GeekBrains

Основы Django

Подготовительная работа

На этом уроке

1. Узнаем, почему стоит вкладывать силы в изучение каркаса Django
2. Рассмотрим основы клиент-серверного взаимодействия и познакомимся с протоколом HTTP
3. Познакомимся с окружением разработчика

Оглавление

На этом уроке **1**

Оглавление **2**

Почему Django? **3**

Интернет как явление 3

Язык Python 4

Django vs Flask 4

Python + Django 5

Клиент-серверное взаимодействие **5**

Протокол HTTP 5

HTTP-сообщения 6

Путь к ресурсу. URL 10

Django и паттерн MVT 11

Окружение разработчика **12**

Операционная система 12

Гипервизоры 14

Виртуальное окружение python 17

Запуск статичного сайта 18

Заключение **25**

Практическое задание **25**

Глоссарий **26**

Дополнительные материалы **27**

Используемые источники **27**

Почему Django?

Интернет как явление

В 1984 году компьютеров, подключенных к сети Интернет, было всего около тысячи. К началу 2001 года их число превысило 90 млн. Для сравнения: радио потребовалось 38 лет, чтобы набрать 50 миллионов пользователей, а интернету — всего 4 года.

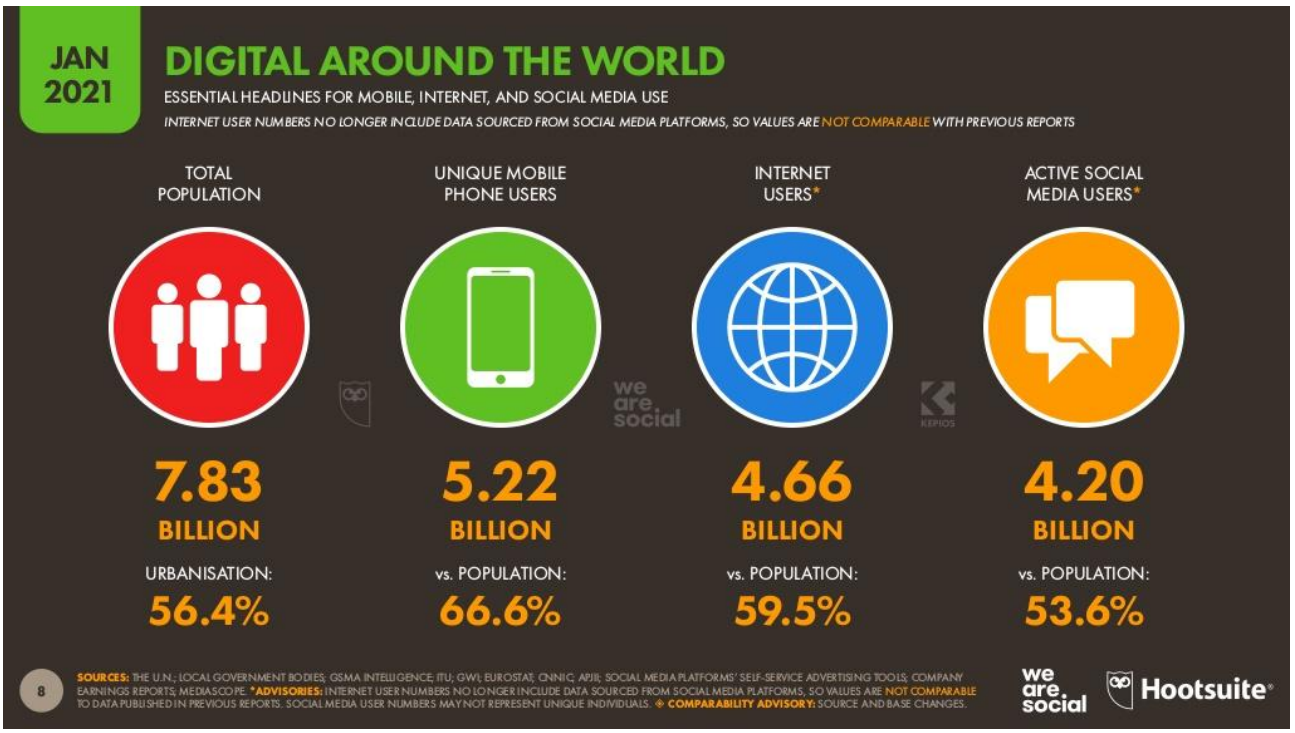

Рис. 1. Статистика использования сети Интернет на начало 2021 года

Согласно отчету на wearesocial.com, по состоянию на начало 2021 года общее количество пользователей интернета уже составляет 4.66 миллиардов. И это число продолжает расти. Пандемия, которая накрыла мир в 2020 году, показала, что современная цивилизация может выстроить многие аспекты взаимодействия между людьми через сеть. Это подтолкнуло развитие технологий: появление мобильных сетей пятого поколения, массовое введение домашних маршрутизаторов с технологией Wi-Fi 6, развитие спутникового интернета сети StarLink.

Интернет становится главной точкой коммуникации благодаря развитию сервисов интернет-торговли, бизнес-площадок, мессенджеров, цифровых сервисов здравоохранения, видеохостингов, сервисов генерации пользовательского контента, социальных сетей. Если человек или компания хочет заявить о себе, то без присутствия в интернете не обойтись.

Язык Python

За время существования компьютерных наук было создано множество самых разных языков программирования: одни нужны для создания абстракции от аппаратного обеспечения (язык C), другие — в академических целях (например, Pascal), а третьи являются языками специального назначения (например, COBOL).

Языки низкого уровня или специального назначения для развития интернета не подошли из-за сложности и множества возникающих проблем. Написание простого интернет-магазина на таких языках требует от специалиста понимания многих областей знания: от электротехники до тонкостей обработки данных на конечных устройствах. Разработка простых задач стоила бы огромных временных и денежных затрат.

Затем появились удобные для программистов языки, предоставляющие высокий уровень абстракции: Java, PHP, Golang и Python.

Благодаря своей простоте и поддержке сообщества язык общего назначения Python пополнился большим количеством библиотек для решения разнообразных задач: математика, нейронные сети, работа с офисными документами, обработка изображений, задачи генеративной архитектуры и создание игр.

В области веб-разработки широкое распространение получили каркасы (frameworks) Django, Flask, Bottle, FaspAPI, Tornado. Наиболее популярными считаются именно Django и Flask: их очень часто сравнивают между собой.

Django vs Flask

Каркас Django был создан для построения веб-приложений на языке Python. Об этом фреймворке принято говорить «все батарейки включены». Это значит, что он предоставляет разработчику полный инструментарий для создания приложений:

- ORM — система абстракции над базой данных
- Встроенный административный интерфейс
- Расширяемая система шаблонов
- Система кэширования
- Интернационализация
- Система авторизации и аутентификации
- Система генерации и обработки форм

Фреймворк Flask гораздо примитивнее. Он предоставляет только базовые возможности для разработки веб-приложений:

- Система обработки запросов от клиентской стороны

- Генерация HTML-страниц для ответа (с помощью сторонней библиотеки Jinja2)

Flask часто называют микро-фреймворком. Разработчик сам решает, какой функционал должен быть в приложении и устанавливает необходимые модули по мере необходимости.

В этом и состоит основное отличие двух фреймворков.

Python + Django

Благодаря большому количеству различных библиотек, простоте использования Python и широкому инструментарию фреймворка, Django взяли на вооружение многие крупные компании. На его основе были построены такие интернет-ресурсы как:

- Instagram
- Pinterest
- Сайт Mozilla
- Сайт журнала National Geographic
- И др.

Такое широкое распространение — показатель зрелости фреймворка. Иными словами, он является production-ready, то есть его можно использовать для решения серьезных бизнес-задач. В связке с дополнительными библиотеками Python-фреймворк дает практически неограниченные возможности для реализации самых смелых идей.

Клиент-серверное взаимодействие

При разработке веб-приложений необходимо понимать, что такое клиент и сервер, как они взаимодействуют и какое участие в этом принимает Django.

Протокол HTTP

Протокол — это правила общения между конечными устройствами. Существует множество протоколов: FTP \ sFTP, HTTP \ HTTPs, SSH, Telnet, TCP, WebSocket и т.д.

Разные протоколы предназначены для разных задач. Например, задействуя протокол SSH, можно подключиться к удалённом компьютеру для установки программного обеспечения, а используя FTP или sFTP — передавать файлы между компьютерами.

Протокол передачи гипертекста (Hypertext Transfer Protocol) предназначен для передачи гипертекстовых документов, таких как HTML. Он следует классической клиент-серверной модели, когда клиент открывает соединение для создания запроса, а затем ждет ответа.

Важно! HTTP — это **протокол без сохранения состояния** (stateless), то есть сервер не сохраняет никаких данных между парами «запрос-ответ».

В клиент-серверной архитектуре задания или сетевая нагрузка распределены между поставщиками услуг (серверами) и заказчиками услуг (клиентами). Фактически клиент и сервер — это программное обеспечение. Браузер всегда является клиентской программой. Однако клиентами могут быть и другие программы. Например, при обмене данными между приложением на мобильном устройстве и сервером, который хранит данные: банковские приложения, онлайн-игры и т.д., это приложение является клиентом. А сервер принимает входящие сообщения от клиентов, обрабатывает их и отправляет ответ. Принципиальное отличие состоит в том, что сервер никогда не не отправляет сообщение первым в рамках HTTP-протокола.

HTTP-сообщения

Все примеры будут демонстрироваться в браузере FireFox. В других браузерах могут быть незначительные отличия в отображении данных, но скорее всего все будет совпадать.

