ЦИФРОВОЕ ИЗДАНИЕ · РУССКИЙ

Новый *креатор* технический.

БЕСПЛАТНЫЙ ФРАГМЕНТ

Введение · Полное содержание · Глава 2

АВТОР

Hernán Capucci

Независимое издание · 2026

Этот бесплатный фрагмент включает введение книги, полное содержание и представительную главу. Так ты сможешь оценить тон, подход и способ обучения, прежде чем перейти к полной версии.

Что включает этот фрагмент

- Полное введение (Глава 0).
- Полное содержание книги.
- Глава 2 целиком: *Как работает интернет*.
- Информация о продолжении.

Полное содержание

Глава 0 — Почему эта книга существует

Однажды ты просишь ИИ сгенерировать тебе что-нибудь — код для какой-то функции, автоматизацию, которая свяжет два сервиса, фрагмент, решающий конкретную задачу.

То, что он тебе возвращает, вроде бы работает. Ты этим пользуешься. И в какой-то момент этого процесса возникает вопрос, который ты, возможно, не решаешься задать вслух: как мне понять, что это правильно?

Этот вопрос не пустяковый. Это разница между тем, чтобы действовать с собственным суждением, и тем, чтобы действовать в надежде, что никто не найдёт ошибку.

Эта книга существует для того, чтобы ты мог на него ответить.

Доступ без понимания

Ещё несколько лет назад, чтобы создавать **программное обеспечение**, требовалась специальная техническая подготовка. Выучить язык программирования, разобраться в структурах данных, настроить среду разработки, читать документацию, которая предполагала, что ты уже многое знаешь. Это был высокий и вполне реальный барьер.

Потом появились инструменты **генеративного искусственного интеллекта**. Внезапно любой человек мог написать инструкцию на естественном языке и получить работающий код, настроенную автоматизацию, систему, подключённую к внешним сервисам. Технический барьер существенно снизился.

Но снизился технический барьер, а не концептуальный.

То, что инструмент генерирует код за тебя, не означает, что ты понимаешь, что он сгенерировал. То, что платформа развёртывает твоё приложение в один клик, не означает, что ты понимаешь, что такое развёртывание, что может пойти не так или как откатить назад, когда что-то ломается. То, что ИИ объясняет тебе ошибку, не означает, что ты знаешь, почему она возникла и как не допустить её повторения.

Доступ улучшился. Понимание **системы**, которая лежит в основе, во многих случаях не поспевало за тем же темпом.

Создавать — не то же самое, что понимать

Сегодня людей, которые создают цифровые продукты без технической подготовки, больше, чем когда-либо прежде. Приложения, автоматизации, интеграции между сервисами, контентные платформы, оцифрованные внутренние процессы.

Эта демократизация реальна и ценна. Проблема возникает, когда что-то работает не так, как ожидалось, или когда приходится принимать решения о построенной системе.

Предприниматель, который с помощью ИИ создал свой интернет-магазин, не знает, почему «упала база данных». Он не знает, защищена ли информация его клиентов. Он не знает, хорошо ли выполнена работа, которую сдал нанятый им разработчик, или нет. Он не знает, какие вопросы задать, чтобы это выяснить.

Специалист, который автоматизировал свои процессы с помощью ИИ, точно не понимает, что делает каждая часть собранного им механизма. Он принимает результат, потому что вчера всё работало. Когда работать перестаёт, у него нет инструментов, чтобы понять, где искать.

Основательница, которая работает с внешней технической командой, кивает, когда ей говорят про **API**, про **репозитории** или про архитектуру системы. Но она не понимает. И не спрашивает, потому что не знает, о чём спросить.

Во всех этих случаях проблема не в нехватке ума или способностей. Она в нехватке словаря и концептуального каркаса, которые позволяют действовать в цифровой экосистеме с собственным суждением.

ИИ как отправная точка

Именно искусственный интеллект сделал эту книгу насущной. Он — будильник.

Но предназначение книги не в том, чтобы говорить об искусственном интеллекте. Оно в том, чтобы говорить о системах, которые существуют в основе любого цифрового проекта, с ИИ или без него.

API — это не понятие из области ИИ. Оно существует уже десятилетиями. **Репозиторий** кода тоже не нов. Разницу между средой разработки и средой производства придумала не какая-то языковая модель. Аутентификация пользователей, реляционные базы данных, жизненный цикл **деплоя** — всё это существовало до генеративного ИИ и продолжит существовать после него.

