Московский государственный технический университет им. Н.Э.Баумана

Кафедра «Системы обработки информации и управления»

Утверждаю: _____ Согласовано: _____

"___"_____2016 г. "___"_____2016 г.

Отчет по лабораторной работе №7

(вид документа)

писчая бумага формата А4

(вид носителя)

_____

(количество листов)

Исполнитель: студент группы РТ5-51

_____ Коньшин К.И.

"_____"_____ 2016 г.

Москва – 2016

Исходники:

1. Создайте view, которая возвращает форму для регистрации.

```
class reg1(View):
    def get(self, request):
        return render(request, 'register.html')
```

```
{% for error in errors %}
    {{ error }}
    </br>
{% endfor %}

<form action="/reg1" method="POST">
   <label for="last_name">Фамилия</label>
   <input type="text" name="last_name" id="last_name" value="{{ last_name }}"><br>

   <label for="first_name">Имя</label>
   <input type="text" name="first_name" id="first_name" value="{{ first_name }}"><br>

   <label for="login">Логин</label>
   <input type="text" name="login" id="login" value="{{ login }}"><br>

   <label for="password">Пароль</label>
   <input type="password" name="password" id="password"><br>

   <label for="password2">Пароль еще раз</label>
   <input type="password" name="password2" id="password2"><br>

   <label for="email">Email</label>
   <input type="text" name="email" id="email" value="{{ email }}"><br>

    {% csrf_token %}

   <button type="submit">Зарегистрироваться</button>
</form>
```

2. Создайте view, которая возвращает форму для авторизации

```
class Login(View):
    def get(self, request):
        return render(request, 'login.html')
```

```
{% for error in errors %}
    {{ error }}
    </br>
{% endfor %}

<form action="/login" method="POST">
   <label for="login">Логин</label>
   <input type="text" name="login" id="login" value="{{ login }}"><br>

   <label for="password">Пароль</label>
   <input type="password" name="password" id="password"><br>

    {% csrf_token %}

   <button type="submit">Войти</button>
</form>
```

3+4 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

При возникновении ошибок в момент отправки формы, введенные значения в полях ввода, кроме пароля, не должны исчезать

```python
class reg1(View):
    def get(self, request):
        return render(request, 'register.html')

    def post(self, request):
        errors = []
        login = request.POST.get('login', '')
        password = request.POST.get('password', '')
        password2 = request.POST.get('password2', '')
        email = request.POST.get('email', '')
        last_name = request.POST.get('last_name', '')
        first_name = request.POST.get('first_name', '')

        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 not len(email) or not len(last_name) or not len(first_name):
            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=last_name, first_name=first_name)
                u.set_password(password)
                u.save()

        if len(errors) > 0:
            return render(request, 'register.html', {'errors': errors, 'login': login,
                                                     'email': email, 'last_name': last_name, 'first_name': first_name})
        return redirect('/login', {'errors': 'Вы зарегестрировались, авторизируйтесь.'})
```

5. Переписать view регистрации с использованием Django Form, правила валидации удалить из view, использовать встроенный механизм валидации полей.

```python
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 2', min_length=8, widget=forms.PasswordInput)
    email = forms.CharField(label='E-mail', min_length=1)
    last_name = forms.CharField(label='Фамилия', min_length=1)
    first_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("Логин занят")
```

```
{% for error in errors %}
    {{ error }}
    </br>
{% endfor %}


<form action="/reg2" method="POST">
   {{ form }}

    {% csrf_token %}

   <button type="submit">Зарегистрироваться</button>
</form>
```

6. Во view авторизации реализовать логин при POST запросе. При успешной авторизации должен происходить переход на страницу успешной авторизации

```
class Login(View):
    def get(self, request):
        return render(request, 'login.html')

    def post(self, request):
        log = request.POST.get('login', '')
        password = request.POST.get('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, 'login.html', {'errors': errors, 'login': login})
```

7. Страница успешной авторизации должна проверять, что пользователь авторизован. Иначе делать перенаправление на страницу авторизации.

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


def index2(request):
    a = 'You are authenticated'
    if request.user.is_authenticated():
        return render(request, 'index.html', {'auth': a})
    else:
        return redirect('/login')
```

8. Реализовать view для выхода из аккаунта

```
def log(request):
    logout(request)
    return redirect('/')
```

9. Заменить проверку на авторизацию на декоратор login_required

см. Пункт 7

10. Добавить superuser'а через комманду manage.py
Суперпользователь был добавлен

11. Подключить django.contrib.admin и войти в панель администрирования

Было подключено, затем мы авторизовались

12.Зарегистрировать все свои модели в django.contrib.admin

сделал

13.Для выбранной модели настроить страницу администрирования:

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

```
class Events(models.Model):
    class Meta:
        verbose_name = 'Мероприятие'
        verbose_name_plural = 'Мероприятия'
    name = models.CharField(max_length=15)
    city = models.IntegerField()
```

```
class Event(admin.ModelAdmin):
    list_display = ('name', 'city')
    list_filter = ('name',)
    search_fields = ('name',)
```