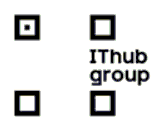
**Автономная некоммерческая организация**

**профессионального образования**

**«Международная Академия Информационных Технологий «ИТ ХАБ»**

115093, Москва, вн.тер.т. Муниципальный округ Даниловский, ул. Дубининская, д. 96

ОГРН: 1207700265255, ИНН: 9709063913

|  |  |
| --- | --- |
| Специальность | УТВЕРЖДАЮ |
| 09.02.07  (шифр) | Зам. директора по методической работе |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Г.И. Янгирова / |
| дневное | (подпись) (инициалы, фамилия) |
| (отделение) | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г. |
|  |  |

|  |
| --- |
| **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** |
| к дипломному проекту |
| Разработка Telegram-бота с автоматизированной рассылкой новостей |
| (тема дипломного проекта) |
| (09.02.072302000ПЗ) |
| (обозначение документа) |

|  |  |  |
| --- | --- | --- |
| Рецензент | Выполнил студент группы 4Р2: | |
| \_\_\_\_\_\_\_\_ И.О.Борисов | \_\_\_\_\_ Л. Т. Александрович |  |
| (подпись) (инициалы, фамилия) | (подпись) (инициалы, фамилия) |  |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_2024 г. | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 2024 г. |
|  | Руководитель проекта | |
|  | \_\_\_\_\_\_ А. М. Пастушков |  |
|  | (подпись) (инициалы, фамилия) |  |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_ | 2024 г. |

Москва 2024

Содержание

[Введение 3](#_Toc170072557)

[1. Постановка задачи 5](#_Toc170072558)

[1.1. Анализ предметной области 5](#_Toc170072559)

[1.2. Область применения 7](#_Toc170072560)

[1.3. Анализ конкурентных продуктов 8](#_Toc170072561)

[1.4. Выводы по первой главе 9](#_Toc170072562)

[2. Проектирование программного продукта 10](#_Toc170072563)

[2.1. Выбор методов и средств проектирования 10](#_Toc170072564)

[2.1.1. Выбранные библиотеки 10](#_Toc170072565)

[2.1.2. Средства тестирования и разработки 13](#_Toc170072566)

[2.2. Логическое прототипирование системы 13](#_Toc170072567)

[2.3. Создание пользовательского интерфейса 14](#_Toc170072568)

[2.4. Требования к программному продукту 21](#_Toc170072569)

[4. Экономическое обоснование 30](#_Toc170072570)

[4.1. Целевая аудитория 30](#_Toc170072571)

[4.2. Модель монетизации 31](#_Toc170072572)

[4.3. Выводы по четвертой главе 35](#_Toc170072573)

[Заключение 37](#_Toc170072574)

[Список литературы 39](#_Toc170072575)

Введение

В настоящее время с быстрым течением жизни многие люди стремятся быть в курсе последних новостей и событий, обращаясь к своим предпочитаемым новостным источникам. Чем больше потребность человека в получении свежих новостей, тем явнее стоит проблема неудобства последовательного посещения каждого новостного источника, будь это сайт или автономное приложение. Каждое утро миллионы людей начинают свой день с просмотра новостей, однако обилие различных платформ делает этот процесс утомительным и неэффективным. Множество новостных сайтов, блогов и информационных порталов требуют постоянного переключения между ними, что занимает много времени и снижает продуктивность. Кроме того, разные источники предоставляют новости в разных форматах и с различными временными задержками, что может привести к пропуску важных событий. В условиях современного ритма жизни, когда каждая минута на счету, этот подход становится все менее приемлемым. Очевидным решением для повышения эффективности данного процесса является агрегация всей нужной информации в одном месте. Задумавшись над этим, ответ пришел сам собой. Таковым местом может отлично стать сверхпопулярная соцсеть Telegram, а именно Telegram-бот.

Telegram-боты в своем подавляющем большинстве – крайне легковесные приложения, имеющие крайне скромное количество функций. По своей задумке боты как раз и должны решать одну небольшую конкретную задачу. Telegram, в свою очередь, сегодня есть чуть ли не у каждого и любой возрастной категории. Не придется ничего устанавливать и учится пользоваться, что, кстати, является несомненным плюсом для людей преклонного возраста, положительно сказываясь на привлечении пользовательской базы.

Итак, бот позволяет получать актуальные данные из различных источников в режиме реального времени, ускоряя процесс информирования пользователя. Бот обеспечивает автоматизацию процесса, экономя время и усилия, которые могли бы быть затрачены на ручной поиск и обработку данных.

Актуальность данной темы связана с необходимостью быстро получать свежие новости в условиях информационной перегрузки. Современные пользователи хотят мгновенно узнавать о событиях и экономить время, избегая посещения множества сайтов. Централизованные решения упрощают доступ к новостям из различных источников. Персонализация новостного потока повышает удовлетворенность пользователей. В эпоху глобализации удобный доступ к разнообразной информации становится все более важным.

Разработанный проект не только позволит закрыть актуальную во все времена потребность в получении новой информации быстрее и проще, но также персонализировать рассылку под пользователя, даже не ограничиваясь строго новостными источниками в классическом значении слова.

Цель дипломного проекта – разработать Telegram-бота с автоматизированной рассылкой новостей.

Для достижения цели поставлены следующие задачи:

* изучить средства парсинга актуального контента с веб-сайтов
* провести анализ конкурентных продуктов
* разработать прототип бота
* разработать базу данных
* провести тестирование с привлечением потенциальной аудитории

Объектом исследования является исследование принципов работы и методов парсинга актуального контента с RSS-лент и дальнейшая интеграция с Telegram-ботом.

Предметом исследования является разработка Telegram-бота на языке Python и последующая реализация RSS-парсера.

**1. Постановка задачи**

**1.1. Анализ предметной области**

С развитием интернета возникла потребность в актуальном контенте на сайтах. Поначалу обновление сайта вручную не представляло особых трудностей, но с ростом объема контента и инструментов редактирование становилось все более сложным. В этом контексте важную роль стали играть RSS ленты, представленные в 1999 году. RSS, что расшифровывается как “Really Simple Syndication”, позволили пользователям получать обновления с сайтов в автоматическом режиме.

RSS ленты стали незаменимым инструментом для агрегаторов новостей и других сервисов, нуждающихся в постоянном обновлении информации. С их помощью сайты могли публиковать обновления, которые автоматически доставлялись подписчикам. В результате потребители контента начали использовать программы-агрегаторы для получения новостей и статей в удобном формате, что значительно упростило доступ к актуальной информации.