Рассмотрим сообщения HTTP-протокола:

1. Открываем браузер
2. Открываем панель разработчика (клавиша F12)
3. Переходим во вкладку «Сеть»
4. Вводим в адресную строку адрес https://gb.ru
5. Из списка запросов выбираем верхнюю строку
6. В правом окне выбираем вкладку «Заголовки» и выставляем переключатели в режим «Необработанный»

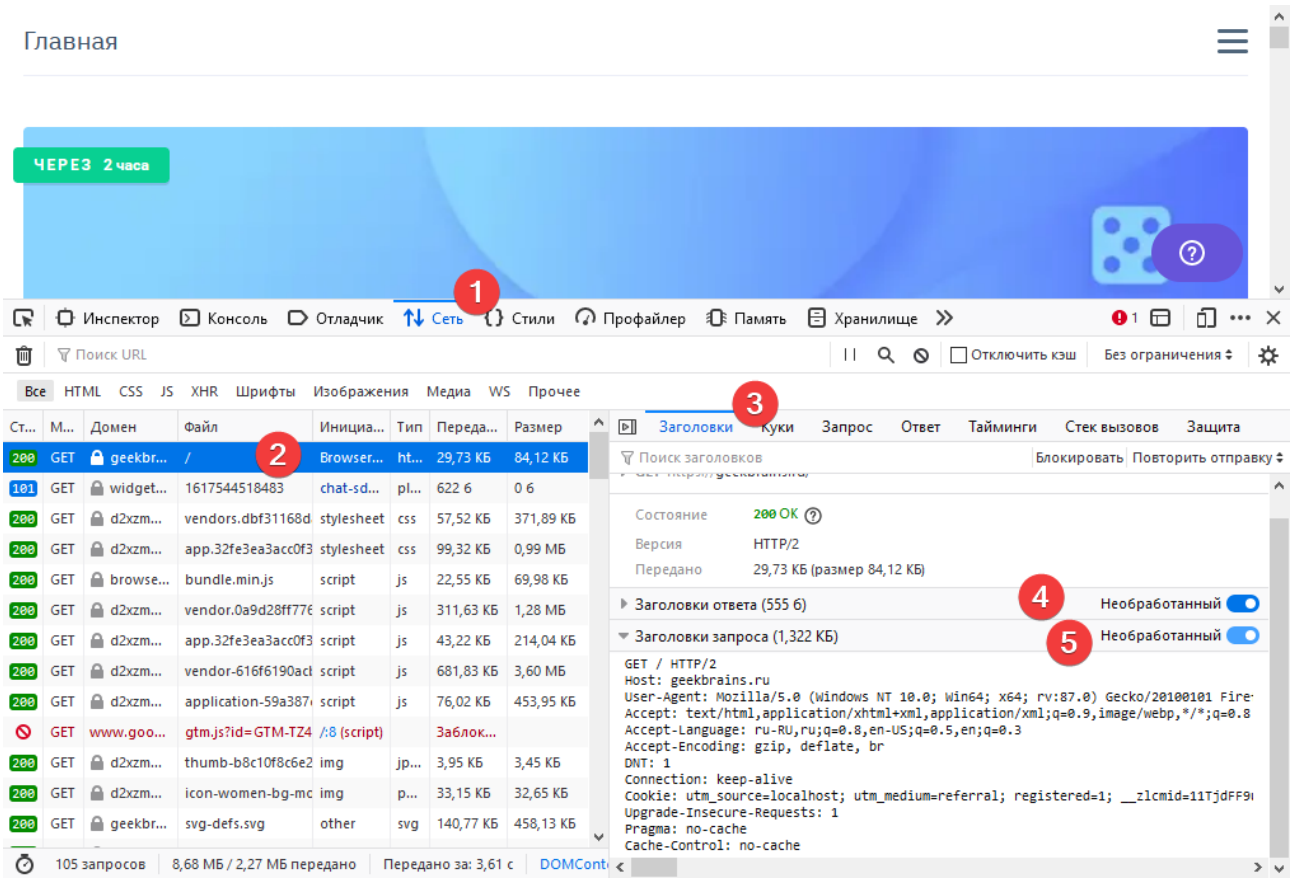

Рис. 2. 1.

Когда мы переводим ползунок параметра «Необработанный» в активное положение, видим тело сообщения HTTP-протокола. Браузер формирует его и отсылает на сервер.

Давайте рассмотрим структуру сообщения запроса «Заголовки запроса»:

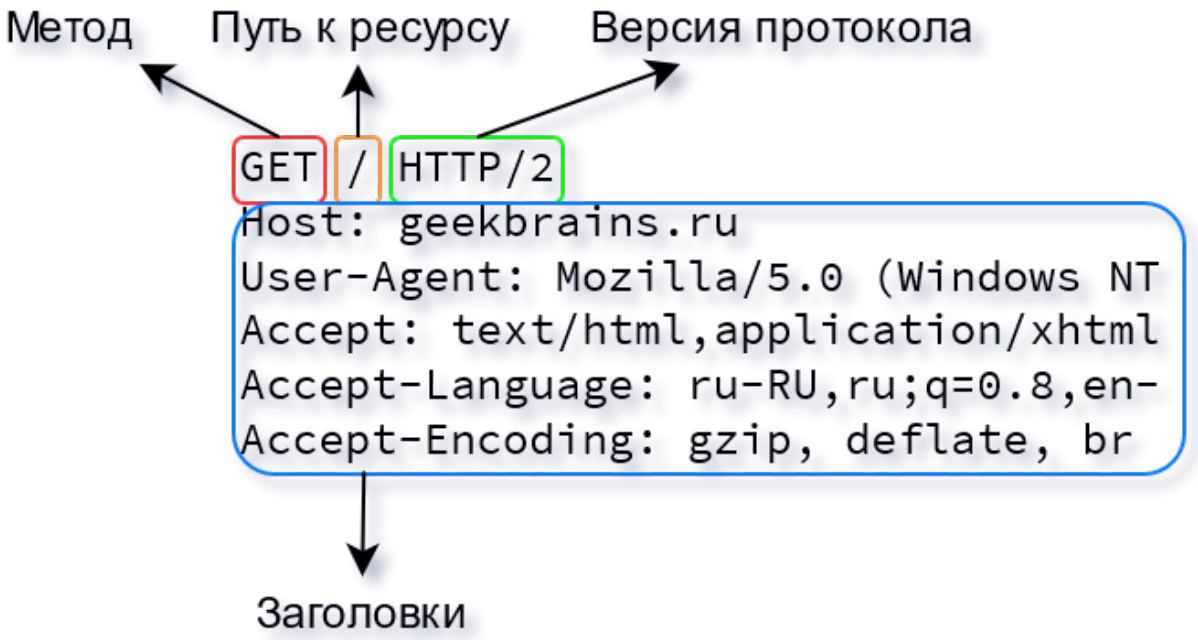

Рис. 3. Структура HTTP-запроса

Запрос состоит из нескольких строк. Первая строка содержит:

1. Метод, определяющий, какую операцию хочет выполнить клиент. На этом курсе мы рассмотрим всего два метода: GET и POST. Однако методом может быть любая последовательность символов. Главное условие: сервер должен понимать метод, с помощью которого идет обращение.
2. Путь к ресурсу указывает полную строку той страницы или адреса, к которому идет обращение, без указания явных элементов (протокола (http://), домена (geekbrains.ru) и TCP порта (в данном случае 80)).
3. Версия протокола.

Сообщение обычно содержит в конце дополнительную пустую строку.

Актуальные версии протокола: 1.0, 1.1, 2.0. Есть еще экспериментальная версия 3.0, которая сейчас готовится к выходу. Об их различиях вы можете узнать из дополнительных материалов.

Заголовки сообщения содержат служебную информацию, необходимую для согласования работы с сервером. HTTP-протокол не сохраняет состояния между парами «запрос-ответ». А значит, чтобы распознать, с какого именно клиента пришел запрос, необходимо включить в заголовки всю необходимую информацию. Однако клиентов может быть сотни тысяч. И в этом кроется главный недостаток HTTP-протокола: на сервер с каждым запросом приходит большое количество данных, которые необходимо обработать. Благодаря заголовкам можно управлять кэшированием на стороне клиента, проводить базовую аутентификацию, создавать сессии и пр.

Если HTTP-протокол не сохраняет состояние, то как сайты понимают, что к ним пришёл тот же пользователь? Это осуществляется с помощью механизма куки: сервер в ответе посылает идентификатор, который клиент сохраняет на своей стороне и при каждом последующем запросе добавляет в запрос. Таким образом создается видимость сохранения состояния.

Теперь рассмотрим сообщение ответа:

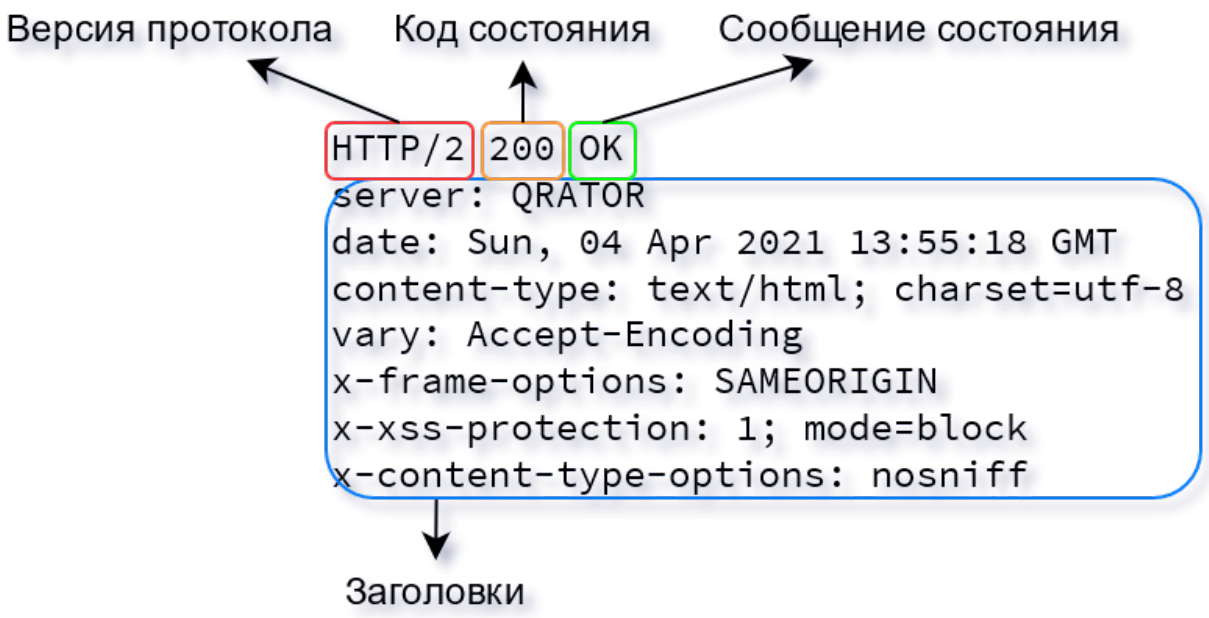

Рис. 4. Структура HTTP-ответа.

