

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS TOLEDO
COTSI - CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

LUCAS PEREIRA MACHADO

**FERRAMENTAS DE DESENVOLVIMENTO WEB: UMA
DOCUMENTAÇÃO TÉCNICA**

PROPOSTA DE TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
15 DE DEZEMBRO DE 2025

LUCAS PEREIRA MACHADO

FERRAMENTAS DE DESENVOLVIMENTO WEB: UMA DOCUMENTAÇÃO TÉCNICA

Proposta de Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 1, do COTSI - Curso de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná - UTFPR - Câmpus Toledo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Mr. Eduardo Pezutti Beletato dos Santos

TOLEDO
15 DE DEZEMBRO DE 2025

Agradecimentos

Agradeço primeiramente a Deus, por me conceder saúde, sabedoria e perseverança para a desenvolver este trabalho e por sua infinita bondade.

Agradeço a minha família, pelo amor, apoio e incentivo para que eu pudesse alcançar mais esta etapa em minha vida.

Agradeço ao meu orientador, Prof. Mr. Eduardo Pezutti B. Dos Santos, pela paciência, ensinamentos e auxilio durante todo o desenvolvimento deste trabalho.

Agradeço também aos meus amigos e colegas, que de modo presencial ou online, se fizeram presentes nesta etapa, e contribuíram de diversas formas para a realização deste trabalho.

Por fim, agradeço a todos que, de forma direta ou indireta, me ajudaram a concluir esta etapa importante.

Não tenhais medo das novas tecnologias! Elas incluem-se "entre as coisas maravilhosas" que Deus pôs à nossa disposição para as descobrirmos, usarmos, fazer conhecer a verdade, também a verdade acerca do nosso destino de filhos seus, e herdeiros do seu Reino eterno.

— Papa São João Paulo II

RESUMO

As ferramentas de desenvolvimento (devtools) integradas aos navegadores modernos são essenciais para a criação e manutenção de aplicações web. Este estudo realiza uma análise exploratória das devtools dos principais navegadores, incluindo Chrome, Firefox, Edge e Safari, com o objetivo de compreender suas funcionalidades, interfaces e aplicações práticas. A pesquisa investiga como os desenvolvedores utilizam essas ferramentas, identificando práticas comuns, desafios e necessidades não atendidas. Além disso, avalia-se o impacto das devtools na produtividade, na depuração de erros e na qualidade do desenvolvimento web. Espera-se que os resultados contribuam para o aprimoramento do uso dessas ferramentas, tornando o processo de desenvolvimento mais eficiente.

Palavras-chave: Ferramentas de desenvolvimento. Navegador web. Experiência do desenvolvedor.

ABSTRACT

The developer tools (devtools) integrated into modern browsers are essential for building and maintaining web applications. This study conducts an exploratory analysis of the devtools in major browser, including Chrome, Firefox, Edge, and Safari, aiming to understand their functionalities, interfaces, and practical applications. The research investigates how developers utilize these tools, identifying common practices, challenges, and unmet needs. Furthermore, the study evaluates the impact of devtools on productivity, debugging, and overall web development quality. The findings are expected to enhance the use of these tools, making the development process more efficient.

Palavras-chave: developer tools, web browser, developer experience.

LISTA DE FIGURAS

Figura 1 – Chrome Devtools	3
Figura 2 – Firefox Developer Tools	3
Figura 3 – Edge DevTools	3
Figura 4 – Ferramenta "Elementos"do Chrome Devtools	5
Figura 5 – Ferramenta "Console"do Chrome Devtools	6
Figura 6 – Ferramenta "Problemas"do Chrome Devtools	6
Figura 7 – Ferramenta "Lighthouse"do Chrome Devtools	7
Figura 8 – Ferramenta "Memória"do Chrome Devtools	7
Figura 9 – Ferramenta "Rede"do Chrome Devtools	8
Figura 10 – Ferramenta "Desempenho"do Chrome Devtools	9
Figura 11 – Ferramenta "Gravador"do Chrome Devtools	9
Figura 12 – Ferramenta "Renderização"do Chrome Devtools	10
Figura 13 – Ferramenta "Segurança"do Chrome Devtools	10
Figura 14 – Ferramenta "Fontes"do Chrome Devtools	11
Figura 15 – Ferramenta "Autofill"do Chrome Devtools	11
Figura 16 – Ferramenta "Animações"do Chrome Devtools	12
Figura 17 – Ferramenta "Aplicativo"do Chrome Devtools	12
Figura 18 – Ferramenta "Alterações"do Chrome Devtools	13
Figura 19 – Ferramenta "Cobertura"do Chrome Devtools	13
Figura 20 – Ferramenta "Visão geral de CSS"do Chrome Devtools	15
Figura 21 – Ferramenta "Recursos para desenvolvedores"do Chrome Devtools	16
Figura 22 – Ferramenta "Camadas"do Chrome Devtools	16
Figura 23 – Ferramenta "Mídia"do Chrome Devtools	17
Figura 24 – Ferramenta "Inspetor de memória"do Chrome Devtools	17
Figura 25 – Ferramenta "Condições de rede"do Chrome Devtools	18
Figura 26 – Ferramenta "Bloqueio de solicitações de rede"do Chrome Devtools	18
Figura 27 – Ferramenta "Monitor de Desempenho"do Chrome Devtools	19
Figura 28 – Ferramenta "Monitor de protocolo"do Chrome Devtools	19
Figura 29 – Ferramenta "Origem rápida"do Chrome Devtools	21
Figura 30 – Ferramenta "WebAudio"do Chrome Devtools	22
Figura 31 – Ferramenta "Sensores"do Chrome Devtools	22
Figura 32 – Ferramenta "WebAuthn"do Chrome Devtools	23
Figura 33 – Ferramenta "Inspetor"do Firefox Devtools	23
Figura 34 – Ferramenta "Console"do Firefox Devtools	24
Figura 35 – Ferramenta "Depurador"do Firefox Devtools	24
Figura 36 – Ferramenta "Rede"do Firefox Devtools	25
Figura 37 – Ferramenta "Desempenho"do Firefox Devtools	25
Figura 38 – Ferramenta "Inspetor de Acessibilidade"do Firefox Devtools	26
Figura 39 – Ferramenta "Aplicações"do Firefox Devtools	26
Figura 40 – Ferramenta "Memória"do Firefox Devtools	26
Figura 41 – Ferramenta "Inspetor de Armazenamento"do Firefox Devtools	27
Figura 42 – Ferramenta "Visualizador de Propriedades DOM"do Firefox Devtools	27
Figura 43 – Ferramenta "Conta-gotas"do Firefox Devtools	28
Figura 44 – Ferramenta "Captura de Tela"do Firefox Devtools	28
Figura 45 – Ferramenta "Régua"do Firefox Devtools	29

