ЧАСТЬ I

Что у нас нового

Мир получил огромную пользу от изобретения Всемирной паутины (или просто «веб» как транслитерации названия Web) сэром Тимом Бернерсом-Ли[1] и языка программирования Python Гвидо ван Россумом.

Единственная небольшая проблема заключается в том, что безымянное издательство компьютерных книг часто помещает пауков и змей на свои обложки книг по тематике Web и Python соответственно. Если бы только сеть называлась Всемирный Гав, а язык Python был бы (Винни) Пухом, эта книга могла бы получить обложку как на рис. I.1.

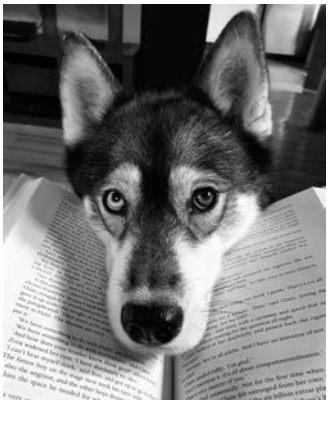

Рис. I.1. FastAPI: современная разработка Гав-Пух

[1] Однажды я пожал ему руку. Я не мыл свою в течение месяца, но могу поспорить, что он сделал это сразу же.

Но я отвлекся[1]. Эта книга посвящена таким темам, как:

- *Всемирная паутина* — особенно эффективная технология, как она изменилась и как теперь разрабатывать для нее программное обеспечение;
- *Python* — очень продуктивный язык для веб-разработки;
- *FastAPI* — особенно производительный веб-фреймворк для Python.

В двух главах части I книги обсуждаются новые темы в веб-разработке и языке Python — сервисы и API, конкурентность, многоуровневые архитектуры и большие-большие данные.

Часть II — это обзор FastAPI, свежего веб-фреймворка на Python. В этой части содержатся ответы на заданные в части I вопросы.

В части III мы углубляемся в инструментарий FastAPI, включая советы, полученные в процессе разработки.

Наконец, в части IV представлена галерея веб-примеров FastAPI. Для них использовался общий источник данных — список воображаемых существ, что может быть немного интереснее и целостнее, чем обычные случайные представления данных. Это должно дать вам отправную точку для конкретного применения этого веб-фреймворка.

[1] И точно не в последний раз.

ГЛАВА 1

Современная Всемирная паутина

> Всемирную паутину, какой я ее себе представлял, мы еще не видели. Будущее все еще намного больше, чем прошлое.
>
> *Тим Бернерс-Ли*

Обзор

Когда-то Всемирная паутина была маленькой и простой. Разработчикам было так весело отправлять вызовы PHP, HTML и MySQL в отдельные файлы и с гордостью говорить всем, что они могут заглянуть на свой веб-сайт. Но со временем Сеть разрослась до невообразимого количества страниц и развивающаяся игровая площадка превратилась в метавселенную тематических парков.

В этой главе отмечены некоторые все более актуальные для современной Всемирной паутины области:

- сервисы и API;
- конкурентность;
- уровни (слои);
- данные.

В следующей главе я расскажу о том, какие возможности предоставляет Python для работы в этих областях. После этого погрузимся в веб-фреймворк FastAPI и посмотрим, что он может предложить.

Сервисы и API

Паутина — это отличная соединительная ткань. Несмотря на то что большая часть деятельности по-прежнему происходит на стороне *контента* — HTML, JavaScript, изображений и т. д., все большее внимание уделяется интерфейсам прикладного программирования (API), соединяющим различные элементы программ.

Обычно *веб-сервис* управляет низкоуровневым доступом к базе данных и бизнес-логикой среднего уровня (часто их объединяют в *бэкенд*), а JavaScript или мобильные приложения обеспечивают богатый *фронтенд* верхнего уровня (интерактивный пользовательский интерфейс). Эти миры становятся все более сложными и разнообразными, что обычно требует от разработчиков специализации в том или ином направлении.

Быть разработчиком *полного стека* (фулстека) сейчас сложнее, чем раньше[1].

Эти два мира общаются друг с другом с помощью API. В современном Интернете дизайн API так же важен, как и дизайн самих сайтов. API — это контракт, подобный схеме базы данных. Определение и модификация API — это уже серьезная работа.

Виды API

Каждый API определяет следующее:

- *протокол* — структуру управления;
- *формат* — структуру содержимого.

Многочисленные методы API развивались по мере эволюции технологий от изолированных машин до многозадачных систем и сетевых серверов. Вероятно, в какой-то момент вы столкнетесь с одним или несколькими из них, поэтому далее приведу краткое описание, прежде чем перейти к языку *HTTP* и его друзьям, описанным в этой книге.

