МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Донецкий национальный технический университет»

Факультет ИСП

Кафедра ПИ им Л.П.Фельдмана

Лабораторная работа № 4

на тему: «Создание самодокументирующегося кода»

по дисциплине «Профессиональная практика программной инженерии»

Выполнил: студент группы ПИ-20а

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Синяева Э.В.

(подпись) «\_\_\_» \_\_\_\_\_\_\_2021г.

Принял:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Незамова Л.В.

(подпись) «\_\_\_» \_\_\_\_\_\_\_2024г.

2024г.

**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.

1. Во всех файлах с исходным кодом добавить докблоки ко всем классам, методам, свойствам и функциям.

**hardware\_emulation.py**

class ЭмуляторОборудования:

"""

Класс для эмуляции работы аппаратного оборудования.

Attributes:

включено (bool): Флаг, указывающий, включено ли устройство.

кнопки (dict): Словарь, хранящий состояние каждой кнопки.

"""

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

"""

Функция инициализации эмулятора оборудования.

"""

включено = Ложь

кнопки = {}

def включить\_питание():

"""

Включает питание устройства.

"""

включено = Истина

def выключить\_питание():

"""

Выключает питание устройства.

"""

включено = Ложь

def нажать\_кнопку(кнопка):

"""

Нажимает указанную кнопку.

Args:

кнопка (str): Название кнопки, которую нужно нажать.

"""

если включено:

кнопки[кнопка] = Истина

вывести(f"Кнопка {кнопка} нажата.")

иначе:

вывести("Устройство не включено.")

def отпустить\_кнопку(кнопка):

"""

Отпускает указанную кнопку.

Args:

кнопка (str): Название кнопки, которую нужно отпустить.

"""

если включено:

кнопки[кнопка] = Ложь

вывести(f"Кнопка {кнопка} отпущена.")

иначе:

вывести("Устройство не включено.")

**user\_interface.py**

class ПользовательскийИнтерфейс:

"""

Класс для работы с пользовательским интерфейсом.

Attributes:

кнопки (dict): Словарь, хранящий состояние каждой кнопки.

"""

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

"""

Функция инициализации пользовательского интерфейса.

"""

кнопки = {}

def нарисовать\_кнопку(название\_кнопки):

"""

Рисует кнопку на экране.

Args:

название\_кнопки (str): Название кнопки для отображения.

"""

вывести(f"Кнопка '{название\_кнопки}' нарисована на экране.")

def нажать\_кнопку(название\_кнопки):

"""

Нажимает указанную кнопку.

Args:

название\_кнопки (str): Название кнопки, которую нужно нажать.

"""

кнопки[название\_кнопки] = Истина

вывести(f"Кнопка '{название\_кнопки}' нажата.")

def отпустить\_кнопку(название\_кнопки):

"""

Отпускает указанную кнопку.

Args:

название\_кнопки (str): Название кнопки, которую нужно отпустить.

"""

кнопки[название\_кнопки] = Ложь

вывести(f"Кнопка '{название\_кнопки}' отпущена.")

**settings\_management.py**

class МенеджерНастроек:

"""

Класс для управления настройками программы.

Attributes:

разрешение (tuple): Кортеж с шириной и высотой разрешения экрана.

громкость\_звука (int): Уровень громкости звука.

скорость\_эмуляции (float): Скорость эмуляции программы.

"""

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

"""

Функция инициализации менеджера настроек.

"""

разрешение = (800, 600)

громкость\_звука = 50

скорость\_эмуляции = 1.0

def установить\_разрешение(ширина, высота):

"""

Устанавливает разрешение экрана.

Args:

ширина (int): Ширина экрана.

высота (int): Высота экрана.

"""

разрешение = (ширина, высота)

вывести(f"Разрешение установлено на {ширина}x{высота}.")

def установить\_уровень\_звука(громкость):

"""

Устанавливает уровень громкости звука.

Args:

громкость (int): Уровень громкости звука.

"""

громкость\_звука = громкость

вывести(f"Уровень звука установлен на {громкость}.")

def установить\_скорость\_эмуляции(скорость):

"""

Устанавливает скорость эмуляции программы.

Args:

скорость (float): Скорость эмуляции программы.

