Основная идея, реализованная в библиотеке, состоит в возможности формирования листов и книг Excel в виде react-компонента, при его отрисовке react сгенерирует xlsx документ и разместит в месте отрисовки кнопку для его скачивания.

Рассмотренная связка библиотек будет хорошей альтернативой применяемому в данный момент разработчиками информационной системы ВГТУ способу формирования отчётов на стороне серверной части с последующей его передачей через локальную сеть университета.

Список использованных источников

1. SheetJS [Электронный ресурс]. – Режим доступа: https://www.npmjs.com/package/xlsx. – Дата доступа: 14.04.2023
2. react-excel-workbook [Электронный ресурс]. – Режим доступа: https://softwareconnect.com/advanced-planning-scheduling/. – Дата доступа: 14.04.2023

УДК 004.415.25

РАЗРАБОТКА ПРИЛОЖЕНИЙ С REST API НА PYTHON

Антонова Т.А., студ., Бизюк А.Н., ст. преп.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье рассмотрены вопросы создания веб-приложений на языке Python. Язык Python очень активно развивается и технологии создания веб-приложений на этом языке имеют хорошие перспективы. Фреймворк Django значительно упрощает создание программных интерфейсов для веб-приложений.*

Ключевые слова: веб-сервис, REST, фреймворк, Python, Django.

Веб-сервис (англ. web service) – идентифицируемая веб-адресом программная система со стандартизированными интерфейсами. Веб-службы могут взаимодействовать друг с другом и со сторонними приложениями посредством сообщений, основанных на определённых протоколах (XML, JSON и т. д.). Веб-служба является единицей модульности при использовании сервис-ориентированной архитектуры приложения.

Одним из подходов создания веб сервиса является REST.

REST (сокр. англ. Representational State Transfer, «передача состояния представления») – стиль построения архитектуры распределенного приложения. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол (как и HTTP) должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ». Утверждается, что такой подход обеспечивает масштабируемость системы и позволяет ей эволюционировать с новыми требованиями [1].

На данный момент можно найти фреймворк для создания приложений в стиле REST практически для каждого языка программирования, используемого в веб-разработке.

На сегодняшний день наиболее функциональным фреймворком для создания веб-приложений на языке Python является фреймворк Django [2]. Django можно назвать MVC-фреймворком, так он реализует взаимодействие пользователя и системы:

- Model (хранит данные пользователя);
- View (отображает данные пользователя);
- Controller (принимает изменения данных от пользователя).

Внутри Django эта терминология звучит немного иначе (Model, View, Template), но суть остается той же.

Фреймворк не просто так пользуется популярностью среди бизнеса и разработчиков.

Преимущества использования Django:

- безопасность;
- защита от SQL-инъекций, межсайтового скриптинга (XSS) и подделки межсайтовых запросов (CSRF);

- хорошая масштабируемость;
- Django подключаемый по своей природе. Разработчики используют плагины для расширения веб-приложения;
- большой набор библиотек и модулей.

Одной из важных библиотек фреймворка является Django REST Framework (DRF) – это мощная и гибкая библиотека, отвечающая за создание API. Её главное преимущество в том, что она значительно упрощает сериализацию.

API должно поддерживать множество дополнительных возможностей, в частности:

- создание, чтение, изменение и удаление данных (CRUD);
- проверку корректности передаваемых данных и защиту от возможных хакерских атак;
- авторизацию и регистрацию пользователей;
- права доступа к данным через API.

Преимущество DRF не только в относительной простоте создания API сайта, но и в высокой скорости работы.

Принцип работы DRF:

На рисунке 1 продемонстрирован процесс обработки API-запроса фреймворком Django.

Рисунок 1 – Архитектура DRF

Вначале приходит запрос от клиента, который обрабатывается маршрутизатором фреймворка Django. Далее, с этим запросом связано одно из представлений, которое реализует API-сайта. Задачей представления является обработка запроса и отправка результата пользователю. Так как для обработки необходимо сформировать некоторые данные, как правило, в формате JSON, то для этого управление передается специальному объекту – сериализатору. Сериализаторы – это главные составляющие Django REST Framework. Именно они формируют данные для ответа на API-запросы, а также выполняют парсинг входной информации. Например, сериализатор может взять данные из таблиц БД и вернуть JSON-строку узлу, который отправил запрос.Также сериализатор может удалить или изменить данные в таблицах БД по определенному запросу.

В ходе работы с помощью Django и DRF был разработан API блога с аутентификацией и многими распространенными паттернами API. Имеются эндпоинты для получения, создания, обновления и удаления постов, категорий и комментариев, просмотра списка пользователей, их личных профилей. Также настроены отношения «многие-к-одному» и «многие-ко-многим» между этими ресурсами.

