

# django

Официальный [учебник](#) на английском языке

Официальная [документация](#) на английском языке

# О преподавателе

## Баринова *Вера Олеговна*



- Окончила факультет вычислительной математики и кибернетики Московского Государственного Университета имени М.В. Ломоносова
- Имеет многолетний опыт практической работы программистом, веб-разработчиком, системным и администратором Баз Данных
- Является автором курса Администрирование GNU/Linux в МГУ
- Ведет научную деятельность в области физики плазмы и физики космоса, автор публикаций в журналах, участник международных конференций.

# История и развитие веб-программирования, различные подходы.



Без своего сайта уже давно себя не мыслит ни частное лицо, ни компания или группа компаний. Люди испытывают потребности как в самовыражении, так и в поиске единомышленников и *клиентов*. Сайт «работает за нас» 24/7, а для обеспечения большей надежности требуется не усложнение, А УПРОЩЕНИЕ разработки. И, конечно, ключевую роль здесь играет цена не только разработки, но И ПОДДЕРЖКИ сайта.

# Предупреждение и разъяснения о приводимых здесь "инструкциях"

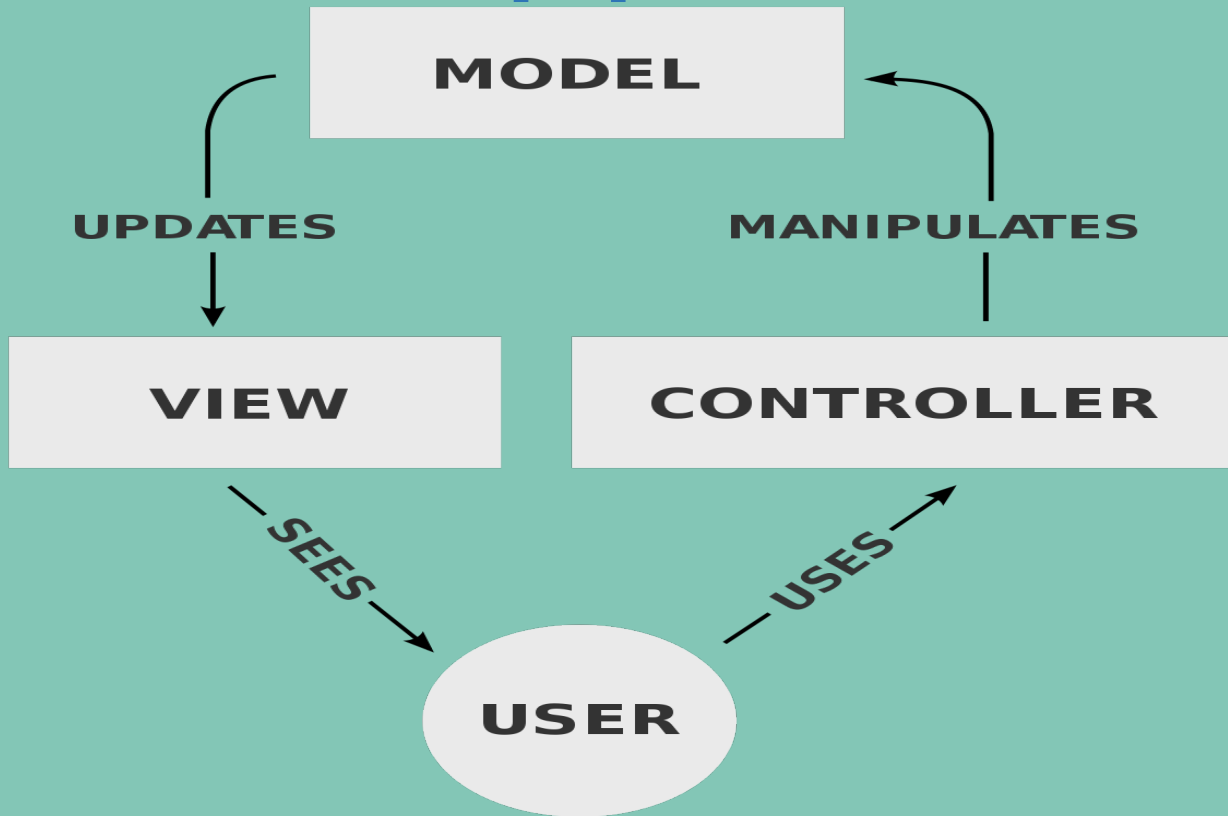
Приходят к физику, просят его чайку организовать. Он ставит чайник, наливает воды, включает газ, дожидается бурления и пузырьков, выключает газ, достает заварку и наливает кипяток.

Приходят с физиком в кухню с кипящим чайником и просят чай организовать. Физик снимает чайник с огня, достает заварку, наливает кипяток.

Приводят математика в кухню с кипящим чайником и просят чай организовать. Математик выключает газ, выливает воду и сообщает, что свёл ситуацию к случаю уже решённой ранее задачи.

*Так вот: здесь нет ИНСТРУКЦИЙ. Перед вами несколько блоков, помогающих осознать, как построить ту или иную связь. Но если Вы оказались в ситуации, когда что-то не работает, убедитесь, что директория та же, что на слайде, включена ли среда, запущен ли веб-сервер и не выдаёт ли он ошибок...*

# MVC-подход: разделение логики, интерфейса и данных.





Что такое MV\* паттерн? *(а что такое паттерн?)*

<https://habr.com/ru/company/mobileup/blog/313538/>

Зачем нужны паттерны проектирования?

MVC



# Почему Framework - не библиотека?

Библиотека - это пакет (набор) функций

Фреймворк включает помимо функций

- структуру проекта
- способы управления не только ПО, но и содержимым (content=>CMS)
- настройки и сопряжение с другими технологиями
- авторизацию
- претендует на решение "под ключ" не отдельной задачи, а целого проекта

Чем отличается "проект" от "задачи"?



# Предполагается, что вы знаете:

- Что такое командная строка, и умеете ей пользоваться
- Язык гипертекстовой разметки HTML
- Язык стилей CSS
- Язык программирования Python
- Систему контроля версий [Git](#) (необязательно, но желательно)

Также предполагается, что Python и Git уже установлены

Но я могу это тоже рассказать в минимальном необходимом объёме



# Операционная система и виртуальная среда

- Операционная система «сама знает», где у нее стоят программы, службы (демоны), что должно работать постоянно, что в каком порядке запускать
- Эти знания хранятся в специальных файлах, откуда загружаются в переменные среды
- Чтобы изменить настройки, нужно менять значения переменных или конфигурационные файлы или создать отдельные виртуальные среды и переключаться между ними
- Также в виртуальной среде могут быть установлены другие версии программ
- **ЗАДАНИЕ:** каковы преимущества и недостатки виртуальной среды?

# Установка и настройка Git и виртуальной среды (Linux)

```
user@computer:~$ sudo apt-get install git virtualenv
```

```
user@computer:~$ git config --global user.name "Wera Barinova"
```

```
user@computer:~$ git config --global user.email alisawera@ya.ru
```

Я использую и некоторые другие, необязательные программы:

```
user@computer:~$ sudo apt-get install vim tree
```

```
user@computer:~$
```



# Установка и настройка Git и виртуальной среды (Windows)

- Скачиваем и устанавливаем из пакета msi
- Открываем командную строку (cmd)

Git должен знать, от чьего имени ведётся разработка

```
C:\Users\Student> git config --global user.name "Wera Barinova"
```

```
C:\Users\Student> git config --global user.email alisawera@ya.ru
```

# Настройка виртуальной среды (Linux)

```
user@computer$ python venv -p python3 py3ve
```

Для активации среды (нужно не всегда!) вводим:

```
user@computer$ source py3ve/bin/activate
```

```
(py3ve) user@computer$
```

Это надо делать каждый раз перед:  
Запуском встроенного веб-сервера! (он запускается в среде)  
и всех команд `./manage.py`

Для деактивации среды:

```
user@computer$ deactivate
```



# Настройка виртуальной среды (Windows)

```
C:\Users\Student> py -m venv py3ve
```

Для активации среды (нужно не всегда!) вводим:

```
C:\Users\Student> C:\Users\Student\py3ve\Scripts\activate.bat
```

Это надо делать каждый раз перед:  
Запуском встроенного веб-сервера! (он запускается в среде)  
и всех команд `./manage.py`

Для деактивации среды:

```
(py3ve) C:\Users\Student> deactivate
```



# Установка Web-фреймворка Django и запуск веб-сервера

```
user@computer:~$ source py3ve/bin/activate
```

```
(py3ve) user@computer:~$ pip install django
```

```
(py3ve) user@computer:~$ django-admin startproject wcity
```

```
(py3ve) user@computer:~$ cd wcity
```

```
(py3ve) user@computer~wcity$ ./manage.py runserver
```

# Установка Web-фреймворка Django и запуск веб-сервера (Windows)

```
(py3ve) C:\Users\Student> pip install django
```

```
(py3ve) C:\Users\Student> django-admin startproject wcity
```

```
(py3ve) C:\Users\Student> cd wcity
```

```
(py3ve) C:\Users\Student\wcity> python manage.py runserver
```

# Структура проекта





```
(py3ve) user@computer:~$ tree
```









```
.  
├── manage.py  
└── wcity  
    ├── asgi.py  
    ├── __init__.py  
    ├── settings.py  
    ├── urls.py  
    └── wsgi.py
```

```
1 directory, 6 files
```



# Встроенный Web-сервер

    Not secure | 0.0.0.0:8000

Bookmarks  Article database  Space Physics  howto  home  people  sinp  Мультитики  conference

## DisallowedHost at /

Invalid HTTP\_HOST header: '0.0.0.0:8000'. You may need to add '0.0.0.0' to ALLOWED\_HOSTS

