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)

01.03.02 Прикладная математика и информатика»
(код и наименование направления подготовки)

Компьютерные технологии и математическое моделирование
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка ПО для анализа и преобразования шаблонов PDF документов в программируемый код

Обучающийся В.О. Москалев
(Инициалы Фамилия) (личная подпись)

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

Тольятти 2023

Аннотация

Название дипломной работы: "Разработка ПО для анализа и преобразования шаблонов PDF документов в программируемый код".

Работа выполнена студентом Тольяттинского государственного университета, Института математики, физики и информационных технологий.

В дипломной работе основное внимание уделяется разработке программного обеспечения, способного анализировать и преобразовывать шаблоны PDF-документов в программируемый код. В работе рассматриваются методы и технологии, необходимые для создания такого программного обеспечения, а также его потенциальные применения.

Ключевым вопросом дипломной работы является разработка программного решения, позволяющего автоматически анализировать структуру и содержание шаблонов PDF-документов, а затем преобразовывать их в программируемый код, который может быть использован для дальнейшей обработки и автоматизации различных задач.

Цель работы заключается в создании эффективного и точного инструмента для анализа и преобразования шаблонов PDF-документов в программируемый код. Разработанное программное обеспечение должно обеспечивать высокую степень автоматизации и точности преобразования, упрощая рабочий процесс с шаблонами и повышая эффективность программистов и разработчиков.

Дипломная работа состоит из следующих логически связанных частей: анализ существующих подходов и инструментов для работы с PDF-документами; разработка алгоритмов для анализа и преобразования шаблонов PDF-документов; создание программного обеспечения на основе разработанных алгоритмов; тестирование и оценка эффективности разработанного программного обеспечения.

В результате исследования представлена дипломная работа,

описывающая разработку программного обеспечения, способного анализировать и преобразовывать шаблоны PDF-документов в программируемый код. Результаты исследования демонстрируют эффективность и точность разработанного программного обеспечения, а также его потенциальное применение в различных областях, где требуется автоматизация работы с шаблонами PDF-документов.

Окончательные выводы подчеркивают актуальность и значимость данной работы в контексте разработки инструментов для упрощения работы с PDF-документами и повышения производительности программистов и разработчиков. Разработанное программное обеспечение может быть использовано для анализа и преобразования шаблонов PDF-документов в различных проектах, включая разработку программного обеспечения и автоматизацию бизнес-процессов.

Abstract

The title of the thesis: "Development of Software for Analysis and Conversion of PDF Template Documents into Programmable Code".

The work was performed by a student of Tolyatti State University, Institute of Mathematics, Physics, and Information Technology.

The thesis focuses on the development of software capable of analyzing and converting PDF template documents into programmable code. The work discusses the methods and technologies required to create such software, as well as its potential applications.

The key question of the thesis is the development of a software solution that can automatically analyze the structure and content of PDF template documents, and then convert them into programmable code that can be used for further processing and automation of various tasks.

The aim of the work is to create an efficient and accurate tool for analyzing and converting PDF template documents into programmable code. The developed software should provide a high degree of automation and accuracy in the conversion process, simplifying the workflow with templates and enhancing the efficiency of programmers and developers.

The thesis consists of the following logically interconnected parts: analysis of existing approaches and tools for working with PDF documents; design and development of algorithms for analyzing and converting PDF template documents; creation of software based on the developed algorithms; testing and evaluation of the effectiveness of the developed software.

As a result of the research, a thesis is presented that describes the development of software capable of analyzing and converting PDF template documents into programmable code. The research results demonstrate the effectiveness and accuracy of the developed software, as well as its potential application in various fields where automation of work with PDF templates is required.

The final conclusions emphasize the relevance and significance of this work in the context of developing tools to simplify work with PDF documents and improve the productivity of programmers and developers. The developed software can be used for the analysis and conversion of PDF template documents in various projects, including software development and business process automation.

Оглавление

Введение

Разработка ПО для анализа и преобразования шаблонов PDF документов в программируемый код в программируемый код является актуальной задачей в области информационных технологий. Шаблоны PDF-документов широко используются в различных отраслях, включая банковское дело, медицину, правительственные учреждения и другие сферы, где требуется обработка большого количества документов. Конвертация PDF-документов в программируемый код может упростить и автоматизировать процесс обработки документов, что позволит существенно сократить затраты времени и ресурсов на их обработку. В данной теме мы рассмотрим основные аспекты разработки программного обеспечения для анализа и преобразования шаблонов PDF-документов в программируемый код, а также возможные применения данной технологии в различных отраслях.

Конечная цель данной дипломной работы - разработать ПО, которое будет автоматически анализировать и преобразовывать шаблоны PDF-документов в программируемый код. Это имеет большое значение для разработчиков, которые работают в области создания веб-сайтов и приложений, так как это позволит им существенно ускорить процесс разработки и избежать рутинной работы по ручному вводу кода.

В первой главе будет рассмотрено описание формата PDF, его структура и основные компоненты. Во второй главе будут рассмотрены методы анализа и преобразования PDF-документов в программируемый код, и будет произведено сравнение и выбор оптимальных методов. В третьей главе будут выбраны инструменты и технологии для разработки ПО, и будет проведена разработка самого ПО.

Задачи:

- Изучение формата PDF-документов и структуры их содержимого;
- Анализ требований к ПО для анализа и преобразования шаблона PDF-

документа в программируемый код;

- Изучение существующих методов и алгоритмов для обработки PDF-документов и их преобразования в программируемый код;
- Разработка алгоритмов и кода для анализа и преобразования шаблона PDF-документа в программируемый код;
- Тестирование и отладка ПО на различных примерах PDF-документов;
- Сравнительный анализ полученного ПО с существующими аналогами на основе критериев производительности, точности и удобства использования;
- Описание полученных результатов в рамках дипломной работы;

Глава 1 Основы формата PDF

1.1 Введение в формат PDF

Формат PDF (Portable Document Format) - это широко распространенный формат электронных документов, который был разработан фирмой Adobe Systems в 1993 году. Он предназначен для представления и обмена документами независимо от программного и аппаратного обеспечения, операционной системы и устройства.

PDF-формат обеспечивает сохранность и точное отображение содержимого документа, включая текст, изображения, графику, таблицы, шрифты и другие элементы. Он позволяет создавать документы с фиксированным форматированием, которые выглядят одинаково на всех устройствах и печатаются без потери качества. Это делает PDF идеальным для распространения и обмена электронными документами, такими как отчеты, инструкции, брошюры, контракты и многое другое.

Основные преимущества формата PDF включают кросс-платформенность, что означает, что документы могут быть открыты и просмотрены на различных операционных системах, включая Windows, macOS, Linux и мобильные устройства; сохранение оригинального форматирования, что позволяет точно воспроизводить документы, созданные в различных программах; возможность добавления интерактивных элементов, таких как гиперссылки, закладки, формы и мультимедиа; поддержка защиты документов с помощью паролей, шифрования и цифровых подписей для обеспечения безопасности информации.

Одним из ключевых элементов формата PDF является его структура. PDF-документ состоит из набора объектов, каждый из которых может представлять текст, изображение, графику, шрифт, аннотацию и другие элементы. Эти объекты организованы в структуру, которая описывает их

взаимодействие и расположение на странице. Такая структура позволяет программам, работающим с PDF, правильно интерпретировать и отображать содержимое документа.

Существуют различные инструменты и библиотеки, которые позволяют работать с форматом PDF. Одним из наиболее популярных инструментов является Adobe Acrobat, который предоставляет множество функций для создания, редактирования, просмотра и взаимодействия с PDF-документами. Кроме того, существуют такие библиотеки, как PyPDF2, PDFMiner [6], PyMuPDF и pdfplumber для работы с PDF в различных языках программирования, включая Python.

