



Quem se prepara, não para.

# Arquitetura de Aplicações Web

5º período

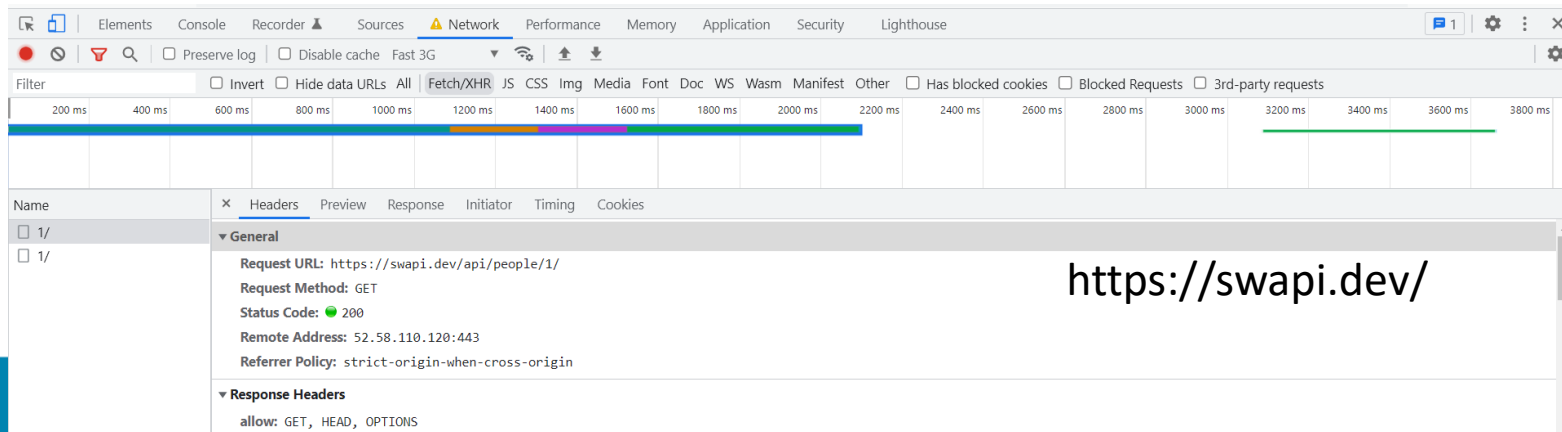
Professora: Michelle Hanne

# Sumário

- Consumindo biblioteca pública em JavaScript
  - Requisição Assíncrona com AJAX (XMLHttpRequest, JQuery e Fetch)
- StarWars API - <https://swapi.dev/>

# AJAX

- AJAX (***Asynchronous JavaScript and XML***) – ou seja, requisição assíncrona
- Surgiu no Internet Explorer, nos anos 2000, por Jesse Gareth
- Originalmente, usava-se JavaScript para fazer uma requisição de dados ao servidor, que respondia no formato XML (em vez de HTML)
- Hoje em dia, responde-se com qualquer objeto reconhecido pelo navegador (o mais comum é JSON)
- Podemos ver as requisições Ajax (Fetch/XHR) da página de inspeção (ferramenta do desenvolvedor):



The screenshot shows the Chrome DevTools Network tab. The top toolbar includes buttons for Elements, Console, Recorder, Sources, Network (active), Performance, Memory, Application, Security, and Lighthouse. Below the toolbar, there are checkboxes for 'Preserve log', 'Disable cache', and 'Fast 3G'. A filter bar shows 'Fetch/XHR' selected. A timeline at the top displays a request starting at approximately 1200ms and ending at 1600ms. The main pane shows the details of the selected request:

- Name:** 1/
- General:**
  - Request URL: <https://swapi.dev/api/people/1/>
  - Request Method: GET
  - Status Code: 200
  - Remote Address: 52.58.110.120:443
  - Referrer Policy: strict-origin-when-cross-origin
- Response Headers:**
  - allow: GET, HEAD, OPTIONS

# AJAX

## Exemplos de utilização:

- Carregar mais tweets
- Carregar mais produtos
- Carregar avaliações, comentários
- Carregar a próxima fase de um jogo
- Avisar ao back-end que algo mudou, sem sair da página. Exemplos:
  - Botão curtir
  - Enviar um comentário
  - Adicionar produto ao carrinho

# XMLHttpRequest

- Cada requisição Ajax é um objeto XMLHttpRequest. Exemplo do Twitter:

```
let requisicao = new XMLHttpRequest();
requisicao.onreadystatechange = callbackMaisTweets;
requisicao.responseType = 'json';
requisicao.open('GET', '/tweets/pagina/5');
requisicao.send();
```

```
function callbackMaisTweets() {
  if (requisicao.readyState === 4) { // 4: DONE (resp. recebida)
    if (requisicao.status === 200) { // 200: código Ok do HTTP
      // a resposta chegou e foi um arquivo
      // .json com um array de tweets:
      let novosTweets = requisicao.response.arrayComNovosTweets;
      novosTweets.forEach(colocaTweetNaPagina);
    } else {
      console.log('Erro ao carregar mais tweets. Código HTTP: '
        + requisicao.status);
    }
  }
}
```

- Uma callback (definida em **onreadystatechange**) é invocada a cada mudança de estado da requisição.

```
{
  "quantidade": 20,
  "arrayComNovosTweets":
    [ { "autor": "Sensacionalista",
      "texto": "Grupo de feministas pró-
      Bolsonaro cria novo grupo:
      Vegetarianos pró carne mal passada",
      "curtidas": 2
    } /* mais 19 tweets aqui... */ ] }
```