Request Method: GET  
Request URL: http://0.0.0.0:8000/  
Django Version: 2.1.7  
Exception Type: DisallowedHost  
Exception Value: Invalid HTTP\_HOST header: '0.0.0.0:8000'. You may need to add '0.0.0.0' to ALLOWED\_HOSTS.  
Exception Location: /home/wera/anaconda3/envs/py3django/lib/python3.6/site-packages/django/http/request.py in get\_host, line 106  
Python Executable: /home/wera/anaconda3/envs/py3django/bin/python  
Python Version: 3.6.8  
Python Path: ['/home/wera/wcity',  
'/home/wera/anaconda3/envs/py3django/lib/python3.6.zip',  
'/home/wera/anaconda3/envs/py3django/lib/python3.6',  
'/home/wera/anaconda3/envs/py3django/lib/python3.6/lib-dynload',  
'/home/wera/anaconda3/envs/py3django/lib/python3.6/site-packages']  
Server time: Sat, 13 Apr 2019 12:58:13 +0000

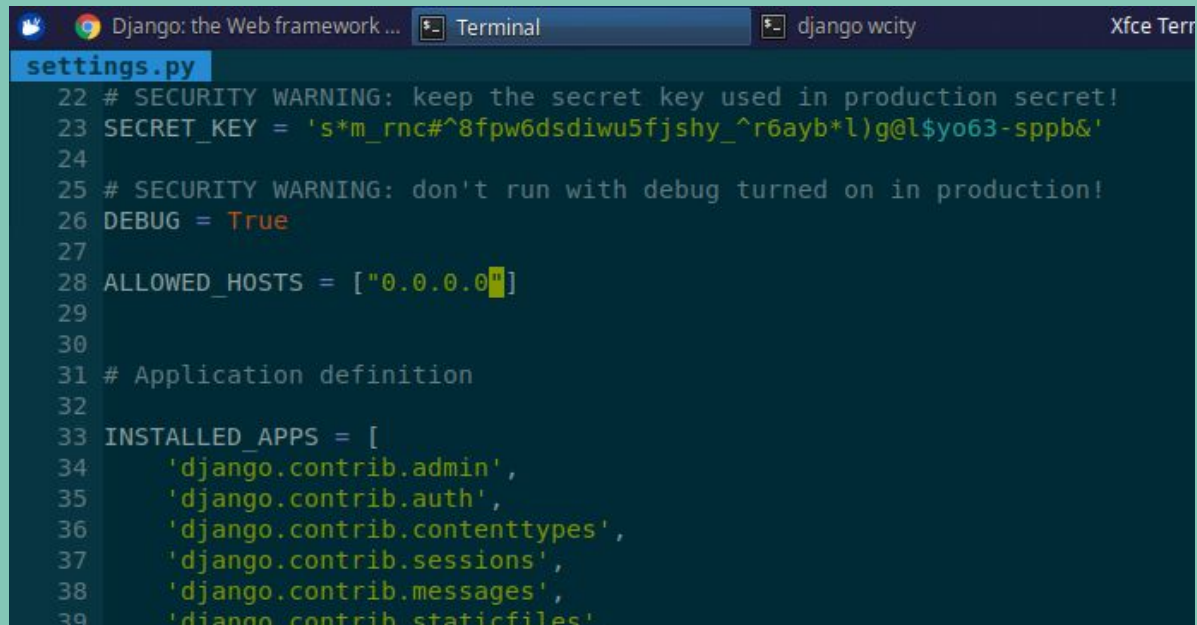
### Traceback [Switch to copy-and-paste view](#)

/home/wera/anaconda3/envs/py3django/lib/python3.6/site-packages/django/core/handlers/exception.py in inner

```
34.         response = get_response(request)
```

► Local vars

# Структура: settings.py



```
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 's*m_rnc#^8fpw6dsdiwu5fjshy_^r6ayb*l)g@l$yo63-sppb&'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = ["0.0.0.0"]
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles']
```

# Структура: settings.py

Предупреждение безопасности требует убрать ключ из settings.py, если мы размещаем его в репозитории

```
from .secret import SECRET_KEY
```

## vim secret.py

```
# SECURITY WARNING: keep the secret key used in production
secret!
SECRET_KEY =
'&*gs^qlx5^vtiahs3ui)ur6@uso+-pqb(+*u-r3*18i-sy9fdk'
```

Веб-сервер перезапустится автоматически и сайт продолжит работать

# Decouple

Хорошей практикой в продакшене, является сокрытие настроек database url, password, secret key, debug status, email host, allowed hosts

```
(py3ve) user@computer:~$ pip install python-decouple
```

vim .psw

```
SECRET_KEY=&*gs^qlx5^vtiahs3ui)ur6@uso+-pqb(+*u-r3*18i-sy9fdk  
DEBUG=True
```

файл wcity/settings.py

```
from decouple import config  
SECRET_KEY = config('SECRET_KEY')  
DEBUG = config('DEBUG', cast=bool)
```

# Текстовый редактор VIM

Внизу отображается состояние: если там нет INSERT, или VISUAL то это режим команд. В этом режиме работают:

:w filename – сохранить в файл с именем

:q – выход

:wq – сохрани и выйди

:wqa – сохранить всё и выйди

:q! – выйди без сохранения

dbd – вырезать 6 строк от текущей вниз (в буфер)

p – вставить после курсора или вниз

ci' ci" ci[ ci( – стереть внутри (скобочек или кавычек) и перейти в режим вставки

i – перейти в режим ввода (вставки)

Если отображается, можно просто набирать текст. Выйти из режима

# Сохранение в репозитории с участием github

```
user@computer$ git clone git@github.com:username/wcity.git
```

```
/* НЕМНОГО МАГИИ С ОДИНАКОВЫМИ ИМЕНАМИ */
```

```
user@computer:~$ cd wcity
```

```
user@computer:~/wcity$ echo __pycache__ >> .gitignore
```

```
user@computer:~/wcity$ vim .gitignore
```

Добавим ещё \*.swp

```
user@computer:~/wcity$ git add .
```

```
user@computer:~/wcity$ git commit -m "Создали новый проект"
```

```
user@computer:~/wcity$ git push
```

# Домашнее задание 1

- Установить к себе виртуальную машину
- Установить в нее xubuntu
- Установить виртуальную среду
- Установить git
- Установить django
- Склонировать проект из репозитория
- Запустить сайт
- Открыть браузер и в нем 0.0.0.0:8000
- *\*\*Запустить на другом порту и другом IP*

Принцип клиент-серверного взаимодействия, схема-путь запроса и ответа

GET и POST запросы

тело запроса, переменные

респонз: статус, стандартные номера статусов (200, 404, 503...)

Автоматическое формирование "обвязки" запроса

6. написание теста ре퀘стов и респонзов "ручное": плохо тем и этим

встроенный клиент - нет обращения к тестовому серверу, эмуляция запроса

анализ запроса



# Тестирование и TDD

```
from django.test import TestCase
```

```
from django.test import Client
```

```
# Create your tests here.
```

```
from django.http import HttpResponse
```

```
from django.http import HttpRequest
```

```
from .views import index
```

# Тестирование и TDD

```
class PollTestCase(TestCase):  
    def test_index(self):  
        self.assertEqual(  

```

```
# не работает, потому что строки b''  
        index(HttpRequest()),  
        HttpResponse("<b>Kuku</b>")  
    )
```

# Тестирование и TDD

```
class PollTestCase(TestCase):  
  
    def test_index(self):  
  
        c = Client() # TODO: перенести в функцию setUp  
  
        response = c.get('/polls/')#'/customer/details/')  
  
        self.assertEqual(response.status_code, 200)  
  
        self.assertEqual(response.content,b'<b>Kuku</b>')
```

# Запуск теста (в среде!)

```
(django_bot) C:\Users\Student\project>coverage run --source='.' manage.py test
```

```
Found 1 test(s).
```

```
Creating test database for alias 'default'...
```

```
System check identified no issues (0 silenced).
```

```
..
```

```
-----  
Ran 2 tests in 0.019s
```

```
OK
```

```
Destroying test database for alias 'default'...
```

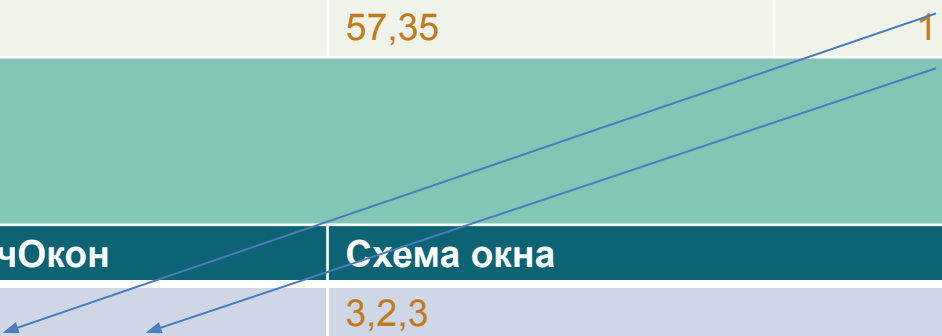
# Способы хранения данных

- В переменных
- В файлах (текстовых, бинарных)
- В базе данных
- В облаке (чужая бд)

# Схема БД: связи и ключи

КлючКвартиры	Адрес квартиры	Тип окон ( <i>внешний ключ</i> )
1	55,37	1
2	57,35	1

КлючОкон	Схема окна	
1	3,2,3	
2	1,2	



Установка PostgreSQL под Windows  
[рассмотрена в отдельной презентации](#)

# Администрирование БД



```
user@computer$ sudo apt-get install postgresql
user@computer$ sudo su - postgres
postgres@computer$ psql
```

Пароль  
приду-  
мывайте  
сами

```
postgres=# CREATE USER king WITH PASSWORD 'queen';
CREATE ROLE
postgres=# ALTER USER king CREATEDB;
```

