Факультатив ФКН

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ НА БАЗЕ ПЛАТФОРМЫ

Преподаватель
Папулин Сергей Юрьевич (***papulin_hse@mail.ru***)

Москва, 2018

Темы

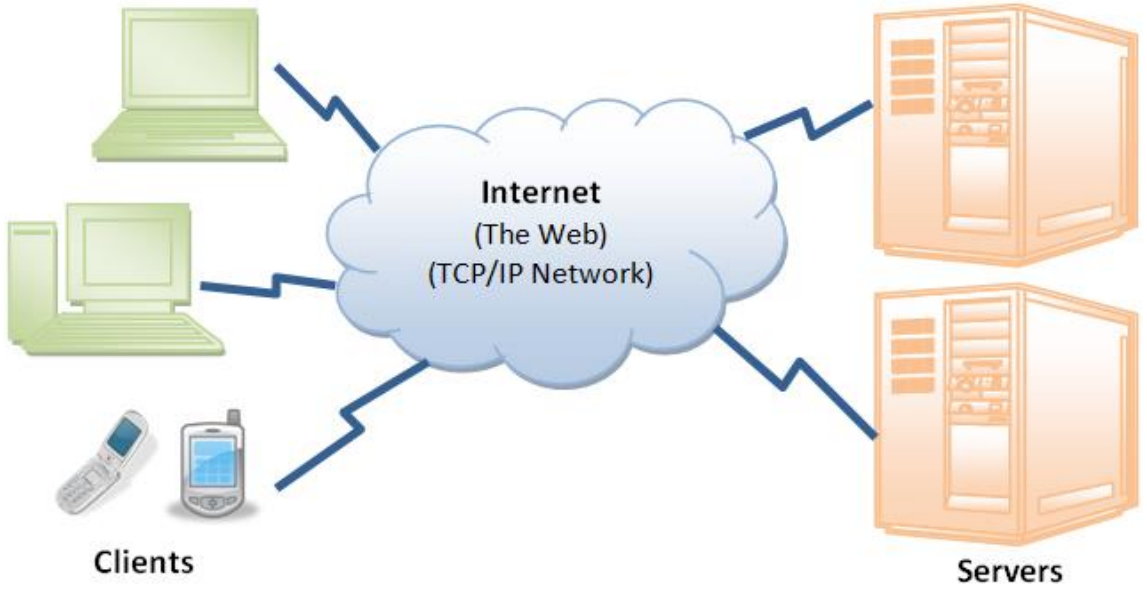
- Основные компоненты
- Протокол HTTP
- WSGI
- Django Framework
- Архитектура серверной части
- Первое приложение

Основные компоненты

Клиент и сервер

В качестве клиент выступает ***user-agent***, (например, веб-браузер). Он выполняет запросы (*requests*) серверу от имени пользователя.

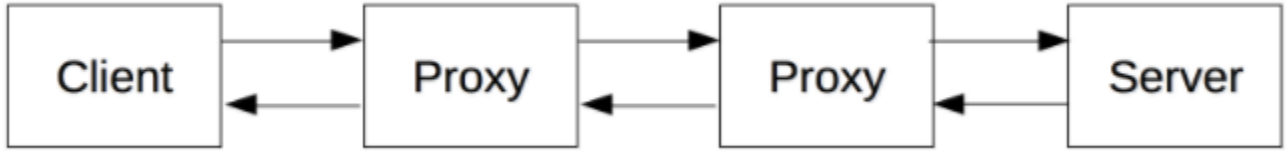
Сервер обрабатывает индивидуальные запросы и отправляет ответ (*response*)



https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Прокси

Между клиентом и сервером



Прокси перехватывает запросы и отправляет ответы, может продвигать запросы и вносить изменения в заголовки

Forward proxy обрабатывает запросы от кого-либо или к кому-либо в Интернете

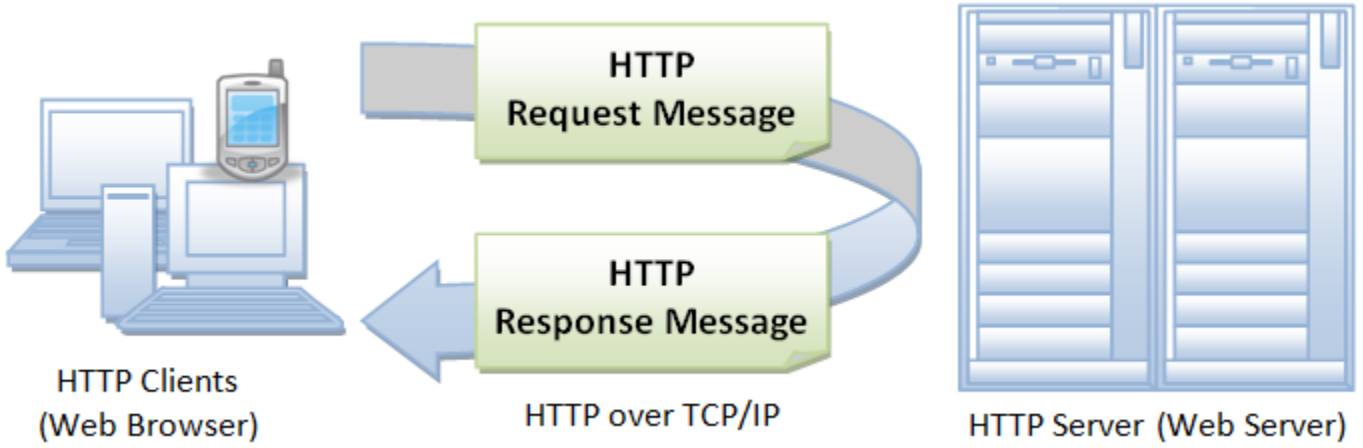
Reverse proxy обрабатывает запросы из Интернета и перенаправляет их в сервер внутренней сети

Примеры

- Кэширование
- Фильтрация трафика
- Балансировка нагрузки
- Аутентификация
- Логирование

Протокол HTTP

Запрос-ответ



Особенности HTTP

HTTP (Hypertext Transfer Protocol)

- Асимметричный, т.е. клиент запрашивает информацию у сервера, но не наоборот
- Не хранит информацию о предыдущих запросах (stateless)
- Позволяет обговаривать типы данных и их представления