Что изменилось с ИИ — так это то, что теперь эти вещи стали доступны людям, которые раньше их не касались. И эта доступность без понимания порождает новые зависимости.

Зависимость от инструмента: если он изменится, если он даст сбой, если он выдаст неожиданные результаты, у тебя нет способа их оценить. Зависимость от разработчика, который действительно понимает: если он уйдёт, если он поднимет расценки, если он исчезнет, ты не сможешь продолжать без него. Зависимость от того, кто утверждает, что знает больше: без собственного каркаса ты не сможешь отличить настоящее знание от пустых обещаний.

ИИ открыл дверь. Эта книга даёт словарь, чтобы понять, что находится внутри.

Недостающий словарь

Есть распространённая путаница, которую стоит развеять с самого начала.

Проблема, которой посвящена эта книга, не в том, что ты не умеешь программировать. Неумение программировать — не проблема. Огромное количество людей, очень эффективных в цифровом мире, не умеют программировать и не нуждаются в том, чтобы этому учиться.

Важно не умение писать код. Важно умение читать его с достаточным суждением, чтобы понимать, что он делает. Важно понимать, что такое **база данных**, чтобы принимать решения о том, как хранить информацию. Знать, что такое репозиторий, чтобы уметь проверить то, что сдал тебе разработчик. Знать, что такое **эндпоинт**, чтобы уметь точно попросить ИИ его построить. Знать, что такое среда производства, чтобы по ошибке не сломать что-то работающее вживую.

Ничто из этого не требует написать ни единой строчки кода.

Представь такую конкретную ситуацию: разработчик говорит тебе «упала миграция в среде staging». Без контекста эта фраза — шум. Со словарём этой книги ты понимаешь, что «миграция» — это изменение в структуре базы данных, что «staging» — это тестовая среда, предшествующая производству, и что проблема, вероятно, пока не затронула пользователей, но её нужно решить, прежде чем продолжать. Эта разница не требует программировать. Она требует понимать понятия.

А что действительно требуется — так это технический словарь. Словарь, который позволяет задавать правильные вопросы, толковать ответы, оценивать предложения, распознавать тревожные сигналы и принимать решения с суждением. Этот словарь не приобретается использованием инструментов. Он приобретается через понимание понятий, которые стоят за ними.

Именно это выстраивает эта книга. Понятие за понятием, с нуля, с универсальными примерами и без допущения о каких-либо предварительных знаниях.

Для кого эта книга

Эта книга написана для людей, которые создают цифровые вещи без технической подготовки и хотят делать это с бóльшим суждением.

Сюда входит тот, кто собирает свой первый цифровой продукт с помощью ИИ и не до конца понимает, что именно он создаёт. Тот, кто работает с внешней технической командой и хочет уметь говорить с ней так, чтобы каждый разговор не превращался в усилие по переводу. Тот, кто пользуется инструментами автоматизации в повседневной работе и хочет понимать, что они на самом деле делают. Тот, у кого есть идея и кому нужно оценить её технически, прежде чем вкладывать время и деньги в её разработку.

Не важна ни отрасль, ни область работы. Не важны также ни возраст, ни уровень формального образования в области технологий.

А что действительно важно — так это то, что ты свободно пользуешься компьютером в своей работе или проекте. Что ты попробовал хотя бы какой-нибудь инструмент ИИ. И что ты хочешь лучше понимать **цифровую экосистему**, в которой действуешь.

Отправной точкой может быть то, что ты никогда не открывал репозиторий кода. Что ты никогда не настраивал среду разработки. Что тебе не ясно, в чём разница между сервером и облаком. Ни одна из этих отправных точек не является препятствием. Эта книга начинается именно оттуда.

Чего ты здесь не найдёшь

Прежде чем продолжить, стоит чётко проговорить, чего эта книга не делает.

Она не учит программировать. Это не руководство ни по какому языку программирования и ни по какому конкретному инструменту. Если ты ищешь именно это, для этого есть лучшие и более специализированные ресурсы.

Она не утверждает, что ИИ делает всё. Он этого не делает. У него есть реальные возможности и реальные ограничения. Эта книга честно показывает и то, и другое.

Она не обещает, что за определённый срок твой проект будет готов. У технических процессов своя сложность и своё время. Приуменьшать их было бы ложью.

Она не зависит ни от какого конкретного инструмента, платформы или сервиса. Когда упоминается конкретный инструмент, это в качестве примера. Понятия, которым она учит, применимы одинаково с любым стеком или окружением, которым ты пользуешься.

Она не рассказывает об отдельных историях успеха или реальных проектах как об образцах для подражания. Примеры по замыслу общие и универсальные.

