МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ С ИСПОЛЬЗОВАНИЕМ ФРЕЙМВОРКА DJANGO

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Леоновой Полины Александровны

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ А. С. Иванов

Саратов 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире веб-приложения стремительно развиваются и становятся одной из важнейших частей бизнеса. Они являются визитной карточкой компании, позволяют осуществлять продажи круглосуточно из любого удобного места, что способствует легкому развитию малого бизнеса путем расширения аудитории покупателей.

Целью бакалаврской работы является разработка веб-приложения для магазина ягодных букетов, которое упростит процессы оформления заказов, сбора и хранения данных покупателей, а также предоставит возможность удобного добавления новых товаров и их отображение в пользовательском интерфейсе.

Предполагается, что на главной странице будет размещен каталог товаров, разделенных на категории. У каждого товара будет собственная страница, на которой представлены его фотографии, название, цена и описание. Вместо регистрации покупателю просто нужно будет ввести личные данные при оформлении заказа на соответствующей странице. Добавлять товары в корзину можно будет как с главной страницы, так и со страницы самого товара. С любой страницы будет доступ к просмотру содержимого корзины и удалению товаров из нее.

Необходимо создать приложение, удовлетворяющее следующим требованиям:

- сотрудник имеет доступ к информации о подписчиках, товарах и заказах, а также возможность ее редактирования;
- покупатель может добавлять товары в корзину и делать заказы;
- у товара должны быть наименование, цена, описание и категория;
- товар может иметь неактивное состояние (быть скрытым от покупателя);
- возможность добавления скидки и фотографий для товара;
- фотографии товара можно добавлять при его редактировании;
- у заказа должны быть сумма, адрес доставки, способ оплаты и статус;
- в заказе должна храниться информация об имени, телефоне и электронной почте покупателя;
- при добавлении товара в заказе должна автоматически пересчитывается его сумма;
- в приложении должна быть страница сбора информации потенциальных

покупателей до момента открытия интернет-магазина;

- сохранять данные подписчика при их отправке из формы на странице лендинга;
- наличие домашней страницы с отображением всех товаров, возможность добавления их в корзину;
- у каждого товара должна быть отдельная страница с подробным описанием, информацией о доставке и оплате;
- при обновлении страницы содержимое корзины не должно обнуляться;
- наличие страницы оформления заказа с возможностью изменения содержимого корзины и автоматическим подсчетом общей суммы заказа;
- при оформлении заказа передавать его в базу данных.

Для достижения поставленной цели требуется решить ряд задач:

- описать модели данных и отношения между ними;
- спроектировать и разработать веб-приложение.

Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и семи приложений.

В первом разделе «Средства разработки и архитектура приложения» представлен обзор средств разработки, использованных при реализации веб-приложения, описаны модели данных и отношения между ними, полностью удовлетворяющие указанным требованиям, а также приведена схема базы данных приложения.

Второй раздел «Процесс разработки приложения» содержит описание полученного продукта, как серверной части, так и пользовательской. Также в разделе продемонстрированы примеры работы приложения и описаны детали его реализации.

1 Средства разработки и архитектура приложения

1.1 Средства разработки

В данном разделе рассматриваются некоторые средства и технологии, применяемые для разработки данного проекта. Заметим, что при их выборе учитывались популярность и необходимость изучения технологий до начала работы, разработка проекта за небольшой срок, а также возможность дальнейшего расширения функционала.

Было решено использовать фреймворк [1], так как с его помощью можно достичь высокой производительности и надежности решений при большой скорости разработки. Как правило, фреймворк включает только базовые программные модули, а разработчик на их основе реализует все специфичные компоненты проекта. Поскольку в качестве языка разработки выбран Python [2], рассматривались такие популярные фреймворки, как Django и Flask.

В результате принято решение использовать Django [3]. Основными его преимуществами является следующее:

— Django доступен со встроенными формами, которые интегрируются с ORM и панелью администратора;
— Django доступен со встроенной ORM и системой миграции, которая может управлять базами данных;
— Django предоставляет приложение аутентификации, которое предоставляет реализацию по умолчанию для пользовательского управления и привилегий;
— Django включает в себя полностью интегрированный админ-интерфейс для управления данными приложения [4].

