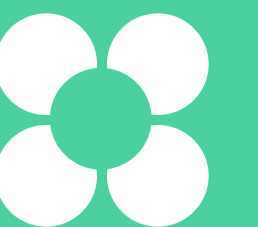


Знакомство с Django. Подготовка и запуск проекта



План занятия

- 1 Django: установка и запуск
- 2 Клиент и сервер
- 3 MVC и Django
- 4 Базовые возможности. Роутинг в Django
- 5 Дебаг проекта
- 6 Что почитать

Django: установка и запуск

Что такое Django

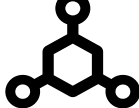
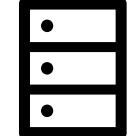



Django — фреймворк для быстрого создания веб-приложений, полностью написанный на Python.

Django — очень популярный проект и используется многими крупными компаниями.

Исходный код Django доступен на [Github](#)



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
```

Для установки Django выполните команду в консоли:

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

Что такое проект и приложение

Под проектом можно понимать полноценный сайт. Это:

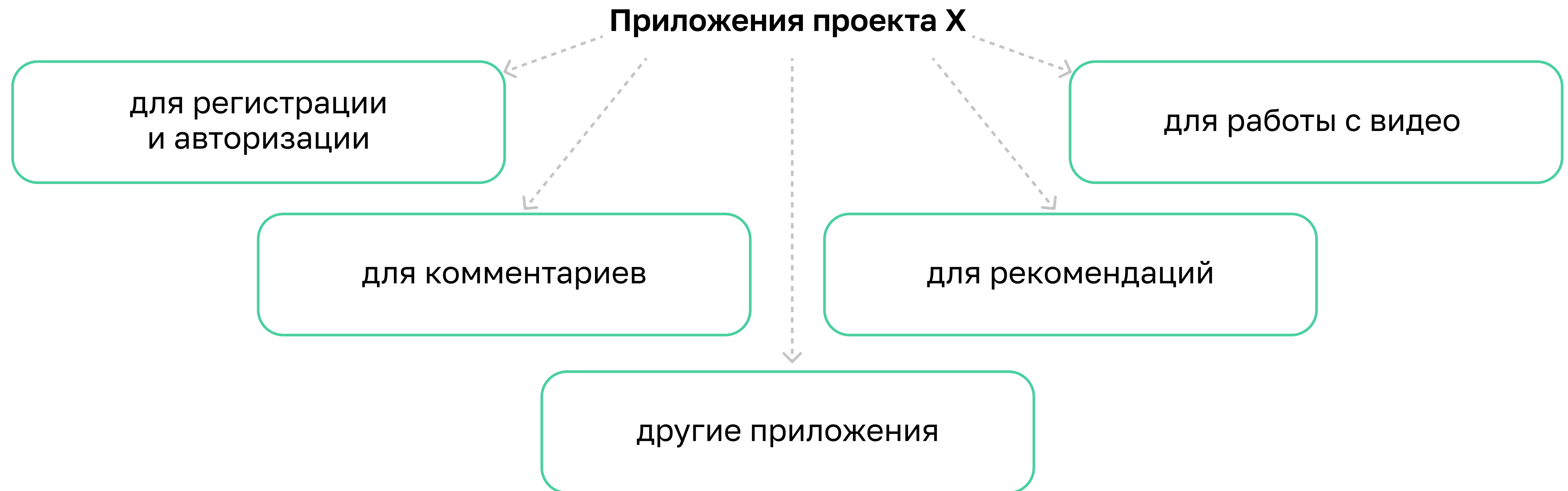
- коллекция настроек
- база данных
- подключённые приложения

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



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

Клиент и сервер



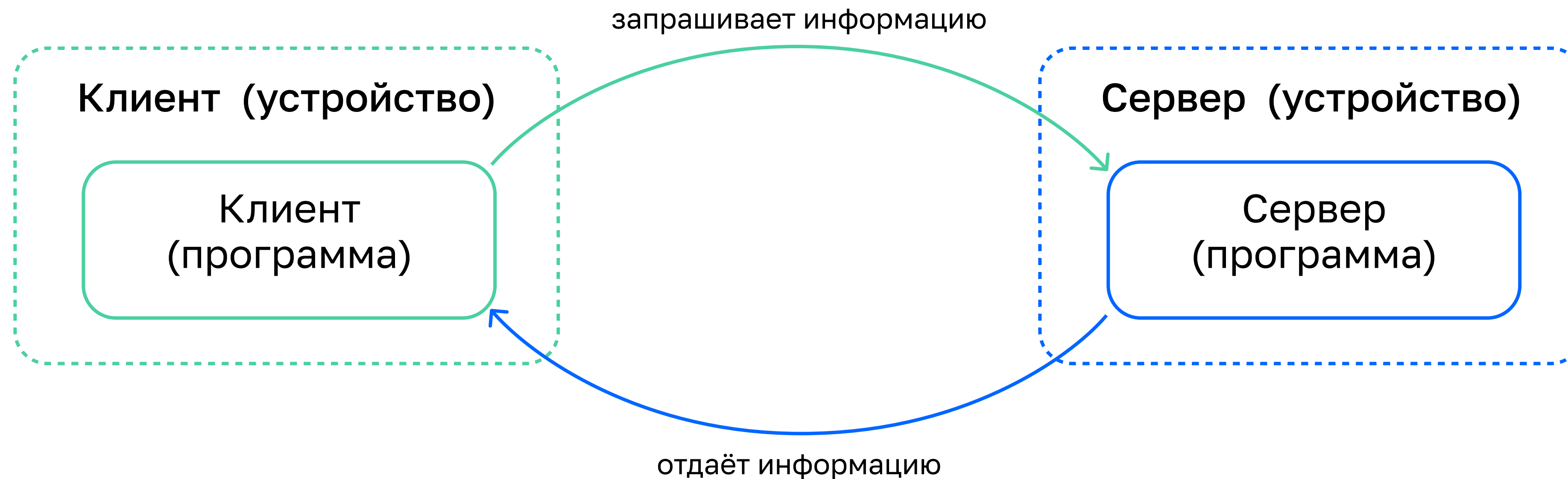
Клиент и сервер

Клиент:

- 1 программа, которая хочет получить информацию
- 2 физическое устройство, на котором работает программа-клиент

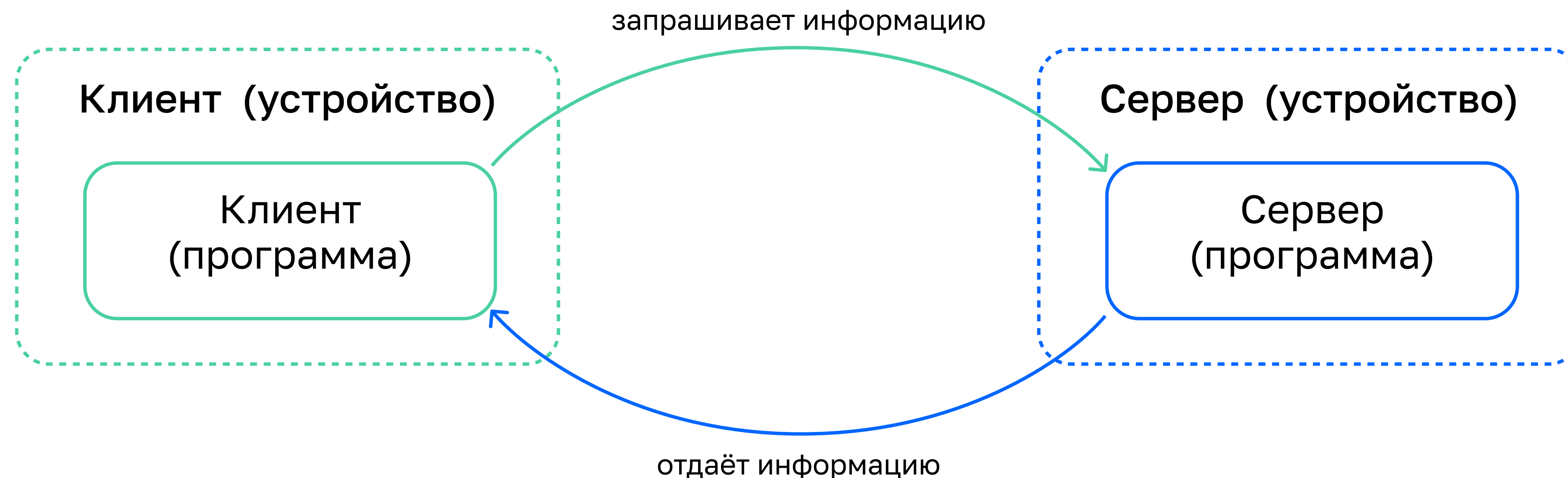
Сервер:

- 1 специальная программа, которая даёт информацию
- 2 физическое устройство, на котором запущена программа-сервер



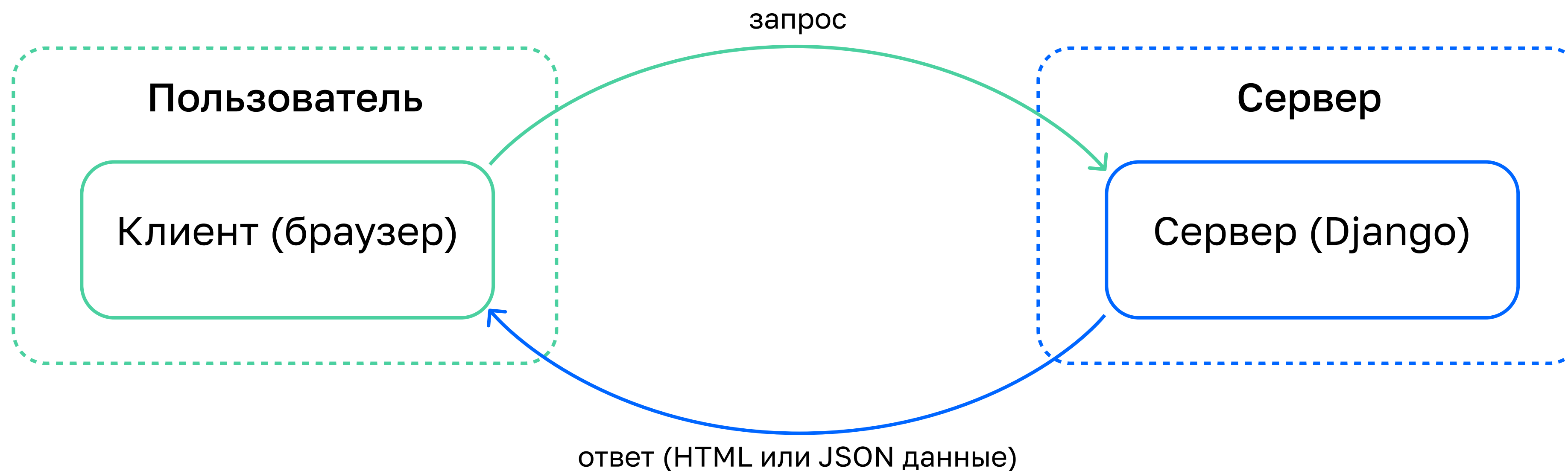
Клиент и сервер

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



Клиент и сервер

Веб-приложение реализует **клиент-серверное взаимодействие**. Пользователи шлют запросы к серверу, он выдаёт им результат в виде данных html или JSON



Клиент и сервер

Веб-приложение реализует **клиент-серверное взаимодействие**. Пользователи шлют запросы к серверу, он выдаёт им результат в виде данных html или JSON.

Django-проект выступает в роли **сервера**. Чтобы запустить проект, выполните команды:

```
$ ./manage.py migrate # создаёт базу данных  
$ ./manage.py runserver # запускает проект
```


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)

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

Базовые возможности и роутинг в Django

View

base/views.py:

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

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

Необходимо добавить view-функцию в обработчик URL [django_netology/urls.py](#):

```
...

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

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 URL по имени. Это позволяет приложению не ломаться, если URL будет меняться, и делает код более понятным

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

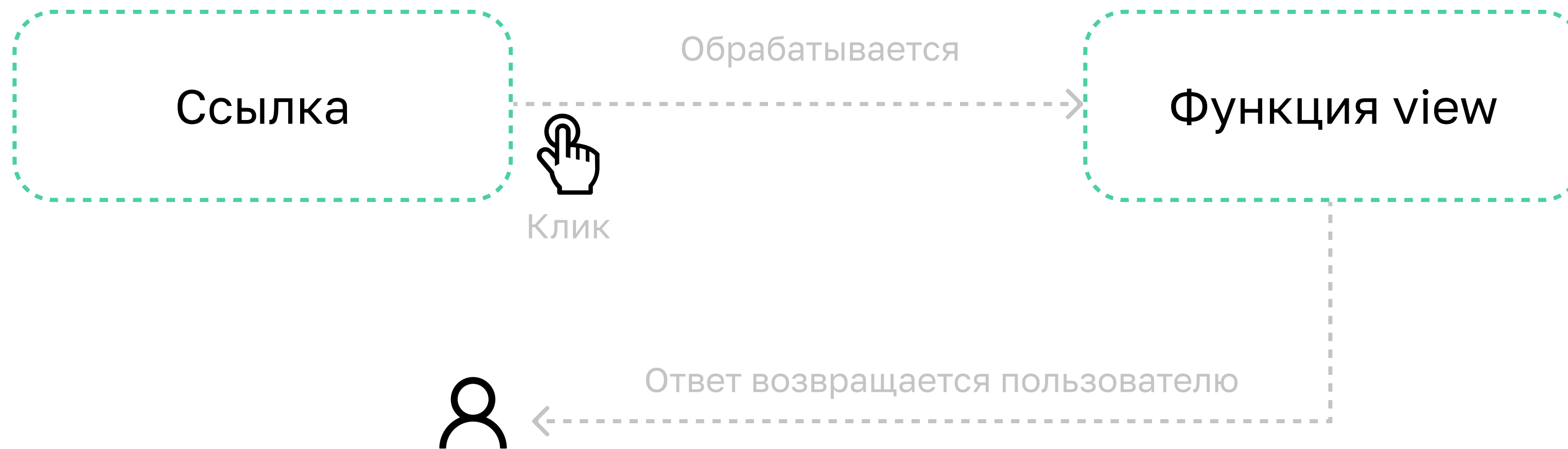
В функцию `reverse` можно также передавать параметры. Это нужно для формирования **динамических URL**. Например:

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

Об этом мы будем говорить подробнее на лекции про обработку запросов

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



Дебаг проекта



Как дебажить Django-проект

Print-функции

Django-проект — это Python-приложение. Поэтому можно использовать возможности Python и `print` для дебага и отладки кода

`manage.py shell`

Запускает интерактивный интерпретатор в контексте Django-проекта

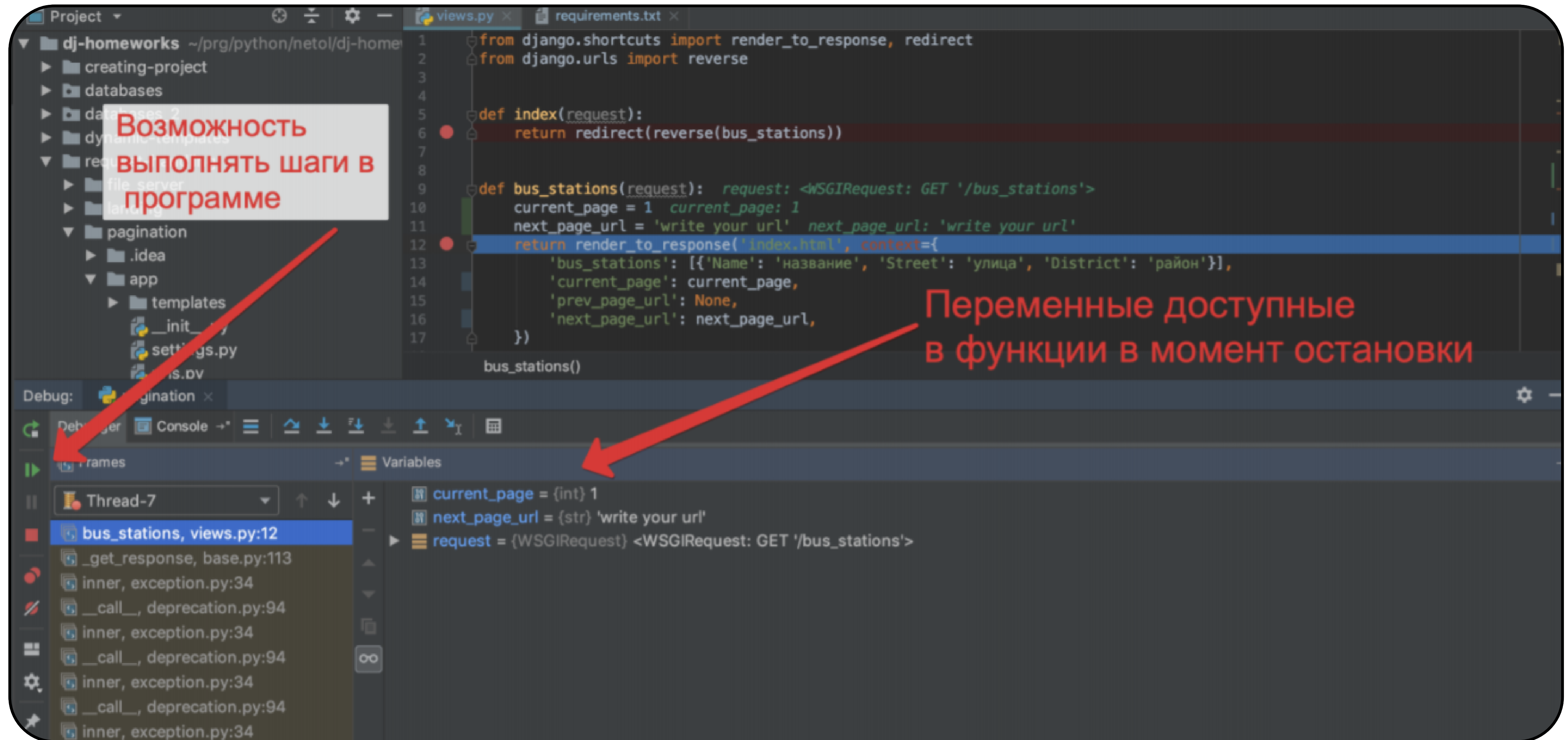
Точки останова (они же `breakpoints`)

