**Московский государственный технический**

**университет им. Н.Э. Баумана.**

Факультет «Информатика и системы управления»

Кафедра ИУ5. Курс «Разработка интернет приложений»

Отчет по лабораторной работе №6,7.

**«Работа с СУБД. Авторизация, работа с формами и Django Admin.»**

Выполнил: Проверил:

студент группы ИУ5-52 доцент каф. ИУ5

Ефимов К.А. Гапанюк Ю.Е.

Подпись и дата: Подпись и дата:

Москва, 2017 г.



Задание и порядок выполнения

В этой лабораторной работе вы познакомитесь с популярной СУБД MySQL, создадите свою базу данных. Также вам нужно будет дополнить свои классы предметной области, связав их с созданной базой. После этого вы создадите свои модели с помощью Django ORM, отобразите объекты из БД с помощью этих моделей и ClassBasedViews.

Для сдачи вы должны иметь:

1. Скрипт с подключением к БД и несколькими запросами.
2. Набор классов вашей предметной области с привязкой к СУБД (класс должен  уметь хотя бы получать нужные записи из БД и преобразовывать их в объекты  этого класса)
3. Модели вашей предметной области
4. View для отображения списка ваших сущностей

Задание и порядок выполнения

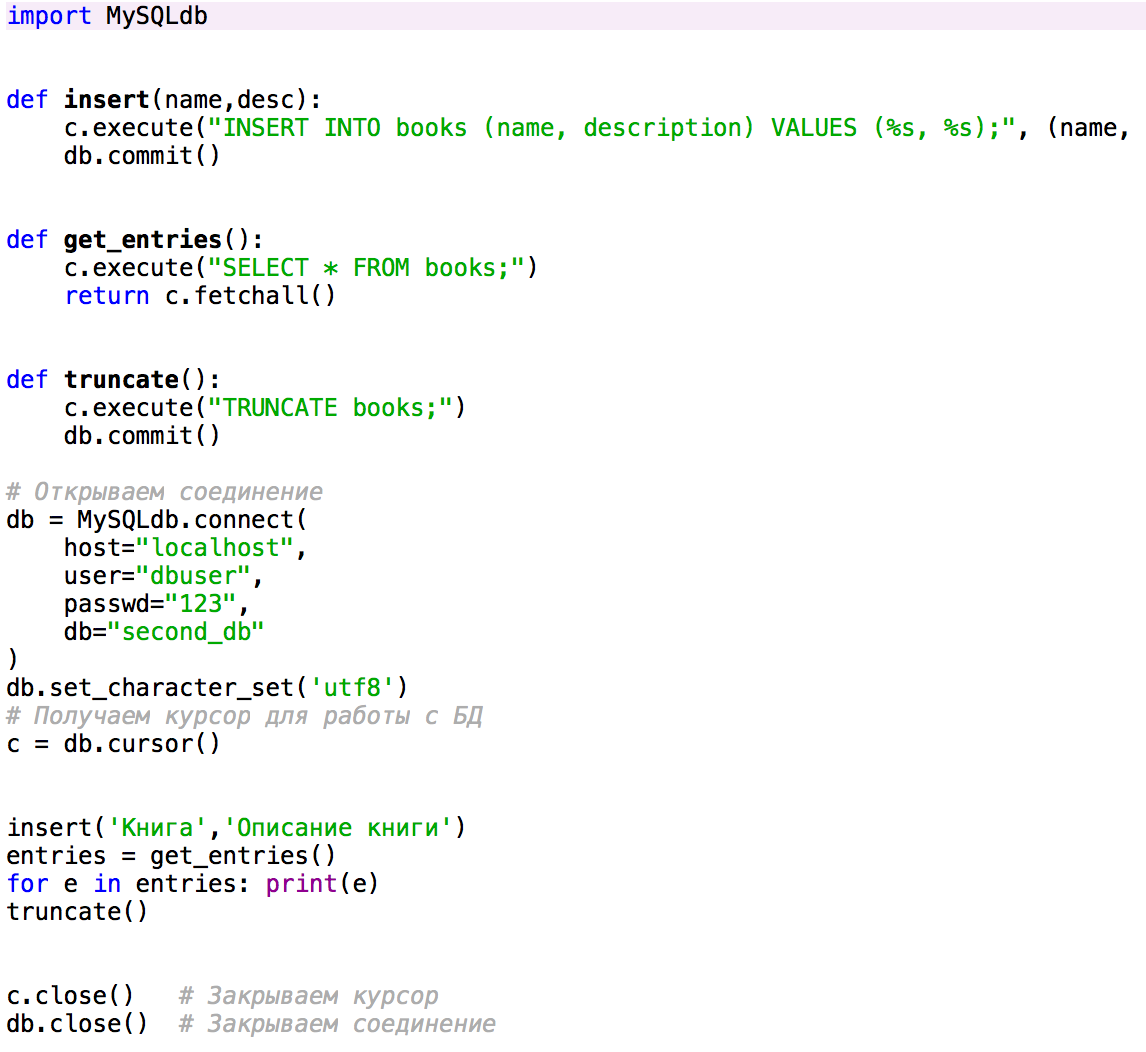
Основная цель данной лабораторной работы – научиться обрабатывать веб-формы на стороне приложения, освоить инструменты, которые предоставляет Django, по работе с формами. Также в этой лабораторной работе вы освоите инструменты Django по работе с авторизацией и реализуете простейшую авторизацию. 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’a через комманду 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