В первой строке ответа от сервера содержится:

1. Версия HTTP-протокола
2. Код состояния, сообщающий об успешности запроса или причине неудачи
3. Сообщение состояния — краткое описание кода состояния

Наиболее часто встречающиеся коды состояний:

- 200 - успешно
- 301 - перенаправление
- 403 - доступ запрещён
- 404 - объект не найден
- 500 - внутренняя ошибка сервера

Полный список кодов состояний можно посмотреть в справочных материалах или в спецификации протокола.

Заголовки несут в себе служебную информацию. Например, в Django через них происходит управление интернационализацией ресурса (переключение языка).

Опционально к ответу можно добавить тело ответа (пересылаемый ресурс). В случае запроса интернет-страницы в теле будет содержаться HTML-документ.

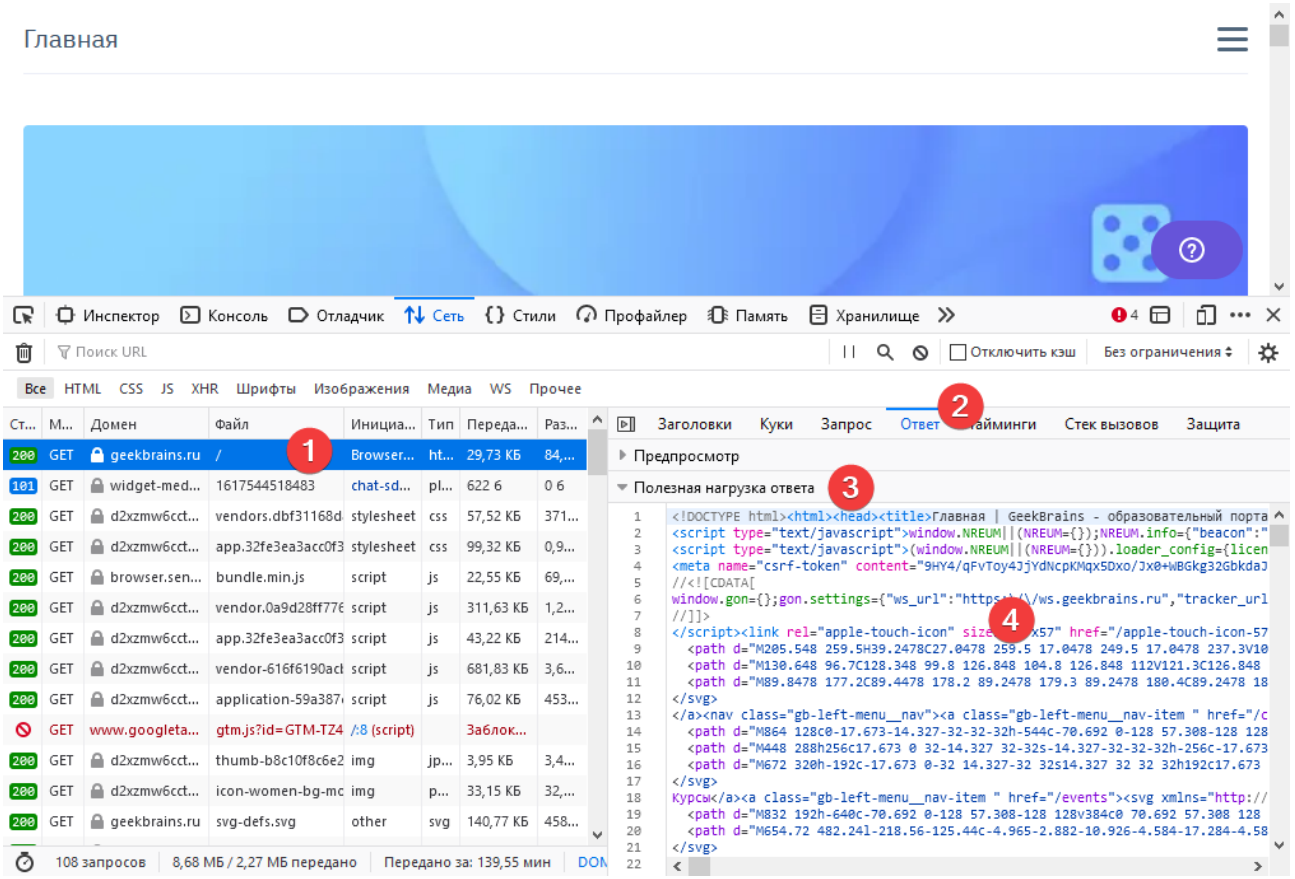

Рис. 5. 1. Выбираем первый запрос из списка 2. Выбираем вкладку «Ответ» 3. Раскрываем вкладку «Полезная нагрузка ответа» 4. HTML-документ из тела сообщения

Путь к ресурсу. URL

Текст, который вводят в адресную строку называют «адресом» или URL. URL (Uniform Resource Locator) — система унифицированных адресов электронных ресурсов или единообразный определитель местонахождения ресурса. Ресурсом может быть всё, что угодно: файл, изображение, сформированный HTML-документ и т.п. Стандарт, определяющий формирование URL — RFC3986 (ссылка в дополнительных материалах). URL не является уникальным для HTTP-протокола. То есть подобные URL могут использоваться и в других протоколах.

Точкой входа в любой фреймворк будет система маршрутизации. В Django тоже есть система диспетчеров адресов или модулей, которые перенаправляют запрос на соответствующий обработчик:

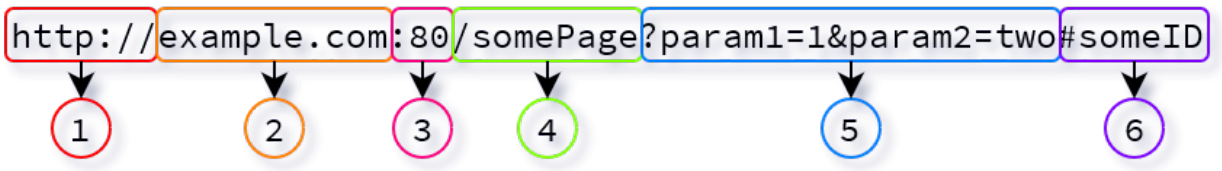

Рис. 6. Схема URL.

На рисунке (Рис. 6) представлена схема URL, которая состоит из:

1. Схемы обращения к ресурсу. Обычно указывается сетевой протокол.
2. Хоста. Это полностью прописанное доменное имя хоста в системе DNS или его IP-адрес в форме четырех групп десятичных чисел, разделенных точками.
3. Порта. Порт 80 задан для протокола HTTP по умолчанию и дополнительно его указывать не нужно. Для протокола HTTPs по умолчанию используется порт 443.
4. URL-пути. Уточняющая информация о месте нахождения ресурса. Зависит от протокола.
5. GET-параметров. Строка запроса с параметрами, которые передаются на сервер методом GET. Начинается с символа ?, разделитель параметров — знак &.
6. Идентификатора якоря. Якорем может быть заголовок внутри документа или его атрибут id. По такой ссылке браузер откроет страницу и переместит окно к указанному элементу.

Иногда между указателем протокола и хостом может быть добавлена пара логин-пароль для авторизации. Например, ftp://username:password@example.com/.

Сочетание хоста и порта еще называют «адресной парой». В дальнейшем на адресной паре 0.0.0.0:8000 мы будем запускать отладочный Django сервер. Хост 0.0.0.0 означает, что сервер будет доступен со всех сетевых интерфейсов. Это удобно, когда нужно продемонстрировать свои наработки коллегам, которые находятся в той же локальной сети.

Примеры URL:

- https://www.google.com/search?client=firefox-b-d&q=django
- https://docs.djangoproject.com/en/3.1/search/?q=url

Django и паттерн MVT

После того, как вы ввели в адресную строку браузера URL и запустили процедуру перехода, по указанному адресу отправляется HTTP-сообщение о запрашиваемом ресурсе. Задача Django — обработать пришедшее сообщение и сформировать ответ. Обработка сообщения внутри Python-кода практически линейна, и, для того, чтобы она легче воспринималась, её разносят по разным модулям, каждый из которых ответственен за свою часть. Это процесс обычно называют *паттерном проектирования*.

В рамках Django используется паттерн MVT:

- M = models. Модели — это часть кода, которая будет взаимодействовать с базой данных.
- V = view. Отображение, представление, контроллер, *вьюшка* — это часть кода, которая обрабатывает запрос и формирует ответ. В контроллерах выполняется основная работа по подготовке данных для последующего отображения.

- T = template. Шаблоны обычно представляют собой HTML-файл или вёрстку, написанные с использованием дополнительного синтаксиса. Фактически шаблоны — это то, что в итоге увидит пользователь.

Схема обработки запроса с точки зрения Django и с использованием паттерна MVT представлена на Рис. 7.

