PROGRAMAÇÃO CONCORRENTE E DISTRIBUÍDA

Introdução à Comunicação e REST

Aula 07 — Protocolos, tipos de comunicação e arquitetura REST

Prof. Newton S. B. Miyoshi

newton.miyoshi@baraodemaua.br

Centro Universitário Barão de Mauá — Ribeirão Preto

PARTE 1

Camadas de Protocolos

Como processos combinam o formato das mensagens que trocam.

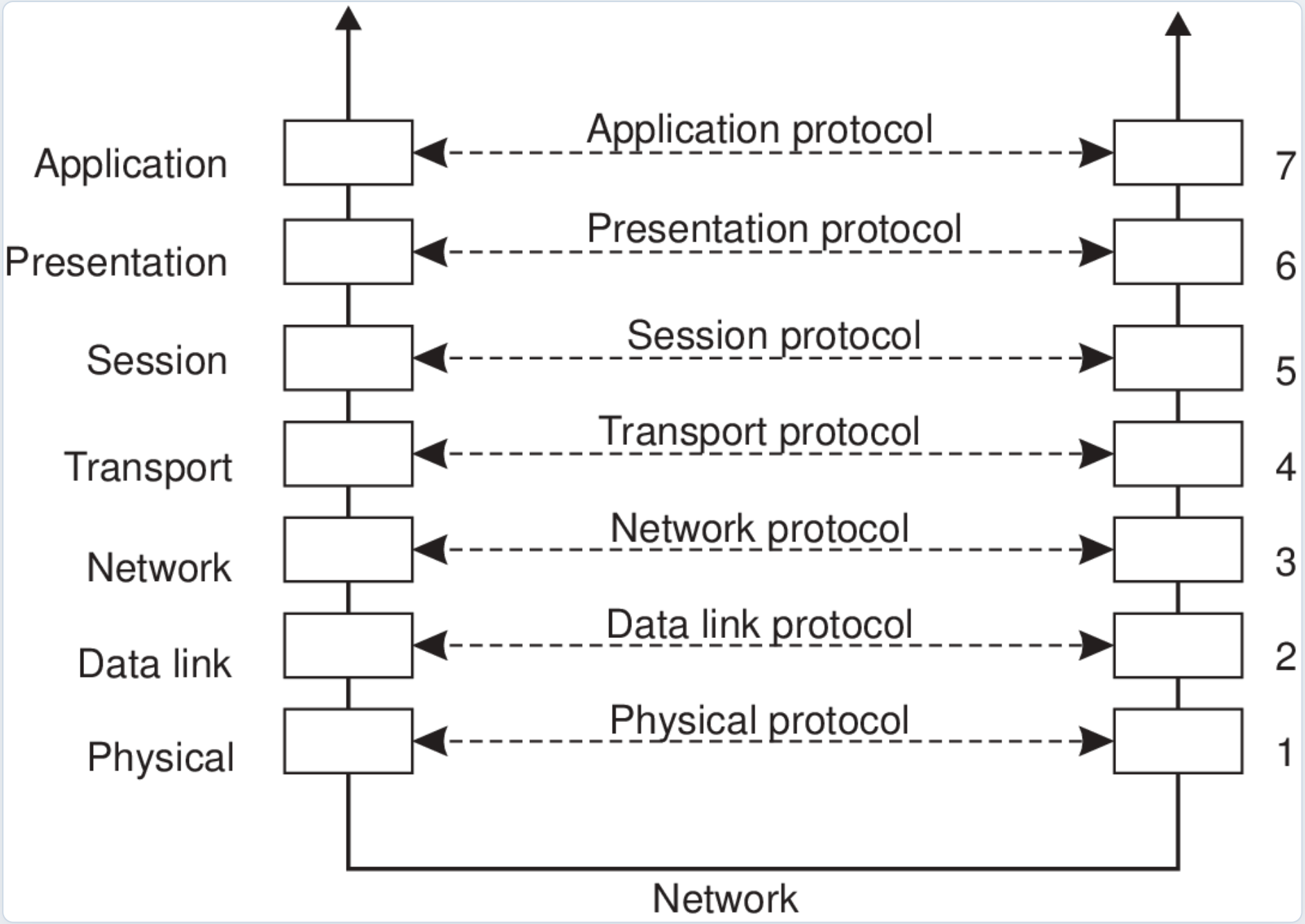
Comunicação — Introdução

- Programação distribuída depende de comunicação interprocesso — troca de informação.
- Trocar mensagens em uma rede é mais difícil do que usar primitivas em memória compartilhada.
- A base das comunicações em sistemas distribuídos são os protocolos de comunicação em rede.

