PUC Minas

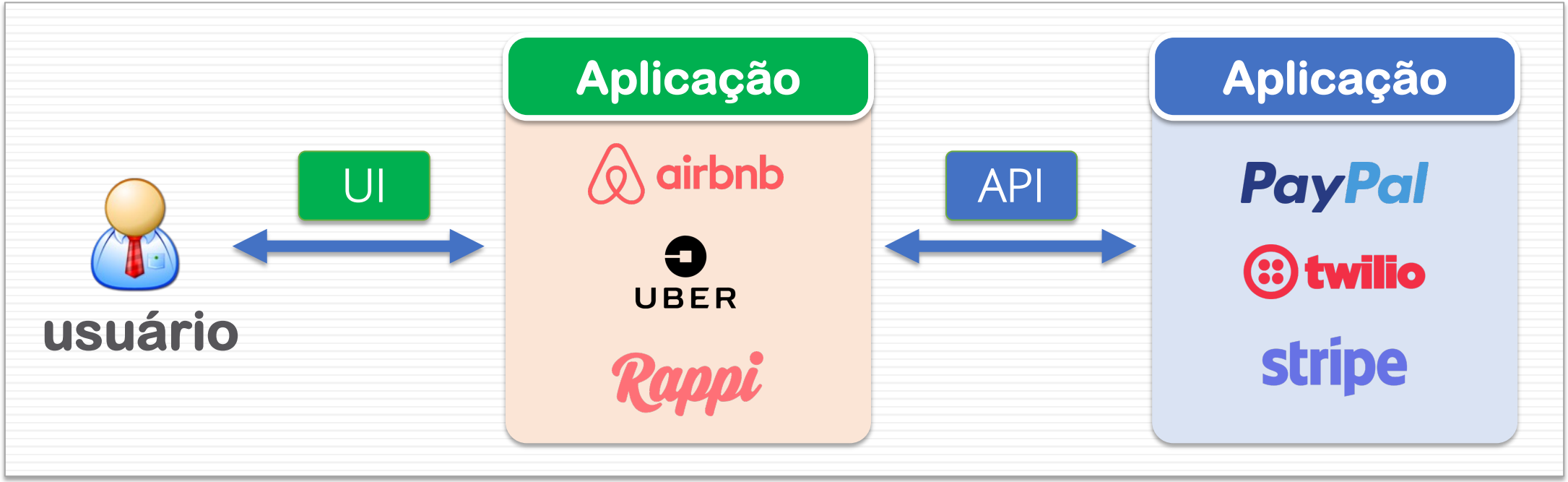
Plataforma Node.js

Fundamentos de APIs e RESTful APIs

Prof. Rommel Vieira Carneiro

Introdução ao mundo das APIs

Application Programming Interface (API) é uma interface de uma aplicação que é voltada para outras aplicações, proporcionando integração entre sistemas.



Tipos de APIs

Web APIs ou Web Services

- SOAP
- REST e RESTful
- GraphQL
- WebSockets
- WebHooks

APIs Legadas

- Remote Procedure Call (RPC)
- Java Remote Method Invocation (RMI)
- CORBA
- DCOM

Fontes:

- API Is Not Just REST - https://apievangelist.com/2018/02/03/api-is-not-just-rest/
- 4 estilos de APIs mais usados e comentados do mercado - https://www.ca.com/pt/blog-latam/estilos-de-apis.html
- A guide to REST and API Design - http://docs.huihoo.com/api/A-Guide-to-REST-and-API-Design.pdf
- Undisturbed REST – A guite to designing the perfect API - https://www.mulesoft.com/lp/ebook/api/restbook
- Web API Design – apigee - https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf
- Pragmatic REST: APIs without hypermedia and HATEOAS - http://www.ben-morris.com/pragmatic-rest-apis-without-hypermedia-and-hateoas/
- API Design: Harnessing Hypermedia Types - https://apigee.com/about/blog/api-technology/api-design-harnessing-hypermedia-types

RESTful API – Introdução

REST é um estilo arquitetural para construção de serviços Web que significa a Transferência de Estado Representacional (Representational State Transfer).

O termo **RESTful** caracteriza serviços Web que seguem integralmente as recomendações REST, diferentemente daqueles (**RESTlike**) que implementam parcialmente suas recomendações.

