

ОСНОВЫ HTTP

OBI MODEL, TCP/IP, CLIENT-SERVER ARCHITECTURE, HTTP PROTOCOL, HTTP STREAM, HTTP METHODS, HTTP CODES



МОДЕЛЬ OSI (OPEN SYSTEMS INTERCONNECTION)

7 Прикладной (Application Layer)

6 Представления (Presentation Layer)

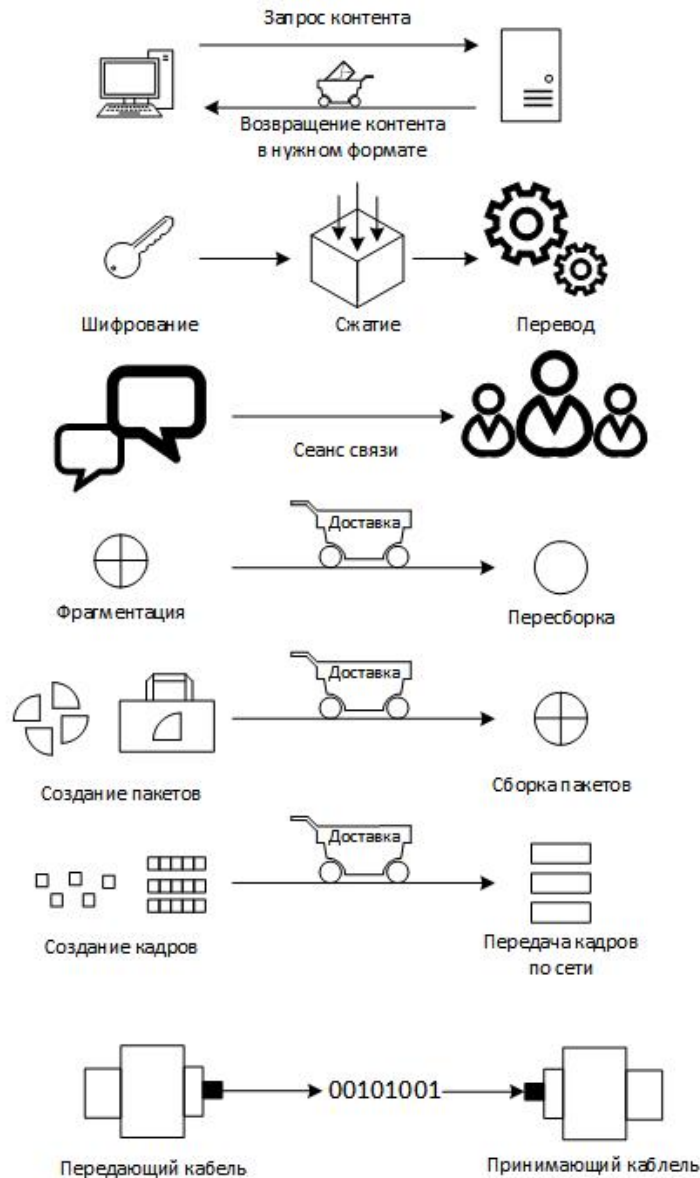
5 Сеансовый (Session Layer)

4 Транспортный (Transport Layer)

3 Сетевой (Network Layer)

2 Канальный (Data Link Layer)

1 Физический (Physical Layer)



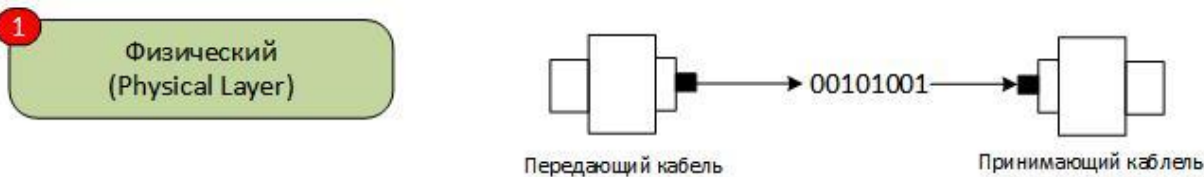
OSI — это образец эталонного процесса передачи данных по сети. В нем пошагово описаны этапы, которые проходит информация по ходу движения от отправителя к получателю.

Модель **OSI** состоит из семи слоев, где на самом нижнем слое (физическом) информация представлена в виде бит. А на самом верхнем слое (прикладном) в виде данных.

