МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

Создание телеграм бота помощника для анализа правильности форматирования документа

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Баранова Антона Дмитриевича

Научный руководитель:
Старший преподаватель _____ М.С. Портенко

подпись, дата

Зав. кафедрой:
к. ф.-м. н., доцент _____ М. В. Огнева

подпись, дата

Саратов 2025

ВВЕДЕНИЕ

В наше время люди всё чаще пользуются мобильными устройствами. На данный момент наиболее используемыми приложениями являются мессенджеры. Сейчас для пользователей важно: минимальное количество усилий для совершения действия; удобство взаимодействия при помощи мобильного устройства; отсутствие необходимости выходить из привычных приложений; мгновенная реакция на запросы пользователя. Соответствовать данным требованиям возможно при помощи чат-ботов. В данной работе выбрана тема автоматизации проверки работ для студентов, обучающихся в университете, в мессенджере Telegram. В каждом году студенты при написании дипломной или курсовой работы сталкиваются с различными трудностями, одной из них является оформление данной работы.

Значимость выпускной квалификационной работы обусловлена высокой популярностью мессенджеров и таких средств автоматизации как чат-боты среди пользователей сети Интернет. Чат-боты легко справляются с задачей облегчения рутинных задач, такие как получение информации о погоде, пробках, последних новостях и другие. Главным достоинством относительно обычных рядовых приложений является возможность совмещения всех возможностей в рамках одного мессенджера.

Цель бакалаврской работы – разработка решения, которое будет проверять правильность форматирования и составлять отчет об ошибках с возможностью автоматического исправления документа в формате docx. Это позволит ускорить процесс редактирования, повысить эффективность, обеспечить единый стиль оформления согласно стандарту университета.

Поставленная цель определила **следующие задачи**:

– Изучить принципы работы и архитектуру Telegram API

– Ознакомиться с примерами использования методов Telegram API

– Изучить библиотеку asyncio

– Изучить библиотеку Python docx

– Реализовать алгоритм, определяющий правильность форматирования документа заданным критериям (шрифт, размер, интервалы, отступы и т.д.).

– Создать телеграм-бота для загрузки документов и получения результатов проверки, а также выгрузки отчета об ошибках с возможностью автоматического исправления ошибок.

Методологические основы

Методологической основой исследования выступает интеграция трех ключевых направлений: принципы диалоговой коммерции (conversational commerce) по К. Мессине [6], задавшие модель пользовательского взаимодействия; требования стандарта оформления СТО 1.04.01–2019 СГУ [5], ставшие эталоном для алгоритмов проверки документов; и спецификации Telegram Bot API [1], на основе которой реализованы методы работы с документами (sendDocument, getUpdates, обработка callback). Инструментальная реализация базируется на асинхронном фреймворке Aiogram [3], библиотеке python-docx [4] для анализа структур документов (SectionProperties, ParagraphFormat) и паттернах backend-разработки для создания отказоустойчивых систем на Python по Колисниченко Д.Н. [7] и Lubanovich B. [8]. Интеграционные подходы включают использование Telegram Mini Apps для генерации отчетов, webhook-механизмов для обработки файлов и схем безопасного хранения данных пользователей по Сергееву А.В. [9] и Гуннарссону Й. [10]. Выбор методологии обоснован сочетанием практико-ориентированных решений (API [1,3], библиотеки [4], стандарт [5]) и теоретического фундамента (диалоговая модель [6], backend-методология [7,8]).

Теоретическая значимость бакалаврской работы заключается в систематизации подходов к автоматизированному анализу структуры и форматирования документов .docx с использованием библиотеки aiogram и интеграции этой функциональности в платформу мессенджера Telegram.

Практическая значимость бакалаврской работы обусловлена созданием готового к использованию инструмента (Telegram-бота), который позволяет:

1. Значительно сократить время студентов и преподавателей на проверку оформления работ.
2. Повысить точность выявления нарушений стандартов оформления.
3. Обеспечить единообразие оформления работ в соответствии с требованиями СГУ.
4. Упростить процесс исправления ошибок форматирования через функцию автоматического исправления.

