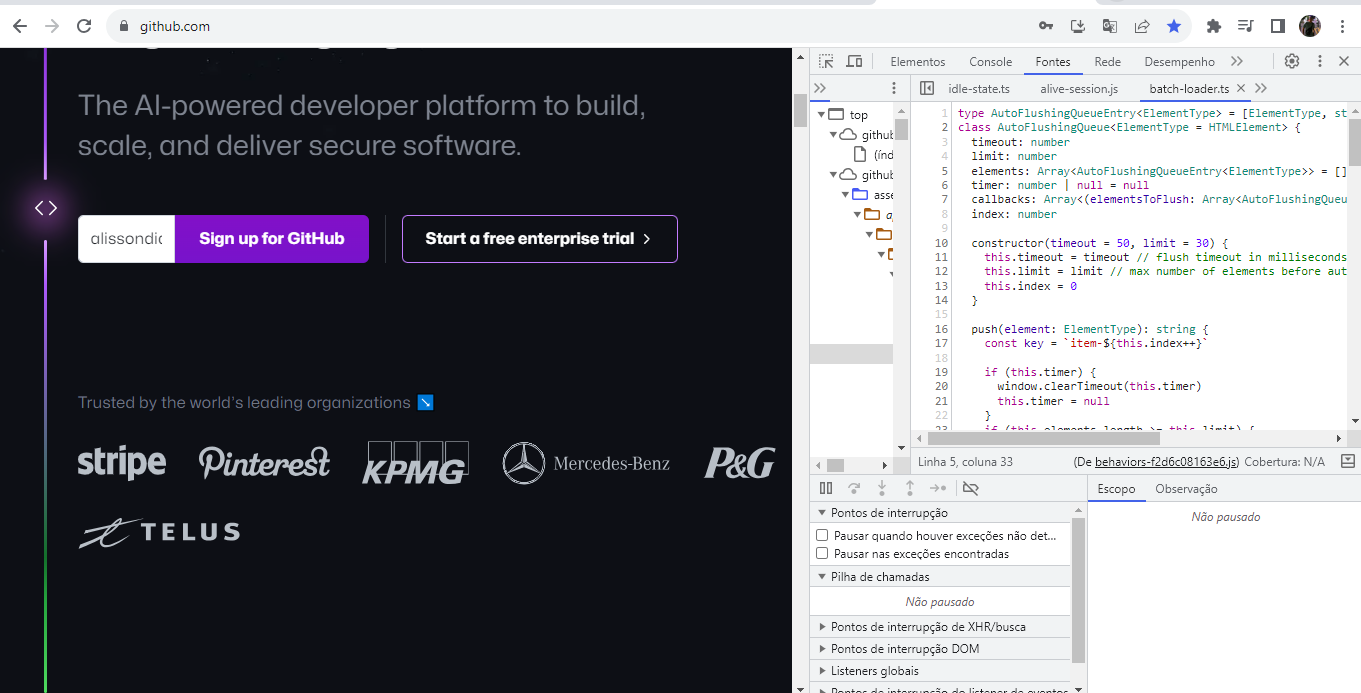
**Atividade Inspeção de Sites**

**1° Site (**[**https://github.com/**](https://github.com/)**)**

* **Página de login GitHub, onde é possível acessá-lo com seu usuário e senha.**

**figura 1.1 (GitHub)**



type AutoFlushingQueueEntry<ElementType> = [ElementType, string]

class AutoFlushingQueue<ElementType = HTMLElement> {

timeout: number

limit: number

elements: Array<AutoFlushingQueueEntry<ElementType>> = []

timer: number | null = null

callbacks: Array<(elementsToFlush: Array<AutoFlushingQueueEntry<ElementType>>) => Promise<void>> = []

index: number

constructor(timeout = 50, limit = 30) {

this.timeout = timeout // flush timeout in milliseconds

this.limit = limit // max number of elements before autoflushing

this.index = 0

}

push(element: ElementType): string {

const key = `item-${this.index++}`

if (this.timer) {

window.clearTimeout(this.timer)

this.timer = null

}

if (this.elements.length >= this.limit) {

this.flush()

}

this.timer = window.setTimeout(() => {

this.timer = null

this.flush()

}, this.timeout)

this.elements.push([element, key])

return key

}

onFlush(callback: (elementsToFlush: Array<AutoFlushingQueueEntry<ElementType>>) => Promise<void>): void {

this.callbacks.push(callback)

