|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МТУ) |

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Объектно-ориентированное программирование»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту (работе) на тему:**

**Разработка Web приложения «Интернет-магазин по продаже пиццы» при помощи Django Framework**

Студент: Кузьмин М.А.

Группа БСБО-01-15 шифр 15Б0158

Проект (работа) защищен(а) на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

МОСКВА 2017 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МТУ) |

Институт ИКБСП специальность (направление) 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Объектно-ориентированное программирование»

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)**

Кузьмин М.А. шифр 15Б0158 группа БСБО-01-15

1 Тема:

**Разработка Web приложения «Интернет-магазин по продаже пиццы» при помощи Django Framework**

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

Высокоуровневый язык программирования Python

4 Содержание пояснительной записки:

Титульный лист

Содержание

Введение

1 Исследовательский раздел

2 Составляющие проекта

3 Программное обеспечение

4 Техническое задание

Заключение

Приложения

Список использованных источников

Руководитель проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

Содержание

[Цели 2](#_Toc504092661)

[Задачи 2](#_Toc504092662)

[Введение 3](#_Toc504092663)

[Техническое задание 4](#_Toc504092664)

[Основания для разработки 4](#_Toc504092665)

[Назначение разработки 4](#_Toc504092666)

[Требования к программе 4](#_Toc504092667)

[1. Требования к функциональным характеристикам 4](#_Toc504092668)

[2. Требования к составу и параметрам технических средств 5](#_Toc504092669)

[3. Требования к информационной и программной совместимости. 5](#_Toc504092670)

[Стадии и этапы разработки 5](#_Toc504092671)

[Разработка базы данных 6](#_Toc504092672)

[Тестирование приложения 9](#_Toc504092673)

[Заключение 18](#_Toc504092674)

[Приложение 1. Исходный код 19](#_Toc504092675)

[Список использованной литературы 64](#_Toc504092676)

Цели

Разработать Web-приложение с применением фреймворка Django, осуществляющее функции интрнет-магазина по продаже пиццы.

Задачи

Для реализации функций приложения следует разработать:

* Организацию оформления заказа с помощью технологий JQuery, Ajax.
* Организацию отображения товара на странице каталога.
* Выбор размера пиццы с помощью технологии JQuery.
* Указание количества товаров для заказа с помощью технологии JQuery.
* Организация добавления товаров в корзину с помощью технологий JQuery, Ajax.
* Организация корзины с возможностью удаления товаров из неё с помощью технологии JQuery.

Введение

В условиях современной стремительной жизни фастфуд стал одной из составляющих человеческой деятельности. Блюда быстрого приготовления существуют в национальной кухне различных стран мира. Преимуществом этих блюд является быстрота приготовления и употребления, дешевизна (относительно обычных ресторанов с официантами). С одной стороны, это находит отклик у потребителя (особенно в крупных городах с насыщенным ритмом жизни). С другой стороны, такая пища очень технологична, что позволяет сетям фастфуд быстро развиваться.

Именно глобальная востребованность фастфуда в современном мире привела к возникновению необходимости в новых инструментах оптимизации большинства процессов взаимодействия заказчика с компанией.

В основе разработанного Web-приложения лежит идея о создании быстрого, эффективного и удобного средства организации этих самых процессов, которое бы существенно упрощало работу обеим сторонам.

Для реализации данной идеи была применена технология Django (Джанго) — свободный фреймворк для Web-приложений на языке Python, использующий шаблон проектирования MVC.

Техническое задание

Основания для разработки

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «Информационные системы и Технологии» кафедры КБ4 «Автоматизированные системы управления», Московский технологический университет (МТУ), 2017 г.

Назначение разработки

Разработанное Web-приложение предназначено для организации взаимодействия клиента с компанией по изготовлению и доставке пиццы. Предполагается повышение эффективности труда за счет автоматизации процесса заказа «товара» клиентом, исключая оператора.

Требования к программе

К разработанной программе предъявляются следующие требования:

1. Требования к функциональным характеристикам

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* 1. Организация оформления заказа.
  2. Организация отображения товара на странице каталога.
  3. Выбор размера пиццы.
  4. Указание количества товаров для заказа.
  5. Организация добавления товаров в корзину.
  6. Организация корзины с возможностью удаления товаров из неё.

1. Требования к составу и параметрам технических средств
   1. Оперативная память объемом от 2 Гб.
   2. Свободное место на SSD или HDD диске объемом не менее 1 Гб.
   3. Доступ в сеть интернет.
2. Требования к информационной и программной совместимости.
3. Поддержка как минимум одного из браузеров: Google chrome, Яндекс.Браузер, Mozilla Firefox, Opera, Internet Explorer, Safari или их мобильной версии.
4. Наличие: Django Framework версии 1.11.6, PY-Postgresql версии 1.2.1, Psycopg2 версии 2.7.3.1.

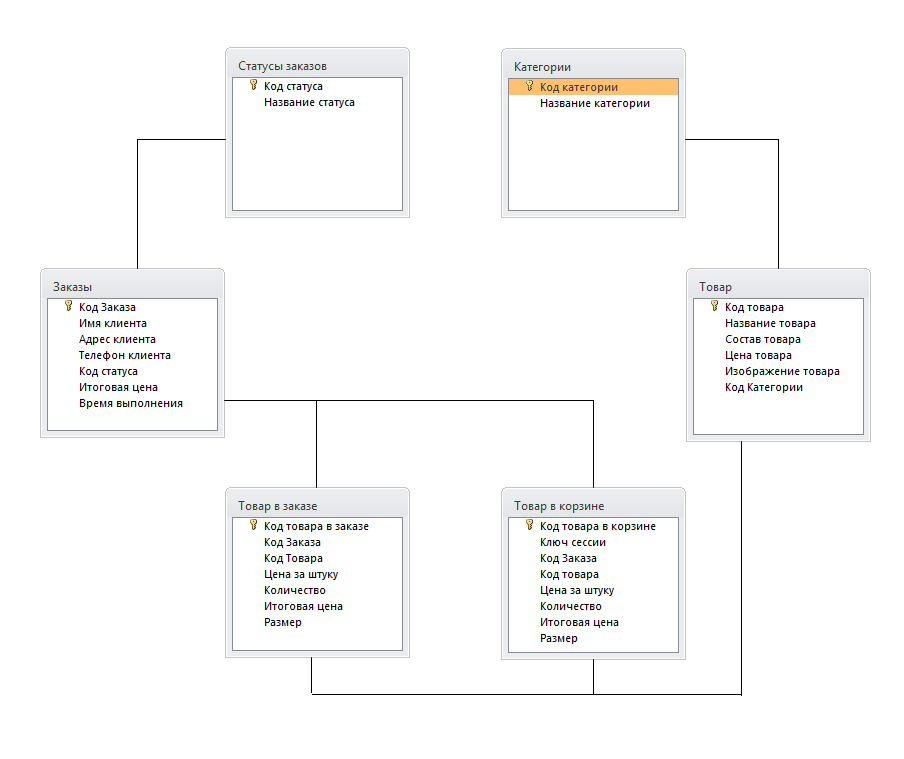
Стадии и этапы разработки

1. Анализ предметной области.
2. Разработка БД.
3. Разработка технического задания на разработку Программы.
4. Разработка структур данных, предназначенных для представления сущностей в оперативной памяти.
5. Разработка программного кода, реализующего функциональные требования к программе.
6. Тестирование приложения.

Разработка программного кода должна происходить на языке программирования Python в среде разработки PyCharm от JetBrains s.r.o.

