Московский государственный технический

университет им. Н.Э. Баумана.

Факультет «Системы обработки информации и управления»

Кафедра ИУ5

Отчёт

по лабораторной работе №6

по дисциплине «Разработка интернет приложений»

«**Работа с формами, авторизация, Django admin**»

Вариант 3

Студент:

Белкина Е.В.

Группа ИУ5-51

Преподаватель:

Гапанюк Ю.Е.

Москва, 2019

Задание и порядок выполнения ЛР №6

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'а через команду 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

urls.py

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^lab6/', include('lab6.urls')),

]
```

views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from django.views import View
from django.views.generic import ListView
from django.views.generic import TemplateView
from django.contrib.auth.decorators import login_required

#from lab6.models import Bullet
from lab6.forms import *
from lab6.models import *
from django import forms
from django.contrib.auth.hashers import make_password
from django.contrib.auth import authenticate,logout
from django.contrib import auth

# Create your views here.

class ExampleView(View):
    def get(self, request):
        return render(request, 'base.html')

class BulletsView(ListView):
    model = Bullet
    context_object_name = 'bullets'
    template_name = 'bullets.html'

    def get_queryset(self):
        qs = Bullet.objects.all().order_by('id').values()
        return qs

class BulletView(View):
    def get(self, request, id):
        data = Bullet.objects.get(id__exact=id)
        return render(request, 'bullet.html', {'bullet':data})

def registration_old(request):
    errors = []
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors.append('Login required')
        elif len(username)<5:
            errors.append('Login should be 5 or more symbols')

        password = request.POST.get('password')
        if not password:
            errors.append('Password required')
        elif len(password)<6:
            errors.append('Password length should be 6 or more')

        password_repeat = request.POST.get('password2')

        if password != password_repeat:
```

```
            errors.append('Password should be similar')

        if not errors:

            return HttpResponseRedirect('/lab6/login')
    return render(request, 'lab6/logon.html', {'errors': errors})

def registration(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('/lab6/')
        return render(request, 'signup.html', {'form': form})
    else:
        form = RegistrationForm()
    return render(request, 'signup.html', {'form': form})

def authorization(request):
    redirect_url = '/lab6/'
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            user = auth.authenticate(username=form.cleaned_data['login'],
                                     password=form.cleaned_data['password'])
            if user is not None:
                auth.login(request, user)
                return HttpResponseRedirect(redirect_url)
            else:
                form.add_error(None, 'Wrong login or password')
    else:
        form = LoginForm()
    return render(request, 'login.html', {'form':form, 'continue': redirect_url})

@login_required
def exit(request):
    logout(request)
    return render(request, 'logout.html')
```

lab6.urls.py

```
from django.conf.urls import url

from . import views

urlpatterns = [
    #url(r'^function_view/', views.function_view),
    #url(r'^class_based_view/', views.ExampleClassBase.as_view()),
    url(r'^$', views.ExampleView.as_view(), name='start'),
    url(r'^bullets/', views.BulletsView.as_view()),
    url(r'^bullet/(?P<id>\d+)', views.BulletView.as_view(), name='bullet_url'),
    url(r'^signupold/', views.registration_old, name='signupNOFORM'),
    url(r'^signup/', views.registration, name='signup'),
    url(r'^login/', views.authorization, name='login'),
    url(r'^logout/', views.exit, name='logout'),
]
```

models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser

# Create your models here.

class Bullet(models.Model):
```

```
    name = models.CharField(max_length=30)
    description = models.CharField(max_length=255)
    datetime = models.DateTimeField(auto_now=True)

class MyUser(AbstractUser):
    regdate = models.DateTimeField(auto_now_add=True)

    class Meta(AbstractUser.Meta):
        swappable = 'AUTH_USER_MODEL'
```

forms.py

```
from django import forms
from django.core.exceptions import ValidationError
from django.core.validators import validate_email
from django.contrib.auth.models import User


class LoginForm(forms.Form):
    login = forms.CharField(label='Логин')
    password = forms.CharField(label='Пароль', widget=forms.PasswordInput)


class RegistrationForm(forms.Form):
    login = forms.CharField(label='Логин', min_length=5)
    password = forms.CharField(label='Пароль', min_length=8, widget=forms.PasswordInput)
    repeat_password = forms.CharField(label='Повторите пароль', widget=forms.PasswordInput)
    email = forms.CharField(label='Адрес электронной Почты')
    first_name = forms.CharField(label='Имя')
    last_name = forms.CharField(label='Фамилия')

    def clean_login(self):
        login = self.cleaned_data['login']
        if User.objects.filter(username=login):
            raise ValidationError('Этот login уже занят')
        return login

    def clean_email(self):
        email = self.cleaned_data['email']
        validate_email( self.cleaned_data['email'])
        if User.objects.filter(email=email):
            raise ValidationError('Этот email уже зарегистрирован')
        return self.cleaned_data['email']

    def clean(self):
        cleaned_data = super(RegistrationForm, self).clean()
        if self.cleaned_data.get('password') and self.cleaned_data.get('repeat_password'):
            if self.cleaned_data['password'] != self.cleaned_data['repeat_password']:
                raise ValidationError('Пароли не совпадают')
        return cleaned_data

    def save(self):
        user = User.objects.create_user(username=self.cleaned_data['login'],
                                        email=self.cleaned_data['email'],
                                        password=self.cleaned_data['password'],
                                        first_name=self.cleaned_data['first_name'],
                                        last_name=self.cleaned_data['last_name'],
                                        )
        return user
```

admin.py

