

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: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Вычислительные машины, комплексы, системы и сети»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

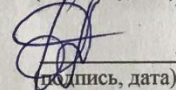
на тему

**ПРОГРАММА ДЛЯ ИНФОРМАЦИОННОЙ
ПОДДЕРЖКИ**

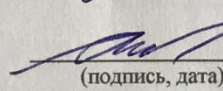
Студент

 Харитонов Даниил Алексеевич
(подпись, дата) 31.05.2023 (ФИО полностью)

Руководитель

 7.06.2023 Федюнин Р.Н.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

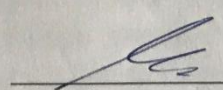
 7.06.2023 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент

зав.каф. Программирование Назгульханов 08.06.23 Мартышкин А.И.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

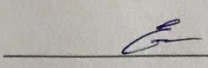
Работа допущена к защите (протокол зас. кафедры от 2.06.2023 № 14)

Заведующий кафедрой

 М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 9.06.2023 № 9)

Секретарь ГЭК

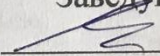
 Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2023

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

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
« 29 » 11 20 21 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Харитонов Даниил Алексеевич гр. 21ВВм1 факультета
вычислительной техники

2. Тема работы «Программа для информационной поддержки»

Тема утверждена приказом ПГУ № 924/10 от " 29 " 11 20 21 г.

3. Руководитель магистерской диссертации к.т.н., доцент Федюнин Р. Н.

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

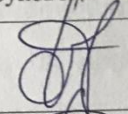
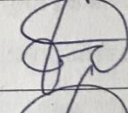
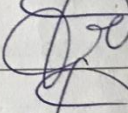
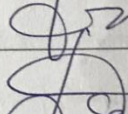
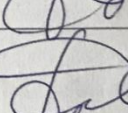
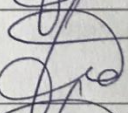
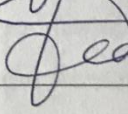
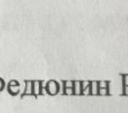
Разработать и отладить чат-бота Telegram для информационной поддержки студентов и абитуриентов ПГУ, основанные на языке Python с использованием базы данных для хранения пользователей.

На основе полученных данных, проанализировать имеющиеся чат-боты, составить концептуальную модель программного продукта, а также добавить административную панель для внесения изменений пользователя.

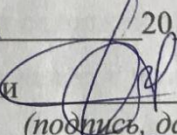
5. Перечень подлежащих разработке вопросов

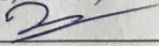
Требуется провести анализ предметной области, установить задачу, провести сравнительный анализ имеющихся программных продуктов и методов взаимодействия вуза с абитуриентами и студентами, определить концептуальную модель, проанализировать актуальность поставленной задачи, определить срезу разработки и языка программирования, реплицировать модель, отладить модель и провести тестирование разработанной модели.

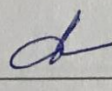
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
	Изучение предметной области и постановка задачи	5%	21.09.2021-21.10.2021	
	Анализ аудитории пользователей мессенджеров	10%	22.10.2021-22.11.2021	
	Анализ чат-ботов Telegram и обзор существующих решений	15%	23.11.2021-23.12.2021	
	Анализ инструментов разработки	20%	24.12.2021-24.04.2022	
	Регистрация чат-бота и размещение на удаленном сервере	10%	25.04.2022-25.08.2022	
	Реализация клиентской и серверной части чат-бота	20%	26.09.2021-26.12.2021	
	Проектирование и реализация базы данных чат-бота	15%	27.12.2023-30.03.2023	
	Оформление выпускной работы	5%	31.03.2023-31.05.2023	

Дата выдачи задания " 29 " 11 20 21 г.

Руководитель магистерской диссертации  Федюнин Р.Н.
(подпись, дата)

Задание к исполнению принял студент  Харитонов Д.А.
(подпись, дата)

Работу к защите допустить
Декан факультета  08.06.2023 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Магистерская работа содержит стр., 36 рис., 3 табл., 28 источн., прил.

Структура работы включает в себя введение, 7 разделов, заключение и список использованной литературы.

ЧАТ-БОТ TELEGRAM, PYTHON, БАЗА ДАННЫХ, СРЕДА РАЗРАБОТКИ PYCHARM, БИБЛИОТЕКИ, HEROKU

Цель работы – разработать программу для информационной поддержки, чат-бот Telegram, основанный на языке Python в среде разработки PyCharm позволяющий взаимодействовать с программой посредством чата – получать информацию о вузе, регистрироваться, получать актуальную рассылку и новости, а также модуль для управления пользователями. Улучшить взаимодействие приемной комиссии и сотрудников вуза со студентами и абитуриентами.

В процессе работы проводился анализ актуальности работы, обзор средств для разработки, была построена концептуальная модель программного продукта и разработана база данных.

В результате проведенной работы был спроектировано и разработано программное обеспечение для информационной поддержки студентов посредством чат-бота в Telegram.

Научная новизна состоит в том, что проанализировано большое количество программных средств и решений, продемонстрировано большое количество применений чат-ботов для поддержки абитуриентов и студентов, а также разработан уникальный чат-бот непосредственно для ПГУ.

					ПГУ1.090401.10.001 ПЗ		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб		Харитонов Д.А.			Программа для информационной поддержки		
Провер.		Федюнин Р.Н.					
Н. Контр.		Бычков А.С.					
Утверд.		Митрохин М.А.					
					Лит.	Лист	Листов
						2	
					ФВТ, каф. ВТ гр. 21ВВм1		

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ	5
ВВЕДЕНИЕ	6
1 Анализ предметной области и постановка задачи.....	8
1.1 Анализ методов взаимодействия вуза с абитуриентами и студентами	8
1.2 Анализ аудитории пользователей мессенджеров	13
1.3 Обзор существующих решений	14
2 Чат-боты Telegram.....	17
2.1 Интерфейс телеграмм бота.....	19
2.2 Успешный опыт в использовании Telegram университетами	21
3 Проектирование чат-бота Telegram	23
3.1 Концептуальная модель программного продукта Проектирование чат-бота Telegram	23
4 Анализ инструментов разработки.....	25
4.1 Язык программирования Python	25
4.2 Среда разработки PyCharm.....	27
4.3 Анализ выбора базы-данных для хранения пользователей.	29
4.3.1 База данных SQLite.....	29
4.3.2 База данных MySQL	32
4.3.3 База данных PostgreSQL	33
5 Разработка программного продукта	34
5.1 Анализ API и библиотек для создания чат-бота Telegram.....	34
5.1.1 Telegram Bot API.....	34
5.1.2 Библиотека pyTelegramBotAPI, Aiogram, Telebot.....	36
5.2 Разработка чат-бота Telegram	37
5.2.1 Регистрация чат-бота для Telegram Bot API.....	37
5.2.2 Реализация клиентской части чат-бота	43
5.2.3 Проектирование и реализация базы данных чат-бота	45
5.2.4 Реализация серверной части чат-бота	48
5.3 Административная панель чат бота	50

5.3.1 Разработка и реализация компонентной структуры чат-бота.....	51
6 Результаты разработки чат-бота	52
6.1 Тестирование системы чат-бота.....	52
7 Удаленный сервер Heroku	58
7.1 Анализ и размещение облачного сервера Heroku.....	58
ЗАКЛЮЧЕНИЕ	64
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	66
Приложение А Листинг программы.....	69
Приложение Б Презентация	89

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей работе применяют следующие сокращения и обозначения

API (Application Programming Interface) – Интерфейс прикладного программирования

JSON (JavaScript object notation) - Обозначение объекта JavaScript

MVC (Model-View-Controller) – Модель-Представление-Контроллер

СУБД - Система управления базами данных

БД – База данных

HTTP (HyperText Transfer Protocol) - протокол передачи гипертекста

ID (Identifier) – идентификатор

SSL-сертификат (Secure Sockets Layer) - Уровень защищенных сокетов

IDE (Integrated development environment) - Интегрированная среда разработки

ПО – Программное обеспечение

URL-запросы (Uniform Resource Locator) - Унифицированный указатель ресурса

ВВЕДЕНИЕ

Цель каждого учебного заведения в сфере высшего образования состоит в подготовке и выпуске высококвалифицированных и востребованных специалистов, готовых применять свои знания и навыки на рынке труда. В условиях острой конкуренции среди высших учебных заведений очень важно поддерживать активное общение с потенциальными поступающими.

Пензенский государственный университет является одним из ведущих образовательных центров, где формируются кадры в различных сферах. Университет предлагает широкий выбор образовательных программ и направлений, что заставляет абитуриентов затратить много времени, чтобы узнать все детали о каждом образовательном учреждении.

Актуальность этой работы обусловлена состоянием рынка информационных технологий. Каждый год появляются новые программные продукты, веб-сервисы и мобильные приложения, которые не только обеспечивают коммуникацию, но и помогают укрепить бренд университета как ведущего образовательного центра.

Цель данного исследования заключается в разработке инновационного чат-бота для абитуриентов и студентов Пензенского государственного университета, который способствует эффективному взаимодействию с ключевыми участниками образовательного процесса - будущими студентами.

Для достижения этой цели были поставлены следующие задачи:

1. Тщательно изучить теоретические основы и ознакомиться с существующими методами взаимодействия с абитуриентами и студентами приемной комиссии Пензенского государственного университета,
2. Провести детальный анализ инструментов и методов разработки чат-ботов для платформы Telegram,
3. Разработать чат-бота, используя язык программирования Python,
4. Создать эффективную базу данных для хранения информации о различных направлениях подготовки и других данных, необходимых для работы чат-бота.

Предметом исследования является чат-бот, который будет разработан для платформы Telegram и будет результатом данной работы.

Telegram — это сервис обмена сообщениями, который предоставляет возможность интерактивного взаимодействия пользователей. Разработчики Telegram предоставляют возможность создания групп, каналов и ботов, чтобы пользователи могли общаться с аккаунтами компаний и организаций и минимизировать время, затрачиваемое на повторяющиеся вопросы.

Чат-бот представляет собой встроенную функцию мессенджера, разработанную с использованием современных технологий, и его целью является организация эффективного общения пользователей и предоставление им необходимой информации в рамках определенных тематик.

Таким образом, решение разработать чат-бот было обусловлено желанием оптимизировать работу сотрудников и студентов приемной комиссии университета, которые тратят значительное количество времени на передачу информации, доступной в открытых источниках. В данной работе были рассмотрены актуальность проблемы, методы ее решения и аналогичные разработки.

1 Анализ предметной области и постановка задачи

Принятие решения о создании чат-бота было вдохновлено желанием оптимизировать эффективность работы сотрудников и абитуриентов приемной комиссии университета. В настоящей исследовательской работе мы осветили актуальность данной проблемы, рассмотрели различные подходы к ее решению и изучили сопоставимые разработки, существующие на данный момент. В результате этого анализа мы пришли к выводу о необходимости разработки собственного чат-бота, который будет уникально адаптирован к потребностям нашей приемной комиссии и обеспечит более эффективный обмен информацией без необходимости повторного доступа к общедоступным источникам.

1.1 Анализ методов взаимодействия вуза с абитуриентами и студентами

В целях оптимизации эффективности взаимодействия сотрудников приемной комиссии и абитуриентов, принято решение разработать инновационного чат-бота. Данная инициатива была вдохновлена стремлением улучшить процесс передачи информации, который до сих пор требовал значительного времени и усилий со стороны сотрудников, несмотря на доступность той же самой информации в открытых источниках. В ходе данного исследования мы провели тщательный анализ актуальности проблемы, рассмотрели различные методы ее решения и изучили существующие аналогичные разработки. В результате нашего анализа мы пришли к выводу о необходимости разработки собственного чат-бота, специально адаптированного к потребностям нашей приемной комиссии. Таким образом, мы стремимся обеспечить более эффективный обмен информацией, минимизировать дублирование работы и облегчить доступ к необходимой информации без лишних затрат времени и ресурсов. [1,2,3]

Взаимодействие приемной комиссии университета с абитуриентами осуществляется двумя основными способами - очным и дистанционным.

Очные методы предоставления информации включают в себя следующие возможности:

1. Очные консультации, где абитуриенты могут получить подробную информацию от членов приемной комиссии, прямо в университете,
2. Посещение Дня открытых дверей, где абитуриенты могут ознакомиться с университетом и его программами,
3. Участие в научных мероприятиях, таких как олимпиады и конференции, организуемых университетом,
4. Принятие участия в креативных мероприятиях, таких как конкурсы и специальные мероприятия, вроде "Дня открытых дверей ПГУ",
5. Распространение информации о занятиях, мероприятиях и возможных изменениях в расписании через различные каналы связи,
6. Организация экскурсий для абитуриентов и студентов, позволяющих им более подробно познакомиться с ПГУ.

Дистанционные методы взаимодействия включают следующие способы передачи информации:

1. Телефонные консультации со стороны сотрудников приемной комиссии для абитуриентов,
2. Консультации по электронной почте со стороны сотрудников приемной комиссии,
3. Коммуникация с абитуриентами через электронную почту,
4. Предоставление консультаций на веб-сайте приемной комиссии,
5. Взаимодействие с абитуриентами в социальных сетях, таких как "ВКонтакте" и "Viber", через аккаунты сотрудников приемной комиссии,
6. Общение и консультации с абитуриентами через мессенджеры,
7. Предоставление советов и поддержки студентам по вопросам, связанным с учебной деятельностью и запланированными мероприятиями.

Таким образом, приемная комиссия университета использует разнообразные методы и каналы связи, чтобы обеспечить максимальную

доступность и удобство для абитуриентов и студентов в получении необходимой информации.

Рисунки 1 и 2 иллюстрируют очно-заочные способы взаимодействия, рисунок 3 - дистанционные способы взаимодействия.

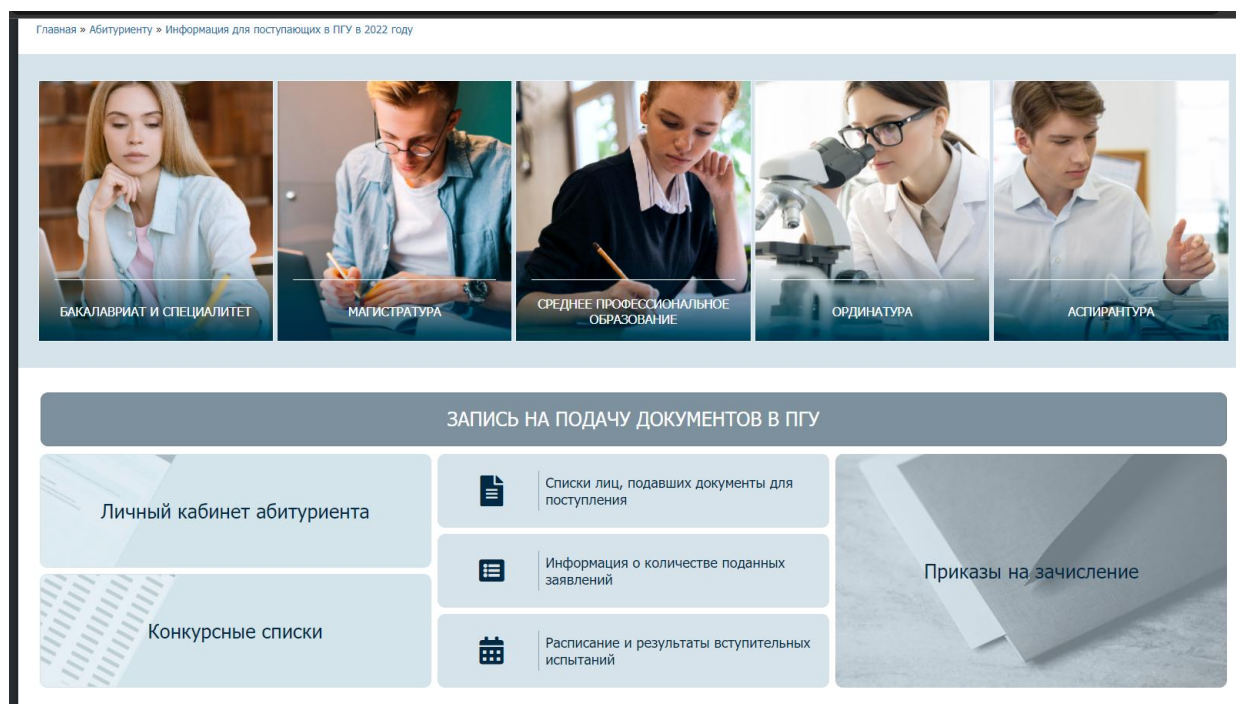


Рисунок 1– Страница Приемной комиссии на официальном веб-сайте университета

The image shows a screenshot of a contact form on a website. At the top, there is a navigation bar with links: "Главная", "ОБ УНИВЕРСИТЕТЕ", "ОБРАЗОВАНИЕ", "НАУКА", and "INTERNATIONAL". Below the navigation bar is a "Список вопросов" section. The form includes fields for "Фамилия Имя Отчество:", "E-mail:", "Телефон:", "Статус:" (with a dropdown menu), "Тема сообщения:", and "Текст сообщения:". There is a "Я не робот" checkbox and a "geCAPTCHA" logo. At the bottom, there are "Отправить" and "Отмена" buttons. A small disclaimer at the bottom states: "* нажатием на кнопку 'отправить', отправитель дает согласие на обработку своих персональных данных в соответствии со стандартом университета СТУ 3.01-2011 'Обработка и защита персональных данных'".

Рисунок 2 – Страница Приемной комиссии на официальном веб-сайте университета - главный информационный ресурс для абитуриентов.



Рисунок 3 Официальная страница университета в социальной сети "ВКонтакте" - центральный канал связи и информации для студентов и абитуриентов.

Действительно, современная приемная комиссия все больше ориентируется на использование дистанционных способов взаимодействия с абитуриентами, что приносит явные преимущества по сравнению с традиционными очными методами. [4]

В последние годы технологический прогресс и доступность интернета позволили приемной комиссии университетов эффективно использовать различные онлайн-инструменты для коммуникации с абитуриентами. Дистанционные методы взаимодействия, такие как электронная почта, телефонные консультации, веб-сайты и социальные сети, обеспечивают быстрый и удобный доступ к информации, а также гибкость в общении. Дистанционные методы взаимодействия обладают следующими достоинствами:

1. **Общедоступность:** С использованием дистанционных средств коммуникации, таких как веб-сайты, социальные сети, и мессенджеры,

абитуриенты могут получить доступ к информации и консультации в любое удобное для них время, независимо от местоположения,

2. Низкая стоимость: Дистанционные методы взаимодействия требуют меньше финансовых затрат по сравнению с организацией очных мероприятий. Нет необходимости оплачивать аренду помещений, транспортные расходы и другие связанные издержки,

3. Практичность для всех участников: Дистанционные методы позволяют абитуриентам и работникам приемной комиссии взаимодействовать из любого удобного места, без необходимости присутствия в университете или ограничений по расписанию,

4. Сохранение истории переписки - одно из значимых преимуществ дистанционного взаимодействия с абитуриентами через сеть Интернет и мессенджеры. По сравнению с традиционными очными встречами, эта возможность предоставляет удобство в том, что каждое сообщение остается записанным и доступным для обращения в дальнейшем. История переписки может быть полезна при разрешении спорных ситуаций, а также для уточнения и проверки информации при необходимости.

Одним из эффективных способов дистанционного консультирования абитуриентов является использование различных онлайн-платформ, таких как официальные веб-сайты университета, страницы в социальных сетях и мессенджеры. Особое внимание следует уделять использованию мессенджеров, которые обеспечивают удобную и практически неисчерпаемую возможность для контакта между абитуриентами и сотрудниками приемной комиссии.

Исходя из перечисленных преимуществ, важно признать, что развитие дистанционных форм взаимодействия с абитуриентами, особенно через сеть Интернет и мессенджеры, становится важным направлением работы приемной комиссии. Это обеспечивает высокую эффективность и комфорт для обеих сторон, а также способствует улучшению и совершенствованию процесса приема в университет.

1.2 Анализ аудитории пользователей мессенджеров

Мессенджеры действительно стали наиболее популярным средством коммуникации, превращаясь в мощную платформу для обмена сообщениями, а также для создания публичных чатов и каналов. Они привлекают все больше пользователей, предоставляя удобную основу для потребления контента.

В особенности, Телеграмм является одним из лидеров в этой области и продолжает набирать обороты. Пользователи предпочитают Телеграмм благодаря подборкам с анонсами новостей, которые регулярно публикуются на каналах несколько раз в день. В отличие от традиционных СМИ, которые буквально бомбардируют подписчиков новостями в течение всего дня, Телеграмм предлагает более организованный и сфокусированный подход к предоставлению информации.

Однако использование мессенджеров не ограничивается только индивидуальным общением. Они также стали мощной платформой для вступления в публичные чаты и чтения публичных каналов. Этот тренд стал особенно заметным в последние годы, и мессенджеры все больше превращаются в своего рода социальные медиа, где пользователи могут обмениваться и находить интересный контент.

Тренд увеличения интереса к мессенджерам продолжает развиваться и не замедляется. Это свидетельствует о важности использования мессенджеров в сфере коммуникации и распространении информации, и приемная комиссия университета может успешно воспользоваться этим инструментом для эффективного взаимодействия с абитуриентами. Telegram стал особенно популярным и продолжает активно развиваться. Публичные каналы стали одной из основных причин увеличения интереса к Телеграмму. Русскоязычные каналы привлекают все больше пользователей и предлагают разнообразный контент: от развлекательного до образовательного и политического.

