Введение

На учебную практику была поставлена задача разработать специального Telegram-бота на тему: «Информационный Telegram-бот СШ №13».

Основной задачей данного проекта является создание Telegram-бота, который будет представлять из себя помощника с определенным рядом функций, таких как: предоставление пользователю краткой информации о школе, ссылками на ее веб-ресурсы и порталы, расписанием работы заведения, краткими новостными сводками, расписания.

Так же в зависимости от выбора подпункта пользователя такого как, родителя, учащегося, учителя будет разниться выдаваемая информация, подходящая более каждому из пользователей, к примеру у ученика, будет предложена информация для выпускника.

Далее приведем краткое описание разделов пояснительной записки.

Первый раздел «Анализ задачи» описывает такие подпункты как, постановку задачи, диаграмму вариантов использования, выбор стратегии разработки и модели жизненного цикла, инструменты разработки, Разработка плана работы над проектом.

Во втором разделе «Проектирование задачи», будет описана разработка непосредственно структуры бота, системы меню, навигации. Разработка UML-диаграмм, разработка пользовательского интерфейса, а так, же тест-кейсы.

В разделе «Реализация» будет описано руководство программиста, а так, же диаграмма компонентов.

Раздел «Тестирование» будет содержать отчет о результатах тестирования программы. Отчет будет оформлен в виде таблицы.

В разделе «Руководство пользователя» будет описано руководство к использованию программного продукта.

В «Заключение», располагается краткая формулировка задачи, как выполнена поставленная задача, использованные методы и средства, степень соответствия проектных решений заданию, причины несоответствия (если имеются). Найденные нетрадиционные способы решения задачи, возможность модификации.

В «Списке использованных источников», содержит перечень, использованных при выполнении курсового проекта источников.

В приложениях к пояснительной записке будут расположены Диаграммы к проекту.

1. Анализ задачи

1.1 Постановка задачи

Организационно-экономическая сущность задачи

Наименование задачи: Информационный Telegram-бот СШ №13.

Цель разработки: Создание информационного Telegram-бота для взаимодействия с различными видами пользователей, таких как, учитель, ученик, родитель. С целью помощи в нахождении информации об учреждении, интересующей пользователя, предоставление информации, подходящей конкретному пользователю. Так же не мало важным является доступность, чтобы к боту можно было очень быстро обратиться с нескольких устройств.

Назначение: Данный продукт разрабатывается для учащихся, их родителей и учителей для упрощения взаимодействия с информационным полем школы без переполнения второстепенной информацией с акцентом внимания на конкретного пользователя, но с возможностью менять пользователя с целью получения большей информации.

Периодичность использования: по мере необходимости.

Источники и способы получения данных: Официальный сайт «Средней школы №13 имени В.Т.Цабо г. Гродно»

Обзор существующих аналогичных ПП: Рассмотрим подобного телеграмм бота, «@iTMOHelpBot». Это бот, создан для студентов университета ИТМО. В функционал бота входят такие функции как: полные сведения о расписания, предоставления адресов кампусов на карте, расписания экзаменов, поиска расписаний по группе, поиск экзаменов по группе. Из плюсов у бота, это возможность производить поиск нужной информации, так же это быстрота работы, из минусов это работа только с помощью команд.

Функциональные требования

Описание перечня функций и задач, которые должен выполнять будущий ПП:

Пользователь

1 Просмотр информации об учреждении

2 Просмотр контактной информации

3 Просмотр педагогического коллектив

4 Получение ссылок на веб-ресурсы

5 Просмотр расписания звонков

6 Просмотр расписания факультативных занятий

7 Просмотр Календаря четвертей и занятий

8 Просмотр новостных сведений

9 Получение ссылки на новость
Ученик
1 Просмотр профориентации для школьников
2 Просмотр информации для выпускника
3 Просмотр информации СППС
4 Просмотр календаря выпуска
Учитель
1 Просмотр СППС для педагогов
2 Просмотр перечня ШАГ
3 Поиск по датам в ШАГ
4 Поиск по темам ШАГ
5 Просмотр нормативных документов
6 Просмотр методических рекомендаций
7 Просмотр инструктивно-методических писем
Родитель
1 Просмотр родительского комитета
2 Просмотр плана работы родительского комитета
3 Просмотр попечительского совета
4 Просмотр состава попечительского совета
5 Просмотр отчета работы попечительского совета
6 Просмотр СППС для родителей
Родитель, ученик и учитель наследуют информацию пользователя.
Входная информация:
1 Данные сайта школы
Выходная информация:
1 Информация об учреждении
2 Контактная информация
3 Информация о педагогическом коллективе
4 Расписания звонков
5 Расписания факультативных занятий
6 Ссылки на веб-ресурсы школы
7 Новостные сведенья
8 Календарь четвертей и занятий
9 СППС для педагогов
10 Профориентации для школьников
11 Информации для выпускника
12 Информации СППС для школьников
13 Календарь выпуска

14 Перечень ШАГ
15 Дата ШАГ
16 Тема ШАГ
17 Нормативные документы
15 Методические рекомендации
16 Инструктивно-методические письма
17 СППС для родителей
18 Родительский комитет
19 План работы родительского комитета
20 Попечительский совет
21 Состав попечительского совета
22 Отчет работы попечительского совета
Условно-постоянная:
1 Ссылка на веб-ресурс школы
2 Таблица Календарь учебных мероприятий
3 Список педагогического состава
4 Список контактной информации
Эксплуатационные требования

Требования к применению: Telegram-бот должен выдавать интересующую информацию пользователю, так же работать с запросами пользователя и уметь на них ответить.

