МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и

информационные технологии»

Курсовая работа

**«Создание проекта Интернет-Магазина»**

по дисциплине

**«Технологии и методы программирования»**

Выполнил: студент гр. БИБ2103 Мылтыкбаев А.Р.

Проверил:

**Оглавление**

[Создание моделей каталога продуктов 2](#_Toc988485649)

[Регистрация моделей catalog на сайте администрирования 4](#_Toc1513270314)

[Создание шаблонов каталога 4](#_Toc1138787978)

[Хранение корзины покупок в сессиях 6](#_Toc868674851)

[Добавление элементов в корзину 7](#_Toc1262986529)

[Создание шаблона для отображения корзины 10](#_Toc1358498699)

[Добавление товаров в корзину 11](#_Toc742981207)

[Создание обработчика контекста для текущей корзины 11](#_Toc1630318326)

[Регистрация заказов 12](#_Toc814908817)

[Создание заказов клиентов 13](#_Toc676552525)

[Конечный вид сайта 16](#_Toc1721720905)

Вывод 18

Создание моделей каталога продуктов



Рисунок 1-отрывок кода создания моделей

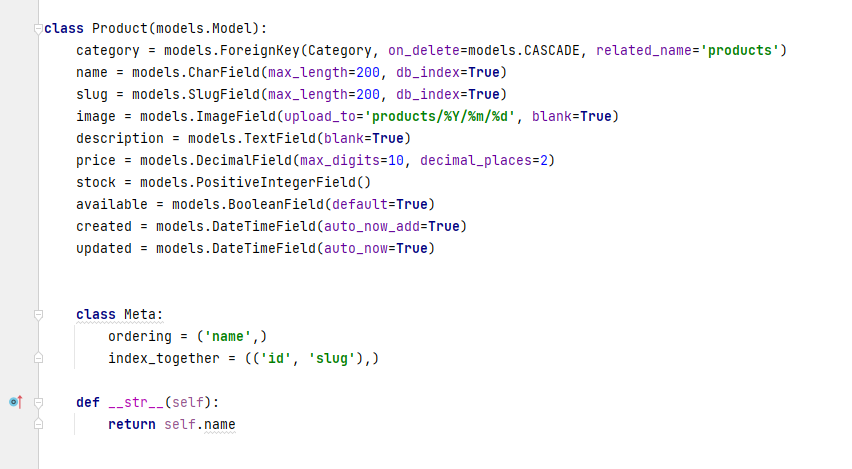


Рисунок 2-отрывок кода создания моделей

Это модели Category и Product. Модель Category состоит из

поля **name** и **slug**. Рассмотрим поля модели Product:

• **category** : Это ForeignKey модели Category. Это отношение "многие к

одному": продукт относится к одной категории, а категория содержит

несколько продуктов

• **name** : Название продукта.

• **slug** : Алиас продукта(его URL).

• **image** : Изображение продукта.

• **description** : Необязательное описание для продукта.

• **price** : Это поле DecimalField. В нем используется десятичное число

Python. Десятичный тип для хранения десятичного числа с

фиксированной точностью. Максимальное число цифр (включая

десятичные разряды) задается с помощью атрибута max\_digits и

десятичных знаков с атрибутом decimal\_places

• **stock** : Это поле PositiveIntegerField для хранения остатков данного

продукта.

• **available** : Это булево значение, указывающее, доступен ли продукт или

нет. Позволяет включить/отключить продукт в каталоге.

• **created** : Это поле хранит дату когда был создан объект.

• **updated** : В этом поле хранится время последнего обновления объекта.

Для поля **price** мы используем **DecimalField** вместо **FloatField**, чтобы избежать проблем округления.

Регистрация моделей catalog на сайте администрирования

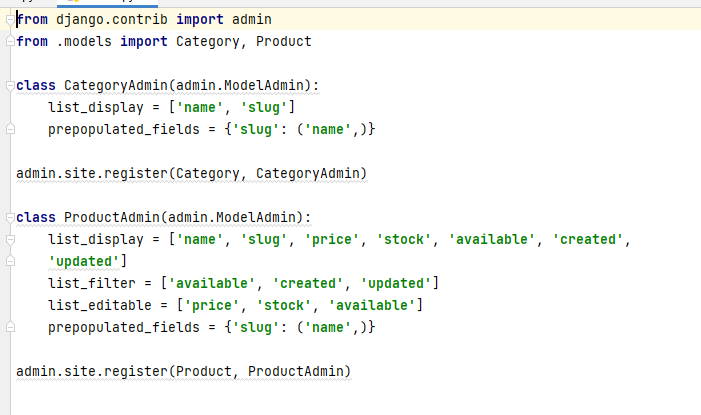


Рисунок 3-код создания каталога на сайте администрирования

Мы используем атрибут **prepopulated\_fields**, чтобы указать поля,

в которых значение автоматически задается с использованием значения

других полей. Атрибут **list\_editable** в классе **ProductAdmin** используется для задания полей, которые могут быть отредактированы на странице отображения списка сайта администрирования. Это позволит редактировать несколько строк одновременно.