RESTful API – Princípios de design

Princípios importantes de design de uma API RESTful:

- Definições do protocolo HTTP
- Estrutura uniforme de Endpoint
- Abordagem stateless (sem estado)
- Abordagem HATEOAS
- Controle do versionamento da API
- Controle adequado do cache
- Documentação clara e adequada da API

Fontes:

- Best Practices for Designing a Pragmatic RESTful API - https://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api
- RESTful Web Services: The basics, Alex Rodriguez - http://www.ibm.com/developerworks/webservices/library/ws-restful/
- RESTful API Designing guidelines—The best practices - https://hackernoon.com/restful-api-designing-guidelines-the-best-practices-60e1d954e7c9
- RESTful API Design. Best Practices in a Nutshell - https://blog.philipphauer.de/restful-api-design-best-practices/
- API design - https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

RESTful API – Princípios de design

Definições do protocolo HTTP – Métodos HTTP

Utilizar os métodos HTTP de maneira clara em função do seu propósito semântico:

GET	Recuperar um recurso (SELECT)
POST	Criar um recurso no servidor (INSERT)
PUT | **PATCH**	Alterar um estado de um recurso ou atualizá-lo (UPDATE)
DELETE	Remover um recurso (DELETE)
OPTIONS	Lista as operações de um recurso

IMPORTANTE: Métodos GET e parâmetros de query não devem alterar estado de recursos.

RESTful API – Princípios de design

Definições do protocolo HTTP – Métodos HTTP

Exemplo 1

Antes – Criação com GET

```
GET /adduser?name=Robert HTTP/1.1
```

Depois – Criação com POST

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>
   <name>Robert</name>
</user>
```

1) Evite utilizar verbos (adduser) na URL, os métodos devem ser suficientes para descrever a operação
2) Utilize o método correto para a operação (Criar → POST)

RESTful API – Princípios de design

Definições do protocolo HTTP – Métodos HTTP

Não RESTful		
Verbo	HREF	Ação
POST	/bookmarks/create	Criar (Create)
GET	/bookmarks/show/1	Visualizar (Read)
POST	/bookmarks/update/1	Atualizar (Update)
POST/GET	/bookmarks/delete/1	Apagar (Delete)
RESTful		
Verbo	URI	Ação
POST	/bookmarks	Criar (Create)
GET	/bookmarks/1	Visualizar (Read)
PUT	/bookmarks/1	Atualizar (Update)
DELETE	/bookmarks/1	Apagar (Delete)

RESTful API – Princípios de design

Definições do protocolo HTTP – Códigos de status da resposta HTTP

Code	Propósito	Descrição	Exemplo
1xx	Informacional	Requisição recebida,	Processo em continuidade
2xx	Sucesso	200 – Sucesso na requisição	Requisição de informações (GET) com sucesso
		201 – Recurso Criado	Inclusão de recurso (POST) com sucesso
		202 – Requisição aceita para processamento	Processo assíncrono sem retorno imediato
		204 – Requisição processada com sucesso	Exclusão de recurso (DELETE) com sucesso
3xx	Redirecionamento	301 – Movido permanentemente	Site transferido de servidor
		302 – Movido temporariamente	Recurso temporário no lugar
4xx	Erro no cliente	400 – Requisição incorreta	Problemas de sintaxe, rotas inexistentes
		401 – Autenticação Requerida	Primeira requisição sem dados de autenticação
		403 – Acesso negado	Recurso privado não acessível pelo requisitante
		404 – Recurso não encontrado	Recurso solicitado não existe
		405 – Método não permitido	PUT ou DELETE em endpoints de GET ou POST
5xx	Erro no servidor	O servidor falhou em completar um pedido aparentemente válido	

RESTful API – Princípios de design

Definições do protocolo HTTP – Uso de MIME Types no Content Type

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Requisição

```
GET /cliente/2 HTTP/1.1
Host: http://servidor
Accept: application/json
```

Resposta

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 109
{
    id: 2
    name: 'Rommel Vieira Carneiro',
    email: 'rommel@email.com',
    alcohol:  '5.21'
}
```

RESTful API – Princípios de design

Estrutura uniforme de Endpoint

- Uma URI intuitiva direta, auto-documentada, e compreensível
- Ter uma estrutura hierárquica semelhares a diretórios

Exemplo

```
http://myservice.org/discussion/{year}/{day}/{month}/{topic}
http://myservice.org/discussion/2008/12/10/{topic}
```