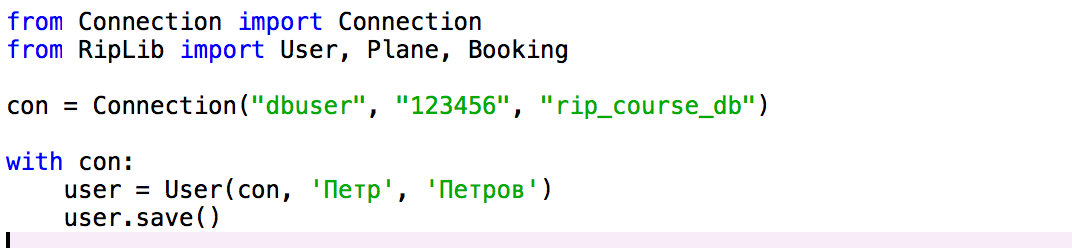
Подключение к БД MySQL



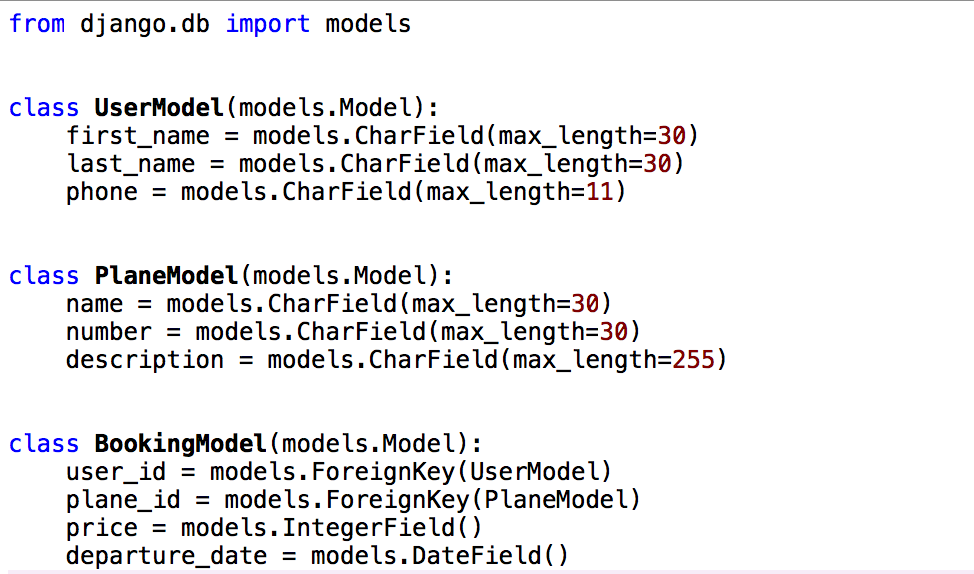
first\_script.py



second\_script.py



RipLibModels.py



RipLib.py

class User:

def \_\_init\_\_(self, db\_connection, first\_name, last\_name, phone=None):

self.db\_connection = db\_connection.connection

self.first\_name = first\_name

self.last\_name = last\_name

self.phone = phone

def save(self):

c = self.db\_connection.cursor()

c.execute("INSERT INTO users (first\_name, last\_name, phone) VALUES (%s, %s, %s);",

(self.first\_name, self.last\_name, self.phone))

self.db\_connection.commit()

c.close()

class Plane:

def \_\_init\_\_(self, db\_connection, name,number,description=None):

self.db\_connection = db\_connection.connection

self.name = name

self.number = number

self.description = description

def save(self):

c = self.db\_connection.cursor()

c.execute("INSERT INTO planes (name, number, description) VALUES (%s, %s, %s);",

(self.name, self.number, self.description))

self.db\_connection.commit()

c.close()

class Booking:

def \_\_init\_\_(self, db\_connection, user\_id,plane\_id,price,departure\_date):

self.db\_connection = db\_connection.connection

self.user\_id = user\_id

self.plane\_id = plane\_id

self.price = price

self.departure\_date = departure\_date

def save(self):

c = self.db\_connection.cursor()

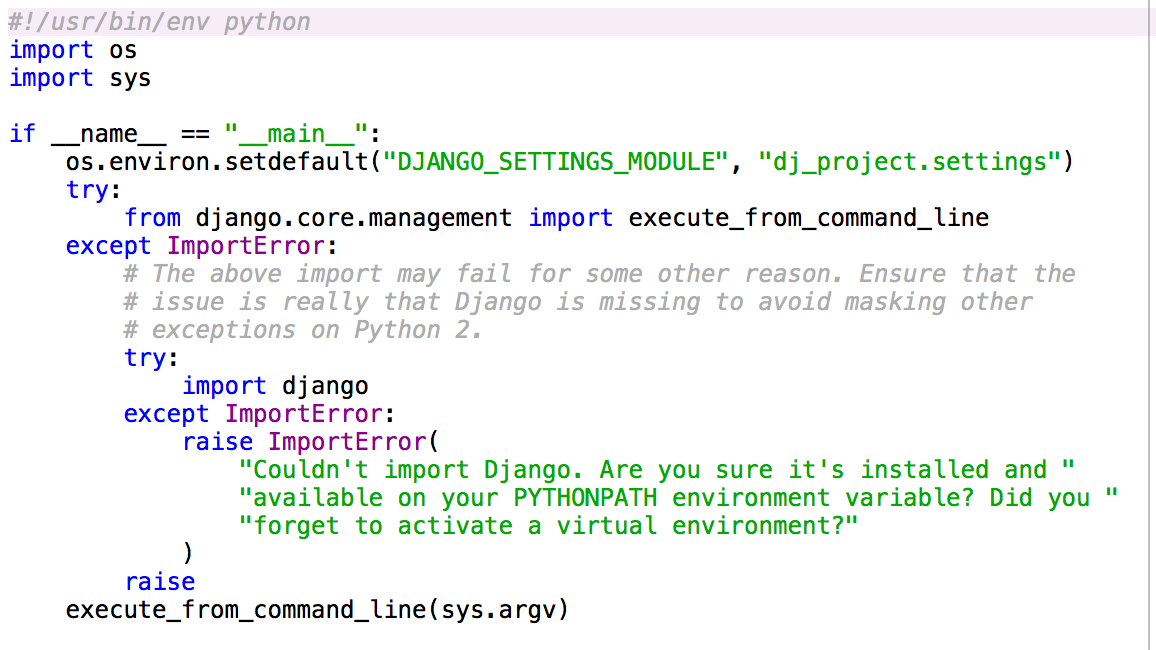
c.execute("INSERT INTO bookings (user\_id, plane\_id, price, departure\_date) VALUES (%d, %d, %d, %s);",

(self.user\_id, self.plane\_id, self.price, self.departure\_date))

self.db\_connection.commit()

c.close()

manage.py



Настройки

import os

# Build paths inside the project like this: os.path.join(BASE\_DIR, ...)

BASE\_DIR = os.path.dirname(os.path.dirname(os.path.abspath(\_\_file\_\_)))

# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/1.11/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!

SECRET\_KEY = '9+\_zz16(292\_8)h$j\_gns-4si-rayue\*q&u11\*%82!hxi%n2i('

# SECURITY WARNING: don't run with debug turned on in production!

DEBUG = True

ALLOWED\_HOSTS = []

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

MIDDLEWARE = [

'django.middleware.security.SecurityMiddleware',

'django.contrib.sessions.middleware.SessionMiddleware',

'django.middleware.common.CommonMiddleware',

'django.middleware.csrf.CsrfViewMiddleware',

'django.contrib.auth.middleware.AuthenticationMiddleware',

'django.contrib.messages.middleware.MessageMiddleware',

'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

ROOT\_URLCONF = 'dj\_project.urls'

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': [],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

],

},

},

]

WSGI\_APPLICATION = 'dj\_project.wsgi.application'

# Database

# https://docs.djangoproject.com/en/1.11/ref/settings/#databases

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'second\_db',

'USER': 'dbuser',

'PASSWORD': '123',

'HOST': 'localhost',

'PORT': 3306,

'OPTIONS': {'charset':'utf8'},

'TEST\_CHARSET': 'utf8',

}

}

# Password validation

# https://docs.djangoproject.com/en/1.11/ref/settings/#auth-password-validators

AUTH\_PASSWORD\_VALIDATORS = [

{

'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

},

{

'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

},

]

# Internationalization

# https://docs.djangoproject.com/en/1.11/topics/i18n/

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'

USE\_I18N = True

USE\_L10N = True

USE\_TZ = True

MY\_APPS = ['labs']

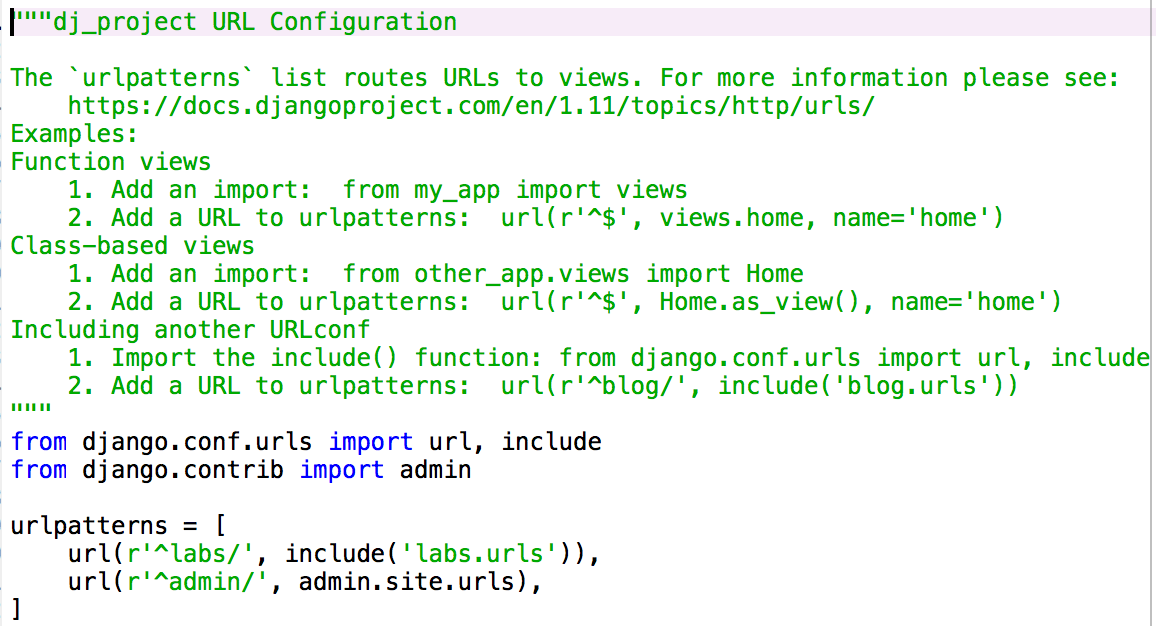
INSTALLED\_APPS += MY\_APPS

# Static files (CSS, JavaScript, Images)

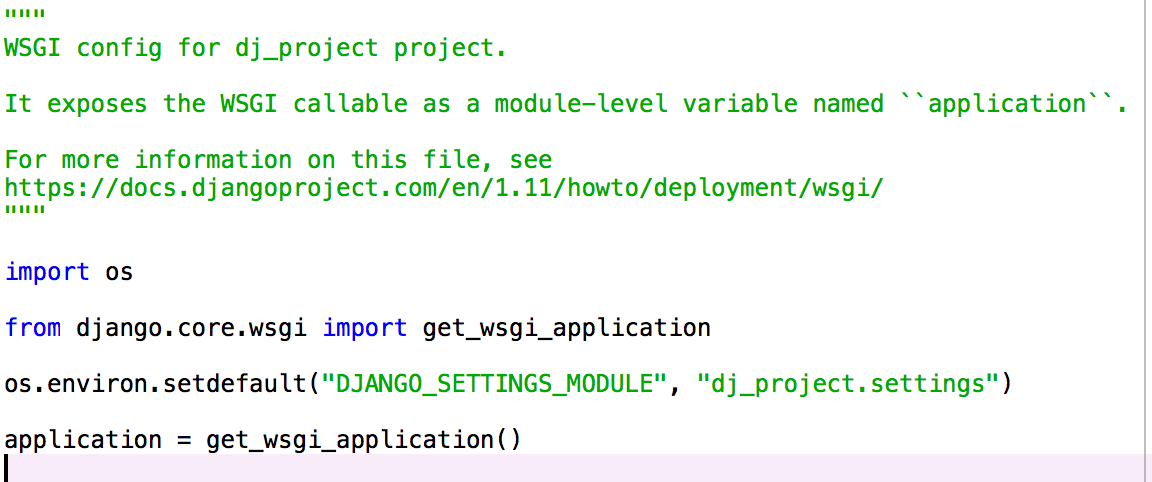
# https://docs.djangoproject.com/en/1.11/howto/static-files/

STATIC\_URL = '/static/'

