МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

ФГБОУ «ДонНТУ»

Институт КНТ

Кафедра ПИ

Лабораторная работа №4

по курсу: «Профессиональная практика программной инженерии»

по теме: «Создание самодокументирующегося кода»

Выполнил:

ст. гр. ПИ-21а

Бунчук Д.Э.

Проверил:

асс. каф. ПИ

Филипишин Д.А.

Донецк 2024

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.

Задание по лабораторной работе

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

2. Закоммитить изменения в репозиторий.

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

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

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

6. Проверить корректность созданного DocBook файла с помощью средств редактора или одного из онлайн валидаторов, например, mashuosoft.com/docbook/validator

Содержание отчёта

1. Титульный лист.

2. Тема выбранного проекта.

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

4. Скриншот полученного справочного руководства.

5. Ссылка на аккаунт.

6. Ссылка на репозиторий.

7. Руководство высылать вместе с отчётом.

Вариант 14: Интерактивная книга

Ссылка на репозиторий:

<https://github.com/daniilbunc/InteractiveBook.git>

Ссылка на аккаунт:

<https://github.com/daniilbunc>

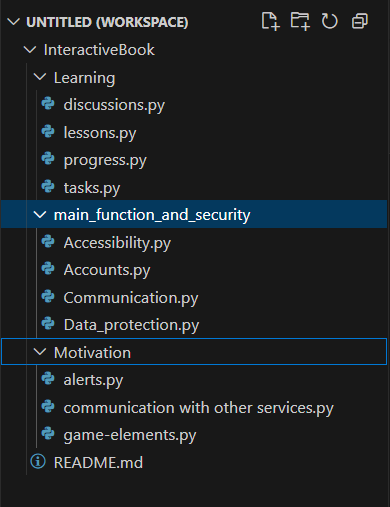


Рисунок 1- Структура проекта

Код с док блоками:

# lessons.py

class CourseManager:

"""Менеджер для создания курсов и управления уроками."""

def create\_course(self, title: str, description: str) -> int:

"""Создает новый учебный курс.

Args:

title (str): Название курса.

description (str): Описание курса.

Returns:

int: ID созданного курса.

"""

pass

def add\_lesson(self, course\_id: int, video\_url: str, text: str) -> int:

"""Добавляет урок к существующему курсу.

Args:

course\_id (int): ID курса.

video\_url (str): Ссылка на видео-материал.

text (str): Текстовое содержание урока.

Returns:

int: ID созданного урока.

"""

pass

class AssignmentManager:

"""Менеджер для работы с учебными заданиями."""

def submit\_test(self, user\_id: int, answers: dict) -> float:

"""Отправляет тест на проверку.

Args:

user\_id (int): ID пользователя.

answers (dict): Ответы пользователя в формате {question\_id: answer}.

Returns:

float: Результат теста в процентах.

"""

pass

def submit\_code(self, user\_id: int, code: str) -> str:

"""Проверяет код пользователя.

Args:

user\_id (int): ID пользователя.

code (str): Исходный код для проверки.

Returns:

str: Результат проверки (например, 'passed' или ошибки компиляции).

"""

pass

class ProgressTracker:

"""Отслеживает прогресс пользователя в курсах."""

def get\_progress(self, user\_id: int) -> dict:

"""Возвращает статистику по пройденным урокам.

Args:

user\_id (int): ID пользователя.

Returns:

dict: Статистика в формате {course\_id: {'completed': int, 'total': int}}.

"""

pass

def get\_recommendations(self, user\_id: int) -> list:

"""Генерирует рекомендации для пользователя.

Args:

user\_id (int): ID пользователя.

Returns:

list: Список ID уроков для повторения.

"""

pass

class DiscussionManager:

"""Управляет обсуждениями и вопросами к урокам."""

def post\_question(self, user\_id: int, lesson\_id: int, text: str) -> int:

"""Создает новый вопрос к уроку.

Args:

user\_id (int): ID пользователя.

lesson\_id (int): ID урока.

text (str): Текст вопроса.

Returns:

int: ID созданного вопроса.