С тех пор развитие технологий парсинга и автоматизации сбора данных позволило создавать более сложные системы, которые не только собирают данные из различных источников, но и анализируют их, предоставляя пользователям персонализированный контент. Эти инструменты стали важной частью экосистемы интернета, обеспечивая оперативное и эффективное распространение информации.

Популярность RSS лент обеспечивают следующие факторы:

1. Автоматическое обновление контента: RSS-ленты позволяют пользователям получать свежие обновления с сайтов автоматически, без необходимости вручную проверять каждый источник.

2. Удобство использования: программы-агрегаторы, такие как Feedly и Inoreader, собирают и отображают контент из различных источников в одном месте, что упрощает чтение и управление информацией.

3. Персонализация: пользователи могут подписываться только на те ленты, которые их интересуют, получая таким образом контент, релевантный их потребностям и интересам.

4. Экономия времени: с помощью RSS-лент пользователи могут быстро просматривать заголовки и краткие описания статей, выбирая для чтения только то, что им действительно интересно.

5. Поддержка множества форматов контента: RSS-ленты поддерживают не только текстовые статьи, но и мультимедийный контент, такой как подкасты и видеоролики, что расширяет возможности использования.

6. Кроссплатформенность: RSS-ленты можно использовать на разных устройствах и операционных системах, что обеспечивает доступ к контенту в любое время и в любом месте.

7. Отсутствие рекламы и алгоритмов: в отличие от социальных сетей и некоторых других источников, RSS-ленты предоставляют контент без дополнительных рекламных вставок и алгоритмических фильтров, что делает их более прозрачными и предсказуемыми для пользователей.

8. Простота интеграции для разработчиков: для создателей сайтов интеграция RSS лент является относительно простой задачей, что способствует их широкому распространению.

9. Стабильность и независимость: RSS-ленты продолжают работать, даже если основной сайт недоступен временно, обеспечивая стабильный доступ к информации.

10. Открытый стандарт: RSS является открытым стандартом, что позволяет любому разработчику создавать программы для работы с лентами, способствуя развитию экосистемы вокруг этого формата.

RSS ленты имеют несколько преимуществ над обычным парсингом веб-сайтов. Во-первых, они предоставляют данные в структурированном и стандартизированном формате, что упрощает их обработку и интеграцию. В отличие от парсинга, который требует анализа HTML-кода и может быть чувствителен к изменениям структуры сайта, RSS-ленты остаются стабильными и надежными источниками информации.

Парсинг данных возможен даже с сайтов, которые не являются строго новостными. Например, с помощью парсинга можно извлекать информацию о продуктах с интернет-магазинов или данные из социальных сетей. Однако многие не новостные сайты также предоставляют RSS ленты, что упрощает получение обновлений.

Примеры таких сайтов включают:

* YouTube: предлагает RSS-ленты для подписок на каналы и плейлисты, позволяя получать уведомления о новых видео.
* GitHub: предоставляет RSS-ленты для репозиториев, чтобы отслеживать изменения в коде и новые релизы.
* Reddit: каждый сабреддит имеет свою RSS-ленту, что позволяет получать обновления о новых постах.
* Amazon: предлагает RSS-ленты для отслеживания новых продуктов и изменений в ценах.

Использование RSS-лент на таких сайтах позволяет пользователям легко следить за обновлениями и получать нужную информацию без необходимости сложного парсинга и обработки данных.

**1.2. Область применения**

Добавление RSS-ленты на сайт может значительно улучшить его функциональность и удобство для пользователей, особенно если сайт регулярно обновляется новыми статьями, новостями или блог-постами. Для сайтов с широкой и разнообразной аудиторией RSS-ленты становятся полезным инструментом, позволяя подписчикам легко отслеживать актуальную информацию. Мультимедийные ресурсы, такие как видеохостинги и подкаст-платформы, могут использовать RSS ленты для уведомления пользователей о новых выпусках, а образовательные и информационные ресурсы, включая учебные платформы и научные журналы, получают возможность эффективно распространять новые материалы и публикации.

Интернет-магазины и платформы электронной коммерции могут внедрить RSS-ленты, чтобы информировать клиентов о новых продуктах и акциях, а форумы и социальные сети — для уведомления о новых обсуждениях и постах. Технологические ресурсы, такие как репозитории кода и сайты с технической документацией, могут использовать RSS-ленты для информирования о новых версиях и обновлениях. В маркетинговых целях компании и бренды могут распространить новости о своей деятельности, пресс-релизы и другие материалы с помощью RSS-лент.

Кроме того, наличие RSS-ленты может улучшить SEO сайта и привлечь больше трафика благодаря удобству доступа к обновлениям и их распространению на других платформах. Если ваш сайт соответствует одному или нескольким из этих критериев, добавление RSS-ленты может значительно повысить его полезность и привлекательность для пользователей.

**1.3. Анализ конкурентных продуктов**

При анализе продуктов-конкурентов можно отметить, что на рынке уже присутствуют решения, которые предоставляют схожие функции по рассылке новостей с использованием RSS-лент. Такие сервисы, как FeedlyBot и NewsBot, успешно собирают и распространяют контент из различных источников, предлагая пользователям актуальные обновления.

Однако мой бот выделяется среди конкурентов благодаря форме Telegram бота, более продвинутому парсингу статического контента и поддержке формата Atom. В отличие от аналогов, мой парсер способен эффективно извлекать данные даже с тех сайтов, которые не предоставляют RSS-ленты, обеспечивая более широкий охват и актуальность информации.

Формат Atom, как и RSS, используется для синдикации контента, однако Atom является более новым стандартом с улучшенной поддержкой метаданных и гибкостью в структуре данных. В то время как RSS фокусируется на простоте и легкости использования, Atom предлагает больше возможностей для детальной настройки и расширения. Поддержка Atom в моем боте позволяет пользователям получать информацию из источников, использующих этот формат, что делает бот более универсальным и функциональным.

**1.4. Выводы по первой главе**

Итак, были рассмотрены возможности создания Telegram-бота для рассылки новостей с использованием RSS-лент. Была исследована область применения RSS-лент, включая их использование на различных типах сайтов, таких как YouTube и GitHub. Кроме того, был проведен анализ потребностей пользователей, выделены такие важные аспекты, как актуальность информации, персонализация, удобство использования и настройка уведомлений. Были выделены преимущества данного бота перед конкурентами, такие как продвинутый парсинг статического контента и поддержка формата Atom.

