ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»

Институт энергетики, информационных технологий и управляющих систем

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем

Расчётно-графическое задание  
по дисциплине: «Технологии web-программирования»

Выполнил: ст. группы ВТ-191

Федосеев Артём Дмитриевич

Проверил: Картамышев С.В.

Белгород 2022

Оглавление

[1. Дизайн веб-сайта 3](#_Toc122990907)

[2. Перенос проекта на движок Vue. 4](#_Toc122990908)

[3. Размещение в Docker 6](#_Toc122990909)

[4. Добавление функционала API 7](#_Toc122990910)

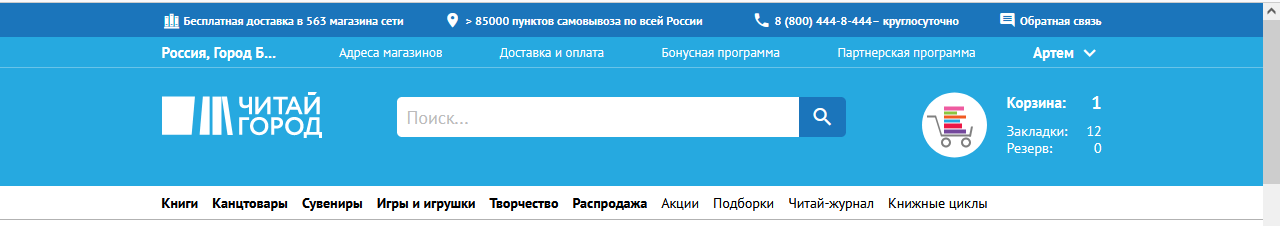
[5. База данных 9](#_Toc122990911)

# 1. Дизайн веб-сайта

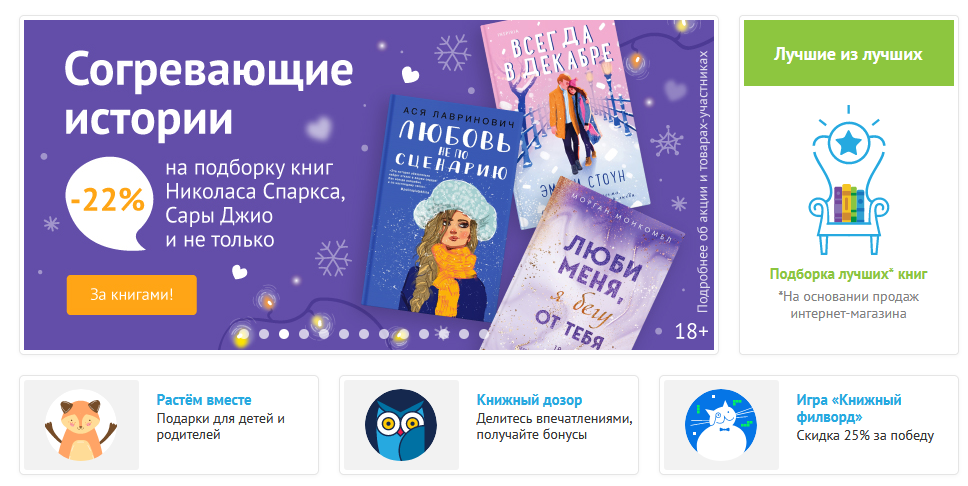
Первым шагом к созданию проекта является выбор тематики сайта, концепт дизайна страниц и элементов страниц.

Я решил разрабатывать сайт книжного магазина.

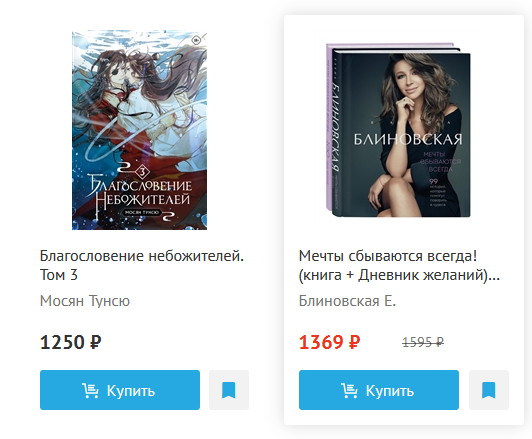
Концепт шапки страницы (остаётся на всех страницах сайта):