Figura 46 – Ferramenta "Traçador JavaScript"do Firefox Devtools	30
Figura 47 – Ferramenta "Elementos"do Safari Web Inspector	31
Figura 48 – Ferramenta "Console"do Safari Web Inspector	31
Figura 49 – Ferramenta "Fontes"do Safari Web Inspector	32
Figura 50 – Ferramenta "Rede"do Safari Web Inspector	33
Figura 51 – Ferramenta "Linha do Tempo"do Safari Web Inspector	33
Figura 52 – Ferramenta "Armazenamento"do Safari Web Inspector	34
Figura 53 – Ferramenta "Gráficos"do Safari Web Inspector	35
Figura 54 – Ferramenta "Camadas"do Safari Web Inspector	35
Figura 55 – Ferramenta "Auditoria"do Safari Web Inspector	36
Figura 56 – Ferramenta "Elementos"do Microsoft Edge Devtools	37
Figura 57 – Ferramenta "Console"do Microsoft Edge Devtools	37
Figura 58 – Ferramenta "Fontes"do Microsoft Edge Devtools	38
Figura 59 – Ferramenta "Rede"do Microsoft Edge Devtools	38
Figura 60 – Ferramenta "Desempenho"do Microsoft Edge Devtools	39
Figura 61 – Ferramenta "Aplicação"do Microsoft Edge Devtools	40
Figura 62 – Ferramenta "Exibição 3D"do Microsoft Edge Devtools	40
Figura 63 – Ferramenta "Animações"do Microsoft Edge Devtools	41
Figura 64 – Ferramenta "Alterações"do Microsoft Edge Devtools	41
Figura 65 – Ferramenta "Cobertura"do Microsoft Edge Devtools	42
Figura 66 – Ferramenta "Analizador de Falhas"do Microsoft Edge Devtools	42
Figura 67 – Ferramenta "Descrição Geral de CSS"do Microsoft Edge Devtools	43
Figura 68 – Ferramenta "Farol"do Microsoft Edge Devtools	44
Figura 69 – Ferramenta "Mídia"do Microsoft Edge Devtools	44
Figura 70 – Ferramenta "Inspetor de Memória"do Microsoft Edge Devtools	45
Figura 71 – Ferramenta "Condições de Rede"do Microsoft Edge Devtools	45
Figura 72 – Ferramenta "Console de Rede"do Microsoft Edge Devtools	46
Figura 73 – Ferramenta "Bloqueio de Solicitações de Rede"do Microsoft Edge Devtools	46
Figura 74 – Ferramenta "Monitoramento de Desempenho"do Microsoft Edge Devtools	47
Figura 75 – Ferramenta "Gravador"do Microsoft Edge Devtools	47
Figura 76 – Ferramenta "Renderização"do Microsoft Edge Devtools	48
Figura 77 – Ferramenta "Segurança"do Microsoft Edge Devtools	48
Figura 78 – Ferramenta "Sensores"do Microsoft Edge Devtools	49
Figura 79 – Ferramenta "Monitor de Mapas de Origem"do Microsoft Edge Devtools	49
Figura 80 – Ferramenta "WebAuthn"do Microsoft Edge Devtools	50
Figura 81 – Ferramenta "WebAudio"do Microsoft Edge Devtools	50

LISTA DE TABELAS

Tabela 1 – Comparação das ferramentas nativas dos navegadores - Parte 1	51
Tabela 2 – Comparação das ferramentas nativas dos navegadores - Parte 2	52