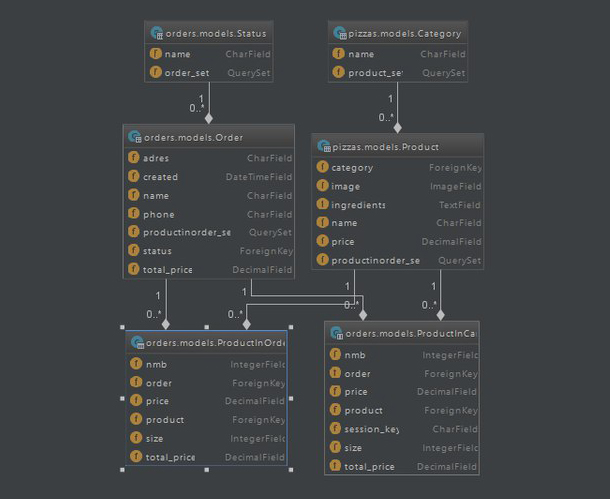
Разработка базы данных

Логическая модель:



*Рис 1. Логическая модель*

Физическая модель:

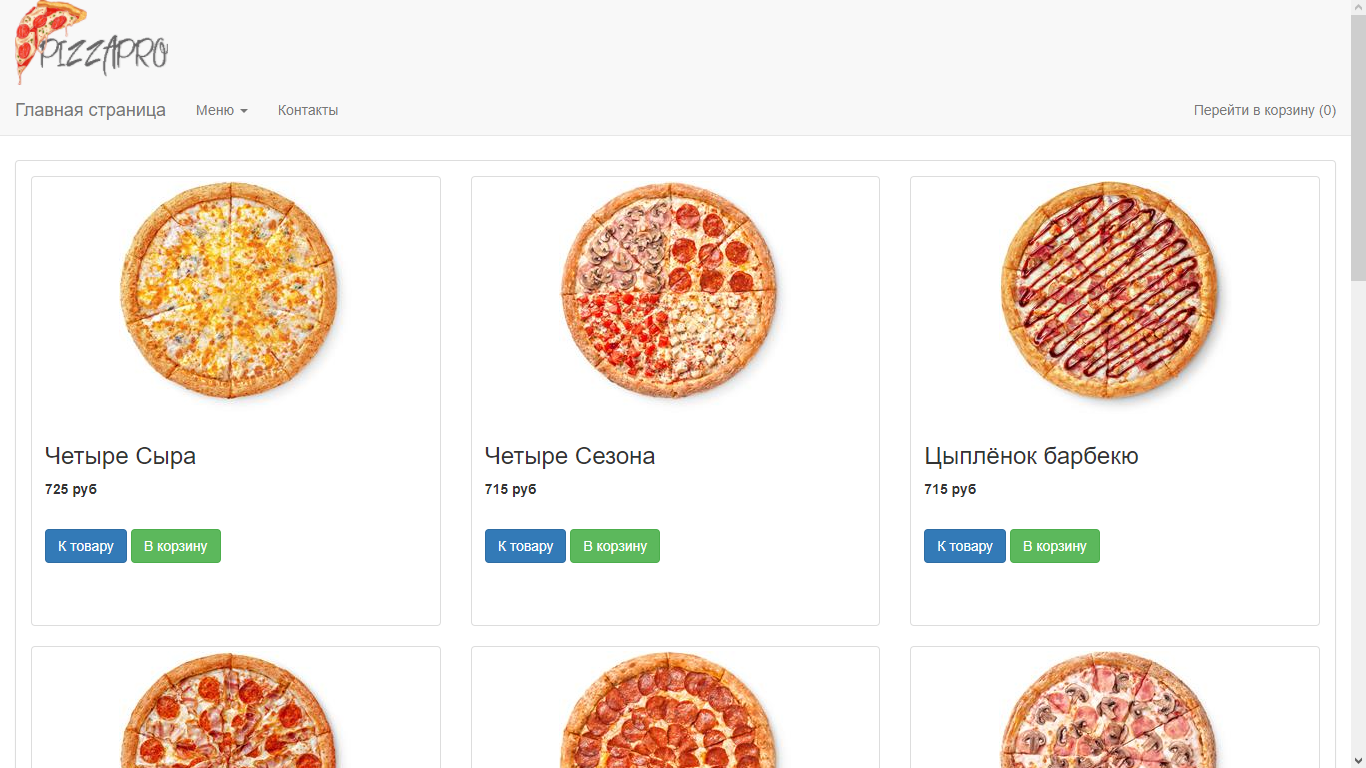


*Рис 2. Физическая модель*

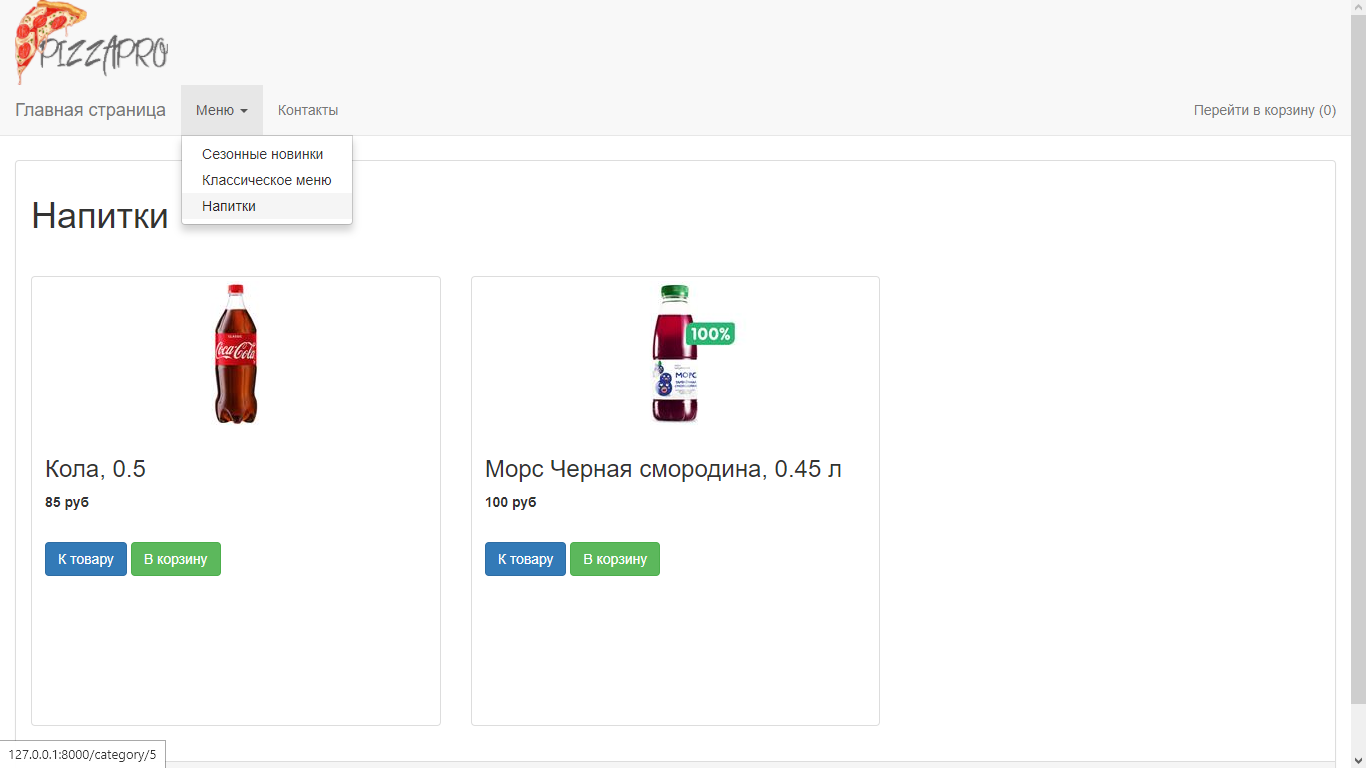
Выделенные сущности представлены в таблице 1:

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Примечание |
| Orders.models.status | | |
| name | Varchar | Название статуса |
| order\_set | Integer | Код статуса |
| pizza.models.category | | |
| name | Varchar | Название категории |
| product\_set | Integer | Код категории |
| Orders.models.order | | |
| Adres | Varchar | Адрес |
| Created | Date | Время выполнения |
| Name | Varchar | Имя |
| Phone | Varchar | Телефон |
| Productionorder\_set | Integer | Код заказа |
| Status | Integer | Код статуса |
| Total\_price | Integer | Итоговая цена |
| pizza.models.product | | |
| Category | Integer | Код категории |
| Image | Varchar | Изображение |
| Ingredients | Varchar | Ингридиенты |
| Name | Varchar | Наименование товара |
| Price | Integer | Цена товара |
| Productionorder\_set | Integer | Код товара |
| Orders.models.ProductionOrders | | |
| Nmb | Integer | Количество |
| Order | Integer | Код заказа |
| Price | Integer | Цена |
| Product | Integer | Код продукта |
| Size | Integer | Размер |
| Total\_price | Integer | Итоговая цена |
| Orders.models.ProductionCategory | | |
| Nmb | Integer | Количество |
| Order | Integer | Код заказа |
| Price | Integer | Цена |
| Product | Integer | Код продукта |
| Session\_key | Integer | Ключ сессии |
| Size | Integer | Размер |
| Total\_price | Integer | Итоговая цена |

