

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

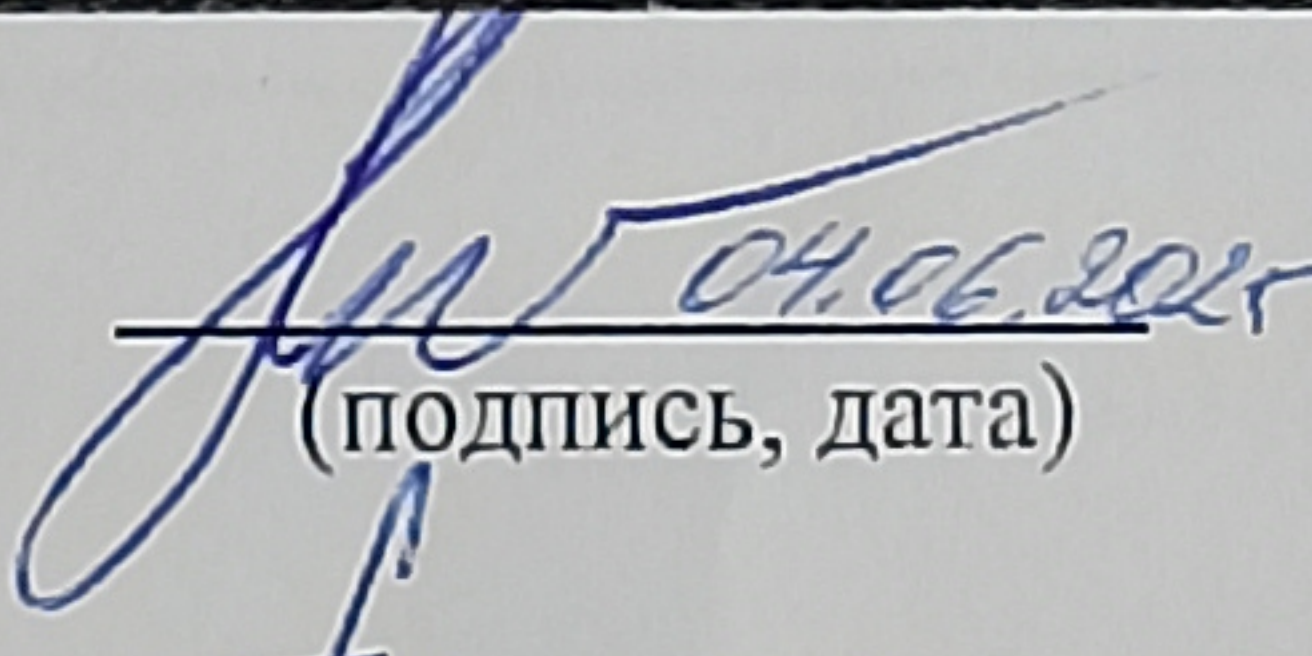
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

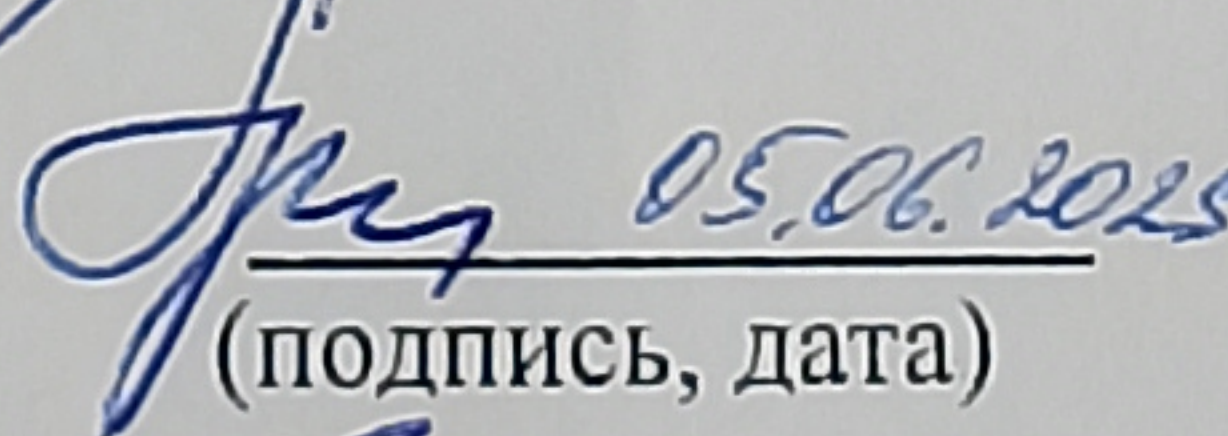
TELEGRAM-БОТ ДЛЯ ПОМОЩИ ПРИ
ВСТУПЛЕНИИ В РОССИЙСКИЕ
СТУДЕНЧЕСКИЕ ОТРЯДЫ

Студент


(подпись, дата)

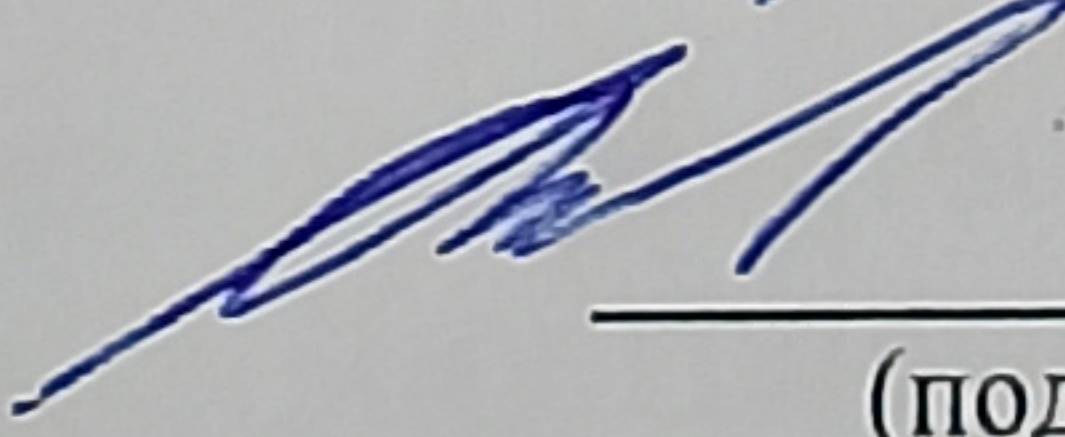
Кирсанова Татьяна Игоревна
(ФИО полностью)

Руководитель


(подпись, дата)

Токарев А.Н.
(фамилия, инициалы)

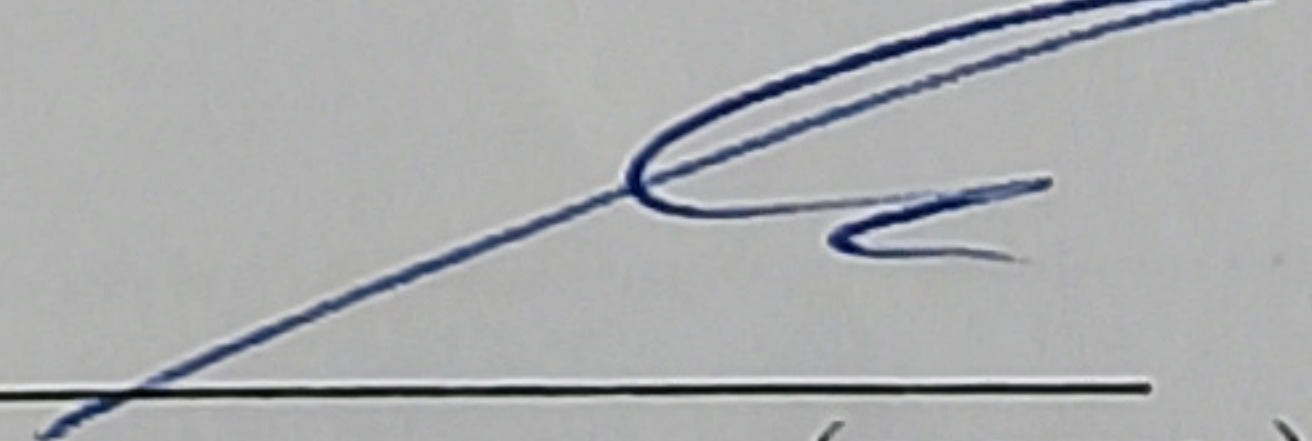
Нормоконтролёр


(подпись, дата)

Бычков А.С.
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой

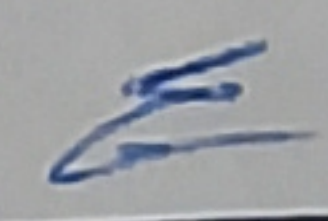

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

6 июня 2025 № 7)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

М.А. Митрохин

«06» мая 2025 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Кирсанова Татьяна Игоревна гр. 21BBB1 факультета ФВТ

2. Тема работы Telegram-бот для помощи при вступлении в Российские студенческие отряды

Тема утверждена приказом ПГУ № 601/о от "06" мая 2025 г.

3. Руководитель бакалаврской работы Токарев А.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать телеграмм-бота интегрированного в Российские студенческие отряды для автоматизированного сбора заявок на вступление и информирования пользователей. Язык программирования python, среда разработки PyCharm. Программа должна обладать интуитивно понятным интерфейсом в мессенджере telegram и использовать следующие технологии:

1.Telegram

2.Telebot

3.Асинхронное программирование

4.Сетевое программирование

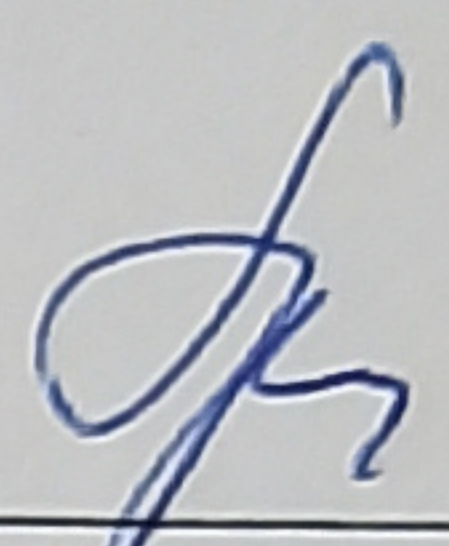
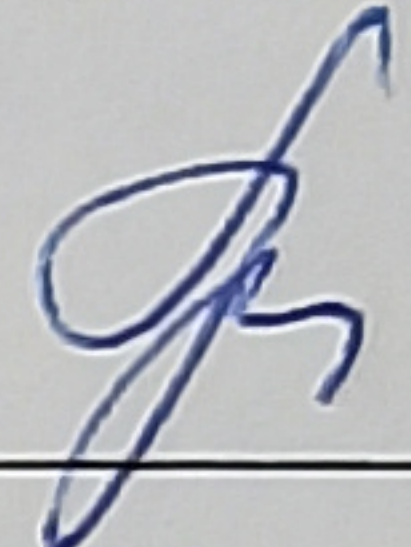
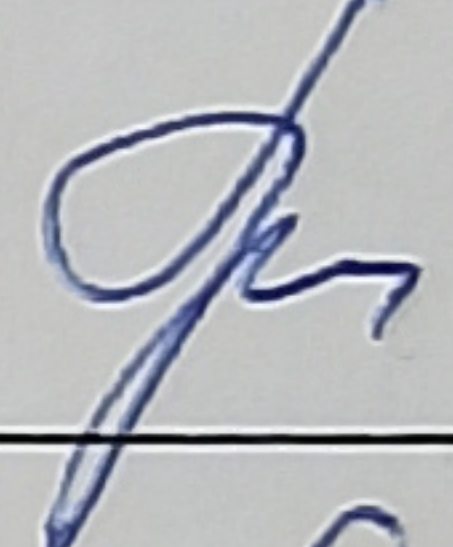
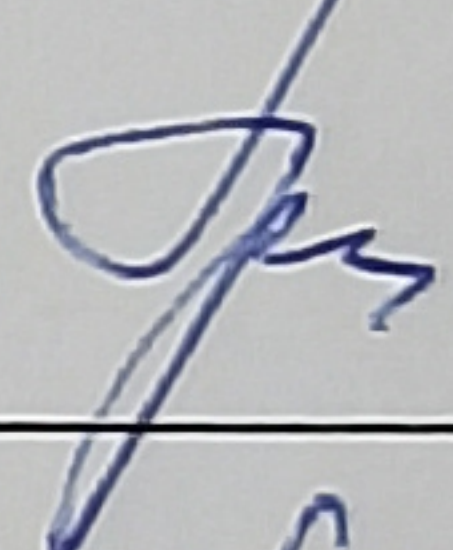
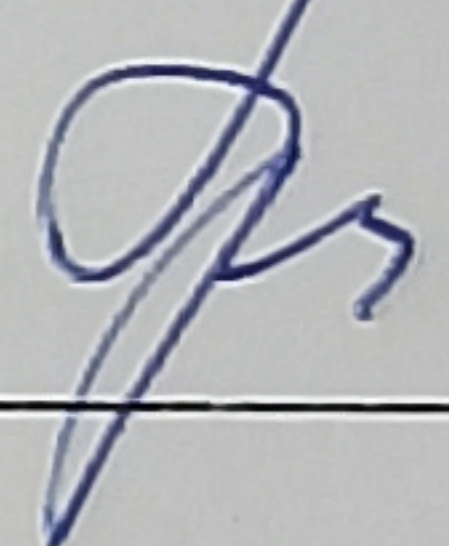
5.Beautiful Soup

6.SMTP

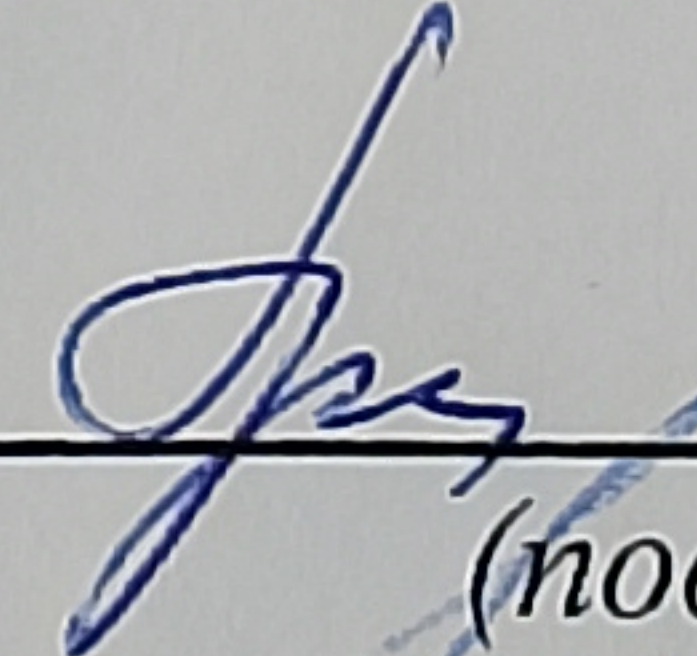
5. Перечень подлежащих разработке вопросов

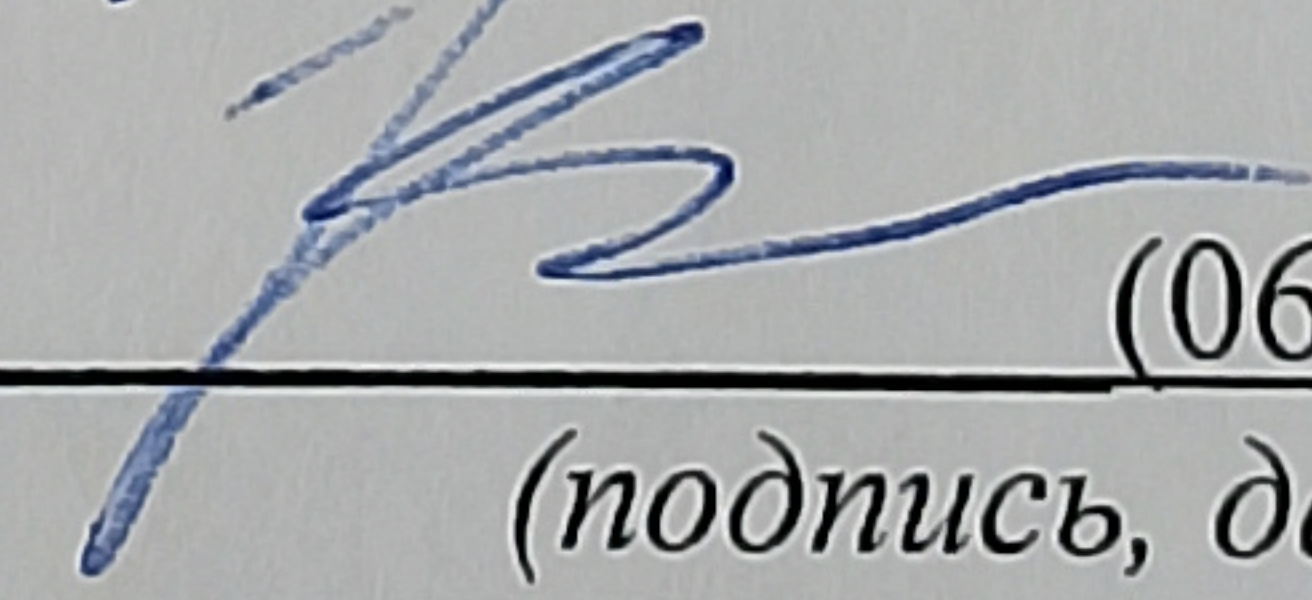
Требуется произвести анализ предметных областей: изучить существующие информационные системы, выделить их достоинства и недостатки. Выявить потребности пользователей. Выбрать язык программирования, среду разработки, СУБД которые подойдут для создания телеграм-бота. Определить архитектуру приложения и вспомогательные средства разработки. Изучить функционал библиотек для разработки информационной системы на основе телеграм-бота.

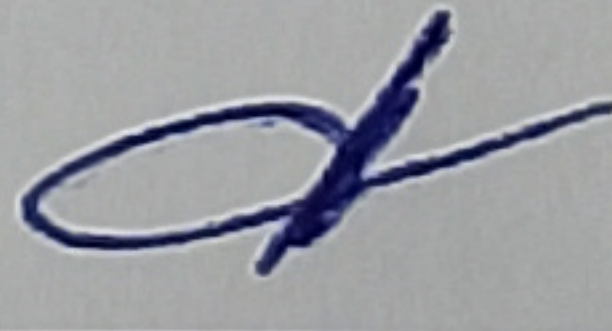
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	26.04.2025 – 28.05.2025	
2	Анализ требований	32	29.04.2025 – 30.05.2025	
3	Создание макета графического интерфейса	30	31.04.2025 – 3.05.2025	
4	Выбор технологий, применяемых при проектировании и разработке	40	4.05.2025 – 8.05.2025	
5	Реализация графического интерфейса	66	9.05.2025 – 15.05.2025	
6	Разработка основной части программы	70	16.05.2025 – 23.05.2025	
7	Оформление пояснительной записки	54	24.05.2025 – 31.05.2025	

Дата выдачи задания "06" ____мая____ 2025 ____г.

