

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА ИНФОРМИРОВАНИЯ СТУДЕНТОВ
ПО ОРГАНИЗАЦИОННЫМ ВОПРОСАМ В
ПРОЦЕССЕ ОБУЧЕНИЯ

Студент


(подпись, дата)

31.05.22

Холодков Данила Валерьевич
(ФИО полностью)

Руководитель


(подпись, дата)

31.05.22

С.А. Зинкин
(фамилия, инициалы)

Нормоконтролёр

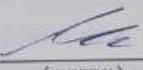

(подпись, дата)

31.05.22

А.С. Бычков
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 02.06.2022 г. № 13)

Заведующий кафедрой

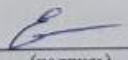

(подпись)

М.А. Митрохин
(фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от

20 июня 2022 № 13)

Секретарь ГЭК


(подпись)

Е.И. Гурин
(фамилия, инициалы)

Пенза, 2022

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

Факультет
вычислительной техники

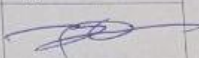
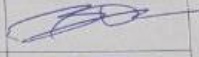
Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» 04 2022 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Холодков Данила Валерьевич гр. 18ВВ2 факультета вычислительной техники.
2. Тема работы: «Система информирования студентов по организационным вопросам в процессе обучения»
Тема утверждена приказом ПГУ № 357/о от "26" апреля 2022 г.
3. Руководитель работы д.т.н., профессор каф. ВТ ПГУ Зинкин С.А.
4. Задание на работу (назначение разработки, исходные данные и т.п.)
Разработать приложение «Система информирования студентов по организационным вопросам в процессе обучения», выполняющее обработку, обмен и структурирование информации, обеспечивающее взаимодействие пользователей, повышающую эффективность процесса информирования студентов в целом. Приложение должно иметь клиент-серверную архитектуру. Клиентская часть должна обладать графическим интерфейсом. Система должна включать в себя функционал сбора, хранения и обработки данных, расположенных в таблицах Google Sheets.
5. Перечень подлежащих разработке вопросов
Требуется произвести анализ технического задания, предметных областей, изучить существующие аналоги и используемые библиотеки, а также обосновать выбор технологий, применяемых при разработке, разработать собственное клиент-серверное приложение, отладить и провести тестирование разработанного приложения.

6. Календарный график выполнения работы

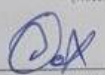
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ технического задания	32 ч.	26.04.2022-30.04.2022	
2.	Анализ предметной области	76 ч.	01.05.2022-02.05.2022	
3.	Выбор технологий, применяемых при проектировании и разработки	76 ч.	03.05.2022-07.05.2022	
4.	Разработка приложения «Система информирования студентов по организационным вопросам в процессе обучения»	72 ч.	08.05.2022-11.05.2022	
5.	Отладка и тестирование приложения	32 ч.	12.05.2022-29.05.2022	
6.	Оформление пояснительной записки	36 ч.	30.05.2022-31.05.2022	

Дата выдачи задания "26" апреля 2022 г.

Руководитель бакалаврской работы

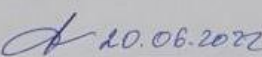

26.04.22 Зинкин С.А.
(подпись, дата)

Задание к исполнению принял студент


26.04.22 Холодков Д.В.
(подпись, дата)

Работу к защите допустить

Декан факультета


10.06.2022 Фионова Л.Р.
(подпись, дата)

Резюме

Пояснительная записка содержит 78 страниц формата А4, 1 таблица, 29 рисунков, 5 источников, 3 приложения.

ИНФОРМИРОВАНИЕ, СЕРВЕР, JSON, СБОР И ОБМЕН ДАННЫМИ, РАБОТА С API

Цель работы: разработка информационного приложения, позволяющего студентам получать информацию об организации учебного процесса, а преподавателям информировать студентов, используя выборку по группе.

Объектом разработки является приложение, обеспечивающее выполнение следующих функций: получение актуальных данных, взаимодействие с API интернет-ресурсов, обработка данных в формате JSON, асинхронная работа обработчиков для запросов клиентов, контроль доступа пользователям.

В результате проделанной работы было разработано приложение, удовлетворяющее поставленным задачам.

В пояснительной записке приведён анализ предметной области, обзор технологии разработки и отладки программы.

					ПГУ 1.090301.43.001 ПЗ		
Изм.	Лист	№ докум	Подпись	Дата			
Разраб.		Холодков Д.В.			Лит.	Лист	Листов
Провер.		Зинкин С.А.				2	78
Н. Контр.		Бычков А.С.			ФВТ, каф. ВТ гр. 18ВВ2		
Утверд.		Митрахин М.А.					
					Система информирования студентов по организационным вопросам в процессе обучения		
					Пояснительная записка		

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Система информирования студентов по организационным вопросам в процессе обучения	8
1.1 Анализ тех задания.....	8
1.1.1 Постановка задачи.....	8
1.1.2 Актуальность решения	9
1.2 Анализ аналогов	10
1.2.1 Web-сайт	10
1.2.2 Электронная почта	11
1.2.3 Социальные сети	11
1.3 Выбор используемых программных средств	12
1.3.1 Выбор языка программирования	12
1.3.2 Выбор библиотеки	15
1.3.3 Хранение данных	16
1.3.4 Клиент приложения	16
1.3.5 Выбор среды разработки	16
1.3.6 Система контроля версий	19
1.4 Описание архитектуры решения	21
1.4.1 Клиент приложения	21
1.4.2 Сервер приложения.....	22
1.4.2.1 Описание основных функций сервера.	23
1.4.2.2 Описание классов сервера.....	26
1.4.3 Хранение данных.....	28
1.5 Тестирование и отладка.....	32
1.5.1 Тестирование сервера	32
1.5.2 Тестирование web-приложения <i>Apps Script</i>	32
1.5.3 Тестирование клиента приложения.....	35

2 Безопасность жизнедеятельности.....	49
2.1 Потенциально-опасные факторы.....	49
Заключение	52
Список использованных источников	53
Приложение А – <i>UML</i> -диаграммы	54
Приложение Б – Листинг программы	59
Приложение В – Презентация.....	72

Перечень сокращений и обозначений

API (Application Program Interface) – программный интерфейс приложения
IDE (Integrated development environment) – интегрированная среда разработки
JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на языке программирования *JavaScript*
UML (Unified Modeling Language) – унифицированный язык моделирования
URL (Uniform Resource Locator) – унифицированный указатель ресурса
БД – база данных
ПО – программное обеспечение
ПЭВМ – персональная электронно-вычислительная машина

					ПГУ 1.090301.43.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

В современном мире информация стала ценным ресурсом. Задачами по сбору, обработке и хранению информации занимается огромное количество специалистов, и их количество растёт, обгоняя специалистов, связанных с физическим трудом. Грамотная работа с информацией позволяет упростить и автоматизировать рабочие процессы увеличивая общую эффективность любой организации.

Информирование студентов – это важная задача. Насколько не был бы усерден студент, если он не узнает вовремя, что экзамен был перенесён, то это может обернуться для него проблемой. В такой сложной организации как Пензенский Государственный Университет изменения происходят очень часто и этого не избежать. Планы, графики, помещения и т.д. Всё это организовано в большую и сложную систему, где всё взаимосвязано. Небольшие изменения приводят к реорганизации многих частей системы. Кроме того, могут происходить бытовые ситуации, когда преподаватель заболел, занятие проходит в другом помещении или в другое время, от чего студенты приходят не в то время или не в то место, и проводят часы в ожидании, вместо полезной деятельности.

Ещё два десятка лет назад это было настоящей проблемой. Чтобы вовремя всё узнавать, студентам нужно было ежедневно проверять доску объявлений на стене университета, а преподавателям лично связываться с группой. Это усложняло процесс обучения и негативно влияло на успеваемость, потому что студенты и преподаватели тратили много времени на простые организационные вопросы. Сейчас технологии развились достаточно чтобы решить данную проблему. У каждого человека есть доступ в интернет и множество инструментов для того, чтобы быть на связи и получать информацию в любой точке нашей страны.

С развитием информационных технологий многие задачи решаются гораздо быстрее, что даёт студентам больше свободного времени и они могут поддерживать хорошую успеваемость, при этом чаще участвуя в творческой и научной деятельности университета.

В данной работе будет рассмотрена реализация приложения «Система информирования студентов по организационным вопросам в процессе обучения». Целью создания приложения является упрощение процесса распространения информации среди студентов с помощью современных технологических решений.

Практическая значимость данной работы заключается в получении завершённого и автономного продукта, готового к использованию в качестве дополнительного инструмента информирования пользователей.

1 Система информирования студентов по организационным вопросам в процессе обучения

1.1 Анализ тех задания

Суть данного проекта заключается в том, чтобы вынести такие данные как расписание занятий преподавателей, расписание приёма задолженностей, данные и контакты преподавателей в таблицы *Google Sheets*. Затем разработать программу-сервер, которая по запросу обращается к этим таблицам и выводит необходимые данные в упорядоченном виде. Также реализовать возможность для преподавателей выполнять запрос о передачи объявления студентам. Интерфейс, через который преподаватели и студенты будут взаимодействовать с программой, планируется выполнить в виде текстового диалога в мессенджере Telegram с помощью технологии Telegram Bot.

1.1.1 Постановка задачи

Разработать программу, реализующую следующие функции:

- сбор данных, внесенных в таблицы *Google Sheets*;
- получение запросов пользователей из диалога приложения Telegram;
- отправка данных, запрошенных пользователем;
- отправка сообщения преподавателями отдельным группам.

Для реализации функции сбора данных, внесённых в таблицы *Google Sheets*, требуется организовать на сервере возможность получить данные из таблицы *Google Sheets*, используя проект *AppsScript*. Для работы с таблицами необходимо выполнить развёртывание веб-приложения и написать программу приложения.

Для получения запросов пользователей из диалога приложения *Telegram* необходимо создать ботов *Student* и *Professor*, чтобы получить их

коды доступа, с помощью которых мы получим возможность управлять действиями бота.

Для реализации отправки данных, запрошенных пользователем, собранные данные необходимо обработать и настроить корректную выдачу пользователю.

Чтобы реализовать отправку преподавателями сообщений для отдельных групп, следует подготовить все сценарии взаимодействия пользователя с ботом и с соответствующими действиями в ответ на запросы пользователя.

1.1.2 Актуальность решения

Данное решение использует принципиально новый способ взаимодействия пользователя с приложением. Удобный и давно освоенный диалог в мессенджерах превращается в командную строку для основной программы, которая выполняется на отдельном сервере. В отличие от сайта, данному решению не требуется сложная разработка интерфейса, что упрощает и удешевляет разработку в целом, и даёт возможность сосредоточиться на других функциях. Для системы выполняющей функции по распространению информации, возможностей данного интерфейса достаточно.

Если сравнивать данное решение с электронной почтой, то мы получаем расширенные возможности. Такие как выборка получателей сообщения по группе и автономный ответ на стандартные вопросы студентов.

В мире происходят события, влияющие на все сферы, в том числе на сектор информационных технологий. Увеличилась вероятность ограничения работы и блокировок сервисов, а также временных неполадок в связи с изменениями в компаниях, которые обеспечивают работу тех или иных приложений и услуг. Поэтому основным аргументом, подтверждающим актуальность данного решения, служит то, что дополнительный способ

					ПГУ 1.090301.43.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

информирования студентов диверсифицирует риски. Другими словами, дополнительный независимый способ информирования студентов не позволит остаться без инструментов информирования студентов, в случае выхода из строя уже используемых.

1.2 Анализ аналогов

Аналогами систем информирования являются любые инструменты позволяющие поддерживать связь и своевременно передавать информацию без необходимости посещать университет. Рассмотрим несколько аналогов и проанализируем преимущества и недостатки.

1.2.1 Web-сайт

Самым распространённым на протяжении долгого времени остаётся веб-сайт, на которой выкладываются все новости и объявления. Простой вариант сайта является аналогом доски объявлений на стене университета.

Плюсы данного решения:

- сайт не ограничен в своих возможностях и при желании можно добавлять любой функционал, вплоть до внедрения собственной площадки с трансляцией лекций, поэтому данное решение отличается универсальностью;
- подобно функционалу, сайт можно настроить под определённые задачи и добиться высокого удобства использования;
- собственный сайт не зависит от сторонних платформ, что позволяет добиться высокой надёжности и безопасности.

Минусы данного решения:

- чем больше функций выполняет сайт, тем сложнее его разработка, что требует времени и средств;
- чтобы сайт легко и удобно было поддерживать, а именно размещать объявления и информацию, необходимо разработать интерфейс для администрирования, что увеличивает сложность разработки.

- если с сервером возникают неполадки, то сайт перестаёт работать, что является большой проблемой, решение которой приводит к усложнению и удорожанию всей системы;

- размещение информации на сайте не гарантирует, что все увидят эту информацию, что является ключевым недостатком, когда речь идёт о неотложном объявлении;

- студенты не могут через веб-сайт обратиться с вопросом к отдельным преподавателям, а добавление такой функции увеличивает сложность и затраты разработки.

1.2.2 Электронная почта

Электронная почта позволяет связаться напрямую с каждым отдельным студентом, и донести до него необходимую информацию.

Плюсы данного решения:

- простейшие сообщения, которые отправляются лично каждому студенту, что не требует никаких вложений и подготовки;

- есть выбор кому отправлять сообщение, поэтому могут решаться вопросы, касающиеся отдельных лиц;

- не только преподаватели могут связываться со студентами, но и студенты могут обратиться с индивидуальным вопросом;

- если почтовый сервис выйдет из строя, то можно воспользоваться другим, что исключает проблемы с доступом.

Минусы данного решения:

- электронная почта — это общий инструмент для обмена сообщениями, в нём нельзя реализовать какой-либо функционал для упрощения конкретных учебных задач.

1.2.3 Социальные сети

Рассмотрение данного аналога производится на примере социальной сети «ВКонтакте».

На данный момент – это самый успешный способ информирования студентов. Он включает в себя все плюсы веб-сайта и электронной почты. Публичные страницы и группы выполняют функции веб-сайта, а личные сообщения и беседы роль электронной почты.

Плюсы данного решения:

- является готовым решением, имеющим широкий функционал;
- администрирование публичной страницы сводится к написанию объявления и нажатию кнопки «Опубликовать», с чем может справиться любой человек, даже далёкий от работы с компьютером.
- есть гарантия получения студентом объявления в личных сообщениях в виде пометки «Прочитано»;
- крупная социальная сеть принимает все меры для того, чтобы добиться высокой отказоустойчивости, что минимизирует неполадки;

Минусы данного решения:

- «ВКонтакте» имеет широкий функционал, но это функции, которые направлены на решение общих задач, не имеющих отношения к обучению, что приводит к ситуации когда для поиска такой простой информации как расписание, необходимо найти её среди всех объявлений и сообщений, а затем убедиться, что она не устарела;
- использование сторонней, общедоступной платформы не гарантирует конфиденциальности.

1.3 Выбор используемых программных средств

1.3.1 Выбор языка программирования

Сервер приложения разработан на языке программирования *Python*, реализация языка: *Python 3*.

Python – высокоуровневый язык программирования, автором которого является Гвидо ван Россум. Данный язык имеет минималистичный синтаксис

и динамическую типизацию данных, поэтому код выглядит лаконично и легко читается, что означает низкий порог вхождения в освоении языка.

Python является объектно-ориентированным – *Python* поддерживает объектно-ориентированный стиль или технику программирования, которая инкапсулирует код в объектах [1].

Данный язык является интерпретируемым. Интерпретатор – это такая программа, которая выполняет другие программы. Когда вы пишете программу на языке *Python*, интерпретатор *Python* читает вашу программу и выполняет составляющие ее инструкции [2].

Интерпретируемые языки программирования имеют свой ряд преимуществ и недостатков в сравнении с компилируемыми языками программирования. Из преимуществ можно выделить: кроссплатформенность, динамическая типизация данных, пошаговое отслеживание выполнения программы и использование динамической области видимости и замыканий, а также отсутствие этапа компиляции программы перед выполнением. Перечисленные преимущества обеспечивают быструю разработку приложений и упрощают поддержку уже разработанных приложений, что приводит к удешевлению проекта в целом.

Из недостатков интерпретируемых языков программирования я хотел бы выделить только один: скорость выполнения. Скорость выполнения кода интерпретируемого языка уступает компилируемому языку программирования. Разница в скорости может быть стократной величины. Цирюлик Олег Иванович в рамках своей работы «Производительность языков программирования» провёл тестирование скорости выполнения кода на разных языках программирования. Результаты тестирования отражены в таблице 1.1.

Таблица 1.1 – результат сравнения скорости выполнения тестовых программ на разных языках.

Фактор скорости	Язык
1 ... 2	<i>GCC C/C++, Clang C++, Java, Kotlin</i>
2 ... 5	<i>GCC Go, GoLang, Scala, PureBasic</i>
5 ... 10	-
10 ... 20	<i>Ocaml</i>
20 ... 50	<i>Scheme</i>
50 ... 100	<i>Ruby, PHP, Haskell</i>
100 ... 200	<i>JavaScript, Python 2/3, Lua, Euphoria</i>
200 ... 500	<i>Tcl</i>
> 500	<i>bash</i>

В результате тестирования было обнаружено, что разница в скорости выполнения интерпретируемого языка *Python* относительно компилируемого языка *GCC C/C++* варьируется от ста до двух ста раз [3]. Это связано с тем, что код интерпретируемого языка требует предварительной обработки перед каждым выполнением. Данный недостаток приводит к ряду других неудобств при разработке, такие как использование дополнительных анализаторов кода и более строгое отношение при оформлении кода. Кроме того, интерпретатор языка при выполнении программы должен всегда находиться в памяти, что увеличивает требуемый объем ресурсов.

