WEB APIs

Conceitos iniciais, métodos, status code, constraints e requisições com requests

Ely – elydasilvamiranda@gmail.com

Referências

- Esses slides usam como base as seguintes fontes:
  - http://www.restapitutorial.com/lessons/whatisrest.html
  - https://classroom.udacity.com/courses/ud388
  - https://app.pluralsight.com/library/courses/creating-web-apis-developers-love/table-of-contents

O que é uma API?

- Application Programing Interface;
- É um conjunto de rotinas e padrões de programação para acesso a um aplicativo;
- Conecta diferentes sistemas computacionais;
- São projetadas para fácil interação/utilização por outros sistemas;
- Não são fortemente acopladas, ao contrário de funções dentro de um sistema:
- São fáceis de expandir;
- Tipicamente disponibilizam funcionalidades CRUD autocontidas.

APIs clássicas

- Bancos de dados;
- DLLs;
- RPC e CORBA;
- APIs do Windows;
- SOAP.

WEB APIs

- Estendem o produto original "fora" do sistema principal;
- Permitem que aplicações de terceiros acessem/alterem dados de outra aplicação;
- A troca de dados em WEB APIs é feita tipicamente utilizando JSON ou XML;

WEB APIs

- Uma comunicação via WEB API se dá geralmente da seguinte forma:
  - É disparada uma requisição com algum verbo HTTP (GET, POST, PUT...) para uma URL
    - Dados podem ser enviados nessa requisição no cabeçalho, URL ou no corpo da requisição;
  - Um servidor processa a requisição;
  - Dados são retornados, tipicamente utilizando arquivos JSON ou XML;

Exemplo:

- Requisição via GET na URL abaixo:
  - https://viacep.com.br/ws/64000040/json
- Resposta do servidor:


```
{
  "cep": "64000-040",
  "logradouro": "Praça da Liberdade",
  "complemento": "",
  "bairro": "Centro",
  "localidade": "Teresina",
  "uf": "PI",
  "unidade": "",
  "ibge": "2211001",
  "gia": ""
}
```

Algumas WEB APIs

- Google maps:
  - Dados de localizações;
  - https://developers.google.com/maps/
- Numbers API:
  - Curiosidades sobre números e datas
  - http://numbersapi.com/
- Postmon:
  - Consulta de dados postais
  - http://postmon.com.br/

Algumas WEB APIs

- Clima Tempo:
  - Dados sobre clima e tempo;
  - https://advisor.climatempo.com.br/
- GitHub:
  - API do Gi
  - https://developer.github.com/v3/
- IP Location:
  - Obter a localizaçãopelo IP
  - https://www.iplocation.net/

Algumas WEB APIs

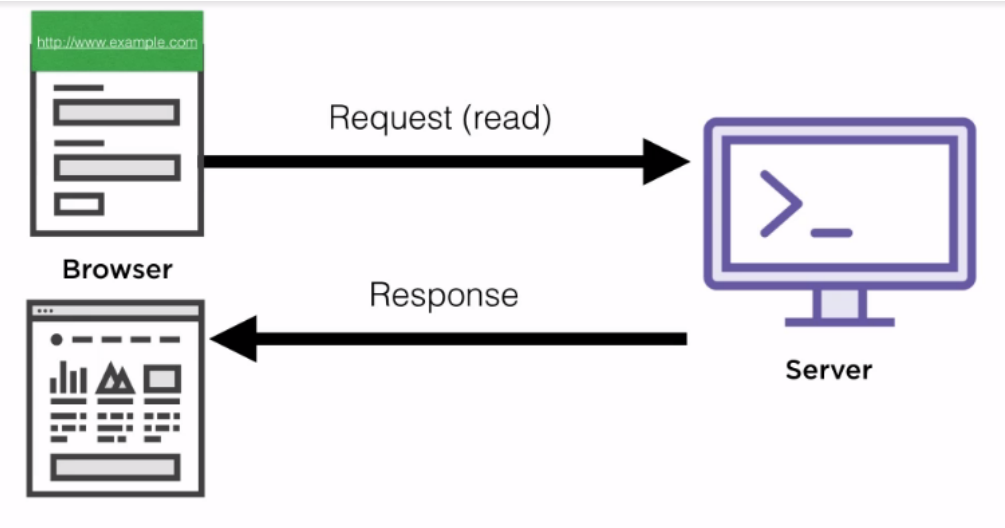
- Translation API:
  - API para traduções entre idiomas;
  - https://cloud.google.com/translate/?hl=pt-br
- Vagalume API:
  - Pesquisa sobre artistas e músicas;
  - https://api.vagalume.com.br/
- JSON Placeholder:
  - API fake que aceita vários métodos;
  - https://jsonplaceholder.typicode.com/

