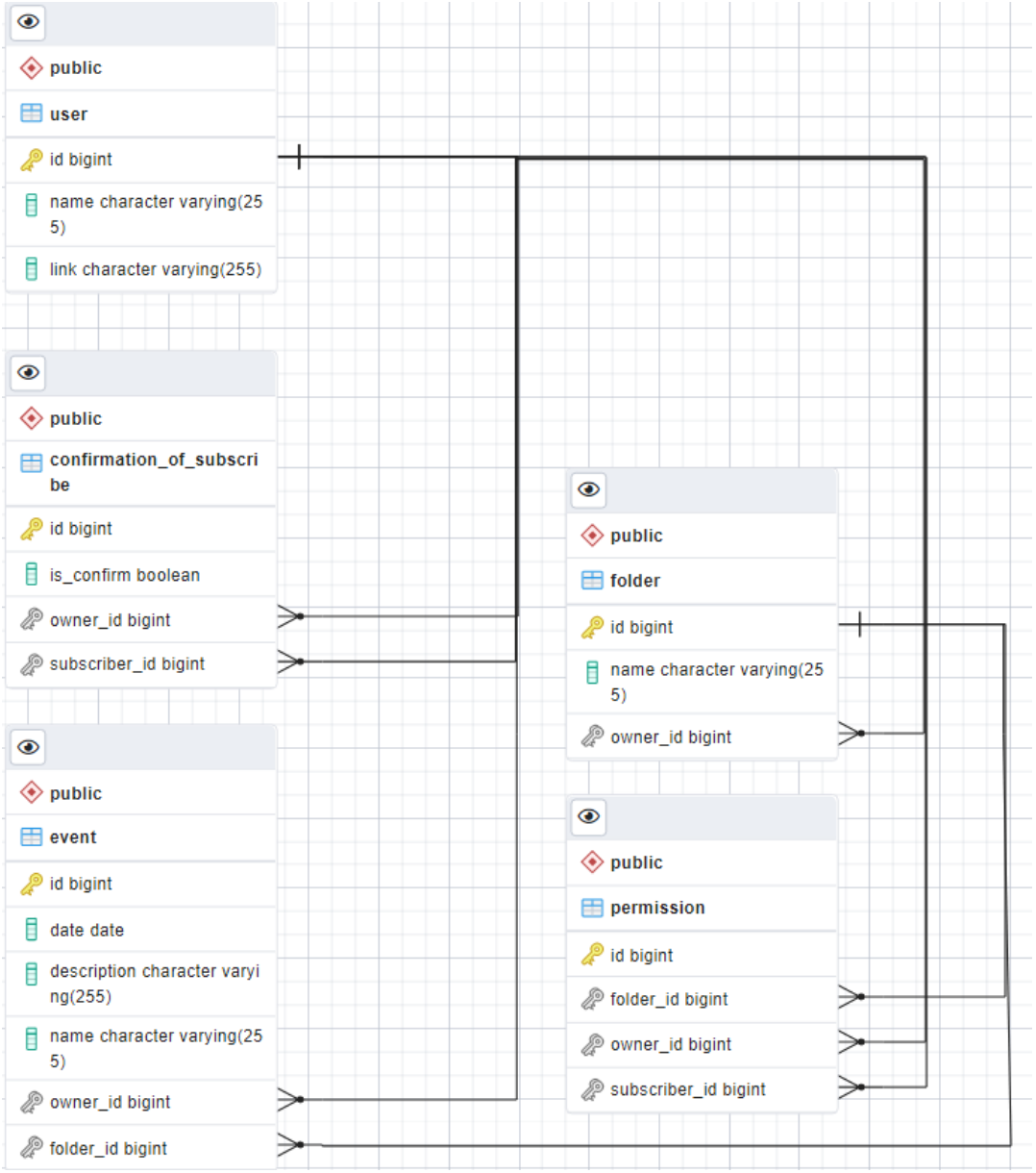

Рисунок 3 - Схема базы данных

На рисунке 3 показана схема, демонстрирующая взаимодействие таблиц между собой.

Таблицы:

— user – таблица пользователей;

— id – уникальный ключ;

— name – имя пользователя;

— link – ссылка на пользователя в Телеграм.

— confirmation_of_subscribe – таблица подписок;

— id – уникальный ключ;

— is_confirm – булево значение, указывающее на статус доступа;

— owner_id – уникальный ключ пользователя, запрашивающего подписку;

— subscriber_id – уникальный ключ пользователя-подписчика.

— event – таблица событий;

— id – уникальный ключ;

— date – дата, которой соответствует событие;

— description – описание события;

— name – название события;

— owner_id – уникальный ключ пользователя, создавшего событие.

— folder – таблица папок;

— id – уникальный ключ;

— name – название папки;

— owner_id – уникальный ключ пользователя, создавшего папку.

— permission – таблица разрешений на отслеживание папок;

— id – уникальный ключ;

— fodler_id – уникальный ключ папки;

— owner_id – уникальный ключ пользователя, создавшего папку;

— subscriber_id – уникальный ключ пользователя-подписчика.

3.4 Регистрация бота в Телеграм

Регистрация нового бота в Телеграм является важным шагом для создания функционального и эффективного Телеграм-бота. Этот процесс включает в себя несколько этапов, которые следует выполнить для успешной регистрации бота и получения необходимых доступов.

Необходимо воспользоваться специальным ботом в Телеграме – BotFather. Написав команду /start (рис. 4), можно начать взаимодействие с ним. BotFather — это официальный Телеграм-бот, созданный командой Telegram, который предоставляет удобный интерфейс для регистрации и настройки новых ботов внутри системы.

