МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка ПО для преобразования шаблона в формате pdf и текстового файла в документ формата docx»

Обучающийся Н.А. Илларионов _____
(Инициалы Фамилия) (личная подпись)

Руководитель С.В. Митин
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант (ы) к.п.н., доцент, А.В. Егорова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Тема выпускной квалификационной работы - «Разработка ПО для преобразования шаблона в формате pdf и текстового файла в документ формата docx».

Дипломная работа посвящена вопросу о разработке программного обеспечения для автоматизации заполнения шаблона документа в формате pdf данными из текстового файла и формирование заполненного документа в формате docx.

Шаблоном можно считать форматированный текстовый документ с пустыми полями, обычно такие поля обозначены символом подчеркивания, например, справка или титульный лист.

Объектом исследования дипломной работы является формат документа pdf, текст и способы его форматирования с целью сохранения стилей форматирования текста, таблиц.

Предметом дипломной работы является поиск способов и алгоритмов для выделения текстовой информации pdf документа и стилей текста, а также способов автоматического форматирования файла в формате docx.

Целью работы является разработка программного обеспечения, предназначенного для заполнения заданного шаблона и формирование заполненного документа в формате docx.

Выпускная квалификационная работа состоит из введения, 3 глав, заключения и списка литературы, состоящего из 20 источников, включая 46 страниц, 20 рисунков.

В введении указывается актуальность темы, а также, описание объекта и предмета исследования, формулируются цели и задачи, необходимые для решения в данной работе.

Первая глава заключается в описании основных понятий и типовых способов выделения данных из файла формата pdf, выбор титульного листа и текстового файла с данным.

Вторая глава посвящена поиску подходящих библиотек, для разбора pdf документа, формирования файла в формате docx и для создания графического интерфейса.

Третья глава посвящена вопросам выбора среды реализации программного обеспечения, разработке программного обеспечения, реализации проработки структур данных и автоматической подготовки файла данными с заданными ключами полей для заполнения и тестирования, также созданию графического интерфейса.

В заключении описываются основные выводы, которые были сделаны в ходе выполнения выпускной квалификационной работы.

Abstract

The title of the graduation work is «Development of software to convert a pdf template and a text file into a docx document».

This qualification work devoted to developing software to automate the filling of the document template in pdf format by the data from the text file and the formation of the filled document in docx format is investigated.

The template can be considered a formatted text document with empty fields, usually such fields are marked with an underscore symbol, such as a reference or title page.

The object of the qualification work is The pdf document format, text and how to format it in order to preserve text formatting styles, tables.

The subject of the qualification work is finding ways and algorithms to select text information pdf document and text styles, as well as ways to automatically format the file in docx format.

The aim of the work is to develop software designed to fill a given template and the formation of a filled document in docx format.

The graduation work consists of an in introduction, three chapters, a conclusion, a list of 20 references, including 46 pages, 20 figures.

In the introduction indicates the relevance of the topic, as well as a description of the object and subject of the study, formulated goals and objectives needed to address in this work.

The first chapter consists of a description of the basic concepts and typical ways of data extraction from a pdf file, selection of the cover page and the text file with the data.

The second chapter is devoted to finding suitable libraries for parsing a pdf document, generating a docx file and to create graphical interface.

The third chapter is devoted to the choice of environment for software implementation, software development, implementation of the study of data structures and automatic preparation of the data file with the specified keys of

fields for filling and testing, as well as the creation of a graphical interface.

In the conclusion describes the results of the graduation work.

Оглавление

Введение

Документооборот, это неотъемлемая часть рабочего процесса любой организации. Один из пунктов заполнение различных справок или шаблонов. Как правило пользователю предлагается набор инструментов (word, libre office) для создания текстового документа с нуля. Актуально иметь ПО которое автоматически заполняет документ данными из заданного шаблона, это позволит сократить время на заполнение и получение такого документа. Наличие такого ПО позволит внедрить автоматическое заполнение документа в электронной системе, например, из базы данных можно по запросу извлечь необходимые поля и сформировать заполненный документ или справку, либо форму, соответствующую заданному стандарту.

В последнее время все растет востребованность в преобразовании одних фалов в другие. На сегодняшний день формат документов PDF является одним из самых надежных и важным форматов для передачи и распространения информации в сети Интернет и по электронной почте. Однако и им присуще свои недостатки, особенно при изменении и редактировании. Чтобы избежать таких проблем, пользователям может понадобится программное обеспечение, которое способно преобразовать PFD в формат, который легко редактируется, как например DOCX.

Таким образом, актуальность данной темы моей выпускной квалификационной работы объясняется необходимостью разработки программного обеспечения, для преобразования файлов.

Целью данной выпускной квалификационной работы (ВКР) является создание программного обеспечения, способного преобразовать файлы, а именно шаблоны в формате PDF и текстовые фалы, в формат DOCX, на примере титульного листа.

Программное обеспечение извлекает тест и форматирование шаблона PDF, соединяет данные с текстового документ и заполняет их в необходимом формате, преобразовывает полученное в документ формата DOCX.

Для достижения поставленной цели необходимо решить следующие

задачи:

- Рассмотреть типовые способы выделения особенностей из формата PDF;
- Выбрать титульный лист формата PDF и текстовый файл с данными;
- Выбрать средства разработки программы;
- Разработать программное обеспечение;
- Реализовать графический интерфейс.

Практическая значимость работы заключается в создании программного обеспечения.

Глава 1 Основные понятия и типовые способы выделения данных из pdf файла

1.1 Файл формата pdf

PDF (Portable Document Format) - это формат электронного документа, который был разработан фирмой Adobe Systems в 1993 году. Он предназначен для создания документов, которые могут быть просмотрены и распечатаны на различных устройствах без изменения формата и содержания.

Значительное количество современного профессионального печатного оборудования имеет аппаратную поддержку формата PDF, что позволяет производить печать документов в данном формате без использования какого-либо программного обеспечения. Традиционным способом создания PDF-документов является виртуальный принтер, то есть документ как таковой готовится в своей специализированной программе — графической программе или текстовом редакторе, САПР и т. д., а затем экспортируется в формат PDF для распространения в электронном виде, передачи в типографию и т. п. [1].

PDF файлы имеют ряд преимуществ перед другими форматами документов. Они сохраняют оригинальный вид и форматирование документа, что позволяет избежать ошибок при открытии на разных устройствах и программных платформах. Они могут быть созданы из различных программ, таких как MS Word, Excel, PowerPoint и других.

Так же такой формат включает себя и другие понятия, такие как:

- Структура PDF-файла: PDF-файл состоит из объектов, которые могут быть графическими, текстовыми, шрифтами, цветами и т.д;
- Шрифты: PDF-файлы могут содержать встроенные шрифты, которые не зависят от системных шрифтов и могут быть использованы на любом устройстве;
- Редактирование: PDF-файлы могут быть редактированы с

помощью специальных программ, таких как Adobe Acrobat и т.п;

- Защита: PDF-файлы могут быть защищены паролем и ограничены на чтение, печать, копирование и т.д.
- Размер: PDF-файлы могут быть очень компактными, что делает их удобными для передачи по электронной почте или загрузки на веб-сайты;
- Интерактивность: PDF-файлы могут содержать интерактивные элементы, такие как гиперссылки, формы и мультимедиа;
- Конвертация: PDF-файлы могут быть сконвертированы в другие форматы, такие как Word, Excel, HTML и т.д;
- Совместимость: PDF-файлы могут быть открыты на любом устройстве, поддерживающем программу для чтения PDF-файлов.

Файлы формата PDF ограничены не только текстом, но и прекрасно работают с изображениями. Он способен поддерживать изображения, которые могут быть графическими, фотографическими или векторными. Качество таких изображений остается высоким даже после сжатия файла, что позволяет использовать его и в профессионально.

30 сентября 2014 года компания Abbyy Россия сообщила о завершении исследования рынка PDF-документов в России и сценариях работы с PDF-документами (рисунок 1). В опросе приняли участие 500 офисных сотрудников из Москвы, Санкт-Петербурга и ряда городов, работающие в различных отраслях индустрии государственного и частного секторов. Согласно этому документу, в 2014 году респонденты отмечали рост объемов документов в формате PDF. Ещё результаты исследования дали понять, что переход на электронный документооборот произошел далеко не везде: 68% офисных служащих считают, что поток бумажных документов возрос. [2]

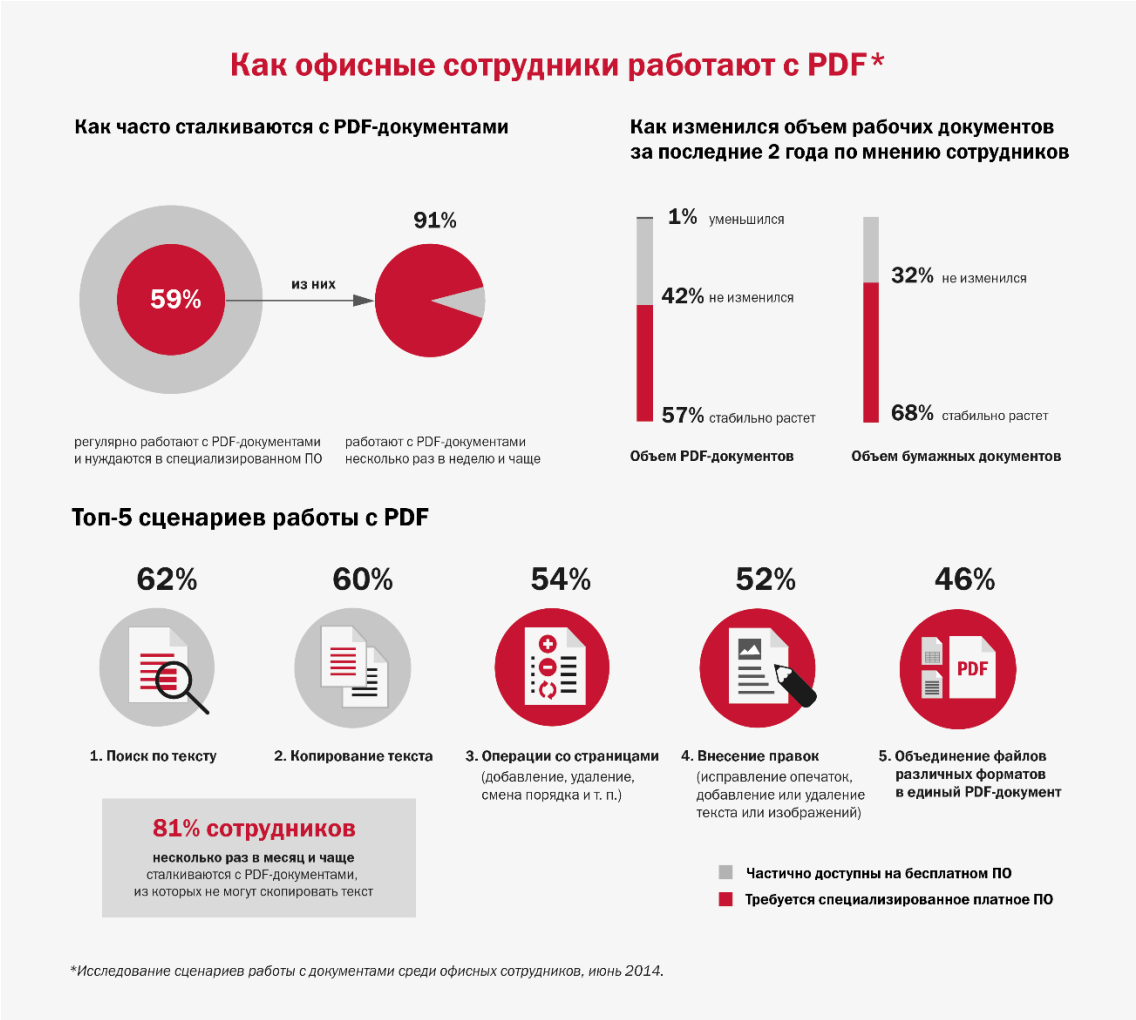


Рисунок 1 – Как используют PDF в России

Так же PDF файл является одним из наиболее распространенных форматов файлов в IT-индустрии. Он широко используется для обмена документами и информацией между различными устройствами и операционными системами.