Она не говорит «это легко», когда это не так. Некоторые понятия в этой книге просты. Другие требуют внимания и повторного прочтения. Когда что-то обладает реальной сложностью, книга это отмечает.

Что ты здесь найдёшь

Дочитав эту книгу, ты обретёшь конкретное понимание понятий, которые сегодня, возможно, знаешь только по названию.

Ты поймёшь, что такое приложение изнутри: какая часть управляет тем, что ты видишь на экране, какая часть обрабатывает логику и данные, и как эти две части общаются между собой. Это не сделает так, что ты сможешь построить его в одиночку, но зато ты будешь понимать, о чём говорят, когда тебе его описывают или когда что-то ломается.

Ты сможешь открыть репозиторий кода и прочитать его структуру: какие есть папки, что делает каждая часть проекта, что говорит история изменений. Эта информация доступна в любом репозитории, и сегодня ты, возможно, не знаешь, что можешь использовать её, чтобы понять то, что кто-то тебе сдал.

Ты сможешь написать **промпт** с достаточным контекстом, чтобы ИИ выдал что-то полезное. Разница между расплывчатой инструкцией и точной инструкцией — не в таланте: это знание того, какая информация нужна инструменту, чтобы работать хорошо.

Ты сможешь проверять то, что сгенерировал ИИ или разработчик, конкретными вопросами: что именно делает эта часть? Что произойдёт, если это поле придёт пустым? Есть ли какой-то раздел, который обращается к чувствительной информации? Эти вопросы не требуют умения программировать. Они требуют понимания задействованных понятий.

Ты сможешь использовать глоссарий этой книги как рабочий инструмент: открыть определение, когда появляется незнакомый термин, понять его и вернуться к разговору или документу с бóльшей ясностью, чем прежде.

И ты сможешь распознавать, когда ответ — человека или инструмента — невозможно оценить с твоим нынешним знанием. Это не всегда означает знать правильный ответ. Это означает знать, когда вопрос, который тебе нужно задать, лучше того, что ты задал.

Как она устроена

Книга состоит из шести частей плюс эта вводная глава, практических приложений и технического глоссария.

Часть I охватывает основы: что такое программное обеспечение, как работает интернет, что такое модель «клиент — сервер», что такое база данных и что такое код. Это самые базовые понятия цифровой экосистемы. Каждая следующая часть считает их известными.

Часть II углубляется в архитектуру приложения: **фронтенд**, **бэкенд**, API, базы данных в деталях, аутентификация и сессии. Это внутренняя структура любого современного цифрового продукта.

Часть III охватывает инструменты и процессы, которые заставляют проект работать в реальном мире: контроль версий с **Git**, командную строку, среды разработки, деплой и базовую безопасность.

Часть IV посвящена умной работе с ИИ: что он может делать, где он систематически даёт сбой, как сформулировать эффективный промпт, как проверять то, что он выдаёт, и как использовать модели, **агентов** и автоматизации, не теряя контроль над процессом.

Часть V собирает всё воедино в контексте создания чего-то конкретного: как подготовиться перед началом, как общаться с техническими командами, как перейти от идеи к работающему продукту.

Часть VI — это практические приложения: чек-листы для быстрой справки, базовые команды терминала и Git, шаблоны переиспользуемых промптов и ресурсы для углубления в каждую область.

В конце книги есть **технический глоссарий** с более чем 200 терминами, объяснёнными ясным языком, с примерами и без жаргона. Это не декоративное приложение. Это неотъемлемая часть книги.

Порядок частей не произволен. Каждая строится на предыдущей. Части о системах и инструментах нуждаются в основах Части I. Части об ИИ и создании нуждаются в архитектуре и инструментах. Если ты решаешь читать по порядку, вот в чём причина.

Как ею пользоваться

Есть два способа читать эту книгу, и оба верны.

Первый — подряд, от начала до конца. Он рекомендуется тому, у кого мало или совсем нет предыдущего контакта с цифровой экосистемой. Это чтение выстраивает понимание накопительно: каждая глава готовит почву для следующей.

Второй — как справочник. Кто-то упомянул тебе API, а ты не понял, что это: открываешь Главу 8. Тебе предстоит сделать твой первый деплой, а ты не знаешь, что он подразумевает: открываешь Главу 14. Тебе нужно понять, что такое **аутентификация**, перед встречей о безопасности: открываешь Главу 10. Каждую главу можно читать независимо, хотя перекрёстные ссылки указывают, что стоит прочитать раньше.