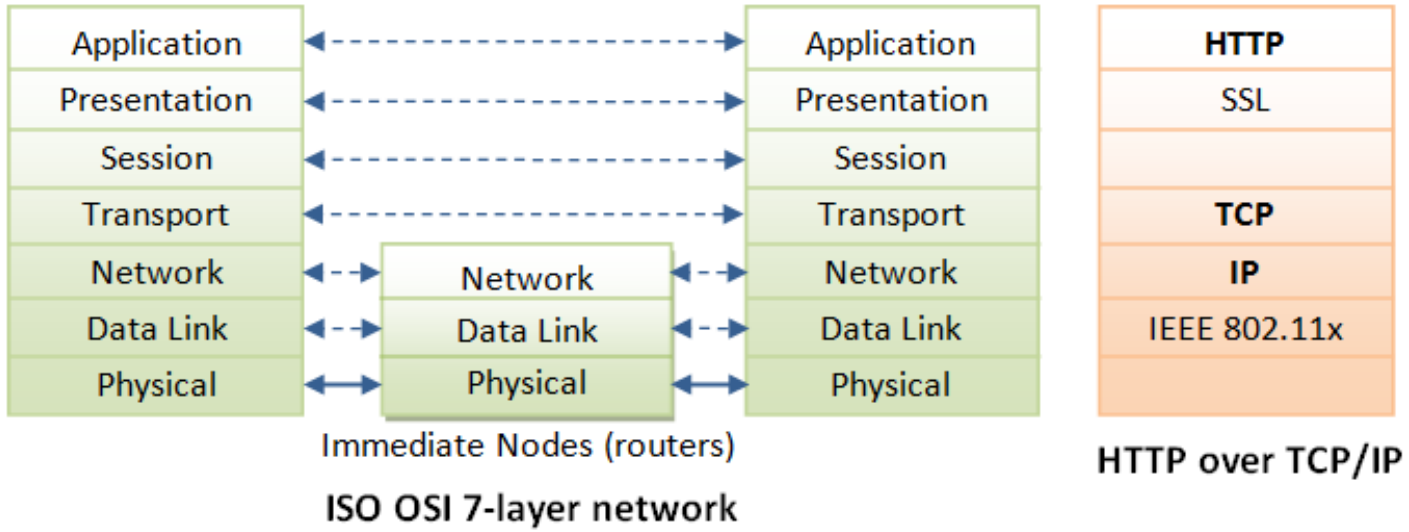
HTTP 1.1

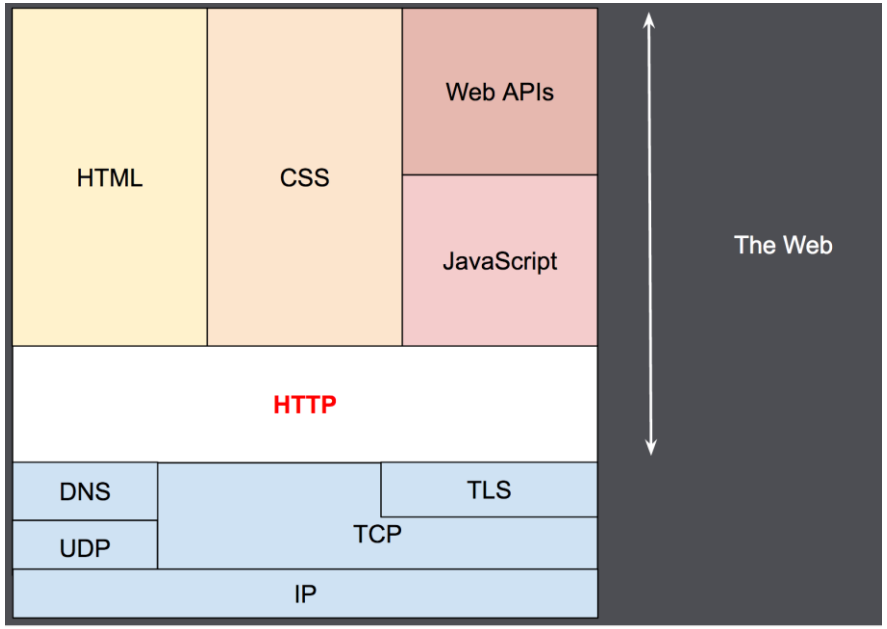
- Кэш
- Одно соединение на несколько запросов
- Прокси

HTTP 2

- Опережающая загрузка
- Сжатие заголовков
- Мультиплексирование запросов и ответов