Algumas WEB APIs

- API Jarbas:
  - Gastos relacionados a deputados;
  - https://jarbas.serenatadeamor.org
- Fixer API
  - Conversão de moedas;
  - https://github.com/fixerAPI

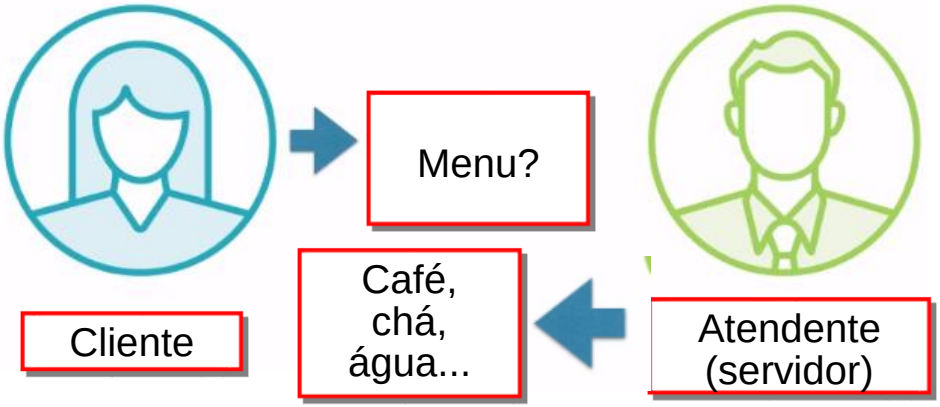
Interação Cliente-Servidor

- No contexto de navegadores web, um dos mais comuns usuários do HTTP;
- Quando um cliente faz uma requisição, ele digita uma URL ou submete um formulário;
- O servidor então faz algum processamento e retorna uma resposta ao cliente.



Café e uma analogia com uma WEB API

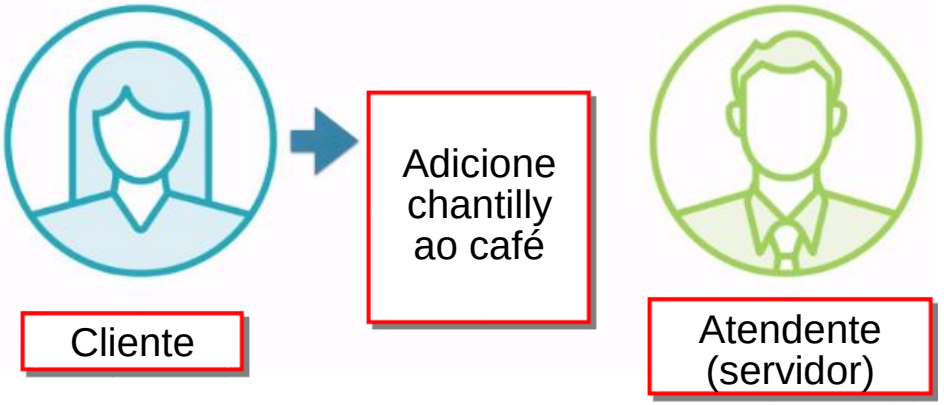
- Perguntar as opções de um menu significa obter (GET) opções;



