МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Национальный исследовательский университет

"Высшая Школа Экономики"

Факультет компьютерных наук

Департамент программной инженерии

СОГЛАСОВАНО
Научный руководитель, приглашенный преподаватель департамента программной инженерии

_____ Сосновский Г. М.
«11» мая 2023 г.

УТВЕРЖДАЮ
Академический руководитель образовательной программы «Программная инженерия», кандидат технических наук

_____ Шилов В. В.
«11» мая 2023 г.

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл	RU.17701729.06.11-01 81 01-1

Генератор документации "Радость Научника"

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.06.11-01 81 01-1-ЛУ

Исполнитель:

студент группы БПИ217

_____
/ Шаповалов А. С. /
«11» мая 2023 г.

Москва 2023

УТВЕРЖДЕН

RU.17701729.06.11-01 81 01-1-ЛУ

Инв. № подл	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата
RU.17701729.06.11-01 81 01-1				

Генератор документации "Радость Научника"

Пояснительная записка

RU.17701729.06.11-01 81 01-1

Листов 28

Москва 2023

СОДЕРЖАНИЕ

АННОТАЦИЯ

Данный программный документ представляет собой пояснительную записку к программному проекту «Генератор документации "Радость Научника"».

Раздел «Введение» включает в себя наименование программы и документ, на основании которого ведётся разработка, с указанием организации, утвердившей данный документ.

В разделе «Назначение и область применения» содержатся функциональное и эксплуатационное назначение программы и краткая характеристика области её применения.

В разделе «Технические характеристики» присутствуют следующие подразделы: постановка задачи на разработку программы, описание функционирования программы, описание и обоснование алгоритма работы программы, описание и обоснование выбора метода организации входных и выходных данных, описание работы с базой данных, описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Программный документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 Виды программ и программных документов [1];
2. ГОСТ 19.102-77 Стадии разработки [2];
3. ГОСТ 19.103-77 Обозначения программ и программных документов [3];
4. ГОСТ 19.104-78 Основные надписи [4];

5. ГОСТ 19.105-78 Общие требования к программным документам [5];
6. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом [6];
7. ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [7].

Изменения к Пояснительной записке оформляются согласно ГОСТ 19.603-78 [8], ГОСТ 19.604-78 [9].

Перед прочтением данного документа следует ознакомиться с терминологией, приведенной в Приложении 1.

1 ВВЕДЕНИЕ

1.1 Наименование программы

Наименование темы разработки: «Генератор документации "Радость Научника"».

Наименование темы разработки на английском языке: «Documentation Constructor “Mentors Joy”».

Условное обозначение темы разработки: «Documentation Constructor “Mentors Joy”**».**

1.2 Документ, на основании которого ведется разработка

Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденная академическим руководителем тема курсового проекта — Генератор документации «Радость Научника».

2 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1 Назначение программы

2.1.1 Функциональное назначение

Функциональным назначением программы является удобная организация конструктора для различных видов технической документации, быстрая и удобная работа с ним, чёткое и безошибочное соблюдение всей нормативов и стандартов для соответствующего вида документации.

2.1.2 Эксплуатационное назначение

Программа предназначена для оформления технической документации к различным проектам в соответствии со всеми актуальными ГОСТами и дальнейшего сохранения данных в различных форматах.

2.2 Краткая характеристика области применения

«Генератор документации ‘‘Радость Научника’’» — прикладная программа, разрабатываемая с целью облегчения формирования и оформления документации.

3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1 Постановка задачи на разработку программы

Главная задача, которая была поставлена – создать программу, которая позволяла бы пользователям создавать технические документы, такие как, например, техническая и пояснительная записки, без особого труда, не теряя время на механическую работу. Таким образом, продукт должен быть легок и понятен в использовании, должен поддерживать государственные стандарты в области технической документации и не требовать завышенных системных характеристик от устройства, которое используется для работы с продуктом.