Данный недостаток нивелируется развитием современных вычислительных систем. Для большинства проектов, в том числе и крупных, стало выгоднее упрощать и удешевлять процесс разработки и поддержки. Если скорости выполнения программы становится недостаточно, то докупается дополнительное оборудование. Благодаря развитию технологий,

вычислительные машины стали дешевле и доступнее, в то время как труд специалистов в области информационных технологий с каждым годом оценивается дороже. Это привело к ситуации, когда заказчикам дешевле оплатить разработку приложения на языке *Python*, но докупить дополнительное оборудование, чем оплачивать разработку на языке *GCC C/C++*, что потребует больше времени, которое заказчику потребуется оплатить, но сэкономить на оборудовании.

Таким образом язык программирования *Python* стал отличным выбором для разработки проектов, в которых приоритетнее простота разработки и поддержки, и идеальным выбором для одноразовых программ и прототипов.

На основании всего вышеперечисленного, при выполнении данной Выпускной Квалификационной работы было принято решение использовать язык программирования *Python*, реализация языка: *Python 3*.

1.3.2 Выбор библиотеки

Для упрощения работы с *Telegram Bot API* необходимо воспользоваться специализированной библиотекой. За время существования *Telegram Bot API* было создано много библиотек, но одни из самых эффективных и популярных это: *pyTelegramBotAPI*, *aiogram*, *python-telegram-bot*. Это свободно распространяемые библиотеки, которые развиваются сообществом разработчиков. Каждый желающий может получить исходные файлы, внести изменения и пользоваться модернизированной под свои нужды библиотекой, так же есть возможность внести свои изменения в общий проект. Таким образом, общими усилиями данные библиотеки продолжают развиваться.

Все выделенные библиотеки имеют весь необходимый функционал для реализации проекта. Для разработки проекта была выбрана асинхронная библиотека *aiogram*. Основанием выбора служит наличие опыта работы с данной библиотекой.

1.3.3 Хранение данных

Данные о расписании занятий, данные и контакты преподавателей и студентов, а также другие вспомогательные данные решено хранить в таблицах *Google Sheets*. Редактировать данные таблицы можно в браузере с компьютера или с телефона из любой точки мира. Есть возможность ограничить доступ к этим таблицам и выделить доступ отдельным людям, что исключает влияние сторонних лиц, и решает проблемы с безопасностью. Благодаря такому решению нет необходимости исправлять данные на сервере.

Есть возможность экспорта и импорта файлов *Excel*, а также написание скриптов, которые в будущем могут полностью автоматизировать работу с таблицами. Данные, получаемые из таблицы представлены в формате *JSON*.

JSON или *JavaScript Object Notation* – это легкий текстовый открытый стандарт, разработанный для удобочитаемого обмена данными. Соглашения, используемые *JSON*, известны программистам, включая *C*, *C++*, *Java*, *Python*, *Perl* [4].

1.3.4 Клиент приложения

Интерфейсом для взаимодействия с пользователями является *Telegram Bot*. Пользователь пишет запрос в диалоге с ботом, этот запрос получает программа-сервер через *Telegram Bot API* и отправляет ответ пользователю. Возможности *Telegram Bot* позволяют реализовать интерфейс в виде удобных кнопок, по нажатию на которые отправляется запрос. Также в нём присутствуют некоторые средства форматирования текста, что позволит сделать вывод данных удобочитаемым.

1.3.5 Выбор среды разработки

В рамках данного проекта было решено использовать бесплатные *IDE*, доступные для скачивания и установки на территории Российской Федерации. Основанием для данного требования служит нестабильная политическая ситуация в мире, а в частности прецеденты запрета на

распространение программного обеспечения на территории Российской Федерации.

Наличие средств отладки кода – это обязательное требование к *IDE*. Программы на *Python* можно писать, используя стандартное приложение *Windows* – блокнот, а также любой другой текстовый редактор. Однако, отсутствие отладчика существенно влияет на скорость и качество разработки. В процессе разработки часто случаются ситуации, когда код работает неверно. В таких случаях программист должен перепроверить код и потратить часы на поиск ошибки в коде. Отладчик предоставляет интерфейс для отслеживания состояния переменных и возможность построчного выполнения кода, что значительно ускоряет процесс поиска ошибок.

Система управления пакетами – программное обеспечение, используемое для упрощения процесса установки, удаления и контроля версий установленных пакетов. Наличие в *IDE* такого инструмента позволяет оперативно работать с пакетами и отслеживать их наличие, что значительно упрощает процесс разработки в сравнении с использованием ручного скачивания пакетов и сети интернет.

Наличие навигация в проекте. Данное требование обусловлено потребностью в работе с несколькими файлами проекта. Наличие интерфейса для отслеживания и переключения между файлами проекта стали нормой, и отсутствие данного инструмента способно затормозить разработку из-за постоянного ручного открытия и закрытия файлов проекта.

Во время разработки используется две машины. На первой установлена операционная система *Windows 10*, а на второй машине установлена операционная система *Linux*. Таким образом, для упрощения работы, требуется *IDE*, которая поддерживает данные операционные системы.

Существует множество сред разработки подходящим данным требованиям. Самые распространённые из них: *Eclipse*, *PyCharm Community Edition*, *Visual Studio Code*.

Eclipse - свободная интегрированная среда разработки модульных кроссплатформенных приложений. Была выпущена в 2001 году и из выделенных *IDE*, она самая старшая. Несмотря на это, она до сих пор развивается и широко используется. Свою популярность данная *IDE* получила за счет того, что изначально была разработана как платформа для разработки расширений. Это означает, что любой разработчик имеет возможность расширить её своими модулями. За двадцать один год истории было разработано огромное количество модулей для широкого спектра задач, и это делает *Eclipse* универсальной *IDE*. Из недостатков данной *IDE* следует выделить сложность освоения и организации рабочего пространства. Широкий выбор расширений требует определённой подготовки и изучения возможностей расширений, прежде чем сделать выбор.

Visual Studio Code – легковесный редактор исходного кода. Данный редактор предоставляет самые широкие возможности персонализированных настроек: настройка горячих клавиш, внешний вид интерфейса, файлы конфигурации, установка расширений. При этом отличается нетребовательностью к ресурсам вычислительной машины. С 2018 года появилось расширение *Python* для *Visual Studio Code* с открытым исходным кодом. Оно предоставляет разработчикам широкие возможности для редактирования, отладки и тестирования кода. Отлично подходит для работы с небольшими проектами.

PyCharm Community Edition – бесплатная версия интегрированной среды разработки *PyCharm* для языка программирования *Python*. Данная *IDE* предоставляет широкий набор инструментов для разработки проектов на языке программирования *Python* и доступна как на *Windows*, так и на *Linux*. Относительно предыдущих *IDE*, *PyCharm* узконаправленная и полноценно поддерживает разработку только на языке *Python*. Из преимуществ данной *IDE* следует выделить легкую настройку и быстрое освоения. Так как основные инструменты уже настроены и готовы к работе сразу после

установки. Есть возможность, но нет необходимости загружать плагины и расширения, а потом настраивать их, окружение и среда уже настроены и готовы к работе. Это является отличительной чертой *PyCharm* по сравнению с ранее рассмотренными *IDE*. На рисунке 1 представлен интерфейс *PyCharm*.

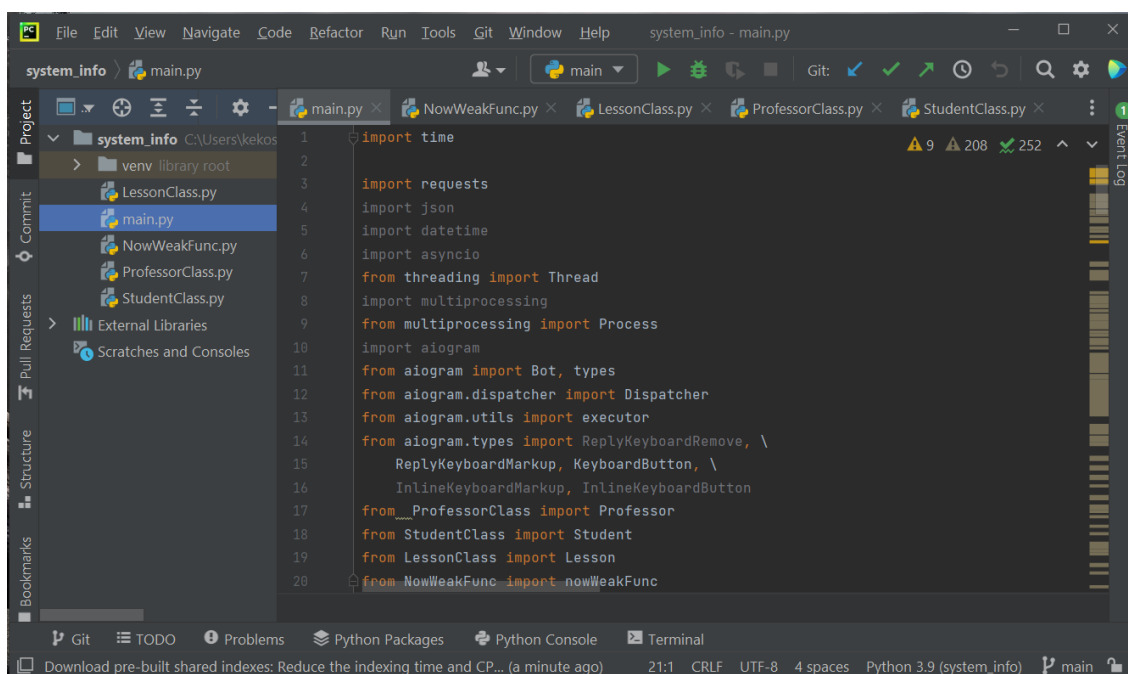


Рисунок 1 – Интерфейс *PyCharm Community Edition*

На основании вышеперечисленных аргументов, для разработки данного проекта была выбрана среда разработки *PyCharm Community Edition*.

1.3.6 Система контроля версий

Система контроля версий — это система, записывающая изменения в файл или набор файлов в течение времени и позволяющая вернуться позже к определенной версии [5]. При работе с одним проектом на разных машинах требуется передача данных и постоянный контроль версии проекта. Кроме того, часто появляется ситуация, когда необходимо отменить ряд изменений и вернуться к определённому моменту разработки. Подобные задачи решаются с помощью систем управления версиями. Для данного проекта была использована распределённая система управления версиями *git*. Работа

с данной системой осуществляется как через командную строку, но существуют доступные клиенты с графическим интерфейсом. В данном проекте был использован клиент *GitHub Desktop*. Он предоставляет графический интерфейс для работы с репозиториями, а также предоставляет сведения о изменениях в проекте. Репозиторий был создан в сервисе *GitHub*, поэтому при работе с разных машин можно осуществлять синхронизацию версий проектов.

Благодаря тому, что при реализации проекта была использована эта система контроля версий, появилась история всех изменений проекта и возможность вернуться на определённый этап. На рисунке 2 представлен интерфейс приложения *Github Desktop*.

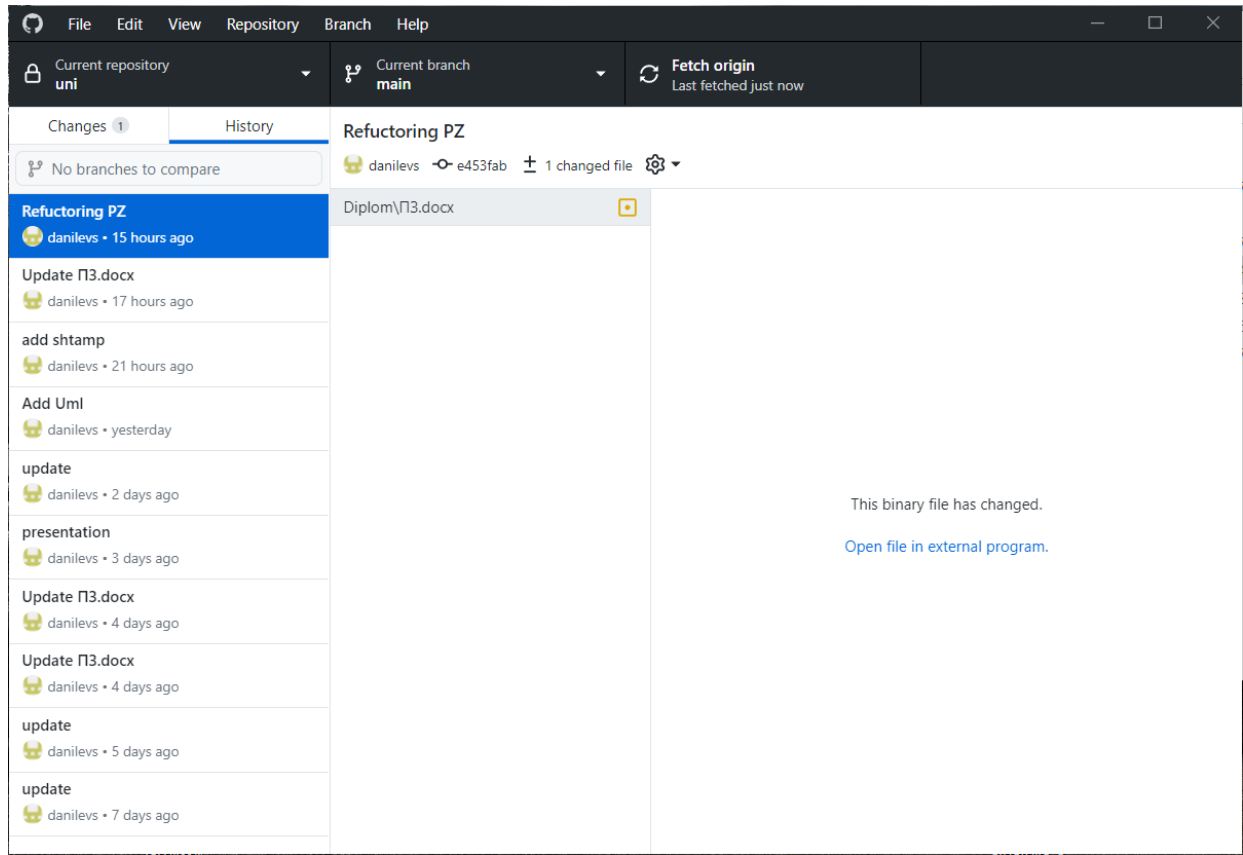


Рисунок 2 – Интерфейс *GitHub Desktop*

1.4 Описание архитектуры решения

Сбор данных из сети требует большой пропускной способности сетевой карты, а их обработка требует высокой мощности, поэтому решение делать монолитную программу для ПЭВМ является не рациональным, кроме того, такой подход сократит количество платформ, на которых можно развернуть приложение. Наиболее оптимальным решением является реализация приложения на архитектуре клиент-сервер. Архитектура клиент-сервер представляет из себя набор различных устройств: серверов, на которых развернута программа-сервер, и клиентов, на которых развернута программа-клиент. Как правило, клиенты — это обычные персональные компьютеры, планшеты, мобильные устройства или Интернет-вещи, а серверы - высокопроизводительные компьютеры, способные обеспечить обработку от нескольких тысяч (обычный сервер) до миллионов запросов в секунду (кластерные серверы).

1.4.1 Клиент приложения

В роли клиента выступает приложение *Telegram*. Пользователи пишут запрос и получают ответ в специальном диалоге, который в данном приложении называется «бот». Отправленные сообщения боту, попадают на сервера *Telegram* и хранятся там.

Функционал данного решения был расширен разработчиками *Telegram*. Помимо текстовых команд, пользователю предоставляется панель кнопок, и отдельное меню с основными командами. Панель кнопок можно перенастраивать в ответ на каждую команду пользователя, что придаёт гибкость и универсальность интерфейсу. С помощью кнопок разработчик способен предоставить удобный и интуитивно понятный интерфейс для пользователя.

Чтобы создать и получить возможность управления данным ботом, необходимо быть авторизованным пользователем приложения *Telegram* и

создать собственного бота в специальном боте *BotFather*. Написав сообщению *BotFather*, нужно проследовать инструкциям и в результате получить ссылку на диалог с ботом, и токен, с помощью которого можно управлять действиями бота. На рисунке 3 представлено создание двух ботов для реализации данного проекта.