В процессе передачи данные проходят все слои, двигаясь от устройства-отправителя к устройству-получателю. При этом при отправке они инкапсулируются, двигаясь от седьмого к первому слою и превращаясь из данных в биты, а при получении декапсулируются — трансформируются обратно из битов в данные.



МОДЕЛЬ OSI (OPEN SYSTEMS INTERCONNECTION)



На первом слое идет работа с физическими сигналами. Эти сигналы передаются от устройства-отправителя к устройству-получателю либо по проводам и кабелям, либо «по воздуху» — через Wi-Fi, Bluetooth, GSM и другие беспроводные сети.



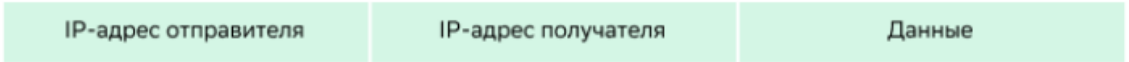
На следующем слое данные проверяются на целостность и отсутствие ошибок, распознаются как биты и упаковываются в специальные блоки – кадры (frames).



Структура кадра



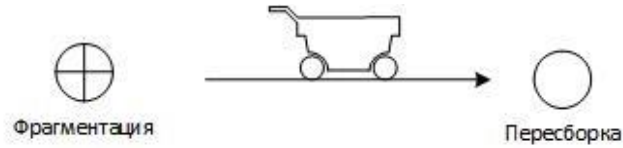
На сетевом слое данные маршрутизируются между устройствами в сети. Для этого используются маршрутизаторы — физические устройства, которые соединяют сети и управляют трафиком между ними.



Структура кадра

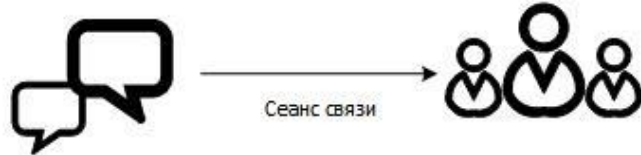
МОДЕЛЬ OSI (OPEN SYSTEMS INTERCONNECTION)

4 Транспортный (Transport Layer)



Транспортный слой отвечает за передачу данных по сети. На нем задействованы два ключевых сетевых протокола — TCP и UDP. Они транспортируют пакеты, созданные на предыдущем слое.

5 Сеансовый (Session Layer)



Сеансовый слой управляет сеансами связи, которые можно синхронизировать для параллельного обмена информацией.

6 Представления (Presentation Layer)



На шестом слое данные преобразуются, кодируются и сжимаются, а также шифруются при необходимости. Здесь уже идет работа с привычными нам форматами данных — JPEG, MPEG, MP3 и так далее.

7 Прикладной (Application Layer)



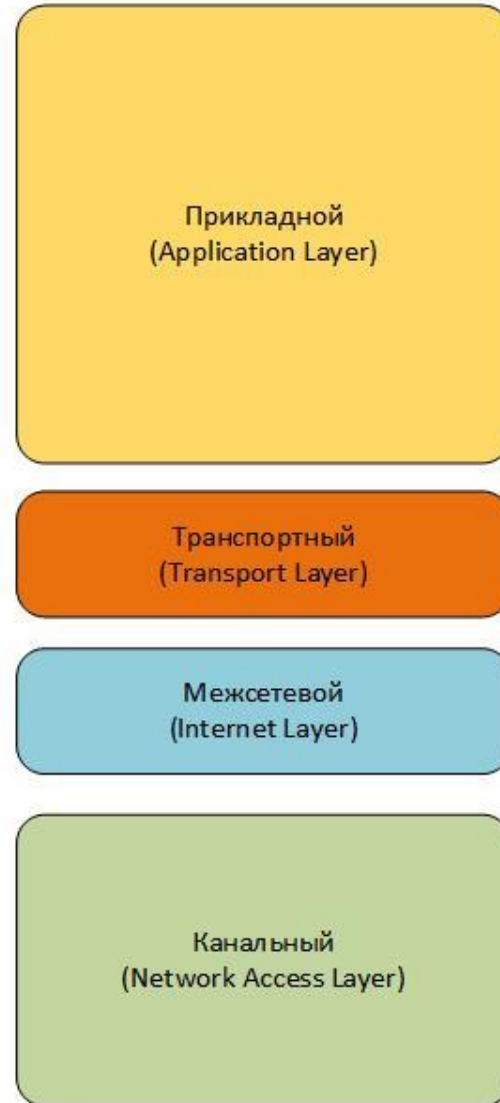
Самый верхний седьмой слой OSI — тот, на котором с данными взаимодействуют пользователи. Он представляет собой своего рода графический интерфейс, а его основная задача заключается в представлении информации в таком виде, которая будет понятна человеку.