Глоссарий работает так же. Когда технический термин появляется в главе впервые, он выделен **жирным** со ссылкой на глоссарий. В цифровой версии книги эта ссылка — активная гиперссылка: ты можешь перейти прямо к определению и вернуться к главе одним кликом. В печатной версии ссылка указывает номер страницы.

Замечание о примерах: на протяжении книги ты будешь встречать одни и те же типы контекстов — интернет-магазин, приложение для управления записями, платформу

курсов. Это общие ситуации, выбранные потому, что они не требуют никаких предварительных знаний ни об одной отрасли. Понятия применяются точно так же в любом другом типе проекта.

Замечание об инструментах: когда инструмент упоминается как пример, это именно пример. Понятия, которым учит эта книга, не зависят ни от какой конкретной платформы. Если инструмент, которым ты пользуешься сегодня, завтра изменится, понятия остаются в силе.

Создавать с собственным суждением

Есть разница между тем, кто пользуется цифровыми инструментами, не понимая их, и тем, кто пользуется ими, зная, что он делает.

Эта разница не в умении программировать. Она в том, чтобы обладать словарём для правильных вопросов. Концептуальным каркасом для оценки ответов. Способностью распознать, когда что-то не так, даже если ты точно не знаешь почему. Самостоятельностью принимать решения, не завися целиком от других в понимании того, что происходит.

Именно этот образ выстраивает эта книга: новый технический креатор. Тот, кто использует ИИ, работает с техническими командами, создаёт цифровые вещи — и делает это с собственным суждением.

Он не программист. Он и не претендует на это. Но он и не действует вслепую, не принимает результаты, не имея возможности их оценить, и не зависит от того, чтобы кто-то другой всё ему объяснял.

Он понимает систему, в которой работает. Он знает, о чём спросить, как оценить то, что получает, и когда признать, что что-то не так. Это понимание не требует писать код. Оно требует понимать, как работает то, чем ты пользуешься.

Чему ты научился в этой главе

- ИИ демократизировал доступ к техническому исполнению, но не понимание цифровой экосистемы.
- Неумение программировать — не проблема. А вот нехватка минимального технического словаря — проблема.

- Эта книга не учит программировать. Она учит понятиям, которые позволяют создавать с суждением, говорить с техническими командами и проверять то, что выдаёт ИИ.
- Книга состоит из шести частей, практических приложений и глоссария из более чем 200 терминов. Её можно читать подряд или использовать как справочник.
- В цифровой версии термины глоссария — активные гиперссылки в каждой главе.

Что дальше

Глава 1 начинает с самого начала: что такое программное обеспечение и почему важно его понимать, даже если ты никогда не будешь его писать. Это первый блок Части I и основа всего, что идёт дальше.

Глава 2 — Как работает интернет

Ты вводишь адрес в браузере и нажимаешь Enter. Меньше чем за секунду появляется страница: текст, изображения, актуальные данные.

Откуда всё это пришло? Как оно добралось до твоего экрана? Почему иногда оно не доходит?

Чтобы ответить на эти вопросы, не нужно уметь программировать. Нужно понимать инфраструктуру, которая соединяет любое устройство с любым сервером в мире. Это понимание — основа для того, чтобы диагностировать, почему что-то работает, — и почему иногда не работает.

Путь одного запроса

Когда ты вводишь адрес в браузере, то, что ты вводишь, называется **URL** — Uniform Resource Locator, унифицированный указатель ресурса. Это адрес, который идентифицирует конкретный ресурс в интернете: страницу, изображение, файл, элемент данных.

Браузеру нужно найти сервер, на котором находится этот ресурс. Но серверы не идентифицируются по своим читаемым именам: они идентифицируются по числам. У каждого сервера есть **IP-адрес** — строка чисел, которая уникально размещает его в сети, подобно тому как почтовый адрес размещает здание в городе.