Руководитель бакалаврской работы  (06.05.2025) Токарев А.Н.
(подпись, дата)

Задание к исполнению принял студент  (06.05.2025) Кирсанова Т.И.
(подпись, дата)

Работу к защите допустить
Декан факультета  09.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

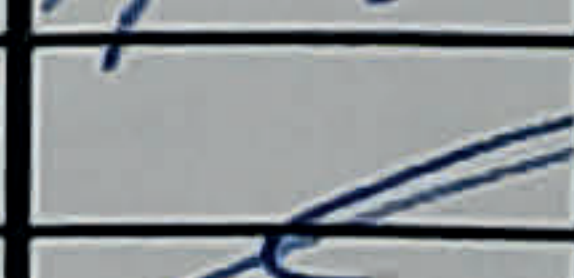
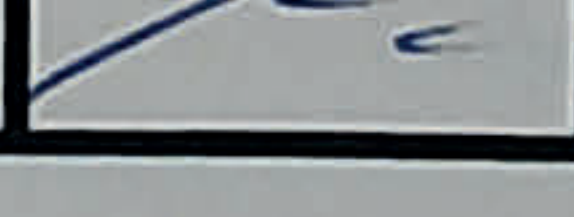
Бакалаврская работа содержит 80 листов, 35 рисунков, 1 таблицу, 7 источников, 2 приложения.

БОТ, ОБРАБОТКА СОСТОЯНИЙ, ПОДАЧА ЗАЯВКИ, ВЫВОД НОВОСТЕЙ, БАЗА ДАННЫХ, TELEBOT, PSYCORG2, PYTHON, VS CODE, TIMEWEB

Цель работы – разработать инструмент, позволяющий автоматизировать процесс сбора заявок на вступление в студенческие отряды.

Одним из путей достижения поставленной задачи является информационная система на основе телеграм-бота, обеспечивающая выполнение следующих функций: интерактивное взаимодействие с пользователями через структурированное меню с возможностью выбора различных разделов и команд, сбор и хранение данных о кандидатах в студенческие отряды с проверкой корректности вводимой информации и сохранением в базу данных SQLite, предоставление информации об имеющихся направлениях студенческих отрядов, информационная поддержка процесса вступления в отряды, автоматическое обновление новостной ленты с официального сайта движения с отображением новостей в формате с изображениями, управление контактами с возможностью получения информации о руководителе.

В конечном итоге разработана информационная система на основе телеграм-бота, которая обеспечивает интерактивное взаимодействие с пользователями через структурированное меню, сбор и хранение данных о кандидатах в базу SQLite, при этом система полностью соответствует цели данной выпускной квалификационной работы и была разработана с использованием следующих технологий: Python, telebot, sqlite3, requests, BeautifulSoup, и готова к эксплуатации после успешного тестирования.

					ПГУ1.090301.39.001 ПЗ		
Изм.	Лист	№ докум.	Подпись	Дата	Telegram-бот для помощи при вступлении в Российские студенческие отряды		
Разраб.		Кирсанова Т.И.		04.06.25			
Провер.		Токарев А.Н.		05.06.25			
Н. Контр		Бычков А.С.		05.06.25			
Утверд.		Митрохин М.А.		05.06.25			
					Лит.	Лист	Листов
						2	80
					ФВТ, каф. ВТ гр. 21BBB1		

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Telegram-бот для помощи при вступлении в Российские студенческие отряды	8
1.1 Анализ предметной области	8
1.2 Анализ решения поставленной задачи	9
1.3 Анализ актуальности разработки	11
1.4 Анализ аналогов информационных систем	12
1.5 Анализ архитектуры и структура бота	17
1.6 Постановка задачи	20
1.7 Выбор инструментов разработки	20
1.7.1. Платформа для реализации системы.....	21
1.7.2. Выбор языка программирования.....	21
1.7.3. Выбор СУБД.....	22
1.8 Реализация	24
1.8.1. Регистрация телеграм-бота	24
1.8.2. Работа с БД	27
1.8.3. Описание доступных функций	27
1.8.4. Описание программы.....	39
2 Безопасность жизнедеятельности	39
2.1 Правила безопасности при использовании ПК	57
Заключение	61
Список использованных источников	65
Приложение А – UML – диаграммы	66
Приложение Б – Листинг программы	70

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД (база данных) - структурированная коллекция данных, организованных и хранимых в электронном виде, используемая в Telegram-боте для хранения информации о пользователях и их анкетах.

ИС (информационная система) - комплексное информационное программно-аппаратное решение, в данном случае Telegram-бот, предназначенный для автоматизации процессов взаимодействия со студенческими отрядами.

ПГУ (Пензенский государственный университет) - образовательное учреждение, в рамках которого реализуется проект, а также контекст, в котором действуют студенческие отряды.

ПК (персональный компьютер) - электронное устройство, используемое для разработки и тестирования Telegram-бота.

РСО (Российские студенческие отряды) - молодежное движение, для поддержки которого разработан Telegram-бот, упрощающий процесс вступления и информирования кандидатов.

СУБД (система управления базами данных) - программное обеспечение, такое как SQLite, применяемое для создания, управления и обработки баз данных в рамках проекта.

ЯП (язык программирования) - формальный язык, в данном случае Python, используемый для написания кода Telegram-бота.

API (Application Programming Interface) - описание способов взаимодействия Telegram-бота с платформой Telegram через Telegram API.

IDE (Integrated Development Environment) - интегрированная среда разработки, например Visual Studio Code, используемая для написания и отладки кода проекта.

SQL (Structured Query Language) - язык структурированных запросов, применяемый для работы с базой данных SQLite в Telegram-боте.

					ПГУ1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

Введение

В современном мире, когда количество внеучебных мероприятий и студенческих движений постоянно растет, возникает острая проблема поиска информации о возможностях вступления и участия в них. Российские студенческие отряды, являются крупнейшим молодежным движением и занимают важное место в подготовке кадров, сталкиваются с тем, что потенциальные кандидаты тратят значительное время на поиск необходимых данных среди открытых источников, что затрудняет процесс принятия решений.

Одним из лучших способов решения этой проблемы могут быть телеграм-боты. Они становятся всё более популярными в образовании. Это удобные инструменты, которые помогают общаться с кандидатами и держать их в курсе.

Плюсы телеграм-бота очевидны: не нужно устанавливать дополнительное ПО, достаточно иметь аккаунт в Telegram, а безопасность передачи данных на высоком уровне.

Информационная система «Телеграм-бот РСО», созданная в рамках этой работы, объединяет все нужные функции для комфортного взаимодействия с кандидатами. Она упрощает процесс предоставления информации о направлениях студенческих отрядов, облегчает работу руководителей и помогает быстро находить нужные данные и рекомендации.

Цели выпускной квалификационной работы:

- освоение современных средств программной разработки;
- изучение существующих проблем в системе информирования абитуриентов о возможностях участия в студенческих отрядах;
- исследование возможностей применения телеграмм-ботов для автоматизации взаимодействия с кандидатами;
- разработка информационной системы «Телеграм-бот РСО» с необходимым функционалом;
- определение требований к интерфейсу и управлению контентом бота;
- создание прототипа системы и тестирование его функциональности;

					ПГУ1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

- оценка эффективности внедрения бота в работу студенческих отрядов;
- анализ результатов и формирование выводов о преимуществах использования телеграм-ботов;
- разработка рекомендаций по дальнейшему развитию системы на основе обратной связи от пользователей;
- формирование обобщенных рекомендаций по применению информационных технологий для оптимизации работы студенческих отрядов.

Идея создать Telegram-бота появилась, потому что нужно было упростить работу для руководителей и кандидатов. Этот чат-бот поможет быстрее находить нужную информацию из открытых источников и сделает процесс проще и удобнее.

Основная польза чат-бота в том, что он ориентирован на кандидатов, предоставляя им актуальную информацию и советы. В процессе его разработки использовались официальные документы для разработчиков и информацию о направлениях подготовки студентов, рабочих местах и новостях из студенческих отрядов.

1 Telegram-бот для помощи при вступлении в Российские студенческие отряды

1.1 Анализ предметной области

Чтобы создать хорошую информационную систему, нужно внимательно рассмотреть, как работают студенческие отряды, и понять, что именно нужно всем, кто в этом заинтересован. В рамках аналитического исследования была проведена комплексная оценка процессов работы студенческих отрядов и их системы управления.

Анализ деятельности студенческих отрядов является основополагающим этапом в разработке информационной системы «Telegram-бот РСО». Проведенные исследования позволили выявить основные проблемы и недостатки, которые могут быть решены с помощью автоматизированной системы на базе телеграм-бота.

Первым этапом анализа стало изучение основных процессов работы студенческих отрядов. Это включало описание процесса подачи заявок на вступление, прохождения обучения, участия в проектах, оформления документации и требований для участия в отрядах. Каждый процесс был детально проанализирован.

Ранее кандидатам необходимо было лично посещать офис отряда, знакомиться с условиями участия и оставлять свои контактные данные. Теперь все это можно сделать удаленно через телеграм-бот, сделав несколько кликов. Управление календарём мероприятий и проектов, которое раньше осуществлялось по телефону, теперь происходит через удобный цифровой интерфейс.

Система подачи заявок на участие в проектах стала более эффективной: кандидат может подать заявку только после прохождения необходимого обучения и получения соответствующих навыков. Это не только ускоряет

процесс, но и исключает ситуации, когда один человек подавал заявки на участие во множестве проектов одновременно.

Изменения коснулись и документооборота – теперь все необходимые документы можно хранить в электронном виде прямо в боте. Достаточно отправить номер документа, а координатор отряда проверит его подлинность. Вся важная информация доступна непосредственно в телеграм-боте, что существенно упрощает взаимодействие.

В ходе исследования были определены потребности всех участников процесса: будущих членов отрядов и командного состава. Основные требования к системе были простыми: она должна быть удобной, безопасной, быстрой и надежной.

В нынешней системе есть проблемы, такие как трудности с координацией мероприятий, ручной учет участников и организация аттестации. Все это можно решить с помощью телеграм-бота.

По результатам анализа мы определили главные требования к новой системе: она должна быть легкой в использовании, безопасной и надежной. Важно, чтобы система была автоматизированной, чтобы сократить ручной труд и ускорить выполнение рутинных задач.

Разработанная информационная система на основе телеграм-бота позволит оптимизировать процессы управления студенческими отрядами, повысить эффективность работы координаторов и улучшить качество взаимодействия между всеми участниками движения.

1.2 Анализ решения поставленной задачи

Telegram-бот представляет собой специализированное программное обеспечение, функционирующее в экосистеме мессенджера Telegram и обеспечивающее интерактивное взаимодействие с пользователями посредством системы сообщений и чат-интерфейса [1]. Согласно имеющимся данным, функционал ботов охватывает широкий спектр задач: от управления

					ПГУ1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

информационными потоками до обработки транзакций и выполнения платежных операций.

В настоящее время боты Telegram получили широкое распространение в различных секторах экономики. Их успешно применяют финансовые учреждения, предприятия общественного питания, медицинские организации, авиаперевозчики и ритейлеры для оптимизации клиентского сервиса и внутренних бизнес-процессов.

Ключевые достоинства телеграм-ботов при создании масштабных информационных систем:

- универсальная доступность – возможность использования для всех обладателей аккаунта в Telegram;

- интуитивно понятный интерфейс – отсутствие необходимости в освоении сложных команд;

- высокая производительность – моментальная обработка пользовательских запросов;

- процессная автоматизация – возможность автоматизации записи, управления расписанием, приема платежей и тестирования;

- криптографическая защита – обеспечение безопасности данных через протокол TLS;

- техническая стабильность – функционирование на инфраструктуре Telegram гарантирует бесперебойную работу;

- многофункциональность – применимость в различных информационных системах;

- экономическая эффективность – меньшие затраты на разработку по сравнению с мобильными приложениями;

- адаптивность – возможность настройки под конкретные пользовательские требования;

- системная совместимость – потенциал для интеграции с различными программными решениями;

– функциональная масштабируемость – применимость в образовании, бизнесе и развлекательной сфере;

– расширенный инструментарий – поддержка платежей, обмена файлами и массовых рассылок;

– пользовательская настройка – возможность адаптации под индивидуальные предпочтения.

Telegram-боты реально помогают улучшить сервис для клиентов. Они дают пользователям быстрый доступ к консультациям, позволяют заказывать услуги и получать поддержку в любое время, что делает людей более довольными.

1.3 Анализ актуальности разработки

Создание Telegram-бота для автоматизации работы студенческих отрядов – это действительно актуальная задача.

Во-первых, студенческие отряды играют большую роль в жизни студентов, помогая развивать лидерские навыки и активно участвовать в социальной жизни. Но управлять такими отрядами бывает непросто, особенно если делать всё вручную.

Во-вторых, современные технологии могут сильно упростить общение и взаимодействие среди членов отрядов. Telegram-бот мог бы сделать такие вещи, как подача заявок, управление документами и координация мероприятий, намного проще.

Также Telegram-боты становятся популярными для автоматизации разных процессов в бизнесе и учебе. Они позволяют быстро получать информацию и делают общение легче, а рутинные задачи — менее утомительными.

Наконец, всем нам сейчас привычнее получать информацию и услуги через цифровые каналы, не выходя из дома. Так что такой бот мог бы стать отличным средством для участников движения. взаимодействия с организаторами, получения актуальной информации и участия в проектах.

Таким образом, разработка Telegram-бота для студенческих отрядов имеет высокую значимость как для улучшения качества работы координаторов, так и для повышения эффективности взаимодействия между всеми участниками движения. Проект способствует развитию современных технологий в сфере студенческого самоуправления и создает основу для дальнейшего совершенствования системы управления студенческими отрядами.

1.4 Анализ аналогов информационных систем

Анализ существующих решений играет ключевую роль в процессе разработки информационной системы, поскольку позволяет провести детальное исследование аналогичных продуктов и выявить их сильные и слабые стороны. В рамках данного проекта был осуществлен комплексный анализ рынка информационных систем для студенческих отрядов с особым вниманием к решениям на базе Telegram-ботов. Хотя на текущий момент не существует полностью интегрированной системы, построенной исключительно на базе Telegram, представляется целесообразным провести сравнительный анализ с существующими веб-платформами и мобильными приложениями, предоставляющими схожие функции [2].

В ходе исследования были выявлены следующие альтернативные решения: официальный сайт РСО, группа РСО в социальной сети ВКонтакте, Telegram-бот @findworkerBot и ряд других информационных платформ. Данные системы предоставляют функционал для работы с информацией о направлениях деятельности, организации мероприятий и взаимодействия с участниками движения, однако имеют существенные ограничения в плане удобства использования и полноты предоставляемых услуг.

Рассмотрим сайт РСО как один из основных аналогов разрабатываемого решения. Данная платформа позволяет пользователям получать информацию о направлениях деятельности, просматривать новости и участвовать в мероприятиях. Однако сайт имеет ряд существенных недостатков: сложный и

неинтуитивный интерфейс, отсутствие автоматического обновления данных в личном кабинете, что требует от пользователей самостоятельного внесения информации. Пример работы аналога «Сайт РСО» рассмотрен на рисунке 1.1.

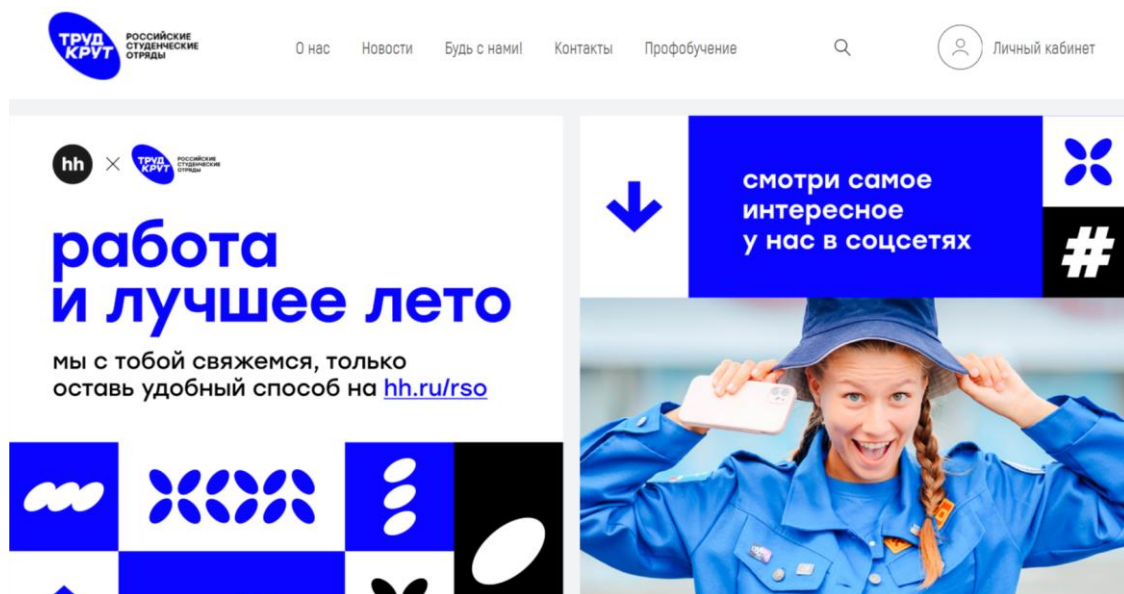


Рисунок 1.1 – Обзор аналога «Сайт РСО»

Группа РСО в социальной сети ВКонтакте предоставляет функционал для ознакомления с деятельностью студенческих отрядов, включая возможность прохождения анкеты для вступления в ряды РСО. Однако из-за стандартного формата группы возникают сложности с поиском информации о различных направлениях деятельности, отсутствует удобный личный кабинет и нет возможности автоматизированного обновления данных. Кроме того, в группе не реализованы интерактивные элементы и современные инструменты взаимодействия с пользователями. Пример работы аналога «Группа РСО» рассмотрен на рисунке 1.2.

Ф.И.О*

Ваш ответ

Дата рождения*

Введите ответ

Место учебы*

Ваш ответ

Ссылка на страницу в ВК*

Ваш ответ

Рисунок 1.2 – Обзор аналога «Твиспо»

В мессенджере Telegram функционирует бот @findworkerBot, специализирующийся на поиске работы для студентов. Данный бот предоставляет удобный поиск вакансий, возможность фильтрации предложений и простой интерфейс. Однако у него есть существенные ограничения: информация о вакансиях не всегда актуальна, отсутствует специализированный функционал для студенческих отрядов, нет возможности подробно ознакомиться с направлениями деятельности и нет системы подачи заявок в конкретные отряды. В отличие от разрабатываемого решения, бот не ориентирован на специфику студенческих отрядов и не предоставляет комплексного подхода к взаимодействию с целевой аудиторией. Пример работы аналога «findworkerBot» рассмотрен на рисунке 1.3.