**2. Проектирование программного продукта**

**2.1. Выбор методов и средств проектирования**

В процессе разработки мною были выбраны такие ключевые технологии как язык программирования Python и библиотека telebot. Использование Python и библиотеки telebot для создания Telegram-бота обусловлено их простотой, удобством и функциональностью. Python обеспечивает быстрое и удобное написание кода благодаря своему чистому синтаксису и богатой экосистеме библиотек. telebot, в свою очередь, предоставляет мощный и простой в использовании интерфейс для работы с Telegram API, что позволяет сосредоточиться на логике бота, а не на технических деталях реализации. telebot является активно развиваемой библиотекой с регулярными обновлениями и улучшениями. Это гарантирует совместимость с последними версиями Telegram API и добавление новых возможностей по мере их появления. Эти факторы сделали Python и telebot идеальным выбором для моего проекта.

**2.1.1. Выбранные библиотеки**

Как уже было сказано ранее, для работы с Telegram API я выбрал библиотеку telebot, которая предоставляет удобный интерфейс для создания и управления ботами. Эта библиотека позволяет легко обрабатывать команды, сообщения и взаимодействовать с пользователями через чат.

Преимущества библиотеки telebot

1. Простота использования:

* + telebot предоставляет удобный и интуитивно понятный интерфейс для взаимодействия с Telegram API. Библиотека упрощает создание и управление ботами, предоставляя готовые функции для обработки сообщений, команд и других событий.

2. Богатый функционал:

* + telebot поддерживает широкий спектр функций Telegram API, включая отправку сообщений, управление клавиатурами, работу с медиафайлами и многое другое. Это позволяет создавать ботов с разнообразным функционалом без необходимости писать сложный код.

3. Активное развитие и поддержка:

* + telebot является активно развиваемой библиотекой с регулярными обновлениями и улучшениями. Это гарантирует совместимость с последними версиями Telegram API и добавление новых возможностей по мере их появления.

Для хранения данных я использовал SQLite, который обеспечивает надежное и легковесное решение для хранения информации о пользователях и их подписках, о чем будет рассказано позже. Для хранения данных в моем Telegram-боте я выбрал SQLite, поскольку она предоставляет простое и эффективное решение, идеально подходящее для задач, которые решает бот. SQLite поставляется вместе с Python, что значительно упрощает процесс установки и настройки, позволяя сосредоточиться на разработке основной логики бота. Несмотря на свою простоту, SQLite обеспечивает надежное хранение данных, достаточное для небольших и средних проектов, таких как мой бот. Использование SQLite также исключает необходимость настройки и администрирования более сложных систем управления базами данных, что делает процесс разработки быстрее и менее ресурсозатратным. Этот выбор позволил мне создать эффективное и устойчивое приложение без излишней сложности. Я использовал соответственную стандартную библиотеку sqlite3 для управления базой данных и обработки SQL-запросов.

Парсинг новостных лент осуществляется с помощью специально разработанных утилит rss\_utils, являющихся обёрткой библиотеки feedparser, которые включают функции parse\_feed и find\_rss\_feeds. Эти функции позволяют находить RSS ленты на указанных сайтах и извлекать актуальные новости из них. В этом также ключевую роль играет библиотека BeautifulSoup. Она предоставляет мощные инструменты для парсинга HTML и XML документов, что делает ее идеальным выбором для задачи поиска и извлечения RSS и Atom лент.

Сама библиотека feedparser — это мощный инструмент для работы с RSS и Atom лентами, который я использовал в своем Telegram-боте. Она позволяет легко извлекать и обрабатывать данные из различных новостных источников. feedparser славится своей гибкостью и способностью работать с разными форматами новостных лент, предоставляя разработчикам удобный интерфейс для доступа к содержимому лент.

Однако, несмотря на свою гибкость, feedparser сталкивается с одной важной проблемой: создатели различных сайтов сами решают, как генерировать свои ленты. Это означает, что структура и содержание RSS и Atom лент могут значительно варьироваться от одного сайта к другому. Некоторые ленты могут содержать дополнительные теги и метаданные, другие могут использовать нестандартные форматы или включать специфические элементы, которые сложно предсказать.

Из-за этого feedparser не всегда может обработать все возможные вариации лент идеально. Разработчикам часто приходится адаптировать свой код для обработки специфических особенностей отдельных лент или внедрять дополнительные проверки и фильтры, чтобы обеспечить корректное извлечение данных. Несмотря на эти проблемы, feedparser остается незаменимым инструментом для работы с новостными лентами, предоставляя обширные возможности для обработки и использования информации из самых разных источников.

Кроме всего прочего, я использовал библиотеку dataclasses, чтобы упростить управление данными и улучшить структуру кода. dataclasses в Python позволяют легко создавать классы для хранения данных с минимальными усилиями по написанию кода. Это обеспечивает более чистый и понятный код, упрощает работу с объектами данных и их атрибутами. В моем проекте dataclasses использовались для хранения временных данных и состояний пользователей, что повысило читаемость и поддерживаемость кода, а также облегчило дальнейшее расширение функциональности бота.

**2.1.2. Средства тестирования и разработки**

Для разработки моего Telegram-бота я использовал Visual Studio Code и Telegram по ряду причин.

Visual Studio Code — это мощный и гибкий редактор кода, который идеально подходит для разработки на Python. Он предлагает множество полезных функций, таких как автодополнение кода, возможность установки расширений для улучшения функциональности, и интеграцию с терминалом. Благодаря этим возможностям, VS Code значительно ускоряет процесс разработки, помогает избежать ошибок и облегчает управление проектом.

Telegram, очевидно, был выбран для тестирования и отладки бота непосредственно в среде, в которой он будет работать. Это позволило легко проверять его функциональность в реальных условиях.

**2.2. Логическое прототипирование системы**

Общий сценарий использования бота выглядит следующим образом:

1. Запуск бота: Пользователь находит и запускает бота в Telegram.
2. Подписка на новости: Пользователь отправляет боту URL сайта, с которого он хочет получать новости. Бот анализирует страницу сайта и находит ссылки на новостные ленты.
3. Обработка контента: Бот проверяет доступность и корректность найденных новостных лент, загружает их содержимое и извлекает данные о новостях, такие как заголовок, ссылка, дата публикации и изображение.
4. Отправка новостей: Пользователь выбирает нужную ленту, и бот начинает регулярную проверку на наличие новых записей. При появлении новых записей бот отправляет пользователю уведомления с основной информацией о новости.
5. Получение обновлений: Пользователь получает обновления новостей в режиме реального времени.