Требования к реализации: Для реализации бота будет использоваться язык программирования Python с соответствующими библиотеками, как «python-telegram-bot» для написания команд и структуры бота, библиотека BeautifulSoup4 для получения данных с сайта. Среда разработки PyCharm.

Требования к надежности: Система может быть недоступна не более чем 24 часа в семестр. После сбоя в работе системный администратор перезапускает бота, для продолжения его корректной работы.

Требования к интерфейсу: Интерфейс должен корректно отображаться на устройствах с разными экранами, быть логично понятным, так же в нем должны присутствовать функциональные кнопки для облегчения работы.

1.2 Диаграмма вариантов использования

В разработанной диаграмме вариантов использования описано четыре актера с вариантами использования.

Первый актером является «Пользователь». Пользователь базовый актёр, который может просматривать общую информацию об учреждении, расписания и

новости. Базовые функции у этого актера расширены дополнительными деталями, такими как Контактной информацией, педагогическим коллективом, ссылками на веб-ресурсы, расписанием звонок, расписанием факультативных занятий, календарем четвертей и каникул.

Вторым актером является «Ученик». Ученик — наследует возможности "Пользователя" и имеет дополнительные функции: доступ к СППС для учащихся, профориентации, информации для выпускников и календарю выпуска.

Третьим актером является «Учитель». Учитель — наследует возможности "Пользователя" и имеет доступ к СППС для педагогов, нормативным документам, ШАГ (поиск по датам и темам), методическим рекомендациям и инструктивно-методическим письмам.

Четвертым актером является «Родитель». Родитель — наследует возможности "Пользователя" и имеет функции, связанные с родительским комитетом, попечительским советом и СППС для родителей.

Разработанная диаграмма вариантов использования представлена в приложении А

1.3 Выбор стратегии разработки и модели жизненного цикла

Таблица 1 – Выбор модели жизненного цикла на основе характеристик требований

№ критерия	Критерии категории требований	Каскадная	V-образная	RAD	Инкрементная	Быстрого прототипирования	Эволюционная
1.	Являются ли требования к проекту легко определимыми и реализуемыми?	Да	Да	Да	-	-	-
2.	Могут ли требования быть сформулированы в начале ЖЦ?	Да	Да	Да	Да	-	-
3.	Часто ли будут изменяться требования на протяжении ЖЦ?	Нет	Нет	Нет	Нет	-	-
4.	Нужно ли демонстрировать требования с целью их определения?	-	-	Да	-	Да	Да
5.	Требуется ли проверка концепции программного средства или системы?	-	-	Да	-	Да	Да

Продолжение таблицы 1

6.	Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	-	-	-	Да	Да	Да
7.	Нужно ли реализовать основные требования на ранних этапах разработки?	-	-	Да	Да	Да	Да

Вычисления: 2 за каскадную, 2 за V- образную, 6 за RAD, 4 за инкрементную, 4 за быстрого прототипирования и 4 за эволюционную.

Итог: На основе результатов заполнения табл. 3 подходящей является RAD модель.

Таблица 2 – Выбор модели жизненного цикла на основе характеристик команды разработчиков

№ критерия	Критерии категории команды разработчиков Проекта	Каскадная	V-образная	RAD	Инкремент-ная	Быстрого прототипирования	Эволюцион-ная
1.	Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет	Нет	Нет	Нет	-	-
2.	Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	-	-	Нет	Нет	Нет	-
3.	Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет	Нет	Нет	-	-	-
4.	Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	-	-	Нет	-	Нет	Нет
5.	Важна ли легкость распределения человеческих ресурсов проекта?	Да	Да	Да	Да	-	-
6.	Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Да	Да	-	Да	Да	Да

Вычисления: 4 за каскадную, 4 за V-образную, 5 за RAD, 4 за инкрементную, 3 за быстрого прототипирования и 2 за эволюционную.

Итог: На основе результатов заполнения табл. 4 подходящими является RAD модель.

Таблица 3 – Выбор модели жизненного цикла на основе характеристик коллектива пользователей

№ критерия	Критерии категории коллектива пользователей	Каскадная	V-образная	RAD	Инкремент-ная	Быстрого прототипирования	Эволюцион-ная
1.	Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Да	Да	-	Да	-	Да
2.	Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	-	-	-	Да	Да	Да
3.	Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет	Нет	-	Нет	-	Нет
4.	Будет ли заказчик отслеживать ход выполнения проекта?	Нет	Нет	Нет	Нет	-	-

Вычисления: 3 за каскадную, 3 за V-образную, 1 за RAD, 4 за инкрементную, 1 за быстрого прототипирования и 3 за эволюционную.

Итог: На основе результатов заполнения табл. 5 подходящей является инкрементная модель.