"""

скорость\_эмуляции = скорость

вывести(f"Скорость эмуляции установлена на {скорость}x.")

2. Закоммитить изменения в репозиторий.

Прежде чем добавить докблоки в код проекта, необходимо убедиться, что все изменения готовы к коммиту. Для этого проверяем статус репозитория с помощью команды **git status** (Рисунок 1).

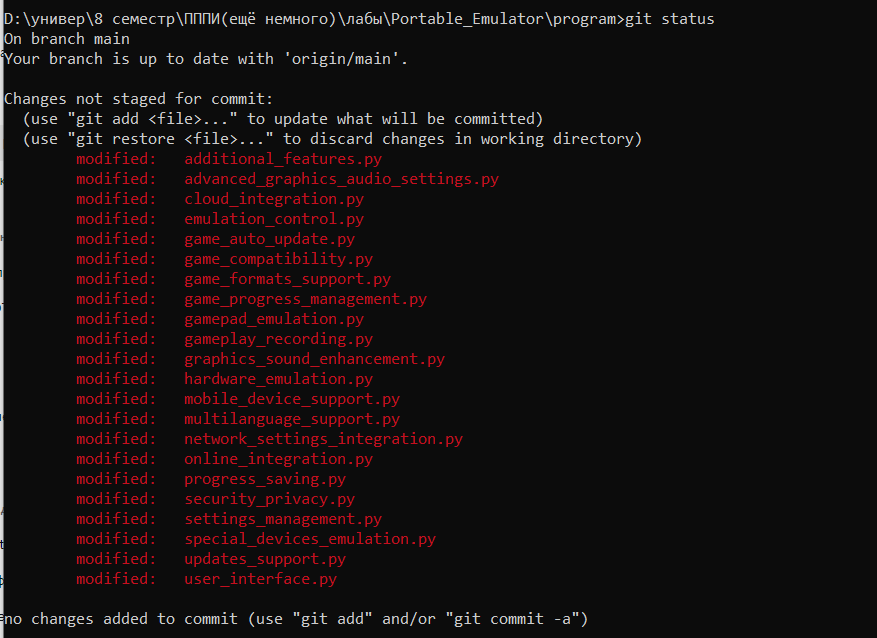


Рисунок 1 – Проверяем статус репозитория

Добавляем все изменения в индекс с помощью команды **git add**, чтобы подготовить их к коммиту (Рисунок 2).

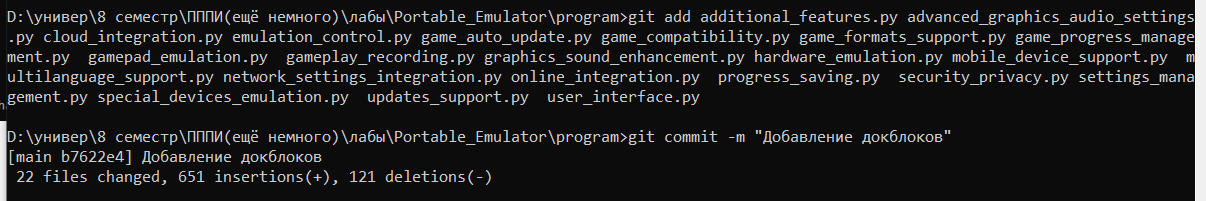


Рисунок 2 – Добавляем изменения

Создаем коммит с добавленными докблоками и сообщением "Добавление докблоков" с помощью команды **git commit -m "Добавление докблоков"** (Рисунок 3).

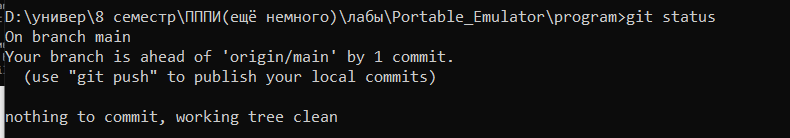


Рисунок 3 – Создаем коммит

Отправляем изменения в удаленный репозиторий с помощью команды **git push origin main** (Рисунок 4).