LISTA DE ABREVIATURAS E SIGLAS

DOM: Modelo de objeto de documento, do inglês *Document Object Model*

CSS: Folhas de estilo em cascata, do inglês *Cascading Style Sheets*

HTML: Linguagem de Marcação de Hipertexto, do inglês *HyperText Markup Language*

XML: Linguagem de Marcação Extensível, do inglês *eXtensible Markup Language*

JS: Linguagem de Programação JavaScript, do inglês *JavaScript*

REPL: Ambiente de execução interativo, do inglês *Read-Eval-Print Loop*

CSP: Política de Segurança de Conteúdo, do inglês *Content Security Policy*

CORS: Compartilhamento de Recursos de Origem Cruzada, do inglês *Cross-Origin Resource Sharing*

SEO: Otimização para Motores de Busca, do inglês *Search Engine Optimization*

FCP: Primeira Pintura com Conteúdo, do inglês *First Contentful Paint*

TBT: Tempo Total de Bloqueio, do inglês *Total Blocking Time*

CPU: Unidade Central de Processamento, do inglês *Central Processing Unit*

LCP: Maior Pintura com Conteúdo, do inglês *Largest Contentful Paint*

CLS: Deslocamento Acumulado de Layout, do inglês *Cumulative Layout Shift*

INP: Interação com a próxima pintura, do inglês *Interaction to Next Paint*

API: Interface de Programação de Aplicações, do inglês *Application Programming Interface*

ASCII: Código Padrão Americano para o Intercâmbio de Informação, do inglês *American Standard Code for Information Interchange*

C++: Linguagem de Programação C++, do inglês *C++ Programming Language*

Wasm: WebAssembly, do inglês *WebAssembly*

HTTP: Protocolo de Transferência de Hipertexto, do inglês *HyperText Transfer Protocol*

HTTPS: Protocolo de Transferência de Hipertexto Seguro, do inglês *HyperText Transfer Protocol Secure*

URL: Localizador Uniforme de Recursos, do inglês *Uniform Resource Locator*

CDP: Protocolo do Chrome DevTools, do inglês *Chrome DevTools Protocol*

JSON: Notação de Objetos JavaScript, do inglês *JavaScript Object Notation*

USB: Barramento Serial Universal, do inglês *Universal Serial Bus*

NFC: Comunicação por Campo de Proximidade, do inglês *Near Field Communication*

CTAP2: Protocolo do Cliente para Autenticador 2, do inglês *Client to Authenticator Protocol 2*

U2F: Fator Universal 2, do inglês *Universal 2nd Factor*

ID: Identificador, do inglês *Identifier*

HAR: Arquivo HTTP, do inglês *HTTP Archive*

XHR: Requisição HTTP XML, do inglês *XMLHttpRequest*

SSE: Eventos Enviados pelo Servidor, do inglês *Server-Sent Events*

DPR: Razão de Pixels do Dispositivo, do inglês *Device Pixel Ratio*

W3C: Consórcio World Wide Web, do inglês *World Wide Web Consortium*

CLI: Interface de Linha de Comando, do inglês *Command Line Interface*

SQL: Linguagem de Consulta Estruturada, do inglês *Structured Query Language*

SVG: Gráficos Vetoriais Escaláveis, do inglês *Scalable Vector Graphics*

AVIF: Formato de Arquivo de Imagem AV1, do inglês *AV1 Image File Format*

WebP: Formato de Imagem Web, do inglês *Web Picture format*

WAI: Iniciativa de Acessibilidade Web, do inglês *Web Accessibility Initiative*

ARIA: Aplicações Ricas de Internet Acessíveis, do inglês *Accessible Rich Internet Applications*

REST: Transferência de Estado Representacional, do inglês *Representational State Transfer*

FPS: Quadros por Segundo, do inglês *Frames Per Second*

Sumário

1 – Introdução	1
1.1 Objetivos	1
1.1.1 Objetivo Geral	1
1.1.2 Objetivos específicos	1
1.2 Justificativa	2
1.3 Materiais utilizados	2
1.4 Metodologia	4
2 – Ferramentas disponibilizadas pelos navegadores	5
2.1 Google Chrome	5
2.2 Mozilla Firefox	23
2.3 Apple Safari	30
2.4 Microsoft Edge	36
2.5 Tabela comparativa das ferramentas nativas dos navegadores	51
3 – Resultados	53
4 – Considerações Finais	54
Referências	55

1 Introdução

No cenário dinâmico do desenvolvimento web, as ferramentas de desenvolvimento (*devtools*) integradas aos navegadores se tornaram indispensáveis para a criação e manutenção de aplicações web complexas e interativas. Essas ferramentas, acessíveis mediante um simples atalho de teclado (geralmente na tecla F12) (??), oferecem aos desenvolvedores um conjunto robusto de funcionalidades que facilitam a depuração de código, a inspeção do DOM, a análise de desempenho e diversas outras tarefas importantes para a construção de experiências digitais de alta qualidade (??).

Este estudo pretende realizar uma análise exploratória das principais ferramentas de desenvolvimento presentes nos navegadores web mais utilizados atualmente, sendo eles o Chrome, Firefox, Edge e Safari ¹. Por meio de uma análise de suas funcionalidades, interfaces e aplicações práticas, busca-se compreender como essas ferramentas influenciam o dia a dia dos desenvolvedores e quais são os benefícios que elas proporcionam para o desenvolvimento de aplicações web.