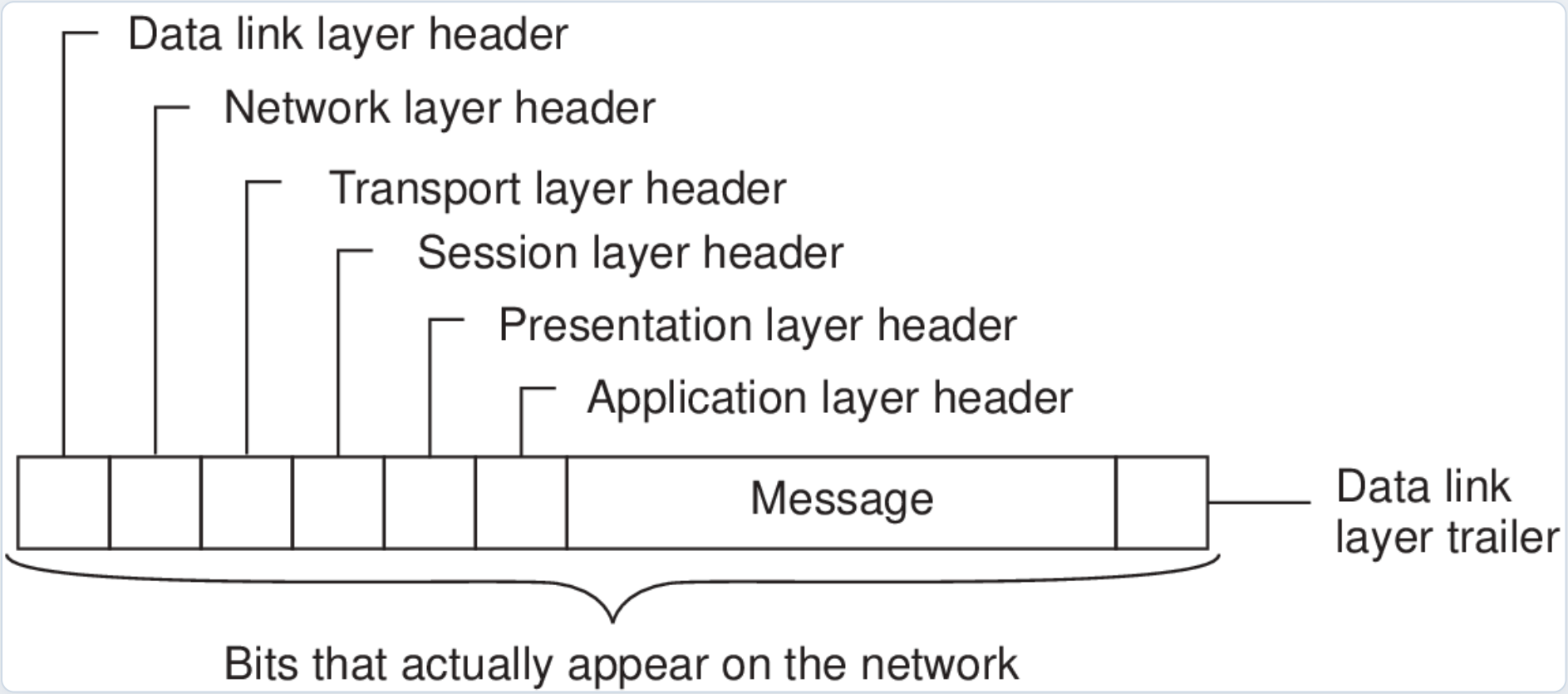
Protocolos em Camadas

- Não há memória compartilhada entre processos em máquinas diferentes.
- A comunicação acontece por troca de mensagens.
- Formato e significado das mensagens precisam estar combinados entre os dois processos — isso é um protocolo.
- Um protocolo de comunicação define um serviço, e pode ser orientado à conexão ou sem conexão.