HTTP в модели OSI

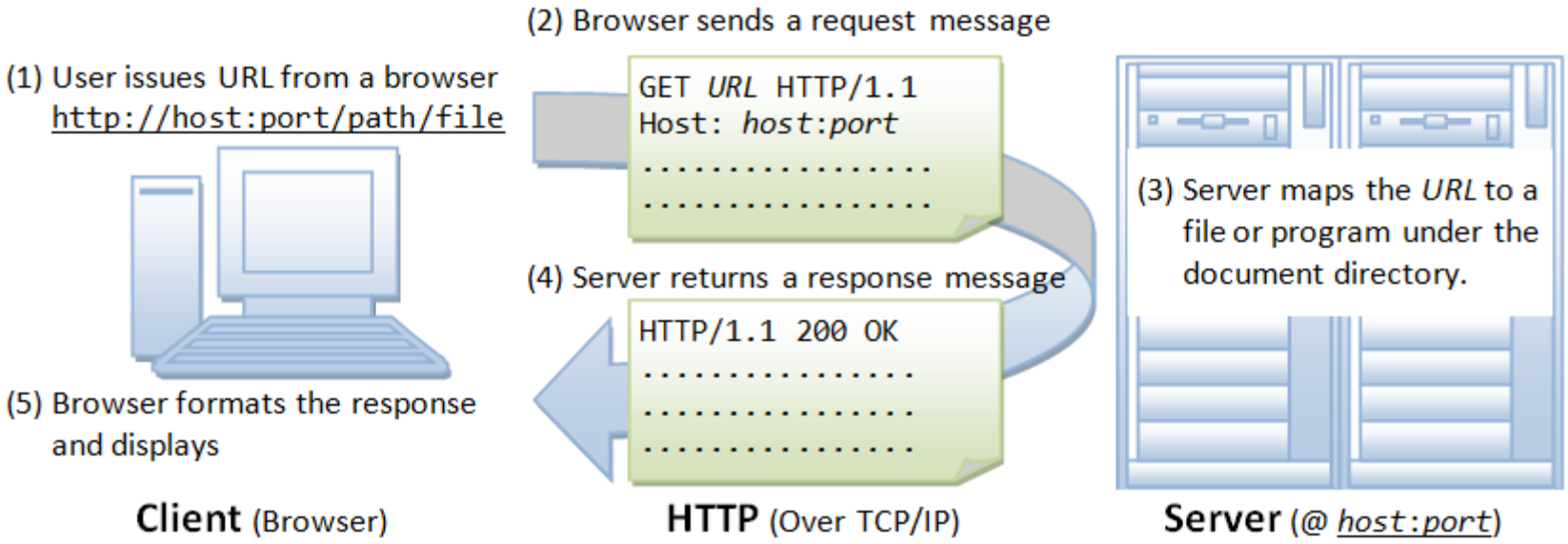




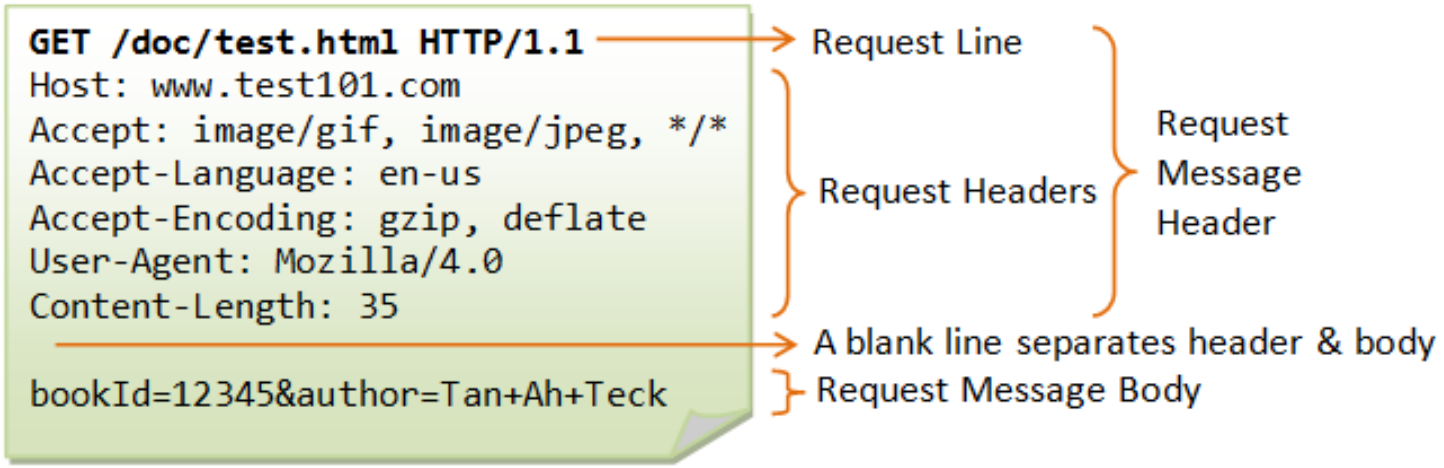
https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

Запрос-ответ 2



Структура запроса HTTP



Структура запроса HTTP. Передача параметров в GET и POST

http://example.com?name1=value1&name2=value2&name3=value3&...

query string

GET

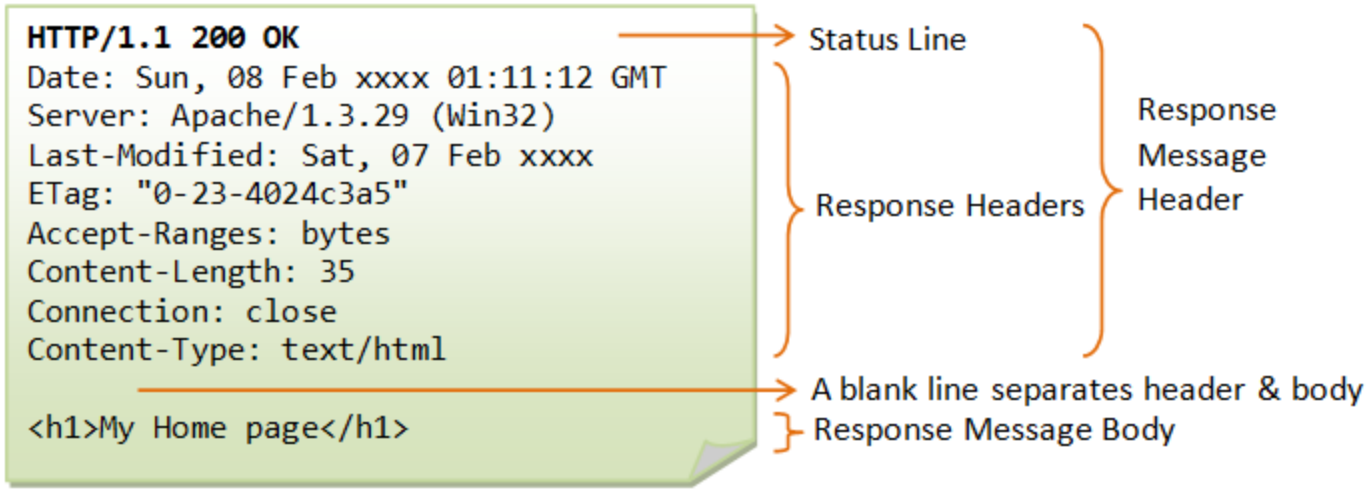
```
GET request-URI?query-string HTTP-version
(other optional request headers)
(blank line)
(optional request body)
```

POST

```
POST request-URI HTTP-version
Content-Type: mime-type
Content-Length: number-of-bytes
(other optional request headers)

(URL-encoded query string)
```

