Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Системы обработки информации и управления»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторной работе №6

«Работа с СУБД»

Выполнил:
студент группы ИУ5-53
Белков А.Д.

Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.

Подпись и дата:

Москва, 2017 г.

Задание и порядок выполнения ЛР №6

В этой лабораторной работе вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также вам нужно будет дополнить свои классы предметной области, связав их с созданной базой. После этого вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей и ClassBasedViews.
Для сдачи вы должны иметь:

1. Скрипт с подключением к БД и несколькими запросами.
2. Набор классов вашей предметной области с привязкой к СУБД (класс должен уметь хотя бы получать нужные записи из БД и преобразовывать их в объекты этого класса)
3. Модели вашей предметной области
4. View для отображения списка ваших сущностей

Код программы

models.py

```python
from django.db import models
from django.contrib.auth.models import User
# -*- coding: utf-8 -*-


class Product(models.Model):
    class Meta:
        db_table = 'lab_app_product'

    # Название товара
    name = models.CharField(max_length=255)

    # Описание товара
    description = models.CharField(max_length=1000)

    # Продавец
    seller = models.CharField(max_length=255)

    # Ссылка на картинку товара
    image = models.ImageField(upload_to='lab_app/static/product_images',

default='lab_app/static/product_images/default.png')

    # Полный путь до картинки товара
    def image_path(self):
        return self.image.name.replace('lab_app/', '')

    # Короткое описание товара
    def short_description(self):
        return self.description[:126]

    def __str__(self):
        return ' '.join([
            self.name,
            ' from ',
            self.seller,
        ])
```

```
class Review(models.Model):
    class Meta:
        db_table = 'lab_app_review'

    # Пользователь, который оставил отзыв
    user = models.ForeignKey(User, on_delete=models.CASCADE)

    # Товар, под которым оставлен отзыв
    product = models.ForeignKey(Product, on_delete=models.CASCADE)

    # Текст отзыва
    description = models.CharField(
        max_length=500,
    )

    def __str__(self):
        return ' '.join([
            'review \'',
            str(self.description),
            ' \' from user @',
            str(self.user.username),
        ])
```

connection.py

```
import MySQLdb


class Connection:

    def __init__(self, user, password, db, host='localhost'):
        self.host = host
        self.user = user
        self.password = password
        self.db = db
        self.use_unicode = True
        self.charset = "utf8"
        self._connection = None

    @property
    def connection(self):
        return self._connection

    def __enter__(self):
        self.connect()

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.disconnect()

    def connect(self):
        if not self._connection:
            self._connection = MySQLdb.connect(
                host=self.host,
                user=self.user,
                password=self.password,
                db=self.db,
                use_unicode=self.use_unicode,
                charset=self.charset
            )

    def disconnect(self):
        if self._connection:
```

```
            self._connection.close()


class Product:
    def __init__(self, db_connection, name, description):
        self.db_connection = db_connection.connection
        self.name = name
        self.description = description

    def save(self):
        c = self.db_connection.cursor()
        c.execute("INSERT INTO 'lab_app_product' ('name', 'description')
VALUES (%s, %s);", (self.name, self.description))

        self.db_connection.commit()
        c.close()


connection = Connection('dbuser', '12345', 'productsDatabase', 'localhost')
with connection:
    product = Product(connection, 'iPhone X', 'Nice computer')
    product.save()
```