Анализ, проведенный изданием "Telegram-stoore", показывает, что пользователи Телеграмма в русскоязычном секторе в основном имеют возраст

от 18 до 35 лет, высшее образование и работу интеллектуального характера, среди которых значительная доля IT-экспертов, статистика предоставлена на рисунке 4. [5,6]

В заключение можно отметить, что конкуренция между учреждениями, предоставляющими образовательные сервисы, огромна. Дистанционные способы взаимодействия, включая использование мессенджеров, являются наиболее результативными. Поэтому использование мессенджеров, особенно Телеграмма, для взаимодействия с абитуриентами является эффективным и актуальным подходом.

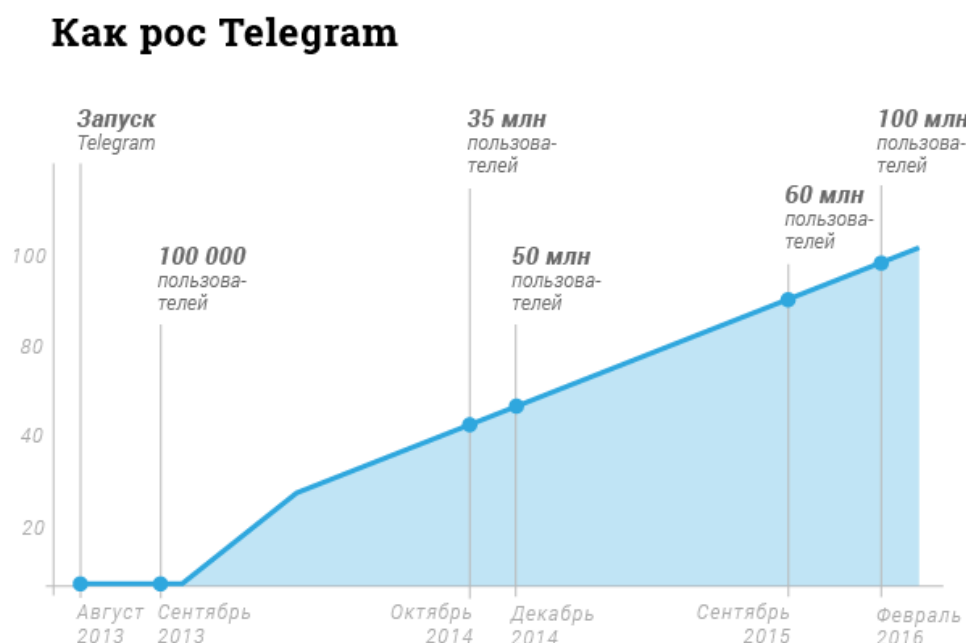


Рисунок 4 - Рост мессенджера Telegram за последнее время

1.3 Обзор существующих решений

При анализе конкурентов в городе Пенза, где разрабатывалось приложение для вуза, одним из релевантных решений, обладающим схожим функционалом, является мобильное приложение «ЭИОС ПГУ» для абитуриентов и студентов Пензенского Государственного Университета. Приложение предлагает удобный интерфейс, позволяющий пользователям войти в личный кабинет и получить доступ к информации о посещаемости и успеваемости. Также в нем реализована возможность изменить статус на студента или абитуриента.

Однако, несмотря на множество функциональных возможностей приложения, стоит отметить, что некоторые из них перенаправляют пользователей на веб-сайт, что в некоторой степени ограничивает автономность приложения.

В данном контексте, разрабатываемое приложение может выделиться своей преимущественной автономностью, предлагая полный функционал внутри приложения без необходимости переключения на веб-сайт. Это создаст более удобное и единообразное пользовательское взаимодействие, обеспечивая абитуриентов и студентов. Приложение «ЭИОС ПГУ» предоставлен на рисунке 5.

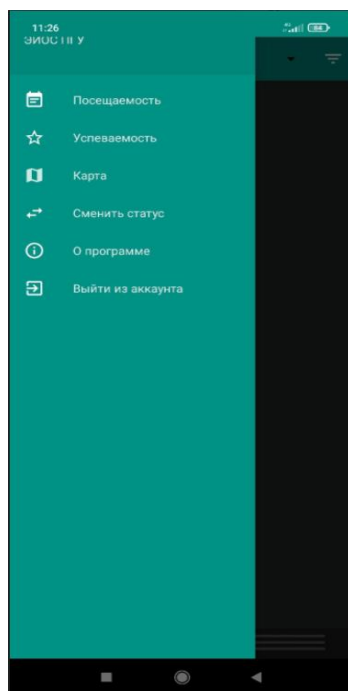


Рисунок 5 приложение "ЭИОС ПГУ"

Для наглядного представления о достоинствах и недостатках систем составлена таблица 1.

Таблица 1- Достоинства и недостатки обоих приложений.

Параметры	Приложения	
	«Телеграмм бот ПГУ»	«ЭИОС ПГУ»

Параметры	Приложения	
	«Телеграмм бот ПГУ»	«ЭИОС ПГУ»
Приоритетное размещение на телефоне	+	+
Обзор направлений вуза	Много информации	Их попросту нет
Новости и мероприятия	Уведомления прямо в чате	Проблема перехода на веб-сайт, потеря автономности
Дизайн	+	+
Поддержка разных платформ	Доступен на всех платформах	Ограничен только определенным инновационным системам
Автономность	+	+
Интуитивный и понятый интерфейс	Прост и быстр в реакции	Не интуитивный непонятный интерфейс
Получение информации по электронной почте	Оповещения и уведомления в одном месте	Уведомлений нет

Исходя из таблицы 1, можно сделать вывод, что "Телеграмм-бот ПГУ" является наиболее идеальным и удобным программным продуктом, который наиболее полно воплощает все необходимые характеристики. Исходя из этого, было принято решение начать разработку программного обеспечения "Телеграмм-бот ПГУ". Это приложение предоставляет пользователям

удобный и простой интерфейс, а также обладает всеми необходимыми функциями для эффективного взаимодействия с абитуриентами и студентами.

Первым шагом в разработке чат-бота является полное определение его основного назначения. Взаимодействие через мессенджер обеспечивает высокий уровень вовлеченности пользователей. Кроме того, использование чат-бота позволяет обеспечить быстрое взаимодействие. Большие группы в социальных сетях также не гарантируют моментального ответа, что особенно важно для абитуриентов. Взаимодействие с чат-ботом обеспечивает моментальный отклик на вопросы пользователей. Поэтому разработка «Телеграмм бот ПГУ» представляется как эффективное решение для достижения поставленных целей.

2 Чат-боты Telegram

Боты в мессенджере Telegram — это специальные аккаунты, созданные для автоматической обработки и отправки сообщений. Они используются различными организациями, компаниями и университетами в разных целях. Пользователи могут общаться с ботами, отправляя сообщения через обычные чаты или группы. [7,8,9,10]

Одним из преимуществ использования чат-ботов является их возможность интеграции с другими сервисами. Например, боты могут отправлять комментарии или управлять "умным домом". Они также могут отправлять уведомления пользователям о различных событиях и действиях.

Боты представляют собой полезные инструменты, предоставляющие разнообразные функции. Например, они могут показывать погоду, выполнять перевод текстов или предоставлять информацию о предстоящих событиях по запросу пользователей. Благодаря этим возможностям, они становятся полезными помощниками, предоставляющими полезную информацию и решающими различные задачи по запросу пользователей.

Еще одним вариантом использования ботов являются игры. Боты могут предложить пользователям одиночные или многопользовательские игры, проводить викторины и т.д.

Боты также могут быть полезными в социальных сервисах. Например, они могут помогать пользователям находить собеседников на основе общих интересов и увлечений.

Чат в Telegram представляет собой беседу нескольких пользователей, где они могут обмениваться сообщениями, изображениями, документами или музыкальными файлами. Создатель чата может добавить бота в беседу. Однако стоит отличать чаты от каналов, поскольку каналы предназначены для публикации материалов, а не для общения.

Роботы в Telegram не требуют указания номера телефона при создании и играют роль интерфейса к сервису, работающему на удаленном сервере. Они имеют ограниченное место на серверах, и все сообщения будут удалены после обработки.[11]

В общем, боты в Telegram предоставляют широкий спектр функций и возможностей, позволяющих пользователям взаимодействовать с различными сервисами, играть, получать информацию и наслаждаться социальными сервисами, упрощая и обогащая их коммуникацию и взаимодействие.

Они действительно предоставляют удобный способ автоматизации обработки сообщений и взаимодействия с пользователями.

Боты в Telegram используются для различных целей, начиная от предоставления информации и услуг до развлекательных и игровых функций. Они могут быть интегрированы с другими сервисами и системами, позволяя отправлять уведомления, получать данные или даже управлять умным домом.

Команды и сообщения, отправляемые ботам, позволяют пользователям взаимодействовать с ботами и получать ответы на свои запросы. Кастомизированные клавиатуры и интерфейсы облегчают пользовательский опыт и позволяют ботам предоставлять более сложные функции.

Режим приватности в ботах по умолчанию обеспечивает конфиденциальность общения пользователей. Они могут отключить этот режим, если им необходимо разрешить боту получать все сообщения. Статус

режима приватности виден всем участникам группы, чтобы они могли быть осведомлены о том, какие данные получает бот.

Механизм внешнего связывания позволяет передавать дополнительные параметры и данные при запуске бота. Это может быть полезно для связи с внешними сервисами и получения дополнительной информации или авторизации пользователя.

В целом, боты в Telegram предоставляют широкий спектр возможностей и гибкость взаимодействия с пользователями, что делает их полезными инструментами для различных организаций и компаний.

Пример взаимодействие чат-бота по покупке билетов с пользователями по средствам Webhook представлен на рисунке 6.[12]

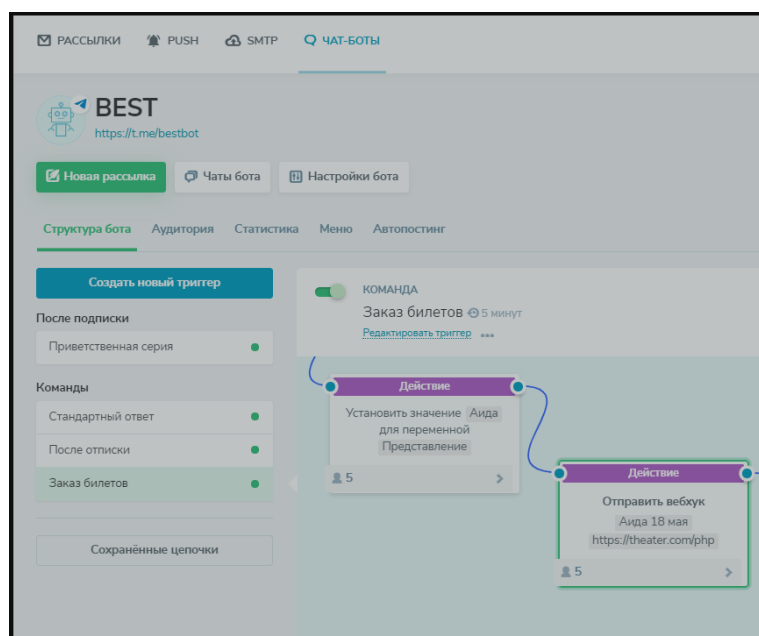


Рисунок 6 взаимодействие Телеграмм-чат-бота с пользователями

2.1 Интерфейс телеграмм бота

Основные элементы интерфейса чат-ботов универсальны, но каждое приложение для обмена сообщениями имеет свои особенности. Согласно документации Телеграмм API, существуют два способа взаимодействия бота с серверами Телеграмм:

1. Метод "getUpdates" предполагает, что бот постоянно обращается к серверу Телеграмм для проверки наличия новых сообщений. Этот подход

требует постоянной активности бота и запросов к серверу, чтобы получить новые сообщения,

2. Метод "setWebhook" предлагает обратный подход, где сервер Телеграмм отправляет новые сообщения боту по мере их поступления. Для этого необходимо наличие веб-сервера или аналогичного механизма, способного принимать входящие сообщения. Кроме того, для обеспечения безопасной связи через протокол HTTPS, требуется наличие SSL-сертификата.

Рисунок 7 иллюстрирует разницу между этими двумя способами взаимодействия. Хотя метод "setWebhook" может быть более эффективным и удобным для всех участников процесса, он также требует дополнительных компонентов, таких как веб-сервер и SSL-сертификат.

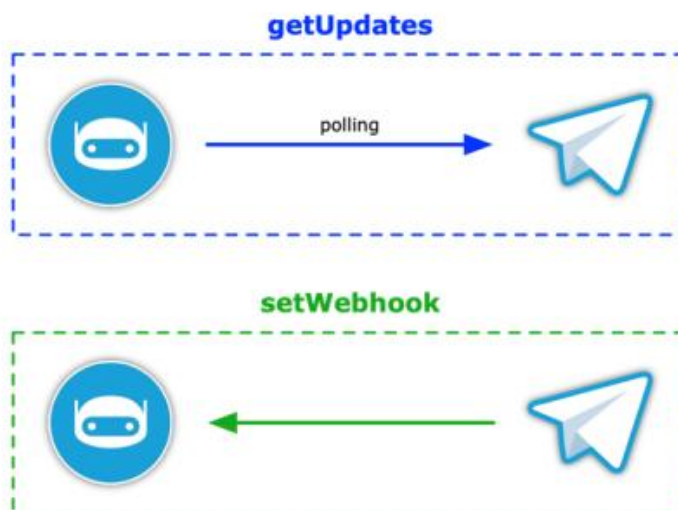


Рисунок 7 способы взаимодействия с ботом

В приложении, которое было разработано, используется веб-сервер, который состоит из серверного программного обеспечения Телеграмм бота. Он обеспечивает обработку сообщений от пользователей в режиме реального времени, предоставление пользователю мгновенных вариантов ответа и обработку решений пользователей с последующей отправкой ответов. Кроме того, веб-сервер взаимодействует с базой данных и выполняет запрошенные пользователем команды.

Таким образом, веб-сервер и Телеграмм бот составляют основные компоненты приложения, которые позволяют обеспечить взаимодействие с пользователями и обработку их запросов.[13]

2.2 Успешный опыт в использовании Telegram университетами

Возникшее предложение о внедрении чат-ботов и платформы Telegram в качестве коммуникационных инструментов нашего университета является весьма инновационным и уникальным. В маркетинговой литературе часто рассматриваются различные сегменты потребителей в зависимости от их скорости и готовности принять новые продукты или услуги:

- Группа новаторов, открытых для экспериментов и внедрения инноваций,
- Активные пользователи, которые быстро принимают новые технологии,
- Раннее большинство, которое становится основной массой пользователей после первых двух групп,
- Позднее большинство, которое присоединяется к использованию продукта или услуги с некоторой задержкой,
- Консерваторы, более консервативные пользователи, которые остаются несколько отстраненными от нововведений.[14]

Понимание сути и поведения каждой из этих групп потребителей имеет особое значение в маркетинге. Следуя модели Роджерса, в которой группы потребителей описываются по отношению к инновациям, можно утверждать, что в нашей стране высшие учебные заведения склонны принадлежать к позднему большинству, в то время как региональные вузы иногда проявляют запоздалую реакцию. Модель Роджерса представлена на рисунке 8.

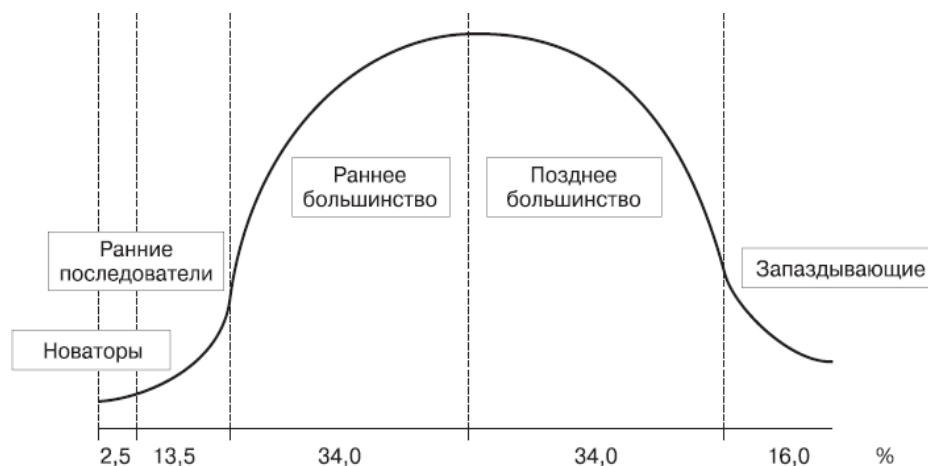


Рисунок 8 - Кривая распределения склонности к инновациям

Однако несмотря на то, что данное предложение нов образное и величественное, важно помнить, что оригинальность и инновационность обладают возможностью внести важный вклад в развитие и прогресс университета, улучшая его коммуникационные процессы и устанавливая более эффективные взаимодействия с различными группами пользователей.

Существуют положительные примеры внедрения инновационных решений в повседневную жизнь студентов, как в случае с Пензенским государственным университетом (ПГУ). Здесь пользователи получили возможность воспользоваться уникальной технологией, предоставленной Электронной информационно-образовательной средой ЭИОС ПГУ. Эта платформа обеспечивает доступ всех участников образовательного процесса к разнообразным электронным информационным и образовательным ресурсам. Она объединяет информационные и телекоммуникационные технологии, обеспечивая эффективное взаимодействие обучающихся, преподавателей и административного персонала.[15]

Основным функционалом ЭИОС ПГУ является обеспечение информационной прозрачности университета в соответствии с современным законодательством. Этот функционал ориентирован в первую очередь на пользователей, активно использующих персональные компьютеры в своей повседневной работе и развлечениях. Однако, чтобы получить доступ к

необходимой информации, пользователю необходимо посетить веб-сайт, пройти верификацию и искать информацию среди множества разделов.

В целях улучшения коммуникационных процессов ПГУ внедрил новую платформу - канал в Telegram. На этом новом ресурсе пользователи получают оперативные новости, объявления и напоминания о важных событиях, включая предстоящее 80-летие университета. Каналы в Telegram являются одной из самых динамично развивающихся медийных площадок. Пользователи отмечают, что главное отличие этой платформы заключается в изоляции от информационного шума, присущего социальным сетям, а также в высокой степени персонализации.

В основном, на канале публикуются анонсы событий, важные новости, фотографии и развлекательный контент, отличающийся от публикаций на официальном сайте ПГУ и в сообществах в социальных сетях. Управление информационной политики отмечает, что данная площадка ориентирована, прежде всего, на студенческую аудиторию, которая всегда имеет смартфон под рукой. Благодаря новому ресурсу студенты могут быть в курсе событий, таких как отмена или перенос занятий, среди первых получая соответствующую информацию.

3 Проектирование чат-бота Telegram

3.1 Концептуальная модель программного продукта

Проектирование чат-бота Telegram

Модель-Представление-Контроллер (MVC) — это эффективная архитектурная схема, применяемая для разделения компонентов приложения: данных, логики управления и пользовательского интерфейса. Основная идея заключается в том, чтобы обеспечить независимость каждой части и возможность их независимого развития и модификации. Модель является ключевым компонентом в MVC и отвечает за предоставление данных и методов для работы с ними, включая запросы к базе данных и проверку их правильности. Что касается объектно-реляционного отображения (ORM), это мощная технология, которая облегчает взаимодействие между объектно-

ориентированными языками программирования и базами данных. Она позволяет создавать "виртуальные" объектные базы данных, упрощая и ускоряя доступ к данным и устраняя необходимость писать прямые запросы SQL. ORM существуют в различных вариантах, включая проприетарные и свободные реализации, и выбор конкретной зависит от потребностей проекта и предпочтений разработчика. [16,17]

Опишем ключевые сущности, с которыми мы будем работать:

- Профиль студента ПГУ (ФИО, дата рождения, контакты, факультет и никнейм),
- Редактирование профиля студента и абитуриента ПГУ (Редактирование основных данных ФИО, дата рождения, контакты, факультет и никнейм),
- Посещаемость и успеваемость студента (Личный кабинет ЭИОС ПГУ),
- Институты ПГУ (Все институты);
- Информация для поступающих в 2023 году (Результаты прошлых лет, основная информация для приема в 2023 году),
- Наука ПГУ (полезные ссылки, статьи ПГУ, основные ресурсы);
- Контакты ПГУ (E-mail адрес, телефон приемной комиссии ПГУ, юридический адрес, социальные сети),
- Новости ПГУ (Последние три новости, которые были опубликованы на сайте ПГУ).

После описания моделей создали необходимые таблицы в базе данных, изображенных на Рисунке 9.

