**Общая логика проекта StoryTracker**

Основная идея StoryTracker – предоставить пользователю простой и удобный инструмент для отслеживания прогресса чтения книг и просмотра сериалов. Программа разделена на два основных направления: книги и сериалы, каждое из которых имеет собственный интерфейс и логику. Данные хранятся в базе данных PostgreSQL и загружаются при запуске приложения.

**Структура проекта**

Проект состоит из следующих основных модулей:

1. **Main.py (Главный модуль):**

Роль: Точка входа в приложение. Отвечает за создание главного окна и предоставление пользователю начального выбора.

Логика:

Создает главное окно с двумя кнопками: “Книги” и “Сериалы”.

При нажатии на кнопку “Книги” открывает окно BookPage.py.

При нажатии на кнопку “Сериалы” открывает окно MoviesPage.py.

После выбора одного из направлений окно Main.py закрывается.

Взаимодействие:

Запускает BookPage.py или MoviesPage.py в зависимости от выбора пользователя.

1. **BookPage.py (Модуль для книг):**

Роль: обеспечивает интерфейс и логику для работы со списком книг.

Логика:

Создает окно с полем ввода (название книги), кнопкой добавления и списком книг.

Позволяет пользователю вводить название книги и добавлять ее в список (с сохранением в БД).

Отображает список добавленных книг.

При нажатии на название книги, меняет статус (например, “прочитано”/” не прочитано”).

При нажатии клавиши “Del” на выбранном элементе списка, удаляет книгу из списка (и из БД).

Взаимодействие:

Использует *db\_connection.py* для работы с базой данных.

Обновляет отображение списка книг после изменений.

1. **MoviesPage.py (Модуль для сериалов):**

Роль: обеспечивает интерфейс и логику для работы со списком сериалов.

Логика:

Аналогична BookPage.py, но для работы с сериалами (поле ввода - название).

Позволяет добавлять сериалы в список (с сохранением в БД).

Отображает список добавленных сериалов.

При нажатии на название сериала, меняет статус (“просмотрено”/” не просмотрено”).

При нажатии клавиши “Del” на выбранном элементе списка, удаляет сериал из списка (и из БД).

Взаимодействие:

Использует *db\_connection.py* для работы с базой данных.

Обновляет отображение списка сериалов после изменений.

1. **db\_connection.py (Модуль для работы с базой данных):**

Роль: управляет соединением с базой данных PostgreSQL и выполняет запросы.

Логика:

Содержит функции для подключения к базе данных.

Предоставляет функции для сохранения, загрузки, изменения и удаления данных о книгах и сериалах.

Содержит SQL-запросы для взаимодействия с базой данных.

Взаимодействие:

Используется как *BookPage.py*, так и *MoviesPage.py* для доступа к данным.

**Взаимодействие модулей:**

Старт: Приложение запускается с Main.py.

Выбор: Пользователь выбирает либо “Книги”, либо “Сериалы”, что открывает соответствующее окно (*BookPage.py* или *MoviesPage.py*).

Данные: Выбранный модуль взаимодействует с *db\_connection.py* для сохранения, загрузки и обновления данных в PostgreSQL.

UI: Взаимодействие пользователя с интерфейсом (ввод данных, изменение статуса, удаление) приводит к вызову соответствующих функций, которые в свою очередь меняют данные в базе и обновляют интерфейс.

**Общая структура проекта (папки и файлы):**

StoryTracker/

├── main.py

├── bookpage.py

├── moviespage.py

└── db\_connection.py

**Ключевые концепции:**

**Модульность**: Каждый модуль имеет свою четкую роль и отвечает за определенную часть функциональности.

Разделение ответственности: Main.py управляет потоком приложения, BookPage.py и MoviesPage.py обрабатывают данные и интерфейс, db\_connection.py отвечает за базу данных.

**Использование PyQt5**: используется для создания графического интерфейса пользователя.

Работа с базой данных: *db\_connection.py* обеспечивает сохранение и извлечение данных через PostgreSQL.

**Преимущества такой структуры:**

Легче разрабатывать: Разделение на модули позволяет работать над частями приложения независимо.

Легче тестировать: Каждый модуль можно протестировать отдельно.

Легче поддерживать: Изменения в одном модуле меньше влияют на другие модули.

Масштабируемость: Добавление новых функций упрощается благодаря модульной структуре.

В целом, эта структура обеспечивает чистый, эффективный и масштабируемый способ создания приложения.