Для хранения данных выбрана SQLite [5]. Она очень популярна и проста в использовании, к тому же имеет полностью свободную лицензию. Это отличная встраиваемая БД, предоставляющая высокую скорость и производительность операций, удобное перемещение базы, поскольку вся база данных хранится в одном файле. Также SQLite является кроссплатформенной, надежной, поддерживает стандарты SQL и использует динамическое типизирование данных [6].

Создание пользовательского интерфейса решено делать с помощью фреймворка Bootstrap [7]. Он очень популярен и используется разработчиками по всему миру, так как позволяет в несколько раз ускорить процесс разработки. Этот

фреймворк представляет собой набор CSS и JavaScript файлов, необходимо просто подключить их к странице. Можно выделить следующие достоинства Bootstrap:

- высокая скорость разработки;
- кроссбраузерность и кроссплатформенность (корректное отображение и работа сайта во всех поддерживаемых этим фреймворком браузерах и операционных системах);
- открытый исходный код, позволяющий изучать библиотеки и изменять их под собственный проект;
- низкий порог вхождения (для работы с фреймворком не обязательно иметь глубокие знания по HTML, CSS, JavaScript и jQuery, достаточно знать только основы вышеперечисленных технологий) [8].

1.2 Описание моделей и их взаимосвязей

Модель – это объект Python, через который в Django веб-приложение получает доступ к данным и управляет ими. Она определяет структуру хранимых данных. Благодаря моделям не нужно работать напрямую с базой данных [9].

При проектировании удобно создавать несколько отдельных моделей для каждого объекта, связанных определенным образом между собой.

Были выделены основные объекты системы, необходимые для покрытия всех требований к веб-приложению и его удобной эксплуатации:

- объект «Подписчик»;
- объект «Товар»;
- объект «Заказ».

1.2.1 Модели объекта «Подписчик»

До момента открытия интернет-магазина планируется наличие страницы, где потенциальный покупатель сможет подписаться на новостную рассылку.

Для этого необходимо создать модель «Подписчик». Очевидно, что у данной модели должны быть поля для имени и почты подписчика.

Поскольку страница сбора информации будет существовать лишь до открытия магазина, данная модель никак не будет связана с другими.

1.2.2 Модели объекта «Товар»

Для товара планируется создание следующих моделей:

— «Товар»;
— «Категория товара»;
— «Фотографии товара».

Проанализировав требования к проекту, сделан вывод, какие поля необходимо добавить к моделям. У товара должны быть поля с именем, ценой, категорией. Требуется добавить поле скидки для дальнейшего информирования покупателя о том, что она распространяется на тот или иной товар. Для отображения товара на главной странице необходимо краткое описание, а на странице самого товара более подробное описание. Данные поля также нужно добавить в модель товара. Чтобы была возможность скрыть товар от покупателя (например, при его отсутствии) и не удалять его при этом из базы, также можно добавить отдельное поле. Для удобства сотрудников предложено добавить поля с датами создания и последнего обновления товара.

У модели категории товара будет поле с названием категории и состоянием (активным или нет).

Для модели фотографии товара нужна возможность загрузки фотографии, установка главного фото на товар (для отображения на главной странице), поля состояния, создания и обновления.

Понятно, что у каждого товара должна быть категория, следовательно, товар должен ссылаться на нее.

Наоборот, каждое фото должно соответствовать какому-либо товару, то есть ссылаться на него.

1.2.3 Модели объекта «Заказ»

Для заказа планируется создание следующих моделей:

— «Заказ»;
— «Способ оплаты»;
— «Статус заказа»;
— «Товар заказа»;
— «Товар в корзине».

В соответствии с вышеизложенными требованиями, у заказа необходимо добавить поля:

— сумма заказа;
— имя, почта, телефон, адрес покупателя;
— способ оплаты;
— комментарий;
— статус;
— даты создания и редактирования заказа.

Для статуса заказа нужны только поля имени, состояния, создания и обновления. Такие же поля требуются для способа оплаты.

В моделях товар заказа и товар в корзине будут поля заказа и продукта, к которым они относятся, количество, цена, сумма. Также поля состояния, создания и обновления.

