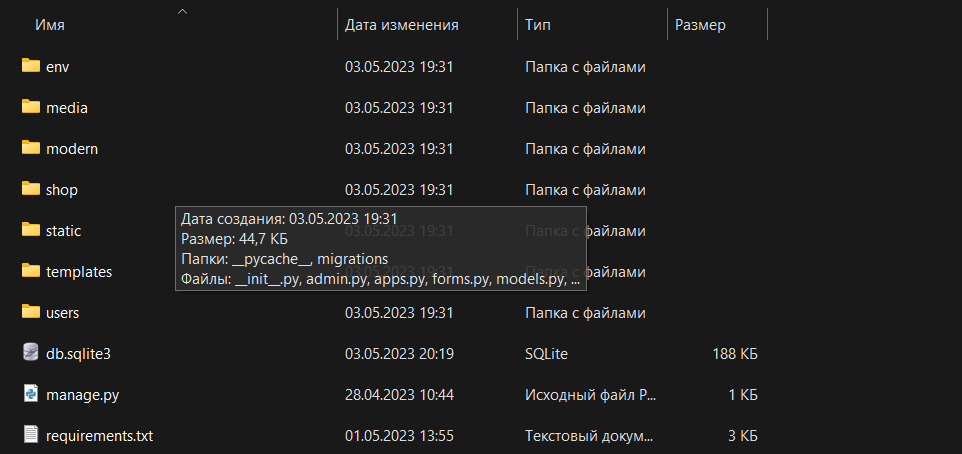
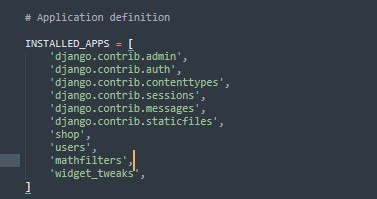
интернет-магазин на Django

Для начала над проектом создании интернет магазина я установил Django(pip install django) Создал новый проект с помощью команда в терминале, Django-admin startproject modern далее создал два приложения, users и shop, команда python manage.py startapp shop и users. Я это всё сделал в виртуальной среде заранее установив его(env), это считается хорошим тоном для таких больших проектов, как интернет – магазин. Виртуальная среда - это изолированное пространство, в котором устанавливаются зависимости для проекта, отделенное от остальных зависимостей на компьютере. Это помогает избежать конфликтов между различными версиями зависимостей, которые могут возникнуть при работе над несколькими проектами на одном компьютере.