Além de comparar as funcionalidades e interfaces das *devtools* dos principais navegadores, este estudo também busca entender como os desenvolvedores utilizam essas ferramentas em seus projetos. Por meio de uma pesquisa com profissionais da área, será possível identificar as práticas mais comuns, as dificuldades encontradas e as necessidades não atendidas. Ao final, espera-se que este estudo contribua para o avanço do conhecimento sobre as *devtools* e para a melhoria das práticas de desenvolvimento web, tornando o trabalho dos desenvolvedores mais eficiente e produtivo.

1.1 Objetivos

1.1.1 Objetivo Geral

Este estudo visa fornecer uma análise aprofundada das ferramentas de desenvolvimento (*devtools*) integradas aos principais navegadores web. Por meio de uma análise exploratória de suas funcionalidades, interfaces e aplicações práticas, busca-se compreender como essas ferramentas influenciam o dia a dia dos desenvolvedores e quais são os benefícios que elas proporcionam para o desenvolvimento de aplicações web.

1.1.2 Objetivos específicos

- Comparar as funcionalidades oferecidas pelos *devtools* dos principais navegadores da atualidade, identificando semelhanças, diferenças e características únicas de cada ferramenta.
- Identificar as práticas mais comuns de utilização das *devtools* entre os desenvolvedores, incluindo as ferramentas e recursos mais utilizados.
- Avaliar o impacto do uso das *devtools* na produtividade dos desenvolvedores, na correção de erros e na qualidade final dos sistemas.
- Avaliar a necessidade de plugins e extensões durante o desenvolvimento de sistemas web.

¹<<https://gs.statcounter.com/browser-market-share#monthly-202401-202501>>

1.2 Justificativa

A crescente demanda por soluções digitais e a evolução constante da internet impulsionam o desenvolvimento de aplicações web cada vez mais sofisticadas e complexas. Nesse contexto, as ferramentas de desenvolvimento web desempenham um papel fundamental, permitindo que os desenvolvedores criem experiências online inovadoras e eficientes. No entanto, a diversidade de navegadores disponíveis no mercado e a constante atualização de suas funcionalidades tornam a escolha da ferramenta ideal e a garantia de compatibilidade tarefas desafiadoras.

Essa complexidade não é apenas uma questão técnica, pois a relevância do desenvolvimento web transcende os limites da área tecnológica, impactando diretamente a sociedade na totalidade. A internet se tornou uma plataforma indispensável para comunicação, comércio, educação e entretenimento, e as aplicações web são os alicerces que sustentam essa infraestrutura digital (??). Portanto, a compreensão das ferramentas e técnicas utilizadas nesse processo é crucial para acompanhar a evolução tecnológica e atender às demandas de um mercado cada vez mais exigente.

Diante desse cenário, este trabalho delimita seu escopo de análise técnica aos quatro navegadores de maior relevância mercadológica e estrutural na atualidade: Google Chrome, Microsoft Edge, Apple Safari e Mozilla Firefox. A escolha deste conjunto justifica-se pela representatividade de mercado, visto que, somados, estes softwares concentram a vasta maioria do tráfego web global, garantindo o alcance demográfico das aplicações. Além disso, sob a ótica da Engenharia de Software, este grupo abrange as três principais famílias de motores de renderização (Rendering Engines) vigentes — Blink, WebKit e Gecko —, tornando-os indispensáveis para a validação de interoperabilidade (cross-browser testing) e aderência aos padrões web (W3C). Assim, ao dissecar as especificidades de cada ferramenta, o estudo visa auxiliar desenvolvedores a tomar decisões fundamentadas, impulsionando a qualidade final dos projetos web.

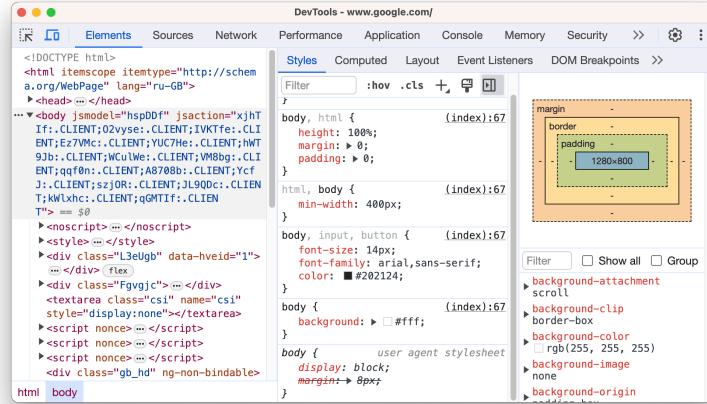
1.3 Materiais utilizados

As ferramentas de desenvolvimento do navegador, também conhecidas como *DevTools*, são conjuntos de utilitários incorporados nos navegadores da web, como Google Chrome e Mozilla Firefox. Esses recursos permitem que os desenvolvedores inspecionem e depurem o código-fonte, analisem o desempenho e verifiquem o uso de memória de seus aplicativos (??). Durante o desenvolvimento deste trabalho, foram utilizadas as principais ferramentas disponíveis no mercado, conforme detalhado a seguir.