Так как у каждого заказа есть статус и способ оплаты, они будут связаны с заказом.

На заказ должны ссылаться товары заказа и товары в корзине. Очевидно, что они также должны быть связаны с моделью товара.

1.3 Схема базы данных

Схема базы данных — это наглядное представление логической конфигурации этой БД или ее части. Схема показывает, как сущности связаны между собой. Процесс создания схемы базы данных называется моделированием.

Реляционная модель сортирует данные в таблицы (отношения), каждая из которых состоит из столбцов и строк. В каждом столбце указан атрибут рассматриваемой сущности, например, название, цена или скидка товара. Определенный атрибут выбирается в качестве первичного ключа, на который можно ссылаться в других таблицах, когда он называется внешним ключом. Каждая строка, также называемая кортежем, содержит данные о конкретном экземпляре рассматриваемой сущности, например, конкретный товар [10].

Модель также учитывает типы отношений между таблицами, включая отношения «один к одному», «один ко многим» и «многие ко многим». Например, между статусом и заказом должна быть связь «один к одному».

Исходя из описания моделей и взаимосвязей между ними, можно изобразить схему базы данных (рис. 1).

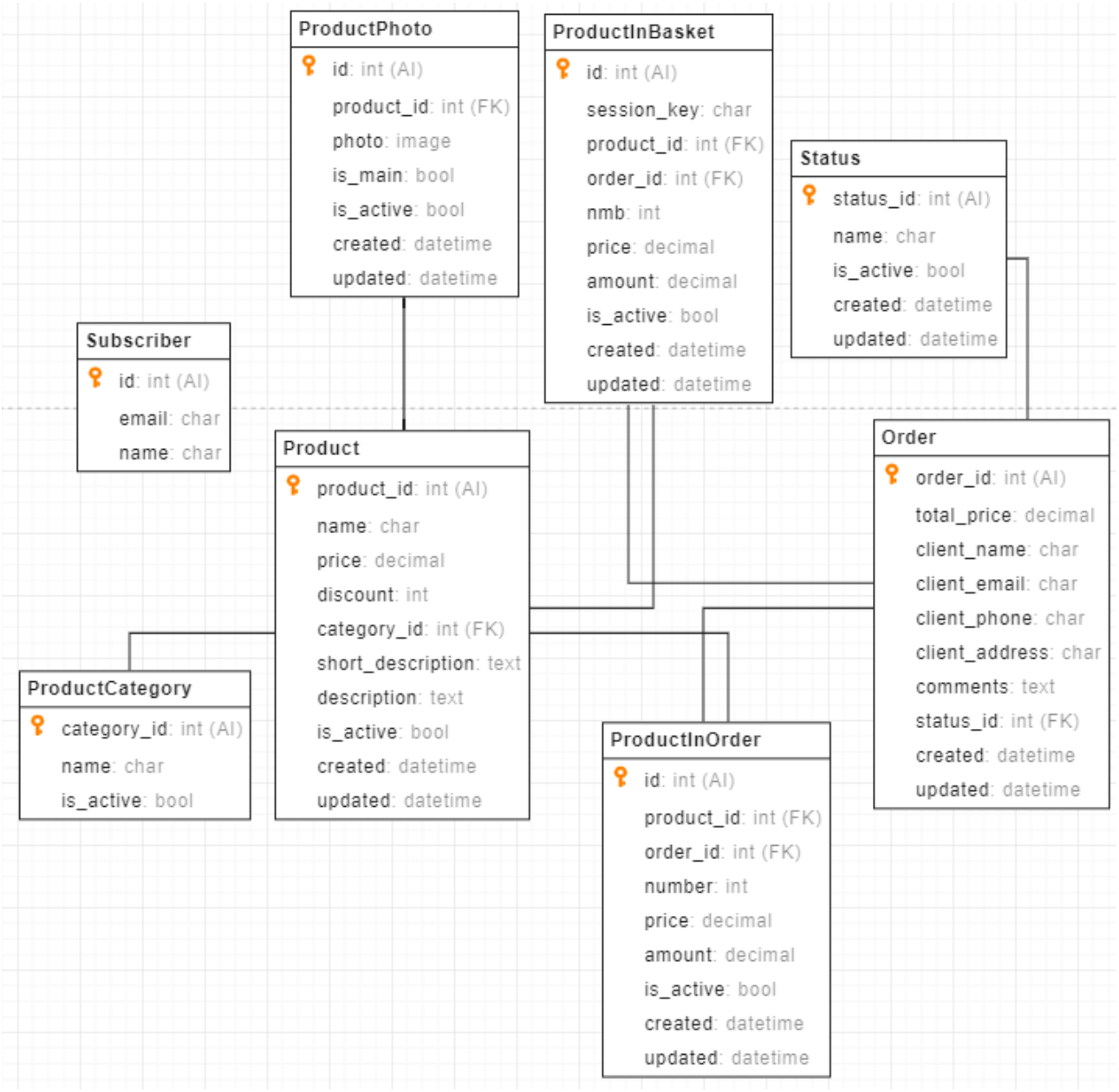

Рисунок 1 – Схема базы данных

1.4 Backend- и frontend-разработка

В разделе описывается, что такое backend и frontend и за что они отвечают.

Современная разработка — это сложный многоуровневый процесс, который можно разделить на серверную (backend) и пользовательскую (frontend) части.

Серверная часть отвечает за логику приложения и его правильное функционирование. Она скрыта от пользователя. Лишь администратор может управлять ей через специальный интерфейс [11].

Цель backend-разработки в реализации серверной стороны приложения,

интеграция базы данных и связь ее с пользовательской частью.

Frontend-разработка отвечает за пользовательскую часть. Перед разработчиком стоит задача создать понятный и удобный в использовании интерфейс. Помимо дизайна frontend отвечает за взаимодействовие с пользователем.

Когда пользователь делает запрос, он передается на сервер, сначала на сервер пользователя — frontend. Запрос обрабатывается и отправляется на свободный backend-сервер. Тот в свою очередь тоже обрабатывает его, обращается к базе данных и посылает ответ обратно frontend-серверу, который уже отображает ответ пользователю в виде HTML-страницы [11].

Рассмотрим инструменты, которые требуются для создания клиентской части сайта.