В целом, PDF-формат является важным инструментом для обмена информацией и документами в IT-индустрии и широко используется в различных отраслях, включая бизнес, образование, науку и технологии.

1.2 Типовые способы выделения данных из PDF файла

Для выделения данных из PDF файла существует множество различных способов. Они начинают от простого копирования текста, до использования специальных программ и плагинов для браузера, которые значительно могут упростить работу с документом.

Разберем способы выделения данных из PDF файла:

- Копирование и вставка текста - это самый простой способ выделения данных из PDF-файла. Однако он может быть неэффективным для документов с большим количеством данных или сложным форматированием;
- Преобразование в текстовый формат - с помощью специальных программ можно преобразовать PDF-файл в текстовый формат, который можно легко редактировать и использовать для дальнейшей обработки данных;
- Использование OCR - оптическое распознавание символов (OCR) позволяет преобразовать отсканированные документы в текстовый формат. Этот метод может быть полезен при работе с документами, которые не были созданы в цифровом формате;
- Использование специализированных инструментов - существуют различные инструменты, которые могут помочь выделить данные из PDF-файла, такие как конвертеры, плагины для браузеров и программы для чтения PDF-файлов;
- Использование библиотек языка Python – с помощью специальных библиотек, можно выделять необходимые данные различными способами, в зависимости от того, какие нам нужны в данный момент.

Выбор способа выделения данных из PDF-файла зависит от конкретной задачи и требований к выделению данных. Например, если необходимо преобразовать PDF-файл в другой формат, то лучше использовать

конвертеры. Если нужно выделить текст из PDF-файла, то лучше использовать программы для чтения PDF-файлов с функцией выделения текста. Если необходимо распознать текст на изображениях, то лучше использовать программы с OCR-технологией.

1.3 Обоснование выбора выделения данных с помощью Python

В рамках изучения материалов, ранее описанных выше и проанализировав их, оптимальным решением будет использовать библиотеки языка Python для решения поставленной задачи.

Это обусловлено тем, что такой выбор дает множество возможностей:

- Универсальность: Python может работать с различными типами PDF-файлов, включая сканированные документы;
- Гибкость: с помощью библиотек Python можно настроить процесс выделения данных из PDF-файла под конкретные нужды пользователя;
- Автоматизация: использование Python позволяет автоматизировать процесс выделения данных из PDF-файла, что экономит время и уменьшает вероятность ошибок;
- Бесплатность: большинство библиотек Python для работы с PDF-файлами доступны бесплатно и имеют открытый исходный код.
- Широкое сообщество: Python является одним из самых популярных языков программирования, что означает наличие большого сообщества разработчиков и обширной документации.

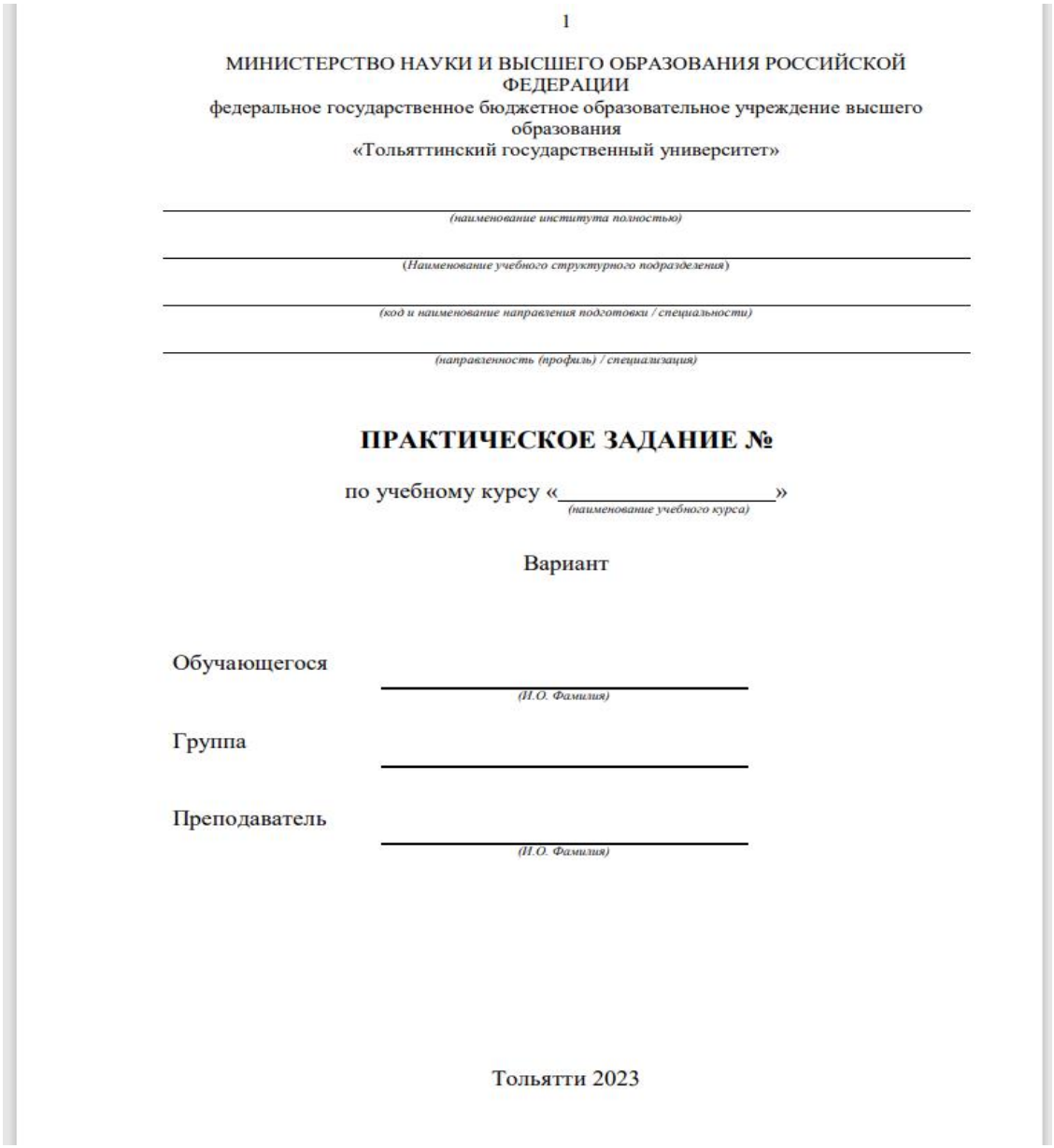
Использование библиотек Python для выделения данных из PDF-файлов является эффективным и удобным способом работы с этим форматом. Благодаря универсальности, гибкости, автоматизации, бесплатности и поддержке широкого сообщества разработчиков, Python позволяет легко и быстро извлекать необходимую информацию из PDF-документов. Это особенно актуально для компаний и организаций, которые

работают с большим объемом документации и нуждаются в автоматизации процесса обработки данных.

1.4 Выбор шаблона в формате PDF и текстового листа с данным для заполнения

Для примера рассмотрим заполнение титульного листа ВКР данными из текстового файла. В общем можно использовать любой шаблон для заполнения данными и формирования заполненного документа.

В качестве шаблона я выбрал лист, который используется в практических работах нашего института.



1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

_____
(наименование института полностью)

_____
(Наименование учебного структурного подразделения)

_____
(код и наименование направления подготовки / специальности)

_____
(направленность (профиль) / специализация)

ПРАКТИЧЕСКОЕ ЗАДАНИЕ №

по учебному курсу «_____»
(наименование учебного курса)

Вариант

Обучающегося _____
(И.О. Фамилия)

Группа _____

Преподаватель _____
(И.О. Фамилия)

Тольятти 2023

Рисунок 2 – Титульный лист в формате PDF

Согласно шаблону я составил текстовый файл с данными для заполнения.

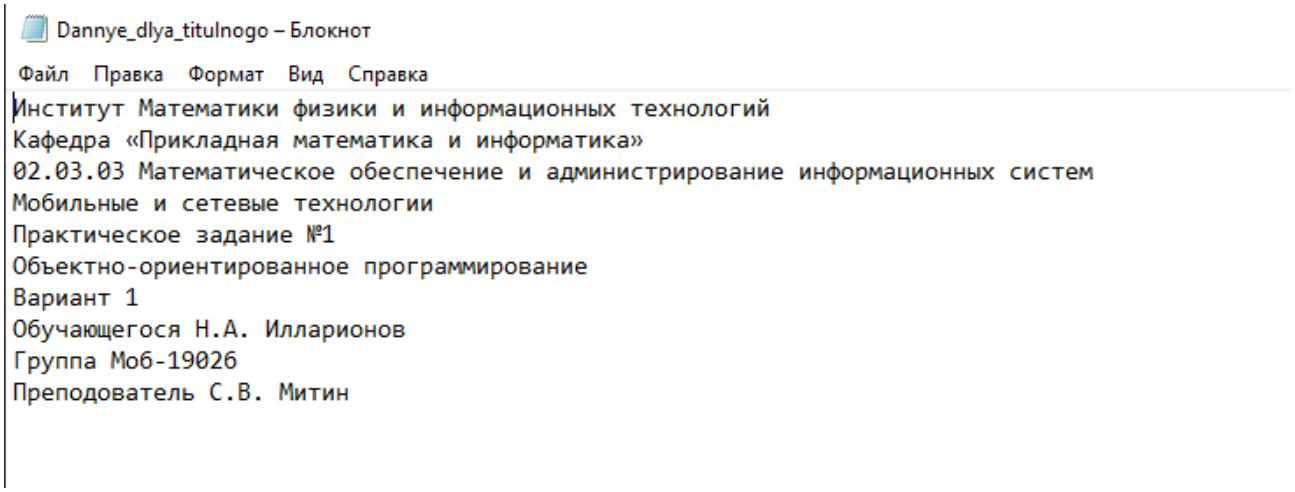


Рисунок 3 – Тестовый файл с данными

На основе этих данных и будет разрабатываться мое программное обеспечение.