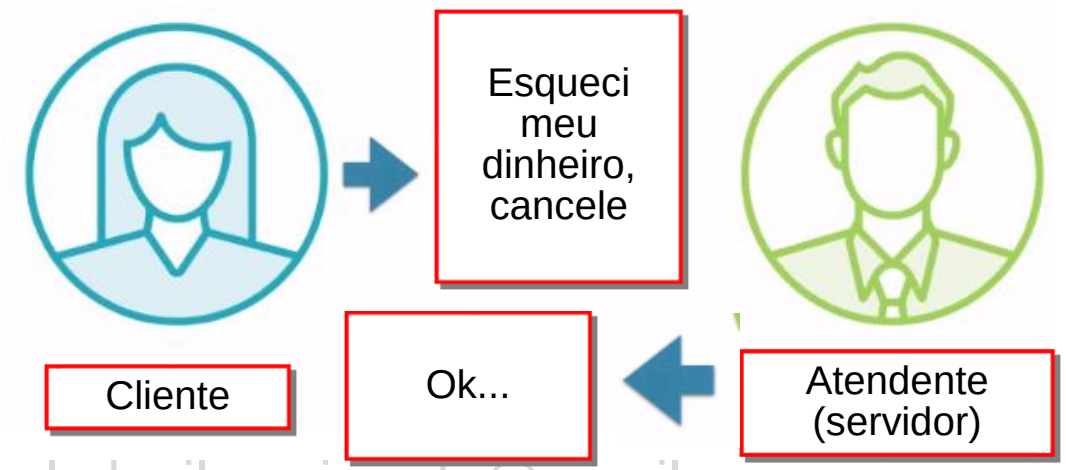
- Pedir um café significa em algum momento "criar" (CREATE) um pedido;

Café e uma analogia com uma WEB API

- Ao pedir complementar o café, você está atualizando (UPDATE) o pedido



- Cancelar o pedido, significa excluir (DELETE) o pedido;



APIs com a arquitetura REST

- Embora existam vários tipos de WEB APIs, focaremos nas APIs que usam a arquitetura REST;
- REST = REpresentional State Transfer;
- Forma simples de prover “serviços” e “recursos”;
- Alternativa ao desenvolvimento baseado em formulários (retornam sempre HTML);
- Permite diversos “front-ends” façam uso de uma API bem definida.

REST

- Usa semânticas de métodos HTTP;
- São fáceis de testar: uma API bem documentada pode ser acessada de um navegador comum;
- Protegem métodos internos de acesso não autorizados
- Expõem dados de um sistema como objetos;
- Protegem/isolam tanto cliente como servidor de mudanças inesperadas:
  - pois a API deve ser mantida independente da implementação