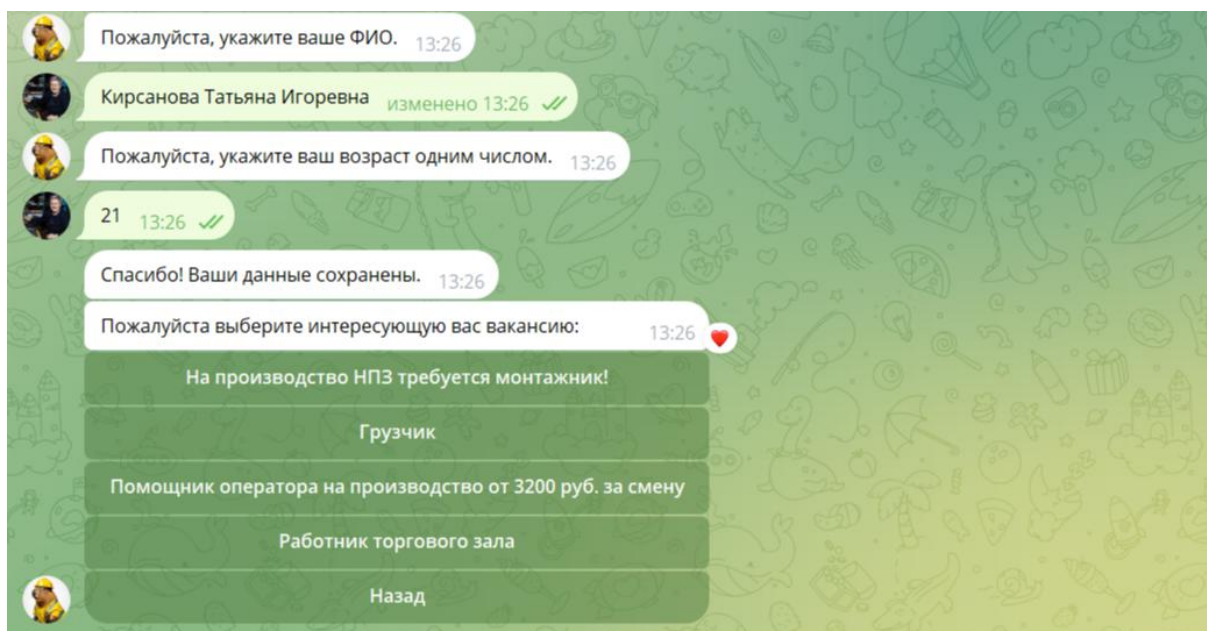


Рисунок 1.3 – Обзор аналога «findworkerBot»

Для наглядного сравнения существующих решений и определения их соответствия требованиям разрабатываемого продукта проведем анализ ключевых характеристик каждого из аналогов. Это позволит выявить сильные и слабые стороны существующих решений, а также определить уникальные преимущества разрабатываемого Telegram-бота РСО. Рассмотрим основные функциональные и технические параметры каждого из рассматриваемых продуктов.

Таблица 1.1 – Сравнение функционала систем

Параметры	Приложения			
	Телеграмм бот РСО	Сайт РСО	Fineworkerbot в телеграмм	Группа РСО в ВК
Отдельное место в телефоне для приложения	+	-	+	+
Возможность просмотра информации о направлениях	+	+	-	+
Возможность просмотра новостей и мероприятий	+	+	-	+
Дизайн	+	+	-	—
Кроссплатформенность	+	+	+	+
Автономность	+	+	+	+
Удобный и отзывчивый интерфейс	+	—	-	—

Исходя из таблицы 1.1 можно заметить, что наиболее полно включает в себя характеристики эталонного программного продукта для кандидатов РСО. Так же можно заметить, что наибольшим удобством в использовании обладает именно данное приложение.

Существующие аналоги (сайт РСО, группа РСО в ВКонтакте, бот @findworkerBot) предоставляют ограниченный функционал для работы со

студенческими отрядами. Разработка Telegram-бота для РСО представляется более эффективным решением, так как обладает рядом существенных преимуществ.

Использовать Telegram-бота на самом деле проще для людей, потому что не нужно скачивать дополнительное приложение или регистрироваться на сайте. Все происходит в привычном мессенджере, где у пользователей уже есть аккаунты.

Одно из главных преимуществ – это хороший уровень безопасности данных, благодаря протоколу MTProto. Это особенно важно, когда речь идет о личной информации участников и подаче заявок в группы.

Telegram-бот позволяет быстро общаться с пользователями, что очень нужно абитуриентам и студентам, которым срочно нужна информация о направлениях, условиях поступления и мероприятиях. В отличие от соцсетей, где нет гарантии, что ответят быстро, бот отвечает на запросы моментально.

Экономическая эффективность также играет важную роль: разработка и поддержка Telegram-бота обходится значительно дешевле создания и обслуживания веб-сайта или мобильного приложения. При этом бот остается кроссплатформенным решением, доступным как с мобильных устройств, так и с ПК.

Таким образом, Telegram-бот РСО представляет собой более современное, удобное и эффективное решение для организации взаимодействия со студентами, автоматизации процессов подачи заявок и информирования участников движения, чем существующие аналоги.

1.5 Анализ архитектуры и структура бота

Успех реализации проекта Telegram-бота для студенческих отрядов во многом зависит от грамотного выбора технологических решений. Правильный подбор инструментов разработки позволяет не только ускорить процесс создания

					ПГУ1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		17

системы, но и гарантировать её надёжность, безопасность и эффективность в долгосрочной перспективе.

От выбора стека технологий зависит не только качество конечного продукта, но и такие важные аспекты как масштабируемость системы, стоимость поддержки, а также возможность дальнейшего развития функционала. При разработке бота для РСО необходимо учитывать специфику проекта и подбирать инструменты, наилучшим образом соответствующие поставленным задачам.

Современные технологии предоставляют широкий спектр возможностей для реализации подобных проектов, позволяя выбрать оптимальное сочетание инструментов с учётом конкретных требований и особенностей системы. Подробную диаграмму прецедентов можно изучить на рисунке 1.4.

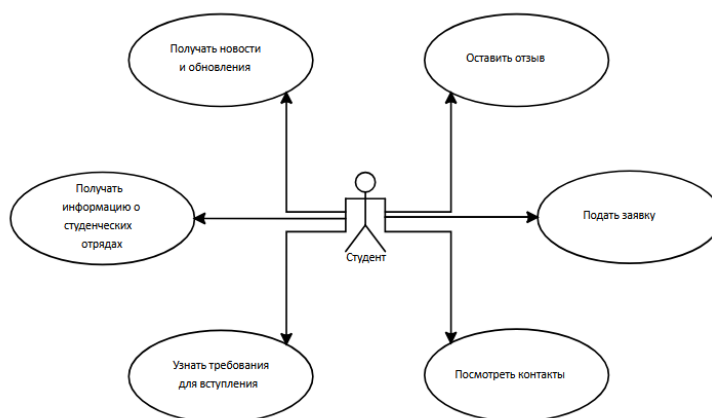


Рисунок 1.4 – UML диаграмма прецедентов интерфейса пользователя

Кроме того, в состав клиентского приложения входят интерфейс администратора. Администраторский интерфейс обеспечивает возможность управления данными всех пользователей системы, включая управление пользователями, доступом к материалам, информации о транзакциях, а также просмотром заявок. Подробнее диаграммы прецедентов этих интерфейсов можно изучить на рисунке 1.5.

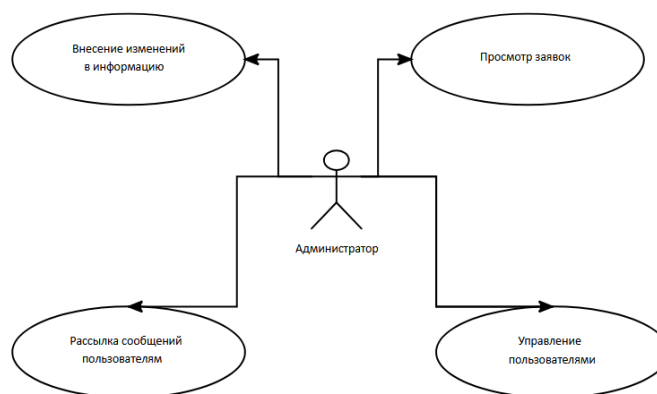


Рисунок 1.5 – UML диаграмма прецедентов интерфейса администратора

Для четкого понимания взаимодействия компонентов бота, необходимо создать UML-диаграмму последовательности. На рисунке 1.6 показана диаграмма последовательности.

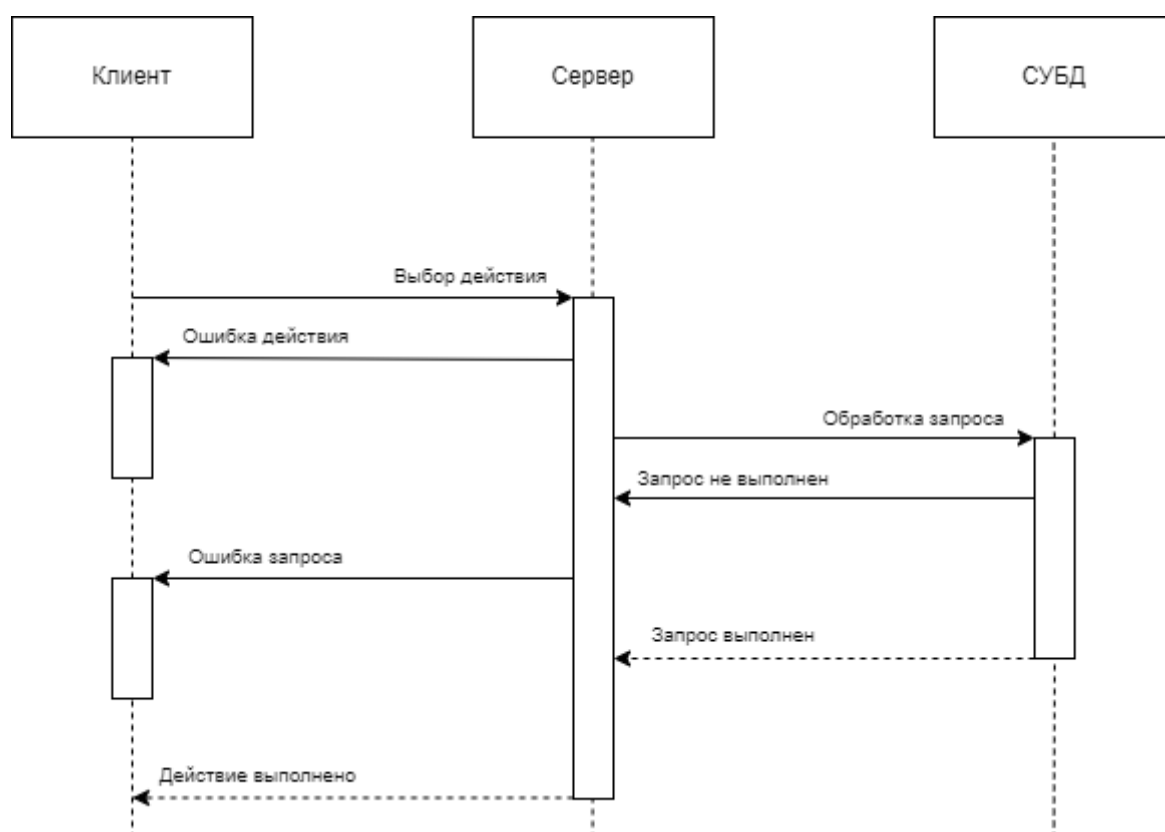


Рисунок 1.6 – UML-диаграмма последовательности

1.6 Постановка задачи

Нужно сделать Telegram-бота для упрощения работы со студенческими отрядами. Он поможет улучшить процесс регистрации новых участников и сделает взаимодействие с абитуриентами более простым. Бот должен включать функции для регистрации пользователей, подачи заявок, получения информации о направлениях работы и возможностях трудоустройства, а также для обратной связи и уведомлений о мероприятиях.

Основная цель бота - сделать общение между отрядами и абитуриентами проще, автоматизируя рутинные задачи и предоставляя доступ к нужной информации. Интерфейс должен быть удобным, а данные - надежно храниться с учетом безопасности.

Важно, чтобы бот можно было легко обновлять - добавлять новые функции по мере увеличения пользователей. Сервис должен быть недорогим в обслуживании, чтобы затраты на его запуск и поддержку были минимальными. Внедрение такого решения поможет сделать студенческие отряды более привлекательными и сильнее укрепит их роль в студенческом самоуправлении.

1.7 Выбор инструментов разработки

Выбор правильных инструментов для разработки – это важный шаг при создании информационной системы для студенческих отрядов. Это помогает сделать процесс проще, сэкономить время и деньги, а также повысить качество и надежность продукта. Если выбрать не те инструменты, можно столкнуться с проблемами во время работы и при использовании конечного продукта. Поэтому важно правильно подойти к выбору инструментов разработки, учитывая конкретные требования и характеристики проекта.

При создании Telegram-бота для студенческих отрядов можно использовать различные инструменты и технологии в зависимости от требований и целей проекта.

					ПГУ1.090301.39.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

1.7.1. Платформа для реализации системы

Выбор технологической платформы играет решающую роль в успехе проекта информационной системы студенческих отрядов. После тщательного анализа различных решений было принято решение реализовать проект на базе Telegram.

Telegram зарекомендовал себя как один из наиболее востребованных мессенджеров, что обеспечивает широкую аудиторию потенциальных пользователей и гарантирует удобство взаимодействия с системой. Telegram-боты обладают существенными преимуществами: молниеносная скорость работы, отлаженная система безопасности, высокая надежность функционирования, а также широкие возможности для интеграции с внешними сервисами [3].

Разработчикам платформа предлагает привлекательные условия: бесплатный доступ к инструментам создания ботов, открытый API, гибкие настройки и широкие возможности для масштабирования функционала. Встроенная система групповых чатов позволяет организовывать эффективное взаимодействие между участниками системы и предоставлять доступ к различным функциям одновременно нескольким пользователям.

Именно совокупность этих факторов определила выбор Telegram в качестве технологической основы для создания информационной системы студенческих отрядов, обеспечивая оптимальное сочетание функциональности, производительности и удобства использования.

1.7.2. Выбор языка программирования

При разработке информационной системы студенческих отрядов первостепенное значение имеет правильный выбор языка программирования, который определит эффективность всего проекта. Если говорить о языках для создания Telegram-ботов, то есть несколько популярных вариантов:

– Python - современный язык с простым синтаксисом, который легко использовать. Его гибкость делает его хорошим выбором для ботов;

– JavaScript - тоже популярный язык, особенно в веб-разработке. У него много библиотек;

– PHP - в основном используется на сервере, тоже хорошо подходит для веб-разработки;

– Java - более строгий язык, который работает быстро, но требует больше кода.

Python часто выбирают для разработки ботов по нескольким причинам:

– легкий в освоении синтаксис помогает быстро писать и поддерживать код, что очень важно при работе в команде;

– библиотека python-telegram-bot делает интеграцию с API Telegram проще;

– язык справляется с большими объемами данных и сложными вычислениями;

– у Python активное сообщество, которое создает новые инструменты и библиотеки, поэтому всегда есть поддержка.

Python имеет все нужные инструменты для создания функционального Telegram-бота, включая работу с API, обработку данных и пользовательский интерфейс. Его простота и эффективность делают его идеальным выбором для реализации информационной системы студенческих отрядов, требующей быстрой разработки и масштабируемости [5].

1.7.3. Выбор СУБД

В минимальной конфигурации, ориентированной на начальный этап эксплуатации Telegram-бота, предназначенного для помощи при вступлении в студенческие отряды, допустимо использование базы данных SQLite для хранения анкетных данных пользователей. SQLite, будучи встраиваемой СУБД, характеризуется низким потреблением ресурсов и не требует развертывания отдельного серверного процесса, что позволяет значительно снизить требования к аппаратному обеспечению. В связи с этим для стабильной работы бота на начальном этапе эксплуатации достаточно VPS со следующими характеристиками: оперативная память (ОЗУ) объемом 512 МБ

— 1 ГБ, одноядерный процессор (1 CPU) и дисковое пространство на твердотельном накопителе (SSD) объемом 10–20 ГБ. В качестве операционной системы рекомендуется использовать Linux (например, Ubuntu, Debian) как наиболее распространенную и стабильную платформу для развертывания серверных приложений.

Однако следует учитывать, что SQLite имеет ограничения по масштабируемости и параллельной обработке запросов. По мере роста числа пользователей, увеличения объёма данных и усложнения логики обработки анкет производительность SQLite может стать недостаточной. Кроме того, SQLite не поддерживает некоторые расширенные возможности, характерные для более мощных СУБД, такие как сложные SQL-запросы, механизмы управления транзакциями и репликация данных.

В связи с этим для обеспечения дальнейшего масштабирования проекта и сохранения приемлемой производительности при увеличении нагрузки рекомендуется заранее планировать возможность перехода на более мощные СУБД, такие как PostgreSQL или MySQL. Переход на PostgreSQL или MySQL потребует выделения дополнительных ресурсов, в частности увеличения объема оперативной памяти до 2 ГБ и дискового пространства до 20 ГБ, а также может потребовать более мощного процессора (хотя для большинства сценариев может быть достаточно одноядерного процессора). Важно понимать, что PostgreSQL и MySQL требуют больше внимания со стороны администраторов, чем SQLite. Это может потребовать лишних усилий. Выбор между PostgreSQL и MySQL зависит от того, что конкретно нужно для вашего проекта. PostgreSQL часто считается более мощной и универсальной базой данных с множеством функций и поддержкой сложных данных. В то же время, MySQL может быть проще в установке и настройке, что делает её хорошим вариантом для простых веб-приложений. Для Telegram-бота, который помогает с поступлением в студенческие отряды, можно начать с SQLite, так как требования к

оборудованию будут ниже. Но если проект начнет расти и потребует больше возможностей, стоит подумать о переходе на более серьезные базы данных, что потребует дополнительных ресурсов и настройки среды.

1.7.4. Выбранные технологии

В ходе проведенного технического анализа были приняты следующие ключевые решения по реализации информационной системы:

- Telegram определен как основная платформа для размещения чат-бота;
- язык программирования: Python;
- библиотека для работы с Telegram: telebot;
- среда разработки: VS Code;
- система управления базами данных: SQLite;
- библиотека для работы с БД: sqlite3;
- BeautifulSoup для парсинга веб-страниц;
- requests для работы с HTTP-запросами;
- re для работы с регулярными выражениями;
- string для работы с шаблонами.