Разветвления сценария использования бота:

1. Неверный или недоступный URL:
   * Если пользователь отправляет недействительный URL или сайт недоступен, бот уведомляет пользователя об ошибке и предлагает повторить попытку с другим URL.
2. Отсутствие RSS/Atom лент на сайте:
   * Если бот не находит новостных лент на указанном сайте, он информирует пользователя об этом и предлагает попробовать другой сайт.
3. Выбор из нескольких лент:
   * Если на сайте найдено несколько новостных лент, бот предоставляет пользователю список найденных лент для выбора. Пользователь может выбрать одну или несколько лент для подписки.
4. Проверка новых записей:
   * Если пользователь хочет изменить частоту проверки новых записей или временно приостановить получение уведомлений, бот позволяет настроить эти параметры через команды в интерфейсе.
5. Управление подписками (подробнее в главе 4):
   * Пользователь может управлять своими подписками, добавлять новые, удалять старые или изменять настройки получения уведомлений. Бот предоставляет интерфейс для управления подписками и просмотра текущего статуса (смотреть главу 4).

**2.3. Создание пользовательского интерфейса**

Интерфейс взаимодействия с Telegram-ботом спроектирован быть минималистичным и интуитивно понятным. Пользователи могут взаимодействовать с ботом через простые команды и кнопки, обеспечивая легкий доступ к основной функциональности. Функционал перечислен в кнопке ‘Menu’ около поля ввода сообщения.

**Команды**

* **/**start (рисунок 1): запускает бота.
* **/**help (рисунок 1): предоставляет справочную информацию о командах и использовании бота.
* /my\_feeds (рисунок 2): отображает список текущих подписок пользователя.
* **/**unsubscribe (рисунок 3): удаляет подписку по ее номеру из списка.
* **/**poll\_start: запускает новостную рассылку.
* /poll\_stop: прекращает новостную рассылку.

**Кнопки**

* Подписаться (рисунок 5): Кнопка для начала процесса подписки на новостную ленту.
* Удалить подписку (рисунок 3): Кнопка для удаления выбранной подписки.

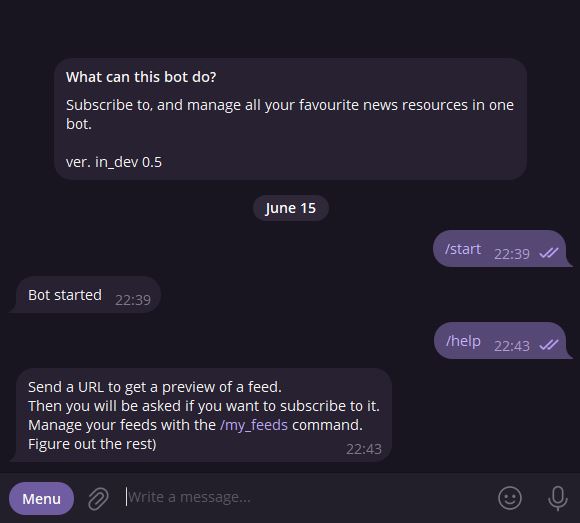


Рисунок 1. Начало работы

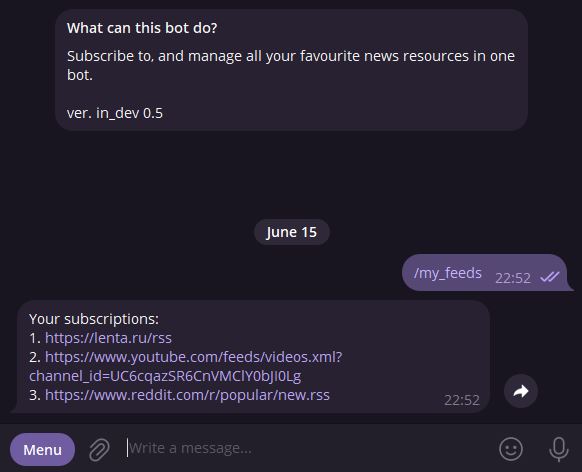


Рисунок 2. Отображение списка подписок

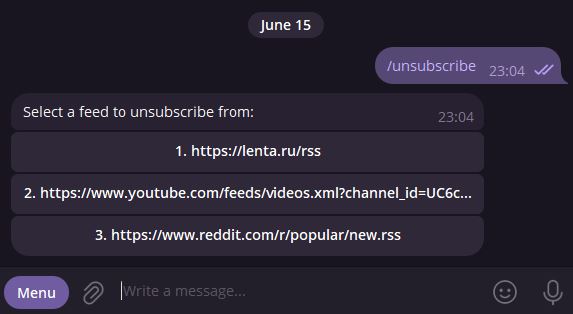


Рисунок 3. Отображение лент к удалению

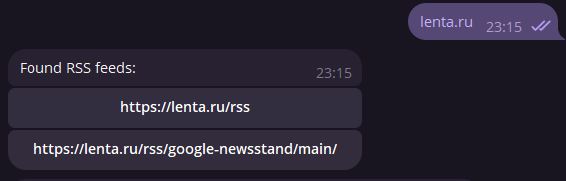


Рисунок 4. Результат поиска RSS-лент на ресурсе 'lenta.ru''

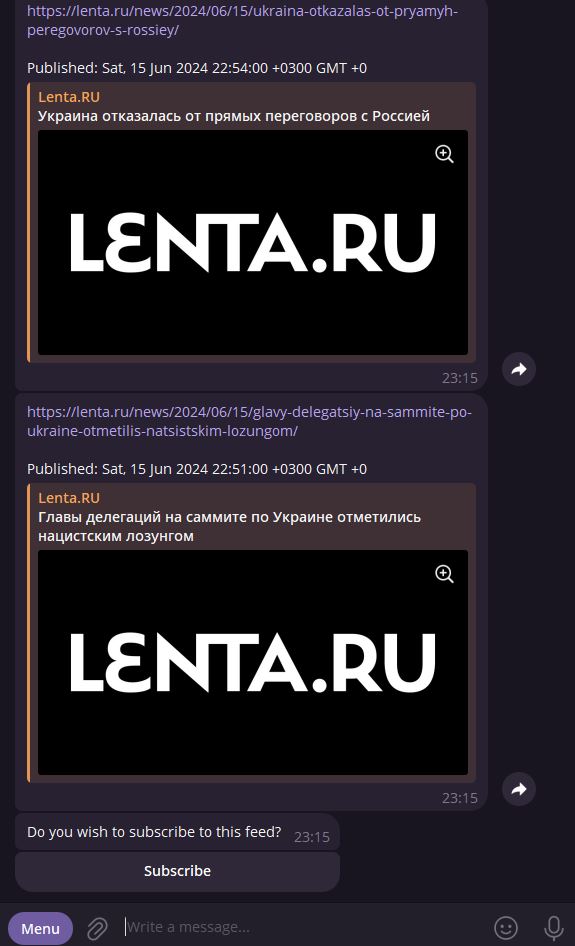


Рисунок 5. Запрос на подписку

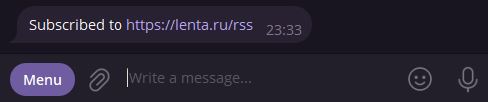


Рисунок 6. Успешная подписка

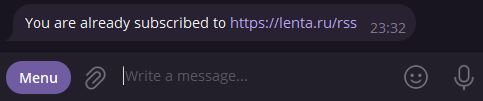


Рисунок 7. Пользователь уже подписан на ленту

Для подписки на ленту достаточно лишь ввести её URL-ссылку в строку ввода сообщения. Бот сначала проанализирует ресурс, отобразит найденные RSS-ленты, выведет кнопки с ссылками найденных лент, и затем будет ожидать выбора пользователя. Далее, бот пришлёт пользователю первые [значение, указанное в файле конфигурации; по умолчанию 5] постов в качестве предпросмотра и спросит, хочет ли пользователь подписаться на данный ресурс.