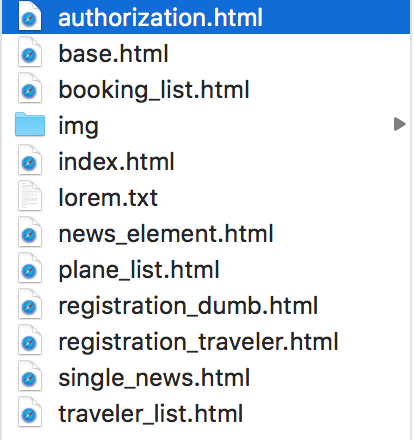
URLS.PY



WSGI.PY



Файлы html



Файл views.py

from django.shortcuts import render

from django.http import HttpResponseRedirect, HttpResponse

from django.views.generic import ListView

from django import forms

from django.contrib.auth import authenticate,login, logout

from django.contrib.auth.decorators import login\_required

from .models import \*

# Create your views here.

class TravelerList(ListView):

model = Traveler

template\_name = 'traveler\_list.html'

class PlaneList(ListView):

model = Plane

template\_name = 'plane\_list.html'

class BookingList(ListView):

model = Booking

template\_name = 'booking\_list.html'

def registration\_dumb(request):

errors = {}

request.encoding = 'utf-8'

if request.method == 'POST':

username = request.POST.get('username')

if not username:

errors['uname']='Введите логин'

elif len(username) < 5:

errors['uname']='Длина логина должна быть не меньше 5 символов'

if User.objects.exist(username=username):

errors['uname']='Такой логин уже занят'

password = request.POST.get('password')

if not password:

errors['psw']='Введите пароль'

elif len(password) < 2:

errors['psw']='Длина пароля должна быть не меньше 2 символов'

password\_repeat = request.POST.get('password2')

if password\_repeat != password\_repeat:

errors['psw2']='Пароли должны совпадать'

email = request.POST.get('email')

if not email:

errors['email']='Введите email'

last\_name = request.POST.get('last\_name')

if not last\_name:

errors['lname']='Введите фамилию'

first\_name = request.POST.get('first\_name')

if not first\_name:

errors['fname']='Введите имя'

if not errors:

user = User.objects.create\_user(username, email, password)

trav = Traveler()

trav.user = user

trav.first\_name = first\_name

trav.last\_name = last\_name

trav.save()

return HttpResponseRedirect('/labs/travelers')

else:

context = {'errors':errors, 'username':username, 'email': email, 'last\_name': last\_name, 'first\_name': first\_name}

return render(request, 'registration\_dumb.html',context)

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

class RegistrationForm(forms.Form):

username = forms.CharField(min\_length=5,label='Логин')

password = forms.CharField(min\_length=8,widget=forms.PasswordInput, label='Пароль')

password2 = forms.CharField(min\_length=8, widget=forms.PasswordInput, label='Повторите ввод')

email = forms.EmailField(label='Email')

last\_name = forms.CharField(label='Фамилия')

first\_name = forms.CharField(label='Имя')

def registration\_traveler(request):

if request.method == 'POST':

form = RegistrationForm(request.POST)

is\_val = form.is\_valid()

data = form.cleaned\_data

if data['password']!=data['password2']:

is\_val = False

form.add\_error('password2',['Пароли должны совпадать'])

if User.objects.filter(username=data['username']).exists():

form.add\_error('username',['Такой логин уже занят'])

is\_val = False

if is\_val:

data = form.cleaned\_data

user = User.objects.create\_user(data['username'], data['email'], data['password'])

trav = Traveler()

trav.user = user

trav.first\_name = data['first\_name']

trav.last\_name = data['last\_name']

trav.save()

return HttpResponseRedirect('/labs/authorization')

else:

form = RegistrationForm()

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

@login\_required(login\_url='/labs/authorization')

def success\_authorization(request):

return HttpResponseRedirect('/labs')

def success\_authorization\_dumb(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('/labs/')

else:

return HttpResponseRedirect('/labs/authorization')

def authorization(request):

errors = {}

if request.method == 'POST':

username = request.POST.get('username')

if not username:

errors['uname']='Введите логин'

elif len(username) < 5:

errors['uname']='Длина логина должна быть не меньше 5 символов'

password = request.POST.get('password')

if not password:

errors['psw']='Введите пароль'

elif len(password) < 2:

errors['psw']='Длина пароля должна быть не меньше 2 символов'

user = authenticate(request, username=username, password=password)

#user = authenticate(request, username='petrov',password='12345678')

if user is None and 'uname' not in errors.keys() and 'psw' not in errors.keys():

errors['login'] = 'Логин или пароль введены неправильно'

if not errors:

login(request,user)

#return HttpResponseRedirect('/labs/success\_authorization\_dumb')

return HttpResponseRedirect('/labs/success\_authorization')

else:

context = {'errors':errors}

return render(request, 'authorization.html',context)

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

def logout\_view(request):

logout(request)

return HttpResponseRedirect('/labs/')

class AutorizationForm(forms.Form):

pass

def index(request):

lorem = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris venenatis sem eu neque elementum semper. Aenean a quam enim. Praesent dictum, magna non tincidunt luctus, tortor tellus lobortis lorem, eu sodales felis libero sit amet nulla. Nam rhoncus faucibus fermentum. Sed nec est tellus. Vestibulum vitae volutpat sem, et ullamcorper tortor. Donec venenatis libero vel metus luctus eleifend. Praesent eleifend metus tincidunt, tempus leo a, volutpat dui." \

"Integer mattis cursus ante, non maximus ligula pellentesque eu. Phasellus semper libero ac tortor auctor placerat. Quisque quam ipsum, gravida vitae risus at, fringilla vestibulum nibh. Donec fermentum accumsan velit vel rutrum. In imperdiet, leo nec finibus vehicula, tellus massa vehicula arcu, non feugiat leo ex ac dolor. In sagittis augue quis metus suscipit dapibus. Curabitur ultrices erat malesuada, viverra arcu ornare, accumsan est." \

"Suspendisse dignissim odio at nibh malesuada, sit amet volutpat ante pellentesque. Ut id lorem commodo, mollis sapien sit amet, vestibulum magna. Curabitur a convallis augue, eu hendrerit ipsum. Cras semper dolor id mauris porttitor, a sagittis felis mattis. In hac habitasse platea dictumst. Curabitur eu ex vel orci ultrices commodo. Integer ac ligula massa."

news = [{'id':i} for i in range(10)]

for n in news:

id = n['id']

n['title'] = 'Title #'+str(id+1)

n['content'] = lorem[:150]+'...'

context = {'news\_list': news[:5]}

return render(request,'index.html',context)

def single\_news(request,id):

lorem = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris venenatis sem eu neque elementum semper. Aenean a quam enim. Praesent dictum, magna non tincidunt luctus, tortor tellus lobortis lorem, eu sodales felis libero sit amet nulla. Nam rhoncus faucibus fermentum. Sed nec est tellus. Vestibulum vitae volutpat sem, et ullamcorper tortor. Donec venenatis libero vel metus luctus eleifend. Praesent eleifend metus tincidunt, tempus leo a, volutpat dui." \

"Integer mattis cursus ante, non maximus ligula pellentesque eu. Phasellus semper libero ac tortor auctor placerat. Quisque quam ipsum, gravida vitae risus at, fringilla vestibulum nibh. Donec fermentum accumsan velit vel rutrum. In imperdiet, leo nec finibus vehicula, tellus massa vehicula arcu, non feugiat leo ex ac dolor. In sagittis augue quis metus suscipit dapibus. Curabitur ultrices erat malesuada, viverra arcu ornare, accumsan est." \

"Suspendisse dignissim odio at nibh malesuada, sit amet volutpat ante pellentesque. Ut id lorem commodo, mollis sapien sit amet, vestibulum magna. Curabitur a convallis augue, eu hendrerit ipsum. Cras semper dolor id mauris porttitor, a sagittis felis mattis. In hac habitasse platea dictumst. Curabitur eu ex vel orci ultrices commodo. Integer ac ligula massa."

news = [{'id':i} for i in range(10)]

for n in news:

idx = n['id']

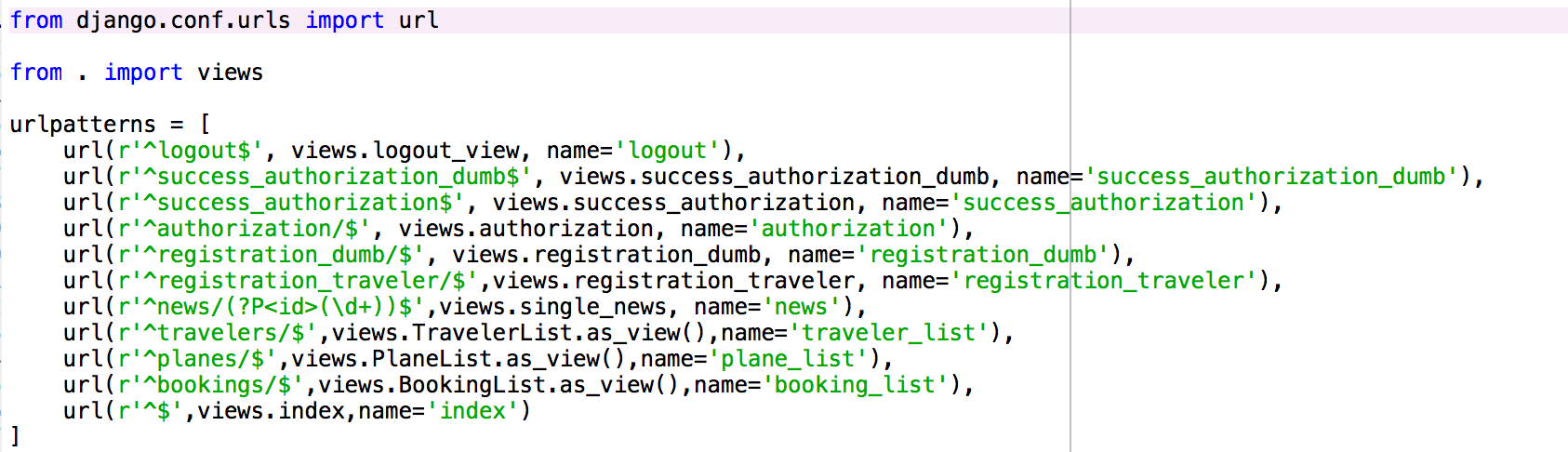
n['title'] = 'Title #'+str(idx+1)