МОДЕЛЬ TCP/IP

OSI



TCP/IP



OSI закрепилась в истории как теоретический стандарт для выстраивания компьютерных сетей, облачных сервисов и совместного использования сетевых ресурсов. На практике же применение нашли другие модели.

Модель TCP/IP (Transmission Control Protocol и Internet Protocol).

Эта модель, которая состоит всего из четырех слоев — канального, межсетевого, транспортного и прикладного. Благодаря своей простоте и скорости внедрения именно она смогла стать основой того, что сегодня мы знаем как интернет.

Особенности модели:

- Первый уровень – канальный. В данной модели он объединяет L1 и L2 уровни модели OSI.
- Второй уровень – межсетевой. Он идентичен сетевому уровню L3 модели OSI.
- Третий уровень – транспортный. Он идентичен транспортному уровню L4 модели OSI.
- Четвертый уровень – прикладной. Он объединяет L5 – L7 уровни модели OSI.

МОДЕЛЬ **TCP/IP**. СЕТЕВЫЕ ПРОТОКОЛЫ

■ UDP

User Datagram Protocol - представляет собой протокол передачи данных, не требующий предварительной установки соединения между хостами. Он быстрый, но часто теряет пакеты данных во время доставки.

■ TCP

Transmission Control Protocol — протокол сквозной связи, созданный в 1974 году и до сих пор востребованный в мире. Надёжный, но медленный. Перед тем, как начинать передачу данных, достигается рукопожатие для установления соединения, и лишь затем начинается передача пакетов. При необходимости пакеты дублируются.

■ FTP

File Transfer Protocol — протокол прикладного уровня для передачи файлов, появившийся в 1971 году. Использует два канала для передачи данных. Первый, управляющий процессом передачи, называют командным. Вторым, передающим информацию — транспортным. FTP — надёжный протокол. Сервер в данном случае называют удалённым хостом, а клиент — локальным. Работает по клиент-серверной модели. После аутентификации пользователь получает доступ к файловой системе. В некоторых случаях возможен анонимный доступ.

МОДЕЛЬ TCP/IP. СЕТЕВЫЕ ПРОТОКОЛЫ

■ RTP

Real-time Transfer Protocol — транспортный протокол, работающий в реальном времени. Нужен для потоковой передачи аудио- и видеоданных. Умеет передавать данные нескольким получателям одновременно. Чаще всего применяется для передачи голоса в IP-телефонии.

■ HTTP

HyperText Transfer Protocol — протокол, который создавался для передачи HTML-файлов, но впоследствии пригодился для всех файлов в сети. Этот протокол клиент-серверного типа не сохраняет промежуточное состояние. Для обмена информацией обычно используется TCP/IP. Также существует защищенная версия HTTPS. Она поддерживает шифрование, а данные передаются поверх криптопротокола TLS.

■ SSH

Secure Shell — защищённый протокол прикладного уровня, необходимый для удалённого управления ОС через протокол TCP. В SSH весь трафик шифруется с использованием заданного вами алгоритма шифрования. Протокол позволяет обрабатывать другие сетевые протоколы передачи, включая аудио- или видеопоток. Его удобно использовать для удалённого подключения клиента к серверу и работы оттуда.

КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА



Клиент-серверная архитектура — это архитектура, в которой задания или сетевая нагрузка распределены между клиентами и серверами.

Клиент — это приложение или устройство, которое запрашивает ресурс.

