Módulo 2: REST APIs

Djonathan Barros

API (Application Programming Interface)

API é um conjunto de definições e protocolos usado no desenvolvimento e na integração de software de aplicações. API é um acrônimo em inglês que significa interface de programação de aplicações.

[REDHAT, 2021].


EXTRA BILLE A
70,000,000
T H I N G

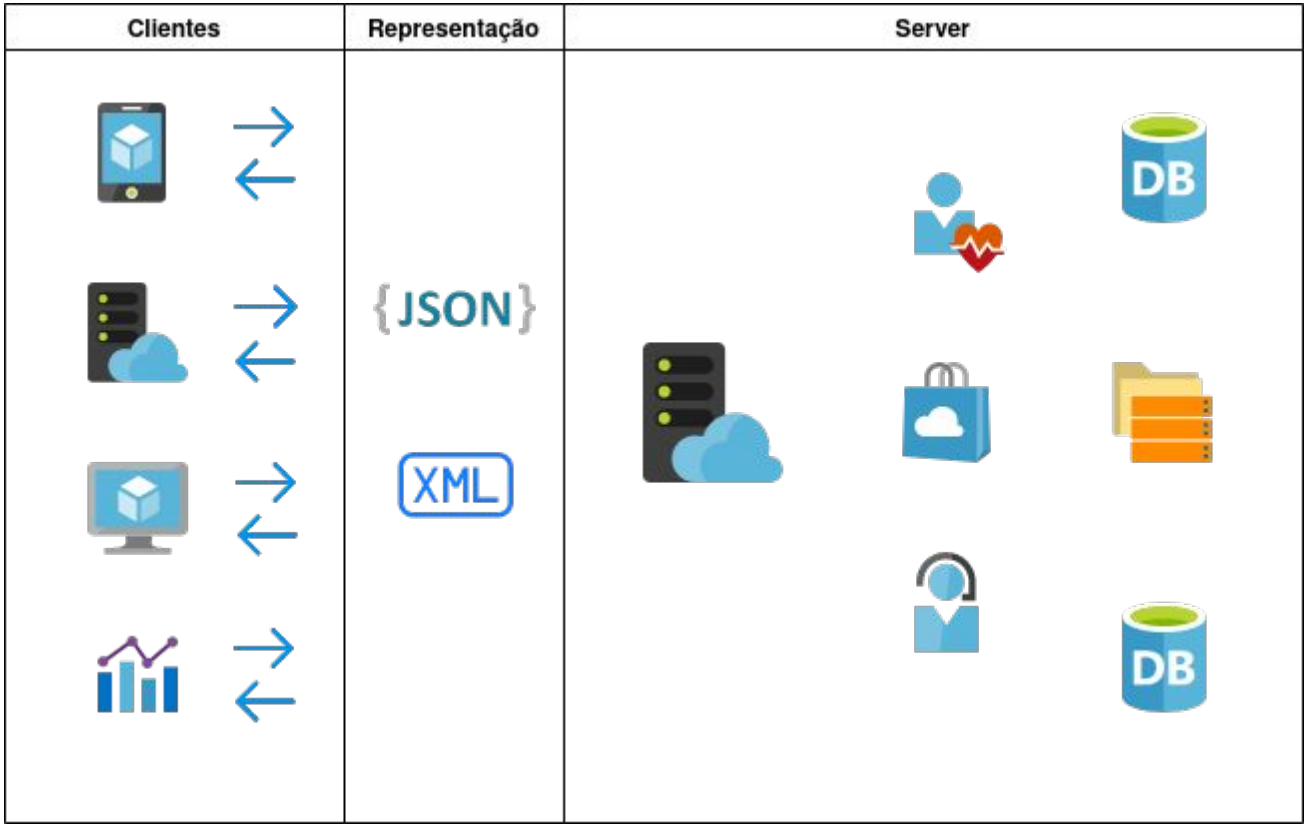
REST (Representional State Transfer)

Definição

O estilo de Transferência de Estado Representacional (REST) é uma abstração dos elementos arquiteturais de um sistema de distribuição de hipermídia.