Формат PDF является универсальным стандартом для представления электронных документов. Он обеспечивает сохранность и точность отображения содержимого документа, а также позволяет добавлять интерактивные элементы и обеспечивать безопасность информации. С развитием технологий и доступностью соответствующих инструментов, работа с PDF-документами становится все более удобной и эффективной для широкого круга пользователей.

1.2 Описание структуры PDF-документа

Структура PDF-документа организована в виде коллекции объектов. Каждый объект в PDF-документе имеет свой уникальный номер, называемый Object Number, который идентифицирует его внутри документа. Объекты могут быть простыми или сложными, в зависимости от их структуры и содержимого.

Простые объекты включают базовые типы данных, такие как числа, строки, булевы значения и некоторые специальные типы данных, такие как нулевой объект. Например, число может представлять координаты или размеры объекта на странице, а строка может содержать текст или имена

ресурсов.

Сложные объекты включают словари, массивы и потоки. Словари являются наборами ключ-значение, где каждый ключ является именем свойства, а значение может быть любым объектом. Словари широко используются для представления различных свойств и атрибутов объектов в PDF-документе, таких как параметры шрифтов или аннотаций.

Массивы являются упорядоченными списками объектов, которые могут содержать объекты разных типов. Массивы используются, например, для представления последовательности страниц в документе или списка элементов формы.

Потоки представляют собой последовательность байтов, которые могут содержать различные данные, такие как изображения, шрифты или сжатый текст. Потоки часто используются для хранения и передачи больших объемов данных в PDF-документе.

Объекты в PDF-документе могут ссылаться друг на друга, используя их Object Number. Например, словарь может содержать ссылки на другие объекты, указывая их Object Number в своих значениях. Это позволяет объектам взаимодействовать и связываться друг с другом, образуя сложную иерархию структуры документа.

Структура объектов в PDF-документе обеспечивает гибкость и масштабируемость при создании и представлении различных типов контента. Каждый объект имеет свою уникальность и явно определенные свойства, что позволяет программам-чтениям и интерпретаторам правильно обрабатывать и визуализировать содержимое PDF-документа.PDF-документы состоят из следующих компонентов:

- Заголовок документа (Header);
- Тело документа (Body);
- Таблица перекрестных значений (Cross-Reference Table);
- Таблица объектов (Object Table).

1.3 Основные компоненты PDF-документа

Заголовок документа (Header) является одной из частей PDF-документа и содержит информацию о его версии и метаданные. Вот более подробная информация о заголовке документа:

Номер версии PDF: Заголовок документа содержит информацию о версии PDF, использованной для создания документа. Это может быть версия 1.0, 1.3, 1.4, 1.5 и т.д. Версия PDF определяет набор функциональных возможностей, поддерживаемых в документе.

Поддерживаемые функциональные возможности: Заголовок документа также содержит информацию о функциональных возможностях, поддерживаемых в документе. Например, это может включать поддержку встроенных шрифтов, цветов, гиперссылок, защиты паролем и других функций PDF.

Метаданные: Заголовок документа содержит различные метаданные, связанные с документом. Некоторые из наиболее распространенных метаданных включают в себя:

Название документа: Заголовок может содержать название документа, которое обычно отображается в заголовке окна или в списке документов.

Автор: Информация об авторе документа, т.е. человеке или организации, ответственном за его создание.

Дата создания: Дата, когда документ был создан или сконвертирован в формат PDF.

Ключевые слова: Список ключевых слов или фраз, которые помогают описать содержимое документа для поиска и индексации.

Описание: Краткое описание содержимого документа.

Язык: Информация о языке, на котором написан документ.

Программа, используемая для создания документа: Имя программы или приложения, которое использовалось для создания документа или его конвертации в формат PDF.

Эти метаданные могут быть полезными для идентификации, классификации и организации документов, а также для облегчения их поиска и управления. [18]

Заголовок документа является важной частью PDF-файла и обычно находится в начале файла перед содержимым документа. Он может быть прочитан и использован программами для обработки и отображения PDF-документов, а также при поиске и анализе информации внутри документа.

Тело документа (Body) является основной частью PDF-документа и содержит все объекты, составляющие его содержимое. Вот более подробная информация о теле документа:

Объекты: Тело документа состоит из множества объектов. Каждый объект представляет собой отдельный элемент или компонент документа. Эти объекты могут быть представлены в форме ASCII-текста, бинарных данных или сжатых данных.

Текстовые блоки: Текстовые блоки являются одним из наиболее распространенных типов объектов в теле документа. Они содержат текст, который отображается на странице документа. Текст может быть представлен в различных шрифтах, размерах и стилях.

Изображения: Тело документа может содержать изображения, которые встраиваются в страницы документа. Изображения могут быть представлены в различных форматах, таких как JPEG, PNG или TIFF, и могут использоваться для вставки фотографий, иллюстраций, диаграмм и других визуальных элементов.

Шрифты: PDF-документ может содержать встроенные шрифты, которые используются для отображения текста на страницах. Шрифты могут быть описаны в теле документа, чтобы обеспечить правильное отображение текста

с заданными стилями и вариантами шрифта.

Аннотации: Тело документа может содержать аннотации, такие как заметки, комментарии или всплывающие подсказки, которые могут быть связаны с определенными областями текста или изображений на странице. Аннотации позволяют добавлять дополнительную информацию или комментарии к содержимому документа.

Ссылки: PDF-документ может содержать ссылки, которые позволяют пользователю переходить к другим разделам документа, внешним ресурсам или выполнять действия, такие как отправка электронной почты или запуск веб-страницы. Ссылки могут быть представлены в виде гиперссылок, кнопок или других элементов интерактивности.

Медиафайлы: Тело документа может содержать встроенные медиафайлы, такие как звуковые или видеофайлы. Эти медиафайлы могут быть воспроизведены или отображены внутри PDF-документа при использовании соответствующих программных средств.

Структурирование информации: Тело документа также может содержать информацию о структуре документа, такую как заголовки, разделы, списки и т.д. Эта информация может быть использована для навигации по документу, создания оглавлений или автоматического извлечения текста.

Тело документа является основой для представления и структурирования информации в PDF-документе. Объекты в теле документа определяют содержимое и внешний вид документа, позволяют вставлять различные типы данных и обеспечивают интерактивность и мультимедийные возможности.

Таблица перекрестных значений (Cross-Reference Table), также известная как XRef, является важной структурой данных в формате PDF. Она содержит список всех объектов, используемых в PDF-документе, и информацию о их позициях в файле. Вот более подробная информация о таблице перекрестных значений:

Список объектов: Таблица перекрестных значений содержит список всех объектов, которые используются в PDF-документе. Каждый объект имеет свой уникальный идентификатор (Object ID), который позволяет обращаться к нему из других частей документа.

Позиции объектов: Для каждого объекта в таблице перекрестных значений указывается его позиция в файле PDF. Это позволяет быстро находить и обращаться к объектам, зная их идентификаторы и позиции в файле. Обычно позиция объекта определяется смещением (offset) от начала файла или относительно другого объекта.

Тип объекта: Кроме позиции в файле, таблица перекрестных значений также содержит информацию о типе каждого объекта. Тип определяет, какие данные содержит объект и как они должны быть интерпретированы. Некоторые из типов объектов включают словари, массивы, строки, числа, булевые значения и другие.