Inicialmente, destaca-se o Chrome DevTools (Figura 1). Este representa um conjunto abrangente de ferramentas integradas ao navegador Google Chrome, essenciais para o desenvolvimento e depuração de aplicações web. Com ele, é possível inspecionar o DOM, analisar o desempenho da página, depurar JavaScript e simular diferentes dispositivos (??). Como uma alternativa robusta ao ambiente da Google, o Firefox Developer Tools (Figura 2) apresenta um foco maior em privacidade e personalização. Essas ferramentas proporcionam um ambiente de desenvolvimento completo, permitindo a inspeção de elementos, a análise de redes, a depuração de JavaScript, a visualização de estilos CSS e a otimização de desempenho. Ademais, o Firefox Developer Tools integra-se ao ecossistema Mozilla, incluindo o editor Visual Studio Code, o que facilita o fluxo de trabalho dos desenvolvedores (??).

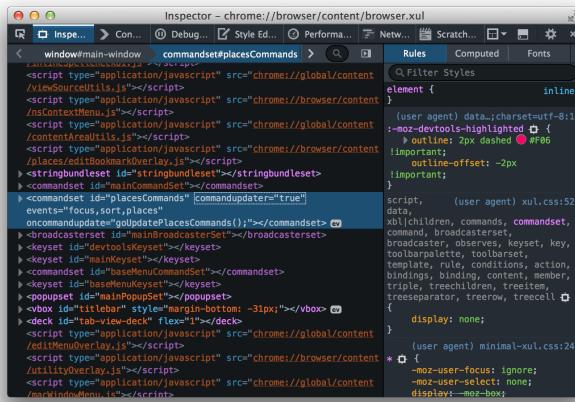
No mesmo cenário de navegadores modernos, os Edge DevTools (Figura 3), incorporados ao Microsoft Edge, oferecem uma experiência eficiente com uma interface visualmente atraente. Apresentando funcionalidades similares às do Chrome, esta ferramenta permite inspe-

Figura 1 – Chrome Devtools



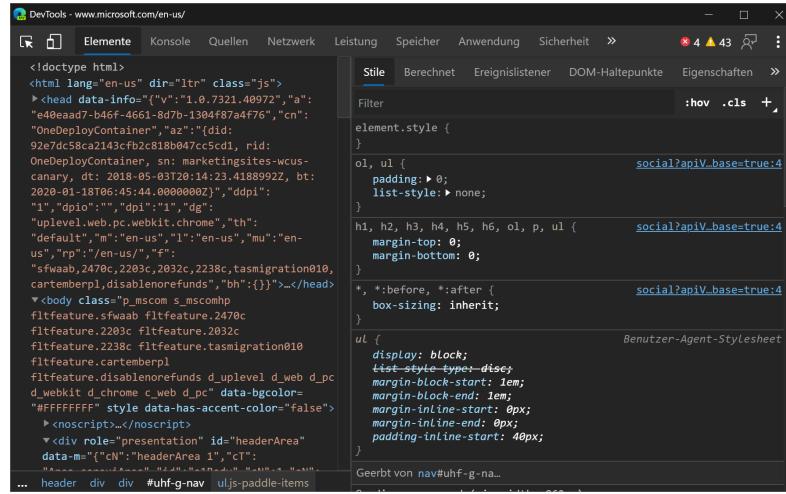
Fonte: Chrome for Developers

Figura 2 – Firefox Developer Tools



Fonte: Firefox Source Tree Documentation

Figura 3 – Edge DevTools



Fonte: Microsoft Learn

cionar elementos, depurar JavaScript e analisar o desempenho da página. A sua forte integração com o Microsoft Visual Studio Code torna os Edge DevTools uma opção particularmente atraente para desenvolvedores que utilizam a plataforma Windows (??).

Por fim, o Safari Web Inspector (Figura ??) é a ferramenta de desenvolvimento integrada ao navegador Apple Safari, projetada para oferecer uma experiência de depuração de alta qualidade. Com foco em desempenho e usabilidade, o Web Inspector permite inspecionar elementos, depurar JavaScript e simular diferentes dispositivos dentro do ecossistema Apple (??).

1.4 Metodologia

Para o desenvolvimento deste trabalho, adotou-se a metodologia de **Pesquisa Documental**, fundamentada na análise rigorosa de **Documentação Técnica**. Esta abordagem é imperativa para o tema, visto que as ferramentas de desenvolvimento web (DevTools) são regidas por especificações técnicas complexas, documentação oficial fornecida pelos fabricantes (como Google, Mozilla, Microsoft e Apple) e padrões da web (W3C). Nesse cenário, a pesquisa documental permite extrair dados fidedignos e objetivos diretamente das fontes primárias, garantindo uma comparação baseada em fatos técnicos e não apenas em observações empíricas.

Teoricamente, a pesquisa documental distingue-se pelo uso de fontes que ainda não receberam tratamento analítico, servindo como base para investigações onde se busca precisão técnica e histórica (??). Diferente da revisão bibliográfica, que analisa obras de outros autores, a análise de documentação técnica foca em manuais, *release notes* (notas de lançamento), especificações de API e guias oficiais de implementação. Isso elimina a subjetividade, permitindo avaliar as ferramentas com base no que elas foram projetadas tecnicamente para realizar.