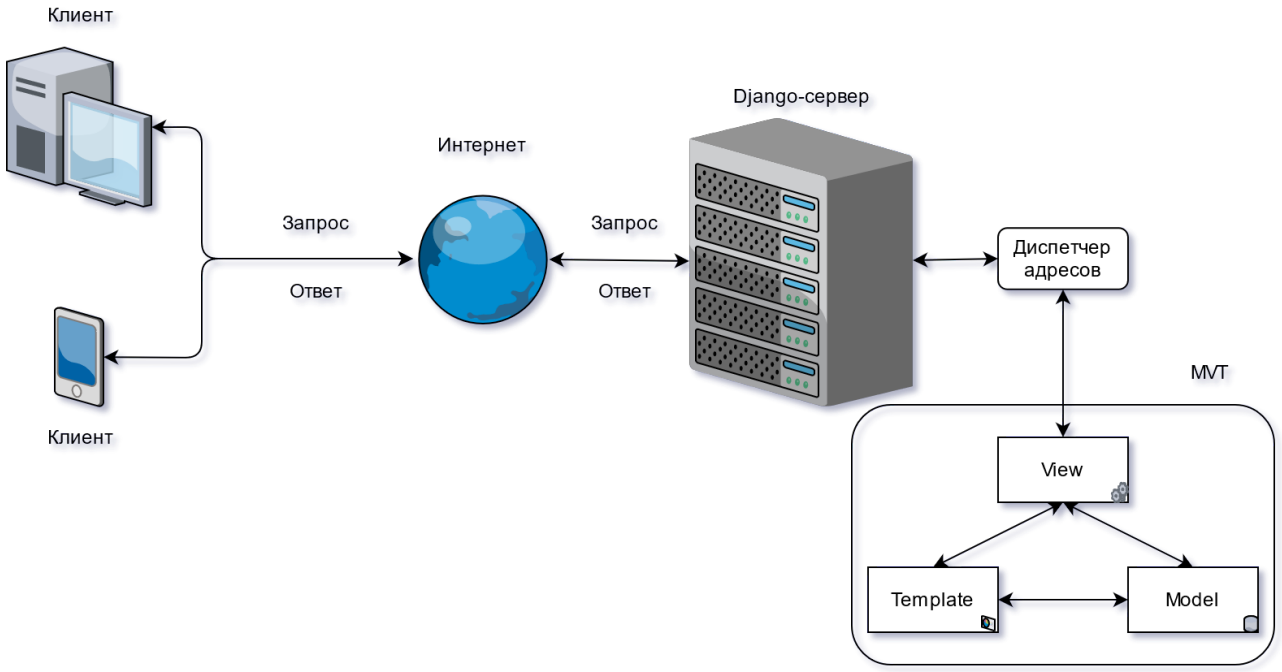

Рис. 7. Обработка запроса внутри фреймворка Django с использованием паттерна MVT.

Окружение разработчика

Операционная система

После того, как вы ввели URL в адресной строке и отправили запрос, браузер автоматически формирует HTTP-сообщение и направляет его на сервер. Затем всю работу делает операционная система. А чтобы сервер Django принял запрос на вход, он должен быть предварительно обработан серверной операционной системой.

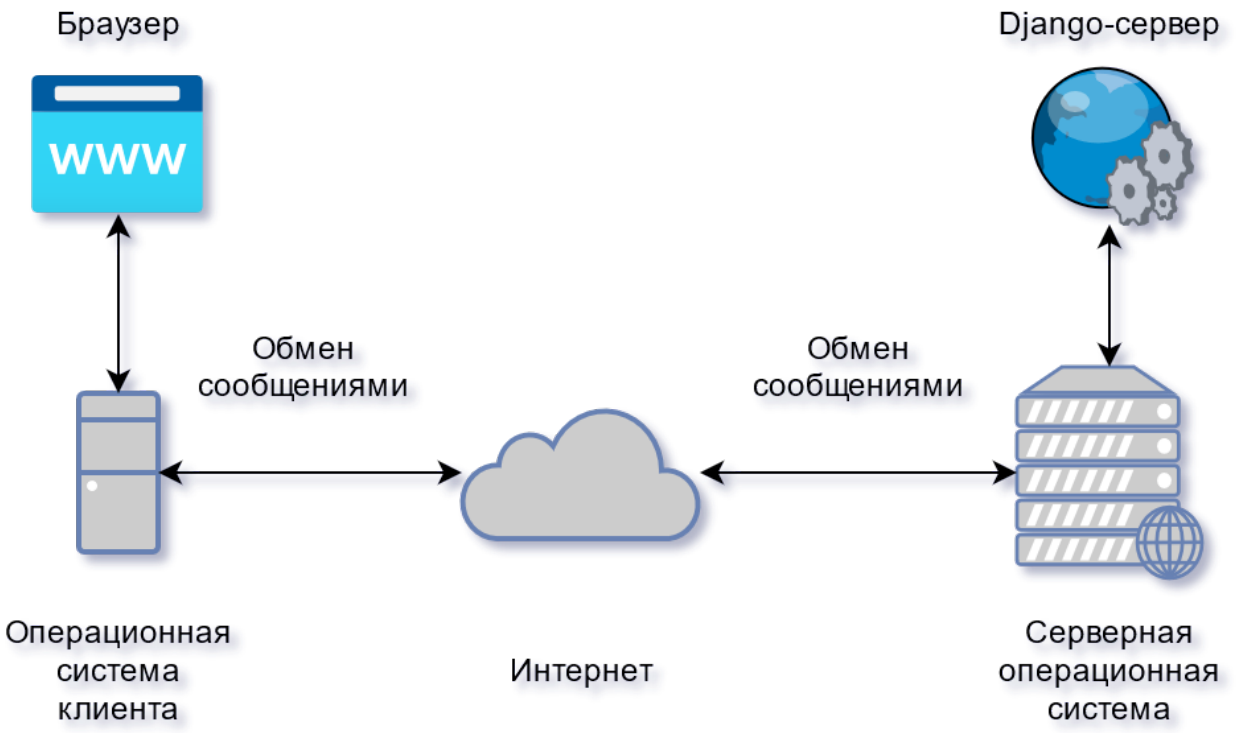

Рис. 8. Схема обмена сообщениями между компьютерами в рамках сети Интернет.

Существует множество операционных систем для выполнения разных задач.

Пользовательская операционная система помогает человеку взаимодействовать с устройством через графический интерфейс. Как правило, такие системы требуют минимального погружения для выполнения повседневных задач: просмотр фото и видео, чтение новостей в интернете, работа с файлами. Наиболее распространенными пользовательскими ОС являются Windows, Android, MacOS, iOs.

Задача серверных ОС — бесперебойный доступ к ресурсам и оптимизация для максимальной производительности. Серверные ОС, как правило, не имеют графической оболочки и используют только командную строку для взаимодействия с пользователем. Наиболее распространенные серверные операционные системы: Linux, BSD-based (OpenBSD, FreeBSD).

При использовании фреймворка Django в проекте необходимо учитывать, что в 95% случаев этот проект будет использоваться на ОС Linux. Linux — это бесплатная операционная система, которую можно кастомизировать под свои нужды. Она распространяется в дистрибутивах: ядро Linux и окружение из программного обеспечения.

Несмотря на использование кроссплатформенного языка Python, в качестве основы необходима именно ОС Linux. Так мы приближаем условия разработки к условиям реальной работы проекта. Чем меньше разница между ОС, которую использует разработчик и ОС, на которой будет работать проект, тем ниже шанс возникновения непредвиденных проблем.

Выбор Linux дистрибутива, его установка и первичная настройка выходят за рамки нашего курса. Однако все дальнейшие приёмы ориентируются на linux-дистрибутив Ubuntu. В качестве ОС для разработчика мы рекомендуем использовать дистрибутив Ubuntu Mate (версии 20.04 LTS) или Linux Mint (Cinnamon; версии 20.1). У них дружелюбный графический интерфейс и они не требовательны к аппаратному обеспечению.

Большинство Linux-дистрибутивов бесплатные, как и фреймворк Django. Это способствует их активному использованию среди небольших организаций и стартапов.

Ещё один нюанс, на который стоит обратить внимание, это тип процессора, с которым будет работать операционная система. Наиболее распространенные архитектуры процессоров: x86 в двух разрядностях x32 и x64 и ARM. В рамках курса мы рассмотрим работу в ОС для архитектуры x86 с разрядностью x64.

Гипервизоры

После того, как вы определились с ОС, её необходимо установить. На большинстве пользовательских компьютеров установлены Microsoft Windows или Apple MacOS. Это операционные системы закрытого типа, которые требуют платной лицензии. И если MacOS в принципе невозможно удалить с компьютера, то в случае с Windows это сделать можно. Есть два варианта решения этой проблемы:

1) Установка ОС параллельно основной;
2) Установка ОС в рамках виртуальной машины.

Первый вариант достаточно сложный и поэтому не рекомендуется для новичков.

Второй — гораздо удобнее. Установка ОС в рамках **виртуальной машины** или **с использованием гипервизора** позволяет не только использовать целевую ОС, но и изолировать рабочее пространство. Таким образом, можно работать в разных окружениях для разных проектов с разными дистрибутивами. Ограничением остается лишь архитектура процессора и объем памяти (оперативной и постоянной).

Чтобы гипервизор работал корректно, нужно его активировать в настройках BIOS. Как это делается, вы всегда можете уточнить в инструкции к материнской плате или в интернете.

