**Корпоративная система лояльности**

**РАБОЧАЯ ДОКУМЕНТАЦИЯ**

**Руководство программиста**

**Листов – 6**

Киров

2022

**О программном стеке разработки.**

В качестве рабочего стека был выбран фреймворк Django.

Django – это бесплатный фреймворк для создания веб-приложений написанный на языке Python.

Как и любой другой фреймворк, «Джанго» — это программная платформа, на который разработчик добавляет дополнительные компоненты приложения, например:

* 1. Интерфейс пользователя.
  2. Скрипты (сценарии) работы элементов приложения/сайта.
  3. Программные библиотеки.
  4. Мультимедийный контент (фото, видео, аудио).
  5. Модули безопасности.
  6. Базы данных.

Фреймворк отвечает за базовые аспекты работы продукта. А подключаемые модули — за специфичные функции конкретного приложения.

**О программе.**

Весь проект состоит из 2 приложений – mysite, scorestore.

В папке mysite лежат все файла для настройки всего проекта. Там прописаны пути для статики, настройки debug, подключенные внутренние и сторонние библиотеки, домен и тд.

В папке scorestore развернут сам сайт.

В него входят следующие файлы:

1. Views – определяет функции, которые получают запросы пользователей, обрабатывают их и возвращают ответ (обработчик веб-страниц).

2. Models – хранит определение моделей, которые описывают используемые в приложении данные.

3. Apps – определяет конфигурацию приложения.

4. Admin – предназначен для административных функций, в частности, здесь производится регистрация моделей, которые используются в интерфейсе администратора.

5. Init – указывает интерпретатору python, что текущий каталог будет рассматриваться в качестве пакета.

6. папка migrations – предназначена для хранения миграций - скриптов, которые позволяют синхронизировать структуру базы данных с определением моделей.

7.urls – определяет url адреса для будущих страниц.

В качестве дополнительных библиотек были использованы – telepot, django-phonenumbers.

Для того чтобы развернуть проект на локальном сервере, необходимо иметь установленный фреймворк Django. Далее дойдя по каталогу до корневой папки с проектом в командной строке, нужно прописать python manage.py runserver. После чего сайт будет доступен по ссылке в консоли. Убедитесь, что у вас установлено виртуальное окружение venv.

На рисунке 1 изображен весь проект в каталоге.

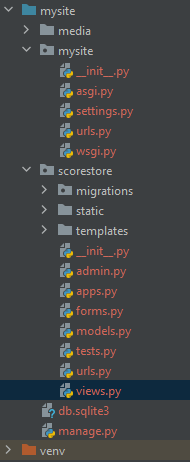


Рисунок 1 – файловая архитектура сайта.

На сайте предусмотрено внесение новых продуктов на витрину посредством специальной формы

Сначала был создан класс модели для описания параметров продукта.

Ниже приведен код для класса Goods.

# Создаем класс для продукта  
class Goods(models.Model):  
 # Наименование продукта  
 name = models.CharField(max\_length=255,  
 verbose\_name="Наименование")  
 # Описание продукта  
 description = models.TextField(blank=True,  
 verbose\_name="Описание")  
 # Цена продукта  
 price = models.IntegerField(default=5,  
 verbose\_name="Цена")  
 # Количество продукта  
 count = models.IntegerField(default=5,  
 verbose\_name="Количество")  
 # Фото продукта  
 photo = models.ImageField(upload\_to="photos/%Y/%m/%d/",  
 verbose\_name="Фото")  
 # Время когда продукт был создан  
 time\_create = models.DateTimeField(auto\_now\_add=True,  
 verbose\_name="Время создания")  
 # Время когда продукт был обновлен  
 time\_update = models.DateTimeField(auto\_now=True,  
 verbose\_name="Время обновления")  
 # Опубликован ли продукт  
 is\_published = models.BooleanField(default=True,  
 verbose\_name="Публикация")

Обязательно после описания моделей в файле models, нужно сделать миграции в базу данных. Делается это двумя консольными командами:

1. Python manage.py makemigrations
2. Python manage.py migrate

Первая консольная программа создает миграции и создает файл с описанием всех моделей, которые вы сделали.

Так же следует применять эту команду для обновления моделей, если вы ее изменили.

Вторая же консольная команда уже создает таблицу и раздел из описанных вами моделей.

Затем мы прописываем код для формы, с помощью которой мы и будем добавлять новые записи о продуктах в базу данных.

# Форма для добавления предмета на витрину  
class AddItemForm(forms.ModelForm):  
 class Meta:  
 # Модель использующаяся в форме = Goods  
 model = Goods  
 # Поля из модели использующиеся в форме  
 fields = ['name', 'price', 'count', 'photo', 'is\_published']

После чего мы должны прописать использование данной формы в файле views, откуда мы затем передадим данную форму в шаблон html

# Добавление предмета  
def additem(request):  
 # Проверка на запрос  
 if request.method == 'POST':  
 form = AddItemForm(request.POST, request.FILES)  
 # Проверка формы на валидность  
 if form.is\_valid():  
 form.save()  
 return redirect('shop')  
 else:  
 form = AddItemForm()  
 return render(request, 'scorestore/additem.html', {'form': form, 'menu': menu, 'title': 'Добавление товара'})

{% extends 'scorestore/base.html' %}  
  
{% block content %}  
<h1>{{title}}</h1>  
<form action="{% url 'additem' %}" method="post" enctype="multipart/form-data">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Добавить</button>  
</form>  
{% endblock %}