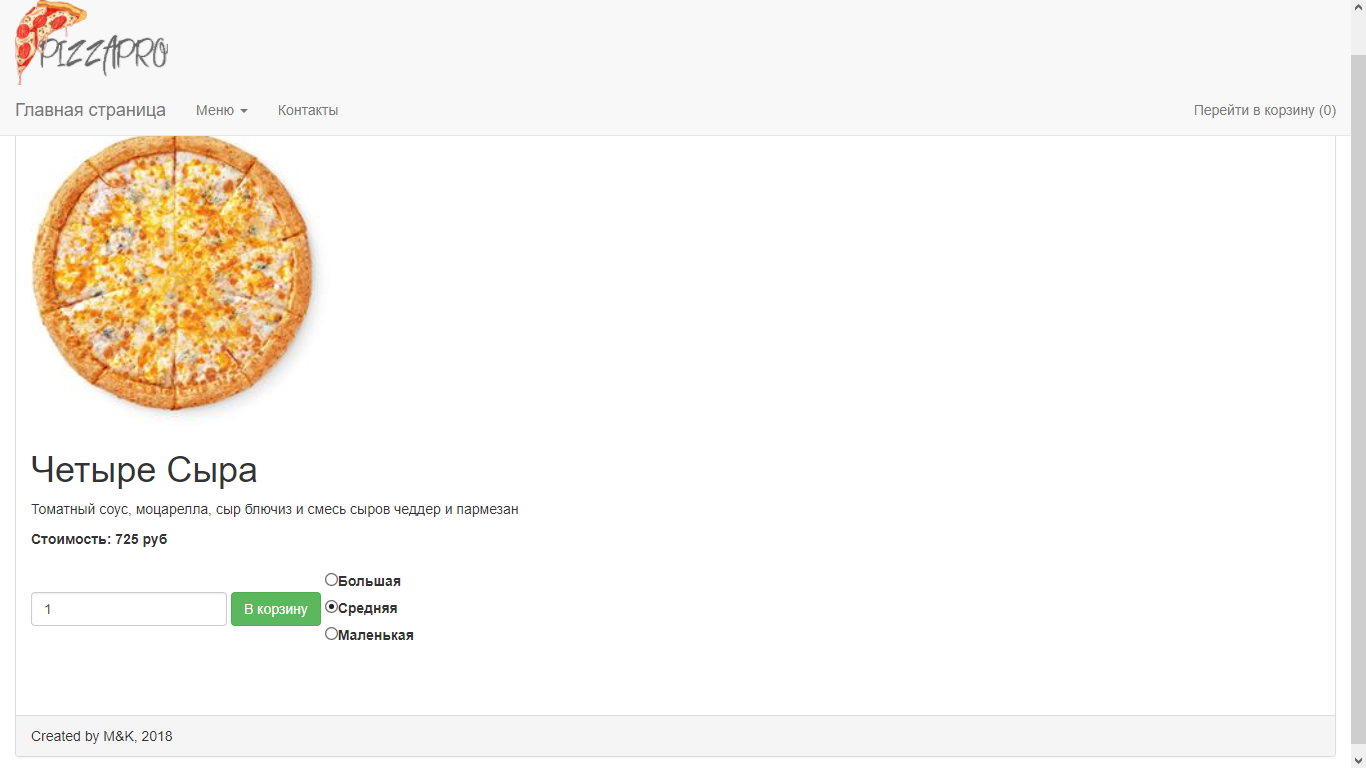
Тестирование приложения



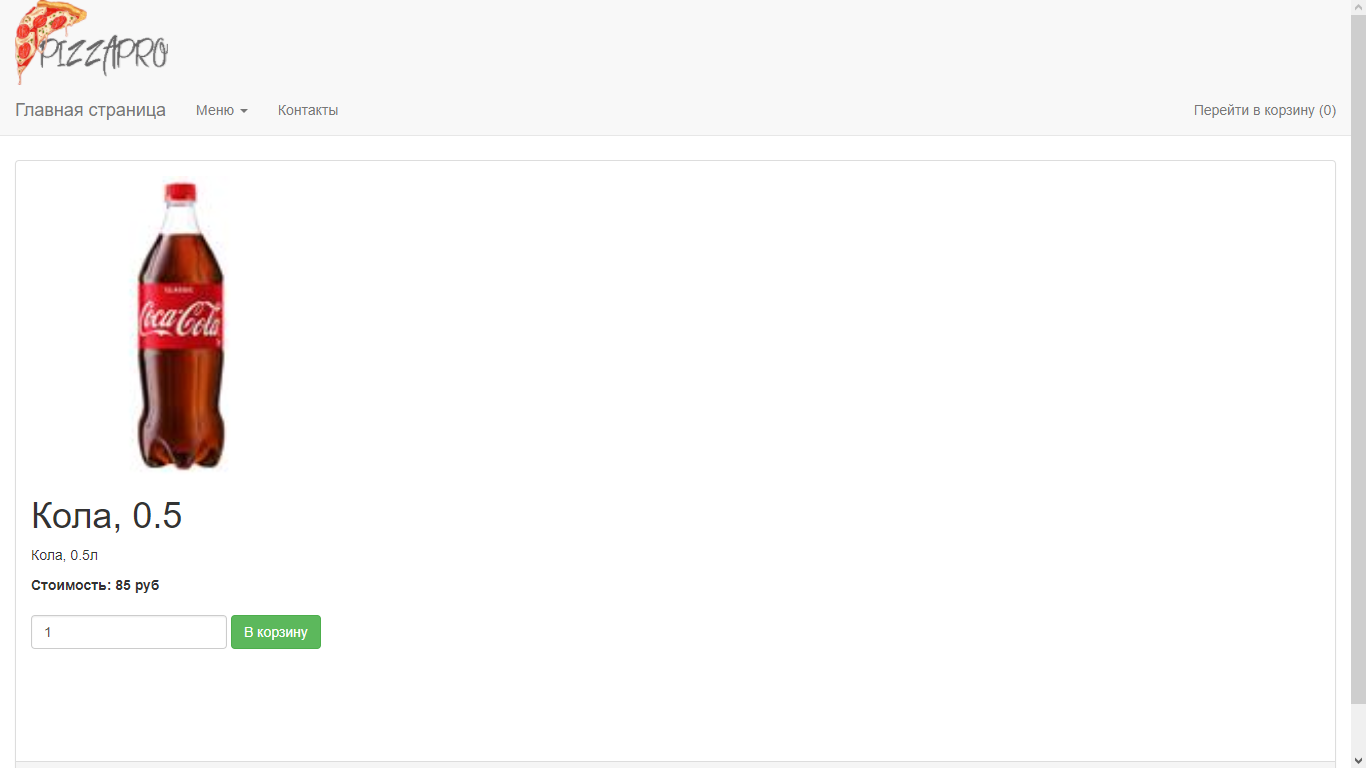
*Рис 3. Главная страница*



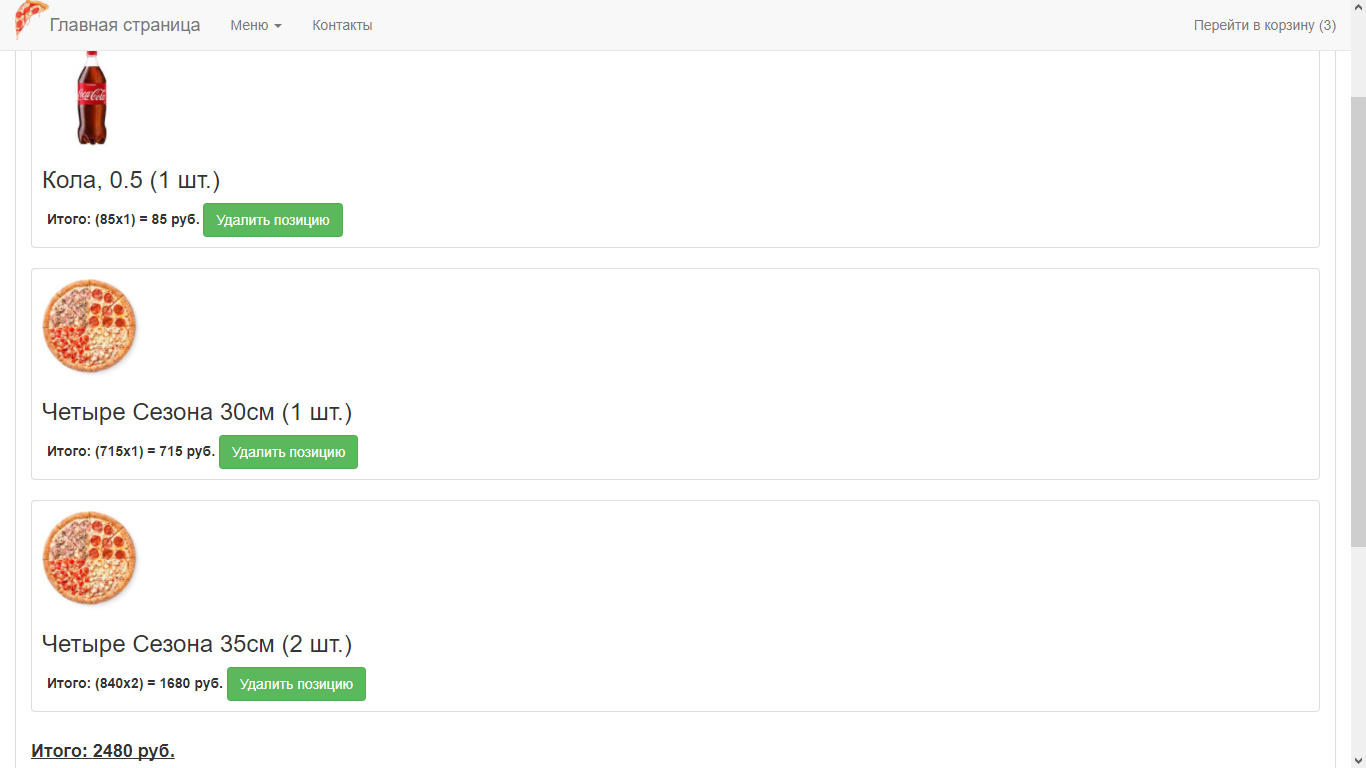
*Рис 4. Страница отображения товаров выбранной категории*



