

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

**Разработка интернет-приложений
Лабораторная работа № 7
“Авторизация, работа с формами и
DjangoAdmin.”**

Выполнил:
студент группы ИУ5-53
Кузнецов В.И.

Подпись:
Дата:

Москва 2017г.

1.Создайте view, которая возвращает форму для регистрации.

Поля формы:

- Login
- Пароль
- Повторный ввод пароля
- Email
- Фамилия
- Имя

```
def registration1(request):
    errors = []
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors.append('Введите логин')
        elif len(username) < 4:
            errors.append('Длина пароля должна быть >=4')

        password = request.POST.get('password')
        if not password:
            errors.append('Введите пароль')
        elif len(password) < 3:
            errors.append('Длина пароля должна быть >=3')
        password_repeat = request.POST.get('password2')

        if password != password_repeat:
            errors.append('Пароли не совпадают')

        if not errors:

            return HttpResponseRedirect('/login/')

    return render(request, 'registration.html', {'errors': errors})
```

Результат

Регистрация

ФИО:

Логин:

Пароль:

Телефон:

Повторите пароль:

[Уже зарегистрированы?](#)

2. Создайте view, которая возвращает форму для авторизации.

Поля формы:

- Login
- Пароль

```
def auth_view(request): # авторизация
    errors = []
    if request.method == 'POST':
        form = UserAuthenticationForm(request.POST)
        print('check valid')

        if form.is_bound:
            print('form is valid', request.POST.get('login'), request.POST.get('password'))

            try:
                print('trying to get data')

                bd_data = CustomerModel.objects.get(login=request.POST.get('login'))

                print('получил бд', bd_data)

                if request.POST.get('password') == bd_data.password:
                    print('авторизация этого пользователя')

                    user = authenticate(username=request.POST.get('login'), password=request.POST.get('password'))
                    print('success - ', user)
                    login(request, user)
                    return HttpResponseRedirect('/home/')

                else:
                    errors.append('неправильный пароль')

            except CustomerModel.DoesNotExist:
                errors.append('неправильный логин')

        else:
            form = UserAuthenticationForm()

    # print(form.errors.as_data())
    print(errors)
    return render(request, 'authentication.html', {'form': form, 'errors': errors})
```

Результат

Авторизация

Логин:

Пароль:

[Зарегистрироваться](#)

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

```
def reg_view(request): # регистрация
    form = UserRegistrationForm(request.POST or None)
    if request.method == 'POST':
        if form.is_valid():
            form.save(commit=True)

            user = User.objects.create_user(username=request.POST.get('login'), password=request.POST.get('password'),
                                             last_name=request.POST.get('fio'))

            # user.first_name = request.POST.get('fio')
            user.save()
            return HttpResponseRedirect('/auth/')

        else:
            form = UserRegistrationForm()

    return render(request, 'registration.html', {'form': form})
```

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

Пример(Неправильно введен логин, появляется ошибка, введенный логин не исчезает)

неправильный логин

Авторизация

Логин:

Пароль:

[Зарегистрироваться](#)

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

См п.2 и 3

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

Для логина используем метод `login_user`, `authenticate`

```
if request.POST.get('password') == bd_data.password:
    print('авторизация этого пользователя')

    user = authenticate(username=request.POST.get('login'), password=request.POST.get('password'))
    print('success - ', user)
    login(request, user)
    return HttpResponseRedirect('/home/')

```

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

```
class CustomerAccounts(View):
    def get(self, request):
        usr_type, usr_id = user_type(request)
        if not request.user.is_authenticated():
            return HttpResponseRedirect('/auth/')
        else:
            data = AccountModel.objects.all()
            return render(request, 'accounts_main.html', {'accts': data, 'usr_type': usr_type, 'usr_id': usr_id})

```

8. Реализовать view для выхода из аккаунта.

```
def logout_view(request):
    logout(request)
    return HttpResponseRedirect('/home/')

```

9. Заменить проверку на авторизацию на декоратор `login_required` для `ClassBasedView`

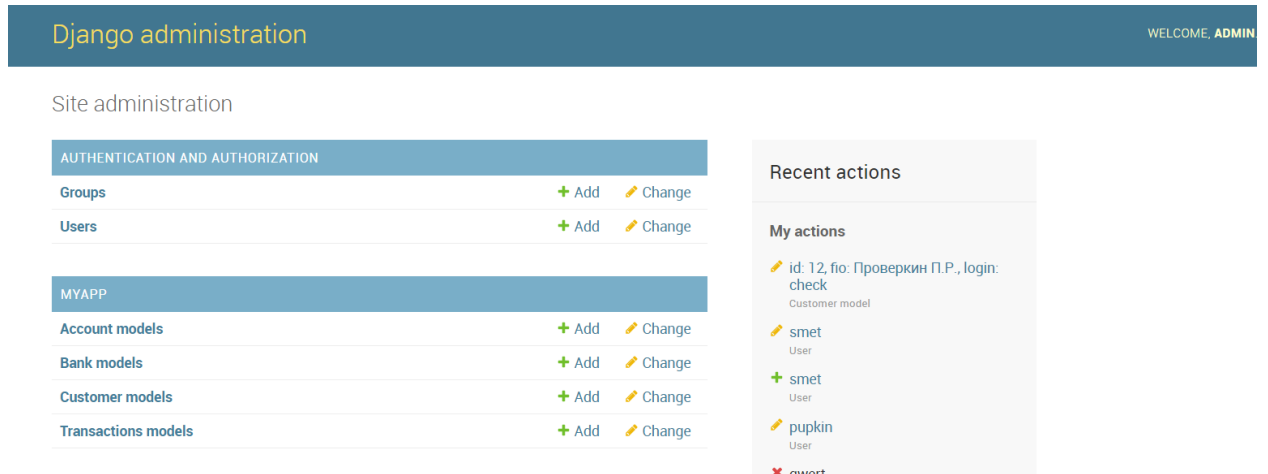
```
class CustomerTransactions(View):
    @method_decorator(login_required(login_url='/auth/'))
    def get(self, request):
        usr_type, usr_id = user_type(request)
        data = TransactionsModel.objects.all()
        return render(request, 'transactions_main.html', {'trans': data, 'usr_type': usr_type, 'usr_id': usr_id})