Создание шаблонов каталога

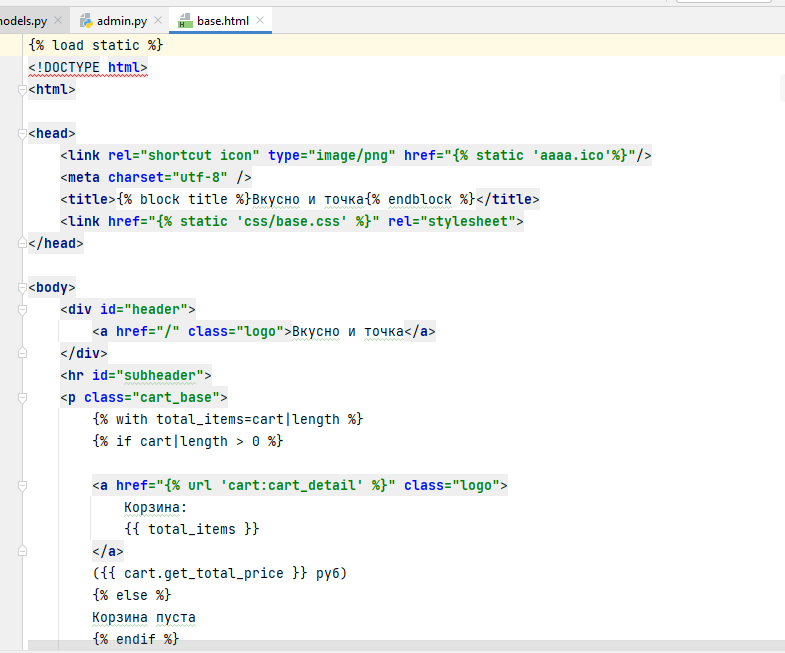


Рисунок 4-отрывок создания шаблонов каталога



Рисунок 5-отрывок создания шаблонов каталога



Рисунок 6-отрывок создания шаблонов каталога

Также, чтобы Джанго обрабатывал загруженные мультимедийные файлы с помощью сервера разработки, я импортировал static **from django.conf.urls.static import static**