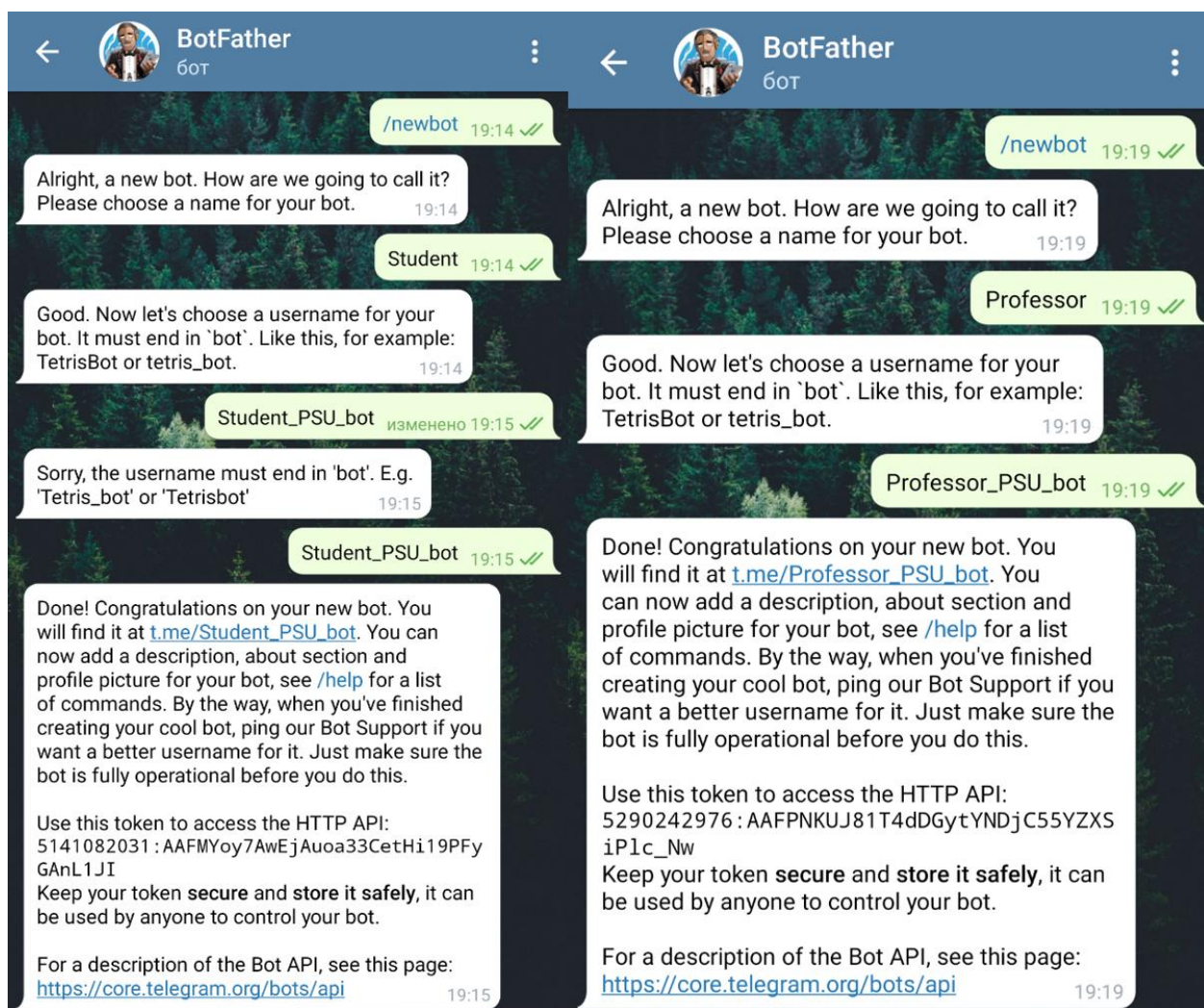


Рисунок 3 – Создание и регистрация ботов

1.4.2 Сервер приложения

Получает и обрабатывает запросы клиентов сервер, разработанный на языке *Python*. Взаимодействие с сервером *Telegram* происходит по *Telegram Bot API*. Для упрощения работы с *Telegram Bot API*, при разработке сервера использовалась библиотека *aiogram*. *Aiogram* содержит готовые функции для

взаимодействий с сервером *Telegram* по *Telegram Bot API*. Отдельные функции способны выполняться асинхронно, что стало весомым аргументом в вопросе об используемых инструментах.

Для того чтобы получить запрос клиента и обработать его, серверу необходимо отправить запрос на сервер *Telegram*, потому что сообщения пользователей хранятся там.

Если было принято сообщение, то наш сервер получает данные об отправителе и текст сообщения, которые попадают в специальные функции-обработчики.

Обработчики проверяют данные и формируют ответ пользователю. Ответ пользователю отправляется на сервера *Telegram* вместе с токеном бота и данными пользователя, тогда по данному токenu сервер *Telegram* направляет сообщение в диалог бота с соответствующим пользователем.

Пример алгоритма обработчика запроса пользователя на получение расписания:

```
@dispatcher_student_bot.message_handler(lambda message:
message.text == "Расписание")
async def without_pureel(msg: types.Message):
    print(f"Пользователь {msg.chat.id} ввёл Расписание")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Расписание"))
    keyboard.add(KeyboardButton("Какая идёт неделя?"))
    keyboard.add(KeyboardButton("Преподаватели"))
    await
msg.answer(generate_timetable_group(msg.chat.id,1),reply_markup=
keyboard,parse_mode="Markdown")
```

1.4.2.1 Описание основных функций сервера.

Update_Data() – функция, которая включает в себя алгоритм выполнения запросов к веб-приложениям с данными из таблиц *Google Sheet*, и обработку полученных данных. Для каждой таблицы разработан отдельный

алгоритм и в сумме выполняется четыре запроса к веб-приложениям. Данная функция выполняется сразу при запуске, а затем запускается каждые 5 минут в параллельном потоке. Таким образом используемые данные сохраняются в программе сервера и каждые 5 минут обновляются. Такое решение было принято для ускорения работы сервера и более оперативных ответов пользователям.

Внутри функции реализован сбор данных с таблицы расписания. Полученные от веб-приложения данные представлены в формате *JSON*, где первый объект — это первая строка, второй объект — это вторая строка и так далее.

Для работы с ними производится обработка. Сначала данные собираются в экземпляры класса *Lesson* и заносятся в список. Затем из списка удаляются одинаковые объекты, которые сформировались из-за совпадения занятия у двух преподавателей, и формируется новый объект *Lesson*, но с двумя преподавателями.

Получение данных из таблиц со списком преподавателей и студентов проще, потому что полученный от веб приложения *JSON* ответ содержит массив объектов, каждый объект которого содержит данные об одном студенте и преподавателе. Такой *JSON* файл легко обрабатывается. Пример получения данных из таблицы преподавателей:

```
response = requests.get(urlStudents)
a = response.json()
for item in a["data"]:
    arr_students.append(Student(item[0], item[1],
item[2], item[3]))
```

nowWeakFunc(day,month,year) – функция, которая вычисляет какая идёт неделя на данный момент. Учебное расписание в Пензенском Государственном университете – двухнедельное. То есть по «первой» неделе одно расписание, по «второй» неделе другое, так они чередуются. Функция

принимает 3 параметра: день, месяц, год. Параметры данной функции — это дата, соответствующая «первой» неделе. От этой даты вычисляется количество прошедших недель. Если нечётное, то значит текущей дате соответствует расписание по «второй» неделе, а если чётное, то «первой».

generate_timetable_group(param, mode) – данная функция обрабатывает массив с экземплярами класса *Lesson*, который хранит все занятия, и возвращает текст расписания. Входной параметр *mode* определяет поведение функции. Если *mode* равен 1, тогда массив занятий отсеивается по обучаемой группе, которая задаётся параметром *param* и формируется расписание группы. Если *mode* равен 2, тогда массив занятий отсеивается по преподавателю, проводящему занятие, имя которого задаётся параметром *param*, тогда результатом функции будет расписание конкретного преподавателя.

Работа функции состоит из четырёх этапов. Сначала определяется как фильтровать массив занятий, по преподавателям или по группе. Затем выборка соответствующих занятий. После чего сортировка. Сортировка по типу недели, по дню недели, а затем по времени занятия. Последний этап — это формирование текстового сообщения с расписанием.

address_delivery(string_groups) – функция, которая принимает на вход название группы, а возвращает список пользователей, которые находятся в данной группе. Работа функции представляет из себя перебор массива с экземплярами класса *Student*, в котором хранятся все студенты, указанные в таблице Студенты.

Обработчики сообщений представлены асинхронными функциями. Для каждой команды есть своя функция. Сообщение пользователя принимается диспетчером бота и подставляется по порядку в каждую функцию, если текстовое сообщение пользователя совпадает с параметром функции, тогда эта функция выполняется. Пример асинхронной функции, для обработки запроса пользователя по получению расписания:

```

@dispatcher_student_bot.message_handler(lambda message:
message.text == "Расписание")
async def process_timetable(msg: types.Message):
    print(f"Пользователь {msg.chat.id} ввёл Расписание")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Расписание"))
    keyboard.add(KeyboardButton("Какая идёт неделя?"))
    keyboard.add(KeyboardButton("Преподаватели"))
    await
msg.answer(generate_timetable_group(msg.chat.id,1),reply_markup=
keyboard,parse_mode="Markdown")

```

Если текстовое сообщение не подходит ни под одну функцию, тогда оно обрабатывается функцией без параметра. Как правило такая функция нужна для обработки неверно введенных команд и вывода предупреждения пользователю.

1.4.2.2 Описание классов сервера

Класс *Lesson* необходим для хранения информации об одном занятии. Экземпляр данного класса представляет из себя структуру, описывающую одно занятие. То есть из объекта мы можем получить название предмета, имена преподавателей, группа, день, время и тип недели. Так же класс имеет два метода: сеттер для поля *first_prof* и сеттер для поля *second_prof*. На рисунке 4 представлена диаграмма класса *Lesson*.

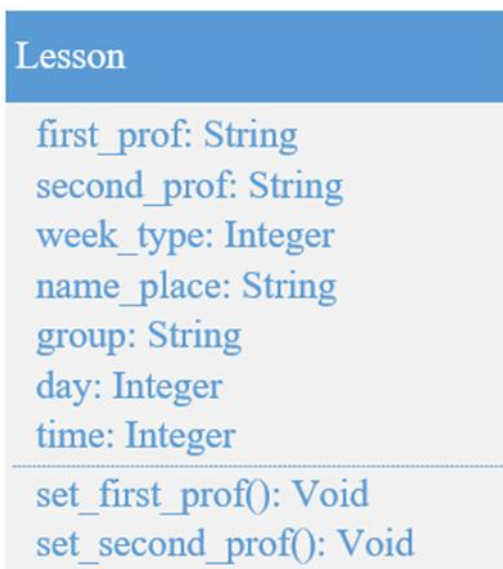


Рисунок 4 – Диаграмма класса *Lesson*

Класс *Professor* – используется для хранения информации об одном преподавателе и включает в себя такие поля как ФИО, контакты и *id* чата. На рисунке 5 представлена диаграмма класса *Professor*.



Рисунок 5 – Диаграмма класса *Professor*

Класс *Student* – используется для хранения информации об одном студенте. Имеет поля: ФИО, контакты, группа и *id* чата. На рисунке 6 представлена диаграмма класса *Student*.

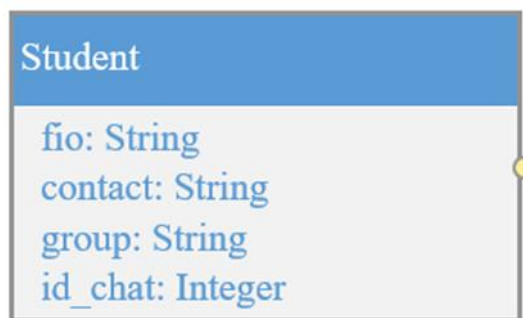


Рисунок 6 – Диаграмма класса *Student*

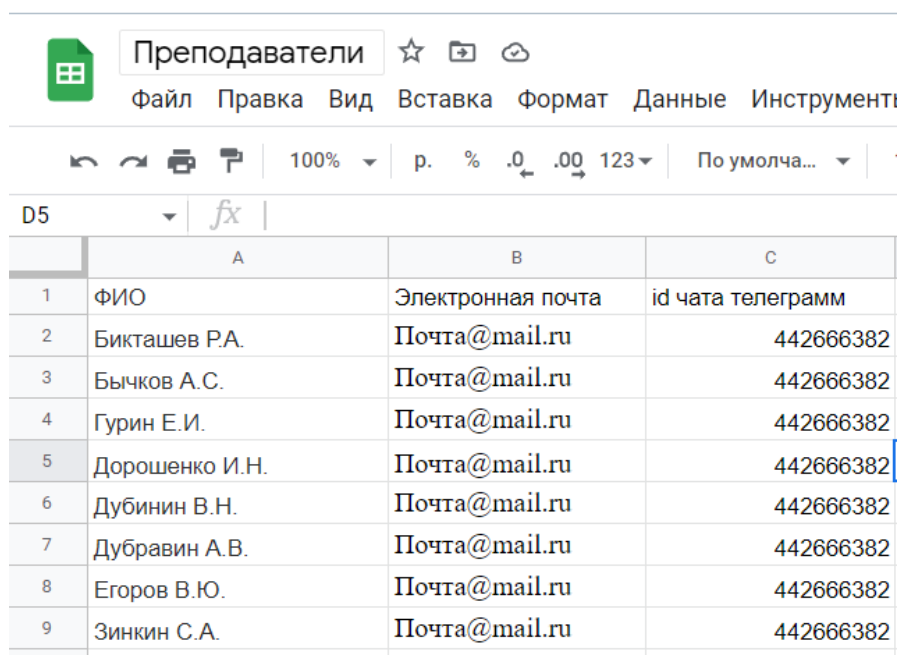
1.4.3 Хранение данных

Для хранения данных было использовано три независимые таблицы *Google Sheets*. Для хранения информации о расписании была создана таблица с именем «Расписание». На рисунке 7 представлена таблица с расписанием.

А1	В	С	Д	Е	Ф	Г	Н	И
07.06.2022	← Введите дату первой недели ДД.ММ.ГГГГ							
	Понедельник							
ФИО Преподавателей	8.00-9.35	9.50-11.25	11.40-13.15	13.45-15.20				
Митрохин М.А.							Кураторский час	20В02
Митрохина Н.Ю.								
Бикташев Р.А.							Кураторский час	20В02
Бычков А.С.								

Рисунок 7 – Таблица с расписанием

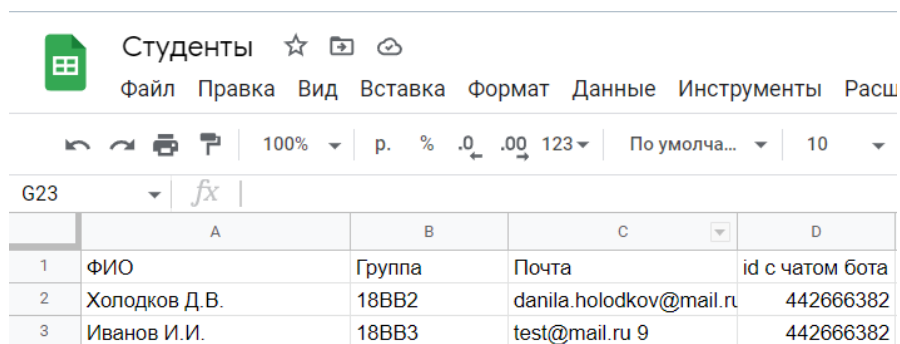
Для хранения информации о преподавателях была создана таблица с именем «Преподаватели». На рисунке 8 представлена таблица с данными преподавателей.



	A	B	C
1	ФИО	Электронная почта	id чата телеграмм
2	Бикташев Р.А.	Почта@mail.ru	442666382
3	Бычков А.С.	Почта@mail.ru	442666382
4	Гурин Е.И.	Почта@mail.ru	442666382
5	Дорошенко И.Н.	Почта@mail.ru	442666382
6	Дубинин В.Н.	Почта@mail.ru	442666382
7	Дубравин А.В.	Почта@mail.ru	442666382
8	Егоров В.Ю.	Почта@mail.ru	442666382
9	Зинкин С.А.	Почта@mail.ru	442666382

Рисунок 8 – Таблица с данными преподавателей

Для хранения информации о студентах была создана таблица с именем «Студенты». На рисунке 9 представлена таблица с данными студентов.



	A	B	C	D
1	ФИО	Группа	Почта	id с чатом бота
2	Холодков Д.В.	18BB2	danila.holodkov@mail.ru	442666382
3	Иванов И.И.	18BB3	test@mail.ru 9	442666382

Рисунок 9 – Таблица с данными студентов

Для получения доступа к данным одной таблицы по *HTTP* интерфейсу, был создан проект *Apps Script*. На рисунке 10 представлен интерфейс для работы с проектом *Apps Script*.

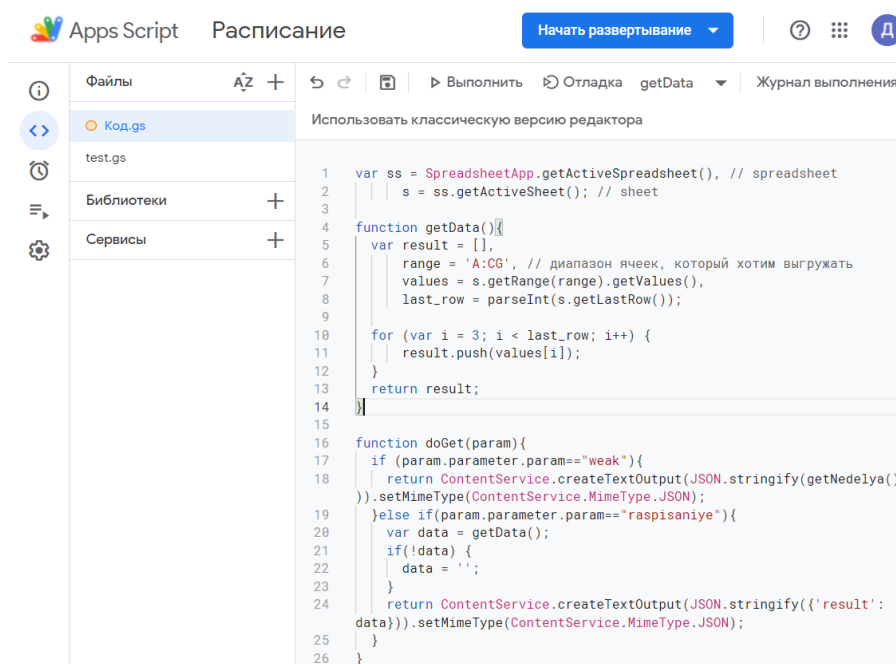


Рисунок 10 – Интерфейс App Script

В проекте реализован алгоритм на языке *Google Apps Script*, который собирает данные с таблицы, формирует из них данные в формате *JSON* и отправляет их в ответ на *GET* запрос к данному *web*-приложению. Для того чтобы данное *web*-приложение работало и обрабатывало запросы к нему по сети интернет, было выполнено развёртывание проекта на серверах *Google*. Данные о развёртывании *web*-приложения для таблицы со списком преподавателей представлены на рисунке 11.