Таблица 4 – Выбор модели жизненного цикла на основе характеристик типа проектов и рисков

№ критерия	Критерии категории типов проекта и рисков	Каскадная	V-образная	RAD	Инкремент-ная	Быстрого прототипирования	Эволюцион-ная
1.	Разрабатывается ли в проекте продукт нового для организации направления?	Нет	Нет	Нет	-	-	-
2.	Будет ли проект являться расширением существующей системы?	-	-	-	-	Нет	Нет
3.	Будет ли проект крупно- или среднемасштабным?	Нет	Нет	Нет	-	-	-
4.	Ожидается ли длительная эксплуатация продукта?	Да	Да	-	Да	-	Да
5.	Необходим ли высокий уровень надежности продукта проекта?	-	Да	-	Да	-	Да

Продолжение таблицы 4

6.	Предполагается ли эволюция продукта проекта в течение ЖЦ?	-	-	-	Да	Да	Да
7.	Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Нет	Нет	Нет	-	-	-
8.	Является ли график сжатым?	Нет	Нет	-	-	-	-
9.	Предполагается ли повторное использование компонентов?	Нет	Нет	-	-	-	-
10.	Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	-	-	-	-	Да	Да

Вычисления: 6 за каскадную, 7 за V-образную, 3 за RAD, 3 за инкрементную, 3 за быстрого прототипирования и 5 за эволюционную.

Итог: На основе результатов заполнения таблицы 6 подходящей является V-образная модель.

Общий итог: в итоге заполнения табл. 3 – 6 наиболее подходящей является RAD модель.

1.4 Инструменты разработки

Для разработки данного проекта будет выбрана среда разработки PyCharm, которая является наиболее актуальной средой для создания приложений данного типа.

Разработка будет производится на таком языке программирования, как:

Python – самый популярный выбор для Telegram-ботов благодаря библиотеке python-telegram-bot. «Python-telegram-bot» — это популярная библиотека для создания ботов в Telegram на языке Python. Она предоставляет удобный интерфейс для взаимодействия с Telegram Bot API, поддерживая все основные функции, включая отправку сообщений, обработку команд, inline-клавиатуры и другие инструменты для разработки ботов Python прост в использовании и обладает множеством готовых инструментов для взаимодействия с Telegram API. Так же будет использоваться библиотека Beautiful Soup. BeautifulSoup4 (BS4) — это библиотека Python для парсинга HTML и XML документов. Она упрощает извлечение данных из веб-страниц, позволяя работать с элементами структуры HTML, такими как теги, атрибуты и текст.Иные инструменты, используемые при разработке и написании сопутствующей документации:

1 WEB-ресурс DRAW.IO – будет использоваться для создания графической части и разработки UML-диаграмм;

2 Microsoft Office Word – для написания документации к программному продукту;

3 Microsoft Visio – для составление плана и графика работы над проектом.

Разработка проекта будет происходить на компьютере со следующими параметрами:

- процессор AMD Ryzen 7 4800H with Radeon Graphics 2.90 GHz
- объем оперативной памяти 16.00 GB;
- объем места на жестком диске 512 GB;
- видеокарта NVIDIA GeForce RTX 3050 Ti with 4 GB VRAM;
- ОС Windows 10 Pro.

1.5 Разработка плана работы над проектом

Диаграмма Ганта предназначена для наглядного отображения плана проекта, включая последовательность выполнения задач, сроки их начала и завершения. Она позволяет эффективно управлять проектом, отслеживать прогресс выполнения задач, выявлять критические пути и оптимизировать распределение ресурсов. Диаграмма помогает участникам проекта видеть временные рамки каждой задачи, точки пересечения и зависимости между этапами.

Представленная на рисунке 1 диаграмма Ганта отображает последовательность задач по разработке проекта, начиная с титульного листа и содержания, включая этапы проектирования, разработки и тестирования, и завершая подготовкой к распечатке. В ней показаны сроки выполнения каждой задачи, позволяя отслеживать их продолжительность и последовательность во времени.

Разработанная диаграмма Ганта представлена в приложении Б

2 Проектирование задачи

2.1 Разработка структуры, системы меню, навигации

В ходе разработки чат-бота были выявлены требования к его структуре. Чат-бот должен быть интуитивно понятным и обеспечивать минимальное количество шагов для выполнения задачи. Структура будет базироваться на иерархичной схеме. Так же каждый пользователь (ученик, учитель, родитель) имеет свой уникальный набор функций, что обеспечивает персонализированный доступ. Новости, расписания, информация об учреждении доступны для всех.

Структура будет строиться следующим образом. В начале идет главное меню, где происходит выборка вида пользователя, учитель, ученик, родитель. Затем в зависимости от выбора пользователя открывается доступ к его подменю с уникальным функционалом. К примеру, у учителя открывается доступ к таким подменю как: ШАГ, СППС, Нормативные документы, в каждом из которых имеются свои особенности и функционал.

Навигация строится на основе кнопок и текстовых команд. В каждом разделе есть кнопки для возврата в предыдущее меню или главное меню. Минимальное количество шагов для достижения целевой информации (не более 3 кликов). Пользователь может вернуться назад или перейти в другое меню из любой точки взаимодействия. Этот подход обеспечивает простоту и удобство взаимодействия пользователя с чат-ботом, исключая путаницу или избыточные шаги.

Разработанная диаграмма системы навигации по проекту представлена в приложении В.

2.2 Разработка UML-диаграмм

В ходе выполнения работы были разработаны следующие UML-диаграммы: диаграмма взаимодействия, диаграмма состояния, диаграмма последовательности.