n['content'] = lorem

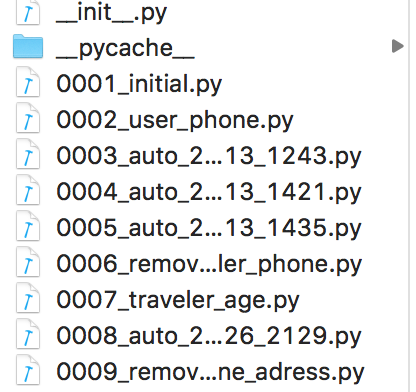
context = {'news': news[int(id)]}

return render(request,'single\_news.html',context)

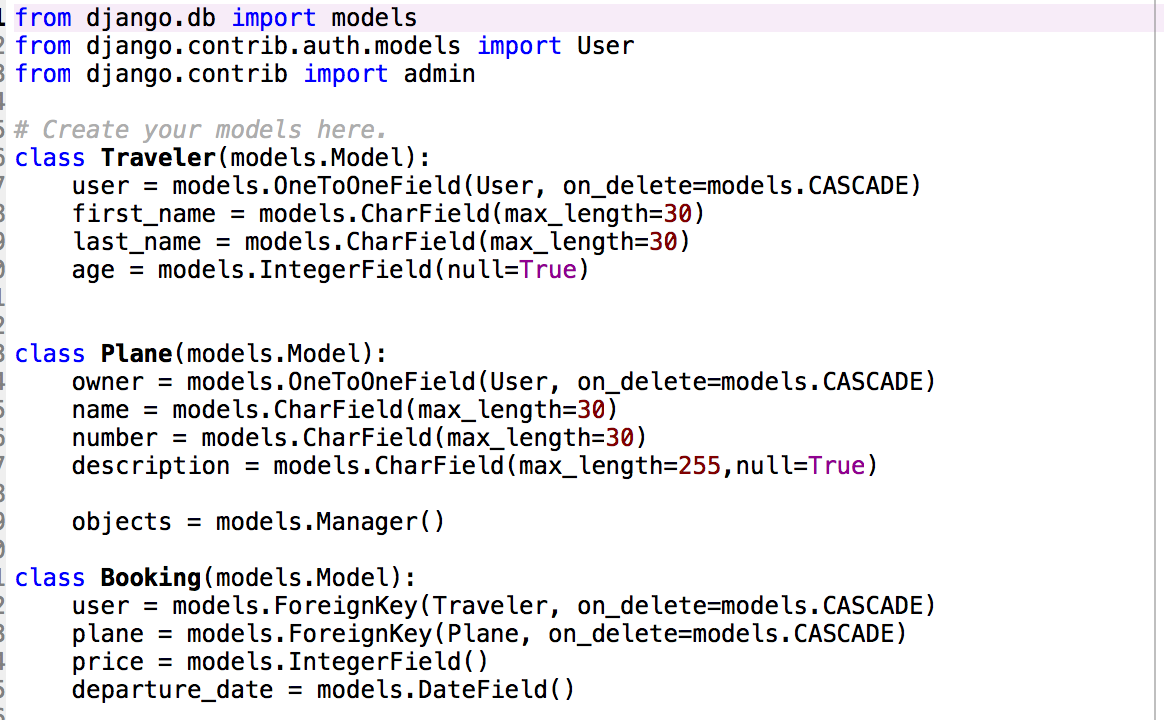
Файл urls.py



migrations



models.py