Вывод к главе 1

В первой главе выпускной квалификационной работы описывает PDF файл, выбор титульного листа в данном формате и текстового листа с данным для заполнения.

Рассматриваются различные способы и виды выделения данных из файла.

Выбирается и обосновывается вид нужного способа для выделения данных, который послужит основой для дальнейшей работы над ВКР.

Глава 2 Поиск подходящих библиотек, для разбора pdf документа, формирования файла в формате docx и графического интерфейса

2.1 Поиск библиотек языка Python для реализации ПО

Одной из первоначальных задач для реализации программного обеспечения [19] является правильный выбор подходящих библиотек для написания кода нашей программы.

Для правильного поиска библиотек нужно выполнить несколько шагов, такие как:

- Использовать ресурсы, такие как GitHub, браузера Yandex или Google, для поиска рекомендаций и перечня библиотек для выбранного языка программирования;
- Оценить популярность и актуальность выбранных библиотек, а также их документацию и сообщество поддержки;
- Смотреть официальные сайты библиотек, чтобы узнать о их возможностях и ограничениях. Обратить внимание на документацию и примеры кода;
- Проверить, совместимы ли выбранные библиотеки с требованиями и проектом;
- Использовать тестирование и отзывы других разработчиков, чтобы выбрать подходящие библиотеки для вашего проекта.

Поскольку основной функционал моего ПО заключается в работе с файлами формата PDF и Docx и текстовым документом, то мне нужны библиотеки, которые могут работать с ними и преобразовывать. Так же будет необходим поиск библиотеки для реализации графического интерфейса.

Для поиска и установки нужных мне библиотек на языке Python, я буду использовать репозиторий PyPi[3]. Его главная страница расположена по адресу pypi.org (рисунок 4).

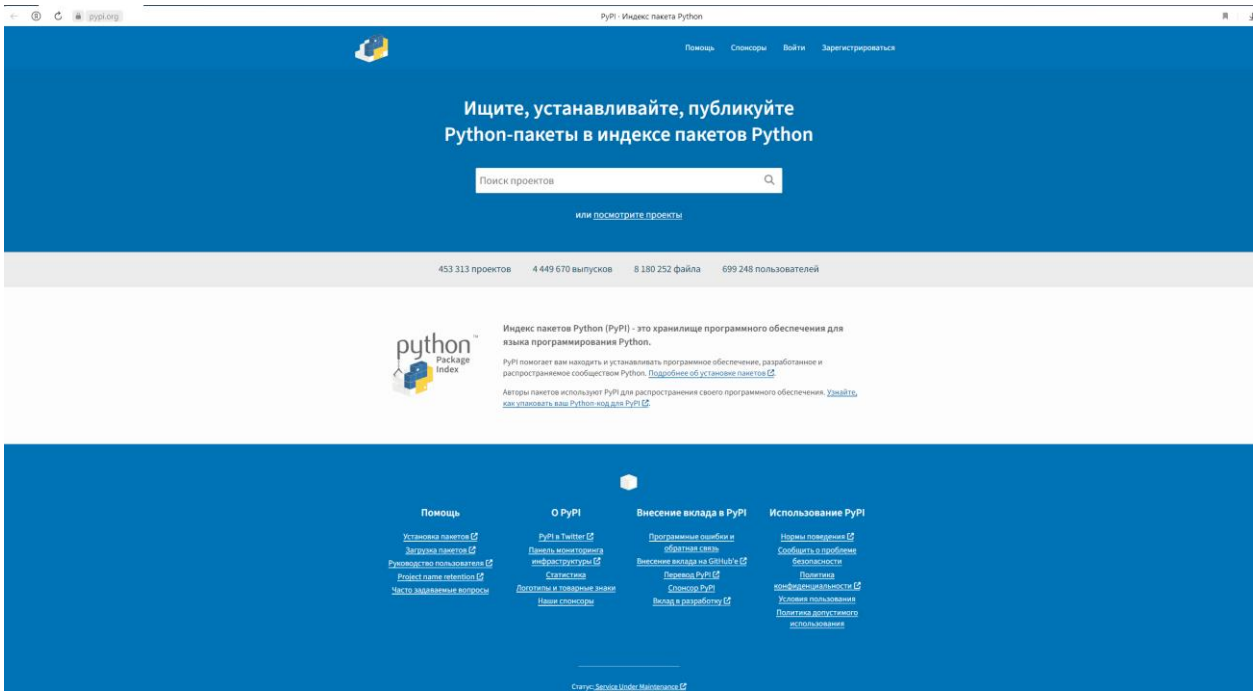


Рисунок 4 – Главная страница сайта PyPi

Данный репозиторий содержащий более 300 тысяч пакетов для Python. Здесь можно найти и скачать различные библиотеки, фреймворки, утилиты и другие инструменты, которые помогут разработчикам создавать приложения на Python. PyPI поддерживает автоматическую установку пакетов с помощью утилиты pip (Python Package Installer), что делает процесс установки и обновления пакетов быстрым и простым. Кроме того, PyPI предоставляет возможность создавать и публиковать свои собственные пакеты, что способствует распространению и повторному использованию кода в сообществе Python.

Учитывая, что данный репозиторий позволяет автоматически устанавливать необходимые библиотеки, то я буду использовать данную функцию.

Для скачивания библиотек из PyPI используется утилита pip, которая входит в стандартную поставку Python. Для скачивания библиотеки необходимо выполнить команду в терминале или командной строке:

```
pip install <имя библиотеки>
```

Например, для скачивания библиотеки requests необходимо выполнить команду:

```
pip install requests
```

После выполнения команды pip загрузит и установит библиотеку requests и ее зависимости. Если библиотека уже установлена, то pip обновит ее до последней версии.

Также можно указать конкретную версию библиотеки, используя опцию --version:

```
pip install <имя библиотеки>==<версия>
```

Например, для установки версии 2.22.0 библиотеки requests необходимо выполнить команду:

```
pip install requests==2.22.0
```

С помощью данного примера и будет производится скачивание и установка библиотек.

2.2 Библиотека для форматирования и извлечения текста из PDF

Чтобы правильно найти библиотеку, которая сможет форматировать и извлекать текст, надо чтобы она соответствовала нескольким критерия.

Критерии для библиотеки включают в себя несколько пунктов, таких как:

- Надежность: библиотека должна быть надежной и стабильной, чтобы не возникало ошибок при извлечении текста из PDF-файлов;
- Качество извлечения: библиотека должна обеспечивать высокое качество извлечения текста из PDF-файлов, сохраняя форматирование, шрифты и стили;
- Поддержка различных языков: библиотека должна поддерживать извлечение текста на различных языках;
- Поддержка различных форматов: библиотека должна поддерживать различные форматы PDF-файлов, включая зашифрованные, сжатые и многостраничные файлы;
- Простота использования: библиотека должна быть простой в использовании и иметь хорошую документацию;
- Скорость работы: библиотека должна работать быстро и эффективно, особенно при обработке больших PDF-файлов;
- Совместимость: библиотека должна быть совместима с различными операционными системами и языками программирования;
- Лицензия: библиотека должна иметь подходящую лицензию, чтобы ее можно было использовать в коммерческих проектах.

Воспользовавшись критериями, описанными выше, мой выбор остановился на библиотеке pdf2docx.

pdf2docx — одна из мощных библиотек Python с открытым исходным кодом, которая позволяет программистам с легкостью создавать и преобразовывать PDF-документы в формат файла Word DOCX. Библиотека очень проста в обращении и имеет простой графический интерфейс, который позволяет пользователям легко получать доступ и использовать различные функции библиотеки.

Библиотека pdf2docx включает в себя различные функции для обработки операций с PDF, таких как доступ к документам PDF, преобразование PDF в файлы других форматов, анализ и повторное создание макета страницы, поддержка полей страницы, извлечение метаинформации, извлечение текста из файлов PDF, анализ и повторная обработка. -создание абзаца, вставка текста в PDF, поддержка стилей списка, анализ и повторное создание изображения, прозрачное изображение, анализ и повторное создание таблицы, объединенные ячейки, таблица с частично скрытыми границами, поддержка вложенных таблиц, анализ страниц с мульти-обработка и многое другое [6].

Основным отличием библиотеки pdf2docx от других подобных инструментов является ее высокая точность конвертации, которая обеспечивается использованием продвинутых алгоритмов распознавания текста и изображений. Это позволяет сохранять максимальное количество информации при конвертации документов из PDF в DOCX.

Кроме того, pdf2docx обладает широким набором настроек, которые позволяют пользователю оптимизировать процесс конвертации под свои потребности. Например, можно выбрать режим конвертации с сохранением изображений или без них, а также настроить параметры распознавания текста.

Еще одним преимуществом pdf2docx является возможность работы с защищенными паролем PDF-файлами. Библиотека позволяет удалять пароль и конвертировать защищенный файл в формат DOCX.

В целом, использование библиотеки pdf2docx может быть выгодным

решением для тех, кто ищет точный и удобный инструмент для конвертации документов из PDF в DOCX.

На рисунке 5 изображена главная страница на сайте Pypi, где содержится информация о версиях Python, которые поддерживает библиотека, рейтинг на GitHub, и описание возможностей данной библиотеки [11].