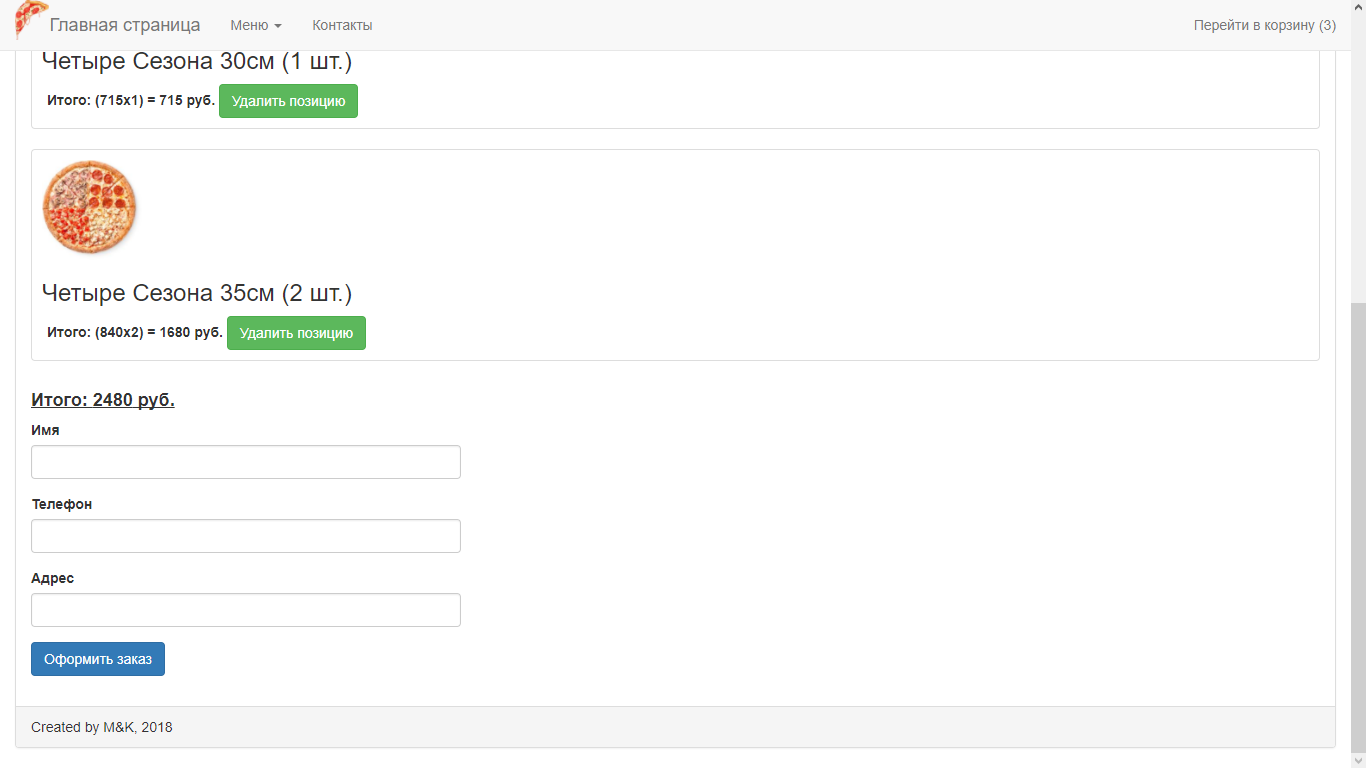
*Рис 5. Страница товара (не напиток) с формой выбора количества товара и его размера*



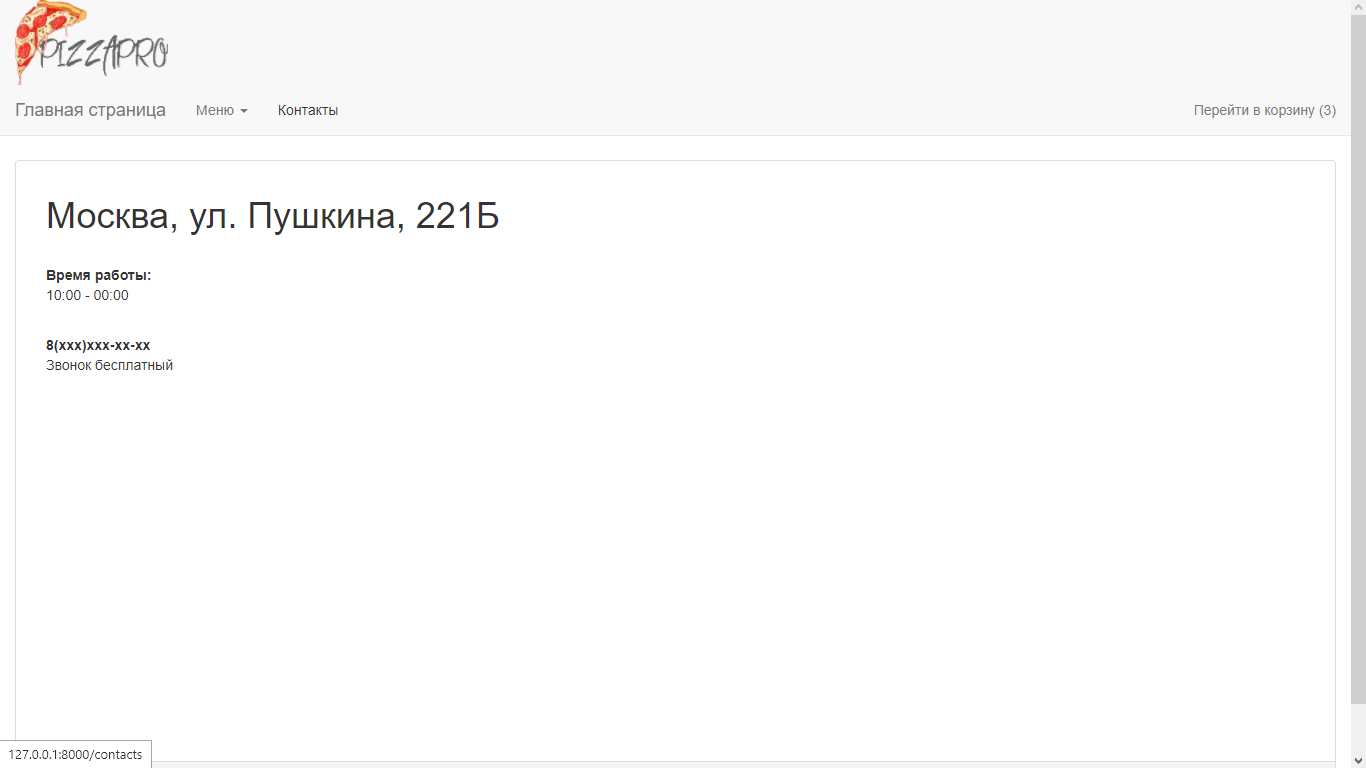
*Рис 6. Страница товара (напиток) c формой выбора количества товара*



*Рис 7. Страница корзины*



*Рис 8. Форма оформления заказа*



*Рис. 9. Страница контактов*

Заключение

В результате проделанной работы было создано Web-приложение, согласно требованиям, описанным в техническом задании.