Структура ответа HTTP



Статусы ответа

- 1xx – информационные
- 2xx – успешное выполнение

  200 OK
- 3xx – перенаправление

  301 Move Permanently
- 4xx – ошибка на стороне клиента

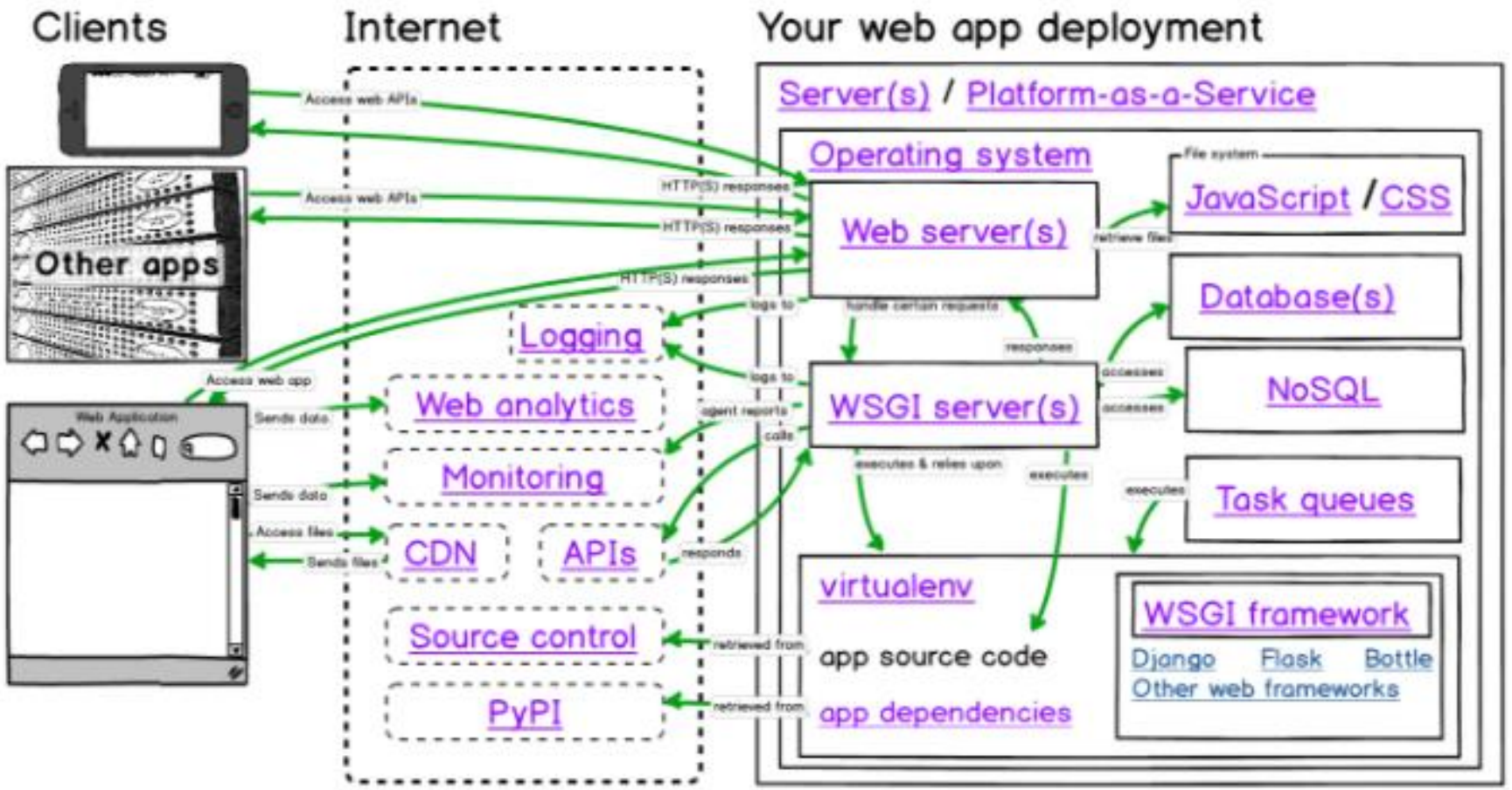
  403 Forbidden

  404 Not Found
- 5xx – ошибка на стороне сервера

  500 Internal Server Error

Web Server Gateway Interface (WSGI)

Архитектура Сервера



https://www.fullstackpython.com/deployment.html

Формирование динамического контента

- Python – WSGI – gunicorn, mod_wsgi, uWSGI
- Ruby – Rack – unicorn, Phusion Passenger
- JavaScript – Node.js
- Java – Servlets – Tomcat, Jetty

WSGI

WSGI (Web Server Gateway Interface) – стандарт запуска python веб приложений

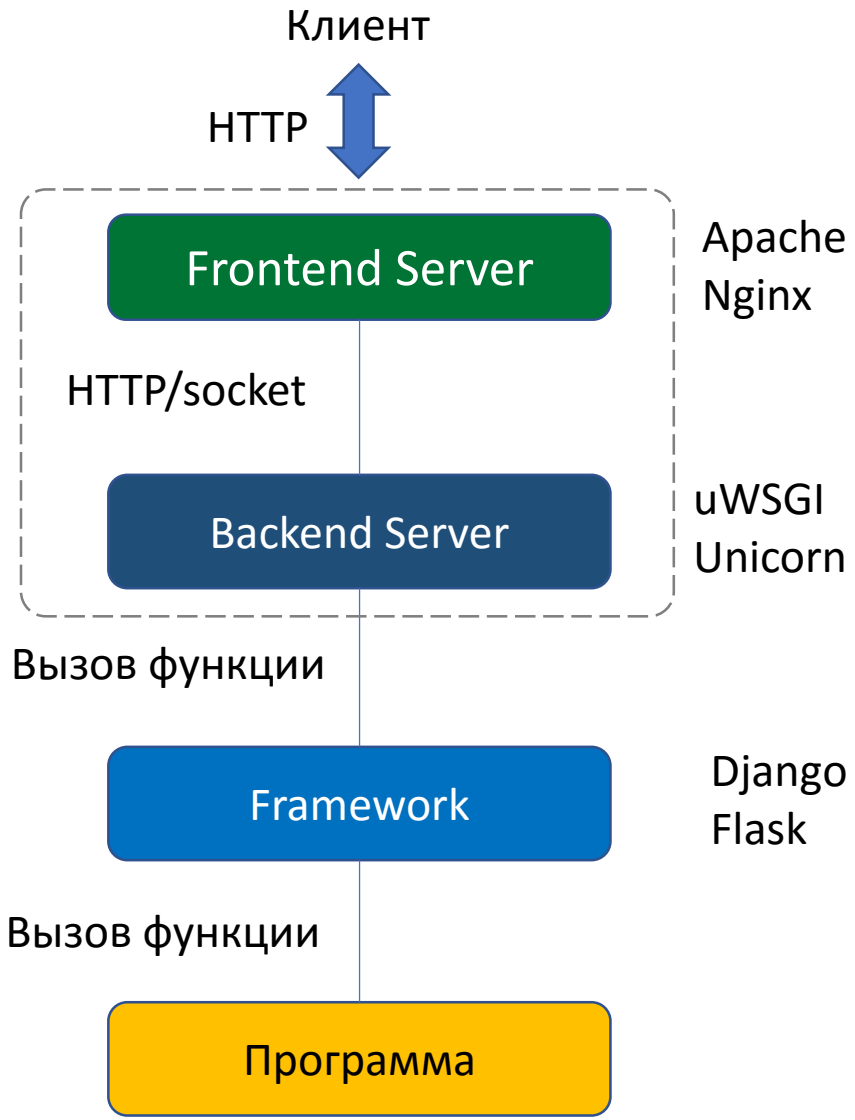
WSGI контейнер – процесс, в котором запускается приложение