На рисунке 2 представлен веб-интерфейс, предоставляемый фреймворком Django для работы с API проекта.

Таким образом, DRF является полезной утилитой фреймворка Django. Он удовлетворяет всем требованиям к возможностям REST-систем, а также представляет из себя удобные инструменты его создания.

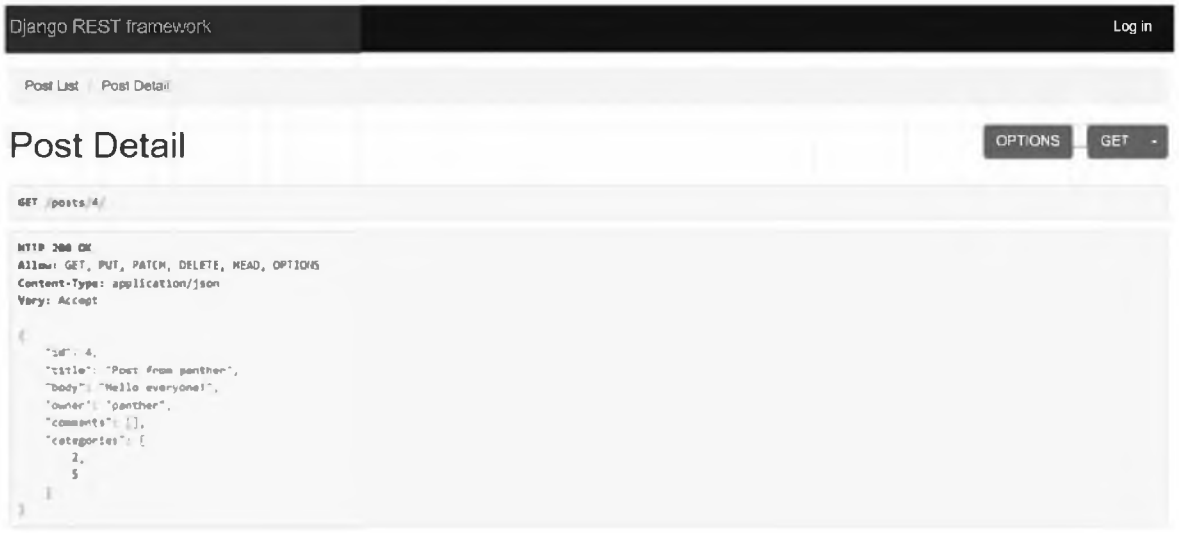

Рисунок 2 – Интерфейс фреймворка для тестирования API

Список использованных источников

1. Wilde E., Pautasso C. REST: From Research to Practice. Springer New York, 2011.
2. Bendoraitis A. Web Development with Django Cookbook: Over 70 Practical Recipes to Create Multilingual, Responsive, and Scalable Websites with Django. Packt Publishing Limited, 2014.

УДК 004.415.25

КОНТЕЙНИРИЗАЦИЯ И РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЙ С ПОМОЩЬЮ DOCKER И DOCKER-COMPOSE

Захарчёнок В.Ф., студ., Бизюк А.Н., ст. преп.
Витебский государственный технологический университет,
г. Витебск, Республика Беларусь

Реферат. *В статье рассмотрены вопросы использования Docker и docker-compose для разработки быстро разворачиваемых приложений. Разработаны Docker-файл и compose-файл для компиляции и запуска веб-приложения, использующего базу данных.*

Ключевые слова: веб-сервис, C#, Docker, doccker-compose.

Docker впервые увидел свет в 2011 году во Франции, когда Соломон Хайкс предвидел создание независимых от платформы механизмов для развертывания всех ваших двоичных файлов разработки таким образом, чтобы все библиотеки, зависимости уровня ОС и коды могли быть развернуты на любом компьютере. Облако или система, дающая разработчику уверенность в том, что его программное обеспечение будет работать должным образом, без каких-либо переделок и сбоев [1].

Основная причина концептуализации контейнера докеров заключалась в том, чтобы упростить жизнь разработчика. Настройка конфигурации рабочей среды для поддержки различных версий программного обеспечения на любой конкретной машине была большой проблемой. Но с докером вам просто нужно создать изолированный пакет, состоящий из всех конфигураций (называемых контейнером докеров), поддерживающих разные версии, и каждый контейнер здесь содержит отдельную конфигурацию изолированной среды, которая не влияет на конфигурации уровня ОС или любой другой работающий контейнер.

Чем отличается Docker от Виртуальных Машин?

Docker и виртуальные машины (VM) имеют одну общую цель – изолировать и запускать приложения. Однако у них есть ряд важных отличий: