20 DE FEVEREIRO DE 2018

PADRONIZAÇÃO DE API

INTERFILE REST

RODRIGO CARLSTROM
INTERFILE BPO
www.interfilebpo.com.br

Sumário

Objetivo

Este documento tem como finalidade definir as regras para a padronização de APIs REST da Interfile BPO. Este documento se propõe responder as seguintes questões:

- Como definir as URIs dos serviços?
- Qual o status HTTP retornar?
- Quando utilizar parâmetro *Path Parameter* ou *Query Parameter*?
- Utilizar XML ou JSON?

O objetivo deste guia é tornar a API simples, intuitiva e prática para o seu utilizador. Assim a API deve ter:

- Documentação adequada;
- Ser mais intuitiva possível;
- Deve-se levar em conta a experiência do desenvolvedor que utilizará a API;
- Seguir tendência de mercado;
- Padronizar a API em sua URI, Verbos, Parâmetros, *Request* e *Response*.

Padrão HTTP

O padrão HTTP adotado será o REST, utilizando as características nativas do HTTP (como código de retorno no *status*, por exemplo) onde elas façam sentido.

Uso do JSON

O JSON é menos verboso que XML e mais fáceis de ler, diminui tráfego de rede e tem sido uma tendência de mercado.

Encoding

Será utilizado como *encoding* UTF-8.

URI

Versão

Deve-se incluir a versão da API no primeiro nível da API. Por exemplo:

/v1/costumers

Recurso (substantivo)

Serão usados os recursos semânticos, de forma a fazer sentido para o usuário. Deve-se utilizar um substantivo no segundo nível, em inglês e no plural. Exemplo:

/v1/issuers

/v1/products

/v1/costumers

Identificador

O terceiro nível deve ser o identificador do recurso, por exemplo:

/v1/costumers/23	Cliente número 23

Agregados

O quarto nível poderá ser uma informação agregada ao segundo nível (podendo ter o identificador do agregado no quinto nível):

/v1/costumers/23/adresses	Lista de endereços do cliente 23
/v1/costumers/23/addresses/1	Endereço 1 do cliente 23

Ações

O quarto nível poderá ser um verbo representando uma ação não cobertas pelos verbos HTTP (Vide Verbos HTTP):

/v1/costumers/23/activate	Ativar o cliente 23

Outros níveis

Os demais níveis seguem a regra anterior. As utilizações de níveis somente devem ser feitas sobre as informações agregadas a um recurso primário (coeso). Os recursos primários devem ficar sempre no primeiro nível para que se aumente a facilidade de uso para o usuário. Por exemplo:

/v1/issuers/56/costumers/23/accounts/53/cards/55	URI incorreda
/v1/cards/55	URI correta

Substantivo Composto

Utiliza o caracter "-" quando se tratar de substantivo composto:

/product-setup

Verbos HTTP

Verbo HTTP	Função	Exemplo	
GET	Ler o recurso	/v1/costumers	Lista de clientes.
		/v1/costumers/23	Recupera um cliente específico.
POST	Cria um novo recurso	/v1/costumers	Cria um novo cliente.
PUT	Atualiza todos os dados do recurso	/v1/costumers/23	Atualiza todos os dados de um cliente específico.
PATCH	Atualiza atributos específicos do recurso	/v1/costumers/23	Atualiza parcialmente um recurso específico.
DELETE	Remove o recurso	/v1/costumers	Remove todos os clientes
		/v1/costumers/23	Remove um cliente específico
OPTIONS	Permite verificar o conjunto de operações disponíveis para o usuário logado.	/v1/costumers	Operações disponíveis ao usuário logado, sobre a coleção de clientes.
		/v1/costumers/23	Operações disponíveis ao usuário logado, sobre um cliente específico.

Tratamento de erros

HTTP Status Code

Verbo HTTP	*Status Code*	*Descrição*
SUCCESS	*200, ok.*	*Sucesso. Utilizado na maioria dos casos, como um GET/PUTH/PATH.*
	201, Created.	*Sucesso. Utilizado quando o recurso é criado, com um POST.*
	204, No Content.	*Sucesso. Utilizado quando não há dados de retorno. Por exemplo um GET com parâmetros.*
	206, Partial Content.	*Sucesso. O recurso está incompleto, utilizando em casos de paginação.*
CLIENT ERROR	*400, Bad Request*	*Recurso foi acessado, porém aconteceu algum erro na validação de alguma regra de negócio da aplicação.*
	401, Unauthorized	*Erro de autenticação.*
	403, Forbidden	*Autenticação bem sucedida, mas sem privilégios suficientes.*
	404, Not Found	*O recurso solicitado não existe. Aplica-se também a um ID não existente, por exemplo. Para este caso é importante deixar explícito o motivo no body da mensagem de erro.*
	422, Unprocessable Entity	*A entidade não pode ser processada. Por exemplo, essa condição de erro pode ocorrer se um corpo de solicitação JSON bem-formadas (ou seja, sintaticamente corretas), mas semanticamente erradas.*
SERVER ERROR	*500, Internal Server Error*	*Algum erro inesperado na aplicação. Exemplo: Algum problema na infraestrutura.*

Return Code & Error Message

Além do HTTP Status Code, a API deve retornar código de retorno e mensagem. Esta tabela deve ser revisitada e atualizada a cada projeto.

Código	Mensagem	Recomendação
`incorrect_zip`	O código postal / postal está incorreto.	O cliente deve tentar novamente usando o código postal / postal correto.