views.py

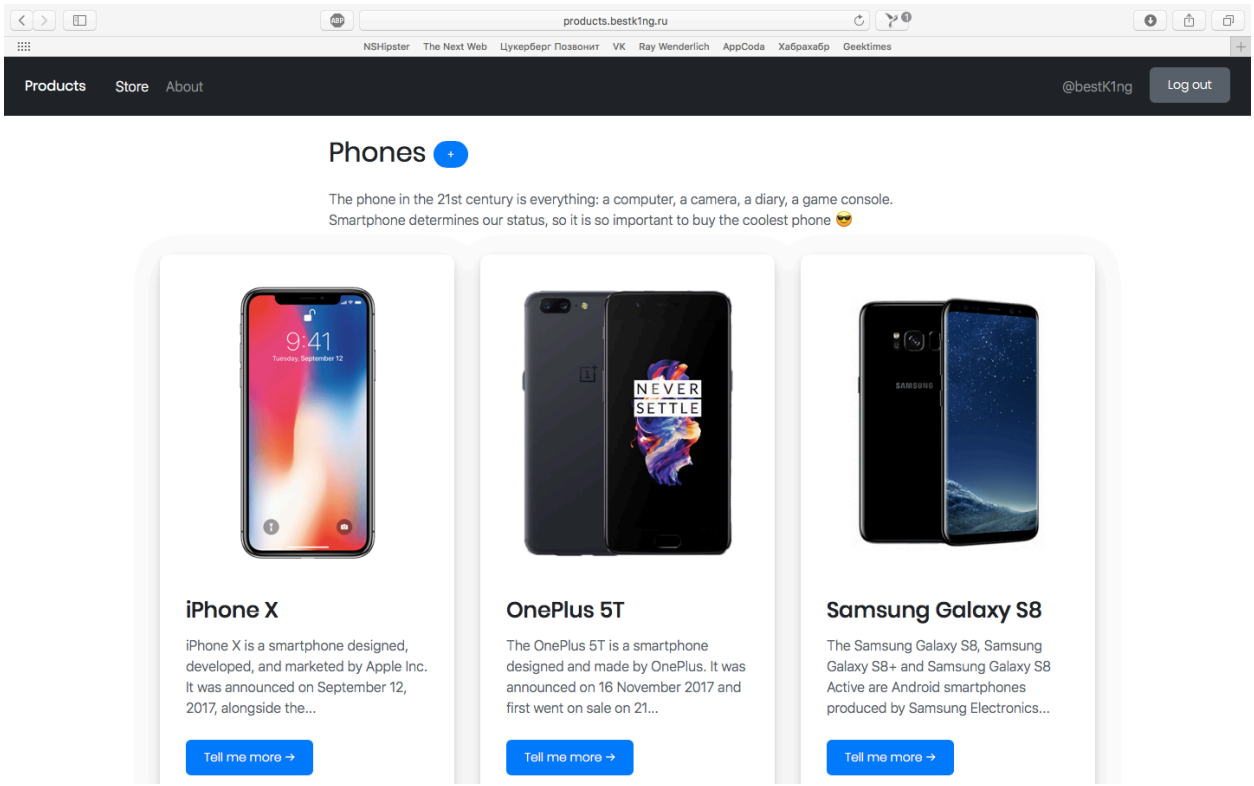
```
from django.contrib.auth import authenticate
from django.contrib.auth.models import User
from django.contrib.auth.views import logout, login
from django.shortcuts import render, redirect

from django.http import HttpResponse
from django.views.generic import View, ListView

from lab_app.models import Product, Review

from django.views.decorators.csrf import csrf_exempt
import json
import math

# TODO: Добавить проверку на superuser для отображения кнопки 'Добавить
продукт'


# Список продуктов
class ListProductView(ListView):

    model = Product
    template_name = 'product_list.html'
    context_object_name = 'products'
    paginate_by = 3

    def get(self, request, page=1):

        # Количество продуктов на странице
        elements_on_page = 9

        # Количество продуктов в строке
        elements_in_row = 3

        products = Product.objects.all()
        pages_count = math.ceil(len(products) / elements_on_page)
```

```
        start_index = (int(page) - 1)*elements_on_page
        end_index = start_index + elements_on_page
        products = products[start_index:end_index]

        index = 1
        rows = []
        row = []
        for product in products:
            row.append(product)

            if index == elements_in_row:
                rows.append(row)
                row = []
                index = 1
            else:
                index += 1

        if len(row) > 0:
            rows.append(row)

        return render(request, 'product_list.html',  {"products": rows,
"page": page, "pages_count": pages_count})


# Страница добавления продукта
class AddProductView(View):

    def post(self, request):
        if request.POST:
            name = request.POST['productName']
            description = request.POST['productDescription']
            seller = request.POST['productSeller']
            image = request.FILES['productImage']

            product = Product(name=name, description=description,
seller=seller, image=image)
            product.save()
            if product is not None:
                return redirect("/")

        return redirect("/invalidProduct")


# Страница с информацией о продукте и отзывами
class ProductView(View):

    def get(self, request, product_id):

        elements_in_row = 2
        product = Product.objects.get(id=product_id)
        reviews = Review.objects.filter(product_id=product_id)
        reviews_count = len(reviews)

        index = 1
        rows = []
        row = []
        for review in reviews:
            row.append(review)

            if index == elements_in_row:
                rows.append(row)
```

```
                row = []
                index = 1
            else:
                index += 1

        if len(row) > 0:
            rows.append(row)

        if len(rows) == 0:
            rows = None

        return render(request, 'product.html',  {"product": product,
"reviews": rows, "reviews_count": reviews_count})


# Страница регистрации
class SignUpView(View):