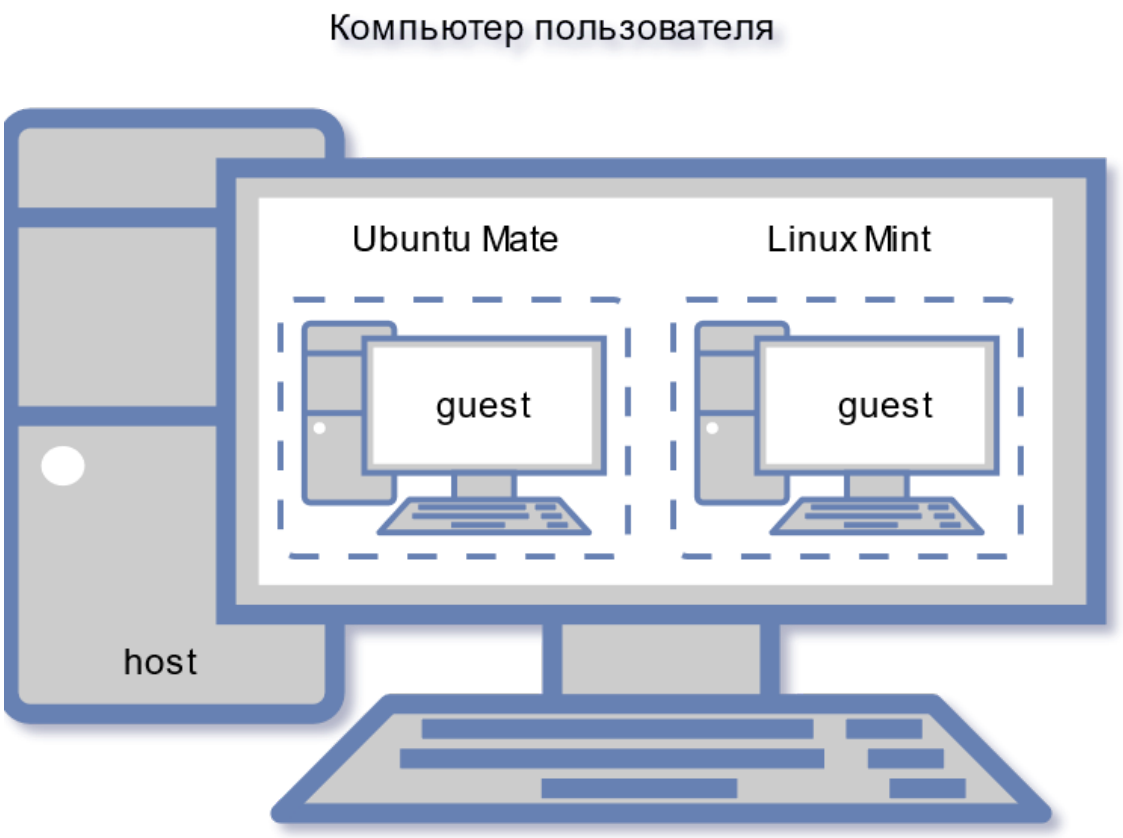

Рис. 9. Использование виртуальных машин.

Виртуальные машины полностью повторяют архитектуру реальных компьютеров, но существуют исключительно в рамках логики программного обеспечения.

Главный недостаток использования виртуальных машин — менее отзывчивый графический интерфейс, чем на основном компьютере.

ОС, на которой запускается виртуальная машина, называют хостовой (от слова host — ведущий), а ОС, которая запускается в рамках виртуальной машины — гостевой.

Наиболее распространенными программами для создания виртуальных машин считаются:

- Oracle VirtualBox
- VMWare
- Microsoft HyperV

VirtualBox и VMWare выпускаются в платной и бесплатных редакциях. HyperV распространяется как компонент хостовой ОС. По указанным ссылкам Вы можете скачать и установить ПО для создания виртуальных машин.

Для пользователей ОС Windows рекомендуется использовать VMWare Player. Он бесплатный для некоммерческого использования, а отзывчивость гостевой системы сравнима с хостовой. Для владельцев MacOS мы рекомендуем VirtualBox.

Еще один нюанс работы виртуальной машины — конфигурация сетевого режима. Всего доступно 3 режима:

- Режим моста (bridge). Если в локальной сети хоста есть маршрутизатор, то рекомендуется использовать именно этот режим. В нем гостевая ОС будет являться таким же полноправным участником локальной сети, как и хостовая. В рамках сети будет выдан уникальный сетевой адрес, по которому можно будет к ней обращаться. Этот режим обычно выбран по умолчанию.
- Режим точки доступа (NAT). В нем хостовая ОС будет являться точкой выхода сетевого трафика. Мы рекомендуем использовать этот режим, если в локальной сети хостовой ОС нет маршрутизатора. В нем для соединения на гостевую ОС потребуется дополнительно настроить проброс портов, т.е. соединить порт хостовой и виртуальной машины. Проброс портов может понадобиться для передачи файлов на гостевую ОС или демонстрации наработок коллегам.
- Приватный режим. Здесь создается изолированный сегмент сети для хостовой и гостевой ОС. У гостевой ОС нет возможности выйти в интернет, а значит, в нашем случае этот вариант не подходит.

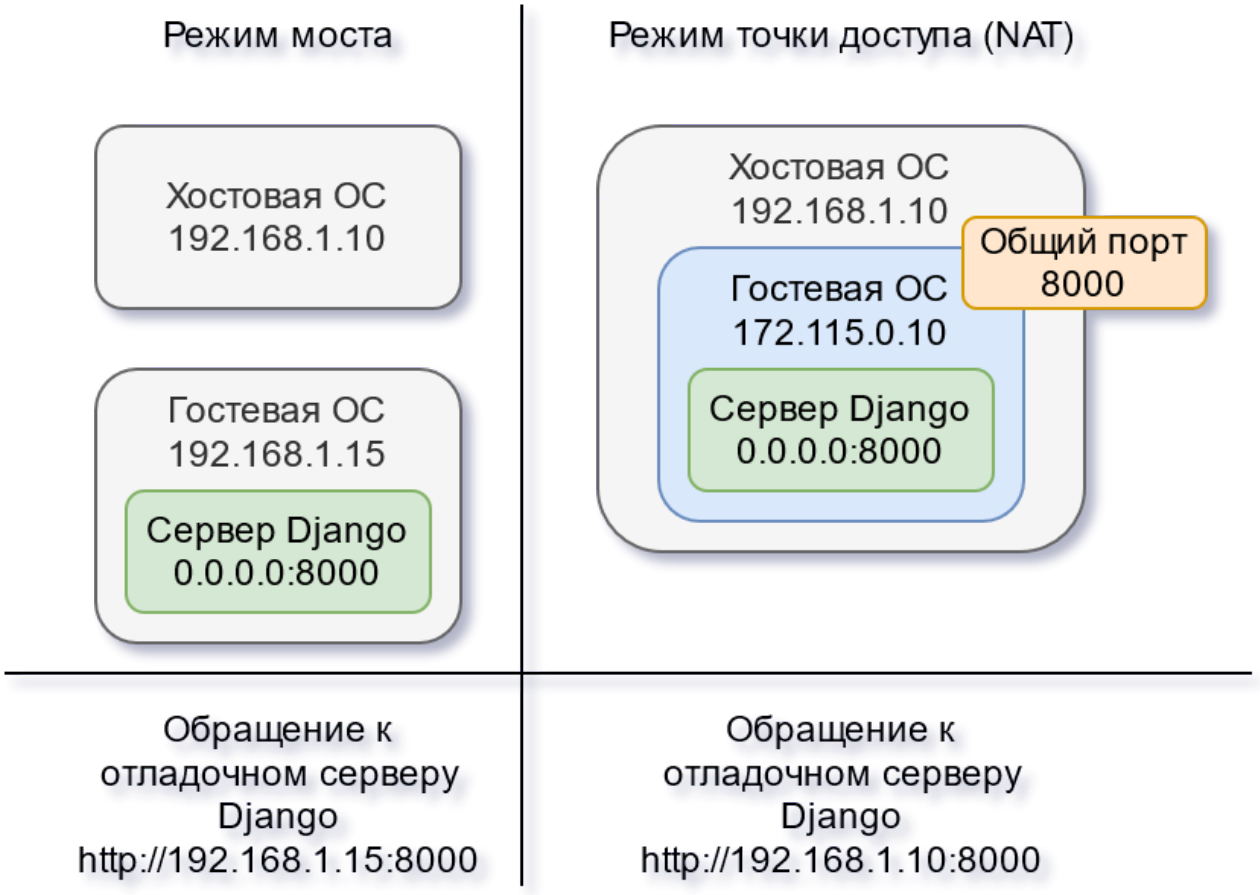

Рис. 10. Схемы сетевой конфигурации виртуальных машин.

