Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Отчет по лабораторной работе № 7 по дисциплине «Разработка интернет приложений».

ИСПОЛНИТЕЛЬ: студент группы ИУ5-53

Шубникова М.В.

Москва, 2017

# Задание

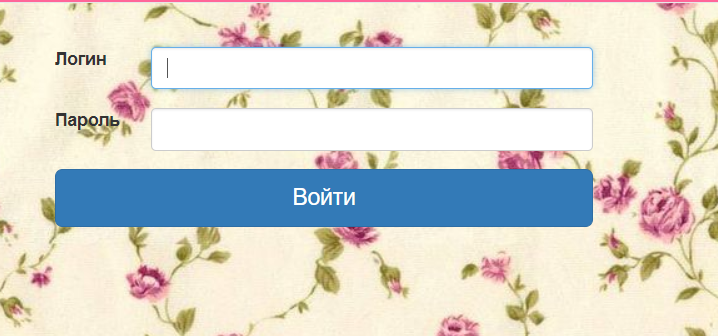
Основная цель данной лабораторной работы – научиться обрабатывать веб-формы на стороне приложения, освоить инструменты, которые предоставляет Django, по работе с формами. Также в этой лабораторной работе вы освоите инструменты Django по работе с авторизацией и реализуете простейшую авторизацию. Напоследок, вы познакомитесь с инструментом администрирования Django – как в несколько строчек кода сделать панель администратора сайта.

# Код программы

**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 = **"/"  
  
 def** form\_valid(self, form):  
 *# Получаем объект пользователя на основе введённых 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)  
*# Create your views here.***class** flowers\_view(ListView):  
 model = models.Flowers  
  
 **def** get(self, request):  
 d = self.model.objects.values(**'id'**,**'name'**,**'picture'**)  
 paginator = Paginator(d, 6) *# Show 25 contacts per page* page = request.GET.get(**'page'**)  
 **try**:  
 contacts = paginator.page(page)  
 **except** PageNotAnInteger:  
 *# If page is not an integer, deliver first page.* contacts = paginator.page(1)  
 **except** EmptyPage:  
 *# If page is out of range (e.g. 9999), deliver last page of results.* contacts = paginator.page(paginator.num\_pages)  
 **if**(contacts.paginator.num\_pages <= 4):  
 r = range(1, contacts.paginator.num\_pages+1)  
 **else**:  
 r = range(contacts.number-1, contacts.number+4)  
 **return** render(request, **'flowers.html'**, {**"contacts"**: contacts, **'user'**: request.user, **'range'**: r})  
  
**class** flower\_view(View):  
 model = models.Flowers  
 model1 = models.Flower\_user  
 model2 = User  
 **def** get(self, request,id1):  
  
 b = dict(flower=self.model.objects.filter(id=id1))  
  
 a = dict(users=self.model1.objects.filter(flower\_id=id1))  
  
 b.update(a)  
 c={**'user'**:request.user}  
 b.update(c)  
 d={**'id'**:id1}  
 b.update(d)  
 **return** render(request, **"flower.html"**, b)  
  
**class** create\_flower(FormView):  
 form\_class = form.Flower\_form  
 model = models.Flowers  
 template\_name = **"create\_flower.html"** success\_url = **"/flowers/"  
 def** form\_valid(self, form):  
 *# Создаём пользователя, если данные в форму были введены корректно.* form.save()  
  
 *# Вызываем метод базового класса* **return** super(create\_flower, self).form\_valid(form)  
  
 **def** post(self, request, \*args, \*\*kwargs):  
 form1 = form.Flower\_form(request.POST)  
 **if** form1.is\_valid() **and** request.FILES.get(**'picture'**):  
 obj = models.Flowers()  
 obj = form1.save(commit=**False**)  
 obj.picture = request.FILES[**'picture'**]  
 obj.save()  
 b = dict(flower=self.model.objects.filter(name=obj.name))  
 **return** HttpResponseRedirect(**"/flower/"**+str(b[**'flower'**][0].id))  
 **else**:  
 form1.add\_error(**'picture'**, **'Добавьте картинку'**)  
 **return** render(request, **'create\_flower.html'**, {**'form'**:form1})  
  
**class** add(View):  
 model = models.Flowers  
 model1 = models.Flower\_user  
 model2 = User  
 **def** get(self, request, id2):  
 **if** request.user.is\_authenticated:  
 u = self.model2.objects.filter(id=request.user.id)  
 a = dict(users=self.model1.objects.filter(flower\_id=id2))  
 b = dict(flower=self.model.objects.filter(id=id2))  
 ab = dict(users=self.model1.objects.filter(flower\_id=id2))  
 b.update(ab)  
 er = 0  
 **for** i **in** a[**'users'**]:  
 **if** i.user\_id == request.user:  
 er = 1  
 **break  
 if** er == 1:  
 ca = {**'uze'**: **"Вы уже заказали эти цветы"**}  
 b.update(ca)  
 k = {**'message'**:**""**}  
 ca = {**'error'**:**''**}  
 b.update(k)  
 b.update(ca)  
 d = {**'id'**: id2}  
 b.update(d)  
 **return** render(request, **"flower.html"**, b)  
 **else**:  
  
 a=models.Flower\_user()  
  
 a.flower\_id=b[**'flower'**][0]  
 a.user\_id=u[0]  
 a.save()  
  
 k = {**'message'**:**"Вы успешно заказали цветы"**}  
 ca = {**'error'**:**''**}  
 b.update(k)  
 b.update(ca)  
 d = {**'id'**: id2}  
 b.update(d)  
  
 **return** render(request, **"flower.html"**, b)  
 **else**:  
 b = dict(flower=self.model.objects.filter(id=id2))  
 a = {**'error'**: **"Вы не авторизованны. Войдите или зарегистрируйтесь"**}  
 c = {**'message'**:**''**}  
 b.update(a)  
 b.update(c)  
 d = {**'id'**: id2}  
 b.update(d)  
 ab = dict(users=self.model1.objects.filter(flower\_id=id2))  
 b.update(ab)  
 **return** render(request, **"flower.html"**, b)

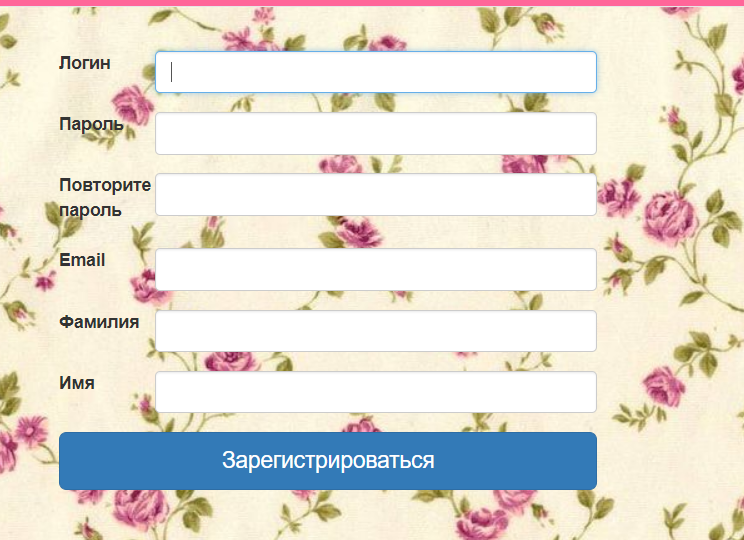
# Результаты работы

Форма для регистрации



Форма для авторизации

Возникновение ошибок при авторизации (выведение ошибки)



Страница администрирования

