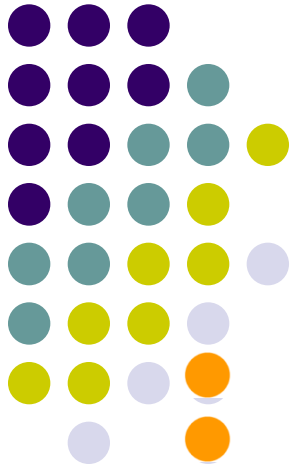
Arquitetura de Software

JSON e YAML

Linguagens para serialização de informações

José Motta Lopes
josemotta@bampli.com



Agenda

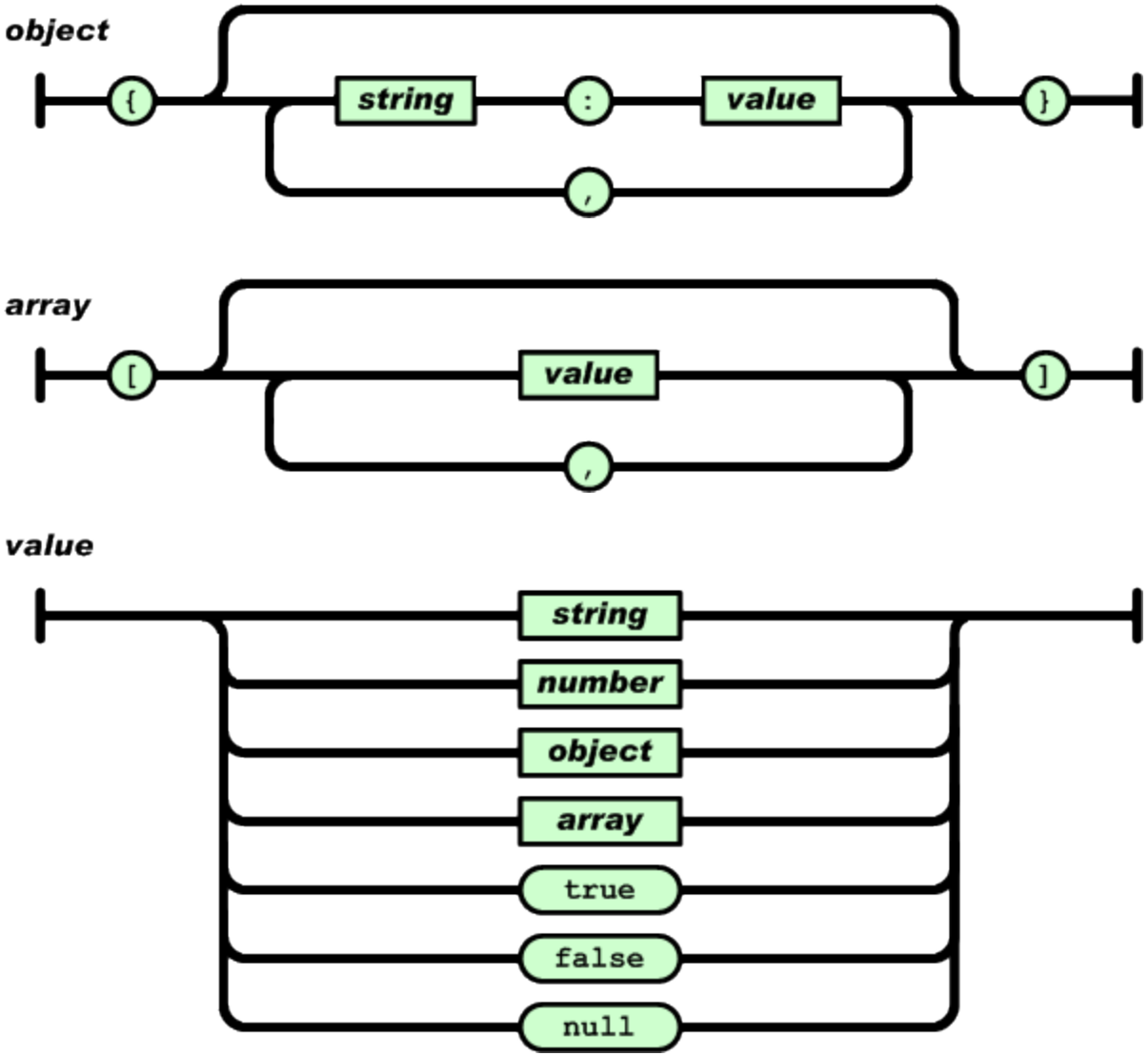
- JSON
- Sintaxe JSON
- Exemplo JSON
- JSON x XML
- YAML
- Sintaxe YAML
- API com YAML
- Exemplo YAML

