**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Системы обработки информации и управления»

Кафедра ИУ5. Курс «РИП»

Отчет по лабораторной работе №7

## «Авторизация, работа с формами и Django Admin.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-53 |  | преподаватель каф. ИУ5 |
| Белков А.Д. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2017 г.

**Задание и порядок выполнения ЛР №7**

Основная цель данной лабораторной работы – научиться обрабатывать веб-формы на стороне приложения, освоить инструменты, которые предоставляет Django, по работе с формами. Также в этой лабораторной работе вы освоите инструменты Django по работе с авторизацией и реализуете простейшую авторизацию. Напоследок, вы познакомитесь с инструментом администрирования Django – как в несколько строчек кода сделать панель администратора сайта.

1. Создайте view, которая возвращает форму для регистрации.  **Поля формы:**  • Логин • Пароль • Повторный ввод пароля • Email • Фамилия • Имя
2. Создайте view, которая возвращает форму для авторизации.  **Поля формы:**  • Логин • Пароль
3. При отправке формы регистрации во view проверять каждый параметр по правилам валидации, если валидация всех полей пройдена, то создавать пользователя и делать перенаправление на страницу логина, а ошибки, если они есть, выводить над формой.  **Правила валидации:**  *•* Логин не меньше 5 символов *•* Пароль не меньше 8 символов *•* Пароли должны совпадать *•* Все поля должны быть заполнены *•* Логин – уникален для каждого пользователя
4. При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать.
5. Переписать view регистрации с использованием Django Form, правила валидации удалить из view, использовать встроенный механизм валидации полей.
6. Во view авторизации реализовать логин при POST запросе. При успешной авторизации должен происходить переход на страницу успешной авторизации.
7. Страница успешной авторизации должна проверять, что пользователь авторизован. Иначе делать перенаправление на страницу авторизации.
8. Реализовать view для выхода из аккаунта.
9. Заменить проверку на авторизацию на декоратор login\_required
10. Добавить superuser’a через комманду manage.py
11. Подключить django.contrib.admin и войти в панель администрирования.
12. Зарегистрировать все свои модели в django.contrib.admin
13. Для выбранной модели настроить страницу администрирования: *•* Настроить вывод необходимых полей в списке *•* Добавить фильтры *•* Добавить поиск  *•* Добавить дополнительное поле в список.

**Код программы**

**models.py**

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),  
 ])

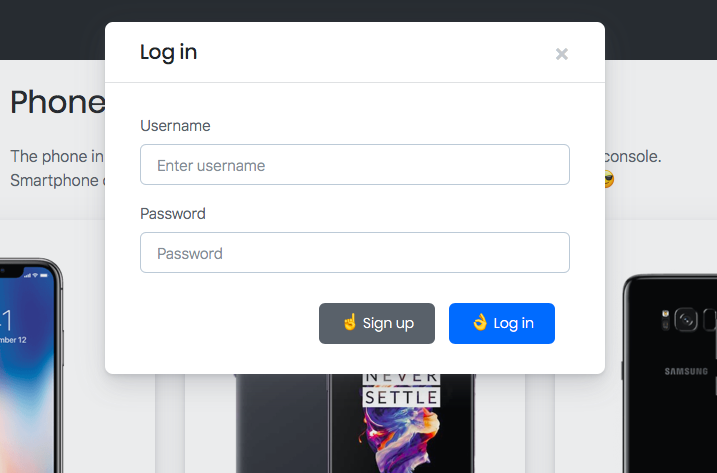
**urls.py**

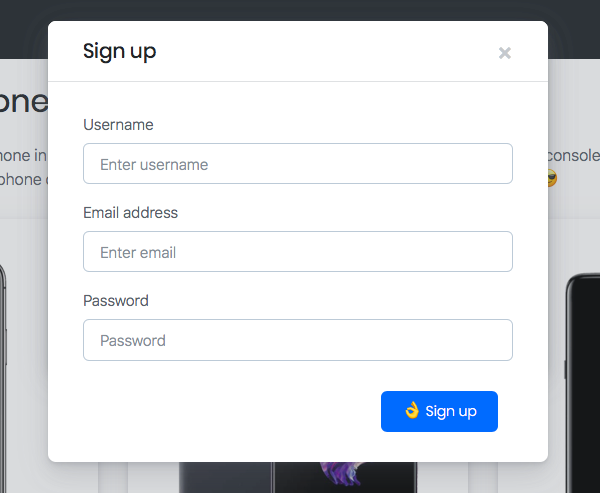
from django.conf.urls import url  
from django.contrib import admin  
  
from lab\_app.views import ProductView, ListProductView, AddProductView, \  
 SignUpView, LoginView, LogoutView, create\_review, AboutView  
  