Файл admin.py



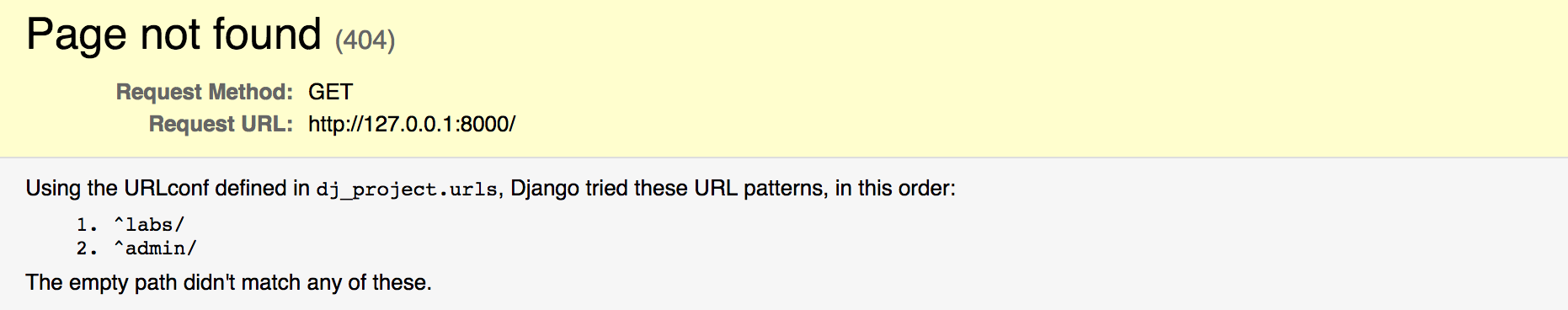
APPS.PY

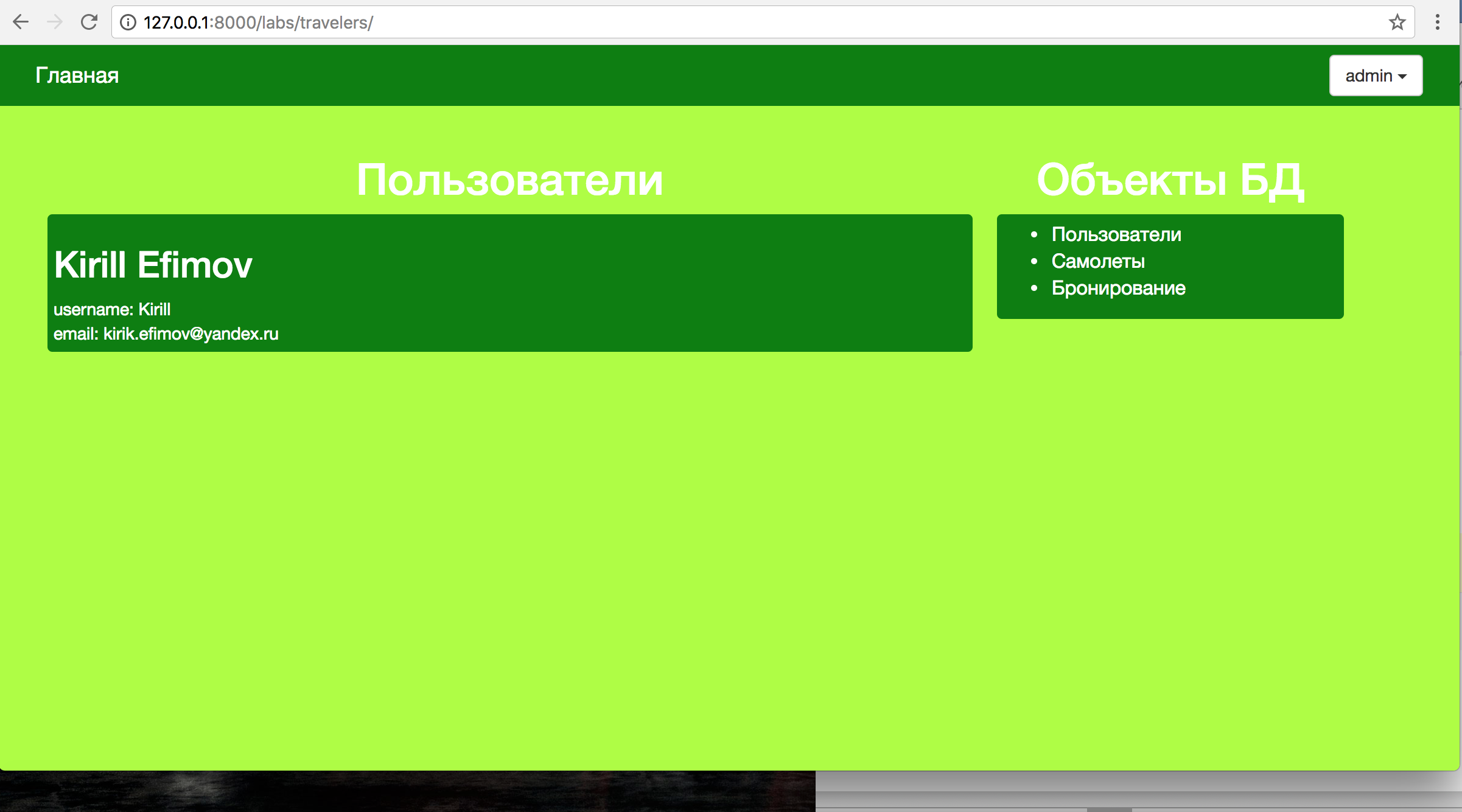
from django.apps import AppConfig

class LabsConfig(AppConfig):

name = 'labs'

Запуск и работа приложения:





Админка