**Покидаем консоль базы и юзера postgres: Ctrl+d, ctrl+d**

```
(py3ve) postgres@computer$ pip install psycopg2
```

```
sudo apt-get install libpq-dev
```

```
sudo apt-get install python3-dev
```

```
Postgresql порт по умолчанию 5432 \ pip install psycopg2-binary
```

# Подружить Django и БД

## файл `/home/student/.pgpass`

`127.0.0.1:5432:wcity:king:ПАРОЛЬ`



```
user@computer$ chmod go-rw ~/.pgpass
user@computer$ psql wcity king -h 127.0.0.1
```

`wcity=>`

Мы убедились, что файл `.pgpass` сформирован верно, ему выданы верные права и база не спрашивает пароль при попытке входа из-под пользователя `king`

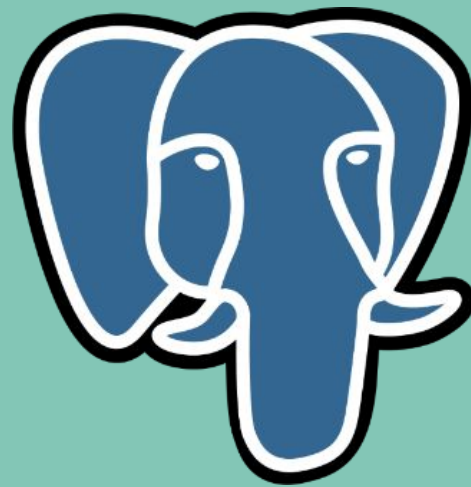
Взять из распространяемых файлов `pgpass_helper.py`

[https://gist.githubusercontent.com/barseghyanartur/a6946cfad9bc03c67de3/raw/593f53e86b091a994b1471d26f72c481831943e1/pgpass\\_helper.py](https://gist.githubusercontent.com/barseghyanartur/a6946cfad9bc03c67de3/raw/593f53e86b091a994b1471d26f72c481831943e1/pgpass_helper.py)



# Подружить Django и БД

## файл `wcity/settings.py`



```
# Database
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases

+ from pgpass_helper import read_pgpass
+
DATABASES = {
-     'default': {
-         'ENGINE': 'django.db.backends.sqlite3',
-         'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
-     }
+     'default': read_pgpass('wcity'),
+     # 'default': {
+         # 'ENGINE': 'django.db.backends.sqlite3',
+         # 'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
+     # }
}
```

<https://gist.github.com/barseghyanartur/a6946cfad9bc03c67de3>

# Подружить Django и БД



```
user@computer$ source py3ve/bin/activate
(py3ve) user@computer$ cd wcity
(py3ve) user@computer:wcity$ ./manage.py dbshell
```

```
wcity=> \dt
```

public		auth_group		table		king
public		auth_group_permissions		table		king
public		auth_permission		table		king

```
.....
```

public		django_content_type		table		king
public		django_migrations		table		king
public		django_session		table		king

```
+ wcity=>
```

# Сохранение и восстановление БД



С некоторой периодичностью, лучше скриптом следует выполнять полный dump базы данных (содержимого)

```
(py3ve) user@computer$ ./manage.py dumpdata > dbdump20191210.json
```

В случае аварии выполняются все миграции в той последовательности, что были в проекте

```
(py3ve) user@computer$ ./manage.py migrate
```

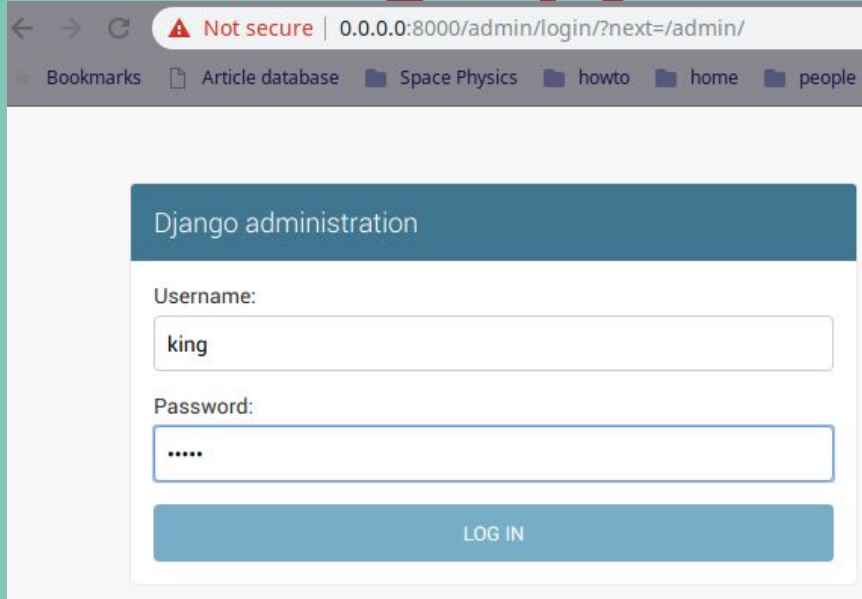
```
(py3ve) user@computer$ ./manage.py loaddata dbdump20191210.json
```

```
Installed 30 object(s) from 1 fixture(s)
```

```
(py3ve) user@computer$
```

# Пользователи

**./manage.py createsuperuser**



A screenshot of a web browser showing the Django administration login page. The browser's address bar displays the URL `0.0.0.0:8000/admin/login/?next=/admin/` with a "Not secure" warning. The browser's bookmark bar shows "Article database", "Space Physics", "howto", "home", and "people". The page itself has a dark blue header with the text "Django administration". Below the header, there is a "Username:" label followed by a text input field containing the text "king". Below that is a "Password:" label followed by a password input field with five dots. At the bottom of the form is a blue button labeled "LOG IN".

<http://0.0.0.0:8000/admin/>

# Пользователи

The screenshot shows the Django administration interface in a web browser. The address bar indicates the URL is 0.0.0.0:8000/admin/. The page title is 'Django administration' and it says 'WELCOME, KING.' with links for 'VIEW SITE', 'CHANGE PASSWORD', and 'LOG OUT'. The main section is 'Site administration' with a sub-section 'AUTHENTICATION AND AUTHORIZATION'. Under this sub-section, there are two items: 'Groups' and 'Users'. Each item has a '+ Add' link and a 'Change' link with a pencil icon. On the right side, there is a sidebar with 'Recent actions' and 'My actions' sections, both of which are currently empty.

← → ↺ ⓘ Not secure | 0.0.0.0:8000/admin/

★ Bookmarks 📄 Article database 📁 Space Physics 📁 howto 📁 home 📁 people 📁 sinp

## Django administration

WELCOME, **KING**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

### Site administration

#### AUTHENTICATION AND AUTHORIZATION

Groups	<a href="#">+ Add</a>	<a href="#">Change</a>
Users	<a href="#">+ Add</a>	<a href="#">Change</a>

#### Recent actions

---

#### My actions

None available

# Приложение

```
(py3ve) user@computer:~/wcity$ ./manage.py startapp fenster
```

## fenster/views.py

```
from django.shortcuts import render
```

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

```
from django.http import HttpResponse
```

```
def index(request):  
    return HttpResponse(  
        '<h1>Я буду приложением<br/> для окошек</h1>'  
    )
```

# Приложение

## fenster/urls.py

```
from django.urls import path

from . import views

urlpatterns = [
    path('', views.index)
]
```

## wcity/urls.py

```
from django.urls import include

urlpatterns = [
    path('', admin.site.urls),
    path('admin/', admin.site.urls),
    path('fenster/', include('fenster.urls')),
]
```

# Починить ракету

## wcity/urls.py

```
from django.urls import include
from django.views.debug import default_urlconf
```

```
urlpatterns = [
    path('', admin.site.urls),
    path('django', default_urlconf),
    path('admin/', admin.site.urls),
    path('fenster/', include('fenster.urls')),
]
```

```
,
```



# Домашнее задание 2

- Установить postgresql server
- Создать базу данных
- Создать пользователя
- Предоставить полный перечень прав этому пользователю на эту бд
- *\*\* Создать новую таблицу – для хранения координат и параметров квартир*
- *\*\* Настроить синхронизацию баз или перенос данных между домом и классом*

# Схема БД: таблицы и модели

```
class Таблица1(models.Model):  
    Столбец1 = models.fieldType(параметры)
```

Таблица1

Ключ	Столбец1	Столбец2
1	Знач1	Val1
2	Знач2	Val2
3	Знач3	Val3

Первичный целочисленный  
автоувеличивающийся **ключ**  
Появляется автоматически

Для остальных полей указываем тип  
данных и иногда – значения по умолчанию

<https://docs.djangoproject.com/en/4.0/ref/models/fields/#charfield>

# Типы полей

- AutoField тип, подходящий для автоувеличивающихся первичных ключей.
- BooleanField - для логических значений
- CharField - для небольших строк (названия, имена, фамилии, адреса и т.д.)
- DateTimeField - для даты и времени
- IntegerField - для целых чисел
- FloatField - для вещественных чисел

# Схема БД: таблицы и модели

Создадим класс для будущей таблицы, в которой каждая строка - окно

*fenster/models.py*

```
class Fenster(models.Model):  
    fenster_height = models.IntegerField()  
  
    # позже:  
    window_view = models.CharField(default='', max_length=1024)
```

Ключ	ширина fenster_width	вид fenster_view
1		
2		
3		

Таблицы в базе ещё нет, есть только её описание на языке классов Python. Необходимы действия, которые «переведут» с языка Python на язык SQL наше пожелание создать таблицу Окошко.

# Отображение данных в БД на объекты приложения (ORM)

Подготовим и применим изменения к базе данных. Сейчас в ней только таблицы и поля системы и администрирования.