Диаграмма объектов используется для визуализации и анализа того, как объекты взаимодействуют друг с другом в рамках определенного процесса или сценария. Ее основная цель — отобразить последовательность информации, которыми объекты обмениваются в рамках выполнения определенной функциональности. В разработанной диаграмме представлено взаимодействие пользователя с меню, меню с системой бота и системы бота с сайтом школы. На ней видно, как происходит обмен информацией между различными объектами системы. Разработанная диаграмма объектов представлена в приложении Г.

Диаграмма состояния — это инструмент моделирования, который используется для описания всех возможных состояний объекта в системе и переходов

между этими состояниями. Она отображает динамическое поведение системы, показывая, как объект реагирует на события и как его состояние изменяется в ответ на них. В разработанной диаграмме представлена событие поиска по дате в разделе ШАГ, у учителя. На ней видно, как система способна изменить свое состояние в ходе выполнения задачи по поиску и в зависимости от этого выдать соответствующий результат. Разработанная диаграмма состояний представлена в приложении Д.

Диаграмма последовательности — это тип диаграммы, который используется для описания взаимодействий между объектами системы в определенной последовательности. Эта диаграмма позволяет показать, как объекты взаимодействуют друг с другом, какие сообщения они отправляют и в каком порядке эти сообщения проходят. В разработанной диаграмме представлено схема взаимодействия пользователя с разделом информации и его подразделами, переходом к педагогическому составу школы. На ней прослеживается пошаговое взаимодействие пользователя между разделами и подразделами бота. Разработанная диаграмма последовательности представлена в приложении Е.

2.3 Разработка пользовательского интерфейса

В ходе разработки пользовательского интерфейса был сформирован следующий стиль. Для текста сообщений используется шрифт Arial, sans-serif, который обеспечит четкость и читаемость. Для заголовков рекомендуется использовать шрифт Roboto, который отличается хорошей визуальной иерархией и современным видом.

Основной текст 16px — размер для обычных сообщений. Заголовки первого уровня (например, приветственные сообщения) 20px — для крупных заголовков. Заголовки второго уровня 18px — для подзаголовков и важных блоков. Тексты на кнопках 18px, жирное начертание — для обеспечения видимости текста на кнопках.

Цветовая палитра Telegram-бота основана на контрастных и гармоничных оттенках для улучшения восприятия и взаимодействия с пользователем. Фоновый цвет — светлый серый для фона сообщений и интерфейса. Темный цвет для текста сообщений, обеспечивающий хорошую читаемость. Цвет заголовков — синий, который используется для выделения заголовков. Голубой для основных кнопок. Цвет активных кнопок — цвет активных элементов, изменяющийся при наведении.

Все сообщения должны быть короткими и понятными. Избегается перегрузка информации, используются простые и ясные фразы. Текстовые ссылки подчеркиваются при наведении, что служит дополнительной индикацией для пользователя.

Кнопки должны быть закругленными, с голубым фоном и белым текстом. Размер текста на кнопках — 18px с жирным начертанием. Кнопки активного состояния меняют свой цвет на темно-синий при наведении, что создает визуальную подсказку для пользователя.

Логотип Telegram-бота был выбран в виде герба школы с ее наименованием. Логотип размещен на нейтральном фоне, который будет контрастировать с его цветами, обеспечивая лучшую видимость.

Макеты UX и UI представлены в приложении Ж.

3 Реализация

3.1 Руководство программиста

Для реализации бота используется язык программирования Python. В ходе разработки были использованы такие библиотеки как: «Python-telegam-bot», «BeautifulSoup4». Библиотека python-telegram-bot, основная библиотека для взаимодействия с Telegram API, BeautifulSoup4, для парсинга HTML-страниц с веб-сайта, requests, для выполнения HTTP-запросов к сайтам. Из библиотеки «telegam-bot» использовались следующие модули: telegram, telegram.ext. Beautifulsoup использовалась как единый модуль, так же как и requests.

Для подключения выше перечисленных библиотек использовался следующий код:

```
import requests
import asyncio
from bs4 import BeautifulSoup
from telegram import Update, InlineKeyboardButton, InlineKeyboardMarkup, ReplyKeyboardMarkup, KeyboardButton
from telegram.ext import Application, CommandHandler, MessageHandler, ContextTypes, filters, CallbackQueryHandler, Updater
```

В работе бота играет ключевую роль прямое получение информации с сайта, для дальнейшего ее корректирования и использования. Для этого используется выше описанная библиотека BeautifulSoup4, совместно с requests для обработки HTTP-запросов к сайту. В коде прослеживается повторяющаяся часть отвечающая за парсинг с сайта.