Структура и объём работы. Бакалаврская работа состоит из введения, четырех разделов, заключения, списка использованных источников и одного приложения (исходный код бота). Общий объем работы – 57 страниц, из них 49 страниц – основное содержание, включая 5 рисунков и 3 таблицы. Список использованных источников информации – 21 наименование.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Теоретические аспекты разработки Telegram-ботов» представляет собой комплексный анализ технологических и концептуальных основ создания Telegram-ботов, начиная с исследования эволюции цифровой коммуникации. В современном цифровом ландшафте мессенджеры вновь заняли лидирующие позиции после периода доминирования социальных сетей, что подтверждается данными исследования Yota (2022), фиксирующего 15%-ный рост аудитории мессенджеров в России. Детальный анализ демографической структуры выявил ключевые пользовательские группы: наиболее активными оказались лица 25-35 лет (32%) и 35-45 лет (26%), тогда как подростки до 18 лет составили лишь 8% аудитории.

Особое внимание в исследовании уделено Telegram как технологической платформе, выделяющейся уникальными характеристиками.

Архитектурные преимущества включают кроссплатформенную реализацию на C++ и распределенную сеть серверов в Германии и США, обеспечивающую глобальную доступность. Система безопасности построена на двухуровневом шифровании MTProto, защищающем как метаданные, так и содержимое сообщений, с дополнительной опцией секретных чатов, где криптографические ключи генерируются локально. Функциональные отличия Telegram проявляются в отсутствии ограничений на размер передаваемых файлов (в отличие от таких платформ, как VK), запрете на встроенную рекламу и развитой экосистеме для разработчиков. Последняя включает открытый API, поддержку ботов (как стандартных, так и inline-версий), каналов для односторонней коммуникации и инновационных интеграций, таких как блокчейн-платформа TON.

Далее исследование переходит к систематизации чат-ботов, выделяя два принципиально разных типа. Кнопочные боты с ограниченным набором опций оптимальны для задач с четко определенными параметрами, тогда как текстовые NLP-боты, основанные на обработке естественного языка, способны анализировать произвольные запросы и адаптироваться через машинное обучение. Исторический экскурс показывает эволюцию от первых развлекательных скриптов (пионером стал чат-бот Майкла Молдина в 1994 году) до современных enterprise-решений, трансформирующих такие отрасли, как здравоохранение (автоматизация записи к врачу), ритейл (управление заказами) и юриспруденция (составление документов).

Обоснование выбора Python в качестве основного языка разработки базируется на совокупности технологических и практических факторов. Ключевые преимущества включают динамическую типизацию, лаконичный синтаксис, минимизирующий объем кода, и богатую экосистему библиотек – от asyncio для асинхронных операций до специализированных инструментов вроде python-docx для работы с документами. Практическая применимость Python подкрепляется его универсальностью: язык одинаково эффективен для

обработки HTTP-запросов, компьютерного зрения, научных вычислений и интеграции с Telegram Bot API.

Завершается раздел сравнительным анализом существующих сервисов автоматизированной проверки документов. Рассмотрены такие решения, как Contract by Embedika с AI-анализом договоров на основе эвристических правил, Azure Document Intelligence для облачной обработки структурированных данных, а также Smart Engines и CORRECT, специализирующиеся на распознавании документов с применением NLP. Критический вывод указывает на отсутствие полных аналогов, способных адресовать специфику вузовских стандартов оформления работ, что подтверждает актуальность и новизну разрабатываемого решения.

Второй раздел «Telegram Bot API» посвящен детальному анализу архитектуры Telegram Bot API и инструментальных средств, используемых для реализации бота-ассистента. Исследование начинается с фундаментальных принципов работы Telegram Bot API, основанного на HTTP-запросах с ответами в формате JSON. Ключевое внимание уделяется критически важным методам API: sendMessage для отправки текстовых сообщений с поддержкой форматирования Markdown/HTML, sendDocument для загрузки файлов через идентификаторы file_id или прямые URL, а также editMessageText для динамического обновления контента после отправки. Особо рассматриваются два механизма получения обновлений: традиционный лонгполлинг через getUpdates и вебхуки (setWebhook) для обработки событий в реальном времени, при этом подчеркиваются требования к безопасности, включая обязательное использование HTTPS для вебхуков и защиту токена бота от несанкционированного доступа.

