Programação Orientada a Serviços

Prof. Diego Cirilo

Aula 11: Clientes para APIs REST

APIs e WebServices

- Application Programming Interfaces
- Web Services: caso especial

Protocolo HTTP

- *Hyper text transfer protocol*
- Camada de aplicação
- Baseado no modelo cliente-servidor
- Padrão de mensagens de requisição e respostas
- Porta 80 (ou 443 para HTTPS)

Métodos HTTP

Método	Descrição
GET	Recebe um recurso existente
POST	Cria um novo recurso
PUT	Atualiza um recurso existente
PATCH	Atualiza parcialmente um recurso existente
DELETE	Remove um recurso

Códigos de status

Faixa	Categoria
2xx	Sucesso
3xx	Redirecionamento
4xx	Erro de cliente
5xx	Erro de servidor

Códigos de status

Código	Significado	Descrição
200	OK	The requested action was successful.
201	Created	A new resource was created.
202	Accepted	The request was received, but no modification has been made yet.
204	No Content	The request was successful, but the response has no content.
400	Bad Request	The request was malformed.
401	Unauthorized	The client is not authorized to perform the requested action.
404	Not Found	The requested resource was not found.
415	Unsupported Media Type	The request data format is not supported by the server.
422	Unprocessable Entity	The request data was properly formatted but contained invalid or missing data.
500	Internal Server Error	The server threw an error when processing the request.

Cabeçalho HTTP

Dado	Descrição
Accept	O tipo de conteúdo que o cliente aceita
Content-Type	O tipo de conteúdo que o servidor retorna
User-Agent	Que software o cliente está usando para comunicar com o servidor
Server	Que software o servidor usa para comunicar com o cliente
Authentication	Quem chama a API que quais suas credenciais

Principais tipos de API

- REST
  - Representational State Transfer
  - Métodos HTTP
- SOAP
  - Simple Object Access Protocol

Princípios do REST

- Interface Uniforme
  - Identificação de recursos
  - Manipulação de recursos através de representações
  - Mensagens auto-descritivas
  - Uso de hyperlinks
- Cliente - Servidor
- *Stateless*
- *Cacheable*
- Sistema em camadas

Interagindo com API REST

- Uma API REST expõe URLs públicas que as aplicações podem acessar
- *API Endpoints*

RESTful resources

Método HTTP	API endpoint	Descrição
GET	/clientes	Lista clientes.
GET	/clientes/<cliente_id>	Cliente específico.
POST	/clientes	Cria um novo cliente.
PUT	/clientes/<cliente_id>	Atualiza um cliente.
PATCH	/clientes/<cliente_id>	Atualiza parcialmente um cliente.
DELETE	/clientes/<cliente_id>	Remove um cliente.

Exemplos

- https://suap.ifrn.edu.br/api/docs/
- https://developer.twitter.com/en/docs/api-reference-index
- https://docs.github.com/en/rest

Interagindo com APIs REST em Python

- Biblioteca `requests`

```
$ pip install requests
```

GET

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos"
response = requests.get(api_url)
print(response.json())
```

Métodos do objeto `Response`

```
response.status_code
response.headers
response.json()
```

POST

- API pública que aceita posts:
  https://jsonplaceholder.typicode.com/todos

```
{
    "userId": 1,
    "title": "Comprar farinha",
    "completed": false
}
```

POST

```
import requests

api_url = "https://jsonplaceholder.typicode.com/todos"
todo = {"userId": 1, "title": "Comprar farinha", "completed": False}
response = requests.post(api_url, json=todo)
print(response.json())
print(response.status_code)
```

PUT

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/10"
response = requests.get(api_url)
print(response.json())

todo = {"userId": 1, "title": "Lavar carro", "completed": True}
response = requests.put(api_url, json=todo)
print(response.json())
print(response.status_code)
```

PATCH

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/10"
todo = {"title": "Cortar grama"}
response = requests.patch(api_url, json=todo)
print(response.json())
```

DELETE

```
import requests
api_url = "https://jsonplaceholder.typicode.com/todos/10"
response = requests.delete(api_url)
print(response.json())
```

Prática

- Acesse a documentação do jsonplaceholder.typicode.com
- Explore as operações com os recursos aninhados, exemplo:
  - *to-dos* do usuário 3
  - comentários do post 3
- Identifique os atributos através do navegador

Exercício

- Implemente uma CLI para CRUD dos usuários da JSON Placeholder
- Apresente as opções de:
  - Listar todos usuários
  - Listar as tarefas de um usuário específico (users/ID/todos)
  - Criar/Ler/Atualizar/Deletar usuários

API *Wrappers*

- Bibliotecas de mais alto nível para acesso a APIs
- Continuam usando URLs, json, requests, etc...
- Simplificam a operação para o dev
- Ex. `python-twitter`, `pygithub`

Exercício

- Desenvolva um wrapper para o CRUD da API dos *users* com as funções `list`, `create`, `read`, `update` e `delete`.
- Desenvolva uma CLI (como no exercício anterior) que use a sua biblioteca.
- Exemplo de uso:

```
import users_wrapper as users

user = users.read(user_id)
print(user["name"])

# Listar usuários
users-list = users.list()

users.delete(user_id)
```

Dúvidas?