*fenster/models.py*

```
class RowTypes(models.Model):  
    scheme = models.CharField(max_length=256)
```

```
user@computer:~wcity$ ./manage.py makemigrations fenster
```

```
Migrations for 'fenster':  
  fenster/migrations/0001_initial.py  
    - Create model Fenster  
    - Add field fenster to apt
```

```
user@computer:~wcity$ ./manage.py migrate
```

fenster\_fenster

Ключ	ширина fenster_height	вид fenster_view
1	130	речка.jpg
2	300	парк.jpg
3	150	арена.jpg

# Таблицы в базе данных

Убедимся в наличии таблицы в БД с помощью консоли СУБД

## fenster\_fenster



```
user@computer:~wcity$ sudo su - postgres
postgres@computer:~$ psql
psql (10.10 (Ubuntu 10.10-0ubuntu0.18.04.1))
Type "help" for help.
```

```
postgres=# \c wcity
```

You are now connected to database "wcity" as user "postgres".

```
wcity=# \d Fenster_fenster
```

```
Table "public.Fenster_fenster"
  Column      | Type   | Collation | Nullable |
Default
-----+-----+-----+-----+
id             | integer |           | not null | nextval(...)
window_height  | integer |           | not null |
```

Indexes:

"Fenster\_fenster\_pkey" PRIMARY KEY, btree (id)

# Получение данных из базы - примитивный запрос

Получим данные из базы данных. Отообразим их на сайт.

*fenster/views.py*

```
from .models import RowTypes

def index(request):
    # RowTypes - это наша таблица моделей окошек
    # Запрос позволяет выбрать из базы все окошки
    rows = RowTypes.objects.all()
    # это список всех объектов!
    one_window_scheme = eval(rows[0].scheme)
    .....
    context['new_height'] = int(one_window_scheme.keys()[0])
```

# Удаление объекта из базы по id

*fenster/views.py*

Взять ("найти") объект в базе и выполнить delete()

```
RowType.objects.filter(id=3).delete()
```

Простейшие запросы к базе из консоли СУБД

```
select * from fenster_rowtype; получить всё из таблицы на экран
```

Если хотите, можно вставить окно через консоль СУБД

```
insert into fenster_rowtype (id, scheme) values (3, '120:80');
```

```
delete from fenster_rowtype where id=1;
```



# HTML

HTML - это язык разметки. Он структурирует текст и картинки на странице

```
<!-- ТЭГИ HTML (краткий экскурс) -->
<h1>Заголовок</h1>
<table> <!-- Таблица -->
  <tr> <!-- строка (ряд) таблицы -->
    <td> <!-- Ячейка. Тут можно что-нибудь написать -->
    </td>
  </tr>
</table>
<input type="radio" name="var1name" id="var1id" />
  <label for="var1id">Кликабельная строка для «кружочка»</label>
<input type="checkbox" name="var2name" id="var2id" />
  <label for="var2id">Кликабельная строка для «галочки»</label>
<select name="var3name">
  <option value="55">Пятьдесят пять</option>
</select>
*/
```

# Шаблон

```
user@computer:~wcity$ mkdir fenster/templates/fenster/ -p
```

```
user@computer:~wcity$ vim fenster/templates/fenster/index.html
```

```
user@computer:~wcity$ git add $1
```

*# первый аргумент предыдущей команды*

Шаблон - это HTML, который содержит переменные для подстановки - динамической генерации страниц.

<http://0.0.0.0:8000/fenster/>

```
<h1>Здесь будет окошко</h1>
<table border="1">
  <tr>
    <td colspan="2"
      width="100"
      height="50">
    </td>
  </tr>
  <tr>
    <td height="50"></td>
    <td></td>
  </tr>
</table>
```

# Шаблон

## fenster/views.py

```
from django.shortcuts import render

def index(request):
    context = {}
    return render(
        request,                # Запрос
        'fenster/index.html',   # путь к шаблону
        context                  # подстановки
    )
```

## wcity/settings.py

```
INSTALLED_APPS = [
    .....
    'fenster',
]
```

<http://0.0.0.0:8000/fenster/>

# Шаблон

git diff --staged

```
+ $ git diff --staged
diff --git a/fenster/views.py b/fenster/views.py
index a7d999b..a60cdb6 100644
--- a/fenster/views.py
+++ b/fenster/views.py
@@a296t+2,9a@@
from django.shortcuts import render
index.html
[13/Apr/2019 17:14:42] "GET /fenster/ HTTP/1.1" 500 74205
^#(Create your views here.16:20:22> <wera@anarinya:~/wcity [master
+from django.http import HttpResponse
+from django.template import loader

def index(request):
    if not hasattr(request, 'messages'):
        return HttpResponse("Das fenster is 'the window' in German")
    +p = loader.get_template('fenster/index.html')
    +u = template.render(context)
    +r = HttpResponse(u)
    +r.render()
    +r.close()
    return r

diff --git a/wcity/settings.py b/wcity/settings.py
index c469807..d7faf33 100644
--- a/wcity/settings.py
+++ b/wcity/settings.py
@@4737,62+37,77@@
INSTALLED_APPS = [
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'fenster',
]

You have 15 unapplied migration(s). Your project may not work properly.
Run 'python manage.py migrate' to apply them.

MIDDLEWARE = [
```

<http://0.0.0.0:8000/fenster/>

# Передача данных из приложения в шаблоны

views.py

```
1 from django.shortcuts import render
2
3 # Create your views here.
4 from django.http import HttpResponse
5 from django.template import loader
6
7 def index(request):
8     template = loader.get_template('fenster/index.html')
9     context = {
10         "window_height": 100,
11         "window_width": 50,
12     }
13     return HttpResponse(template.render(context))
```

{{var\_name}}

t/f/index.html

```
1 <h1>Das Fenster</h1>
2
3 <table border="6">
4     <tr height="50px">
5         <td colspan="2">
6             &nbsp;
7         </td>
8     </tr>
9     <tr height="{{window_height}}px">
10         <td width="50px">
11             &nbsp;
12         </td>
13         <td width="50px">
14             &nbsp;
15         </td>
16     </tr>
17 </table>
```



# Выбор в шаблонах

```
1 <h1>Das Fenster</h1>
2
3 <table border="6">
4   {% if countryside %}
5     <tr height="50px">
6       <td colspan="2">
7         &nbsp;
8       </td>
9     </tr>
10    {% endif %}
11    <tr height="{{window_height}}px">
12      <td width="50px">
13        &nbsp;
14      </td>
15      <td width="50px">
16        &nbsp;
17      </td>
18    </tr>
19  </table>
```

# Перебор в шаблонах

```
1 <h1>Das Fenster</h1>
2
3 {% for countryside in fenstertypes %}
4   <table border="6">
5     {% if countryside %}
6       <tr height="50px">
7         <td colspan="2">
8           &nbsp;
9         </td>
10      </tr>
11    {% endif %}
12    <tr height="{{window_height}}px">
13      <td width="50px">
14        &nbsp;
15      </td>
16      <td width="50px">
17        &nbsp;
18      </td>
19    </tr>
20  </table>
21 {% endfor %}
```

Вопрос: что мы  
*забыли* исправить?

# Удобные, но неочевидные замены в шаблонах

Точка. `{{ foo.bar }}`

Точка в Django обрабатывается в трёх случаях  
(в указанном порядке):

Сначала проверяется вариант "ключ словаря": `foo["bar"]`

Затем - атрибут экземпляра класса: `foo.bar`

Наконец, индекс (массива, кортежа, строки): `foo[bar]`

Т.е., чтоб получить элемент с номером 5 пишем `foo.5`

Счётчик цикла. `{% for elem in iterable %}` их два:

`forloop.counter` - номер витка, начиная с единицы

`forloop.counter0` - номер витка, начиная с нуля



# Шаблон

git status -uno

```
+ $ git status
On branch master
Your branch is up to date with 'origin/master'.
Changes to be committed:
  fenster/index.html
  fenster/te
  te (use "git reset HEAD <file>..." to unstage)
+ $ cp fenster/templates/fenster/index.html
+ $ rm fenster/modified: wcity/settings.py
  templates/ tests.py
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  /bin/rm: remove regular file 'fenster/templates/index.html'? y
  (py3dja central-fed-district-latest-osm@anarinya:~/wcity [master
+ $ vim wcity/settings.py
<Sun Apr 14 11:00:42> <wera@anarinya:~/wcity [master<!?+>] master
```

<http://0.0.0.0:8000/fenster/>

# Домашнее задание 3

- Создать новое приложение freeaddr  
(будем хранить и показывать доступные для аренды адреса)
- Подключить приложение, отобразить в браузере
- *Заполнить в приложении в html различные атрибуты из БД*

# Улучшим `fenster/models.py`

## *`fenster/models.py`*

```
from django.db import models
from django.core.validators import RegexValidator

class RowType(models.Model):
    scheme = models.CharField(
        validators=[
            RegexValidator('\{(\d+:\[\d+(,\d+)*\],)*\}', ),
        ],
        max_length=256)
```

Теперь у нас есть проверять, который не даст ввести схему, не подходящую под наши задачи. Повторите Регулярные Выражения

# Создание данных в БД из приложения

Данные в базу можно добавлять

1. из консоли базы
2. из панели администратора
3. из приложений

# Создание данных в БД из приложения

*fenster/views.py*

```
27 def add(request):
28     f = Fenster(    # создали объект
29         window_height=328
30     )
31     f.save()    # сохранили в базе
32     return HttpResponse('<p>Получили и создали окно: %s</p>' % (
33         request.GET['newheight']
34     ))
```

*fenster/urls.py*

```
# добавляем второй путь (после первого запятая – это же список!)
path('add', views.add, name='add')
```

# Взаимодействие с пользователем

The screenshot shows a web browser window with the address bar displaying "Not secure | 0.0.0.0:8000/fenster/". The browser's bookmarks bar includes "Article database", "Space Physics", and "Other bookmarks".

