ApiService.js: Consumo de APIs REST

Prof. Me Hélio Esperidião

Classe para consumo de APIs REST

- Abstrai chamadas HTTP comuns (GET, POST, PUT, DELETE)
- Suporte a token de autenticação (Bearer Token)
- Facilita comunicação com APIs RESTful

Atributo privado e construtor

- #token é privado e usado para autenticação
- Construtor recebe token opcional para uso em headers

Atributo privado e construtor

```
export default class ApiService {

    #token;  // Atributo privado

    constructor(token = null) {

        this.#token = token;

  }

    // Getter padrão
```

Método simpleGet

- Método simpleGet(uri)
  - Faz um fetch simples (GET) sem headers personalizados
  - Retorna JSON convertido
  - Útil para APIs públicas sem autenticação

```
/**
 * Método para fazer uma requisição GET simples sem headers adicionais.
 * Útil para APIs públicas que não requerem autenticação.
 * @param {string} uri - URL do recurso para a requisição GET.
 * @returns {Promise<Object|Array>} Retorna o JSON obtido da resposta ou array vazio em caso de erro.
 */
async simpleGet(uri) {
    try {
        const response = await fetch(uri);            // Faz a requisição HTTP GET
        const jsonObj = await response.json();        // Converte a resposta para JSON
        console.log("GET:", uri, jsonObj);             // Log para depuração
        return jsonObj;                                // Retorna o JSON obtido

      } catch (error) {
        console.error("Erro ao buscar dados:", error.message);  // Exibe erro no console
        return [];                                     // Retorna array vazio em caso de erro
    }
}
```

Método get

- Método get(uri)
  - GET com headers padrão (Content-Type: application/json)
  - Inclui token no header Authorization se disponível
  - Tratamento de erros e retorno do JSON da resposta

```
    async get(uri) {
      try {
          // Configura headers padrão para JSON
          const headers = {
              "Content-Type": "application/json"
          };

             // Adiciona header Authorization se token estiver configurado
          if (this.#token) {
              headers["Authorization"] = `Bearer ${this.#token}`;
          }
             // Faz requisição GET com headers configurados
          const response = await fetch(uri, {
              method: "GET",
              headers: headers
          });

             const jsonObj = await response.json();   // Converte resposta para JSON
          console.log("GET:", uri, jsonObj);       // Log para depuração
          return jsonObj;                           // Retorna JSON da resposta

        } catch (error) {
          console.error("Erro ao buscar dados:", error.message);
          return [];                                // Retorna array vazio em caso de erro
      }
  }
```

Método getById

- Método getById(uri, id)
  - GET para URI com parâmetro id anexado
  - Exemplo: api/recursos/123
  - Verifica status HTTP, lança erro se não OK
  - Retorna JSON do recurso específico

```
async getById(uri, id) {
        try {
            const headers = {
                "Content-Type": "application/json"
            };
            if (this.#token) {
                headers["Authorization"] = `Bearer ${this.#token}`;
            }
            // Concatena URI com ID para buscar recurso específico
            const fullUri = `${uri}/${id}`;
            const response = await fetch(fullUri, {
                method: "GET",
                headers: headers
            });
            // Verifica se a resposta HTTP foi bem sucedida (status 2xx)
            if (!response.ok) {
                throw new Error(`Erro HTTP: ${response.status}`);
            }
            const jsonObj = await response.json();
            console.log("GET BY ID:", fullUri, jsonObj);
            return jsonObj;

           } catch (error) {
            console.error("Erro ao buscar por ID:", error.message);
            return null;  // Retorna null se houve erro na requisição
        }
    }
```

Método post

- Método post(uri, jsonObject)
  - POST com corpo JSON serializado
  - Headers padrão + token no header se presente
  - Retorna JSON da resposta (ex: objeto criado)
  - Tratamento de erros com retorno seguro

```
async post(uri, jsonObject) {
       try {
           const headers = {
               "Content-Type": "application/json"
           };

              if (this.#token) {
               headers["Authorization"] = `Bearer ${this.#token}`;
           }

              // Executa a requisição POST com headers e corpo JSON
           const response = await fetch(uri, {
               method: "POST",
               headers: headers,
               body: JSON.stringify(jsonObject)
           });

              const jsonObj = await response.json();
           console.log("POST:", uri, jsonObj);
           return jsonObj;

          } catch (error) {
           console.error("Erro ao buscar dados:", error.message);
           return [];  // Retorna array vazio em caso de erro
       }
   }
```

Método put

- Método put(uri, id, jsonObject)
  - PUT para atualizar recurso com id
  - URI formada como uri/id
  - Envia corpo JSON com dados atualizados
  - Retorna resposta JSON da API

```
    async put(uri, id, jsonObject) {
        try {
            const headers = {
                "Content-Type": "application/json"
            };
            if (this.#token) {
                headers["Authorization"] = `Bearer ${this.#token}`;
            }
               // Monta URL final com ID
            const fullUri = `${uri}/${id}`;
               // Faz requisição PUT com corpo JSON
            const response = await fetch(fullUri, {
                method: "PUT",
                headers: headers,
                body: JSON.stringify(jsonObject)
            });
            const jsonObj = await response.json();
            console.log("PUT:", fullUri, jsonObj);
            return jsonObj;
           } catch (error) {
            console.error("Erro ao enviar dados:", error.message);
            return null;  // Retorna null em caso de erro
        }
    }
```

Método delete

- Método delete(uri, id)
  - DELETE para recurso específico uri/id
  - Inclui headers e token, se presente
  - Verifica resposta HTTP (status)
  - Retorna JSON ou null, caso a resposta não tenha corpo

```
    async delete(uri, id) {
        try {
            const headers = {
                "Content-Type": "application/json"
            };
               if (this.#token) {
                headers["Authorization"] = `Bearer ${this.#token}`;
            }
               // Monta URL final com ID
            const fullUri = `${uri}/${id}`;
               // Executa requisição DELETE
            const response = await fetch(fullUri, {
                method: "DELETE",
                headers: headers
            });
               // Verifica sucesso da resposta
            if (!response.ok) {
                throw new Error(`Erro HTTP: ${response.status}`);
            }
               // Tenta converter resposta para JSON, mas se falhar retorna null
            const jsonObj = await response.json().catch(() => null);
            console.log("DELETE:", fullUri, jsonObj);
            return jsonObj;

           } catch (error) {
            console.error("Erro ao deletar dados:", error.message);
            return null;  // Retorna null em caso de erro
        }
    }
```

Getters e Setters

- Permite alterar token após a criação da instância
- Facilita atualização de credenciais

```
    /**
     * Getter para o token privado.
     * @returns {string|null} Retorna o token atual.
     */
    get token() {
        return this.#token;
    }

       /**
     * Setter para atualizar o token privado.
     * @param {string} value - Novo token a ser setado.
     */
    set token(value) {
        this.#token = value;
    }
```

Aplicação

```
<script type="module">
  import ApiService from './ApiService.js';  // importa a classe
   document.addEventListener("DOMContentLoaded", () => {
    const btnClick = document.getElementById("btnClick1");
    const divResposta = document.getElementById("divResposta");
    const uri = "https://api.adviceslip.com/advice";
    const api = new ApiService();
    btnClick.addEventListener("click", async () => {
      const objJson = await api.simpleGet(uri);

        if (objJson && objJson.slip) {
        const { id, advice } = objJson.slip;
        divResposta.innerHTML += `
          <p><strong>ID:</strong> ${id} <strong>ADVICE:</strong> ${advice}</p>
        `;
      } else {
        divResposta.textContent = "Não foi possível obter o conselho.";
      }
    });
  });
</script>
```