

Знакомство с Django. Подготовка и запуск проекта

Александр Мужев

Александр Мужев

О спикере:

- Старший инженер по тестированию в компании "ООО Альянс АйтиТехнолоджи"
- Эксперт продвинутого курса "Инженер по тестированию: с нуля до middle" в образовательной платформе ООО Нетология
- Преподаватель QA (очная форма) по направлению ручное и автоматизированное тестирование ПО в компьютерной академии ТОР
- Репетитор по ручному и автоматизированному тестированию ПО (расширенный курс). Провожу индивидуальные занятия со студентами.
- QA Full Stack-фрилансер
- Неоднократно организовал весь процесс тестирования с нуля
- Опыт работы в тестировании банковских продуктов и приложений сферы медицинских услуг.



План занятия

1. Что такое Django. Установка и запуск
2. Проект и приложение
3. Клиент и сервер
4. MVC и Django
5. Работа с урлами. Роутинг в Django
6. Как дебажить Django-проект
7. Что почитать

Что такое Django

Что такое Django

Django – фреймворк для быстрого создания веб-приложений, полностью написанный на **Python**.

Django является очень популярным проектом и используется многими крупными компаниями.

Исходный код Django доступен на Github:

<https://github.com/django/django>



Почему Django



Грамотно спроектированная архитектура



Прозрачная работа с базой данных



Серьёзное отношение к безопасности



Огромное количество библиотек и написанного кода



Подробная документация (*на английском*)

Установка



Для установки **Django**, выполните команду в консоли

```
$ pip install django
```

Чтобы убедиться, что все установилось **корректно**

```
$ python -m django --version
```

Проект и приложение

Что такое проект и приложение

Под **проектом** можно понимать **полноценный сайт**. Это:

- коллекция настроек
- база данных
- подключенные приложения

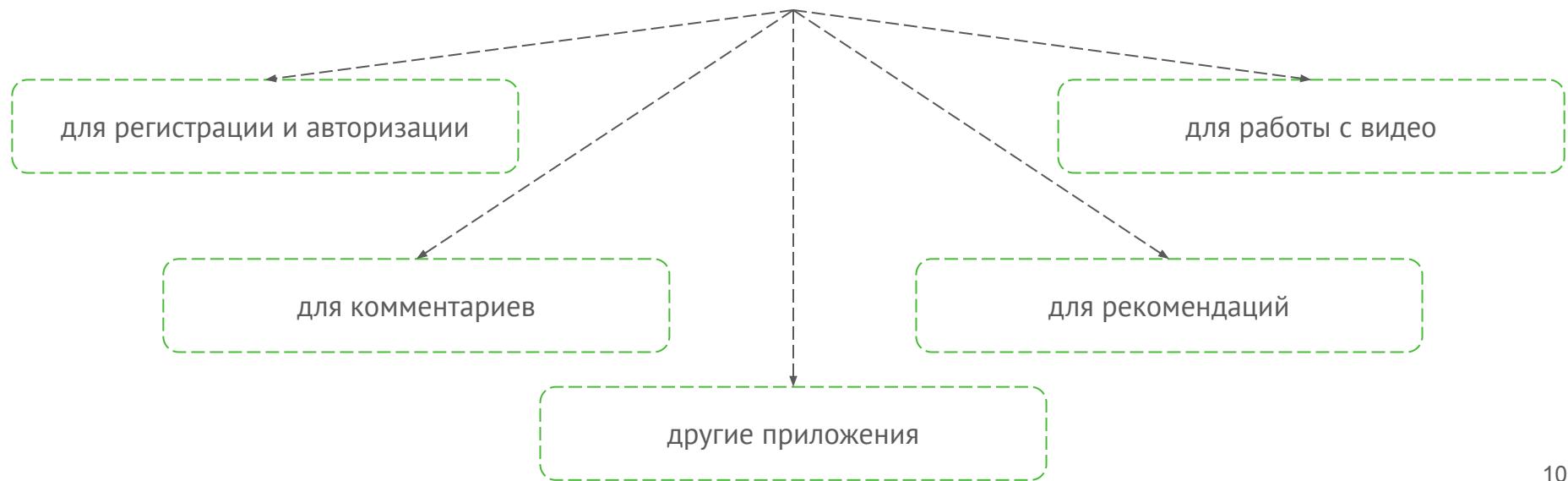
Например, YouTube – это **проект**



Что такое проект и приложение

Приложение – это изолированная часть функциональности. Приложения могут переиспользоваться в различных проектах. Ближайшая аналогия – модули в Python

Приложения проекта X



Создание проекта

Запустите

```
$ django-admin startproject django_netology
```

После этого у вас появится директория *django_netology*.