Protocolos em Camadas — Modelo OSI



Protocolos em Camadas — Mensagens e Cabeçalhos



Protocolos em Camadas — Protocolos

Rede

IP — Internet Protocol

Transporte

TCP, UDP, RTP

Aplicação

HTTP, FTP, SMTP

Protocolos de Comunicação

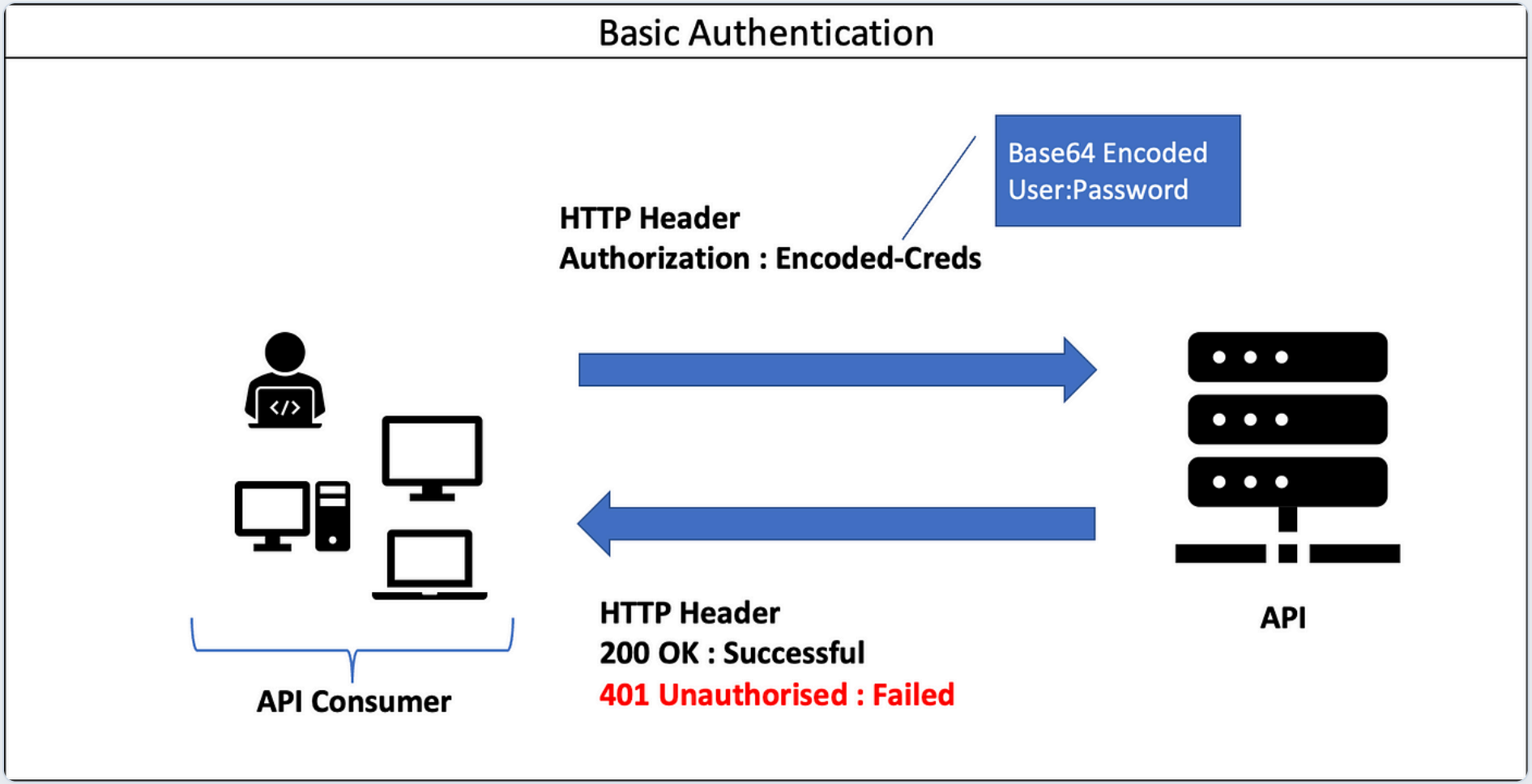
TCP | UDP | HTTP 1.1 | HTTP 2 | WebSockets | REST | GraphQL