Язык гипертекстовой разметки HTML отвечает за содержание сайта, наполнение его информацией и ее расположение на страницах. За оформление сайта и отображение HTML-документа отвечает CSS.

За интерактивность сайта отвечает язык программирования JavaScript. Он дает возможность реализации сложного поведения страницы, например, добавление товара в корзину по нажатию кнопки. Фреймворк этого языка jQuery — библиотека с набором готовых функций, которые упрощают разработку [11].

Таким образом, проект состоит из двух частей:

— интерфейс администратора, который позволит сотрудникам добавлять, удалять или изменять информацию о товарах и заказах;
— сайт, который будет отображать покупателям информацию о товарах и позволит им оформлять заказы.

2 Процесс разработки приложения

В данном разделе бакалаврской работы описывается процесс разработки веб-приложения, приводятся примеры его использования и детали реализации. Описана функциональность приложения, изложенная в требованиях.

2.1 Разработка серверной части

При разработке серверной части был использован автоматический интерфейс администратора Django, который предоставляет готовый к использованию интерфейс для работы с содержимым веб-приложения, что поможет сотрудникам магазина редактировать данные.

Рассмотрены создание первого пользователя и вид главной страницы админ-интерфейса.

На примере подписчика показан процесс создания моделей, их сохранение в базе данных и отображение в интерфейсе администратора. Представлено отображение подписчиков в интерфейсе администратора и вид страницы редактирования данных подписчика.

В разделе рассмотрены добавление формы для передачи введенных данных в созданную модель и функция для сохранения данных о подписчике, в которой принимаются данные с веб-страницы и сохраняются в новую форму.

Также рассмотрены типы полей данных в моделях и их параметры. В качестве примера их использования приведена модель товара. Для объекта «Товар» представлен вид каталога товаров и его редактирование. Описано как можно редактировать связанные объекты на странице родителя на примере добавления фотографий при редактировании самого товара.

В разделе приведено описание функционала приложения для моделей объекта «Заказ»:

- автоматический пересчет суммы заказа при изменении в нем количества товара;
- сохранение информации о содержимом корзины для текущего пользователя;
- функция для пересчета и получения информации о товарах в корзине;
- функция для формирования заказа.