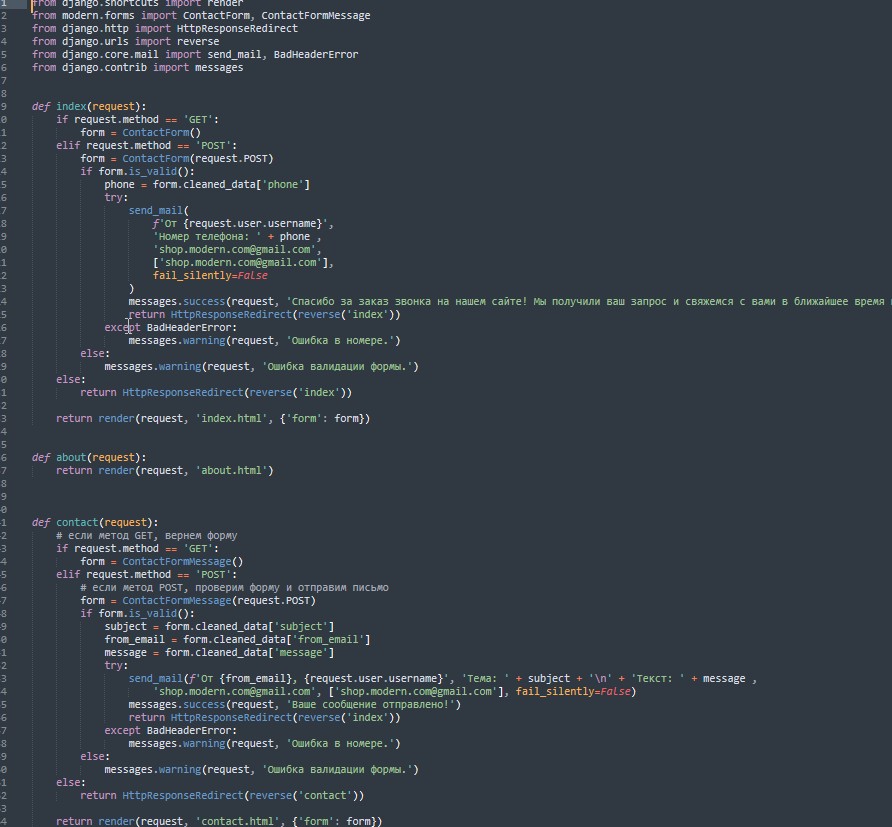


Далее создал настроил в папке главного проекта modern, Settings для работы с датами, установил приложения.



Установленные приложения

В этой же папке(Modern) создал файлы, forms.py, views.py, они отвечают за поведение сайта, формы и post и get запросов на север.



Этот код

- Проверяет, был ли запрос отправлен методом GET или POST;

- Создает экземпляр формы `ContactForm()` в случае метода GET;

- Создает экземпляр формы с POST данными и проверяет, проходит ли эта форма валидацию;

- Если форма проходит валидацию, код отправляет сообщение электронной почты на почту администратора, используя `send\_mail` функцию Django;

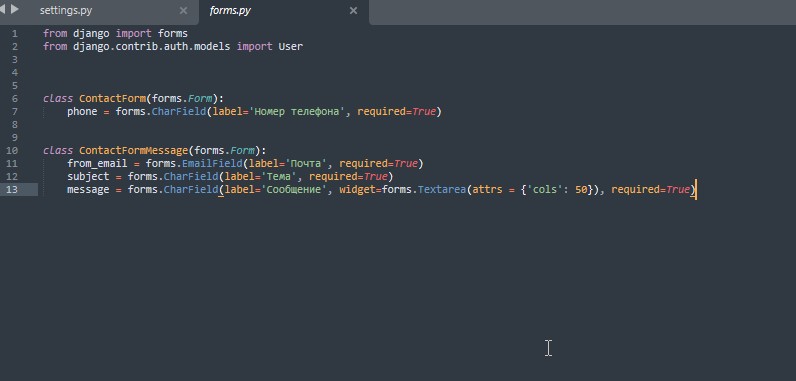
- На экран выводится сообщение об успехе, если отправка прошла успешно;

- Если при отправке возникла ошибка, выводится сообщение с предупреждением об ошибке.

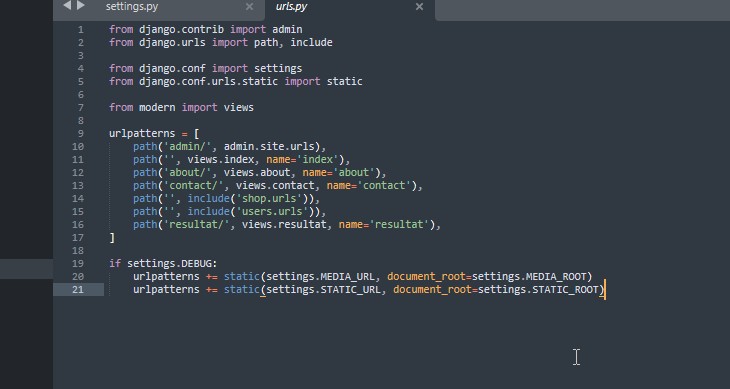
- Наконец, функция возвращает шаблон `index.html` с формой и добавляет сообщения в контекст, чтобы их можно было отобразить на странице.

Функция `about`:

- Возвращает шаблон `about.html`.



Так же немаловажно файл urls.py, который отвечает за переходами между страницами.



Этот код содержит переменную `urlpatterns` - это список объектов URL-адресов, которые связывают конкретный URL-шаблон с функцией представления Django.

- `path('admin/', admin.site.urls)` обрабатывает URL-адрес, используемый для получения доступа к административной панели Django.