The web page content includes the title "Das Fenster" and the text "Всего: 2". Below this, there are two form elements, each containing a table with dimensions and a "Buy" button.

The first form element contains the following text:

Ширина:201
Высота:100
Схема:1,2
<input type="radio"/> fenster №4
<input type="text"/>
<input type="text"/>

The second form element contains the following text:

Ширина:164
Высота:50
Схема:1,2
<input checked="" type="radio"/> fenster №15
<input type="text"/>
<input type="text"/>

Below the second form is a "Buy" button.

The browser's developer tools are open, showing the HTML structure. The selected element is a form with the following HTML code:

```
<html>
  <head>...</head>
  <body>
    <h1>Das Fenster</h1>
    ...
    <form action="/fenster/" method="post"> == $0
      <input type="hidden" name="csrfmiddlewaretoken" value="
        "XjMNXUZUGJepep9GMjWawQq6CcY4GiBgAHkT2EI89ia3hf30YipznurI7QMexrDy">
      <h2>Bcero: 2</h2>
      <table border="6">...</table>
      <table border="6">...</table>
      <input type="submit" value="Buy">
    </form>
  </body>
</html>
```

The developer tools also show the CSS styles for the selected element, including the "margin" and "border" properties.

# Взаимодействие с пользователем

файл `fenster/urls.py`

```
path('', views.index, name='buy'),  
#path('<int:fenster_id>/', views.buy, name='buy'),  
# по 'name' можно вызвать функцию в  
# шаблоне по тэгу {% url %}
```

# POST и GET

Файл `fenster/views.py`

```
if request.method == "POST":  
    if 'selected_fenster' in request.POST:  
        fenster_id = request.POST['selected_fenster']
```



# Формы запросов

файл `fenster/templates/fenster/index.html`

```
<form action="{% url 'buy' %}" method="post">
{% csrf_token %}  <!-- подставится при использовании render -->
    <input
        type="radio"
        name="selected_fenster"
        id="fenster{{afenster.id}}"
        value={{afenster.id}}
        checked="checked"
    />
    <label for=fenster{{afenster.id}}>
        fenster №{{afenster.id}}
    </label>
<input type="submit" value="Buy">
</form>
```

# Создание данных в БД с сайта

Данные в базу можно добавлять из консоли базы, из панели администратора и из приложений

файл `fenster/views.py`

```
27 def add(request):
28     f = RowType(
29         window_height=int(request.GET['newheight'])
30     )
31     f.save()
32     return HttpResponseRedirect(
33         '<p>Получили и создали окно: %s</p>' % (
34             request.GET['newheight']
35         ))
```

# Выборка и изменение объектов

views.py+

```
1 from django.http import HttpResponseRedirect
2 from django.shortcuts import render
3 from .models import Fenster
4
5 def index(request):
6     return display_all(request)
7
8 def buy(request):
9     if request.method == "POST":
10         if 'selected_fenster' in request.POST:
11             f = Fenster.objects.get(
12                 pk=request.POST['selected_fenster']
13             )
14             f.for_rent=False
15             f.save()
16         return HttpResponseRedirect("/fenster") # relative to 127.0.0.1
17
18 def display_all(request, context={}):
19     fenster_list = Fenster.objects.filter(for_rent=True).order_by("id")
20     context["fenster_list"] = fenster_list
21     return render(request, 'fenster/index.html', context)
22
```

# Домашнее задание 4

- Освоить различные input в HTML, предлагая пользователю выбрать из выпадающего списка адрес, а в переменной храня координаты объекта, с помощью checkbox указывать «только с фото»
- Выпадающий список должен создаваться из полученного словаря из views.py
- <https://docs.djangoproject.com/en/2.2/ref/templates/builtins/>
- *\*\*Поработать с другими встроенными операторами*
- Добавить цены
- Выбирать только окна с видом/без в зависимости от «галочки»
- Добавить таблицу покупок
- Выводить после покупки общую сумму дохода )))
- *\*\* Разобраться с передачей параметров через адрес и redirect*

# Пользователи

Django administration

WELCOME, **KING** [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Authentication and Authorization > Users > Add user

Add user

First, enter a username and password. Then, you'll be able to edit more user options.

Username:

Required. 150 characters or fewer. Letters, digits and @/./+/-/\_ only.

Password:

Your password can't be too similar to your other personal information.  
Your password must contain at least 8 characters.  
Your password can't be a commonly used password.  
Your password can't be entirely numeric.

Password confirmation:

Enter the same password as before, for verification.

Save and add another

Save and continue editing

SAVE

# Привилегии

← → ↻ ⓘ Not secure | 0.0.0.0:8000/admin/auth/user/2/change/

🔖 Bookmarks 📄 Article database 📁 Space Physics 📁 howto 📁 home 📁 people 📁 sinp 📁 Мультитки 📁 conference

Choose all ⓘ Remove all ⓘ

The groups this user belongs to. A user will get all permissions granted to each of their groups. Hold down "Control", or "Command" on a Mac, to select more than one.

User permissions:

Available user permissions ⓘ

- admin | log entry | Can add log entry
- admin | log entry | Can change log entry
- admin | log entry | Can delete log entry
- admin | log entry | Can view log entry
- auth | permission | Can add permission
- auth | permission | Can change permission
- auth | permission | Can delete permission
- auth | permission | Can view permission
- auth | user | Can add user
- auth | user | Can change user
- auth | user | Can delete user
- auth | user | Can view user

Choose all ⓘ

Chosen user permissions ⓘ

- fenster | fenster | Can add fenster
- fenster | fenster | Can delete fenster
- fenster | fenster | Can change fenster
- fenster | fenster | Can view fenster
- fenster | apt | Can add apt
- fenster | apt | Can change apt
- Choose | apt | Can delete apt
- fenster | apt | Can view apt

Remove all ⓘ

Specific permissions for this user. Hold down "Control", or "Command" on a Mac, to select more than one.

# Домашнее задание 5

- Добавить пользователя-контентщика
- Дать возможность контентщику добавлять окошки на продажу
- *\*\* Сделать форму для внешнего пользователя, без регистрации с возможностью предложить окошко и контентщику – его выставить*

# Редактирование бд через

<http://127.0.0.1:8000/admin/>

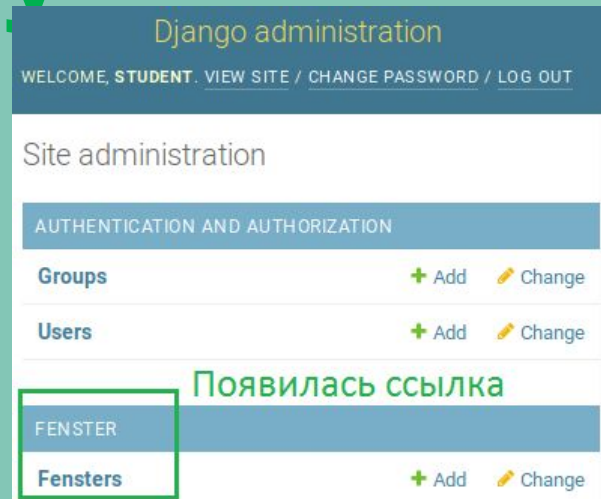
**fenster/admin.py**

```
from django.contrib import admin

# Register your models here.

from .models import Fenster
admin.site.register(Fenster)
```

Когда мы произведём эти изменения, на сайте появится ссылочка для добавления объектов окон





# Редактирование БД через admin

## fenster/admin.py

Если не хочется отдельно  
прописывать каждую...

```
from django.contrib import admin
import django.db.models
import fenster.models
for cls in fenster.models.__dict__:
    try:
        admin.site.register(eval('fenster.models.' + cls))
    except:
        pass # то, что не удалось зарегистрировать
#issubclass(fenster.models.__dict__['RowType'], django.db.models.Model)
#class Smth: pass
for cls in fenster.models.__dict__:
    smth = fenster.models.__dict__[cls]
    if type(smth) == type(Smth) and issubclass(
        smth,
        django.db.models.Model):
```

# Усложнили модель окна

*Не забываем про миграцию!*

```
NORMAL SPELL [EN_US/DE_DE/RU_RU] | master <er/admin.py python utf-8
1 from django.db import models
2 from django.core.validators import int_list_validator
3
4 # Create your models here.
5
6 class Fenster(models.Model):
7     fenster_width = models.IntegerField(default=50)
8     fenster_height = models.IntegerField(default=50)
9     fenster_scheme = models.CharField(
10         validators=[int_list_validator],
11         default='1,2',
12         max_length=1024
13     )
14     fenster_price = models.DecimalField(
15         max_digits=10,
16         decimal_places=2,
17         default=1
18     )
19     window_view = models.CharField(default='', max_length=1024)
20
21 ~/wcity/fenster/models.py
```

# Редактируем БД средствами

← → ↻ http://127.0.0.1:8000/admin/

Bookmarks Article database Space Physics howto home people

**Django administration** WELCOME, **KING**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Fenster > Fensters

Select fenster to change ADD FENSTER +

Action:   0 of 2 selected

<input type="checkbox"/>	FENSTER
<input type="checkbox"/>	Fenster object (15)
<input type="checkbox"/>	Fenster object (4)

2 fensters

# Шаблон добавления окна

*Создание шаблона ни к чему не обязывает*

add.html

```
1 <h1>New Fenster</h1>
2
3 <form action="{% url 'add' %}" method="post">
4 {% csrf_token %}
5 Ширина: <input
6     type="text"
7     name="fenster_width"
8     value="50"
9     id="fenster_width"
10 /><br />
11 Высота: <input
12
13
29 <input type="submit" value="Выставить на продажу" />
30 </form>
```