3.2 Описание и обоснование архитектуры программы

3.2.1 Описание и обоснование архитектуры программы

Чтобы гарантировать возможности расширять, модифицировать, тестировать и документировать проект для его выполнения необходимо было выбрать архитектуру.

Для реализации проекта было принято решение создать телеграмм-бота, потому что этот инструмент популярен среди целевой аудитории, которой являются студенты технических вузов и молодые специалисты.

Архитектура написания телеграм-ботов зависит от многих факторов, включая выбранный язык программирования, масштаб проекта, требования к производительности и долговечности. Но в общих чертах, для написания ботов для Telegram часто используются следующие архитектуры:

1. Монолитная архитектура: Это подход, при котором весь бот написан в виде одного приложения. Такой подход может быть полезен для небольших проектов, где производительность и масштабируемость не являются первостепенными проблемами.
2. Микросервисная архитектура: В этом подходе функционал бота разбивается на отдельные независимые сервисы. Это может быть полезно для больших проектов, где масштабируемость и надежность являются важными факторами.
3. Serverless архитектура: Это подход, при котором функции бота запускаются в ответ на определенные события, и разработчику не нужно заботиться о серверах. Этот подход также может быть полезен для проектов, где требуется масштабируемость и гибкость.
4. Событийно-ориентированная архитектура: В этом подходе бот реагирует на события, которые приходят от Telegram (например, новые сообщения или обновления). Эта архитектура часто используется в комбинации с другими подходами.

Также важно отметить, что для написания телеграм-ботов используются различные библиотеки и SDK, которые предоставляют простой интерфейс для взаимодействия с API Telegram.

Телеграм-боты можно писать на любом языке программирования, который поддерживает HTTP-запросы, так как Telegram Bot API основан на HTTP. Однако некоторые языки имеют официальные и поддерживаемые сообществом библиотеки, которые делают разработку ботов проще и удобнее.

На данный момент наиболее популярными языками для написания телеграм-ботовявляются:

1. Python: Язык имеет простой синтаксис, что делает его популярным среди разработчиков. Множество библиотек, таких как python-telegram-bot, telebot и других делают разработку ботов простой и быстрой. Однако Python может быть медленнее других языков, что может стать проблемой для очень больших или высоконагруженных проектов.
2. JavaScript (Node.js): Благодаря асинхронной природе JavaScript и его производительности, он хорошо подходит для ботов, которые должны обрабатывать большое количество запросов. Библиотека telegraf позволяет легко взаимодействовать с Telegram Bot API.
3. Java: Этот язык обладает большой производительностью и надежностью, и он хорошо подходит для больших, сложных проектов. Библиотеки, такие как Java Telegram Bot API, упрощают разработку ботов на Java.
4. Go: Этот язык известен своей производительностью и эффективностью, а также простым синтаксисом. Библиотека telegram-bot-api делает разработку ботов на Go относительно простой.
5. C# (.NET): C# является мощным языком программирования, поддерживаемым многими корпоративными и крупными приложениями. Telegram.Bot - это библиотека на C#, которая облегчает разработку ботов для Telegram. Однако, C# сложнее в

использовании, чем некоторые другие языки, и он может быть избыточным для маленьких, простых ботов.

После проведенного сравнительного анализа было принято решение использовать для написания бота язык программирования Python и библиотеку telebot. Также была выбрана монолитная архитектура с элементами событийно-ориентированного подхода. Такой выбор обусловлен тем, что генератор документации является небольшим проектом и не требует сложной архитектуры и разбиения на несколько сервисов. А применение событийно-ориентированный подхода обусловлено тем, что бот должен реагировать на различные события, которые происходят в Telegram (например, получение нового сообщения или callback запроса от inline кнопок).

3.3 Описание и обоснование алгоритма работы программы

3.3.1 Описание алгоритма работы программы

Основная идея работы бота заключается в пошаговом взаимодействии с пользователем для создания различных типов документов, сохранении промежуточных результатов и предоставлении возможности конвертировать и получать созданные документы.