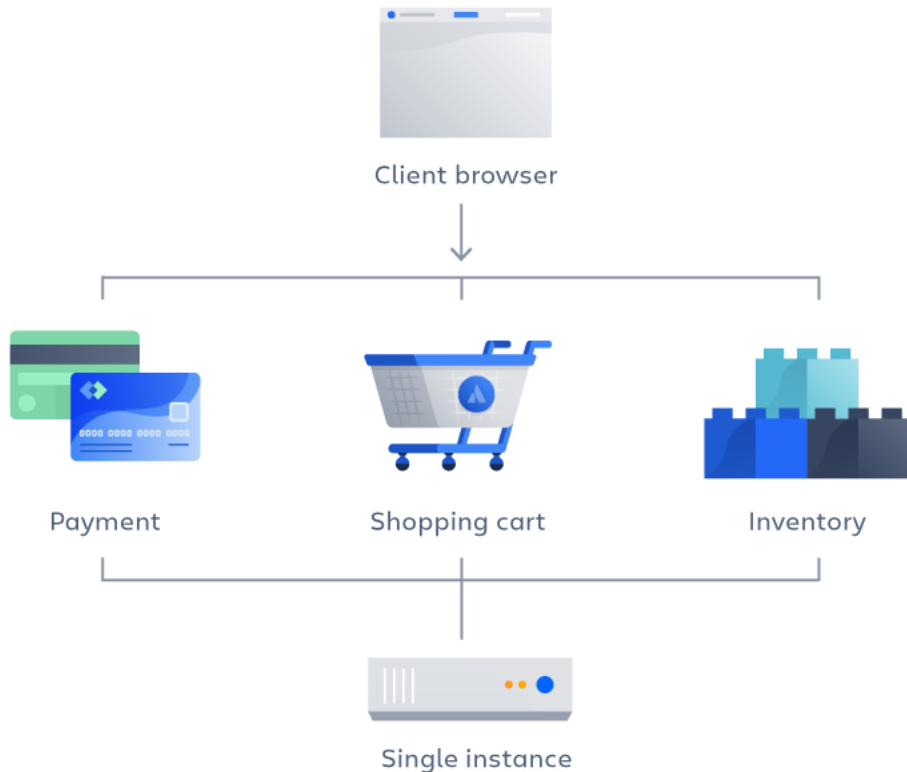
Сервер — это удалённый компьютер или система, которая предоставляет услуги клиентам.

Процесс клиент-серверной архитектуры:

1. Запрос клиента. Пользователь запускает ресурс, инициируя запрос к серверу.
2. Отправка запроса. Ресурс формирует запрос и направляет его на адрес сервера.
3. Приём запроса сервером. Получив запрос, сервер проверяет его.
4. Обработка запроса. Сервер проводит соответствующие операции.
5. Формирование ответа. Обработав запрос, сервер создаёт ответ и отправляет его обратно.
6. Получение ответа клиентом. Приложение получает ответ от сервера и отображает его пользователю.
7. Повторение цикла.

МОНОЛИТНАЯ АРХИТЕКТУРА

Monolithic architecture



Монолитная архитектура - это традиционная модель программного обеспечения, которая построена как унифицированная единица, которая автономна и независима от других приложений.

Преимущества:

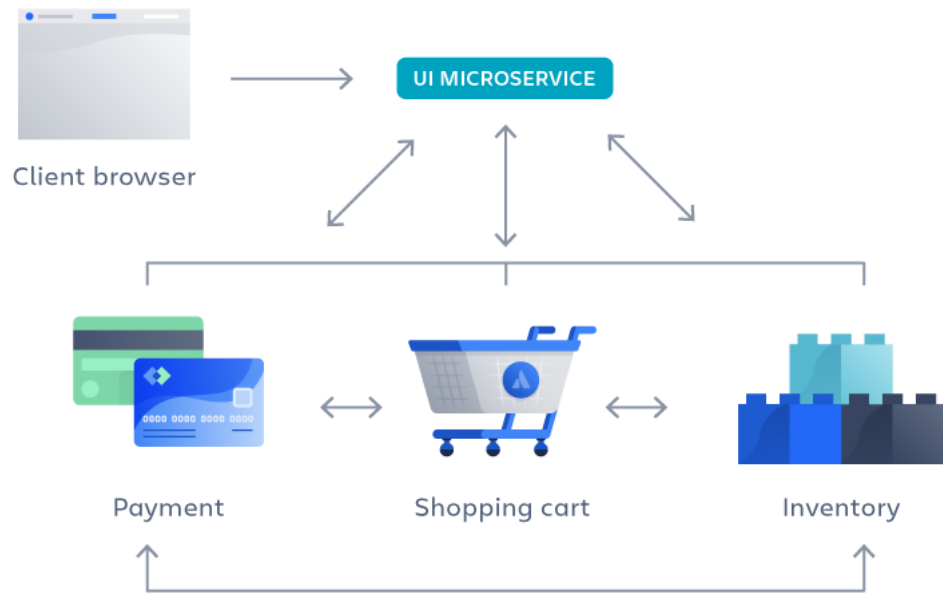
- Простое развертывание
- Простая разработка
- Меньшее количество API
- Упрощенное тестирование
- Простая отладка

Недостатки:

- Медленная разработка
- Сложная масштабируемость
- Низкая надежность
- Дорогостоящие изменения
- Отсутствие гибкости

МИКРОСЕРВИСНАЯ АРХИТЕКТУРА

Microservice architecture



Архитектура микросервисов является архитектурным методом, который опирается на ряд независимо разворачиваемых услуг. Эти сервисы имеют свою бизнес-логику и базу данных с определенной целью. Обновление, тестирование, развертывание и масштабирование происходят в каждой службе.