Orientações adicionais

- Ocultar a extensão da tecnologia empregada (.asp, .jsp, .php, etc)
- Manter tudo em caixa baixa (minúsculo)
- Evitar espaços nos caminhos da URI
- Evite parâmetros de QueryString, o máximo possível

RESTful API – Princípios de design

Estrutura uniforme de Endpoint – CRUD de Recursos

Ação	Operação	Mapeamento da URL
Incluir um curso	**C**REATE	**POST** /cursos/
Obter lista de cursos	**R**ETRIEVE	**GET** /cursos
Obter um curso específico	**R**ETRIEVE	**GET** /cursos/:id
Pesquisar um curso	**R**ETRIEVE	**GET** /cursos?search=param
Alterar um curso	**U**PDATE	**PUT** /cursos/:id **PATCH** /cursos/:id
Excluir um curso	**D**ELETE	**DELETE** /cursos/:id

RESTful API – Princípios de design

Estrutura uniforme de Endpoint – Relacionamentos

Ação	Mapeamento da URL
Listar alunos do curso	**GET** /cursos/:id/alunos ou **GET** /alunos/?curso_id=:id
Obter um aluno do curso	**GET** /cursos/:id/alunos/:id
Incluir aluno no curso	**POST** /cursos/:id/alunos
Remover aluno do curso	**DELETE** /cursos/:id/alunos/:id
Listar cursos do aluno	**GET** /alunos/:id/cursos ou **GET** /cursos/?aluno_id=:id
Obter um curso do aluno	**GET** /alunos/:id/cursos/:id

RESTful API – Princípios de design

Estrutura uniforme de Endpoint

Exemplo: The Movie DB - Endpoint: http://api.themoviedb.org/3

Ação	Mapeamento da URL
Busca por empresas	**GET** /search/company?query=xxx&page=x
Busca por filmes	**GET** /search/movie?query=xxx&page=x **GET** /search/movie?year=XXXX&language=xx
Dados de filme específico	**GET** /movie/{movie_id}
Artistas e equipe técnica do filme	**GET** /movie/{movie_id}/credits
Dados de uma empresa específica	**GET** /company/{company_id}
Dados de uma pessoa específica	**GET** /person/{person_id}

RESTful API – Princípios de design

Abordagem stateless (sem estado)

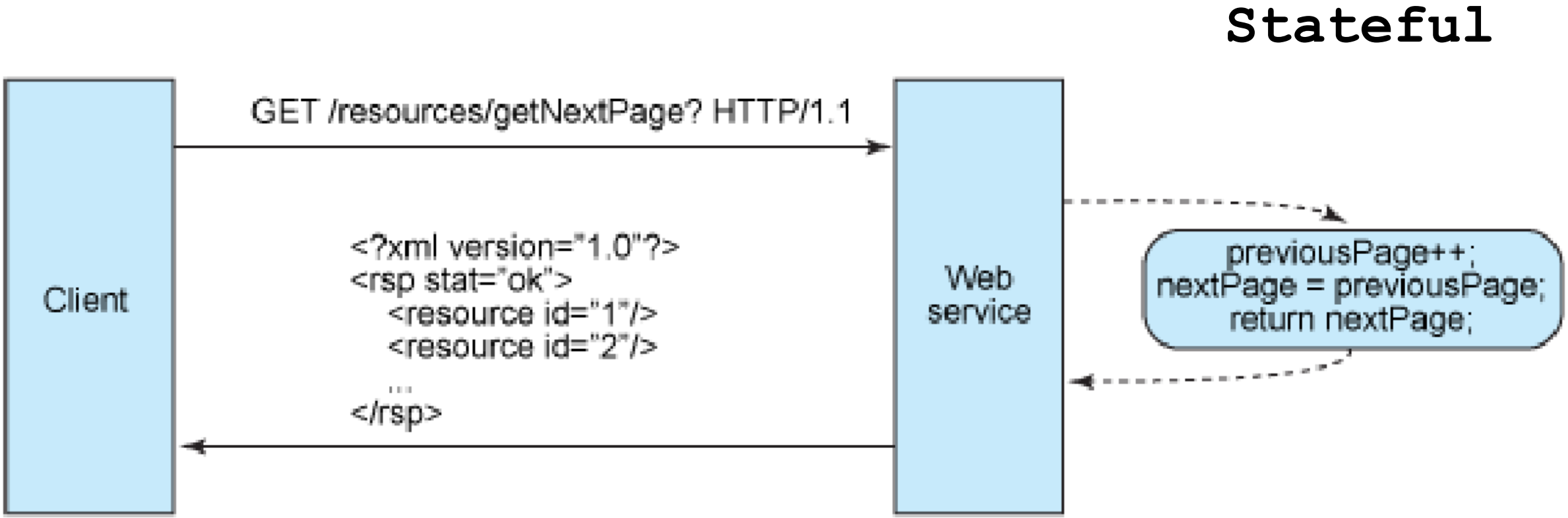
- O servidor não mantem estado sobre a sessão do usuário/aplicação
- Toda requisição deve conter todas as informações requeridas (como parte da URI, na query string, no corpo ou no cabeçalho)