- `path('', views.index, name='index')` указывает на главную страницу приложения, которая будет обрабатываться функцией представления `index` из модуля `views` в приложении `modern`.

- `path('about/', views.about, name='about')` определяет URL-адрес для страницы "О нас" с использованием представления `about` из модуля `views`.

- `path('contact/', views.contact, name='contact')` определяет URL-адрес для страницы контактов с использованием представления `contact`.

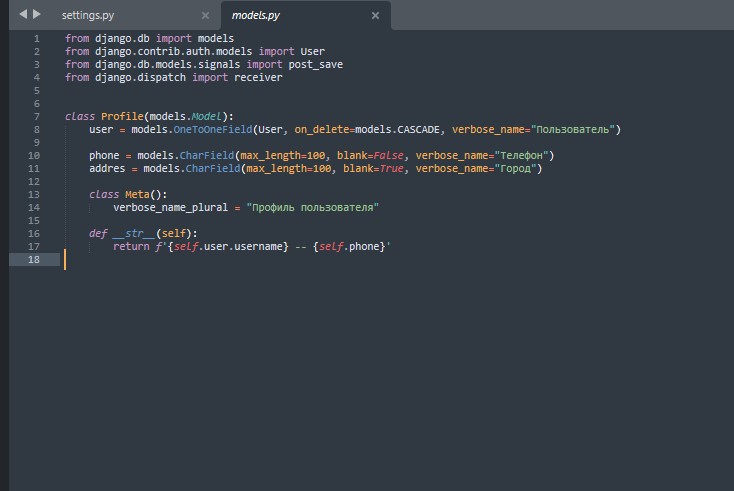
- `path('', include('shop.urls'))` подключает URL-адреса из приложения `shop`. При этом `include` используется для включения URL-адресов из файла `urls.py` в приложении `shop`.

- `path('', include('users.urls'))` подключает URL-адреса из приложения `users`. Перенаправляет URL-адреса в `urls.py` файлах приложения `users`.

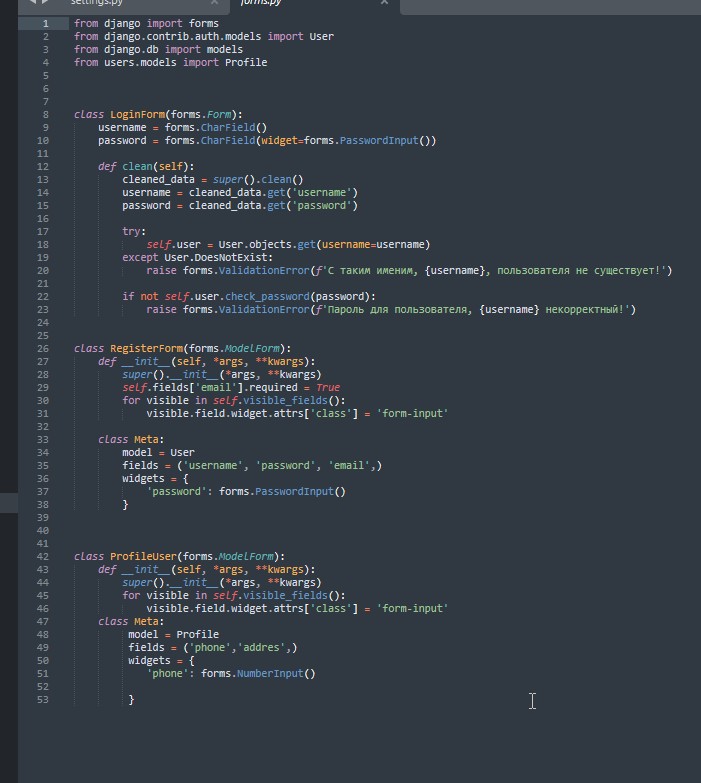
- `path('resultat/', views.resultat, name='resultat')` определяет URL-адрес для страницы результата с использованием представления `resultat`.