# Estados de um XMLHttpRequest

- 0 UNSENT: open() ainda não foi invocado
- 1 OPENED: send() ainda não foi invocado
- 2 HEADERS\_RECEIVED: send() foi invocado e os cabeçalhos da resposta já estão disponíveis (chegaram)
- 3 LOADING: fazendo download da resposta
- 4 DONE: operação finalizada

# Ajax com jQuery

- O jQuery possui uma abstração do objeto XMLHttpRequest para facilitar a realização de requisições Ajax. Exemplo:

```
$.ajax({  
  url: '/tweets/pagina/5',  
  dataType: 'json',  
  success: resposta => {  
    let novosTweets = resposta.arrayComNovosTweets;  
    novosTweets.forEach(colocaTweetNaPagina);  
  }  
});
```

Existe \$.getJSON({...}), equivalente a \$.ajax({...}) com o dataType: 'json'



# Ajax com Fetch

- Mais recentemente, foi introduzida nova forma: `fetch(...)`
- Ele envia uma requisição Ajax e retorna uma Promessa

## Formato:

```
fetch(url, opcoes)
  .then(callbackSucesso)
  .catch(callbackErro);
```

## Exemplo:

```
fetch('http://example.com/movies.json')
  .then(response => response.json())
  .then(data => console.log(data));
```

Busca um arquivo JSON na rede e imprime no console. No exemplo possuí um argumento (`response`) e não retorna diretamente o corpo da resposta JSON, mas a promessa que resolve com um objeto do tipo `Response`.

Exemplos: [https://developer.mozilla.org/en-US/docs/Web/API/Fetch\\_API/Using\\_Fetch](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)

# Ajax com Fetch

`fetch(url, opcoes)` retorna uma promessa

- url é para onde a requisição será enviada
- opcoes serão mostradas no próximo slide
- A promessa retornada é resolvida como uma Response, que pode ser lida de diferentes formas:
  - `response.json()`
  - `response.text()`
  - `response.arrayBuffer()`
  - `response.blob()`
  - `response.formData()`

# API Star Wars

Podemos usar as API públicas para retornar algum tipo informações, como por exemplo a API Star Wars, disponível em <https://swapi.dev/>

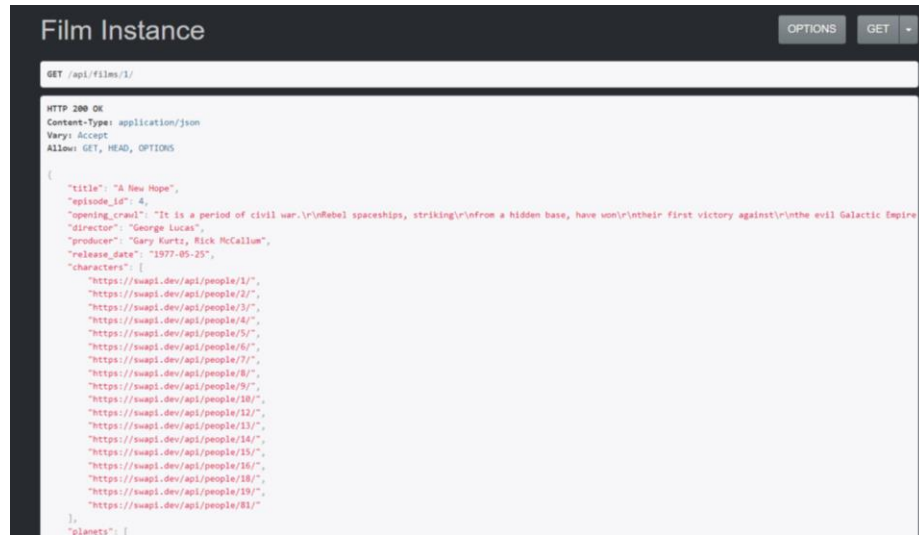
Exemplo:

<https://swapi.dev/api/films/1/>

## Documentação da API

<https://swapi.dev/documentation>

<https://www.npmjs.com/package/swapi-node>



The screenshot shows a web browser displaying the response for the GET request to `/api/films/1/`. The response is a JSON object representing the first Star Wars film, 'A New Hope'. The JSON includes fields for title, episode ID, opening crawl, director, producer, release date, characters, and planets. The characters array lists 19 individual characters with their respective API endpoints. The planets array is currently empty.

```
GET /api/films/1/

HTTP 200 OK
Content-Type: application/json
Vary: Accept
Allow: GET, HEAD, OPTIONS

{
  "title": "A New Hope",
  "episode_id": 4,
  "opening_crawl": "It is a period of civil war. Rebel spaceships, striking from a hidden base, have won their first victory against the evil Galactic Empire.",
  "director": "George Lucas",
  "producer": "Gary Kurtz, Rick McCallum",
  "release_date": "1977-05-25",
  "characters": [
    "https://swapi.dev/api/people/1/",
    "https://swapi.dev/api/people/2/",
    "https://swapi.dev/api/people/3/",
    "https://swapi.dev/api/people/4/",
    "https://swapi.dev/api/people/5/",
    "https://swapi.dev/api/people/6/",
    "https://swapi.dev/api/people/7/",
    "https://swapi.dev/api/people/8/",
    "https://swapi.dev/api/people/9/",
    "https://swapi.dev/api/people/10/",
    "https://swapi.dev/api/people/12/",
    "https://swapi.dev/api/people/13/",
    "https://swapi.dev/api/people/14/",
    "https://swapi.dev/api/people/15/",
    "https://swapi.dev/api/people/16/",
    "https://swapi.dev/api/people/18/",
    "https://swapi.dev/api/people/19/",
    "https://swapi.dev/api/people/81/"
  ],
  "planets": [
  ]
}
```