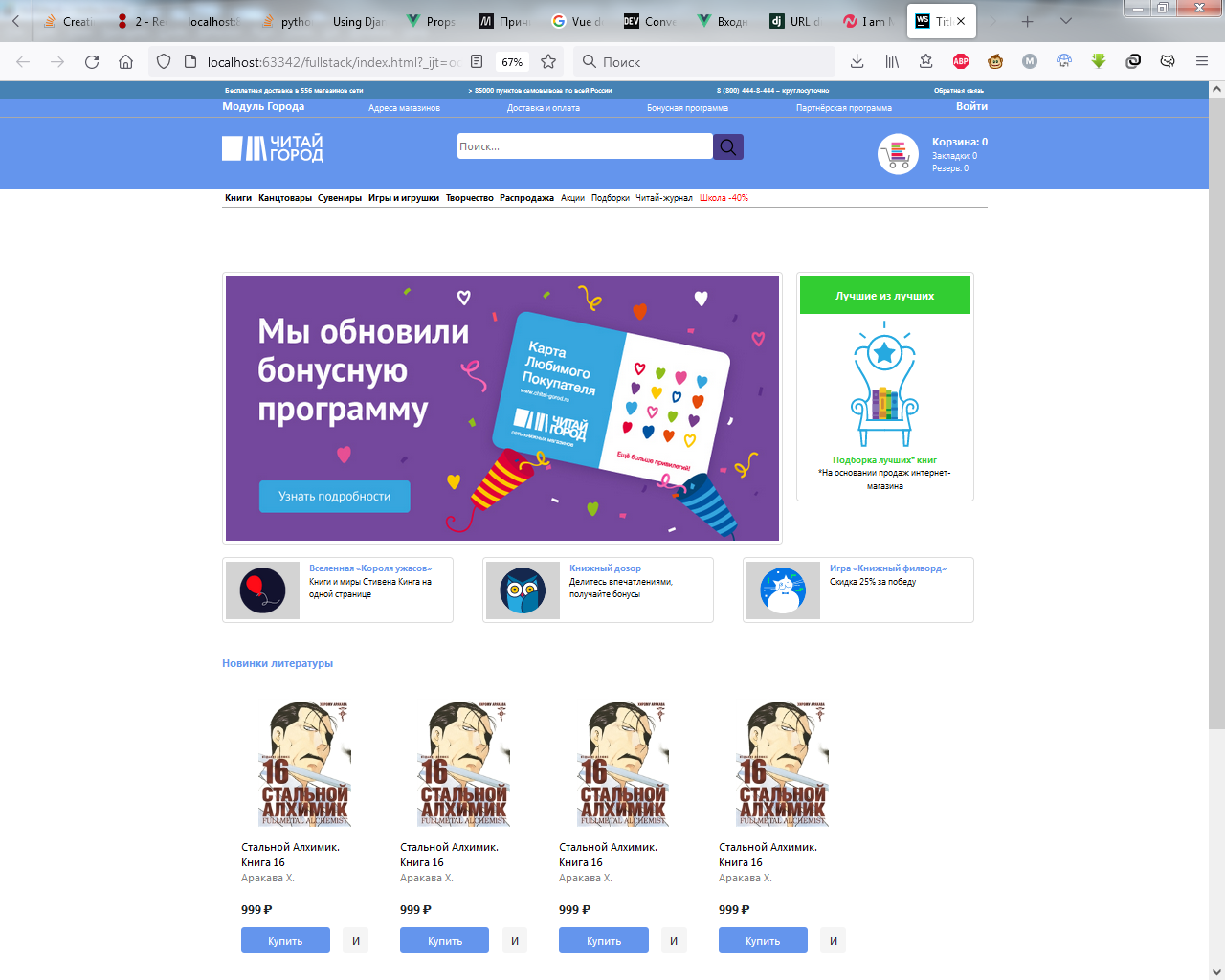
Концепт баннеров на главной странице:



Концепт карточки книги (справа - эффект при наведении):



Для начала, интерфейс сайта реализуется на HTML + CSS.

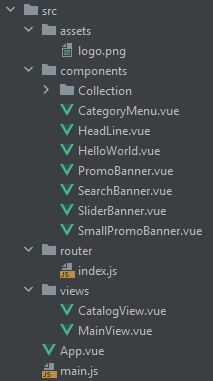


Исходный код можно найти по ссылке https://github.com/Stally18/fullstack

# 2. Перенос проекта на движок Vue.

Т.к. в настоящее время возможности одного только языка разметки HTML ограничены, необходимо воспользоваться движком для web-приложений. В его качестве был выбран Vue.

