Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Автоматизированные системы обработки информации и управления»



**Отчет**

**Лабораторная работа № 6**

**По курсу «Разработка интернет приложений»**

**«Работа с формами, авторизация и модуль администрирования в Django»**

**ИСПОЛНИТЕЛЬ:**

Зубков Андрей

Группа ИУ5-54

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2019 г.

**ПРЕПОДАВАТЕЛЬ:**

Гапанюк Ю.Е.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"\_\_"\_\_\_\_\_\_\_\_\_\_\_2019 г.

Москва 2019

1. **Задание лабораторной 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.
2. Создайте view, которая возвращает форму для регистрации.  
   Поля формы:

* Логин
* Пароль
* Повторный ввод пароля
* Email
* Фамилия
* Имя

1. Создайте view, которая возвращает форму для авторизации.  
   Поля формы:

* Логин
* Пароль

1. 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

1. При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать.
2. Переписать view регистрации с использованием Django Form, правила валидации удалить из view, использовать встроенный механизм валидации полей.
3. Во view авторизации реализовать логин при POST запросе. При успешной авторизации должен происходить переход на страницу успешной авторизации.
4. Страница успешной авторизации должна проверять, что пользователь авторизован. Иначе делать перенаправление на страницу авторизации.
5. Реализовать view для выхода из аккаунта.
6. Заменить проверку на авторизацию на декоратор login\_required
7. Добавить superuser’a через комманду manage.py
8. Подключить django.contrib.admin и войти в панель администрирования.
9. Зарегистрировать все свои модели в django.contrib.admin
10. Для выбранной модели настроить страницу администрирования:

* Настроить вывод необходимых полей в списке
* Добавить фильтры
* Добавить поиск
* Добавить дополнительное поле в список

1. **Форма регистрации  
   class** RegistrationForm(forms.ModelForm):  
    first\_name = forms.CharField(widget=forms.TextInput(attrs={**'class'**: **'form-control col-sm-6'**}))  
    last\_name = forms.CharField(widget=forms.TextInput(attrs={**'class'**: **'form-control col-sm-6'**}))  
    email = forms.CharField(widget=forms.EmailInput(attrs={**'class'**: **'form-control col-sm-6'**}),  
    validators=[validate\_email])  
    username = forms.CharField(widget=forms.TextInput(attrs={**'class'**: **'form-control col-sm-6'**}))  
    password = forms.CharField(widget=forms.PasswordInput(attrs={**'class'**: **'form-control col-sm-6'**}))  
    confirm\_password = forms.CharField(widget=forms.PasswordInput(attrs={**'class'**: **'form-control col-sm-6'**}))  
     
    **class** Meta:  
    model = Author  
    fields = [**'avatar'**]  
     
    field\_order = [  
    **'first\_name'**,  
    **'last\_name'**,  
    **'email'**,  
    **'username'**,  
    **'password'**,  
    **'confirm\_password'**,  
    **'avatar'**  
   ]  
     
    **def** is\_valid(self):  
    ret = forms.Form.is\_valid(self)  
    **for** f **in** self.errors:  
    **if** f != **"\_\_all\_\_"**:  
    self.fields[f].widget.attrs.update(  
    {**'class'**: self.fields[f].widget.attrs.get(**'class'**, **''**) + **' is-invalid'**})  
    **return** ret  
     
    **def** clean(self):  
    confirm\_password = self.cleaned\_data.get(**'confirm\_password'**)  
    password = self.cleaned\_data.get(**'password'**)  
     
    **if** password != confirm\_password:  
    **raise** forms.ValidationError(**'Passwords do not match'**)  
     
    **def** save(self, commit=**True**):  
    u = User.objects.create(username=self.cleaned\_data.get(**'username'**),  
    email=self.cleaned\_data.get(**'email'**),  
    first\_name=self.cleaned\_data.get(**'first\_name'**),  
    last\_name=self.cleaned\_data.get(**'last\_name'**))  
    u.set\_password(self.cleaned\_data.get(**'password'**))  
    u.save()  
     
    a = super().save(commit=**False**)  
    a.user = u  
     
    **if** commit:  
    a.save()  
     
    **return** u
2. **View для регистрации  
   class** SignUpView(View):  
    **def** get(self, request, \*\*kwargs):  
    form = kwargs.get(**'form'**, RegistrationForm())  
    context = {  
    **'form'**: form  
    }  
    **return** render(request, **'signup.html'**, context)  
     
    **def** post(self, request):  
    form = RegistrationForm(data=request.POST, files=request.FILES)  
     
    **if** form.is\_valid():  
    user = form.save()  
    login(request, user)  
    **return** HttpResponseRedirect(reverse(**'ask:index'**))  
    **else**:  
    **return** self.get(request, form=form)
3. **Login form  
   class** LoginForm(forms.ModelForm):  
    **class** Meta:  
    model = User  
    fields = [**'username'**, **'password'**]  
     
    widgets = {  
    **'username'**: forms.TextInput(attrs={**'class'**: **'form-control col-sm-6'**}),  
    **'password'**: forms.PasswordInput(attrs={**'class'**: **'form-control col-sm-6'**})  
    }  
     
    **def** clean(self):  
    username = self.cleaned\_data.get(**'username'**)  
    password = self.cleaned\_data.get(**'password'**)  
    user = authenticate(username=username, password=password)  
    **if not** user **or not** user.is\_active:  
    **raise** forms.ValidationError(**"Login is invalid. Please, try again"**)  
    **return** self.cleaned\_data
4. **Login View  
   class** LoginView(View):  
    **def** get(self, request):  
    next\_url = request.GET.get(**'next'**, reverse(**'ask:index'**))  
    form = LoginForm()  
    context = {  
    **'form'**: form,  
    **'next\_url'**: next\_url  
    }  
    **return** render(request, **'login.html'**, context)  
     
    **def** post(self, request):  
    next\_url = request.GET.get(**'next'**, reverse(**'ask:index'**))  
    form = LoginForm(data=request.POST)  
     
    **if** form.is\_valid():  
    username = form.cleaned\_data.get(**'username'**)  
    password = form.cleaned\_data.get(**'password'**)  
    user = authenticate(username=username, password=password)  
     
    **if** user **is not None**:  
    login(request, user)  
    **return** HttpResponseRedirect(next\_url)  
    **else**:  
    context = {  
    **'next\_url'**: next\_url,  
    **'form'**: form  
    }  
    **return** render(request, **'login.html'**, context)
5. **Logout view  
   def** logout(request):  
    django\_logout(request)  
    url = request.META.get(**'HTTP\_REFERER'**)  
    **if** url **is None**:  
    url = reverse(**'ask:index'**)  
    **return** HttpResponseRedirect(url)
6. **Регистрация моделей в django.contrib.admin**admin.site.register(Author)  
   admin.site.register(Question)  
   admin.site.register(Answer)  
   admin.site.register(Tag)  
   admin.site.register(QuestionVote)  
   admin.site.register(AnswerVote)