Далее приведен пример кода, в котором происходит парсинг информации о родительском комитете, также акцентирую внимание на том, что ссылка на сайт школы, с которого происходит парсинг, описана в локальной переменной (url) функции start_button_callback, отвечающей за обработку сообщения с целью изменения его содержания.

```
        elif query_data == "Pbutton1":
            url="https://sch13.oktobr-grodno.gov.by/%D1%80%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8F%D0%BC/%D1%80%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D0%BA%D0%B8%D0%B9-%D0%BA%D0%BE%D0%BC%D0%B8%D1%82%D0%B5%D1%82"
            try:
```

```
            response = requests.get(url)
            response.raise_for_status()  # Проверяем статус ответа

            soup = BeautifulSoup(response.text, "html.parser")
            items = soup.find("table").find("tbody").find_all("tr")[1:]  # Пропускаем
заголовок таблицы

            # Инициализация переменной для сообщения
            message = (f"📋 Состав родительского комитета:\n\n"
                       )

            for item in items:
                cells = item.find_all("td")  # Используем find_all для всех ячеек
                if len(cells) >= 2:  # Проверяем, что в строке есть хотя бы две ячейки
                    FIo = cells[0].get_text(strip=True)  # ФИО
                    klass = cells[1].get_text(strip=True)  # Класс
                    message += f"• **{FIo}**, класс: `{klass}`\n"  # Добавляем строку
в сообщение
            message+=(f"\n\nС планом работы родителского комитета можно
ознакомиться [здесь]({url})\n\n")
            # Если message пустой после цикла
            if not message:
                message = "Данные не найдены на сайте."

        except requests.RequestException as e:
            message = f"Ошибка при получении данных с сайта: {e}"
        except Exception as e:
            message = f"Произошла ошибка: {e}"

        keyboard=[
            [InlineKeyboardButton("Попечительский                                    со-
вет",callback_data="Pbutton2"),
             InlineKeyboardButton("СППС",callback_data="Pbutton3")],
            [InlineKeyboardButton("Назад", callback_data="Pback")]
        ]
```

Для того чтобы пользователю было удобнее обращаться с ботом я использовал «InLine» кнопки, которые описываются в процедуре, отвечающей за вывод

или изменение содержания сообщения. Так же эти кнопки могут изменяться в последствии, но при этом у каждой кнопки присутствует «Callback_data» отвечающая за то, что после нажатия на кнопку будет записано в переменную «Query.data» с целью дальнейшей работы с этим.

Пример описания кнопок приведен в коде ниже:

```
inline_keyboard = [
    [
        InlineKeyboardButton("Родитель", callback_data='parent'),
        InlineKeyboardButton("Учитель", callback_data='teacher'),
        InlineKeyboardButton("Ученик", callback_data='student')
    ]
]

# Создание разметки для inline-кнопок
inline_markup = InlineKeyboardMarkup(inline_keyboard)
```

Так же для того, чтобы пользователь мог улучшить свой опыт работы с ботом была создана постоянная клавиатура, reply-клавиатура с кнопками навигации по всему основному функционалу принцип ее работы отличен от принципа работы «InLine», reply-клавиатура не отправляет обратные данные, она отправляет команду на которую реагирует обработчик, к примеру у меня это команды начало, помощь, инфо, расписание, новости.

Пример создания такой клавиатуры представлен в следующем коде:

```
# Создание reply-клавиатуры, которая будет отображаться под полем ввода
reply_keyboard = [
    [
        KeyboardButton("Новости"),
        KeyboardButton("Расписание"),
        KeyboardButton("Инфо")
    ],
    [
        KeyboardButton("Помощь")
    ],
    [
        KeyboardButton("Начало")
    ]
```

```
    ]

    # Разметка для reply-клавиатуры
    reply_markup = ReplyKeyboardMarkup(reply_keyboard, resize_keyboard=True, one_time_keyboard=False)

    # Отправка сообщения с inline-кнопками и reply-клавиатурой
    if update.message:
        # Отправка с обеими клавишами
        await update.message.reply_text(welcome_text, reply_markup=inline_markup, parse_mode="Markdown")
        await update.message.reply_text(text="Кнопки активны", reply_markup=reply_markup)
    elif update.callback_query:
        # Для обработки callback_query (например, при нажатии на inline-кнопки)
        await update.callback_query.edit_message_text(welcome_text, reply_markup=inline_markup, parse_mode="Markdown")
        await update.callback_query.message.reply_text(text="Кнопки активны", reply_markup=reply_markup)
```

Теперь подробнее про обработчики в коде, с самого начал нас встречает основной обработчик «Start» первым выводящий приветственное сообщение с line-кнопками. В нем содержится приветственный текст и сами inline-кнопки с reply-кнопками. Так же акцентирую внимание что далее по коду обработчиков встречается «async» перед «def», он предназначен для асинхронного выполнения задач при помощи «await». Это особенно полезно в разработке ботов, где требуется обрабатывать несколько запросов одновременно, не блокируя выполнение программы. Теперь после ознакомления с этой информацией введу корректировку о том, что все обработчики имеют похожую структуру с Inline-кнопкам, в связи с этим не вижу смысла заострять внимание на всех обработчиках далее по коду.