Структура проекта

Содержимое директории

```
manage.py
django_netology
    __init__.py
    settings.py
    urls.py
    wsgi.py
```

Создание приложения

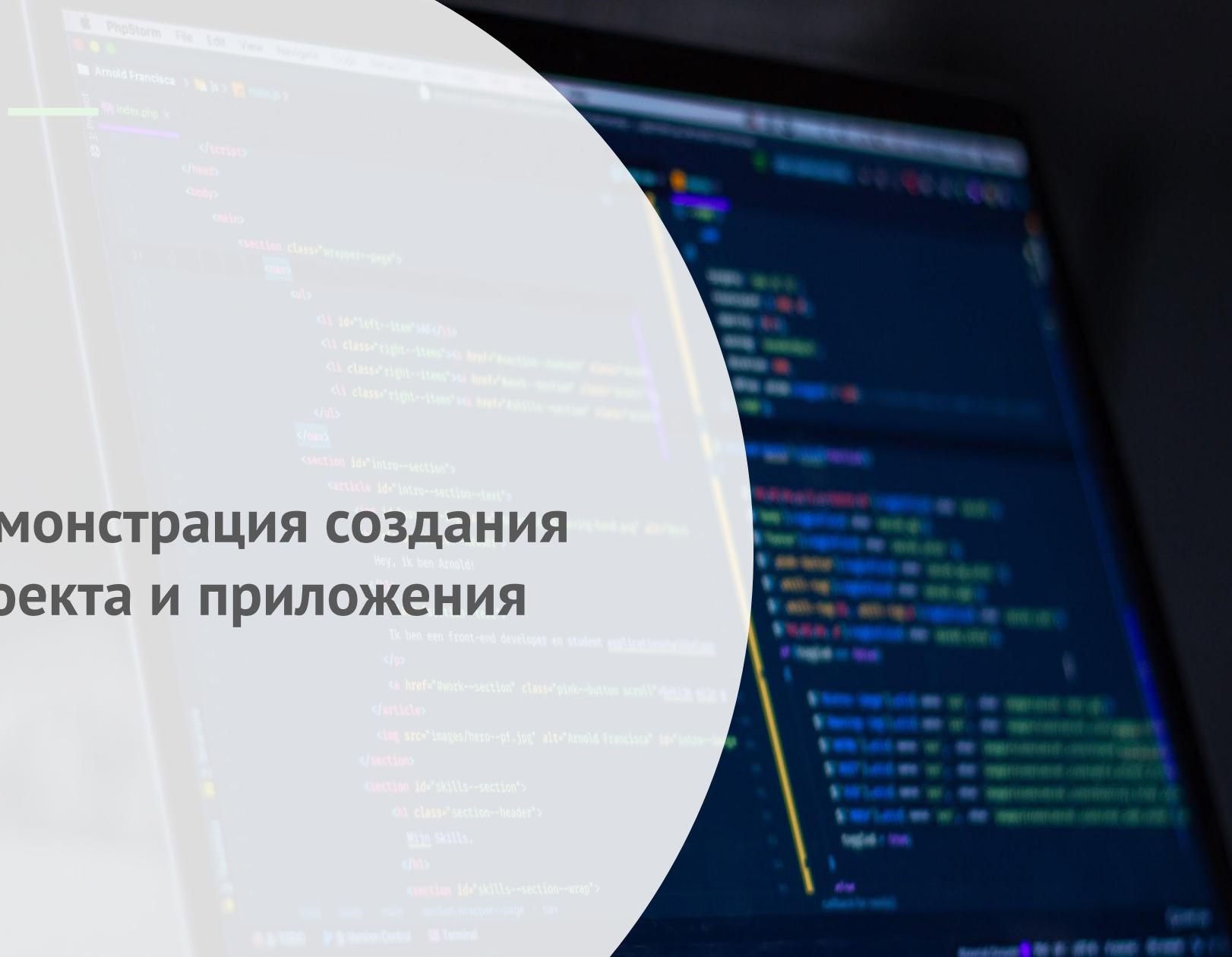
Приложение в Django – это своеобразный модуль с некоторой функциональностью. Например, приложение для работы с email, с пользователями и т. д.

Создание приложения

```
$ cd django_netology  
$ python manage.py startapp app
```