Portanto, no contexto específico deste estudo, tal metodologia se justifica pela necessidade de validar as capacidades nativas de cada navegador através de suas definições oficiais. Consequentemente, a metodologia será utilizada para:

- **Mapear Funcionalidades Oficiais:** Catalogar os recursos existentes em cada ambiente (Chrome, Firefox, Edge e Safari) consultando diretamente a documentação do desenvolvedor (ex: MDN Web Docs, Chrome Developers).
- **Realizar Análise Comparativa Técnica:** Confrontar as especificações das ferramentas para identificar discrepâncias de implementação, suporte a novos padrões CSS/JS e exclusividades de cada ecossistema.
- **Verificar Atualizações e Versionamento:** Analisar os registros de alteração (*change-logs*) para compreender a evolução das ferramentas de depuração e o ciclo de introdução de novos recursos de produtividade.

Por fim, a aplicação dessa metodologia será realizada em etapas sequenciais: seleção das fontes oficiais, extração sistemática de dados técnicos, categorização das funcionalidades e, finalmente, a síntese comparativa dos resultados.

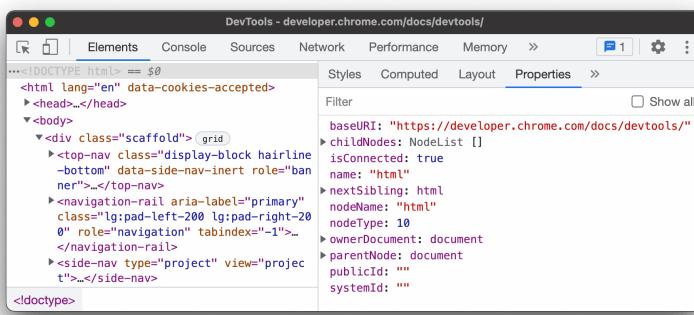
2 Ferramentas disponibilizadas pelos navegadores

Para compreender melhor as possibilidades oferecidas pelas DevTools e como elas podem ser otimizadas, esta seção apresenta uma lista das ferramentas disponíveis nos principais navegadores do mercado atualmente, explorando suas funcionalidades, onde estão disponíveis e aplicações práticas no desenvolvimento web. A lista será apresentada em ordem alfabética.

2.1 Google Chrome

O navegador Google Chrome integra um conjunto de ferramentas de desenvolvimento web, conhecido como Chrome DevTools. Esta suíte é projetada para instrumentar, inspecionar, depurar e criar perfis de aplicações web diretamente no navegador. As ferramentas fornecem funcionalidades para uma variedade de tarefas, incluindo a manipulação do DOM, alteração de CSS, análise de desempenho de carregamento de página e monitoramento de solicitações de rede. A seção subsequente apresenta uma análise detalhada dos painéis que compõem este conjunto, elucidando as capacidades e finalidades específicas de cada ferramenta.

Figura 4 – Ferramenta "Elementos" do Chrome Devtools

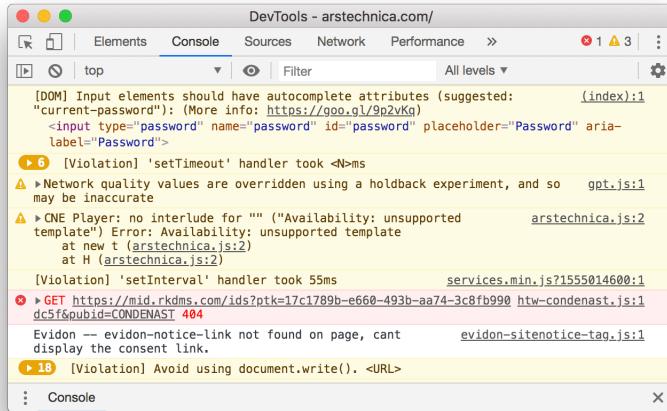


Fonte: Chrome for Developers

A ferramenta “Elementos” (*Elements*) é utilizada para realizar a inspeção e manipulação em tempo real do *Document Object Model (DOM)*. Ele renderiza a estrutura da página em uma árvore *DOM* hierárquica, como apresentado na Figura 4, permitindo a seleção precisa e a modificação direta de nós. As funcionalidades de edição permitem alterar dinamicamente o conteúdo textual, modificar atributos, alterar o tipo de nó e reordenar elementos estruturalmente (??). A ferramenta também facilita a depuração ao permitir forçar estados de elementos (como *:hover* ou *:focus*), ocultar ou excluir nós, e definir pontos de interrupção (*breakpoints*) que pausam a execução do script mediante modificações específicas no *DOM* (??).

Além da manipulação estrutural, esta ferramenta é central para a análise e depuração de *CSS*. A sub-aba “Estilos” (*Styles*) exibe o conjunto de regras *CSS* aplicadas a um nó selecionado, detalhando a cascata e identificando propriedades (????). De forma complementar, agrupa ainda ferramentas especializadas para a depuração de *layouts* complexos, como *Grid* e *Flexbox* (????). O painel “Console” (*Console*) atua como uma interface primária para o diagnóstico e interação com aplicações *web*, como exibido na figura 5, servindo a duas funções principais: a visualização de mensagens registradas e a execução de *JavaScript* (??). Desenvolvedores utilizam a *API* do *Console*, através de métodos, para enviar mensagens de diagnóstico durante

Figura 5 – Ferramenta "Console" do Chrome Devtools



Fonte: Chrome for Developers

a execução do código. Este mecanismo de *logging* é fundamental para verificar a ordem de execução e inspecionar os valores das variáveis em pontos específicos, facilitando a depuração e a análise do comportamento do *script*.