Далее приведен код обработчика «Start»:

```
# Функция для обработки команды /start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    # Текст приветствия
    welcome_text = (
```

"Здравствуйте, вас приветствует информационный телеграмм-бот средней школы №13🎓.\n\n"

"Пожалуйста, выберите пользователя или используйте следующие команды для навигации:\n\n"

" /news — для показа новостей\n"

" /Info — для получения информации о школе\n"

" /schedule — для получения расписания\n"

" /help — для получения большей информации о боте и его функционале"

)

```
# Создание inline-кнопок для выбора пользователя
inline_keyboard = [
    [
        InlineKeyboardButton("Родитель", callback_data='parent'),
        InlineKeyboardButton("Учитель", callback_data='teacher'),
        InlineKeyboardButton("Ученик", callback_data='student')
    ]
]

# Создание разметки для inline-кнопок
inline_markup = InlineKeyboardMarkup(inline_keyboard)

# Создание reply-клавиатуры, которая будет отображаться под полем ввода
reply_keyboard = [
    [
        KeyboardButton("Новости"),
        KeyboardButton("Расписание"),
        KeyboardButton("Инфо")
    ],
    [
        KeyboardButton("Помощь")
    ],
    [
        KeyboardButton("Начало")
    ]
]
```

```
    # Разметка для reply-клавиатуры
    reply_markup = ReplyKeyboardMarkup(reply_keyboard, resize_keyboard=True, one_time_keyboard=False)

    # Отправка сообщения с inline-кнопками и reply-клавиатурой
    if update.message:
      # Отправка с обеими клавишами
      await update.message.reply_text(welcome_text, reply_markup=inline_markup, parse_mode="Markdown")
      await update.message.reply_text(text="Кнопки активны", reply_markup=reply_markup)
    elif update.callback_query:
      # Для обработки callback_query (например, при нажатии на inline-кнопки)
      await update.callback_query.edit_message_text(welcome_text, reply_markup=inline_markup, parse_mode="Markdown")
      await update.callback_query.message.reply_text(text="Кнопки активны", reply_markup=reply_markup)
```

Для реализации взаимодействия с пользователем в коде создана обработка нескольких типов команд и сообщений. Основной обработчик — это CommandHandler, который реагирует на команды, и MessageHandler, реагирующий на текстовые сообщения, соответствующие определённым шаблонам. В дополнение к этому, для работы с интерактивными элементами используется CallbackQueryHandler, обрабатывающий обратные данные от inline-кнопок.

Ниже представлен код, в котором создается среда работы бота и запуск основного цикла:

```
    if __name__ == '__main__':

      app = Application.builder().token("7893771875:AAE1FCyCb9FBVOZAICHm9iJBSKUApEi-Pzo").build()
      app.add_handler(CommandHandler("start", start))
      app.add_handler(MessageHandler(filters.TEXT & filters.Regex("^(Начало|начало|глав|Глав|Главная|главная|Старт|старт|start)$"), start))
```

```
        app.add_handler(CallbackQueryHandler(start_button_callback,pat-
tern="^(parent|Pbutton1|Pbutton2|sostav|otchet|Pbutton3|Pback|teacher|Tbut-
ton1|search_date|search_theme|Tbutton2|Tbutton3|Tback|student|Sbutton1|Sbut-
ton2|Sbutton3|UVO|USSO|Kalend|Sback|back)$"))
        app.add_handler(CommandHandler(["news"], news))
        app.add_handler(MessageHandler(filters.TEXT & filters.Re-
gex("^(Новости|новости|news)$"), news))
        app.add_handler(CommandHandler(["info"], info))
        app.add_handler(MessageHandler(filters.TEXT & filters.Re-
gex("^(Инфо|информация|info)$"), info))
        app.add_handler(CallbackQueryHandler(info_button_callback, pat-
tern="^(kinfo|pkol|veb|Iback)$"))
        app.add_handler(MessageHandler(filters.TEXT & filters.Re-
gex("^(История|история)$"), story))
        app.add_handler(CallbackQueryHandler(delete_story, pattern="de-
lete_story"))
        app.add_handler(CommandHandler("schedule",schedule))
        app.add_handler(MessageHandler(filters.TEXT & filters.Re-
gex("^(расп|рсапис|Распис|Расп|расписание|Расписание|schedule)$"), schedule))
        app.add_handler(CallbackQueryHandler(handle_button_click, pat-
tern="^(fakult|Kon|schedule)$"))
        app.add_handler(CommandHandler("help", help))
        app.add_handler(MessageHandler(filters.TEXT & filters.Re-
gex("^(памагы|помощь|Помощь|help)$"),help))
        app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, han-
dle_search_input))

        # Запускаем бота
        print("Бот запущен!")
        app.run_polling()
```

4 Тестирование

4.1 Тесты на использование

При разработке данного программного продукта многие возникающие ошибки и недоработки были исправлены на этапе реализации проекта. После завершения испытания реализации программы было проведено тщательное функциональное тестирование. Функциональное тестирование должно гарантировать работу всех элементов программы в автономном режиме.

Разработанные тест-кейсы и статус их выполнения представлены в приложение З. Расписание работ над проектом представлено в таблице 5.

Таблица 5 – Расписание работ над проектом

Имя	Дата	Деятельность	Продолжительность, ч
Торгонский Федор	17.12.2024	Разработка тестов	1
Торгонский Федор	18.12.2024	Тестирование	3
Торгонский Федор	19.12.2024	Составление отчетов о найденных дефектах	2
Торгонский Федор	19.12.2024	Исправление найденных ошибок	3
Торгонский Федор	19.12.2024	Проведение регрессивного тестирования	2
Торгонский Федор	19.12.2024	Составление отчета о результатах тестирования	2

4.2 Отчёт о результатах тестирования

Элементы программы были проверены, и было установлено, что все они работают правильно и выполняют задачи, указанные в процедурах. Статистика по всем дефектам представлена в таблице 6.