JSON

JAVASCRIPT OBJECT NOTATION

- **Formato leve para intercâmbio de dados**
- **Fácil leitura e escrita para humanos**
- **Fácil para máquinas analisar e gerar**
- **Convenções familiares aos programadores**
- **Linguagens C, C++, C#, Java, Javascript, Perl, Python, etc.**

Sintaxe JSON

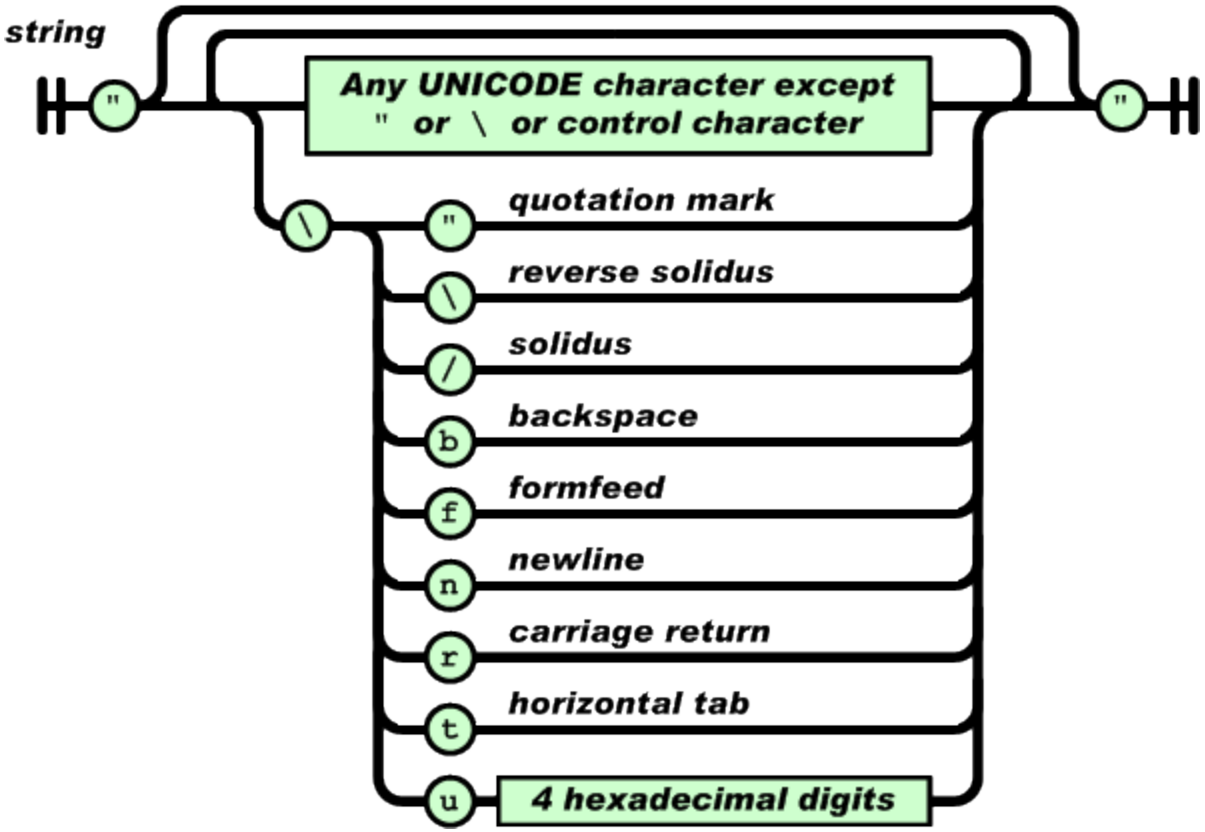


Objeto é um conjunto não ordenado de pares com nome/valor

Array é uma coleção ordenada de valores

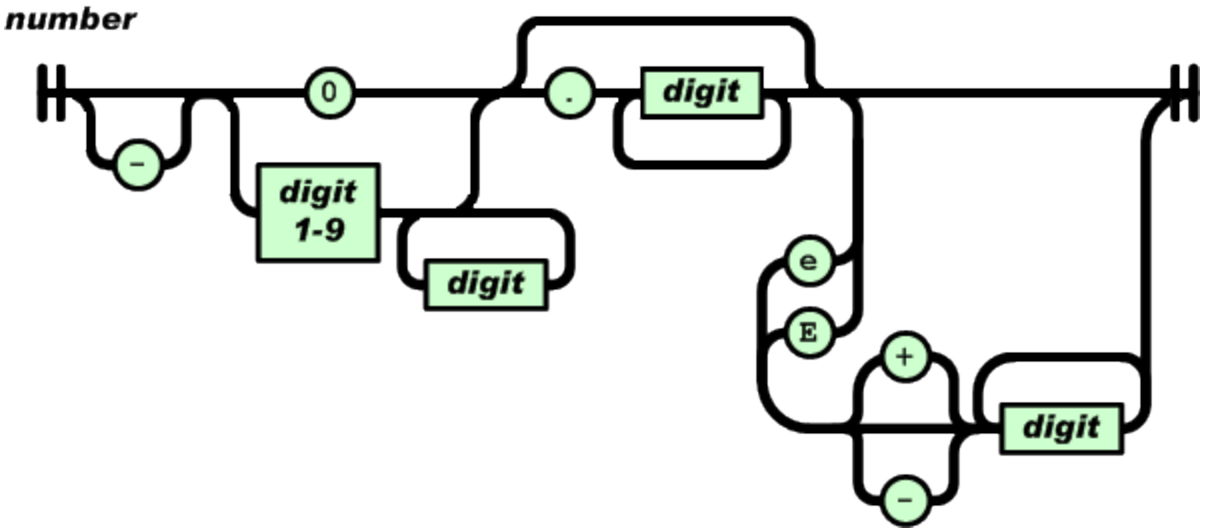
Valor pode ser um string com aspas ou um número, ou true / false / null, ou um objeto ou um array

Sintaxe JSON



String é uma sequência caracteres UNICODE, envoltos em aspas duplas e utilizando escapes de barra invertida

Sintaxe JSON



Número é semelhante aos números C ou Java, exceto que os formatos octal e hexadecimal não são usados