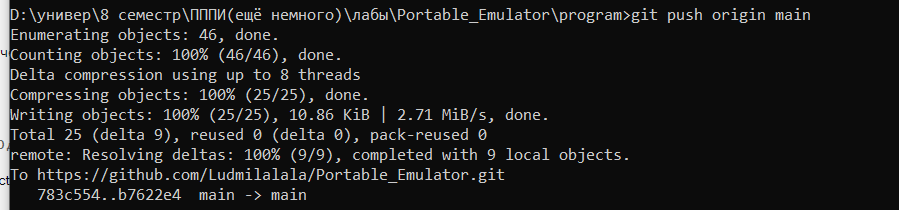


Рисунок 4 – Отправляем изменения

Для подтверждения успешной отправки изменений в удаленный репозиторий просмотрим коммит в онлайн репозитории (Рисунок 5)

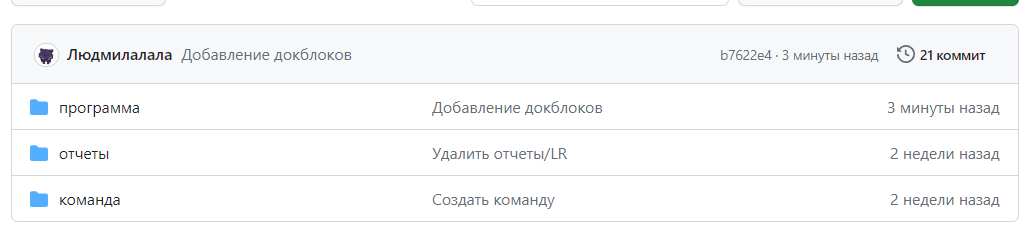


Рисунок 5 – Изменения в удаленном репозитории

3. Установить один из XML-редакторов (по желанию).

Для выполнения задачи был выбран XMLmind XML Editor. Этот редактор известен своей функциональностью и удобством использования для работы с различными типами XML-документов, включая DocBook.

4. Оформить краткое описание разработанного продукта (не менее пяти абзацев).

Скриншот оформления в XMLmind XML Editor (Рисунок 6).