Алгоритм работы этого бота можно описать следующим образом:

1. Импортируются необходимые библиотеки и модули.

2. Устанавливаются настройки логирования и загружаются переменные окружения.
3. Создается экземпляр бота с использованием токена из переменных окружения.
4. Определяются переменные и структуры данных для управления состоянием бота, сохранения данных и идентификации файлов.
5. Определяются вспомогательные функции и обработчики событий для работы с ботом.
6. Основной блок программы запускает цикл опроса бота и обрабатывает события, такие как команды пользователя, нажатия кнопок и другие действия.
7. При получении команды "/start" бот приветствует пользователя и предлагает выбрать тип документа для создания.
8. После выбора типа документа, бот поэтапно запрашивает у пользователя необходимые данные и сохраняет их в блоках кода.
9. После заполнения всех данных, бот создает документ на основе шаблона и сохраняет его в указанной папке.
10. Бот отправляет созданный документ пользователю и предлагает конвертировать его в формат PDF.
11. Если пользователь соглашается на конвертацию, бот выполняет конвертацию в формат PDF и отправляет полученный файл.
12. После отправки документа и выбора дальнейших действий (создание нового документа или завершение работы), бот обновляет кнопки выбора продолжения работы и ожидает новых действий пользователя.

13. При получении команды "/stop" бот прерывает создание документа и отправляет пользователю частично заполненные результаты.
14. При возникновении ошибок, бот записывает информацию об ошибке в журнал и отправляет сообщение об ошибке пользователю.

3.3.2 Обоснование алгоритма работы программы

Рассмотрим алгоритм работы и обоснуем выбор такой реализации:

1. Импортирование необходимых библиотек и модулей: Это необходимо для использования функций и инструментов, предоставляемых этими библиотеками, которые требуются для полной функциональности бота. Помимо импорта стандартных библиотек также импортируются .py файлы, содержащие словари-шаблоны для всех типов создаваемых документов. Словари вынесены в отдельные файлы чтобы не нагружать визуально основной код и для удобства их обновления.
2. Установка настроек логирования и загрузка переменных окружения: Настройка логирования позволяет записывать сообщения об ошибках и другую информацию для последующего анализа и устранения проблем. Необходимость журналирования стала очевидна уже при первых тестах бота, потому что для отладки его работы постоянно приходилось вызывать ошибки искуственно, что было неудобно. Кроме того, важно собирать

возникающие ошибки от всех пользователей. Загрузка переменных окружения позволяет получать доступ к конфиденциальным данным, таким как токен для работы с API Telegram, без их явного указания в коде. Это делается для того, чтобы токен был известен только на устройстве-сервере и никто не мог получить доступ к управлению ботом и его исходным кодам.

3. Создание экземпляра бота: Создание экземпляра бота с использованием токена из переменных окружения позволяет установить связь с Telegram API и обрабатывать входящие и исходящие сообщения.
4. Определение переменных и структур данных необходимо для хранения состояния бота, такого как текущий тип документа, шаги ввода данных и сохранение введенных пользователем данных. Чтобы данные не смешивались между различными пользователями, были созданы словари, ключами к которым являются уникальные id пользователя, предоставляемые Телеграммом. Также используются структуры данных для идентификации файлов и управления ими.
5. Определение вспомогательных функций и обработчиков событий: Вспомогательные функции и обработчики событий определяют логику работы бота, обрабатывают входящие сообщения, команды пользователя, нажатия кнопок и другие события, выполняют необходимые действия для корректного функционирования и взаимодействуют с пользователем.