При выполнении, для создания программы была использована интегрированная среда разработки PyCharm. В ходе работы был разработан понятный и удобный для использования пользовательский интерфейс. Готовое приложение обеспечивает комфортную работу и выполнение требуемых от ПО функций.

Таким образом, при использовании данного Web-приложения в процессе сотрудничества, прогнозируется повышение производительности труда, облегчение взаимодействия.

В ходе написания курсового проекта была использована разнообразная профильная литература и Интернет-ресурсы о разработке и алгоритмах работы Django.

Приложение 1. Исходный код

Исходный код программы:

**pythonkurs/static/js/scripts.js:**

$(document).ready(**function** () {  
 **var** form = $('#form\_number\_products');  
 console.log(form);  
 $('#logo\_min').hide();  
  
 **function** cart\_upd(product\_id, nmb, size, is\_delete){  
 **var** data = {};  
 data.product\_id = product\_id;  
 data.nmb = nmb;  
 data.size = size;  
 **var** csrf\_token = $('#form\_number\_products [name="csrfmiddlewaretoken"]').val();  
 data["csrfmiddlewaretoken"] = csrf\_token;  
  
 **if** (is\_delete){  
 data["is\_delete"]=**true**;  
 }  
  
 **var** url = form.attr("action");  
 console.log(data);  
 $.ajax({  
 url: url,  
 type: 'POST',  
 data: data,  
 cache: **true**,  
 success: **function** (data) {  
 console.log("OK");  
 console.log(data.products\_total\_nmb);  
 **if** (data.products\_total\_nmb){  
 $('#cart\_total\_nmb').text("("+data.products\_total\_nmb+")");  
 }  
 },  
 error: **function** () {  
 console.log('ERROR')  
 }  
 });  
 };  
  
 form.on('submit', **function** (e) {  
 e.preventDefault();  
 **var** nmb = $('#number').val();  
 **if** (document.getElementById('radio\_group')){  
 console.log('is\_size');  
 **var** size = $('input[name = size]:checked', '#form\_number\_products').val();  
 }  
 **else** {  
 console.log('no\_size');  
 **var** size = '30';  
 }  
 console.log(size);  
 console.log(nmb);  
 **var** submit\_btn = $('#submit\_btn');  
 **var** product\_id = submit\_btn.data("product\_id");  
 **var** product\_name = submit\_btn.data("name");  
 **var** product\_price = submit\_btn.data("price");  
 console.log(product\_id);  
 console.log(product\_name);  
 console.log(product\_price);  
 cart\_upd(product\_id, nmb, size, is\_delete=**false**)  
 });  
  
 $('input[name = size]').on('change', **function** () {  
 **var** base\_price = document.getElementById('item\_price').innerHTML;  
 console.log(base\_price);  
 $('#item\_price').hide();  
 **if** ($(**this**).val()==35){  
 **var** change\_price = Number(base\_price)+125;  
 $('#change\_price').text(change\_price);  
 }  
 **else if** ($(**this**).val()==30){  
 $('#change\_price').text(base\_price);  
 }  
 **else if** ($(**this**).val()==25){  
 **var** change\_price = Number(base\_price)-125;  
 $('#change\_price').text(change\_price);  
 }  
 });  
  
 $(document).on('click', '#del\_button', **function** (e) {  
 e.preventDefault();  
 **var** product\_id = $(**this**).data("product\_id");  
 console.log(product\_id);  
 **var** nmb = 0;  
 **var** size = 0;  
 cart\_upd(product\_id, nmb, size, is\_delete = **true**);  
 location.reload();  
 });  
  
 $(document).on('click', '#ind\_btn', **function** (e) {  
 e.preventDefault();  
 **var** product\_id = $(**this**).data("product\_id");  
 **var** nmb = 1;  
 cart\_upd(product\_id, nmb, size = 30, is\_delete = **false**);  
 });  
   
 **function** calcCartPrice() {  
 **var** cart\_total\_price = 0;  
 **var** elems = document.getElementsByClassName('item\_total\_price');  
 **for** (**var** i = 0; i<elems.length; i++){  
 **var** n = (elems[i]).innerHTML;  
 cart\_total\_price+=Number(n);  
 }  
 console.log(cart\_total\_price);  
 $('#cart\_total\_price').text(cart\_total\_price)  
 }  
  
 $(window).scroll(**function** () {  
 **if**($(**this**).scrollTop()>110){  
 $('#navbar\_head').hide();  
 $('#logo\_min').show();  
 }  
 **else** {  
 $('#navbar\_head').show();  
 $('#logo\_min').hide();  
 }  
 });  
 calcCartPrice();  
});

**Pythonkurs/static/css/styles.css:**

body{  
 padding-top: 160px;  
}  
#main\_panel{  
 min-height: 600px;  
}  
#item\_bage{  
 min-height: 450px;  
}  
.error\_span{  
 color: red;  
}

**pythonkurs/pythonkurs/settings.py:**

*"""  
Django settings for pythonkurs project.  
  
Generated by 'django-admin startproject' using Django 2.0.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/2.0/topics/settings/  
  
For the full list of settings and their values, see  
https://docs.djangoproject.com/en/2.0/ref/settings/  
"""*import os  
  
# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)  
BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))  
  
  
# Quick-start development settings - unsuitable for production  
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/  
  
# SECURITY WARNING: keep the secret key used in production secret!  
SECRET\_KEY = 'c\_j\_z0ovpag+mwr!y5-6c6!1=rpl\_pb=)=#x%&#&h9dy81st#t'  
  
# SECURITY WARNING: don't run with debug turned on in production!  
DEBUG = True  
  
ALLOWED\_HOSTS = []  
  
  
# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'pizza\_main',  
 'pizzas',  
 'orders',  
]  
  
MIDDLEWARE = [  
 'django.middleware.security.SecurityMiddleware',  
 'django.contrib.sessions.middleware.SessionMiddleware',  
 'django.middleware.common.CommonMiddleware',  
 'django.middleware.csrf.CsrfViewMiddleware',  
 'django.contrib.auth.middleware.AuthenticationMiddleware',  
 'django.contrib.messages.middleware.MessageMiddleware',  
 'django.middleware.clickjacking.XFrameOptionsMiddleware',  
]  
  
ROOT\_URLCONF = 'pythonkurs.urls'  
  
TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': ['templates'],  
 'APP\_DIRS': True,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 'django.template.context\_processors.media',  
 'orders.context\_processor.gettings\_cart\_info',  
 ],  
 },  
 },  
]  
  
WSGI\_APPLICATION = 'pythonkurs.wsgi.application'  
  
  
# Database  
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases  
  
DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
 }  
}  
  
  
# Password validation  
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators  
  
AUTH\_PASSWORD\_VALIDATORS = [  
 {  
 'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',  
 },  
 {  
 'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',  
 },  
]  
  
  
# Internationalization  
# https://docs.djangoproject.com/en/2.0/topics/i18n/  
  
LANGUAGE\_CODE = 'en-us'  
  
TIME\_ZONE = 'UTC'  
  
USE\_I18N = True  
  
USE\_L10N = True  
  
USE\_TZ = True  
  
  
# Static files (CSS, JavaScript, Images)  
# https://docs.djangoproject.com/en/2.0/howto/static-files/  
  
STATICFILES\_DIRS = (os.path.join(BASE\_DIR, "static"),)  
  
STATIC\_URL = '/static/'  
  
MEDIA\_URL = '/media/'  
  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

**Pythonkurs/pythonkurs/urls.py:**

*"""pythonkurs URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.contrib.staticfiles.urls import staticfiles\_urlpatterns  
from django.urls import path, re\_path, include  
from django.conf import settings  
from django.conf.urls.static import static  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('pizza\_main.urls'),),  
 path('', include('pizzas.urls'),),  
 path('', include('orders.urls'),),  
]  
if settings.DEBUG:  
  
 if settings.MEDIA\_ROOT:  
  
 urlpatterns += static(settings.MEDIA\_URL,  
  
 document\_root=settings.MEDIA\_ROOT)  
  
  
urlpatterns += staticfiles\_urlpatterns()

**Pythonkurs/pythonkurs//wsgi.py:**

*"""  
WSGI config for pythonkurs project.  
  