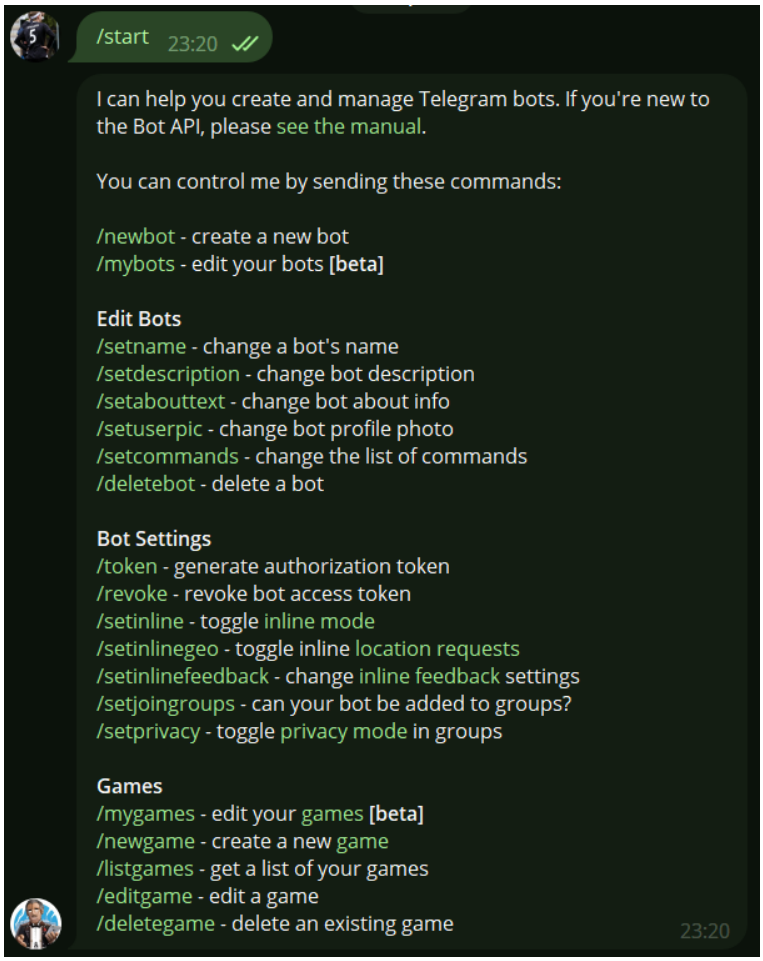

Рисунок 4 - Результат команды /start

Создание собственного бота с BotFather начинается с команды /newbot, которую нужно отправить в чат. После этого BotFather запросит у вас название нового бота. Выбираем уникальное и легко запоминающееся имя для бота (рис. 5).

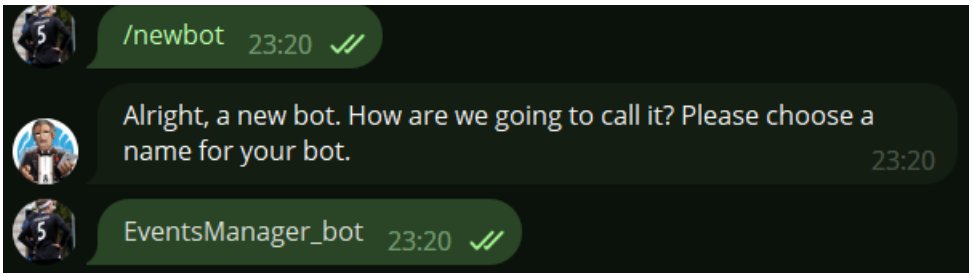

Рисунок 5 - Инициализация бота по команде /newbot

После выбора имени и юзернейма(URL, по которому будет доступен бот), BotFather предоставляет токен доступа к новому боту (рис. 6). Этот токен является ключевым элементом для связи кода с ботом и позволяет управлять его функциональностью.

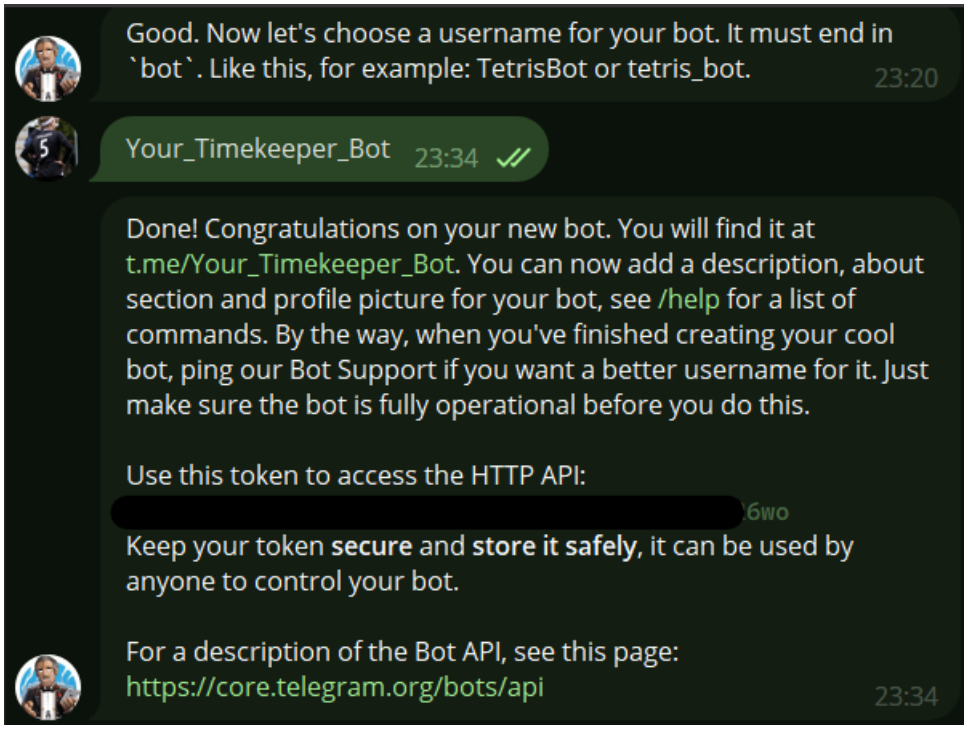

Рисунок 6 - Полученный токен (размыт)

Получив токен, важно сохранить его в надежном месте, поскольку он является уникальным и конфиденциальным. В случае потери токена, можно запросить новый у BotFather, выполнив команду /token в чате с ним.

