

Отчёт по лабораторной работе №7
по курсу «Разработка интернет-приложений»
Работа с формами, авторизация, Django admin

Выполнил:

Волобуев В.Н., ИУ5-54

Преподаватель:

Гапанюк Ю.Е.

2016 г.

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.

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`'а 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

2) Листинг

```
# urls.py
```

```
from django.conf.urls import url
```

```
from . import views
```

```
urlpatterns = [
    url(r'^$', views.ExampleView.as_view(), name='start'),
    url(r'^bullets/', views.BulletsView.as_view()),
    url(r'^bullet/(?P<id>\d+)', views.BulletView.as_view(),
name='bullet_url'),
    url(r'^signup/', views.registration, name='signup'),
    url(r'^login/', views.authorization, name='login'),
    url(r'^logout/', views.exit, name='logout'),
]
```

views.py

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from django.views import View
from django.views.generic import ListView
from django.views.generic import TemplateView
from django.contrib.auth.decorators import login_required
```

```
#from lab7.models import Bullet
from lab7.forms import *
from lab7.models import *
from django import forms
from django.contrib.auth.hashers import make_password
from django.contrib.auth import authenticate, logout
from django.contrib import auth
```

```
# Create your views here.
```

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

```

class BulletsView(ListView):
    model = Bullet
    context_object_name = 'bullets'
    template_name = 'bullets.html'

    def get_queryset(self):
        qs = Bullet.objects.all().order_by('id').values()
        return qs

class BulletView(View):
    def get(self, request, id):
        data = Bullet.objects.get(id__exact=id)
        return render(request, 'bullet.html', {'bullet':data})

def registration_old(request):
    errors = []
    if request.method == 'POST':
        username = request.POST.get('username')
        if not username:
            errors.append('Login required')
        elif len(username)<5:
            errors.append('Login should be 5 or more symbols')

        password = request.POST.get('password')
        if not password:
            errors.append('Password required')
        elif len(password)<6:
            errors.append('Password length should be 6 or
more')

        password_repeat = request.POST.get('password2')

        if password != password_repeat:
            errors.append('Password should be similar')

```

```

        if not errors:

            return HttpResponseRedirect('/lab7/login')
        return render(request, 'lab7/logon.html', {'errors':
errors})

def registration(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('/lab7/')
        return render(request, 'signup.html', {'form': form})
    else:
        form = RegistrationForm()
        return render(request, 'signup.html', {'form': form})

def authorization(request):
    redirect_url = '/lab7/'
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            user =
auth.authenticate(username=form.cleaned_data['login'],
password=form.cleaned_data['password'])
            if user is not None:
                auth.login(request, user)
                return HttpResponseRedirect(redirect_url)
            else:
                form.add_error(None, 'Wrong login or
password')
        else:
            form = LoginForm()

```

```
        return render(request, 'login.html', {'form':form,
'continue': redirect_url})
```

```
@login_required
```

```
def exit(request):
```

```
    logout(request)
```

```
    return render(request, 'logout.html')
```

forms.py

```
from django import forms
```

```
from django.core.exceptions import ValidationError
```

```
from django.core.validators import validate_email
```

```
from django.contrib.auth.models import User
```

```
class LoginForm(forms.Form):
```

```
    login = forms.CharField(label='Логин')
```

```
    password = forms.CharField(label='Пароль',
widget=forms.PasswordInput)
```

```
class RegistrationForm(forms.Form):
```

```
    login = forms.CharField(label='Логин', min_length=5)
```

```
    password = forms.CharField(label='Пароль', min_length=8,
widget=forms.PasswordInput)
```

```
    repeat_password = forms.CharField(label='Повторите
пароль', widget=forms.PasswordInput)
```

```
    email = forms.CharField(label='Адрес электронной Почты')
```

```
    first_name = forms.CharField(label='Имя')
```

```
    last_name = forms.CharField(label='Фамилия')
```

```
def clean_login(self):
```

```
    login = self.cleaned_data['login']
```

```
    if User.objects.filter(username=login):
```

```
        raise ValidationError('Этот login уже занят')
```

```
    return login
```

```

def clean_email(self):
    email = self.cleaned_data['email']
    validate_email( self.cleaned_data['email'])
    if User.objects.filter(email=email):
        raise ValidationError('Этот email уже
зарегистрирован')
    return self.cleaned_data['email']

def clean(self):
    cleaned_data = super(RegistrationForm, self).clean()
    if self.cleaned_data.get('password') and
self.cleaned_data.get('repeat_password'):
        if self.cleaned_data['password'] !=
self.cleaned_data['repeat_password']:
            raise ValidationError('Пароли не совпадают')
    return cleaned_data

def save(self):
    user =
User.objects.create_user(username=self.cleaned_data['login'],

email=self.cleaned_data['email'],

password=self.cleaned_data['password'],

first_name=self.cleaned_data['first_name'],

last_name=self.cleaned_data['last_name'],

        )

    return user

```

models.py

```

from django.db import models
from django.contrib.auth.models import AbstractUser

```

```

# Create your models here.

```



```

class Bullet(models.Model):
    name = models.CharField(max_length=30)
    description = models.CharField(max_length=255)
    datetime = models.DateTimeField(auto_now=True)

class MyUser(AbstractUser):
    regdate = models.DateTimeField(auto_now_add=True)

class Meta(AbstractUser.Meta):
    swappable = 'AUTH_USER_MODEL'

```

admin.py

```

from django.contrib import admin

```

```

# Register your models here.

```

```

from lab7.models import Bullet

```

```

class BulletAdmin(admin.ModelAdmin):
    list_display = ('name', 'description', 'datetime')
    list_filter = ['datetime']
    search_fields = ('id', 'name')

```

```

admin.site.register(Bullet, BulletAdmin)

```

templates/base.html

```

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="{% static
'css/bootstrap.css' %}">
    <title>{% block title %}Тестовый вывод{% endblock
%}</title>

```

```

</head>

<body>

<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile
display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed"
data-toggle="collapse" data-target="#navbar" aria-expanded="false"
aria-controls="navbar">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand">IAD #7</a>
    </div>

    <!-- Collect the nav links, forms, and other content for
toggling -->
    <div class="collapse navbar-collapse" id="navbar">
      <ul class="nav navbar-nav">
        <li><a href="/lab7/">Базовый класс</a></li>
        <li><a href="/lab7/bullets/">Объявления</a></li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        {% if user.is_authenticated %}
          <li><div style="padding-
top:15px"><i><b>Здравствуйте,
{{ user.get_short_name }}</b></i></div></li>
          <li><a href="{% url 'logout' %}">Выйти из
{{ user.get_username }}</a></li>
        {% else %}
          <li><div style="padding-top:15px"><i><b>Вы вошли
как Гость</b></i></div></li>
          <li><a href="{% url 'signup'
%}">Зарегистрироваться</a></li>

```

```

        <li><a href="{% url 'login' %}">Войти</a></li>
    {% endif %}
</ul>

</div><!-- /.navbar-collapse -->

</div><!-- /.container-fluid -->

</nav>

<div>

    {% block body %}Redefined by spawnlings{% endblock %}

</div>

<!-- Bootstrap core JavaScript
===== -->

<!-- Placed at the end of the document so the pages load
faster -->

<script src="{% static 'js/jquery-3.1.1.min.js'
%}"></script>

<script src="{% static 'js/bootstrap.min.js' %}"></script>

</body>

</html>

```

templates/signup.html

```

{% extends 'base.html' %}

{% block title %}Регистрация{% endblock %}

{% block body %}

    {{ form.non_field_errors }}

    <form action="/lab7/signup/" method="post" class="form-
horizontal" enctype="multipart/form-data">