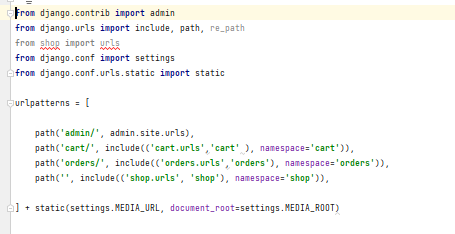


Рисунок 7-Импортирование всех библиотек

Хранение корзины покупок в сессиях

Я создал приложение **cart** и добавил в **INSTALLED\_APPS** в файле settings.py



Рисунок 8-добавил приложение cart.py

Далее создал в каталоге cart **cart.py**

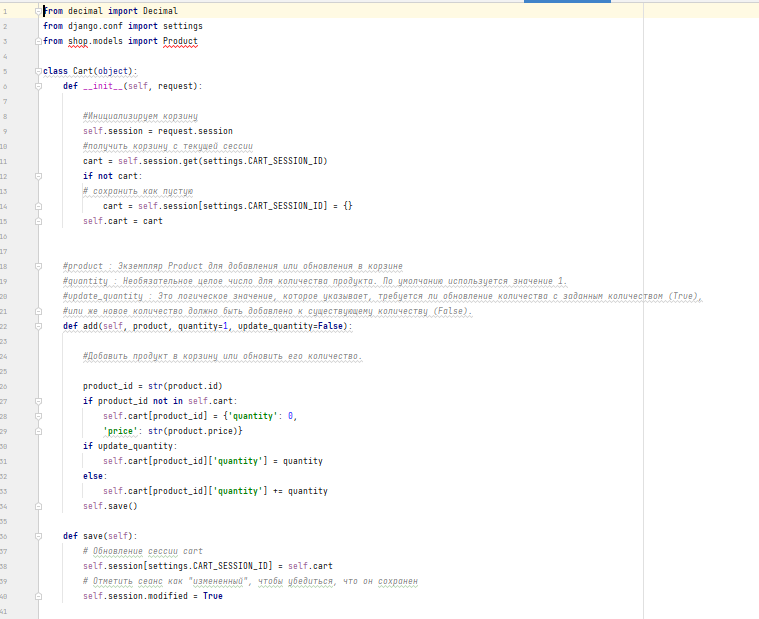


Рисунок 9-Отрывок кода в cart.py

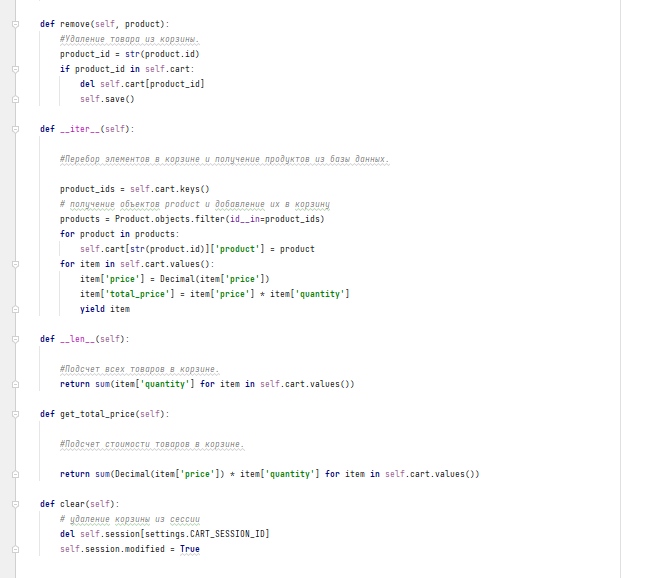


Рисунок 10-Отрывок кода в cart.py

Класс Cart позволит нам управлять корзиной для покупок.

Добавление элементов в корзину

Чтобы добавить элементы в корзину, нам нужна форма, позволяющая

пользователю выбрать количество добавляемого товара. Для этого я создал файл forms.py в каталоге приложения cart и добавил в него следующий код:

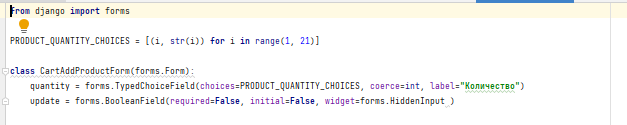


Рисунок 11-Код файла forms.py

Эта форма будет использоваться для добавления продуктов в корзину. Класс **CartAddProductForm** содержит следующие поля:

**quantity :** позволяет пользователю выбрать количество между 1-20. Мы используем поле TypedChoiceField с coerce=int для преобразования ввода в целое число.

**update :** позволяет указать, следует ли добавлять сумму к любому существующему значению в корзине для данного продукта (False) или если существующее значение должно быть обновлено с заданным значением (True). Для этого поля используется графический элемент HiddenInput, поскольку не требуется показывать его пользователю.

Далее я создал представление для добавления элементов в корзину.

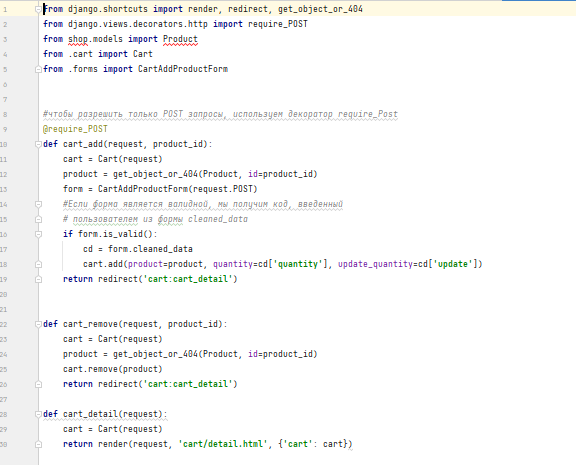


Рисунок 12-Представление для добавление элементов в

корзину

Представление **cart\_detail** выводит на экран текущее состояние корзины.

Далее я добавил шаблоны url адресов приложения **cart** в **urls.py**



Рисунок 13-добавление приложение cart в urls.py

И включил эти адреса в основной urls.py проекта



Рисунок 14-Включение адресов в основной urls.py

Создание шаблона для отображения корзины



Рисунок 15- отрывок кода для отображения, надписей,

изображений

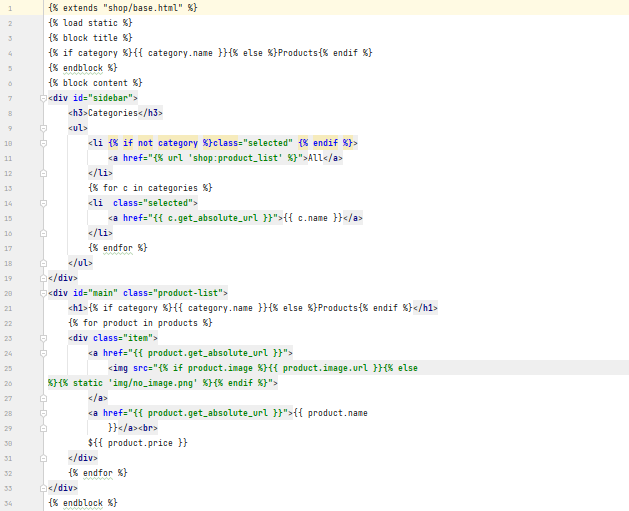


Рисунок 16-заголовки и так далее

Этот шаблон используется для отображения содержимого корзины. Он содержит таблицу с элементами, хранящимися в текущей корзине. Мы разрешаем пользователям изменять количество выбранных продуктов, используя форму, которая учитывается в представлении cart\_add. Мы также разрешаем пользователям удалять элементы из корзины, предоставляя для каждого из них ссылку Remove.