Управление развертываниями

Активно

Конфигурация

?

✎

+

Professors

В архиве

Professors

Professors

Professors

Professors

Версия

Версия 5 от 29 мая, 15:37

Описание

Professors

Идентификатор развертывания

AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsHwwQST3LwmuzWkaHkVaFjgTCCJh...

📄

OK

Веб-приложение

URL

https://script.google.com/macros/s/AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsH...

📄

OK

Запуск от имени

Отмена

Начать развертывание

Рисунок 11 – Данные о развёртывании *web*-приложения

Аналогичные действия были выполнены с другими таблицами.

Алгоритмы для *web*-приложения состоят из двух функций.

doGet(param) – данная функция выполняется при *Get* запросе к развёрнутому *web*-приложению для данного проекта *Apps Script*. Она генерирует ответ на *Get* запрос в формате *json*.

getData() – функция, которая собирает данные из таблицы в массив построчно в заданном диапазоне.

1.5 Тестирование и отладка

1.5.1 Тестирование сервера

Сервер приложения отлажен с использованием среды программирования *PyCharm*.

Отладка проводилась с использованием построчного прохождения программы, точек останова, просмотра и анализа содержимого стеков, переменных, регистров. Также в процессе написания кода был использован инструмент *qDebug()*, позволяющий фиксировать данные на выводе приложений.

Тестирование осуществлялось в рабочем порядке, в процессе разработки, а также после завершения написания программ.

В ходе тестирования были выявлены и устранены проблемы, касающиеся обработки данных в формате *JSON*, передачи данных между клиентом и сервером, взаимодействия сервера с *web*-приложением *Apps Script*.

1.5.2 Тестирование *web*-приложения *Apps Script*

Для тестирования *web*-приложение использовался ручной метод, который заключается в обращении через адресную строку браузера по ссылке. При корректной работе *web*-приложения, *GET*-запрос должен возвращать данные из *Google* таблицы в формате *JSON*.

На рисунке 12 представлена таблица Преподаватели с тестовыми значениями.

Преподаватели ☆ 📁 🔄

Файл Правка Вид Вставка Формат Данные Инструменты Расширения Справка Последнее изменение

100% р. % .0 .00 123 По умолча... 10 В I S A 🔍 📐 📏 ☰

B9		fx				
	A	B	C	D	E	F
1	ФИО	Электронная почта	id чата телеграмм			
2	Холодков Д.В.	Почта@mail.ru	442666382			
3						
4						
5						
6						
7						
8						
9						
10						
11						

Рисунок 12 – Таблица с тестовыми данными

Для обращения к *web*-приложению следует выполнить *GET*-запрос по адресу, представленному в настройках развёртывания проекта *Apps Script*. На рисунке 13 представлены настройки развёртывания.

Управление развёртываниями

Активно	Конфигурация
Professors	Версия Версия 5 от 29 мая, 15:37
В архиве	Описание Professors
Professors	Идентификатор развёртывания
Professors	AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsHwwQST3LwmuzWkaHkVaFjgTCCJh...
Professors	Веб-приложение URL https://script.google.com/macros/s/AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsHwwQST3LwmuzWkaHkVaFjgTCCJhWqwuXU8/exec
Professors	Запуск от имени

Отмена Начать развёртывание

Рисунок 13 – Настройки развёртывания

В настройках развёртывания получен *URL*-адрес, по которому следует
 ВЫПОЛНИТЬ *GET*-запрос:

https://script.google.com/macros/s/AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsHwwQST3LwmuzWkaHkVaFjgTCCJhWqwuXU8/exec

После перехода по этой ссылке в браузере, был получен ответ от web-приложения. Данные в ответе представлены в формате *JSON* и совпадают с содержимым таблицы Преподаватели. На рисунке 14 представлен результат запроса.

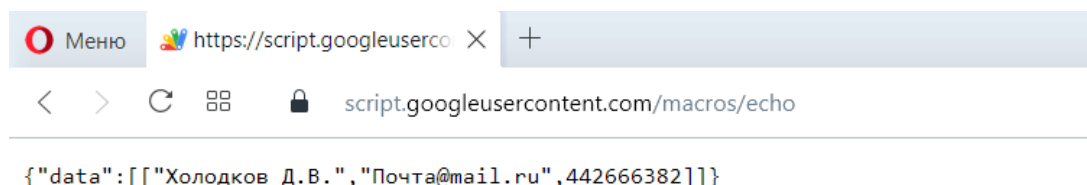


Рисунок 14 – Результат запроса

Следующим шагом были внесены изменения в таблице, а именно изменено поле ФИО у тестовой записи. На рисунке 15 представлена таблица Преподаватели с изменёнными тестовыми данными.

Преподаватели ☆ 📁 Сохранено на Диске.

Файл Правка Вид Вставка Формат Данные Инструменты Расширени

100% ▾ р. % .0 .00 123 ▾ По умолча... ▾ 10 ▾ B

B5	fx			
	A	B	C	D
1	ФИО	Электронная почта	id чата телеграмм	
2	Иванов Д.В.	Почта@mail.ru	442666382	
3				
4				

Рисунок 15 – Изменённые тестовые данные

Данные, полученные в результате нового *GET*-запроса *web*-приложению, совпадают с изменёнными данными в таблице. На рисунке 16 представлен результат нового запроса.

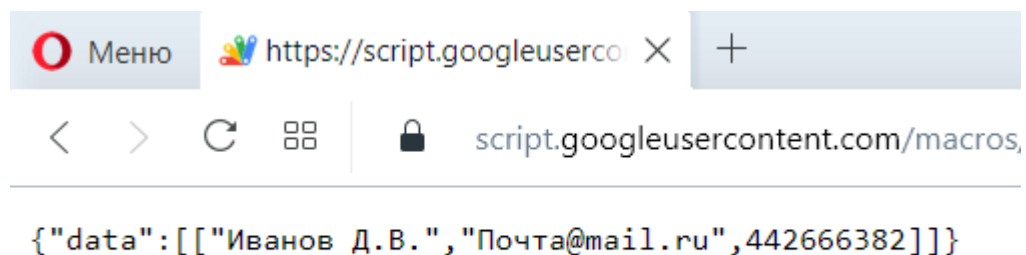


Рисунок 16 – Результат нового запроса

Результат тестирования *web*-приложения *Apps Script* совпал с ожидаемым и подтвердил корректность работы.

1.5.3 Тестирование клиента приложения

При тестировании клиента использовался ручной метод тестирования, заключающийся в проверке каждой отдельной функции приложения.

Наглядно полный набор возможностей пользователя продемонстрирован на рисунке 17.

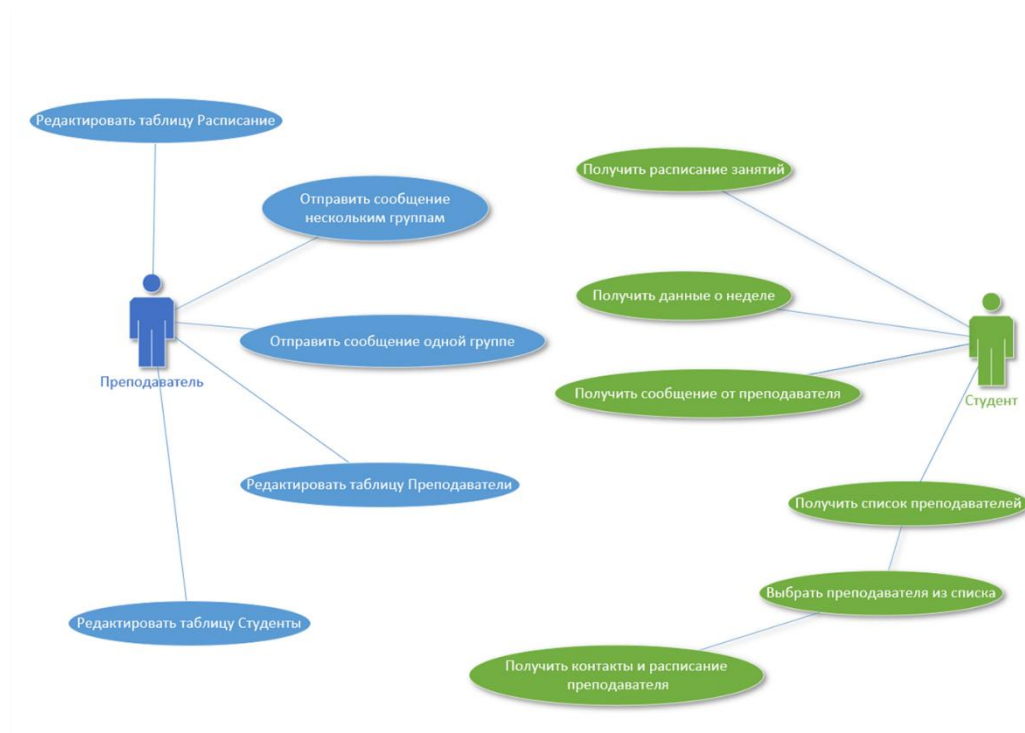


Рисунок 17 – Диаграмма вариантов использования

Для тестирования следует открыть клиент приложения *Telegram* и открыть диалог с пользователем *@Student_PSU_bot*. На рисунке 18 представлен первый запуск бота.

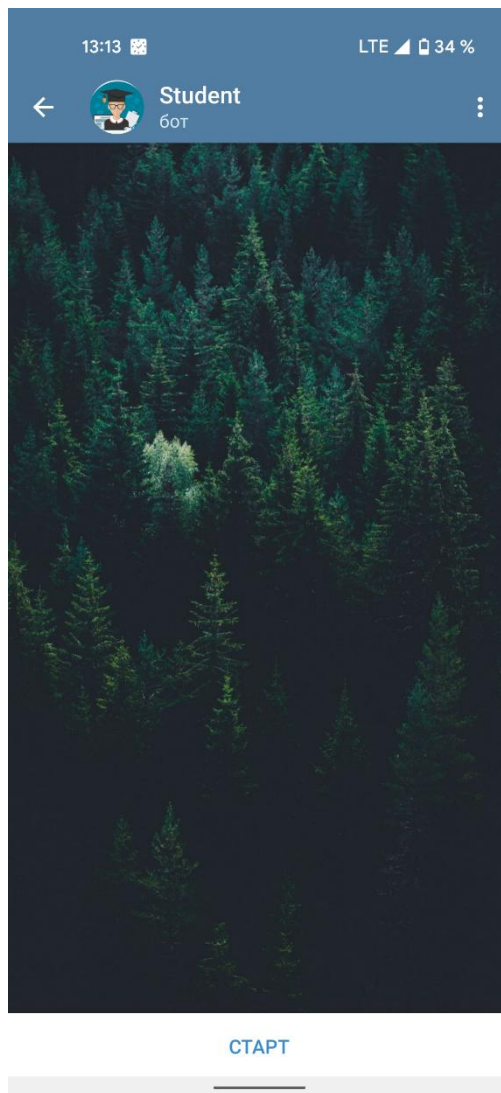


Рисунок 18 – Первый запуск бота

После выполнения запуска с помощью команды «/start», выводится сообщение о готовности бота и меню приложения. На рисунке 19 представлен результат выполнения команды «/start».

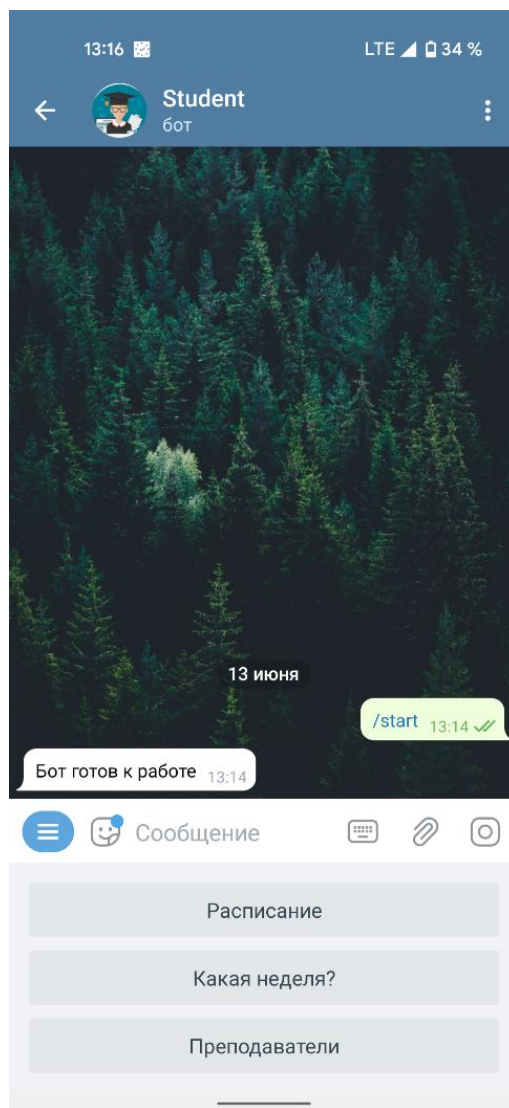


Рисунок 19 – Результат выполнения команды «/start»

Первая тестируемая функция это – Расписание. После нажатия кнопки расписание, сервер получает запрос и генерирует текстовое сообщение с расписанием. На рисунке 20 представлен результат выполнения команды «Расписание».

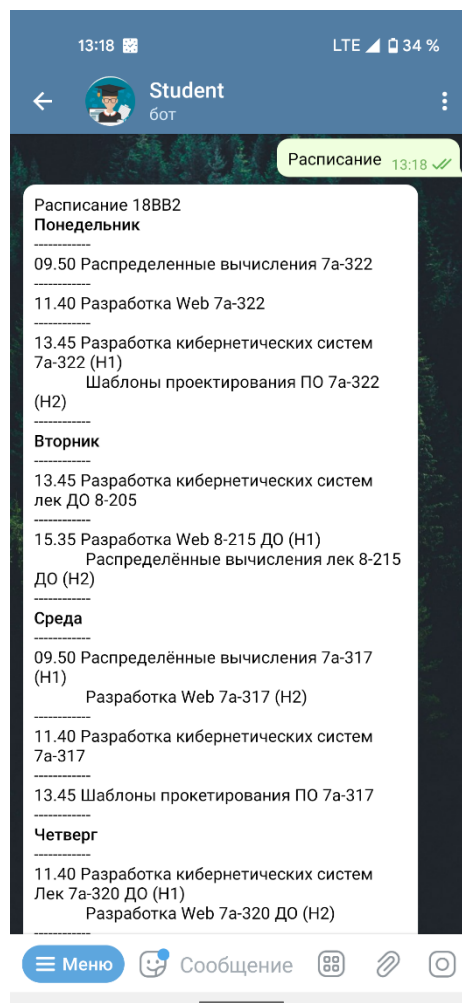


Рисунок 20 – Результат выполнения команды «Расписание»

Результат тестирования функции Расписание является корректным и совпал с ожидаемым.

Следующая тестируемая функция это – получение информации о том, какая сейчас идёт неделя «первая» или «вторая». Так как расписание в университете двухнедельное, то следует отслеживать по какой неделе смотреть расписание. После нажатия кнопки «Какая идёт неделя?», сервер обрабатывает запрос и генерирует сообщение в ответ. На рисунке 21 представлен результат нажатия кнопки «Какая идёт неделя?».

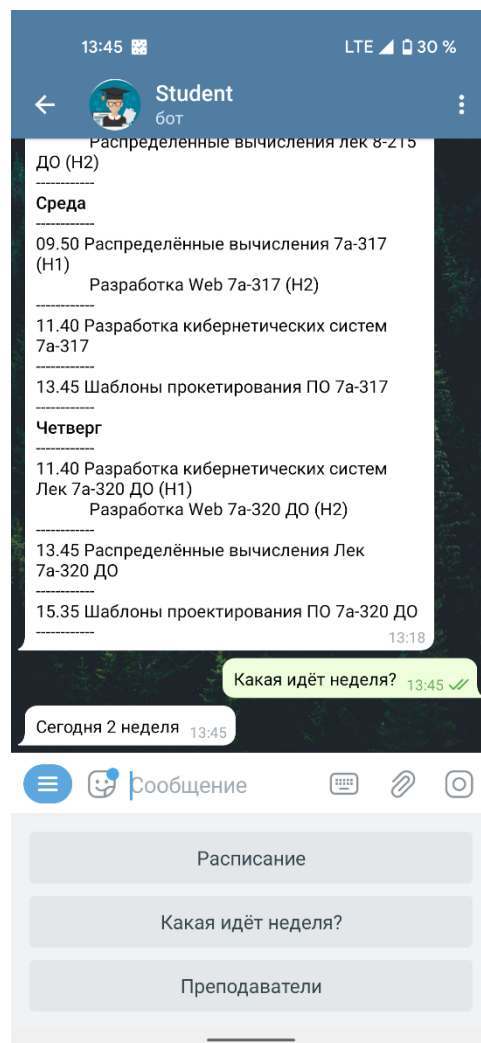


Рисунок 21 – Результат нажатия кнопки «Какая идёт неделя?»