Usos comuns

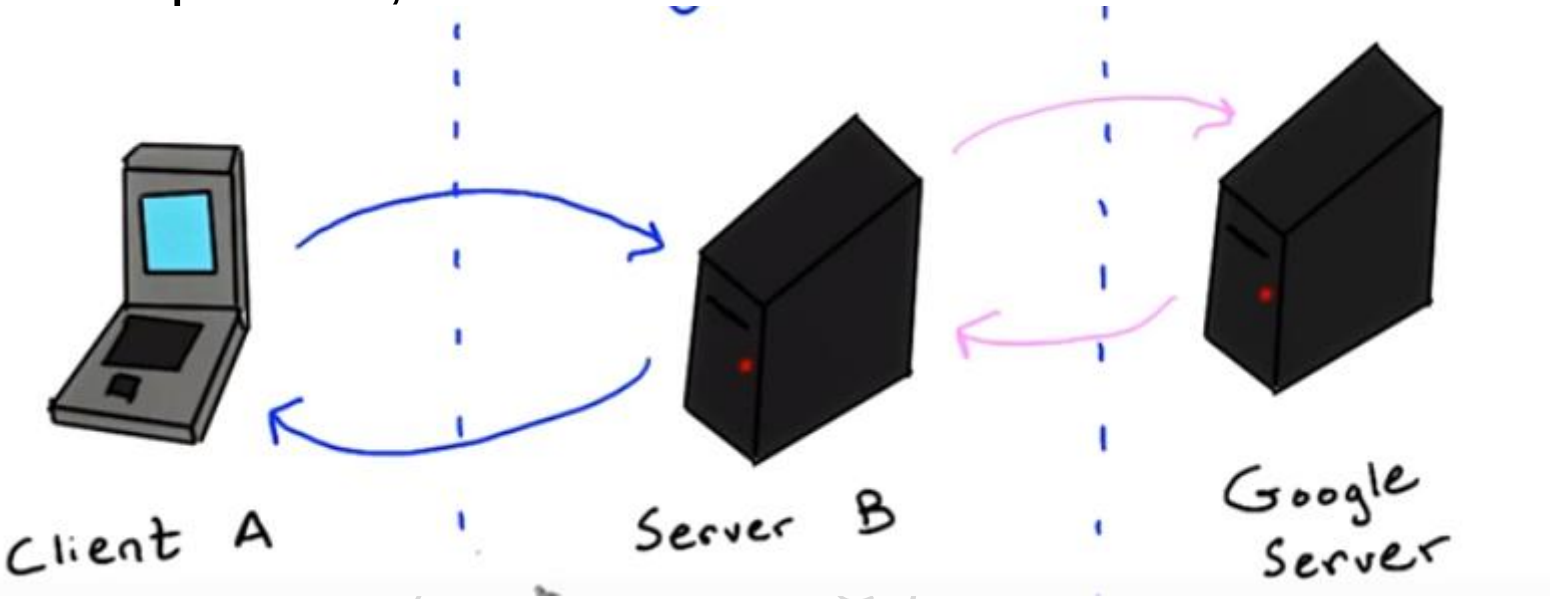
- Single-page Applications;
- Serviços de tempo real;
- Criação de APIs externas;
- Aplicações móveis;
- WebComponents e projeto de sites modulares;
- Uso em conjunto com frameworks JS.

REST Constraints

- Cliente-servidor:
  - ...
- Stateless:
  - O servidor não guarda o estado do cliente entre as requisições;
  - Cada requisição deve ter seu contexto para o processamento;
  - Qualquer sessão de usuário deve ser mantida pelo cliente.
- Cacheable:
  - As informações podem ser mantidas em cache no cliente, caso não tenham mudado no servidor;

REST Constraints

- Interface uniforme:
  - Utiliza verbos HTTP para as requisições do cliente. Tipicamente: GET, PUT, POST e DELETE;
- Sistema em camadas:
  - Siginica que um cliente pode ter acesso a endpoints que se confiam em outros endpoints de forma transparente;



REST Constraints

- Code on demand (opcional):
  - Siginifica que códigos podem ser enviados do servidor para o cliente para posterior execução.

Estratégias de projeto de APIs

Projeto clássico	Projetando API First
Frontend e backend fortemente acoplados	Conjunto de APIs entre frontend e backend
APIs são adicionadas ao backend separadamente	APIS bem testadas e integradas
Código duplicado	Sem duplicação de código
Features e funcionalidades inconsistentes	Melhor isolamento de interfaces e consistência e confiabilidade
APIs são cidadãs de segunda classe	APIs são cidadãs de primeira classe

HTTP Transactions

- HyperText Transfer Protocol;
- Projetado inicialmente para tráfego de páginas em navegadores;
- Request/Response;
- Stateless: cada transação é autocontida e não lembra das demais operações anteriores.

Métodos HTTP

- Os métodos HTTP são conhecidos como "verbos";
- Os mais comuns são análogos às necessidades CRUD;

Métodos HTTP

- GET:
  - uma operação simples de leitura;
  - Usado para obter um ou mais recursos do sistema;
  - O verbo mais comum;
- PUT:
  - Utilizado para se atualizar um recurso específico e já existente;
  - Em alguns casos, é utilizado para atualizar uma lista de recursos
  - Ao utilizá-lo, espera-se que sejam enviadas todas a informações do objeto, não somente as que serão atualizadas.
- POST:
  - Usado para adicionar um novo recurso;
  - A API deve responder à criação de um novo recurso com um ID ou a URL do item recém criado;
- DELETE:
  - Apaga um recurso no servidor;

