

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



Лабораторные работы по курсу:

**«Разработка Интернет Приложений»**

**ЛР7. Авторизация, работа с формами и Django Admin.**

Исполнитель:

Студент группы РТ5-51

Макаров А.В.

Преподаватель:

Гапанюк Ю. Е.

« \_\_\_\_ » \_\_\_\_\_



## Задание и порядок выполнения

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.

Исходный код:

settings.py:

```
"""
Django settings for test5_full project.

Generated by 'django-admin startproject' using Django 1.11.7.

For more information on this file, see
https://docs.djangoproject.com/en/1.11/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/1.11/ref/settings/
"""

import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '+glayd^@rojiipq)-qy33nk8i+2un!kjloug(r9178(f+uw6!4'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['192.168.1.10', '127.0.0.1']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp.apps.MyappConfig',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
```

```

'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'test5_full.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
        ,
        '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 = 'test5_full.wsgi.application'

# Database
# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'userbronkoncert',
        'USER': 'root',
        'PASSWORD': '1111',
        'PORT': 3306, #Стандартный порт Mysql
        'OPTIONS': {
            'autocommit': True,
        },
        'TEST_CHARSET': 'utf8',
    }
}

# Password validation
# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidato
r',
    },
    {
        '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/1.11/topics/i18n/

```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/1.11/howto/static-files/

```

```
STATIC_URL = '/static/'
```

## urls.py

```

from django.conf.urls import url
from django.contrib import admin
from myapp.views import *

```

```

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^register/', RegisterFormView.as_view()),
    url(r'^login/', LoginFormView.as_view()),
    url(r'^users/$', UserList.as_view(), name='my-view2'),
    url(r'^usersList/$', UserList.as_view()),
]

```

## views.py

```
from django.http import HttpResponseRedirect
from django.views.generic.edit import FormView
from django.views.generic import ListView
from django.contrib.auth.forms import UserCreationForm
# Опять же, спасибо django за готовую форму аутентификации.
from django.contrib.auth.forms import AuthenticationForm

# Функция для установки сессионного ключа.
# По нему django будет определять, выполнил ли вход пользователь.
from django.contrib.auth import login
from django.http import HttpResponseRedirect
from django.views.generic.base import View
from django.contrib.auth import logout

from myapp.Connection import *

class UserList(ListView):
    model = User

#####
class UserList(View):
    def get(self, request, *args, **kwargs):
        con = Connection("root", "1111", "userbronkoncert")
        with con:
            user = SelAll(con)
            f=user.save()
        return HttpResponseRedirect(f)

#####

class LogoutView(View):
    def get(self, request):
        # Выполняем выход для пользователя, запросившего данное
        # представление.
        logout(request)

        # После чего, перенаправляем пользователя на главную страницу.
        return HttpResponseRedirect("/")

class LoginFormView(FormView):
    form_class = AuthenticationForm

    # Аналогично регистрации, только используем шаблон аутентификации.
    template_name = "login.html"

    # В случае успеха перенаправим на главную.
    success_url = "/users/"

    def form_valid(self, form):
        # Получаем объект пользователя на основе введенных в форму данных.
        self.user = form.get_user()

        # Выполняем аутентификацию пользователя.
```

```

        login(self.request, self.user)
        return super(LoginFormView, self).form_valid(form)

class RegisterFormView(FormView):
    form_class = UserCreationForm

    # Ссылка, на которую будет перенаправляться пользователь в случае
    # успешной регистрации.
    # В данном случае указана ссылка на страницу входа для
    # зарегистрированных пользователей.
    success_url = "/login/"

    # Шаблон, который будет использоваться при отображении представления.
    template_name = "register.html"

    def form_valid(self, form):
        # Создаём пользователя, если данные в форму были введены корректно.
        form.save()

        # Вызываем метод базового класса
        return super(RegisterFormView, self).form_valid(form)

```

## login.html

```

<!DOCTYPE html>
<html>
<head>
<title>Авторизация</title>
</head>
<body>
<form action="" method="post">
    {% csrf_token %}
    <!-- as_p для того, чтобы каждый элемент формы был с новой строки -->
    {{ form.as_p }}
    <button type="submit">Проверка подлинности человека</button>
</form>
</body>
</html>

```

## register.html

```

<!DOCTYPE html>
<html>
<head>
<title>Регистрация</title>
</head>
<body>
<form action="" method="post">
    {% csrf_token %}
    <!-- as_p для того, чтобы каждый элемент формы был с новой строки -->
    {{ form.as_p }}
    <button type="submit">Зарегистрироваться</button>
</form>
</body>
</html>

```