        {% csrf_token %}

        {% for i in form %}

            <div class="form-group">

                <label class="col-sm-4">{{ i.label }}</label>

                <div class="col-sm-8">

                    <div>

```

```

        {{ i }}
    </div>
    <div>
        {{ i.errors }}
    </div>
</div>
</div>
{% endfor %}

<div class="form-group">
    <div class="col-sm-offset-4 col-sm-8">
        <input type="submit" value="Зарегистрировать меня"
class="btn btn-default"/>
    </div>
</div>
</form>
{% endblock %}

```

templates/login.html

```

{% extends 'base.html' %}

{% block title %}Вход{% endblock %}

{% block body %}

    {{ form.non_field_errors }}

    <form action="/lab7/login/" method="post" class="form-
horizontal">

        {% csrf_token %}

        {% for i in form %}
            <div class="form-group">
                <label class="col-sm-
2">{{ i.label }}</label>

                <div class="col-sm-10">
                    <div>

```

```

                {{ i }}
            </div>

        </div>
    </div>
{% endfor %}

<div class="form-group">
    <div class="col-sm-offset-2 col-sm-
10">
        <input type="submit" value="Войти"
class="btn btn-default"/>
    </div>
</div>

</form>
{% endblock %}

# templates/logout.html
{% extends 'base.html' %}

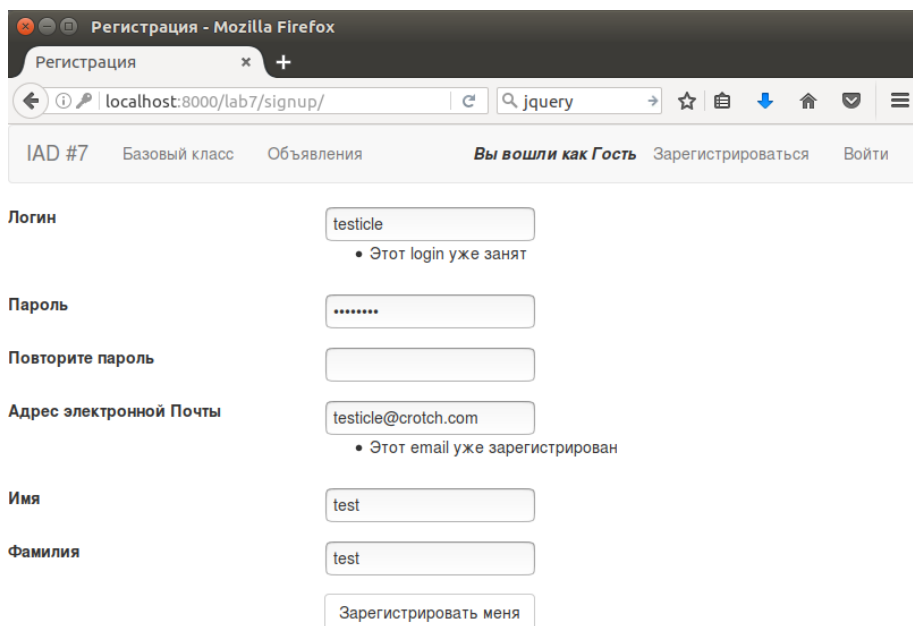
{% block title %}Выход{% endblock %}

{% block body %}
<p>Выход произошёл успешно.</p>
{% endblock %}

```

3) Результаты работы

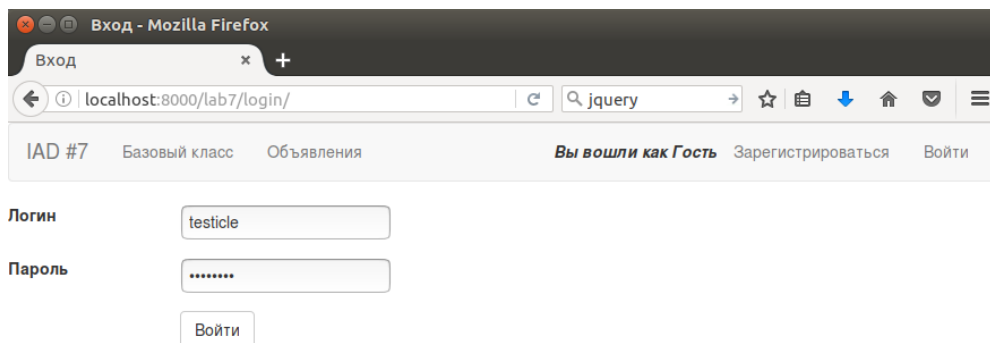
1) Попытка регистрации на уже существующий логин



The screenshot shows a Mozilla Firefox browser window titled "Регистрация". The address bar displays "localhost:8000/lab7/signup/". The page header includes "IAD #7", "Базовый класс", "Объявления", and a navigation bar with "Вы вошли как Гость", "Зарегистрироваться", and "Войти". The registration form contains the following fields and messages:

- Логин:** Input field with "testicle". Below it, a message: "• Этот login уже занят".
- Пароль:** Input field with masked characters ".....".
- Повторите пароль:** Empty input field.
- Адрес электронной Почты:** Input field with "testicle@crotch.com". Below it, a message: "• Этот email уже зарегистрирован".
- Имя:** Input field with "test".
- Фамилия:** Input field with "test".
- Зарегистрировать меня:** Button.