Когда приходит запрос на URL-адрес, Django ищет соответствующий объект в списке `urlpatterns`. Если находит соответствующий объект, то Django вызывает связанную функцию(`view`) и передает в нее запрос пользователя.

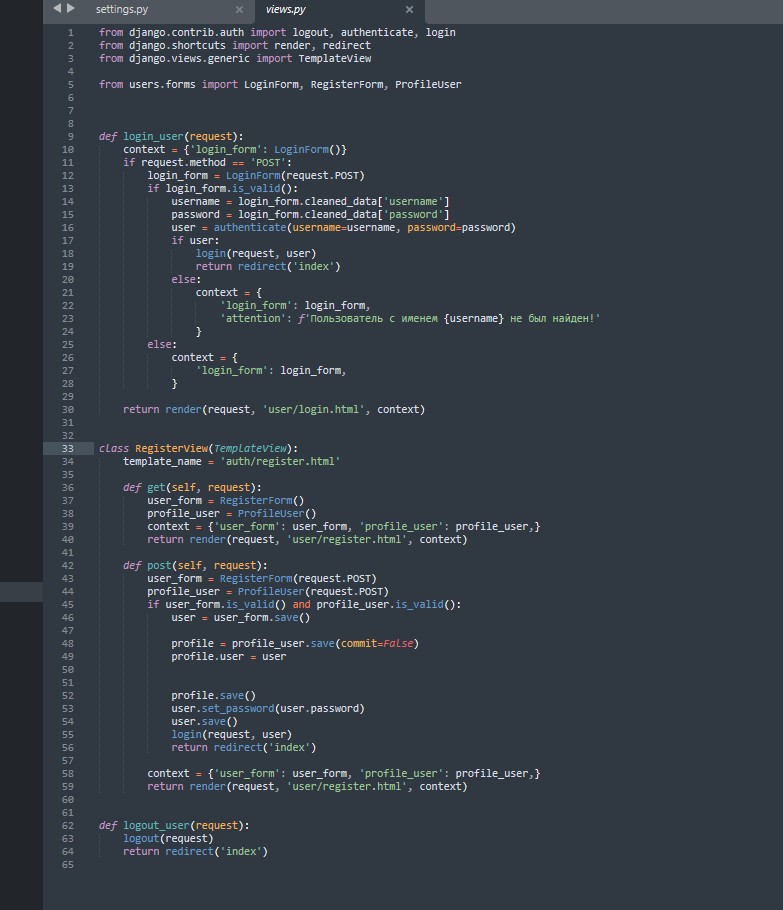
После всего этого я начал добавлять в каждом приложение эти файлы: views.py, urls.py и forms.py. Для начала создал модели. Для приложения users.