Parâmetros

Tipo	Descrição	Exemplo	Quando utilizar
HTTP Parameters	*Diretamente na URI da chamada.*	*/v1/costomers/23*	*Apenas um parâmetro é obrigatório para o serviço.*
QUERY Parameters	*Diretamente na URI da chamada.*	*/v1/costomers?name=joao*	*Parâmetros opcionais ou lista de parâmetros.*
HEADER Parameters	*Passado no cabeçalho do HTTP*	*Cached-control:no-cached*	*Necessidade de fornecer metadados ao servidor e atributos como: tokens, credenciais para autenticação, etc.*
BODY Parameters	*Definido na construção da mensagem HTTP. Os parâmetros são enviados no corpo da mensagem em formato JSON.*	*{ “name”:”João”, “status”:1 }*	*Em funcionalidades que utilizem os verbos PUT, PATH e POST.*

Paginação

Deve ser utilizado *QUERY Parameter* para a paginação. Utilizar pelo menos um dos três parâmetros:

Tipo	Descrição
limit	*O número de objetos a serem retornados, entre 1 e 100.* GET /v1/clients?limit=100
starting_after	*O id do objeto que define a posição de início de sua lista. Por exemplo: Se em uma busca de clientes com limit de 100 e o último cliente tiver o id=111, na chamada subsequente pode ser utilizado o id do último cliente apresentando para resultar na página posterior:* GET /v1/clients?limit=100&starting_after=111
ending_before	*O id do objeto que define a posição de término de sua lista. Por exemplo: Se em uma busca de clientes com limit de 100 e o primeiro cliente tiver o id=112, chamada subsequente pode ser utilizado o id do primeiro cliente apresentando para resultar na página anterior:* GET /v1/clientes?limit=100&ending_before=112

Query Language

Nos *QUERY Parameters* poderão ser utilizadas *Query Language* baseada no padrão FIQL: The Feed Item Query Language.

A expressão poderá ser composta por uma ou mais comparações, relacionadas entre si com operadores lógicos:

- AND: `;` ou `and`;
- OR: `,` ou `or`.

Por padrão, o operador AND tem precedência (ou seja, é avaliado antes de quaisquer operadoras OR). No entanto, uma expressão entre parênteses pode ser usada para alterar a precedência, produzindo qualquer que seja a expressão contida.

A comparação é composta por um seletor, um operador e um argumento.

Seletor deve ser composto de uma *string* se caracteres reservados.

Os operadores possíveis são:

- Igual a: `==`
- Não é igual a: `!=`
- Menor que: `=lt=` ou `<`
- Menor ou igual a: `=le=` ou `<=`
- Maior que o operador: `=gt=` ou `>`
- Maior ou igual a: `=ge=` ou `>=`
- Em: `=in=`
- Não em : `=out=`

Argumento pode ser um valor único, ou vários valores entre parênteses separados por vírgulas. O valor que não contém nenhum caractere reservado ou um espaço em branco pode ser não cotado, outros argumentos devem ser incluídos em aspas simples ou duplas. Para o caso de uma comparação com operador `==`, é possível utilizar o `*` no argumento, tendo o efeito de um operado *like* do *sql*.

Exemplos:

```
- name=="Kill Bill";year=gt=2003
- name=="Kill Bill" and year>2003
- genres=in=(sci-fi,action);(director=='Christopher Nolan',actor==*Bale);year=ge=2000
- genres=in=(sci-fi,action) and (director=='Christopher Nolan' or actor==*Bale) and year>=2000
- director.lastName==Nolan;year=ge=2000;year=lt=2010
- director.lastName==Nolan and year>=2000 and year<2010
- genres=in=(sci-fi,action);genres=out=(romance,animated,horror),director==Que*Tarantino
- genres=in=(sci-fi,action) and genres=out=(romance,animated,horror) or director==Que*Tarantino
```

NOTA: O componente rsql-parser implementa esta especificação.

Outras padronizações

- Informe no *header "accept":"application/json"* e *"content-type":"application/json"*;
- Os formatos de data devem seguir o padrão ISO-8601;
- Informe se o serviço deve ou não utilizar cache, exemplo: *"cache-control":"no-cache"*. No Internet Explorer, para desativar o cache, a configuração é um pouco diferente dos demais browsers: *"pragma": "no-cache".*

Documentação.

A documentação da API deverá ser disponibilizada nos idiomas Português Brasileiro e Inglês em HTML com forte utilização de links, contendo:

- Página com índice de recursos primários (substantivos);
- Página de Visão Geral, contendo explicações e dicas de uso da API;
- Para cada recurso primário, página índice com as operações referentes ao recurso, contendo:
    - Título da operação;
    - Método (Verbo + URI).
- Para cada operação, página com detalhamento da operação, contendo:
    - Título da operação;
    - Descrição da operação;
    - Método (verbo + URI);
    - Parâmetros:
        - Tabela de *Path Parameters* contendo (se for o caso):
            - Nome do parâmetro;
            - Descrição do parâmetro;
        - Tabela de *Query Parameters* contendo (se for o caso):
            - Nome do parâmetro;
            - Tipo de parâmetro;
            - Descrição do parâmetro.
    - Resultado contendo:
        - Descrição;
        - Tabela de propriedade de retorno contendo (se for o caso):
            - Nome da propriedade;
            - Tipo de propriedade;
            - Descrição da propriedade;
    - Códigos de retorno contendo:
        - Código;
        - *Media Type*;
        - Descrição.
    - Exemplo contendo:
        - *Request*:
            - Verbo + URI, com um exemplo prático;
            - Request Body (se for o caso) contendo documento JSON;
        - *Response*:
            - Documento JSON de resultado.