Статус объектов: Каждый объект в таблице перекрестных значений может иметь один из трех статусов: in-use (используется), free (свободен) или compressed (сжат). Объекты со статусом in-use являются активными объектами, используемыми в документе. Объекты со статусом free являются удаленными объектами и могут быть повторно использованы для хранения новых данных. Объекты со статусом compressed являются сжатыми объектами и содержат сжатые данные.

Кросс-ссылки: Таблица перекрестных значений также содержит кросс-ссылки, которые позволяют быстро переходить от одного объекта к другому. Кросс-ссылки определяют связи между объектами на основе их идентификаторов, позволяя эффективно обрабатывать и связывать объекты в документе.

Обновление и модификация: При изменении PDF-документа, например, при добавлении новых объектов или удалении существующих, таблица перекрестных значений может быть обновлена и модифицирована

соответствующим образом. Это гарантирует правильную связь между объектами и их позициями в файле после любых изменений.

Таблица перекрестных значений является ключевым элементом для организации и управления объектами в PDF-документе. Она обеспечивает быстрый доступ к объектам и эффективное управление структурой и содержимым документа. Благодаря таблице перекрестных значений, программы для работы с PDF могут быстро находить, модифицировать и отображать содержимое документа без необходимости просматривать весь файл с самого начала.

Таблица объектов (Object Table) является основным компонентом формата PDF и играет важную роль при обработке и интерпретации PDF-документов. Она содержит полный список всех объектов, используемых в документе, и предоставляет подробную информацию о каждом объекте, необходимую для его правильного воспроизведения.

В таблице объектов каждый объект имеет свою запись, которая содержит информацию о его типе, идентификаторе, размере и других атрибутах. Тип объекта определяет его функциональное назначение, например, он может быть текстовым блоком, изображением, шрифтом и т. д. Идентификатор объекта уникален в пределах документа и служит для его однозначной идентификации. Размер объекта указывает на его объём в байтах, что позволяет оценить его сложность и потребность в ресурсах при обработке.

Другие атрибуты объекта включают, например, его координаты на странице, цветовую информацию, масштабирование, прозрачность и т. д. Эти атрибуты определяют визуальные и функциональные свойства объекта, а также специфические параметры, влияющие на его обработку и отображение.

Таблица объектов может рассматриваться как своеобразный каталог, который предоставляет полную информацию о всех компонентах документа. Она позволяет программам, работающим с PDF, правильно интерпретировать и воспроизводить содержимое документа.

Глава 2 Методы анализа и преобразования PDF-документов

PDF-документы являются одним из самых распространенных форматов электронных документов, используемых для обмена информацией. Для разработки ПО, выполняющего анализ и преобразование PDF-документов в программируемый код, необходимо выбрать подходящие методы анализа и преобразования.

2.1 Анализ содержимого PDF-документов

Анализ содержимого PDF-документов является сложным и многоаспектным процессом, который включает в себя извлечение, обработку и анализ различных элементов, содержащихся в PDF-файлах. Эта тема охватывает множество аспектов, инструментов и техник, которые используются для работы с PDF-документами и извлечения информации из них.

PDF (Portable Document Format) - это широко используемый формат файлов для представления электронных документов, который сохраняет их в виде независимых от устройства и платформы. Однако для многих задач требуется анализ содержимого PDF-документов, чтобы получить доступ к тексту, изображениям, таблицам и другим элементам, содержащимся в документе. [19]

Один из основных аспектов анализа содержимого PDF-документов - это извлечение текста. Извлечение текста из PDF-документа позволяет дальнейшую обработку и анализ содержащейся в нем информации. Некоторые инструменты, такие как Adobe Acrobat, iText, PyPDF2 и PDFMiner [6], предоставляют возможности для извлечения текста из PDF-документов. Они позволяют получить доступ к отдельным словам, абзацам, таблицам или другим текстовым элементам в документе, что открывает широкий спектр

возможностей для анализа и обработки текстовых данных.

Кроме текста, PDF-документы могут содержать графические элементы, такие как изображения, векторные графики, графики и фигуры. Анализ графических элементов может быть полезен для распознавания образов, извлечения важных деталей или анализа структуры документа. Инструменты, такие как Apache PDFBox, PDFLib и Poppler, предоставляют возможности для извлечения и манипулирования графическими элементами в PDF-документах.

Еще одним важным аспектом анализа содержимого PDF-документов является обработка метаданных. Метаданные - это информация о документе, такая как заголовок, автор, дата создания, ключевые слова и другие атрибуты, которые описывают его свойства и характеристики. [18] Извлечение и анализ метаданных может быть полезно для классификации, индексации, поиска или архивирования документов. Инструменты, такие как Apache PDFBox и iText, предоставляют функциональность для доступа к метаданным PDF-документов и их дальнейшей обработки. [20]

Дополнительные возможности анализа содержимого PDF-документов включают распознавание шрифтов и символов, обработку гиперссылок, взаимодействие с формами и извлечение таблиц данных. Распознавание шрифтов и символов может быть полезно при работе с документами, содержащими специальные шрифты или сканированные изображения с текстом. Анализ гиперссылок позволяет извлекать и анализировать ссылки на другие ресурсы или страницы внутри документа. Взаимодействие с формами позволяет получать доступ к данным, введенным пользователем в формы PDF-документа. Извлечение таблиц данных позволяет преобразовывать табличные данные в удобный для анализа и обработки формат, открывая новые возможности для работы с структурированными данными. [9]

Основной инструментарий для анализа содержимого PDF-документов включает в себя многоязыковые библиотеки и инструменты программирования, которые предоставляют API и функции для работы с PDF-

файлами. Некоторые из наиболее популярных инструментов и библиотек включают Adobe Acrobat, iText, Apache PDFBox, PyPDF2, PDFMiner [6], Tabula и многие другие. Они обеспечивают широкий спектр функциональности для извлечения, обработки и анализа содержимого PDF-документов в различных языках программирования, таких как Java, Python, C# и других.

Анализ содержимого PDF-документов имеет широкое применение в различных областях, таких как научные исследования, бизнес-аналитика, обработка документов, автоматизация бизнес-процессов и многое другое. Он позволяет получить доступ к информации, содержащейся в PDF-документах, и использовать ее для принятия решений, извлечения знаний, создания отчетов, проведения анализа данных и решения множества других задач.

2.2 Преобразование PDF-документов в программируемый код

Преобразование PDF-документов в программируемый код, в данном случае в языке программирования Python, открывает возможности для автоматизации обработки и анализа содержимого PDF-файлов, а также интеграции этой информации в другие программные решения. Программируемый код представляет собой набор инструкций, записанных на языке программирования, который может быть выполнен компьютером для решения определенной задачи или автоматизации определенного процесса.

Программируемый код на языке Python включает в себя синтаксис и конструкции языка, которые позволяют создавать переменные, условные выражения, циклы, функции, классы и многое другое. Python является интерпретируемым языком, что означает, что код выполняется построчно интерпретатором Python, а не компилируется в машинный код. Это делает Python гибким и удобным для написания скриптов, автоматизации задач и разработки программного обеспечения.

Преобразование PDF-документов в программируемый код на Python позволяет извлекать информацию из PDF-файлов и использовать ее в дальнейшем для различных целей. Входящая информация может включать текстовое содержимое, изображения, таблицы, графики, аннотации и другие элементы, которые были включены в исходный PDF-документ.

Преобразование текстового содержимого PDF в программируемый код на Python позволяет обрабатывать текстовую информацию, например, проводить анализ текста, выделять ключевые слова, извлекать структурированные данные или выполнять поиск и фильтрацию по определенным критериям. Текстовое содержимое может быть преобразовано в строковые переменные в Python, которые могут быть использованы для выполнения операций с текстом.