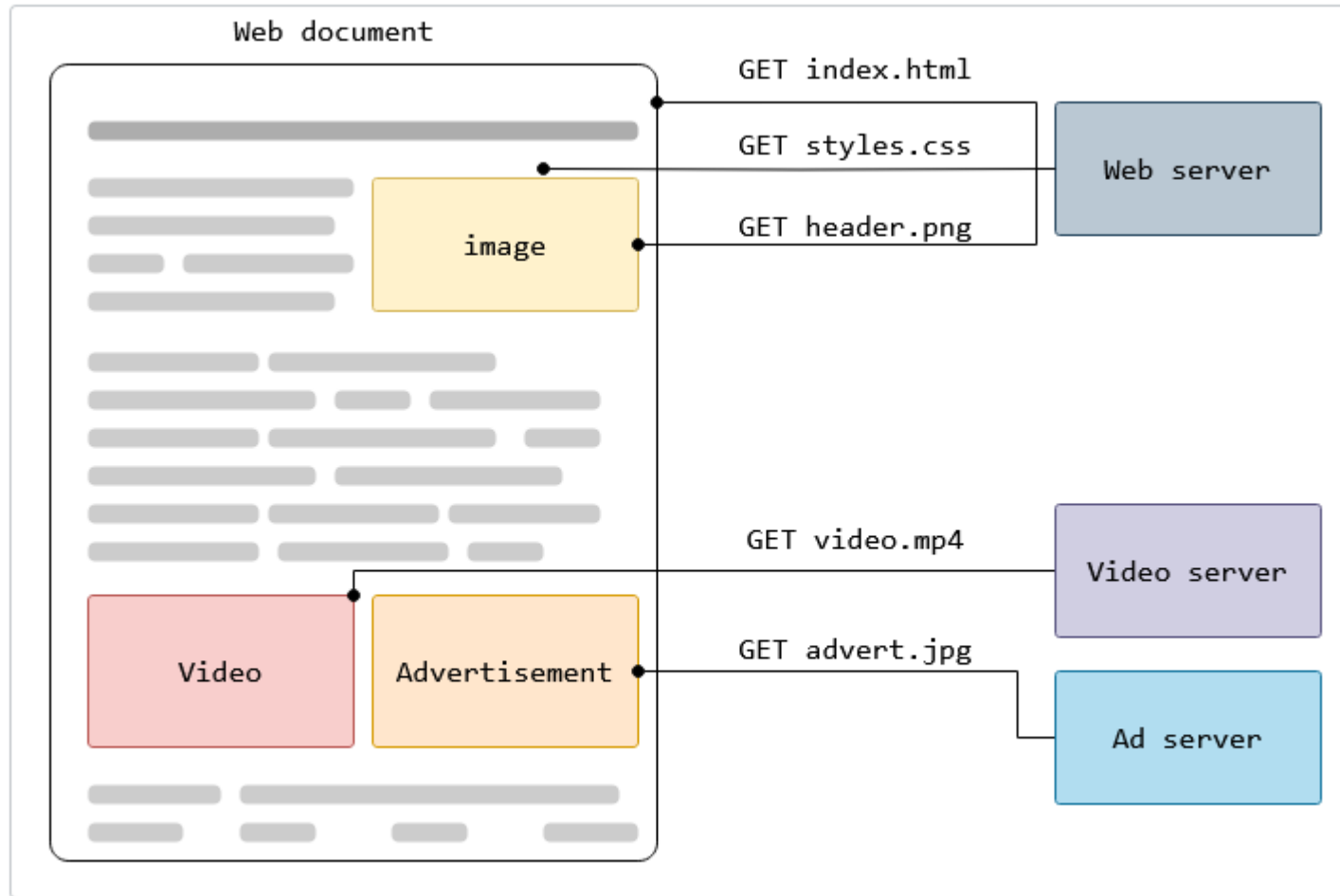
Преимущества:

- Гибкое масштабирование
- Непрерывное развертывание
- Скорость разработки
- Гибкость технологий
- Высокая надежность

Недостатки:

- Затраты на инфраструктуру
- Организационные накладные расходы
- Усложнение отладки

ПРОТОКОЛ HTTP (HYPERTEXT TRANSFER PROTOCOL)



HTTP — это протокол для получения ресурсов, например, HTML-документов. Он лежит в основе обмена данными в Интернете и является протоколом клиент-серверного взаимодействия, что означает инициирование запросов к серверу самим получателем, обычно веб-браузером.

