

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

**Отчет по лабораторной работе № 7 по курсу  
разработка интернет-приложений**

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-71Ц \_\_\_\_\_  
(подпись)

Баглай П.С

"\_\_" \_\_\_\_\_ 2017 г.

ПРОВЕРИЛ:

Гапанюк Ю.Е \_\_\_\_\_  
(подпись)

"\_\_" \_\_\_\_\_ 2017 г.

**Москва – 2017 г.**

## Порядок работы

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 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

## Листинг программы

### Model.py

```
from django.db import models
from django import forms
from django.contrib.auth.models import User
# Create your models here.

class RegisterForm(forms.Form):
    login = forms.CharField(label='ЛОГИН', min_length=5)
    password = forms.CharField(label='ПАРОЛЬ', min_length=8,
    widget=forms.PasswordInput)
    password2 = forms.CharField(label='ПАРОЛЬ ПОВТОРИТЬ', min_length=8,
    widget=forms.PasswordInput)
    email = forms.CharField(label='ПОЧТА', min_length=1)
    surname = forms.CharField(label='ФАМИЛИЯ', min_length=1)
    name = forms.CharField(label='ИМЯ', min_length=1)

    def clean(self):
        cleaned_data = super(RegisterForm, self).clean()
        password = cleaned_data.get('password')
        password2 = cleaned_data.get('password2')
        if password != password2:
            raise forms.ValidationError("Пароли не совпадают")
        usrs = User.objects.filter(username=cleaned_data.get('login'))
        if len(usrs) > 0:
            raise forms.ValidationError("Пользователь с данным логином уже
существует")

class LoginForm(forms.Form):
    login = forms.CharField(label='Login', min_length=5)
    password = forms.CharField(label='Password', min_length=8,
    widget=forms.PasswordInput)

class Book_User(models.Model):
    user=models.CharField(max_length=200)
    book=models.CharField(max_length=100)
    number=models.IntegerField()
```

### views.py

```
from django.shortcuts import render, redirect, render_to_response
from django.http import HttpResponseRedirect
from django.views import View
from django.utils.safestring import mark_safe
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
# Create your views here.
from lab.models import RegisterForm
class Register(View):
    def get(self, request):
        form = RegisterForm()
        return render(request, 'lab/register2.html', {'errors': '', 'form':
form.as_p()})
```

```

def post(self, request):
    form = RegisterForm(request.POST)

    if not form.is_valid():
        return render(request, 'lab/register2.html', {'errors': '',
'form': form.as_p()})

    u = User(username=form.cleaned_data['login'],
email=form.cleaned_data['email'],
            last_name=form.cleaned_data['surname'],
first_name=form.cleaned_data['name'])
    u.set_password(form.cleaned_data['password'])
    u.save()
    return redirect('/succ/')

#def get(self, request):
#    return
render(request, 'lab/register.html', {'errors': '', 'login': '', 'email': '', 'surname': '', 'name': ''})

#def post(self, request):
#    login = request.POST['login']
#    password = request.POST['password']
#    password2 = request.POST['password2']
#    email = request.POST['email']
#    surname = request.POST['surname']
#    name = request.POST['name']
#    errors = []
#    if len(login) < 5:
#        errors.append("Логин короткий")
#    if len(password) < 8:
#        errors.append("Пароль короткий")
#    if password != password2:
#        errors.append("Пароли не совпадают")
#    if len(email) < 1 or len(surname) < 1 or len(name) < 1:
#        errors.append("Все поля должны быть заполнены")
#    if len(errors) == 0:
#        usrs = User.objects.filter(username=login)
#        if len(usrs) > 0:
#            errors.append("Пользователь с данным логином уже
существует")
#        else:
#            u = User(username=login, email=email, last_name=surname,
first_name=name)
#            u.set_password(password)
#            u.save()
#    if len(errors) > 0:
#        return render(request, 'lab/register.html', {'errors':
mark_safe('<br>'.join(errors)), 'login': login,
#            'email': email,
'surname': surname, 'name': name})
#    return redirect('/login/')

class Login(View):
    def get(self, request):
        return render(request, 'lab/login.html', {'errors': '', 'login': ''})

    def post(self, request):
        log = request.POST['login']
        password = request.POST['password']
        errors = []

        user = authenticate(username=log, password=password)

        if user is not None:
            login(request, user)
            return redirect('/')
        errors.append('Логин или пароль неверны')

```

```
        return render(request, 'lab/login.html', {'errors':
mark_safe('<br>'.join(errors)), 'login': login})

@login_required(login_url='/login')
def home(request):
    a = 'You are authenticated'
    return render(request, 'lab/home.html', {'auth': a})

class Logout(View):
    success_url = "/"
    def get(self, request):
        logout(request)
        return HttpResponseRedirect("/")
```

## Результаты работы программы

### Войти

Логин

Пароль

[Регистрация](#)

---

ЛОГИН:

ПАРОЛЬ:

ПАРОЛЬ ПОВТОРИТЬ:

ПОЧТА:

ФАМИЛИЯ:

ИМЯ:

# Войти

Логин

pasbaglai

Пароль

●●●●●●●●

Войти

[Регистрация](#)

You are authenticated

Привет, pasbaglai [Выход](#)