Trabalho: Criação de uma aplicação Front-End para uma API de desenhos de sucesso dos anos 90 e 2000

Aluna: Fernanda Messa Menegassi
Disciplina: Web Programming for Front-End
Instituição: UNIFECAF

1. O que é uma API e como funciona o consumo de dados

Uma API (Interface de Programação de Aplicações) é basicamente um conjunto de regras e protocolos que permite a comunicação entre diferentes sistemas.
No contexto web, as APIs REST fornecem endpoints HTTP que retornam dados estruturados, geralmente em formato JSON.

O consumo de dados em tempo real ocorre quando a aplicação faz requisições a esses endpoints sob demanda — por exemplo, ao carregar a página, ao pesquisar ou ao solicitar mais registros — e atualiza de forma dinâmica a interface com as informações recebidas, sem precisar recarregar toda a página.

2. Conceito de manipulação do DOM e criação de elementos dinâmicos

O DOM (Document Object Model) é a representação em árvore dos elementos HTML de uma página.
A manipulação do DOM permite ao JavaScript acessar, criar, modificar e remover elementos durante a execução do código.

No projeto, os cards dos personagens são gerados dinamicamente: o script recebe os objetos da API e, para cada item, cria elementos (<article>, <img>, <h3>, <p>) com document.createElement(), define seu conteúdo e atributos (por exemplo, textContent e src), e os insere no container principal usando appendChild() ou append().

3. Principais funções utilizadas

- **fetch(url):** realiza requisições HTTP ao endpoint indicado.
- **response.json():** converte a resposta em formato JSON (objeto JavaScript).
- **async / await** ou **then():** tratam o fluxo assíncrono das *Promises*.
- **document.createElement():** cria elementos HTML dinamicamente.
- **element.appendChild() / element.append():** insere elementos no DOM.
- **addEventListener():** faz o registro de ouvintes de eventos (busca, clique em “carregar mais”, etc.).

Essas funções possibilitaram o carregamento inicial dos itens, a paginação (função “carregar mais”), a busca por nome e a criação automática dos cards a partir dos dados da API.

4. Motivo da escolha da API

A API selecionada foi a PokéAPI (https://pokeapi.co/).

Motivos da escolha:

- É pública, gratuita e não exige cadastro ou token de autenticação;
- É estável e amplamente documentada;
- Fornece imagens (sprites e official artwork) e informações úteis (tipos, peso, altura, etc.);
- A franquia Pokémon foi um dos maiores sucessos dos anos 90 e 2000, atendendo ao tema proposto pelo trabalho.

A PokéAPI permite construir cards com imagem, nome e atributos.

5. Regras de acesso e estrutura de resposta da API

- **Endpoint base:** https://pokeapi.co/api/v2/pokemon
- **Paginação:** A API utiliza paginação por meio dos parâmetros limit (quantos itens serão retornados) e offset (a partir de qual posição começar).
- Exemplo: ?offset=0&limit=30 retorna os 30 primeiros registros.
- **Fluxo de uso no projeto:**
    1. Requisição inicial à listagem de Pokémon → campo results (com name e url);
    2. Nova requisição para cada url → retorna dados detalhados (sprites, types, weight, etc.).
- **Autenticação:** não é necessária.

Fontes consultadas:

- Documentação oficial da PokéAPI (https://pokeapi.co/docs/v2)
- Referências MDN sobre fetch e manipulação do DOM

6. Conclusão

Esse projeto demonstrou a integração de uma aplicação Front-End com uma API pública, utilizando exclusivamente HTML, CSS e JavaScript.
Implementei requisições via fetch, tratamento assíncrono, criação dinâmica de elementos via DOM e funcionalidades de busca e paginação.

Faz parte da entrega o código-fonte (HTML/CSS/JS), que seria a parte prática, o relatório teórico (este documento) e o vídeo demonstrativo.

Referências

- MDN — Using Fetch
- MDN — Document.createElement
- MDN — Node.appendChild / Document.append
- PokéAPI — Documentação oficial