Далее исследование переходит к интерактивным элементам и интеграционным возможностям. Анализируются UI-компоненты, такие как кастомные клавиатуры с кнопками-шаблонами, упрощающие пользовательское взаимодействие, и inline-кнопки, встраиваемые непосредственно в тело сообщений, с обработкой через метод

answerCallbackQuery. Рассматриваются практические аспекты интеграции бота с внешними системами: подключение к СУБД для хранения истории проверок и отчетов, использование вебхуков для синхронизации с CRM-системами образовательных учреждений, а также потенциал подключения платежных шлюзов для реализации премиум-функционала.

В части практического применения API демонстрируются реальные сценарии использования методов. На примерах иллюстрируется обработка входящих документов: при получении файла формата .docx бот мгновенно подтверждает прием файла и инициирует процедуру анализа. Особое внимание уделено динамическому редактированию сообщений через editMessageText, что позволяет реализовывать интерактивные интерфейсы с прогресс-барами или обновляемыми результатами проверки.

Центральное место в разделе занимает сравнительный анализ библиотек для работы с Telegram API, где обосновывается выбор Aiogram в качестве основного инструмента. Подробно рассматриваются его преимущества перед альтернативами (такими как python-telegram-bot): полностью асинхронная архитектура на базе asyncio, обеспечивающая обработку свыше 1000 запросов в секунду; строгая аннотация типов, повышающая надежность кода; своевременная поддержка новых функций Telegram API. Особо выделяются продвинутые возможности Aiogram: реализация конечных автоматов (FSM) для многошаговых сценариев (например, уточнение параметров проверки), встроенные middleware для логирования и ограничения запросов, а также удобная система фильтров для маршрутизации сообщений.

Завершается раздел исследованием библиотеки python-docx как инструмента анализа документов. Описывается её функционал для решения поставленной задачи: чтение структурных элементов документа (заголовков, параграфов, секций), извлечение метаданных форматирования (размер шрифта, выравнивание, отступы, интервалы), программная проверка соответствия стилевым требованиям через анализ paragraph_format. Подчеркивается роль библиотеки в реализации функции автокоррекции,

которая позволяет автоматически исправлять выявленные нарушения без ручного вмешательства пользователя.

Третий раздел «Разработка телеграмм-бота» описывает процесс создания бота. Приведены этапы регистрации бота через @BotFather и получения токена. Детально описана реализация обработки базовых типов сообщений (команда /start, прием документов) с использованием асинхронной модели Aiogram. Разработан и реализован ключевой алгоритм проверки документа, включающий:

Проверку полей документа (левое: 25 мм, правое: 15 мм, верхнее/нижнее: 20 мм ±1 мм).

Анализ структуры на наличие обязательных разделов (Введение, Заключение, Список источников литературы).

Проверку форматирования текста: шрифт (Times New Roman), размер (14 пт), межстрочный интервал (1.5), отступ первой строки (1.25 см), выравнивание (по ширине для основного текста, по центру для обязательных заголовков, по левому краю для остальных), наличие полужирного начертания у заголовков.

Проверку оформления приложений: использование разрешенных букв кириллицы (исключая Ё, Й, О и др.), отсутствие дубликатов, соблюдение алфавитного порядка (А, Б, В...).

Алгоритм учитывает прямое форматирование, стили и низкоуровневую XML-структуру .docx. Ошибки группируются по типам, приводятся конкретные примеры из текста (до 3 уникальных на тип). Реализована функция fix_document, выполняющая автоматическое исправление выявленных ошибок форматирования (кроме структурных) с сохранением оригинального оформления титульного листа и генерацией исправленного файла. Вывод по разделу: Разработанный бот обладает функционалом, полностью отвечающим поставленной цели – автоматизированной проверке и исправлению оформления документов по стандарту СГУ.

Четвёртый раздел «Проверка работоспособности функционала бота для проверки оформления документов» посвящен тестированию. Проверены различные сценарии:

- Документы с заведомо неправильными полями (успешно обнаружены).
- Документы без обязательных структурных разделов (успешно обнаружено отсутствие).
- Документы с ошибками форматирования текста (неверный шрифт, размер, интервал, выравнивание – успешно обнаружены).
- Документы с ошибками в оформлении заголовков (отсутствие жирного начертания, неверное выравнивание – успешно обнаружены).
- Документы с некорректно оформленными приложениями (запрещенные буквы, дубликаты, нарушение порядка – успешно обнаружены).
- Загрузка файлов неподдерживаемых форматов (PDF – корректное сообщение об ошибке).
- Загрузка документа, полностью соответствующего СТО (успешное подтверждение соответствия).