Представлено как выглядят страницы заказов и редактирования заказа, а также отображение товаров корзины в интерфейсе администратора.

2.2 Разработка пользовательской части

На примере страницы лендинга рассмотрено как происходит взаимодействие между браузером и Django, а также процесс создания и внешний вид страницы сбора информации.

Рассмотрены шаблоны в Django и их особенности, создание базового шаблона и использованные в нем теги, подключение статических файлов. Показано каким образом определяются дочерние шаблоны.

В разделе описан процесс создания навигационной панели сайта и представлен ее вид при разных размерах окна браузера.

Аналогично описан процесс создания главной страницы, где покупатель может просматривать каталог товаров, добавлять их в заказ и просматривать содержимое корзины.

Представлен внешний вид главной страницы при разных размерах окна браузера.

На странице товара расположены его фотографии, название, цена и описание, а также информация о доставке и оплате. Добавлена форма с указанием количества товара и кнопкой добавления в корзину. Внешний вид страницы товара представлен на рисунке 2.

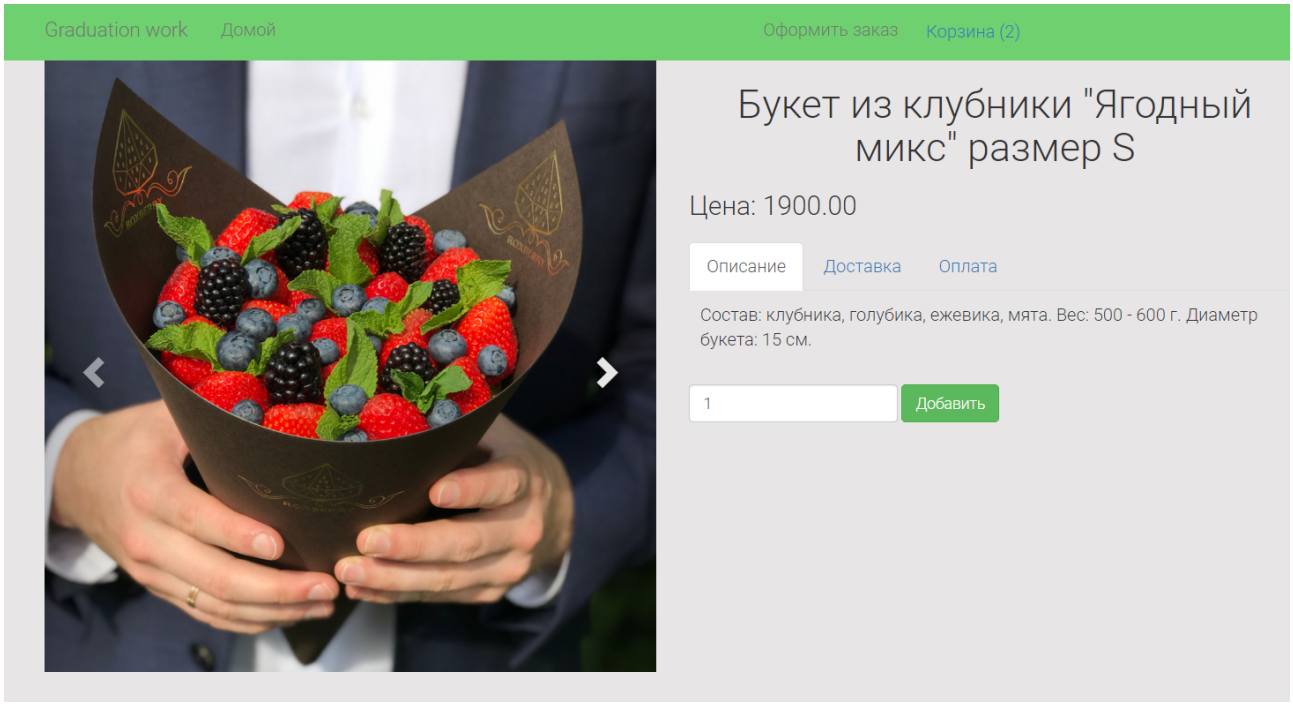

Рисунок 2 – Страница товара

Рассмотрено использование карусели для отображения фотографий товара. Слайды переключаются автоматически, но можно делать это самостоятельно, используя компьютерную мышь или клавиатуру.