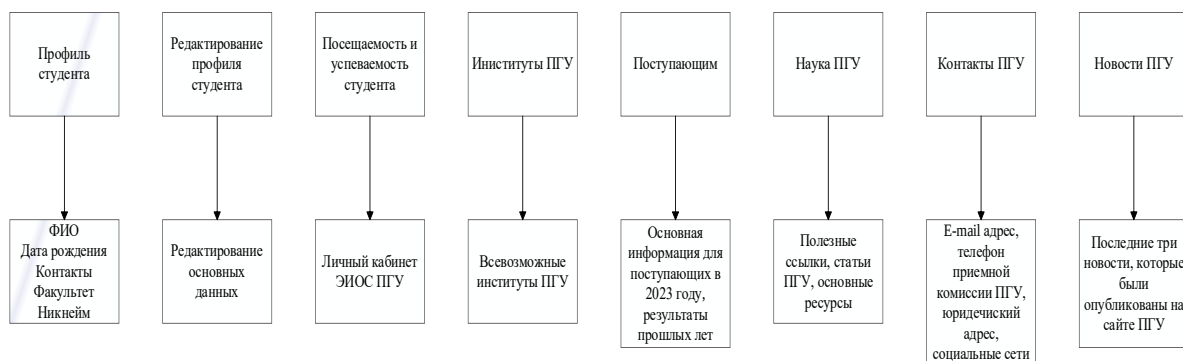


Рисунок 9 концептуальная модель программного продукта

4 Анализ инструментов разработки

В работе по созданию чат-бота были использованы Python 3.11.1, PyCharm 2023.3.1, DataGrip 2022.2.1, Telegram Bot API, библиотека pyTelegramBotAPI 4.9.0, библиотека aiogram, библиотека datetime СУБД Sqlite 3.4.4.

4.1 Язык программирования Python

Python является скриптовым языком программирования, разработанным Гвидо ван Росс умом с целью создания языка, который был простым в изучении и использовании. В настоящее время Python пользуется широкой популярностью и находит применение во многих сферах, включая: [18,19]

- Разработку прикладного программного обеспечения: Python широко используется для создания различных приложений, включая настольные программы, мобильные приложения и серверные приложения,
- Разработку веб-приложений: Python обладает разнообразными фреймворками, вроде Django и Flask, которые значительно облегчают процесс создания веб-приложений и предоставляют мощные инструменты для работы с веб-технологиями,
- Встроенный скриптовый язык в играх: Python активно применяется в игровой индустрии для разработки скриптов и модификаций игровых процессов,
- Математические расчеты: Python обладает обширной библиотекой инструментов, включая такие популярные пакеты, как NumPy и SciPy,

которые обеспечивают высокую производительность при выполнении научных и математических вычислений,

Python поддерживает различные парадигмы программирования, включая структурное, объектно-ориентированное, функциональное, императивное и аспектное-ориентированное программирование. Благодаря этому Python является гибким языком, который можно применять для решения широкого спектра задач,

Python находит применение как в крупных компаниях, вроде Google, Intel, Cisco, Hewlett-Packard, Seagate, Qualcomm, IBM, так и в небольших проектах, разрабатываемых студентами и инженерами. Он применяется для создания приложений, тестирования аппаратного обеспечения, обработки данных, научных вычислений и других задач,

Синтаксис языка Python отличается простотой и понятностью. Важную роль играют отступы, которые определяют блоки кода, в отличие от фигурных скобок. Такой подход делает код на Python более читаемым и помогает избежать некоторых ошибок,

Python предоставляет множество функций и структур данных, включая списки, словари и строки. Кроме того, он обеспечивает мощные средства для обработки и управления объектами. Python также поддерживает создание модулей, классов и исключений, что позволяет создавать крупные системы,

В целом, Python является мощным и гибким языком программирования, который находит применение в разных областях. Его простой синтаксис, обширная библиотека инструментов и активное сообщество разработчиков делают его предпочтительным выбором для создания разнообразных приложений и решения различных задач,

Кроме того, Python обладает высокой переносимостью, что означает, что программы, написанные на Python, могут работать на разных операционных системах, таких как Windows, macOS и Linux, без необходимости внесения значительных изменений. Это облегчает разработку и развертывание

программного обеспечения на разных платформах. Далее на рисунке 10 предоставлена простейшая функция на языке Python

```
def add(x, y):  
    return x + y
```

Рисунок 10 простейшая функция на языке Python

Важно отметить, что Python имеет огромное сообщество разработчиков и активную поддержку со стороны его создателей и разработчиков. Это обеспечивает доступность обучающих материалов, руководств, форумов и онлайн-ресурсов, которые помогают новичкам освоить язык и развиваться в нем.

В заключение, Python — это мощный, гибкий и широко используемый язык программирования, который применяется во многих областях. Его простой синтаксис, богатый набор инструментов и активное сообщество разработчиков делают его привлекательным выбором для создания разнообразных приложений и решения различных задач.

4.2 Среда разработки PyCharm

PyCharm – это не только ведущая интеллектуальная Python IDE, но и истинное воплощение творческого вдохновения, предлагающее разработчикам полный арсенал инновационных инструментов для максимально продуктивного творчества на языке Python. Независимо от того, выбираете ли вы свободную от платы PyCharm Community Edition или жаждете раскрыть все грани своего потенциала с помощью расширенной PyCharm Professional Edition, у вас открываются неограниченные возможности для качественного и увлекательного программирования.[20]

PyCharm восхищает не только возможностью инспектировать код в режиме реального времени, но и умением предлагать авто дополнение, опираясь на динамические данные, полученные в процессе выполнения кода. Но это еще не все – с помощью этой магической IDE вы сможете легко наигрывать по своему коду, осуществлять масштабные рефакторинги и даже

проводить анализ ключевых отличий между Community Edition и Professional Edition, чтобы точно определить, какой вариант отвечает вашим потребностям и творческим амбициям.

PyCharm Community Edition — это идеальный выбор для энтузиастов, которые хотят полностью сосредоточиться на разработке на языке Python. Это бесплатная версия с открытым исходным кодом, которая обладает всеми необходимыми инструментами для эффективной работы разработчика.

PyCharm Community Edition предлагает мощные функции отладки, рефакторинга и инспекции кода, которые помогают вам создавать высококачественные программы. Он также обладает удобной интеграцией с системами контроля версий, что позволяет вам легко отслеживать изменения в вашем проекте и сотрудничать с другими разработчиками.

Навигация по проекту в PyCharm Community Edition интуитивно понятна и удобна, что позволяет быстро находить нужные файлы и классы. Вы также можете настроить пользовательский интерфейс по своему вкусу, чтобы максимально комфортно работать в IDE.

PyCharm Community Edition поддерживает горячие клавиши Vim, что позволяет вам использовать привычные команды Vim внутри PyCharm, добавляя дополнительный комфорт и эффективность вашей работе.

В целом, PyCharm Community Edition — это мощная и гибкая среда разработки, которая поможет вам сосредоточиться на создании качественного кода на языке Python.

PyCharm Professional Edition, в свою очередь, предлагает полный комплект возможностей для мастеров разработки на Python, включая разработку сложных веб-приложений с помощью различных фреймворков и поддержку многих языков программирования. С поддержкой фреймворков Django, Flask, Google App Engine, Pyramid, web2py и сыпучего ассортимента языков, включая JavaScript, CoffeeScript, TypeScript, CSS, Cython и другие, Professional Edition обеспечивает вам безграничные возможности для

творческой реализации самых сложных проектов. Далее на рисунке 10 предоставлен интерфейс среды разработки PyCharm.

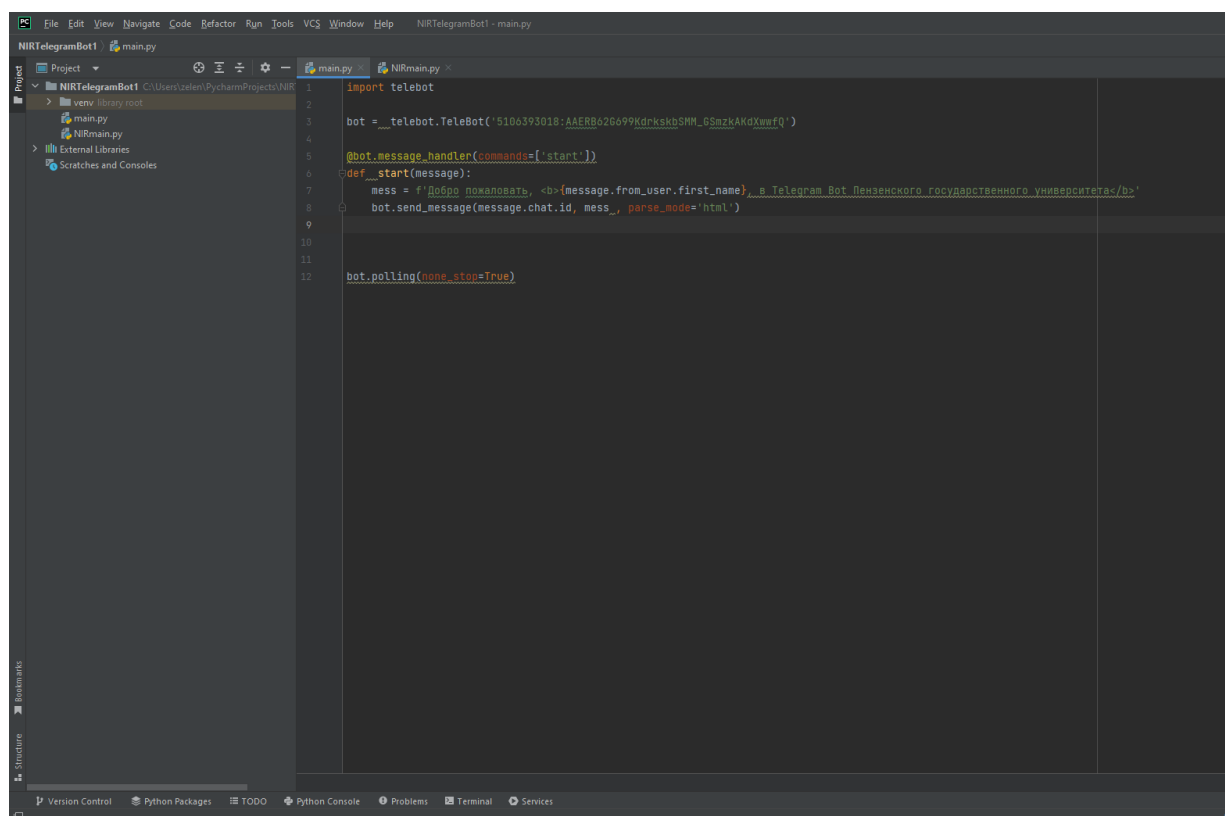


Рисунок 10 Интерфейс PyCharm

4.3 Анализ выбора базы-данных для хранения пользователей.

4.3.1 База данных SQLite

Для обеспечения надежного хранения данных, необходимых для безупречной работы нашего чат-бота, мы приняли решение выбрать базу данных SQLite. SQLite — это компактная интегрированная реляционная база данных с открытым исходным кодом. Она предлагает широкий набор функций и возможностей, что делает ее идеальным выбором для наших потребностей.

Одним из главных преимуществ SQLite является его способность обрабатывать сложные запросы, поддерживать внешние ключи, триггеры, изменяемые представления, а также обеспечивать транзакционную целостность и много версионность. Это позволяет нам эффективно управлять и манипулировать данными внутри нашего чат-бота.[21]

Кроме того, SQLite предоставляет возможность расширения его функциональности путем создания собственных типов данных, функций,

операторов, агрегатных функций, методов индексирования и даже процедурных языков. Это позволяет нам настроить базу данных в соответствии с уникальными потребностями и требованиями нашего проекта.

Управление базами данных является важной составляющей разработки динамических веб-продуктов, так как позволяет систематизировать все необходимые данные и обеспечивает быстрый доступ и оптимизацию работы нашего приложения или сайта. В контексте нашего проекта мы провели анализ различных популярных СУБД, таких как MySQL и SQLite, и выбрали SQLite как наиболее подходящую для наших целей.

SQLite предоставляет чисто реляционную модель баз данных, где данные хранятся в таблицах и связаны между собой согласно реляционной модели. Она поддерживает различные типы данных, включая NULL, INTEGER, REAL, TEXT и BLOB, что позволяет нам гибко работать с разнообразными данными, включая целые числа, числа с плавающей запятой, текстовые строки и двоичные данные.

Среди преимуществ SQLite следует отметить его файловую структуру, благодаря которой вся база данных хранится в одном файле, что упрощает ее перемещение. Также стоит отметить, что SQLite использует стандартизированный язык запросов SQL, что облегчает работу с базой данных и обеспечивает совместимость с другими системами управления базами данных.

SQLite идеально подходит для разработки и тестирования, поскольку она предоставляет достаточно функциональности и простоты использования для работы с одним файлом и связанной библиотекой. Это позволяет нам эффективно разрабатывать и проверять функциональность нашего чат-бота, а также обеспечивает масштабируемое решение для будущего развития проекта.

SQLite является легковесной и встроенной системой управления базами данных, которая стала популярной благодаря своей простоте и эффективности. Однако у нее также есть некоторые ограничения и недостатки. Вот некоторые из них:

1. Отсутствие пользовательского управления: в отличие от продвинутых СУБД, SQLite не предоставляет обширных функций управления пользователями. Он не обеспечивает возможность управлять связями в таблицах в соответствии с привилегиями,

2. Невозможность дополнительной настройки: SQLite не предоставляет возможности для дополнительной настройки и оптимизации производительности. Вы не можете внести изменения, чтобы сделать его более эффективным для конкретных потребностей,

3. Сложность освоения: Понимание принципов работы SQLite требует знания баз данных. Для некоторых пользователей может быть сложно освоить и начать использовать эту СУБД,

4. Стоимость: Сама SQLite может быть бесплатной, но обслуживание базы данных, приобретение стороннего программного обеспечения и размещение больших баз данных на хорошем хостинге могут потребовать определенных затрат,

5. Увеличение объема данных: при использовании SQLite для хранения файлов и данных объем базы данных может существенно возрасти. Файлы имеют дополнительные функции, находятся в логической связи и вызываются скриптами, что требует дополнительного пространства на сервере.

Несмотря на эти недостатки, SQLite все же обладает рядом преимуществ, таких как экономия памяти, избежание дублирования информации, поддержка санкционированного доступа к базе данных и простота использования. Важно учитывать свои потребности и требования при выборе СУБД для конкретного проекта. Далее на рисунке 11 предоставлена развернутая база данных database в SQLite.

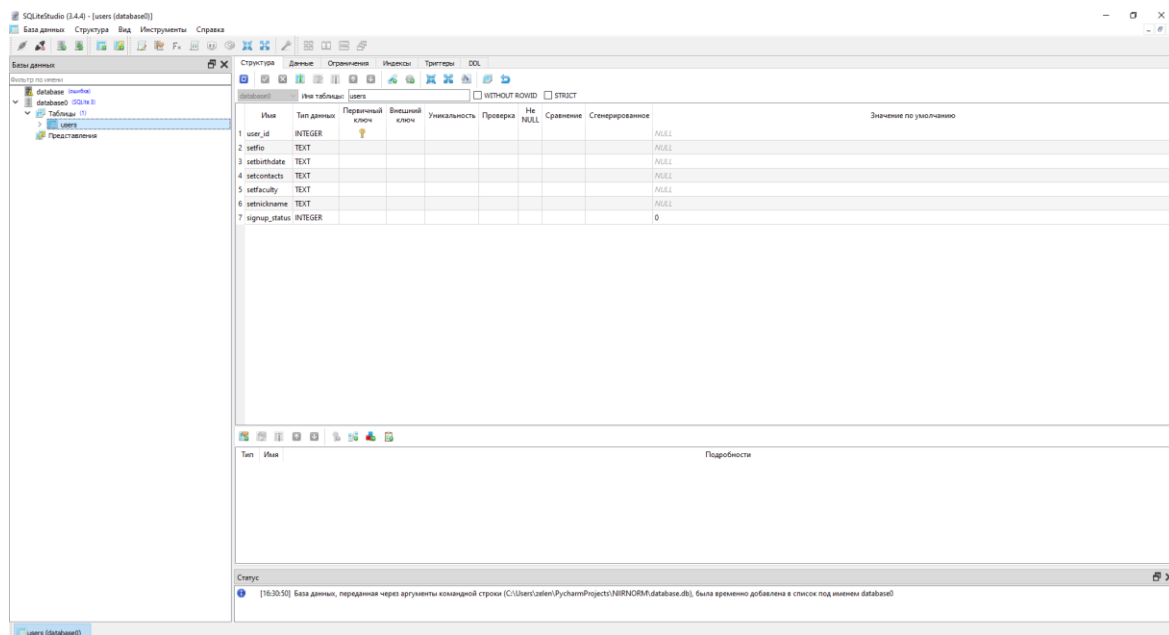


Рисунок 11 Развернутая database в SQLite

4.3.2 База данных MySQL

MySQL - популярная СУБД, которая используется для разработки разнообразных приложений, включая веб-приложения и сложное программное обеспечение. Ее функциональность постоянно совершенствуется, и разработчики регулярно выпускают новые версии ПО. MySQL находится под управлением компании Oracle с момента приобретения компании Sun Microsystems, включая MySQL, в 2010 году.

MySQL была разработана компанией MySQL AB в 1995 году, а затем использовались языки программирования C, C++ и HTML для создания стабильной и быстрой СУБД с непрерывной поддержкой. Сегодня MySQL включает в себя сервер, базу данных и скриптовый язык программирования, и является частью так называемых "LAMP-наборов" (Linux, Apache, MySQL, PHP/Perl/Python), которые широко используются для разработки веб-приложений.

Одним из явных преимуществ MySQL перед конкурентами является его широкая распространенность. Популярное ПО обычно имеет большую базу пользователей, что упрощает работу с ним. Баги быстро обнаруживаются и исправляются, а развитие MySQL активно поддерживается сообществом разработчиков. Всегда появляются новые плагины и расширения для MySQL.

4.3.3 База данных PostgreSQL

PostgreSQL представляет собой впечатляющую объектно-реляционную систему управления базами данных (ОРСУБД), отличающуюся высокой гибкостью и множеством функциональных возможностей. Эта мощная и надежная СУБД разработана "от программистов для программистов" и является одной из наиболее профессиональных и мощных систем на современном рынке баз данных. PostgreSQL произошла из долголетнего университетского проекта, который получил название Postgres, и продолжает активно развиваться и поддерживаться качественным сообществом разработчиков.[22]

Одной из главных преимуществ PostgreSQL является его универсальность в работе с различными типами данных и поддержка пользовательских объектов и поведения. Благодаря этому, PostgreSQL способен эффективно хранить и обрабатывать сложные структуры данных, включая геометрические типы, массивы значений и JSON. Кроме того, PostgreSQL обеспечивает соблюдение требований ACID (атомарность, согласованность, изолированность и долговечность) и обладает высокой ссылочной и транзакционной целостностью.

Одной из ключевых особенностей PostgreSQL является его масштабируемость. Система успешно применяется в крупных проектах обработки Big Data, благодаря своей стабильности и высокой производительности. PostgreSQL также предлагает несколько вариантов пользовательского интерфейса, что делает его удобным для настройки и персонализации в соответствии с потребностями проекта.

Благодаря своей свободной лицензии, PostgreSQL доступна для использования, модификации и распространения бесплатно, что делает ее привлекательным выбором для широкого круга задач - от коммерческих проектов до учебных целей. Эта СУБД также отличается активной поддержкой и дружественным сообществом разработчиков, которые постоянно работают над улучшением и расширением ее функциональности.

5 Разработка программного продукта

5.1 Анализ API и библиотек для создания чат-бота Telegram

5.1.1 Telegram Bot API

Telegram Bot API является универсальным инструментом для создания чат-ботов на платформе Telegram. Он поддерживает множество языков программирования, но для удобства разработки был выбран язык Python, благодаря его гибкости и широкой поддержке сообщества.[23,24]

Работа с API осуществляется через отправку HTTPS-запросов. Все запросы к Telegram Bot API должны быть защищены протоколом HTTPS и иметь следующий формат: `https://api.telegram.org/bot<token>/METHOD_NAME`, где `<token>` представляет собой уникальный токен вашего бота, а `METHOD_NAME` - название метода API, который вы хотите вызвать.

Bot API предоставляет HTTP-интерфейс для взаимодействия с ботами. Каждый бот — это особый аккаунт, созданный для автоматической обработки и отправки сообщений. Для получения обновлений от бота можно использовать два основных подхода:

- Long polling (длительное ожидание) - приложение периодически опрашивает серверы Telegram на предмет наличия новых обновлений для бота. Задержка между запросами по умолчанию составляет 100 мс, но может быть настроена в соответствии с требованиями вашего приложения,
- Webhook (перехватчик) - в этом режиме серверы Telegram сами оповещают ваш сервер о появлении новых обновлений. Пользователь должны настроить веб-сервер и указать URL-адрес, на который Telegram будет отправлять обновления. На рисунке 12 предоставлен принцип работы чат-бота на платформе Telegram.

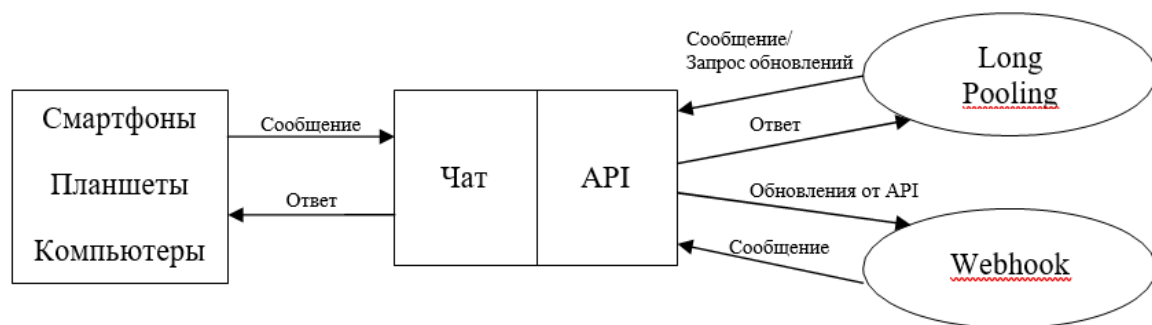


Рисунок 12 Принцип работы чат-бота на платформе Telegram

Для взаимодействия с API Telegram Bot доступны различные методы передачи параметров, что обеспечивает гибкость в работе с ним. Один из методов — это передача параметров в URL запроса. Этот метод часто используется для простых запросов, когда данные передаются в строке запроса.