manage.py – запускает команды в контексте Django-приложения

Демонстрация создания проекта и приложения



Клиент и сервер

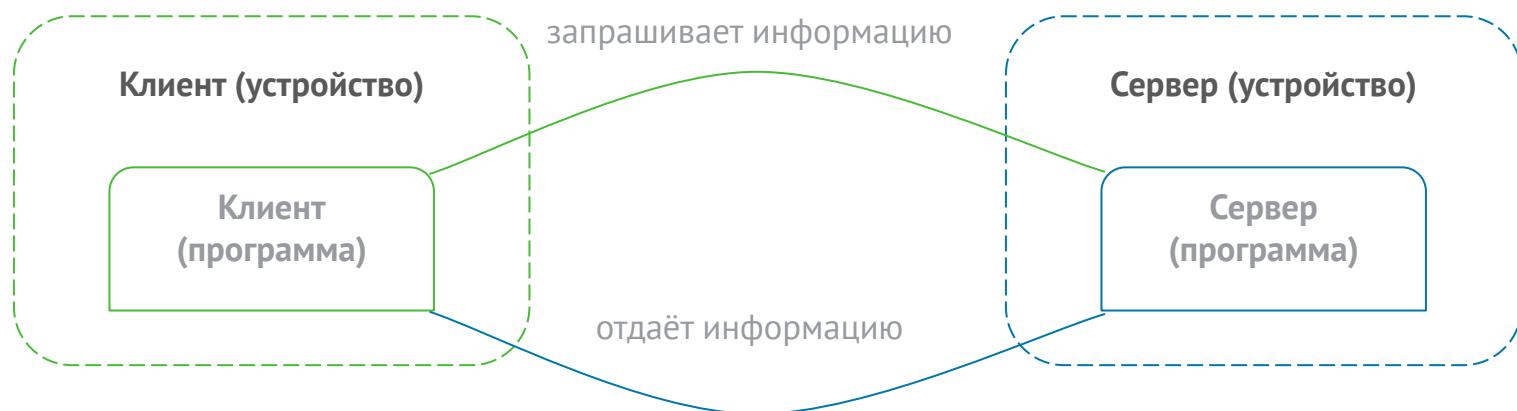
Клиент и сервер

Клиент

1. Программа, которая хочет получить информацию
2. Физическое устройство, на котором работает программа-клиент

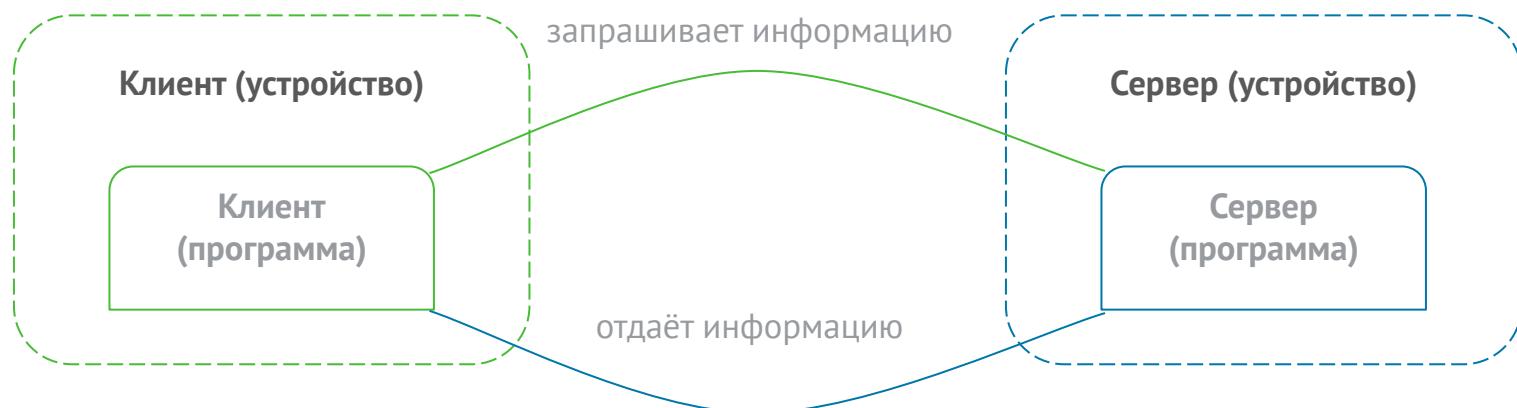
Сервер

1. Специальная программа, которая даёт информацию
2. Физическое устройство, на котором запущена программа-сервер



Клиент и сервер

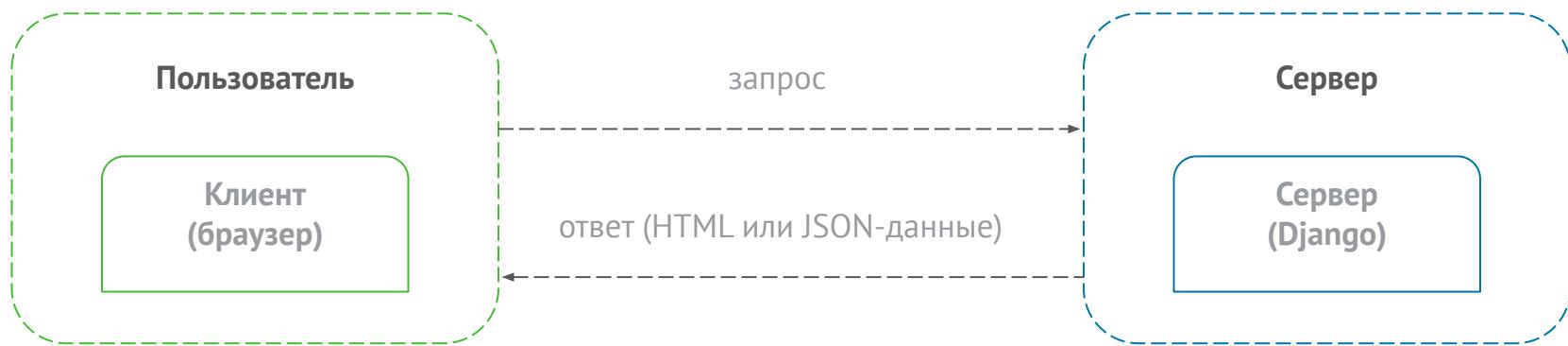
Обычно эти программы расположены на **разных вычислительных машинах** и взаимодействуют между собой по различным **протоколам**, но они могут быть расположены и на одной машине