6. Основной блок программы: Этот блок запускает цикл опроса бота, который непрерывно проверяет наличие новых событий, таких как входящие сообщения или нажатия кнопок. При обнаружении событий вызываются соответствующие обработчики, которые выполняют необходимые в данном контексте действия.
7. Обработка команды "/start": Когда пользователь отправляет команду "/start", бот приветствует пользователя и предлагает выбрать тип документа, который он хочет создать. Это позволяет пользователям управлять процессом создания документов и выбирать интересующий их тип. Кроме того команда "/start" может вызываться пользователем в любой другой момент времени, в таком случае используется специальный обработчик, который отправляет сразу опрос с выбором документа без приветственного сообщения. Это сделано для того, чтобы визуально не нагружать чат.
8. Пошаговый ввод данных: После выбора типа документа, бот поэтапно запрашивает у пользователя необходимые данные, сохраняет их в соответствующих блоках кода по угникальному идентификатору пользователя и переходит к следующему шагу ввода данных. Это позволяет пользователю последовательно заполнить все необходимые разделы документа.
9. Создание документа: После заполнения всех данных, бот создает документ на основе соответствующего шаблона и сохраняет его в указанной папке. Для этого используется библиотека docxtpl,

которая позволяет заполнять шаблоны документов на основе данных, хранящихся в блоках кода.

10. Отправка документа пользователю: После создания документа, бот отправляет его пользователю с помощью метода send_document Telegram API. Благодаря этому, пользователь получает готовый документ, соответствующий выбранному типу и заполненный введёнными им данным.
11. Конвертация в формат PDF: После отправки документа, бот предлагает пользователю конвертировать его в формат PDF. Если пользователь соглашается на конвертацию, бот использует библиотеку docx2pdf для конвертации документа из формата DOCX в формат PDF. Затем полученный PDF-файл отправляется пользователю.
12. Обновление кнопок выбора продолжения работы: После отправки документа и выбора пользователем дальнейших действий (создание нового документа или завершение работы), бот обновляет кнопки выбора продолжения работы. Это позволяет пользователю выбрать следующее действие без необходимости вводить команды вручную. Подобная реализация используется для всех inline-клавиатур, используемых в данном боте. После того как пользователь нажал на одну из кнопок, данные кнопок обновляются: на выбранной появляется зеленая галочка, а на остальных – красный крестик, кроме того, все кнопки становятся неактивными, для того чтобы не нарушался общий алгоритм работы и не возникали ошибки.

13. Обработка команды "/stop": При получении команды "/stop", бот прерывает процесс создания документа и отправляет пользователю частично заполненные результаты. Это даёт пользователю возможность получить уже введенные данные, даже если документ не был полностью заполнен. Такая реализация обусловлена тем, что пользователь может захотеть посмотреть, что получается на текущем этапе заполнения документа. В таком случаем пользователю отправляется частично заполненный шаблон, в котором сохраняется общая структура, но убираются все оставшиеся незаполненные технические блоки. Благодаря такому подходу пользователь может затем самостоятельно заполнить не до конца оформленный документ.
14. Обработка ошибок: В случае возникновения ошибок в процессе работы бота, информация об ошибке записывается в журнал и отправляется пользователю. Это помогает обнаружить и исправить проблемы, а также предоставляет обратную связь пользователю о возникших проблемах. Пользователю отправляется лишь частичная информация об ошибке и небольшая инструкция, помогающая вновь приступить к заполнению документа. Это сделано для того, чтобы в случае возникновения ошибки не получалось так, что у пользователя все просто остановилось и перестало работать. Кроме того, в большинстве случаев при возникновении ошибки, бот самостоятельно начинает создание нового документа для пользователя.

3.4 Описание и обоснование выбора способа организации входных и выходных данных

Входные данные бот получает от пользователей через интерфейс чата, и они включают текстовые сообщения с ответами на вопросы бота. Эти данные затем используются для заполнения шаблонов документов, которые впоследствии сохраняются в документах формата .docx, конвертируются в PDF и отправляются обратно пользователю.