Веб-сервер перенаправляет запросы WSGI контейнерам и получает ответ в форме HTML

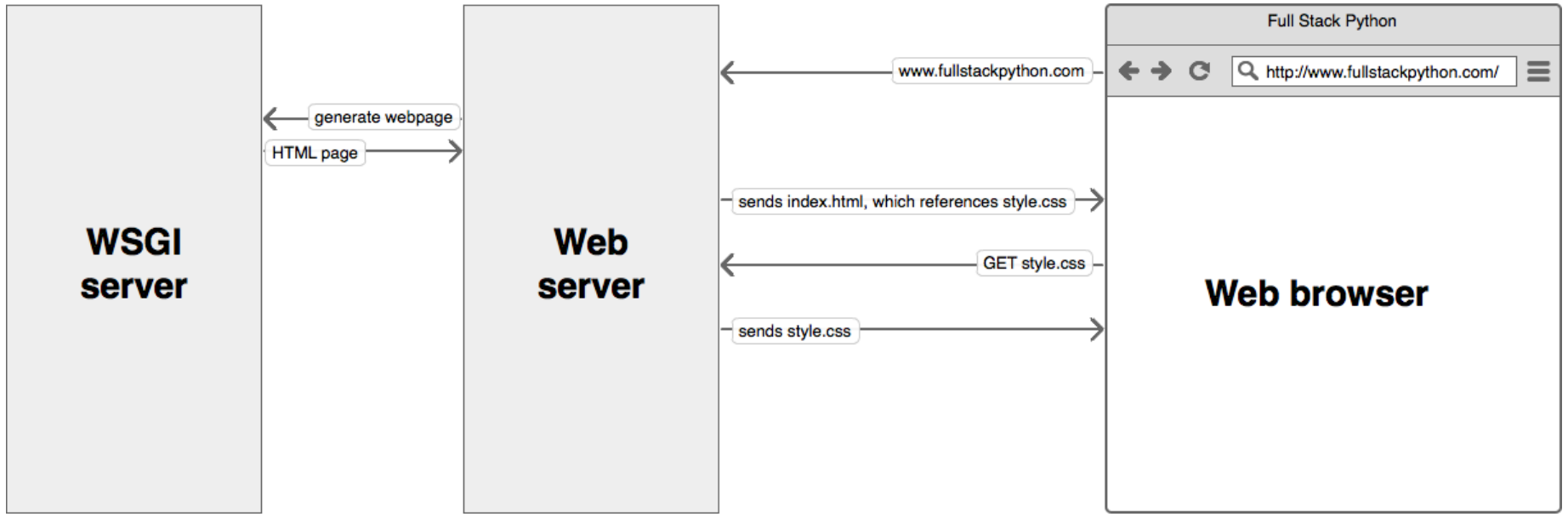
WSGI серверы (контейнеры):

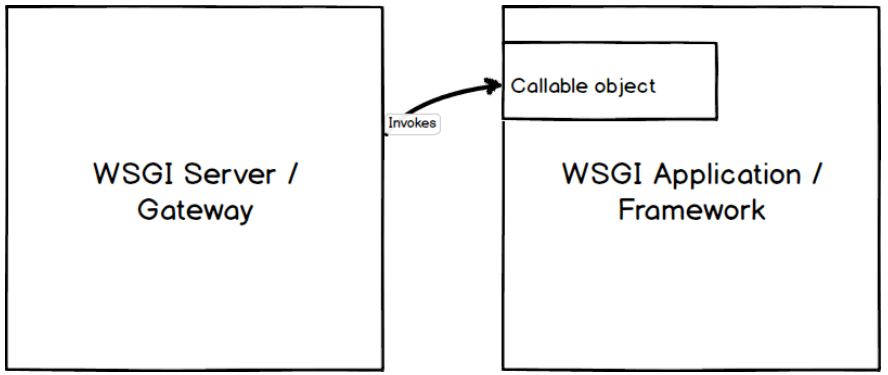
- uWSGI
- Unicorn
- mod_wsgi

Обработка запроса



Архитектура Клиент-Сервер





Django Framework

Особенности Django (с официального сайта)

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django was designed to help developers take applications from concept to completion as quickly as possible.

Django includes dozens of extras you can use to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks — right out of the box.

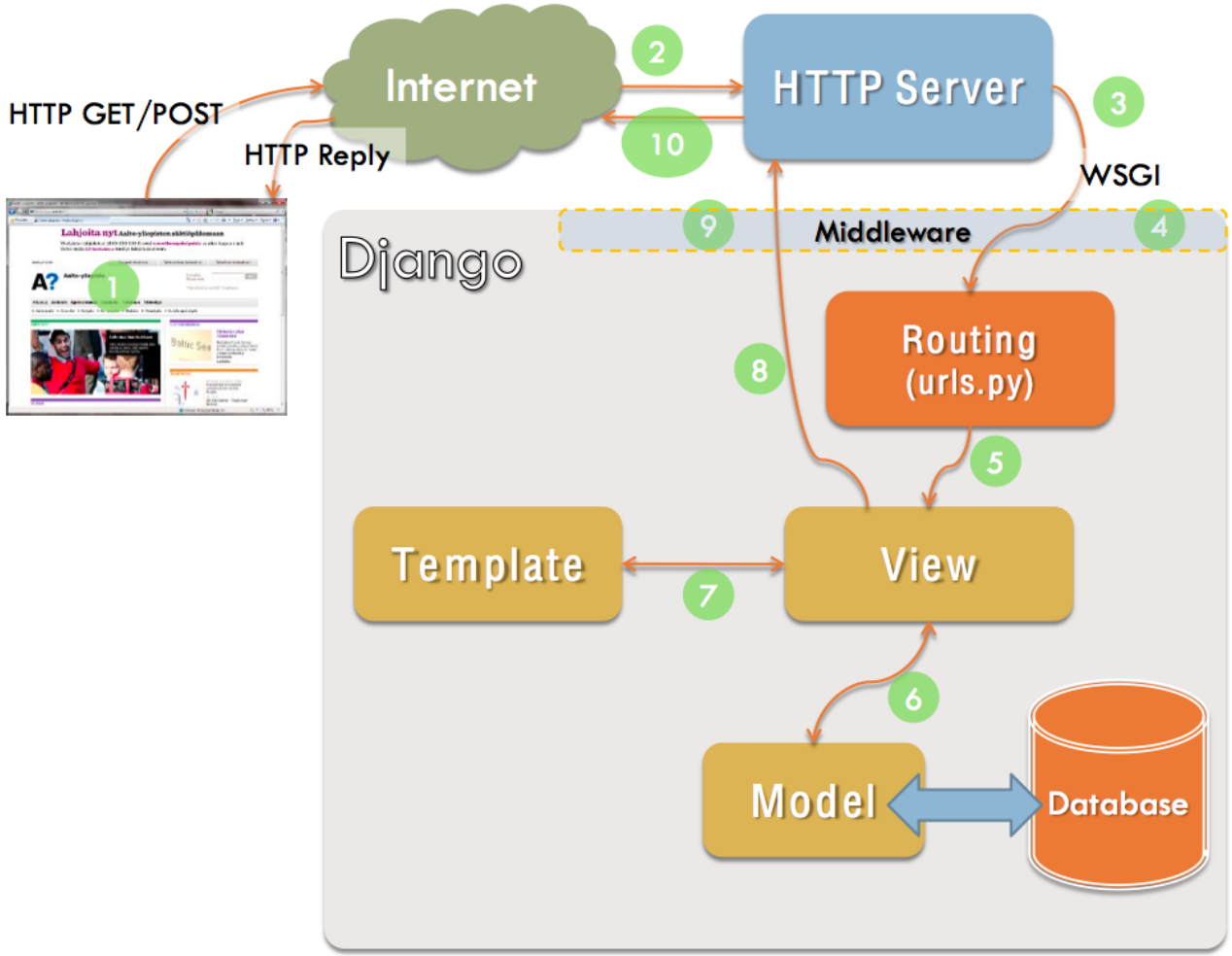
Django takes security seriously and helps developers avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery and clickjacking. Its user authentication system provides a secure way to manage user accounts and passwords.