Exemplo JSON

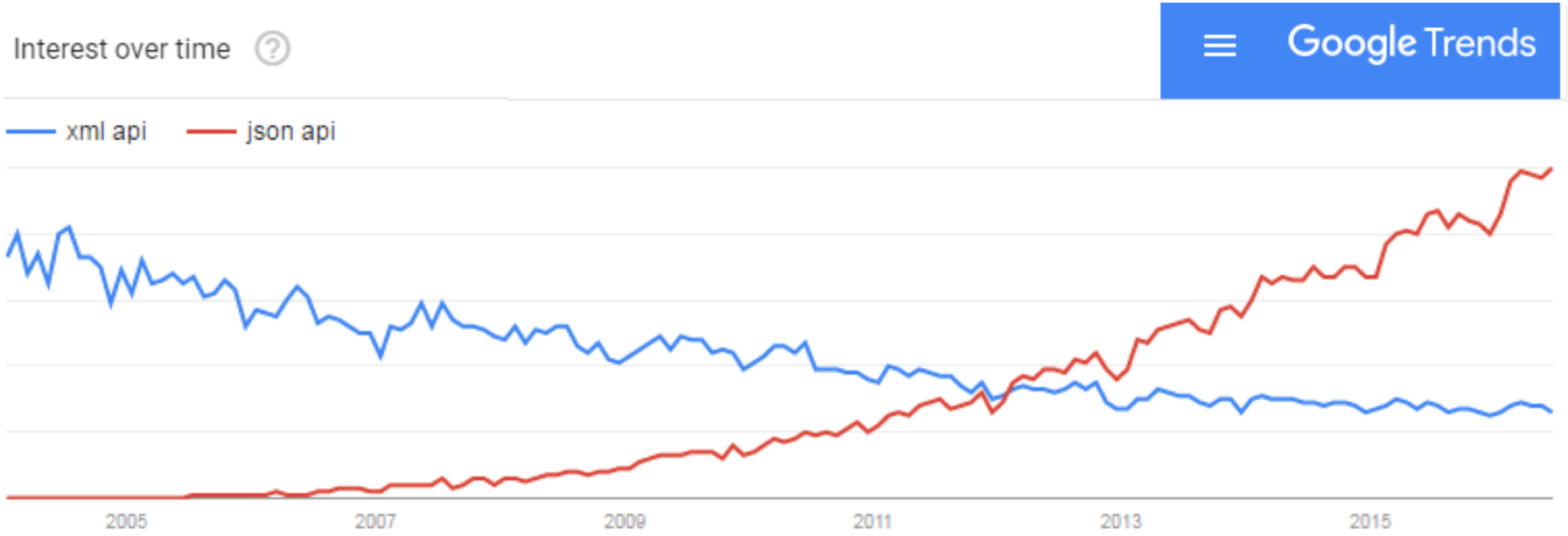
```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Possível representação JSON para descrição de uma pessoa

JSON x XML

API TENDENDO PARA RESPOSTAS JSON

- **XML ficando para trás, mais difícil de ler e analisar**
- **Legado do mercado corporativo usa XML**



YAML

YAML AIN'T MARKUP LANGUAGE

- **Formato leve comumente usado em arquivos de configuração**
- **Linguagem de serialização de informações**
- **Projetada para fácil leitura e escrita de humanos**
- **Fácil para máquinas analisar e gerar**
- **Menos complexa que XML e JSON**

Sintaxe YAML

Sequência Simples em YAML?

```
- maçã
- banana
- cenoura
```

Uma lista é especificada com cada membro em uma linha, iniciada com um traço

Sequências Aninhadas em YAML?

```
-
 - foo
 - Barra
 - baz
```

Incluir uma sequência dentro de outra sequência com traço vazio e lista recuada

Sequências Mistas em YAML?

```
- maçã
-
 - foo
 - Barra
 - x123
- banana
- cenoura
```

Sequências podem conter quaisquer dados YAML, incluindo outras sequências

Sequências profundamente aninhadas em YAML?

```
-
 -
  - uno
  - dos
```

Sequências são aninhadas, com cada nível de recuo representando profundidade

Sintaxe YAML

Mapeamento Simples em YAML?

```
foo: whatever
bar: coisas
```

Seqüência em um mapeamento em YAML?

```
foo: whatever
Barra:
 - uno
 - dos
```

Mapeamentos aninhados em YAML?

```
foo: whatever
Barra:
 fruta: maçã
 nome: steve
 esporte: beisebol
```

Mapeamento Misto em YAML?

```
foo: whatever
Barra:
 -
   fruta: maçã
   nome: steve
   esporte: beisebol
 - Mais
 -
   python: rochas
   perl: papers
   rubi: scissorses
```

Mapeamento é uma lista com chave de dicionário ou hash separada por <:>

Valor no mapeamento pode ser uma sequência

Valor no mapeamento pode ser outro mapeamento

Mapeamento pode conter qualquer variedade de mapeamentos e sequências como valores

Sintaxe YAML

Atalho de Mapeamento em Sequência em YAML?

```
- trabalhe em YAML.py:
   - trabalhar na loja
```

Adicione um mapeamento a uma sequência, usando atalho na mesma linha do traço

Atalho de Sequência no Mapeamento em YAML?

```
permitir:
- 'localhost'
- '% .sourceforge.net'
- '% .freepan.org'
```

Traço em uma sequência conta como recuo, não precisa recuar mais espaços

Matriz Inline Simples em YAML?

