Отчет по лабораторной работе № 4 по курсу
"Разработка Интернет-Приложений"

Выполнил:
Студент группы
ИУ5-55Б
Хижняков Вадим Максимович

Москва, МГТУ – 2021

Задание:

1. Создайте прототип веб-приложения с использованием фреймворка Django:
    - Создайте виртуальное окружение.
    - Установите в него Django.
    - Создайте проект и приложение Django.
2. Создайте представления и шаблоны (по желанию можно использовать модели), реализующие концепцию master/detail со следующей функциональностью:
    - На странице master в виде списка HTML выводится информация о трех объектах (например, о трех сортах мороженого). Каждая строка списка представляет собой гиперссылку, при нажатии на которую происходит переход к странице detail.
    - Страница detail содержит детальное описание объекта (сорта мороженого), фотографию, гиперссылку на master-страницу.
    - Фотография относится к статическому содержимому сайта.
    - Страница detail должна выводить данные с использованием таблицы HTML.
    - Шаблон страницы detail получает от представления данные о детальном объекте с использованием контекста.
    - НЕОБЯЗАТЕЛЬНЫЙ ПУНКТ. По желанию можно использовать верстку с применением Bootstrap (или аналогичного фреймворка), а также представления на основе классов (class-based views).

Текст программы:

admin.py

```
from django.contrib import admin
from .models import Car, Manufacturer
```

```
admin.site.register(Car)
admin.site.register(Manufacturer)
```

settings.py

```
"""
Django settings for lab4 project.

Generated by 'django-admin startproject' using Django 3.1.4.

For more information on this file, see
https://docs.djangoproject.com/en/3.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.1/ref/settings/
"""

from pathlib import Path
import environ
import os

env = environ.Env(DEBUG=(bool, True))

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.1/howto/deployment/checklist/

# Take environment variables from .env file
environ.Env.read_env(os.path.join(BASE_DIR, '.env'))
SECRET_KEY = env('SECRET_KEY')

DEBUG = env('DEBUG')

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'app.apps.AppConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'corsheaders.middleware.CorsMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

CORS_ALLOW_ALL_ORIGINS = True # If this is used then `CORS_ALLOWED_ORIGINS` will not have any effect
CORS_ALLOW_CREDENTIALS = True
CORS_ALLOWED_ORIGINS = [
    'http://localhost:3000',
] # If this is used, then not need to use `CORS_ALLOW_ALL_ORIGINS = True`
CORS_ALLOWED_ORIGIN_REGEXES = [
    'http://localhost:3000',
]

ROOT_URLCONF = 'lab4.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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',
            ],
        },
    },
]

WSGI_APPLICATION = 'lab4.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.1/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/3.1/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/3.1/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'Europe/Moscow'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.1/howto/static-files/

STATIC_URL = '/static/'

STATICFILES_DIRS = [
  BASE_DIR / "static",
]

# Media path
MEDIA_URL = '/media/'

MEDIA_ROOT = BASE_DIR / 'media'
```

models.py

```
from django.db import models


class Manufacturer(models.Model):
```

```
    name = models.CharField(max_length=200)

    def __str__(self):
        return self.name


class Car(models.Model):
    name = models.CharField(max_length=200, verbose_name="Name")
    price = models.PositiveIntegerField(verbose_name="Price")
    description = models.TextField(verbose_name="Description")
    main_image = models.ImageField(
        upload_to='car_images/',
        blank=True
    )
    manufacturer = models.ForeignKey(
        'Manufacturer',
        on_delete=models.CASCADE,
    )
    score = models.PositiveSmallIntegerField(verbose_name="Score")

    def __str__(self):
        return self.name
```

views.py

```
from django.http.response import Http404, JsonResponse
from django.core import serializers
from django.shortcuts import render
from django.forms.models import model_to_dict
import json

from .models import Car

# Create your views here.


def index(req):
    cars = list(Car.objects.all().values())
    return JsonResponse({'cars': cars})


def details(req, car_id):
    try:
        car = list(Car.objects.filter(pk=car_id).values())[0]
    except Car.DoesNotExist:
        raise Http404('Car Does Not Exist')

    return JsonResponse({'car': car})
```