Также в процессе регистрации бота имеется возможность настроить другие параметры, такие как аватар бота, описание и команды, которые он будет выполнять.

Завершив процесс регистрации, новый Телеграм-бот готов к использованию. Можно приступить к разработке его функциональности, взаимодействию с пользователем и настройке необходимых команд и ответов.

Важно отметить, что при разработке Телеграм-бота необходимо соблюдать политику конфиденциальности и правила использования Telegram API. Следует учитывать ограничения и рекомендации, чтобы предоставить пользователям безопасный и удобный опыт использования бота.

3.5 Пользовательский интерфейс

Диаграмма экранов взаимодействия с ботом представлена на рисунке 7.

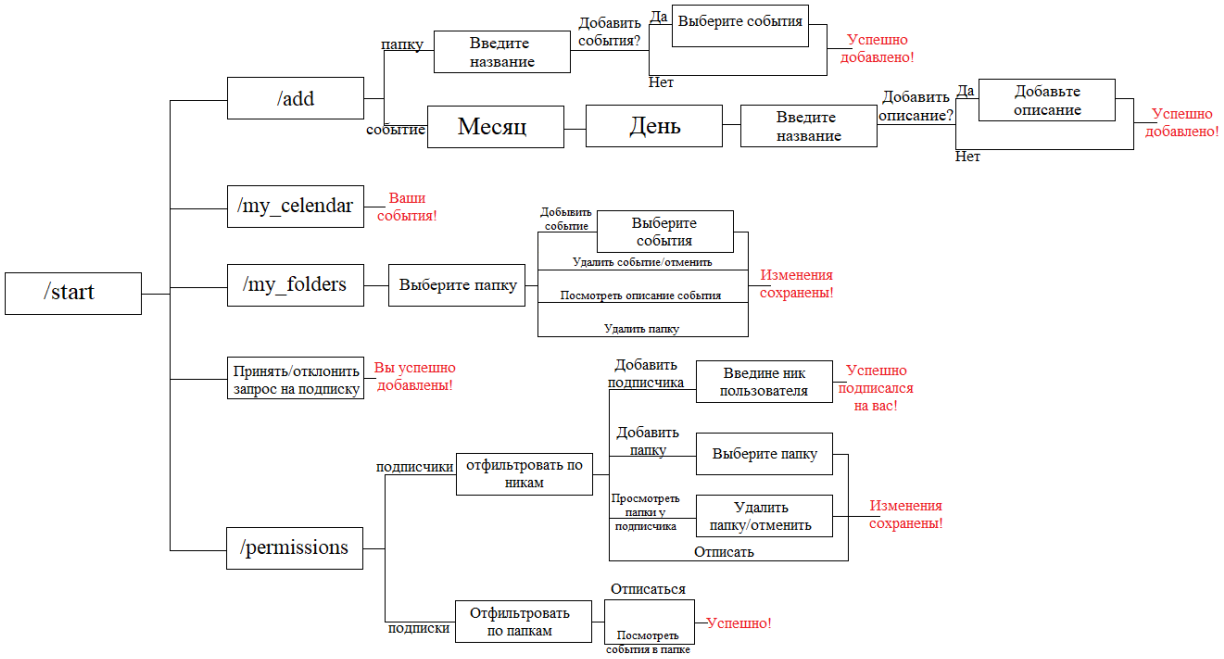

Рисунок 7 - Диаграмма взаимодействия

Следует обратить внимание, что работа с ботом начинается с команды «/start». Рассмотрим некоторые из вариантов взаимодействия.

3.5.1 Добавление события

Чтобы добавит событие необходимо написать команду «/add» и выбрать «Событие» (рис. 8).

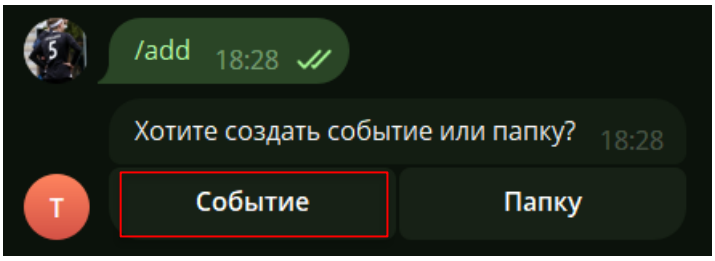

Рисунок 8 - Результат команды «/add»

Следующим шагом является выбор месяца и конкретного дня, когда событие должно отработать (рис. 9 и рис. 10).