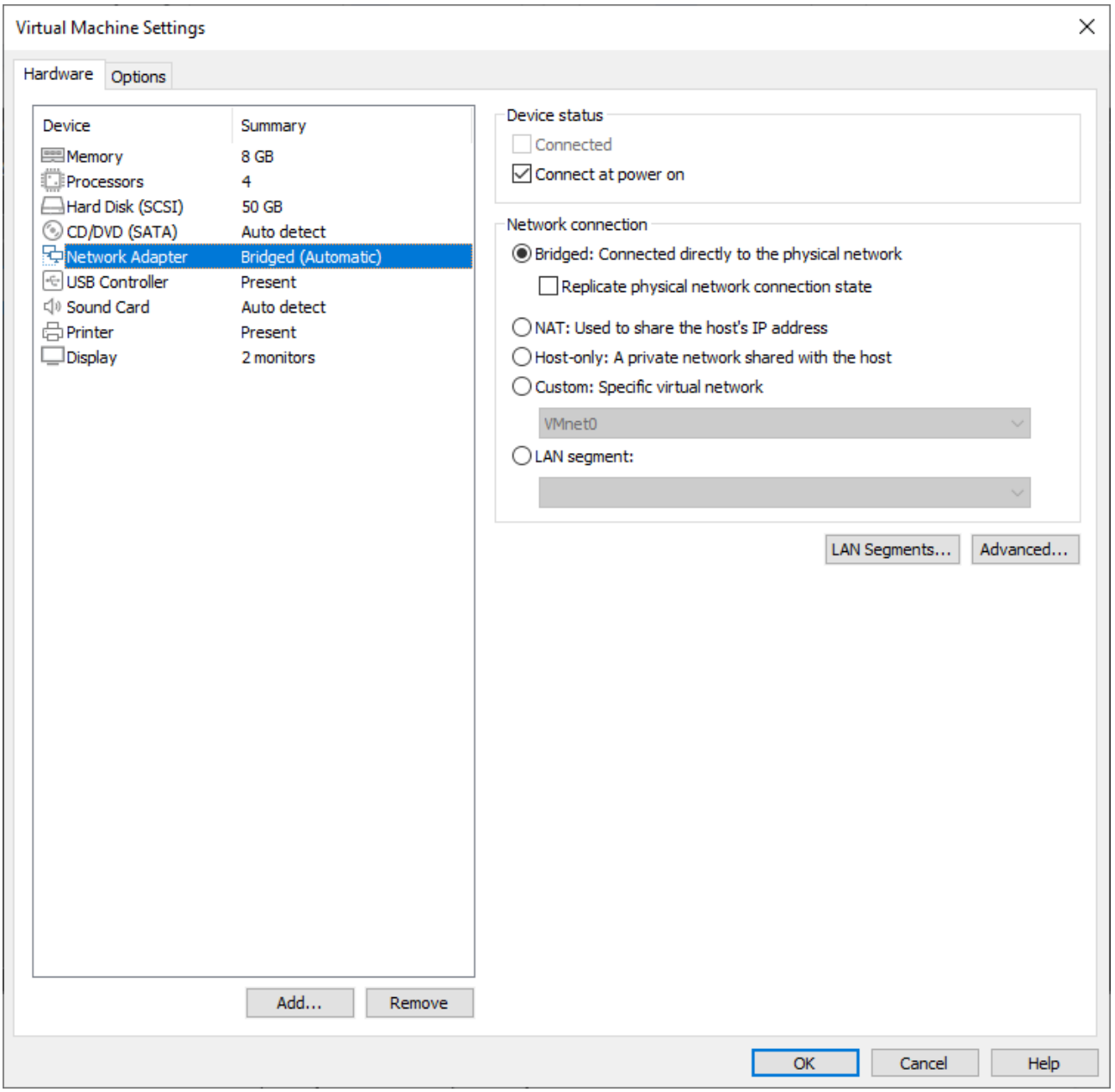

Рис. 11. Настройка режима моста для виртуальной машины в программе VMWare Player.

В рамках ОС Windows есть подсистема запуска Linux-приложений — WSL (windows subsystem linux). Однако мы не рекомендуем вам её использовать, потому что там есть сложные нюансы работы с различным ПО.

Виртуальное окружение python

Python — интерпретируемый язык общего назначения. Он стал настолько популярен, что в итоге интерпретатор Python встроили в дистрибутивы Linux. До определённого момента дистрибутивы поставлялись с двумя интерпретаторами: Python 2.x и Python 3.x В последних редакциях популярных дистрибутивов остался только один — Python 3.

Встроенный интерпретатор еще называют системным. С точки зрения разработчика, системный интерпретатор можно использовать до момента установки дополнительных пакетов. Поскольку внутри операционный системы существует ПО, которое основывается на библиотеках, поставляемых с системным интерпретатором, при установке сторонних библиотек эти зависимости могут быть перезаписаны. Так, обновив пакет или библиотеку для системного интерпретатора, вы можете сломать внутренние механизмы ОС. Для того, чтобы этого избежать, разработчики пользуются ПО для создания виртуального окружения.

Существует три популярных способа создания виртуального окружения:

- Пакет стандартной библиотеки venv. Он позволяет создать изолированное окружение Python, но с одним нюансом: оно будет напрямую зависеть от интерпретатора, с помощью которого создавалось. То есть если вы удалите интерпретатор, виртуальное окружение тоже перестанет работать.
- Сторонний пакет создания виртуальных окружений virtualenv позволяет создавать полную копию интерпретатора и окружение, изолированное от системного интерпретатора. Главный минус такого подхода — зависимость от версии интерпретатор, в котором создается виртуальное окружение.
- Пакет ПО pyenv. Это комплексное ПО, которое позволяет гибко настраивать работу с интерпретаторами Python: устанавливать их из списка и создавать виртуальные окружения. Его главный минус — сложность.

В рамках курса мы рассмотрим первые два пакета, а третий мы рекомендуем изучить самостоятельно в разделе дополнительных материалов.

Для установки пакета virtualenv воспользуйтесь терминальным установщиком системных пакетов apt:

```
// Установка пакета создания виртуального окружения
sudo apt install python3-virtualenv
```

После установки в командной строке появится возможность использовать команду virtualenv.

Запуск статичного сайта

Создадим виртуальное окружение и запустим статичный сайт, используя HTTP-сервер из стандартной библиотеки Python.

На рабочем столе создайте папку “djangoBasics”.

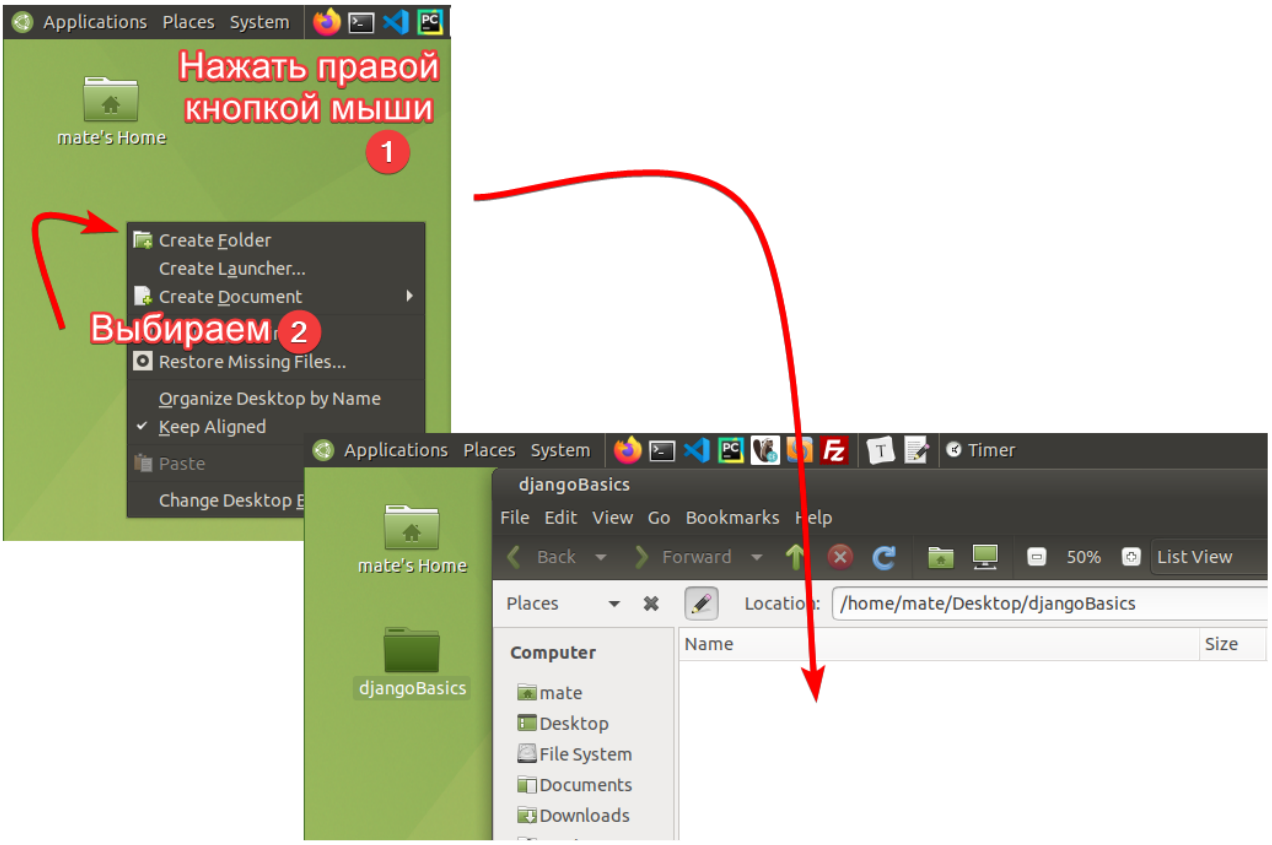

Рис. 13. Создание и открытие новой папки “djangoBasics”

Теперь необходимо открыть новую папку в терминале. Для этого в пустом поле окна папки необходимо щелкнуть правой кнопкой мыши и выбрать пункт «Открыть в терминале».