HTTP является расширяемым протоколом. Это протокол прикладного уровня, который использует соединение TCP (или TLS – защищенный TCP) для пересылки сообщений. Благодаря своей расширяемости, HTTP используется не только для получения гипертекстовых документов, но и для изображений и видео, а также для отправки содержимого серверам.

HTTP-ПОТОК

1. Открытие TCP соединения
2. Отправка HTTP-сообщения

```
GET / HTTP/1.1
Host: developer.mozilla.org
Accept-Language: fr
```

3. Чтение ответа от сервера

```
HTTP/1.1 200 OK
Date: Sat, 09 Oct 2010 14:28:02 GMT
Server: Apache
Last-Modified: Tue, 01 Dec 2009 20:18:22 GMT
ETag: "51142bc1-7449-479b075b2891b"
Accept-Ranges: bytes
Content-Length: 29769
Content-Type: text/html

<!doctype html>... (here come the 29769 bytes of the requested web page)
```

4. Закрытие или переиспользование соединения для следующих запросов

ЗАПРОС

Method Path Protocol version

GET

/

HTTP/1.1

Host: developer.mozilla.org
Accept-Language: fr

Headers

ОТВЕТ

Protocol version Status code Status message

HTTP/1.1

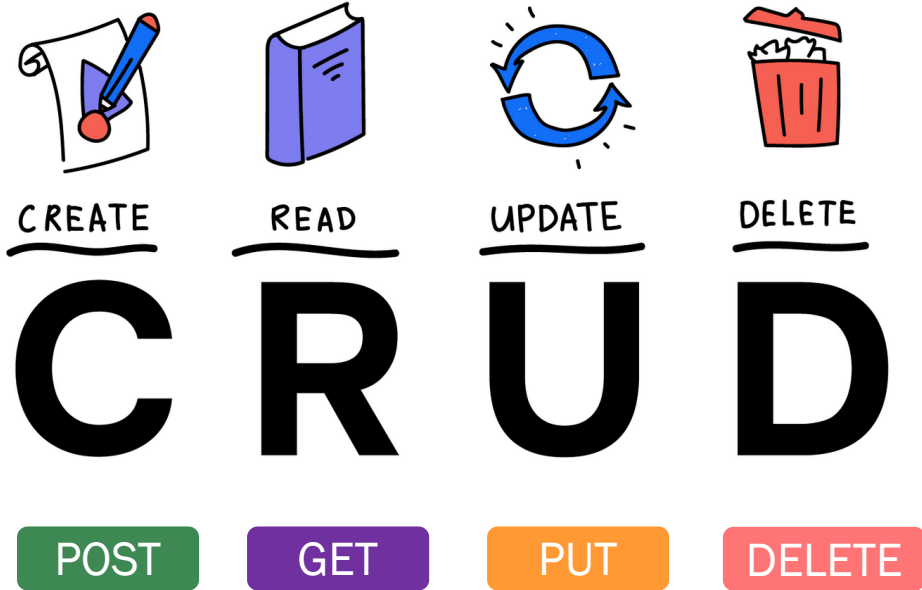
200

OK

date: Tue, 18 Jun 2024 10:03:55 GMT
cache-control: public, max-age=3600
content-type: text/html

Headers

HTTP-МЕТОДЫ



В программировании часто используется аббревиатура CRUD. Она обозначает четыре базовых операции над информацией:

- операции создания — метод POST
- операции чтения — метод GET
- операции редактирования — PUT или PATCH
- операции удаления — метод DELETE.

HTTP определяет множество методов запроса, которые указывают, какое желаемое действие выполнится для данного ресурса. Каждый реализует свою семантику, но каждая группа команд разделяет общие свойства:

■ Безопасность

Метод HTTP является безопасным, если он не изменяет состояние сервера. Другими словами, безопасный метод производит операции "только для чтения".

■ Идемпотентность

Метод HTTP является идемпотентным, если повторный идентичный запрос, сделанный один или несколько раз подряд, имеет один и тот же эффект, не изменяющий состояние сервера.

■ Кэшируемость

Кэшируемые ответы - это HTTP-ответы, которые могут быть закэшированы, то есть сохранены для дальнейшего восстановления и использования позже, тем самым снижая число запросов к серверу.

