Documento Teórico - Aplicação Front-End Rick and Morty

1. Introdução e Escolha da API

Este projeto consiste no desenvolvimento de uma aplicação *Front-End* estática, utilizando **HTML**, **CSS** e **JavaScript** puro, com o objetivo de consumir dados de uma API pública e exibir informações de forma dinâmica.

A API escolhida para este trabalho foi a **Rick and Morty API** [1].

Motivo da Escolha da API

A Rick and Morty API foi selecionada por ser uma **API RESTful** pública, gratuita e de fácil acesso, que não exige chaves de autenticação (*API Keys*) para requisições básicas de leitura. Sua documentação é clara e o tema (desenho animado de sucesso) se alinha perfeitamente com o requisito do projeto.

Regras de Acesso e Estrutura de Resposta da API

A API é acessada através do endpoint base `https://rickandmortyapi.com/api/`. Para buscar personagens, utiliza-se o endpoint `/character`.

- **Regras de Acesso:** Não há necessidade de token de acesso. O limite de requisições é alto o suficiente para uso em projetos de desenvolvimento e aprendizado.
- **Estrutura de Resposta:** A resposta da API é um objeto JSON que contém dois campos principais:
    - `info`: Metadados sobre a requisição (total de resultados, total de páginas, links para próxima e página anterior).
    - `results`: Um array contendo os objetos dos personagens. Cada objeto de personagem inclui campos como `id`, `name`, `status`, `species`, `gender`,

`origin`, `location` e `image` (URL da imagem).

2. O que é uma API e Consumo de Dados em Tempo Real

O que é uma API?

API (*Application Programming Interface*) é um conjunto de definições e protocolos que permite que diferentes softwares se comuniquem entre si. No contexto da web, uma API geralmente se refere a um sistema que permite que um cliente (como nosso site *Front-End*) solicite e receba dados de um servidor.

> *"Uma API é um intermediário de software que permite que dois aplicativos se comuniquem. Em termos simples, é um mensageiro que entrega sua solicitação ao provedor e, em seguida, entrega a resposta de volta para você." [2]*

Consumo de Dados em Tempo Real

O consumo de dados em tempo real, neste contexto, refere-se à capacidade da aplicação de buscar informações atualizadas do servidor no momento em que o usuário interage com a página (por exemplo, ao carregar a página ou mudar de página na paginação). Isso é feito através de requisições **HTTP** (como `GET`), que solicitam os dados e os injetam na interface do usuário.

3. Manipulação do DOM e Criação de Elementos Dinâmicos

Conceito de DOM Manipulation

O **DOM** (*Document Object Model*) é uma interface de programação para documentos HTML e XML. Ele representa a estrutura da página como uma árvore de objetos, onde cada nó é um elemento, atributo ou texto. A **Manipulação do DOM** é o processo de usar JavaScript para alterar essa estrutura, modificando o conteúdo, estilo ou a própria estrutura da página após ela ter sido carregada.

Criação de Elementos Dinâmicos via JavaScript

Para atender ao requisito de criar os *cards* de personagens dinamicamente, o JavaScript é utilizado para construir os elementos HTML do zero, com base nos dados recebidos da API.

O processo básico envolve:

1. **Criação:** Utilizar `document.createElement('tag')` para criar um novo nó HTML (ex: `<div>`, `<img>`, `<h3>`).
2. **Configuração:** Definir atributos e conteúdo (ex: `element.className = 'classe'`, `element.textContent = 'texto'`, `element.src = 'url'`).
3. **Injeção:** Utilizar `parentElement.appendChild(childElement)` para inserir o novo elemento na árvore do DOM, tornando-o visível na página.

4. Funções Básicas Utilizadas para Consumo da API

O consumo da API e a manipulação do DOM são realizados por meio de funções nativas do JavaScript:

Função	Categoria	Propósito
`fetch()`	Consumo de API	Inicia uma requisição HTTP para buscar um recurso (os dados da API). Retorna uma `Promise` que resolve para o objeto `Response`.
`.then()`	Consumo de API	É encadeado após o `fetch` para lidar com o resultado da `Promise`. É usado para processar a resposta (ex: converter para JSON) e, em seguida, para manipular os dados.
`async/await`	Consumo de API	Sintaxe moderna para trabalhar com `Promises`, tornando o código assíncrono mais legível e sequencial, substituindo o encadeamento excessivo de `.then()`.
`document.createElement()`	DOM Manipulation	Cria um novo nó de elemento HTML especificado por seu nome de tag.
`element.appendChild()`	DOM Manipulation	Adiciona um nó (elemento) como o último filho de um nó pai especificado.
`element.textContent`	DOM Manipulation	Define ou retorna o conteúdo de texto de um elemento.
`element.className`	DOM Manipulation	Define ou retorna o valor do atributo `class` de um elemento.

O uso dessas funções garante que a aplicação seja leve, rápida e cumpra o requisito de ser desenvolvida com **JavaScript puro**.

Referências

[1] Rick and Morty API. Disponível em: https://rickandmortyapi.com/ [2] O que é uma API? Red Hat. Disponível em: https://www.redhat.com/pt/topics/api/what-is-an-api [3] Usando a Fetch API. MDN Web Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/Fetch_API/Using_Fetch [4] Document.createElement(). MDN Web

Docs. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/API/Document/createElement