Такой технологический стек обеспечивает надежную работу системы, возможность масштабирования и удобство поддержки. Выбор конкретных инструментов и библиотек обусловлен их эффективностью для решения поставленных задач и простотой интеграции между собой.

1.8 Реализация

1.8.1. Регистрация телеграм-бота

Первый шаг в создании информационной системы на базе Telegram — это регистрация собственного бота. Ниже представлена пошаговая инструкция по регистрации.

Для начала необходимо войти в систему Telegram. Откройте официальное приложение Telegram на мобильном устройстве или компьютере. Если у вас нет

аккаунта — создайте его. После этого войдите в существующий аккаунт. Внешний вид главного меню представлен на рисунке 1.7.

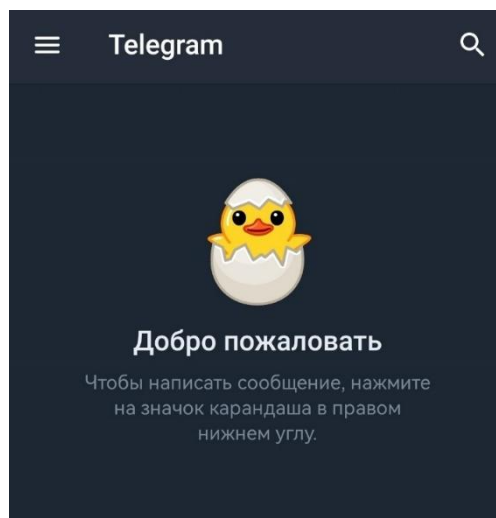


Рисунок 1.7 – Главное меню

Далее нужно найти в поиске Telegram официального бота “BotFather” — это специальный инструмент для создания новых ботов в Telegram. Процесс поиска показан на рисунке 1.8.



Рисунок 1.8 – Поиск BotFather

После того как вы открыли чат с BotFather, просто начните разговор с ботом. Для этого введите команду “/start”. Это запустит разговор с ботом, как видно на рисунке 1.9.



Рисунок 1.9 – Начало диалога с BotFather

Для создания любого бота необходимо выполнить команду `/newbot`, а затем присвоить боту имя, которое будет отображаться в поиске, и логин, который будет отображаться в ссылке. Как это сделала я, показано на рисунке 1.10.



Рисунок 1.10 – Выбор имени бота

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.39.001 ПЗ

Лист

26

После создания бота BotFather выдаст уникальный токен доступа к вашему боту. Важно: сохраните полученный токен в надежном месте. Этот токен необходим для аутентификации бота при работе с Telegram API. Пример получения токена показан на рисунке 1.11.

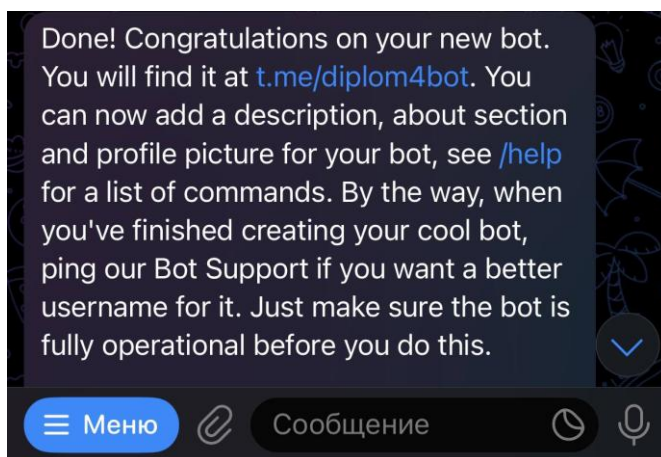


Рисунок 1.11 – Получение ссылки на бот

После завершения всех этих шагов телеграм-бот успешно зарегистрирован. Теперь можно использовать токен бота для настройки и интеграции бота в систему сбора данных о кандидатах РСО.

1.8.2. Работа с БД

В системе реализована база данных со следующей структурой:

- users - основная таблица для хранения информации о пользователях;
- id (уникальный идентификатор);
- name (имя пользователя);
- age (возраст);
- city (факультет и ВУЗ);
- napravlenie (выбранное направление отряда).

1.8.3. Описание доступных функций

Разработанная информационная система включает следующие ключевые функции:

- работа с анкетами. Система обеспечивает сбор и обработку анкет

кандидатов. Реализована валидация данных, включая проверку возраста и формата ввода. Вся информация автоматически сохраняется в базу данных. Предусмотрено формирование отчетов по поданным заявкам для дальнейшего анализа;

– управление пользователями: в системе реализован сбор данных о пользователях через интерактивную анкету в Telegram. Пользователи указывают имя, возраст, факультет/вуз и интересующий отряд. Данные безопасно сохраняются в базе SQLite. Функционал позволяет отслеживать взаимодействие пользователей с ботом через логирование в консоли;

– информационное взаимодействие: бот предоставляет актуальную информацию о студенческих отрядах ПГУ, включая описания и изображения. Доступны ответы на часто задаваемые вопросы о вступлении, а также контактная информация. Реализован парсинг новостей с сайта трудкрут.рф, что обеспечивает пользователей свежими обновлениями;

– интерактивные возможности: навигация организована через основное меню и подменю с помощью кнопок Telegram. Пользователи могут выбирать интересующие отряды, получать подробные данные о них и взаимодействовать с ботом через сообщения. Обратная связь реализована через автоматические ответы на запросы;

– административные функции: в текущей версии управление контентом ограничено изменением кода. Информация об отрядах обновляется через редактирование словаря `pravleniya_info`, а контакты — через словарь `contacts_info`. Вы можете отслеживать активность пользователей через логи в консоли. Бот также может собирать новости с внешнего сайта и отправлять их в виде HTML с картинками, если они есть. Обработка ошибок настроена, чтобы отправлять сообщения и фото, что помогает боту работать без сбоев.

Что касается преимуществ, то бот имеет удобный интерфейс в Telegram, автоматическую обработку данных и хранение в SQLite. Контент обновляется через парсинг новостей, а гибкое меню помогает пользователям легко находить

нужную информацию и выполнять задачи, как, например, заполнять анкеты. Все эти функции помогают эффективно управлять взаимодействием с кандидатами, организовывать работу студенческих отрядов и предоставлять актуальную информацию.

Чтобы начать использовать бота, его нужно найти. Сделать это можно тремя способами: перейти по ссылке, найти его в поиске (как показано на картинке 1.12) или воспользоваться QR-кодом, который находится на картинке 1.13.

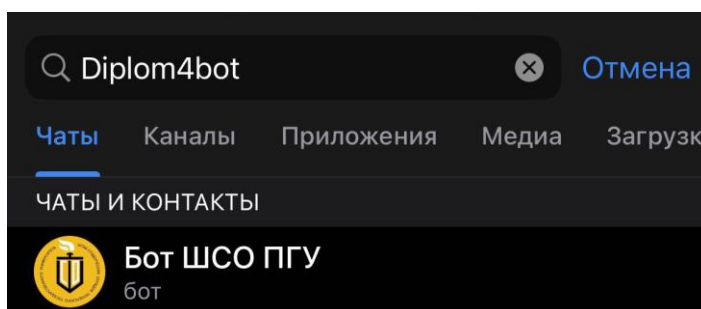


Рисунок 1.12 – Поиск бота в поиске



Рисунок 1.13 – Доступ к боту по QR

Когда пользователь откроет бота, он получит сообщение с описанием возможностей. Чтобы начать, нужно ознакомиться с этой информацией и нажать кнопку “Старт”. Графика этого процесса показана на изображении 1.14.

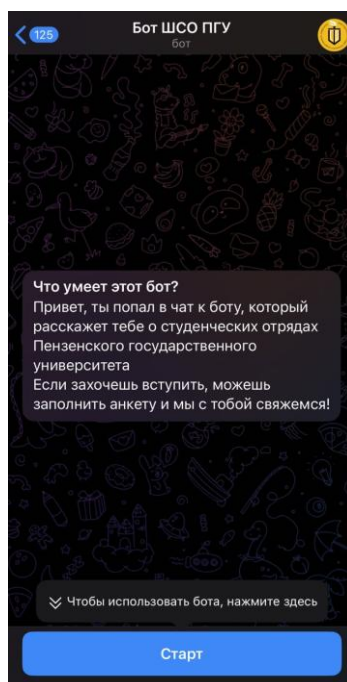


Рисунок 1.14 – Начало работы с ботом

После старта работы, бот предлагает выбрать, подать анкету или открыть основное меню, как показано на рисунке 1.15, тем самым показать пользователю выбор из других разделов. Пример работы этой кнопки, будет рассмотрен ниже.

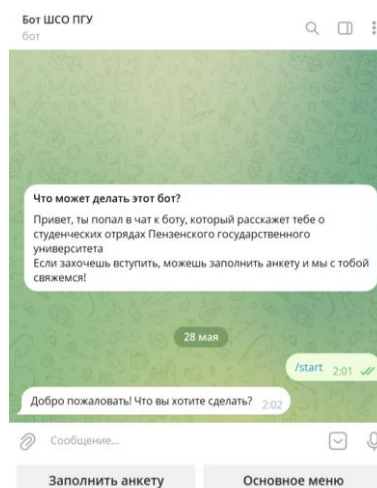


Рисунок 1.15 – Стартовое сообщение

После нажатия кнопки «Основное меню» открывается меню, в котором есть разделы: контакты, заполнить анкету, отряды ПГУ, новости, как вступить в студенческие отряды? Пример работы на рисунке 1.16.



Рисунок 1.16 – Основное меню.

Рассмотрим раздел «Как вступить в студенческие отряды?». В этом разделе пользователь получает информацию о требованиях к кандидатам, процессе подачи заявки и обучении. (Раздел «Как вступить в студенческие отряды» частично реализован. Код содержит функцию `как_vstupit_menu`, которая отображает кнопки «Требования к кандидатам», «Процесс подачи заявки», «Обучение и подготовка».) Можно увидеть на рисунке 1.17

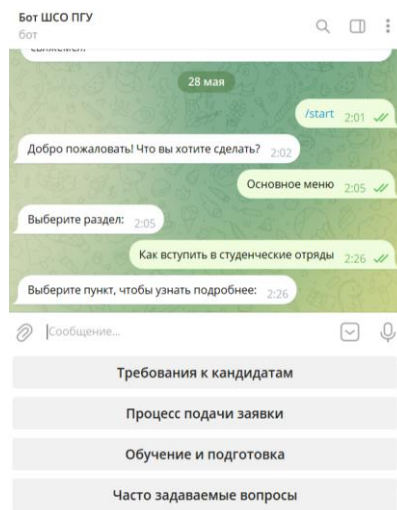


Рисунок 1.17 – Выбор категории.

Далее, в зависимости от выбранной кнопки, пользователь получает соответствующую информацию. (Отображение информации в зависимости от

выбора — реализовано в коде с помощью словаря `kak_vstupit_info`). Пример можно увидеть на рисунке 1.18.

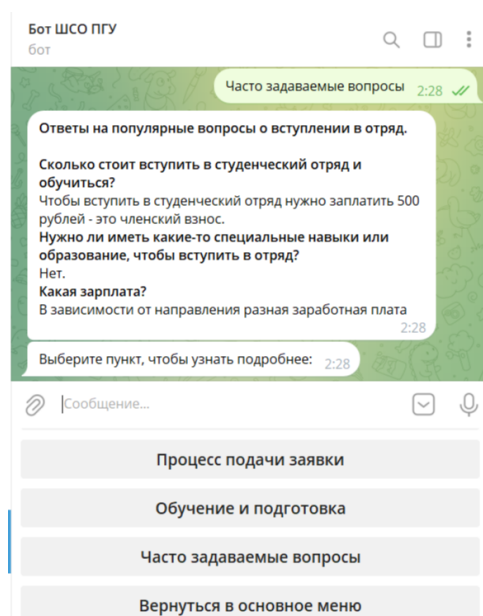


Рисунок 1.18 – Вывод информации

Далее рассмотрим все пункты по порядку. Рисунок 1.19 показывает, как выглядит раздел «Требования к кандидатам».

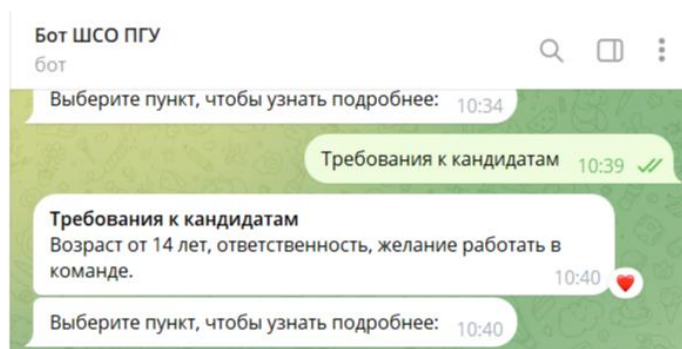


Рисунок 1.19 – Требования к кандидатам.

Рисунок 1.20 показывает, как выглядит раздел «Процесс подачи заявки».

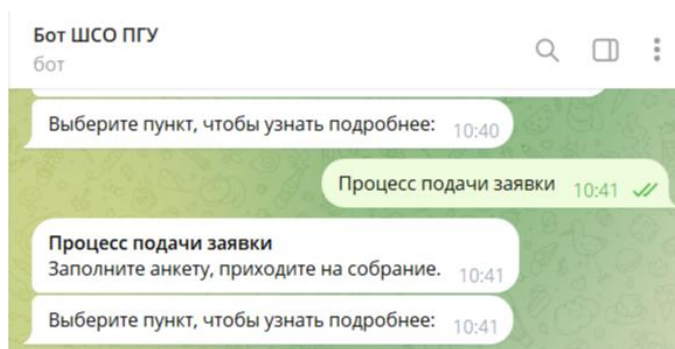


Рисунок 1.20 – Процесс подачи заявки.

Рисунок 1.21 показывает, как выглядит раздел «Обучение и подготовка».

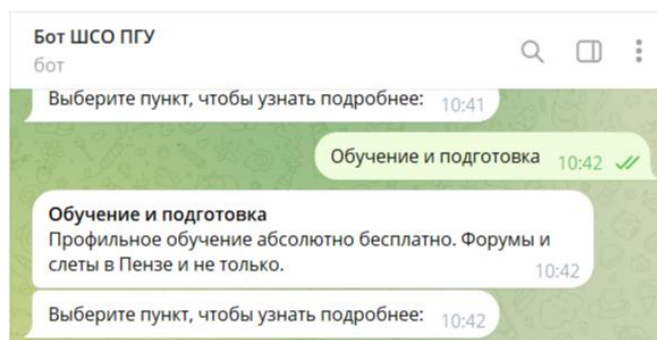


Рисунок 1.21 – Обучение и подготовка.

Рисунок 1.22 показывает, как выглядит раздел «Часто задаваемые вопросы».

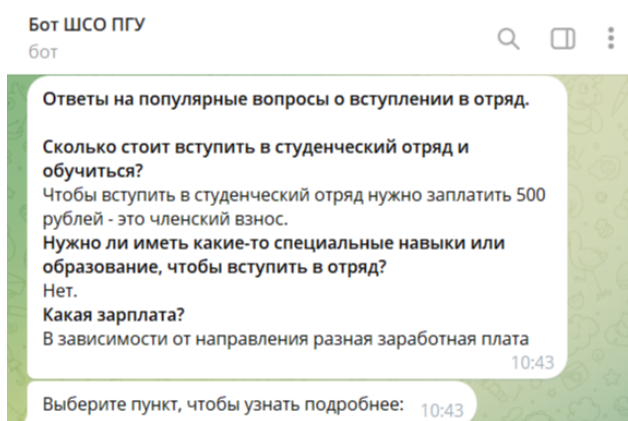


Рисунок 1.22 – Часто задаваемые вопросы.

Далее рассмотрим раздел «Заполнение анкеты». После выбора пункта «Заполнить анкету» в главном меню пользователь начинает процесс заполнения, автоматически сохраняется имя пользователя. (В коде реализована функция `start_anketa`, запускаемая при нажатии на кнопку «Заполнить анкету», которая инициализирует процесс заполнения). Бот последовательно запрашивает информацию о пользователе, начиная с имени (функция `start_anketa` задает первый вопрос). Далее пользователь вводит ответ на каждый вопрос. Пример показан на рисунке 1.23. (Реализованы функции `ask_age`, `ask_city`, `ask_napravlenie` для последовательного запроса данных у пользователя).

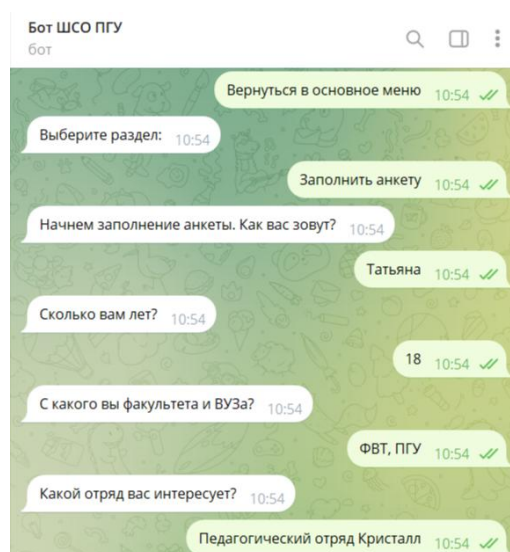


Рисунок 1.23 – Заполнение анкеты

После заполнения анкеты. Бот присылает всю информацию в виде одного текстового сообщения, а также благодарит за заполнение анкеты. Пример можно увидеть на рисунке 1.24

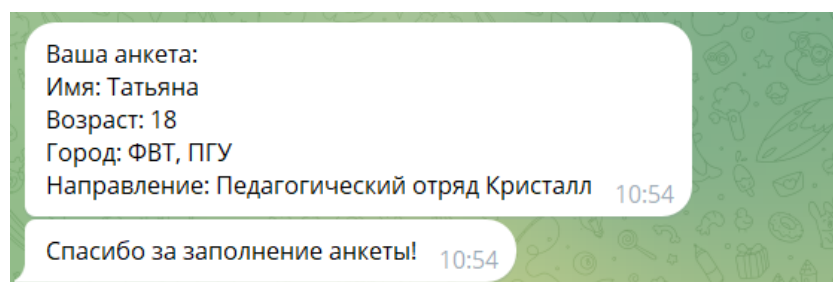


Рисунок 1.24 – Вывод информации

В функции `ask_age` есть проверка возраста. Так как вступить в студенческие отряды можно только с четырнадцати лет, при вводе числа меньше – приходит сообщение с ссылкой на основное меню. Пример показан на рисунке 1.25

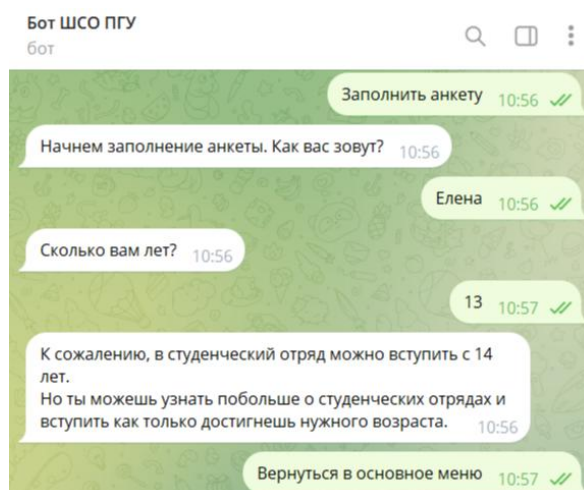


Рисунок 1.25 – Проверка возраста

Также при вводе возраста в любом формате кроме числового, бот предложит ввести возраст в виде числа. Работа показана на рисунке 1.26.