Métodos HTTP

- Nota:
  - antigas APIs usavam apenas GET e POST para todas as requisições, o que não é mais o caso;

Uso correto de métodos HTTP

- Sempre use o verbo com a semântica correta;
- Um contra exemplo é a API do Flickr, que sobrecarrega o uso do POST

Flickr Call	RESTful Call
GET /services/rest/?method=flickr.activity.userPhotos	GET /services/rest/activity/userPhotos
POST /services/rest/?method=flickr.favorites.add	POST /services/rest/favorites
POST /services/rest/?method=flickr.favorites.remove	DELETE /services/rest/favorites/:id
POST /services/rest/?method=flickr.galleries.editPhoto	PUT /services/rest/galleries/photo/:id

Status codes

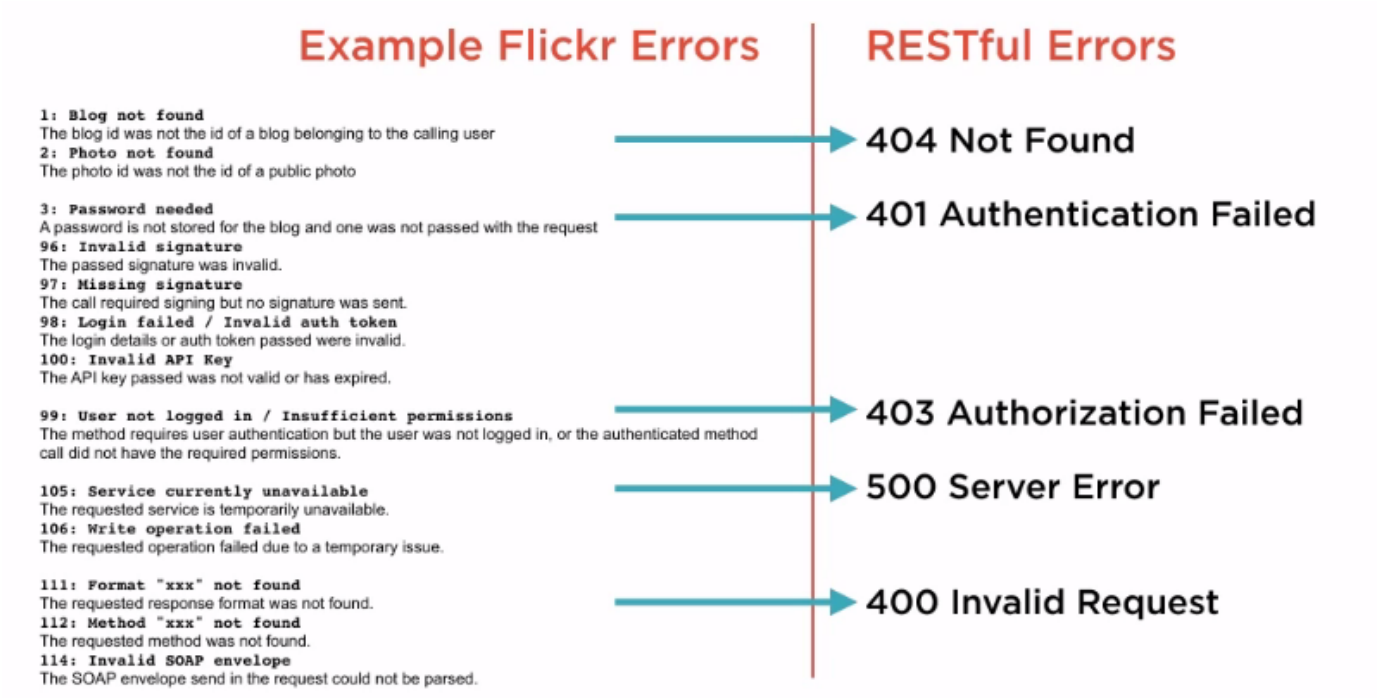
- Utilizamos os verbos HTTP para indicar ações no servidor e o servidor responde com Status Codes
- Cada resposta possui um status code associada a ela e a um eventual conteúdo;
- Os Status Codes cobrem uma grande faixa de mensagens possíveis, como: ok, não encontrado, proibido...
- Essas mensagens são números de 3 dígitos e seu significado amplamente conhecidos;