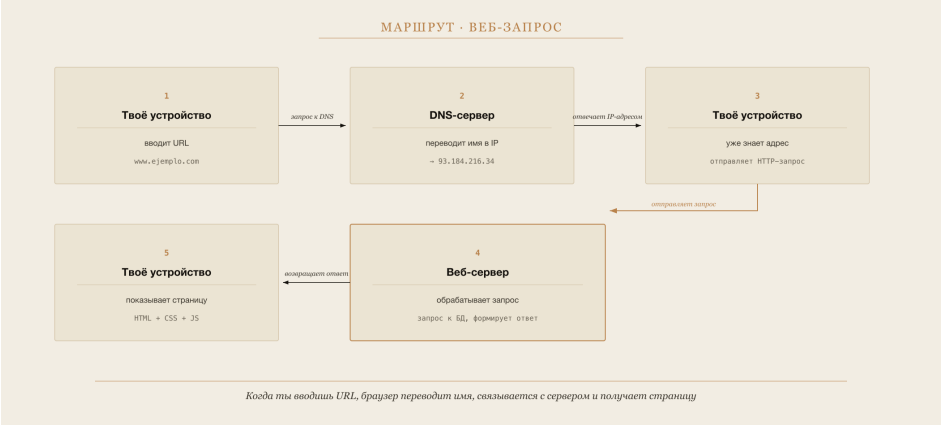

Рис. 1. *Когда ты вводишь адрес, браузер не идёт в волшебное облако: он запускает цепочку переводов, запросов и ответов.*

Проблема в том, что ты не вводишь числа. Ты вводишь `www.enciclopedia.org` или `noticias.ejemplo.com`. Чтобы браузер мог найти соответствующий сервер, ему нужно перевести это имя в IP-адрес. Этот процесс называется разрешением **DNS** — Domain Name System, система доменных имён.

DNS работает как телефонный справочник интернета. Браузер обращается к DNS-серверу и спрашивает: «какой IP у `noticias.ejemplo.com`?» DNS-сервер отвечает числом. Теперь у браузера есть реальный адрес.

С этим адресом браузер отправляет запрос на сервер. Этот запрос путешествует по сети — по кабелям, оптоволокну и беспроводным соединениям — пока не достигнет сервера, который его ждёт. Сервер получает его, обрабатывает то, что нужно обработать, и возвращает ответ. Этот ответ путешествует обратно к браузеру. Браузер интерпретирует содержимое и показывает его на экране.

Весь этот процесс — DNS-запрос, путь туда, обработка, ответ обратно — в нормальных условиях происходит за доли секунды.

Стоит упомянуть одну оптимизацию: **кэш**. Браузер не обращается к DNS каждый раз, когда ты заходишь на один и тот же сайт. Он сохраняет ответ — перевод имени в IP — на некоторое время. То же делает и операционная система. Из-за этого повторные визиты быстрее. Но это также означает, что если сервер недавно сменил IP-адрес, твоё устройство может всё ещё использовать старый адрес. Очистка кэша браузера или ожидание его истечения решает эту проблему.

Когда ты вводишь адрес, браузер не идёт в волшебное облако: он запускает цепочку переводов, запросов и ответов.

Что такое домен

Домен — это читаемое имя, которое идентифицирует сайт или сервис в интернете. `enciclopedia.org`, `noticias.ejemplo.com`, `miapp.io` — это домены.

Домены не существуют сами по себе: кто-то их регистрирует. Есть компании, специализирующиеся на регистрации доменов, — они называются регистраторами, — и домен арендуется на периоды времени, обычно на год. Если его не продлить, домен истекает и становится доступным для регистрации другим человеком.

Финальная часть домена — `.com`, `.org`, `.io`, `.ar` — называется доменом верхнего уровня, или TLD (Top Level Domain). Она указывает на тип организации или страну происхождения. Это различие имеет историческую и позиционную ценность, но технически не меняет работу системы.

Часть, которая может запутать, — это связь между доменом и сервером. Домен — это не сервер: это метка, которая указывает на сервер. Один и тот же сервер может иметь несколько доменов, указывающих на него. Один и тот же домен может перенаправлять на разные серверы в зависимости от времени суток или региона мира, откуда идёт обращение. DNS — это система, которая поддерживает эту связь между именами и адресами в актуальном состоянии.

Когда домен настроен неправильно — или когда он истёк, — DNS не может разрешить адрес. Браузер не знает, на какой сервер идти. Страница не загружается, хотя сервер, который её содержит, работает совершенно исправно.

Что такое сервер — и что такое хостинг

Сервер — это компьютер, спроектированный для того, чтобы получать запросы и возвращать ответы. По сути он не отличается от любого другого компьютера: у него есть процессор, память, хранилище и подключение к сети.

Практическая разница в том, что сервер включён постоянно, подключён к интернету с высокой доступностью и настроен на обслуживание множества одновременных запросов. У него нет ни экрана, ни клавиатуры. Он не стоит ни у кого на столе. Он живёт в

дата-центре — сооружении, спроектированном так, чтобы поддерживать работу множества серверов непрерывно и безопасно.