Добавление товаров в корзину

Отредактировал файл views.py приложения shop и добавил CartAddProductForm в представление product\_detail, следующим образом:

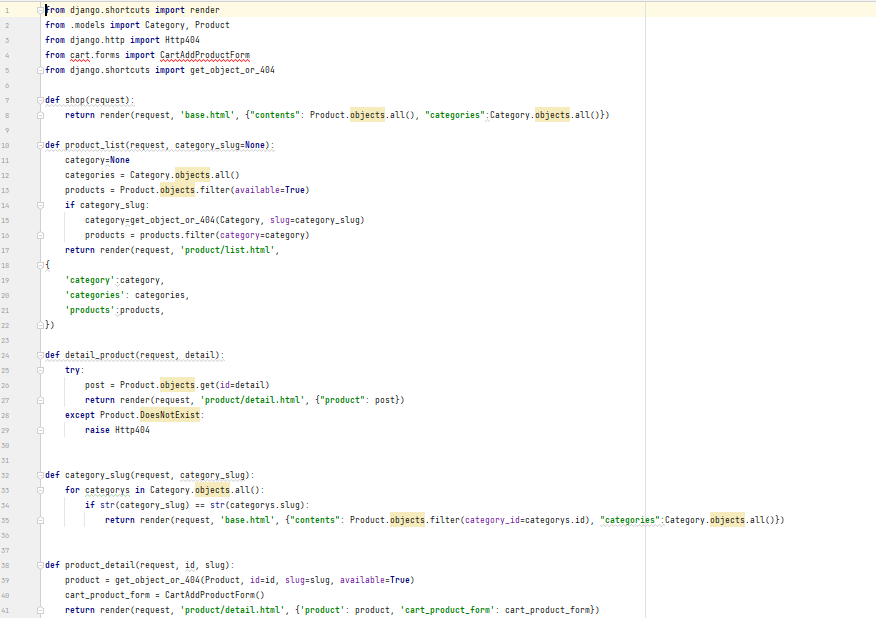


Рисунок 17- отредактированный файл views.py в shop

Создание обработчика контекста для текущей корзины

Создал новый файл в каталоге приложения cart и назвал его **context\_processors.py.**

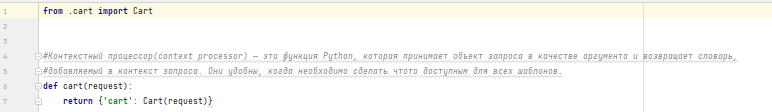


Рисунок 18- context\_processors.py

Далее отредактировал Settings.py проекта, добавив этот процессор



Рисунок 19- отредактированный в Settings.py

И для главной страницы (html) добавил обработчик:

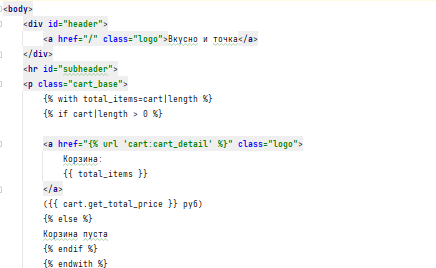


Рисунок 20-html с обработчиком

Регистрация заказов

Создал новое приложение orders. И добавил его в файл settings.py в параметры INSTALLED\_APPS:



Рисунок 21- с добавленная с orders

Потребуется модель для хранения сведений о заказе и второй модели для

хранения купленных товаров, включая их количество и цену. Для этого я создал класс order в файле **models.py** приложения **orders**:



Рисунок 22-класс orders

Также нужно включить модели заказов на сайте администрирования, для этого я добавил модель заказов, изменив файл **admin.py** приложения **orders**:

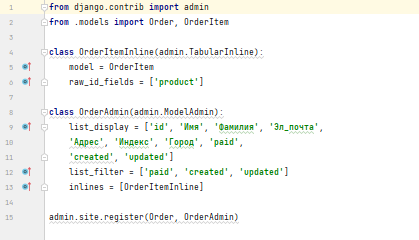


Рисунок 23- измененный admin.py в orders

Создание заказов клиентов

Во-первых, я создал новый файл в каталоге приложения orders и назвал его forms.py:

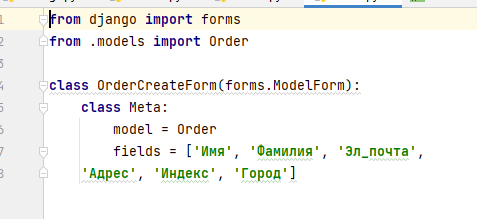
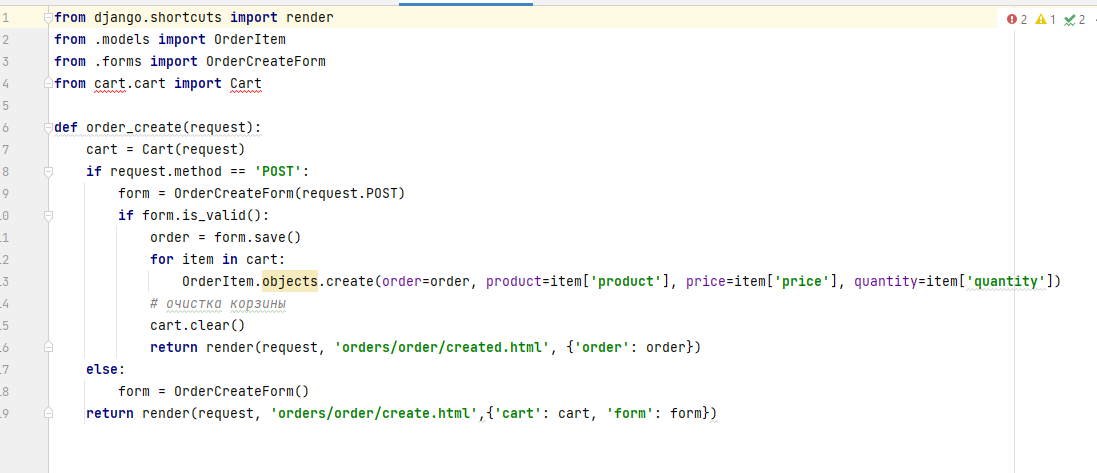


Рисунок 24-forms.py в orders

Это форма, которая будет использоваться для создания новых

объектов Order

Теперь мне нужно создать представление, чтобы обработать форму и создать новый заказ, для этого я добавил след. Код в файл views.py приложения orders:

  
Рисунок 25-Код в файле views.py приложения orders

В представлении order\_create мы получаем текущую корзину из сесссии с cart = Cart(request)

Также я создал новый файл ursl.py приложения orders:

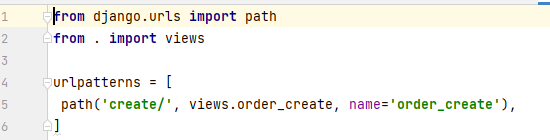


Рисунок 26-новый urls.py в orders

И включил этот файл в основной файл ursl.py проекта (26 строчка кода):



Рисунок 27- добавленние файла в основной urls.py

И добавил в шаблон страницы следующий код (56 строчка):

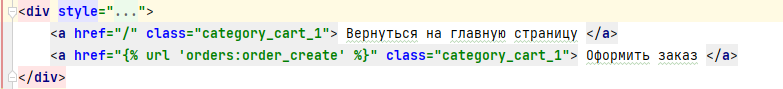


Рисунок 28- Заголовок активных кнопок

Отредактировал шаблон orders/order/create.html (страница оформления заказа):

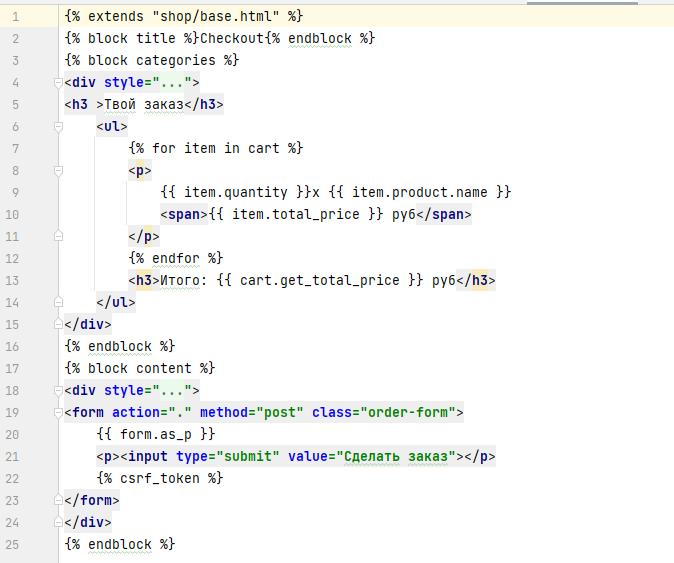


Рисунок 29-Отредактированный шаблон оформления заказа

:

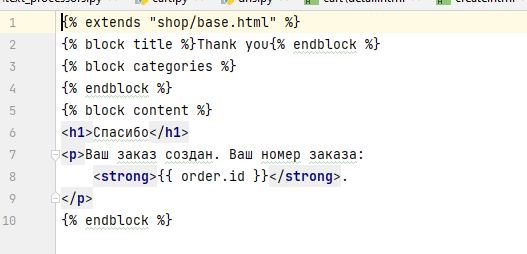


Рисунок 30- шаблон orders/order/created.html (страница после оформления заказа, с выводом номера заказа)

Добавление иконки на сайт:



Рисунок 31- Добавление иконки

Конечный вид сайта

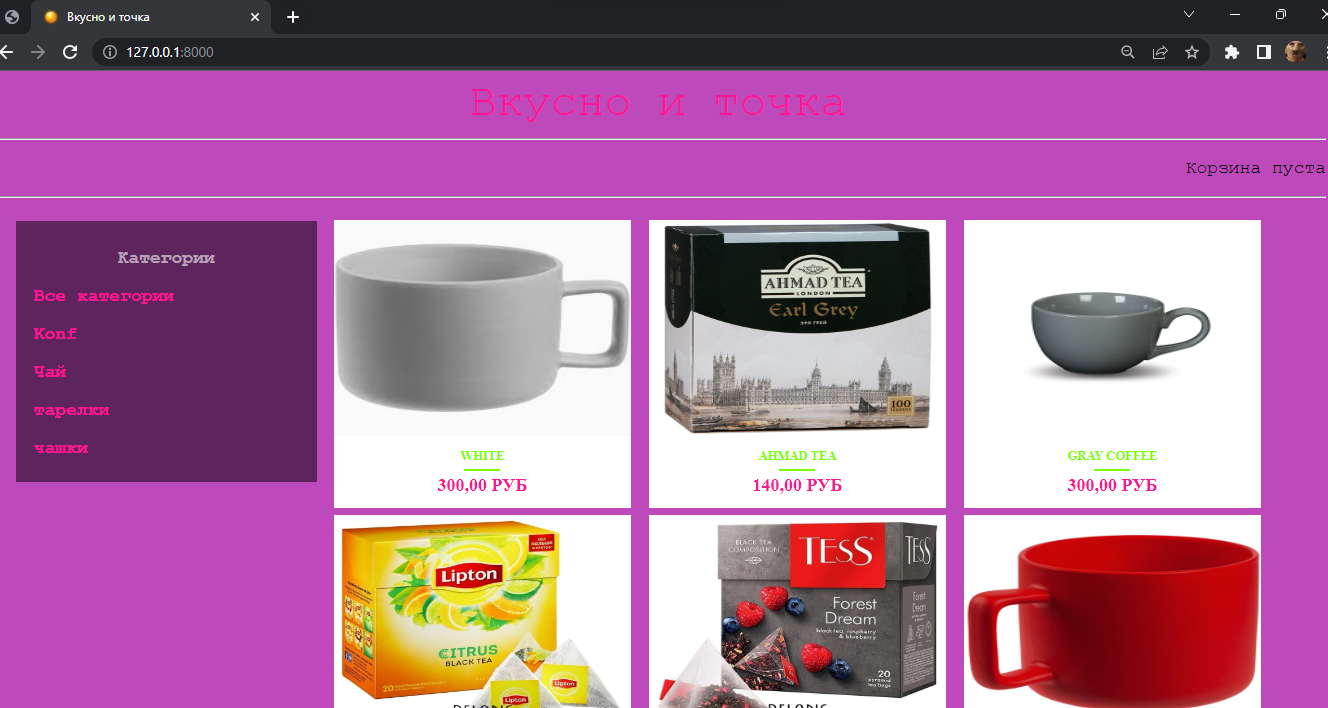


Рисунок 32-Главная страница сайта



Рисунок 33-Сортировка по категориям (в данном случае конфеты):



Рисунок 34- Сортировка по категориям

(в данном случае конфеты):

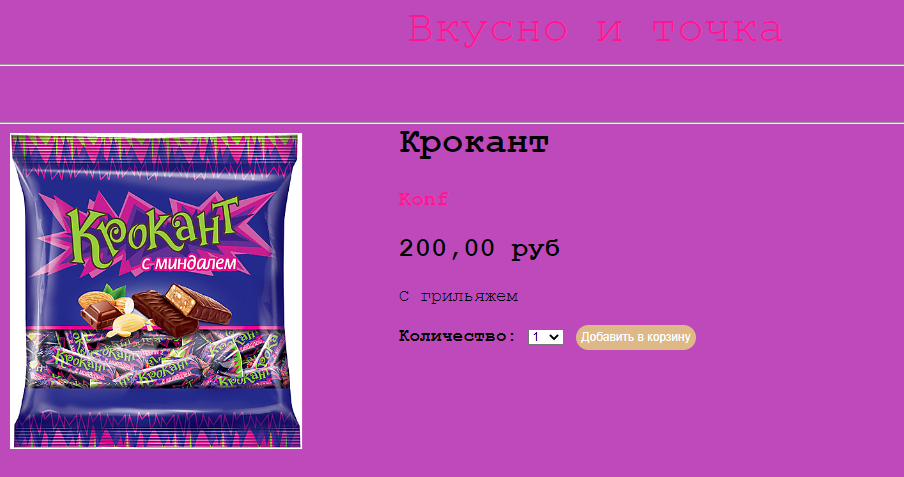


Рисунок 35-Страница с отображением товара, его

описания и кол-во единиц для покупки:

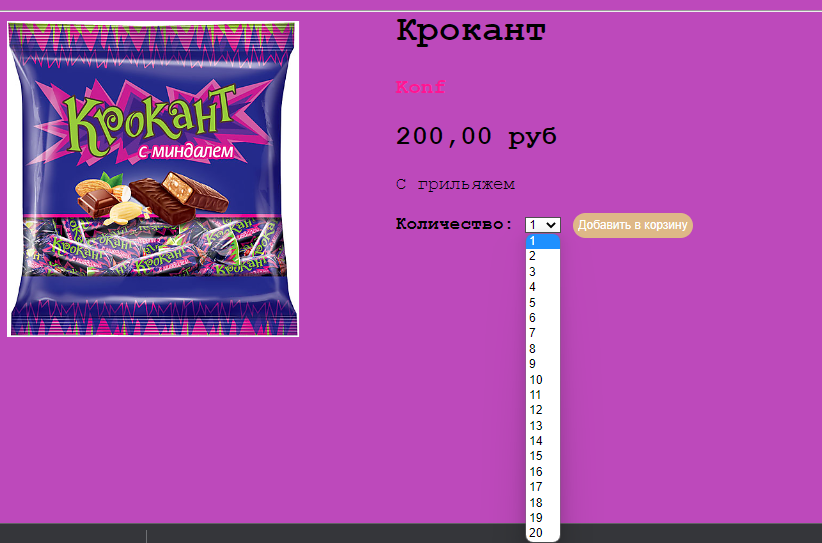


Рисунок 36- Страница с выбором количества товаров в корзину

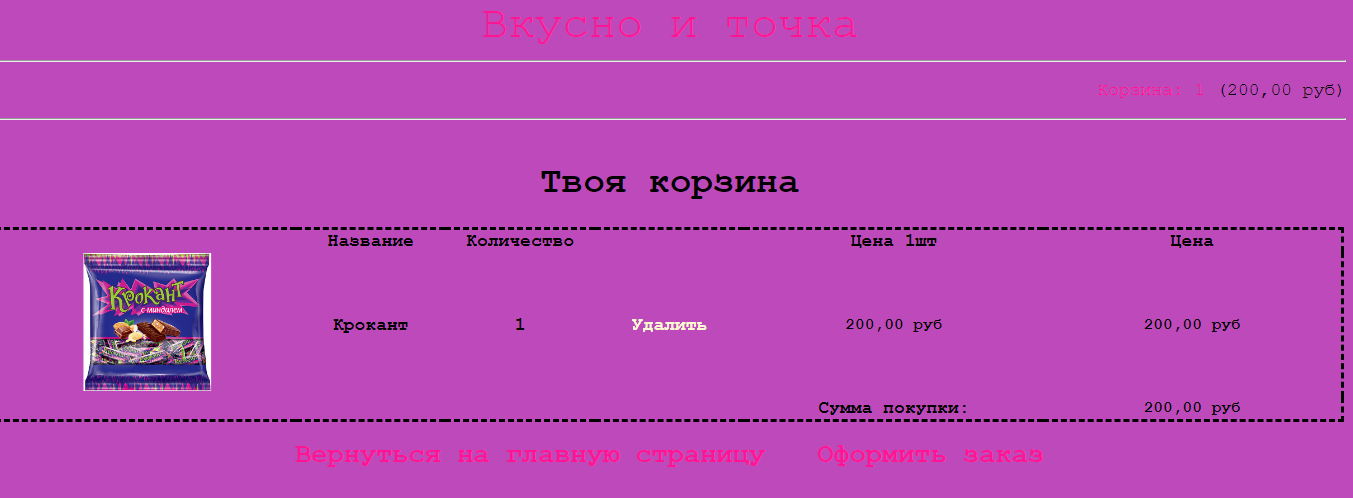


Рисунок 37- Страница Корзины

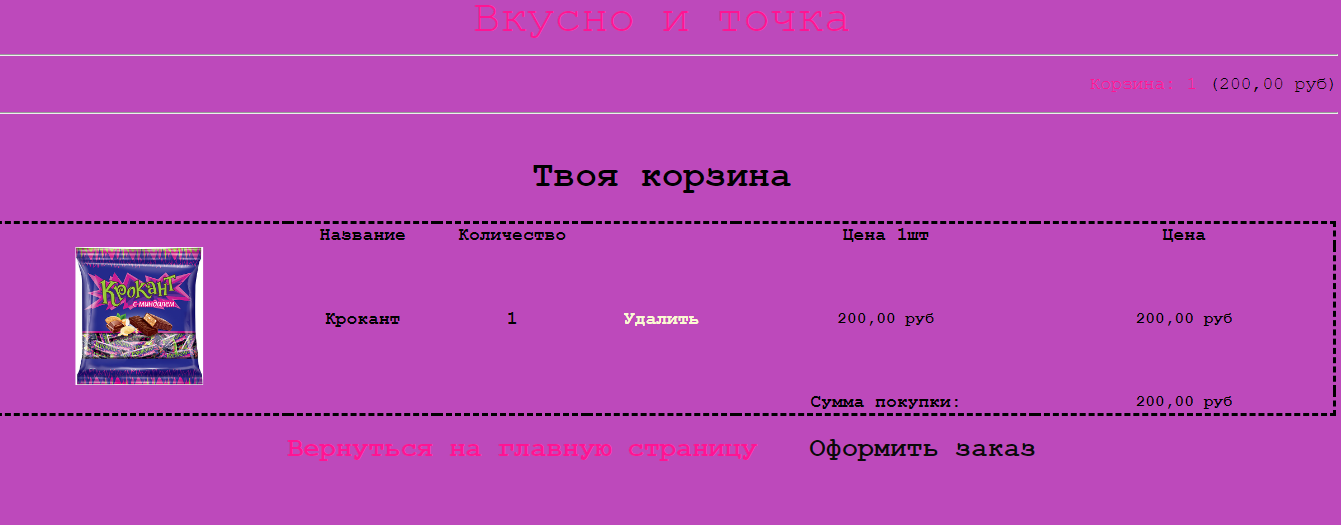


Рисунок 38-Наведение на кнопку оформление заказа

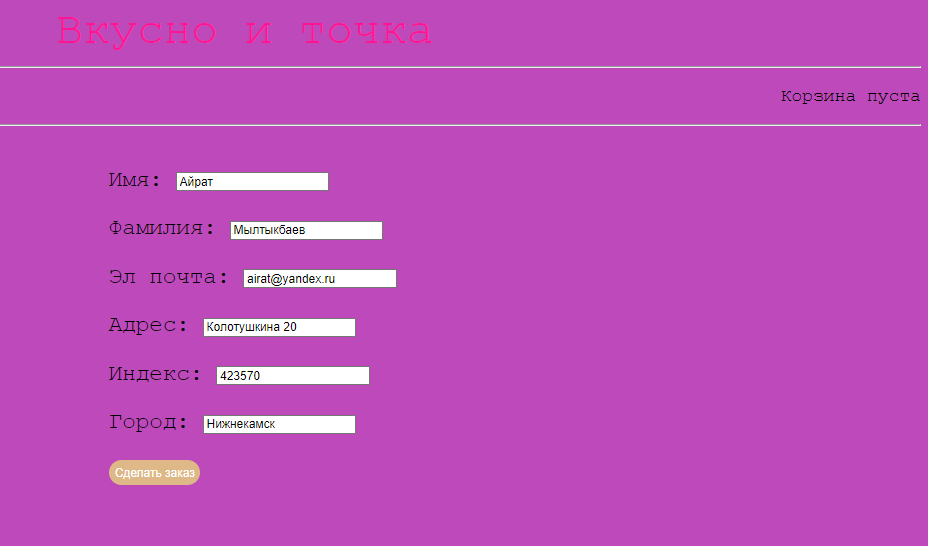


Рисунок 39- Заполнение данных заказчика

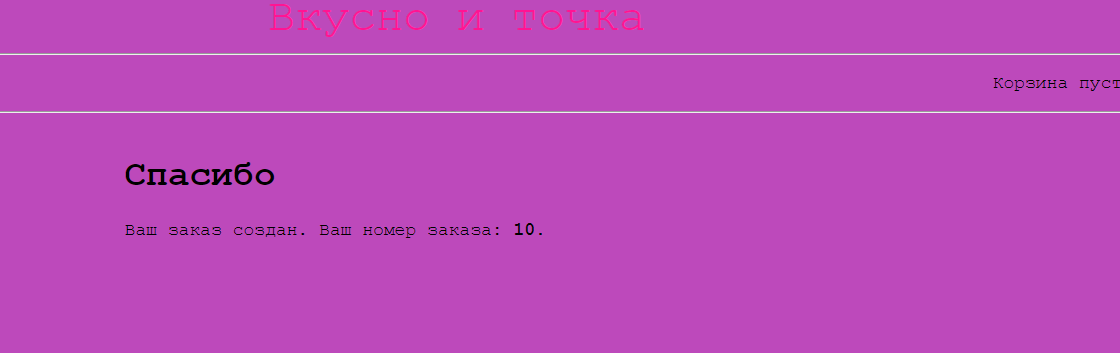


Рисунок 40-Страница успешного заказа.

**Вывод:**

Научились созданию моделей каталога продуктов, добавление их на сайт администрирования и созданию основных представлений для отображения каталога, созданию системы корзины для покупок с помощью сессий Джанго, чтобы пользователи могли сохранять выбранные продукты при просмотре сайта, созданию форм и функциональных возможностей для размещения заказов, отправке по электронной почте подтверждение пользователям при размещении заказа. Размещение товаров по категориям. Создание уникальной иконки сайта.