"""

pass

def moderate\_message(self, message\_id: int) -> bool:

"""Модерирует сообщение (удаляет при нарушении правил).

Args:

message\_id (int): ID сообщения.

Returns:

bool: True если сообщение было удалено, иначе False.

"""

pass

class UserManager:

"""Управляет учетными записями пользователей."""

def register(self, email: str, password: str, social\_auth: dict = None) -> bool:

"""Регистрирует нового пользователя.

Args:

email (str): Email пользователя.

password (str): Пароль.

social\_auth (dict, optional): Данные соцсетей. Defaults to None.

Returns:

bool: True при успешной регистрации.

"""

pass

def login(self, email: str, password: str) -> bool:

"""Аутентифицирует пользователя.

Args:

email (str): Email пользователя.

password (str): Пароль.

Returns:

bool: True при успешном входе.

"""

pass

def update\_profile(self, user\_id: int, photo: str, name: str, role: str) -> bool:

"""Обновляет профиль пользователя.

Args:

user\_id (int): ID пользователя.

photo (str): Путь к фото.

name (str): Новое имя.

role (str): Роль (например, 'student' или 'teacher').

Returns:

bool: True при успешном обновлении.

"""

pass

# communication.py

class SecurityManager:

"""Обеспечивает безопасность данных."""

def encrypt\_password(self, password: str) -> str:

"""Шифрует пароль.

Args:

password (str): Пароль в открытом виде.

Returns:

str: Зашифрованный пароль.

"""

pass

def backup\_data(self, user\_id: int) -> bool:

"""Создает резервную копию данных пользователя.

Args:

user\_id (int): ID пользователя.

Returns:

bool: True при успешном резервном копировании.

"""

pass

class AccessibilityManager:

"""Управляет настройками доступности."""

def set\_font\_size(self, user\_id: int, size: str) -> bool:

"""Устанавливает размер шрифта.

Args:

user\_id (int): ID пользователя.

size (str): Размер ('small', 'medium', 'large').

Returns:

bool: True при успешном изменении.

"""

pass

def toggle\_dark\_mode(self, user\_id: int) -> bool:

"""Переключает темную тему.

Args:

user\_id (int): ID пользователя.

Returns:

bool: Новое состояние темы (True - темная, False - светлая).

"""

pass

# alerts.py

class GamificationManager:

"""Управляет игровыми элементами системы."""

def award\_badge(self, user\_id: int, badge\_type: str) -> bool:

"""Награждает пользователя значком.

Args:

user\_id (int): ID пользователя.

badge\_type (str): Тип значка ('bronze', 'silver', 'gold').

Returns:

bool: True при успешной выдаче.

"""

pass

def update\_leaderboard(self, user\_id: int) -> dict:

"""Обновляет рейтинг пользователя.

Args:

user\_id (int): ID пользователя.

Returns:

dict: Текущий рейтинг {user\_id: position}.

"""

pass

class NotificationManager:

"""Управляет уведомлениями."""

def send\_deadline\_reminder(self, user\_id: int) -> bool:

"""Отправляет напоминание о дедлайне.

Args:

user\_id (int): ID пользователя.

Returns:

bool: True при успешной отправке.

"""

pass

def toggle\_notifications(self, user\_id: int, is\_enabled: bool) -> bool:

"""Включает/выключает уведомления.

Args:

user\_id (int): ID пользователя.

is\_enabled (bool): Включить (True) или выключить (False).

Returns:

bool: Новое состояние уведомлений.

"""

pass

class SocialIntegration:

"""Обеспечивает интеграцию с соцсетями."""

def share\_achievement(self, user\_id: int, platform: str) -> bool:

"""Публикует достижение в соцсетях.

Args:

user\_id (int): ID пользователя.

platform (str): Платформа ('facebook', 'twitter', etc.).

Returns:

bool: True при успешной публикации.