Хостинг — это сервис, который сдаёт тебе в аренду место и ресурсы на этих серверах. Когда ты «публикуешь» что-то в интернете — страницу, приложение, файл, — ты размещаешь этот контент на сервере хостинг-компании, настраиваешь домен, чтобы он указывал туда, и позволяешь серверу отдавать его тому, кто его запросит.

Есть разные формы хостинга. В самой базовой ты делишь сервер с другими сайтами — это дешевле, но с ограниченными ресурсами. В более продвинутых формах у тебя есть отдельный виртуальный или физический сервер — дороже, но с большим контролем и мощностью. В облачной модели ресурсы выделяются динамически в зависимости от спроса.

Для того, кто строит что-то цифровое, понимать, что такое хостинг, значит понимать, что созданная тобой система живёт в каком-то физическом месте, что у этого места есть пределы мощности и что если это место даёт сбой — или если домен указывает на него неправильно, — система недоступна, хотя она и построена совершенно исправно.

Полезное различие: веб-сервер — это компонент, который получает HTTP-запросы и возвращает контент. Сервер приложений — это тот, что выполняет логику системы, о чём подробнее пойдёт речь в Главе 3. Во многих простых системах обе функции работают на одной и той же машине. В более крупных системах они разделены. Пока важно то, что «сервер» — это не одна монолитная вещь: это набор частей, которые работают вместе, чтобы отвечать на запросы.

HTTP и HTTPS: язык веба

Когда браузер отправляет запрос, а сервер возвращает ответ, им обоим нужно говорить на одном языке. Этот язык называется **протоколом**. Протокол — это набор правил, который определяет, как форматируются, передаются и интерпретируются сообщения.

Протокол веба — **HTTP** — HyperText Transfer Protocol. Он определяет структуру запросов и ответов: какая информация идёт первой, как указывается тип контента, как сообщается результат операции.

Базовый HTTP-запрос включает: метод (что ты хочешь сделать — получить информацию, отправить данные, что-то удалить), адрес ресурса и заголовки, которые содержат

дополнительную информацию о том, кто спрашивает и какой тип ответа принимает. HTTP-ответ включает: код состояния (сработало или нет), заголовки с информацией о контенте и тело — сам контент.

HTTPS — это безопасная версия HTTP. «S» — от «Secure». Техническая разница в том, что в HTTPS вся коммуникация путешествует зашифрованной: данные, которые браузер отправляет на сервер, и те, что сервер возвращает, не могут быть прочитаны кем-то, кто перехватит соединение по пути.

Это шифрование делает возможным **сертификат SSL/TLS** — цифровой файл, который сервер предъявляет браузеру, чтобы доказать, что он тот, за кого себя выдаёт, и который активирует шифрование коммуникации. Современные браузеры показывают замочек в адресной строке, когда соединение использует HTTPS.

Почему это важно для того, кто строит: любая система, которая обрабатывает данные пользователей — пароли, личную информацию, платежи, — должна использовать HTTPS без исключений. Без HTTPS эти данные путешествуют по сети в открытом виде и могут быть перехвачены. При HTTP браузер может показать предупреждение «небезопасный сайт», которое отпугивает пользователей и снижает доверие к системе.

> ***Ключевое понятие: протокол*** *Набор правил, который определяет, как две стороны общаются. HTTP — это протокол, который используют браузеры и серверы для обмена информацией в вебе. HTTPS — его зашифрованная версия.*

Что отвечает сервер

Каждый раз, когда сервер получает запрос, он возвращает ответ, который включает число: **код состояния**. Это число указывает, что произошло с запросом.

Коды группируются по категориям. Те, что начинаются с 2, означают успех. Те, что начинаются с 3, — перенаправления. Те, что начинаются с 4, указывают на ошибку на стороне клиента — что-то в запросе не так. Те, что начинаются с 5, указывают на ошибку на стороне сервера — сервер получил запрос, но не смог его корректно обработать.

Три самых важных для того, кто работает с системами:

200 — OK. Запрос обработан корректно, и сервер вернул ожидаемый контент. Это нормальный результат. Если ты видишь страницу, сервер ответил 200.

404 — Not Found. Сервер получил запрос, но ресурса, который ты запросил, на этом сервере нет. Адрес, который ты использовал, не соответствует ни одному доступному файлу, странице или элементу данных. Это не ошибка сервера: просто того, что ты искал, там нет.

500 — Internal Server Error. Сервер получил запрос, попытался его обработать, и что-то дало сбой в его собственной работе. Проблема не в том, что ты запросил: она в том, как сервер обработал запрос. Это ошибка кода или конфигурации сервера.