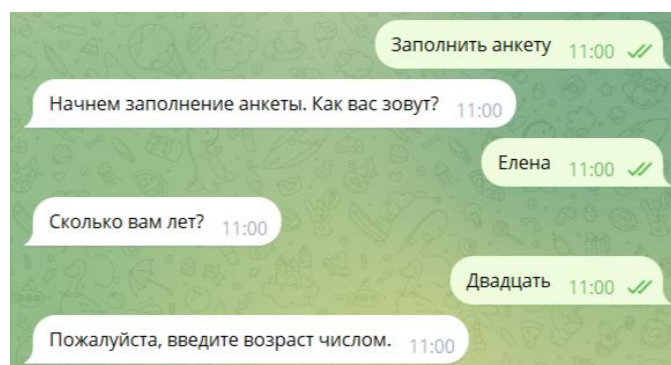


Рисунок 1.26 – Ввод возраста

После этого рассмотрим раздел «Новости» предоставляет пользователям актуальную информацию со стороннего ресурса – официальный сайт РСО трудкрут.рф. Для получения новостей используется функция `get_news()`, она

проверяет последнюю новость на сайте и отправляет ее пользователю по запросу. Её работа показана на рисунке 1.27.

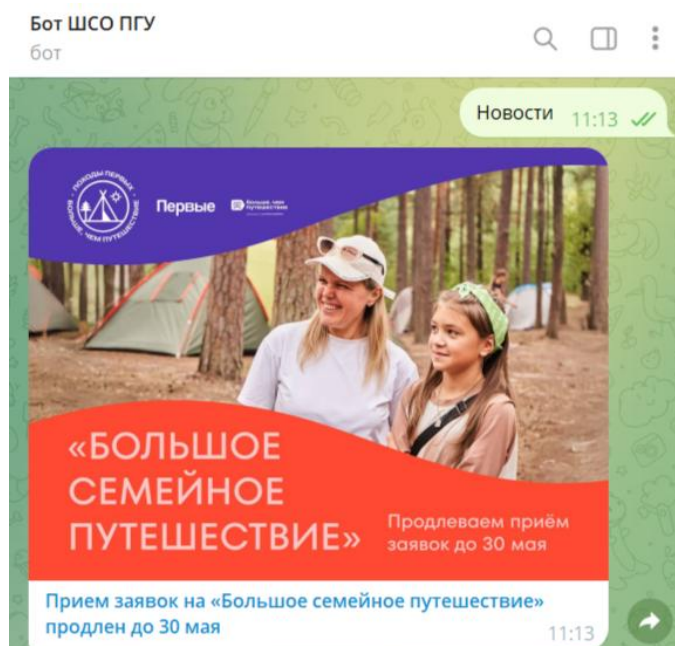


Рисунок 1.27 – Новости

Раздел «Отряды ПГУ» предоставляет пользователям информацию о студенческих отрядах. При выборе этого пункта меню бот отображает список доступных отрядов (реализована функция `pravleniya_menu`).

Функция `pravleniya_menu()` создает клавиатуру с названиями отрядов, перечисленными в словаре `pravleniya_info`. добавляет кнопку «Вернуться в главное меню», отправляет пользователю клавиатуру с предложением выбрать отряд. Это показано на рисунке 1.28.

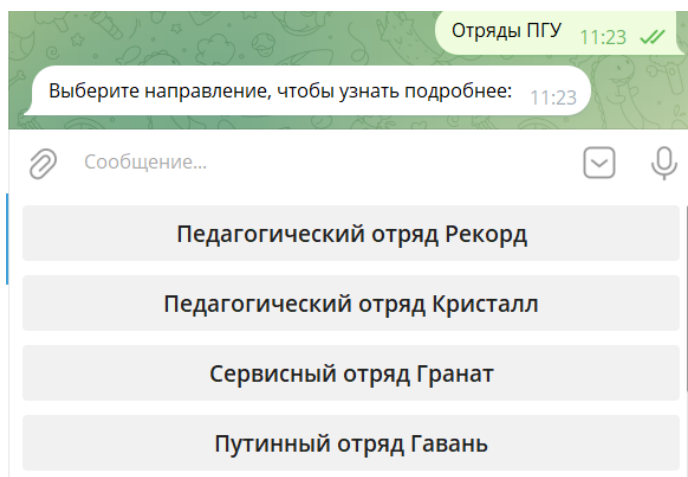


Рисунок 1.28 – Отряды ПГУ

При выборе конкретного отряда бот предоставляет подробную информацию о нем, включая описание, изображение и контактную информацию, пример можно увидеть на рисунке 1.29.

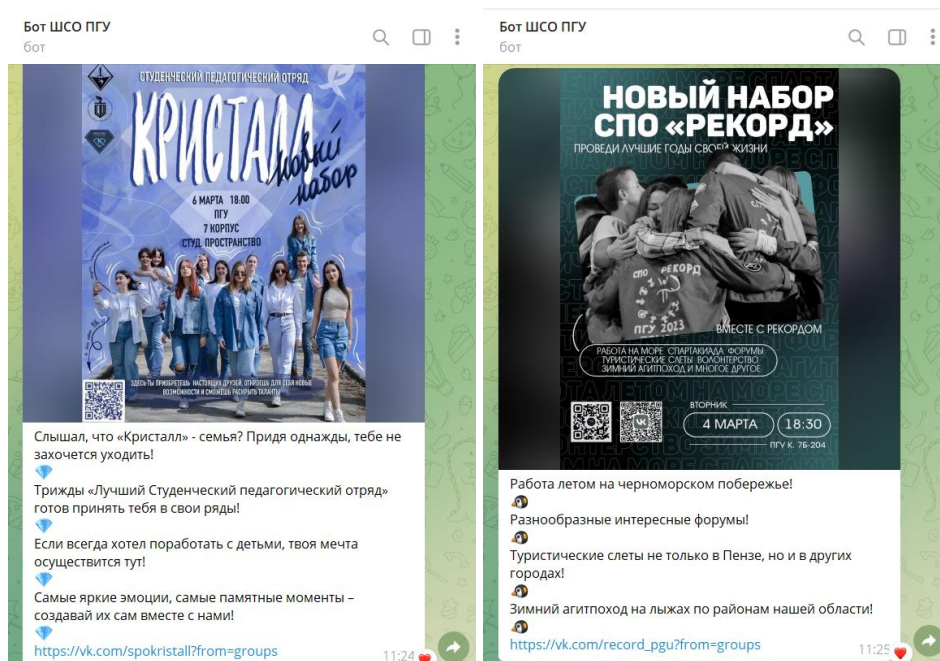


Рисунок 1.29 – Пример выбора конкретных отрядов

Рассмотрим раздел «Контакты». В этом разделе пользователь получает контактную информацию об автошколе (раздел «Контакты» реализован. Код содержит функцию contacts, отображающую контактную информацию, взятую из словаря contacts_info, и отображающую ее с помощью HTML-разметки. Работа продемонстрирована на рисунке 1.30.

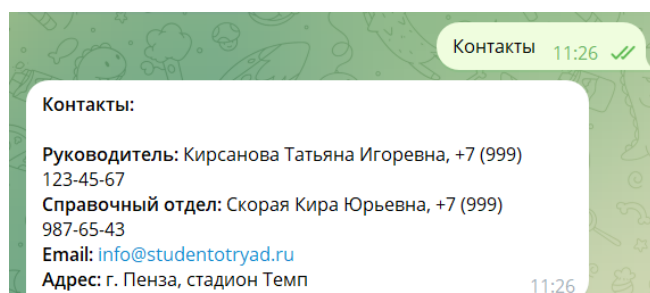


Рисунок 1.30 – Просмотр контактов

Также реализована функция ответа, на любое сообщение пользователя, не заложенное в код. Пример можно увидеть на рисунке 1.31.

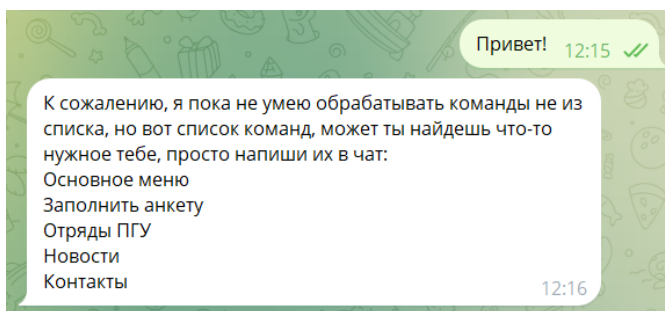


Рисунок 1.31 – Обработка сообщений пользователя

В коде присутствует функция, благодаря которой администратор может рассылать всем пользователям сообщения, достаточно вставить /send перед сообщением. ID администратора учтено в коде. Функционал продемонстрирован на рисунке 1.32.

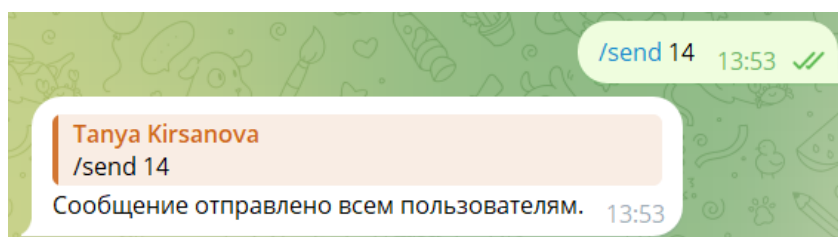


Рисунок 1.32 – Отправка сообщения пользователю

Если пользователь успешно заполнил анкету, данные автоматически высылаются администратору с заранее сохраненным ID, для дальнейшей коммуникации. Пример продемонстрирован на рисунке 1.33.

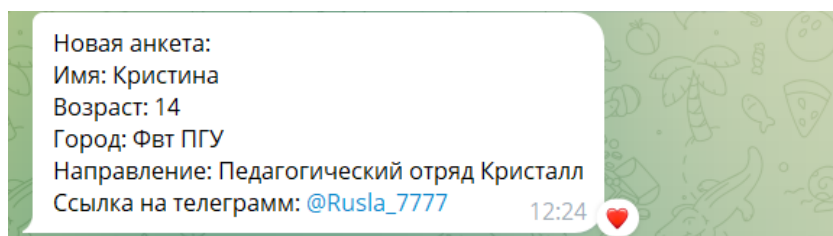


Рисунок 1.33 – Уведомление о новой заявке

Посмотрим как анкета нашего пользователя выглядит в СУБД. Пример показан на рисунке 1.34.

Rusla_7777	Кристина	14	ФВТ ПГУ	Педагогический отряд Кристалл
etkopatel	Татьяна	16	ФВТ ПГУ	Педагогический отряд Кристалл
lena228	Елена	14	ФВТ ПГУ	Педагогический отряд Кристалл

Рисунок 1.34 – Анкеты в СУБД

1.8.4. Описание программы

Диаграмма развертывания, приведенная на рисунке 1.35, используется для визуализации физического размещения компонентов системы и их взаимодействия на уровне аппаратного и программного обеспечения. Она позволяет представить архитектуру системы, указывая, на каких устройствах или серверах расположены компоненты приложения и как они связаны друг с другом.



Рисунок 1.35 - UML-диаграмма развертывания

Этот код реализует Telegram-бот, предназначенный для предоставления информации о студенческих отрядах Пензенского государственного университета (ПГУ), заполнения анкеты для вступления, получения последних новостей с сайта и связи с организаторами. Бот использует библиотеку telebot для взаимодействия с Telegram API, sqlite3 для хранения данных пользователей и requests с BeautifulSoup для веб-скрейпинга новостей.

Бот использует несколько библиотек Python:

```
import telebot
from telebot import types
```

```
import sqlite3
import requests
from bs4 import BeautifulSoup
import re
from string import Template
```

telebot и types: из библиотеки pyTelegramBotAPI. telebot создает бота, а types отвечает за интерфейс, включая клавиатуры (ReplyKeyboardMarkup).

sqlite3 — это встроенная библиотека Python для работы с базой данных SQLite, которая используется для хранения информации анкеты.

requests позволяет отправлять HTTP-запросы, чтобы получать новости с сайта трудкрут.рф.

BeautifulSoup — это инструмент для парсинга HTML, который нужно для извлечения новостей.

re — модуль для поиска ссылок на новости с помощью регулярных выражений.

Бот настраивается с помощью токена и использует глобальные переменные для управления действиями:

```
BOT_TOKEN = '7110963982:AAG0jKvMOjPeTmaG6gV5lNqfF26383bi8OA'
bot = telebot.TeleBot(BOT_TOKEN)
anketa = {}
CURRENT_QUESTION = None
USER_ID = None
```

BOT_TOKEN: уникальный токен от Telegram для аутентификации бота.

bot: экземпляр класса TeleBot, через который выполняются все действия бота (отправка сообщений, обработка команд и т.д.).

anketa: словарь для временного хранения данных анкеты. Ключ — ID чата, значение — словарь с полями (Имя, Возраст, Город, Направление).

CURRENT_QUESTION: отслеживает текущий вопрос анкеты (например, "Имя" или "Возраст"), изначально None.

USER_ID: хранит ID чата пользователя, заполняющего анкету, для предотвращения путаницы при одновременной работе нескольких пользователей.

Бот подключается к базе данных SQLite для хранения данных анкет:

```
try:
    conn = sqlite3.connect('database.db',
check_same_thread=False) # Создаем соединение с базой данных
    cursor = conn.cursor()
    # Создаем таблицу, если она еще не существует
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY,
            name TEXT,
            age INTEGER,
            city TEXT,
            napravlenie TEXT
        )
    """)
    conn.commit()
    print("Подключение к базе данных успешно установлено")
except Exception as e:
    print(f"Ошибка при подключении к базе данных: {e}")
```

Подключение: создаётся соединение с файлом database.db. Параметр check_same_thread=False позволяет использовать соединение в разных потоках, что важно для многопользовательского бота.

Создание таблицы: создаётся таблица users, если она еще не существует, с полями:

id: автоинкрементный ключ.

name: имя пользователя (текст).

age: возраст (целое число).

city: факультет и вуз (текст).

napravlenie: выбранный отряд (текст).

Обработка ошибок: логирует успешное подключение или ошибки в консоль.

Бот использует словари для хранения статической информации об отрядах, процессе вступления и контактах:

```
napravleniya_info = {
    "Педагогический отряд Рекорд": {
        "description": "Работа летом на черноморском побережье!
\n👤 \nРазнообразные интересные форумы!\n👤 \nТуристические слеты
не только в Пензе, но и в других городах!\n👤 \nЗимний агитпоход на
лыжах по районам нашей области!
\n👤\nhttps://vk.com/record_pgu?from=groups ",
        "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Рекорд.jpg"
    },
    "Педагогический отряд Кристалл": {
        "description": "Слышал, что «Кристалл» – семья? Придя
однажды, тебе не захочется уходить!\n💎\nТрижды «Лучший
Студенческий педагогический отряд» готов принять тебя в свои
ряды!\n💎\nЕсли всегда хотел поработать с детьми, твоя мечта
осуществится тут!\n💎\nСамые яркие эмоции, самые памятные моменты –
создавай их сам вместе с
нами!\n💎\nhttps://vk.com/spokristall?from=groups",
        "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Кристалл.jpg"
    },
    # ... (аналогичные записи для Гранат, Гавань, Династия)
}

kak_vstupit_info = {
    "Требования к кандидатам": "<b>Требования к
кандидатам</b>\nВозраст от 14 лет, ответственность, желание работать
в команде.",
```

"Процесс подачи заявки": "Процесс подачи заявки\nЗаполните анкету, приходите на собрание.",

"Обучение и подготовка": "Обучение и подготовка\nПрофильное обучение абсолютно бесплатно. Форумы и слеты в Пензе и не только.",

"Часто задаваемые вопросы": "Ответы на популярные вопросы о вступлении в отряд.\n\nСколько стоит вступить в студенческий отряд и обучиться?\nЧтобы вступить в студенческий отряд нужно заплатить 500 рублей - это членский взнос.\nНужно ли иметь какие-то специальные навыки или образование, чтобы вступить в отряд?\nНет.\nКакая зарплата? \nВ зависимости от направления разная заработная плата"

}

contacts_info = {

"Руководитель": "Кирсанова Татьяна Игоревна, +7 (999) 123-45-67",

"Справочный отдел": "Скорая Кира Юрьевна, +7 (999) 987-65-43",

"Email": "info@studentotryad.ru",

"Адрес": "г. Пенза, стадион Темп"

}

pravleniya_info: содержит информацию о пяти студенческих отрядах, включая description (описание с эмодзи и ссылкой на VK) и image_path (путь к изображению на локальном диске).

как_vstupit_info: предоставляет данные о вступлении: требования, процесс подачи заявки, обучение и FAQ, с использованием HTML-разметки для форматирования в Telegram.

contacts_info: хранит контактные данные: руководитель, справочный отдел, email и адрес.

Эти словари содержат статическую информацию, отправляемую пользователям по запросу.

Бот обеспечивает надежную отправку сообщений:

```
def send_message_safe(chat_id, text, reply_markup=None,
parse_mode=None):
    """Отправляет сообщение с обработкой возможных ошибок."""
    try:
        bot.send_message(chat_id, text,
reply_markup=reply_markup, parse_mode=parse_mode)
    except Exception as e:
        print(f"Ошибка при отправке сообщения: {e}")
```

Назначение: обертка для bot.send_message, обрабатывающая ошибки (например, проблемы с сетью или форматом).

Параметры:

- chat_id: ID чата для отправки сообщения.
- text: Текст сообщения.
- reply_markup: Необязательная клавиатура.
- parse_mode: Режим форматирования (например, HTML).

Обработка ошибок: логирует исключения, предотвращая остановку бота.