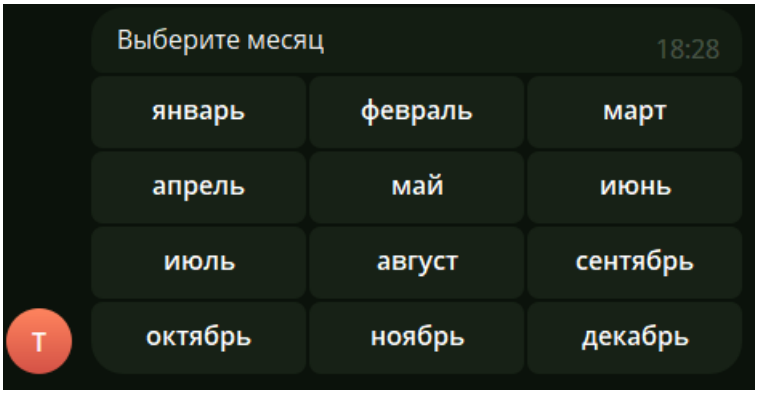

Рисунок 9 - Выбор месяца

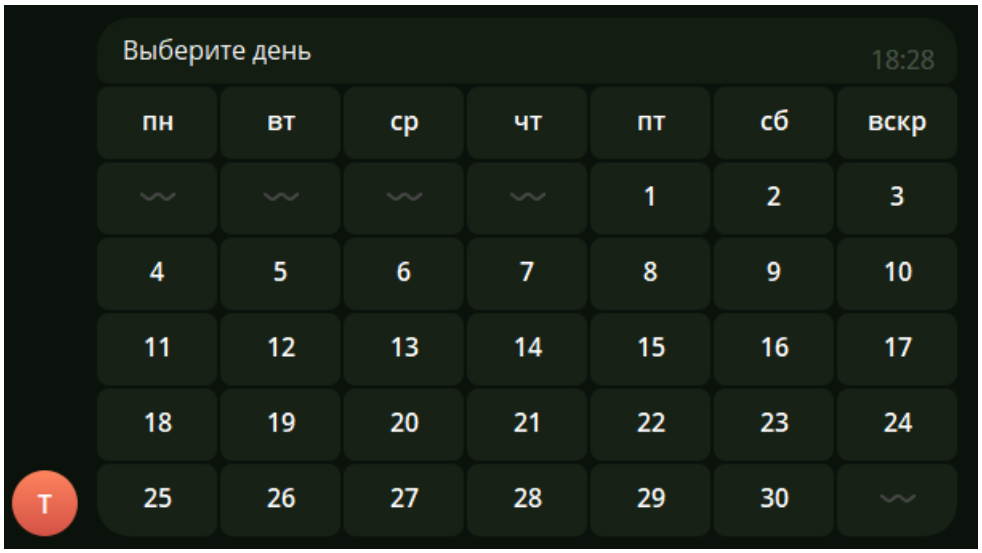

Рисунок 10 - Выбор дня

Теперь необходимо задать название событию (рис. 11).

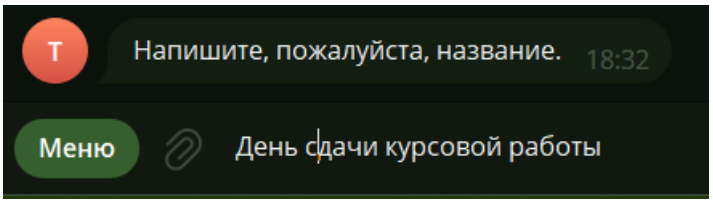

Рисунок 11 - Ввод названия события

Далее имеется возможность задать описание событию или важную информацию (рис. 12).

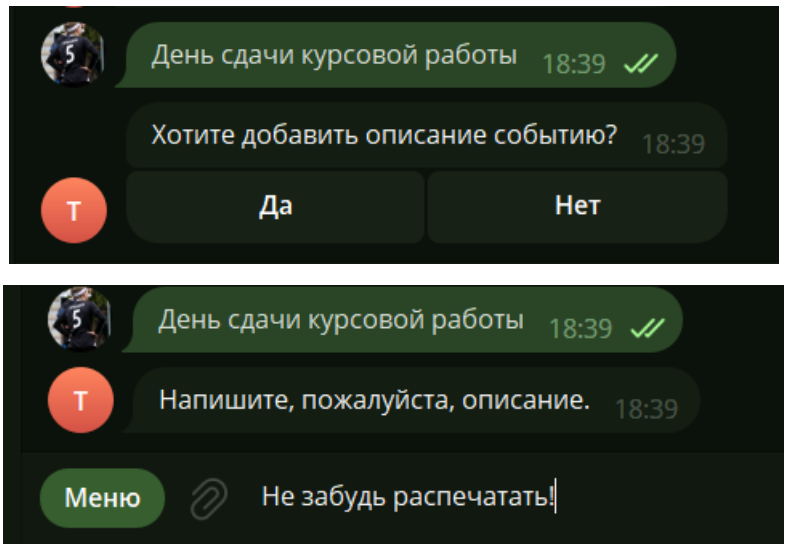

Рисунок 12 - Добавление описания

Затем следует сообщение с информацией, что событие успешно добавлено (рис.13).

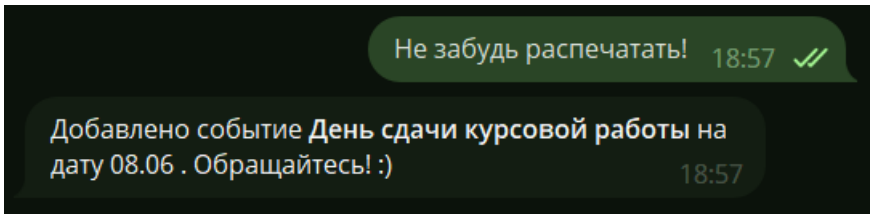

Рисунок 13 - Сообщение об успешном добавлении

После добавления события в назначенный день придет сообщение от бота, уведомляющее о его наступлении.

3.5.2 Добавление подписчика

Чтобы добавить пользователя, которому будут приходить уведомления из папок, которые другой пользователь предоставит на отслеживание, необходимо ввести команду «/permissions» и выбрать раздел «Подписчики» (рис. 14).