Таблица 6 – Статистика по всем дефектам

Статус	Количество	Важность			
		Низкая	Средняя	Высокая	Критическая
Найдено	0	0	0	0	0
Исправлено	0	0	0	0	0
Проверено	0	0	0	0	0
Открыто заново	0	0	0	0	0
Отклонено	0	0	0	0	0

5 Руководство пользователя

Основной целью telegram-бота является помощь различным группам пользователей ориентироваться в информационном поле школы и получать актуальную, корректную информацию, в связи с этим в telegram-боте реализована система, где пользователь может свободно получить интересующую его информацию, со ссылками на оригинальные источники информации. Telegram-бот ориентирован на следующие группы пользователей: учащихся, учителей, родителей. Стоит отметить, что пользователи могут смело менять свою группу с целью получения большего спектра информации.

Одной из особенностей является то, что telegram-бот является кросс-платформенным - это значит что с ним можно взаимодействовать с различных устройств на которых возможен запуск мессенджера telegram, к примеру: смартфон, персональный компьютер, ТВ приставка и т.д.

Для запуска telegram-бота пользователь должен зайти в распространенный мессенджер telegram. Ввести в поисковую строку «тег» бота (@Info_13_Bot), затем выбрать из перечня нужного информационного бота и перейти в чат с ним, пример представлен на рисунке 1.

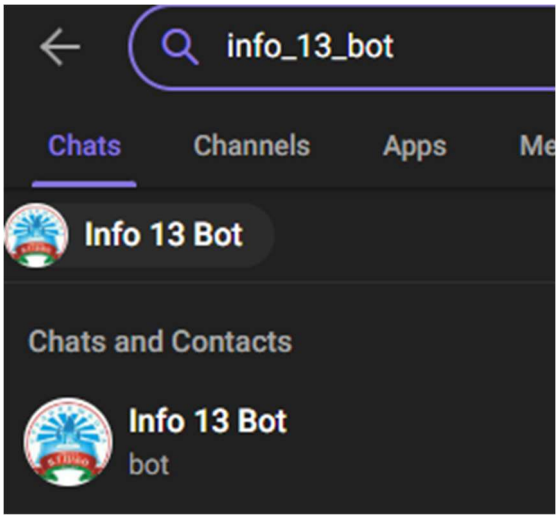

Рисунок 1 – Окно поиска telegram-бота по метке.

Для начала диалога с ботом, когда пользователь перешел в чат с ним, нужно нажать на кнопку «Start», после чего бот выведет приветственное сообщение и предложит выбрать пользователя либо же получить общие сведения, пример сообщения показан на рисунке 2.

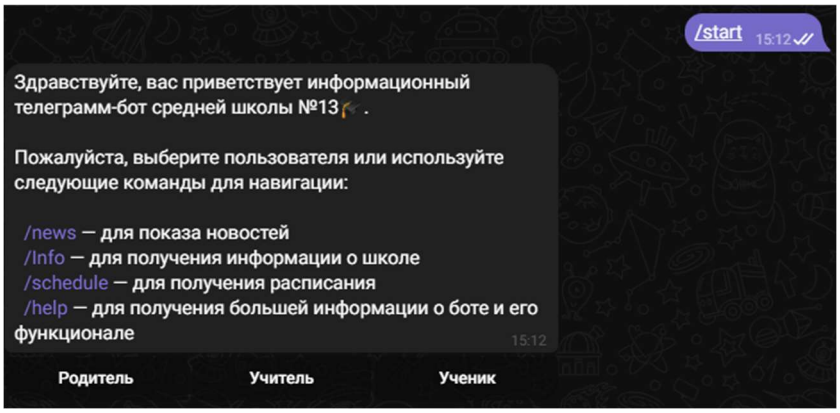

Рисунок 2 – приветственное сообщение, отправленное после нажатия кнопки старт.

Затем у пользователя может не выбирать свою вид пользователя, а взаимодействовать с общеми сведениями, такими как: новости, информация о школе, расписание, справкой по взаимодействие с ботом. Пример информационного сообщения представлен а рисунке 3

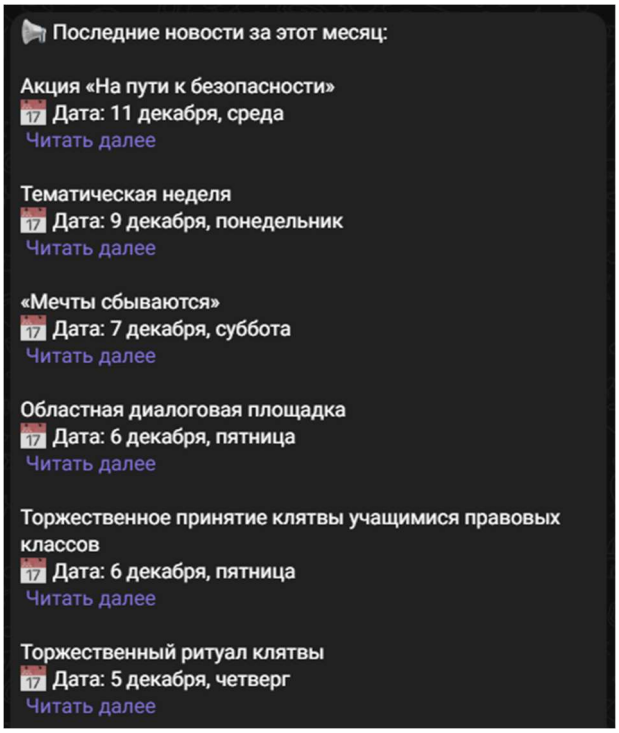

Рисунок 3 – Информационное сообщение.