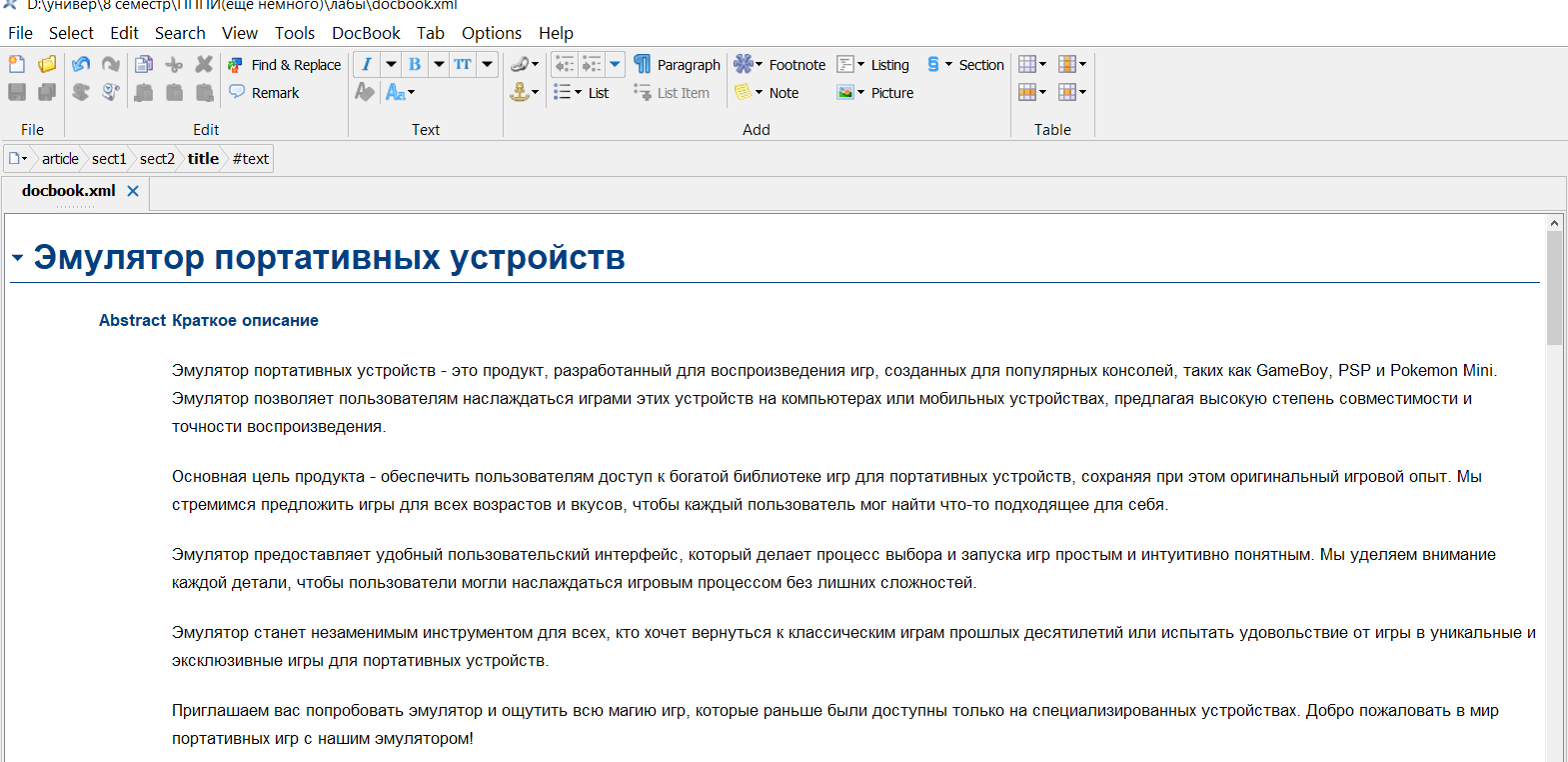


Рисунок 5 – Описание разрабатываемого продукта

Код и текст описания:

<?xml version="1.0"?>

<ns:clipboard

xmlns:ns="http://www.xmlmind.com/xmleditor/namespace/clipboard"

><para

>Эмулятор портативных устройств - это продукт, разработанный для воспроизведения игр, созданных для популярных консолей, таких как GameBoy, PSP и Pokemon Mini. Эмулятор позволяет пользователям наслаждаться играми этих устройств на компьютерах или мобильных устройствах, предлагая высокую степень совместимости и точности воспроизведения.</para

><para

>Основная цель продукта - обеспечить пользователям доступ к богатой библиотеке игр для портативных устройств, сохраняя при этом оригинальный игровой опыт. Мы стремимся предложить игры для всех возрастов и вкусов, чтобы каждый пользователь мог найти что-то подходящее для себя.</para

><para

>Эмулятор предоставляет удобный пользовательский интерфейс, который делает процесс выбора и запуска игр простым и интуитивно понятным. Мы уделяем внимание каждой детали, чтобы пользователи могли наслаждаться игровым процессом без лишних сложностей.</para

><para

>Эмулятор станет незаменимым инструментом для всех, кто хочет вернуться к классическим играм прошлых десятилетий или испытать удовольствие от игры в уникальные и эксклюзивные игры для портативных устройств.</para

><para

>Приглашаем вас попробовать эмулятор и ощутить всю магию игр, которые раньше были доступны только на специализированных устройствах. Добро пожаловать в мир портативных игр с нашим эмулятором!</para

></ns:clipboard

>

5. Рассмотреть с примерами кода не менее трех сценариев использования вашего приложения.

Рассматриваем сценарии использования ЭмуляторОборудования - hardware\_emulation.py, МенеджерНастроек - settings\_management.py, ПользовательскийИнтерфейс - user\_interface.py. (Рисунки 6 - ).