Для проверки результата использовался официальный сайт Пензенского Государственного Университета, на котором также можно найти информацию о текущей неделе: <https://pnzgu.ru/schedule>. На рисунке 22 представлена информация о расписании на сайте.

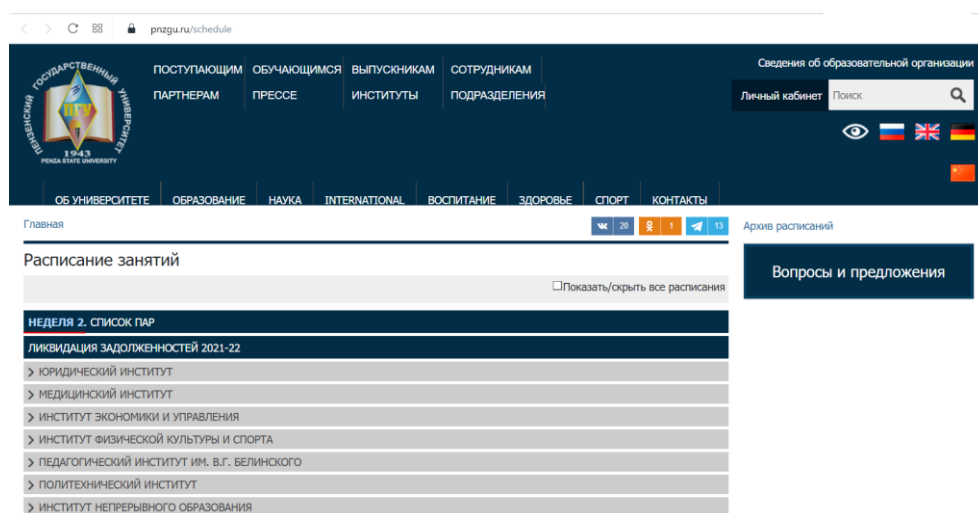


Рисунок 22 – Данные о расписании на сайте

На сайте Пензенского Государственного Университета указано «НЕДЕЛЯ 2». Результат работы приложения доказал корректность работы данной функции приложения.

Для проверки вывода списка преподавателей была нажата в меню кнопка «Преподаватели». На рисунке 23 представлен результат нажатия кнопки «Преподаватели».

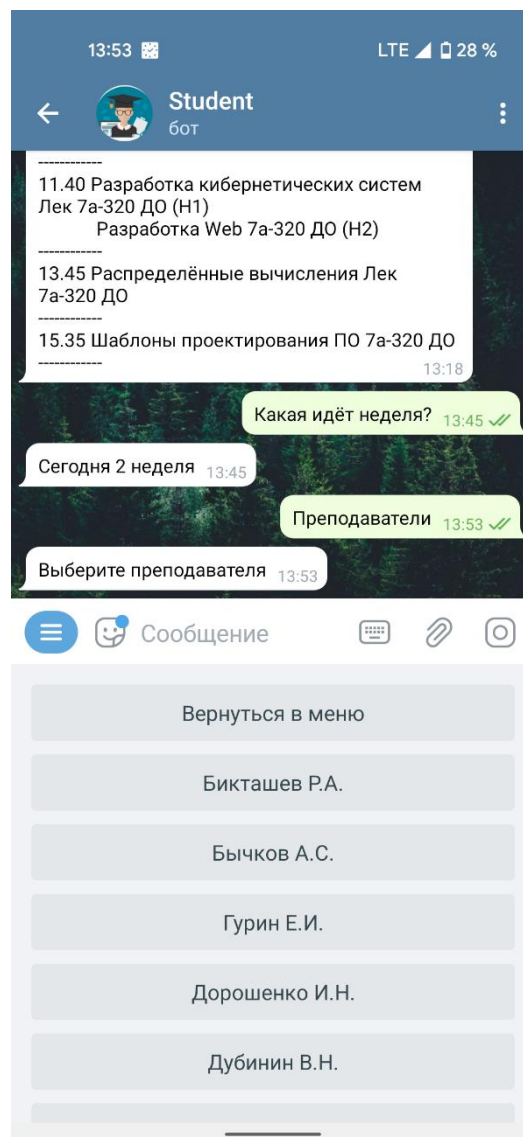


Рисунок 23 – Результат нажатия кнопки «Преподаватели»

В результате был выведен список всех преподавателей, и кнопка «Вернуться в меню». Результат работы совпал с ожидаемым.

Для проверки вывода контактных данных и расписания конкретного преподавателя, не выходя из списка следует нажать кнопку с именем интересующего преподавателя. Результат выбора преподавателя представлен на рисунке 24.

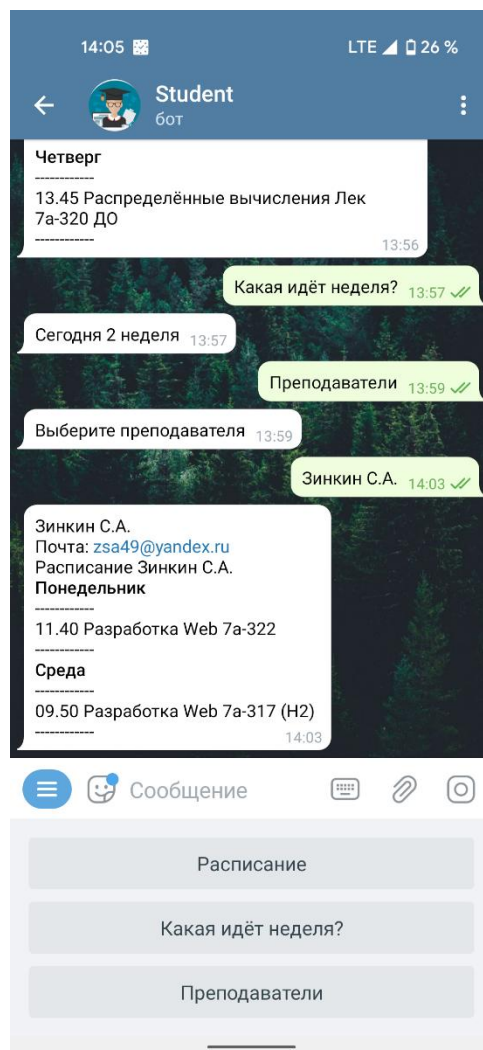


Рисунок 24 – Результат выбора преподавателя

Были выведены корректные контактные данные и расписание выбранного преподавателя. Результат совпал с ожидаемым.

Для тестирования функции отправки преподавателем сообщения используя выборку по группе, следует открыть клиент приложения *Telegram* и зайти в диалог с пользователем *@Professor_PSU_bot*. На рисунке представлен первый запуск бота.

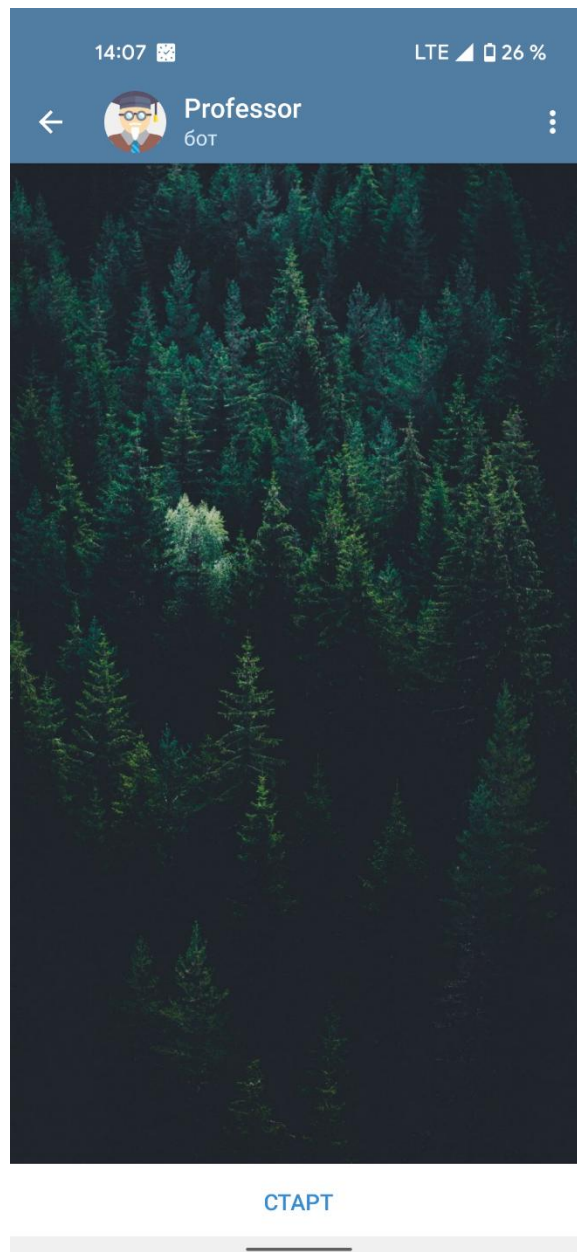


Рисунок 25 – Первый запуск бота

После выполнения команды «/start» выводится сообщение с инструкцией, в которой описываются правила отправки сообщения и корректный шаблон при использовании выборки получателей по группам. На рисунке 26 представлен результат выполнения команды «/start».

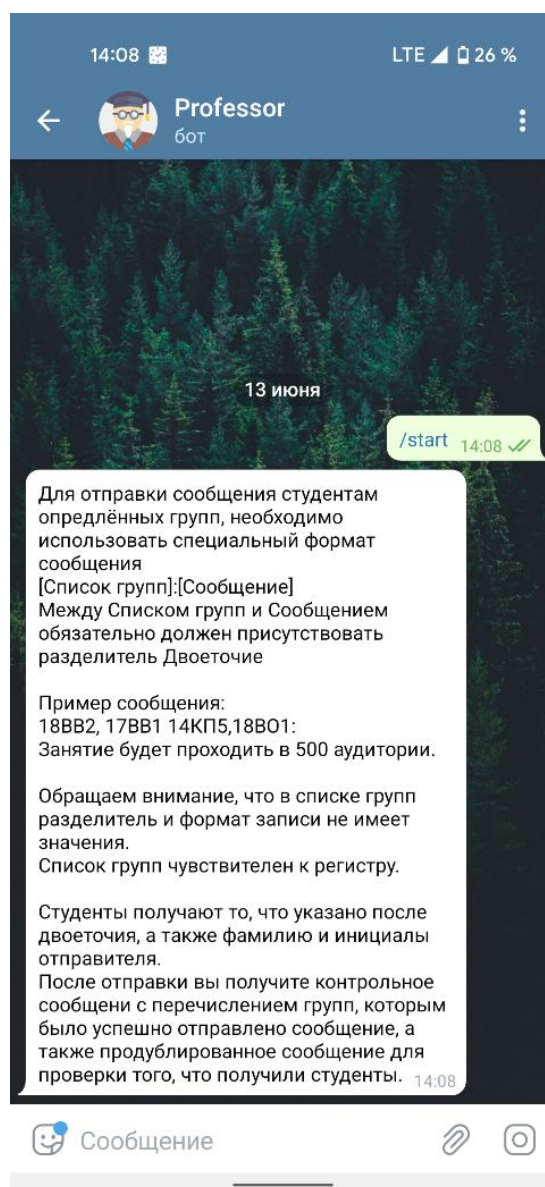


Рисунок 26 – Результат выполнения команды «/start»

Для тестирования отправки сообщения с использованием выборки, в таблицу студентов были добавлены два пользователя из различных групп. На рисунке 9 представлена таблица с тестовыми данными. Таким образом пользователь с *id* равным 442666382, будет получать сообщения, предназначенные для групп 18BB2 и 18BB3. В тестовом сообщении указаны три группы получателей: 18BB2, 18BB3, 18BB9. На рисунке 27 представлено тестовое сообщение.

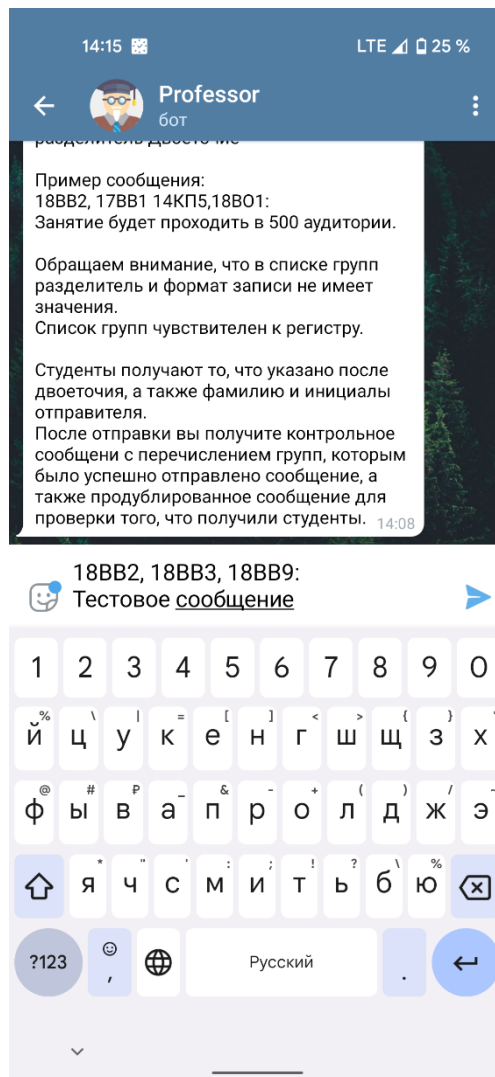


Рисунок 27 – Тестовое сообщение

Пользователи с соответствующими группами должны получить данное тестовое сообщение, а также фамилию и инициалы отправителя. На рисунке 28 представлены полученные сообщения.

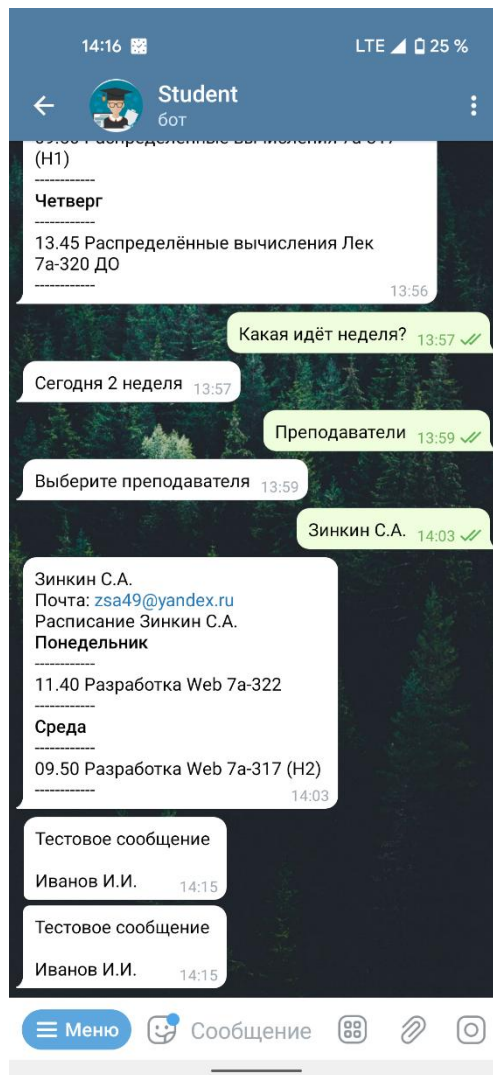


Рисунок 28 – Полученные сообщения

В результате пользователь с *id* равным 442666382 получил два экземпляра сообщения, так как он зарегистрирован в группе 18BV2 и 18BV3. Также после отправки сообщения было получено сообщение с результатом отправки, а именно перечисление групп, пользователи которых получили сообщение. На рисунке 29 представлено сообщение с информацией о доставке сообщения.