- До появления сетей API обычно означал очень тесную связь, например вызов функции из *библиотеки* на том же языке, что и ваше приложение, — скажем, вычисление квадратного корня в математической библиотеке.

[1] Я бросил попытки несколько лет назад.

- *Удаленные вызовы процедур* (remote procedure call, RPC) были придуманы для вызова функций в других процессах на той же или другой машине, как если бы они находились в вызывающем приложении. Популярным примером в настоящее время служит система gRPC (https://grpc.io).
- С помощью *системы передачи сообщений* отправляются небольшие фрагменты данных по конвейеру между процессами. Сообщения могут быть глагольными командами или просто обозначать интересующие вас *события*, похожие на существительные. В настоящее время популярными решениями для обмена сообщениями, которые варьируются от наборов инструментов до полноценных серверов, являются брокеры Apache Kafka (https://kafka.apache.org), RabbitMQ (https://www.rabbitmq.com), NATS (https://nats.io) и ZeroMQ (https://zeromq.org). Взаимодействие может строиться по разным схемам:
  - *запрос — ответ*. Точно так, как браузер вызывает веб-сервер;
  - *издатель — подписчик*, или *pub-sub*. Издатель (publisher, или pub) рассылает сообщения, а подписчики (subscribers, или sub) обрабатывают каждое из них в соответствии с некоторыми данными, содержащимися в сообщении, например с темой;
  - *очереди*. Работают как подход pub-sub, но только один подписчик из пула получает сообщение и действует в соответствии с ним.

Любой из этих подходов может использоваться вместе с веб-сервисом, например, для выполнения медленной задачи бэкенда, такой как отправка электронной почты или создание уменьшенного изображения.

HTTP

Бернерс-Ли предложил для своей Всемирной паутины три компонента:

- *HTML* — язык для отображения данных;
- *HTTP* — протокол «клиент — сервер»;
- *URL* — схему адресации для веб-ресурсов.

Хотя в ретроспективе все кажется очевидным, на деле это оказалось до смешного полезной комбинацией. По мере развития Интернета люди экспериментировали, и некоторые идеи, такие как тег `IMG`, выжили в борьбе по Дарвину. По мере того как потребности пользователей прояснялись, люди всерьез занялись определением стандартов.

REST(ful)

В одной из глав докторской диссертации (https://oreil.ly/TwGmX) Роя Филдинга есть определение *передачи репрезентативного состояния* (Representational State Transfer, REST) — *архитектурного стиля* для использования HTTP[1]. Несмотря на то что на эту работу часто ссылаются, ее по большей части неправильно понимают (https://oreil.ly/bsSry).

В современной Паутине развилась и доминирует примерно одинаковая адаптация. Она называется *RESTful* и обладает следующими характеристиками:

- использует HTTP и протокол «клиент — сервер»;
- не имеет состояния (каждое соединение независимо);
- кэшируема;
- основана на ресурсах.

Ресурс — это данные, которые можно определять и с которыми можно выполнять операции. Веб-сервис предоставляет *конечную точку* — отдельный URL и HTTP-*глагол* (действие) — для каждой функции. Конечную точку называют также *маршрутом*, поскольку она направляет URL к функции.

Пользователи баз данных знакомы с акронимом *CRUD* для процедур: создание (create), чтение (read), модификация (update), удаление (delete). HTTP-глаголы довольно хорошо вписываются в понятие CRUD:

- `POST` — создание (запись);
- `PUT` — полная модификация (замена);
- `PATCH` — частичная модификация (обновление);
- `GET` — получение (считывание, извлечение);
- `DELETE` — удаление.

Клиент отправляет запрос на конечную точку RESTful с данными в одной из таких областей HTTP-сообщения, как:

---

[1] Под стилем понимается шаблон более высокого уровня, например «клиент — сервер», а не конкретная конструкция.

- заголовки;
- строка URL;
- параметры запроса;
- значения в теле сообщения.

В свою очередь, HTTP-ответ возвращает:

- целочисленное значение *кода состояния* (https://oreil.ly/oBena), определяющее такие состояния, как:
  - группа кодов 100 — информация, продолжение выполнения;
  - группа кодов 200 — успешное выполнение;
  - группа кодов 300 — перенаправление;
  - группа кодов 400 — ошибка на стороне клиента;
  - группа кодов 500 — ошибка на стороне сервера;
- различные заголовки;
- тело сообщения, которое может быть пустым, единым или разделенным на *части* (последовательные фрагменты).

По крайней мере один код состояния можно считать пасхалкой — 418 (I'm a teapot, https://www.google.com/teapot). На странице должен появиться подключенный к сети чайник. Если попросить, он нальет вам чашечку чая.

В широком доступе существует множество сайтов и книг о проектировании RESTful API, и все они содержат полезные практические указания. Эта книга станет вашим помощником в пути.