Benefícios

- Simplifica o servidor
- Maior escalabilidade uma vez que o servidor não mantém informações sobre sessão
- Servidores de balanceamento de carga não precisam se preocupar com dados de sessão
- Maior confiabilidade (recuperação de falhas)

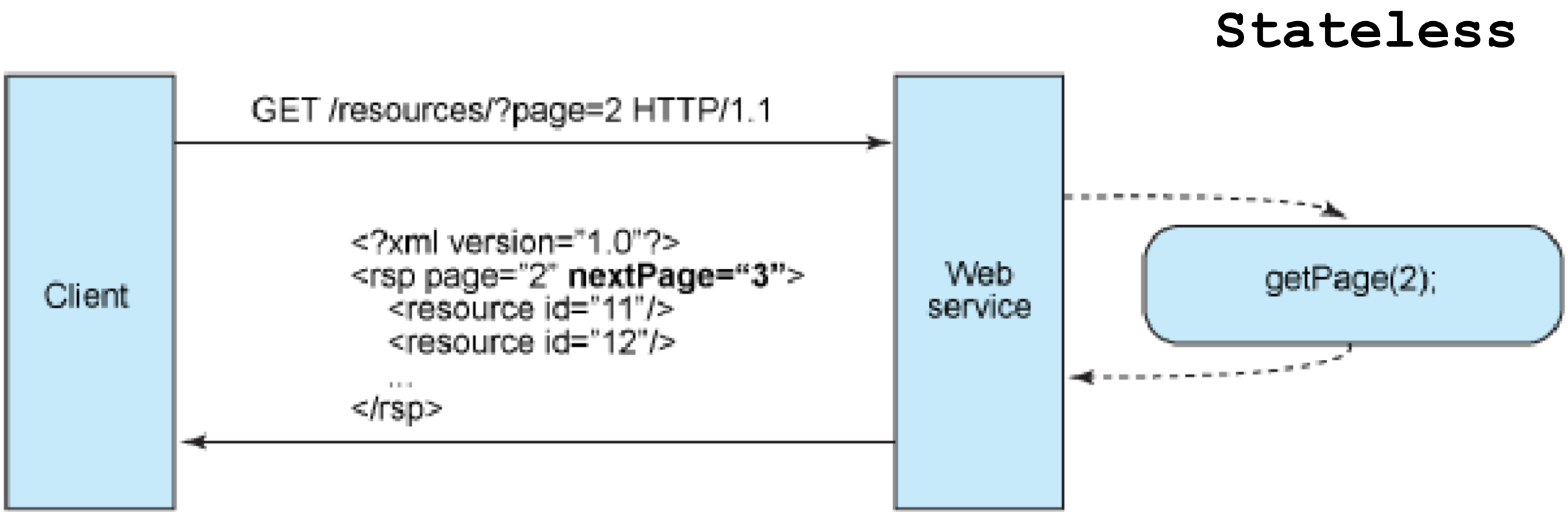
RESTful API – Princípios de design

Abordagem stateless (sem estado)



RESTful API – Princípios de design

Abordagem stateless (sem estado)



RESTful API – Princípios de design

Abordagem HATEOAS - Hipertexto como mecanismo de estado da aplicação

Resposta p/ saldo de 100,00

```
GET /account/12345 HTTP/1.1

------

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```


Fonte: http://en.wikipedia.org/wiki/HATEOAS

RESTful API – Princípios de design

Abordagem HATEOAS - Hipertexto como mecanismo de estado da aplicação