Эта функция используется повсеместно для надежной отправки сообщений. Бот начинает взаимодействие с командой /start:

```
@bot.message_handler(commands=['start'])
def start(message):
    print(f"Функция start вызвана пользователем
{message.from_user.id}, ID чата: {message.chat.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    item1 = types.KeyboardButton("Заполнить анкету")
    item2 = types.KeyboardButton("Основное меню")
    markup.add(item1, item2)
    bot.send_message(message.chat.id, "Добро пожаловать! Что вы
хотите сделать?", reply_markup=markup)
```

Триггер: команда /start.

Действие: логирует ID пользователя и чата. Создает клавиатуру с кнопками "Заполнить анкету" и "Основное меню". Отправляет приветственное сообщение с клавиатурой.

Назначение: запускает взаимодействие, предлагая выбор между анкетой и основным меню.

Основное меню предоставляет доступ к основным функциям бота:

```
@bot.message_handler(func=lambda message: message.text ==
"Основное меню" or message.text == "Вернуться в основное меню")
def main_menu(message):
    print(f"Функция main_menu вызвана пользователем
{message.from_user.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    item1 = types.KeyboardButton("Отряды ПГУ")
    item2 = types.KeyboardButton("Новости")
    item3 = types.KeyboardButton("Как вступить в студенческие
отряды")
    item4 = types.KeyboardButton("Контакты")
    item5 = types.KeyboardButton("Заполнить анкету")
    markup.add(item1, item2, item3, item4, item5)
    bot.send_message(message.chat.id, "Выберите раздел:",
reply_markup=markup)
```

Триггер: сообщения "Основное меню" или "Вернуться в основное меню".

Действие: логирует ID пользователя. Создает клавиатуру с пятью кнопками: "Отряды ПГУ", "Новости", "Как вступить в студенческие отряды", "Контакты", "Заполнить анкету". Запрашивает выбор раздела.

Назначение: служит центральным узлом для навигации по функциям бота.

Пользователи могут изучить информацию об отрядах:

```
@bot.message_handler(func=lambda message: message.text ==
"Отряды ПГУ")
def napravleniya_menu(message):
```

```

        print(f"Функция napravleniya_menu вызвана пользователем
{message.from_user.id}")

        markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)

        for napravlenie in napravleniya_info.keys():
            markup.add(types.KeyboardButton(npravlenie))
        markup.add(types.KeyboardButton("Вернуться в основное
меню"))

        bot.send_message(message.chat.id, "Выберите направление,
чтобы узнать подробнее:", reply_markup=markup)

```

Триггер: сообщение "Отряды ПГУ".

Действие: логирует ID пользователя. Создает клавиатуру с названиями отрядов и кнопкой "Вернуться в основное меню". Запрашивает выбор отряда.

Назначение: позволяет выбрать отряд для получения информации.

При выборе отряда бот предоставляет подробности:

```

@bot.message_handler(func=lambda message: message.text in
napravleniya_info and CURRENT_QUESTION is None)
def napravlenie_info(message):
    print(f"Функция napravlenie_info вызвана пользователем
{message.from_user.id}")

    napravlenie = napravleniya_info[message.text]
    description = napravlenie["description"]
    image_path = napravlenie["image_path"]

    try:
        with open(image_path, "rb") as photo:
            bot.send_photo(message.chat.id, photo.read(),
caption=description, reply_markup=types.ReplyKeyboardRemove())
        except FileNotFoundError:
            bot.send_message(message.chat.id, f"Файл {image_path}
не найден.")

            main_menu(message)
            return
        except Exception as e:

```

```

        print(f"Ошибка при отправке фото: {e}")
        bot.send_message(message.chat.id, "Произошла ошибка при
отправке фото.")
        main_menu(message)
        return

```

```
main_menu(message)
```

Триггер: сообщение, совпадающее с названием отряда, при отсутствии активной анкеты.

Действие: логирует ID пользователя. Извлекает описание и путь к изображению отряда. Пытается отправить фото с описанием. Если файл не найден, отправляет сообщение об ошибке. При других ошибках логирует их и уведомляет пользователя. Удаляет клавиатуру и возвращает в основное меню.

Назначение: показывает информацию об отряде с изображением.

Бот предоставляет информацию о вступлении в отряды:

```

@bot.message_handler(func=lambda message: message.text == "Как
вступить в студенческие отряды")
def kak_vstupit_menu(message):
    print(f"Функция kak_vstupit_menu вызвана пользователем
{message.from_user.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    for item in kak_vstupit_info.keys():
        markup.add(types.KeyboardButton(item))
    markup.add(types.KeyboardButton("Вернуться в основное
меню"))
    send_message_safe(message.chat.id, "Выберите пункт, чтобы
узнать подробнее:", reply_markup=markup)

```

Триггер: сообщение "Как вступить в студенческие отряды".

Действие: логирует ID пользователя. Создает клавиатуру с пунктами из kak_vstupit_info и кнопкой "Вернуться в основное меню". Запрашивает выбор пункта.

Назначение: отображает подменю с информацией о вступлении.

Обработывает запросы конкретной информации о вступлении:

```
@bot.message_handler(func=lambda message: message.text in
kak_vstupit_info)
def kak_vstupit_info_handler(message):
    print(f"Функция kak_vstupit_info_handler вызвана
пользователем {message.from_user.id}")
    send_message_safe(message.chat.id,
kak_vstupit_info[message.text],
reply_markup=types.ReplyKeyboardRemove(), parse_mode="HTML")
    kak_vstupit_menu(message)
```

Триггер: сообщение, совпадающее с пунктом из kak_vstupit_info.

Действие: логирует ID пользователя. Отправляет выбранную информацию с HTML-форматированием. Удаляет клавиатуру и возвращает в меню вступления.

Назначение: показывает подробности о вступлении.

Отображает контактную информацию:

```
@bot.message_handler(func=lambda message: message.text ==
"Контакты")
def contacts(message):
    print(f"Функция contacts вызвана пользователем
{message.from_user.id}")
    contacts_text = "<b>Контакты:</b>\n\n"
    for key, value in contacts_info.items():
        contacts_text += f"<b>{key}</b> {value}\n"
    bot.send_message(message.chat.id, contacts_text,
parse_mode='HTML')
    main_menu(message)
```

Триггер: сообщение "Контакты".

Действие: логирует ID пользователя. Форматирует контакты с использованием HTML. Отправляет текст и возвращает в основное меню.

Назначение: предоставляет контактную информацию.

Заполнение анкеты реализовано через последовательность обработчиков, собирающих данные пользователя (имя, возраст, факультет/вуз, отряд).

```
@bot.message_handler(func=lambda message: message.text ==
"Заполнить анкету")
def start_anketa(message):
    print(f"Функция start_anketa вызвана пользователем
{message.from_user.id}")
    global CURRENT_QUESTION, USER_ID
    anketa[message.chat.id] = {}
    USER_ID = message.chat.id
    CURRENT_QUESTION = "Имя"
    bot.send_message(message.chat.id, "Начнем заполнение
анкеты. Как вас зовут?")
```

Триггер: сообщение "Заполнить анкету".

Действие: логирует ID пользователя. Инициализирует пустой словарь для данных анкеты. Устанавливает USER_ID и CURRENT_QUESTION. Запрашивает имя.

Назначение: запускает процесс заполнения анкеты.

Запрос возраста

```
@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Имя" and message.chat.id == USER_ID)
def ask_age(message):
    print(f"Функция ask_age вызвана пользователем
{message.from_user.id}")
    anketa[message.chat.id]["Имя"] = message.text
    global CURRENT_QUESTION
    CURRENT_QUESTION = "Возраст"
    bot.send_message(message.chat.id, "Сколько вам лет?")
```

Триггер: сообщение, когда текущий вопрос — "Имя".

Действие: логирует ID пользователя. Сохраняет имя в anketa. Обновляет CURRENT_QUESTION на "Возраст". Запрашивает возраст.

Назначение: собирает имя и переходит к следующему вопросу.

Запрос факультета/вуза

```
@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Возраст" and message.chat.id == USER_ID)
def ask_city(message):
    print(f"Функция ask_city вызвана пользователем
{message.from_user.id}")
    try:
        age = int(message.text)
        if age < 14:
            bot.send_message(message.chat.id, "К сожалению, в
студенческий отряд можно вступить с 14 лет. \nНо ты можешь узнать
побольше о студенческих отрядах и вступить как только достигнешь
нужного возраста.")
            return
        main_menu(message) # Недостижимо из-за return
        anketa[message.chat.id]["Возраст"] = age
    except ValueError:
        bot.send_message(message.chat.id, "Пожалуйста, введите
возраст числом.")
        return
    global CURRENT_QUESTION
    CURRENT_QUESTION = "Город"
    bot.send_message(message.chat.id, "С какого вы факультета и
ВУЗа?")
```

Триггер: сообщение, когда текущий вопрос — "Возраст".

Действие: логирует ID пользователя. Проверяет возраст: если не число — запрашивает числовой ввод, если меньше 14 — прерывает процесс с сообщением. Сохраняет возраст и обновляет CURRENT_QUESTION на "Факультет". Запрашивает факультет и вуз.

Назначение: собирает и валидирует возраст.

					ПГУ1.090301.39.001 ПЗ	Лист
						50
Изм.	Лист	№ докум.	Подпись	Дата		

Запрос отряда

```
@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Город" and message.chat.id == USER_ID)
def ask_napravlenie(message):
    print(f"Функция ask_napravlenie вызвана пользователем
{message.from_user.id}")
    anketa[message.chat.id]["Город"] = message.text
    global CURRENT_QUESTION
    CURRENT_QUESTION = "Направление"
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    napravleniya = list(npravleniya_info.keys())
    for napravlenie in napravleniya:
        markup.add(types.KeyboardButton(npravlenie))
    bot.send_message(message.chat.id, "Какой отряд вас
интересует?", reply_markup=markup)
```

Триггер: сообщение, когда текущий вопрос — "Город".

Действие: логирует ID пользователя. Сохраняет факультет/вуз. Обновляет CURRENT_QUESTION на "Направление". Создает клавиатуру с отрядами и запрашивает выбор.

Назначение: собирает факультет/вуз и предлагает выбрать отряд.

Завершение анкеты

```
@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Направление" and message.chat.id == USER_ID)
def anketa_done(message):
    print(f"Функция anketa_done вызвана пользователем
{message.from_user.id}")
    napravleniya = list(npravleniya_info.keys())
    if message.text not in napravleniya:
        send_message_safe(message.chat.id, "Пожалуйста,
выберите отряд из списка.")
    return
```

```

anketa[message.chat.id]["Направление"] = message.text
global CURRENT_QUESTION, USER_ID
CURRENT_QUESTION = None
USER_ID = None

try:
    cursor.execute("""
        INSERT INTO users (name, age, city, направление)
        VALUES (?, ?, ?, ?)
        """, (anketa[message.chat.id]["Имя"],
            anketa[message.chat.id]["Возраст"],
            anketa[message.chat.id]["Город"],
            anketa[message.chat.id]["Направление"]))
    conn.commit()
    print(f"Данные пользователя {message.chat.id} успешно
сохранены в базе данных")
    print(f"Последняя вставленная строка:
{cursor.lastrowid}")
except Exception as e:
    print(f"Ошибка при сохранении данных в базу данных:
{e}")

anketa_text = "Ваша анкета:\n"
for key, value in anketa[message.chat.id].items():
    anketa_text += f"{key}: {value}\n"
send_message_safe(message.chat.id, anketa_text,
reply_markup=types.ReplyKeyboardRemove())
send_message_safe(message.chat.id, "Спасибо за заполнение
анкеты!")

```

#main_menu(message) # Закомментировано

Триггер: сообщение, когда текущий вопрос — "Направление".

Действие: логирует ID пользователя. Проверяет, что выбранный отряд в списке, иначе запрашивает корректный выбор. Сохраняет отряд, сбрасывает

глобальные переменные. Сохраняет данные в базу, логирует результат. Отправляет пользователю итог анкеты и благодарность.

Назначение: завершает анкету и сохраняет данные.

Бот извлекает и отправляет новости с сайта:

```
def get_news():
    """Получает последнюю новость с сайта трудкрыт.рф."""
    url = 'https://трудкрыт.рф/novosti.html'

    try:
        response = requests.get(url)
        response.raise_for_status()
        response.encoding = 'utf-8'

        soup = BeautifulSoup(response.text, 'html.parser')

        # Находим все элементы a, содержащие новости
        news_items = soup.find_all('a',
href=re.compile(r"/view-news\.html\?id=\d+"))

        if news_items:
            # Получаем только первую новость
            item = news_items[0]
            link = "https://трудкрыт.рф" + item['href']
            b_tag = item.find('b')
            text = b_tag.text.strip() if b_tag else "Заголовок
не найден"

            # Извлекаем URL картинки
            img_tag = item.find('img')
            img_url = "https://трудкрыт.рф" + img_tag['src'] if
img_tag else None

            return {'link': link, 'text': text, 'img_url':
img_url}
```

```

else:
    return "Новости не найдены"

except requests.exceptions.RequestException as e:
    return f"Ошибка при запросе к сайту: {e}"
except Exception as e:
    return f"Ошибка при парсинге страницы: {e}"

```

Назначение: извлекает последнюю новость с сайта трудкрут.рф/novosti.html.

Процесс: отправляет GET-запрос, устанавливает кодировку UTF-8. Парсит HTML, ищет ссылки на новости по шаблону /view-news.html?id=\d+. Извлекает ссылку, заголовок и URL картинки первой новости.

Обработка ошибок: возвращает сообщения об ошибках при проблемах с запросом или парсингом.

```

def send_news_with_image(chat_id, news_item):
    """Отправляет новость с картинкой в Telegram."""
    text = f"<b><a
href='{news_item['link']}'>{news_item['text']}'</a></b>"
    img_url = news_item['img_url']

    if img_url:
        try:
            bot.send_photo(chat_id, photo=img_url,
caption=text, parse_mode="HTML")
        except Exception as e:
            print(f"Ошибка при отправке фото: {e}")
            bot.send_message(chat_id, text + f"\n(Не удалось
загрузить изображение: {img_url})", parse_mode="HTML")
    else:
        bot.send_message(chat_id, text, parse_mode="HTML")

```

Назначение: отправляет новость, предпочтительно с картинкой.

Процесс: форматирует заголовок как кликабельную ссылку в HTML. Пытается отправить фото с подписью; при ошибке отправляет текст. Если картинки нет, отправляет только текст.

Обработка ошибок: логирует проблемы с отправкой фото.

```
@bot.message_handler(func=lambda message: message.text ==
"Новости")

def send_news_menu(message):
    """Отправляет последнюю новость с сайта трудкрыт.рф."""
    chat_id = message.chat.id
    news = get_news()

    if isinstance(news, str):
        # Если get_news вернула строку (ошибку), отправляем ее
        send_message_safe(chat_id, news)
    else:
        # Если get_news вернула словарь с новостью, отправляем
        ее с картинкой
        send_news_with_image(chat_id, news)

    main_menu(message)
```

Триггер: сообщение "Новости".

Действие: получает новость через get_news, если возвращена строка (ошибка), отправляет ее, если возвращен словарь, отправляет новость с картинкой. Возвращает в основное меню.

Назначение: доставляет пользователю последнюю новость.

Бот запускается в режиме опроса для обработки сообщений:

```
if __name__ == '__main__':
    try:
        bot.polling(none_stop=True)
    except Exception as e:
        print(f"Ошибка при запуске бота: {e}")

if __name__ == '__main__':
```

```

try:
    bot.polling(none_stop=True, timeout=60)
except Exception as e:
    print(f"Ошибка при запуске polling: {e}")

```

Назначение: запускает цикл опроса для обработки сообщений.

Параметры:

- none_stop=True: продолжает работу после ошибок.
- timeout=60 (во втором блоке): устанавливает тайм-аут 60 секунд.

2 Безопасность жизнедеятельности

2.1 Правила безопасности при использовании ПК

В данном разделе необходимо учесть риски, перечисленные в ГОСТ 12.0.003-2015 "Система стандартов безопасности труда. Электронно-вычислительная техника. Общие требования безопасности" [7]. Ниже приведен перечень рисков, которым следует придать особое значение:

– длительное взаимодействие с компьютерными мониторами без соблюдения режима труда и отдыха может привести к серьезным проблемам со зрением. Особенно опасно непрерывная работа без перерывов, что вызывает повышенную утомляемость глазных мышц и сухость конъюнктивы;

– не менее важным фактором является правильная организация рабочего пространства. Некорректное расположение монитора относительно глаз пользователя, наличие бликов на экране из-за неудачного освещения или недостаточная освещенность рабочего места создают дополнительную нагрузку на органы зрения;

– неправильная организация рабочего места часто становится причиной развития заболеваний опорно-двигательного аппарата. Некорректная осанка, неудобная поза, неправильное расположение конечностей при работе с клавиатурой и мышью могут привести к хроническим заболеваниям;

– отсутствие специализированной эргономичной мебели, такой как регулируемые кресла с подлокотниками и специальные подставки для рук, существенно повышает риск развития профессиональных заболеваний, связанных с длительным статическим напряжением мышц;

– использование устаревшего или неисправного компьютерного оборудования может создавать повышенные уровни электромагнитного излучения. Особую опасность представляет некорректная установка техники, когда устройства располагаются слишком близко друг к другу или к рабочему месту пользователя;

					ПГУ1.090301.39.001 ПЗ	Лист
						57
Изм.	Лист	№ докум.	Подпись	Дата		

– нарушение правил эксплуатации электрооборудования, включая использование поврежденной проводки и перегрузку электрических розеток, создает риск поражения электрическим током и возникновения пожара;

– размещение легковоспламеняющихся материалов вблизи компьютерной техники, а также использование несертифицированных электроприборов в непосредственной близости от ПК может привести к серьезным последствиям;

– слабые пароли и небезопасное использование учетных записей создают предпосылки для несанкционированного доступа к конфиденциальной информации. Существует риск утечки данных из-за недостаточных мер защиты или действий злоумышленников;

– работа в условиях повышенного шума, вызванного как внутренними компонентами компьютера (вентиляторами, кулерами), так и внешними источниками, может негативно влиять на концентрацию внимания и общее самочувствие пользователя.

Для минимизации всех перечисленных рисков необходимо строго соблюдать правила организации рабочего места, регулярно проводить перерывы в работе и использовать только исправное оборудование, соответствующее всем требованиям безопасности.

При использовании современного смартфона необходимо учитывать ряд факторов, способных оказать негативное влияние на пользователя и сохранность данных.

Рассмотрим основные виды рисков:

– аддиктивное поведение и психофизиологические риски проявляются через формирование зависимости от социальных платформ, игровых приложений и постоянного пребывания в сети. Это может привести к снижению социальной активности и ухудшению общего самочувствия;

– влияние на качество сна проявляется в нескольких аспектах. Воздействие синего спектра экрана устройства перед отходом ко сну может нарушать естественные биоритмы. Активное использование гаджета в ночное время

способно подавлять выработку мелатонина, что негативно сказывается на качестве отдыха;

– угрозы информационной безопасности возникают при потере или хищении устройства. В таких случаях существует риск несанкционированного доступа к содержимому и компрометации личных данных злоумышленниками;

– электромагнитное воздействие представляет собой потенциальную опасность при длительном контакте с устройством. Особенно это актуально в зонах с нестабильным сигналом сотовой связи, где уровень излучения может быть повышен;

– дорожная безопасность существенно нарушается при использовании телефона во время управления транспортным средством. Чтение, набор сообщений или ведение разговоров за рулем отвлекают внимание водителя и повышают риск дорожно-транспортных происшествий.