gRPC | Web Service | QUIC / HTTP 3 | VoIP

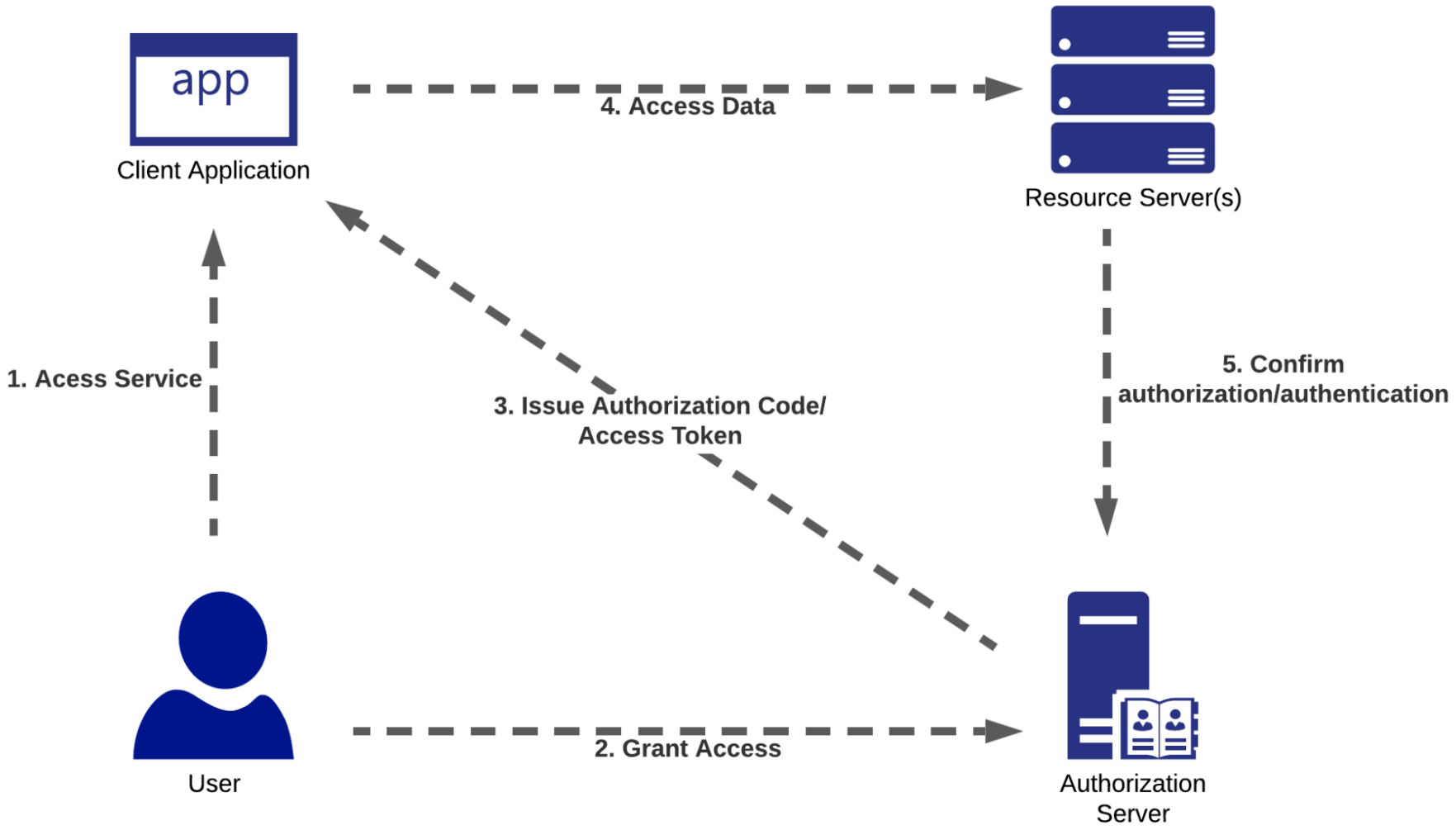
Protocolos em Camadas — Protocolos de Middleware

- Middleware é uma aplicação que busca simplificar a complexidade da comunicação usando o modelo OSI.
- Exemplos de protocolos utilizados por middleware:
  - DNS pode ser visto como uma aplicação de propósito geral que pode ser parte de um middleware.
  - Protocolos de autenticação e autorização — ex.: OAuth.
  - Protocolos de transação e travamento (lock).

Autenticação Básica (Basic Auth)



OAuth — Autorização



PARTE 2

Tipos de Comunicação

Persistência e sincronicidade das mensagens.

Middleware como serviço entre cliente e servidor

Um middleware pode oferecer vários tipos de serviço às aplicações — considere-o como um serviço adicional entre o cliente e o servidor, a nível de aplicação.