Следующими идут два возможных сценария:

1. Пользователь успешно подписан на RSS-ленту (рисунок 6).

2. Пользователь уже подписан на RSS-ленту (рисунок 7).

Для непосредственного начала рассылки пользователь может использовать команду /poll\_start. Команда будет проходиться по каждой из пользовательских лент заданными в файле конфигурации и отправлять актуальные данные в виде обычных сообщений. Для прекращения рассылки пользователю достаточно воспользоваться командой /poll\_stop.

Итак, интерфейс бота минималистичен, что делает его удобным для пользователя. Такой подход позволяет пользователям быстро и эффективно получать доступ к новостям и управлять своими подписками, не тратя время на изучение сложного интерфейса.

2.4. Требования к программному продукту

Это техническое задание предназначено для предоставления подробного описания требований и результатов разработки Telegram-бота под названием “GodRSS”.

**Цель проекта:**

Целью приложения является создание Telegram-бота, который будет автоматически отправлять пользователям актуальные новости и обновления из различных источников через RSS и Atom ленты. Бот должен быть простым в использовании и предоставлять пользователям удобный способ подписки на новостные ленты и получения уведомлений о новых записях.

**Описание:**

Telegram-бот будет принимать URL сайтов от пользователей, искать на них RSS и Atom ленты, парсить содержимое этих лент и отправлять пользователям уведомления о новых записях. Бот должен предоставлять минималистичный и интуитивно понятный интерфейс для управления подписками и получения новостей.

**Общий функционал**

* Запуск:
  + Команда /start для запуска бота.
* Подписка на новостные ленты:
  + Ввод пользователем URL-ссылки для подписки на новостные ленты с указанного URL.
  + Бот автоматически находит и предлагает пользователю список доступных лент для подписки.
* Управление подписками:
  + Команда /my\_feeds для отображения списка текущих подписок пользователя.
  + Команда /unsubscribe для удаления подписки
* Получение новостей:
  + Бот регулярно проверяет новостные ленты на наличие новых записей и отправляет пользователю уведомления с основной информацией о новых записях, включая дату публикации в формате GMT+0.
* Справочная информация:
  + Команда /help для предоставления справочной информации о командах и использовании бота.

**Средства разработки:**

Приложение будет создано с использованием языка программирования Python и следующих библиотек:

* + telebot для взаимодействия с Telegram API.
  + requests для загрузки веб-страниц.
  + BeautifulSoup для парсинга HTML-кода и извлечения ссылок на RSS/Atom ленты.
  + feedparser для обработки и анализа RSS/Atom лент.
  + sqlite3 для хранения данных о пользователях и их подписках.
  + dataclasses для структурированного хранения данных.

**Целевая аудитория:**

Целевой аудиторией являются пользователи Telegram, которые хотят получать актуальные новости и обновления из различных источников в удобном формате. Это могут быть:

* Журналисты и редакторы, которые хотят быть в курсе последних новостей.
* Исследователи и аналитики, которые отслеживают информацию по конкретным темам.
* Обычные пользователи, интересующиеся новостями и событиями в мире.

2.5. Оценка безопасности труда

**Выявление опасностей:**

Обеспечение безопасности труда является важным для успеха любой организации, поскольку опасности на рабочем месте могут привести к травмам и заболеваниям. В контексте разработки программных продуктах основные опасности включают напряжение глаз, плохую осанку и длительное сидение. Дополнительно стоит отметить риск синдрома запястного канала из-за продолжительного использования клавиатуры и мыши, а также стресса, вызванного высокими нагрузками и сроками выполнения задач.

**Оценка рисков:**

После выявления опасностей необходимо оценить риски. Это включает оценку вероятности и серьезности каждой опасности, чтобы определить уровень риска. Основные риски в разработке программного продукта включают воздействие компьютерного излучения и эргономические проблемы. Синдром запястного канала может привести к постоянным болям и дискомфорту в руках, что существенно снизит продуктивность. Стресс и выгорание, вызванные продолжительными рабочими часами и высокими требованиями, могут привести к снижению мотивации и повышению уровня ошибок в коде.

**Реализация элементов управления:**

Для минимизации или устранения рисков внедряются средства контроля, которые могут быть административными, инженерными или средствами индивидуальной защиты (СИЗ). Для разработчиков программных продуктов это включает обеспечение эргономичных рабочих мест, регулярные перерывы и использование фильтров синего света для уменьшения нагрузки на глаза. Регулярные проверки безопасности помогают убедиться в эффективности контроля и выявить новые опасности. Для предотвращения синдрома запястного канала рекомендуется использование эргономичных клавиатур и мышей, а также выполнение регулярных упражнений для рук. Чтобы снизить уровень стресса и предотвратить выгорание, полезно внедрять гибкие графики работы и обеспечивать доступ к психологической поддержке.

Оценка безопасности труда в разработке программных продуктов включает выявление опасностей, оценку рисков и внедрение средств контроля. Это помогает обеспечить безопасность и благополучие разработчиков, минимизируя потенциальные риски.

2.6. Выводы по второй главе

В процессе проектирования данного Telegram-бота были выбраны следующие инструменты разработки:

* Python – как основной язык программирования для реализации функционала бота.
* Для взаимодействия с Telegram API использовалась библиотека Telebot.
* Для хранения данных использовалась база данных SQLite, компактная и удобная, где будет храниться информация о пользователях и их подписках.
* Для парсинга HTML-кода и извлечения RSS/Atom лент применялась библиотека BeautifulSoup.
* Библиотека feedparser использовалась для обработки и анализа RSS и Atom лент.

Был проведен анализ требований, в результате которого были сформированы как технические, так и функциональные требования к боту. Получено более четкое представление о взаимодействии компонентов системы, изменяемых данных, а также конечной цели приложения – удобной и эффективной доставки новостей пользователям через Telegram.