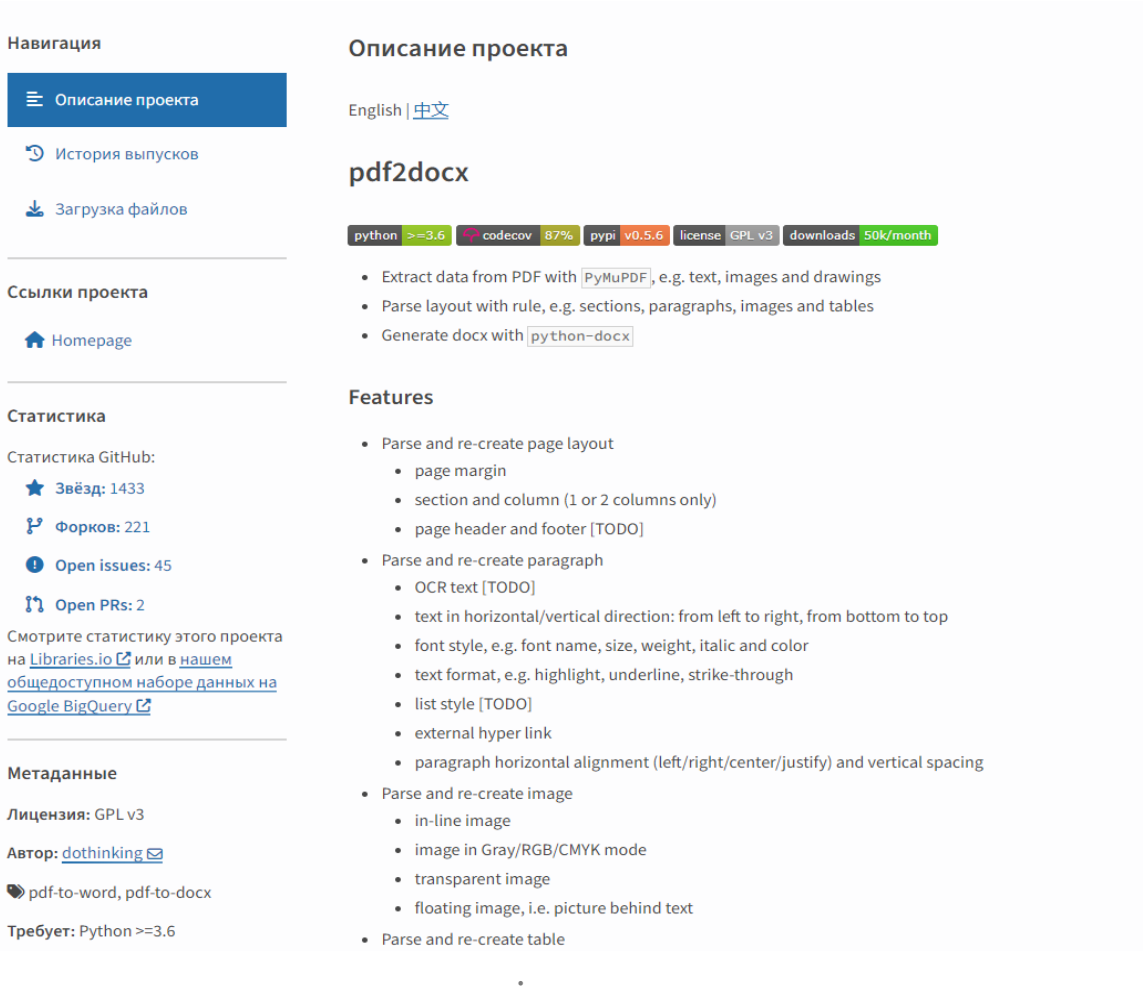


Рисунок 5 – страница библиотеки pdf2docx на сайте PyPi

Основные плюсы и минусы pdf2docx в сравнении с другими похожими библиотеками:

- Библиотека pdf2docx обладает хорошей скоростью конвертации файлов.
- Она может обрабатывать файлы различных форматов, включая PDF, DOCX, RTF и другие.
- Библиотека имеет простой и интуитивно понятный интерфейс, что делает ее использование удобным для пользователей с любым уровнем опыта.
- Pdf2docx поддерживает конвертацию файлов с изображениями и другими графическими элементами.

Минусы:

- Ограничения функционала: pdf2docx имеет ограниченные функции и не может обрабатывать сложные PDF-файлы, содержащие много графики и изображений. Это может привести к потере важной информации и неполадкам в конечном документе.
- Не всегда точность: В некоторых случаях конвертация PDF в DOCX может привести к изменению или потере некоторых элементов документа, что может привести к ошибкам или потере важной информации.
- Нет поддержки других форматов: PDF2DOCX не может конвертировать файлы в другие форматы, кроме DOCX, что ограничивает его функциональность и не позволяет использовать его в различных ситуациях.
- Ограниченный функционал: Библиотека не позволяет редактировать документы в формате DOCX, она только конвертирует их из PDF в этот формат.

Библиотека pdf2docx является удобным и быстрым инструментом для конвертации файлов различных форматов. Ее простой и интуитивно

понятный интерфейс делает ее доступной для пользователей с любым уровнем опыта. В целом, библиотека pdf2docx является полезным инструментом для конвертации файлов, но необходимо учитывать ее ограничения.

2.3Библиотека для работы с DOCX-файлами

Для моего программного обеспечения, мне понадобится библиотека, которая способна создавать сами документы формата docx, но и иметь функции форматирования текста, включая шрифты, размеры и стили.

Поскольку в своей ВКР я использую шаблон титульного листа, то мне понадобится библиотека, которая хорошо с ними функционирует и может точечно их форматировать.

Под функционал, который мне нужно, отлично подходит библиотека python-docx.

Python-DOCX — это библиотека Python с открытым исходным кодом, которая дает разработчикам программного обеспечения возможность работать с Microsoft Word (Docx) в своих собственных приложениях. API может создавать и изменять документы Word с расширением файла docx.

API очень производительный и поддерживает несколько важных функций обработки текстов, таких как открытие документа, добавление абзаца, добавление заголовка, добавление разрыва страницы, добавление таблицы, вставка изображений, применение стиля абзаца, форматирование текста и многое другое [9]. Так же данная библиотека отлично сочетается по функционалу с pdf2docx, поскольку позволяется перекрыть некоторые минусы, такие как невозможность редактирования docx.

На рисунке 6 изображена главная страница на сайте PyPi, где содержится ссылка на документацию, где более подробно описано о библиотеке и ее возможностях и краткое описание самое библиотеки. Как

можно заметить на рисунке 6, у данной библиотеки хороший рейтинг на GitHub и много сделанных проектов с помощью этой библиотеки [12].

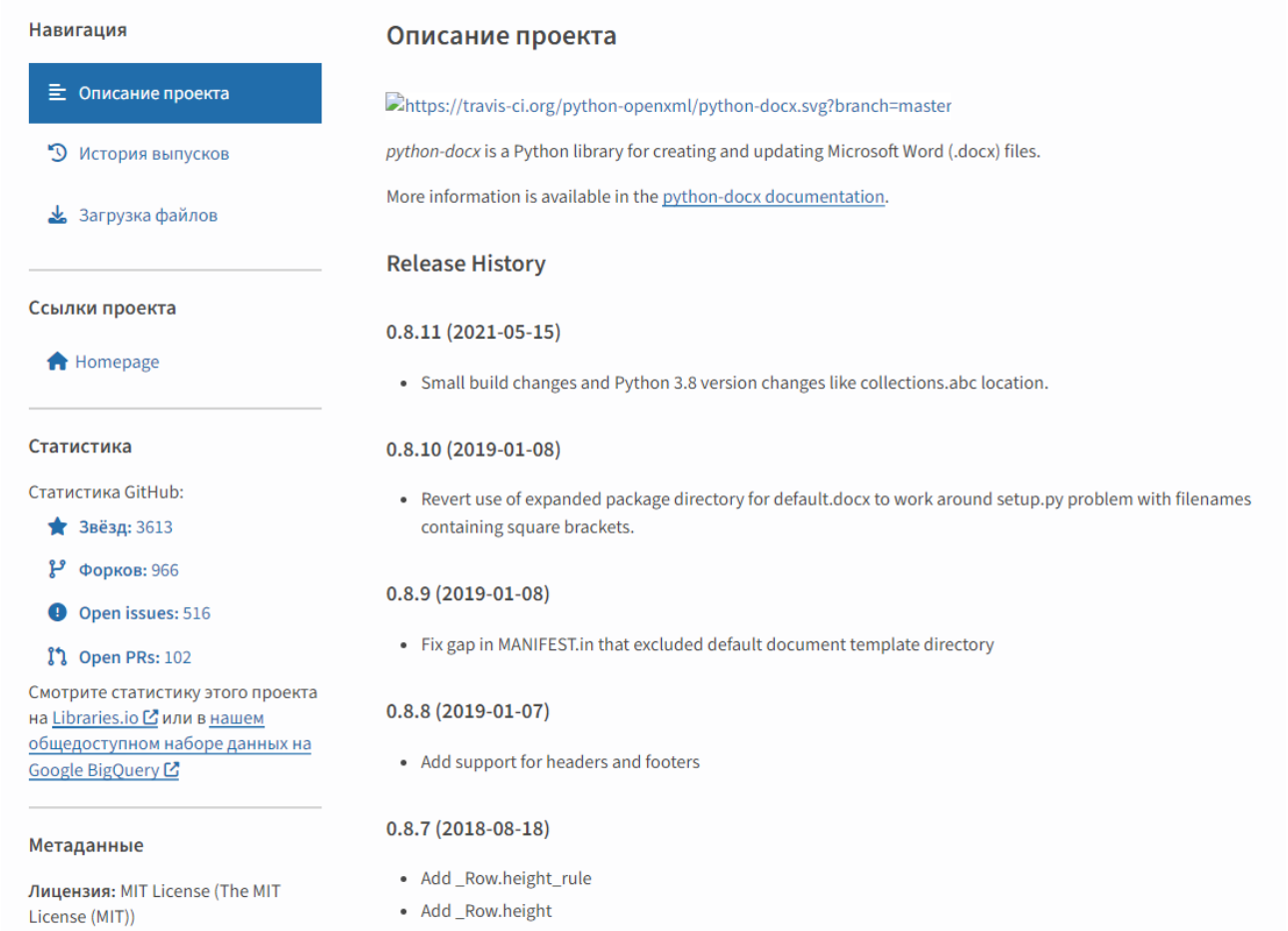


Рисунок 6 – Страница библиотеки python-docx на сайте PyPi

Основные плюсы python-docx:

- Простота использования. Библиотека имеет простой и понятный API, который позволяет легко создавать и редактировать документы;
- Возможность создания и редактирования документов Microsoft Word. Python-docx позволяет создавать и редактировать документы в формате Microsoft Word, что делает его очень удобным для работы с текстовыми документами;

- Поддержка форматирования текста. Библиотека позволяет форматировать текст, добавлять различные стили и настраивать отступы и интервалы между строками;
- Кроссплатформенность. Библиотека может работать на различных операционных системах, включая Windows, Mac и Linux;
- Возможность добавления изображений и таблиц. Python-docx позволяет добавлять изображения и таблицы в документы, что делает его еще более универсальным;
- Открытый исходный код. Python-docx является бесплатной библиотекой с открытым исходным кодом, что позволяет разработчикам адаптировать ее под свои нужды или вносить свои изменения в код.

Так же у данной библиотеки есть и свои минусы, такие как:

- Ограниченная поддержка других форматов документов. Python-docx специализируется на работе с документами Microsoft Word, поэтому его возможности для работы с другими форматами документов ограничены;
- Отсутствие графического интерфейса. Python-docx является библиотекой для программистов и не имеет графического интерфейса, что может быть неудобно для некоторых пользователей;
- Невозможность работы с файлами в старых форматах, такие как .doc.

