МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Интеллектуальных систем и программирования

Кафедра "Программная инженерия" им. Л.П. Фельдмана

Лабораторная работа №4

по курсу: «Профессиональная практика программной инженерии»

на тему: «Создание самодокументирующегося кода»

Выполнил:

ст. гр. ПИ-21б

Зосимов В. В.

Проверил(а):

Филипишин Д. А

Ищенко А. П.

ДОНЕЦК – 2025

**Цель работы**: научиться добавлять в программный код специальным образом оформление докблок-комментарии, для последующей автоматической генерации API reference, а также познакомиться с форматом оформления документации DocBook.

**Задание к лабораторной работе:**

Внедрить в ранее разработанный программный код докблоки, соответствующие выбранному языку программирования.

Разработать DocBook и проверить корректность созданного DocBook файла с помощью средств редактора или одного из онлайн валидаторов.

Автоматически сформировать справочное руководство для программистов и выслать вместе с отчётом. В руководстве должны присутствовать все модули, описанные в лабораторной работе №1.

Добавление DocBook-комментариев

**Тема выбранного проекта**: 8. Блокнот (с реализацией поддержки семантики языков программирования, учесть наличие тем оформления);

Несколько примеров докблоков из программного кода разрабатываемого проекта подходящих к выбранному языку программирования, а также XML.

class Application:

"""

Класс приложения.

Этот класс отвечает за инициализацию и запуск приложения на различных платформах.

Атрибуты:

platform (str): Платформа, на которой будет работать приложение ("web" или "desktop").

"""

def \_\_init\_\_(self, platform):

self.platform = platform

if platform == "web":

self.init\_web\_app()

elif platform == "desktop":

self.init\_desktop\_app()

def init\_web\_app(self):

"""Инициализация веб-приложения."""

print("Инициализация веб-приложения")

def init\_desktop\_app(self):

"""Инициализация настольного приложения."""

print("Инициализация настольного приложения")

def run(self):

"""Запуск приложения."""

print(f"Запуск приложения на платформе: {self.platform}")

class AutoComplete:

""" Класс автозаполнения.

Этот класс предоставляет функции автозаполнения для языков программирования.

Атрибуты:

keywords (list): Список ключевых слов для автозаполнения.

"""

def \_\_init\_\_(self):

self.keywords = ["def", "class", "import", "from", "if", "else", "elif", "while", "for", "return"]

def get\_suggestions(self, input\_text):

"""

Получение предложений по автозаполнению.

Аргументы:

input\_text (str): Вводимый текст для получения предложений.

Возвращает:

list: Список предложений по автозаполнению.

"""

suggestions = []

for keyword in self.keywords:

if keyword.startswith(input\_text):

suggestions.append(keyword)

return suggestions

class ChangeHistory:

"""

Класс для отслеживания истории изменений текста.

Этот класс позволяет добавлять версии текста и управлять историей изменений.

Атрибуты:

history (list): Список версий текста.

current\_index (int): Индекс текущей версии.

"""

def \_\_init\_\_(self):

self.history = []

self.current\_index = -1

def add\_version(self, text):

"""Добавление новой версии текста."""

self.history.append(text)

self.current\_index = len(self.history) - 1

def undo(self):

"""Отмена последнего изменения."""

if self.current\_index > 0:

self.current\_index -= 1

return self.history[self.current\_index] if self.history else None

def redo(self):

"""Повтор последнего изменения."""

if self.current\_index < len(self.history) - 1:

self.current\_index += 1

return self.history[self.current\_index] if self.history else None

Файл docbook.xml

<book xmlns="http://docbook.org/ns/docbook" version="5.0">

<title>Документация к проекту Блокнот</title>

<chapter>

<title>Введение</title>

<para>Это документация к проекту Блокнот с поддержкой семантики языков программирования.</para>

</chapter>

<chapter>

<title>Модули</title>

<section>

<title>Application</title>

<para>Класс приложения, отвечающий за инициализацию и запуск.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_(platform)</para>

</listitem>

<listitem>

<para>init\_web\_app()</para>

</listitem>

<listitem>

<para>init\_desktop\_app()</para>

</listitem>

<listitem>

<para>run()</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>AutoComplete</title>

<para>Класс автозаполнения для языков программирования.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_()</para>

</listitem>

<listitem>

<para>get\_suggestions(input\_text)</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>ChangeHistory</title>

<para>Класс для отслеживания истории изменений текста.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_()</para>

</listitem>

<listitem>

<para>add\_version(text)</para>

</listitem>

<listitem>

<para>undo()</para>

</listitem>

<listitem>

<para>redo()</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>FileHandler</title>

<para>Класс для работы с файлами.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>open\_file(file\_path)</para>

</listitem>

<listitem>

<para>save\_file(file\_path, content)</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>PerformanceOptimizer</title>

<para>Класс для оптимизации производительности.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>cache\_data(data)</para>

</listitem>

<listitem>

<para>async\_process(task)</para>

</listitem>

<listitem>

<para>optimize\_large\_files(file)</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>SyntaxHighlighter</title>

<para>Класс для подсветки синтаксиса.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_()</para>

</listitem>

<listitem>

<para>highlight\_syntax(code)</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>UserInterface</title>