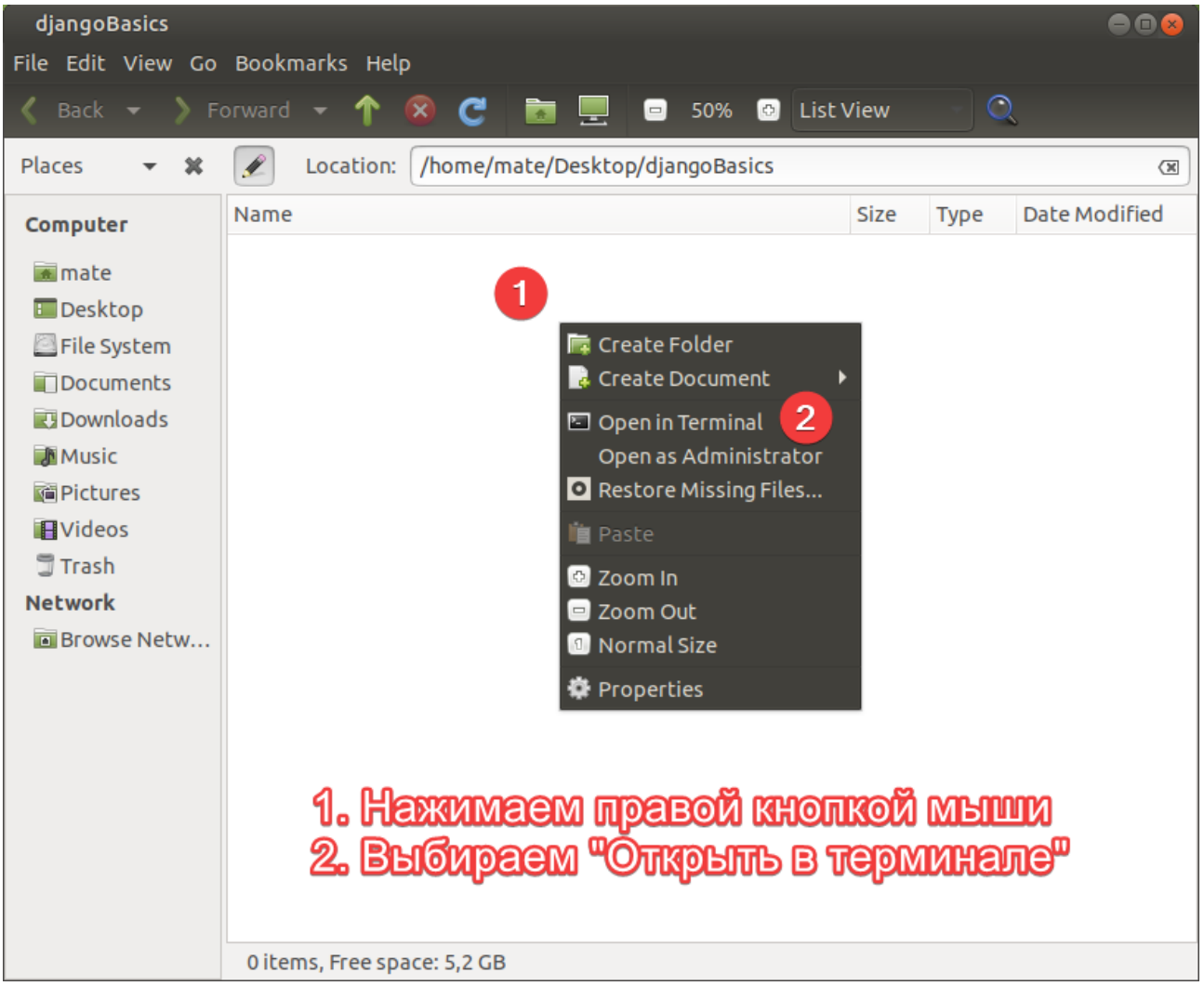

Рис. 14. Открытие папки в терминале

Воспользуемся командой:

```
// Создание виртуального окружения Python
virtualenv -p python3 venv
```

Флаг -p python3 указывает, какой интерпретатор мы хотим использовать в качестве основного для виртуального окружения. Последний параметр venv указывает на имя папки, в которой будет создано виртуальное окружение.

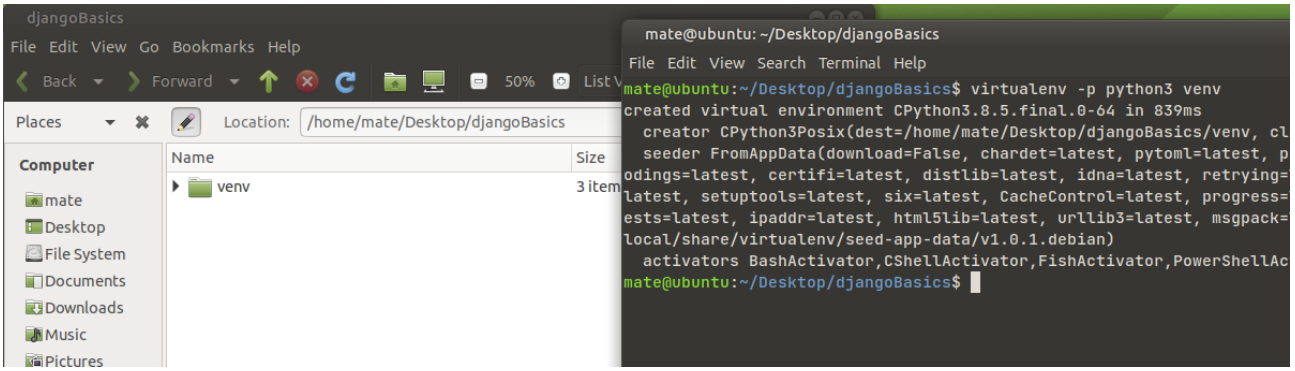

Рис. 15. Успешное создание папки виртуального окружения.

Для использования виртуального окружения его необходимо активировать командой:

```
1  // Активация виртуального окружения
2  source ./venv/bin/activate
```

./venv/bin/activate указывает относительный путь к расположению скрипта активации виртуального окружения.

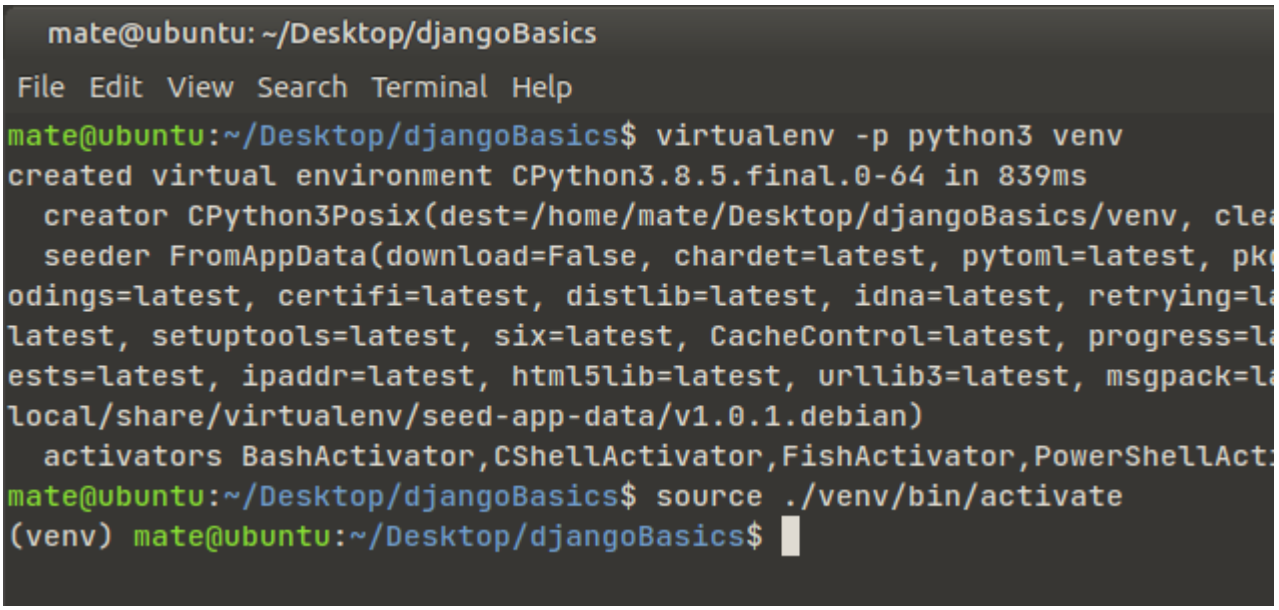

Рис. 16. Активация виртуального окружения.

Префикс командной строки (Рис. 16) (venv) указывает на то, что виртуальное окружение активировано.

Скачайте из материалов к уроку статичный сайт и распакуйте его рядом с папкой виртуального окружения.

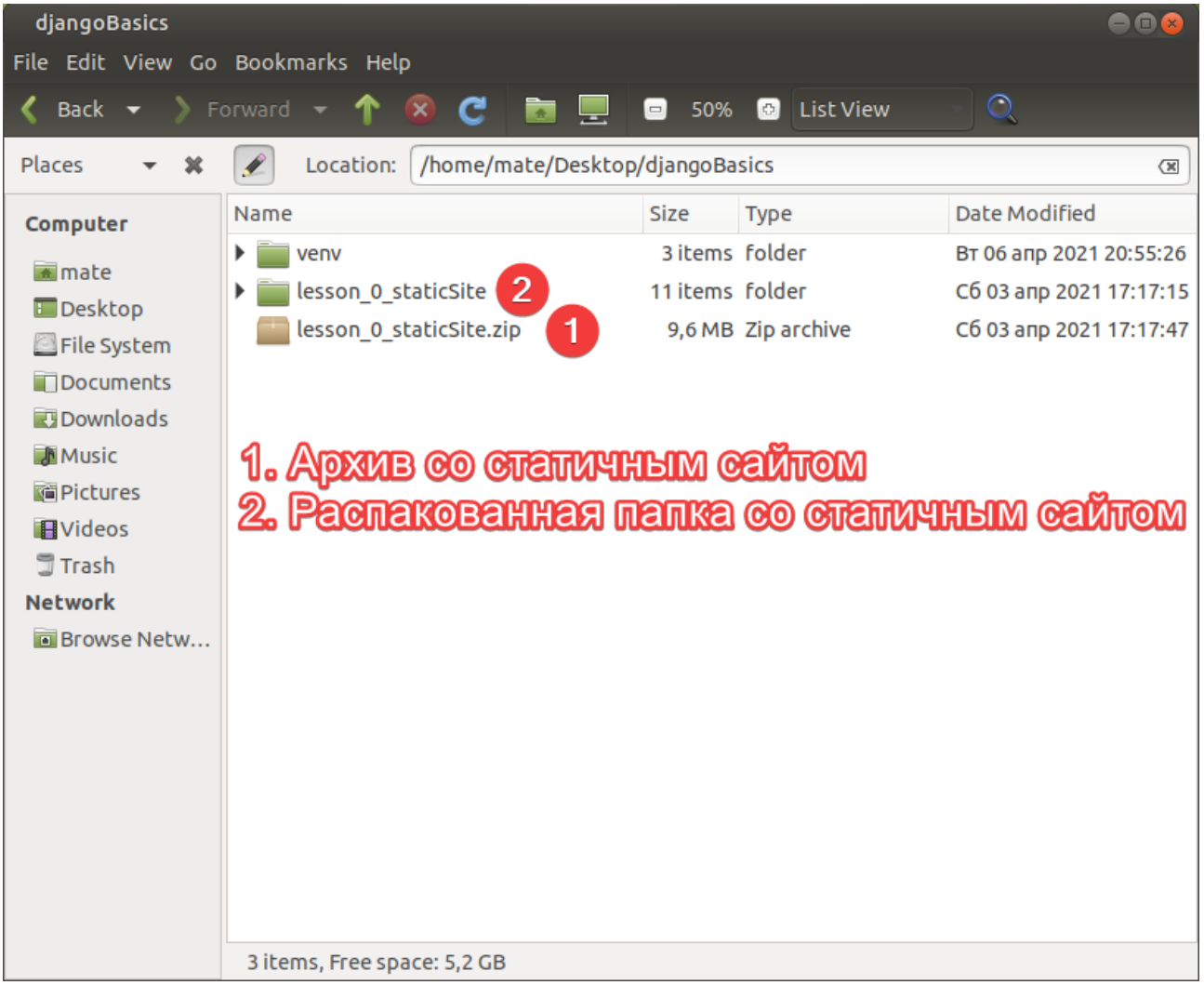

Рис. 17. Распаковка архива со статичным сайтом.