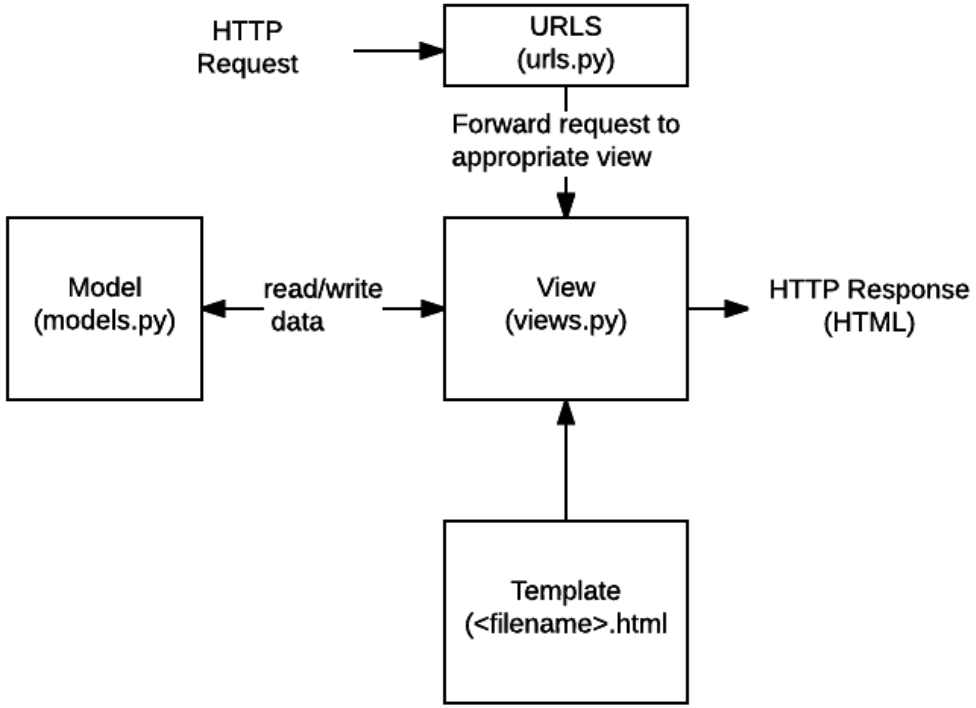
Some of the busiest sites on the planet use Django's ability to quickly and flexibly scale to meet the heaviest traffic demands.

Companies, organizations and governments have used Django to build all sorts of things — from content management systems to social networks to scientific computing platforms.

https://www.djangoproject.com/start/overview/

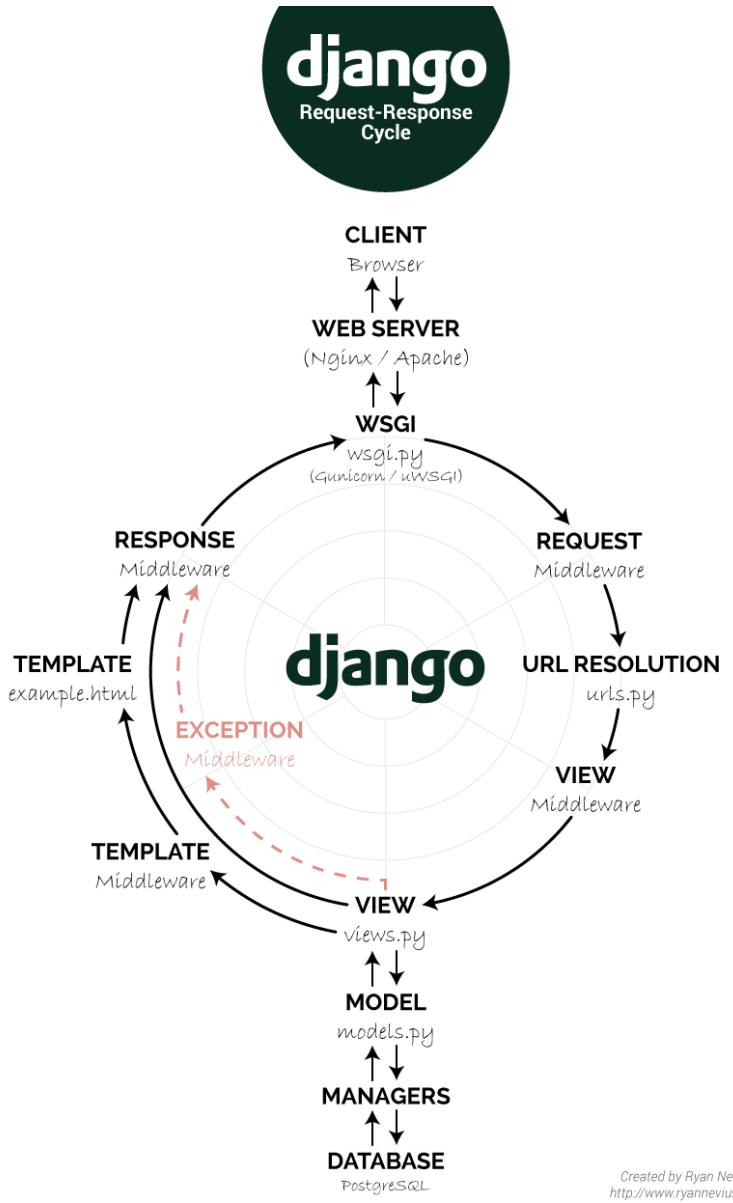
Структура Django





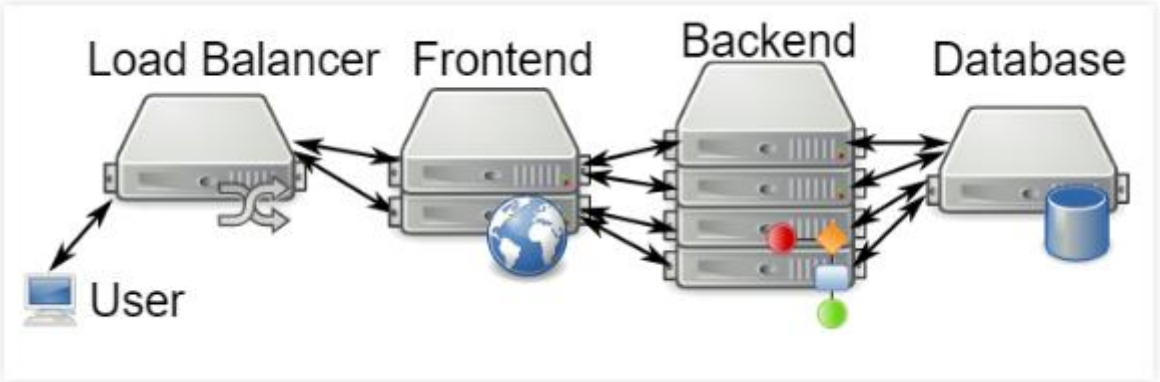
https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