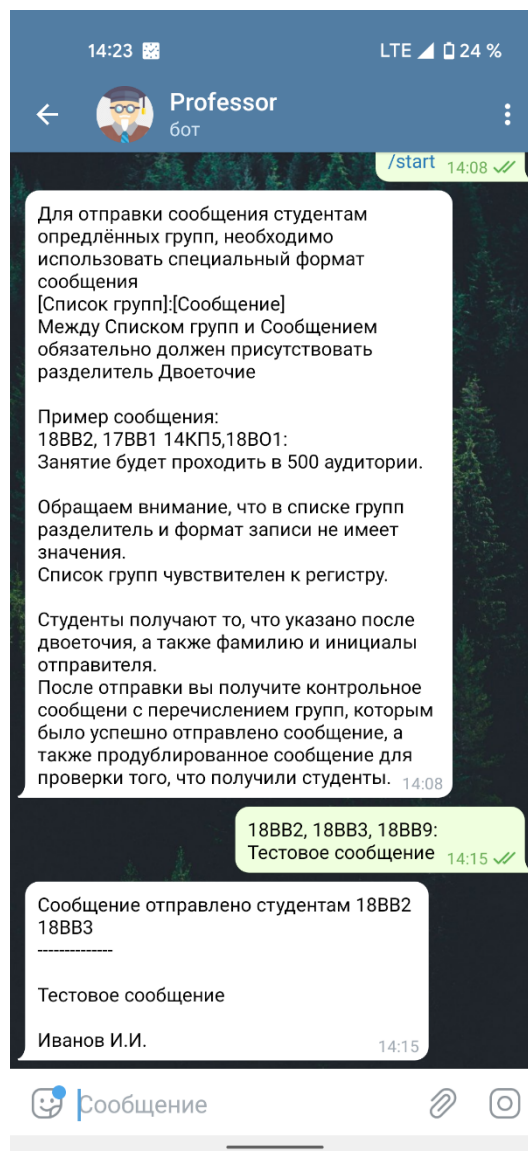


Рисунок 29 – Сообщение с информацией и доставке сообщения

Как видно из рисунка, сообщение получили пользователи группы 18BB2 и 18BB3, а пользователи группы 18BB9 не получили сообщения, так как ни одного пользователя с данной группы не было найдено.

Таким образом, были протестированы все функции приложения, а результаты тестирования доказали корректность работы приложения.

2 Безопасность жизнедеятельности

Приложения и другое программное обеспечение не представляют угрозу здоровью пользователям и разработчикам. Касаясь вопроса безопасности, в данном случае следует рассматривать устройства, на которых будет использоваться данное приложение, а именно ПЭВМ. Помещение, в котором предполагается взаимодействие с разрабатываемым продуктом, его размеры (площадь, объем) должны соответствовать количеству полностью оборудованных рабочих мест сотрудников. Кроме того, должны быть учтены соответствующие нормы по температуре, освещению, чистоте воздуха, изоляции от производственного шума.

Для нормальной работы с ПЭВМ, должны быть соблюдены рекомендации по обеспечению общих санитарно-технических норм к помещению и рабочим местам для минимизирования влияния потенциально-опасных факторов.

2.1 Потенциально-опасные факторы

При работе с ПЭВМ необходимо предотвращать влияние всех потенциально-опасных и вредных факторов, которые могут привести к заболеванию и любому отрицательному влиянию на здоровье человека. Вредные факторы при работе с ПЭВМ можно разделить на следующие группы:

- факторы тяжести трудового процесса;
- биологические факторы;
- физические факторы.

К факторам тяжести трудового процесса относится вся деятельность человека, связанная с нагрузкой на опорно-двигательный аппарат и другие функциональные системы организма. К такой деятельности относится любая физическая активность, а в частности передвижение и ношение тяжестей, что

характерно для физического труда. В частности, для работы с ПЭВМ характерно продолжительное нахождение в статичном положении, что негативно влияет на глаза, опорно-двигательный аппарат и другие функциональные системы организма. Пренебрежительное отношение к данному вредному фактору при работе с ПЭВМ может привести к застою крови, необратимым изменениям в опорно-двигательном аппарате и ухудшению зрения.

Из нормативов и правил Министерства здравоохранения Российской Федерации "Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. СанПиН 2.2.2/2.4.1340-03", утвержденные Главным государственным санитарным врачом Российской Федерации 30 мая 2003 года следует, что высота рабочей поверхности стола для взрослых пользователей должна регулироваться в пределах 680 - 800 мм, а при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм. Также модульными размерами рабочей поверхности стола для ПЭВМ, на основании которых должны рассчитываться конструктивные размеры, следует считать: ширину 800, 1000, 1200 и 1400 мм, глубину 800 и 1000 мм при нерегулируемой его высоте, равной 725 мм. Также из норматива следует, что рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной - не менее 500 мм, глубиной на уровне колен - не менее 450 мм и на уровне вытянутых ног - не менее 650 мм. Следование данным нормативам уменьшит нагрузку на опорно-двигательный аппарат и снизит влияние фактора тяжести трудового процесса.

К биологическим факторам относятся микроорганизмы и бактерии, пагубно влияющие на здоровье человека и способные привести к заболеванию. Такие микроорганизмы представляют угрозу, когда находятся на поверхностях, с которыми соприкасается человек или находится в непосредственной близости. В клавиатуре, компьютерной мыши и других

манипуляторов, используемых при работе с ПЭВМ, из-за частого контакта с кожей человека, образуется благоприятная среда для размножения микроорганизмов и бактерий, что увеличивает риск заражения патогенными микроорганизмами и вреда здоровью. Для того, чтобы снизить влияние данного фактора, следует соблюдать гигиену и тщательно дезинфицировать все устройства, с которыми приходится соприкасаться во время работы с ПЭВМ.

К физическим факторам относят микроклимат помещения, неионизирующие электромагнитные поля и излучения, электростатическое и постоянное магнитное, геомагнитное поля, источники ионизирующего излучения, производственный шум, ультразвук, инфразвук, вибрацию, аэрозоли преимущественно фиброгенного действия – пыли, световую среду, аэроионный состав воздуха.

Министерством здравоохранения Российской Федерации был разработан комплекс нормативов и правил, следуя которому сотрудники и работодатели обеспечат снижение негативного влияния всех вышеперечисленных факторов на человека.

Заключение

В ходе выполнения выпускной квалификационной работы был проведён анализ предметной области, который показал, что у уже существующих методов информирования студентов имеется ряд недостатков.

В результате было разработано приложение, позволяющее информировать студентов по организационным вопросам в процессе обучения. В данной системе устранена часть недостатков аналогичных систем и реализованы все предъявленные функциональные требования.

Выполненные отладка и тестирование доказали работоспособность и корректность работы приложения.

В процессе выполнения работы были изучены различные технологии, в частности *Telegram Bot API*, методы хранения информации с помощью *Google Sheet* и реализации *web*-приложения на основе проекта *Google Apps Scripts*, а также методы разработки программ на языке *Python*.

Для данной системы есть возможность дальнейшей доработки и модернизации функционала, в результате которой она станет более удобной и универсальной.

Список использованных источников

1. *Python 3* Обзор [Электронный ресурс]/ – Режим доступа: <https://coderlessons.com/tutorials/python-technologies/izuchite-python-3/python-3-obzor>, свободный (дата обращения 04.06.2022 г.).
2. Марк Лутц. Изучаем *Python*, 3-е издание – Пер. с англ. – СПб.: Символ-Плюс, 2009. – 848 с.
3. Цилюрик О.И. Производительность языков программирования. Редакция 10, 2018. – 25 с.
4. *JSON* – Краткое руководство [Электронный ресурс]/ – Режим доступа: <https://coderlessons.com/tutorials/java-tekhnologii/vyuchit-json/json-kratkoe-rukovodstvo>, свободный (дата обращения 07.06.2022 г.).
5. О системе контроля версий [Электронный ресурс]/ – Режим доступа: <https://git-scm.com/book/ru/v2/Введение-О-системе-контроля-версий>, свободный (дата обращения 07.06.2022 г.).

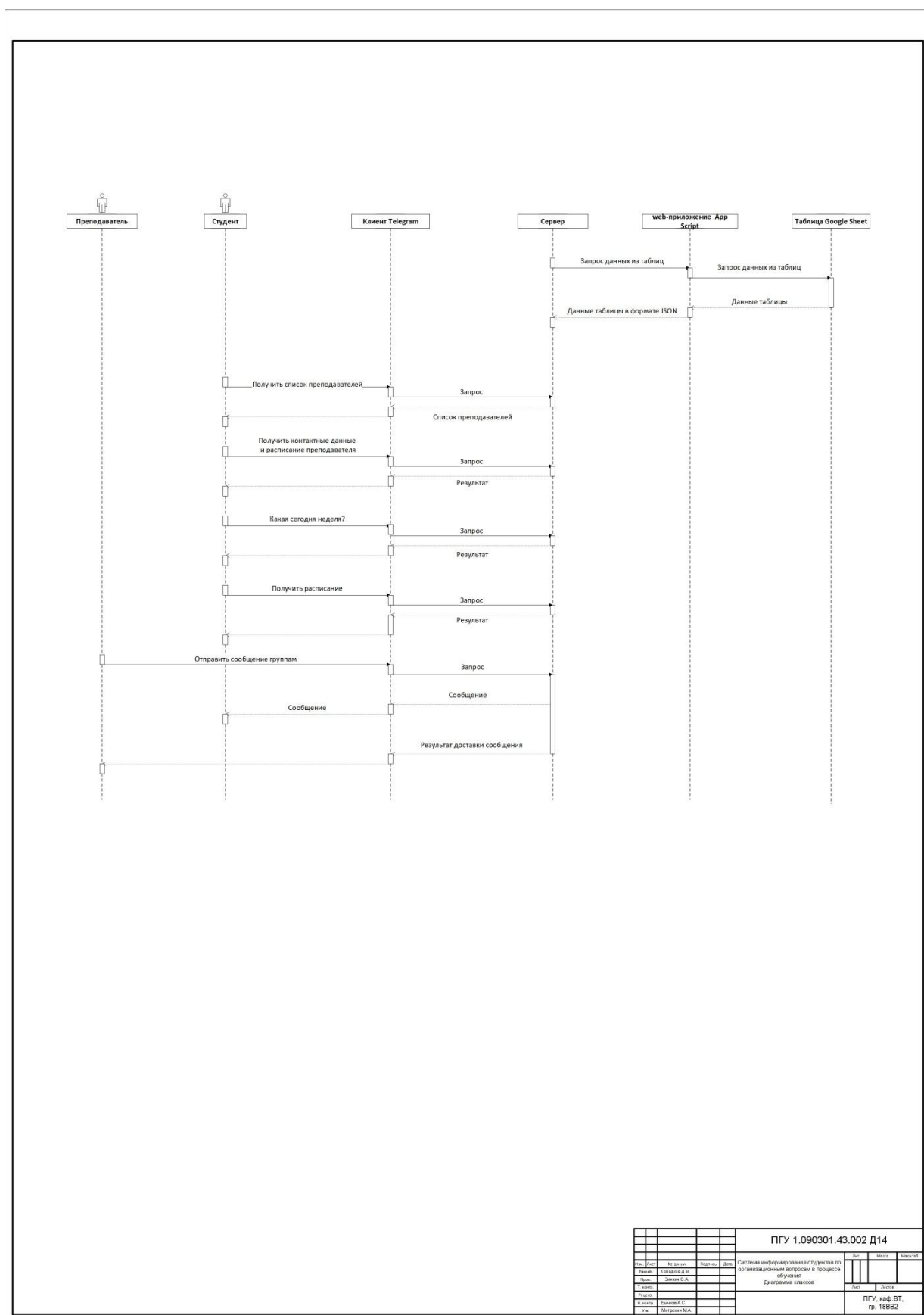
Приложение А

(Обязательное)

Система информирования студентов по организационным вопросам в
процессе обучения
UML-диаграммы

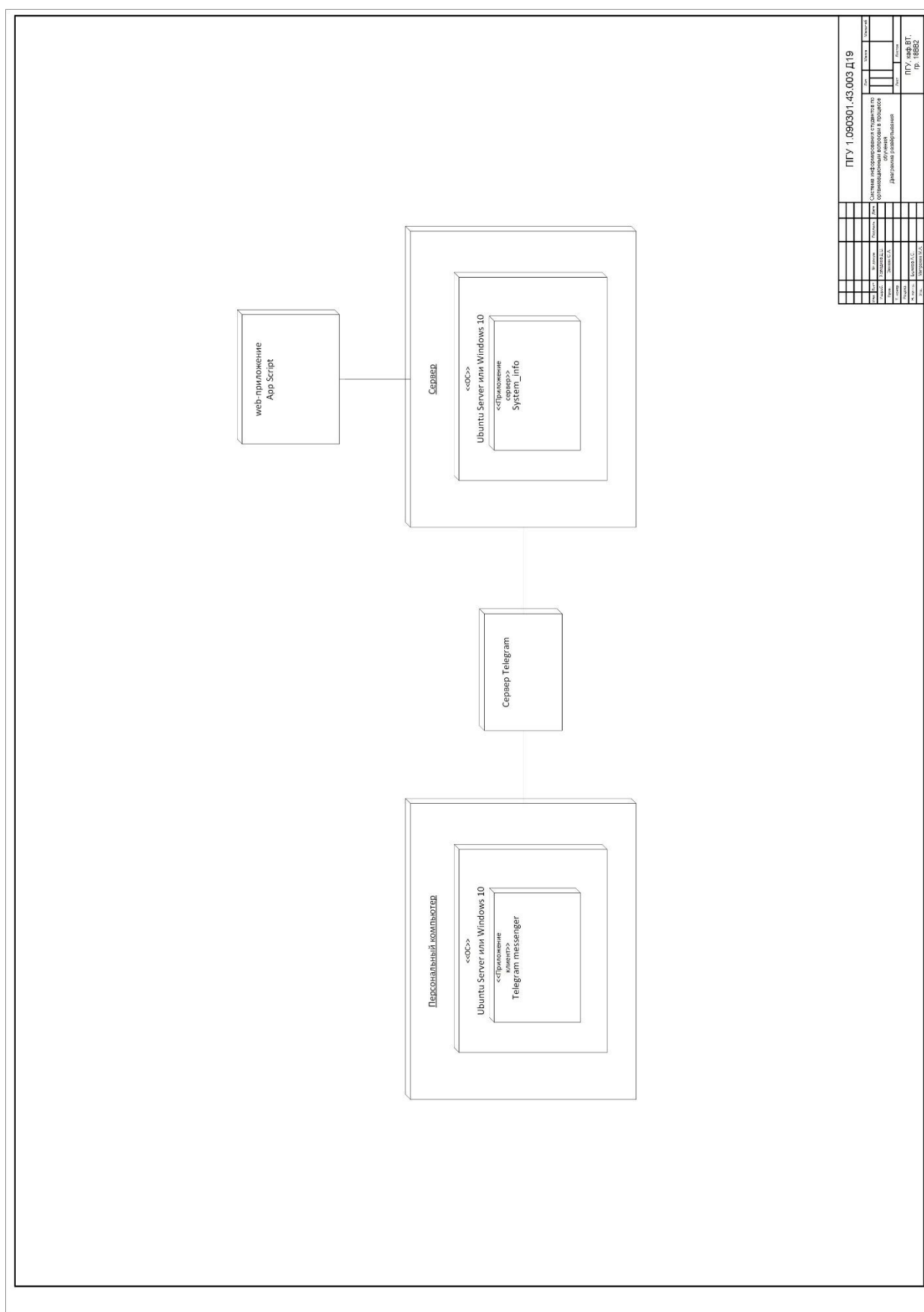
					ПГУ 1.090301.43.001 ПЗ	Лист
						54
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А1 – Диаграмма последовательностей

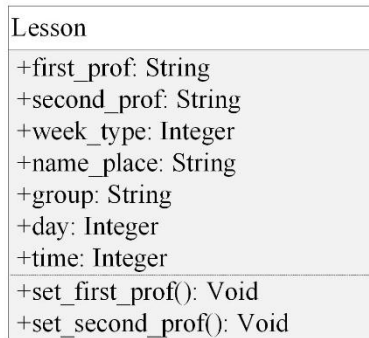
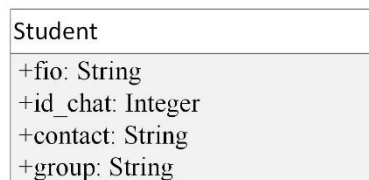
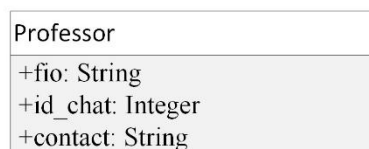


ПГУ 1.090301.43.002 Д14					ПГУ, каф. ВТ, пр. 18882		
Изм.	Лист	№ докум.	Подпись	Дата	Имя	Фамилия	Отчество
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов
					Иванов	Иванов	Иванов