```
GET /account/12345 HTTP/1.1

------

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Resposta p/ saldo de -25,00

Fonte: http://en.wikipedia.org/wiki/HATEOAS

RESTful API – Princípios de design

Versionamento da API

- Incorporar a versão do serviço na URI
  http://myservice.org/v1/discussion/{year}/{day}/{month}/{topic}

- Incluir informações da versão na representação de estado do recurso

```
<beer version="1.0">
    <name>Asahi Draft Beer</name>
    <brewer>
        <name>Asahi</name>
        <country>Japan</country>
    </brewer>
    <calories>41</calories>
    <alcohol>5.21</alcohol>
</beer>
```

RESTful API – Princípios de design

Controle adequado do cache – Cabeçalhos HTTP

Cabeçalho	Aplicação
Date	Data e hora de geração da representação de estado.
Last Modified	Data e hora de última alteração da representação do estado.
Cache-Control	O cabeçalho utilizado pelo HTTP 1.1 para controle de cache.
Expires	Data e hora de expiração a representação do estado. Para compatibilizar com clientes HTTP 1.0.
Age	Tempo passado em segundos desde que o resultado foi extraído do servidor. Pode ser inserido por um componente intermediário.

RESTful API – Princípios de design

Controle adequado do cache – Valores para o Cache-Control

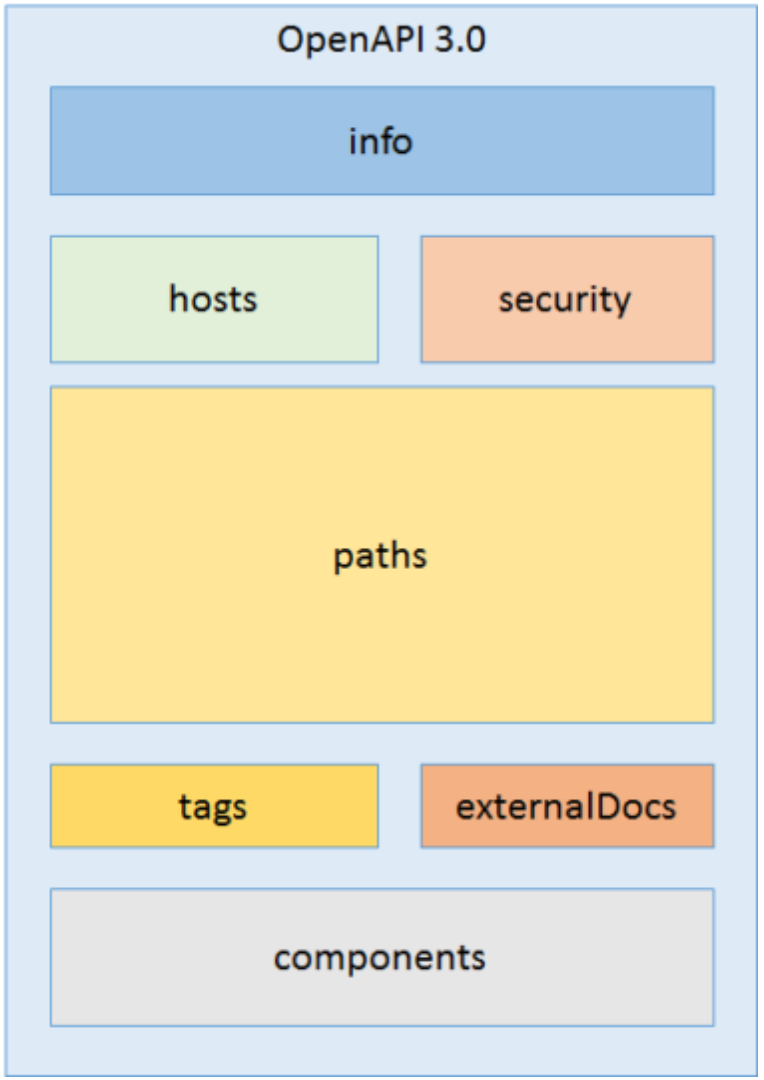
Directive	Application
Public	The default. Indicates any component can cache this representation.
Private	Intermediary components cannot cache this representation, only client or server can do so.
no-cache/no-store	Caching turned off.
max-age	Duration in seconds after the date-time marked in the Date header for which this representation is valid.
s-maxage	Similar to max-age but only meant for the intermediary caching.
must-revalidate	Indicates that the representation must be revalidated by the server ifmax-age has passed.
proxy-validate	Similar to max-validate but only meant for the intermediary caching.