Клиент и сервер

Веб-приложение реализует **клиент-серверное взаимодействие**.

Пользователи шлют запросы к серверу, он выдаёт им результат в виде HTML или JSON-данных



Клиент и сервер

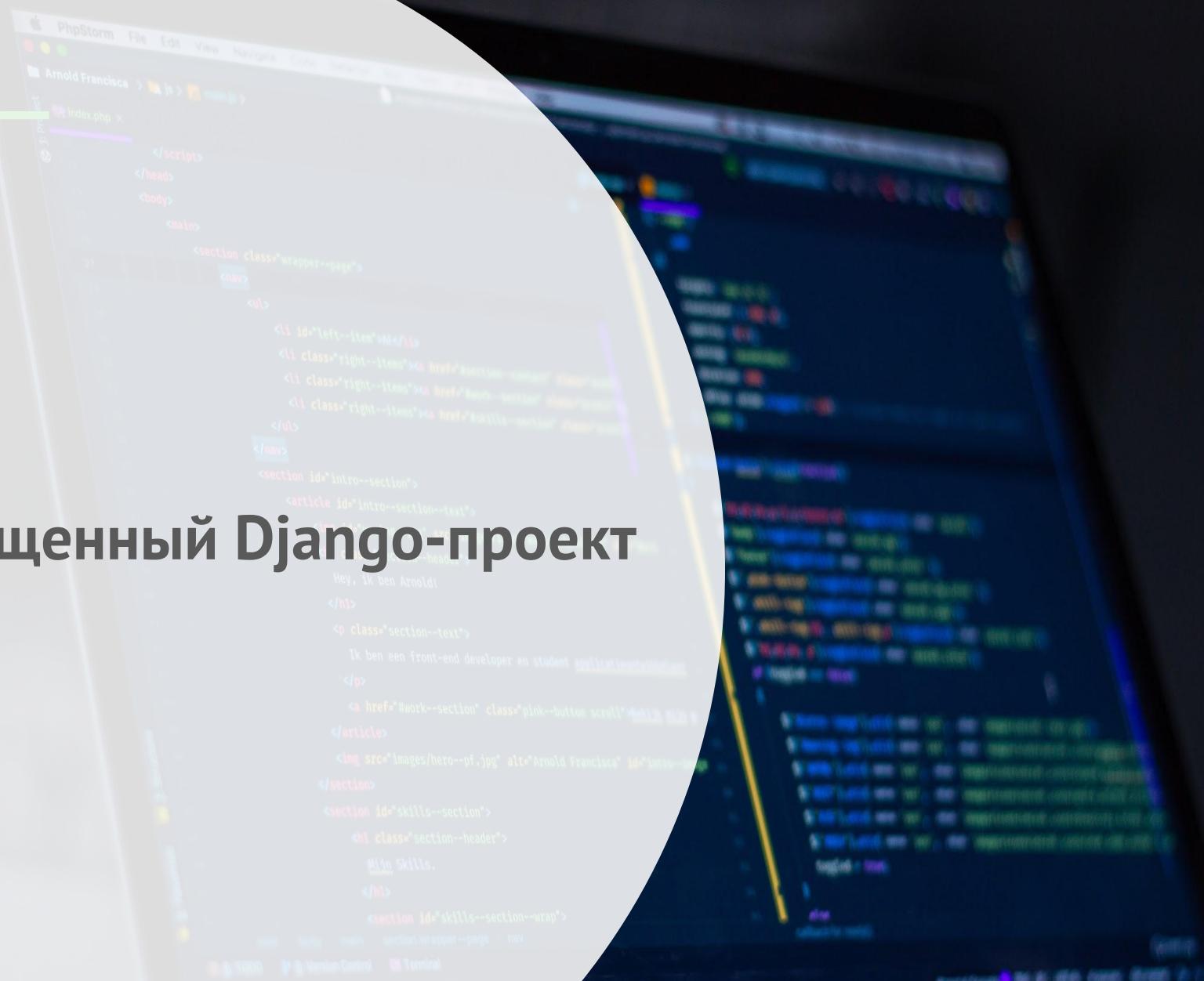
Веб-приложение реализует **клиент-серверное взаимодействие**.