**3. Разработка компонентов информационной системы**

**3.1. Развертывание виртуального окружения**

В первую очередь, следует установить последнюю версию Python с официального сайта [python.org](https://www.python.org/downloads/), чтобы иметь все необходимые инструменты для разработки. Затем создать директорию для проекта и перейти в нее:

“mkdir my\_telegram\_bot”

“cd my\_telegram\_bot”

Для комфорта и безопасности разработки следует развернуть виртуальное окружение, используя библиотеку venv. Развертывание виртуального окружения venv помогает изолировать зависимости проекта и избежать конфликтов с глобальными библиотеками Python.

Создать виртуальное окружение можно с помощью следующей команды:

“python -m venv venv”

Эта команда создала папку venv, содержащую копию интерпретатора Python и стандартной библиотеки.

Затем следует активировать виртуальное окружение. Команды активации зависят от операционной системы:

* Windows:

“venv\Scripts\activate”

* macOS и Linux:

“source venv/bin/activate”

После активации виртуального окружения командная строка изменилась, добавив префикс “(venv)”, что подтвердило успешную активацию.

Далее, хорошей идеей будет создать файл зависимостей в корневой директории проекта, в который будут добавлены все необходимые библиотеки с помощью следующей команды:

“python -m pip freeze > requirements.txt”

Создание файла зависимостей предоставляет множество преимуществ при разработке проектов на Python. Он позволяет четко указать, какие библиотеки и какие версии этих библиотек необходимы для работы проекта. Это помогает избежать конфликтов версий и других проблем, связанных с несовместимыми библиотеками. С помощью файла зависимостей другие разработчики или системы, на которых будет разворачиваться проект, могут легко установить все необходимые библиотеки. Достаточно выполнить команду “pip install -r requirements.txt”, чтобы воспроизвести среду разработки. Файл зависимостей фиксирует версии библиотек, что гарантирует стабильность и совместимость проекта. Это особенно важно при обновлении библиотек, так как новые версии могут содержать изменения, нарушающие работу приложения. И последнее, что важно отметить, это то, что файл зависимостей служит документацией проекта, указывая, какие библиотеки используются. Это упрощает процесс анализа и поддержки кода, так как разработчики сразу видят все используемые библиотеки и их версии. Пример файла requirements.txt для проекта Telegram-бота может выглядеть так:

pyTelegramBotAPI==4.4.0

requests==2.26.0

beautifulsoup4==4.10.0

feedparser==6.0.8

Для установки всех необходимых сторонних библиотек для проекта, выполните следующую команду:

“pip install pyTelegramBotAPI requests beautifulsoup4 feedparser”

**3.2. Разработка базы данных**

Как было сказано во второй главе, при разработке данного проекта была применена база данных SQLite. Чтобы ее использовать, не нужно устанавливать никаких сторонних библиотек, ведь эта база данных поставляется вместе со стандартным Python.

В базе данных, по сути, имеется лишь одна таблица – таблица с пользователями (users), обеспечивающая основные операции CRUD (создание, чтение, обновление и удаление) для управления пользователями, их статусами и RSS-лентами.

**Подключение к базе данных**

Чтобы подключиться к базе данных следует выполнить следующие шаги:

1. Импортировать модуль sqlite3:

“import sqlite3”

1. Установить соединение с базой данных (если базы данных с указанным именем не существует, она будет создана автоматически.):

“conn = sqlite3.connect('database\_name.db')”

1. Создать объект курсора:

“cursor = conn.cursor()”

Далее, уже можно выполнять SQL-запросы.

**Функции для реализации**

* Инициализация базы данных:
  + Функция init\_db() создает таблицу users с полем premium\_status, если она еще не существует.
* Получение информации о пользователе:
  + Функция get\_user(user\_id) возвращает информацию о пользователе с указанным user\_id.
* Создание нового пользователя:
  + Функция create\_user(user\_id) добавляет нового пользователя в базу данных с пустым списком RSS-лент.
* Добавление RSS-ленты пользователю:
  + Функция add\_feed\_to\_user(user\_id, rss\_feed) добавляет указанную RSS-ленту в список RSS-лент пользователя.
* Удаление RSS-ленты у пользователя:
  + Функция remove\_feed\_from\_user(user\_id, rss\_feed) удаляет указанную RSS-ленту из списка пользователя.
* Получение списка RSS-лент пользователя:
  + Функция get\_feeds(user\_id) возвращает список RSS-лент для указанного пользователя.
* Получение статуса премиум-подписки пользователя:
  + Функция get\_user\_premium\_status(user\_id) возвращает текущий статус премиум-подписки пользователя.
* Изменение статуса премиум-подписки пользователя:
  + Функция change\_user\_premium\_status(user\_id, status) обновляет статус премиум-подписки пользователя.

Вот пример того, как может выглядеть база данных, хранящая пользователя с идентификатором “1013991164”, обычным (не премиум) статусом и следующим набором RSS-лент, рисунок 8.

* <https://www.youtube.com/feeds/videos.xml?channel_id=UC6cqazSR6CnVMClY0bJI0Lg>
* <https://www.reddit.com/r/popular/new.rss>
* <https://lenta.ru/rss>

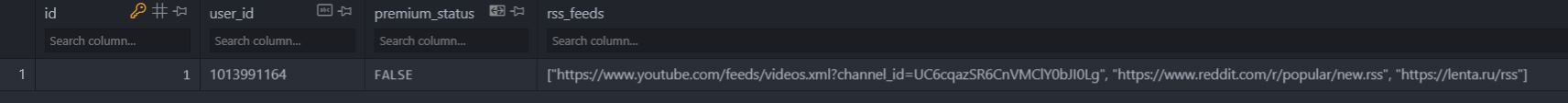


Рисунок 8. Пример записи о пользователе

**3.3. Выводы по третьей главе**

Из главы можно сделать вывод, что она содержит все необходимые шаги для успешной разработки и развертывания Telegram-бота для рассылки новостей с RSS-лент. В первую очередь, было установлено и настроено виртуальное окружение с помощью `venv`, что позволило изолировать проект и управлять зависимостями. Также был описан процесс подключения и взаимодействия с базой данных SQLite.

Полный код прикладывается к пояснительной записке на электронном носителе.

4. Экономическое обоснование

Экономическое обоснование — важная часть любого бизнес-плана или проекта. Оно помогает определить целесообразность и экономическую эффективность проекта, способствуя принятию правильных управленческих решений. В этой главе представлены расчёты трудозатрат на разработку продукта, экономические расчёты его использования, модель продаж и её обоснование, а также описание выбранной монетизации. Эти данные помогут определить потенциальную прибыль и рентабельность проекта, а также принять решение об инвестировании.