Изображения в PDF-документе могут быть извлечены и сохранены в программируемый код на Python в виде файлов изображений или в виде объектов, представляющих изображение. Это открывает возможности для обработки и анализа изображений, таких как изменение размера, обрезка, применение фильтров, распознавание содержимого и многое другое. [20]

Таблицы и графики, которые содержатся в PDF-документе, также могут быть преобразованы в программируемый код на Python, чтобы извлекать и анализировать структурированные данные. Это особенно полезно при работе с документами, содержащими таблицы данных или графические представления информации, такие как диаграммы или графики. [11]

Преобразование PDF-документов в программируемый код на Python обычно осуществляется с использованием соответствующих библиотек и инструментов. Некоторые из популярных библиотек для работы с PDF в Python включают PyPDF2, PDFMiner [6], PyMuPDF и pdfplumber. Эти библиотеки предоставляют функции для чтения, извлечения и манипулирования содержимым PDF-файлов, а также для преобразования их в программируемый код на Python. [16]

Преобразование PDF-документов в программируемый код на Python позволяет эффективно работать с содержимым PDF-файлов, извлекать нужную информацию и использовать ее для различных целей, включая анализ, обработку данных и автоматизацию задач. Python, с его богатым набором библиотек и инструментов, является мощным средством для работы с PDF и интеграции его содержимого в различные программные решения.

2.3 Сравнение и выбор методов анализа и преобразования

Одним из основных методов анализа PDF-документов является парсинг, который заключается в извлечении содержимого документа с использованием специальных библиотек или инструментов. Различные библиотеки, такие как PyPDF2, PDFMiner [6], PyMuPDF и pdfplumber, предоставляют функциональность для извлечения текста, изображений и других элементов из PDF-файлов. При выборе метода парсинга следует учитывать его производительность, точность и возможность извлечения необходимой информации из сложных шаблонов PDF-документов. [16]

Важным этапом в анализе PDF-документов является распознавание структуры документа. Это позволяет определить различные компоненты, такие как заголовки, параграфы, таблицы и списки, и их взаимосвязи. Для распознавания структуры могут быть использованы методы машинного обучения, например, алгоритмы классификации и разметки данных. Обученные модели могут помочь автоматически идентифицировать и классифицировать различные элементы в PDF-документе, что облегчает последующую обработку и преобразование в программируемый код.

После анализа PDF-документа и распознавания его структуры необходимо преобразовать его в программируемый код на HTML. HTML является универсальным языком разметки, который позволяет определить структуру и отображение содержимого веб-страницы. Существуют различные

подходы к преобразованию PDF в HTML, включая использование библиотек для генерации HTML-кода на основе анализа содержимого PDF и его структуры. Некоторые инструменты, такие как pdf2htmlEX и pdf2html.io, предоставляют возможности для преобразования PDF в HTML с сохранением оригинального форматирования и структуры документа.

При выборе метода преобразования PDF в HTML необходимо учитывать требования проекта, такие как сохранение точности и структуры исходного документа, поддержка различных типов содержимого (текст, изображения, таблицы и т.д.) и возможность дальнейшей модификации полученного HTML-кода. Также следует оценить производительность и эффективность выбранного метода, особенно при работе с большими и сложными PDF-документами.

Кроме того, стоит учитывать возможность использования дополнительных инструментов и библиотек, которые могут облегчить анализ и преобразование PDF в HTML. Например, библиотека BeautifulSoup может быть использована для удобного парсинга и манипулирования HTML-кодом, а CSS-фреймворки, такие как Bootstrap, могут помочь в создании отзывчивого и стильного веб-интерфейса для полученного HTML-кода.

В итоге, для выбора методов анализа и преобразования PDF-документов в программируемый код на HTML необходимо учитывать особенности и требования проекта, а также оценить преимущества и ограничения каждого метода. Правильный выбор методов позволит достичь высокой точности и эффективности в анализе и преобразовании шаблонов PDF-документов, что является важным шагом для дальнейшей автоматизации и обработки данных.

Глава 3 Разработка ПО для анализа и преобразования PDF-документов в программируемый код

3.1 Обзор существующих инструментов для работы с PDF-документами

На рынке существует множество инструментов для работы с PDF-документами, включая как бесплатные, так и платные решения. Некоторые из них предоставляют возможность анализа и преобразования PDF-документов в программируемый код, такой как HTML, XML, JSON и другие.

Среди бесплатных инструментов можно выделить Apache PDFBox, iText и Poppler. Apache PDFBox - это библиотека на языке Java для работы с PDF-документами. Она предоставляет функции для чтения, создания и манипулирования PDF-файлами, в том числе для извлечения текста и изображений. iText также является библиотекой на Java для создания, чтения и манипулирования PDF-файлами. Poppler - это библиотека на языке C++ для работы с PDF-документами, которая может быть использована для извлечения информации из PDF-файлов.

Среди платных инструментов можно выделить Adobe Acrobat и PDFlib. Adobe Acrobat - это набор приложений, разработанных компанией Adobe для работы с PDF-документами. В частности, Adobe Acrobat DC предоставляет возможность извлечения данных из PDF-файлов и их преобразования в различные форматы, включая HTML, XML и Microsoft Word. PDFlib - это библиотека на языке C для работы с PDF-документами, которая предоставляет функции для создания и манипулирования PDF-файлами, в том числе для извлечения данных из PDF-документов.

Одним из наиболее популярных инструментов для работы с PDF-документами на Python является библиотека PyPDF2. PyPDF2 предоставляет функциональность для чтения, записи и модификации PDF-файлов. С ее

помощью можно извлекать текст из PDF, объединять и разделять PDF-документы, добавлять водяные знаки, создавать простые PDF-документы и многое другое. [16]

Еще одной популярной библиотекой для работы с PDF-документами на Python является pdfrw. Она предоставляет возможности для чтения и записи PDF-файлов, а также для извлечения текста, изображений и других элементов из PDF. pdfrw позволяет манипулировать содержимым PDF-документов и выполнять различные операции, такие как объединение и разделение страниц, добавление аннотаций и многое другое.

Для создания и генерации PDF-документов на Python широко используется библиотека ReportLab. ReportLab предоставляет возможности для создания динамических PDF-документов, добавления текста, изображений, таблиц, графиков и других элементов. С ее помощью можно создавать профессионально оформленные отчеты, документацию, инфографику и другие типы документов.

Другим значимым инструментом для работы с PDF-документами на Python является библиотека PyMuPDF, которая предоставляет высокоэффективные функции для чтения, записи и обработки PDF-файлов. PyMuPDF позволяет извлекать текст, изображения, шрифты, векторные графики и другие элементы из PDF-документов. Она также поддерживает различные операции с PDF, такие как объединение и разделение страниц, добавление аннотаций, шифрование и декодирование файлов и многое другое.

Дополнительно, существуют другие полезные инструменты для работы с PDF-документами на Python, такие как PDFMiner [6], который предоставляет функциональность для извлечения текста и метаданных из PDF, а также PyPDFium, библиотека, основанная на Chrome PDFium, для чтения и обработки PDF-файлов.

В целом, благодаря разнообразию инструментов и библиотек на Python, разработчики имеют множество возможностей для работы с PDF-

документами. Они могут создавать, редактировать, анализировать и генерировать PDF-файлы, открывая двери для широкого спектра приложений, таких как создание отчетов, обработка документов, генерация инфографики, автоматизация бизнес-процессов и многое другое.