urls.py

```
"""lab4 URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/3.1/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
from app.views import index, details

from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index, name="index"),
    path('<int:car_id>/', details, name="details")
]

urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Код клиента для лабораторной работы №4 был написан с помощью библиотеки React. В качестве аналогичной работы будет изложен код РК2, написанный с использованием шаблонов django с применением css фреймворка bootstrap5

templates/index.html

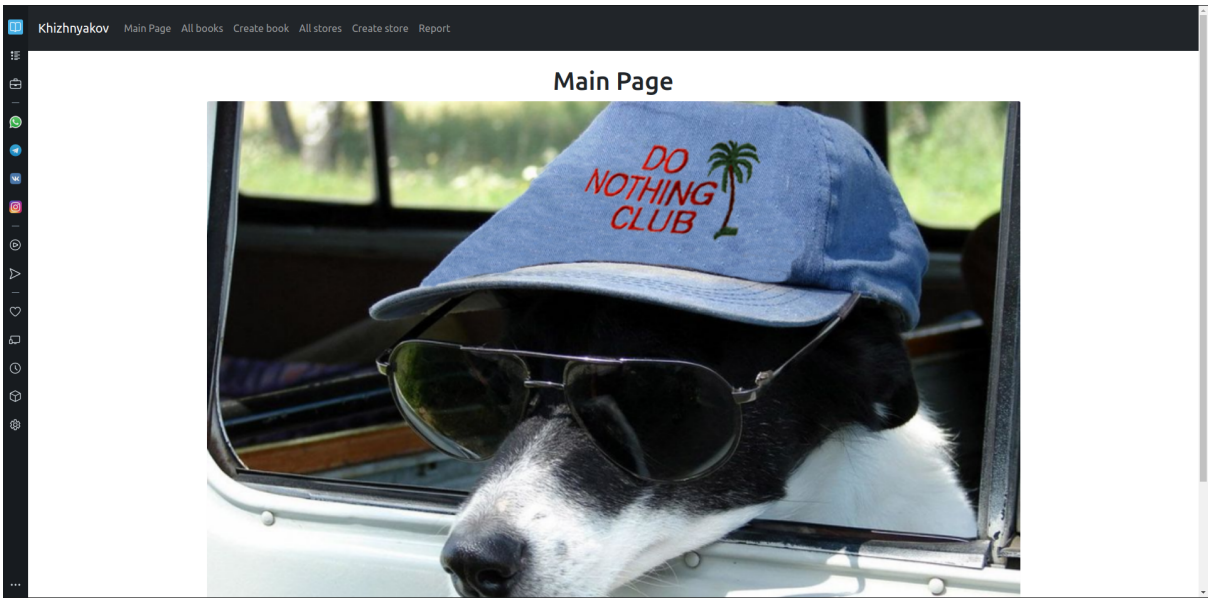
```
{% extends 'inc/base.html' %} {% load static %} {% block content %}
<div class="container">
    <h1 class="text-center mt-4">All Books</h1>
</div>
<!-- Page Content -->
<div class="d-flex flex-column">
    {% for book in books %} {% include 'book/book_card.html' with book=book %} {% endfor %}
</div>
{% endblock content %}
```

templates/details.html

```
{% load static %}

<div class="card mb-3">
  <div class="row g-0">
    <div class="col-md-4">
      <img src="{% static 'img/pes_ava.png' %}" class="img-fluid rounded-start" />
    </div>
    <div class="col-md-8">
      <div class="card-body">
        <h5 class="card-title">{{ book.name }}</h5>
        <p class="card-text">{{ book.store.name }}</p>
        <p class="card-text">
          This is a wider card with supporting text below as a natural lead-in to additional content. This
          content is a little bit longer.
        </p>
        <p class="card-text"><small class="text-muted">{{ book.cost }}</small></p>
        <a href="{% url 'delete_book' book.pk %}" class="btn btn-danger"> Delete </a>
        <a href="{% url 'update_book' book.pk %}" class="btn btn-primary"> Update </a>
      </div>
    </div>
  </div>
</div>
```

Результат работы программы:



Khizhnyakov Main Page All books Create book All stores Create store Report

All Books

latte

bloom-n-brew

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

123

Delete Update

capuchino

brrrew

This is a wider card with supporting text below as a natural lead-in to additional content. This content is a little bit longer.

123123

Delete Update

work pls

brrrew

Khizhnyakov Main Page All books Create book All stores Create store Report

Update Book

Book Name

latte

Book Cost

123

Select Store

brrrew

Save