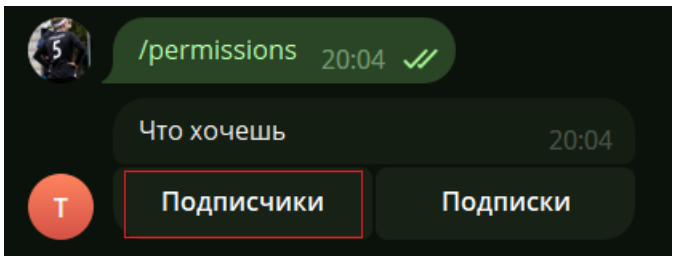

Рисунок 14 - Переход в раздел с подписчиками

На данном экране пользователь имеет возможность отписать подписчика, посмотреть, какие папки ему доступны и отредактировать этот список, добавить папку или добавить подписчика (рис. 15). Рассмотрим последнее подробнее.

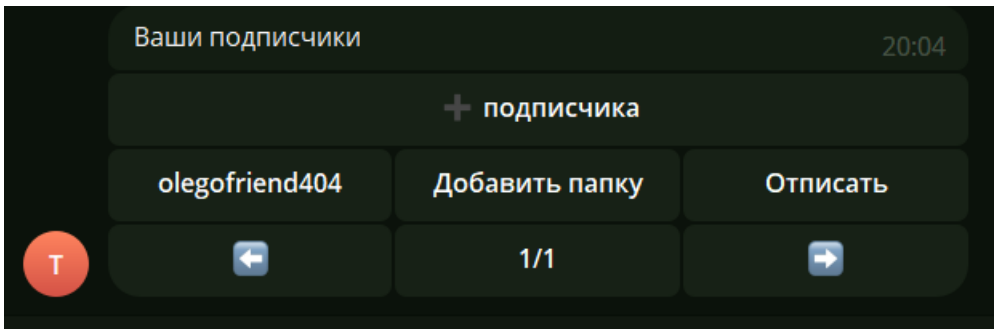

Рисунок 15 - Интерфейс экрана списка подписчиков

Чтобы добавить пользователя, необходимо нажать кнопку «+ подписчика». Далее необходимо ввести ник человека (рис. 16), которого хотите подписать, чтобы отправить ему запрос на подписку. После этого ему будет необходимо подтвердить, что дает свое согласие на отслеживание событий (рис. 17). Когда будет принято разрешение, отправляющему придет сообщение об успешном подписании (рис. 18). Следующим шагом можем быть добавление отслеживаемых папок, от событий в которых будут приходить уведомления всем, кому доступна папка.

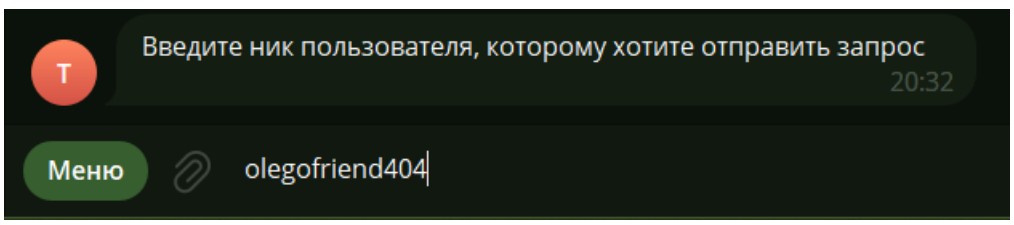

Рисунок 16 - Отправка запроса на подписку

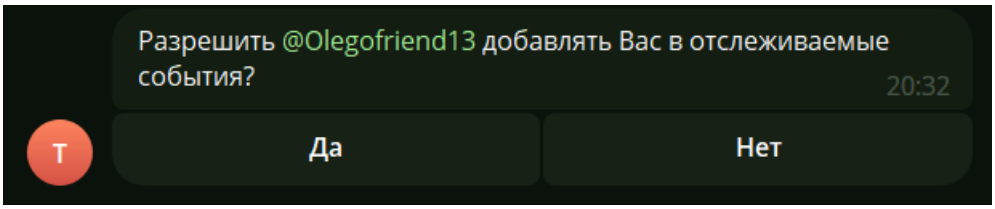

Рисунок 17 - Подтверждение отслеживания

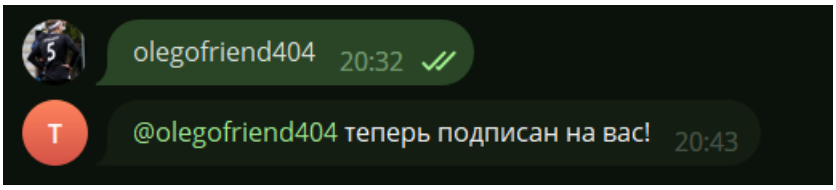

Рисунок 18 - Уведомление об успешном добавлении подписчика

3.5.3 Просмотр папок и событий

Пользователь имеет возможность посмотреть все события, написав команду «/my_calendar» (рис. 19).