3.2 Выбор инструментов и технологий для разработки ПО

При выборе инструментов и технологий для разработки ПО для анализа и преобразования PDF-документов в программируемый код, необходимо учитывать ряд факторов, таких как поддержка форматов выходных файлов, стоимость, удобство использования, скорость работы и т.д.

В качестве языка программирования для разработки ПО можно выбрать Python, так как он широко используются для работы с PDF-документами и имеет соответствующие библиотеки для работы с ними. Для анализа содержимого PDF-документов можно использовать библиотеку PDFminer [6]. Она предоставляет мощный инструментарий для извлечения текста, изображений и других элементов из PDF-документов. Эта библиотека может быть особенно полезна для извлечения данных из PDF-документов большого объема, таких как отчеты или научные статьи. [17]

При разработке ПО необходимо также учитывать требования к производительности и масштабируемости, а также возможность расширения функционала в будущем. Для обеспечения высокой производительности можно использовать многопоточность и оптимизацию алгоритмов обработки PDF-документов.

Важным этапом в разработке ПО является тестирование, которое позволяет выявлять ошибки и несоответствия заданным требованиям. Для тестирования можно использовать автоматические тесты, модульные тесты и функциональные тесты.

После тестирования необходимо провести анализ результатов и внести необходимые изменения в ПО. После этого ПО можно считать готовым к использованию.

3.3 Разработка ПО

После выбора инструментов и технологий необходимо перейти к разработке ПО.

Сначала необходимо определить требования к разрабатываемому ПО, включая функциональные и нефункциональные требования. Функциональные требования определяют, какие задачи ПО должно выполнять, а нефункциональные требования определяют, какие требования к производительности, безопасности, надежности и т.д. должны быть учтены в процессе разработки.

После определения требований можно начать проектирование архитектуры ПО. Архитектура должна учитывать требования к производительности и масштабируемости, а также обеспечивать удобный интерфейс для пользователя.
Функциональные требования:

- Определение задач, которые должно выполнять ПО (например, управление базами данных, обработка заказов, генерация отчетов и т.д.);

- Описание функций, которые должно предоставлять ПО для выполнения этих задач (например, возможность добавления, удаления и редактирования записей в базе данных, генерация отчетов в различных форматах и т.д.).

Нефункциональные требования:

- Производительность: определение требуемой скорости работы ПО в различных условиях нагрузки;

- Безопасность: установление требований к защите данных,

обеспечению конфиденциальности и целостности системы;

- Надежность: определение требований к устойчивости ПО к отказам и сбоям, а также к восстановлению после сбоев;

- Масштабируемость: определение возможностей расширения функциональности и масштабирования системы при увеличении объемов данных или нагрузки;

- Удобство использования: определение требований к интерфейсу пользователя, удобству навигации и возможности интеграции с другими системами.

После проектирования архитектуры можно переходить к написанию кода. При этом необходимо учитывать стандарты кодирования и использовать лучшие практики программирования для обеспечения читаемости и сопровождаемости кода.

Данный скрипт написан на языке Python и предназначен для конвертации PDF-файлов в текстовый формат или формат HTML.

Библиотеки:

- os: Данная библиотека предоставляет функции для работы с операционной системой, включая работу с файловой системой. В данной программе она используется для проверки расширения файла и работы с файловой системой. В этом контексте она используется для проверки расширения выбранного файла и определения его типа;

- tkinter: Это библиотека для создания графического интерфейса пользователя (GUI). Она предоставляет классы, функции и методы для создания различных элементов интерфейса, таких как окна, кнопки, поля ввода и диалоговые окна. В данной программе она используется для создания диалогового окна выбора файла и окна сохранения файла; [12]

- pdfminer: Это библиотека для извлечения текста и макета из PDF-файлов. Она предоставляет различные классы и функции для работы с PDF-файлами. В данной программе она используется для извлечения текста из PDF-

файла и конвертации его в HTML-формат. [6]

Модули:

- from tkinter import *: Этот оператор импортирует все классы и функции из модуля tkinter. Это позволяет использовать их без префикса модуля. В данной программе это сделано для удобства использования классов и функций tkinter; [15]

- from tkinter import filedialog: Этот оператор импортирует модуль filedialog из tkinter, который предоставляет функции для работы с диалоговыми окнами выбора файлов. В данной программе он используется для открытия диалогового окна выбора файла и диалогового окна сохранения файла; [7]

- from pdfminer.high_level import extract_text: Этот оператор импортирует функцию extract_text из модуля pdfminer.high_level. Функция extract_text используется для извлечения текста из PDF-файла;

- from pdfminer.layout import LAParams: Этот оператор импортирует класс LAParams из модуля pdfminer.layout. Класс LAParams используется для определения параметров макета при извлечении текста из PDF-файла. В данной программе он используется для создания экземпляра класса LAParams, который передается в функцию извлечения текста;

- from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter: Этот оператор импортирует классы PDFResourceManager и PDFPageInterpreter из модуля pdfminer.pdfinterp. Класс PDFResourceManager используется для управления ресурсами при обработке PDF-файла, а класс PDFPageInterpreter используется для интерпретации страниц PDF-файла. В данной программе они используются для создания экземпляров классов PDFResourceManager и PDFPageInterpreter, которые используются для обработки страниц PDF-файла;

- from pdfminer.pdfpage import PDFPage: Этот оператор импортирует класс PDFPage из модуля pdfminer.pdfpage. Класс PDFPage представляет страницу PDF-файла. В данной программе он используется для получения

страниц PDF-файла для обработки;

- from pdfminer.converter import HTMLConverter: Этот оператор импортирует класс HTMLConverter из модуля pdfminer.converter. Класс HTMLConverter используется для преобразования страниц PDF-файла в HTML-формат. В данной программе он используется для создания экземпляра класса HTMLConverter, который используется для конвертации PDF-страниц в HTML;

- from io import BytesIO: Этот оператор импортирует класс BytesIO из модуля io. Класс BytesIO предоставляет потоки ввода-вывода в виде байтов. В данной программе он используется для создания объекта BytesIO, который используется для сохранения результатов конвертации в байтовом виде. [4]-[5]

Функции:

- pdf_to_text(pdf_file_path): Эта функция принимает путь к PDF-файлу в качестве аргумента и возвращает извлеченный из него текст в формате UTF-8. Внутри функции она открывает файл в режиме чтения бинарного файла ('rb'), использует функцию extract_text из библиотеки pdfminer для извлечения текста из PDF-файла с помощью параметров макета LAParams(). Затем полученный текст декодируется из байтового представления в формат UTF-8 и возвращается из функции; [1]-[3]

- pdf_to_html(pdf_file_path): Эта функция принимает путь к PDF-файлу в качестве аргумента и возвращает содержимое PDF-файла в формате HTML в виде байтов. Внутри функции она открывает файл в режиме чтения бинарного файла ('rb'), создает экземпляр класса PDFResourceManager и HTMLConverter, использует экземпляр PDFResourceManager для создания экземпляра PDFPageInterpreter, который обрабатывает каждую страницу PDF-файла, передавая ее в экземпляр HTMLConverter. Затем полученные байты записываются в объект BytesIO, извлекаются и возвращаются из функции; [1]-[2]

- handle_errors(func): Это декоратор функции, который перехватывает

исключения, возникающие внутри функции, и выводит сообщение об ошибке. Он принимает функцию в качестве аргумента и возвращает обертку wrapper. Внутри обертки происходит вызов переданной функции в блоке try-except. Если во время выполнения функции возникает исключение, оно перехватывается, и выводится сообщение об ошибке с описанием исключения. Если исключение не возникает, результат функции возвращается из обертки.