Adicionalmente, o *Console* funciona como um ambiente *REPL* (*Read-Eval-Print Loop*), permitindo a execução interativa de código *JavaScript* no contexto da página inspecionada (??). Por possuir acesso total ao objeto 'window' da página, os desenvolvedores podem utilizá-lo para interagir programaticamente com o *DOM*, modificar dinamicamente o conteúdo da página ou testar novas funcionalidades isoladamente (??).

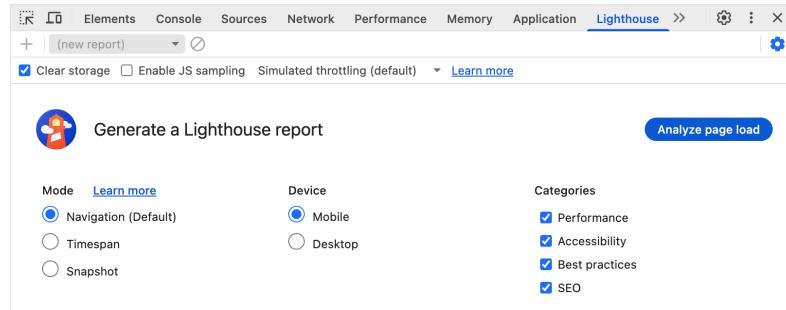
Figura 6 – Ferramenta "Problemas" do Chrome Devtools

Fonte: Chrome for Developers

O painel “Problemas” (*Issues*), apresentado na figura 6, é utilizado para diagnósticos e anomalias detectadas pelo navegador, como problemas de *cookies*, conteúdo misto, erros de *CORS* e violações da Política de Segurança de Conteúdo (*CSP*) (??). Sua função é reduzir a verbosidade e a desordem de notificações no “Console”, apresentando os problemas de forma estruturada, agregada e acionável. Cada item reportado inclui uma descrição contextual, uma solução recomendada e uma seção de “Recursos Afetados” que fornece *links* diretos para o contexto relevante em outras ferramentas, como “Elementos”. O recurso também permite agrupar os problemas por tipo e filtrar a exibição de problemas relacionados a *cookies* de terceiros (??).

Esta ferramenta é projetada para realizar auditorias abrangentes da qualidade de aplicações *web*, como exibido na figura 7. Sua principal função é gerar um relatório detalhado que avalia o *site* em diversas categorias fundamentais, including Desempenho, Acessibilidade, Práticas Recomendadas e *SEO* (??). A ferramenta permite a configuração do ambiente de teste, como a emulação de dispositivos móveis, a seleção de modos de análise e a simulação de condições de rede (limitação simulada) (??). A execução de uma auditoria estabelece um referencial quantitativo, essencial para medir o impacto de otimizações subsequentes, e fornece recomendações práticas sobre as mudanças que podem gerar maior impacto na *performance*.

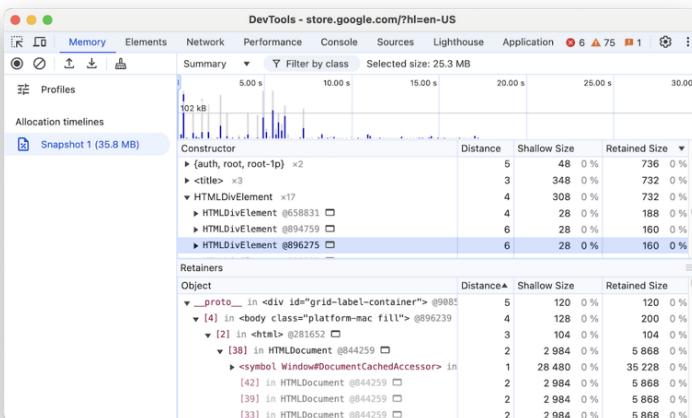
Figura 7 – Ferramenta "Lighthouse"do Chrome Devtools



Fonte: Chrome for Developers

O relatório resultante da auditoria apresenta uma pontuação de desempenho geral e detalha métricas quantitativas cruciais, como *FCP* (Primeira Exibição de Conteúdo) e *TBT* (Tempo total de bloqueio) (??). O painel “Memória” (*Memory*), apresentado na figura 8,