Пользователи шлют запросы к серверу, он выдает им результат в виде HTML или JSON-данных.

Django-проект выступает в роли **сервера**. Для того, чтобы запустить проект, выполните команды

```
$ python manage.py migrate # создает базу данных  
$ python manage.py runserver # запускает проект
```

Запущенный Django-проект

A screenshot of a MacBook Pro screen showing a Django project setup. On the left, a code editor in PhpStorm displays the file 'Index.php' with PHP and HTML code. The code includes sections for navigation, an introduction, and skills. On the right, a terminal window shows the command 'python manage.py runserver' being run, with the output indicating the server is running at 'http://127.0.0.1:8000/'.

```
</script>
</head>
<body>
    <main>
        <section class="wrapper--page">
            <div>
                <ul id="left--item"></ul>
                <ul class="right--items"><a href="#section--about">About</a>
                    <a href="#section--work">Work</a>
                    <a href="#section--skills">Skills</a>
                    <a href="#section--contact">Contact</a>
                </ul>
            </div>
            <div>
                <h1>Hey, ik ben Arnold!</h1>
                <p>Ik ben een front-end developer en student applicatiedeveloping</p>
                <a href="#work--section" class="pink--button scroll">View my work</a>
            </div>
        </section>
        <section id="skills--section">
            <h2>My Skills.</h2>
            <div>
                <ul>
                    <li>HTML</li>
                    <li>CSS</li>
                    <li>JavaScript</li>
                    <li>React</li>
                    <li>Node.js</li>
                    <li>MongoDB</li>
                    <li>Django</li>
                    <li>Python</li>
                    <li>MySQL</li>
                    <li>Git</li>
                </ul>
            </div>
        </section>
    </main>
</body>
</html>
```

```
MacBook Pro: ~ 2023 Arnold Franciscus
```

MVC и Django

Теперь напишем что-нибудь своё

Django генерирует структуру проекта **самостоятельно**. Благодаря этому даже новые разработчики знают, где и что можно искать.

При разработке Django-приложений **очень важно** придерживаться **соглашений**.

Проекты на Django должны придерживаться **паттерна MVC**:
model-view-controller (*модель-представление-контроллер*)

Дополнительная информация про MVC: <https://habr.com/post/181772/>

Django и разделение ответственности

- управление логикой при ответе -> *view*
- как будет выглядеть страница -> *template*
- состояние приложения -> *model*

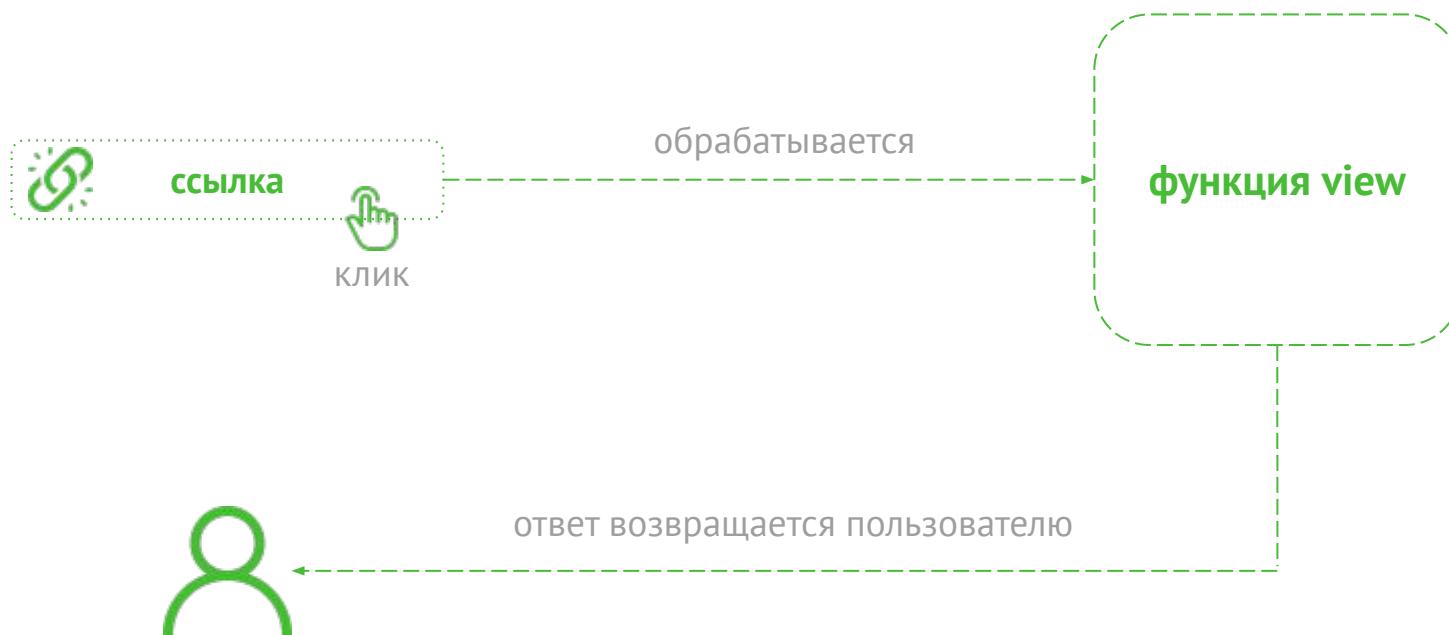
[Дополнительный материал](#)