Для минимизации перечисленных рисков необходимо придерживаться комплексного подхода к использованию смартфона. Важно контролировать продолжительность взаимодействия с устройством, особенно в вечернее время. При работе с гаджетом следует поддерживать правильную осанку и эргономичное положение тела.

Необходимо использовать только сертифицированные аксессуары для зарядки и принимать меры по предотвращению перегрева устройства. Для защиты данных рекомендуется устанавливать надежные пароли и активировать экранную блокировку. В общественных местах следует проявлять особую осторожность и обеспечивать надежное хранение устройства.

Важным аспектом безопасности является регулярное создание резервных копий важной информации. Для защиты от вредоносного программного обеспечения необходимо устанавливать антивирусное ПО. Особое внимание следует уделять безопасности на дороге – полностью исключить использование телефона во время управления транспортным средством.

Следование данным рекомендациям позволит существенно снизить потенциальные риски и обеспечить безопасное взаимодействие с мобильным устройством.

					ПГУ1.090301.39.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения дипломной работы был спроектирован и разработан Telegram-бот, предназначенный для помощи абитуриентам и заинтересованным лицам в процессе вступления в студенческие отряды. Были достигнуты поставленные цели и задачи, а именно: проведен анализ целевой аудитории, разработана архитектура бота, спроектирован пользовательский интерфейс, сформирована информационная модель и реализованы основные функциональные модули.

Основные результаты исследования:

- проведен детальный анализ целевой аудитории: определены основные группы пользователей (абитуриенты, студенты, организаторы) и их потребности в информации и функциях, связанных со студенческими отрядами;

- разработана трехуровневая архитектура: архитектура бота построена на принципах модульности и разделения ответственности, что обеспечивает гибкость, масштабируемость и удобство сопровождения;

- разработан интуитивно понятный пользовательский интерфейс: интерфейс бота спроектирован с учетом потребностей целевой аудитории и обеспечивает удобную навигацию по функциям бота с помощью кнопок, команд и форматированного текста;

- сформирована информационная модель: определены основные сущности, используемые ботом (пользователь, отряд, новость, вопрос, анкета), и реализована база данных SQLite для хранения информации о пользователях и их анкетах;

- реализованы основные функциональные модули: разработаны модули подачи заявки, получения новостей, предоставления информации о отрядах. В частности, был создан функциональный модуль получения новостей, который автоматически собирает информацию с веб-сайта трудкрут.рф и предоставляет пользователям актуальные сведения о студенческом движении. Этот модуль

демонстрирует возможности использования Telegram-бота для автоматизации сбора и распространения информации.

Вот несколько практических рекомендаций на основе опыта разработки и тестирования бота:

- используйте возможности Telegram API для улучшения работы с пользователями. Это значит, что стоит добавить клавиатуры, делать текст форматированным и использовать изображения, чтобы интерфейс выглядел более удобно и привлекательно;

- применяйте модульную конструкцию в коде. Делите код на логические части, чтобы разработка, тестирование и поддержка шли легче. Используйте функции и классы, чтобы разрабатывать код организованно;

- обратите внимание на скорость работы. Бот должен быть быстрым. Оптимизируйте код и избегайте долгих операций, которые могут замедлить обработку сообщений. Подумайте о кэшировании и асинхронной обработке;

- заботьтесь о безопасности. Помните, что бот может обрабатывать личные данные, поэтому важно защищать их, используя шифрование и проверку введенных данных. Ограничьте доступ к базе данных;

- следите за надежностью. Бот должен работать без сбоев. Используйте обработку ошибок, ведите логи и мониторьте работу, чтобы быстро реагировать на проблемы;

- взаимодействуйте с пользователями. Их отзывы и идеи могут помочь улучшить бот. Проводите опросы и внимательно следите за тем, что они пишут;

- соблюдайте правила использования Telegram API, чтобы избежать блокировки бота. Обязательно ознакомьтесь с документацией и не нарушайте ограничения.

Разработанный Telegram-бот имеет значительный потенциал для дальнейшего развития и расширения функциональности. Возможны следующие направления развития:

– реализация модуля тестирования кандидатов: разработка модуля тестирования позволит оценить знания и навыки абитуриентов и подобрать наиболее подходящий отряд;

– интеграция с другими сервисами: возможность интеграции с другими сервисами, такими как система управления учебным процессом (например, для получения информации о студентах) или социальные сети (для распространения информации о студенческом движении), может значительно расширить функциональность бота;

– персонализация контента: бот может анализировать интересы и предпочтения пользователей и предоставлять им персонализированный контент (например, новости, информацию о отрядах, рекомендации по тестированию);

– улучшение интерфейса: дальнейшая оптимизация пользовательского интерфейса с использованием интерактивных элементов (например, встроенных клавиатур) и визуального контента (например, изображений и видео) может сделать взаимодействие с ботом еще более удобным и привлекательным;

– использование машинного обучения: технологии машинного обучения могут быть использованы для автоматического анализа анкет, классификации пользователей по интересам и потребностям, а также для улучшения качества ответов бота на вопросы пользователей;

– поддержка нескольких языков: добавление поддержки нескольких языков позволит расширить аудиторию бота и сделать его доступным для иностранных студентов;

– автоматизация процесса набора в отряды: разработка более сложной системы подачи заявок и их обработки с возможностью автоматического отбора кандидатов по заданным критериям.

В заключение можно отметить, что разработанный Telegram-бот является перспективным инструментом для автоматизации и улучшения процесса взаимодействия с абитуриентами и студентами, заинтересованными в участии в студенческих отрядах. Дальнейшее развитие и расширение функциональности

бота позволит создать эффективную платформу для привлечения новых участников в студенческое движение и повышения его популярности.

					ПГУ1.090301.39.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. Демиденко, А. Telegram Bot. Руководство по созданию бота в мессенджере Телеграм. - 2025. - 23 с.
2. Одинцов, Б. Е. Информационные системы управления эффективностью бизнеса. - 2025. - 206 с.
3. Habr.com. [Электронный ресурс]. URL: <https://habr.com/ru/articles/697052> (дата обращения: 27.04.2025).
4. Рамальо, Л. Python. К вершинам мастерства. - 2018. - 770 с.
5. Цилюрик, О. И. Сравнение языков программирования. - 2018. - 206 с.
6. Оуэнс, М. The Definitive Guide to SQLite. - 2010. - 440 с.
7. ГОСТ 12.0.003-2015. Система стандартов безопасности труда. Электронно-вычислительная техника. Общие требования безопасности.

Приложение А

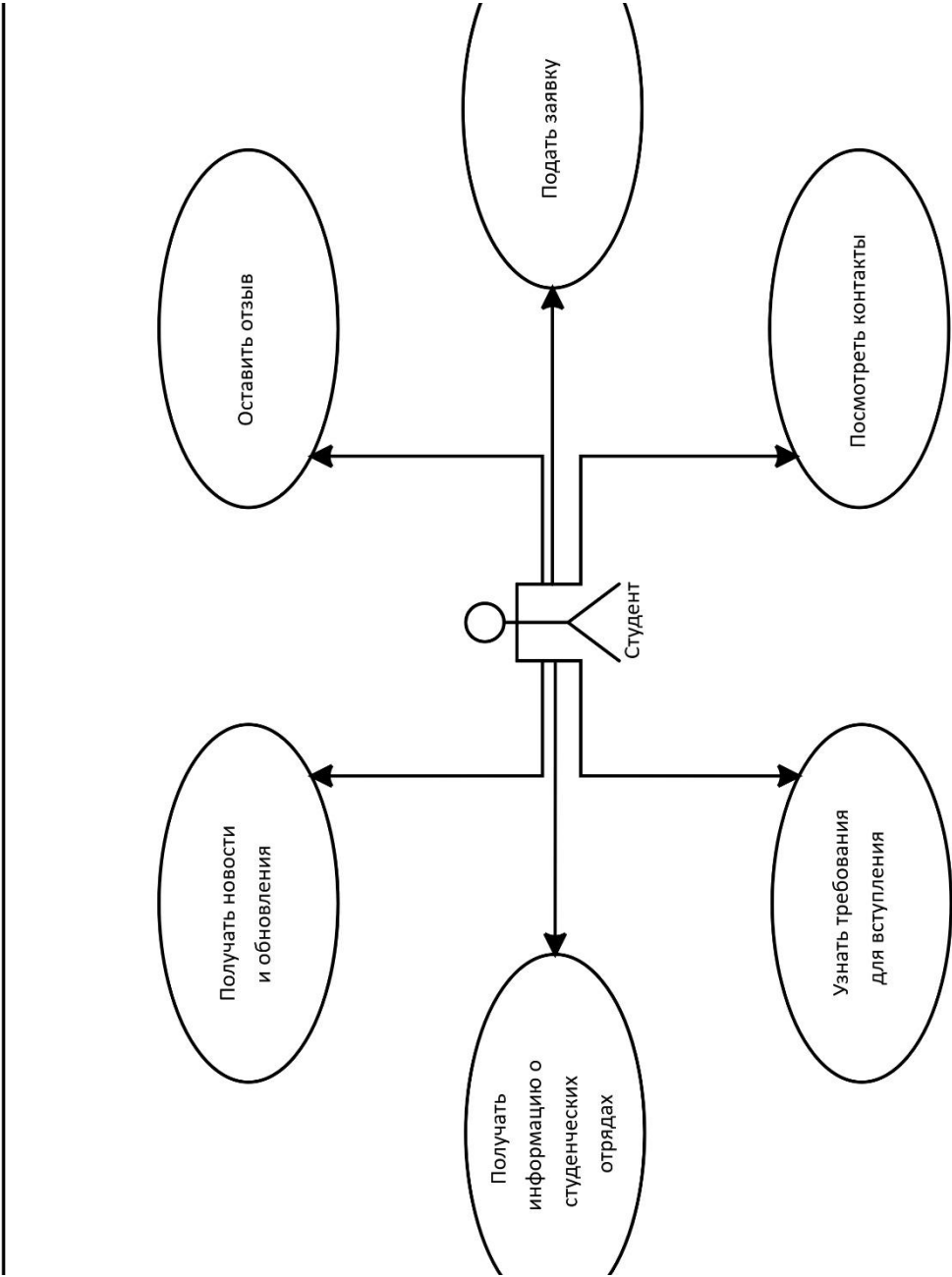
(Обязательное)

Telegram-бот для помощи при вступлении в российские студенческие отряды

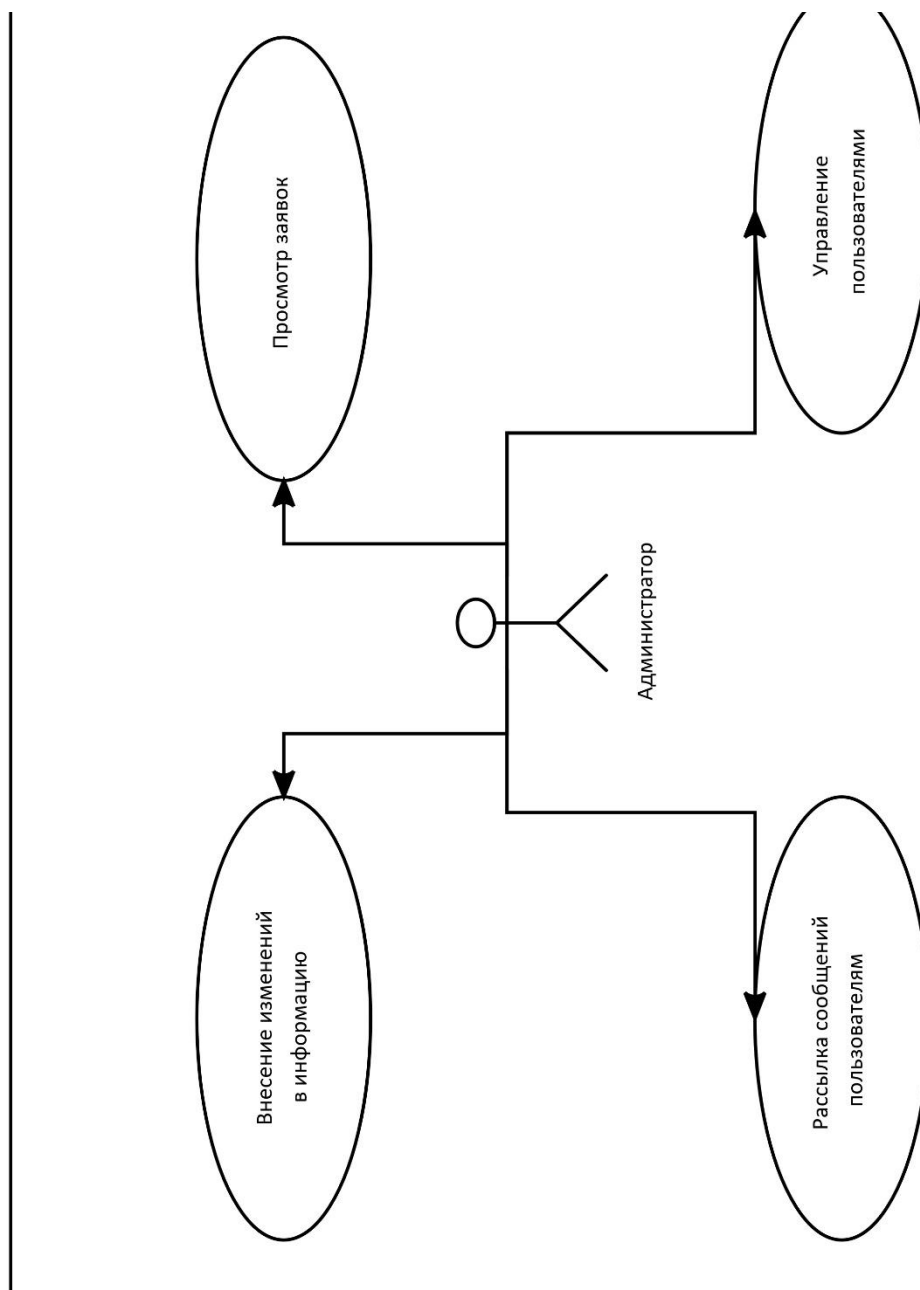
UML диаграммы

					ПГУ1.090301.39.001 ПЗ	Лист
						66
Изм.	Лист	№ докум.	Подпись	Дата		

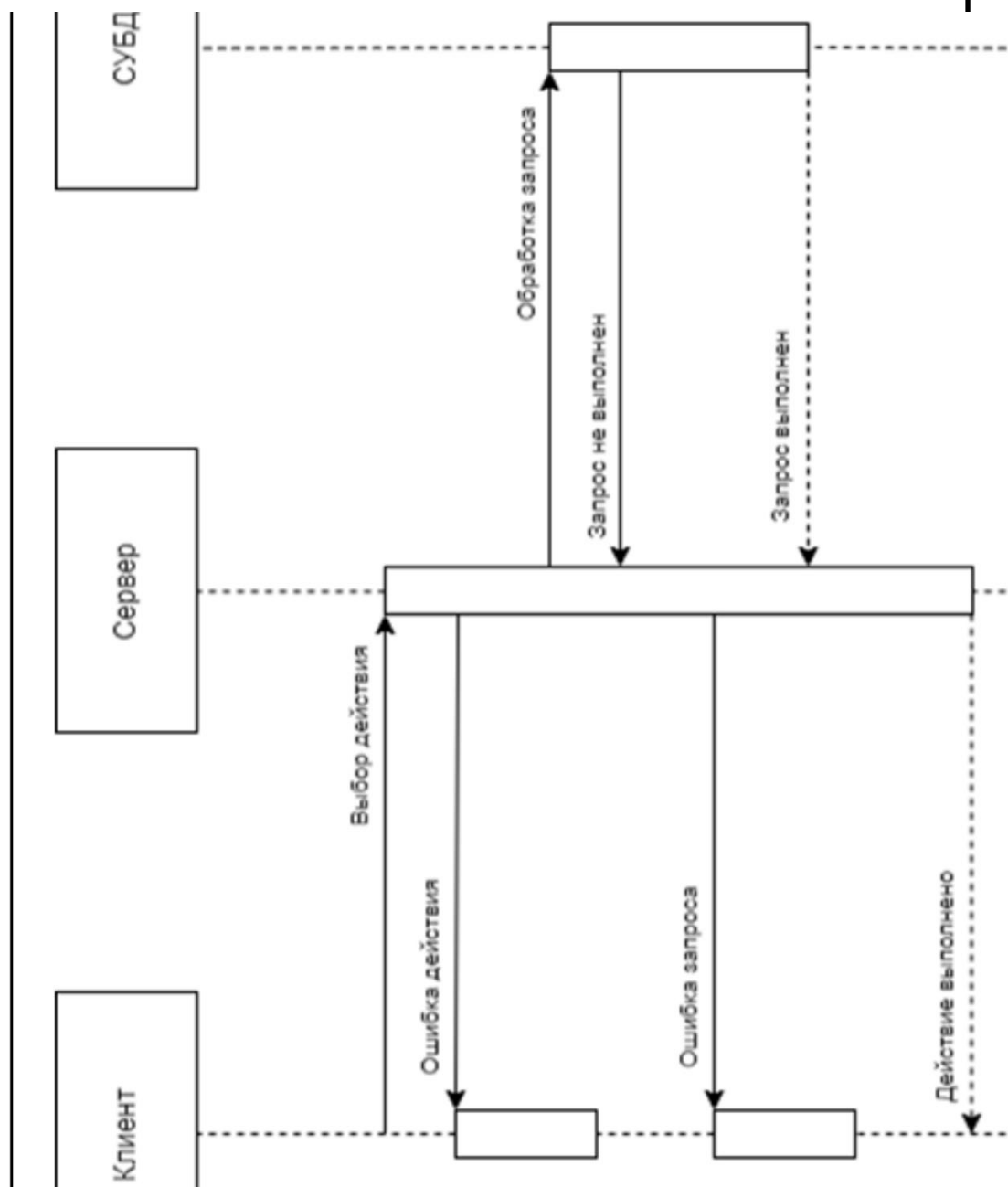
Приложение А.1 – UML – диаграмма прецедентов пользователя