# Функция добавления окна

*Тоже ни к чему не обязывает!*

views.py

```
1 from django.shortcuts import render
2 from django.http import HttpResponseRedirect
3 from django.contrib.auth.decorators import login_required
4 from .models import Fenster
5
6
7 @login_required
8 def add(request):
9     if request.method == "POST":
10         f = Fenster(
11             fenster_width=request.POST["fenster_width"],
12             fenster_height=request.POST["fenster_height"],
13             fenster_scheme=request.POST["fenster_scheme"],
14             fenster_price=request.POST["fenster_price"]
15         )
16         f.save()
17         return render(request, 'fenster/add.html', context)
```

# Документация offline + собственные добавки

```
(py3ve) user@computer$ pip install docutils
```

ФАЙЛ `wcity/settings.py`

```
INSTALLED_APPS = [
```

```
.....
```

```
+     'django.contrib.admindocs',
```

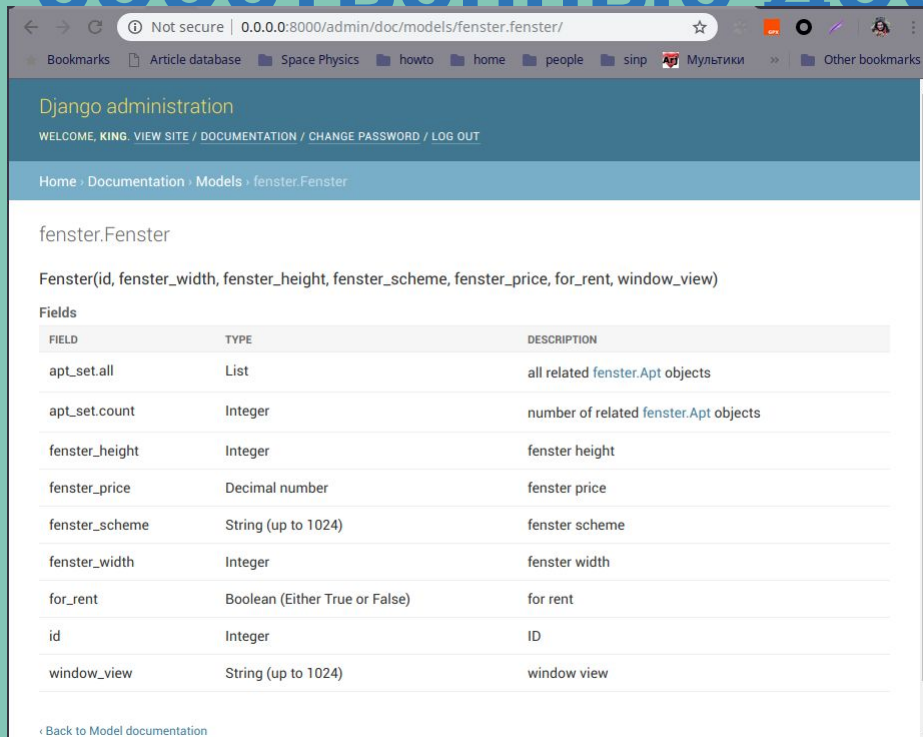
```
.....
```

```
]
```

Файл `wcity/urls.py`

```
path('admin/doc/', include('django.contrib.admindocs.urls')),  
path('admin/', admin.site.urls),
```

# Документация offline + собственные добавки



The screenshot shows a web browser window displaying the Django administration interface. The address bar shows the URL `0.0.0.0:8000/admin/doc/models/fenster.fenster/`. The page title is "Django administration". Below the title, there is a navigation bar with links: "WELCOME, KING", "VIEW SITE", "DOCUMENTATION", "CHANGE PASSWORD", and "LOG OUT". The breadcrumb trail is "Home > Documentation > Models > fenster.Fenster". The main content area shows the model name "fenster.Fenster" and its fields: "Fenster(id, fenster\_width, fenster\_height, fenster\_scheme, fenster\_price, for\_rent, window\_view)". Below this, there is a table titled "Fields" with three columns: "FIELD", "TYPE", and "DESCRIPTION". The table lists the following fields:

FIELD	TYPE	DESCRIPTION
apt_set.all	List	all related <a href="#">fenster.Apt</a> objects
apt_set.count	Integer	number of related <a href="#">fenster.Apt</a> objects
fenster_height	Integer	fenster height
fenster_price	Decimal number	fenster price
fenster_scheme	String (up to 1024)	fenster scheme
fenster_width	Integer	fenster width
for_rent	Boolean (Either True or False)	for rent
id	Integer	ID
window_view	String (up to 1024)	window view

At the bottom of the page, there is a link: "Back to Model documentation".

## Отделение статического контента от динамического

```
user@computer:~wcity/$ mkdir static
```

```
STATIC_URL = '/static/'  
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, 'static')]
```

ФАЙЛ wcity/settings.py

```
function cry() {  
    alert('WoW!');  
};
```

ФАЙЛ static/alert.js

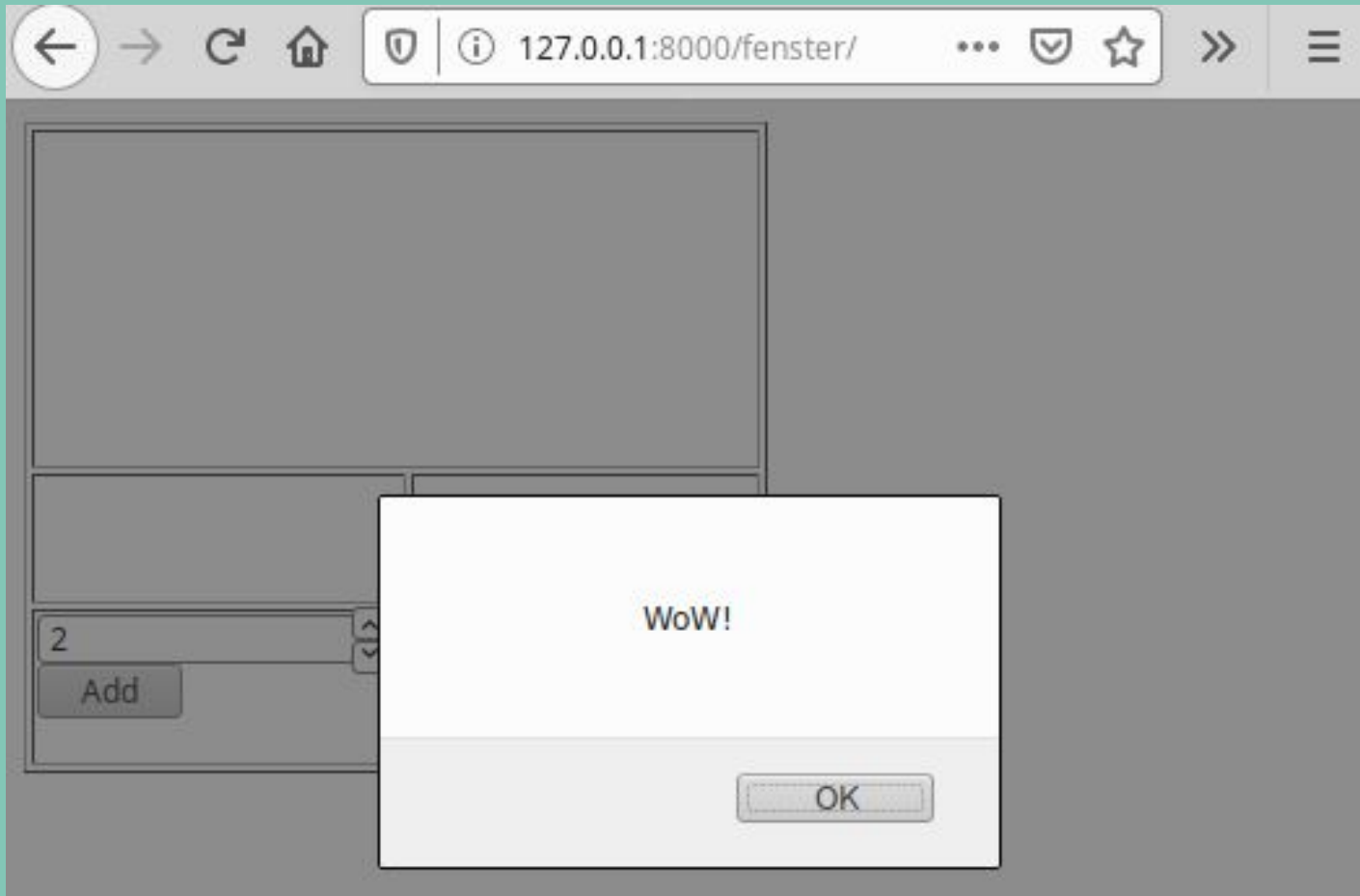
ФАЙЛ fenster/templates/fenster/index.html

```
{% load static %}  
<script src="{% static 'alert.js' %}" > </script>  
<input type="button" value="нажми меня" onClick="cry()">
```

В браузере отображается файл текстом по пути <http://127.0.0.1:8000/static/alert.js>



## Отделение статического контента от динамического



## Запрос к серверу с CSRF-ключом без Cookie

ФАЙЛ `fenster/templates/fenster/index.html`

```
{% csrf_token %}  
<script src="{% static 'alert.js' %}" > </script>
```

```
const csrftoken = document.querySelector(  
  '[name=csrfmiddlewaretoken]').value;  
var post_body = {'3': ['26']};  
var request_header = {  
  'X-CSRFToken': csrftoken,  
  'Accept': 'text/html',  
  'Content-Type': 'application/json'  
};  
fetch('/lessons/3/',    // Путь (url)  
  {  
    method: 'POST',  
    body: post_body,  
    headers: request_header  
  }).then(response => console.log(response));
```