Организация входных данных:

1. Вопросы для каждого типа документа (title_page_questions, tech_spec_questions, explanatory_note_questions) хранятся в отдельных файлах. Это упрощает их обновление и поддержку.
2. Бот использует словари для хранения ответов пользователей на вопросы (title_page_code_blocks, tech_spec_code_blocks, explanatory_note_code_blocks). Ключи в этих словарях - это уникальные идентификаторы чатов, что позволяет поддерживать множество активных сессий одновременно.
3. Обработчики для каждого вопроса создаются динамически с использованием функции create_ask_and_save_handlers. Это позволяет легко добавлять новые вопросы и обработчики к ним без изменения основного кода бота.

Организация выходных данных:

1. Готовые документы сохраняются в папке output_files. Уникальные идентификаторы файлов генерируются с помощью функции generate_file_id и хранятся в file_id_dict, что позволяет найти нужный файл по его идентификатору.

2. После создания документа бот предлагает пользователю конвертировать его в PDF. Это удобно, так как PDF - это универсальный формат, который можно открыть на любом устройстве.
3. В конце сессии бот предлагает пользователю начать создание нового документа или закончить работу. Это делает бота удобным для использования, так как пользователь может сразу начать новую сессию без необходимости отправлять команду /start.

Таким образом, организация входных и выходных данных в этом боте обеспечивает его масштабируемость, гибкость и удобство использования.

3.5 Описание и обоснование выбора состава технических и программных средств

Телеграмм бот написан с использованием следующих технических и программных средств:

1. Python: Язык программирования Python выбран из-за его простоты, читаемости и широкого спектра библиотек и фреймворков. Python поддерживает множество парадигм программирования, включая процедурное, объектно-ориентированное и функциональное программирование, что делает его универсальным инструментом для разработки различных видов приложений.
2. pyTelegramBotAPI (telebot): Эта библиотека Python предоставляет простой в использовании интерфейс для API Telegram Bot. Она включает функции для обработки различных типов обновлений, которые бот может получить от Telegram, таких как сообщения,

обратные вызовы от inline-кнопок и многое другое. Выбор этой библиотеки обосновывается её простотой использования и широким функционалом.

3. DocxTemplate: Этот модуль используется для создания документов в формате .docx на основе шаблонов. Он позволяет вставлять данные, полученные от пользователя, в определенные места в документе, что делает его полезным инструментом для автоматизированного создания документов.
4. os: Этот модуль Python используется для взаимодействия с операционной системой. Он позволяет выполнять такие операции, как изменение прав доступа к файлам и удаление файлов, что необходимо для управления файлами, созданными ботом.

Такой выбор технологий и библиотек обоснован необходимостью разработки бота, который может взаимодействовать с пользователями через Telegram, получать от них информацию и создавать на основе этой информации документы в формате .docx. Python и выбранные библиотеки обеспечивают все необходимые инструменты для выполнения этих задач.

4 ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1 Ориентировочная экономическая эффективность

Расчет экономической эффективности не предусмотрен в данной работе.

4.2 Предполагаемая потребность

Данный бот может использоваться в образовательных целях: Студенты или преподаватели могут использовать этого бота для создания технической документации, связанной с проектами или курсовыми работами. Бот может сэкономить время на форматировании и структурировании документов, позволяя пользователям сосредоточиться на содержании.

Профессиональное использование: Разработчики программного обеспечения или инженеры могут использовать бота для быстрого создания технической документации для своих проектов. Это может быть особенно полезно для стартапов или небольших команд, где ресурсы могут быть ограничены.

Помощь в управлении проектами: Проектные менеджеры могут использовать этого бота для создания документации, которая поможет в управлении и координации проектов.

Кроме того, бот может использоваться для обучения техническому письму и пониманию структуры технических документов.

Таим образом, данный бот будет полезен для всех, кто работает над проектами, требующими технической документации, и поможет сэкономить значительное количество времени, обеспечивая стандартизацию и качество документов.