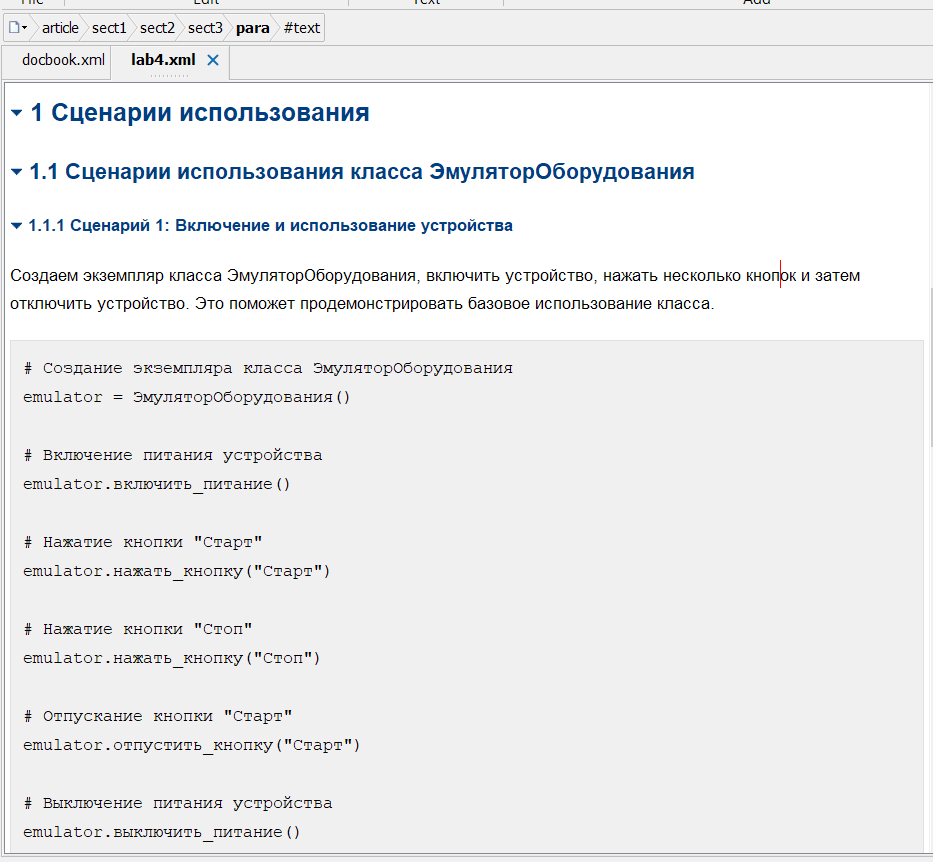


Рисунок 6 – ЭмуляторОборудования Сценарий1

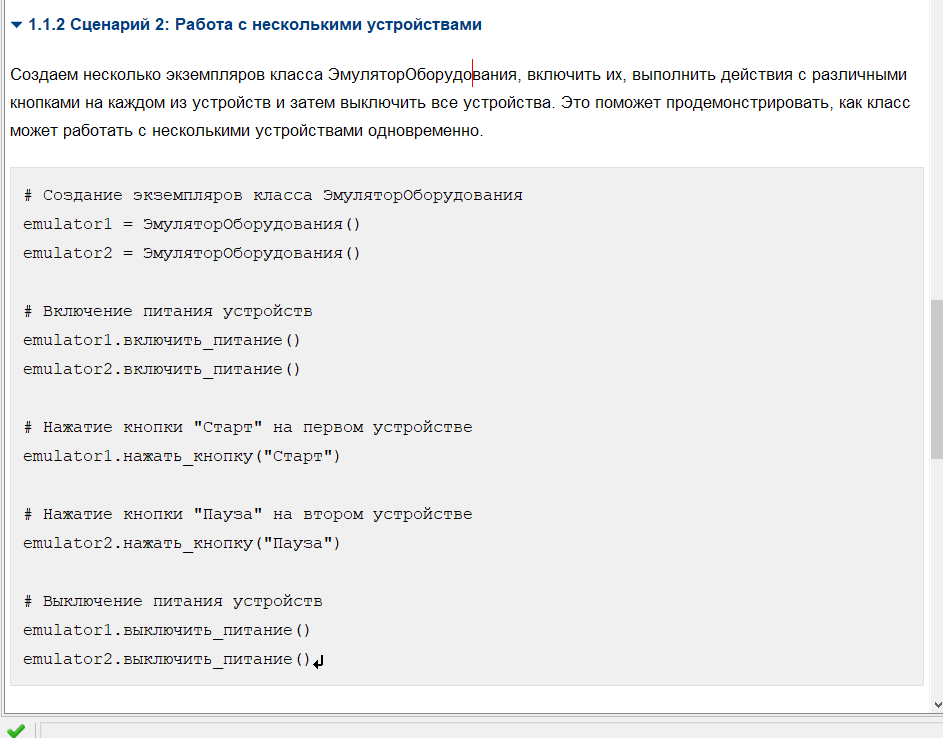


Рисунок 7 – ЭмуляторОборудования Сценарий2

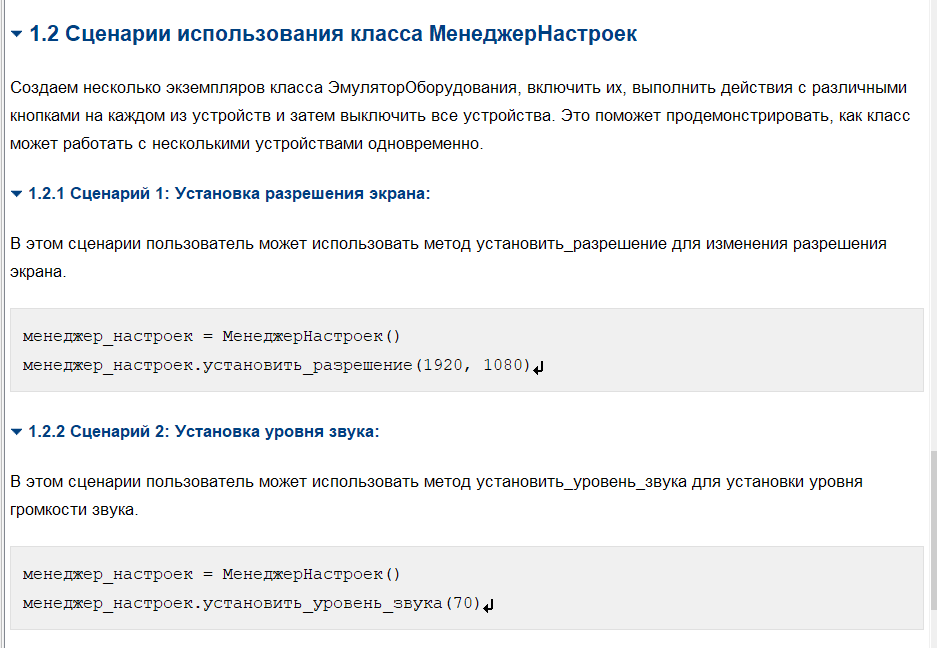


Рисунок 8 – МенеджерНастроек