}

private async flush(): Promise<void> {

const elementsToFlush = this.elements.splice(0, this.limit)

if (elementsToFlush.length === 0) return

await Promise.all(this.callbacks.map(callback => callback(elementsToFlush)))

}

}

interface HTMLElementWithInputs extends HTMLElement {

inputs: HTMLInputElement[]

}

export class BatchLoader<ResponseType> {

autoFlushingQueue: AutoFlushingQueue<HTMLElementWithInputs>

url: string

callbacks: Map<string, (value: ResponseType) => void>

validate: (value: unknown) => asserts value is ResponseType

constructor(url: string, validator: (value: unknown) => asserts value is ResponseType) {

this.url = url

this.callbacks = new Map()

this.autoFlushingQueue = new AutoFlushingQueue()

this.autoFlushingQueue.onFlush(async elements => {

this.load(elements)

})

this.validate = validator

}

loadInBatch(element: HTMLElementWithInputs): Promise<ResponseType> {

const key = this.autoFlushingQueue.push(element)

return new Promise(resolve => this.callbacks.set(key, resolve))

}

async load(elements: Array<AutoFlushingQueueEntry<HTMLElementWithInputs>>) {

const elementsByKey = new Map<string, HTMLElementWithInputs>()

for (const [batchElement, key] of elements) {

elementsByKey.set(key, batchElement)

}

const consolidatedData = new FormData()

for (const [key, batchElement] of elementsByKey.entries()) {

for (const input of batchElement.inputs) {

consolidatedData.append(`items[${key}][${input.name}]`, input.value)

}

}

if (Array.from(consolidatedData.values()).length === 0) {

return

}

consolidatedData.set('\_method', 'GET')

const response = await fetch(this.url, {

method: 'POST',

body: consolidatedData,

headers: {

Accept: 'application/json',

'X-Requested-With': 'XMLHttpRequest',

},

})

if (response.ok) {

const json = await response.json()

if (!json || typeof json !== 'object' || Array.isArray(json)) {

throw new Error('Malformed batch response')

}

for (const key in json) {

const callback = this.callbacks.get(key)

if (callback) {

const value = json[key]

this.validate(value)

callback(value)

}

}

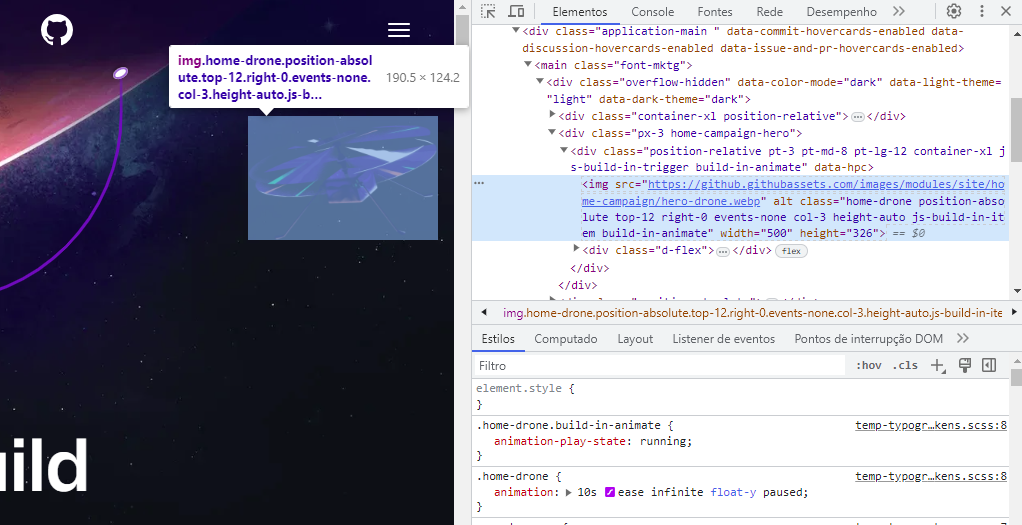
}

}

}

**Verifiquei uma parte do código em Javascript do site, nele conseguimos verificar inúmeros elementos do Javascript, como a declaração de classe, construtores, estrutura de condição, array, entre outros.**