QUANTO À PERSISTÊNCIA

Comunicação Persistente

- A mensagem é armazenada no middleware de comunicação pelo tempo necessário até ser entregue ao destinatário.
- O remetente não precisa aguardar nem o destinatário estar em execução.
- Ex.: e-mail, WhatsApp.

QUANTO À PERSISTÊNCIA

Comunicação Transiente

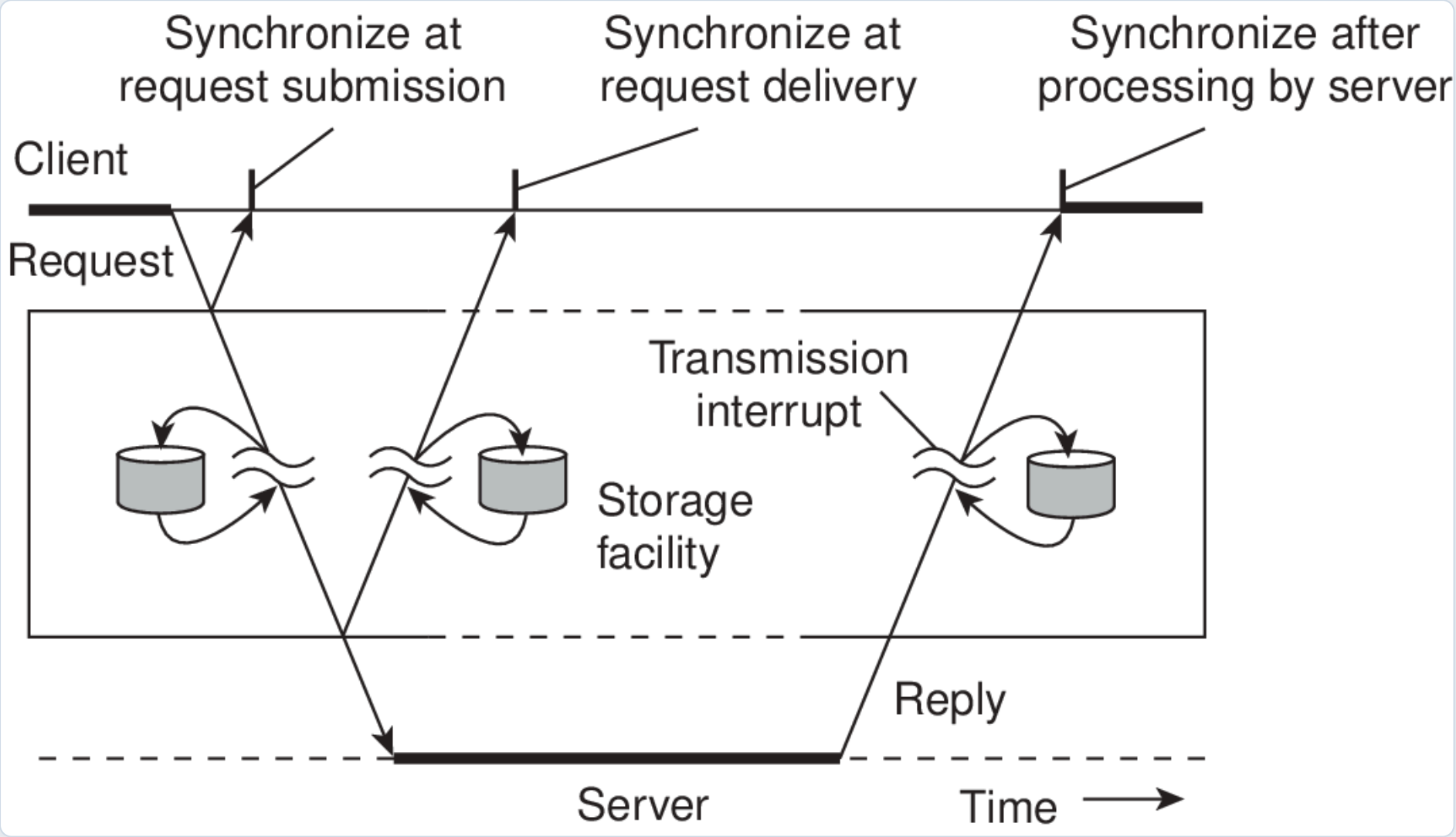
- A mensagem só é armazenada enquanto ambas as aplicações (remetente e destinatária) estiverem em execução.
- Se houver problema na transmissão, a mensagem é descartada — não fica guardada no middleware.
- Ex.: todos os protocolos de transporte são transientes.