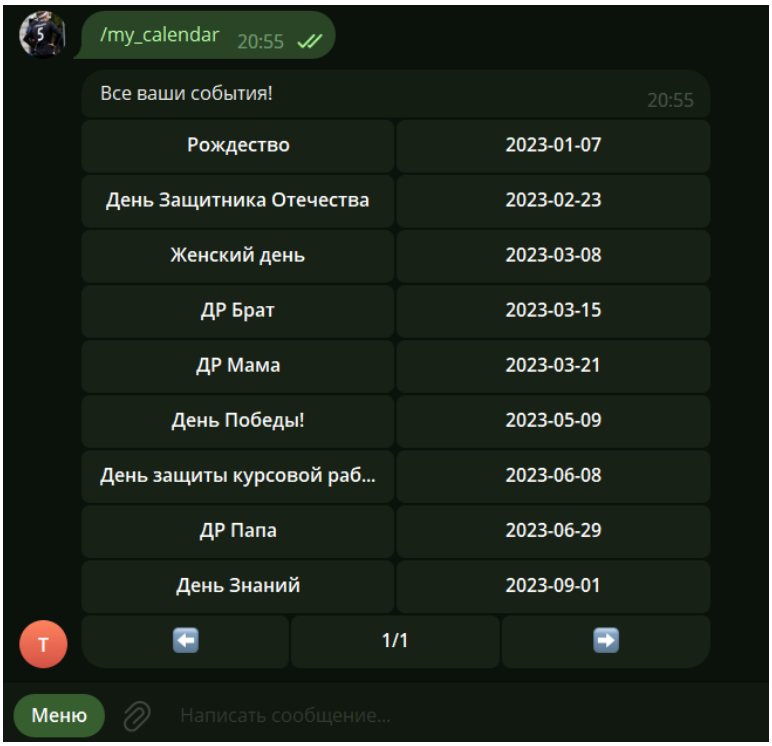

Рисунок 19 - Результат команды «/my_calendar»

Увидеть папки можно используя «/my_folders» (рис. 20). Нажав на конкретную, пользователь увидит все события, содержащиеся в папке (рис. 21). Чтобы узнать подробную информацию о событии, необходимо на него нажать (рис. 22).

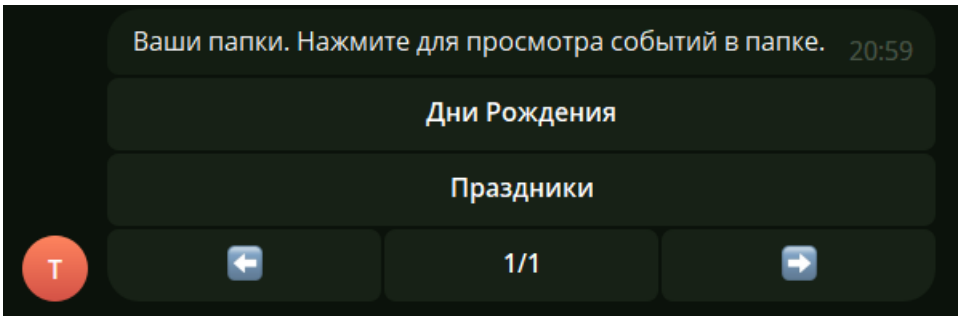

Рисунок 20 - Результат команды «/my_folders»