Приложение А2 – Диаграмма развёртывания

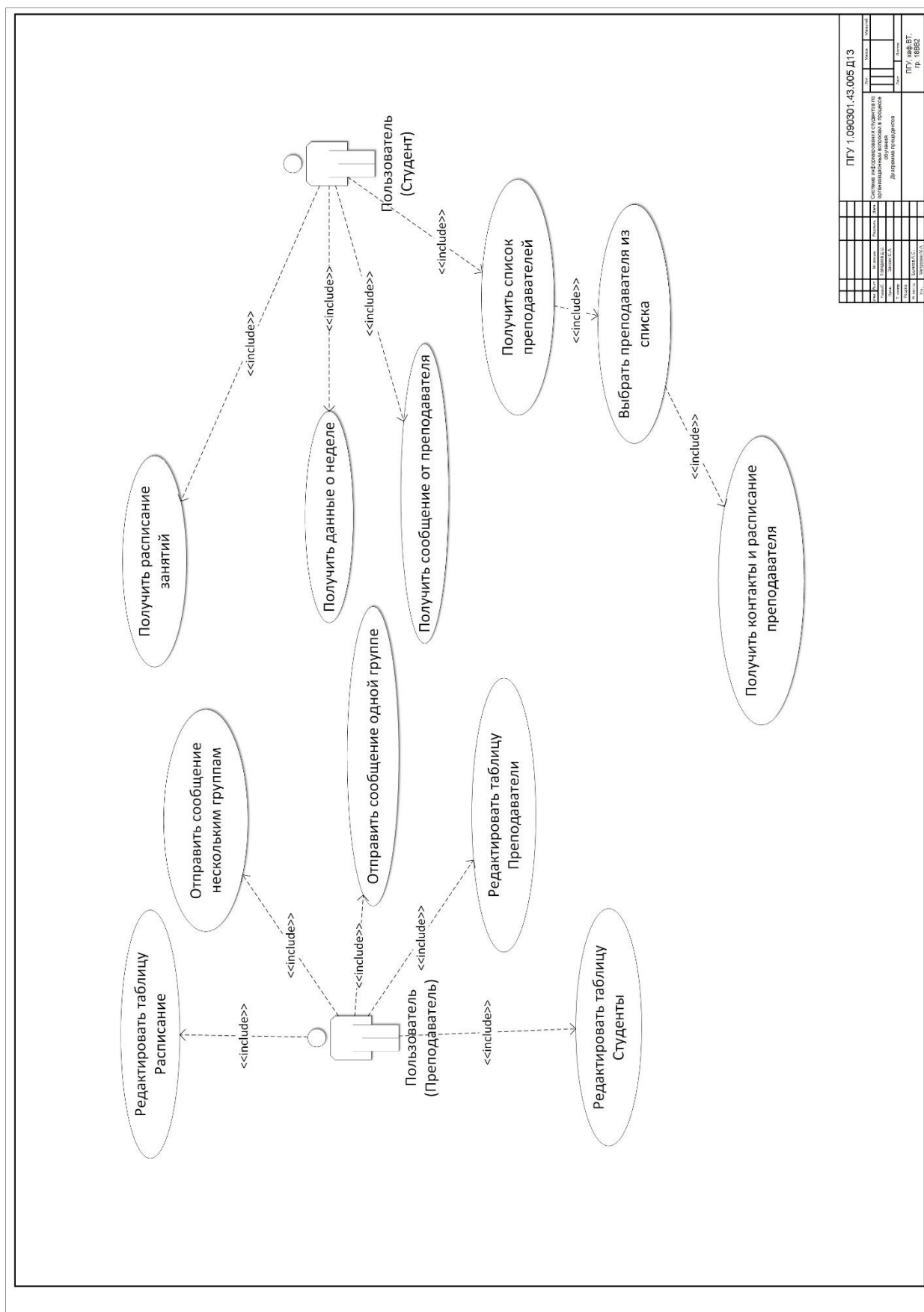


Приложение А3 – Диаграмма классов



						ПГУ 1.090301.43.004 Д11					
Имя, Фамилия	М. д. рождения	Подпись	Дата	Система информатизации и структуры по организации учебных занятий и процессам обучения Диаграмма классной					Вс.	Минут	Минут
Дмитров	Ульяновский Д. В.										
Григорьев	Зинченко С. А.										
С. номер											
Д. номер											
Имя, Фамилия	М. д. рождения	Подпись	Дата						Вс.	Минут	Минут
С. номер	Васильев Д. С.										
Д. номер											
							ПГУ, каф. ВТ, гр. 18882				

Приложение А4 – Диаграмма прецедентов



ПГУ 1.090301.43.005 Д13									
№ п/п	И.И.О.	Ф.И.О.	Д.И.	М.И.	П.И.	С.И.	С.И.	С.И.	С.И.
1									
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									
23									
24									
25									
26									
27									
28									
29									
30									
31									
32									
33									
34									
35									
36									
37									
38									
39									
40									
41									
42									
43									
44									
45									
46									
47									
48									
49									
50									
51									
52									
53									
54									
55									
56									
57									
58									
59									
60									
61									
62									
63									
64									
65									
66									
67									
68									
69									
70									
71									
72									
73									
74									
75									
76									
77									
78									
79									
80									
81									
82									
83									
84									
85									
86									
87									
88									
89									
90									
91									
92									
93									
94									
95									
96									
97									
98									
99									
100									

Приложение Б

(Обязательное)

Система информирования студентов по организационным вопросам в

процессе обучения

Листинг программы

					ПГУ 1.090301.43.001 ПЗ	Лист
						59
Изм.	Лист	№ докум.	Подпись	Дата		

Файл «main.py»

```
import time
import requests
import json
import datetime
import asyncio
from threading import Thread
import multiprocessing
from multiprocessing import Process
import aiogram
from aiogram import Bot, types
from aiogram.dispatcher import Dispatcher
from aiogram.utils import executor
from aiogram.types import ReplyKeyboardRemove, \
    ReplyKeyboardMarkup, KeyboardButton, \
    InlineKeyboardMarkup, InlineKeyboardButton
from ProfessorClass import Professor
from StudentClass import Student
from LessonClass import Lesson
from NowWeakFunc import nowWeakFunc
day_text=['Понедельник' , 'Вторник' , 'Среда' , 'Четверг' , 'Пятница' ,
'Sуббота' , 'Воскресенье']
time_text= [ '08.00 - 09.35','09.50 - 11.25','11.40 - 13.15','13.45 -
15.20','15.35 - 17.10','17.25 - 19.00','19.15 - 20.50']
student_token='5141082031:AAFMYoy7AwEjAuoa33CetHi19PFyGAnL1JI'
professor_token='5290242976:AAFPNKUJ81T4dDGytYNDjC55YZXSiPlc_Nw'
dd='https://api.telegram.org/bot5141082031:AAFMYoy7AwEjAuoa33CetHi19PFyGAnL1J
I/sendMessage?chat_id=442666382&text=pesos'
# Шаблоны для ответа бота
MyRequest="https://api.telegram.org/bot"
sendMes="/sendMessage?"
text="text="
user_id="442666382"
# ссылка для get запроса таблицы расписания
urlLessons = 'https://script.google.com/macros/s/AKfycbz-
fxcXXePkXhjJHK_ki7RyIngN36ZBCEHBRsISWIg8FSFwd6T3hvEAds_EjzNopI-
GsA/exec?param=raspisanije'
# ссылка для get запроса таблицы преподадов
```

```

urlProfessors
'https://script.google.com/macros/s/AKfycbzZ9kl_44yhQeswYgOo56bsKc2cNFTdHsHww
QST3LwmuzWkaHkVaFjgTCCJhWqwuXU8/exec'
#   ссылка для get запроса даты 1 недели в таблице расписания
urlWeak          =          'https://script.google.com/macros/s/AKfycbz-
fxcXXePkXhjJHK_ki7RylngN36ZBCEHBRsISWIg8FSFwd6T3hvEAds_EjzNopI-
GsA/exec?param=weak'
#   ссылка для get запроса таблицы студентов
urlStudents='https://script.google.com/macros/s/AKfycbxTsAzzxhWRbzJa9BMEOuoyZ
UjTFoYjlmhGetcagESetT4vmdvqESSRzu6WCKM0f0riqg/exec'
arr_students = []
arr_lessons = []
arr_professors = []
nowWeak=0
def Update_Data():
    arr_students.clear()
    arr_lessons.clear()
    arr_professors.clear()
    nowWeak=0
#   запрос даты 1 недели с таблицы расписания
response = requests.get(urlWeak)
a = response.json()
#   вычисление Какая неделя
nowWeak = nowWeakFunc(a["day"], a["month"], a["year"]);
#   запрос данных с таблицы преподав
response = requests.get(urlProfessors)
a = response.json()
for item in a["data"]:
    arr_professors.append(Professor(item[0], item[1], item[2]))
#   print(item[0], item[1], item[2], item[3], '\n')
#   запрос данных с таблицы студентов
response = requests.get(urlStudents)
a = response.json()
for item in a["data"]:
    arr_students.append(Student(item[0], item[1], item[2], item[3]))
#   запрос данных с таблицы расписания
response = requests.get(urlLessons)
a = response.json()
counter = 0
arr_json = []
arr_lessons_repeat=[]

```

```

# Обработка json и формирование массива пар
for item in a["result"]:
    arr_json.append(item)
for row in range(0, len(arr_json)):
    for col in range(1, len(arr_json[0]), 2):
        if (arr_json[row][col] != ""):
            if (arr_json[row][col] != "*"):
                week_type = 1
                if ((row!=(len(arr_json)-
1))and(arr_json[row+1][col]=="*")):
                    week_type=3
                professor_this_lesson = ""
                if (row % 2 > 0):
                    # 2 неделя
                    # print(arr_json[str-1][0])
                    professor_this_lesson = arr_json[row - 1][0]
                    week_type = 2
                else:
                    # 1 неделя
                    # print(arr_json[str][0])
                    professor_this_lesson = arr_json[row][0]
                day_this_lesson = col // 14
                group_this_lesson = arr_json[row][col + 1]
                time_this_lesson = ((col % 14) - 1) // 2
                arr_lessons_repeat.append(
                    Lesson(professor_this_lesson,"*", week_type,
arr_json[row][col], group_this_lesson, day_this_lesson,
                        time_this_lesson))

# Объединение пар у двух преподавателей
for i in range(0,len(arr_lessons_repeat)-1):
    for j in range(i+1,len(arr_lessons_repeat)):
        if((arr_lessons_repeat[i].group==arr_lessons_repeat[j].group)and(arr_lessons_
repeat[i].week_type==arr_lessons_repeat[j].week_type)and(arr_lessons_repeat[i
].day==arr_lessons_repeat[j].day)and(arr_lessons_repeat[i].time)==arr_lessons
_repeat[j].time):
            arr_lessons_repeat[i].second_prof=arr_lessons_repeat[j].first_prof
            arr_lessons_repeat[j].first_prof=""
for i in range(0,len(arr_lessons_repeat)):
    if(arr_lessons_repeat[i].first_prof!=""):
        arr_lessons.append(arr_lessons_repeat[i])
print("Данные Обновлено")

```

```

Update_Data()
# Отбор расписания по группе и сортировка по дню и времени
def generate_timetable_group(param, mode):
    timetable_group=[]
    if(mode==1):
        check=True
        for item in arr_students:
            print(f"param={param}          {type(param)}\nid          {item.id_chat}
{type(item.id_chat)}")
            if(item.id_chat==param):
                check=False
                param=item.group
                print(f"Найдена группа {param}")
                break
        if(check):
            print("сработал check после поиска группы")
            return "Не обнаружено"
        for item in arr_lessons:
            if(item.group==param):
                timetable_group.append(item)
        print(f"Было найдено пар {len(timetable_group)}")
        if (len(timetable_group)==0):
            print("сработал return от пустого массива пар")
            return "Не обнаружено"
    if(mode==2):
        for item in arr_lessons:
            if((item.first_prof==param)or(item.second_prof==param)):
                timetable_group.append(item)
        if (len(timetable_group)==0):
            return "Не обнаружено"
    if((mode!=1)and(mode!=2)):
        print("Сработал return от mode")
        return "Не обнаружено"
    for i in range(0,len(timetable_group)-1):
        for j in range(i+1,len(timetable_group)):
            if(timetable_group[i].time<=timetable_group[j].time):
                buf=timetable_group[i]
                timetable_group[i]=timetable_group[j]
                timetable_group[j]=buf

    for i in range(0,len(timetable_group)-1):

```

```

        for j in range(i+1,len(timetable_group)):
if (timetable_group[i].day>=timetable_group[j].day):
            buf=timetable_group[i]
            timetable_group[i]=timetable_group[j]
            timetable_group[j]=buf
#Отсортировать по типу недели
for i in range(0,len(timetable_group)-1):
    if ((timetable_group[i].time == timetable_group[i + 1].time) and (
        timetable_group[i].day == timetable_group[i + 1].day) and (
            timetable_group[i].week_type      >      timetable_group[i      +
1].week_type)):
        buf=timetable_group[i]
        timetable_group[i]=timetable_group[i+1]
        timetable_group[i+1]=buf
#Форматированный вывод
timetable_text=f"Расписание {param}\n"
separator="-----\n"
separator2="++++++\n"
name_week_add=-1
next=False
for i in range(0,len(timetable_group)):
    if next:
        next=False
        continue
    if (name_week_add!=timetable_group[i].day):
timetable_text=timetable_text+f"*{day_text[timetable_group[i].day]}*\n{separa
tor}"
        name_week_add=timetable_group[i].day
        if (timetable_group[i].week_type==3):
            timetable_text      =      timetable_text      +
f"{time_text[timetable_group[i].time][0:5]}
{timetable_group[i].name_place}\n{separator}"
            continue
        if (i!=(len(timetable_group)-1)):
if ((timetable_group[i].day==timetable_group[i+1].day) and (timetable_group[i].t
ime==timetable_group[i+1].time) and (timetable_group[i].week_type==1) and (timeta
ble_group[i+1].week_type==2)):
            next=True
            timetable_text      =      timetable_text      +
f"{time_text[timetable_group[i].time][0:5]}      {timetable_group[i].name_place}
(H{timetable_group[i].week_type})\n"

```

```

timetable_text=timetable_text+f"
{timetable_group[i+1].name_place}
(H{timetable_group[i+1].week_type})\n{separator}"
continue
if(timetable_group[i].week_type==1):
timetable_text=timetable_text+
f"{time_text[timetable_group[i].time][0:5]} {timetable_group[i].name_place}
(H{timetable_group[i].week_type})\n{separator}"
#timetable_text=timetable_text+f"
{timetable_group[i+1].name_place}
(H{timetable_group[i+1].week_type})\n{separator}"
continue
timetable_text=timetable_text+
f"{time_text[timetable_group[i].time][0:5]} {timetable_group[i].name_place}
(H{timetable_group[i].week_type})\n{separator}"
return timetable_text
#print(generate_timetable_group("18BB2"))
#####
print("готово")
student_bot=Bot(token=student_token)
dispatcher_student_bot=Dispatcher(student_bot)
professor_bot=Bot(token=professor_token)
dispatcher_professor_bot=Dispatcher(professor_bot)
def address_delivery(string_groups):
arr_adress=[]
find_groups=""
for item in arr_students:
if(item.group in string_groups):
print(f"Найден студент {item.fio} {item.id_chat}")
arr_adress.append(item.id_chat)
if (not(item.group in find_groups)):
find_groups=find_groups+item.group+' '
return [arr_adress, find_groups]
# Запросы для STUDENT_BOT
@dispatcher_student_bot.message_handler(commands=['start'])
async def process_start_command(msg: types.Message):
print(f"Пользователь {msg.chat.id} выполнил /start")
keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
keyboard.add(KeyboardButton("Расписание"))
keyboard.add(KeyboardButton("Какая неделя?"))
keyboard.add(KeyboardButton("Преподаватели"))

```

```

        await msg.answer("Бот готов к работе",reply_markup=keyboard)
@dispatcher_student_bot.message_handler(commands=['menu'])
async def process_start_command1(msg: types.Message):
    print(f"Пользователь {msg.chat.id} выполнил команду /menu")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Расписание"))
    keyboard.add(KeyboardButton("Какая идёт неделя?"))
    keyboard.add(KeyboardButton("Преподаватели"))
    await msg.answer("МЕНЮ",reply_markup=keyboard)
@dispatcher_student_bot.message_handler(lambda message: message.text ==
"Меню" or message.text == "Вернуться в меню")
async def process_start_command2(msg: types.Message):
    print(f"Пользователь {msg.chat.id} ввёл Меню или Вернуться в меню")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Расписание"))
    keyboard.add(KeyboardButton("Какая идёт неделя?"))
    keyboard.add(KeyboardButton("Преподаватели"))
    await msg.answer("МЕНЮ",reply_markup=keyboard)
@dispatcher_student_bot.message_handler(lambda message: message.text ==
"Преподаватели")
async def process_start_command3(msg: types.Message):
    print(f"Пользователь {msg.chat.id} ввёл Преподаватели")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Вернуться в меню"))
    for item in arr_professors:
        keyboard.add(KeyboardButton(item.fio))
    await msg.answer("Выберите преподавателя",reply_markup=keyboard)
@dispatcher_student_bot.message_handler(lambda message: message.text ==
"Расписание")
async def without_puree1(msg: types.Message):
    print(f"Пользователь {msg.chat.id} ввёл Расписание")
    keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
    keyboard.add(KeyboardButton("Расписание"))
    keyboard.add(KeyboardButton("Какая идёт неделя?"))
    keyboard.add(KeyboardButton("Преподаватели"))
    await
msg.answer(generate_timetable_group(msg.chat.id,1),reply_markup=keyboard,pars
e_mode="Markdown")
@dispatcher_student_bot.message_handler(lambda message: message.text ==
"Какая идёт неделя?")
async def without_puree2(msg: types.Message):