It exposes the WSGI callable as a module-level variable named ``application``.  
  
For more information on this file, see  
https://docs.djangoproject.com/en/2.0/howto/deployment/wsgi/  
"""*import os  
  
from django.core.wsgi import get\_wsgi\_application  
  
os.environ.setdefault("DJANGO\_SETTINGS\_MODULE", "pythonkurs.settings")  
  
application = get\_wsgi\_application()

Pythonkurs/pizzas/admin.py:

from django.contrib import admin  
from .models import \*  
# Register your models here.  
  
  
class ProductAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in Product.\_meta.fields]  
  
 class Meta:  
 model = Product  
  
admin.site.register(Product, ProductAdmin)  
  
  
class CategoryAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in Category.\_meta.fields]  
  
 class Meta:  
 model = Category  
  
admin.site.register(Category, CategoryAdmin)

**pythonkurs/pizzas/apps.py:**

from django.apps import AppConfig  
  
  
class PizzasConfig(AppConfig):  
 name = 'pizzas'

**pythonkurs/pizzas/models.py:**

from django.db import models  
  
class Category(models.Model):  
 name = models.CharField(max\_length=50, unique=True)  
  
 class Meta:  
 verbose\_name = 'Категория'  
 verbose\_name\_plural = 'Категории'  
  
 def \_\_str\_\_(self):  
 return '%s %s' % (self.id, self.name)  
  
  
class Product(models.Model):  
 name = models.CharField(max\_length=255, unique=True)  
 ingredients = models.TextField(default=None)  
 price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 image = models.ImageField(upload\_to='product\_image/')  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
  
 class Meta:  
 verbose\_name = 'Товар'  
 verbose\_name\_plural = 'Товары'  
  
 def \_\_str\_\_(self):  
 return '%s' % self.name

**pythonkurs/pizzas/tests.py:**

from django.test import TestCase  
  
# Create your tests here.

**Pythonkurs/pizzas/urls.py:**

*"""pythonkurs URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.urls import path, re\_path, include  
from pizzas import views  
  
urlpatterns = [  
]

**Pythonkurs/pizzas/views.py:**

from django.shortcuts import render  
from pizzas.models import \*  
# Create your views here.

**pythonkurs/pizza\_main/admin.py**:

from django.contrib import admin  
  
# Register your models here.

**pythonkurs/pizza\_main/apps.py:**

from django.apps import AppConfig  
  
  
class PizzaMainConfig(AppConfig):  
 name = 'pizza\_main'

**pythonkurs/pizza\_main/models.py**

from django.db import models  
  
# Create your models here.

**Pythonkurs/pizza\_main/tests.py**

from django.test import TestCase  
  
# Create your tests here.

**pythonkurs/pizza\_main/urls.py:**

*"""pythonkurs URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.urls import path, re\_path, include  
from pizza\_main import views  
  
urlpatterns = [  
 path('', views.home, name='home'),  
 re\_path(r'^item/(?P<id>[^/]+)', views.item, name='item'),  
 path('contacts/', views.contacts, name='contacts'),  
 re\_path(r'^category/(?P<id>[^/]+)', views.get\_category, name='get\_category')  
  
]

**Pythonkurs/pizza\_main/views.py**:

from django.shortcuts import render  
from django.http import HttpResponse, Http404  
from pizzas.models import \*  
from django.template.loader import render\_to\_string  
  
# Create your views here.  
  
  
def home(request):  
 tovar = Product.objects.all()  
 category\_list = Category.objects.all()  
 context = {  
 'tovars' : tovar,  
 'categories' : category\_list,  
 }  
 return render(request, 'index.html', context)  
  
  
def get\_category(request, id):  
 try:  
 category = Category.objects.get(id = id)  
 tovar = Product.objects.filter(category=category)  
 category\_list = Category.objects.all()  
 except:  
 raise Http404('Category not found')  
 context = {  
 'tovars' : tovar,  
 'category' : category,  
 'categories' : category\_list,  
 }  
 return render(request, 'index\_category.html', context)  
  
  
def item(request, id):  
 session\_key = request.session.session\_key  
 if not session\_key:  
 request.session.cycle\_key()  
  
 print(request.session.session\_key)  
 try:  
 tovar = Product.objects.get(id=id)  
 category\_list = Category.objects.all()  
 except:  
 raise Http404('Object not found')  
 context = {  
 'tovar' : tovar,  
 'categories' : category\_list,  
 }  
 return render(request, 'item.html', context)  
  
def contacts(request):  
 category\_list = Category.objects.all()  
 context = {  
 'categories' : category\_list,  
 }  
 return render(request, 'contacts.html', context)

**pythonkurs/orders/admin.py:**

from django.contrib import admin  
from .models import \*  
# Register your models here.  
  
  
class OrderAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in Order.\_meta.fields]  
  
 class Meta:  
 model = Order  
  
admin.site.register(Order, OrderAdmin)  
  
  
class StatusAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in Status.\_meta.fields]  
  
 class Meta:  
 model = Status  
  
  
admin.site.register(Status, StatusAdmin)  
  
  
class ProductInOrderAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in ProductInOrder.\_meta.fields]  
  
 class Meta:  
 model = ProductInOrder  
  
  
admin.site.register(ProductInOrder, ProductInOrderAdmin)  
  
  
class ProductInCartAdmin(admin.ModelAdmin):  
 list\_display = [field.name for field in ProductInCart.\_meta.fields]  
  
 class Meta:  
 model = ProductInCart  
  
admin.site.register(ProductInCart, ProductInCartAdmin)

**pythonkurs/orders/apps.py:**

from django.apps import AppConfig  
  
  
class OrdersConfig(AppConfig):  
 name = 'orders'

**pythonkurs/orders/context\_processor**:

from .models import ProductInCart  
  
  
def gettings\_cart\_info(request):  
 session\_key = request.session.session\_key  
 if not session\_key:  
 request.session.cycle\_key()  
 products\_in\_cart = ProductInCart.objects.filter(session\_key=session\_key)  
 products\_total\_nmb = products\_in\_cart.count()  
  
 return locals()

**pythonkurs/orders/forms.py:**

from django import forms  
from .models import \*  
  
  
class CheckForm(forms.Form):  
 name = forms.CharField(required=True)  
 phone = forms.CharField(required=True)  
 adres = forms.CharField(required=True)

**pythonkurs/orders/models.py**:

from django.db import models  
from pizzas.models import Product  
from django.db.models.signals import post\_save  
from django.contrib.auth.models import User  
  
  
# Create your models here.  
  
  
class Status(models.Model):  
 name = models.CharField(max\_length=24)  
  
 class Meta:  
 verbose\_name = 'Статус заказа'  
 verbose\_name\_plural = 'Статусы заказа'  
  
 def \_\_str\_\_(self):  
 return 'Статус %s' % self.name  
  
  
class Order(models.Model):  
 name = models.CharField(max\_length=255, verbose\_name='Имя покупателя')  
 phone = models.CharField(max\_length=48, default=None, null=True)  
 adres = models.CharField(max\_length=255)  
 status = models.ForeignKey(Status, on\_delete=models.CASCADE)  
 total\_price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 created = models.DateTimeField(auto\_now\_add=True)  
  
 class Meta:  
 verbose\_name = 'Заказ'  
 verbose\_name\_plural = 'Заказы'  
  
 def \_\_str\_\_(self):  
 return "Заказ %s %s" % (self.id, self.status.name)  
  
 def save(self, \*args, \*\*kwargs):  
 super(Order, self). save(\*args, \*\*kwargs)  
  
  
class ProductInOrder(models.Model):  
 order = models.ForeignKey(Order, default=None, on\_delete=models.CASCADE, blank=True, null=True)  
 product = models.ForeignKey(Product, default=None, on\_delete=models.CASCADE, blank=True, null=True)  
 price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 nmb = models.IntegerField(default=1, verbose\_name='Количество товара в заказе')  
 total\_price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 size = models.IntegerField(default=30, verbose\_name='Размер(см)')  
  
 class Meta:  
 verbose\_name = 'Товар в заказе'  
 verbose\_name\_plural = "Товары в заказах"  
  
 def \_\_str\_\_(self):  
 return '%s' % self.product.name  
  
 def save(self, \*args, \*\*kwargs):  
 price\_per\_item = self.product.price  
 item\_category = self.product.category  
 item\_size = self.size  
 self.price = price\_per\_item  
 if item\_category.name != "Напитки":  
 if int(item\_size) < 30:  
 self.price = self.price - 125  
 self.total\_price = int(self.nmb) \* (price\_per\_item-125)  
 elif int(item\_size) > 30:  
 self.price = self.price + 125  
 self.total\_price = int(self.nmb) \* (price\_per\_item+125)  
 else:  
 self.total\_price = int(self.nmb)\*price\_per\_item  
 else:  
 self.total\_price = int(self.nmb)\*price\_per\_item  
  