HTTP-МЕТОДЫ

GET

```
GET /http-api/tasks HTTP/1.1
HOST: http.hexlet.app
```

POST

```
POST /http-api/tasks HTTP/1.1
HOST: http.hexlet.app
Content-Length: 53
Content-Type: application/json

{"title":"new task","description":"task description"}
```

■ GET — получение данных

Метод предназначен для запроса данных с сервера.

При использовании GET-запроса параметры передаются через строку запроса в URL: можно закешировать запросы и делать их повторно без отправки данных.

Повторное выполнение запроса не изменит состояние сервера.

Не подходит для передачи больших объемов данных или конфиденциальной информации, т. к. содержимое отображается в адресной строке.

■ POST — отправка данных

Метод используется для отправки данных на сервер, чтобы создать или обновить ресурсы.

Повторная отправка POST-запроса может привести к созданию нескольких одинаковых ресурсов или повторному выполнению операции.

POST используется при отправке форм на веб-сайтах, загрузке файлов, в API-запросах. Он поддерживает передачу сложных структур данных, включая JSON или XML.

HTTP-МЕТОДЫ

PUT

```
PUT /http-api/tasks/1 HTTP/1.1
HOST: http.hexlet.app
Content-Length: 21
Content-Type: application/json

{"title":"new title"}
```

DELETE

```
DELETE /http-api/tasks/1 HTTP/1.1
HOST: http.hexlet.app
```

■ PUT — обновление данных

Применяется для создания нового ресурса или полного обновления существующего на сервере по указанному URI.

Клиент отправляет запрос с полным представлением ресурса, которое сервер должен сохранить. Если ресурс по указанному URI не существует, сервер создаст его.

■ DELETE — удаление данных

Используется для удаления ресурса по URI. При успешной обработке запроса веб-страница становится недоступной.

Как и PUT, метод DELETE идемпотентный: повторные запросы удаления не приведут к ошибке, хотя сервер может сообщить, что ресурс уже недоступен.

HTTP-КОДЫ СОСТОЯНИЯ ОТВЕТА

- Информационные ответы (100 - 199)
- Успешные ответы (200 - 299)
- Сообщения о перенаправлении (300 - 399)
- Ошибки клиента (400 – 499)
- Ошибки сервера (500 – 599)

■ 200 OK

Запрос успешно выполнен. Значение результата «успех» зависит от метода HTTP:

GET: Ресурс был получен и передан в теле сообщения.

PUT или **POST:** Ресурс, описывающий результат действия, передан в теле сообщения.

■ 400 Bad Request

Сервер не может или не будет обрабатывать запрос из-за чего-то, что воспринимается как ошибка клиента

■ 404 Not Found

Сервер не может найти запрошенный ресурс

■ 500 Internal Server Error

На сервере произошла ошибка, в результате которой он не может успешно обработать запрос

■ 503 Service Unavailable

Сервер не готов обработать запрос в данный момент

OPENROUTER НАЗНАЧЕНИЕ И МОДЕЛЬ РАБОТЫ

<https://openrouter.ai>

The Unified Interface For LLMs

Better **prices**, better **uptime**, no subscription.

Start a message...



Featured Models

[View Trending](#)

Gemini 2.5 Pro

by **google**

143.7B

Tokens/wk

2.5s

Latency

-6.82%

Weekly growth

GPT-5

by **openai**

157.1B

Tokens/wk

7.4s

Latency

+20.69%

Weekly growth

Claude Sonnet 4.5

by **anthropic**

2.6B

Tokens/wk

2.1s

Latency

--

Weekly growth

12T

Monthly Tokens

4.2M+

Global Users

60+

Active Providers

500+

Models

OpenRouter это сервис единого окна к различным поставщикам ИИ моделей - ChatGPT, Claude и др. Эта платформа разработана для упрощения интеграции различных моделей искусственного интеллекта в приложения и проекты. Основная цель **OpenRouter** — предоставить централизованный доступ к множеству языковых моделей от различных поставщиков, таких как OpenAI, Anthropic, Google и других.