Другой метод - использование форматов `application/x-www-form-urlencoded` и `application/json`. Формат `application/x-www-form-urlencoded` используется для передачи данных в виде пар ключ-значение, а формат `application/json` позволяет передавать сложные структуры данных в формате JSON. Эти методы особенно полезны при выполнении запросов с более сложной логикой.

Также для загрузки файлов используется формат `multipart/form-data`. Этот формат позволяет передавать файлы вместе с другими параметрами запроса, что особенно полезно при создании ботов, требующих обработки медиафайлов.

Python является популярным выбором для разработки ботов на основе Telegram Bot API. В языке Python существуют богатые библиотеки, такие как `python-telegram-bot`, которые облегчают процесс разработки и предоставляют удобные функции для работы с API. Это позволяет разработчикам создавать ботов на Python с минимальными усилиями и получать максимальную гибкость в настройке функциональности бота. Благодаря этим библиотекам

разработчики могут сосредоточиться на логике бота, не заботясь о низкоуровневых деталях взаимодействия с API.3.1.5

5.1.2 Библиотека pyTelegramBotAPI, Aiogram, Telebot

В море возможностей, предоставляемых стандартными инструментами PyCharm, иногда встречаются задачи, которые требуют использования сторонних библиотек. Эти библиотеки не только приносят пользу, но и являются неотъемлемой частью разработки, позволяя избежать сложных конструкций и упростить синтаксис. В нашем проекте сторонние библиотеки играют важную роль, так как без них невозможно получить необходимые данные с серверов, с которыми мы взаимодействуем.

Одной из ключевых составляющих работы бота с людьми являются HTTP-запросы. Однако обработка "сырых" данных может быть непростой задачей. Для облегчения процесса отправки и получения запросов мы можем использовать библиотеку pyTelegramBotAPI, которая берет на себя все трудности данного процесса, позволяя нам сконцентрироваться непосредственно на разработке логики бота.

Также стоит обратить внимание на библиотеку aiogram, которая представляет собой современное решение, пользующееся популярностью. Она работает с асинхронным подходом, что позволяет боту продолжать свою работу, не ожидая ответа от пользователя. Библиотека aiogram обладает обширной документацией и активным русскоязычным сообществом.

Перед выбором оптимальной библиотеки для работы с Telegram Bot API был проведен анализ различных вариантов. В результате анализа мы выбрали библиотеку pyTelegramBotAPI, которая предоставляет удобные функции и инкапсулирует все вызовы API в одном классе

Класс TeleBot позволяет отправлять сообщения различных типов (таких как текстовые сообщения или документы) с помощью функций send_xyz. Кроме того, TeleBot предоставляет несколько способов для прослушивания входящих сообщений.

Для установки библиотеки pyTelegramBotAPI достаточно выполнить простую команду "pip install pytelegrambotapi" в командной строке. После установки мы можем начать использовать функциональность библиотеки и сосредоточиться на разработке бота, не беспокоясь о низкоуровневых деталях обработки запросов и данных.

5.2 Разработка чат-бота Telegram

В предыдущих разделах выпускной квалификационной работы был проведен анализ существующих методов взаимодействия между абитуриентами и приемной комиссией университета, а также был выбран язык программирования и среда разработки для создания чат-бота в мессенджере Telegram.

В рамках работы над ВКР была разработана последовательность этапов для создания чат-бота, которая включает в себя:

1. Изучение теоретической базы и ознакомление с существующими методами взаимодействия абитуриентов, студентов и приемной комиссии ПГУ,
2. Детальный анализ инструментов и методов разработки ботов для мессенджера Telegram,
3. Разработка алгоритма действий для создания чат-бота,
4. Написание кода чат-бота на языке программирования Python,
5. Внедрение функциональности, команд и административных возможностей в чат-бот,
6. Освоение и создание базы данных для хранения информации о ПГУ.

Целью работы является создание эффективного чат-бота, который будет обеспечивать удобное и быстрое взаимодействие с абитуриентами и студентами ПГУ, а также обладать необходимыми функциями и возможностями для обеспечения эффективного администрирования и хранения информации о университете.

5.2.1 Регистрация чат-бота для Telegram Bot API

При разработке приложения первым шагом является регистрация чат-бота у специального сервиса "BotFather". Регистрация начинается с отправки команды `/newbot`, после чего требуется ввести название чат-бота с обязательным условием: в конце названия должно быть указано "Bot" или "_bot". Например, "MagistrHelpBot_bot". Если все условия удовлетворены, "BotFather" выдаст токен - специальный набор символов для доступа к HTTP API Telegram Bot, и URL-адрес для доступа к чат-боту. Пример регистрации чат-бота представлен на рисунке 13.[25]

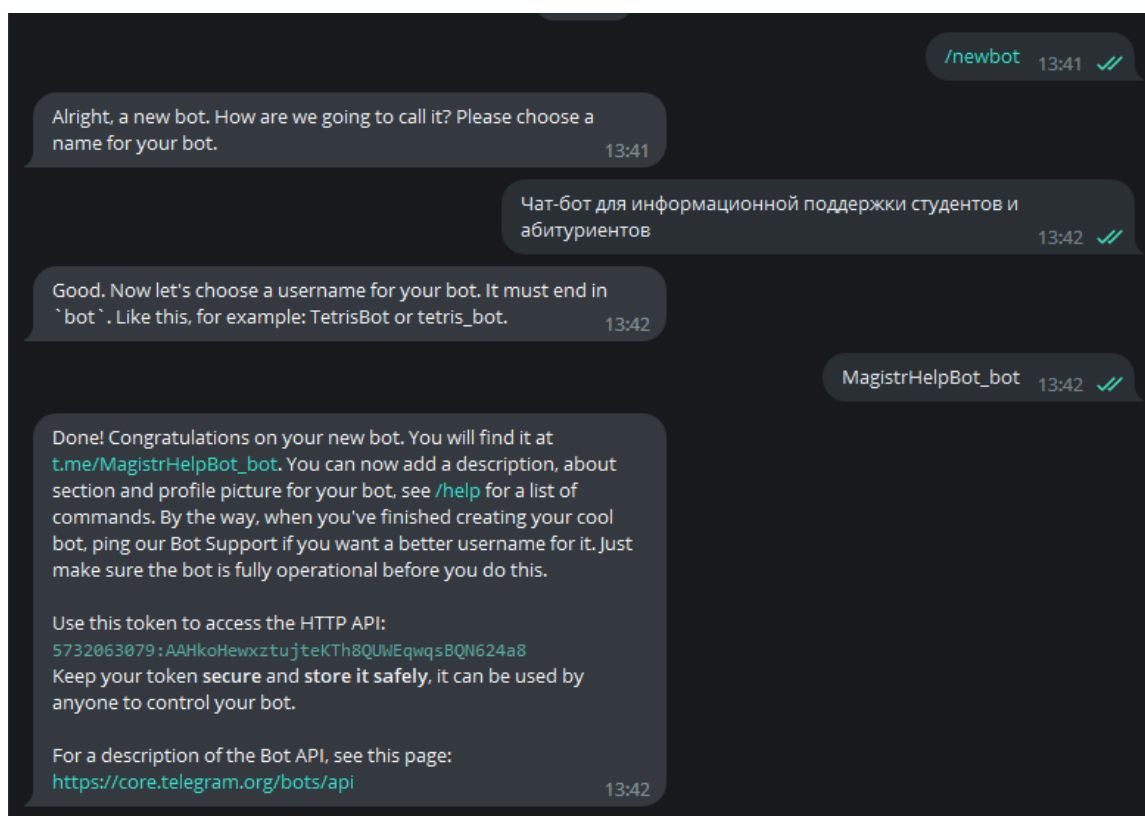


Рисунок 13 – Пример регистрации чат-бота

Для настройки дополнительных параметров, таких как иконка чат-бота, приветственное сообщение и описание чат-бота, а также для удаления существующих чат-ботов, существуют следующие команды предоставлены на таблице 2.

Таблица 2 - Доступные команды для изменения чат-ботов

Команда	Описание
/setname	Изменяет текущее имя чат-бота
/setdescription	Задаёт текст, который будет отображаться при первом открытии чат-бота
/setabouttext	Задаёт текст, отображаемый в разделе «О чат-боте»
/setuserpic	Задаёт выбранную изображение для чат-бота
/setcommands	Создаёт список доступных команд для чат-бота
/deletebot	Удаляет выбранного чат-бота

Кроме того, существуют команды, позволяющие выводить неизменяемые параметры (токен) и устанавливать значения для дополнительной настройки чат-бота, как указано в таблице 3.

Таблица 3 - Доступные команды для дополнительной настройки чат-бота

Команда	Описание
/token	Возвращает ранее полученный токен для выбранного бота
/revoke	Аннулирует токен доступа к боту
/setinline	Включает и выключает возможность вызова бота из других чатов
/setinlinegeo	Включает и выключает возможность передачи местоположения

	бота из другого чата
Команда	Описание
/setinlinefeedback	Позволяет получать информацию о количестве выбранных пользователем команд
/setjoingroup	Определяет, может ли бот быть добавлен в групповые диалоги
/setprivacy	Включает режим конфиденциальности, при котором бот обрабатывает информацию отдельно для каждого пользователя в чате

После настройки на стороне Telegram и получения токена можно приступить к разработке программной части чат-бота.

Когда вы получите токен для вашего чат-бота, вам потребуется проверить его для подтверждения его действительности и настройки параметров профиля. Этот процесс позволяет убедиться, что ваш чат-бот связан с правильным аккаунтом и готов к разработке функциональности.[26]

Чтобы проверить токен, нужно выполнить следующие шаги:

Откройте веб-браузер и введите следующий URL: "https://api.telegram.org/bot<your_bot_token>/getMe". Замените "<your_bot_token>" на фактический токен вашего чат-бота.

Если токен действителен, вы увидите JSON-объект с информацией о вашем чат-боте, такой как его идентификатор, имя и другие данные.

Убедитесь, что информация о вашем чат-боте отображается корректно в полученном JSON-ответе.

Примеры регистрации чат-бота представлены на рисунках 14 и 15.

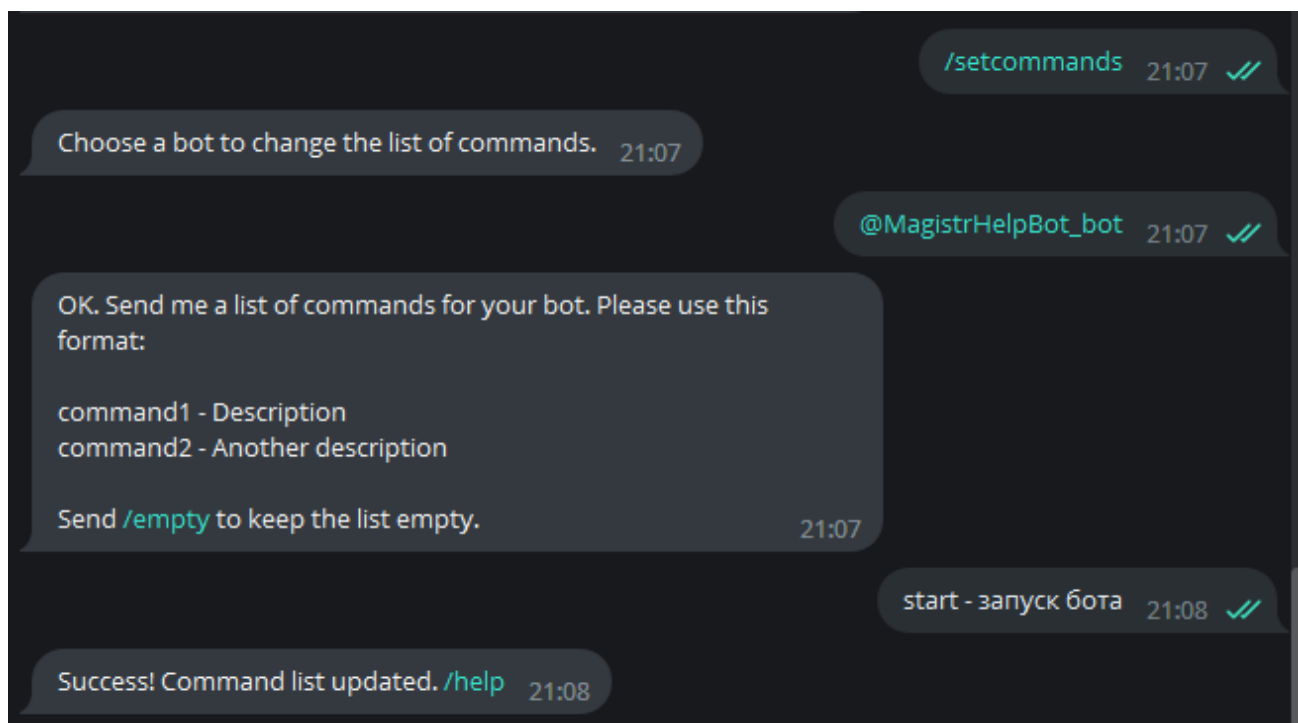


Рисунок 14 – Выполнение команды «/setcommands»

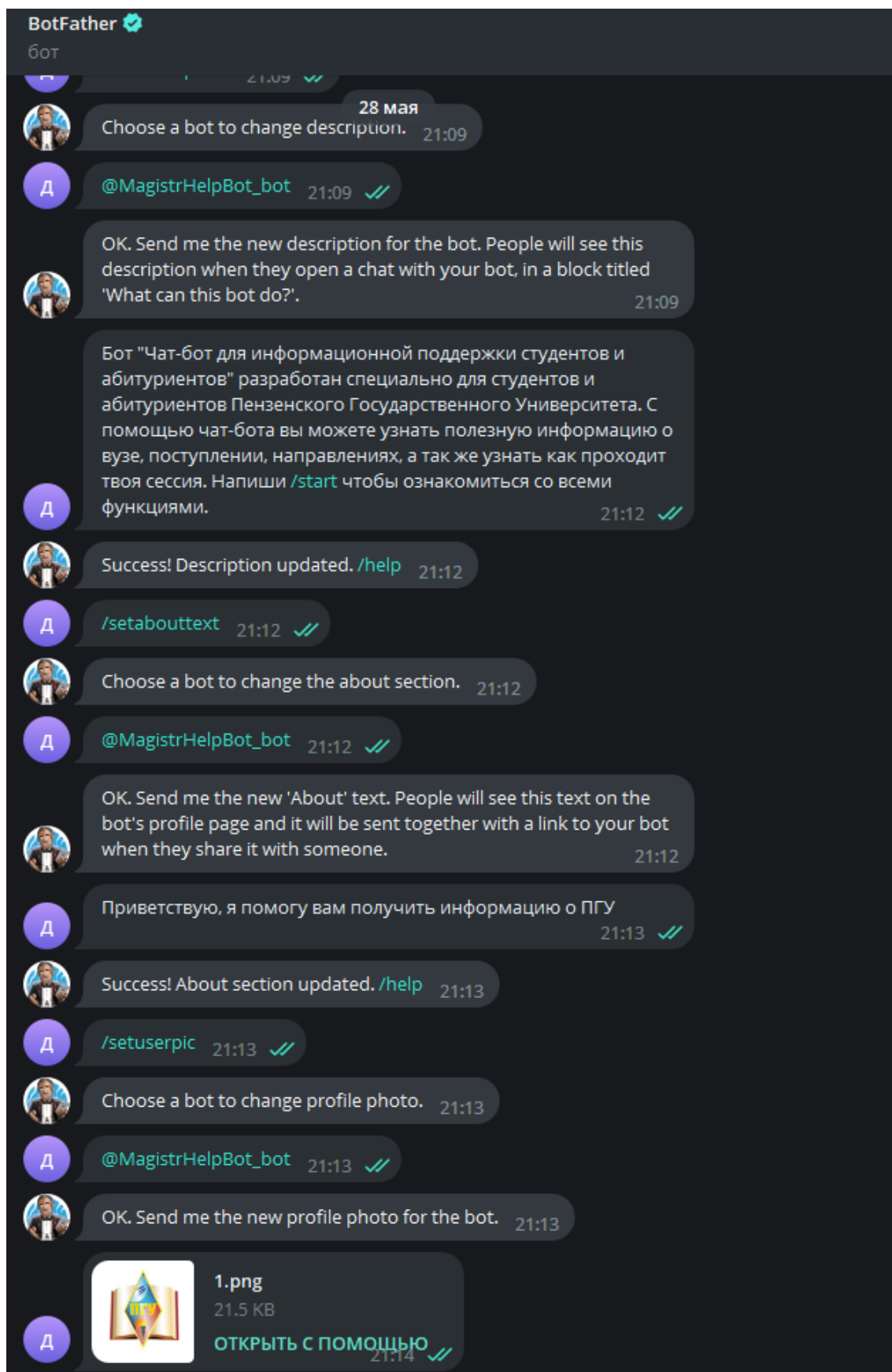


Рисунок 15 – Выполнения команд «/setdescription, /setabouttext, /setuserpic»

5.2.2 Реализация клиентской части чат-бота

Для реализации поставленных целей потребуется восемь различных меню:

- 1 Профиль студента,
- 2 Редактирование профиля,
- 3 Посещаемость и успеваемость студента,
- 4 Институты ПГУ,
- 5 Поступающим,
- 6 Наука,
- 7 Контакты ПГУ,
- 8 Новости ПГУ.

Основной файл является `main.py`, данный файл является скриптом на языке Python, использующим библиотеку `aiogram` для создания Telegram-бота. Рассмотрим основные компоненты:

`aiogram` - библиотека для работы с Telegram Bot API,

`logging` - модуль для настройки логирования,

`markups` - пользовательский модуль с различными клавиатурными разметками,

`db` - модуль для работы с базой данных,

`datetime` - модуль для работы с датой и временем,

`Requests` — это модуль для языка Python, который используют для упрощения работы с HTTP-запросами. Он удобнее и проще встроенного `Urllib` настолько, что даже в документации Python рекомендовано использовать `Requests`. [27]

Рассмотрим основные команды для инициализации бота и диспетчера:

`bot = Bot(token=TOKEN)` - создание объекта бота с использованием предоставленного токена,

`dp = Dispatcher(bot)` - создание диспетчера для обработки входящих сообщений.

Рассмотрим основные команды для обработчика сообщений:

@dp.message_handler(commands=['start']) - обработчик команды "/start", выполняющийся при запуске бота,

@dp.message_handler() - обработчик всех остальных сообщений.

Рассмотрим основные функции-обработчика сообщений:

Функция start обрабатывает команду /start и регистрирует пользователя в базе данных. Она запрашивает у пользователя его ФИО и устанавливает статус регистрации для последующего ввода информации,

Функция handle_text_message обрабатывает все остальные сообщения от пользователя. Она проверяет тип чата (должен быть "private") и обрабатывает различные варианты текстовых сообщений в зависимости от текущего статуса регистрации пользователя. Также, функция отвечает на различные команды, предоставляя информацию о профиле студента, редактировании профиля, посещаемости и успеваемости студента, новостях и других разделах,

В функции __main__ выполняется запуск бота с помощью метода executor.start_polling.

Рассмотрим основную точку входа в программы:

if __name__ == '__main__': - проверка, запущен ли файл напрямую, а не импортирован как модуль,

executor.start_polling(dp, skip_updates=True) - запуск бота и ожидание входящих обновлений.

Разработка мессенджеров и их функциональности продолжает привлекать все больше внимания с каждым днем. В основе этой популярности лежит возможность обмениваться сообщениями и медиаконтентом на расстоянии, что делает мессенджеры неотъемлемой частью нашей повседневной коммуникации. Вместе с этим развивается и технология чат-ботов, которые работают на платформах мессенджеров.

В Telegram появление кнопок было представлено вместе с Bot API еще в 2015 году. Эти кнопки представляют собой шаблоны сообщений, где текст, указанный на кнопке, будет отправлен в текущий чат при ее нажатии. Для

обработки нажатий кнопок, боту необходимо распознавать входящие текстовые сообщения.

В библиотеке aiogram, предназначенной для разработки Telegram-ботов, существует удобный механизм под названием "фабрика колбэков". Она позволяет создавать объекты Callback Data с определенным префиксом и дополнительными аргументами. При создании кнопок, эти аргументы могут быть использованы для связывания с соответствующими колбэками.

Таким образом, разработчики, использующие aiogram и Telegram Bot API, получают преимущество в создании кнопок и обработке пользовательских действий. Фабрика колбэков в aiogram облегчает этот процесс, позволяя гибко определять параметры и логику кнопок, создавая удобный и интерактивный пользовательский опыт.