Файлы с расширением .docx обладают развитой внутренней структурой. В модуле python-docx эта структура представлена тремя различными типами данных. На самом верхнем уровне объект `Document` представляет собой весь документ. Объект `Document` содержит список объектов `Paragraph`, которые представляют собой абзацы документа. Каждый из абзацев содержит список,

состоящий из одного или нескольких объектов `Run`, представляющих собой фрагменты текста с различными стилями форматирования [4]. Пример показан на рисунке 7.

Рисунок 7 – Структура python-docx

Поскольку данная библиотека подходит по своему функционалу для разработки ПО, в своем коде я буду использовать ее.

2.4 Библиотека для создания графического интерфейса

В программном обеспечении так же будет реализовать просто GUI.

GUI (Graphical User Interface) - программная оболочка, которая предоставляет пользователю удобный интерфейс для работы с операционной системой. Она визуализирует многие компоненты в виде графических объектов, например, кнопки, меню, стрелки и т. д. К преимуществам графического интерфейса относится наличие более дружелюбной системы управления ОС по сравнению со стандартной консолью [7].

Для работы с GUI в Python есть несколько библиотек, но я остановился на PySimpleGUI, поскольку мне нужен небольшой и компактный графический интерфейс.

PySimpleGUI - это библиотека для создания графического интерфейса пользователя на языке Python [8]. Она была создана с целью упрощения процесса создания GUI и сделать его доступным для начинающих программистов.

Так же можно на сайте PyPi просмотреть рейтинг данной библиотеки и количество написанных работ с помощью данной библиотеки (рисунок 8) и краткое описание (рисунок 8.1).

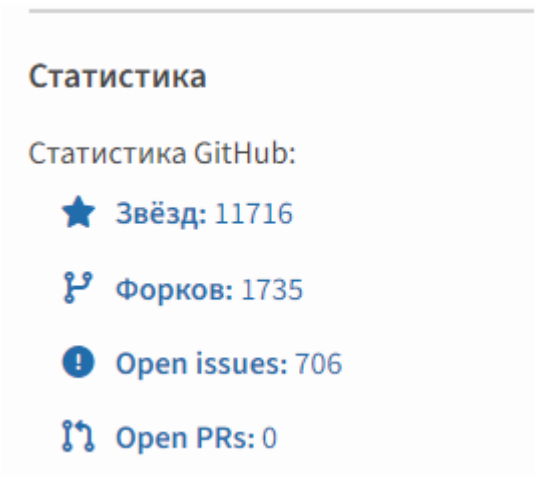


Рисунок 8.1 – Рейтинг библиотеки PySimpleGUI и количество работ с ней

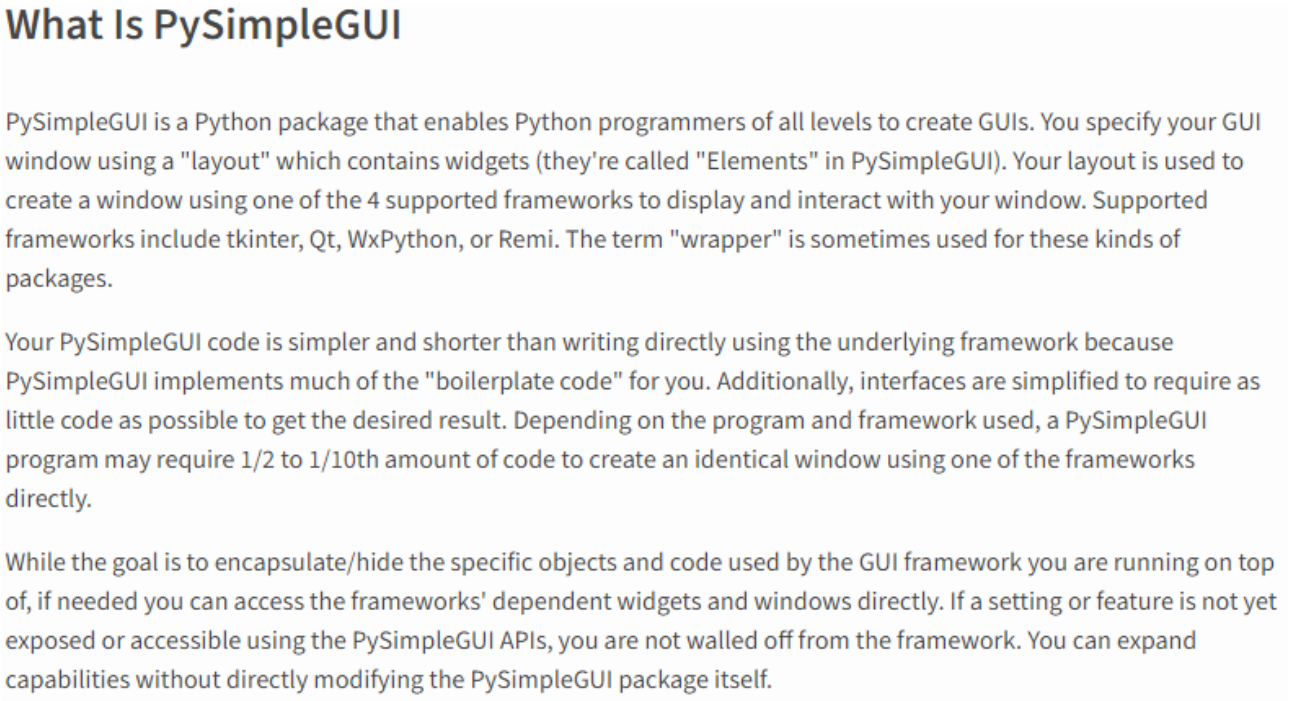


Рисунок 8.2 – Описание библиотеки на PyPi

Особенностью PySimpleGUI является простота использования и возможность создания кросс-платформенных приложений. Библиотека предоставляет простые и интуитивно понятные методы для создания элементов интерфейса, таких как кнопки, текстовые поля, таблицы и многое другое. Пример как выглядит готовый GUI, написанный с помощью PySimpleGUI, показан на рисунке 9.

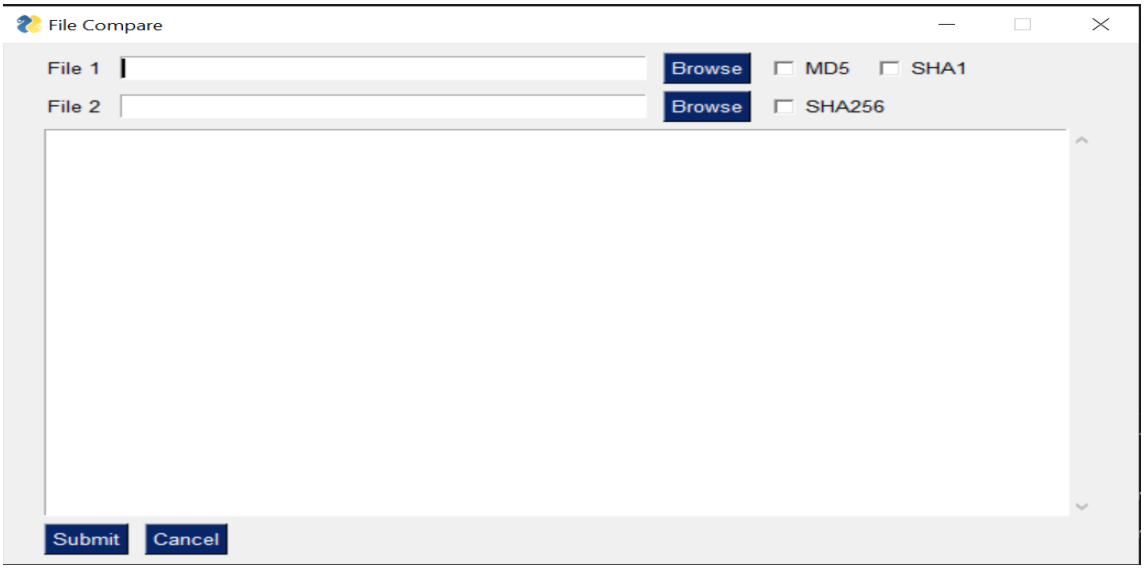


Рисунок 9 – пример GUI библиотеки PySimpleGUI

К плюсам данной библиотеки можно отнести:

- Простота и удобство использования: PySimpleGUI предоставляет простой и интуитивно понятный интерфейс для создания графических интерфейсов;
- Кроссплатформенность: библиотека работает на различных операционных системах, включая Windows, macOS и Linux;
- Гибкость: PySimpleGUI позволяет создавать как простые, так и сложные графические интерфейсы с различными элементами управления;
- Широкий выбор тем оформления: библиотека предоставляет множество тем оформления, которые можно легко применить к своим приложениям.

Так же у данной библиотеки есть свои минусы:

- Ограниченный функционал: PySimpleGUI не предоставляет некоторых продвинутых возможностей, которые могут быть полезны при создании сложных приложений.
- Ограниченный выбор элементов управления: хотя библиотека предоставляет широкий выбор элементов управления, некоторые элементы могут отсутствовать или быть ограничены в функционале.
- Не подходит для создания крупных проектов: PySimpleGUI может быть ограничен в функционале для создания крупных проектов, требующих продвинутых возможностей и инструментов.

Строчка кода для импорта библиотеки выгладит так:

```
import PySimpleGUI as sg
```

Так как, в программном обеспечении будет создан простой GUI, то для этого и будет выбрана данная библиотека, поскольку она проста в использовании и подходит для создания небольших и средних проектов.

Вывод к главе 2

Во второй главе выпускной квалификационной работы описывается принцип поиска основных библиотек при написании кода и их краткое описание.

Рассматриваются плюсы и минусы выбранных библиотек.

Обосновывается выбор данных библиотек.

Глава 3 Разработка программного обеспечение и тестирование

3.1 Выбор средств разработки

Перед началом разработки нужно определится с правильным выбором IDE. IDE (Integrated Development Environment) или «интегрированная среда разработки» представляет готовый комплекс средств, необходимых для разработки создания ПО. Во время работы в IDE программист Python использует широкий набор инструментов, в число которых входят редакторы, библиотеки, платформы для запуска, отладки и тестирования кода. Благодаря средам разработки, программист может не только сэкономить время, но и сделать код более качественным и читаемым [5].

Так же сформулируем основные возможности IDE в виде наглядного списка:

1. Редактор кода с подсветкой синтаксиса и автодополнением;
2. Отладчик для поиска и исправления ошибок в коде;
3. Интеграция с системами контроля версий для управления исходным кодом;
4. Поддержка различных языков программирования и технологий;
5. Возможность создания проектов и управления файлами проекта;
6. Инструменты для автоматизации и ускорения процесса разработки, такие как генераторы кода, шаблоны проектов и т.д.
7. Интеграция с базами данных для управления данными приложения;
8. Встроенные инструменты для тестирования кода;

9. Поддержка различных операционных систем и платформ;

10. Возможность настройки среды разработки в соответствии с требованиями конкретного проекта или разработчика.