Vue - фреймворк, использующий в своей основе HTML + CSS, но сильно расширяющий их возможности посредством скриптов языка Javascript. Характеризуется очень лёгкой интеграцией в и с другими проектами, потому и был выбран.

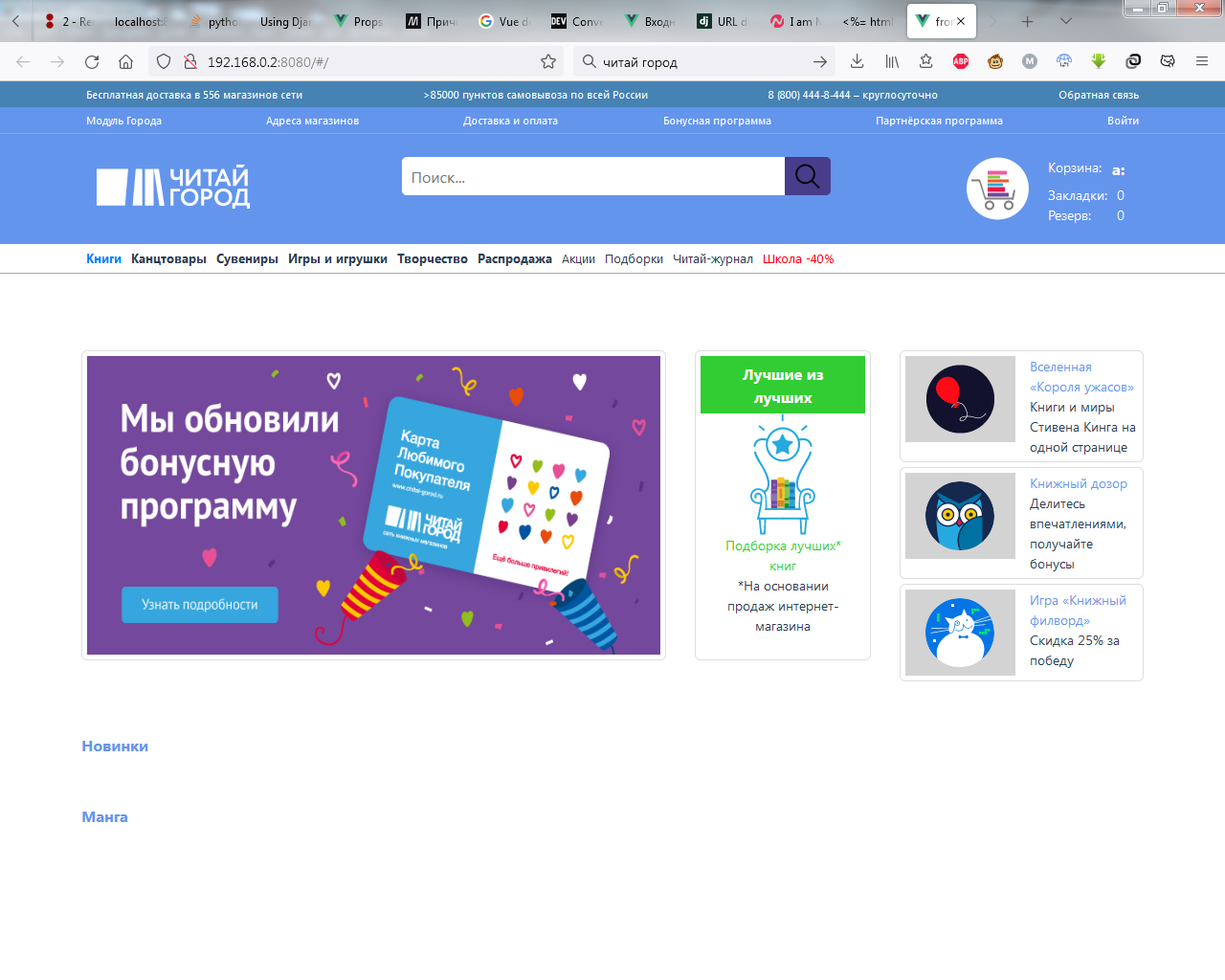
Структура Vue-приложения основывается на компонентах. Компонентом может являться как целая страница, так и некоторый контейнер, шаблон или небольшой элемент.