Пример реализации кнопки-клавиатуры одного из меню чат-бота изображен на рисунке 16.

```
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    if not db.user_exists(message.from_user.id):
        db.add_user(message.from_user.id)
        await bot.send_message(message.from_user.id, "Введите ваше ФИО:")
        db.set_signup_status(message.from_user.id, "setfio")
    else:
        await bot.send_message(message.from_user.id, "Вы уже зарегистрированы!", reply_markup=nav.mainMenu)
```

Рисунок 16 – Реализованная клавиатура основного меню

Таким образом, была реализована клиентская часть чат-бота для информационной поддержки студентов.

5.2.3 Проектирование и реализация базы данных чат-бота

Для реализации поставленных задач появилась необходимость в базе данных (БД). В проекте используется Sqlite. База данных чат-бота представляет собой таблицу. Диаграмма структуры БД представлена на рисунке 17.

Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Сгенерированное
1 user_id	INTEGER	🔑						NULL
2 setfio	TEXT							NULL
3 setbirthdate	TEXT							NULL
4 setcontacts	TEXT							NULL
5 setfaculty	TEXT							NULL
6 setnickname	TEXT							NULL
7 signup_status	INTEGER							0

Рисунок 17 – Структура базы данных

Приведенная таблица выполняет свою функцию:

`user_id` – это уникальный `id`, который присваивается каждому пользователю, который запустил чат-бота,

`setfio` – это фамилия, имя и отчество уникального пользователя, который прошел регистрацию в чат-боте,

`setbirthdate` – это дата рождения уникального пользователя, который записывается в базу данных при регистрации в чат-боте,

`setcontacts` – это контакты уникального пользователя, которые записываются в базу данных при регистрации в чат-боте,

`setfaculty` – это факультет уникального пользователя, которые записываются в базу данных при регистрации в чат-боте,

`setnickname` – это никнейм уникального пользователя, которые записываются в базу данных при регистрации в чат-боте,

`signup-status` – это статус, который присваивается каждому пользователю, если пользователь зарегистрирован в системе, ему присваивается значение «done».

Программа, представляет собой класс на языке Python с названием Database, который предоставляет методы для взаимодействия с базой данных SQLite. Он позволяет создавать таблицу с названием "users" и определенными столбцами, добавлять пользователей, проверять существование пользователя и выполнять различные операции для получения и установки информации о пользователе.

Обзор методов, предоставляемых классом Database:

- `__init__(self, db_file)`. Это конструктор класса, который инициализирует подключение к базе данных SQLite. Он принимает аргумент `db_file`, который представляет собой путь к файлу базы данных,

- `create_table(self)`. Метод для создания таблицы "users" в базе данных, если она не существует,

`add_user(self, user_id)`: Метод для добавления пользователя в таблицу "users". Принимает аргумент `user_id`, представляющий идентификатор пользователя,

`user_exists(self, user_id)`: Метод для проверки существования пользователя в таблице "users". Принимает аргумент `user_id` и возвращает True, если пользователь существует, иначе False,

`set_signup_status(self, user_id, status)`: Метод для установки статуса регистрации пользователя. Принимает аргументы `user_id` и `status`, где `user_id` - идентификатор пользователя, а `status` - значение статуса регистрации,

`get_signup_status(self, user_id)`: Метод для получения статуса регистрации пользователя. Принимает аргумент `user_id` и возвращает значение статуса регистрации пользователя,

`set_fio(self, user_id, fio)`: Метод для установки ФИО пользователя. Принимает аргументы `user_id` и `fio`, где `user_id` - идентификатор пользователя, а `fio` - ФИО пользователя,

`get_fio(self, user_id)`: Метод для получения ФИО пользователя. Принимает аргумент `user_id` и возвращает ФИО пользователя,

`set_birthdate(self, user_id, birthdate)`: Метод для установки даты рождения пользователя. Принимает аргументы `user_id` и `birthdate`, где `user_id` - идентификатор пользователя, а `birthdate` - дата рождения пользователя,

`get_birthdate(self, user_id)`: Метод для получения даты рождения пользователя. Принимает аргумент `user_id` и возвращает дату рождения пользователя,

`set_contacts(self, user_id, contacts)`: Метод для установки контактной информации пользователя. Принимает аргументы `user_id` и `contacts`, где

`user_id` - идентификатор пользователя, а `contacts` - контактная информация пользователя,

`get_contacts(self, user_id)`: Метод для получения контактной информации пользователя. Принимает аргумент `user_id` и возвращает контактную информацию пользователя,

`set_faculty(self, user_id, faculty)`: Метод для установки факультета пользователя. Принимает аргументы `user_id` и `faculty`, где `user_id` - идентификатор пользователя, а `faculty` - факультет пользователя,

`get_faculty(self, user_id)`: Метод для получения факультета пользователя. Принимает аргумент `user_id` и возвращает факультет пользователя,

`set_nickname(self, user_id, nickname)`: Метод для установки никнейма пользователя. Принимает аргументы `user_id` и `nickname`, где `user_id` - идентификатор пользователя, а `nickname` - никнейм пользователя,

`get_nickname(self, user_id)`: Метод для получения никнейма пользователя. Принимает аргумент `user_id` и возвращает никнейм пользователя.

Эти методы позволяют взаимодействовать с базой данных SQLite, добавлять пользователей, получать и устанавливать информацию о пользователях.

5.2.4 Реализация серверной части чат-бота

Для эффективной работы серверной части чат-бота на платформе Telegram, необходимо решить следующие задачи:

- Получение и распознавание сообщения: Серверная часть должна принимать сообщения, отправленные пользователями, и распознавать их содержимое. Это включает анализ текста, изображений, аудио или других медиафайлов, которые могут быть отправлены пользователем,

- Сравнение информации: Полученная информация должна быть сопоставлена с предопределенными шаблонами или правилами для определения нужного ответа. Например, если пользователь задает вопрос о

погоде, сервер должен распознать этот запрос и определить, что нужно отправить информацию о погоде,

– Извлечение информации из базы данных: Серверная часть должна иметь доступ к базе данных, где хранится необходимая информация. С помощью выполнения запросов с полученными параметрами, сервер может извлечь нужные данные для формирования ответа. Например, если пользователь запрашивает информацию о расписании, сервер может выполнить запрос в базе данных, чтобы получить соответствующую информацию,

– Составление и отправка ответа: на основе полученной информации и данных из базы данных, серверная часть должна составить ответ в виде текста или другого формата (например, изображения или видео) и отправить его обратно пользователю. Ответ должен быть сформирован таким образом, чтобы соответствовать запросу пользователя и предоставить ему необходимую информацию. В данной работе используется библиотека `aiogram` для создания клавиатурных кнопок и меню. Код определяет несколько кнопок и создает две клавиатуры: `mainMenu` и `facultyMenu`.

Для того чтобы чат-бот имел возможность распознавать сообщение пользователя каждая кнопка `inline-клавиатуры` имеет так называемую обратную информацию (англ. `callback data`). После нажатия кнопки пользователем серверная часть сравнивает полученный ответ со значением или массивом значений (Рис 18).

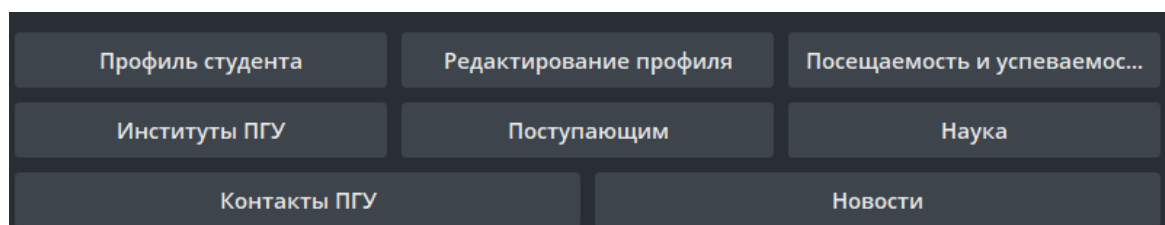


Рисунок 18 – Окно авторизации административной панели

Таким образом, были реализованы следующие функции чат-бота на платформе Telegram:

- 1 Профиль студента ПГУ,

- 2 Редактирование профиля студента ПГУ,
- 3 Посещаемость и успеваемость студента ПГУ,
- 4 Институты ПГУ,
- 5 Поступающим (абитуриентам),
- 6 Наука ПГУ,
- 7 Контакты ПГУ,
- 8 Новости ПГУ.

5.3 Административная панель чат бота

Данный модуль проектируется и реализуется по шаблону MVC. Концепция MVC широко используется в веб-программировании в последние годы. MVC представляет собой конструктивный шаблон, который определяет структуру приложения, области ответственности и взаимодействие между его компонентами.

В рамках шаблона MVC, модель (Model) представляет собой слой данных и бизнес-логики приложения. Она отвечает за обработку данных, взаимодействие с базой данных и выполнение бизнес-логики.[28]

Представление (View) отвечает за отображение данных пользователю. Оно представляет собой пользовательский интерфейс, через который пользователи взаимодействуют с приложением. Представление получает данные от модели и отображает их пользователю в удобном для восприятия виде.

Контроллер (Controller) является посредником между моделью и представлением. Он отвечает за обработку пользовательских действий и управление потоком данных между моделью и представлением. Контроллер получает запросы от пользователей, обрабатывает их, взаимодействует с моделью для получения необходимых данных и передает их в представление для отображения.

Использование шаблона MVC позволяет разделить логику приложения на логические компоненты, что упрощает разработку, поддержку и масштабирование приложения. Модель, представление и контроллер имеют

четко определенные роли и ответственности, что способствует лучшей организации кода и повышает его читаемость и переиспользуемость.

Первое что увидит при входе в административную панель администратор приложения будут данные пользователя и его уникальный id. Страница авторизации графически показана на рисунке 19.

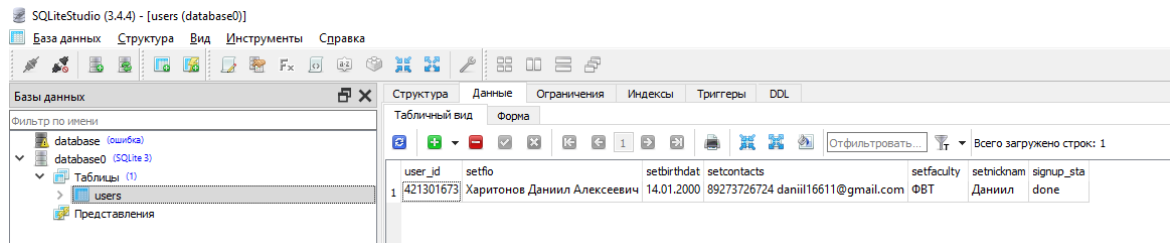


Рисунок 19 – Окно авторизации административной панели

У администратора приложения будут неограниченные права на создание, чтение, редактирование и удаление данных.

5.3.1 Разработка и реализация компонентной структуры чат-бота

- Реализация компонентной структуры на основе алгоритма новостей:

```
def get_latest_news():
```

```
    url = "https://example.com/news" #
```

```
    response = requests.get(url)
```

```
    soup = BeautifulSoup(response.text, "html.parser")
```

```
    news_items = soup.find_all("div", class_="news-item")
```

```
    latest_news = []
```

```
    for news in news_items[:3]:
```

```
        title = news.find("h2").text.strip()
```

```
        content = news.find("p").text.strip()
```

```
        latest_news.append({"title": title, "content": content})
```

```
    return latest_news
```

```
btnNews = KeyboardButton("Новости")
```

```
mainMenu = ReplyKeyboardMarkup(resize_keyboard=True)
```

```
mainMenu.add(btnProfile, btnEditProfile, btnAttendance, btnInstitutes,
```

```
btnApplicants, btnScience, btnContacts, btnNews)
```

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

```

async def handle_news(message: Message):
    news = get_latest_news()
    response = "Последние новости:\n\n"
    for item in news:
        response += f"{item['title']}\n{item['content']}\n\n"
    await message.answer(response)

```

Алгоритм реализации запроса «Новости» реализован на рисунке 20.



Рисунок 20 – Выполнение запроса на запись новости

6 Результаты разработки чат-бота

6.1 Тестирование системы чат-бота

Ниже приведены результаты разработки чат-бота для мессенджера Telegram и его функции.

В самом начале работы бот предлагает воспользоваться командой /start – стартовая команда, с которой начинается работа с ботом, при вызове присылается приветственное сообщение и начинается этап регистрации. Далее этап регистрации представлен на рисунках 21, 22, 23, 24 и 25.

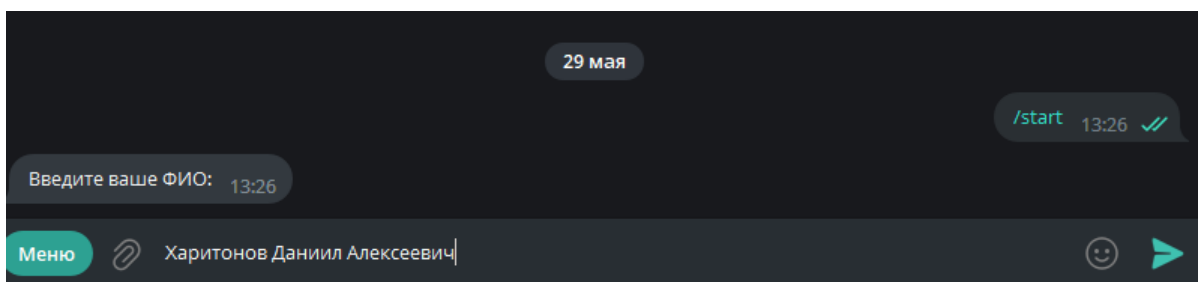


Рисунок 21 – Ответ на команду /start

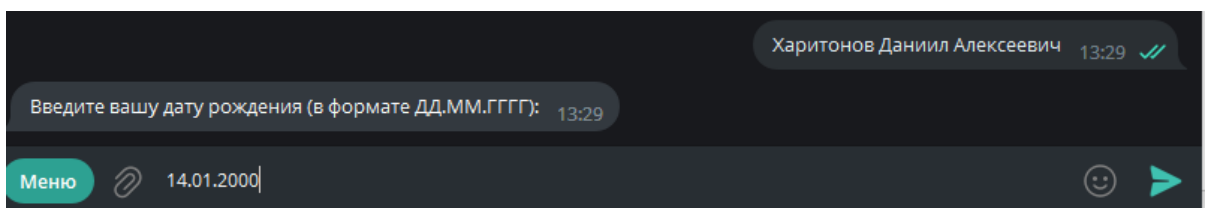


Рисунок 22 – Ответ на команду /start

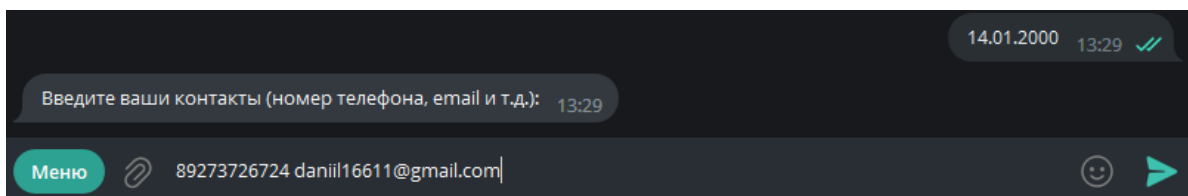


Рисунок 23 – Ответ на команду /start



Рисунок 24 – Ответ на команду /start

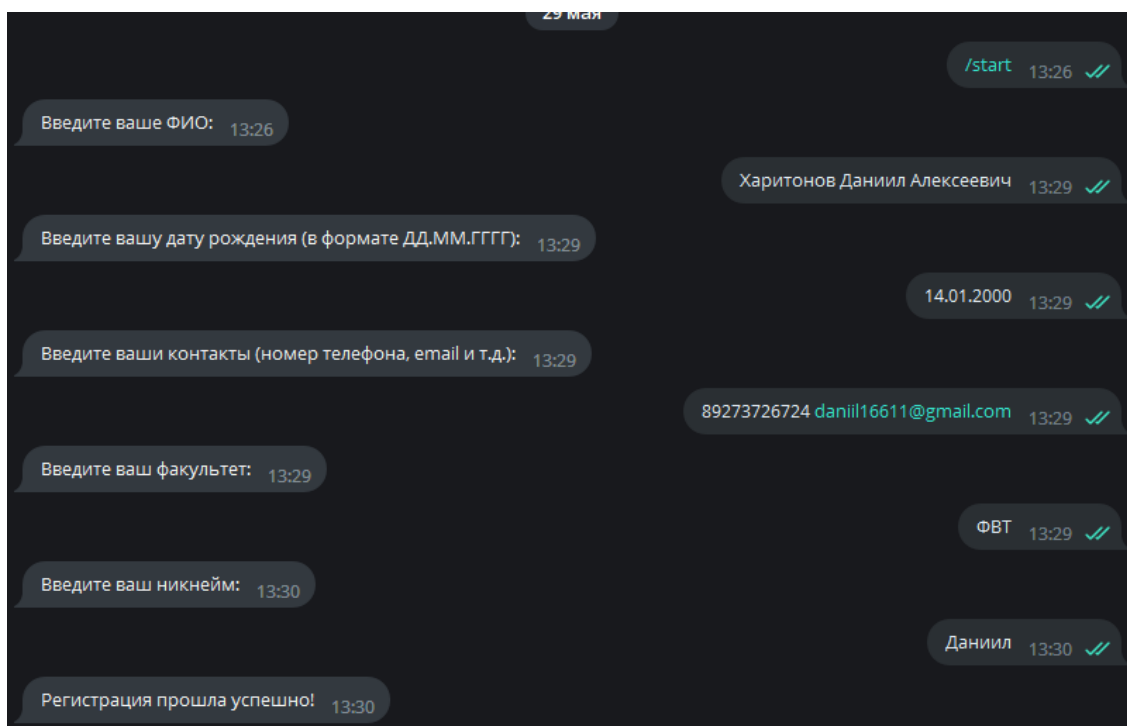


Рисунок 25 – Ответ на команду /start

После того, как все этапы регистрации будут успешно пройдены, чат-бот оповестит вас об этом сообщением «Регистрация прошла успешно!»

Далее отображается главное меню, в котором на выбор есть 8 кнопок «Профиль студента», «Редактирование профиля», «Посещаемость и успеваемость студента», «Институты ПГУ», «Поступающим», «Наука», «Контакты ПГУ» и «Новости». Есть возможность воспользоваться справкой, вызвав команду /help – команда, выводящая справку по боту и список команд.

Согласно информации о боте, в нем содержится основная информация для студентов и абитуриентов ВУЗа. В первую очередь было реализовано главное меню (Рис.26).

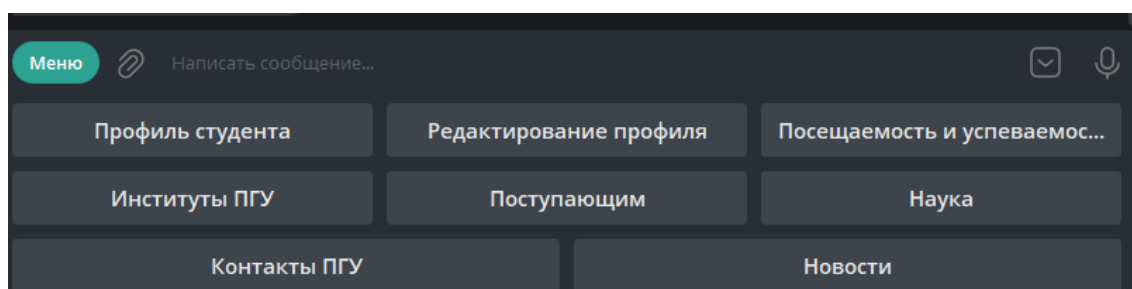


Рисунок 26– Главное меню чат-бота Telegram

Кнопка «Профиль студента» – в ответ на команду присылается сообщение с информацией о профиле, которую указывал пользователь на этапе регистрации. Далее на рисунке 27 представлен ответ на команду «Профиль студента»

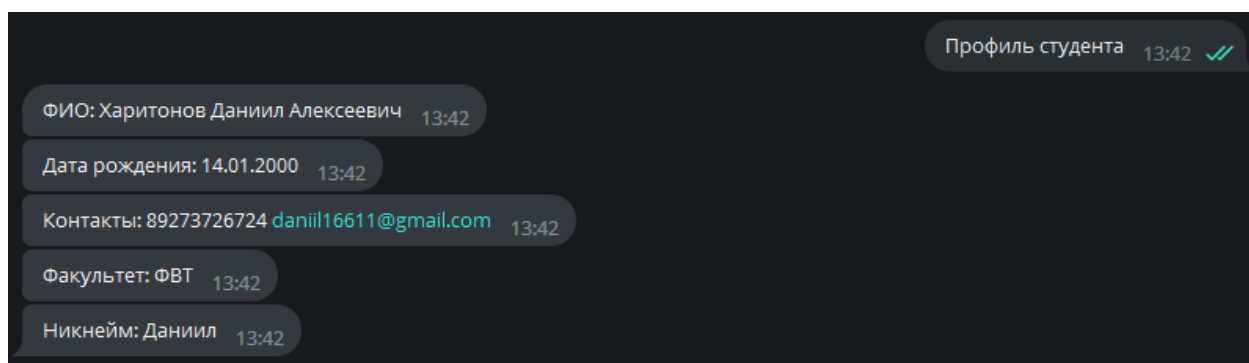


Рисунок 27– Ответ на команду «Профиль студента»

Следующая кнопка «Редактировать профиль», в ней можно поменять любую информацию пользователя, который указывал на этапе Регистрации (ФИО, дата рождения, контакты, факультет и никнейм).