```
---
seq: [a, b, c]
```

Sequência em linha, sintaxe inline. Separar cada entrada com vírgula e colocar entre colchetes

Hash Inline Simples em YAML?

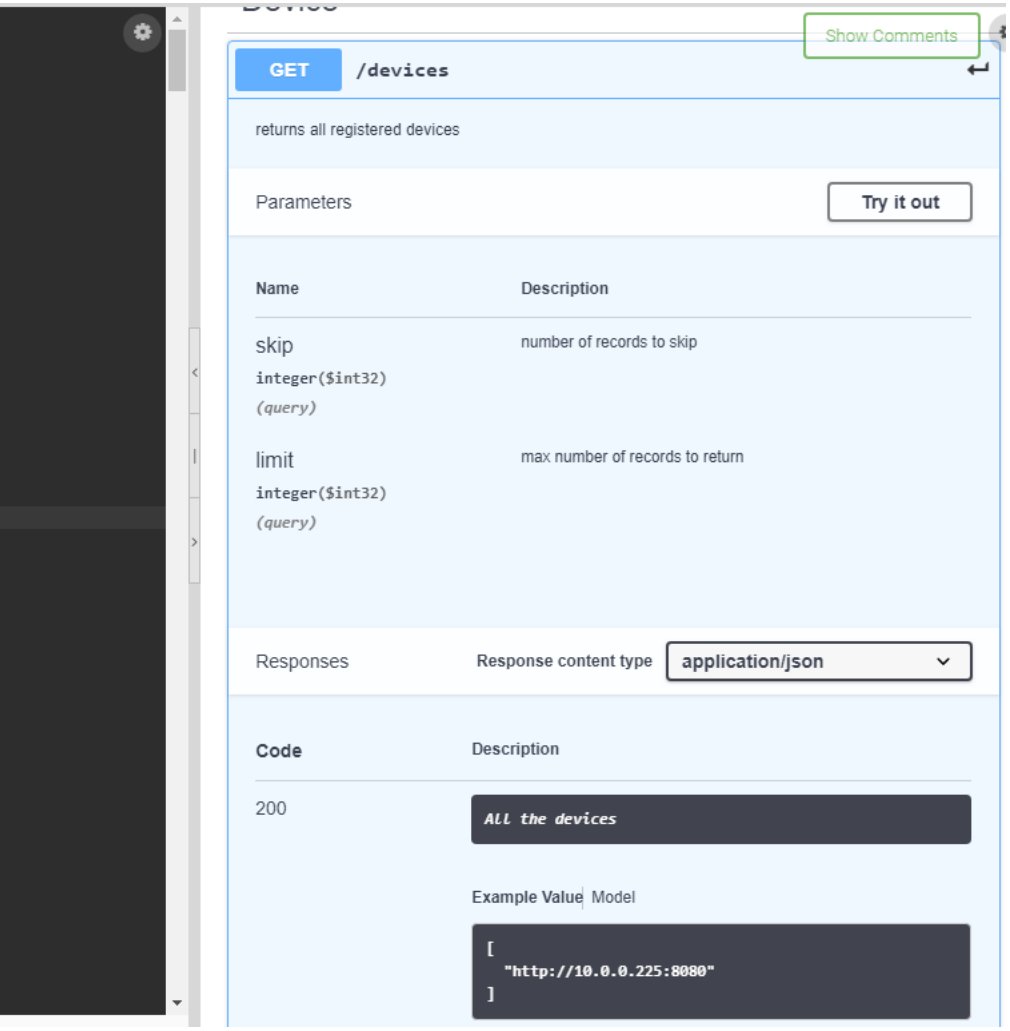