ФАЙЛ `static/alert.js`

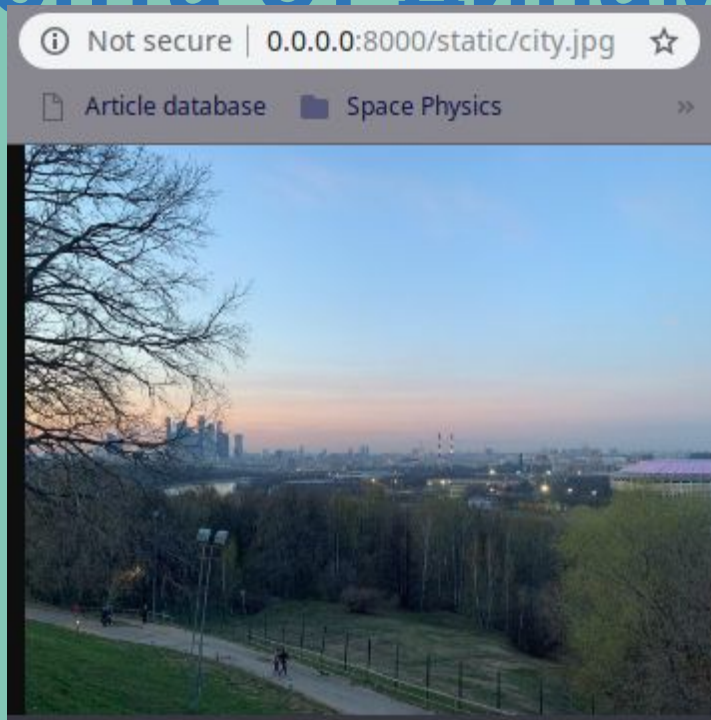
# Сборка статических файлов всех приложений

## ФАЙЛ `wcity/settings.py`

```
STATIC_ROOT = BASE_DIR
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, 'static')
]
```

```
(py3ve) user@computer:~wcity/$ ./manage.py collectstatic
```

# Отделение статического контента от динамического



# Отправка почтовых уведомлений

## ФАЙЛ `wcity/settings.py`

```
# Email
# https://docs.djangoproject.com/en/3.0/topics/email/
EMAIL_PORT = 465
EMAIL_HOST = 'smtp.yandex.ru'
EMAIL_HOST_USER and EMAIL_HOST_PASSWORD se
auth_user='alisawera', #
                        auth_password=yapass
EMAIL_USE_SSL = True
```

# Отправка почтовых уведомлений

views.py+

```
23 from django.core.mail import send_mail #
24
25 from .pss import yapass # import from fenster/pss.py #
26 if request.method == "POST":
27     if 'selected_fenster' in request.POST:
28         f = Fenster.objects.get(
29             pk=request.POST['selected_fenster']
30         )
31         f.for_rent=False
32         f.save()
33         send_mail(
34             subject='Fenster was sold',
35             message='Fenster #%i was sold.' % f.id,
36             send_mail( #
37                 subject='Fenster', #
38                 message='sold', #
39                 from_email='alisawera@yandex.ru', #
40                 recipient_list=['alisawera@gmail.com'], #
41                 ) #
42             ) #
```

# Домашнее задание 8

- Сделать стиль для странички
- Настроить стиль в шаблоне base.html
- Сделать названия картинок координатами + направлением
- Подбирать к окну вид по координатам

# Журналирование (logs)

Python уже содержит удобный модуль logging, настроим его

```
logging.config.dictConfig({
    'version': 1,
    'disable_existing_loggers': False,
    'handlers': {
        'file': {
            'level': 'DEBUG',
            'class': 'logging.FileHandler',
            'filename': 'fenster/debug.log',
        },
    },
    'loggers': {
        'django': {
            'handlers': ['file'],
            'level': 'DEBUG',
            'propagate': True
        },
    },
})
```

wcity/settings.py:



# Журналирование (logs)

Python уже содержит удобный модуль logging, настроим его

▼ 4 ■■■■■ fenster/views.py

111 -4,6 +4,9

```
4   4   from django.contrib.auth.decorators import login_required
5   5   from django.core.mail import send_mail
6   6   from .pss import yapass
```

```
7   +   import logging
8   +
9   +   logger = logging.getLogger(__name__+'.log')
```

```
7   10
8   11   def index(request):
9   12       # test_creation()
```

112 -35,6 +38,7

```
111 def buy(request, fenster_id):
```

```
35  38       )
36  39       f.for_rent=False
37  40       f.save()
```

```
41  +   logger.info('Fenster #%s was sold' %(request.POST['selected_fenster']))
38  42   print("from: %s " % ('*@e1.ru', ))
39  43   send_mail(
40  44       subject='Fenster was sold',
```

# Тестирование

Django расширяет функционал unittest применительно к своей задаче  
Файл tests.py уже создан в момент startproject

```
from django.test import TestCase
from django.test import Client
from django.http import HttpResponseRedirect
from .models import RowType
```

```
class FensterTest(TestCase):
    def setUp(self):
        self.client = Client()
        RowType(
            fenster_width=10,
```

# и другие параметры

```
            price=300,
            window_view=''
```

# Тестирование

Например, можно эмулировать запрос, не запуская браузер!

```
def test_buy(self):  
    response = self.client.post(  
        '/fenster/buy', # путь  
        { # словарь передаваемых переменных  
            'selected_fenster': 128  
        }  
    )  
    self.assertIsInstance(  
        response,  
        HttpResponse  
    )  
    self.assertEqual(  
        response.status_code,
```

# Тестирование

## ФАЙЛ `wcity/settings.py`

```
ALLOWED_HOSTS = ['127.0.0.1', 'testserver', '0.0.0.0']
```

```
(py3ve) user@computer:~wcity$ pip install coverage
```

```
(py3ve) user@computer:~wcity$ coverage run --source='.'  
manage.py test
```

```
(py3ve) user@computer:~wcity$ coverage run manage.py  
test --debug-mode fenster
```

```
(py3ve) user@computer:~wcity$ coverage html
```

Теперь в директории `htmlcov` располагается файл `index.html`

Откройте в браузере <file:///home/student/wcity/htmlcov/index.html>

`django` или `apache` запускать для просмотра НЕ НАДО!

# Тестирование

Django тесты

Как создать специального тестового пользователя для тестирования?

# Домашнее задание 9

- Покрыть проект тестами на 100% по строкам
- Подобрать функциональное покрытие, обсудить его с коллегами
- Отследить по логам все этапы работы приложений

# Как строятся настоящие сайты

- Сервер (железный и настроенный вами или арендованный)
- Система контроля версий (git, ...)
- Система ведения задач (redmine, ...)  
git можно сопрячь с redmine
- Локальные копии
- Тесты (автотесты)

# Как строятся настоящие сайты

## Жизненный цикл разработки

1. Постановка задачи (через redmine)
2. Разработка и локальное тестирование
3. Push (в своей ветке!)
4. Тестирование на сервере-development
5. Передача на основной сервер
6. Обнаружение ошибки
7. Откат
8. goto 1



# Сохранение и восстановление данных

Код хранится в репозитории (git)

1. Перед началом правок сохраняем состояние предыдущих правок (git stash)
2. Вытащили все принятые изменения других пользователей (git pull) - в master-ветке
3. Влили ветку мастер в свою
4. Внесли изменения...

База данных регулярно сохраняется в инкрементной резервной копии

# Apache2 + mod\_wsgi



WSGI -- Web Server Gateway Interface.

Описывает взаимодействие между Web-Server и цепочкой приложений, исполняющих запрос.

```
user@computer:~$ sudo apt-get install apache2
```

```
user@computer:~wcity/$ pip install mod_wsgi-httpd
```

Строго в две строки. В следующем слайде приведена ошибка!

```
user@computer:~wcity/$ pip install mod_wsgi
```

Pip установит mod\_wsgi с учетом среды, поскольку pip выполняется из-под среды и, таким образом, с учетом версии python

