*Скворцов Алексей, Провоторин Лев*

Проект «ИЭБ: Интегрированная Электронная Библиотека»  
Пояснительная записка

*Описание с точки зрения пользователя*

Данный проект представляет собой систему для хранения информации о книгах и обмена книгами. На веб-сайте (yandexlyceum-web-project.herokuapp.com) можно добавлять книги и читать их. При помощи ботов ВКонтакте (vk.com/yandex\_iel) и Discord (IEL#8301) можно искать книги, получать информацию о книгах и скачивать тексты книг.

Без регистрации на сайте пользователь может просмотреть опубликованные книги и скачать какие-нибудь из них. При просмотре книги любой пользователь видит название, автора, жанр, ник добавившего пользователя, описание книги и приложенное изображение.

С регистрацией на сайте пользователь может просматривать, добавлять, редактировать и удалять книги, авторов и жанры. У книг, авторов и жанров присутствует статус, который ограничивает права пользователей. Статус «Принято» делает объект общедоступным (видят все, могут использовать все). Статусы «На модерации» и «Отклонено» делают объект видимым только создателю и модераторам (остальные не видят и не могут использовать). При этом если объект имеет статус «Отклонено», пользователь может его редактировать/удалить (это невозможно при других статусах).

Модераторы, помимо прочего, могут в любой момент редактировать объекты и изменять их статусы. Удалить объект модератор может не всегда: если это автор или жанр, который привязан хотя бы к одной книге, в удалении будет отказано (сначала его нужно отвязать, отредактировав или удалив книгу).

При помощи ботов можно осуществлять поиск по книгам, авторам и жанрам. Также можно получить информацию о любой общедоступной книге и скачать ее.

Все команды ботов начинаются со знака «минус». В этом случае бот пытается распознать команду и выполнить ее, другие сообщения боты игнорируют. На платформе ВКонтакте бота можно добавить в беседу: для этого он должен обладать правами администратора беседы.

Некоторые полезные функции, заметные не сразу

При удалении любого объекта запрашивается подтверждение удаления с возможностью вернуться обратно, не удаляя объект.

Формы поддерживают возвращение не на константную страницу, а на ту, с которой была открыта форма (это реализовано при помощи передачи в URL-е формы параметра from, который обработчики корректно распознают).

Если при создании или редактировании формы оставить поле «Описание» пустым, туда автоматически будет вставлена часть статьи Википедии о данной книге. Ссылка на статью будет также выведена в начале описания. Описание формируется при помощи API Википедии на основе названия книги, автора и жанра. Модератор перед публикацией книги может убрать описание, если оно не соответствует книге.

*Описание с точки зрения кода*

Проект изначально создавался под размещение на хостинге Heroku. С этим связаны некоторые его особенности.

1. Heroku не позволяет надолго размещать какие-то файлы, не загруженные при развертывании проекта. Поэтому при создании базы данных мы использовали не SQLite, а облачную базу на PostgreSQL. По этой же причине мы были вынуждены дублировать картинки и тексты в базе.
2. Хостинг ограничивает количество файлов программ, которые могут запускаться одновременно. Поэтому мы отказались от использования Longpoll API при работе с ВКонтакте и использовали Callback API. Это позволило встроить работу с ВК в веб-компоненту Flask.

Боты

Изначально при разработке ботов мы закладывали для них единый движок. Идея была такова: бот только получает сообщение и отправляет ответ, который ему передадут; «придумыватель ответов» расположен отдельно от хэндлеров сообщений и универсален для обоих ботов. Этот универсальный движок находится в файле bots\_infrastructure/bot\_engine.py. С этой же целью универсальности упрощено создание команд: каждая команда лежит в отдельном файле в папке bots\_infrastructure/commands и является объектом класса BotCommand. Ответы движка являются объектами класса BotAnswer.

Хэндлеры ботов расположены в папке bots\_handlers. Как уже написано выше, они практически не анализируют сообщения (происходит только проверка, что сообщение является командой, иначе сообщение просто игнорируется). При этом discord\_handler.py запускается как отдельный исполняемый файл, а vk\_handler.py импортируется программой main.py.

Данные

Весь код, отвечающий за непосредственную работу с БД, находится в папке data. В проекте используется библиотека sqlalchemy, что позволило нам легко адаптироваться к новой для нас базе на Postgres. Единственное неудобство заключается в доступе к БД: Heroku может без предупреждения менять логин и пароль от базы. На хостинге это отслеживается автоматически средствами самого хостинга, но для локального тестирования приходится сохранять отдельно строку доступа к базе, которая может внезапно стать неактуальной.

Скрипты обслуживания

Скрипты обслуживания находятся в папке maintenance\_scripts. Это служебные скрипты разработки, и они не предназначены для использования в боевых условиях. Два из них связаны с базой: один полностью очищает базу и заполняет ее тестовыми данными (был нужен в процессе разработки, когда ORM-модели часто менялись под задачи проекта), другой сбрасывает логин и пароль от базы (был нужен, когда в процессе разработки мы превышали количество допустимых подключений к базе, и нужно было сбросить подключения). Третий – скрипт для подсчета количества строк и файлов в проекте – был создан исключительно из любопытства. На данный момент в проекте 2011 строк (1402 – python, 600 – HTML, 9 – css) и 38 файлов кода.

Сайт

HTML-шаблоны хранятся в папке templates. Здесь находятся шаблоны всех страниц сайта. Мы в процессе разработки пытались оптимизировать количество файлов. Поэтому все страницы, отображающие списки книг, используют один шаблон помимо базового, с условиями внутри шаблона под разные страницы сайта. Аналогично с шаблоном «данных для книг» – авторов и жанров.

Немного о статических файлах. В папке css хранится единственный файл стилей, в папке img/books хранятся изображения книг, в папке txt/books хранятся тексты книг.

В папке web\_infrastructure находятся все обработчики страниц сайта. Для логической завершенности они разделены на blueprint-ы (технология, аналогичная модулям в Python, но конкретно для Flask). Также здесь находятся модели всех форм сайта (в файле forms\_models.py) и API-инфраструктура (файл book\_resource.py в подпапке API). API реализовано небольшим: идеология проекта не позволяет модифицировать данные без аутентификации на сайте, поэтому доступно только получение опубликованных книг (по одной или в виде списка) методом GET.

Основные файлы проекта

main.py – основной запускаемый файл, который отвечает за объединение всех blueprint-ов Flask в одно целое и за успешный запуск сервера. В том числе, он же импортирует vk\_handler.py, тем самым запуская ВК-бота (vk\_handler тоже содержит blueprint, т.к. должен получать POST-запросы от серверов ВКонтакте через одну из страниц сайта).

settings.py – файл настроек проекта. В этом файле содержатся все секретные ключи, используемые в проекте: ключ csrf, три ключа безопасности бота ВК, ключ бота Discord и строка прямого подключения к БД. Везде в коде необходимые данные импортируются из этого файла. Это упрощает контроль безопасности.

Файлы Procfile и requirements.txt – служебные файлы, необходимые для развертывания на хостинге. Procfile сообщает хостингу, какие из файлов следует запускать, а requirements.txt – какие сторонние библиотеки нужны для работы проекта.