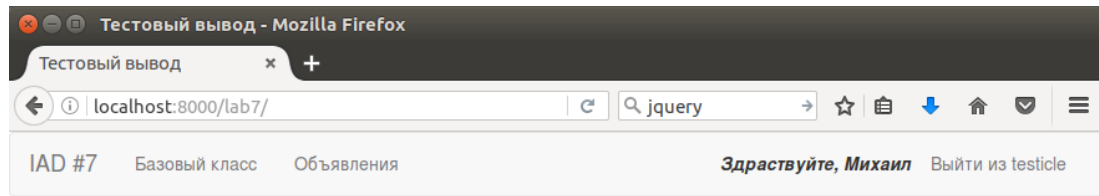
2) Вход под логином testicle



The screenshot shows a Mozilla Firefox browser window titled "Вход". The address bar displays "localhost:8000/lab7/login/". The page header is identical to the registration page. The login form contains the following fields and elements:

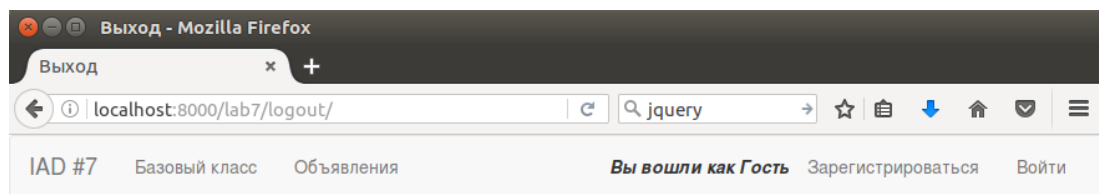
- Логин:** Input field with "testicle".
- Пароль:** Input field with masked characters ".....".
- Войти:** Button.

3) Подтверждение успешного входа (изменился navbar)



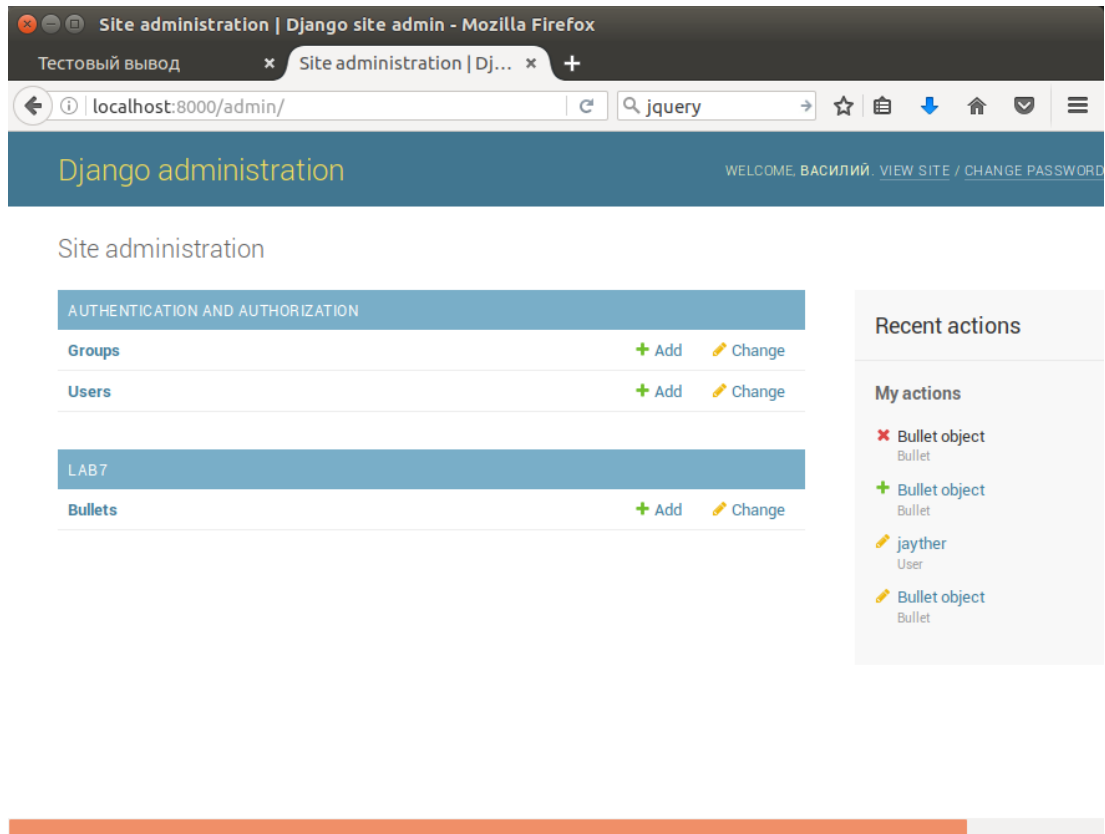
Redefined by spawnlings

4) Успешный logout

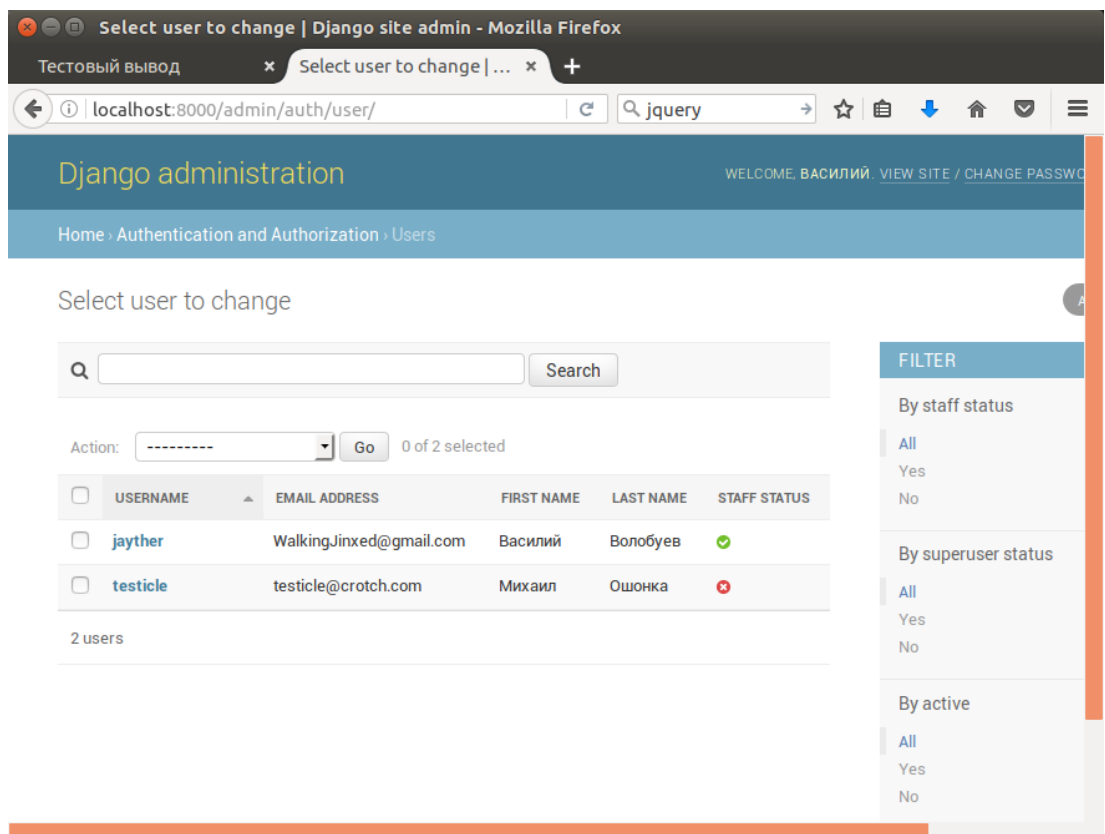


Выход произошёл успешно.

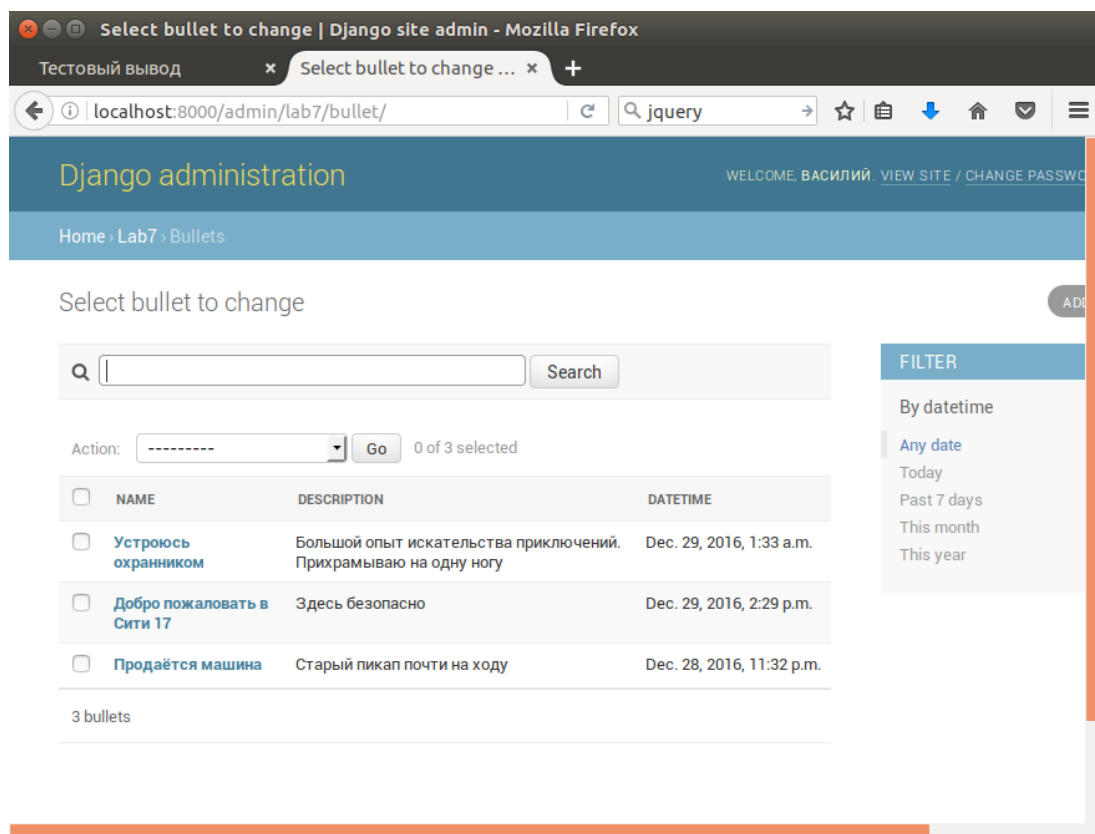
5) Вошёл в Django Admin под логином jayther (superuser)



6) Отображение пользователей в админке



7) Отображение объявлений в админке



The screenshot shows the Django administration interface for the 'lab7' app, specifically the 'Bullets' model. The page title is 'Select bullet to change'. The breadcrumb trail is 'Home > Lab7 > Bullets'. The main content area displays a table with 3 bullets. The table has columns