RESTful API – Princípios de design

Documentação clara e adequada da API

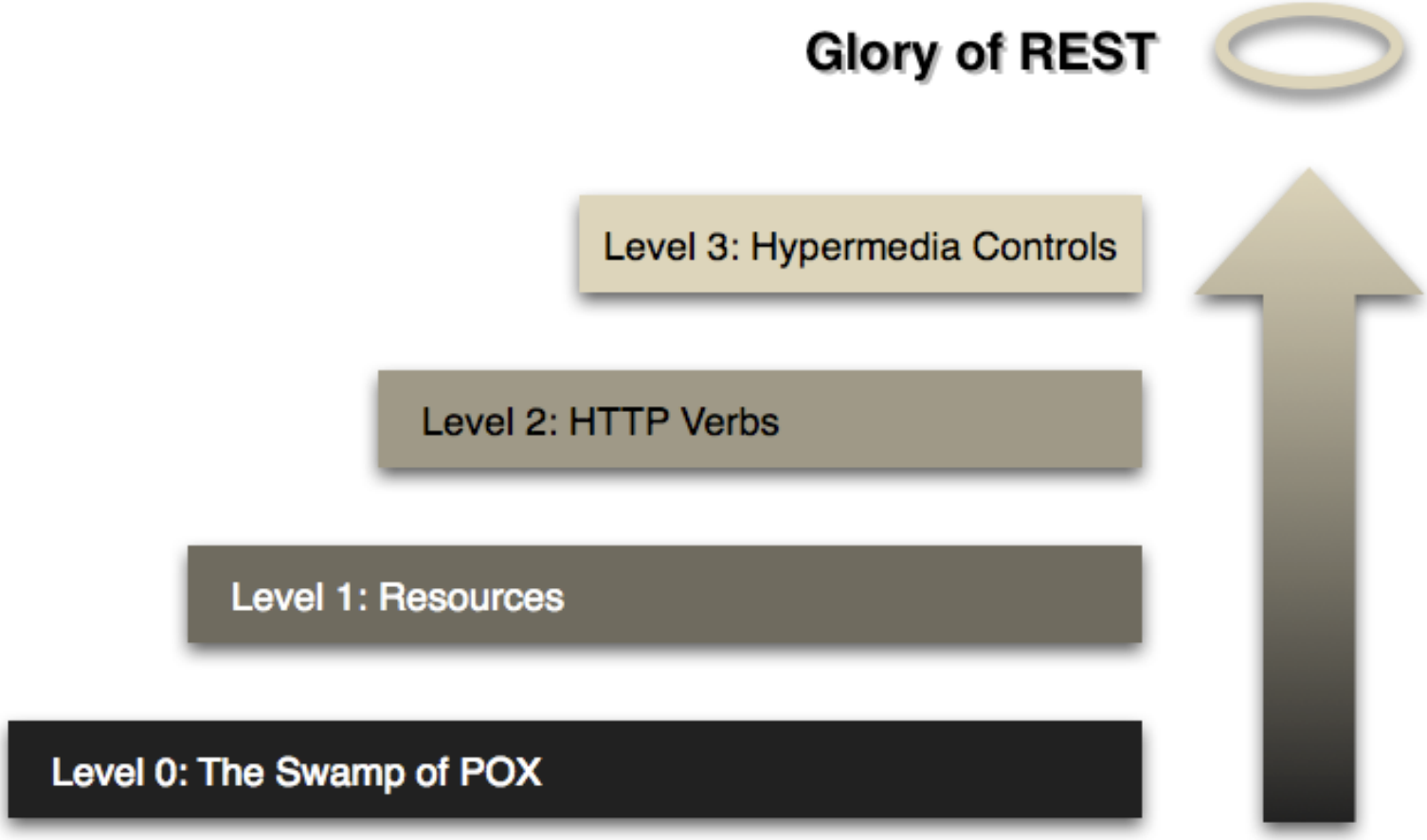
OpenAPI Specification

- Padrão para definição de APIs RESTful
- Independente de linguagem de programação
- Permite a geração automática de código



RESTful API – Princípios de design

Richardson Maturity Model



Fonte: Richardson Maturity Model