Перейдите в терминале в папку со статичным сайтом и выполните команду запуска HTTP-сервера из стандартной библиотеки python:

```
// Запуск сервера из стандартной библиотеки
python -m http.server 8000
```

8000 указывает на порт, на котором будет запущен сервер.

```
(venv) mate@ubuntu:~/Desktop/djangoBasics$ cd lesson_0_staticSite/
(venv) mate@ubuntu:~/Desktop/djangoBasics/lesson_0_staticSite$ python -m http.server 8000
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Рис. 18. Запуск HTTP-сервера из стандартной библиотеки Python.

Этот сервер позволяет запускать статичные сайты для отладки и не предназначен для «боевого режима». Сайт запускается на адресной паре 0.0.0.0:8000, т.е. на широковещательном адресе (сервер доступен всем участникам локальной сети). Теперь можно зайти на сервер, указав в браузере в качестве адреса loopback (закольцованный адрес).

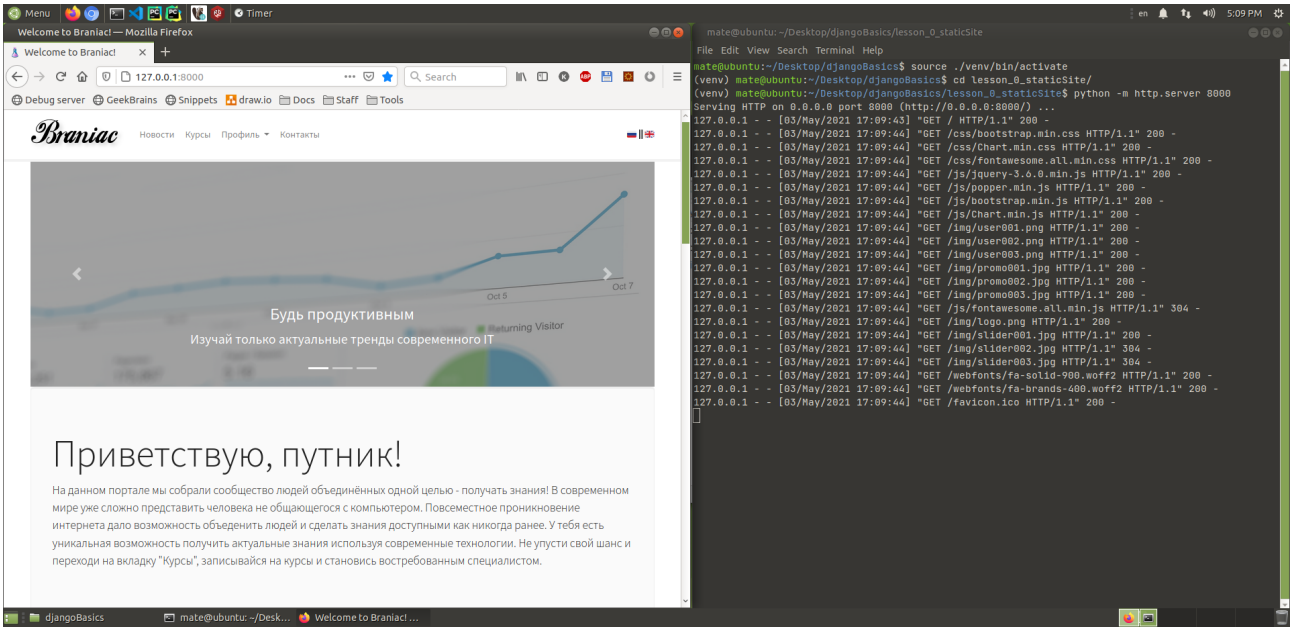

Рис. 19. Открытие статичного сайта в браузере и лог сервера.

Обратите внимание, что в лог-сообщениях сервера отображается HTTP-метод URL ресурса, который запросил браузер и код ответа от сервера.

Заключение

Развитие интернета привело к созданию мощных инструментов, которые позволяют создавать комплексные продукты: интернет-магазины, форумы, социальные сети. Размещаясь на сервере, эти продукты задействуют мощь и простоту HTTP-протокола для взаимодействия с пользователем, который обзорно рассмотрен в рамках урока 1.

Один из инструментов разработки серверной составляющей — фреймворк Django, который создан на языке программирования Python. Python, представляя собой интерпретируемый язык, имеет такие средства изоляции выполнения, как виртуальные среды. После создания виртуальной среды появляется возможность создавать неограниченные по масштабу проекты, которые не вызывают конфликтов во время исполнения: начиная от простого статического сайта и заканчивая социальной сетью мирового масштаба.

Практическое задание

Создайте папку в любом удобном месте хостовой ОС и для каждого задания сделайте по несколько снимков экранов. Для четвёртого задания можно создать простой текстовый файл с расширением .txt. Затем опубликуйте архив папки со скриншотами. Для снимков экрана используйте стороннее ПО: для Windows, например, ShareX или PicPick.

(*) — задания повышенной сложности, (**) — самые сложные задания, поэтому их выполнять необязательно.

1. Установите программное обеспечение для виртуализации и создайте виртуальную машину. Назначьте ей дистрибутив Linux из семейства Ubuntu — Ubuntu Mate, Linux Mint. После этого настройте сетевое соединение в конфигурации виртуальной машины и убедитесь, что на гостевой ОС есть доступ в интернет.

   Снимок экрана: работающая гостевая ОС с открытым в браузере сайтом python.org.

2. Создайте на рабочем столе гостевой ОС папку `djangoBasics`. Затем, используя пакет `virtualenv`, создайте в ней виртуальное окружение Python. Активировав его, убедитесь, что используете версию Python старше 3.7.

   Снимок экрана: терминал с запущенным интерактивным режимом python в активированном виртуальном окружении.

3. Скачайте статичный сайт из дополнительных материалов к уроку и запустите его на HTTP-сервере из стандартной библиотеки Python. Убедитесь, что всё работает, перейдя по кольцевому адресу в браузере.

   Снимок экрана для сдачи ПЗ (практического задания): два окна рядом — браузер и открытый терминал с запущенным сервером.

4. Выберите три рабочих URL на любых ресурсах, которые содержат:
   - URL-путь с двумя или тремя вложениями;
   - несколько GET-параметров;
   - идентификатор якоря.

   В качестве ПЗ принимаются снимки экрана адресной строки браузера или текстовый файл с адресами.

5. (*) В качестве пакета создания виртуального окружения используйте программный комплекс pyenv.

6. (**) В качестве гостевой ОС используйте ОС Ubuntu Server. Организуйте доступ к файловой системе и папке пользователя через сервер Samba.

Глоссарий

Браузер или **веб-обозреватель** — это прикладное программное обеспечение для просмотра страниц, содержания веб-документов, компьютерных файлов и их каталогов, а также управления веб-приложениями.

Виртуальная машина — программная и/или аппаратная система, эмулирующая аппаратное обеспечение платформы и исполняющая программы для guest-платформы на host-платформе.

Виртуальное окружение Python — изолированная среда выполнения программ на Python.

Операционная система — комплекс взаимосвязанных программ, предназначенных для управления ресурсами компьютера и организации взаимодействия с пользователем.

Протокол передачи данных — набор определенных правил или соглашений интерфейса логического уровня, который определяет обмен данными между различными программами и задаёт единообразный способ передачи сообщений и обработки ошибок.

Фреймворк — платформа, которая определяет структуру программной системы или ПО и облегчает разработку и объединение разных компонентов большого проекта.

Шаблон или **паттерн проектирования** — это описание типичных способов решения часто встречающихся проблем при создании программного обеспечения.

Дополнительные материалы

1. Обзор HTTP-протокола.
2. Список кодов состояния HTTP.
3. Стандарт формирования URL — RFC 3986.
4. Flask framework.
5. Ubuntu Mate | Linux Mint Cinnamon.
6. Архитектура процессора x86.
7. Расширение архитектуры x86 для 64-битной адресации.
8. Про гипервизоры.
9. Установка Linux и гипервизора.
    a. Установка Ubuntu 20.04 в VirtualBox.
    b. Installing Linux Ubuntu Version 20.04 in VMware Workstation Player.
    c. How to Install VirtualBox on Mac OS X.
10. Карманный справочник Linux. Рекомендуется для ознакомления с базовыми командами Linux.
11. Pyenv.
12. Loopback.

Используемые источники

1. Интернет в цифрах и фактах
2. Статистика интернета и соцсетей на 2021 год
3. Что означает микро-фреймворк?
4. Фреймворк
5. Протокол передачи данных

6. Операционная система