```
---
hash: {nome: Steve, foo: bar}
```

Mapeamento em linha, sintaxe inline. Cada par valor/chave é separado por <:>, com <,> entre cada entrada. Colocar entre chaves

API com YAML

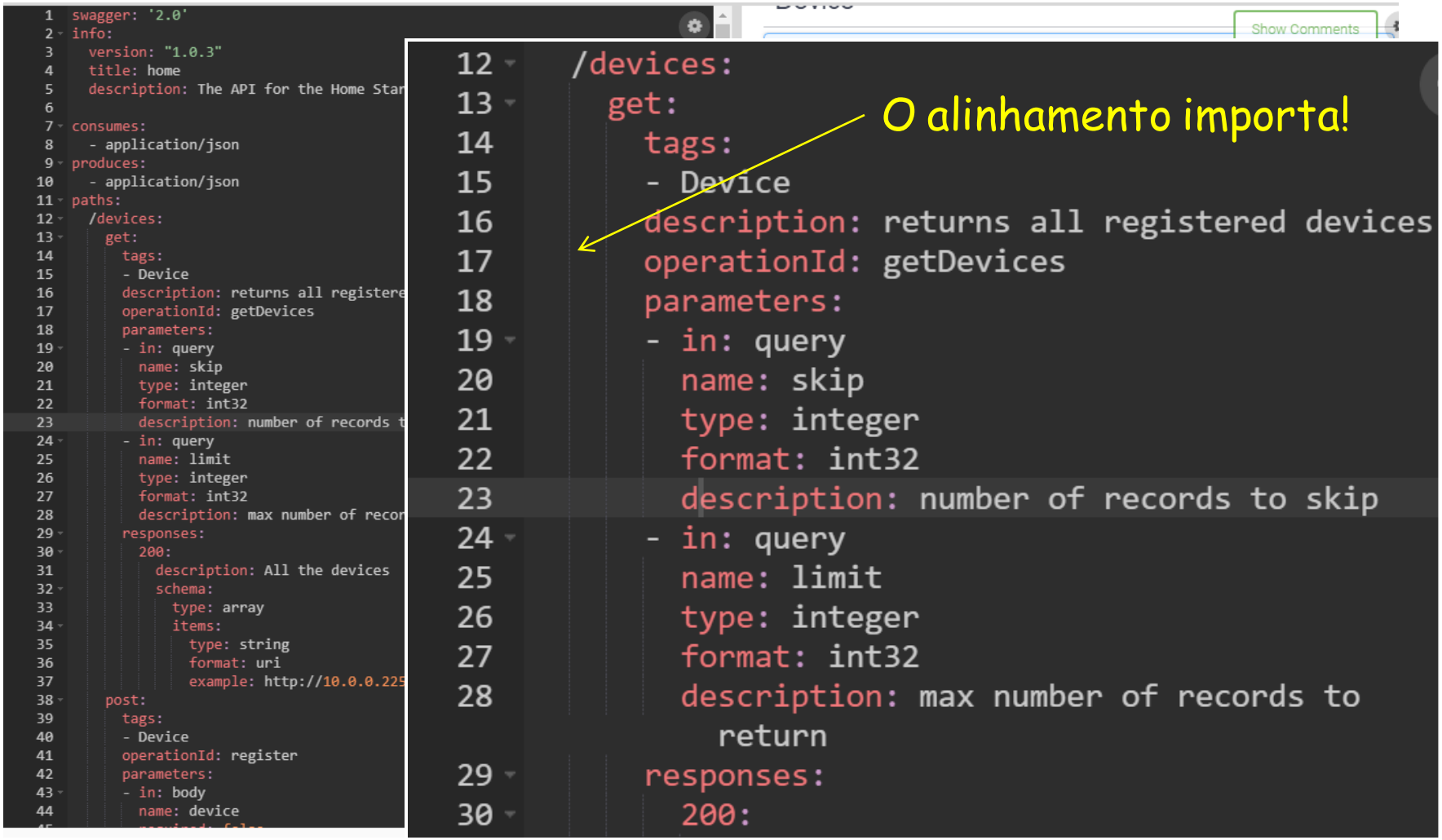
SWAGGER HUB

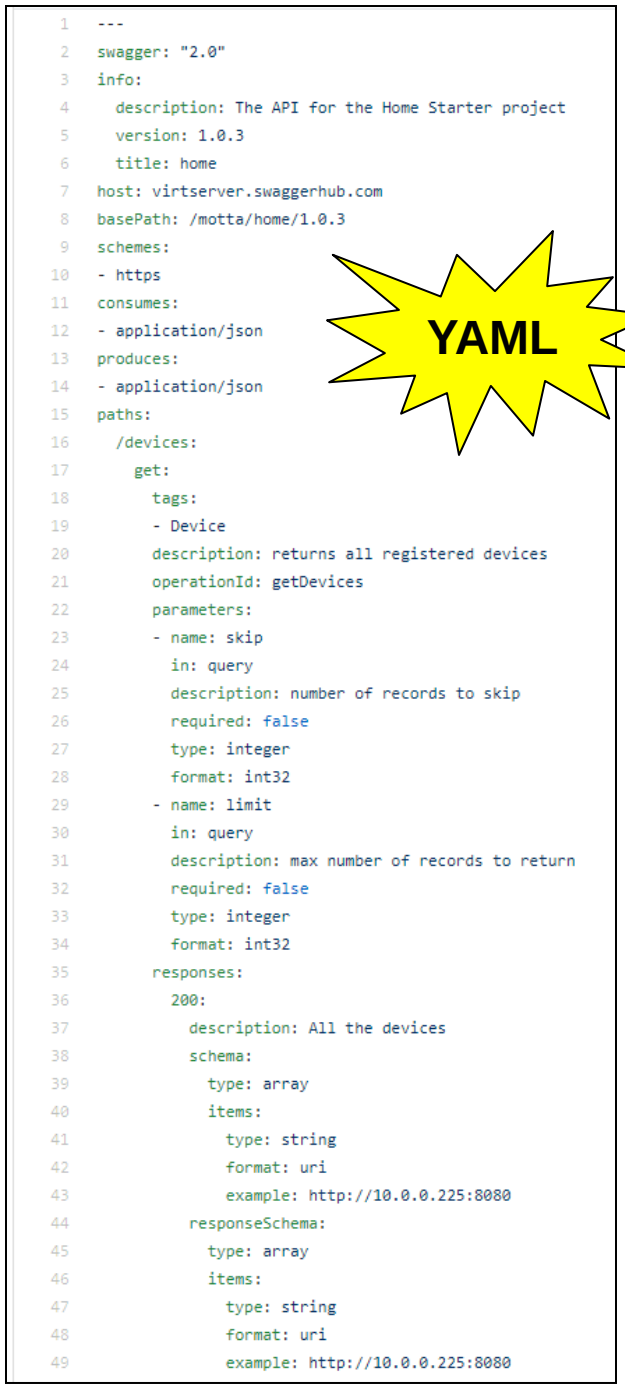
```yaml
swagger: '2.0'
info:
  version: "1.0.3"
  title: home
  description: The API for the Home Starter project