QUANTO À ASSINCRONICIDADE

Comunicação Assíncrona

O remetente continua sua execução logo após o envio da mensagem — sem esperar resposta.

Comunicação Síncrona



Combinando persistência e sincronicidade

Sistemas de fila de mensagens

Comunicação persistente, com sincronização no envio.

Chamadas de procedimento remotas

Comunicação transiente, com sincronização após o processamento.

PARTE 3

REST

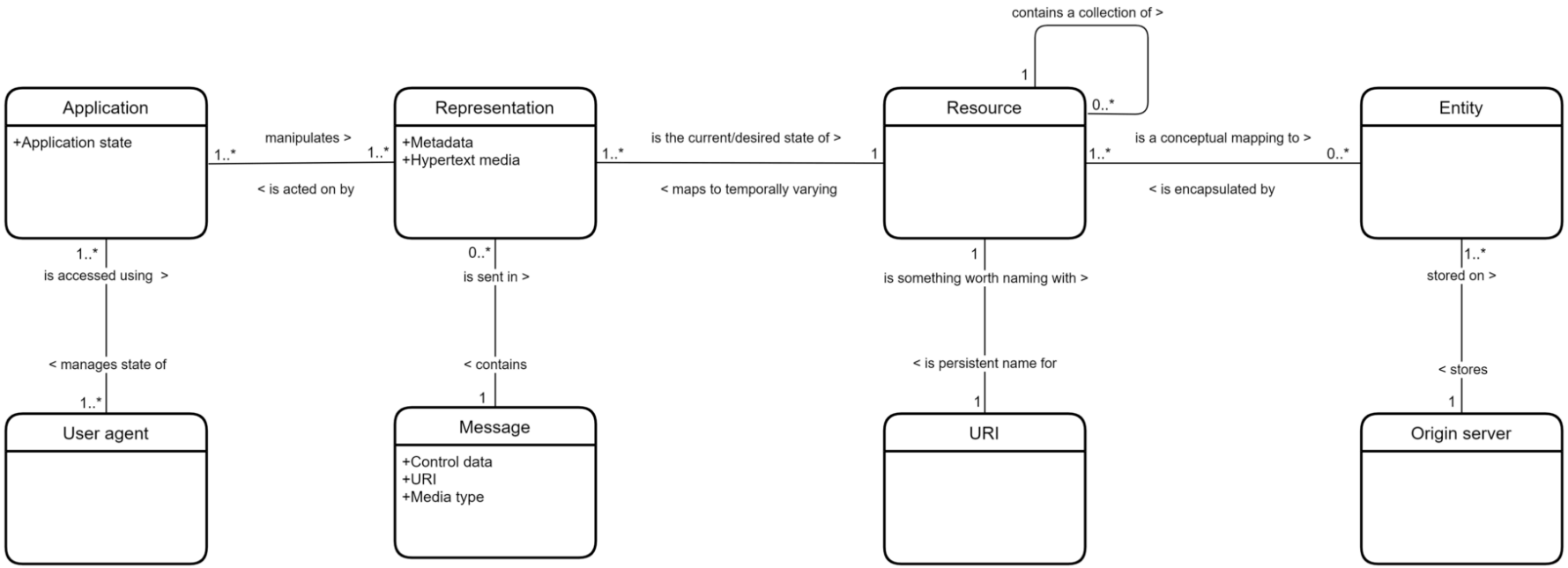
A arquitetura por trás da maioria das APIs web.

REpresentational State Transfer — REST

- “Transferência de Estado Representacional”.
- Proposto na tese de doutorado de Roy Fielding, em 2000.
- Uma nova arquitetura para sistemas na Web.

Roy Fielding

Os conceitos do REST



REpresentational State Transfer — REST

- Propriedades:
  - Sem estado (Stateless).
  - Cacheável.
  - Arquitetura Cliente-Servidor em camadas.
  - Interface Uniforme.

ATENÇÃO

REST ≠ HTTP — REST é um estilo arquitetural; HTTP é apenas o protocolo mais usado para implementá-lo.