Запрос-ответ в Django

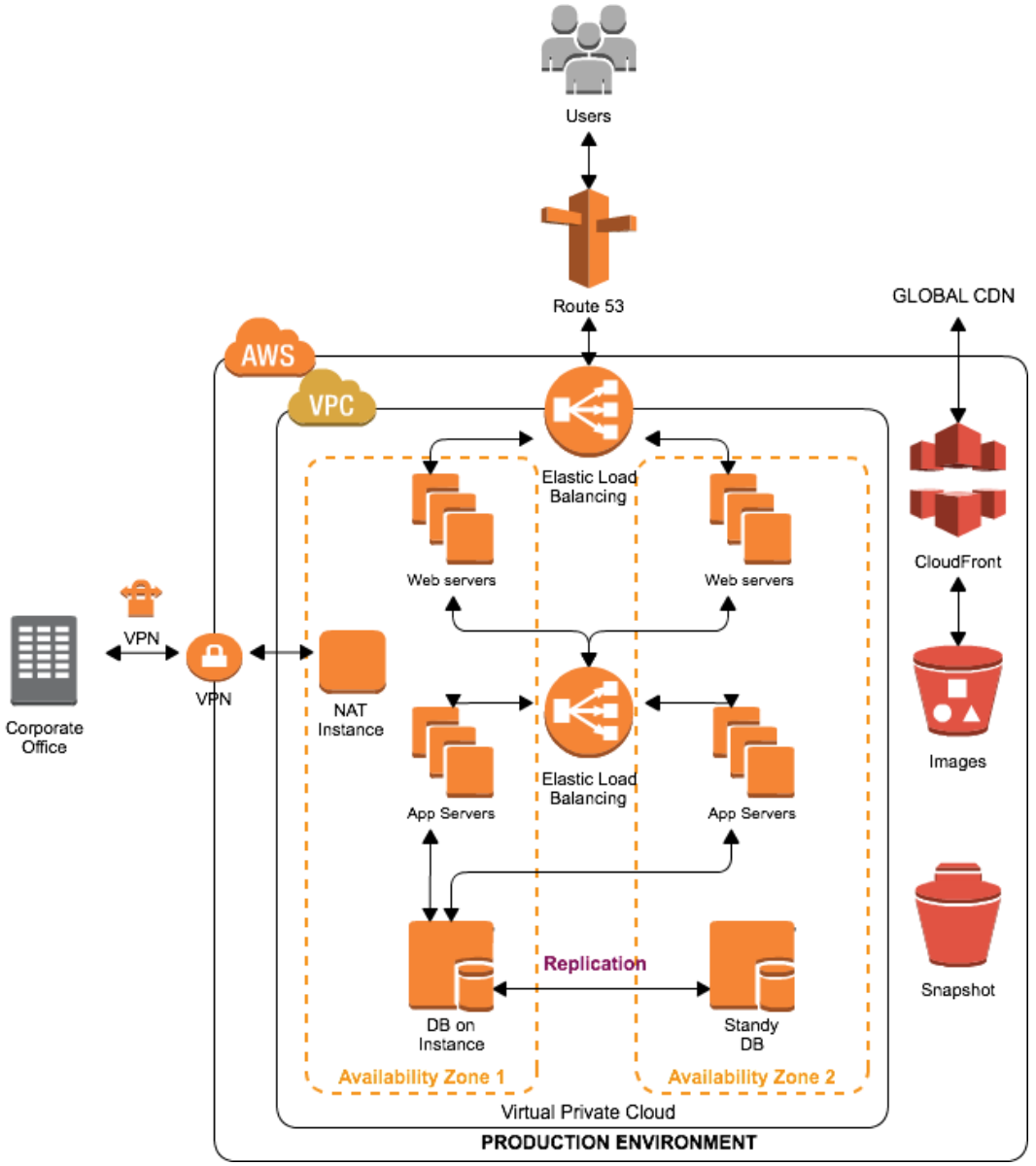


Архитектура серверной части

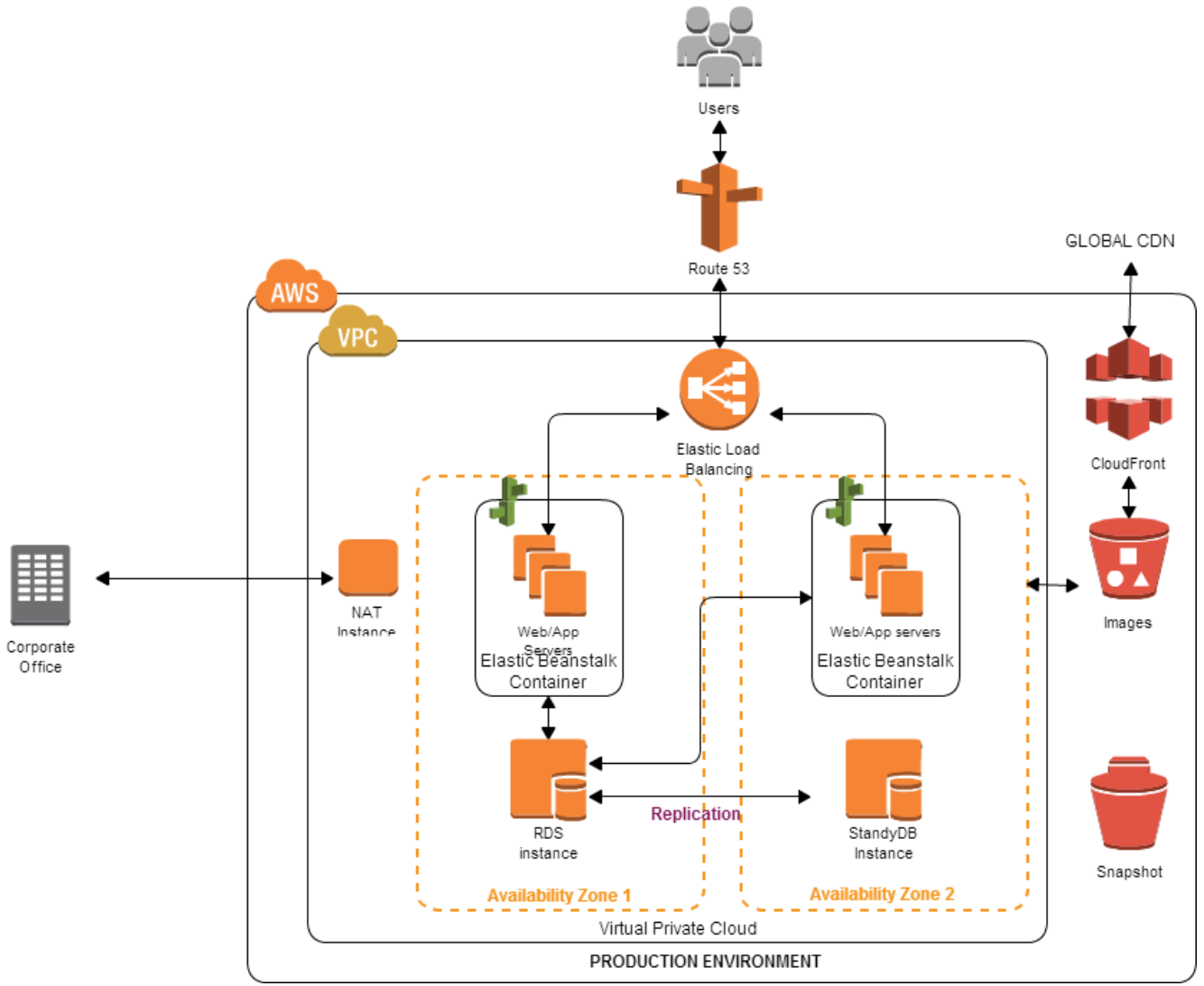
Масштабируемость



Веб-сервер на AWS

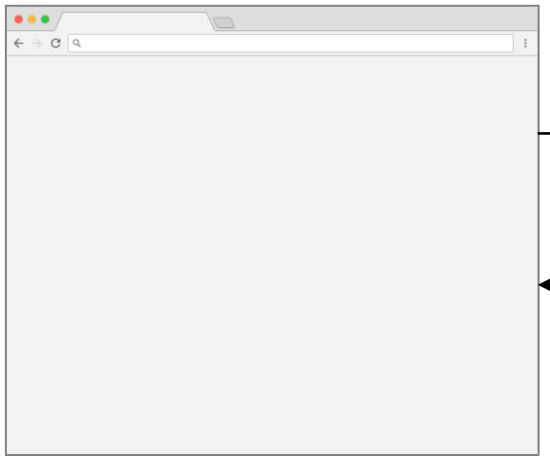


Веб-сервер на AWS. Elastic Beanstalk



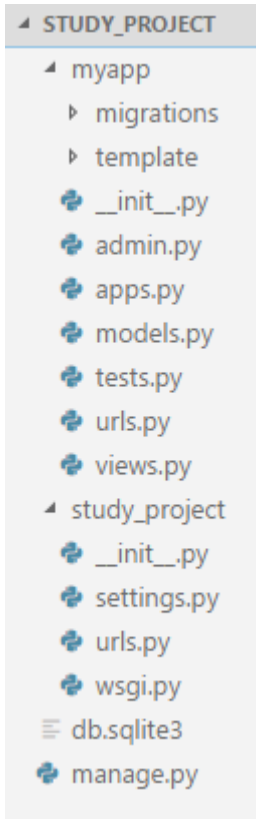
Первое приложение

Пример



Request

Response



Пример

```
# check a python version
python --version
>> Python 3.5.2

# check available python packages
pip freeze
>> Django==1.9.1
>> virtualenv==15.0.1

# if virtualenv is installed, then check its version...
virtualenv --version
>> 15.0.1

# and create a new python environment - django-env. Set a proper interpreter
virtualenv -p /usr/bin/python3.5 django-env
>> Running virtualenv with interpreter /usr/bin/python3.5
>> Using base prefix '/usr'
>> New python executable in /home/student/django-env/bin/python3.5
>> Also creating executable in /home/student/django-env/bin/python
>> Installing setuptools, pkg_resources, pip, wheel...done.
```

Пример

```
# activate the new environment
source django-env/bin/activate

# install the django in it
pip install django
>> Installing collected packages: pytz, django
>> Successfully installed django-2.0.2 pytz-2018.3

# check installed packages
pip freeze
>> Django==2.0.2
>> pkg-resources==0.0.0
>> pytz==2018.3

# create a directory for django projects
mkdir django-projects
cd django-projects
```

Пример

```
# create a django project called study_project
django-admin startproject study_project

# get into the study_project directory
cd study_project

# look at the created files and directories
ls -Rl

# run a lightweight development server on your local machine
python manage.py runserver

>> March 05, 2018 - 16:07:02
>> Django version 2.0.2, using settings 'study_project.settings'
>> Starting development server at http://127.0.0.1:8000/