Код	Название	Что произошло	Бытовой пример
200	OK	Сервер нашёл и отдал то, что ты запросил.	Ты открываешь страницу, и всё загружается без проблем.
301 / 302	Перенаправление	Ресурс находится по другому адресу. Браузер автоматически ведёт тебя туда.	Старый URL отправляет тебя на новый домен сайта.
400	Bad Request	Запрос пришёл искажённым; сервер не смог его интерпретировать.	Ты отправляешь форму с незаполненными обязательными данными.
401	Unauthorized	Ты не предъявил действительных учётных данных. Сервер не знает, кто ты.	Ты пытаешься посмотреть приватный контент, не войдя в аккаунт.
403	Forbidden	Твои учётные данные действительны, но у тебя нет прав на это.	Ты вошёл в свой аккаунт, но этот раздел не соответствует твоей роли.

Код	Название	Что произошло	Бытовой пример
404	Not Found	То, что ты запросил, не существует на этом сервере.	URL был написан с ошибкой или больше не существует.
500	Internal Server Error	Сервер получил твой запрос, но дал сбой при его обработке.	Ты пытаешься завершить покупку, и появляется «неожиданная ошибка».
503	Service Unavailable	Сервер не может обслужить сейчас; он перегружен или на обслуживании.	Сайт падает во время массовой распродажи или запуска.

Не нужно запоминать все коды. Важно распознать закономерность: 2xx обычно означают успех, 3xx — перенаправление, 4xx — проблему на стороне запроса, а 5xx — проблему на стороне сервера.

Почему приложение «упало»

«Страница не загружается», «система лежит», «не работает» — когда что-то перестаёт быть доступным, обычно есть конкретная техническая причина. Понимание возможных точек отказа позволяет сузить проблему прежде, чем звонить тому, кто её решит.

Самые частые причины, упорядоченные по тому, где они возникают:

Сервер лежит. Сервер перестал отвечать. Это может быть из-за того, что у хостинг-сервиса возникла проблема, из-за того, что сервер перезагрузился ради обновления, или из-за того, что системе не хватило ресурсов (памяти, CPU) и она перестала работать. В этом случае браузер не получает никакого ответа.

Домен не разрешается. DNS не может перевести имя домена в IP-адрес. Это может быть из-за того, что домен истёк, из-за того, что конфигурацию DNS изменили неправильно, или из-за проблемы с DNS-сервером. Результат похож на предыдущий: браузер не знает, куда идти.

Истёк сертификат. Если HTTPS-сертификат сервера истёк, браузер блокирует соединение и показывает предупреждение безопасности. Технически страница существует и сервер работает, но браузер защищает пользователя, отклоняя соединение. Видимый результат: экран с предупреждением, а не страница.

Ошибка 500. Сервер отвечает, но внутренней ошибкой. Страница существует, сервер включён, но код столкнулся с проблемой при обработке запроса. Пользователь видит сообщение об ошибке вместо ожидаемого контента.

Проблема с сетью. Соединение между устройством пользователя и сервером прервано в какой-то промежуточной точке. Это может быть интернет-подключение пользователя, сетевой провайдер или точка инфраструктуры между двумя сторонами.

Поверхностное различение помогает понять, с чего начать:

- Если страница не загружает ничего и предупреждения нет → возможно, сервер лежит или DNS не разрешается.
- Если браузер показывает явное предупреждение безопасности → истёкший сертификат или HTTP без шифрования.
- Если страница загружается с сообщением об ошибке в контенте → вероятно, ошибка 500, проблема кода.
- Если только ты не можешь получить доступ, а другие могут → возможно, проблема с сетью или с локальным кэшем.

Знать это не значит уметь решить это напрямую. Это значит уметь точно сообщить об этом тому, кто действительно может решить.

> ***Полезные вопросы, когда сайт не загружается***
>
> *Когда система недоступна, эти вопросы помогают сузить проблему прежде, чем звонить тому, кто может её решить:*
>
> - *Проблема у всех или только у меня?*

> - *Домен разрешается? Имя указывает на какой-то адрес?*
> - *Сервер отвечает? Приходит хоть что-то, пусть даже ошибка?*
> - *Какой код ошибки — 404, 500 — или нет никакого ответа?*
> - *Даёт сбой продакшн или только тестовое окружение?*
> - *Есть ли логи ошибки?*
> - *Было ли какое-то недавнее изменение — деплой, изменение конфигурации, продление домена?*

Есть один случай, который стоит упомянуть, потому что он частый и запутанный: система работает для одних людей и не работает для других. Это почти никогда не означает, что сервер лежит. Обычно это указывает на проблему распространения DNS — когда сделали изменение в конфигурации домена, это изменение доходит до всех DNS-серверов мира от нескольких минут до нескольких часов. В течение этого времени разные пользователи обращаются к разным DNS-серверам и получают разные ответы. Одни видят новый сайт; другие продолжают видеть старый или не видят ничего. Это ожидаемое поведение, а не ошибка, которую нужно срочно решать.