for 'NAME', 'DESCRIPTION', and 'DATETIME'. The first bullet is 'Устроюсь охранником' with a description 'Большой опыт искательства приключений. Прихрамываю на одну ногу' and a datetime of 'Dec. 29, 2016, 1:33 a.m.'. The second bullet is 'Добро пожаловать в Сити 17' with a description 'Здесь безопасно' and a datetime of 'Dec. 29, 2016, 2:29 p.m.'. The third bullet is 'Продаётся машина' with a description 'Старый пикап почти на ходу' and a datetime of 'Dec. 28, 2016, 11:32 p.m.'. A search bar is at the top, and a filter sidebar is on the right.

Django administration

WELCOME, ВАСИЛИЙ. [VIEW SITE](#) / [CHANGE PASSWORD](#)

Home > Lab7 > Bullets

Select bullet to change

Search

Action: Go 0 of 3 selected

<input type="checkbox"/>	NAME	DESCRIPTION	DATETIME
<input type="checkbox"/>	Устроюсь охранником	Большой опыт искательства приключений. Прихрамываю на одну ногу	Dec. 29, 2016, 1:33 a.m.
<input type="checkbox"/>	Добро пожаловать в Сити 17	Здесь безопасно	Dec. 29, 2016, 2:29 p.m.
<input type="checkbox"/>	Продаётся машина	Старый пикап почти на ходу	Dec. 28, 2016, 11:32 p.m.

3 bullets

FILTER

By datetime

Any date

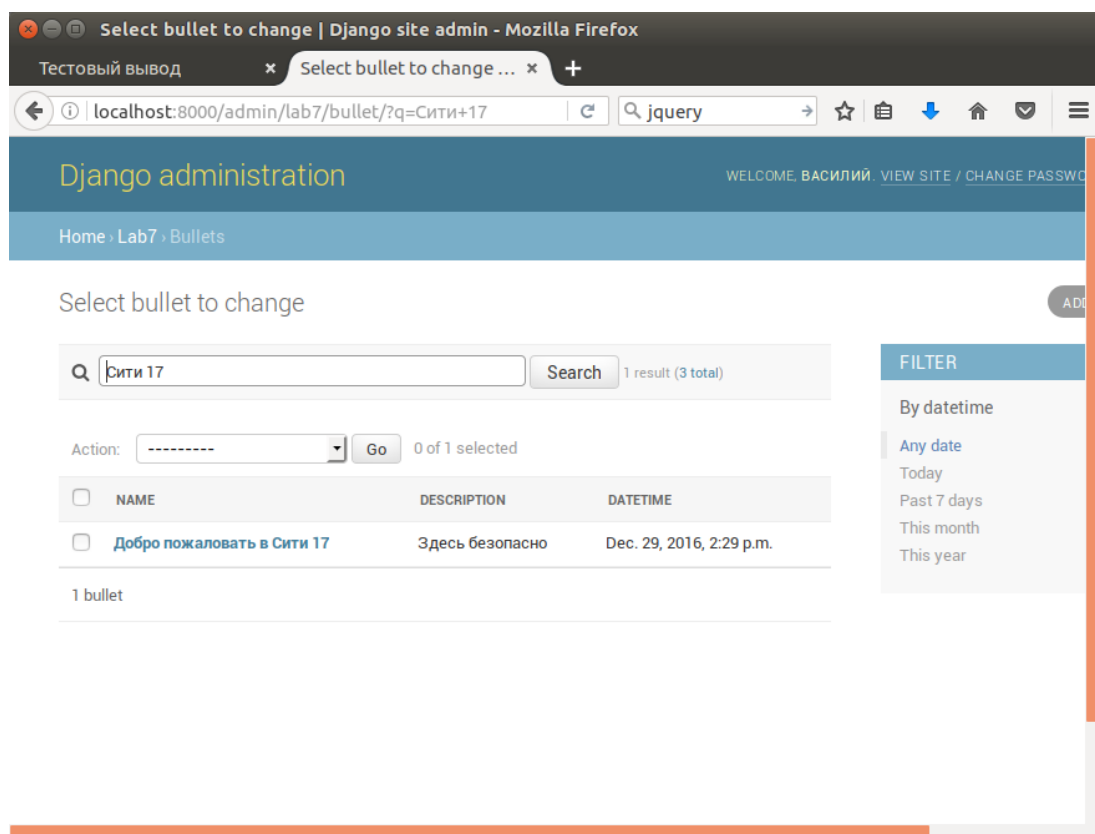