[FIELDING, 2000] *Tradução nossa.*

Cliente x Servidor



Elementos de Dados

Os **elementos de dados** são transportados entre as **camadas** (DNS, Cache, Armazenamento) da aplicação através de componentes e conectores bem definidos e **independentemente da infraestrutura**.

Estruturas como URIs (Coleções vs Recursos vs Hierarquias)

/planos
/planos/1
/planos/1/dependentes/
/planos/1/dependentes/4

Obs: sempre utilizar substantivos e nunca verbos

/planos/cancelar **X**

Padrões para URIs

- Mantenha tudo em caixa baixa
- Substitua espaços com hífens ou underscore

Representação

- Deve utilizar uma *Linguagem Publicada*
  - https://www.json.org/json-en.html
  - https://www.w3.org/XML/
  - png / pdf / txt
- Deve ser consistentes (a mesma requisição deve retornar a mesma resposta caso o recurso não tenha sido modificado)
- O conteúdo pode ser negociado

Métodos HTTP Explícitos

Verbo	Função
[GET] * **	Obter a representação de um recurso
[POST]	Criar um novo recurso em uma coleção
[PUT] **	Tanto inserir quanto alterar um recurso identificável armazenado
[PATCH] **	Atualização parcial de um recurso
[DELETE] **	Remover um recurso
[HEAD][OPTIONS] * **	*Ferramentas de servidor*

* **seguro:** não causa efeitos
** **idempotente:** não causa efeitos colaterais

Códigos de Resposta

3xx e 4xx: há uma série de erros definidos pelo protocolo HTTP que podem ser utilizados para casos específicos (ex. segurança)

Verbo	Função
200 (OK)	Indica sucesso não específico
201 (Created)	Indica o sucesso em uma criação
202 (Aceito)	Indica o início de uma ação assíncrona
204 (No Content)	Indica sucesso sem um corpo de resposta
400 (Bad request)	Parâmetros inválidos informados para o processamento da requisição
404 (Not found)	Recurso não encontrado
500 (Internal server err)	Erro genérico no servidor durante o processamento

Comunicação Sem Estado (Stateless)

- Requisito para escalabilidade
- As requisições devem ter a informação necessária para o processamento
- As requisições devem ser independentes, não necessitando de estado de sessão armazenado no servidor
- Várias instâncias do serviço podem processar as requisições em diferentes momentos

Headers

Diferentes cabeçalhos HTTP podem ser utilizados como metadados de comunicação:

- *Content-type / Accept:* negociação de conteúdo (ex. *application/json* ou *application/xml*)
- *Cache-control:* controle de cache (ex. *no-cache* ou *max-age[=segundos]*)
- *Last-modified:* retornar nova representação se modificada depois da data fornecida (*ETag* pode ser utilizado para identificar versões específicas de um recurso)