Serviços Web RESTful

- Usam todos os métodos HTTP: `GET`, `POST`, `PUT`, `PATCH`, `DELETE`, `OPTIONS`, `HEAD`.
- Gerenciam recursos/entidades.
- Representação padrão: JSON.

REST NA PRÁTICA — E-COMMERCE

Listar e obter detalhes de um recurso

Recurso: Produto — parte do CRUD (Create, Retrieve, Update, Delete).

```
GET /:nome_recurso_plural
GET /products

GET /produtos/:id
GET /produtos/2342
GET /produtos/ram-ddr4-16-adata-23
```

O identificador pode ser um `id` numérico ou um slug — ambos identificam o recurso de forma única.

REST NA PRÁTICA — E-COMMERCE

Criar um novo recurso

```
POST /:nome_recurso_plural
POST /products
```

Os dados do novo recurso vão em JSON no corpo da requisição.

REST NA PRÁTICA — E-COMMERCE

Atualizar todos os dados de um recurso

```
PUT /:nome_recurso_plural/:id_recurso
PUT /products/:id
```

Os dados completos do recurso vão em JSON no corpo — `PUT` substitui o recurso por inteiro.

REST NA PRÁTICA — E-COMMERCE

Atualizar parcialmente um recurso

```
PATCH /:nome_recurso_plural/:id_recurso
PATCH /products/:id
```

Só os campos enviados em JSON são alterados — os demais permanecem como estavam.

REST NA PRÁTICA — E-COMMERCE

Deletar um recurso e outros métodos

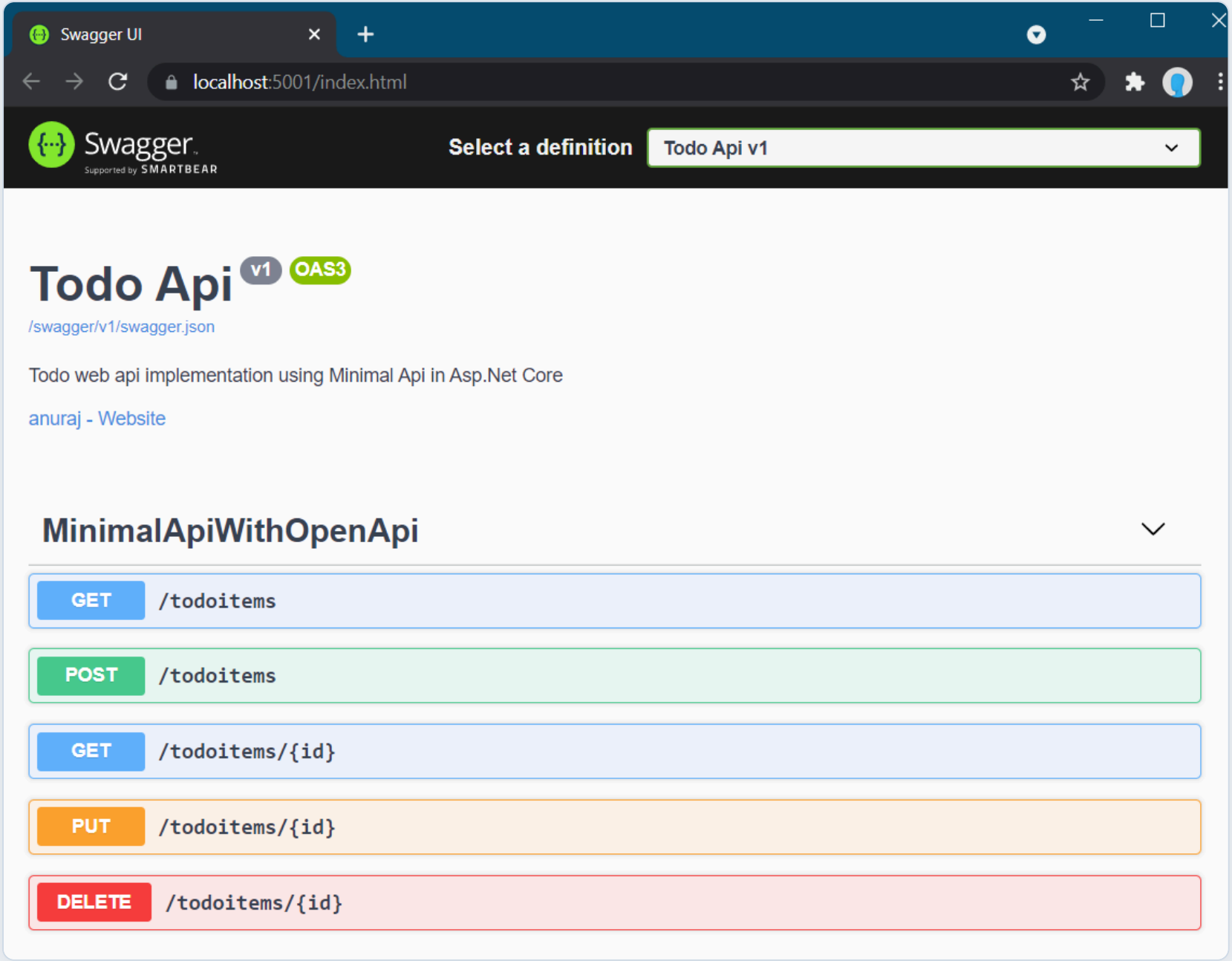
```
1 DELETE /:nome_recurso_plural/:id_recurso
2 DELETE /products/:id
```