Компоненты состоят из трёх частей:

* template - шаблон, описывающий отображение содержимого посредством HTML и пользовательских тегов.
* script - раздел, включающий в себя код инициализации компонента, входных параметров, хранимых данных и/или пользовательских функций
* style - набор стилей, применяемых к этому (и, если указано, дочерним) компоненту.

Vue предоставляет возможность использовать компоненты в шаблонах других компонентов как пользовательские теги, превращая тем самым проект в иерархическую объектно-ориентированную систему.

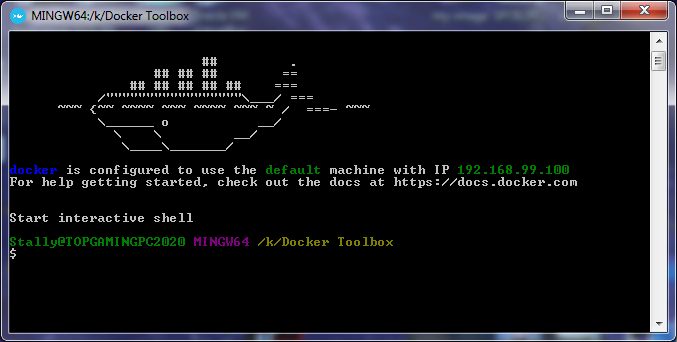
Также в проекте используется необязательный модуль vue-router, позволяющий настраивать перемещение по страницам нашего сайта путём замены контейнеров в определённом месте - таким образом, не нужно описывать и хранить отдельно целые страницы, а достаточно только заменять на одной странице компоненты.



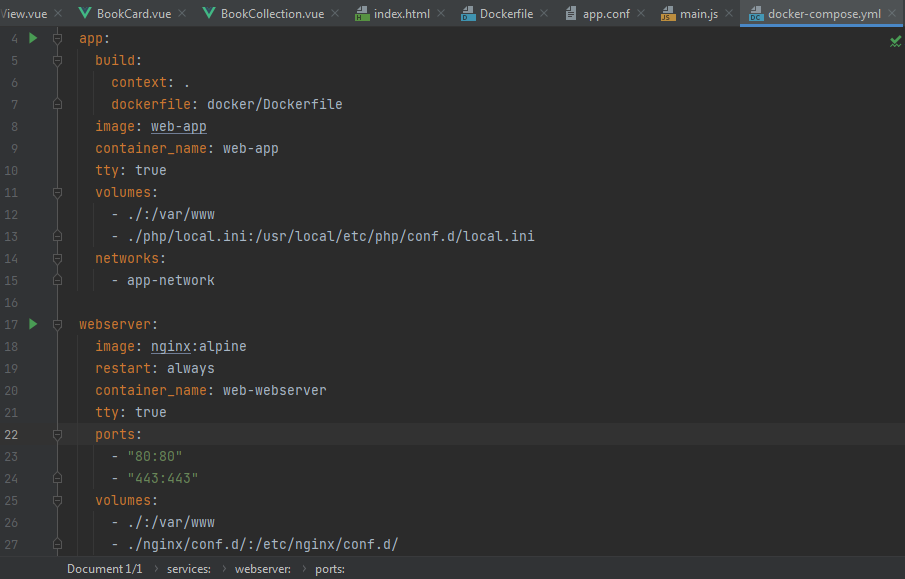
Исходный код проекта можно найти по ссылке https://github.com/Stally18/fullstackVue

# 3. Размещение в Docker

Следующим логичным шагом разработки является движение в сторону платформонезависимости. Для этой цели существует технология Docker - виртуальная машина, позволяющая быстро развернуть образ той или иной системы и запустить на ней исходные файлы проекта. Также Docker предоставляет мощные инструменты по интеграции приложений на разных языках и платформах друг с другом.