Правило: не мешать всё в одну кучу

Работа с урлами Роутинг в Django

View и гипертекст

Пользователь взаимодействует с системой через гипертекст: клики, скролы, нажатия клавиш. Сервер реагирует на эти взаимодействия и с помощью view подготавливает новое состояние – ответ



View

base/views.py:

```
from django.http import HttpResponse
from django.shortcuts import render

def home_view(request):
    return HttpResponse('Здесь будет сайт!')
```

Необходимо добавить view-функцию в обработчик урлов

django_netology/urls.py:

```
...
urlpatterns = [
    path('', home_view, name='home'),
    ...
]
```

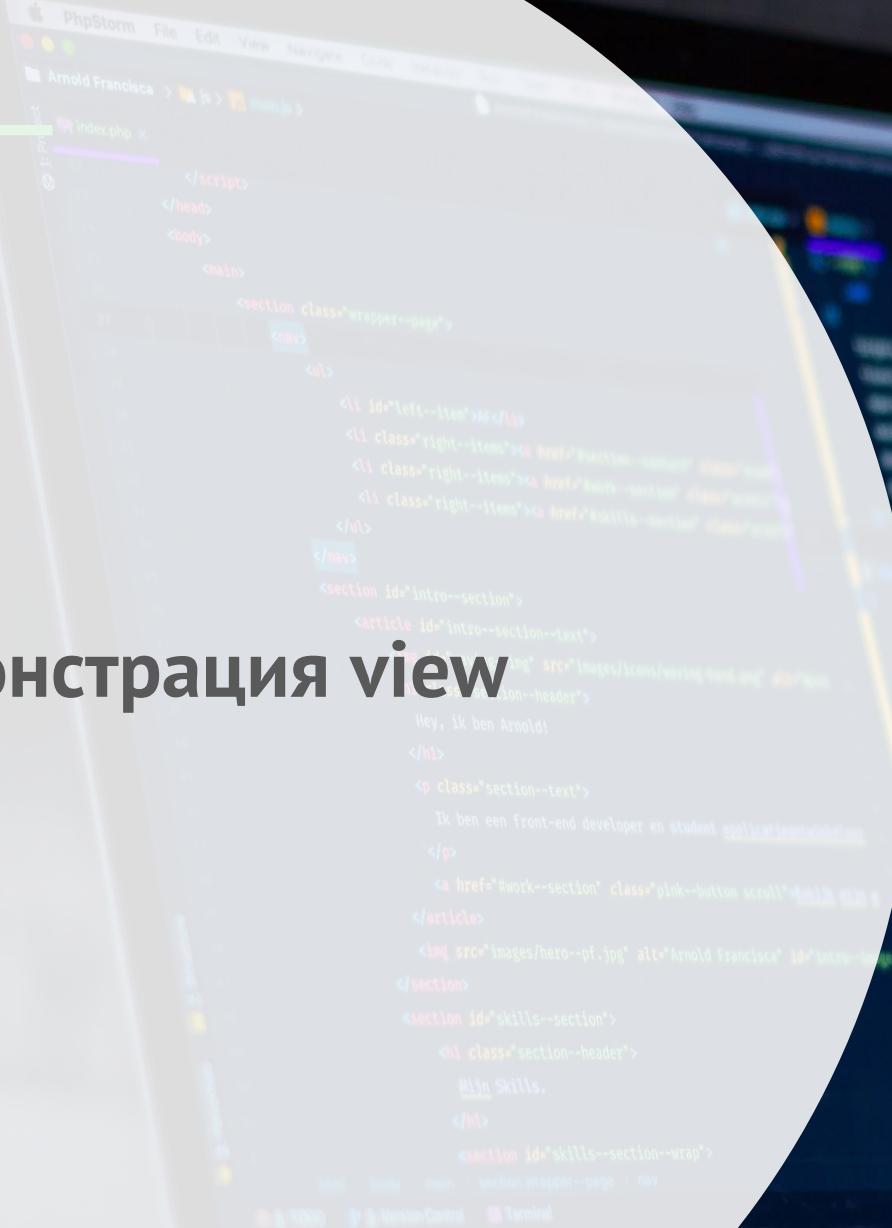
Рассмотрим структуру описания урла подробнее