Кнопка «Редактировать профиль» – в ответ на запрос выводится сообщение с тем какую информацию нужно изменить, например Контакты. Далее на рисунке 28 представлен ответ на команду «Редактировать профиль»

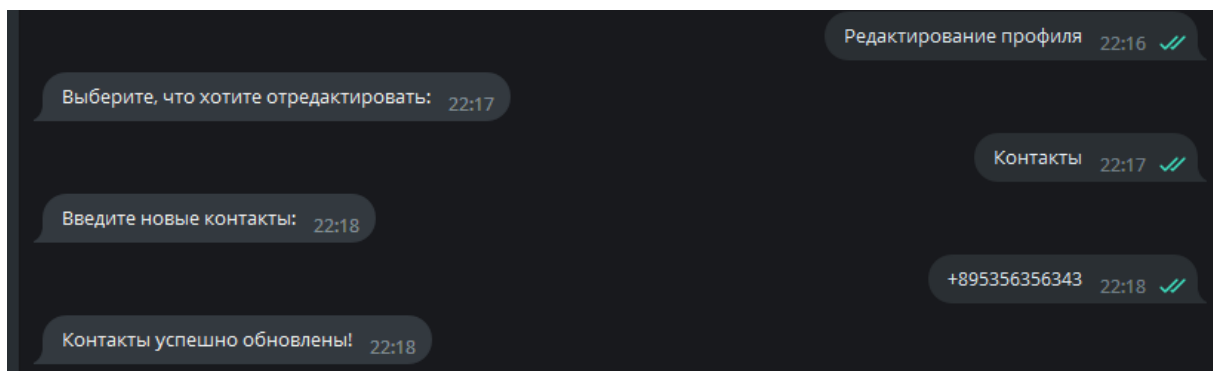


Рисунок 28 – Ответ на команду редактировать профиль

Следующая функция «Посещаемость и успеваемость студента», потенциально, может представлять наибольший интерес для студентов. Чат-бот позволяет узнать о задолженностях и успеваемости студента ПГУ, узнать расписание своего семестра, а также узнать о предстоящих занятиях и курсах. Далее на рисунке 29 представлен ответ на команду «Посещаемость и успеваемость»

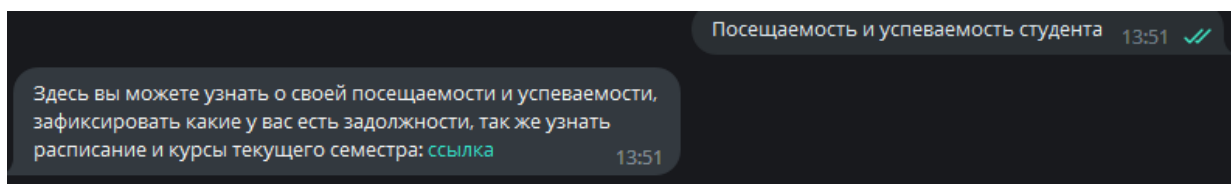


Рисунок 29 – Ответ на команду Посещаемость и успеваемость

Следующая функция, потенциально, может представлять наибольший интерес для студентов и абитуриентов. Кнопка «Институты ПГУ» - чат-бот позволяет узнать о всех институтах ПГУ, не переходя на официальный сайт ПГУ и не путаться в навигации, так же он показывает текущую неделю. В ответ на запрос пользователю необходимо выбрать интересующее направление, после выбора будет выведена специальность и клавиатура для переключения между ними. Далее на рисунке 30 представлен ответ на команду «Институты ПГУ»

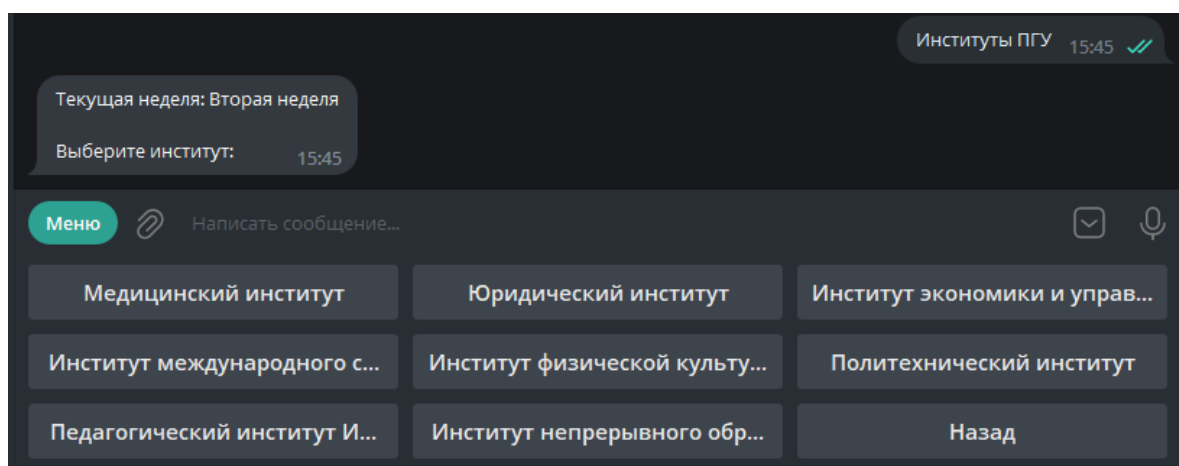


Рисунок 30– Ответ на команду Институты ПГУ

Кнопка «Поступающим» в ответ на запрос пользователю будет выдана ссылка на актуальную страницу для поступающим, а также список документов, включающий в себя «Правила приема на обучение по образовательным программам высшего образования 2023/2024 учебный год». Далее на рисунке 31 представлен ответ на команду «Поступающим».

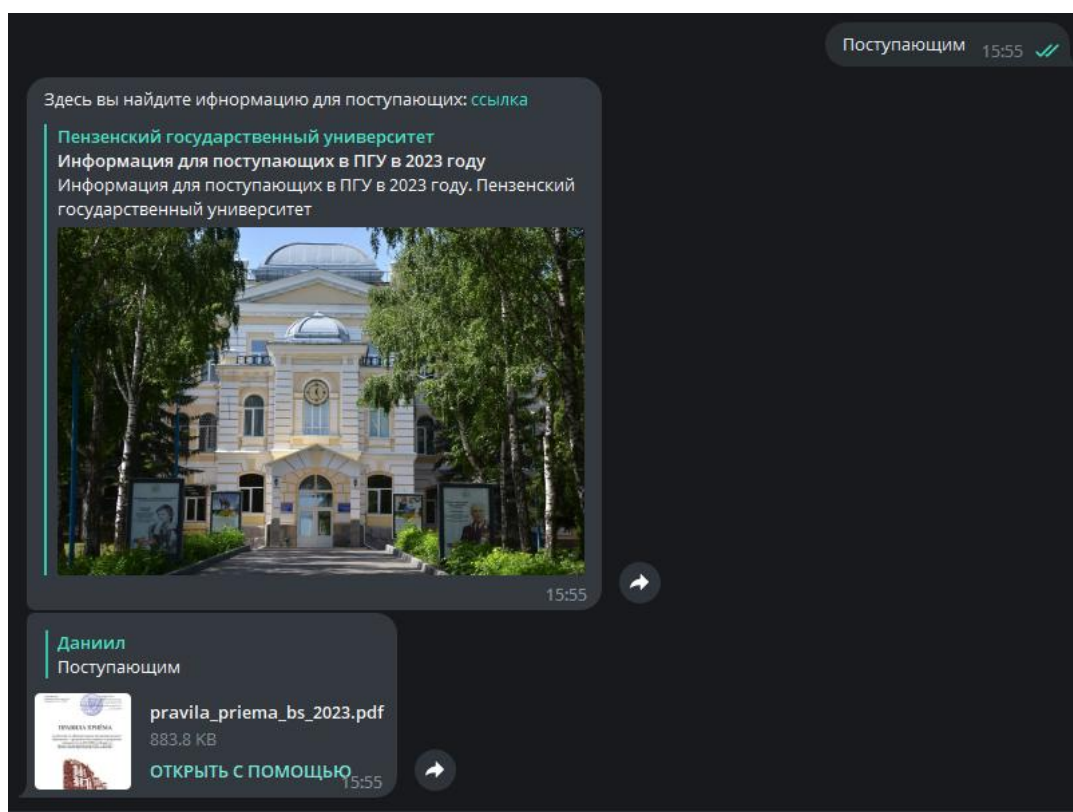


Рисунок 31 – Ответ на команду поступающим

Получить подробную информацию, полезные ссылки, статьи, документацию, а также помощь по многим предметам Вузовской программы можно по кнопке «Наука» она включает в себя огромное множество ресурсов,

которые пригодятся в получении знаний при учебе и поступлении. В ответ на запрос присылается сообщение с краткой инструкцией по науке и ссылкой. Далее на рисунке 30 представлен ответ на команду «Наука»

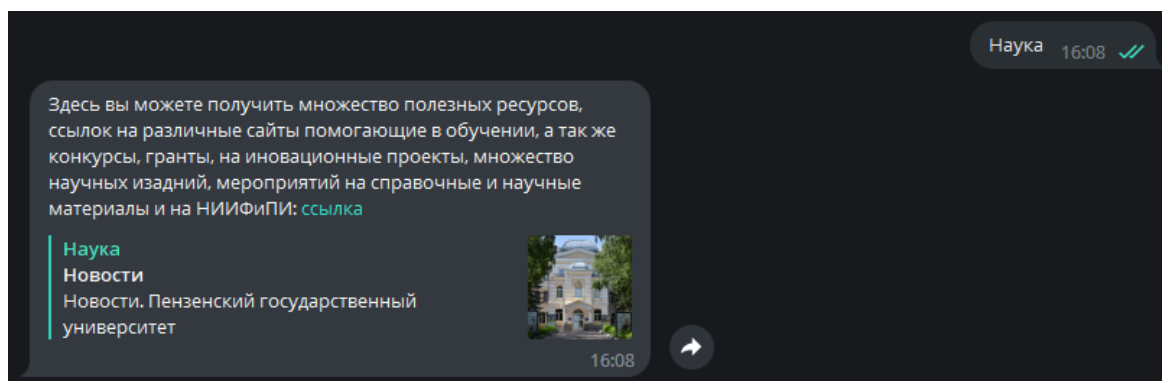


Рисунок 32 – Ответ на команду Наука

Кнопка «Контакты» – команда выводит контактную информацию о приемной комиссии, ссылку на социальную сеть «Вконтакте», карту, при использовании мобильного приложения можно проложить маршрут до университета. Далее на рисунке 30 представлен ответ на команду «Контакты ПГУ»

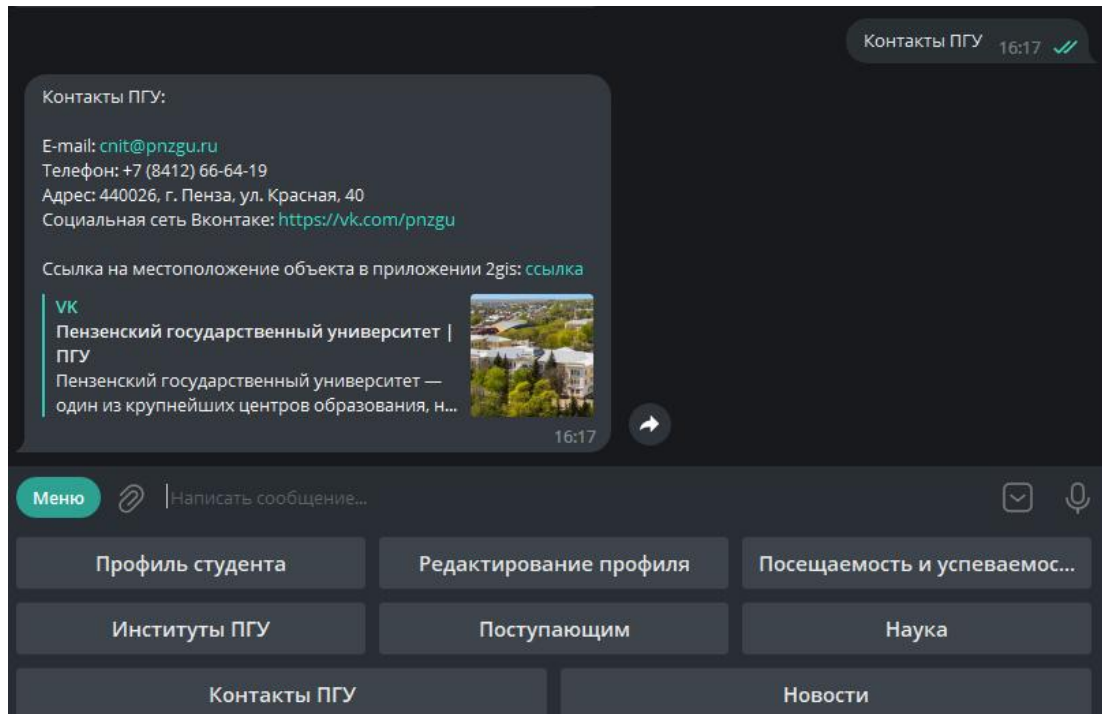


Рисунок 33 – Ответ на команду Контакты ПГУ

Наиболее интересный функционал предоставляет кнопка «Новости». С помощью нее можно получить последние новости три новости, которые

отображаются на сайте <https://pnzgu.ru/news>, так как на момент написания магистерской диссертации свежих новостей не было, он выдает такой же ответ. Далее на рисунке 34 представлен ответ на команду «Новости».

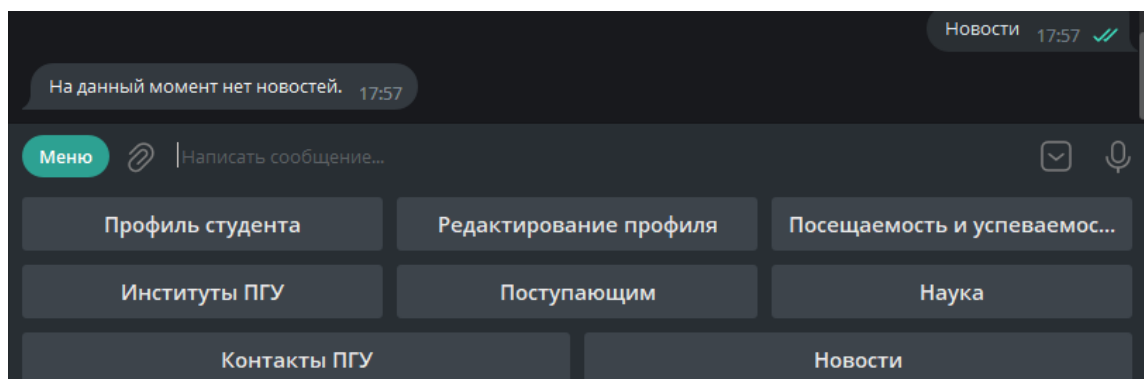


Рисунок 34 – Ответ на команду новости

7 Удаленный сервер Heroku

7.1 Анализ и размещение облачного сервера Heroku

Heroku — популярная облачная платформа, которая предоставляет широкий спектр услуг и простоту в использовании для разработки проектов. Она является платформой как услуга (PaaS) и поддерживает несколько языков программирования.

С самого начала своего развития с июня 2007 года, Heroku была ориентирована на язык Ruby, и она остается одной из ведущих платформ для Ruby-разработки. Однако, со временем Heroku расширила свои возможности и в настоящее время поддерживает также Java, Node.js, Scala, Clojure, Python, PHP и Go. Это делает Heroku привлекательным выбором для разработчиков, работающих на разных языках программирования.

Heroku обеспечивает простоту развертывания и масштабирования приложений. Разработчики могут легко разместить свои приложения на платформе Heroku, используя интуитивный интерфейс и командную строку. Heroku также предоставляет гибкие возможности масштабирования, позволяя приложениям автоматически расти по мере увеличения нагрузки.

Одним из основных преимуществ Heroku является интеграция с системами контроля версий, такими как Git. Разработчики могут легко

развертывать свои приложения, используя средства автоматического развертывания, которые связываются с их репозиторием Git.

Облачные сервисы можно быстро предоставлять с минимальными усилиями по управлению со стороны бизнеса. Хотя облако является привлекательным вариантом для компаний с различными потребностями, особенности бизнес-модели и потребности проекта будут определять, к какому поставщику облачных услуг они в конечном итоге прибегнут.

Heroku и пример решения «Платформа как услуга» обычно считается простым в использовании. Но это наиболее выгодно для бизнеса в конкретных ситуациях. У Heroku есть бесплатная модель обслуживания для небольших проектов, но существуют многоуровневые пакеты услуг для случаев, когда необходимо решать более сложные бизнес-задачи.

Платформа облачных сервисов Heroku основана на системе управляемых контейнеров (называемых dynos в рамках парадигмы Heroku) с интегрированными службами данных и мощной экосистемой для развертывания и запуска современных приложений.

Приложения, работающие на Heroku, обычно имеют уникальные доменные имена, которые используются для маршрутизации HTTP-запросов в нужный контейнер. Приложения в качестве служб используют контейнеры приложений. Контейнеры предназначены для упаковки и запуска сервисов. Каждый из контейнеров приложений представляет собой интеллектуальный контейнер в надежной, полностью управляемой среде выполнения. Контейнеры приложений, называемые «dynos» в контексте платформы Heroku, распределены по «динамической сетке». Он состоит из нескольких серверов. Менеджер Dyno поддерживает и управляет созданными Dyno.

Все это означает, что, поскольку Heroku управляет приложениями и запускает их, нет необходимости управлять операционными системами или другими внутренними системными конфигурациями.

Ваши приложения запускаются внутри смарт-контейнеров в полностью управляемой среде выполнения, мы берем на себя все важные для

производства задачи: настройку, оркестровку, балансировку нагрузки, аварийное переключение, ведение журналов, безопасность и многое другое.

Heroku предлагает широкий спектр возможностей и услуг, которые значительно упрощают разработку и развертывание приложений. В управляемой среде выполнения, основанной на смарт-контейнерах, называемых dynos, Heroku предоставляет полный спектр решений для задач настройки, оркестровки, балансировки нагрузки, аварийного переключения, ведения журналов и обеспечения безопасности.

Heroku также предоставляет гибкую возможность масштабирования приложений, позволяя вертикальное и горизонтальное масштабирование. Это означает, что вы можете легко масштабировать приложения от небольших хобби-проектов до масштабных корпоративных решений, способных справиться с любыми всплесками активности, включая пиковую нагрузку во время Черной пятницы.

Кроме основных возможностей, Heroku предлагает множество дополнительных интегрированных сервисов, таких как New Relic, MongoDB, SendGrid, Searchify, Fastly, Papertrail, ClearDB MySQL, Treasure Data и другие, которые позволяют расширить и усовершенствовать функциональность ваших приложений.

Одним из преимуществ Heroku является возможность мгновенного отката кода и базы данных до предыдущего состояния, обеспечивая безопасность и уверенность в работе с приложениями.

Встроенное мониторинг приложений на Heroku позволяет отслеживать пропускную способность, время отклика, использование памяти, загрузку ЦП и ошибки, чтобы всегда быть в курсе состояния ваших приложений.

Благодаря бесшовной интеграции с GitHub, Heroku позволяет автоматически развертывать ваше приложение при каждом вытягивании (pull request) или отправке в выбранную ветку, обеспечивая быструю и простую интеграцию с вашими существующими инструментами разработки.

Heroku также обладает высокой степенью расширяемости, благодаря инновационной функции Buildpacks. Она позволяет настраивать стек технологий и использовать сотни готовых Buildpacks, созданных сообществом, для запуска приложений на различных языках программирования и технологиях.

С помощью умных контейнеров, известных как dynos, Heroku обеспечивает оптимальную работу ваших приложений, постоянно обновляя системные и языковые стеки внутри контейнеров.

Docker-поддержка на Heroku позволяет упаковывать ваше приложение и его зависимости в контейнер, обеспечивая гибкое и переносимое развертывание.

С возможностью оптимизировать использование ресурсов и объединять несколько приложений в одном контейнере, Heroku позволяет достичь эффективного использования ресурсов и снизить затраты благодаря контейнерным плотностям.

Эти уникальные особенности Heroku делают его мощным и гибким инструментом для разработки, развертывания и масштабирования приложений, позволяя вам фокусироваться на создании высококачественного кода и обеспечении бесперебойной работы ваших приложений. Контейнерная модель удаленного сервера Heroku изображена на рисунке 35.