# Apache2 + mod\_wsgi (Windows)



C:\Program Files (x86)\edb\pem\httpd\apache

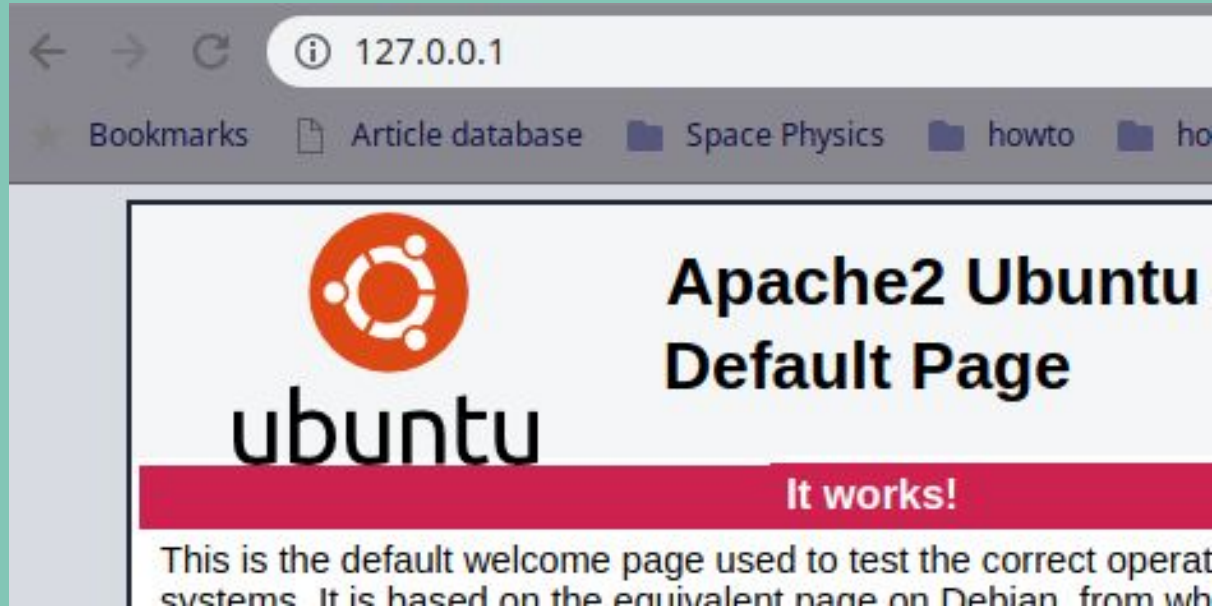
# Apache2 + mod\_wsgi



```
(py3ve) student@student-VirtualBox:~/wcity$  
pip install mod_wsgi-httpd mod_wsgi  
Collecting mod_wsgi-httpd  
  Downloading https://files.pythonhosted.org/packages/07/b5/2ed  
5131c0adc374a9da382ef651c6adcdad51c10c6d47db302d6a6539ee8/mod_w  
sgi-httpd-2.4.41.1.tar.gz (3.7MB)  
    |████████████████████| 3.7MB 3.2MB/s  
Collecting mod_wsgi  
  Downloading https://files.pythonhosted.org/packages/a0/8b/34d  
d82c3e15a031e9c89f5a5d2ca527ec35b7a01e1e7530abb61ffdb4d60/mod_w  
sgi-4.7.0.tar.gz (497kB)  
    |████████████████████| 501kB 570kB/s  
ERROR: Command errored out with exit status 1:  
  command: /home/student/py3ve/bin/python3 -c 'import sys, s  
etuptools, tokenize; sys.argv[0] = ''''/tmp/pip-install-ytq_g7  
dq/mod-wsgi/setup.py''''; __file__ = ''''/tmp/pip-install-ytq_g  
7dq/mod-wsgi/setup.py''''; f=getattr(tokenize, ''''open'''' ,  
open)(__file__); code=f.read().replace(''''\\r\\n''', ''''\\n''')'
```

'missing Apache httpd server packages.' % APXS)

# Apache2 default page





# Apache2 + mod\_wsgi

Теперь необходимо получить конфигурацию для apache2:

```
user@computer:~wcity/$ mod_wsgi-express module-config
```

**НЕ КОПИРОВАТЬ ИЗ PPT!!! СВОЙ!!!**

Получите вывод вроде этого и скопируйте его:

```
LoadModule wsgi_module
"/home/student/py3ve/lib/python3.6/site-packages/mod_wsgi/serve
r/mod_wsgi-py36.cpython-36m-x86_64-linux-gnu.so"
WSGIPythonHome /home/student/py3ve
```



# Apache2 + mod\_wsgi

```
user@computer:~$ sudo -e /etc/apache2/apache2.conf
```

## ФАЙЛ /etc/apache2/apache2.conf

```
# Дописываем в конце:
```

```
WSGIPythonHome /home/student/py3ve  
WSGIPythonPath /home/student/wcity
```

Это был глобальный конфигурационный файл Apache. Теперь отредактируем файл для конкретного сайта – виртуального хоста.

# Apache2 + mod\_wsgi



```
<Fri Apr 26 15:30:56> <wera@anarinya:~>
+ $ ls -lah --color /etc/apache2/sites-enabled
total 8.0K
drwxr-xr-x 2 root root 4.0K Apr 26 14:57 ./
drwxr-xr-x 8 root root 4.0K Apr 26 14:57 ../
lrwxrwxrwx 1 root root   35 Apr 26 14:57 000-default.conf ->
    ../sites-available/000-default.conf
<Fri Apr 26 15:31:02> <wera@anarinya:~>
+ $ ls -lah --color /etc/apache2/sites-available/
total 20K
drwxr-xr-x 2 root root 4.0K Apr 26 14:57 ./
drwxr-xr-x 8 root root 4.0K Apr 26 14:57 ../
-rw-r--r-- 1 root root 1.4K Oct 10 2018 000-default.conf
-rw-r--r-- 1 root root 6.2K Oct 10 2018 default-ssl.conf
```

Apache будет запускаться из-под пользователя www-data. С помощью команды `finger www-data` можно посмотреть, где домашняя директория этого пользователя, и положить туда файл `.pgpass`, чтобы сайт смог продолжать пользоваться базой после установки и настройки этого Веб-Сервера





# Apache2 + mod\_wsgi

```
user@computer:~$ sudo -e  
/etc/apache2/sites-available/000-default.conf
```

ФАЙЛ /etc/apache2/sites-available/000-default.conf

```
WSGISocketPrefix /var/run/wsgi  
<VirtualHost *:80>
```

# Вместо старой ссылки на директорию по умолчанию для Apache

~~# DocumentRoot /var/www/html~~

# Вписать сюда результат выполнения команды

# mod\_wsgi-express module-config

LoadModule wsgi\_module

"/home/student/py3ve/lib/python3.6/site-packages/mod\_wsgi/serve  
r/mod\_wsgi-py36.cpython-36m-x86\_64-linux-gnu.so"

WSGIDaemonProcess wcity python-home=/home/student/py3ve

python-path=/home/student/py3ve:/home/student/wcity

WSGIScriptAlias / /home/student/wcity/wcity/wsgi.py



# Apache2 + mod\_wsgi

# Описание поведения с директорией проекта:

```
<Directory /home/student/wcity/>  
    Order allow,deny  
    Allow from all  
    Options Indexes FollowSymLinks  
    AllowOverride None  
    Require all granted  
</Directory>
```

# И статических файлов! Если collectstatic не выполнен, CSS в админке не будет отображаться

```
Alias /static/ "/home/student/wcity/static/"  
<Directory "/home/student/wcity/static/">  
    Require all granted  
</Directory>
```

# Успехов в настройке!

После сохранения файлов конфигурации и перезапуска apache будет «видеть» django, ваши картинки, CSS и JS

# Развёртывание сайта и БД

Файл requirements.txt

```
django  
psycopg2  
docutils  
coverage  
mod_wsgi  
pandas
```

```
(py3ve)user@computer:~$ pip install -r requirements.txt
```

# Бонус ReactJS

```
(py3ve)user@computer:wcity$ sudo apt-get install npm
```

```
(py3ve)user@computer:wcity$ sudo npm install -g npx
```

```
(py3ve)user@computer:wcity$ npm init -y
```

```
(py3ve)user@computer:wcity$ npm install babel babel-cli@6  
babel-preset-react-app@3
```

# Бонус ReactJS

```
Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
(py3ve) 2019-12-17 18:08:21 <student@student-VirtualBox:~/wcity[master]> $
+ npm install babel-cli@6 babel-preset-react-app@3
npm WARN deprecated core-js@2.6.11: core-js@<3 is no longer maintained and not
recommended for usage due to the number of issues. Please, upgrade your depende
ncies to the actual version of core-js@3.
npm WARN deprecated browserslist@2.11.3: Browserslist 2 could fail on reading B
rowserslist >3.0 config used in other tools.

> core-js@2.6.11 postinstall /home/student/wcity/node_modules/core-js
> node -e "try{require('./postinstall')}catch(e){}"

wcity@1.0.0 /home/student/wcity
├─ babel-cli@6.26.0  extraneous
└─ babel-preset-react-app@3.1.2  extraneous

npm WARN optional Skipping failed optional dependency /chokidar/fsevents:
npm WARN notsup Not compatible with your operating system or architecture: fsev
ents@1.2.11
npm WARN wcity@1.0.0 No description
npm WARN wcity@1.0.0 No repository field.
Активация Windows
Чтобы активировать Windows, перейдите в
раздел "Параметры".
(py3ve) 2019-12-17 18:09:11 <student@student-VirtualBox:~/wcity[master]> $
```

# Бонус ReactJS

```
(py3ve)user@computer:wcity$ mkdir jsx
(py3ve)user@computer:wcity$ npx babel --watch jsx --out-dir
static/ --presets react-app/prod
jsx/alert.jsx -> static/alert.js
jsx/alert.jsx -> static/alert.js
```

## jsx/alert.jsx

```
class Hello extends React.Component {
  render() {
    return <div>Привет, {this.props.toWhat}</div>;
  }
}
function cry() {
  console.log('Tcccc');
  ReactDOM.render(
    <Hello toWhat='Bepa' />,
    document.getElementById('reactplace')
  );
};
```

```
}; /* https://babeljs.io/docs/en/configuration */
```

# Бонус ReactJS

fenster/templates/fenster/index.html

```
<!-- Добавляем библиотеки. Здесь приведены их пути в Интернет,  
но можно скачать к себе и через static добавить локально -->  
<script  
src="https://unpkg.com/react@16/umd/react.development.js"  
crossorigin></script>  
<script  
src="https://unpkg.com/react-dom@16/umd/react-dom.development.j  
s" crossorigin></script>  
  
<!-- По-прежнему ссылаемся на alert.js, будет скомпилирован -->
```

```
{% load static %}  
<script src="{% static 'alert.js' %}" > </script>
```



# Бонус ReactJS через create-react-app (другой способ)

```
(py3ve)user@computer:wcity$ npm install create-react-app
```

```
(py3ve)user@computer:wcity$ sudo npm install -g npx
```

```
(py3ve)user@computer:wcity$ cd fenster
```

```
(py3ve) student@student-Py4:~/wcity/fenster$ npx create-react-app js
(py3ve) student@student-Py4:~/wcity/fenster$ npx create-react-app js
Creating a new React app in /home/student/wcity/fenster/js.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...
loadDep:yargs → fetch
```

В результате у нас страницы могут целиком быть созданы на reactjs без использования шаблонов Django или других, т.е. это замена front-end части Django. Back-end при этом может по-прежнему быть написан на Django, либо, что уже достаточно логично – на NodeJs