- `OPTIONS` — verifica quais métodos são permitidos para uma entidade.
- `HEAD` — obtém metadados de um recurso (campos, tamanho, última versão...) sem baixar o corpo.

Códigos HTTP

Código	Significado
200	Requisição processada com sucesso.
201	Recurso criado com sucesso.
400	Bad Request — dados inválidos enviados.
401	Não autorizado.
404	Recurso não encontrado.
500	Erro geral no servidor.

REST Design / Build / Generate



FERRAMENTAS

Clientes REST

Insomnia

Postman

Bruno

EXEMPLO PRÁTICO

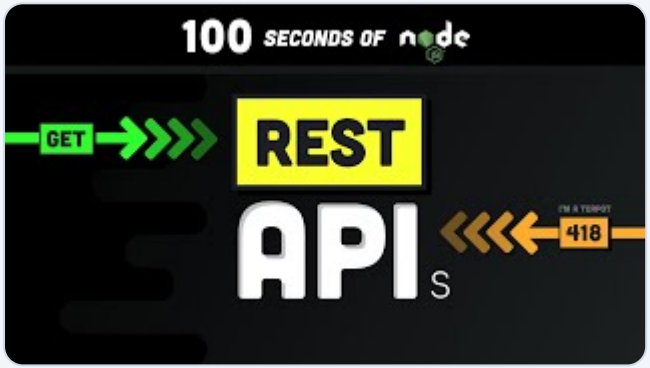
API REST com Django e Django Ninja

Exemplo completo disponível no GitHub da disciplina.

MATERIAL EM VÍDEO

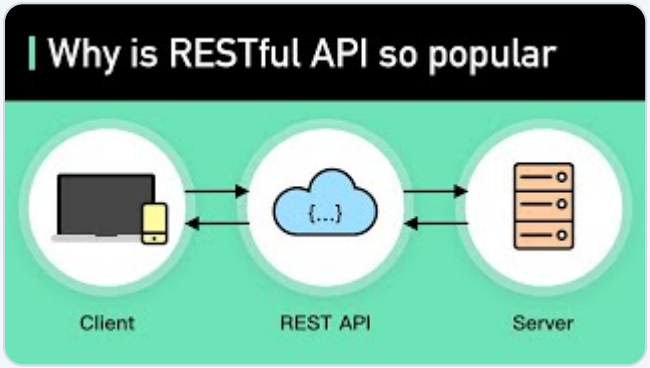
YOUTUBE **REST — Dicionário do Programador** youtube.com/watch?v=S7MduKwvVGk

MATERIAL EM VÍDEO



YOUTUBE **100 Seconds of Node — REST APIs** youtube.com/watch?v=-MTSQjw5DrM

MATERIAL EM VÍDEO



YOUTUBE **Why is RESTful API so popular**

youtube.com/watch?v=-mN3VyJuCjM

MATERIAL EM VÍDEO

PLAYLIST **REST + FastAPI — playlist completa** youtube.com/watch?v=QShMRcicxnE

Playlist bem completa sobre REST e como criar uma API com FastAPI. Para uma visão mais geral, vale começar pelo vídeo 3.

Referências bibliográficas

- TANENBAUM, A. S.; VAN STEEN, M. Distributed Systems. 3. ed. 2018. Cap. 4. ISBN 978-90-815406-2-9. distributed-systems.net