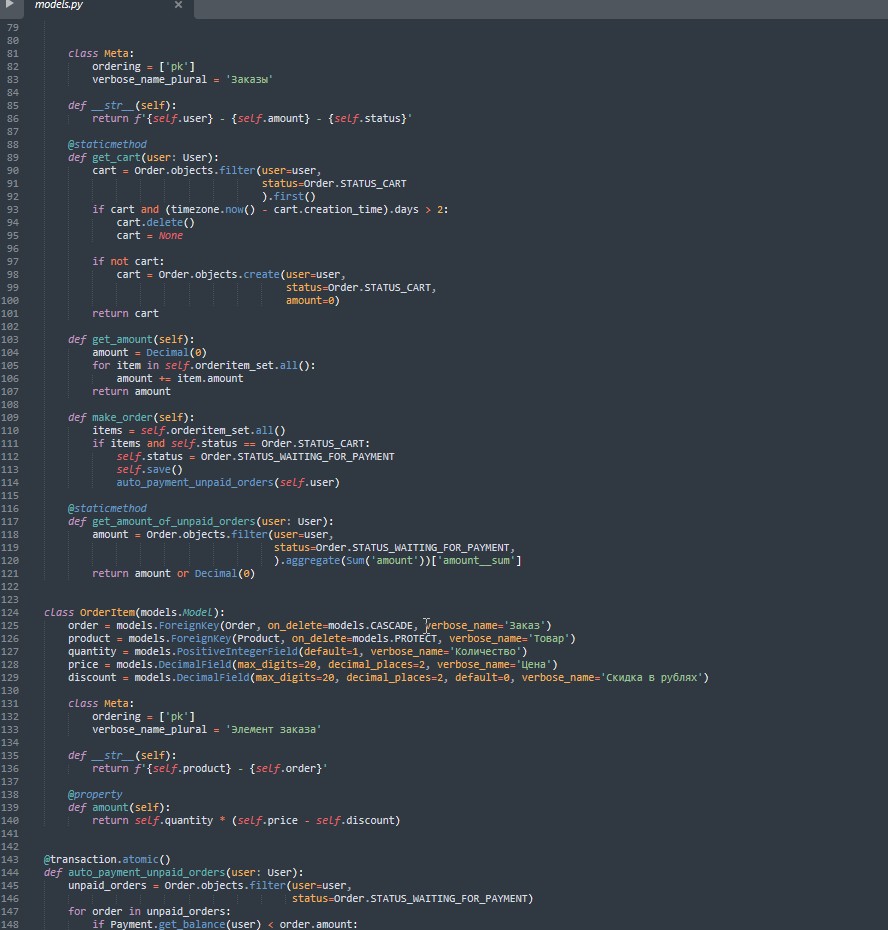
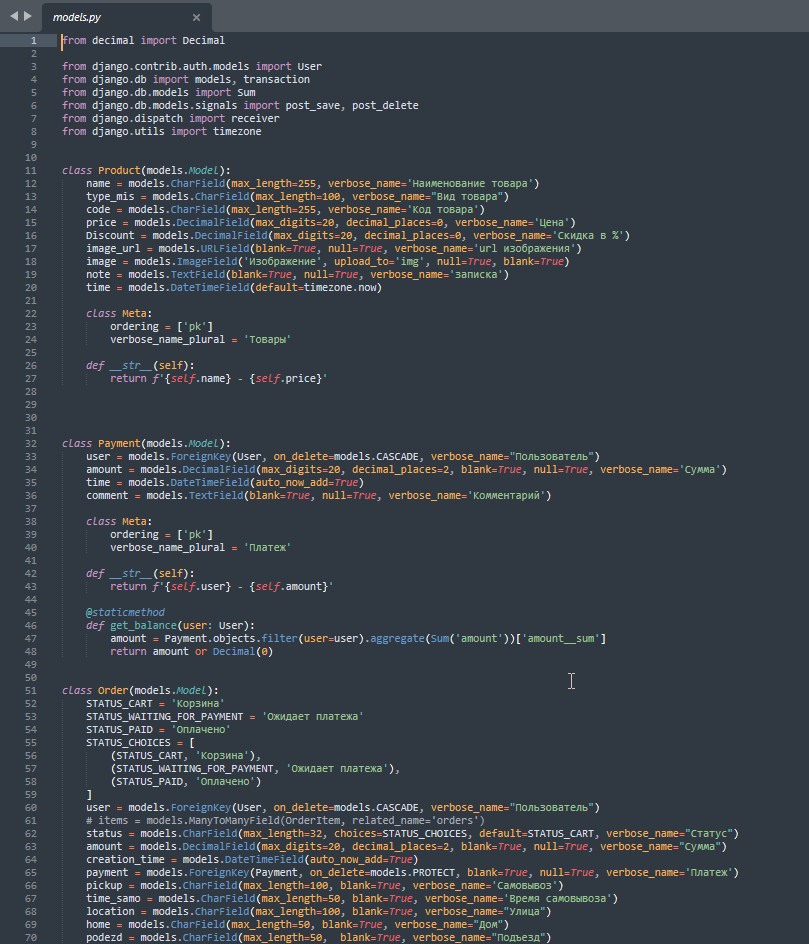
Формы для отображения на странице html.



И views.py для обработки запросов и проверки на сервере, регистрации пользователей и входа.



В приложения таким же сделал, просто там уже для добавлении товаров, для корзины, цены.



`Product`, которая наследуется от базовой модели `models.Model` из Django.