consumes:
  - application/json
produces:
  - application/json
paths:
  /devices:
    get:
      tags:
      - Device
      description: returns all registered devices
      operationId: getDevices
      parameters:
      - in: query
        name: skip
        type: integer
        format: int32
        description: number of records to skip
      - in: query
        name: limit
        type: integer
        format: int32
        description: max number of records to return
      responses:
        200:
          description: All the devices
          schema:
            type: array
            items:
              type: string
              format: uri
              example: http://10.0.0.225:8080
    post:
      tags:
      - Device
      operationId: register
      parameters:
      - in: body
        name: device
```



API com YAML

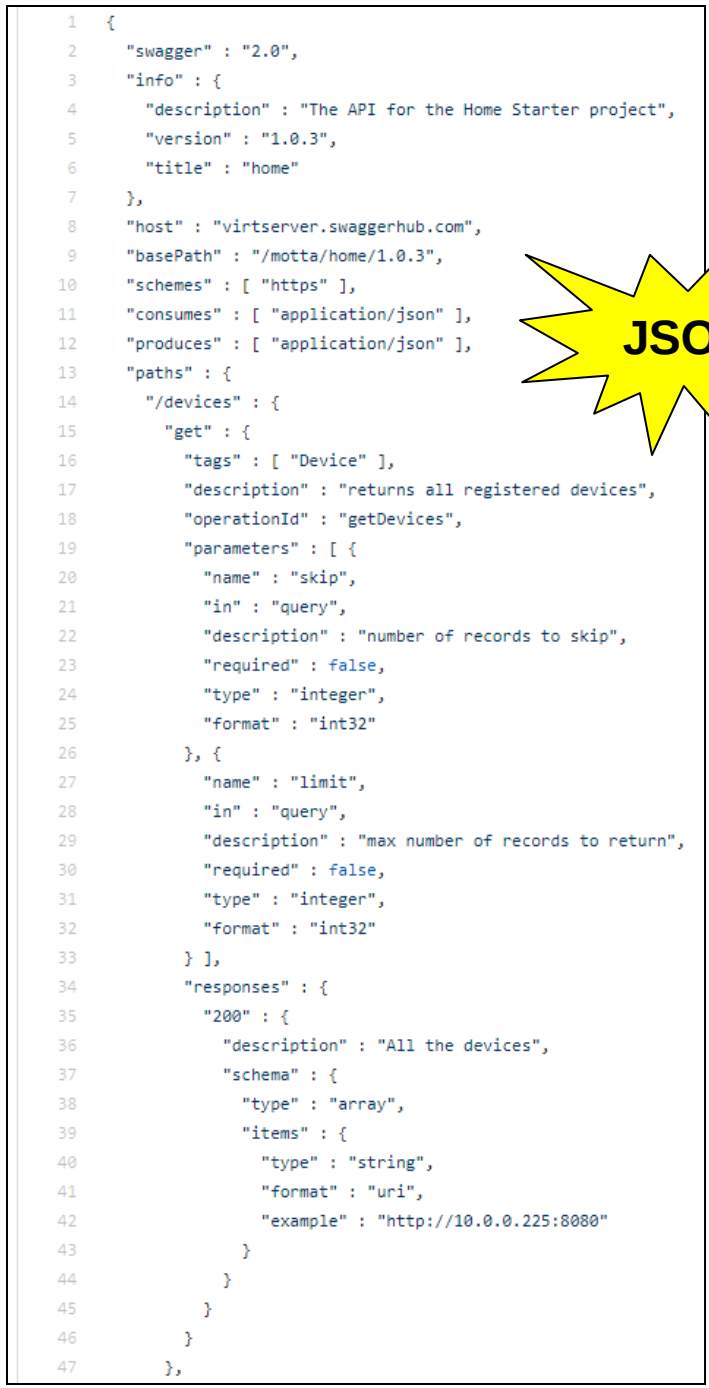
SWAGGER HUB