>> Quit the server with CONTROL-C.

# Go to the link http://127.0.0.1:8000/. Your browser show the django
starting page.
```

Пример

```
# create a web application
python manage.py startapp myapp

# look at the created files and directories
ls -Rl

# create the template directory in the myapp folder
# create the index.html file and add the following code:
<html>
    <head>
        <title>My first Django app</title>
    </head>
    <body>
        <h1>Hello, {{ name }}!</h1>
    </body>
</html>

# change the settings.py file. add 'DIRS': ["/home/student/django-projects/study_project"]
to the TEMPLATES variable
```

Пример

```
# In the veiws.py add the following code:
from django.shortcuts import render

def my_view(request):
        return render(request,
                "myapp/template/index.html",
                {"name": "World"},
                content_type="text/html")

# create the urls.py inside the myapp directory and add the following code:
from django.urls import path

from . import views

urlpatterns = [
path("", views.my_view, name="my_first_view"),
]
```

Пример

```
# change the urls.py file in the main directory to
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
path("myapp/", include("myapp.urls")),
path("admin/", admin.site.urls),
]

# Go to http://127.0.0.1:8000/myapp/

# Stop the server with Cntrl-C

# Deactivate the django-env environment
deactivate

# You have to activate the environment and change a directory to ~/django-
projects/study_project to run the server
```

Источники

https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

https://www.fullstackpython.com/deployment.html

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

https://www.djangoproject.com/start/overview/

https://docs.djangoproject.com/en/2.0/intro/tutorial01/