Удобнее всего использовать в IDE PyCharm или VS Code

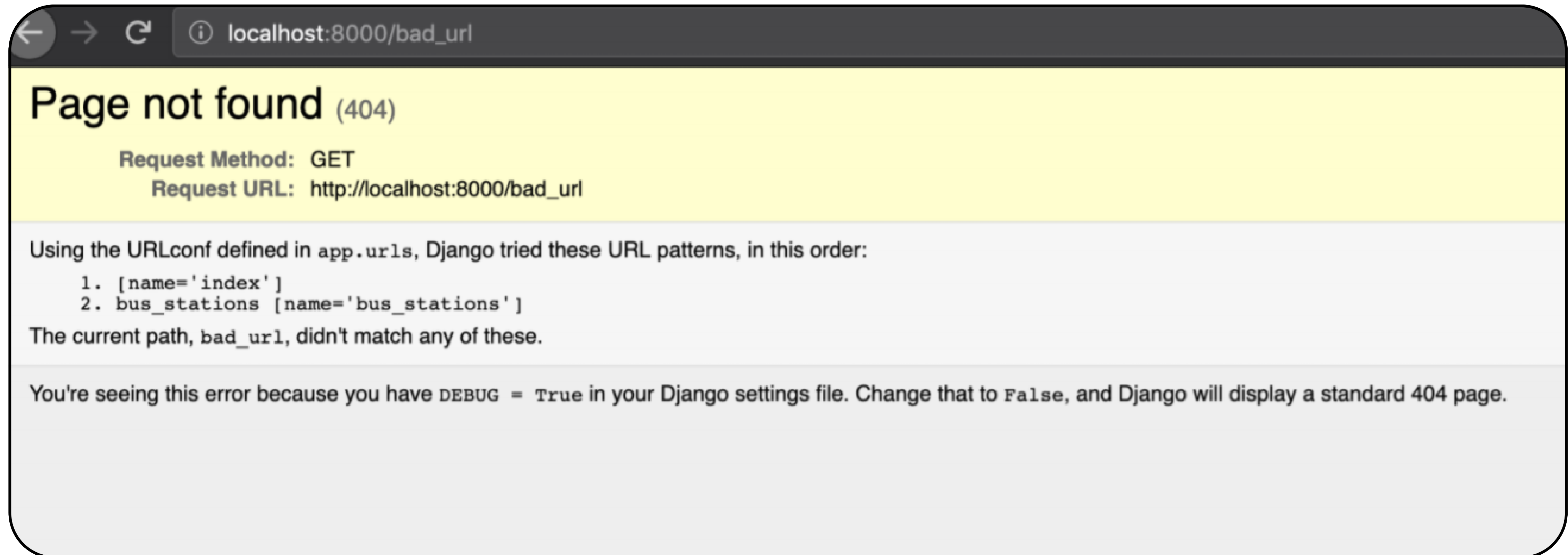
Сообщения об ошибках Django

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

Демонстрация: точки останова



Демонстрация: информация об ошибке



Отладка

Отладка — очень важный процесс. Сохраните себе эту информацию и используйте всегда в домашних работах.

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



Что почитать



Что почитать

- Официальный сайт с [документацией](#) (английский)
- [Веб-фреймворк Django](#), но не все статьи переведены
- Проект [Django Girls](#) будет полезен всем