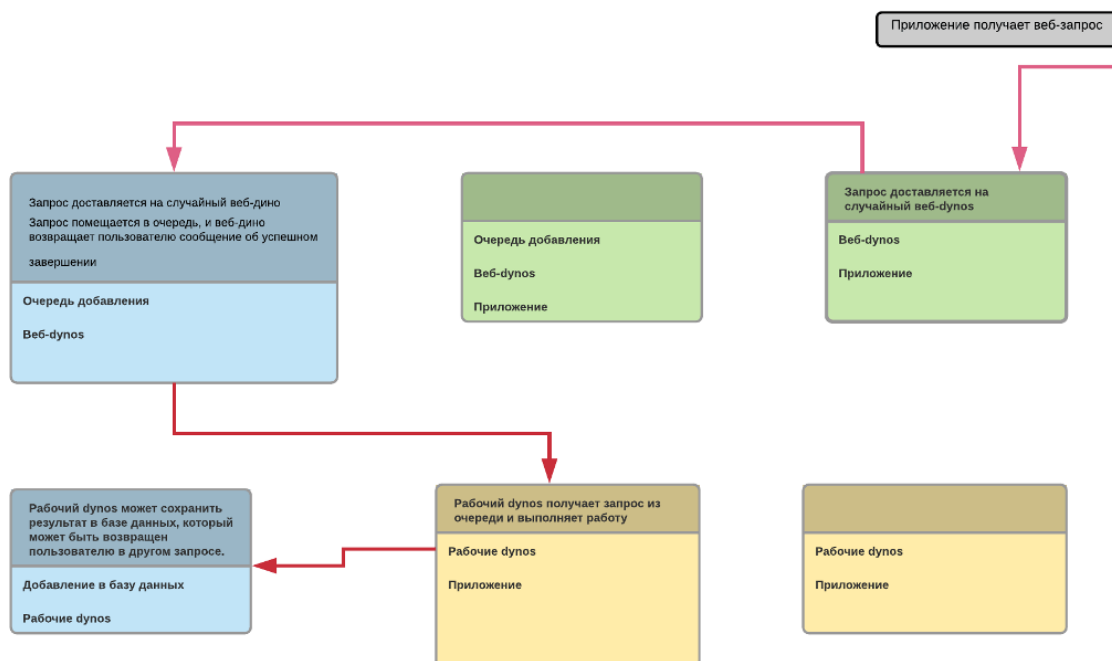


Рисунок 35 Контейнерная модель удаленного сервера Heroku

Чтобы обеспечить постоянный доступ к нашему чат-боту, мы решили разместить его на удаленном сервере с использованием платформы Heroku. Heroku предоставляет облачные услуги и является отличным выбором для развертывания наших приложений.

Процесс подключения к Heroku проходит следующим образом. Сначала мы создали аккаунт на платформе Heroku, чтобы получить доступ к их сервисам. Затем мы установили Heroku CLI (Command Line Interface) на нашу локальную машину. Heroku CLI предоставляет набор инструментов, которые позволяют нам взаимодействовать с Heroku через командную строку.

После установки Heroku CLI мы выполнили вход в наш аккаунт Heroku, используя команду "heroku login". Это позволило нам аутентифицироваться и установить соединение с нашим аккаунтом Heroku.

Затем мы клонировали репозиторий с нашим чат-ботом с удаленного Git-сервера Heroku на нашу локальную машину, используя команду "heroku git:clone -a ". Теперь у нас была локальная копия кода нашего чат-бота.

Мы внесли необходимые изменения в коде чат-бота, чтобы учесть наши требования и функциональность. Затем мы зафиксировали изменения с помощью команды "git add ." для добавления всех измененных файлов в

индекс Git, а затем команды "git commit -am " для создания коммита с описанием изменений.

После этого мы отправили все зафиксированные изменения на удаленный сервер Heroku, используя команду "git push heroku master". Это начало процесс передачи данных на сервера Heroku и развертывания нашего чат-бота.

Если все прошло успешно, мы увидим информацию о состоянии приложения и его доступности в терминале. Это будет означать, что наш чат-бот успешно развернут на серверах Heroku и готов к использованию.

Теперь наш чат-бот доступен на удаленном сервере Heroku, что позволяет пользователям взаимодействовать с ним в любое время. Был успешно пройден процесс подключения к Heroku, теперь чат-бот будет работать стабильно и надежно, обеспечивая непрерывный доступ и качественное обслуживание наших пользователей. Далее на рисунке 36 изображено подключение бота к серверу Heroku.

```
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote:      Procfile declares types      -> (none)
remote:      Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:      Done: 19.4M
remote: -----> Launching...
remote:      Released v3
remote:      https://sfutelegrambotdip.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/sfutelegrambotdip.git
 * [new branch]      master -> master
```

Рисунок 36 – Результат развертывания чат-бота

ЗАКЛЮЧЕНИЕ

С каждым днем, вместе с растущей популярностью мессенджеров в качестве главного средства общения, находят все большее распространение чат-боты, функционирующие на этих платформах.

Чат-боты — это удивительные сущности, которые предлагают пользователю бесконечное разнообразие возможностей, собранных в одном мессенджере. Они являются настоящими "мульти инструментами", обеспечивающими доступ к круглосуточной поддержке, конвертации документов и медиафайлов, заказу такси, поиску информации и многому другому. Больше не нужно тратить время на загрузку и установку множества отдельных приложений - все, что нужно, это просто мессенджер и соответствующий чат-бот. Ведь, по сути, чат-боты — это наши надежные помощники, занимающие минимальное количество памяти на смартфоне, но способные выполнять множество задач, оживляя нашу коммуникацию и делая нашу жизнь проще и удобнее. Данная выпускная квалификационная работа была посвящена разработке и реализации чат-бота на платформе Telegram с целью предоставления информационной поддержки студентам и абитуриентам. В ходе исследования и разработки успешно достигнута поставленная цель и получен ценный опыт в области разработки чат-ботов и применения технологий мессенджеров.

В ходе работы мы провели анализ различных мессенджеров и выбрали Telegram в качестве основной платформы для реализации нашего чат-бота. Telegram предоставляет широкие возможности для разработки и интеграции чат-ботов, а также обладает удобным и документированным Telegram-Bot-API.

В данной ВКР изучили существующие аналоги чат-ботов на платформе Telegram и выявили их достоинства, недостатки и интересные решения. Это позволило нам определить требования и функциональность нашего чат-бота, сосредоточившись на предоставлении информационной поддержки студентам во время учебы или процесса поступления в университет.

Для разработки чат-бота в данной ВКР использовали современные технологии и инструменты. В текущей работе выбрали PyCharm в качестве интегрированной среды разработки (IDE) и язык программирования Python, который обеспечивает гибкость и мощные возможности для разработки чат-ботов. Также мы использовали библиотеку Aiogram для взаимодействия с Telegram Bot API, что позволило нам создать эффективное и отзывчивое взаимодействие с пользователями.

Одной из важных составляющих нашего проекта было развертывание чат-бота на облачной платформе Heroku. Был успешно освоен процесс размещения на Heroku и использовали командную строку Heroku CLI для развертывания и управления нашим приложением. Это обеспечило надежную и стабильную работу нашего чат-бота, обеспечивая бесперебойный доступ для пользователей.

В результате нашей работы полностью реализован чат-бот, который успешно предоставляет информационную поддержку студентам и абитуриентам. Данный чат-бот помогает пользователям получать необходимую информацию, отвечает на вопросы, предоставляет рекомендации и облегчает процесс общения и взаимодействия с университетом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бот (программа) [Электронный ресурс]. — Режим доступа: [https://ru.wikipedia.org/wiki/Бот_\(программа\)#Чат-боты](https://ru.wikipedia.org/wiki/Бот_(программа)#Чат-боты) (дата обращения: 27.11.2021).
2. Документация Telegram: Примеры ботов [Электронный ресурс]. — Режим доступа: <https://tlgrm.ru/docs/bots/samples#c-sharp> (дата обращения: 11.03.2018).
3. Банокин П.И. Методы и средства проектирования информационных систем и технологий: учебное пособие / П.И. Банокин; Томский политехнический университет. — Томск: Изд-во Томского политехнического университета, 2015. — 92 с.
4. Регламент обработки обращений клиентов в чате [Электронный ресурс]. — URL: <https://wiki.skbbkontur.ru/pages/viewpage.action?pageId=313263686> (дата обращения: 11.05.2020).
5. Лапшина С. Н. Архитектура предприятия [Текст]: Учебное электронное текстовое издание / С. Н. Лапшина – ФГАОУ ВО «УрФУ», 2012.
6. Использование компонентной архитектуры в веб приложениях // [Электронный ресурс]. URL: <https://fwdays.com/en/event/js-frameworks-day2015/review/komponentnaia-arkhitektura-v-web-prilozheniiakn> (дата обращения: 29.03.23).
7. Мартынов А.В. Python и SQLite. Взаимодействие с базами данных: учебное пособие / А.В. Мартынов; РГУПС. — Москва: Изд-во РГУПС, 2021. — 120 с.
8. Gensim: Topic Modelling for Humans [Электронный ресурс]. — Режим доступа: <https://radimrehurek.com/gensim/> (дата обращения: 03.06.2023).
9. SQLite: Home Page [Электронный ресурс]. — Режим доступа: <https://www.sqlite.org/index.html> (дата обращения: 03.06.2023).
10. Heroku: Cloud Application Platform [Электронный ресурс]. — Режим доступа: <https://www.heroku.com/> (дата обращения: 03.06.2023).

11. PyCharm: Python IDE for Professional Developers [Электронный ресурс]. — Режим доступа: <https://www.jetbrains.com/pycharm/> (дата обращения: 03.06.2023).

12. NumPy: The fundamental package for scientific computing with Python [Электронный ресурс]. — Режим доступа: <https://numpy.org/> (дата обращения: 03.06.2023).

13. Kovalenko A. Telegram Bot API: от создания до монетизации. — Киев: self-published, 2020. — 160 с.

14. Орлов В.П., Карен П.М., Артамонов А.М. Анализ и проектирование информационных систем. — Москва: Издательство МГТУ им. Н.Э. Баумана, 2015. — 432 с.

15. Рассел С., Норвиг П. Искусственный интеллект: современный подход. — Москва: ДМК Пресс, 2019. — 1168 с.

16. Гринфилд Р., Майфилд С. Python и анализ данных. — Москва: ДМК Пресс, 2020. — 400 с.

17. Pandas: Powerful data structures for data analysis [Электронный ресурс]. — Режим доступа: <https://pandas.pydata.org/> (дата обращения: 03.06.2023).

18. Аванесян Н. Л., Telegram, как пример мессенджера: возможности и перспективы развития. [Электронный ресурс] / Н. Л. Аванесян // Научный потенциал XXI века. – 2018. – Режим доступа: https://elibrary.ru/download/elibrary_29653726_34734017.pdf

19. Агальцов, В. Базы данных: В 2-х книгах. Книга 2: Распределенные и удаленные базы данных : Учебник /В. Агальцов – Москва : Форум, 2022. – 271 с.

20. Иванов А. Д., Чат-бот в Telegram и ВКонтакте, как новый канал распространения новостей. [Электронный ресурс] / А. Д. Иванов // Волжский университет имени В.Н. Татищева. – 2022. – №3 – с. 126-132. – Режим доступа: https://elibrary.ru/download/elibrary_26673675_34058358.pdf

21. Рейтинг мессенджеров 2022 [Электронный ресурс] : Сервис сравнения и выбора приложений и сервисов в сфере ИТ. – 2022. – Режим доступа: <https://coba.tools/compilation/reiting-messendzherov-2018>

22. Сайт Github [Электронный ресурс], URL: <https://groosha.gitbooks.io/telegram-bot-lessons/content/> (Дата обращения: 29.01.2019).

23. Харрисон М., Как устроен Python. Гид для разработчиков, программистов и интересующихся, - 2019 –с. 51.

24. Сайт Telegram [Электронный ресурс], URL: <https://telegram.org/> (Дата обращения: 15.01.2019).

25. Как создать своего бота в BotFather? [Электронный ресурс]. – Режим доступа: <https://botcreators.ru/blog/kak-sozdat-svoego-bota-v-botfather/> –Загл. с экрана.

26. Какая библиотека для создания telegram бота на Python лучше и удобнее? [Электронный ресурс]. Режим доступа: <https://qna.habr.com/q/815357> –Загл. с экрана

27. Мы сделали из чатботов помощников, теперь пора сделать из них собеседников [Электронный ресурс]. Режим доступа: <https://tjournal.ru/tech/58633-my-sdelali-iz-chatbotov-pomoshchnikov-teper-pora-sdelat-iz-nih-sobesednikov> –Загл. с экрана.

28. Основы программирования на примере языка Python [Электронный ресурс]. – Режим доступа: <https://dfedorov.spb.ru/python3/book.pdf> – Загл. с экрана.

ПРИЛОЖЕНИЕ А

ЛИСТИНГ ПРОГРАММЫ

```
from aiogram import Bot, Dispatcher, executor, types
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton,
InlineKeyboardMarkup, InlineKeyboardButton
from aiogram.dispatcher.filters import Command, Text
from aiogram.types import InlineQuery, InputTextMessageContent,
InlineQueryResultArticle
import logging
import markups as nav
from db import Database
from datetime import datetime

import markups

import requests
from bs4 import BeautifulSoup

TOKEN = «»

logging.basicConfig(level=logging.INFO)

bot = Bot(token=TOKEN)
dp = Dispatcher(bot)

db = Database('database.db')
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    if not db.user_exists(message.from_user.id):
        db.add_user(message.from_user.id)
```

```

        await bot.send_message(message.from_user.id, "Введите ваше ФИО:")
        db.set_signup_status(message.from_user.id, "setfio")
    else:
        await bot.send_message(message.from_user.id, "Вы уже
зарегистрированы!", reply_markup=nav.mainMenu)

```

```

@dp.message_handler()
async def handle_text_message(message: types.Message):
    if message.chat.type == 'private':
        if message.text == 'Профиль студента':
            fio = db.get_fio(message.from_user.id)
            birthdate = db.get_birthdate(message.from_user.id)
            contacts = db.get_contacts(message.from_user.id)
            faculty = db.get_faculty(message.from_user.id)
            nickname = db.get_nickname(message.from_user.id)

            if fio:
                await bot.send_message(message.from_user.id, f"ФИО: {fio}")
            else:
                await bot.send_message(message.from_user.id, "ФИО не задано")

            if birthdate:
                await bot.send_message(message.from_user.id, f"Дата рождения:
{birthdate}")
            else:
                await bot.send_message(message.from_user.id, "Дата рождения не
задана")

            if contacts:

```

```

        await bot.send_message(message.from_user.id, f"Контакты:
{contacts}")
    else:
        await bot.send_message(message.from_user.id, "Контакты не
заданы")

    if faculty:
        await bot.send_message(message.from_user.id, f"Факультет:
{faculty}")
    else:
        await bot.send_message(message.from_user.id, "Факультет не
задан")

    if nickname:
        await bot.send_message(message.from_user.id, f"Никнейм:
{nickname}")
    else:
        await bot.send_message(message.from_user.id, "Никнейм не
задан")

    await bot.send_message(message.from_user.id, "Что вы хотите
сделать?", reply_markup=nav.studentProfileMenu)

elif db.get_signup_status(message.from_user.id) == "setfio":
    db.set_fio(message.from_user.id, message.text)
    await bot.send_message(message.from_user.id, "Введите вашу дату
рождения (в формате ДД.ММ.ГГГГ):")
    db.set_signup_status(message.from_user.id, "setbirthdate")
elif db.get_signup_status(message.from_user.id) == "setbirthdate":
    db.set_birthdate(message.from_user.id, message.text)

```

```

        await bot.send_message(message.from_user.id, "Введите ваши
контакты (номер телефона, email и т.д.):")
        db.set_signup_status(message.from_user.id, "setcontacts")
    elif db.get_signup_status(message.from_user.id) == "setcontacts":
        db.set_contacts(message.from_user.id, message.text)
        await bot.send_message(message.from_user.id, "Введите ваш
факультет:")
        db.set_signup_status(message.from_user.id, "setfaculty")
    elif db.get_signup_status(message.from_user.id) == "setfaculty":
        db.set_faculty(message.from_user.id, message.text)
        await bot.send_message(message.from_user.id, "Введите ваш
никнейм:")
        db.set_signup_status(message.from_user.id, "setnickname")
    elif db.get_signup_status(message.from_user.id) == "setnickname":
        if len(message.text) > 60:
            await bot.send_message(message.from_user.id, "Превышено
максимальное количество символов!")
        elif '@' in message.text or '/' in message.text:
            await bot.send_message(message.from_user.id, "Вы ввели
запрещенный символ!")
        else:
            db.set_nickname(message.from_user.id, message.text)
            db.set_signup_status(message.from_user.id, "done")
            await bot.send_message(message.from_user.id, "Регистрация
прошла успешно!", reply_markup=nav.mainMenu)
    elif message.text == 'Редактирование профиля':
        await bot.send_message(message.from_user.id, "Выберите, что
хотите отредактировать:",
                                reply_markup=nav.editProfileMenu)

```

```

elif message.text == 'Редактировать ФИО':
    await bot.send_message(message.from_user.id, "Введите новое
ФИО:")

    db.set_signup_status(message.from_user.id, "editfio")
elif message.text == 'Редактировать дату рождения':
    await bot.send_message(message.from_user.id, "Введите новую дату
рождения (в формате ДД.ММ.ГГГГ):")

    db.set_signup_status(message.from_user.id, "editbirthdate")
elif message.text == 'Редактировать контакты':
    await bot.send_message(message.from_user.id, "Введите новые
контакты:")

    db.set_signup_status(message.from_user.id, "editcontacts")
elif message.text == 'Редактировать факультет':
    await bot.send_message(message.from_user.id, "Введите новый
факультет:")

    db.set_signup_status(message.from_user.id, "editfaculty")
elif message.text == 'Редактировать никнейм':
    await bot.send_message(message.from_user.id, "Введите новый
никнейм:")

    db.set_signup_status(message.from_user.id, "editnickname")

elif db.get_signup_status(message.from_user.id) == "editfio":
    db.set_fio(message.from_user.id, message.text)
    db.set_signup_status(message.from_user.id, "done")
    await bot.send_message(message.from_user.id, "ФИО успешно
обновлено!", reply_markup=nav.studentProfileMenu)
elif db.get_signup_status(message.from_user.id) == "editbirthdate":
    db.set_birthdate(message.from_user.id, message.text)
    db.set_signup_status(message.from_user.id, "done")

```

```

        await bot.send_message(message.from_user.id, "Дата рождения
успешно обновлена!",

                                reply_markup=nav.studentProfileMenu)
    elif db.get_signup_status(message.from_user.id) == "editcontacts":
        db.set_contacts(message.from_user.id, message.text)
        db.set_signup_status(message.from_user.id, "done")
        await bot.send_message(message.from_user.id, "Контакты успешно
обновлены!",

                                reply_markup=nav.studentProfileMenu)
    elif db.get_signup_status(message.from_user.id) == "editfaculty":
        db.set_faculty(message.from_user.id, message.text)
        db.set_signup_status(message.from_user.id, "done")
        await bot.send_message(message.from_user.id, "Факультет успешно
обновлен!",

                                reply_markup=nav.studentProfileMenu)
    elif db.get_signup_status(message.from_user.id) == "editnickname":
        if len(message.text) > 60:
            await bot.send_message(message.from_user.id, "Превышено
максимальное количество символов!")
        elif '@' in message.text or '/' in message.text:
            await bot.send_message(message.from_user.id, "Вы ввели
запрещенный символ!")
        else:
            db.set_nickname(message.from_user.id, message.text)
            db.set_signup_status(message.from_user.id, "done")
            await bot.send_message(message.from_user.id, "Никнейм
успешно обновлен!",

                                    reply_markup=nav.studentProfileMenu)

    elif message.text == 'Посещаемость и успеваемость студента':

```

```

await bot.send_message(message.from_user.id,
                        "Здесь вы можете узнать о своей посещаемости и
успеваемости, зафиксировать какие у вас есть задолжности, так же узнать
расписание и курсы текущего семестра: [ссылка](https://moodle.pnzgu.ru/)",
                        parse_mode='Markdown')
elif message.text == 'Институты ПГУ':
    week_number = datetime.now().isocalendar()[1]
    current_week = "Первая неделя" if week_number % 2 == 1 else
"Вторая неделя"
    schedule_text = f"Текущая неделя: {current_week}\n\nВыберите
институт:"
    await bot.send_message(message.from_user.id, schedule_text,
reply_markup=nav.facultyMenu)
elif message.text == 'Поступающим':

    await bot.send_message(message.from_user.id,
                            "Здесь вы найдете ифнормацию для поступающих:
[ссылка](https://pnzgu.ru/Abitur/abitur_2023)",
                            parse_mode='Markdown')

    file_path =
'C:\\Users\\zelen\\OneDrive\\Документы\\pravila_priema_bs_2023.pdf'
    with open(file_path, 'rb') as file:
        await message.reply_document(file)
elif message.text == 'Наука':
    await bot.send_message(message.from_user.id,
                            "Здесь вы можете получить множество полезных
ресурсов, ссылок на различные сайты помогающие в обучении, а так же
конкурсы, гранты, на иновационные проекты, множество научных изадний,
мероприятий на справочные и научные материалы и на НИИФиПИ:
[ссылка](https://science.pnzgu.ru/)",