```

```

print(f"Пользователь {msg.chat.id} ввёл Какая идёт неделя")
keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
keyboard.add(KeyboardButton("Расписание"))
keyboard.add(KeyboardButton("Какая идёт неделя?"))
keyboard.add(KeyboardButton("Преподаватели"))
await msg.answer(f"Сегодня {nowWeak} неделя", reply_markup=keyboard)
@dispatcher_student_bot.message_handler()
async def echo_message(msg: types.Message):
    for item in arr_professors:
        if (msg.text==item.fio):
            keyboard = ReplyKeyboardMarkup(resize_keyboard=True)
            keyboard.add(KeyboardButton("Расписание"))
            keyboard.add(KeyboardButton("Какая идёт неделя?"))
            keyboard.add(KeyboardButton("Преподаватели"))
            await
                                msg.answer(f"{item.fio}\nПочта:
{item.contact}\n{generate_timetable_group(item.fio,2)}",
reply_markup=keyboard, parse_mode="Markdown")
@dispatcher_student_bot.callback_query_handler(text="Расписание")
async def send_random_value(call: types.CallbackQuery):
    await call.message.answer("Заглушка% вывод расписания")
# Запросы для PROFESSOR_BOT
@dispatcher_professor_bot.message_handler(commands=['start'])
async def process_start_command(msg: types.Message):
    await msg.answer(f"Для отправки сообщения студентам определённых групп,
необходимо использовать специальный"
                    f" формат сообщения\n[Список групп]:[Сообщение]\nМежду
Списком групп и Сообщением обязательно"
                    f" должен присутствовать разделитель Двоеточие\n\nПример
сообщения:\n18BB2, 17BB1 14КП5,18BO1:\n"
                    f"Занятие будет проходить в 500 аудитории.\n\n"
                    f"Обращаем внимание, что в списке групп разделитель и "
                    f"формат записи не имеет значения.\nСписок групп
чувствителен к регистру.\n\nСтуденты получают то, "
                    f"что указано после двоеточия, а также фамилию и
инициалы отправителя.\n"
                    f"После отправки вы получите"
                    f" контрольное сообщени с перечислением групп, которым
было успешно отправлено сообщение,"
                    f" а также продублированное сообщение для проверки того,
что получили студенты.")
@dispatcher_professor_bot.message_handler()

```

```

async def echo_message2(msg: types.Message):
    if(msg.text.find(":")!=-1):
        caption=""
        for item in arr_professors:
            if(msg.chat.id==item.id_chat):
                caption=item.fio
                break
        arr = address_delivery(msg.text[0:msg.text.find(':')])
        if(len(arr[0])>0):
            for item in arr[0]:
                await student_bot.send_message(chat_id=item,
text=f"{msg.text[(msg.text.find(':')+1):]}\n\n{caption}")
                await msg.answer(f"Сообщение отправлено студентам {arr[1]}\n-----
-----\n{msg.text[(msg.text.find(':')+1):]}\n\n{caption}")
            else:
                await msg.answer(f"Сообщение не отправлено студентам, группы не
найжены")
        else:
            await msg.answer(f"Сообщение не отправлено студентам, отсутствует
двоеточие")
def thr1():
    executor.start_polling(dispatcher_student_bot)
def ThreadUpdateData():
    while(True):
        time.sleep(60)
        print("Обновление данных")
        Update_Data()
if __name__ == '__main__':
    #ОСНОВНАЯ ПРОГРАММА
    thread1 = Thread(target=ThreadUpdateData)
    thread1.start()
    proc = Process(target=thr1)
    proc.start()
    executor.start_polling(dispatcher_professor_bot)

```

Файл «NowWeakFunc.py»

```

import datetime
def nowWeakFunc(day,month,year):
    date_first_weak=datetime.date(year,month+1,day+1)
    day_weak_dat_first_weak=date_first_weak.weekday()
    today=datetime.date.today()

```

```

day_weak_today=today.weekday()
dif=int(str(today-date_first_weak).split()[0])
dif=dif+day_weak_dat_first_weak-day_weak_today
return (dif//7)%2+1

```

Файл «LessonClass.py»

```

class Lesson:
    def __init__(self, first_prof, second_prof, week_type,
name_place, group, day, time):
        self.first_prof=first_prof
        self.second_prof=second_prof
        self.week_type=week_type
        self.name_place=name_place
        self.group=group
        self.day=day
        self.time=time
    def set_first_prof(self, first_prof):
        self.first_prof=first_prof
    def set_second_prof(self, second_prof):
        self.second_prof=second_prof

```

Файл «ProfessorClass.py»

```

class Professor:
    def __init__(self, fio, contact, id_chat):
        self.fio = fio
        self.contact = contact
        self.id_chat=id_chat

```

Файл «StudentClass.py»

```

class Student:
    def __init__(self, fio, group, contact, id_chat):
        self.fio = fio
        self.group = group
        self.contact = contact
        self.id_chat=id_chat

```

Файл «Raspisanie.gs»

```

var ss = SpreadsheetApp.getActiveSpreadsheet(), // spreadsheet

```

```

        s = ss.getActiveSheet(); // sheet
function getData(){
    var result = [],
        range = 'A:CG', // диапазон ячеек, который хотим выгрузить
        values = s.getRange(range).getValues(),
        last_row = parseInt(s.getLastRow());
    for (var i = 3; i < last_row; i++) {
        result.push(values[i]);
    }
    return result;
}
function getNedelya(){
    var myDate=s.getRange(1,1).getValue();
    //value.getMonth(), value.getHours(), value.getUTCHours()]);
    var
date={day:myDate.getDate(),month:myDate.getMonth(),year:myDate.getFullYear()}
    // console.log(date);
    return date;
}
function doGet(param){
    if (param.parameter.param=="weak"){
        return
        ContentService.createTextOutput(JSON.stringify(getNedelya())).setMimeType(ContentService.MimeType.JSON);
    }else if(param.parameter.param=="raspisanie"){
        var data = getData();
        if(!data) {
            data = '';
        }
        return ContentService.createTextOutput(JSON.stringify({'result':data})).setMimeType(ContentService.MimeType.JSON);
    }
}

```

Файл «Professors.gs»

```

var ss = SpreadsheetApp.getActiveSpreadsheet(), // spreadsheet
    s = ss.getActiveSheet(); // sheet
function getData(){
    var result = [],
        range = 'A:C', // диапазон ячеек, который хотим выгрузить
        values = s.getRange(range).getValues(),
        last_row = parseInt(s.getLastRow());
    for (var i = 1; i < last_row; i++) {
        result.push(values[i]);
    }
    return result;
}
function doGet(){
    var data = getData();
    if(!data) {
        data = '';
    }
    return ContentService.createTextOutput(
        JSON.stringify({data})) .setMimeType(ContentService.MimeType.JSON);
}

```

Файл «Students.gs»

```

var ss = SpreadsheetApp.getActiveSpreadsheet(), // spreadsheet
    s = ss.getActiveSheet(); // sheet
function getData(){
    var result = [],
        range = 'A:D', // диапазон ячеек, который хотим выгрузить

```

```

        values = s.getRange(range).getValues(),
        last_row = parseInt(s.getLastRow());
        for (var i = 1; i < last_row; i++) {
            result.push(values[i]);
        }
        return result;
    }
}
function doGet() {
    var data = getData();
    if(!data) {
        data = '';
    }
    return ContentService.createTextOutput(
        JSON.stringify({data})) .setMimeType(ContentService.MimeType.JSON);
}

```

					ПГУ 1.090301.43.001 ПЗ	Лист
						71
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Система информирования студентов по организационным вопросам в
процессе обучения

Презентация

					ПГУ 1.090301.43.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		72

Выпускная
Квалификационная работа
на тему: «Система информирования
студентов по организационным вопросам
в процессе обучения»

Автор работы: студент группы 17ВВ2 Холодков Д.В.

Руководитель: д.т.н., профессор Зинкин С.А.

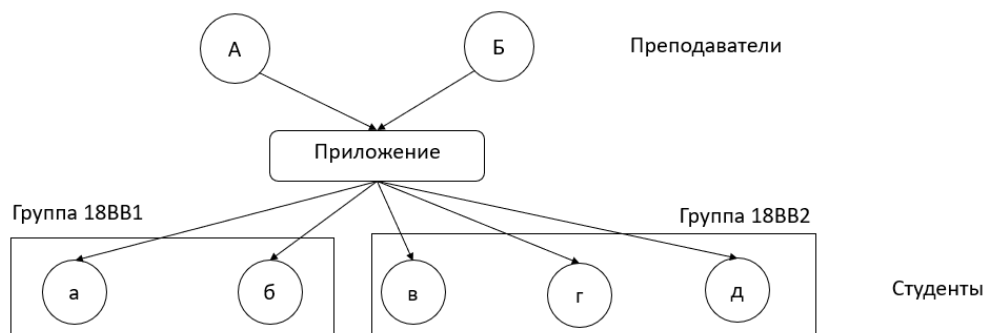
Пенза, 2022

Введение

Информирование студентов – это важная задача для университета. Для решения данной задачи используются различные инструменты позволяющие проинформировать студентов через интернет, однако у них есть ряд недостатков. Основной недостаток, это отсутствие автоматизации при передаче сообщений конкретной группе. Для каждого преподавателя данный недостаток выражается в необходимости хранения контактных данных, для связи с группой.

Цель работы

Цель работы – создание клиент-серверного приложения, реализующего передачу информации студентам с применением автоматизации, а также создание альтернативного инструмента для решения данной задачи.



Объекты работы

Объекты работы – сервер и проект Google App Scripts выполняющие следующие функции:

- Обмен данными с пользователем через клиент Telegram с помощью Telegram Bot API.
- Получение и обработка данных таблиц Google Sheets
- Отправка сообщений пользователям групп используя выборку, без необходимости хранения контактных данных.
- Реализация получения данных для студента касающихся расписания и информации о преподавателях.

Обзор аналогов

- Электронная почта



Яндекс Почта



- Web-сайт



pnzgu.ru

- Социальные сети



Отличия текущей реализации

- | | |
|---|---|
| <ul style="list-style-type: none"> • Удобство использования • Автоматизация • Простота поддержки | <ul style="list-style-type: none"> • Узко направленность |
|---|---|

Используемые технологии



AppScripts



Python



aiogram



GitHub Desktop

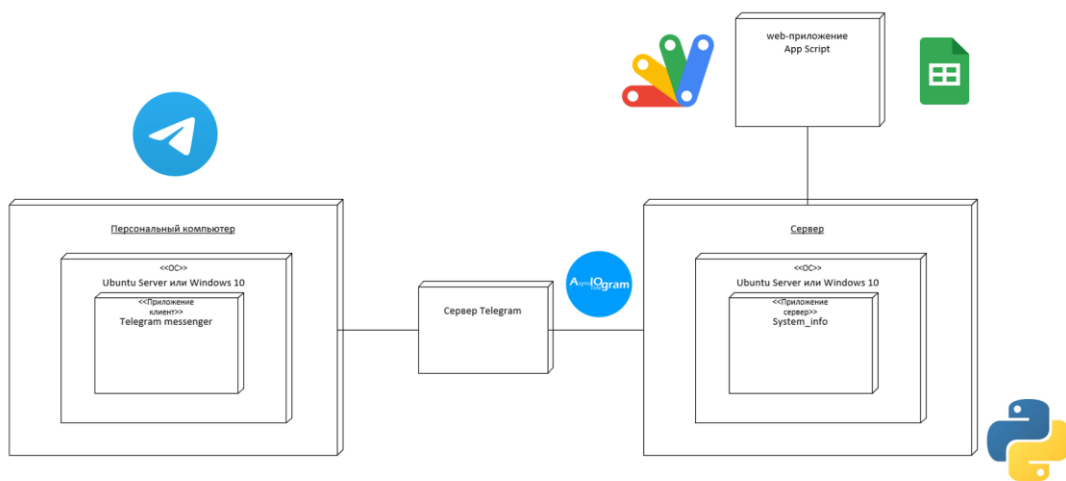


Telegram Bot API



Google Sheets

Архитектура решения



Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.43.001 ПЗ

Лист

76

Хранение данных

Студенты

Файл Правка Вид Вставка Формат Данные Инструменты Расц.

100% 123 По умолча...

	A	B	C	D
1	ФИО	Группа	Почта	id с чатом бота
2	Холодков Д.В.	188B2	danila.holodkov@mail.ru	442666382
3	Иванов И.И.	188B3	test@mail.ru	9 442666382

Преподаватели

Файл Правка Вид Вставка Формат Данные Инструменты

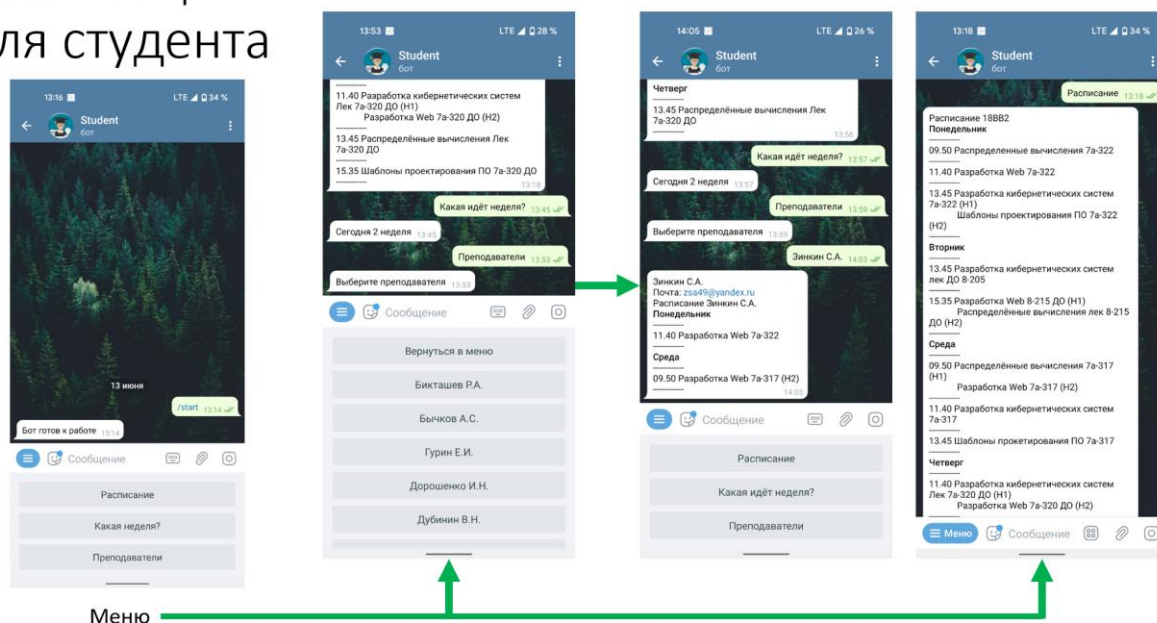
100% 123 По умолча...

	A	B	C
1	ФИО	Электронная почта	id чата телеграмм
2	Бикташев Р.А.	Почта@mail.ru	442666382
3	Бычков А.С.	Почта@mail.ru	442666382
4	Гурин Е.И.	Почта@mail.ru	442666382
5	Дорошенко И.Н.	Почта@mail.ru	442666382
6	Дубинин В.Н.	Почта@mail.ru	442666382
7	Дубравин А.В.	Почта@mail.ru	442666382
8	Егоров В.Ю.	Почта@mail.ru	442666382
9	Зинкин С.А.	Почта@mail.ru	442666382

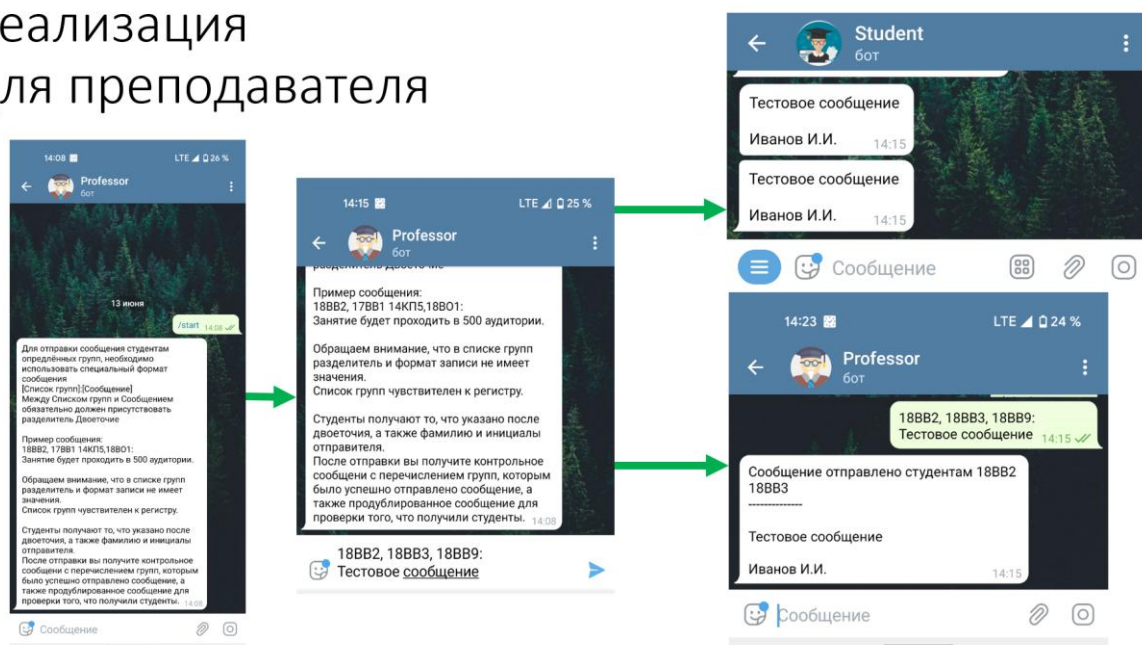
Реализованные возможности приложения



Реализация для студента



Реализация для преподавателя



Изм.	Лист	№ докум.	Подпись	Дата

Заключение

В ходе выполнения выпускной квалификационной работы был проведён анализ предметной области, который показал, что у уже существующих методов информирования присутствует ряд недостатков. В результате была разработана система информирования пользователей.

Выполненные отладка и тестирование доказали работоспособность и корректность приложения.