Маршрутизация

```
...  
urlpatterns = [  
    path('' , home_view, name='home'),  
    ...  
]
```

- `''` (*первый параметр*) – **фактический адрес**, который будет указан в адресной строке браузера
- `home_view` (*второй параметр*) – **view-функция**, которая будет вызвана при обработке запроса
- `name` (*третий параметр*) – позволяет получать конкретный **урл по имени**. Это позволяет приложению не ломаться, если урлы будут меняться, и делает код более понятным

Демонстрация view



```
</script>
</head>
<body>
    <main>
        <section class="wrapper--page">
            <div>
                <ul id="left--item"><a href="#">Home</a>
                    <a href="#">About</a>
                    <a href="#">Skills</a>
                    <a href="#">Work</a>
                </ul>
            </div>
            <div>
                <h1 class="right--items"><a href="#section--about" class="pink--button scroll">About</a></h1>
                <h1 class="right--items"><a href="#section--skills" class="pink--button scroll">Skills</a></h1>
                <h1 class="right--items"><a href="#section--work" class="pink--button scroll">Work</a></h1>
            </div>
        </section>
        <section id="intro--section">
            <article id="intro--section--text">
                
                <h2 data-size="100px" data-align="center" data-weight="bold">Arnold Franciscus
                <h3 data-align="center">Hey, ik ben Arnold!
                <p>Ik ben een front-end developer en student applicatiedeveloping</p>
                <a href="#work--section" class="pink--button scroll">Work</a>
            </article>
            
        </section>
        <section id="skills--section">
            <h2 class="section--header">Mijn Skills.</h2>
            <div>
                <div>
                    <h3>HTML</h3>
                    <div>Progress: 100%</div>
                </div>
                <div>
                    <h3>CSS</h3>
                    <div>Progress: 95%</div>
                </div>
                <div>
                    <h3>JavaScript</h3>
                    <div>Progress: 80%</div>
                </div>
                <div>
                    <h3>React</h3>
                    <div>Progress: 70%</div>
                </div>
                <div>
                    <h3>Node.js</h3>
                    <div>Progress: 60%</div>
                </div>
                <div>
                    <h3>MongoDB</h3>
                    <div>Progress: 50%</div>
                </div>
                <div>
                    <h3>Django</h3>
                    <div>Progress: 40%</div>
                </div>
                <div>
                    <h3>Python</h3>
                    <div>Progress: 30%</div>
                </div>
                <div>
                    <h3>Java</h3>
                    <div>Progress: 20%</div>
                </div>
                <div>
                    <h3>C/C++</h3>
                    <div>Progress: 10%</div>
                </div>
            </div>
        </section>
    </main>
</body>
```

Reverse

Как получить url по имени

```
urlpatterns = [
    path('', home_view, name='home'),
    path('profile', profile_view, name='profile'),
    path('long/address/orders', orders_view, name='orders')
]
```

С помощью `manage.py shell` можно проверить работу

```
> from django.urls import reverse
> reverse('orders')
'/long/address/orders'

> reverse('profile')
'/profile'
```

Reverse

В функцию `reverse` можно также передавать **параметры**. Это нужно для формирования **динамических урлов**. Например

```
reverse('accounts', kwargs={'account_id': 100})
```

Демонстрация reverse



The screenshot shows a MacBook Pro screen with a PhpStorm IDE window open. The window displays the contents of an `index.php` file. The code is a PHP script with embedded HTML and CSS. It includes sections for navigation, introduction, and skills. A yellow arrow points from the text "Hey, ik ben Arnold!" in the code to a placeholder in the code editor's status bar.

```
<ul id="left--item"><a href="#about">About</a></ul>
<ul class="right--items"><a href="#section--home">Home</a>
<a href="#section--about">About</a>
<a href="#section--skills">Skills</a>
<a href="#section--contact">Contact</a>
</ul>
</nav>
<main>
    <section class="wrapper--page">
        <div>
            <ul id="left--item"><a href="#about">About</a></ul>
            <ul class="right--items"><a href="#section--home">Home</a>
<a href="#section--about">About</a>
<a href="#section--skills">Skills</a>
<a href="#section--contact">Contact</a>
</ul>
        </div>
        <div>
            <h1>Hey, ik ben Arnold!</h1>
            <p class="section--text">Ik ben een front-end developer en student applicatiedeveloping</p>
            <a href="#work--section" class="pink--button scroll">View my work</a>
        </div>
    </section>
    <section id="skills--section">
        <h2 class="section--header">Mijn Skills.</h2>
        <div>
            <div>
                <img alt="Placeholder for skills section content" data-bbox="300 650 450 800"/>
            </div>
        </div>
    </section>
</main>
```