```
from django.contrib import admin

# Register your models here.

from lab6.models import Bullet

class BulletAdmin(admin.ModelAdmin):
    list_display = ('name','description','datetime','desc_len')
    list_filter = ['datetime']
    search_fields = ('id','name')

    def desc_len(self, obj):
        return len(obj.description)
    desc_len.short_description = "Длина сообщения"

admin.site.register(Bullet, BulletAdmin)
```

base.html

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="{% static 'css/bootstrap.css' %}">
    <title>{% block title %}Тестовый вывод{% endblock %}</title>
</head>
<body>
<nav class="navbar navbar-default">
  <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="#navbar"
aria-expanded="false" aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand">IAD #6</a>
    </div>-->

    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="navbar">
      <ul class="nav navbar-nav">
        <li><a href="/lab6/">Базовый класс</a></li>
        <li><a href="/lab6/bullets/">Объявления</a></li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        {% if user.is_authenticated %}
            <li><div style="padding-top:15px"><i><b>Здраствуйте, {{ user.get_short_name
}}</b></i></div></li>
            <li><a href="{% url 'logout' %}">Выйти из {{ user.get_username }}</a></li>
        {% else %}
            <li><div style="padding-top:15px"><i><b>Вы вошли как Гость</b></i></div></li>
            <li><a href="{% url 'signup' %}">Зарегистрироваться</a></li>
            <li><a href="{% url 'login' %}">Войти</a></li>
        {% endif %}
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
    <div>
        {% block body %}Redefined by spawnlings{% endblock %}
    </div>
    <!-- Bootstrap core JavaScript
    ================================================ -->
```

```
    <!-- Placed at the end of the document so the pages load faster -->
    <script src="{% static 'js/jquery-3.1.1.min.js' %}"></script>
    <script src="{% static 'js/bootstrap.min.js' %}"></script>
</body>
</html>
```

login.html

```
{% extends 'base.html' %}

{% block title %}Вход{% endblock %}

{% block body %}

{{ form.non_field_errors }}
<form action="/lab6/login/" method="post" class="form-horizontal">
                    {% csrf_token %}
                    {%  for i in form %}
                    <div class="form-group">
                        <label class="col-sm-2">{{ i.label }}</label>
                        <div class="col-sm-10">
                            <div>
                                {{ i }}
                            </div>

                        </div>
                    </div>
                    {% endfor %}

                    <div class="form-group">
                        <div class="col-sm-offset-2 col-sm-10">
                            <input type="submit" value="Войти" class="btn btn-default"/>
                        </div>
                    </div>
</form>
{% endblock %}
```

logout.html

```
{% extends 'base.html' %}

{% block title %}Выход{% endblock %}

{% block body %}
<p>Выход произошёл успешно.</p>
{% endblock %}
```

