# Operações Assíncronas (modo antigo: call-back-hell)

Em cliente-service.js:

http.onload = function() {

    // Pegar o resultado que servidor me devolveu. Precisamos usar o método parse, pois o http.response retorna

    // o resultado (dados dos clientes) em formato string. O parse transforma essa string num objeto interpretável pelo JavaScript.

    const data = JSON.parse(http.response)

    data.forEach(element => {

        tabela.appendChild(criaNovaLinha(element.nome, element.email))

    });

    // Mostrar dados mais antigos (Sequência de operações assíncronas). CALLBACK

    const http2 = new  XMLHttpRequest()

    http2.open('GET', 'http://localhost:3000/profile/semanaPassada')

    http2.onload = function() {

        // código omitido...

        const http3 = new  XMLHttpRequest()

        http3.open('GET', 'http://localhost:3000/profile/mesPassado')

        http3.onload = function() {

            // código omitido...

        }

        http3.send()

        // e assim sucessivamente (http4, http5...)

    }

    http2.send()

}

Esse método foi substituído pelo método “Promises”.

# Promises

Em cliente-service.js:

const criaNovaLinha = function(nome, email) {

// Os dados (nome, email) vêm de 'data' em http.onload, que é um array que contém os dados dos clientes.

    const linhaNovoCliente = document.createElement('tr')

    // Uma Template Literal precisa estar entre crases.

    const conteudo = `

        <td class="td" data-td>${nome}</td>

        <td>${email}</td>

        <td>

            <ul class="tabela\_\_botoes-controle">

                <li><a href="../telas/edita\_cliente.html" class="botao-simples botao-simples--editar">Editar</a></li>

                <li><button class="botao-simples botao-simples--excluir" type="button">Excluir</button></li>

            </ul>

        </td>

    `

    linhaNovoCliente.innerHTML = conteudo

    return linhaNovoCliente

}

// <tbody data-tabela>

const tabela = document.querySelector('[data-tabela]')

const listaClientes = function() {

    const promise = new Promise((resolve, reject) => {

        // Objeto XMLHTTPRequest: faz a comunicação entre a minha aplicação e o servidor que contém os dados.

        const http = new  XMLHttpRequest()

        // Abrir a comunicação entre a minha aplicação e a API (Método Open).

        // Primeiro argumento: ação a ser executada pelo servidor (“pegar”).

        // Segundo argumento: o alvo da ação.

        http.open('GET', 'http://localhost:3000/profile')

        // O que fazer com a resposta que o servidor vai enviar de volta.

        // Ao ler, executar a função. CUIDADO: não estamos declarando uma variável (var, let, const).

        http.onload = function() {

            // A partir de 400, são erros relacionados a cliente-servidor.

            // Precisamos usar o método parse, pois o http.response retorna o resultado (dados dos clientes)

            // em formato string.

            if(http.status >= 400) {

                reject(JSON.parse(http.response))

            }else {

                resolve(JSON.parse(http.response))

            }

        }

        // Enviar a requisição

        http.send()

    })

    // console.log(promise) // para testar se o promise está sendo retornado. Não se esqueça depois de chamar

    // listaCliente() para fazer o teste.

    return promise

}

listaClientes()

.then(data => {

// Quando a Promise tem sucesso e é resolvida, resolve(JSON.parse(http.response)) é executado,

// então o valor da expressão JSON.parse(http.response) é passado para a próxima função, que será a .then (esta em que estamos).

// Então, o data dentro do .then nada mais é do que o JSON.parse(http.response).

    data.forEach(element => {

        tabela.appendChild(criaNovaLinha(element.nome, element.email))

    })

})