Как дебажить Django-проект

Как дебажить Django-проект

print-функции

Django-проект – это Python-приложение. Поэтому можно использовать возможности Python и **использовать** принты для дебага и отладки кода

manage.py shell

Запускает **интерактивный интерпретатор** в контексте Django-проекта

Точки останова, они же *breakpoints*

Удобнее всего использовать в IDE Pycharm или VS Code

Сообщения об ошибках Django

Средство **фреймворка**. Если включен DEBUG-режим (*по умолчанию во всех домашних работах именно так*), то Django собирает и агрегирует **информацию об ошибке**

Демонстрация: точки останова

Возможность выполнять шаги в программе

Переменные доступные в функции в момент остановки

Project: dj-homeworks

views.py:

```
from django.shortcuts import render_to_response, redirect
from django.urls import reverse

def index(request):
    return redirect(reverse(bus_stations))

def bus_stations(request): request: <WSGIRequest: GET '/bus_stations'>
    current_page = 1 current_page: 1
    next_page_url = 'write your url' next_page_url: 'write your url'
    return render_to_response('index.html', context={
        'bus_stations': [{Name: 'название', Street: 'улица', District: 'район'}],
        'current_page': current_page,
        'prev_page_url': None,
        'next_page_url': next_page_url,
    })
bus_stations()
```

Debug: pagination

Frames: bus_stations, views.py:12

Variables: current_page = {int} 1
next_page_url = {str} 'write your url'
request = {WSGIRequest} <WSGIRequest: GET '/bus_stations'>

Демонстрация: информация об ошибке

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:8000/bad_url
- Page Title:** Page not found (404)
- Request Method:** GET
- Request URL:** http://localhost:8000/bad_url
- Text Content:**
 - Using the URLconf defined in app.urls, Django tried these URL patterns, in this order:
 - 1. [name='index']
 - 2. bus_stations [name='bus_stations']
 - The current path, bad_url, didn't match any of these.
- Footnote:** You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

Отладка

The image shows a MacBook Pro screen with the PhpStorm IDE open. On the left, the code editor displays the 'Index.php' file:

```
</script>
</head>
<body>
    <main>
        <section class="wrapper--page">
            <nav>
                <ul>
                    <li id="left--item"><a href="#">Home</a></li>
                    <li class="right--item"><a href="#section--about">About</a></li>
                    <li class="right--item"><a href="#section--work">Work</a></li>
                    <li class="right--item"><a href="#section--skills">Skills</a></li>
                </ul>
            </nav>
            <section id="intro--section">
                <article id="intro--section--text">
                    
                    <h1 class="section--header">
                        Hey, ik ben Arnold!
                    </h1>
                    <p class="section--text">
                        Ik ben een front-end developer en student applicatiedeveloping.
                    </p>
                    <a href="#work--section" class="pink--button scroll" data-aos="fade-up">View my work</a>
                </article>
                
            </section>
            <section id="skills--section">
                <h1 class="section--header">
                    Mijn Skills.
                </h1>
                <section id="skills--section--wrap">
                    <ul>
                        <li data-aos="fade-up"> HTML5</li>
                        <li data-aos="fade-up"> CSS3</li>
                        <li data-aos="fade-up"> JavaScript</li>
                        <li data-aos="fade-up"> React</li>
                        <li data-aos="fade-up"> Node.js</li>
                        <li data-aos="fade-up"> MongoDB</li>
                        <li data-aos="fade-up"> MySQL</li>
                        <li data-aos="fade-up"> Git</li>
                        <li data-aos="fade-up"> GitHub</li>
                    </ul>
                </section>
            </section>
        </main>
    </body>
</html>
```

The right side of the screen shows the terminal window with the following command-line interface:

```
MacBook-Pro: ~ % cd /Users/arnoldfranciscus/Sites/arnoldfranciscus.com
MacBook-Pro: ~ % git status
MacBook-Pro: ~ % ls
MacBook-Pro: ~ %
```

Отладка

Отладка – очень важный процесс. Сохраните себе эту информацию и используйте всегда при работе с домашними заданиями.

Помните, что **проект на Django** – это тот же Python-код, который выполняется интерпретатором



Что почитать

Что почитать

- <https://docs.djangoproject.com/> – официальный сайт с документацией (на английском)
- <https://developer.mozilla.org/ru/docs/Learn/Server-side/Django> – но не все статьи переведены;
- <https://tutorial.djangogirls.org/ru/> – проект Django Girls, но полезно будет всем.



Итоги



Чему мы научились

- **Научились** устанавливать Django и создавать проект и приложение
- **Узнали**, как писать простые страницы
- **Научились** дебажить Django-проект



Задавайте вопросы и
пишите отзыв о лекции