Ключевые особенности:

- Единый API-доступ
- Интерактивная площадка
- Интеграция с популярными фреймворками

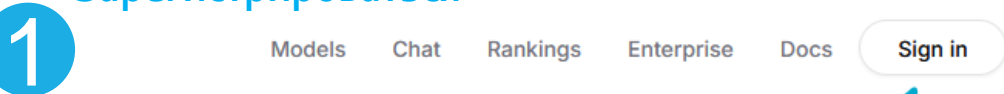
Ограничения бесплатного API:

- 20 запросов в минуту
- 50 запросов в день

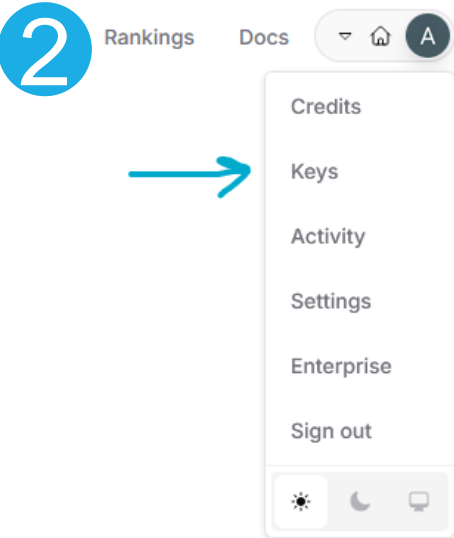
OPENROUTER АВТОРИЗАЦИЯ И ПОЛУЧЕНИЕ ТОКЕНА

https://openrouter.ai

Зарегистрироваться



Перейти на вкладку Keys



Создать ключ API

API Keys

3



Create API Key

Ввести имя ключа и нажать Create

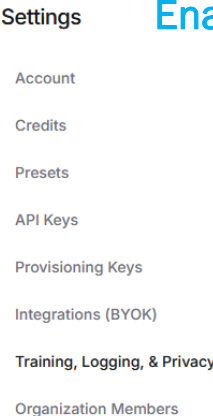
4

A screenshot of the 'Create Key' form in the OpenRouter interface. The 'Name' field is filled with 'botkey'. Below it are fields for 'Credit limit (optional)' and 'Reset limit every...'. A blue circle with the number '4' is next to the 'Name' field. A blue arrow points to the 'Create' button at the bottom right.

После создания
сохранить себе
значение ключа

Перейти в Settings -> Training, Logging, & Privacy и нажать флажок
Enable free endpoints that may publish prompts

5



Training, Logging, & Privacy



A screenshot of the 'Training, Logging, & Privacy' settings page. It shows several privacy settings with toggle switches. The 'Enable free endpoints that may publish prompts' toggle is highlighted by a blue arrow. A blue circle with the number '5' is next to the settings list.



OPENROUTER БЕСПЛАТНЫЕ МОДЕЛИ

https://openrouter.ai/models?max_price=0&q=free

Models

51 models Reset Filters



Sort  

TNG: DeepSeek R1T2 Chimera (free)



70,8B tokens

DeepSeek-TNG-R1T2-Chimera is the second-generation Chimera model from TNG Tech. It is a 671 B-parameter mixture-of-experts text-generation model assembled from DeepSeek-AI's ...



by [tngtech](#) | 164K context | \$0/M input tokens | \$0/M output tokens

DeepSeek: DeepSeek V3.1 (free)



66,6B tokens

DeepSeek-V3.1 is a large hybrid reasoning model (671B parameters, 37B active) that supports both thinking and non-thinking modes via prompt templates. It extends the DeepSeek-V3 base wi...

by [deepseek](#) | 164K context | \$0/M input tokens | \$0/M output tokens


Z.AI: GLM 4.5 Air (free)



36,1B tokens

GLM-4.5-Air is the lightweight variant of our latest flagship model family, also purpose-built for agent-centric applications. Like GLM-4.5, it adopts the Mixture-of-Experts (MoE) architecture bu...



by [z-ai](#) | 131K context | \$0/M input tokens | \$0/M output tokens


DeepSeek: DeepSeek V3.1 (free)

deepseek/deepseek-chat-v3.1:free 

Created Aug 21, 2025 | 163,800 context | \$0/M input tokens | \$0/M output tokens


DeepSeek-V3.1 is a large hybrid reasoning model (671B parameters, 37B active) that supports both thinking and non-thinking modes via prompt templates. It extends the DeepSeek-V3 base with a two-phase long-context training process, reaching up to 128K tokens, and uses FP8 microscaling






Free 


Model weights 

- Overview
- Providers
- Performance
- Apps
- Activity
- Uptime
- API

Providers for DeepSeek V3.1 (free)

OpenRouter [routes requests](#) to the best providers that are able to handle your prompt size and parameters, with fallbacks to maximize [uptime](#). 

OpenInference						Latency	Throughput	Uptime
	US	int8				2,61s	30,30tps	
Total Context	Max Output		Input Price	Output Price	Cache Read	Cache Write	Input Audio	Input Audio Cache
163.8K	163.8K		--	--	--	--	--	--