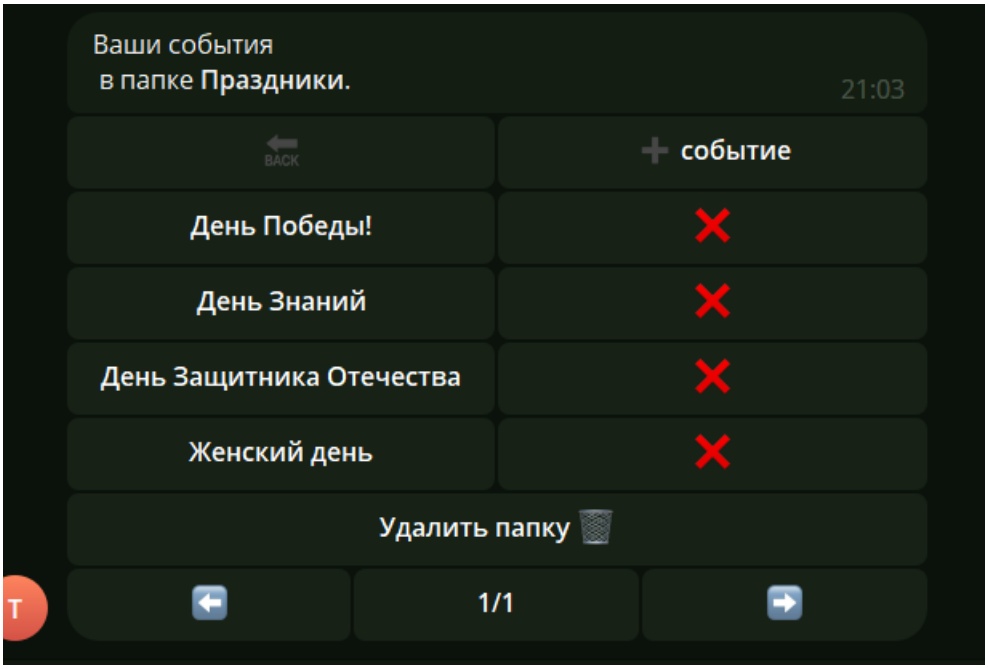

Рисунок 21 - События в папке

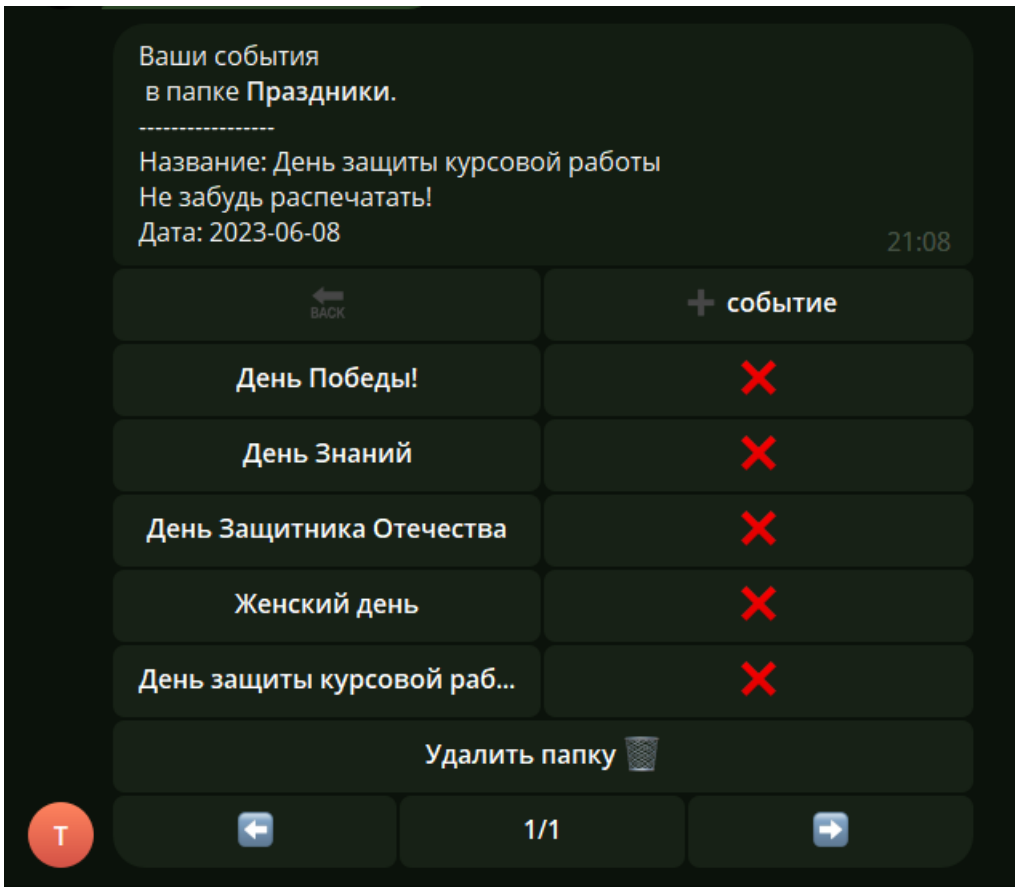

Рисунок 22 - Подробная информация при нажатии на событие «День защиты курсовой работы»

Заключение

В рамках данной курсовой работы была выполнена разработка и создание Телеграм-бота "Календарь", предоставляющего пользователям возможность эффективного управления временем и событиями. Целью работы было создание инструмента, который поможет пользователям организовать свою жизнь более эффективно и продуктивно.

В ходе работы были поставлены и успешно выполнены следующие задачи:

- Разработка функций, позволяющих пользователям создавать события и заметки на определенные даты.
- Создание системы группировки событий в папки для удобного хранения и поиска.
- Возможность делиться папками с другими пользователями для совместного планирования времени и работы над проектами.
- Интеграция с мессенджером Телеграм, обеспечивающая удобный и доступный интерфейс взаимодействия с ботом.

В ходе разработки использовались такие технологии, как Java, PostgreSQL, Telegrambots, Telegram API и Spring framework. Эти инструменты были выбраны из-за их широкой распространенности, функциональности и возможности интеграции с мессенджером Телеграм.

Телеграм-бот "Календарь" представляет собой удобный и мощный инструмент для управления временем и задачами. Его функциональные возможности, включая создание событий, группировку в папки и возможность совместной работы с другими пользователями, помогают организовать и планировать свою жизнь более эффективно. Благодаря удобному интерфейсу взаимодействия с ботом через мессенджер Телеграм, пользователи получают

быстрый и легкий доступ к своему календарю и задачам в любое время и из любого места.

Таким образом, разработка и создание Телеграм-бота "Календарь" являются актуальным и полезным решением для проблемы неэффективного управления временем. Бот предоставляет пользователям инструмент, который помогает им достигать своих целей и задач вовремя, повышая продуктивность и эффективность их жизни и работы.

Список используемых источников

1. Spring.io. Документация для фреймворка Spring [Электронный ресурс]. – Режим доступа: https://spring.io/ - Заглавие с экрана.
2. Документация для библиотеки Telegram Bots API [Электронный ресурс]. – Режим доступа: https://core.telegram.org/bots - Заглавие с экрана.
3. Документация для БД PostgreSQL [Электронный ресурс]. – Режим доступа: https://www.postgresql.org/docs/ - Заглавие с экрана.
4. Разработка Телеграм-бота [Электронный ресурс]. – Режим доступа: https://www.youtube.com/playlist?list=PLV_4DSIw2vvI3_a6L_z5AlNaIdFNqQlW2 – Заглавие с экрана.
5. Руководство по языку программирования Java [Электронный ресурс]. – Режим доступа: https://metanit.com/java/tutorial/ – Заглавие с экрана.
6. Creating a Telegram Bot in Java: from conception to deployment [Электронный ресурс]. – Режим доступа: https://codegym.cc/groups/posts/telegram-bot-in-java – Заглавие с экрана.

Приложение

Программный код конфигурации и инициализации бота и основных команд

```
import
org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.context.event.ContextRefreshedEvent
;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Component;
import org.telegram.telegrambots.meta.TelegramBotsApi;
import
org.telegram.telegrambots.meta.exceptions.TelegramApiEx
ception;
import
org.telegram.telegrambots.updatesreceivers.DefaultBotSe
ssion;
import ru.vsu.cs.Lukashev.service.TelegramBot;

@Component
public class BotInitializer {

    @Autowired
    TelegramBot bot;

    @EventListener({ContextRefreshedEvent.class})
    public void init() throws TelegramApiException{
        TelegramBotsApi telegramBotsApi = new
TelegramBotsApi(DefaultBotSession.class);
        try{
            telegramBotsApi.registerBot(bot);
        }
        catch (TelegramApiException e){

        }
    }

}
```

```
public TelegramBot(BotConfig botConfig) {
        this.botConfig = botConfig;
        List<BotCommand> commandList = new
ArrayList<>();
        commandList.add(new BotCommand(ADD_COMMAND,
"Добавь папку или событие"));
        commandList.add(new
BotCommand(MY_CALENDAR_COMMAND,  "Посмотри свои
ближайшие события"));
        commandList.add(new
BotCommand(MY_FOLDERS_COMMAND,   "Посмотри свои
папки"));
        commandList.add(new
BotCommand(PERMISSIONS_COMMAND,  "Посмотри свои
подписки/наблюдателей"));
        try {
            this.execute(new SetMyCommands(commandList,
new BotCommandScopeDefault(), null));
        } catch (TelegramApiException e) {
            log.error("Error : " + e);
        }
    }
```