```yaml
---
swagger: "2.0"
info:
  description: The API for the Home Starter project
  version: 1.0.3
  title: home
host: virtserver.swaggerhub.com
basePath: /motta/home/1.0.3
schemes:
- https
consumes:
- application/json
produces:
- application/json
paths:
  /devices:
    get:
      tags:
      - Device
      description: returns all registered devices
      operationId: getDevices
      parameters:
      - name: skip
        in: query
        description: number of records to skip
        required: false
        type: integer
        format: int32
      - name: limit
        in: query
        description: max number of records to return
        required: false
        type: integer
        format: int32
      responses:
        200:
          description: All the devices
          schema:
            type: array
            items:
              type: string
              format: uri
              example: http://10.0.0.225:8080
          responseSchema:
            type: array
            items:
              type: string
              format: uri
              example: http://10.0.0.225:8080
```

YAML



```json
{
  "swagger" : "2.0",
  "info" : {
    "description" : "The API for the Home Starter project",
    "version" : "1.0.3",
    "title" : "home"
  },
  "host" : "virtserver.swaggerhub.com",
  "basePath" : "/motta/home/1.0.3",
  "schemes" : [ "https" ],
  "consumes" : [ "application/json" ],
  "produces" : [ "application/json" ],
  "paths" : {
    "/devices" : {
      "get" : {
        "tags" : [ "Device" ],
        "description" : "returns all registered devices",
        "operationId" : "getDevices",
        "parameters" : [ {
          "name" : "skip",
          "in" : "query",
          "description" : "number of records to skip",
          "required" : false,
          "type" : "integer",
          "format" : "int32"
        }, {
          "name" : "limit",
          "in" : "query",
          "description" : "max number of records to return",
          "required" : false,
          "type" : "integer",
          "format" : "int32"
        } ],
        "responses" : {
          "200" : {
            "description" : "All the devices",
            "schema" : {
              "type" : "array",
              "items" : {
                "type" : "string",
                "format" : "uri",
                "example" : "http://10.0.0.225:8080"
              }
            }
          }
        }
      },
```

JSON