Restful

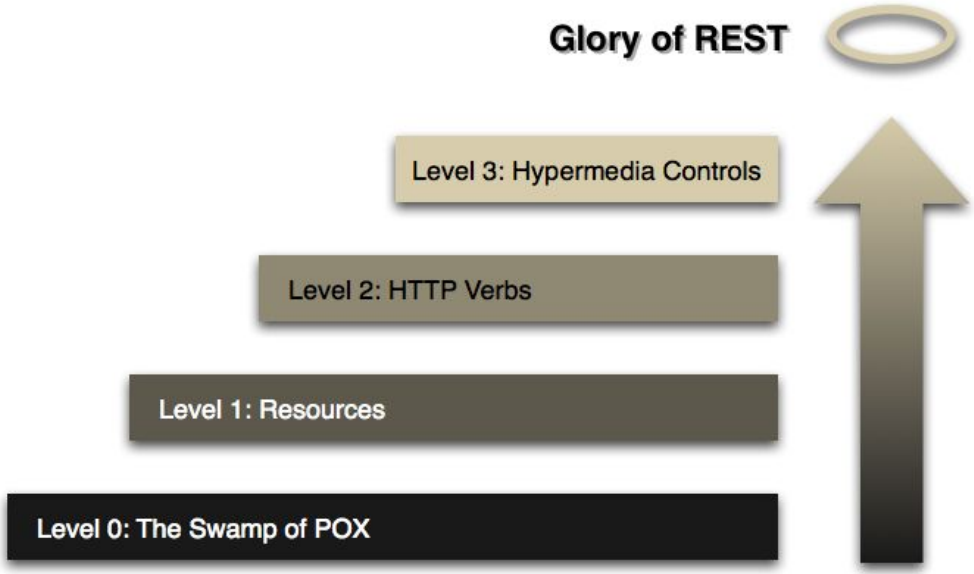
Definição

- Uma API Restful é completamente aderente as restrições do Estilo Arquitetural REST
  - Interface Uniforme
    - (Resource Based, Manipulação através de representação, Mensagens auto descritivas, **HATEOAS**)
  - Stateless
  - Cacheavel
  - Client-Server
  - Sistema em camadas

https://blog.geekhunter.com.br/sua-api-nao-e-restful-entenda-por-que/

Modelo de Maturidade de Richardson

Modelo desenvolvido por Leonard Richardson para medir a aderência de um serviço web a partir de um conjunto de características “REST”.



Modelo de maturidade de Richardson [FOWLER, 2010]

Nível 0 (XML-RPC)

O serviço é mais próximo a um protocolo de Chamadas de Procedimentos Remotos, dirigido por XML, tanto Schema quanto ação são descritos diretamente no corpo da requisição.

```
POST /simulacaoService HTTP/1.1


<dadosSimulacaoRequest tipo="IPCA" aliquotaIPCA=8.44/>
```

Nível 1 (Recursos)

Alternativamente a comunicação identificada por serviços, nesse nível a URI descreve quais os recursos que serão operados.

```
POST /simulacao HTTP/1.1


<dadosSimulacaoRequest tipo="IPCA" aliquotaIPCA=8.44/>
```

Nível 2 (Verbos Http)

A maioria ou todos os verbos Http são utilizados, os códigos de resposta correto também são utilizados.

```
POST /simulacao HTTP/1.1

<dadosSimulacaoRequest tipo="IPCA" aliquotaIPCA=8.44/>

201 CREATED /simulacao/15
```

```
GET /simulacao/15 HTTP/1.1



200 OK <respostaSimulacao .../>
```

Nível 3 (Controles de Hipermídia)

O relacionamento e a navegação entre os recursos é dirigida por hipermídia e sensível ao estado dos recursos (HATEOAS).

```
GET /pedido/15

200 OK

<pedido>

     <id>15</id>

     <link rel="confirmar" uri="/pedido/15/confirmacao">15</link>

<pedido/>
```

```
GET /pedido/15

200 OK

<pedido>

     <id>15</id>

     <link rel="cancelar" uri="/pedido/15/cancelamento">15</link>

<pedido/>
```

Q&A

Bibliografia

FIELDING, Roy Thomas. **Architectural styles and the design of network-based software architectures.** University of California, Irvine, 2000.

FOWLER, Martin. **Richardson Maturity Model: steps toward the glory of REST.** Online at http://martinfowler.com/articles/richardsonMaturityModel. html, p. 24-65, 2010.

KOSCHEL, Arne et al. **Evaluating the RESTfulness of “APIs from the Rough”**. 2019.

REDHAT. **O que é API?.** Red Hat Forum, 2021. https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces.

RODRIGUEZ, Alex. **Restful web services: The basics.** IBM developerWorks, v. 33, p. 18, 2008.