 super(ProductInOrder, self).save(\*args, \*\*kwargs)  
 order = self.order  
 all\_products\_in\_order=ProductInOrder.objects.filter(order=order)  
 order\_total\_price = 0  
 for item in all\_products\_in\_order:  
 order\_total\_price += item.total\_price  
  
 self.order.total\_price=order\_total\_price  
 self.order.save(force\_update=True)  
  
  
class ProductInCart(models.Model):  
 session\_key = models.CharField(max\_length=128, editable=False)  
 order = models.ForeignKey(Order, default=None, on\_delete=models.CASCADE, blank=True, null=True)  
 product = models.ForeignKey(Product, default=None, on\_delete=models.CASCADE, blank=True, null=True)  
 price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 nmb = models.IntegerField(default=1, verbose\_name='Количество товара в заказе')  
 total\_price = models.DecimalField(max\_digits=6, decimal\_places=2, default=0)  
 size = models.IntegerField(default=30, verbose\_name='Размер пиццы(диаметр в см)')  
  
 class Meta:  
 verbose\_name = 'Товар в корзине'  
 verbose\_name\_plural = "Товары в корзинах"  
  
 def \_\_str\_\_(self):  
 return '%s' % self.product.name  
  
 def save(self, \*args, \*\*kwargs):  
 price\_per\_item = self.product.price  
 item\_size = self.size  
 item\_category = self.product.category  
 self.price = price\_per\_item  
 if item\_category.name != "Напитки":  
 if int(item\_size) < 30:  
 self.price = self.price - 125  
 self.total\_price = int(self.nmb) \* (price\_per\_item-125)  
 elif int(item\_size) > 30:  
 self.price = self.price + 125  
 self.total\_price = int(self.nmb) \* (price\_per\_item+125)  
 else:  
 self.total\_price = int(self.nmb)\*price\_per\_item  
 else:  
 self.total\_price = int(self.nmb)\*price\_per\_item  
  
 super(ProductInCart, self).save(\*args, \*\*kwargs)

**pythonkurs/orders/tests.py**

from django.test import TestCase  
  
# Create your tests here.

**Pythonkurs/orders/urls.py**

*"""pythonkurs URL Configuration  
  
The `urlpatterns` list routes URLs to views. For more information please see:  
 https://docs.djangoproject.com/en/2.0/topics/http/urls/  
Examples:  
Function views  
 1. Add an import: from my\_app import views  
 2. Add a URL to urlpatterns: path('', views.home, name='home')  
Class-based views  
 1. Add an import: from other\_app.views import Home  
 2. Add a URL to urlpatterns: path('', Home.as\_view(), name='home')  
Including another URLconf  
 1. Import the include() function: from django.urls import include, path  
 2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))  
"""*from django.contrib import admin  
from django.urls import path, re\_path, include  
from . import views  
  
urlpatterns = [  
 path('cart\_add/', views.cart\_add, name='cart\_add'),  
 path('cart/', views.cart\_page, name='cart\_page'),  
]

**Pythonkurs/orders/views.py**

from django.shortcuts import render  
from django.http import JsonResponse  
from .models import ProductInCart, Order, ProductInOrder  
from pizzas.models import Category  
from .forms import CheckForm  
  
  
def cart\_add(request):  
 return\_dict = dict()  
 session\_key = request.session.session\_key  
 print(request.POST)  
 data = request.POST  
 product\_id = data.get("product\_id")  
 nmb = data.get("nmb")  
 size = data.get("size")  
 is\_delete = data.get("is\_delete")  
  
 if is\_delete:  
 ProductInCart.objects.filter(id=product\_id).delete()  
 else:  
 new\_product, created = ProductInCart.objects.get\_or\_create(size=size, session\_key=session\_key, product\_id=product\_id, defaults={"nmb" : nmb})  
 if not created:  
 new\_product.nmb += int(nmb)  
 new\_product.save(force\_update=True)  
  
 products\_in\_basket = ProductInCart.objects.filter(session\_key=session\_key)  
 products\_total\_nmb = products\_in\_basket.count()  
  
 return\_dict["products\_total\_nmb"] = products\_total\_nmb  
 return JsonResponse(return\_dict)  
  
  
def cart\_page(request):  
 session\_key = request.session.session\_key  
 items = ProductInCart.objects.filter(session\_key=session\_key)  
 category\_list = Category.objects.all()  
 form = CheckForm(request.POST or None)  
 context = {  
 'items' : items,  
 'categories' : category\_list,  
 'form' : form,  
 }  
 if request.POST:  
 if form.is\_valid():  
 print(request.POST)  
 data = request.POST  
 name = data.get("name")  
 phone = data["phone"]  
 adres = data["adres"]  
 order = Order.objects.create(name=name, adres=adres, phone=phone, status\_id=2)  
 for item in items:  
 item\_name=item.product  
 item\_nmb = int(item.nmb)  
 item\_total\_price = item.total\_price  
 item\_price = item.price  
 order = order  
 item\_size = item.size  
 ProductInOrder.objects.create(product = item\_name, nmb = item\_nmb, price = item\_price, total\_price = item\_total\_price, order=order, size = item\_size)  
 ProductInCart.objects.filter(session\_key=session\_key).delete()  
 else:  
 print('error')  
  
 return render(request, 'cart.html', context)

**pythonkurs/templates/cart.html**

{% load staticfiles %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="{% static "styles.css" %}">  
 <link rel="stylesheet" type="text/css" href="{% static "Bootstrap3/css/bootstrap.css" %}">  
 <title>Корзина</title>  
</head>  
<body>  
 {% include 'scripts.html' %}  
 {% include 'navbar.html' %}  
 <div class="container-fluid" >  
 <div class="panel panel-default">  
 <div class="panel-body" id="main\_panel">  
 {% if items %}  
 {% for item **in** items %}  
 <div class="row" id="cart\_row{{ item.id }}">  
 <div class="col-sm-12">  
 <ul class="list-group">  
 <li class="list-group-item">  
 <img src="{{ item.product.image.url }}" alt="" class="list-inline" height="100px">  
 <h3 class="list-inline">{{ item.product.name }} {% if item.product.category.name != "Напитки"%}{{ item.size }}см{% endif %} ({{ item.nmb }} шт.)</h3>  
 <form id="form\_number\_products" class="form-inline" action="{% url 'cart\_add' %}">{% csrf\_token %}  
 <strong>Итого: ({{ item.price }}x{{ item.nmb }}) = <strong class="item\_total\_price">{{ item.total\_price }}</strong> руб.</strong>  
 <button class="btn btn-success" id="del\_button" data-product\_id = {{ item.id }}>Удалить позицию</button>  
 </form>  
 </li>  
 </ul>  
 </div>  
 </div>  
 {% endfor %}  
 <h4><u><strong>Итого: <strong id="cart\_total\_price"></strong> руб.</strong></u></h4>  
 <form action="" class="form-horizontal" method="post" >{% csrf\_token %}  
 <div class="col-lg-4">  
 <div class="form-group">  
 <label for="">Имя</label>  
 <span class="error\_span">{{ form.name.errors }}</span>  
 <input type="text" name="{{ form.name.html\_name }}" class="form-control" value="{{ request.POST.name }}"></input>  
 </div>  
 <div class="form-group">  
 <label for="">Телефон</label>  
 <span class="error\_span">{{ form.phone.errors }}</span>  
 <input type="text" name="{{ form.phone.html\_name }}" class="form-control" value="{{ request.POST.phone }}"></input>  
 </div>  
 <div class="form-group">  
 <label for="">Адрес</label>  
 <span class="error\_span">{{ form.adres.errors }}</span>  
 <input type="text" name="{{ form.adres.html\_name }}" class="form-control" value="{{ request.POST.adres }}"></input>  
 </div>  
 <div class="form-group">  
 <button type="submit" class="btn btn-primary" id="confirm\_btn">Оформить заказ</button>  
 </div>  
 </div>  
 </form>  
 {% else %}  
 <h1 class="text-center">В корзине нет товаров</h1>  
 {% endif %}  
 </div>  
 <div class="panel-footer">Created by M&K, 2018</div>  
 </div>  
 </div>  
  