4.1. Целевая аудитория

Ранее уже была описана целевая аудитория этого продукта, но сделано это было поверхностно, лишь с целью общего понимания запросов пользователей. Сейчас целевая аудитория будет рассмотрена с экономической точки зрения. Итак, целевой аудиторией являются:

1. Индивидуальные пользователи:

* Технически подкованные пользователи: люди, интересующиеся технологиями и новостями, которые хотят получать обновления из множества источников в одном месте.

2. Образовательные учреждения:

* Университеты и исследовательские институты: учебные заведения, где преподаватели и студенты могут использовать бота для получения научных новостей и статей.

3. Корпоративные клиенты:

* Крупные корпорации: компании, которые могут использовать бота для внутреннего распространения новостей и информации среди сотрудников.

4. Медиа и контент-платформы:

* Новостные сайты и блоги: Платформы, которые могут интегрировать бота для автоматического распространения своих новостей и статей подписчикам.

4.2. Модель монетизации

**Экономическая перспектива**

1. Индивидуальные пользователи

**Модель премиум-подписок для индивидуальных пользователей:**

Премиум-подписки предлагают пользователям дополнительные функции и улучшенный пользовательский опыт. Подписку можно оформить через встроенные платежные системы Telegram или сторонние платформы, такие как Patreon или Boosty.

**Преимущества премиум-подписки:**

* Доступ к дополнительным источникам новостей и RSS-лентам.
* Приоритетная обработка запросов и обновлений (реализована путем установления интервала между запросами к RSS-лентам).
* Возможность настройки частоты уведомлений и фильтрации контента.
* Доступ к аналитическим функциям и статистике по полученным новостям.
* Без рекламы.

**Идеи для работы премиум-подписки:**

Премиум-подписка может включать функции, такие как персонализированные подборки новостей на основе предпочтений пользователя, сохранение статей для оффлайн-просмотра и возможность интеграции с другими сервисами (например, Google Calendar для добавления событий на основе новостей).

2. Образовательные учреждения

**Экономическая перспектива для образовательных учреждений:**

Университеты и исследовательские институты могут подписаться на специализированные тарифные планы, предлагающие доступ к новостям и научным публикациям, полезным для преподавателей и студентов.

**Стоимость подписки:**

Для образовательных учреждений можно предложить корпоративные тарифы с фиксированной ежемесячной или годовой оплатой, например, 50 долларов в месяц для небольших учреждений и 200 долларов в месяц для крупных университетов.

**Преимущества для образовательных учреждений:**

* + Доступ к специализированным источникам и базам данных научных публикаций.
  + Возможность создания групповых подписок для классов или исследовательских групп.
  + Инструменты для анализа и обработки полученных данных.
  + Интеграция с внутренними системами управления контентом.

3. Корпоративные клиенты

**Экономическая перспектива для корпоративных клиентов:**

Крупные корпорации могут использовать бота для распространения корпоративных новостей и анализа информации среди сотрудников. Бот может автоматически доставлять важные новости, обновления и объявления в реальном времени, обеспечивая своевременное информирование всех сотрудников. Это особенно полезно для больших компаний с распределенными командами, где быстрая и эффективная коммуникация имеет решающее значение.

**Стоимость подписки:**

Для корпоративных клиентов можно установить более высокие тарифы, например, от 100 до 500 долларов в месяц, в зависимости от размера компании и объема предоставляемых услуг.

Преимущества для корпоративных клиентов:

* + Расширенные возможности для настройки и интеграции с корпоративными системами.
  + Персонализированные уведомления для разных отделов и сотрудников.

4. Медиа и контент-платформы

**Экономическая перспектива для медиа и контент-платформ:**

Медиа-компании могут использовать бота для автоматического распространения своих новостей и увеличения аудитории.

**Стоимость подписки:**

Для медиа и контент-платформ можно предложить тарифы от 50 до 300 долларов в месяц, в зависимости от объема контента и требуемых функций.

**Преимущества для медиа и контент-платформ:**

* Автоматическое распространение новостей через Telegram-каналы и группы.
* Возможность настройки расписания и частоты публикаций.
* Аналитика по охвату и вовлеченности аудитории.
* Инструменты для взаимодействия с подписчиками и сбора обратной связи.

**Модель премиум подписок**

**Как получить премиум-подписку:**

Пользователи могут оформить премиум-подписку через встроенные платежные системы Telegram или через интеграцию с такими платформами, как Boosty. Процесс подписки будет максимально упрощен, с возможностью оплаты через банковские карты, электронные кошельки или мобильные платежи.

**Стоимость и сравнение с конкурентами:**

Некоторые сервисы, предлагающие премиум-подписки для новостных ботов и агрегаторов контента, включают:

1. Inoreader: платформа для чтения и управления RSS-ленты, предлагает премиум-подписку, которая стоит до ~7 долларов в месяц в зависимости от плана (Basic, Pro, или Custom).

2. NewsBlur: RSS-агрегатор, предлагающий премиум-подписку за 3 доллара в месяц или 36 долларов в год, предоставляющий дополнительные функции, такие как более быстрое обновление лент и доступ к полным статьям.

3. BazQux Reader: RSS-читалка, предлагающая подписку за 30 долларов в год (примерно 2,5 долларов в месяц), предоставляющая доступ к премиум-функциям и поддержке.

Вот ссылки на ресурсы, откуда взяты данные о подписках:

* Inoreader: <https://www.inoreader.com/pricing>
* NewsBlur: <https://newsblur.com/>
* BazQuxReader: <https://bazqux.com/>

Эти сервисы показывают примерные диапазоны цен и функциональных возможностей, которые могут быть включены в премиум-подписки для Telegram-бота, предоставляющего новости с RSS-лент.

Для бота можно установить стоимость премиум-подписки на уровне 5 долларов в месяц, что будет конкурентоспособной и доступной ценой.

**Возможная стоимость премиум-подписки:**

* + Индивидуальные пользователи: 5 долларов в месяц.
  + Образовательные учреждения: от 50 до 200 долларов в месяц.
  + Корпоративные клиенты: от 100 до 500 долларов в месяц.
  + Медиа и контент-платформы: от 50 до 300 долларов в месяц.

**Преимущества премиум-подписки:**

* Расширенные функции и доступ к дополнительным источникам.
* Персонализация уведомлений и фильтрации контента.
* Приоритетная поддержка и обработка запросов.
* Аналитические инструменты и доступ к статистике.
* Интеграция с другими сервисами и системами.

**Идеи для работы премиум подписки**

Премиум-подписка может включать функции, такие как:

* Персонализированные подборки новостей на основе предпочтений пользователя.
* Интеграция с календарями и другими приложениями для создания событий на основе новостей.
* Специальные уведомления о важных событиях и новостях.