"""

pass

**DokBook Документация**

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN" "http://www.oasis-open.org/docbook/xml/4.5/docbookx.dtd">

<book lang="ru">

<title>Интерактивный учебник - Полная документация</title>

<chapter id="accounts">

<title>Учётные записи и вход</title>

<section>

<title>Функциональность</title>

<itemizedlist>

<listitem><para>Создание аккаунта (через почту или соцсети)</para></listitem>

<listitem><para>Вход в систему, восстановление пароля</para></listitem>

<listitem><para>Настройки профиля: фото, имя, уровень доступа</para></listitem>

</itemizedlist>

</section>

<section>

<title>Сценарии использования</title>

<example>

<title>Регистрация нового пользователя</title>

<programlisting>

1. Пользователь нажимает "Зарегистрироваться"

2. Выбирает способ регистрации (email/соцсети)

3. Заполняет обязательные поля

4. Получает подтверждение на email

5. Активирует аккаунт по ссылке из письма

</programlisting>

</example>

</section>

</chapter>

<chapter id="courses">

<title>Учебные курсы и уроки</title>

<section>

<title>Функциональность</title>

<itemizedlist>

<listitem><para>Создание и структурирование курсов</para></listitem>

<listitem><para>Добавление мультимедийных уроков</para></listitem>

</itemizedlist>

</section>

<section>

<title>Сценарии использования</title>

<example>

<title>Создание нового курса преподавателем</title>

<programlisting>

1. Преподаватель заходит в раздел "Мои курсы"

2. Нажимает "Создать новый курс"

3. Заполняет метаданные (название, описание)

4. Добавляет модули и уроки

5. Публикует курс для студентов

</programlisting>

</example>

</section>

</chapter>

<!-- Аналогичные главы для других модулей -->

<chapter id="integration-scenarios">

<title>Комплексные сценарии</title>

<section>

<title>Полный цикл обучения</title>

<orderedlist>

<listitem>

<para>Регистрация и вход</para>

<itemizedlist>

<listitem><para>Создание аккаунта студента</para></listitem>

<listitem><para>Вход в систему</para></listitem>

</itemizedlist>

</listitem>

<listitem>

<para>Работа с курсом</para>

<itemizedlist>

<listitem><para>Выбор и прохождение уроков</para></listitem>

<listitem><para>Выполнение заданий</para></listitem>

</itemizedlist>

</listitem>

<listitem>

<para>Анализ результатов</para>

<itemizedlist>

<listitem><para>Просмотр прогресса</para></listitem>

<listitem><para>Получение рекомендаций</para></listitem>

</itemizedlist>

</listitem>

</orderedlist>

</section>

</chapter>

<appendix id="error-handling">

<title>Обработка ошибок</title>

<itemizedlist>

<listitem>

<para>Ошибка аутентификации</para>

<procedure>

<step><para>Проверка подключения к интернету</para></step>

<step><para>Повторная отправка запроса</para></step>

<step><para>Сброс пароля при необходимости</para></step>

</procedure>

</listitem>

</itemizedlist>

</appendix>

</book>

Интерактивный учебник - Полная документация

# Учётные записи и вход

## Функциональность

* Создание аккаунта (через почту или соцсети)
* Вход в систему, восстановление пароля
* Настройки профиля: фото, имя, уровень доступа

## Сценарии использования

Регистрация нового пользователя

1. Пользователь нажимает "Зарегистрироваться"  
2. Выбирает способ регистрации (email/соцсети)  
3. Заполняет обязательные поля  
4. Получает подтверждение на email  
5. Активирует аккаунт по ссылке из письма

# Учебные курсы и уроки

## Функциональность

* Создание и структурирование курсов
* Добавление мультимедийных уроков

## Сценарии использования

Создание нового курса преподавателем

1. Преподаватель заходит в раздел "Мои курсы"  
2. Нажимает "Создать новый курс"  
3. Заполняет метаданные (название, описание)  
4. Добавляет модули и уроки  
5. Публикует курс для студентов

# Комплексные сценарии

## Полный цикл обучения

1. Регистрация и вход
   * Создание аккаунта студента
   * Вход в систему
2. Работа с курсом
   * Выбор и прохождение уроков
   * Выполнение заданий
3. Анализ результатов
   * Просмотр прогресса
   * Получение рекомендаций

# Обработка ошибок

* Ошибка аутентификации
  1. Проверка подключения к интернету
  2. Повторная отправка запроса
  3. Сброс пароля при необходимости