```

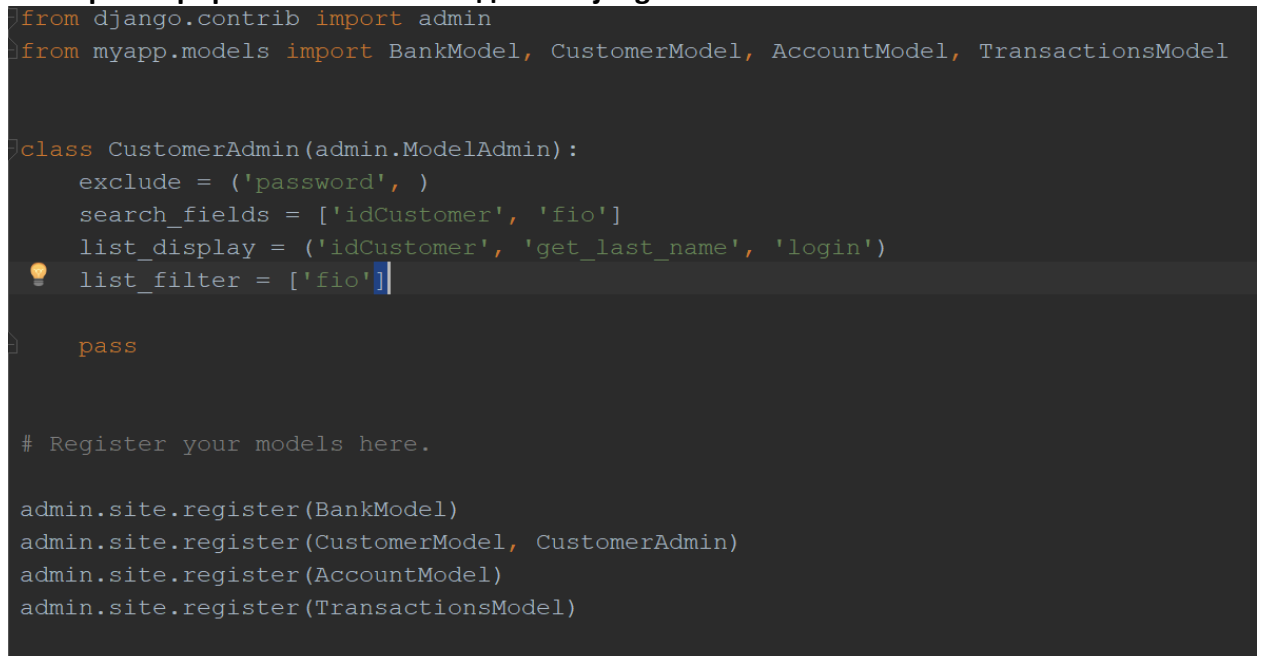
10. Добавить `superuser`'а через команду `manage.py`

	id	password	last_login	is_superuser	username
1	2	pbkdf2_sha256\$36000\$dop3iw9geHdB\$t9nLT/4FtG5K2cmY7gCxliBTBAQ82tWsk...	2017-12-14 21:40:52.108952	0	pupkin
2	4	pbkdf2_sha256\$36000\$li4XzmND9oCI\$UVoh5Ef5wEqQi7ZOFUMe17M+xq3ALciq/...	2017-12-15 11:24:06.951252	1	admin
3	5	pbkdf2_sha256\$36000\$x9mHJkQcjYe9\$KKVjfdgSd/mvkoEWSYIRNojf3w+o4/T7V...	2017-12-15 11:28:43.414882	0	test

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

См п.12

Добавленные пользователи и отображение модели Customer(Поиск, фильтры, и т.д.)

Select customer model to change

Q

Search

Action:

Go

0 of 5 selected

<input type="checkbox"/>	IDCUSTOMER	LAST NAME	LOGIN
<input type="checkbox"/>	13		admin
<input type="checkbox"/>	12	Проверкин	check
<input type="checkbox"/>	11		test
<input type="checkbox"/>	8	Пупкин	pupkin
<input type="checkbox"/>	1	Сметанкин	smet

5 customer models

ADD CUSTOMER MODEL +

FILTER

By fio

All

admin

test

Проверкин П.Р.

Пупкин П.У.

Сметанкин К.И.