4.3 Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

Рассмотрим ряд сервисов и инструментов, которые обладают схожим функционалом:

1. Pandoc: Это универсальный инструмент работы с текстовыми документами и преобразования документов, который может конвертировать файлы из одного формата в другой. Он может использоваться для автоматизации создания технической документации.
2. LaTeX и Overleaf: Это системы подготовки документов, особенно популярные в академическом сообществе для создания технической и научной документации. В Overleaf содержится большое количество различных шалонов, созданных людьми со всего мира, однако не все они оформлены по актуальным стандартам, а работа с Latex требует определенных знаний и навыков. Кроме того, могут возникать трудности с конвертацией .tex документов в .docx.
3. Microsoft Word и Google Docs: Самые популярные приложения для написания документации любых видов предлагают функции для создания и форматирования различных типов документов, имеются

и в онлайн версии и в качестве приложения. Однако, у большинства студентов возникают трудности с форматированием документов, а использование шаблонов, полученных от других студентов влечет накопление ошибок.

4. Jupyter Notebooks: Это инструмент для создания документов, которые могут включать и программный код, и обычный текст. Они часто используются для создания технической документации в области науки о данных. Этот вариант подходит для оформления мелкой документации, но не обладает удобными инструментами для оформления, например, титульных листов.
5. Markdown: Это легковесный язык разметки, который часто используются для написания технической документации, но опять же его использование требует определенных знаний языка, а ограниченный набор инструментов не позволяет легко и быстро выполнить все необходимые преобразования.
6. Dr.Explain: Это сайт, поддерживающий русский язык, который предназначен для написания руководства пользователя, на выбор предоставляется несколько русскоязычных шаблонов, но данный сайт не гарантирует поддержку ГОСТов и не подходит для генерации других видов документации.
7. StepShot: Это приложение, которое подходит для написания руководства пользователя и оператора, потому что предоставляет удобные инструменты для работы со скриншотами, но не предлагает никаких возможностей по автоматизации написания и

не подходит для оформления таких элементов, как колонтитулы, оглавления, титульные листы и т.д.

Каждый из этих инструментов и сервисов имеет свои собственные преимущества и недостатки, и выбор между ними во многом зависит от конкретных потребностей пользователя. Однако, сравнительный анализ показал, что аналоги не предлагают весь функционал, реализованный в данном Телеграмм-боте.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) ГОСТ 19.101-77 Виды программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2) ГОСТ 19.102-77 Стадии разработки. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3) ГОСТ 19.103-77 Обозначения программ и программных документов. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4) ГОСТ 19.104-78 Основные надписи. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5) ГОСТ 19.105-78 Общие требования к программным документам. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6) ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7) ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8) ГОСТ 19.603-78 Общие правила внесения изменений. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.

9) ГОСТ 19.604-78 Правила внесения изменений в программные документы, выполненные печатным способом. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10) ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11) ГОСТ 15150-69 Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды. – М.: Изд-во стандартов, 1997.
12) ГОСТ 19.301-79 Программа и методика испытаний. Требования к содержанию и оформлению. //Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13) В чем писать техническую документацию // Образован URL: https://3dnews.ru/938545/v-chem-pisat-tehnicheskuyu-dokumentatsiyu-5-udobnih-programm (дата обращения: 12.02.2023).
14) pyTelegramBotAPI // Github URL: https://github.com/eternnoir/pyTelegramBotAPI (дата обращения: 01.05.2021).
15) Разаработка Telegram Ботов на Python // Botfather URL: https://botfather.dev/dashboard (дата обращения: 25.04.2023).
16) Документация docx2pdf // pypi URL: https://pypi.org/project/docx2pdf/ (дата обращения: 05.05.2023).
17) Документация docxtpl // pypi URL: https://pypi.org/project/docxtpl/ (дата обращения: 06.05.2023).