**figura 1.2 (GitHub)**

****

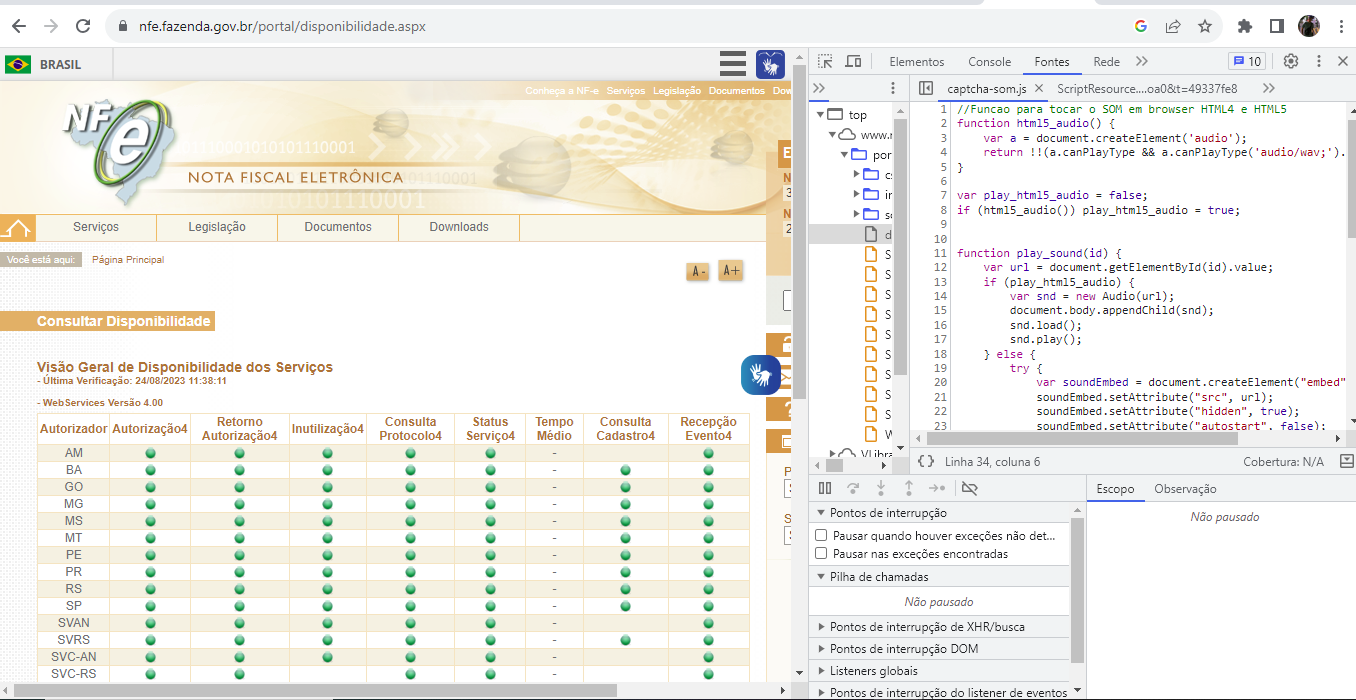
<img src="https://github.githubassets.com/images/modules/site/home-campaign/hero-drone.webp" alt="" class="home-drone position-absolute top-12 right-0 events-none col-3 height-auto js-build-in-item build-in-animate" width="500" height="326">

**Também temos esse parte do código em HTML e CSS que inserem e posicionam a foto da figura no site, ajustando o posicionamento, estilo e comportamentos associados.**

**2° Site (**[**https://www.nfe.fazenda.gov.br/portal/disponibilidade.aspx**](https://www.nfe.fazenda.gov.br/portal/disponibilidade.aspx)**)**

* **Site onde é consultado a disponibilidade da Secretaria da Fazenda (SEFAZ) para emissões e outros processos relacionados a notas fiscais.**

**figura 2.1 (Sefaz)**

****

//Funcao para tocar o SOM em browser HTML4 e HTML5

function html5\_audio() {

var a = document.createElement('audio');

return !!(a.canPlayType && a.canPlayType('audio/wav;').replace(/no/, ''));

}

var play\_html5\_audio = false;

if (html5\_audio()) play\_html5\_audio = true;

function play\_sound(id) {

var url = document.getElementById(id).value;

if (play\_html5\_audio) {

var snd = new Audio(url);

document.body.appendChild(snd);

snd.load();

snd.play();