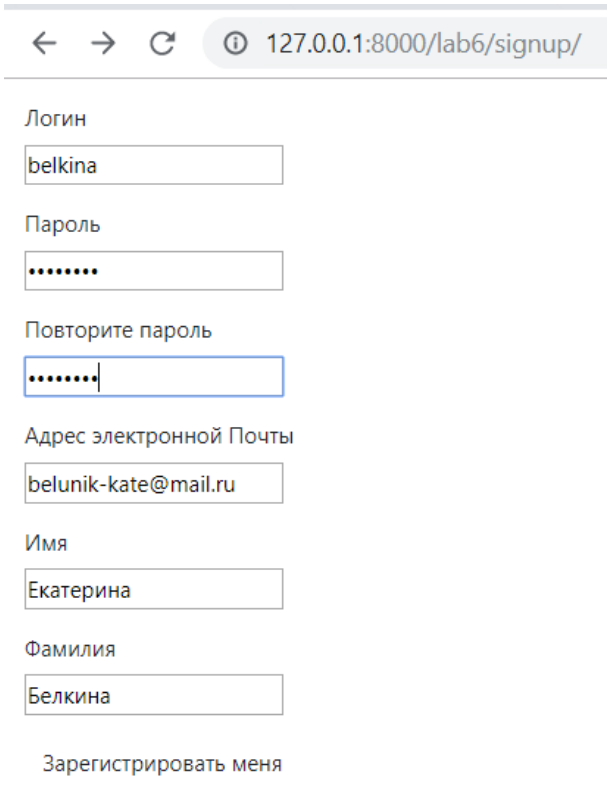
signup.html

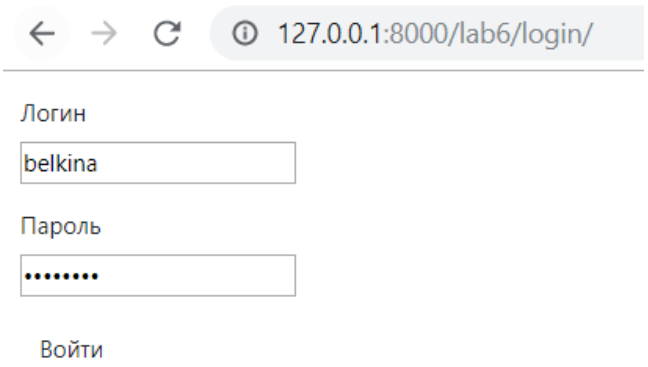
```
{% extends 'base.html' %}

{% block title %}Регистрация{% endblock %}

{% block body %}

{{ form.non_field_errors }}
<form action="/lab6/signup/" method="post" class="form-horizontal" enctype="multipart/form-data">
    {% csrf_token %}
    {%  for i in form %}
    <div class="form-group">
        <label class="col-sm-4">{{ i.label }}</label>
        <div class="col-sm-8">
            <div>
                {{ i }}
            </div>
            <div>
```

```
                {{ i.errors }}
            </div>
        </div>
    </div>
    {% endfor %}

    <div class="form-group">
        <div class="col-sm-offset-4 col-sm-8">
            <input type="submit" value="Зарегистрировать меня" class="btn btn-default"/>
        </div>
    </div>
</form>
{% endblock %}
```

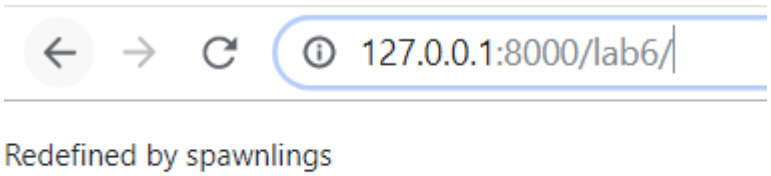
Скриншоты выполнения

Регистрация и Вход



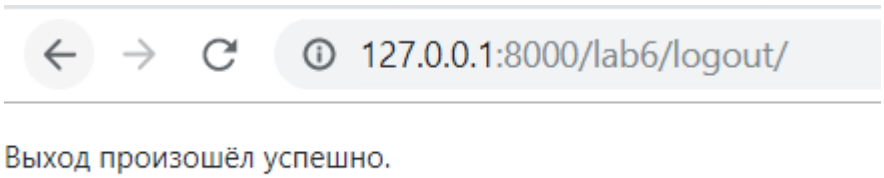


Успешный вход



Redefined by spawnlings

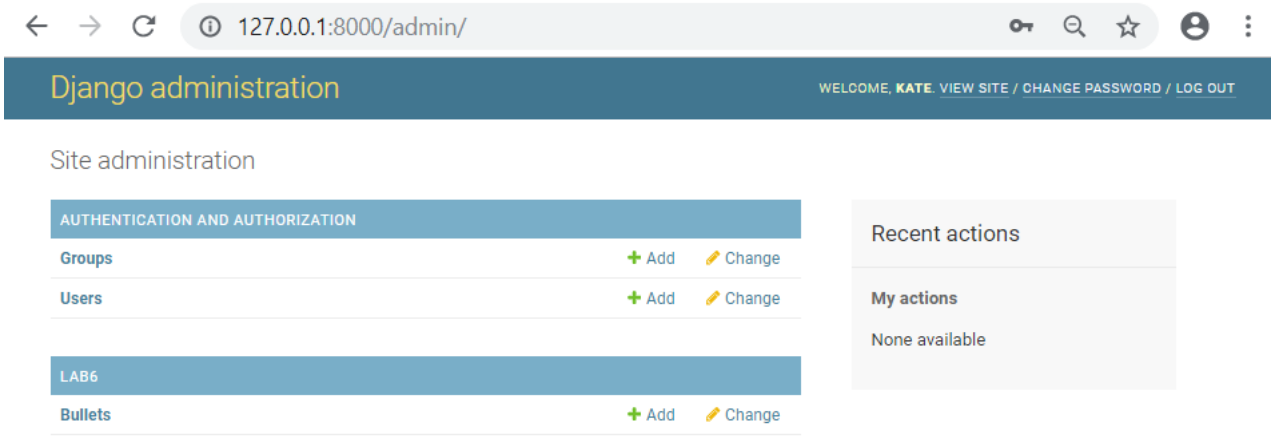
Успешный выход



Создание аккаунта Admin (superuser):

```
C:\Users\belun\Documents\РИП\Lab_6_3>python manage.py createsuperuser
Username (leave blank to use 'belun'): kate
Email address: ex@mail.ru
Password:
Password (again):
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

Django Admin



Отображение пользователей **в** Django Admin