18) Документация Telegram Bot API // Telegram URL: https://core.telegram.org/bots/api (дата обращения: 01.05.2023).
19) Bot API v2: Кнопки и редактирование сообщений // mastergroosha.github.io URL: https://mastergroosha.github.io/telegram-tutorial/docs/lesson_08/ (дата обращения: 08.05.2023).
20) Как написать telegram-бота на python с помощью библиотеки telebot // BookFlow URL: https://bookflow.ru/kak-napisat-telegram-bota-na-python-s-pomoshhyu-biblioteki-telebot/ (дата обращения: 02.05.2023).
21) Как вести лог ошибок Telegram-Bot python // StackOverflow URL: https://ru.stackoverflow.com/questions/1350892/Как-вести-лог-ошибок-telegram-bot-python-Как-автоматически-перезагружать-бота-п (дата обращения: 05.05.2023).

ПРИЛОЖЕНИЕ 1

ТЕРМИНОЛОГИЯ

Таблица 1

Термин	Определение
API (Application Programming Interface)	Это набор правил и протоколов для построения и взаимодействия программного обеспечения.
Словарь (Dictionary)	В Python это структура данных, которая представляет собой набор пар ключ-значение. Каждый ключ в словаре должен быть уникальным, а значение может быть любым типом данных.
Асинхронный режим	В контексте программирования, асинхронность означает возможность выполнения задач вне основного потока выполнения.
Телеграмм-бот	Это автоматический скрипт, который может общаться с пользователями через интерфейс Telegram. Боты могут быть использованы для различных целей, от простых уведомлений до сложных интерактивных сервисов.
Callback-функция	Это функция, которая передается в качестве аргумента другой функции и выполняется после того, как основная функция завершила свое выполнение. В контексте Telegram-бота, callback-функции часто используются для обработки ответов пользователя на вопросы бота или взаимодействия пользователя с интерфейсом бота (например, нажатия кнопок).

ПРИЛОЖЕНИЕ 2

О ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПОЛЕЙ

Таблица 2

Название	**Тип**	**Назначение**
bot	telebot.TeleBot	Экземпляр бота, используется для взаимодействия с Telegram API
last_chat_id	int	Идентификатор последнего чата, в котором производилось взаимодействие
current_document_type	dict	Словарь, связывающий идентификатор чата и тип документа, который пользователь хочет создать
chat_steps	dict	Словарь, связывающий идентификатор чата и текущий шаг в процессе создания документа
tech_spec_code_blocks	dict	Словарь, связывающий идентификатор чата и словарь с данными для создания технического задания
explanatory_note_code_blocks	dict	Словарь, связывающий идентификатор чата и словарь с данными для создания пояснительной записки
title_page_code_blocks	dict	Словарь, связывающий идентификатор чата и словарь с данными для создания титульного листа

ПРИЛОЖЕНИЕ 3

ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ МЕТОДОВ

Таблица 3

Название	**Тип аргументов**	**Назначение**
handle_start	message: telebot.types.Message	Обработчик команды /start
handle_stop_command	message: telebot.types.Message	Обработчик команды /stop
handle_stop	message: telebot.types.Message	Обработчик прерывания процесса создания документа
document_template_handler	call: telebot.types.CallbackQuery	Обработчик выбора типа документа
restart_handler	call: telebot.types.CallbackQuery	Обработчик перезапуска процесса создания документа
create_document	message: telebot.types.Message, code_blocks: Dict, template_name: str, output_name: str	Создает документ на основе переданных данных и шаблона
dummy_callback	call: telebot.types.CallbackQuery	Пустой обработчик, не выполняющий никаких действий
update_conversion_buttons	call: telebot.types.CallbackQuery, conversion_text: str, no_conversion_text: str, disable_buttons: bool	Обновляет кнопки в сообщении
no_conversion_callback	call: telebot.types.CallbackQuery	Обработчик отказа от конвертации в PDF
update_buttons_to_inactive	chat_id: int, message_id: int, text: str	Обновляет кнопки в сообщении на неактивные
start	message: telebot.types.Message	Запускает начальный процесс взаимодействия с ботом
create_code_blocks	chat_id: int	Создает словари code_blocks для каждого типа документа

Изм.	Лист	№ докум.	Подп.	Дата
RU.17701729.06.11-01 81				
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					