Status codes

- Esses dígitos são agrupados em centenas;
- 5xx: erros internos do servidor – problemas de processamento, null pointer exceptions...
- 4xx: erros do cliente – autenticação, dados ou tipos mal formados;
- 3xx: recursos que mudaram de local, redirecionamentos;
- 2xx: indicam que tudo funcionou bem.

Uso correto de status codes

- Apesar de serem erros genéricos, os status codes são universais;
- Complementar ao status code, uma API pode retornar mensagens complementares;
- Mais um contra exemplo para a API do Flickr:



Algumas ferramentas

- Postman
- Curl
- ... Ou em python...

Requisições HTTP

- Requisições via web driver não são performáticas:
  - Uma instância do navegador ficar aberta, ainda que em headless mode;
  - Há um gap de comunicação entre os scripts e o web driver;
- Para raspar dados de uma página, deve-se preferencialmente baixá-la via HTTP;
- A forma mais simples de baixar uma página é utilizar bibliotecas que façam requisições HTTP;

Requests

- Para fazer requisições HTTP com Python, há várias bibliotecas;
- As que mais se destacam são:
  - httplib (http.client) e urllib: nativas do Python, porém muito verbosas e com algumas limitações;
  - Requests: implementação independente, mas não verbosa com várias características relevantes;

Requests

- "HTTP para Humanos";
- Documentação:
  - http://docs.python-requests.org/pt_BR/latest/
- Instalação:
  - pip install requests

Exemplo 1

- Buscando um CEP:

```
# -*- coding: utf-8 -*-
import requests

cep = '64218100'
url = "https://viacep.com.br/ws/{cep}/json/"
url = url.format(cep=cep)

response = requests.get(url).json()
```

Exemplo 2

- Acessando as "propriedades" do JSON
- ```
  # -*- coding: utf-8 -*-
  import requests

  cep = '64218100'
  url = "https://viacep.com.br/ws/$s/json/" % cep

  response = requests.get(url).json()
  # print(response)

  print(" Logradouro: %s \n Bairro: %s \n
          Localidade: %s \n UF: %s" %
        (response['logradouro'],
         response['bairro'],
         response['localidade'], response['uf']))
  ```
  ```

Exemplo 3

- Tradução

```
# coding: utf-8
import requests, json
key = 'sua chave'
q = 'Prezados, boa tarde, gerem uma chave para a API.'
target = 'en'

url =
'https://translation.googleapis.com/language/translate/v2'
params = {'key':key, 'q':q, 'target':target}

response = requests.get(url, params=params)

json = response.json()
print(json['data']['translations'][0]['translatedText'])
```

Exemplo 4

- Localizações

```
import requests

key = 'sua outra chave'
localizacao = "Avenida frei serafim"
url =
'https://maps.googleapis.com/maps/api/geocode/json?address=%s&key=%
s' % (localizacao, key)
response = requests.get(url)
json = response.json()
print (json)
latitude = json['results'][0]['geometry']['location']['lat']
longitude = json['results'][0]['geometry']['location']['lng']

print('Latitude: %f  - Longitude: %f' % (latitude, longitude ))
```

Exemplo 5

- Criando um "registro" no JSON Placeholder e tentando apagar outro

```
import requests

url = 'https://jsonplaceholder.typicode.com/todos/'
dados = {
    "userId": 1,
    "title": 'Prepare class notes',
    "completed": False
}

response = requests.post(url, data = dados)
print(response.json())

response = requests.delete(url + '/210')
# 404
print(response)
```

WEB APIs

Conceitos iniciais, métodos, status code, constraints e requisições com requests

Ely – elydasilvamiranda@gmail.com