`Product` имеет следующие поля:

- `name` - имя товара.

- `type\_mis` - вид товара

- `code` - код товара.

- `price` - цена товара.

- `Discount` - скидка в %.

- `image\_url` - URL-адрес изображения, связанного с товаром.

- `image` - поле загрузки изображения, связанного с товаром.

- `note` - дополнительная информация о товаре.

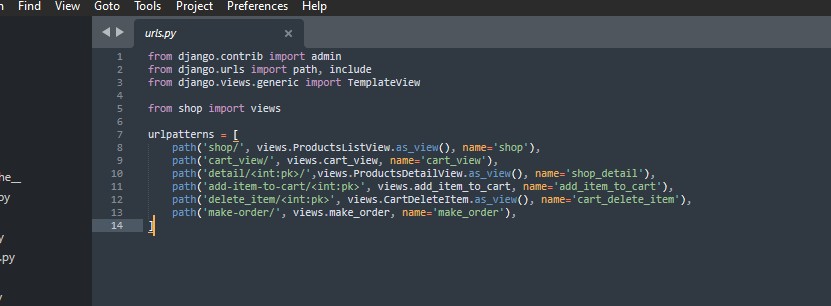
- `time` - дата создания товара.

`Meta` класс используется для управления метаданными модели, в данном случае ordering установлено для сортировки записей по ключу(pk), а verbose\_name\_plural используется для установки человекочитаемого имени модели во множественном числе.

Также в модели определен метод `str()`, который возвращает строковое представление модели. Этот метод возвращает строку, содержащую имя и цену товара.

Модель `Product` будет использоваться для создания таблицы в базе данных, в которой хранится информация о товарах, представленных на сайте. Django автоматически создаст таблицу в базе данных при миграции данных. Для доступа к данным в этой модели, например через запросы к базе данных, можно использовать объекты этого класса в контроллерах Django.

Так же очень важно urls.py в приложение shop,



- `path('shop/', views.ProductsListView.as\_view(), name='shop')` определяет URL-адрес для просмотра списка продуктов с использованием представления `ProductsListView`.

- `path('cart\_view/', views.cart\_view, name='cart\_view')` определяет URL-адрес для просмотра товаров в корзине с использованием функции представления `cart\_view`.

- `path('detail/<int:pk>/',views.ProductsDetailView.as\_view(), name='shop\_detail')` создает URL-адрес для просмотра информации о конкретном товаре с использованием представления `ProductsDetailView`.

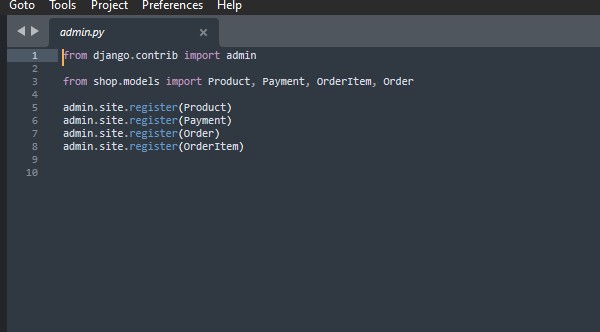
- `path('add-item-to-cart/<int:pk>', views.add\_item\_to\_cart, name='add\_item\_to\_cart')` определяет URL-адрес для добавления товара в корзину с использованием функции представления `add\_item\_to\_cart`.

- `path('delete\_item/<int:pk>', views.CartDeleteItem.as\_view(), name='cart\_delete\_item')` указывает URL-адрес для удаления товаров из корзины с помощью представления `CartDeleteItem`.

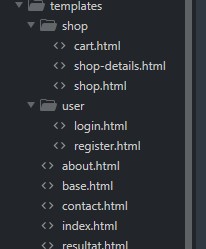
- `path('make-order/', views.make\_order, name='make\_order')` определяет URL-адрес для создания заказа с использованием функции представления `make\_order`.