В качестве среды разработки я решил выбрать PyCharm [20]. Главная страница сайта изображена на рисунке 10.

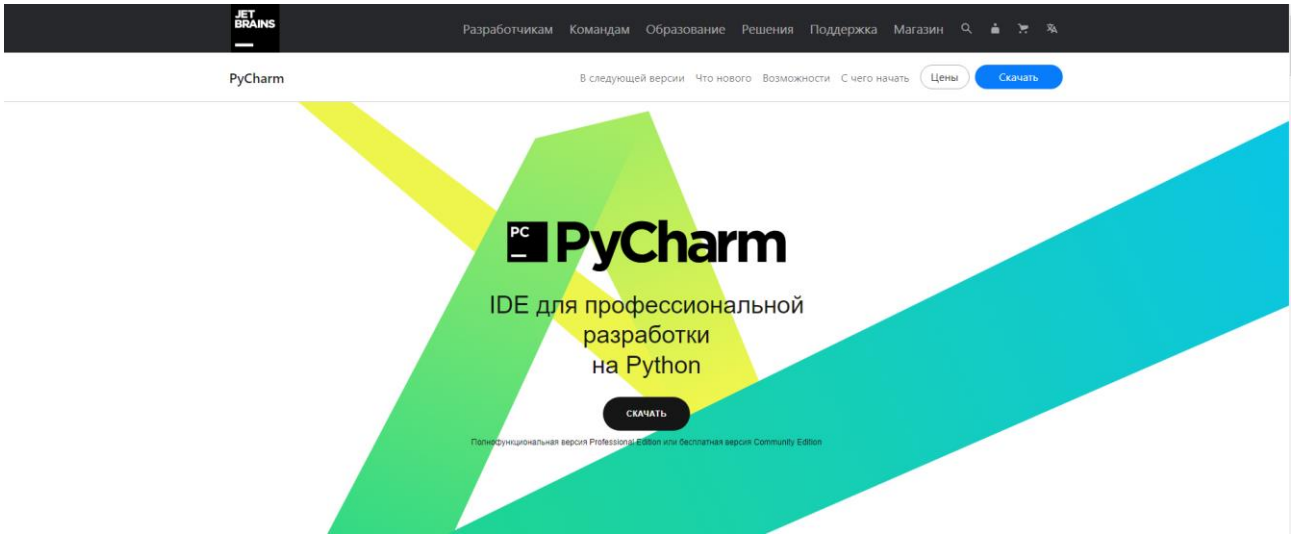


Рисунок 10 – Главная страница PyCharm

PyCharm - это интегрированная среда разработки (IDE) для языка программирования Python. Она разработана компанией JetBrains и предоставляет широкий набор инструментов для разработки, отладки, тестирования и управления проектами на Python. PyCharm является одной из самых популярных IDE для Python и широко используется в сообществе Python-разработчиков.

Плюсы PyCharm:

- Широкий набор инструментов для разработки на Python;
- Автодополнение кода и подсветка синтаксиса;

- Интеграция с системами контроля версий;
- Поддержка виртуальных окружений;
- Отладка кода;
- Поддержка различных фреймворков;
- Доступность в двух версиях: Community Edition и Professional Edition.

Минусы PyCharm:

- Некоторые функции могут быть сложными для новичков;
- Professional Edition платная;
- Не все функции доступны в Community Edition;
- Некоторые пользователи могут считать интерфейс PyCharm слишком громоздким.

Исходя из выше перечисленного мой выбор остановился на IDE PyCharm.

3.2 Функционал программного обеспечения

Разрабатываемая система по преобразованию файлов имеет монолитную архитектуру. Преобразование осуществляется посредством выбора пользователем необходимых фалов и дальнейшей их конвертации.

Основной функциональной частью является конвертация фалов. Настройка работы системы осуществляется посредством графического интерфейса приложения.

Программное обеспечение реализует следующий функционал: графический интерфейс пользователя, конвертация файлов, извлечение определенных элементов из файла, объединение файлов.

Алгоритм работы кода представлен на рисунке 11.

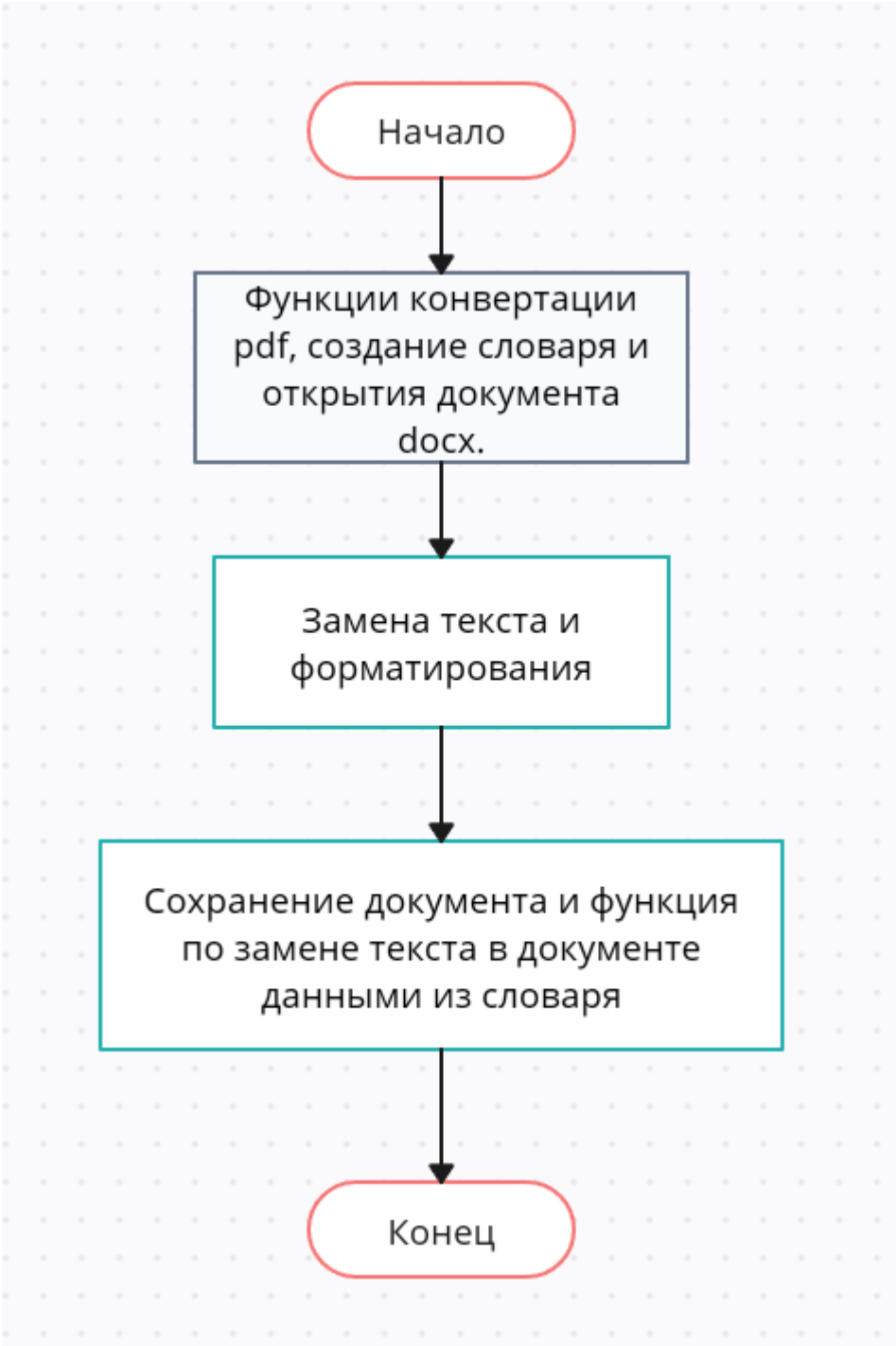


Рисунок 11 – Алгоритм работы кода

Теперь перейдем к разработке программного обеспечения и пользовательского интерфейса.

3.3 Реализация программного обеспечения

Реализуемый код программного обеспечения должен решать задачу по преобразованию pdf и txt документа в docx. Процесс начинается с конвертации шаблона pdf и создания word файла. Далее создается словарь, с помощью которого будет производится замена данных. Затем производится замена текста в файле docx на основной текст, написанный txt файле и добавление форматирования. В конце создается заполненный docx шаблон.

Для реализации поставленной задачи будут использоваться библиотеки, описанные в главах 2.2 и 2.3.

Первоначально код ипмортирует из библиотек необходимые модули [14], [15] для работы с документами в формате docx и pdf, а также установки параметров текста, таких как размер шрифта и выравнивание абзаца в docx (рисунок 12.1).

```
1   from docx import Document
2   from pdf2docx import Converter
3   from docx.shared import Pt, Inches
4   from docx.enum.text import WD_PARAGRAPH_ALIGNMENT, WD_UNDERLINE
```

Рисунок 12.1 – Импорт модулей

Затем используется функции `convert_pdf_to_word` [16], которая принимает два аргумента: `pdf_file` - путь к исходному файлу в формате

PDF, и `output_file` - путь к файлу, в который будет сохранен результат конвертации в формате DOCX. Внутри функции создается экземпляр класса `Converter` из модуля pdf2docx, который принимает путь к исходному файлу. Затем вызывается метод `convert` этого экземпляра, который производит конвертацию исходного файла в формат DOCX и сохраняет результат в файл, указанный в `output_file`. После завершения конвертации вызывается метод `close`, который закрывает экземпляр `Converter` и освобождает ресурсы.

Далее используется функция `replace_text_in_docx`, которая принимает два аргумента: `replacements` и `docx_file`. `Replacements` - это словарь, содержащий пары ключ-значение, где ключ - это строка, которую нужно заменить в документе, а значение - это строка, на которую нужно заменить ключ. `docx_file` - это путь к файлу Microsoft Word в формате .docx, который нужно изменить. Функция использует библиотеку python-docx для открытия документа и создания объекта `Document` [13]. Этот объект представляет собой документ Microsoft Word и содержит всю информацию о форматировании, структуре и содержимом документа (рисунок 12.2).

```
7   def convert_pdf_to_word(pdf_file, output_file):
8       cv = Converter(pdf_file)
9       cv.convert(output_file, start=0, end=None)
10      cv.close()
11
12  def replace_text_in_docx(replacements, docx_file):
13      doc = Document(docx_file)
14
```

Рисунок 12.2 – Функции конвертации pdf, замены текста и открытия документа docx.