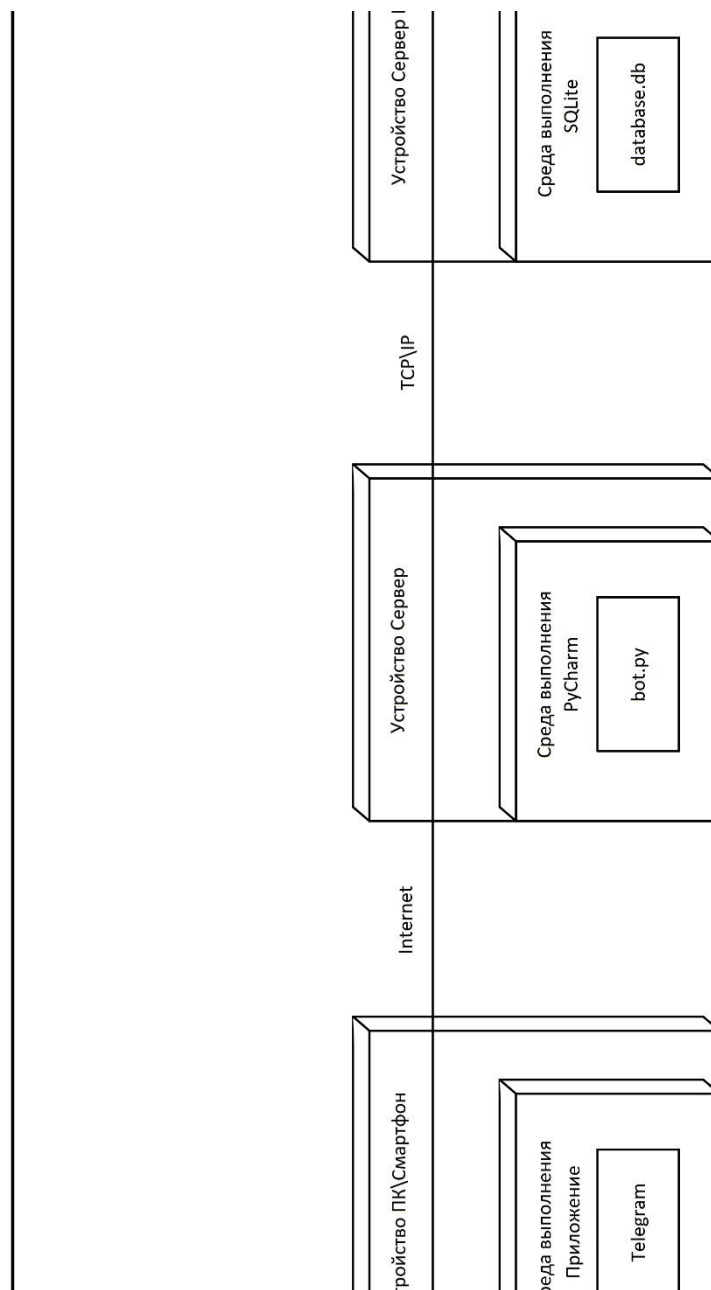
Приложение А.2 – UML – диаграмма прецедентов администратора



Приложение А.3 – UML – диаграмма последовательности



Приложение А.4 – UML – диаграмма развертывания



Приложение Б

(Обязательное)

Telegram-бот для помощи при вступлении в Российские студенческие отряды

Листинг программы

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090301.39.001 ПЗ

Лист

70

Файл «bot.py»

```
import telebot
from telebot import types
import sqlite3
import requests
from bs4 import BeautifulSoup
import re
from string import Template

BOT_TOKEN = '7110963982:AAG0jkbMOjPeTmaG6gV5lNqfF26383bi80A'

bot = telebot.TeleBot(BOT_TOKEN)

anketa = {}
CURRENT_QUESTION = None
USER_ID = None
# --- Подключение к базе данных ---
try:
    conn = sqlite3.connect('database.db',
check_same_thread=False) # Создаем соединение с базой данных
    cursor = conn.cursor()

    # Создаем таблицу, если она еще не существует
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS users (
            id INTEGER PRIMARY KEY,
            name TEXT,
            age INTEGER,
            city TEXT,
            napravlenie TEXT
        )
    """)
    conn.commit()
    print("Подключение к базе данных успешно установлено")

except Exception as e:
    print(f"Ошибка при подключении к базе данных: {e}")

napravleniya_info = {
    "Педагогический отряд Рекорд": {
        "description": "Работа летом на черноморском побережье!
\n👤 \nРазнообразные интересные форумы!\n👤 \nТуристические
слеты не только в Пензе, но и в других городах!\n👤 \nЗимний
агитпоход на лыжах по районам нашей области!
\n👤\nhttps://vk.com/record_pgu?from=groups ",
        "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Рекорд.jpg"
    },
    "Педагогический отряд Кристалл": {
```

```

    "description": "Слышал, что «Кристалл» – семья? Придя
однажды, тебе не захочется уходить!\n💎\nТрижды «Лучший
Студенческий педагогический отряд» готов принять тебя в свои
ряды!\n💎\nЕсли всегда хотел поработать с детьми, твоя мечта
осуществится тут!\n💎\nСамые яркие эмоции, самые памятные
моменты – создавай их сам вместе с
нами!\n💎\nhttps://vk.com/spokristall?from=groups",
    "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Кристалл.jpg"
  },
  "Сервисный отряд Гранат": {
    "description": "Опыт работы в сфере обслуживания
официантом, барменом и помощником повара!\n🍷 \nБесплатное
обучение!\n🍷 \nНовые знакомства и дружная команда!\n🍷
\nНезабываемые впечатления и море позитива!
\n🍷\nhttps://vk.com/granat_pnz?from=groups",
    "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Гранат.jpg"
  },
  "Путинный отряд Гавань": {
    "description": "Мы первыми встречаем самые нежные
рассветы и первыми провожаем яркие закаты!\n🌅\nЗавтракаем
вкуснейшей в мире икрой!\n🌅\nИногда общаемся с
медведями!\n🌅\nЖивём на берегу Тихого
океана!\n🌅\nhttps://vk.com/spuoharbor?from=groups",
    "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Гавань.jpg"
  },
  "Строительный отряд Династия": {
    "description": "Обучение и получение рабочей профессии
штукатура или маляра на бесплатной основе от высококлассных
мастеров своего дела!\n🏠\nОфициальное трудоустройство в ПГУ!
Будем улучшать материально – техническую базу!\n🏠\nДостойный
заработок в летний период!\n🏠\nОткрытие новых
талантов!\n🏠",
    "image_path":
"C:/Users/Tanya/Desktop/dip/Отряды/Династия.jpg"
  }
}

kak_vstupit_info = {
  "Требования к кандидатам": "<b>Требования к
кандидатам</b>\nВозраст от 14 лет, ответственность, желание
работать в команде.",
  "Процесс подачи заявки": "<b>Процесс подачи
заявки</b>\nЗаполните анкету, приходите на собрание.",
  "Обучение и подготовка": "<b>Обучение и
подготовка</b>\nПрофильное обучение абсолютно бесплатно. Форумы

```

и слеты в Пензе и не только.",

"Часто задаваемые вопросы": "Ответы на популярные вопросы о вступлении в отряд.\n\nСколько стоит вступить в студенческий отряд и обучиться?\nЧтобы вступить в студенческий отряд нужно заплатить 500 рублей - это членский взнос.\nНужно ли иметь какие-то специальные навыки или образование, чтобы вступить в отряд?\nНет.\nКакая зарплата? \nВ зависимости от направления разная заработная плата"

}

contacts_info = {

"Руководитель": "Кирсанова Татьяна Игоревна, +7 (999) 123-45-67",

"Справочный отдел": "Скорая Кира Юрьевна, +7 (999) 987-65-43",

"Email": "info@studentotryad.ru",

"Адрес": "г. Пенза, стадион Темп"

}

def send_message_safe(chat_id, text, reply_markup=None, parse_mode=None):

"""Отправляет сообщение с обработкой возможных ошибок."""

try:

bot.send_message(chat_id, text,
reply_markup=reply_markup, parse_mode=parse_mode)

except Exception as e:

print(f"Ошибка при отправке сообщения: {e}")

@bot.message_handler(commands=['start'])

def start(message):

print(f"Функция start вызвана пользователем

{message.from_user.id}, ID чата: {message.chat.id}")

markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)

item1 = types.KeyboardButton("Заполнить анкету")

item2 = types.KeyboardButton("Основное меню")

markup.add(item1, item2)

bot.send_message(message.chat.id, "Добро пожаловать! Что вы хотите сделать?", reply_markup=markup)

@bot.message_handler(func=lambda message: message.text ==

"Основное меню" or message.text == "Вернуться в основное меню")

def main_menu(message):

print(f"Функция main_menu вызвана пользователем
{message.from_user.id}")

markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)

item1 = types.KeyboardButton("Отряды ПГУ")

item2 = types.KeyboardButton("Новости")

item3 = types.KeyboardButton("Как вступить в студенческие
отряды")

item4 = types.KeyboardButton("Контакты")

item5 = types.KeyboardButton("Заполнить анкету")

markup.add(item1, item2, item3, item4, item5)

					ПГУ1.090301.39.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		74

```

        bot.send_message(message.chat.id, "Выберите раздел:",
reply_markup=markup)

@bot.message_handler(func=lambda message: message.text ==
"Отряды ПГУ")
def napravleniya_menu(message):
    print(f"Функция napravleniya_menu вызвана пользователем
{message.from_user.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    for napravlenie in napravleniya_info.keys():
        markup.add(types.KeyboardButton(npravlenie))
    markup.add(types.KeyboardButton("Вернуться в основное
меню"))
    bot.send_message(message.chat.id, "Выберите направление,
чтобы узнать подробнее:", reply_markup=markup)

@bot.message_handler(func=lambda message: message.text in
napravleniya_info and CURRENT_QUESTION is None)
def napravlenie_info(message):
    print(f"Функция napravlenie_info вызвана пользователем
{message.from_user.id}")
    napravlenie = napravleniya_info[message.text]
    description = napravlenie["description"]
    image_path = napravlenie["image_path"]

    try:
        with open(image_path, "rb") as photo:
            bot.send_photo(message.chat.id, photo.read(),
caption=description, reply_markup=types.ReplyKeyboardRemove())
    except FileNotFoundError:
        bot.send_message(message.chat.id, f"Файл {image_path} не
найден.")
    main_menu(message)
    return
    except Exception as e:
        print(f"Ошибка при отправке фото: {e}")
        bot.send_message(message.chat.id, "Произошла ошибка при
отправке фото.")
        main_menu(message)
        return

    main_menu(message)

@bot.message_handler(func=lambda message: message.text == "Как
вступить в студенческие отряды")
def kak_vstupit_menu(message):
    print(f"Функция kak_vstupit_menu вызвана пользователем
{message.from_user.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    for item in kak_vstupit_info.keys():

```

```

        markup.add(types.KeyboardButton(item))
    markup.add(types.KeyboardButton("Вернуться в основное
меню"))
    send_message_safe(message.chat.id, "Выберите пункт, чтобы
узнать подробнее:", reply_markup=markup)

@bot.message_handler(func=lambda message: message.text in
kak_vstupit_info)
def kak_vstupit_info_handler(message):
    print(f"Функция kak_vstupit_info_handler вызвана
пользователем {message.from_user.id}")
    send_message_safe(message.chat.id,
kak_vstupit_info[message.text],
reply_markup=types.ReplyKeyboardRemove(), parse_mode="HTML")
    kak_vstupit_menu(message)

@bot.message_handler(func=lambda message: message.text ==
"Контакты")
def contacts(message):
    print(f"Функция contacts вызвана пользователем
{message.from_user.id}")
    contacts_text = "<b>Контакты:</b>\n\n"
    for key, value in contacts_info.items():
        contacts_text += f"<b>{key}</b> {value}\n"
    bot.send_message(message.chat.id, contacts_text,
parse_mode='HTML')
    main_menu(message)

# --- Функции для работы с анкетой ---
@bot.message_handler(commands=['start'])
def start(message):
    print(f"Функция start вызвана пользователем
{message.from_user.id}, ID чата: {message.chat.id}")
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    item1 = types.KeyboardButton("Заполнить анкету")
    item2 = types.KeyboardButton("Основное меню")
    markup.add(item1, item2)
    bot.send_message(message.chat.id, "Добро пожаловать! Что вы
хотите сделать?", reply_markup=markup)

@bot.message_handler(func=lambda message: message.text ==
"Заполнить анкету")
def start_anketa(message):
    print(f"Функция start_anketa вызвана пользователем
{message.from_user.id}")
    global CURRENT_QUESTION, USER_ID
    anketa[message.chat.id] = {}
    USER_ID = message.chat.id
    CURRENT_QUESTION = "Имя"
    bot.send_message(message.chat.id, "Начнем заполнение анкеты.
Как вас зовут?")

```

```

@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Имя" and message.chat.id == USER_ID)
def ask_age(message):
    print(f"Функция ask_age вызвана пользователем
{message.from_user.id}")
    anketa[message.chat.id]["Имя"] = message.text
    global CURRENT_QUESTION
    CURRENT_QUESTION = "Возраст"
    bot.send_message(message.chat.id, "Сколько вам лет?")

@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Возраст" and message.chat.id == USER_ID)
def ask_city(message):
    print(f"Функция ask_city вызвана пользователем
{message.from_user.id}")
    try:
        age = int(message.text)
        if age < 14:
            bot.send_message(message.chat.id, "К сожалению, в
студенческий отряд можно вступить с 14 лет. \nНо ты можешь
узнать побольше о студенческих отрядах и вступить как только
достигнешь нужного возраста.") # Отправляем сообщение "Ты лох"
            return
        main_menu(message) # Прекращаем дальнейшее выполнение
функции
        anketa[message.chat.id]["Возраст"] = age
    except ValueError:
        bot.send_message(message.chat.id, "Пожалуйста, введите
возраст числом.")
        return

    global CURRENT_QUESTION
    CURRENT_QUESTION = "Город"
    bot.send_message(message.chat.id, "С какого вы факультета и
ВУЗа?")

@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Город" and message.chat.id == USER_ID)
def ask_napravlenie(message):
    print(f"Функция ask_napravlenie вызвана пользователем
{message.from_user.id}")
    anketa[message.chat.id]["Город"] = message.text
    global CURRENT_QUESTION
    CURRENT_QUESTION = "Направление"
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True,
one_time_keyboard=True)
    napravleniya = list(npravleniya_info.keys())
    for napravlenie in napravleniya:
        markup.add(types.KeyboardButton(npravlenie))
    bot.send_message(message.chat.id, "Какой отряд вас
интересует?", reply_markup=markup)

```

```

@bot.message_handler(func=lambda message: CURRENT_QUESTION ==
"Направление" and message.chat.id == USER_ID)
def anketa_done(message):
    print(f"Функция anketa_done вызвана пользователем
{message.from_user.id}")
    napravleniya = list(npravleniya_info.keys())
    if message.text not in napravleniya:
        send_message_safe(message.chat.id, "Пожалуйста, выберите
отряд из списка.")
        return

    anketa[message.chat.id]["Направление"] = message.text
    global CURRENT_QUESTION, USER_ID
    CURRENT_QUESTION = None
    USER_ID = None

# --- Сохранение данных в базу данных ---
try:
    cursor.execute("""
        INSERT INTO users (name, age, city, napravlenie)
        VALUES (?, ?, ?, ?)
        """, (anketa[message.chat.id]["Имя"],
            anketa[message.chat.id]["Возраст"],
            anketa[message.chat.id]["Город"],
            anketa[message.chat.id]["Направление"]))
    conn.commit()
    print(f"Данные пользователя {message.chat.id} успешно
сохранены в базе данных")
    print(f"Последняя вставленная строка:
{cursor.lastrowid}") # Проверяем ID последней вставленной
строки
except Exception as e:
    print(f"Ошибка при сохранении данных в базу данных:
{e}")

anketa_text = "Ваша анкета:\n"
for key, value in anketa[message.chat.id].items():
    anketa_text += f"{key}: {value}\n"
send_message_safe(message.chat.id, anketa_text,
reply_markup=types.ReplyKeyboardRemove())
send_message_safe(message.chat.id, "Спасибо за заполнение
анкеты!")
#main_menu(message) # Убрали вызов main_menu

def get_news():
    """Получает последнюю новость с сайта трудкрут.рф."""
    url = 'https://трудкрут.рф/novosti.html'

    try:
        response = requests.get(url)
        response.raise_for_status()

```

```

response.encoding = 'utf-8'

soup = BeautifulSoup(response.text, 'html.parser')

# Находим все элементы a, содержащие новости
news_items = soup.find_all('a', href=re.compile(r"/view-
news\.html\?id=\d+"))

if news_items:
    # Получаем только первую новость
    item = news_items[0]
    link = "https://трудкрыт.рф" + item['href']
    b_tag = item.find('b')
    text = b_tag.text.strip() if b_tag else "Заголовок
не найден"

    # Извлекаем URL картинки
    img_tag = item.find('img')
    img_url = "https://трудкрыт.рф" + img_tag['src'] if
img_tag else None

    return {'link': link, 'text': text, 'img_url':
img_url}
else:
    return "Новости не найдены"

except requests.exceptions.RequestException as e:
    return f"Ошибка при запросе к сайту: {e}"
except Exception as e:
    return f"Ошибка при парсинге страницы: {e}"

def send_news_with_image(chat_id, news_item):
    """Отправляет новость с картинкой в Telegram."""
    text = f"<b><a
href='{news_item['link']}'>{news_item['text']}'</a></b>"
    img_url = news_item['img_url']

    if img_url:
        try:
            bot.send_photo(chat_id, photo=img_url, caption=text,
parse_mode="HTML")
        except Exception as e:
            print(f"Ошибка при отправке фото: {e}")
            bot.send_message(chat_id, text + f"\n(Не удалось
загрузить изображение: {img_url})", parse_mode="HTML")
    else:
        bot.send_message(chat_id, text, parse_mode="HTML")

@bot.message_handler(func=lambda message: message.text ==
"Новости")

```

```

def send_news_menu(message):
    """Отправляет последнюю новость с сайта трудкрут.рф."""
    chat_id = message.chat.id
    news = get_news()

    if isinstance(news, str):
        # Если get_news вернула строку (ошибку), отправляем ее
        send_message_safe(chat_id, news)
    else:
        # Если get_news вернула словарь с новостью, отправляем
        ее с картинкой
        send_news_with_image(chat_id, news)

    main_menu(message)

if __name__ == '__main__':
    try:
        bot.polling(none_stop=True)
    except Exception as e:
        print(f"Ошибка при запуске бота: {e}")

if __name__ == '__main__':
    try:
        bot.polling(none_stop=True, timeout=60)
    except Exception as e:
        print(f"Ошибка при запуске polling: {e}")

```

Приложение В

(Обязательное)

Telegram-бот для помощи при вступлении в Российские студенческие отряды

Презентация

					ПГУ1.090301.39.001 ПЗ	Лист
						81
Изм.	Лист	№ докум.	Подпись	Дата		