urlpatterns = [  
 url(r'^$', ListProductView.as\_view()),  
 url(r'^page=(?P<page>\d+)', ListProductView.as\_view()),  
 url(r'^product/(?P<product\_id>\d+)', ProductView.as\_view()),  
 url(r'^product/create\_review/$', create\_review, name='create\_review'),  
 url(r'^product/add\_product/$', AddProductView.as\_view()),  
 url(r'^signup/$', SignUpView.as\_view()),  
 url(r'^login/$', LoginView.as\_view()),  
 url(r'^logout/$', LogoutView.as\_view()),  
 url(r'^about/', AboutView.as\_view()),  
 url(r'^admin/', admin.site.urls),  
]

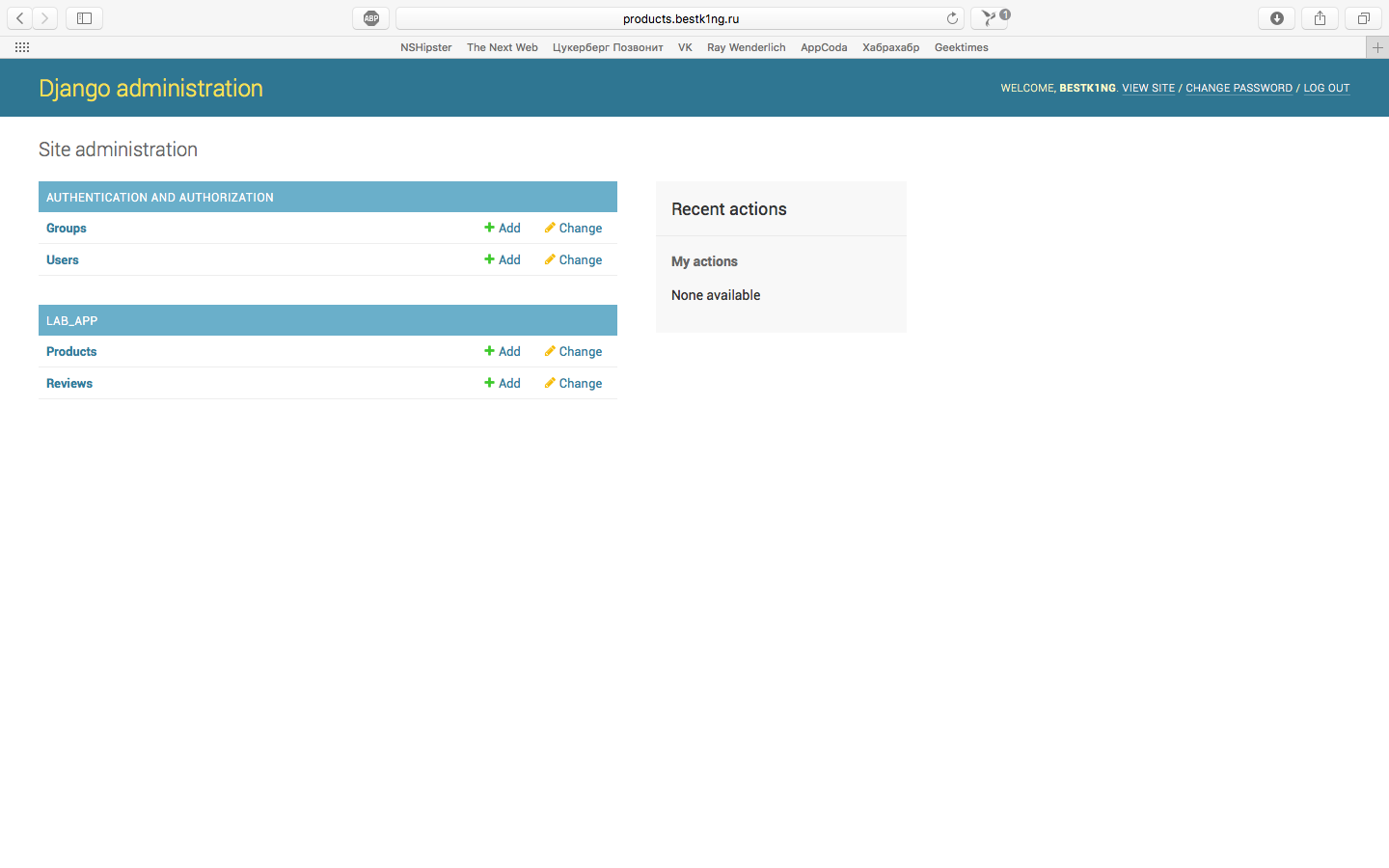
**views.py**

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**

****

****

****