Следующим этапом мы используем цикл `for` [17] который выбирает каждый параграф в документе, после он проверяет содержит ли текст параграфа необходимую строку, на примере “наименование института полностью”, если да, то выполняются следующие действия:

- Находим позицию этой строки в тексте параграфа;
- Заменяем эту строку на значение из словаря с ключом;
- Устанавливаем шрифт и размер шрифта;
- Выравниваем параграф и устанавливаем отступ между ним и следующим параграфом.

Эти действия будут проделаны для всех необходимых для замены значений в тексте (Рисунок 12.3).

```
15  # Замена текста в файле DOCX
16  for paragraph in doc.paragraphs:
17      if '(наименование института полностью) ' in paragraph.text:
18          pos = paragraph.text.find('(Наименование института полностью) ')
19          paragraph.text = paragraph.text.replace('(наименование института полностью)', '')
20
21          # Добавление основного текста
22          main_text = f'{replacements["0"]}'
23          run = paragraph.add_run(main_text)
24          run.font.size = Pt(14)
25          run.font.name = 'Times New Roman'
26
27          # Добавление мелкого текста курсивом
28          small_text = '(Наименование института полностью)'
29          run = paragraph.add_run(small_text)
30          run.font.size = Pt(8)
31          run.font.italic = True
32          run.font.name = 'Times New Roman'
33
34          # Выравнивание по центру
35          paragraph.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
36
37          # Удаление отступа
38          paragraph.paragraph_format.right_indent = Inches(0)
39          paragraph.paragraph_format.space_before = Pt(12)
40
```

Рисунок 12.3 – Замена текста и форматирования

Завершающим этапом мы сохраняем наш получившийся файл в формат docx. Далее используем функцию `generate_word_file`, которая принимает два аргумента, `txt_file` и `output_file`.

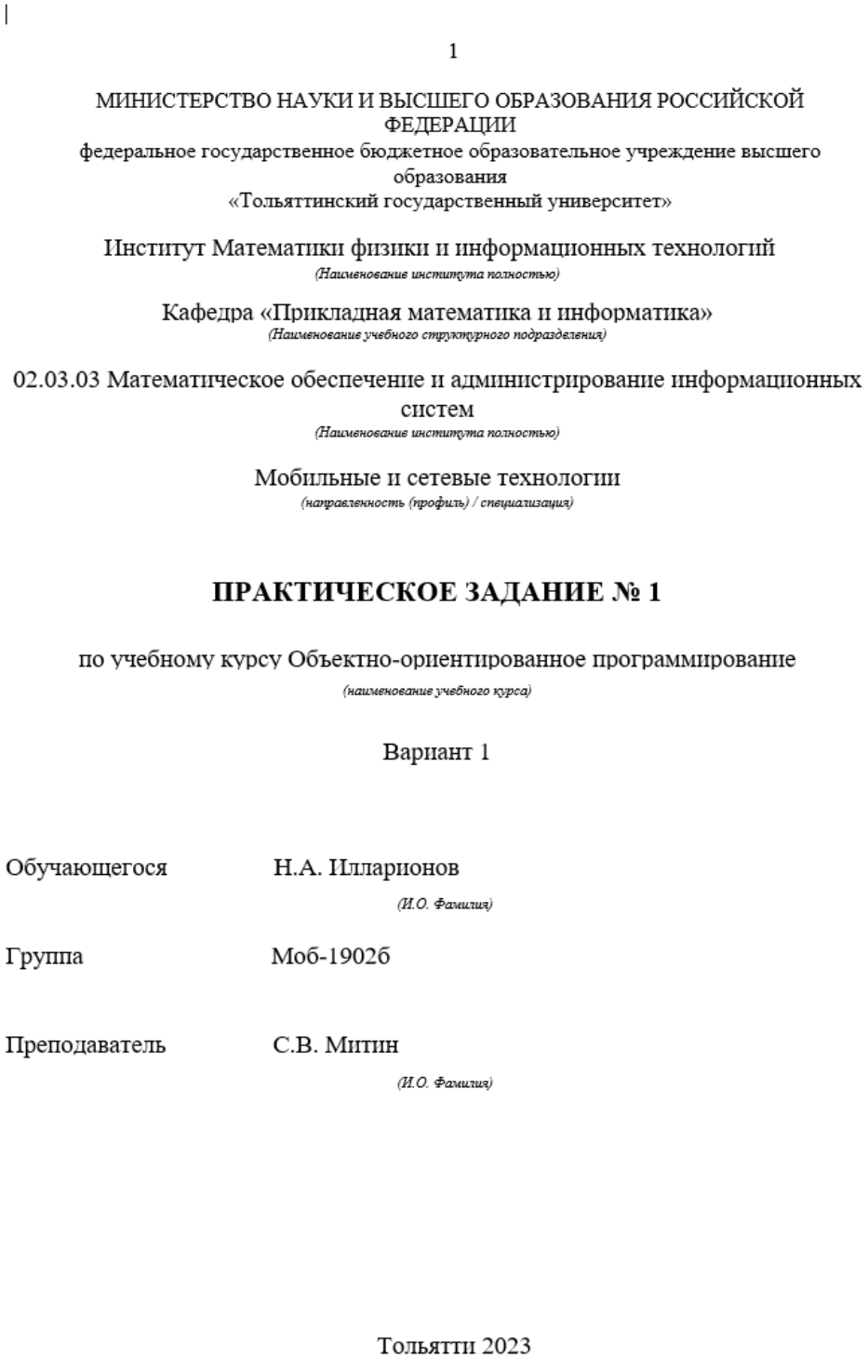
Внутри функции используется оператор `with open` для открытия файла `txt_file` в режиме чтения `r` с использованием кодировки `utf-8`. Затем метод `readlines()` вызывается для чтения всех строк из файла и сохранения их в переменной `lines`. Далее создается пустой словарь `replacements`. В цикле `for` проходится по всем строкам файла `lines`, и каждая строка добавляется в словарь `replacements` с ключом, равным текущему индексу строки `(i)`.

Наконец, вызывается функция `replace_text_in_docx`, которая принимает два аргумента: словарь `replacements` и имя файла `output_file`. Эта функция заменяет все вхождения ключей из словаря на соответствующие значения в файле `output_file` (Рисунок 12.4).

```
216     # # Сохранение результата в новом файле DOCX
217     doc.save('output_file.docx')
218
219 def generate_word_file(txt_file, output_file):
220     with open(txt_file, 'r', encoding='utf-8') as file:
221         lines = file.readlines()
222
223     replacements = {}
224     for i in range(len(lines)):
225         replacements[f'{i}'] = lines[i]
226
227     replace_text_in_docx(replacements, output_file)
228
229
230 convert_pdf_to_word('Titulny.pdf', 'output.docx')
231 generate_word_file('Dannye_dlya_titulnogo.txt', 'output.docx')
```

Рисунок 12.4 – Сохранение документа и функция по замене текста в документе данными из словаря

Результат, который получается после выполнения кода показан на рисунке 12.5.



1

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Институт Математики физики и информационных технологий
(Наименование института полностью)

Кафедра «Прикладная математика и информатика»
(Наименование учебного структурного подразделения)

02.03.03 Математическое обеспечение и администрирование информационных систем
(Наименование института полностью)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ПРАКТИЧЕСКОЕ ЗАДАНИЕ № 1

по учебному курсу Объектно-ориентированное программирование
(наименование учебного курса)

Вариант 1

Обучающегося Н.А. Илларионов
(И.О. Фамилия)

Группа Моб-1902б

Преподаватель С.В. Митин
(И.О. Фамилия)

Тольятти 2023

Рисунок 12.5 - Результат программы

Приведенный код демонстрирует процесс конвертации pdf файла в docx, нахождение и замена ключевых слов с помощью словаря, которые написаны в txt файле и добавление форматирования. Так же можно сделать вывод, что оно работает исправно и полностью выполняет свои задачи.

3.4 Разработка пользовательского интерфейса

Разрабатываемое программное обеспечение обладает графическим интерфейсом. Для его разработки я буду использовать библиотеку, описанную в пункте 2.4.

Созданный графический интерфейс содержит в себе функции такие как:

- Выбор шаблона и файла для заполнения шаблона;
- Выбор папки куда сохранить результат работы и название заполненного шаблона.

Интерфейс изображен на рисунке 13.

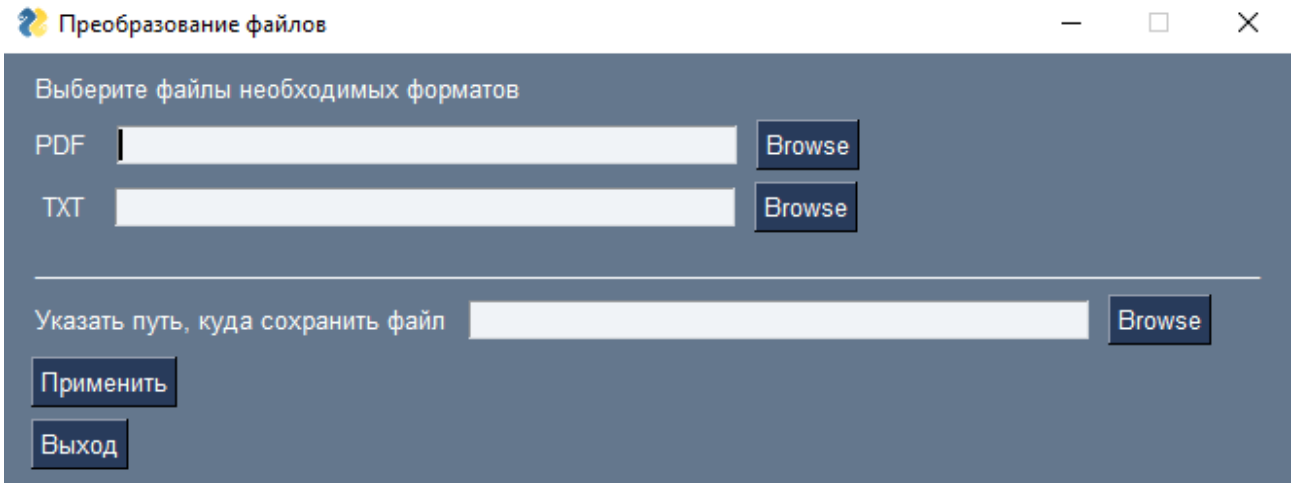


Рисунок 13 – Пользовательский интерфейс.

В интерфейсе доступны следующие пункты меню:

- Выбор из директории компьютера PDF файла
- Выбор из директории компьютера TXT файла
- Выбор куда сохранить файл DOCX в директории компьютера
- Кнопки действий “Применить”, “Выход”

Каждая кнопка в данном интерфейсе отвечает за определенное действие. Кнопка “Применить” выполняет программный код по преобразованию файлов. Кнопка “Выход” закрывает интерфейс.