Для отображения описания товара, информации о доставке и оплате использованы вкладки, что позволяет улучшить организацию данных и внешний вид сайта.

На странице оформления заказа необходимо указывать имя, номер телефона, выбрать способ оплаты. Также выводится содержимое корзины, автоматически подсчитывается общая сумма заказа и есть возможность изменения количества товара.

После успешного оформления заказа выводится сообщение на странице и корзина опустошается.

Чтобы оформить заказ, нужно обязательно указать имя и номер телефона, у способа оплаты задано значение по умолчанию. Если какое-либо поле формы не заполнено, выдается сообщение об ошибке.

Ошибки выводятся у каждого поля независимо друг от друга. Приведены примеры отображения ошибок.

В разделе рассмотрен процесс создания скрипта. Описан следующий функционал:

— функция отображения корзины;
— функция обновления корзины;
— событие по нажатию кнопки добавления товара в корзину;
— событие удаления товара из корзины;
— событие изменения количества товара в корзине;
— функция подсчета суммы корзины.

ЗАКЛЮЧЕНИЕ

В ходе данной работы была поставлена и достигнута задача создания веб-приложения для интернет-магазина. На основе выявленных требований были спроектированы модели данных и определены отношения между ними. В качестве одной из задач был изучен фреймворк Django для Python, средства которого помогли упростить процесс оформления заказов для покупателей и предоставить сотрудникам удобную работу с данными о клиентах, товарах и заказах.

Таким образом, была достигнута цель работы и выполнены все поставленные задачи. Результатом работы стало веб-приложение для интернет-магазина ягодных букетов, в котором покрыты все заданные требования.

В дальнейшем планируется создание системы регистрации пользователей, что позволит им просматривать все совершенные заказы, доработка графического интерфейса и интеграция с мобильными мессенджерами. Также предполагается добавление онлайн-оплаты. Все это должно повысить актуальность приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Фреймворки в веб-разработке [Электронный ресурс]. — URL: https://web-creator.ru/articles/about_frameworks (Дата обращения 15.03.2020). Загл. с экр. Яз. рус.

2 *Лутц, М.* Изучаем Python / М. Лутц. — Москва: Вильямс, 2019.

3 *Дронов, В. А.* Django 2.1. Практика создания веб-сайтов на Python / В. А. Дронов. — Санкт-Петербург: БХВ-Петербург, 2019.

4 Flask или Django [Электронный ресурс]. — URL: https://python-scripts.com/flask-or-django (Дата обращения 15.03.2020). Загл. с экр. Яз. рус.

5 SQLite Documentation [Электронный ресурс]. — URL: https://www.sqlite.org/docs.html (Дата обращения 16.03.2020). Загл. с экр. Яз. англ.

6 SQLite — Базы данных [Электронный ресурс]. — URL: https://lecturesdb.readthedocs.io/databases/sqlite.html (Дата обращения 16.03.2020). Загл. с экр. Яз. рус.

7 Bootstrap [Электронный ресурс]. — URL: https://getbootstrap.com/docs/4.5/getting-started/introduction/ (Дата обращения 18.03.2020). Загл. с экр. Яз. англ.

8 Что такое Bootstrap и зачем он нужен? [Электронный ресурс]. — URL: https://itchief.ru/bootstrap/introduction (Дата обращения 18.03.2020). Загл. с экр. Яз. рус.

9 Django учебник Часть 3: Использование моделей [Электронный ресурс]. — URL: https://developer.mozilla.org/ru/docs/Learn/Server-side/Django/Models (Дата обращения 18.03.2020). Загл. с экр. Яз. рус.

10 What is Database Model [Электронный ресурс]. — URL: https://www.lucidchart.com/pages/database-diagram/database-models (Дата обращения 18.03.2020). Загл. с экр. Яз. англ.

11 Frontend- и Backend-разработка [Электронный ресурс]. — URL: https://skillbox.ru/media/code/frontend_i_backend/ (Дата обращения 18.03.2020). Загл. с экр. Яз. рус.