    def post(self, request):
        logout(request)
        if request.POST:
            user = User.objects.create_user(
                username=request.POST['username'],
                password=request.POST['password'],
                email=request.POST['email']
            )

            if user is not None:
                login(request, user)
                return redirect("/")

        return redirect("/invalidUser")


# Страница авторизации
class LoginView(View):

    def post(self, request):
        logout(request)
        if request.POST:
            username = request.POST['username']
            password = request.POST['password']

            user = authenticate(username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect("/")

        return redirect("/invalidUser")


# Страница выхода
class LogoutView(View):

    def post(self, request):
        logout(request)
        return redirect("/")


# Создание и сохранение отзыва
@csrf_exempt
def create_review(request):
```

```
    if request.method == 'POST':
        review_text = request.POST.get('review_text')
        product_id = request.POST.get('product_id')

        # Создаем отзыв и сохраняем в БД
        user = User.objects.get(id=request.user.id)
        product = Product.objects.get(id=product_id)
        review = Review(description=review_text, user=user, product=product)
        review.save()

        # Формируем json с отзывом для обновления страницы
        response_data = dict()
        response_data["review_description"] = review.description
        response_data["product_id"] = review.product_id
        response_data["user_name"] = review.user.username
        response_data["reviews_count"] =
int(request.POST.get('reviews_count')) + 1

        return HttpResponse(
            json.dumps(response_data),
            content_type="application/json"
        )
    else:
        return HttpResponse(
            json.dumps({"nothing to see": "this isn't happening"}),
            content_type="application/json"
        )


class AboutView(View):

    def get(self, request):

        # ..

        return render(request, 'about.html')
```

Скришоты выполнения


products.bestk1ng.ru
NSHipster
The Next Web
Цукерберг Позвонит
VK
Ray Wenderlich
AppCoda
Хабрахабр
Geektimes
Products
Store
About
@bestK1ng
Log out
Phones
+
The phone in the 21st century is everything: a computer, a camera, a diary, a game console.
Smartphone determines our status, so it is so important to buy the coolest phone 😎
9:41
Tuesday, September 12
NEVER
SETTLE
SAMSUNG
iPhone X
iPhone X is a smartphone designed, developed, and marketed by Apple Inc. It was announced on September 12, 2017, alongside the...
Tell me more →
OnePlus 5T
The OnePlus 5T is a smartphone designed and made by OnePlus. It was announced on 16 November 2017 and first went on sale on 21...
Tell me more →
Samsung Galaxy S8
The Samsung Galaxy S8, Samsung Galaxy S8+ and Samsung Galaxy S8 Active are Android smartphones produced by Samsung Electronics...
Tell me more →