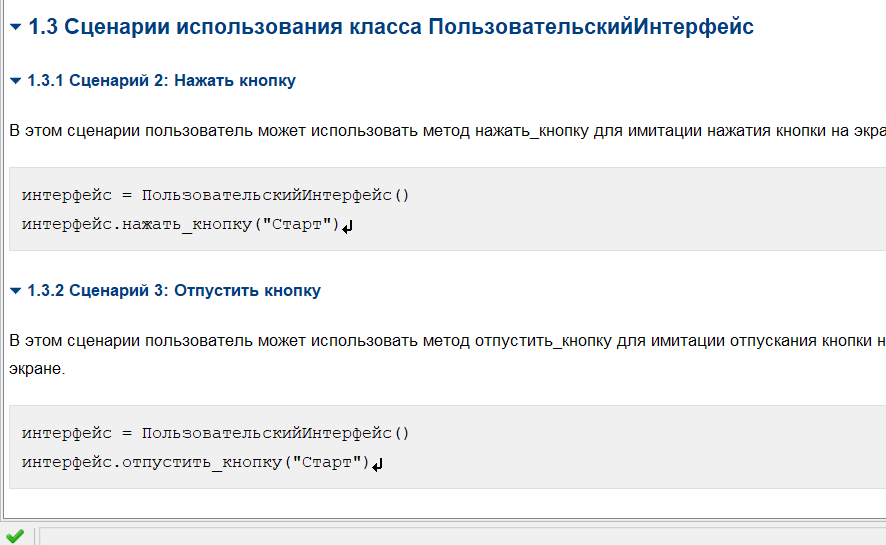


Рисунок 9 – ПользовательскийИнтерфейс

6. Проверить корректность созданного DocBook файла с помощью средств редактора или одного из онлайн валидаторов

Я использовала онлайн валидатор <https://www.freeformatter.com/xml-validator-xsd.html>