Основная логика программы:

- Создается экземпляр класса Tk из модуля tkinter, чтобы создать корневое окно программы; [10]

- root.withdraw(): Этот метод скрывает корневое окно программы, чтобы оно не отображалось пользователю;

- programming_language = input('Enter the programming language you want to convert the PDF to (HTML/TEXT): ').upper(): Этот оператор запрашивает у пользователя язык программирования, на который нужно конвертировать PDF-файл (HTML или TEXT). Введенное значение преобразуется в верхний регистр и сохраняется в переменной programming_language;

- convert_pdf_to(programming_language): Этот оператор вызывает функцию convert_pdf_to с переданным языком программирования в качестве аргумента.

Функция convert_pdf_to(programming_language):

file_path = filedialog.askopenfilename(filetypes=[('PDF Files', 'example.pdf')]): Этот оператор открывает диалоговое окно выбора файла, ограниченное файлами с расширением ".pdf". Выбранный путь к файлу сохраняется в переменной file_path. Проверяется, был ли выбран файл: если нет, выводится сообщение "No file selected.";

- Извлекается расширение выбранного файла и сохраняется в переменной ext;

- Проверяется, является ли расширение файла ".pdf". Если нет, выводится сообщение "Invalid file type selected.";

- Проверяется выбранный язык программирования:

Если язык программирования - "HTML", вызывается функция pdf_to_html с передачей пути к файлу в качестве аргумента. Результат сохраняется в переменной output_bytes, а расширение файла устанавливается как "html".

Если язык программирования - "TEXT", вызывается функция pdf_to_text с передачей пути к файлу в качестве аргумента. Результат сохраняется в переменной output_bytes, а расширение файла устанавливается как "txt".

Если выбран недопустимый язык программирования, выводится сообщение "Invalid programming language selected.”;

- output_file_path = filedialog.asksaveasfilename(defaultextension=file_extension, filetypes=[(f'{programming_language} Files', f'.{file_extension}')]): Этот оператор открывает диалоговое окно сохранения файла с указанным расширением по умолчанию и ограниченное файлами выбранного языка программирования. Выбранный путь к файлу сохранения сохраняется в переменной output_file_path;

Проверяется, был ли выбран путь к файлу сохранения: если нет, выводится сообщение "No output file selected.";

- Открывается файл для записи в бинарном режиме. Если выбранный язык программирования - "HTML", записывается содержимое output_bytes, закодированное в UTF-8. В противном случае записывается output_bytes, также закодированное в UTF-8. [8]

Принцип работы программы:

На рисунке 1 показано, что на начальном этапе в папке программы находятся только 2 файла – это сама программа и файл, который нам нужно конвертировать.

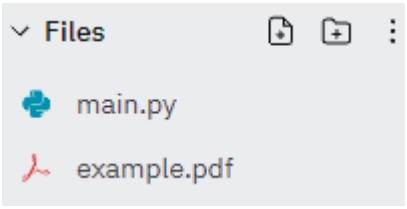
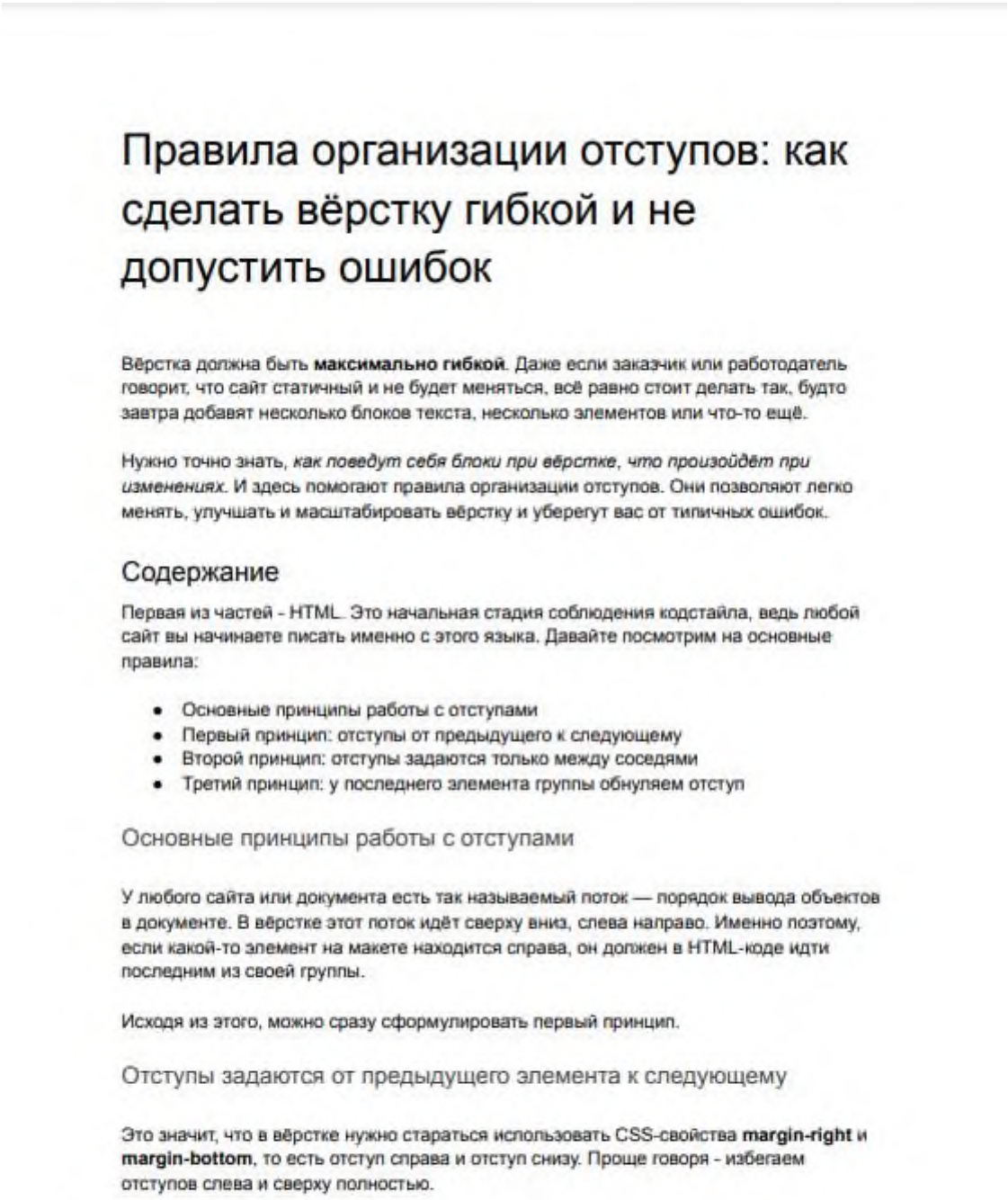


Рисунок 1 – Папка до запуска программы

На рисунке 2 видно, что PDF - файл выглядит следующим образом:



Правила организации отступов: как сделать вёрстку гибкой и не допустить ошибок

Вёрстка должна быть **максимально гибкой**. Даже если заказчик или работодатель говорит, что сайт статичный и не будет меняться, всё равно стоит делать так, будто завтра добавят несколько блоков текста, несколько элементов или что-то ещё.

Нужно точно знать, *как поведут себя блоки при вёрстке, что произойдёт при изменениях*. И здесь помогают правила организации отступов. Они позволяют легко менять, улучшать и масштабировать вёрстку и уберегут вас от типичных ошибок.