Так же поскольку программное обеспечение будет в формате .exe файл, то оно будет выглядеть следующим образом. Вид .exe ПО изображено на рисунке 15. Для реализации я использовал библиотеку PyInstaller [18].

На PyPi можно посмотреть рейтинг библиотеки и количество работ с данной библиотекой, а также ссылки на документацию (рисунок 14).

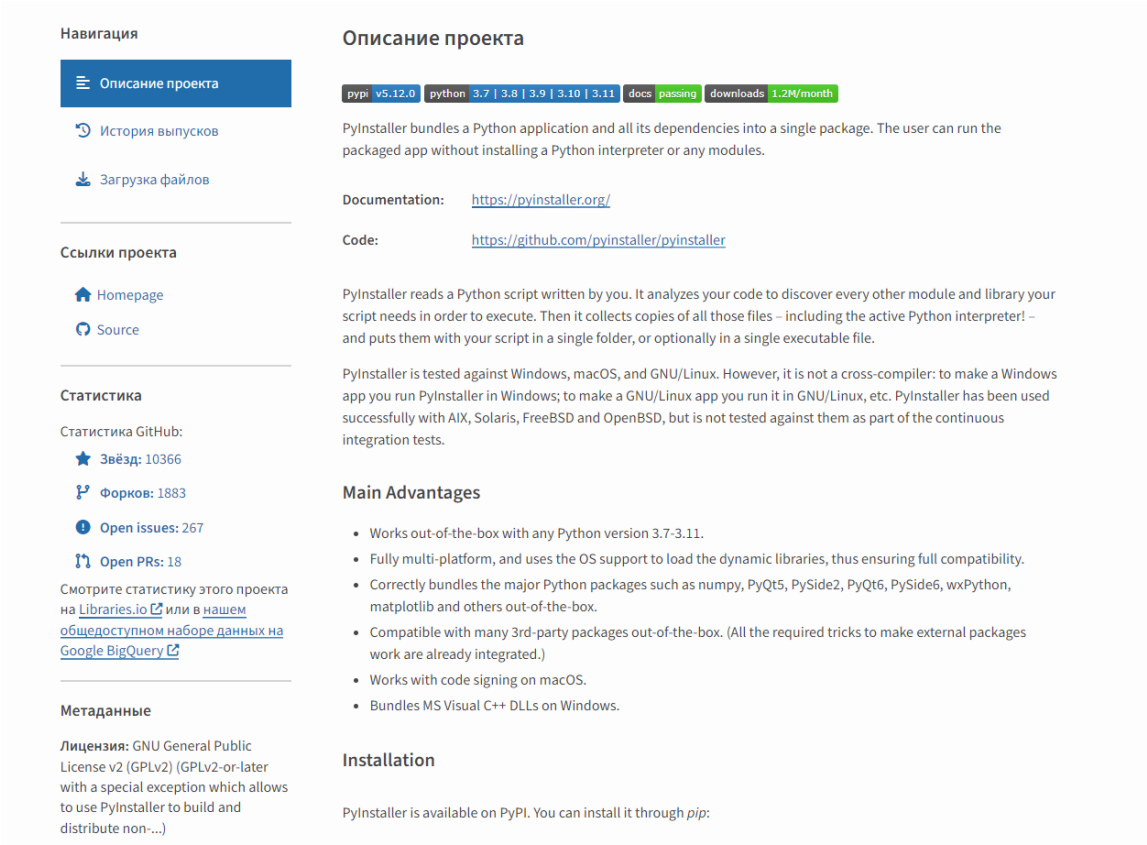


Рисунок 14 - страница библиотеки PyInstaller.

С помощью библиотеки PyInstaller можно собрать все файлы, модули и зависимости в одном сете, привязать к ним интерпретатор Python, а потом обернуть это в один файл. Это значит, что мы получаем как бы виртуальный контейнер, в котором уже есть все, что нужно для запуска скрипта, - без установки на свой компьютер [10].

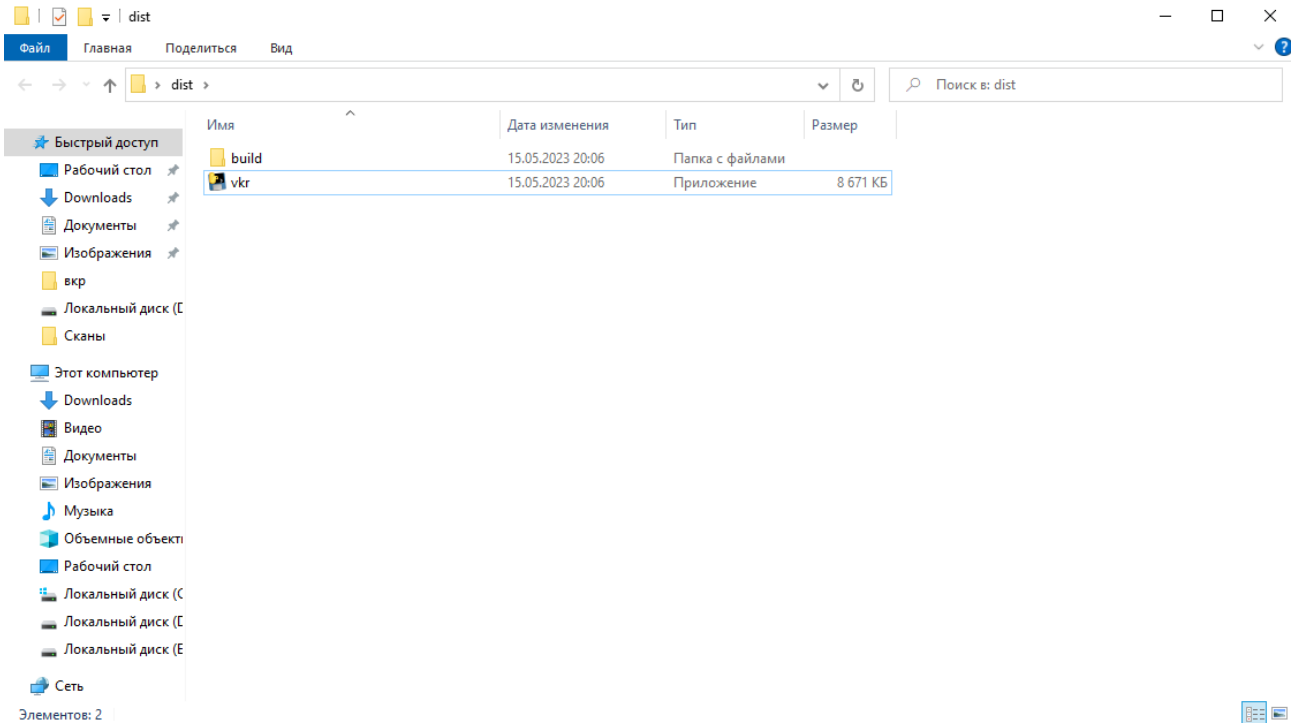


Рисунок 15 – Программное обеспечение в .exe формате

При запуске данного приложения будет открываться наше программное обеспечение.

Вывод к главе 3

В третьей главе выпускной квалификационной работы было разработано программное обеспечение на языке программирования Python.

Был описан пользовательский интерфейс и среда программирования.

Так же был произведен тест программного обеспечения с помощью шаблона титульного листа в формате pdf.

Заключение

Разработанное программное обеспечение позволяет автоматизировать заполнение шаблона в формате pdf данными из текстового файла и сформировать заполненный документ в формате docx.

В результате постановки задачи выполнен обзор предметной области, сформированы требования к программному обеспечению и способ его реализации.

Рассмотрены необходимые библиотеки для написание программного обеспечения, описаны характеристики и возможности, способ их установки.

Произведен выбор среды разработки и описан функционал программного обеспечения. Описан пользовательский интерфейс. Произведен текст программного обеспечения.

Таким образом, все поставленные задачи были успешно выполнены, цель работы – достигнута.

Список используемой литературы и используемых источников

1. Portable Document Format (PDF) [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Portable_Document_Format (дата обращения 25.02.2023)

2. Использование PDF в России [Электронный ресурс]. URL: https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:PDF_(Portable_Document_Format) (дата обращения 25.02.2023)

3. Информация о репозитории PyPi [Электронный ресурс]. URL: https://pypi.org/ (дата обращения 25.04.23)

4. Работа с файлами MS Word в Python [Электронный ресурс]. URL: https://tokmakov.msk.ru/blog/item/78 (дата обращения 25.04.23)

5. Лучшие инструменты Python: IDE и редакторы коды [Электронный ресурс] URL: https://eternalhost.net/blog/razrabotka/python-ide (дата обращения 05.05.2023)

6. Создание и преобразование PDF в Docx через библиотеку Open Source Python [Электронный ресурс] URL: https://products.fileformat.com/ru/pdf/python/pdf2docx/ (дата обращения 25.04.23)

7. Глоссарий основных терминов IT [Электронный ресурс] URL: https://itglobal.com/ru-ru/company/glossary/gui/ (дата обращения 25.04.23)

8. PySimpleGUI [Электронный ресурс] URL: https://pypi.org/project/PySimpleGUI/ (дата обращения 25.04.23)

9. Python API для создания и редактирования документов Microsoft Word [Электронный ресурс] URL: https://products.fileformat.com/ru/word-processing/python/python-docx/ (дата обращения 26.04.23)

10. Превращаем Python-скрипт в исполняемый файл [Электронный ресурс] URL: https://thecode.media/pyinstaller/ (дата обращения 27.04.23)

11. Pdf2docx [Электронный ресурс] URL: https://pypi.org/project/pdf2docx/ (дата обращения 26.04.23)

12. Python-docx [Электронный ресурс] (дата обращения 25.04.23)

13. Объект Document модуля python-docx в Python [Электронный ресурс] URL: https://docs-python.ru/packages/modul-python-docx-python/klass-document/ (дата обращения 29.04.23)

14. Python-docx documentation [Электронный ресурс] URL: https://pypi.org/project/python-docx/ (дата обращения 29.04.23)

15. Pdf2docx documentation [Электронный ресурс] URL: https://dothinking.github.io/pdf2docx/quickstart.convert.html (дата обращения 29.04.23)

16. How to Convert PDF to Docx in Python [Электронный ресурс] URL: https://www.thepythoncode.com/article/convert-pdf-files-to-docx-in-python (дата обращения 29.04.23)

17. Множественное ветвление: if-elif-else. [Электронный ресурс] URL: https://younglinux.info/python/elif (дата обращения 29.04.23)

18. PyInstaller manual [Электронный ресурс] URL: https://pyinstaller.org/en/stable/ (дата обращения 30.04.23)

19. ГОСТ 19781-90. Обеспечение систем обработки информации программное. Термины и определения

20. PyCharm IDE для профессиональной разработки на Python [Электронный ресурс] URL: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения 28.04.23)