Программный код метода для генерации экрана вывода папок, на которые подписан пользователь

```
    private                                          void
generateSubscriptionByFolderButton(long messageId, long
chatId){
            TemplateMes templateMes = new TemplateMes();

            BlockButtons headBlock = new BlockButtons();
            headBlock.add(
                    new LineButtons(
```

```
                        new
ArrayList<>(List.of(EmojiParser.parseToUnicode(":back:"
))),
                        new
ArrayList<>(List.of(CHECK_SUBSCRIPTIONS_BUTTON))));

            BlockButtons bodyBlock = new BlockButtons();
            List<Folder>          folderList          =
getFoldersBySubscriberId(chatId);
            List<Folder>       folderOnPageList       =
Pagination.getEntityListForPage(folderList,
getCurrentPage(chatId), 10);
            var        subscribersLinkListSize        =
folderList.size();
            List<String> textList = new ArrayList<>();
            List<String>    callbackDataList    =    new
ArrayList<>();

            for(Folder f : folderOnPageList){
                textList = new ArrayList<>();
                callbackDataList = new ArrayList<>();

                textList.add(f.getName());
                textList.add(f.getOwnerID().getLink());
                textList.add("Отписаться");


callbackDataList.add(EVENTS_IN_SUBSCRIPTION_FOLDER_BUTT
ON + f.getId());
```

```
callbackDataList.add(CHECK_SUBSCRIPTIONS_BY_FOLDERS_BUT
TON);

callbackDataList.add(DELETE_SUBSCRIPTION_BUTTON        +
f.getId());

                bodyBlock.add(
                        new LineButtons(
                                textList,
                                callbackDataList
                        )
                );
            }

            BlockButtons      underBlock      =      new
BlockButtons();
            underBlock
                    .add(new      LineSwitchButtons(new
ArrayList<>(List.of(SWITCH_PAGE_BUTTON                 +
IN_CHOOSE_EVENT)),              getCurrentPage(chatId),
subscribersLinkListSize));

            templateMes
                    .addBlock(headBlock)
                    .addBlock(bodyBlock)
                    .addBlock(underBlock)
                    .create(chatId,   (int)   messageId,
"Папки, на которые вы подписаны");
```

```
            executeEditMessage(templateMes, "Error: ");
        }
```

Программный код класса Шаблона сообщения от бота пользователю

```
    import lombok.Getter;
    import lombok.Setter;
    import
org.telegram.telegrambots.meta.api.methods.send.SendMes
sage;
    import
org.telegram.telegrambots.meta.api.methods.updatingmess
ages.EditMessageText;
    import
org.telegram.telegrambots.meta.api.objects.replykeyboar
d.InlineKeyboardMarkup;
    import
org.telegram.telegrambots.meta.api.objects.replykeyboar
d.buttons.InlineKeyboardButton;


    import java.util.ArrayList;
    import java.util.Collections;
    import java.util.Iterator;
    import java.util.List;

    @Setter @Getter
    public class TemplateMes {
```

```
        private List<BlockButtons> blockButtonsList;

        private        List<List<InlineKeyboardButton>>
bodyButtonsList;
        private EditMessageText editMessage;
        private SendMessage sendMessage;

        public TemplateMes() {
            this.bodyButtonsList = new ArrayList<>();
            this.blockButtonsList = new ArrayList<>();
        }

        public  TemplateMes  create(Long  chatId,  int
messageId, String text){

            this.editMessage = new EditMessageText();
            editMessage.setChatId(chatId);
            editMessage.setMessageId(messageId);
            editMessage.setText(text);
            editMessage.enableHtml(true);

            InlineKeyboardMarkup markupInline          =
new InlineKeyboardMarkup();
            List<List<InlineKeyboardButton>> rowsInline
= new ArrayList<>();
            Iterator<BlockButtons> blockButtonsIterator
= blockButtonsList.iterator();
            if(!bodyButtonsList.isEmpty())
```

```
rowsInline.addAll(blockButtonsIterator.next().generate(
));
            rowsInline.addAll(bodyButtonsList);

            while (blockButtonsIterator.hasNext())

rowsInline.addAll(blockButtonsIterator.next().generate(
));

            markupInline.setKeyboard(rowsInline);
            editMessage.setReplyMarkup(markupInline);
            return this;
        }

        public EditMessageText getEditMessage() {
            return editMessage;
        }

        public SendMessage getSendMessage(){
            return sendMessage;
        }


        public TemplateMes create(Long chatId, String
text){
            this.sendMessage = new SendMessage();
            sendMessage.setChatId(chatId);
            sendMessage.setText(text);
```

```
            InlineKeyboardMarkup markupInline          =
new InlineKeyboardMarkup();
            List<List<InlineKeyboardButton>> rowsInline
= new ArrayList<>();


            for(BlockButtons block : blockButtonsList)
                rowsInline.addAll(block.generate());

            markupInline.setKeyboard(rowsInline);
            sendMessage.setReplyMarkup(markupInline);


            return this;
        }

        public     TemplateMes     addBlock(BlockButtons
blockButtons){
            this.blockButtonsList.add(blockButtons);

            return this;
        }


    }
```