```

```

        parse_mode='Markdown')
    elif message.text == 'Контакты ПГУ':
        contacts_text = "Контакты ПГУ:\n\nE-mail:
cnit@pnzgu.ru\nТелефон: +7 (8412) 66-64-19\nАдрес: 440026, г. Пенза, ул.
Красная, 40\nСоциальная сеть Вконтакте: https://vk.com/pnzgu\n\nСсылка на
местоположение объекта в приложении 2gis:
[ссылка](https://2gis.ru/penza/geo/5911958058440784/45.003972351551056,53.18
122453061585)"
        await bot.send_message(message.from_user.id, contacts_text,
parse_mode='Markdown')
    elif message.text == 'Медицинский институт':
        await bot.send_message(message.from_user.id, "Открываю веб-сайт
Медицинский институт: [ссылка](https://i_med.pnzgu.ru)",
        parse_mode='Markdown')
    elif message.text == 'Юридический институт':
        await bot.send_message(message.from_user.id, "Открываю веб-сайт
Юридический институт: [ссылка](https://fyur.pnzgu.ru)",
        parse_mode='Markdown')
    elif message.text == 'Институт экономики и управления':
        await bot.send_message(message.from_user.id, "Открываю веб-сайт
Институт экономики и управления: [ссылка](https://feiu.pnzgu.ru)",
        parse_mode='Markdown')
    elif message.text == 'Институт международного сотрудничества':
        await bot.send_message(message.from_user.id, "Открываю веб-сайт
Институт международного сотрудничества: [ссылка](https://ims.pnzgu.ru)",
        parse_mode='Markdown')
    elif message.text == 'Институт физической культуры и спорта':
        await bot.send_message(message.from_user.id, "Открываю веб-сайт
Институт физической культуры и спорта: [ссылка](https://ffkis.pnzgu.ru)",
        parse_mode='Markdown')

```

```

elif message.text == 'Политехнический институт':
    await bot.send_message(message.from_user.id, "Открываю веб-сайт
Политехнический институт: [ссылка](https://politeh.pnzgu.ru)",
                           parse_mode='Markdown')
elif message.text == 'Педагогический институт Им В.Г Белинского':
    await bot.send_message(message.from_user.id, "Открываю веб-сайт
Педагогический институт Им В.Г Белинского: [ссылка](https://ppi.pnzgu.ru)",
                           parse_mode='Markdown')
elif message.text == 'Институт непрерывного образования':
    await bot.send_message(message.from_user.id, "Открываю веб-сайт
Институт непрерывного образования: [ссылка](https://iito.pnzgu.ru)",
                           parse_mode='Markdown')
elif message.text == 'Назад':
    await bot.send_message(message.from_user.id, 'Вы вернулись в
главное меню.', reply_markup=markups.mainMenu)
elif message.text == 'Новости':
    await get_news(message.from_user.id)
else:
    await bot.send_message(message.from_user.id, 'Неизвестная
команда.')
else:
    await bot.send_message(message.from_user.id, 'Привет! Я работаю
только в личных сообщениях.')
@dp.message_handler()
async def get_news(user_id: int):
    url = "https://pnzgu.ru/news"
    try:
        response = requests.get(url)
        response.raise_for_status()
        soup = BeautifulSoup(response.content, 'html.parser')

```

```

news_list = soup.find_all('div', class_='news-list-item')
if news_list:
    for news in news_list:
        title = news.find('div', class_='news-list-item-title').text.strip()
        date = news.find('div', class_='news-list-item-date').text.strip()
        link = news.find('a', class_='news-list-item-link')['href']
        message_text = f"📰 {title}\n📅 {date}\n🔗 {link}"
        await bot.send_message(user_id, message_text)
    else:
        await bot.send_message(user_id, "На данный момент нет новостей.")

except requests.exceptions.RequestException as e:
    await bot.send_message(user_id, "Ошибка при получении новостей.")

def get_latest_news():
    url = "https://example.com/news"
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    news_items = soup.find_all("div", class_="news-item")
    latest_news = []
    for news in news_items[:3]:
        title = news.find("h2").text.strip() #
        content = news.find("p").text.strip() #
        latest_news.append({"title": title, "content": content})
    return latest_news

btnNews = KeyboardButton("Новости")
mainMenu = ReplyKeyboardMarkup(resize_keyboard=True)
mainMenu.add(btnProfile, btnEditProfile, btnAttendance, btnInstitutes,
btnApplicants, btnScience, btnContacts, btnNews)

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

```

```

async def handle_news(message: Message):
    news = get_latest_news()
    response = "Последние новости:\n\n"
    for item in news:
        response += f"{item['title']}\n{item['content']}\n\n"
    await message.answer(response)
if __name__ == '__main__':
    executor.start_polling(dp, skip_updates=True)

```

Файл db.py

```

import sqlite3

```

```

class Database:

```

```

    def __init__(self, db_file):
        self.connection = sqlite3.connect(db_file)
        self.cursor = self.connection.cursor()
        self.create_table()

```

```

    def create_table(self):

```

```

        self.cursor.execute("""
            CREATE TABLE IF NOT EXISTS users (
                user_id INTEGER PRIMARY KEY,
                setfio TEXT,
                setbirthdate TEXT,
                setcontacts TEXT,
                setfaculty TEXT,
                setnickname TEXT,
                signup_status INTEGER DEFAULT 0
            )

```

```

        """)
        self.connection.commit()

    def add_user(self, user_id):
        self.cursor.execute("INSERT INTO users (user_id) VALUES (?)",
        (user_id,))
        self.connection.commit()

    def user_exists(self, user_id):
        self.cursor.execute("SELECT 1 FROM users WHERE user_id=?",
        (user_id,))
        result = self.cursor.fetchone()
        return bool(result)

    def set_signup_status(self, user_id, status):
        self.cursor.execute("UPDATE users SET signup_status=? WHERE
        user_id=?", (status, user_id))
        self.connection.commit()

    def get_signup_status(self, user_id):
        self.cursor.execute("SELECT signup_status FROM users WHERE
        user_id=?", (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]
        return None

    def set_fio(self, user_id, fio):
        self.cursor.execute("UPDATE users SET setfio=? WHERE user_id=?",
        (fio, user_id))

```

```

        self.connection.commit()

    def get_fio(self, user_id):
        self.cursor.execute("SELECT setfio FROM users WHERE user_id=?",
        (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]
        return None

    def set_birthdate(self, user_id, birthdate):
        self.cursor.execute("UPDATE users SET setbirthdate=? WHERE
        user_id=?", (birthdate, user_id))
        self.connection.commit()

    def get_birthdate(self, user_id):
        self.cursor.execute("SELECT setbirthdate FROM users WHERE
        user_id=?", (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]
        return None

    def set_contacts(self, user_id, contacts):
        self.cursor.execute("UPDATE users SET setcontacts=? WHERE
        user_id=?", (contacts, user_id))
        self.connection.commit()

    def get_contacts(self, user_id):

```

```

        self.cursor.execute("SELECT    setcontacts    FROM    users    WHERE
user_id=?", (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]
        return None

    def set_faculty(self, user_id, faculty):
        self.cursor.execute("UPDATE    users    SET    setfaculty=?    WHERE
user_id=?", (faculty, user_id))
        self.connection.commit()

    def get_faculty(self, user_id):
        self.cursor.execute("SELECT    setfaculty    FROM    users    WHERE
user_id=?", (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]
        return None

    def set_nickname(self, user_id, nickname):
        self.cursor.execute("UPDATE    users    SET    setnickname=?    WHERE
user_id=?", (nickname, user_id))
        self.connection.commit()

    def get_nickname(self, user_id):
        self.cursor.execute("SELECT    setnickname    FROM    users    WHERE
user_id=?", (user_id,))
        result = self.cursor.fetchone()
        if result:
            return result[0]

```

```

        return None

Файл markups.py

from aiogram.types import ReplyKeyboardMarkup, KeyboardButton

btnProfile = KeyboardButton("Профиль студента")
btnEditProfile = KeyboardButton("Редактирование профиля")
btnAttendance = KeyboardButton("Посещаемость и успеваемость студента")
btnInstitutes = KeyboardButton("Институты ПГУ")
btnApplicants = KeyboardButton("Поступающим")
btnScience = KeyboardButton("Наука")
btnContacts = KeyboardButton("Контакты ПГУ")
btnNews = KeyboardButton("Новости")
btnFaculty1 = KeyboardButton("Медицинский институт")
btnFaculty2 = KeyboardButton("Юридический институт")
btnFaculty3 = KeyboardButton("Институт экономики и управления")
btnFaculty4 = KeyboardButton("Институт международного сотрудничества")
btnFaculty5 = KeyboardButton("Институт физической культуры и спорта")
btnFaculty6 = KeyboardButton("Политехнический институт")
btnFaculty7 = KeyboardButton("Педагогический институт Им В.Г Белинского")
btnFaculty8 = KeyboardButton("Институт непрерывного образования")
btnBack = KeyboardButton("Назад")
editProfileMenu = KeyboardButton("Редактирование профиля")

mainMenu = ReplyKeyboardMarkup(resize_keyboard=True)
mainMenu.add(btnProfile, btnEditProfile, btnAttendance, btnInstitutes, btnApplicants, btnScience, btnContacts, btnNews)

```

```
facultyMenu = ReplyKeyboardMarkup(resize_keyboard=True)
facultyMenu.add(btnFaculty1, btnFaculty2, btnFaculty3, btnFaculty4,
btnFaculty5, btnFaculty6, btnFaculty7, btnFaculty8, btnBack)
```

Файл runtime.txt

worker: python main.py

aiogram==2.25.1

aiohttp==3.8.4

aiosignal==1.3.1

asgiref==3.6.0

async-timeout==4.0.2

attrs==23.1.0

distlib==0.3.6

Django==4.2.1

Flask==2.3.2

Flask-Navigation==0.2.0

multidict==6.0.4

numpy==1.24.3

ПРИЛОЖЕНИЕ Б

ПРЕЗЕНТАЦИЯ

ВКР: «ПРОГРАММА ДЛЯ ИНФОРМАЦИОННОЙ ПОДДЕРЖКИ СТУДЕНТОВ»

Руководитель:

к.т.н., доцент Федюнин Р.Н.

Выполнил:

Студент группы 21ВВм1 Харитонов Д.А.

: : : : : : : :

: : Цель работы:

- Разработать и отладить чат-бота Telegram для информационной поддержки студентов и абитуриентов ПГУ основанные на языке Python с использованием базы данных для хранения пользователей
- В функционал которого будет входить

Регистрация пользователя

Редактирование профиля,

Посещаемость и успеваемость студента

Институты ПГУ,

Наука,

Контакты ПГУ

Актуальные новости.

:: Актуальность и аналоги

- В целях оптимизации и актуализации эффективности взаимодействия сотрудников приемной комиссии и работников, принято решение разработать инновационного чат -бота. Данная инициатива была вдохновлена стремлением улучшить процесс передачи информации, который до сих пор требовал значительного времени и усилий со стороны сотрудников, несмотря на доступность той же самой информации в открытых источниках.

- Аналоги :

Приложение «ЭИОС ПГУ» ,

Сайт Пензенского Государственного Университета ,

Сайт приемной комиссии ПГУ .

:: Стек технологий используемых в проекте:



Язык программирования Python



Библиотека Aioogram



Среда разработки PyCharm



База данных SQLite



Библиотека Telebot



Мессенджер Telegram

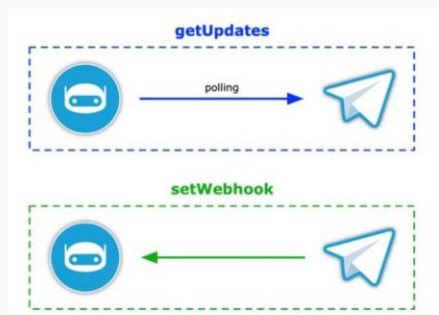
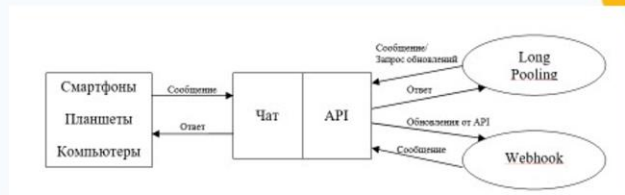
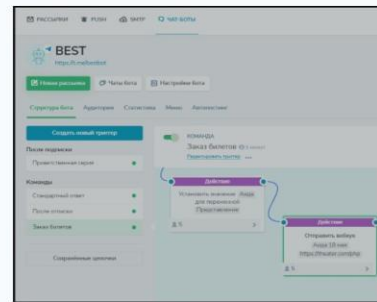


Облачный-сервер Herkoku

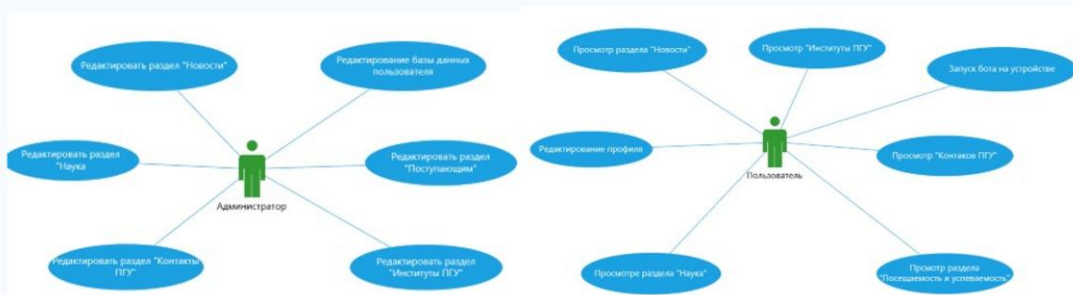


Bot_Father
Telegram Bot Api

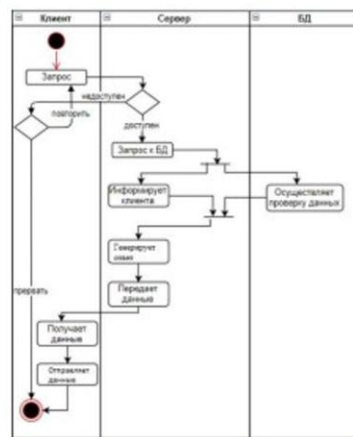
:: Технология разработки



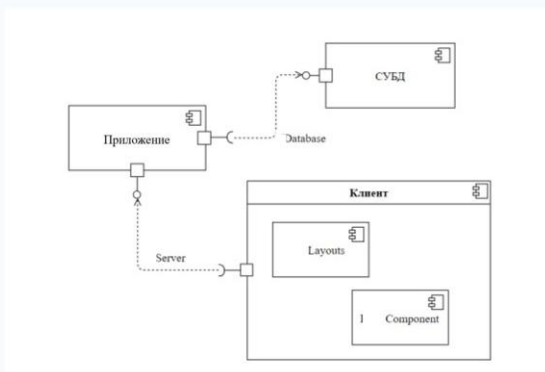
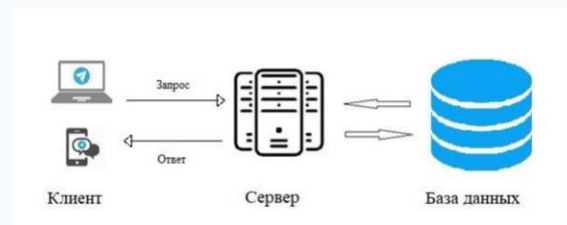
:: Диаграмма прецедентов и компонентная структура



:: Диаграмма деятельности процесса



:: Диаграмма компонентов и архитектура



Регистрация пользователя

Введите ваше ФИО: 13:26 ✓

Введите вашу дату рождения (в формате ДД.ММ.ГГГГ): 13:29 ✓

Введите ваши контакты (номер телефона, email и т.д.): 13:29 ✓

Введите ваш факультет: 13:29 ✓

Введите ваш никнейм: 13:30 ✓

Регистрация прошла успешно! 13:30 ✓

Профиль студента 18:43 ✓

ФИО: Харитонов Даниил Алексеевич 18:43 ✓

Дата рождения: 14.01.2000 18:43 ✓

Контакты: 89273726724 daniil16611@gmail.com 18:43 ✓

Факультет: ФБИТ 18:43 ✓

Никнейм: Даниил 18:43 ✓

Меню

Профиль студента Редактирование профиля Посещаемость и успеваемос...

Институты ПГУ Поступающим Наука

Контакты ПГУ Новости

Главное меню и функции:

Вы уже зарегистрированы! 18:47 ✓

Меню

Профиль студента Редактирование профиля Посещаемость и успеваемос...

Институты ПГУ Поступающим Наука

Контакты ПГУ Новости

Текущая неделя: Первая неделя

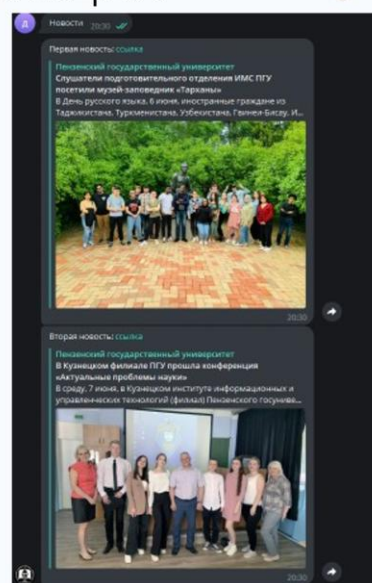
Выберите институт: 20:26

Меню

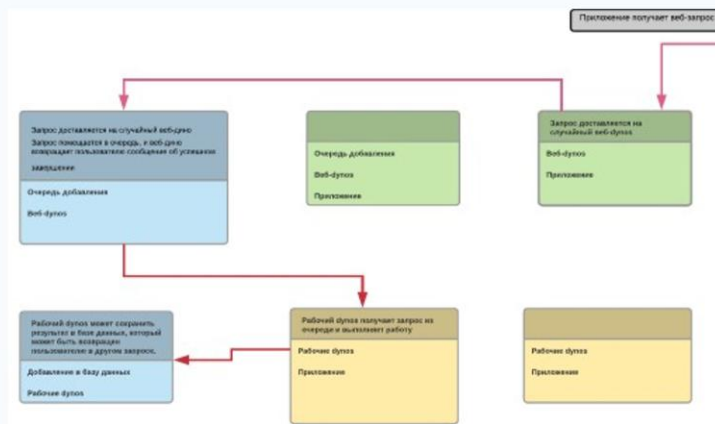
Медицинский институт Юридический институт Институт экономики и управления

Институт международного сотрудничества Институт физической культуры и спорта Политехнический институт

Педагогический институт Им В.Г. Белинского Институт непрерывного образования Назад

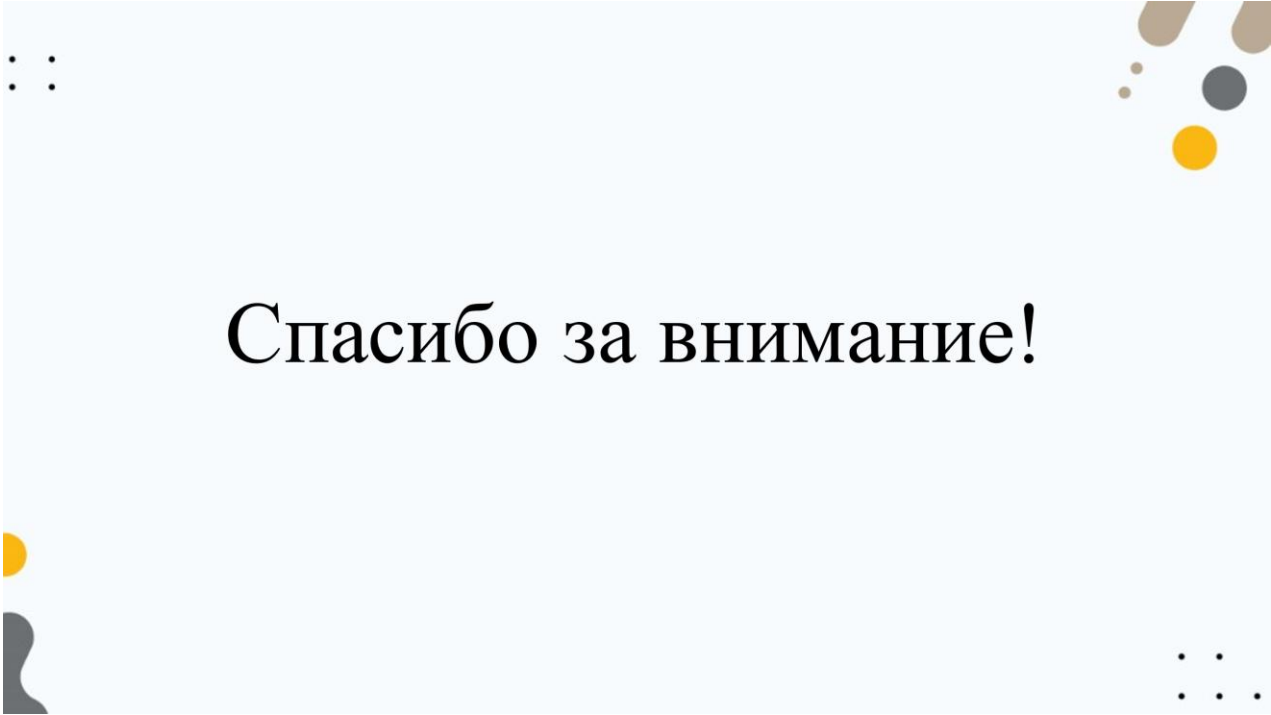


Контейнерная модель Heroku



Заключение:

В ходе разработки проекта были решены актуальные проблемы взаимодействия вуза со студентами и абитуриентами, Была разработана концептуальная модель программного продукта, Был проведен анализ актуальности, аналогов и технологий, В итоге реализована программа для информационной поддержки чат-бот ПГУ.

A light blue rectangular area with decorative corner elements. Top-left: two small grey dots. Top-right: a cluster of brown, grey, and yellow circles. Bottom-left: a yellow circle and a grey shape. Bottom-right: a 2x3 grid of small grey dots.

Спасибо за внимание!