Содержание

Первая из частей - HTML. Это начальная стадия соблюдения кодстайла, ведь любой сайт вы начинаете писать именно с этого языка. Давайте посмотрим на основные правила:

- Основные принципы работы с отступами
- Первый принцип: отступы от предыдущего к следующему
- Второй принцип: отступы задаются только между соседями
- Третий принцип: у последнего элемента группы обнуляем отступ

Основные принципы работы с отступами

У любого сайта или документа есть так называемый поток — порядок вывода объектов в документе. В вёрстке этот поток идёт сверху вниз, слева направо. Именно поэтому, если какой-то элемент на макете находится справа, он должен в HTML-коде идти последним из своей группы.

Исходя из этого, можно сразу сформулировать первый принцип.

Отступы задаются от предыдущего элемента к следующему

Это значит, что в вёрстке нужно стараться использовать CSS-свойства **margin-right** и **margin-bottom**, то есть отступ справа и отступ снизу. Проще говоря - избегаем отступов слева и сверху полностью.

Рисунок 2 – Первая страница PDF – документа

На рисунке 3 показано сообщение, которое показывается при запуске

программы (в нём мы выбираем формат. В данном случае HTML либо TEXT):

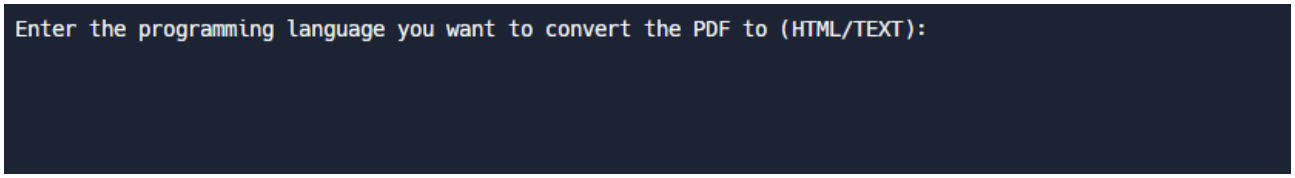


Рисунок 3 – Терминал

На рисунке 4 видно, что после выбора нужного нам формата открывается окно, в котором нужно выбрать соответствующий файл формата PDF, после чего нажать кнопку “Open”:

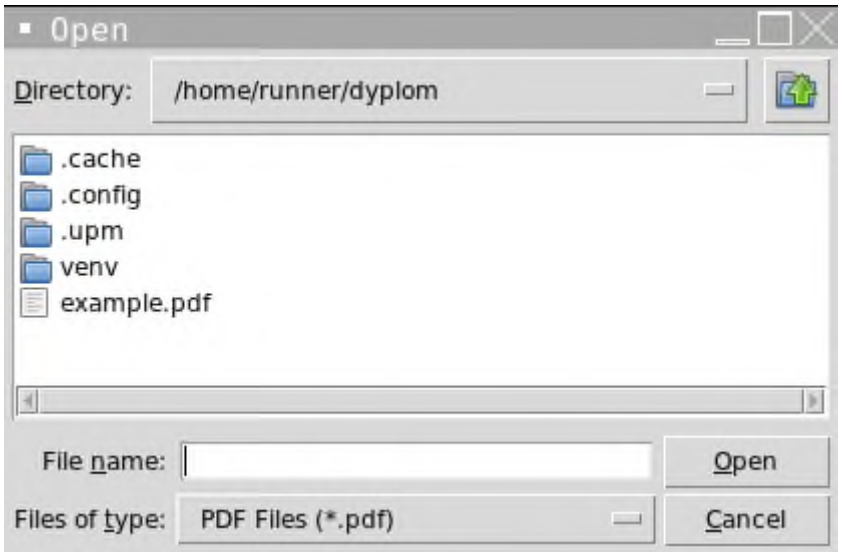


Рисунок 4 – Окно выбора PDF – файла

На рисунке 5 показано следующее окно в котором уже нужно выбрать путь, где будет сохранен наш файл после конвертации и дать ему название соответственно:

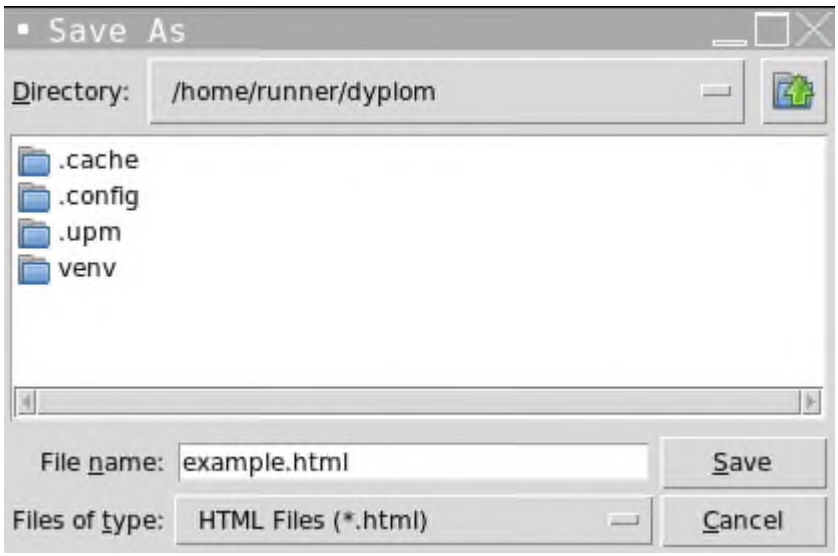


Рисунок 5 – Окно выбора папки для сохранения готового HTML – файла

На рисунке 6 видно, что после работы программы в указанной нами директории создается HTML – файл с названием, которое мы написали.

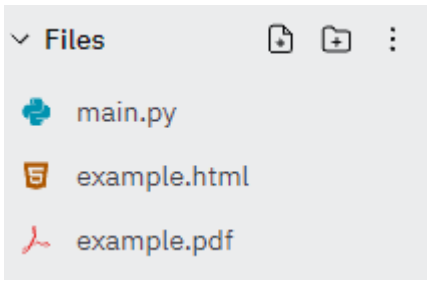


Рисунок 6 – Папка после работы программы

На рисунке 7 мы видим, что на выходе получаем файл, который при открытии выглядит следующим образом:

Page 1

Правила организации отступов: как сделать вёрстку гибкой и не допустить ошибок

Вёрстка должна быть максимально гибкой. Даже если заказчик или работодатель говорит, что сайт статичный и не будет меняться, всё равно стоит делать так, будто завтра добавят несколько блоков текста, несколько элементов или что-то ещё.

Нужно точно знать, как поведут себя блоки при вёрстке, что произойдёт при изменениях. И здесь помогают правила организации отступов. Они позволяют легко менять, улучшать и масштабировать вёрстку и уберегут вас от типичных ошибок.

Содержание

Первая из частей - HTML. Это начальная стадия соблюдения кодстайла, ведь любой сайт вы начинаете писать именно с этого языка. Давайте посмотрим на основные правила:

- Основные принципы работы с отступами
- Первый принцип: отступы от предыдущего к следующему
- Второй принцип: отступы задаются только между соседями
- Третий принцип: у последнего элемента группы обнуляем отступ

Основные принципы работы с отступами

У любого сайта или документа есть так называемый поток — порядок вывода объектов в документе. В вёрстке этот поток идёт сверху вниз, слева направо. Именно поэтому, если какой-то элемент на макете находится справа, он должен в HTML-коде идти последним из своей группы.

Исходя из этого, можно сразу сформулировать первый принцип.