} else {

try {

var soundEmbed = document.createElement("embed");

soundEmbed.setAttribute("src", url);

soundEmbed.setAttribute("hidden", true);

soundEmbed.setAttribute("autostart", false);

soundEmbed.setAttribute("width", 0);

soundEmbed.setAttribute("height", 0);

soundEmbed.setAttribute("enablejavascript", true);

soundEmbed.setAttribute("autostart", true);

document.body.appendChild(soundEmbed);

}

catch (e) {

alert('errou');

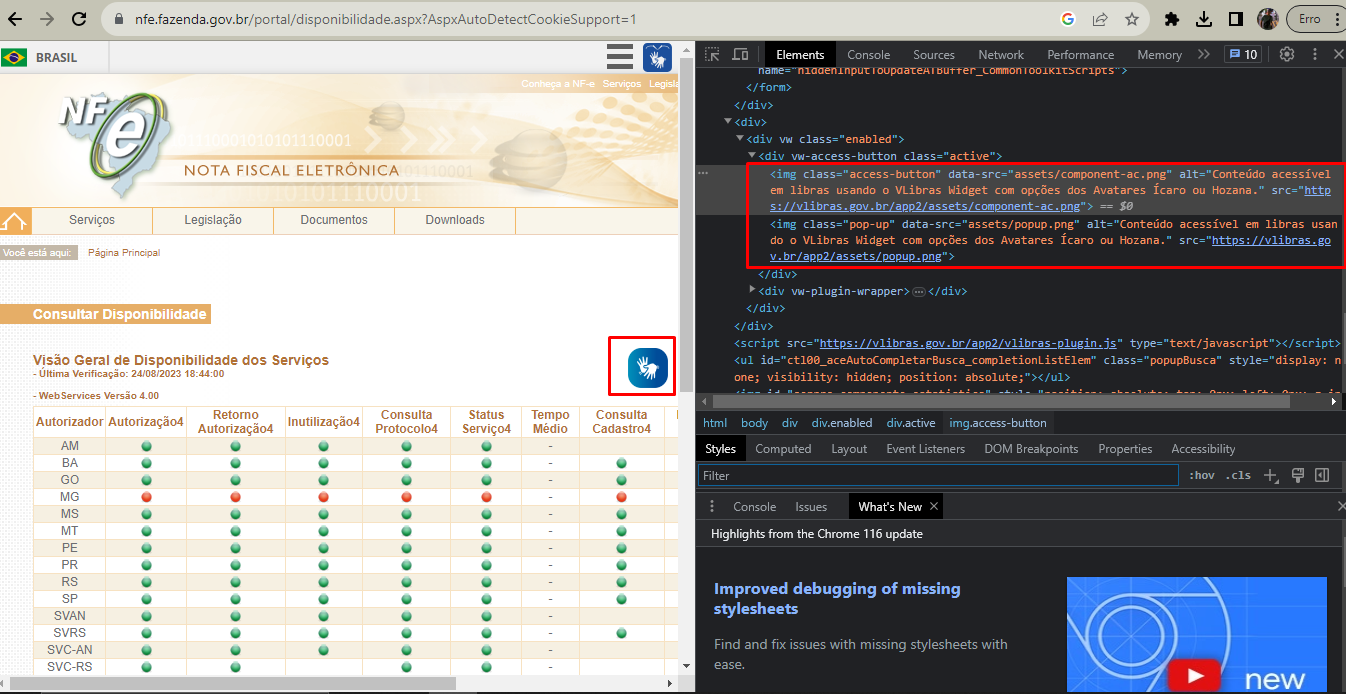
}

}

}

**Vemos neste código Javascript funções e verificações com estrutura de condição, também tendo um alert no código para criar um alerta.**

**figura 2.2 (Sefaz)**

****

<img class="access-button" data-src="assets/component-ac.png" alt="Conteúdo acessível em libras usando o VLibras Widget com opções dos Avatares Ícaro ou Hozana." src="<https://vlibras.gov.br/app2/assets/component-ac.png>">

<img class="pop-up" data-src="assets/popup.png" alt="Conteúdo acessível em libras usando o VLibras Widget com opções dos Avatares Ícaro ou Hozana." src="https://vlibras.gov.br/app2/assets/popup.png">

**Nessa parte do código temos um botão que através dele podemos abrir uma tela para fornecer a tradução em Libras do conteúdo textual da página, sendo traduzido para quem necessita do Vlibras que é uma ferramenta de acessibilidade.**