`path()` используется для определения URL-адресов, которые затем будут разрешаться во время навигации по сайту. Когда приходит запрос на URL-адрес, Django ищет соответствующий в списке `urlpatterns`. Если находит соответствующий объект, то Django вызывает связанную функцию представления и передает ей запрос пользователя.

Таким образом я каждую модель зарегистрировал в админ, чтобы на сервере могли управлять сайтом, внести изменения или добавлять товары на сайт, так же посмотреть оплаты, корзину и так далее.



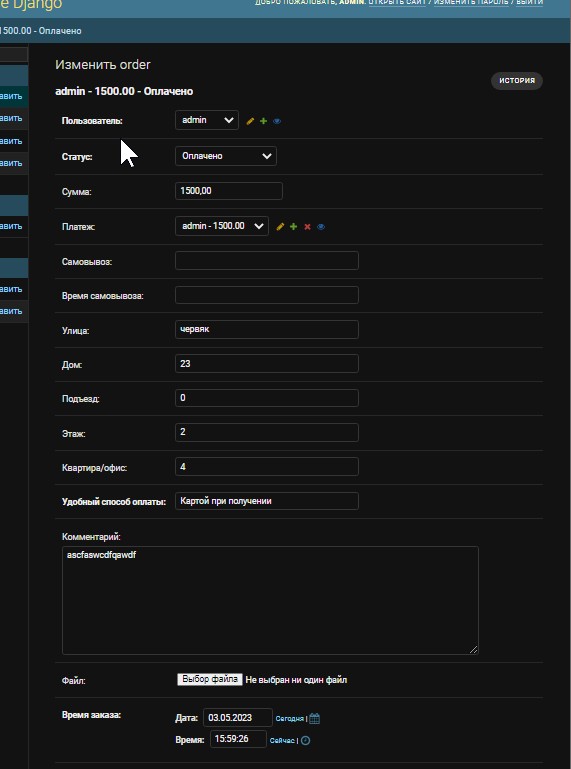
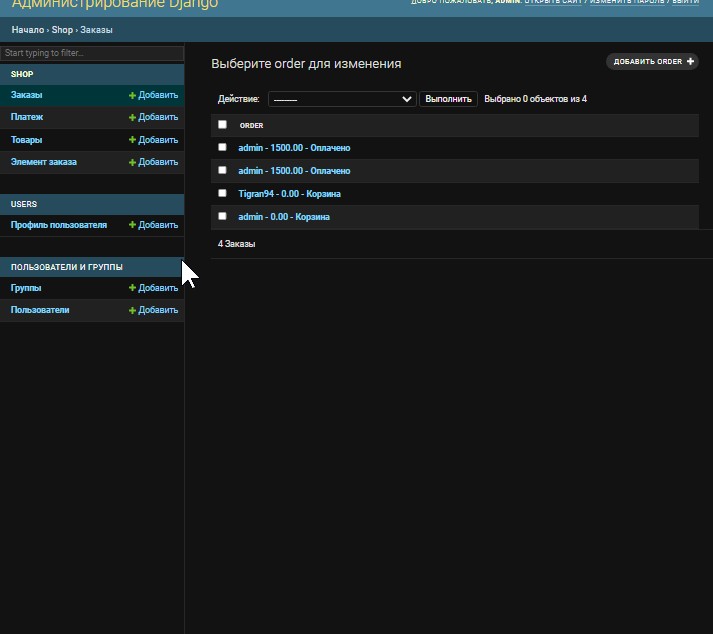
Кроме всего этого создал папку templates для html шаблонов, создал шаблоны для страницы, base.html, index, shop.html и так далее.



Static создал папку для images, css и javascript. Это делается для вида страницы, поведение и стилей.

Во время работы с сайтом, я каждый раз делал команду в терминале для обновления данных в базе. Python manage.py makemigrations и Python manage.py migrate.

На сайте мы можем из страницы магазина добавлять продукты в корзину и заказть доставкой или забронировать, потом забрать. Потом на сервере мы получаем данные о покупке также получаем заказы и на почту.



Таким образом я создал интернет магазин для одежды. Конечно там много нюансов, свои сложности. Но в целом шаг за шагом показал вам, как это делается.