Теперь чтобы вернуться к начальному сообщению после просмотра иной информации пользователь может воспользоваться кнопками навигации,

расположенными в нижней части бота, пример кнопок навигации представлен на рисунке 4.

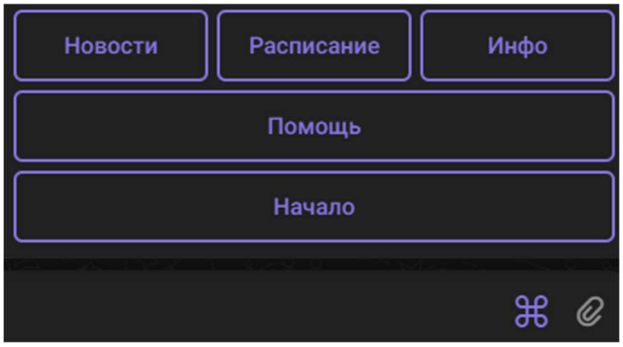

Рисунок 4 – Кнопки навигации по боту.

При помощи этих кнопок пользователь может быстро перемещаться по информации в боте, теперь если нажать на кнопку начало, то бот выведет начальное сообщение с кнопками взаимодействия откуда пользователь может детальнее ознакомиться с интересующей его категорией информации. Пример представлен на рисунке 5.

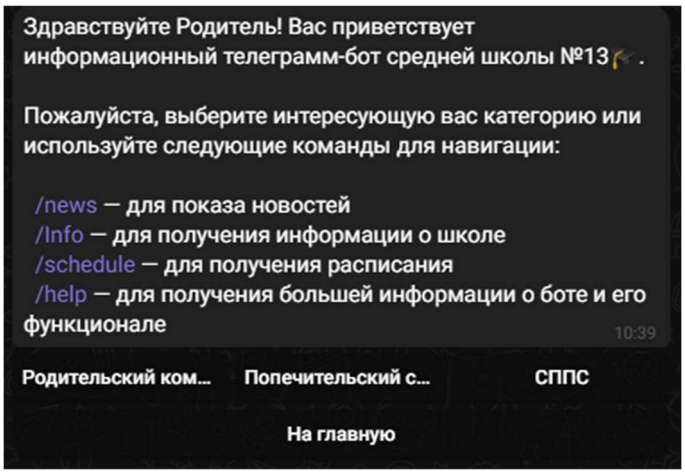

Рисунок 5 – Пример информации для группы пользователей «Родитель».

Заключение

В ходе практики была поставлена цель по разработке информационного программного продукта на тему: «Разработка информационного телеграмм-бота для ГУО «Средняя школа №13 им. В.Т. Цабо г.Гродно». В ходе разработки программного продукта были сформированы четкие цели и функционал, были определены группы пользователей, а так, же корректность предоставляемой информации с оригинальных источников. За счет выбора платформы бота в виде распространенного мессенджера была реализована кроссплатформенность.

Для реализации был использован язык программирования python с основной библиотекой для telegram API – «python-telegram-bot». Для того чтобы получать данные с сайта школы была использована библиотека BeautifulSoup4 совместно с requests.

Как было описано выше вся информация для telegram-бота берется с основного сайта школы, что позволяет получать актуальную информацию без нужды в обновлении кода. К примеру с сайта получается информация о новостях, расписании, информации о учреждении и т.д.

По степени готовности продукт находится на стадии «готов к использованию». По прохождению тест-кейсов в продукте не было выявлено критических ошибок или недостатков. Однако в продукте остались небольшие недоработки в связи с техническими ограничениями, которые никак не повлияют на работоспособность и пригодность бота.

На данный момент бота можно внедрять в эксплуатацию для школы и использовать в качестве информационного помощника с целью быстрого доступа к интересующей информации.

Список используемых источников

1 Bing. [Чат Бот]/ Microsoft Bing Search Assistant. - Режим доступа: https://github.com/microsoft/bing-search-ai-chatbot - Дата доступа: 16.12.2024

2 Gemini. [Чат Бот]/.Google LLC Gemini Assistant. - Режим доступа: https://gemini.google.com – Режим доступа: 17.12.2024

3 Библиотека python-telegram-bot в Python [Электронный ресурс]/ Python3 (https://docs-python.ru/). - Режим доступа: https://docs-python.ru/packages/biblioteka-python-telegram-bot-python/ - Дата доступа:17.12.2024

4 Графические возможности Figma [Электронный ресурс] / Figma. – Режим доступа: https://www.figma.com/ – Дата доступа: 17.12.2024

5 Книги по программированию ботов [Электронный ресурс] / GIThub. Maxim Gudkov - Режим доступа: https://github.com/MaximGudkov/book-bot – Дата доступа: 17.12.2024

6 UML диаграммы [Электронный ресурс] / Википедия. - Режим доступа: https://ru.wikipedia.org/wiki/UML – Дата доступа: 16.12.2024

7 Python-telegram-bot документация [Электронный ресурс] / – Режим доступа: https://docs.python-telegram-bot.org/en/v21.9/ Дата доступа: 17.12.2024