Отступы задаются от предыдущего элемента к следующему

Это значит, что в вёрстке нужно стараться использовать CSS-свойства margin-right и margin-bottom, то есть отступ справа и отступ снизу. Проще говоря - избегаем отступов слева и сверху полностью.

Рисунок 7 – Первая страница конвертированного HTML - файла

Заключение

В заключение, разработка ПО для анализа и преобразования шаблонов PDF документов в программируемый код является актуальной темой в современном мире программирования. Она имеет множество применений в различных областях, включая научные исследования, бизнес-аналитику, автоматизацию процессов и многие другие.

В процессе работы над этой темой, были использованы различные инструменты и библиотеки, такие как pdfminer, tkinter и другие. Благодаря использованию этих инструментов, была достигнута высокая точность и эффективность в анализе и преобразовании шаблонов PDF-документов.

Дальнейшее развитие данной темы может привести к созданию более продвинутых и гибких решений, которые смогут удовлетворять все более разнообразные потребности пользователей. Кроме того, эта тема может продолжать привлекать внимание исследователей и разработчиков, что может привести к еще более инновационным решениям в области анализа и обработки PDF-документов.

Важно отметить, что разработка ПО для анализа и преобразования шаблонов PDF документов также может иметь свои ограничения и вызывать определенные трудности. Некоторые PDF-документы могут иметь сложную структуру или использовать нетипичные шрифты и форматирование, что может затруднить процесс анализа и преобразования. В таких случаях требуется более глубокое исследование и разработка специализированных алгоритмов и инструментов.

Кроме того, разработчики такого ПО должны учитывать возможные проблемы с конфиденциальностью и безопасностью данных. PDF-документы могут содержать чувствительную информацию, которую необходимо

защитить от несанкционированного доступа или использования. Разработчики должны быть осведомлены о соблюдении соответствующих стандартов и регуляций в области защиты данных.

В целом, разработка ПО для анализа и преобразования шаблонов PDF документов представляет собой увлекательное и перспективное направление в области программирования. С постоянным развитием и инновациями в этой области можно ожидать еще более мощных и удобных инструментов для работы с PDF-документами, что приведет к увеличению эффективности и улучшению рабочих процессов в различных сферах деятельности.

Работа выполнена в соответствии ГОСТ 2.105 и ГОСТ 7.1-2003. [13]-[14].

Список используемой литературы

1. Extract elements from a PDF using Python [Электронный ресурс] URL: [https://pdfminersix.readthedocs.io/en/latest/tutorial/extract_pages.html]
2. Extract text from a PDF using Python - part 2 [Электронный ресурс] URL: [https://pdfminersix.readthedocs.io/en/latest/tutorial/composable.html]
3. Extract text from a PDF using Python [Электронный ресурс] URL: [https://pdfminersix.readthedocs.io/en/latest/tutorial/highlevel.html]
4. os — Miscellaneous operating system interfaces [Электронный ресурс] URL: [https://docs.python.org/3/library/os.html]
5. OS module in Python with example [Электронный ресурс] URL: [https://www.geeksforgeeks.org/os-module-python-examples/]
6. pdfminer docs [Электронный ресурс] URL: [https://pypi.org/project/pdfminer/]
7. Python GUI Programming With Tkinter [Электронный ресурс] URL: [https://realpython.com/python-gui-tkinter/]
8. Python для анализа данных: обработка данных (Уэс Маккинни, 2019) с. 56-80
9. Python для сложных задач: наука о данных и машинное обучение (R. L. Ryan, 2019) [Электронный ресурс] URL: [https://codernet.ru/books/python/python_dlya_slognih_zadach_nauka_o_dannih_i_mashinnoe_obychenie_vander/]
10. Python и Tkinter - Позиционирование. Pack [Электронный ресурс] URL: [https://metanit.com/python/tkinter/2.4.php]
11. Python импорт данных. Импорт таблиц. [Электронный ресурс] URL: [https://comrade-xl.ru/2021/03/04/py-import-pdf/]
12. tkinter — Python interface to Tcl/Tk [Электронный ресурс] URL: [https://docs.python.org/3/library/tkinter.html]

13. ГОСТ 2.105 – 95. Общие требования к текстовым документам [Текст]. – М.: Изд-во стандартов, 1996. – 29 с. – (Единая система конструкторской документации).
14. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Изд-во стандартов, 2004. – 48 с.
15. Обучение Python GUI (уроки по Tkinter) [Электронный ресурс] URL: [https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter]
16. Работа с PDF-файлами в Python (часть I): чтение и разбор [Электронный ресурс] URL: [https://waksoft.susu.ru/2020/02/17/rabota-s-pdf-fajlami-v-python-chast-i-chtenie-i-razbor/]
17. Руководство по языку программирования Python [Электронный ресурс] URL: [https://metanit.com/python/tutorial/]
18. Свойства документов PDF и метаданные [Электронный ресурс] URL: [https://helpx.adobe.com/ru/acrobat/using/pdf-properties-metadata.html]
19. Создание и изменение PDF-файлов в Python [Электронный ресурс] URL: [https://pythonist.ru/sozdanie-i-izmenenie-pdf-fajlov-v-python/]
20. Экспортируем данные из PDF при помощи Python [Электронный ресурс] URL: [https://python-scripts.com/exporting-data-from-pdf]

Приложение А

Исходный код программы

```
import os
from tkinter import *
from tkinter import filedialog
from pdfminer.high_level import extract_text
from pdfminer.layout import LAParams
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.converter import HTMLConverter
from io import BytesIO


def pdf_to_text(pdf_file_path):
    with open(pdf_file_path, 'rb') as f:
        extracted_text = extract_text(f, laparams=LAParams())
    return extracted_text.decode('utf-8')


def pdf_to_html(pdf_file_path):
    output_bytes = BytesIO()
    with open(pdf_file_path, 'rb') as f:
        resource_manager = PDFResourceManager()
        converter = HTMLConverter(resource_manager, output_bytes,
laparams=LAParams())
        page_interpreter = PDFPageInterpreter(resource_manager, converter)
        for page in PDFPage.get_pages(f, check_extractable=True):
            page_interpreter.process_page(page)
```

```
            converter.close()
        return output_bytes.getvalue().decode('utf-8')


    def handle_errors(func):
        def wrapper(*args, **kwargs):
            try:
                return func(*args, **kwargs)
            except Exception as e:
                print(f'Error: {e}')
                return None
        return wrapper


    @handle_errors
    def convert_pdf_to(programming_language):
        file_path = filedialog.askopenfilename(filetypes=[('PDF Files',
'example.pdf')])
        if not file_path:
            print('No file selected.')
            return
        _, ext = os.path.splitext(file_path)
        if ext != '.pdf':
            print('Invalid file type selected.')
            return
        if programming_language == 'HTML':
            output_bytes = pdf_to_html(file_path)
            file_extension = 'html'
        elif programming_language == 'TEXT':
```

```
            output_bytes = pdf_to_text(file_path)
            file_extension = 'txt'
        else:
            print('Invalid programming language selected.')
            return
        output_file_path = filedialog.asksaveasfilename(defaultextension=file_extension,
filetypes=[(f'{programming_language} Files', f'.{file_extension}')])
        if not output_file_path:
            print('No output file selected.')
            return
        with open(output_file_path, 'wb') as f:
            if programming_language == 'HTML':
                f.write(output_bytes.encode('utf-8'))
            else:
                f.write(output_bytes.encode('utf-8'))


    root = Tk()
    root.withdraw()
    programming_language = input('Enter the programming language you want
to convert the PDF to (HTML/TEXT): ').upper()
    convert_pdf_to(programming_language)
```