Как ты услышишь это на совещании

На совещании с технической командой или с поставщиком используются фразы, которые предполагают, что все понимают, о чём речь. Вот самые частые по этой теме, что они значат на практике и что ты можешь спросить.

«Похоже на проблему с DNS». Имя домена не разрешается корректно — его нельзя перевести в IP-адрес сервера. Это может быть неправильная конфигурация или недавнее изменение, которое ещё не распространилось полностью.

Полезный вопрос: «Это происходит у всех пользователей или только у некоторых?» Если у всех — вероятно, это ошибка конфигурации. Если только у некоторых — скорее всего, идёт распространение, процесс, который занимает часы и разрешается сам.

Не стоит предполагать: что сервер лежит. DNS и сервер — разные части. Сервер может работать совершенно исправно, пока DNS даёт сбой.

«Сервер отвечает 500». Сервер получил запрос, но столкнулся с ошибкой при его обработке. Проблема в коде или в конфигурации системы, а не в запросе пользователя и не в сети.

Полезный вопрос: «Есть ли логи ошибки?» Логи — это запись того, что делала система перед сбоем. 500 почти всегда оставляет след, который выявляет причину. Без логов диагностика становится намного труднее.

Не стоит предполагать: что это проблема с интернетом или с устройством пользователя. 500 означает, что сервер действительно ответил — просто внутренней ошибкой.

«Сертификат истёк». HTTPS-сертификат сервера истёк. Браузер обнаруживает, что соединение больше нельзя гарантировать как безопасное, и по умолчанию блокирует доступ. Система может работать совершенно исправно внутри, но ни один пользователь не может войти.

Полезный вопрос: «Сколько занимает продление?» В большинстве случаев это быстрый процесс. Важно знать оценочное время решения и есть ли уже затронутые пользователи, которым нужно что-то сообщить.

Не стоит предполагать: что есть проблема с кодом или инфраструктурой. Это административное истечение, похожее на истечение срока действия документа. Оно ничего не говорит о качестве системы.

«Лежит в продакшене». Система, которой пользуются реальные пользователи, недоступна. Это не тестовое и не девелоперское окружение: это то, где реальный трафик и реальные данные. Эта фраза указывает на срочность.

Полезный вопрос: «Над решением уже работают? Каково оценочное время?» Эти два вопроса дают контекст, необходимый, чтобы понять, нужно ли что-то сообщать пользователям или ждать молча.

Не стоит предполагать: что проблема одинаково затрагивает всех. Иногда падение частичное — затрагивает только один регион, устройство или конкретную функцию.

Что ты узнал в этой главе

- URL — это адрес, который идентифицирует ресурс в интернете. Чтобы добраться до него, браузер обращается к DNS, который переводит имя домена в IP-адрес.
- Сервер — это всегда включённый компьютер, который получает запросы и возвращает ответы. Хостинг — это сервис, который сдаёт этот сервер в аренду.

- HTTP — это протокол, который определяет, как общаются браузер и сервер. HTTPS — его зашифрованная версия, обязательная для любой системы, которая обрабатывает чувствительные данные.
- Коды состояния — это язык, которым сервер сообщает, что произошло: 200 — успех, 404 — ресурс не найден, 500 — внутренняя ошибка сервера.
- «Приложение упало» имеет конкретные технические причины: сервер лежит, DNS не разрешается, истёкший сертификат, ошибка кода. Знание их позволяет сузить проблему.

Что дальше

Глава 3 входит в двух действующих лиц того разговора, который ты только что понял: клиент и сервер. Ты уже знаешь, как между ними путешествует информация. Дальше — понять, что делает каждый из них в этом обмене, — и почему это разделение является фундаментальной структурой любого приложения.

Этот фрагмент заканчивается здесь.

Полная книга продолжается полной картой:

клиент, сервер, базы данных, код,

API, безопасность, искусственный интеллект и деплой.

Полное издание включает

22 главы, практические приложения,

технический словарь и версии PDF + EPUB.