Практическое задание

Чтобы закрепить знания по фреймворку Django и подготовиться к собеседованию, предлагаем реализовать несложное приложение.

Практическим заданием будет каталог товаров, состоящий из двух страниц:

- главной со списком товаров и кнопкой их добавления;
- и страницы добавления товара.

Приложение должно функционировать в синхронном режиме: пользователь нажимает кнопку «Добавить товар», переходит на предназначенную для этого страницу, указывает в форме необходимые данные и нажимает «Сохранить». После этого он перенаправляется на главную страницу, где в табличной форме отображается список всех товаров. Проект должен быть привязан к базе данных **SQLite3** (БД по умолчанию) и иметь минимальную сложность стилевого оформления.

На первом этапе необходимо реализовать список товаров, а добавление самих товаров осуществлять через админку приложения

При работе над проектом:

1. Создать виртуальное окружение проекта, под которым установить необходимый инструментарий (файл **requirements.txt**).
2. Под виртуальным окружением создать Django-проект и одно приложение, настроить файл **settings.py**, выполнить базовые миграции. Запустить Django-сервер для проверки работоспособности проекта.
3. В каталоге приложения создать модель, которая должна хранить информацию о поступивших товарах: название, дату поступления, цену, единицу измерения, имя поставщика. Выполнить миграции.
4. Проверить правильность созданной модели, зарегистрировав ее в админке приложения.
5. Настроить файл **urls.py** внутреннего каталога проекта. Он должен содержать привязку к url-адресу админки проекта (будет в файле по умолчанию после создания проекта) и привязку к набору шаблонов url-адресов созданного приложения (оператор **include**).
6. Создать и настроить файл привязок **urls.py** для приложения. В этом файле создать к url-адресу главной страницы проекта. Для нее указать функцию-контроллер и название. Функция-контроллер должна отвечать за загрузку списка товаров на главной странице.
7. В файле **views.py** каталога проекта реализовать контроллер в формате функций либо ListView. Контроллер должен извлекать все записи из модели с каталогом товаром и передавать переменную со списком товаров в контекст шаблона (html-страница со списком товаров).
8. В корне проекта создать директорию **templates** с базовым шаблоном (**base.html**). Базовый шаблон будет соответствовать каркасу главной страницы. В нем необходимо реализовать один динамически обновляемый блок — например, **{% block content %}{% endblock %}**. Он будет содержать таблицу со списком товаров, которая динамически подгружается из шаблона-наследника (html-страница со списком товаров). В файле **base.html** необходимо подключить статику и указать ссылку на CSS-файл со стилизацией проекта. Можно

воспользоваться файлом **bootstrap.min.css** (его нужно скачать и поместить в каталог **.static/css**).

9. В каталоге приложения создать директорию **templates** с шаблоном html-страницы со списком товаров (**goods_list.html**). В нем необходимо указать шаблон-родитель **base.html**, кнопку добавления товара и разметить html-таблицу. Каждая из ее строк (кроме той, что с заголовками) должна формироваться при переборе содержимого переменной со списком товаров — мы ее предварительно передали в контекст данного шаблона из соответствующего контроллера. Для каждого из значений переменной (фактически — это запись базы данных), полученного в цикле, необходимо обратиться к нужному полю и вывести его в соответствующей ячейке.