<para>Класс пользовательского интерфейса.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_()</para>

</listitem>

<listitem>

<para>create\_interface()</para>

</listitem>

<listitem>

<para>change\_theme(theme)</para>

</listitem>

</itemizedlist>

</section>

<section>

<title>ThemeManager</title>

<para>Класс для управления темами оформления в проекте Блокнот.</para>

<para>Этот класс позволяет загружать, переключать и получать информацию о доступных темах оформления.</para>

<para>Методы:</para>

<itemizedlist>

<listitem>

<para>\_\_init\_\_()</para>

</listitem>

<listitem>

<para>load\_theme(theme\_name)</para>

</listitem>

<listitem>

<para>switch\_theme(theme\_name)</para>

</listitem>

<listitem>

<para>get\_available\_themes()</para>

</listitem>

<listitem>

<para>get\_current\_theme()</para>

</listitem>

</itemizedlist>

</section>

</chapter>

<chapter>

<title>Заключение</title>

<para>Документация к проекту Блокнот была создана для облегчения понимания кода и его структуры. Надеемся, что она будет полезна как текущим, так и будущим разработчикам.</para>

</chapter>

</book>

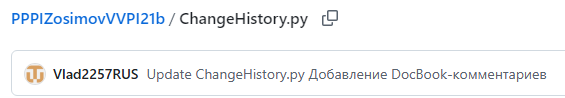


Рисунок 1 – Закоммититивание изменения в репозиторий

Описание разработанного продукта

Разработанный продукт представляет собой текстовый редактор "Блокнот", который включает в себя множество функций, таких как автозаполнение, подсветка синтаксиса, управление историей изменений и оптимизация производительности. Он ориентирован на разработчиков, которым необходимо быстро и эффективно редактировать код на различных языках программирования.

Приложение поддерживает работу с несколькими файлами одновременно, что позволяет пользователю легко переключаться между проектами. Пользовательский интерфейс интуитивно понятен, что делает его доступным для пользователей с разным уровнем опыта. Также предусмотрены настройки для изменения темы оформления интерфейса.

Система автозаполнения помогает ускорить процесс написания кода, предлагая подсказки на основе уже введенного текста. Это значительно снижает количество ошибок и увеличивает производительность разработчиков. Подсветка синтаксиса позволяет легко различать элементы кода, что улучшает читаемость и понимание написанного.

Кроме того, приложение включает систему управления историей изменений, которая позволяет пользователю откатываться к предыдущим версиям текста. Это особенно полезно в случае ошибок или необходимости вернуться к предыдущему состоянию работы.

Примеры сценариев использования приложения

Сценарий 1: Создание нового файла

Пользователь запускает приложение и выбирает опцию "Создать новый файл". Вводит текст и сохраняет его на диск. Приложение предлагает выбрать имя файла и его расположение.

file\_handler = FileHandler()

file\_handler.save\_file("new\_file.txt", "Это новый файл.")

Сценарий 2: Использование автозаполнения

Пользователь начинает вводить код на Python и замечает, что приложение предлагает варианты для автозаполнения.

autocomplete = AutoComplete()

suggestions = autocomplete.get\_suggestions("pri")

print(suggestions) # Вывод: ['print', 'private']

Сценарий 3: Подсветка синтаксиса

Пользователь открывает файл с кодом и видит, что синтаксис автоматически подсвечивается, что облегчает его понимание.

highlighter = SyntaxHighlighter()

highlighter.highlight\_syntax("def my\_function():\n print('Hello, World!')")

Проверка корректности созданного DocBook файла

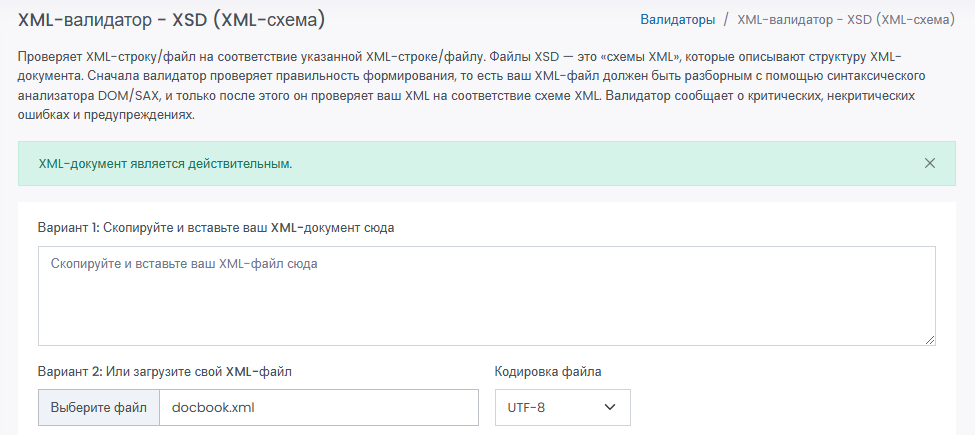


Рисунок 2 - Проверка корректности созданного DocBook файла

Ссылка на аккаунт: https://github.com/Vlad2257RUS

Ссылка на репозиторий: https://github.com/Vlad2257RUS/PPPIZosimovVVPI21b/tree/main