Today

Past 7 days

This month

This year

8) Рабочий поиск по имени объявления



The screenshot shows the Django administration interface for the 'lab7' app, specifically the 'Bullets' model. The page title is 'Select bullet to change'. The breadcrumb trail is 'Home > Lab7 > Bullets'. The main content area displays a table with 1 bullet. The table has columns for 'NAME', 'DESCRIPTION', and 'DATETIME'. The first bullet is 'Добро пожаловать в Сити 17' with a description 'Здесь безопасно' and a datetime of 'Dec. 29, 2016, 2:29 p.m.'. A search bar at the top shows the search term 'Сити 17' and the result count '1 result (3 total)'. A filter sidebar is on the right.

Django administration

WELCOME, ВАСИЛИЙ. [VIEW SITE](#) / [CHANGE PASSWORD](#)

Home > Lab7 > Bullets

Select bullet to change

Search Сити 17 1 result (3 total)

Action: Go 0 of 1 selected

<input type="checkbox"/>	NAME	DESCRIPTION	DATETIME
<input type="checkbox"/>	Добро пожаловать в Сити 17	Здесь безопасно	Dec. 29, 2016, 2:29 p.m.

1 bullet

FILTER

By datetime

Any date

Today

Past 7 days

This month

This year

9) Фильтр по дате последнего изменения/публикации объявления

The screenshot shows the Django administration interface in a Mozilla Firefox browser. The page title is "Select bullet to change | Django site admin". The browser address bar shows "localhost:8000/admin/lab7/bullet/?datetime__gte=2". The page header includes "Django administration" and a welcome message "WELCOME, ВАСИЛИЙ. VIEW SITE / CHANGE PASSWORD". The breadcrumb trail is "Home > Lab7 > Bullets".

The main content area is titled "Select bullet to change". It features a search bar with a magnifying glass icon and a "Search" button. Below the search bar is an "Action:" dropdown menu with a "Go" button. The table below shows two results:

<input type="checkbox"/>	NAME	DESCRIPTION	DATETIME
<input type="checkbox"/>	Устроюсь охранником	Большой опыт искательства приключений. Прихрамываю на одну ногу	Dec. 29, 2016, 1:33 a.m.
<input type="checkbox"/>	Добро пожаловать в Сити 17	Здесь безопасно	Dec. 29, 2016, 2:29 p.m.

Below the table, it says "2 bullets".

On the right side, there is a "FILTER" sidebar. It has a section "By datetime" with the following options: "Any date", "Today", "Past 7 days", "This month", and "This year".