Тестирование подтвердило корректность работы алгоритма проверки, точность формирования отчетов об ошибках, работоспособность функции автоматического исправления и устойчивость бота к нестандартным входным данным. Вывод по разделу: Функционал бота протестирован и готов к практическому использованию.

Выпускная квалификационная работа была направлена на упрощение процесса редактирования, повышения эффективности работы пользователей и обеспечение единого стиля оформления, при помощи разработанного автоматизированного телеграм-бота для проверки форматирования в формате docx.

Поставленные цели были достигнуты, при помощи реализации ключевых этапов разработки. Разработанный алгоритм стал основой проекта, при проверке форматирования, созданной на основе точного анализа

требований к оформлению учебных документов. Этот алгоритм выполняет анализ и сравнивает полученные данные с заданными критериями при помощи детального анализа параметров текста его шрифта, размера, междустрочного интервала, отступов и других аспектов. Также алгоритм позволяет автоматически исправлять ошибки. Такая систематизация процесса проверки позволяет существенно снизить влияние человеческого фактора на качество оформления. Одновременно с этим был создан и внедрен телеграм–бот с простым и понятным интерфейсом для загрузки документов. Бот обеспечивает быстрый анализ и выдает пользователю результаты проверки в виде детальных отчетов. Были изучены библиотеки: python-docx и python telegram bot. На их основе создан функционал. Этот функционал отвечает за чтение документов, извлечение критически важных данных о форматировании и генерации отчетов. Эти отчеты дают точные указания на места в документе, в которых обнаружены нарушения установленных правил оформления. Проведенные испытания подтвердили стабильную работу бота, а также выявления ошибок и корректность формирования отчетов.

Данная разработка имеет высокую практическую ценность, поскольку позволяет значительно ускорить процесс проверки документов, минимизировать ошибки в оформлении и гарантировать их соответствие установленным корпоративным или законодательным требованиям. Внедрение такого инструмента особенно выгодно для образовательных учреждений, государственных структур и компаний, где унифицированный стиль документооборота является критически важным. В будущем планируется расширить возможности системы за счет интеграции с веб-интерфейсами и мессенджерами, добавления поддержки форматов PDF и ODT, а также использования машинного обучения для автоматического исправления ошибок. Кроме того, предусмотрена возможность адаптации решения под индивидуальные потребности различных организаций. В итоге, разработанный инструмент не только эффективно решает проблему контроля

форматирования, но и закладывает фундамент для оптимизации работы с документами в целом.

Основные источники информации:

1. Официальная документация Telegram Bot API. URL: https://core.telegram.org/bots/api (дата обращения 20.02.2025)
2. Python-Telegram-Bot Documentation. URL: https://python-telegram-bot.org/ (дата обращения 20.02.2025)
3. Aiogram Documentation. URL: https://docs.aiogram.dev/ (дата обращения 20.02.2025)
4. python-docx Documentation. URL: https://python-docx.readthedocs.io/ (дата обращения 20.02.2025)
5. СТО 1.04.01–2019 Саратовского национального исследовательского государственного университета имени Н.Г. Чернышевского "Курсовые работы (проекты) и выпускные квалификационные работы. Порядок выполнения, структура и правила оформления".
6. Мессина К. 2016 will be the year of conversational commerce. (Концепция диалоговой коммерции).
7. Колисниченко, Д. Н. Современная разработка backend-приложений на Python. 2023 г.
8. Bill Lubanovich, Introducing Python: Modern Computing in Simple Packages. – 2025.
9. Сергеев А. В. Telegram Mini Apps: Разработка Web-приложений внутри Telegram. (БХВ-Петербург). 2024 г.
10. Гуннарссон, Й. Building Bots with Node.js: Create Conversational Bots for Slack, Facebook Messenger, and Telegram. 2023 г.