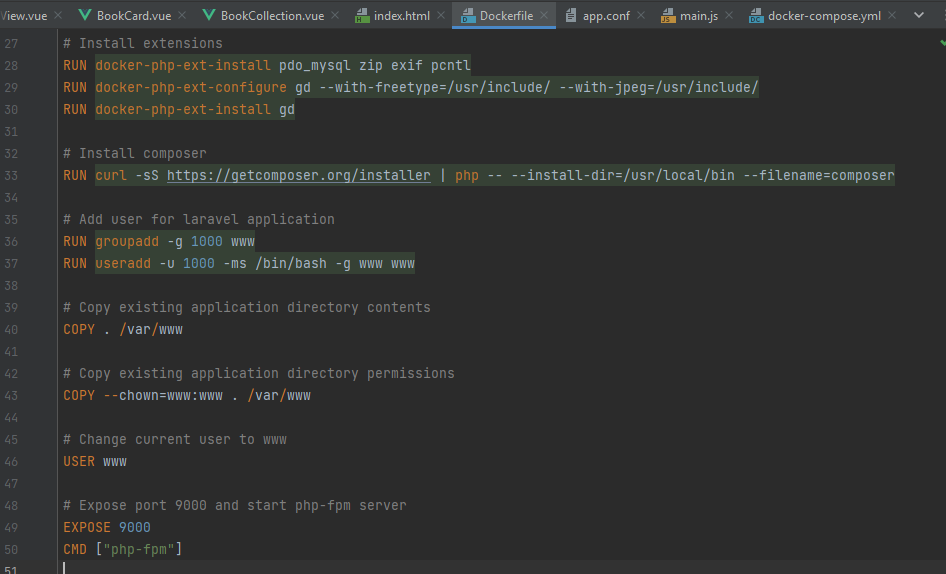


Инструментом разворачивания и связывания проектов является файл docker-compose. На данный момент вместе объединяются два приложения: сам сайт на Vue и вебсервер, обрабатывающий запросы и доступ к сайту на Nginx.



Приложение собирается с нуля на основе написанных нами исходных файлов, а сервер Nginx создаётся из стандартного образа библиотеки образов Docker. Настраиваются разделы диска, которые будут использоваться проектами, они монтируются, проецируя директории на хост-машине в указанные пути на виртуальной машине, тем самым позволяя нашему проекту быть в двух местах "одновременно".

После этого компилятор выполняет последовательность консольных команд, записанных в Dockerfile, и наш проект доступен из виртуальной машины.

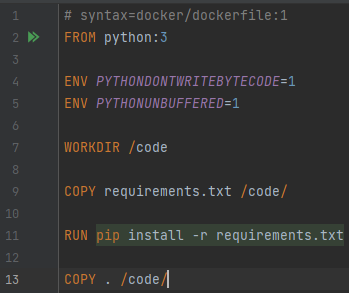
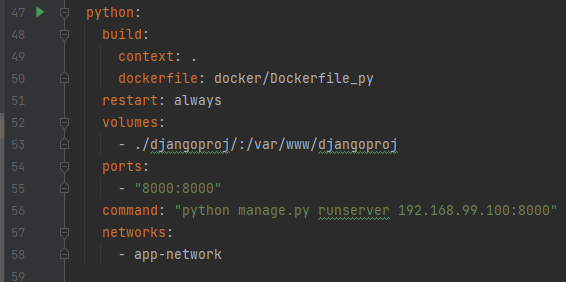


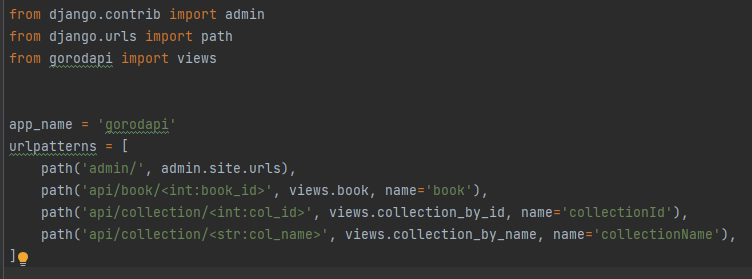
# 4. Добавление функционала API

Для полноценного взаимодействия с сайтом \ сайта с другими проектами или сервисами, сайту требуется API - интерфейс, предоставляющий определённый набор запросов, по которым можно обратиться к сайту или получить от него ответ. Функционал и возможности API определяется разработчиком.