Эти функции сделают использование бота более привлекательным и удобным для различных категорий пользователей, что в свою очередь повысит его популярность и экономическую эффективность.

4.3. Выводы по четвертой главе

В ходе анализа была разработана модель монетизации Telegram-бота, рассылающего новости с RSS-лент. Целевая аудитория включает индивидуальных пользователей, образовательные учреждения, корпоративных клиентов и медиа-компании.

Премиум-подписка обеспечивает дополнительные функции, такие как расширенные возможности фильтрации новостей, приоритетную обработку запросов и обновлений и интеграцию с корпоративными системами. Опираясь на примеры аналогичных сервисов, стоимость подписки была определена в диапазоне от 5 долларов в месяц для индивидуальных пользователей и от 500 долларов в месяц для корпоративных клиентов.

Внедрение премиум-подписки позволит создать устойчивый источник дохода, обеспечивая окупаемость затрат на разработку и поддержку бота.

**Заключение**

Одной из значимых задач, с которыми сталкиваются современные люди, является эффективный и удобный способ оставаться осведомленными об интересующих событиях и новостях. Данная работа по созданию Telegram-бота, рассылающего новости с RSS-лент, играет важную роль в этом процессе, предоставляя пользователям оперативный доступ к актуальным новостям и контенту. Бот сможет помочь пользователям быть в курсе последних событий, экономя время на поиск новостей. Выбранная тема актуальна, так как потребность в быстром доступе к информации растет с каждым годом.

В ходе выполнения исследования первой главы — по постановке задачи и анализу предметной области — были рассмотрены анализ аналогичных систем, изучены методы проектирования и разработки ботов, а также проанализированы потребности пользователей. Также было проведено исследование целевой аудитории, что позволило определить наиболее важные функции, которые должен предоставлять бот.

В ходе выполнения исследования второй главы — проектирование программного продукта — был осуществлен выбор технологий и библиотек, включая Python, telebot, SQLite и другие. Произведено логическое прототипирование системы, определены методы взаимодействия с базой данных и разработаны основные функции бота.

В третьей главе — разработки компонентов информационной системы — было описано, как установить и настроить виртуальное окружение, подключить необходимые библиотеки и создать файл зависимостей. Разработана структура базы данных и проработаны функции для взаимодействия с ней. Также описан процесс развертывания и тестирования бота.

В ходе выполнения исследования по четвертой главе — экономического обоснования — была произведена оценка эффективности разработанного программного продукта. Рассмотрены различные модели монетизации, в том числе внедрение премиум-подписки для доступа к расширенным функциям. Проанализирована целевая аудитория, включая индивидуальных пользователей, образовательные учреждения, корпоративных клиентов и медиа-компании.

Целью работы являлась разработка функционального Telegram-бота для рассылки новостей с RSS-лент, который обеспечивает пользователей актуальной информацией и удобными инструментами для управления подписками. Для реализации поставленной цели были выполнены следующие задачи:

* Проведен анализ аналогичных систем и потребностей пользователей.
* Разработан прототип бота и определен стек технологий.
* Создано и настроено виртуальное окружение для разработки.
* Разработаны функции для взаимодействия с базой данных.
* Определены и реализованы ключевые функции бота.
* Проведена оценка экономической эффективности и разработана модель монетизации.

В процессе выполнения работы были рассмотрены основные преимущества созданного бота, исследована область применения, выполнен анализ рынка и конкурентов, показан стек технологий и проведены экономические расчеты.

**Список литературы**

1. [Электронный ресурс] MDN Web Docs.

URL: <https://developer.mozilla.org> (Дата обращения: 10.02.2024)

1. [Электронный ресурс] Python Official Documentation.

URL: <https://docs.python.org> (Дата обращения: 15.02.2024)

1. [Электронный ресурс] SQLite Documentation.

URL: <https://www.sqlite.org/docs.html> (Дата обращения: 20.02.2024)

1. [Электронный ресурс] Feedparser Documentation.

URL: <https://pythonhosted.org/feedparser/> (Дата обращения: 22.02.2024)

1. [Электронный ресурс] Beautiful Soup Documentation.

URL: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> (Дата обращения: 05.0632024)

1. [Электронный ресурс] Telegram Bot API Documentation.

URL: <https://core.telegram.org/bots/api> (Дата обращения: 09.03.2024)

1. [Электронный ресурс] Python Dataclasses Documentation.

URL: <https://docs.python.org/3/library/dataclasses.html> (Дата обращения: 14.03.2024)

1. [Электронный ресурс] Python venv Documentation.

URL: <https://docs.python.org/3/library/venv.html> (Дата обращения: 16.03.2024)

1. [Электронный ресурс] Python Requests Documentation. URL: <https://docs.python-requests.org/en/latest/> (Дата обращения: 10.04.2024)
2. [Электронный ресурс] Stack Overflow. URL: <https://stackoverflow.com/> (Дата обращения: 18.04.2024)
3. [Электронный ресурс] W3Schools. URL: <https://www.w3schools.com/> (Дата обращения: 20.04.2024)
4. [Электронный ресурс] Pip Documentation. URL: <https://pip.pypa.io/en/stable/> (Дата обращения: 20.04.2024)
5. [Электронный ресурс] RSS 2.0 Specification. URL: <https://cyber.harvard.edu/rss/rss.html> (Дата обращения: 25.04.2024)
6. [Электронный ресурс] The Python Requirements File and How to Create it**.** URL: <https://learnpython.com/blog/python-requirements-file/> (Дата обращения: 26.04.2024)
7. [Электронный ресурс] Mastering Advanced OOP Concepts in Python: Advanced Class Features.

URL: <https://medium.com/@ramanbazhanau/mastering-advanced-oop-concepts-in-python-advanced-class-features-a8790fa2b5a2>(Дата обращения: 22.04.2024

1. [Электронный ресурс] Geeksforgeeks (Python Data Structures). URL: <https://www.geeksforgeeks.org/python-data-structures/> (Дата обращения: 25.04.2024)
2. [Электронный ресурс] SQLite Tutorial – Tutorialspoint. URL: <https://www.tutorialspoint.com/sqlite/> (Дата обращения: 27.04.2024)
3. [Электронный ресурс] SQLZoo. URL: <https://sqlzoo.net/> (Дата обращения: 10.05.2024)
4. [Электронный ресурс] PEP 8 – Style Guide for Python Code. URL: <https://peps.python.org/pep-0008/> (Дата обращения: 15.05.2024)
5. [Электронный ресурс] Black 24.4.2 documentation. URL: <https://black.readthedocs.io/en/stable/usage_and_configuration/the_basics.html> (Дата обращения: 20.05.2024)