</body>  
</html>

**pythonkurs/templates/contacts.html**

{% load staticfiles %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="{% static "styles.css" %}">  
 <link rel="stylesheet" type="text/css" href="{% static "Bootstrap3/css/bootstrap.css" %}">  
 <title>Главная страница</title>  
</head>  
<body>  
 {% include 'scripts.html' %}  
 {% include 'navbar.html' %}  
 <div class="container-fluid">  
 <div class="panel panel-default">  
 <div class="panel-body" id="main\_panel">  
 <div class="col-sm-12">  
 <h1>Москва, ул. Пушкина, 221Б</h1>  
 <br><strong>Время работы:</strong>  
 <p>10:00 - 00:00</p>  
 <br><strong>8(xxx)xxx-xx-xx</strong>  
 <p>Звонок бесплатный</p>  
 </div>  
 </div>  
 <div class="panel-footer">Created by M&K, 2018</div>  
 </div>  
 </div>  
</body>  
</html>

**pythonkurs/templates/index.html**

{% load staticfiles %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="{% static "styles.css" %}">  
 <link rel="stylesheet" type="text/css" href="{% static "Bootstrap3/css/bootstrap.css" %}">  
 <title>Главная страница</title>  
</head>  
<body>  
 {% include 'scripts.html' %}  
 {% include 'navbar.html' %}  
 <div class="container-fluid" >  
 <div class="panel panel-default">  
 <div class="panel-body" id="main\_panel">  
 <div class="row">  
 {% for tovar **in** tovars %}  
 <div class="col-sm-6 col-md-4" >  
 <div class="thumbnail" id="item\_bage">  
 <img src="{{ tovar.image.url}}" alt="...">  
 <div class="caption">  
 <h3>{{ tovar.name }}</h3>  
 <strong>{{ tovar.price }} руб</strong>  
 <p><br>  
 <form id="form\_number\_products" class="form-inline" action="{% url 'cart\_add' %}">{% csrf\_token %}  
 <a href="/item/{{ tovar.id }}" class="btn btn-primary" role="button">К товару</a>  
 <butoon class="btn btn-success btn-buy" id="ind\_btn" data-product\_id = {{ tovar.id }}>  
 В корзину  
 </butoon>  
 </form>  
 </p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 </div>  
 <div class="panel-footer">Created by M&K, 2018</div>  
 </div>  
 </div>  
</body>  
</html>

**pythonkurs/templates/index\_category.html**

{% load staticfiles %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="{% static "styles.css" %}">  
 <link rel="stylesheet" type="text/css" href="{% static "Bootstrap3/css/bootstrap.css" %}">  
 <title>{{ category.name }}</title>  
</head>  
<body>  
 {% include 'scripts.html' %}  
 {% include 'navbar.html' %}  
 <div class="container-fluid" >  
 <div class="panel panel-default" >  
 <div class="panel-body" id="main\_panel">  
 <div class="row">  
 <div class="col-sm-6">  
 <h1>{{ category.name }}</h1>  
 </div>  
 <br>  
 <br>  
 <br>  
 <br>  
 {% for tovar **in** tovars %}  
 <div class="col-sm-6 col-md-4">  
 <h1>{{ category\_list.name }}</h1>  
 <div class="thumbnail" id="item\_bage">  
 <img src="{{ tovar.image.url}}" alt="...">  
 <div class="caption">  
 <h3>{{ tovar.name }}</h3>  
 <strong>{{ tovar.price }} руб</strong>  
 <p><br>  
 <form id="form\_number\_products" class="form-inline" action="{% url 'cart\_add' %}">{% csrf\_token %}  
 <a href="/item/{{ tovar.id }}" class="btn btn-primary" role="button">К товару</a>  
 <butoon class="btn btn-success btn-buy" id="ind\_btn" data-product\_id = {{ tovar.id }}>  
 В корзину  
 </butoon>  
 </form>  
 </p>  
 </div>  
 </div>  
 </div>  
 {% endfor %}  
 </div>  
 </div>  
 <div class="panel-footer">Created by M&K, 2018</div>  
 </div>  
 </div>  
</body>  
</html>

**pythonkurs/templates/item.html**

{% load staticfiles %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">  
 <meta name="viewport" content="width=device-width, initial-scale=1">  
 <link rel="stylesheet" type="text/css" href="{% static "styles.css" %}">  
 <link rel="stylesheet" type="text/css" href="{% static "Bootstrap3/css/bootstrap.css" %}">  
 <title>{{ tovar.name }}</title>  
</head>  
<body>  
 {% include 'scripts.html' %}  
 {% include 'navbar.html' %}  
 <div class="container-fluid" >  
 <div class="panel panel-default">  
 <div class="panel-body" id="main\_panel">  
 <div class="row">  
 <div class="col-sm-8">  
 <img src="{{ tovar.image.url}}" alt="" width="300px">  
 <h1>{{ tovar.name }}</h1>  
 <p>{{ tovar.ingredients }}</p>  
 <strong>Стоимость: <strong id="item\_price">{{ tovar.price }}</strong><strong id="change\_price"></strong> руб </strong>  
 <div>  
 <br>  
 <form id="form\_number\_products" class="form-inline" action="{% url 'cart\_add' %}">{% csrf\_token %}  
 <div class="form-group">  
 <input type="number" class="form-control" name="number" id="number" min="1" value="1">  
 </div>  
 <div class="form-group">  
 <button type="submit" class="btn btn-success btn-buy" id="submit\_btn"  
 data-product\_id = "{{ tovar.id }}"  
 data-name = "{{ tovar.name }}"  
 data-price = "{{ tovar.price }}"  
 data-size = "{{ tovar.size }}"  
 >  
 В корзину  
 </button>  
 </div>  
 {% if tovar.category.name != "Напитки"%}  
 <div class="form-group" id="radio\_group">  
 <label><input type="radio" name="size" value="35">Большая</label><br>  
 <label><input type="radio" name="size" value="30" checked>Средняя</label><br>  
 <label><input type="radio" name="size" value="25">Маленькая</label>  
 </div>  
 {% endif %}  
 </form>  
 </div>  
 </div>  
 </div>  
 </div>  
 <div class="panel-footer"> Created by M&K, 2018</div>  
 </div>  
 </div>  
</body>  
</html>

**pythonkurs/templates/navbar.html**

{% load staticfiles %}  
<nav class="navbar navbar-default navbar-fixed-top">  
 <div class="container-fluid" id="navbar\_head">  
 <img src="{% static "logo.png" %}" alt="" height="85px">  
 </div>  
 <div class="container-fluid">  
 <!-- Brand and toggle get grouped for better mobile display -->  
 <div class="navbar-header">  
 <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">  
 <span class="sr-only">Toggle navigation</span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 <span class="icon-bar"></span>  
 </button>  
 <img src="{% static "logo\_min.png" %}" alt="" height="40px" id="logo\_min" class="navbar-left">  
 <a class="navbar-brand" href="/{{ views.home }}">Главная страница</a>  
 </div>  
  
 <!-- Collect the nav links, forms, and other content for toggling -->  
 <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">  
 <ul class="nav navbar-nav">  
 <li class="dropdown">  
 <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">Меню <span class="caret"></span></a>  
 <ul class="dropdown-menu">  
 {% for category\_list **in** categories %}  
 <li><a href="/category/{{ category\_list.id }}">{{ category\_list.name }}</a></li>  
 {% endfor %}  
 </ul>  
 </li>  
 <li><a href="/contacts">Контакты</a></li>  
 </ul>  
 <ul class="nav navbar-nav navbar-right">  
 <li><a href="/cart/">Перейти в корзину <span id="cart\_total\_nmb">({{ products\_total\_nmb }})</span></a></li>  
 </ul>  
 </div><!-- /.navbar-collapse -->  
 </div><!-- /.container-fluid -->  
 </nav>

**pythonkurs/templates/scripts.html**

{% load staticfiles %}  
<!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->  
 <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>  
 <!-- Include all compiled plugins (below), or include individual files as needed -->  
 <script src="{% static "Bootstrap3/js/bootstrap.js" %}"></script>  
 <script src="{% static "js/scripts.js" %}"></script>

Список использованной литературы

|  |  |  |
| --- | --- | --- |
| [1] |  | Владимир Дронов, «Django: практика создания Web-сайтов на Python», 2016. |
| [2] |  | Ayman Hourieh, «Django 1.0 Website Development», 2009. |
| [3] |  | «Django Web Development with Python» on YouTube. |
| [4] |  | Jeff Forcier, Paul Bissex and Wesley Chun, «Python Web Development with Django», 2009. |