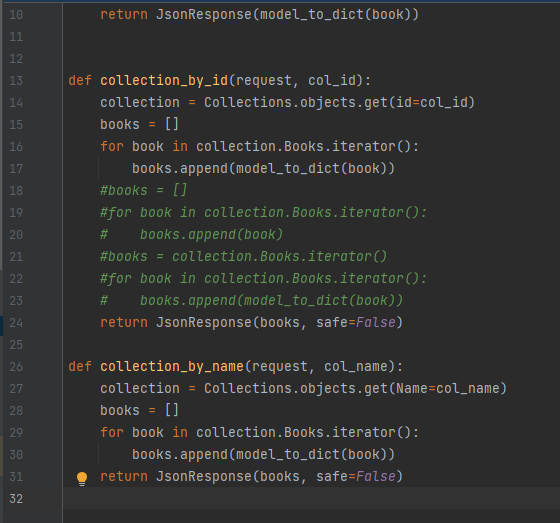
Для реализации API мною был выбран фреймворк Django. Django основан на языке Python и работает по модели MVC (MVT), но в нашем случае будут применяться только Модели (M) и Представления (V), т.к. за отображение уже отвечает Vue.

Для интеграции Django в проект мы дополняем файлы инициализации Docker:



Django позволяет задать набор путей, по которым можно обратиться к функциям API; пути могут иметь или не иметь параметры, возвращая статические или динамические данные. Состав, структура и содержимое данных определяется разработчиком.

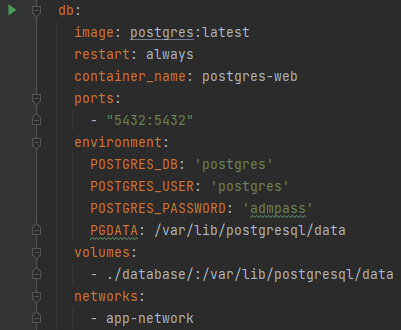


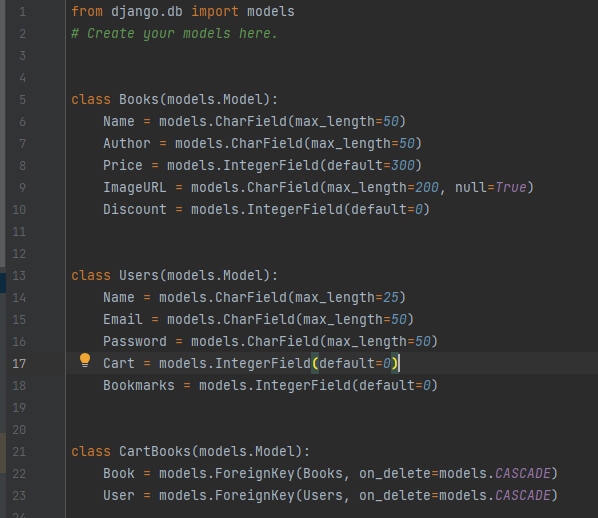
Далее, API запрос обрабатывается сервером Django и возвращает тот или иной ответ, в виде JSON, HTML или файла для скачивания.

Очевидно, чтобы возвращать посредством API какую-то информацию, необходимо её откуда-то взять. Что подводит нас к пункту

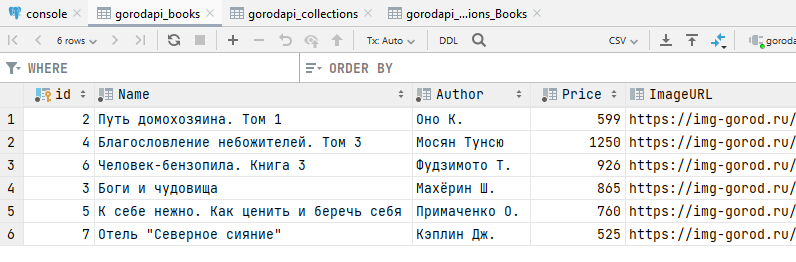
# 5. База данных

Django предоставляет инструменты для интеграции с той или иной базой данных. Мною была выбрана база данных на сервере PostgreSQL. Подключаем её в связку:





В библиотеках Django имеются инструменты для описания моделей сущностей, которые в последствии генерируются в используемой базе данных как таблицы, которые после этого можно заполнить данными.



В итоге, взаимодействие между всеми компонентами проекта строится следующим образом: в базе данных хранятся наборы данных, необходимые для размещения на сайте, Django позволяет сделать выборку данных из базы на основе определённого набора запросов (API) и передать их в формате JSON. Далее эти данные обрабатываются движком Vue и выводятся пользователю для просмотра и взаимодействия.