Figura 8 – Ferramenta "Memória"do Chrome Devtools

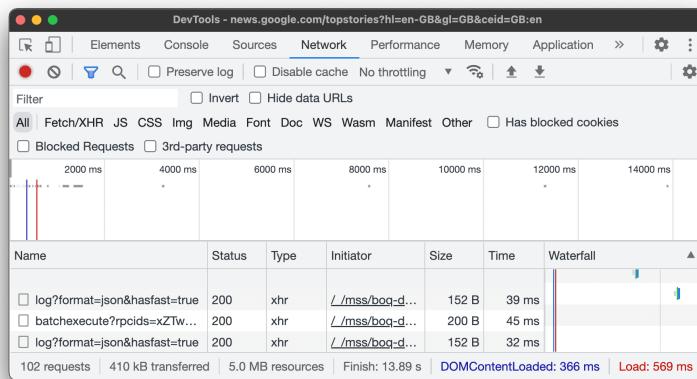


Fonte: Chrome for Developers

fornece um conjunto de ferramentas de diagnóstico para a análise do uso da memória em aplicações web. Ele é projetado para identificar problemas como vazamentos de memória (*memory leaks*), inchaço de memória (*memory bloat*) e coletas de lixo (*garbage collections*) frequentes (??). A ferramenta oferece quatro tipos de perfis de captura: “Instantâneo de alocação heap” (*Heap snapshot*), que exibe a distribuição de memória entre objetos JavaScript e nós *DOM*; “Instrumentação de alocação na linha do tempo” (*Allocation instrumentation on timeline*), que monitora alocações ao longo do tempo para isolar vazamentos; “Amostragem de alocação” (*Allocation sampling*), um método de baixo *overhead* que registra alocações de memória por pilha de execução JavaScript; e “Elementos separados” (*Detached elements*), que identifica objetos retidos por referências JavaScript (??). A análise dos *snapshots* de *heap* permite a identificação de nós *DOM* separados (*detached DOM nodes*), uma causa comum de vazamentos de memória (??).

O painel “Rede” (*Network*) é uma ferramenta de diagnóstico, como exibido na figura 9 abaixo, utilizada para monitorar e analisar a atividade de rede de uma aplicação *web*, registrando todas as solicitações de recursos (*requests*) e suas respectivas respostas (*responses*) em um *log* cronológico (??). Sua finalidade primária é a verificação da transferência de recursos e a inspeção das propriedades de solicitações individuais. A seleção de um recurso específico permite uma análise detalhada através de sub-abas, que expõem os cabeçalhos *HTTP* (*Headers*), o conteúdo da resposta (*Response*), a cadeia de causa da solicitação (*Initiator*) e um detalhamento temporal da atividade de rede (*Timing*) (??). A ferramenta também permite a exportação dos dados de atividade de rede em formato *HAR* (*HTTP Archive*) (??).

Figura 9 – Ferramenta "Rede" do Chrome Devtools

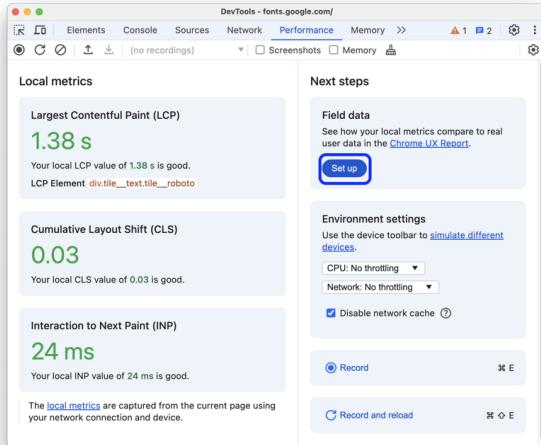


Fonte: Chrome for Developers

O painel “Desempenho” (*Performance*), exibido na figura 10 abaixo, é uma ferramenta de diagnóstico avançada projetada para gravar e analisar perfis de desempenho da *CPU* em aplicações *web* (??). Ele permite a captura detalhada da *performance* tanto em tempo de execução (*runtime*) quanto durante o carregamento da página (*load*), com o objetivo de identificar gargalos e otimizar a utilização de recursos. A ferramenta monitora métricas vitais (*Core Web Vitals*) em tempo real, como *LCP*, *CLS* e *INP*, e oferece a capacidade de comparar dados locais com os dados de campo agregados do *Chrome UX Report* (??). As configurações de captura permitem a simulação precisa das condições do usuário, notadamente através da limitação (*throttling*) de *CPU* e rede. O relatório de *performance* resultante fornece uma visualização detalhada da atividade da *thread* principal através de um gráfico de chamas (*flame graph*), juntamente com métricas de memória, capturas de tela do processo de renderização e análises tabulares, como “Bottom-Up” e “Árvore de Chamadas” (*Call Tree*) (??).

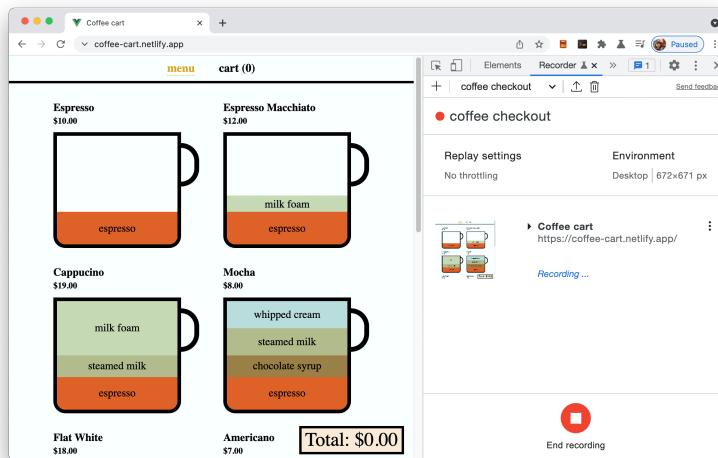
O painel “Gravador” (*Recorder*), apresentado na figura 11 abaixo, constitui uma ferramenta de diagnóstico avançada, projetada para a análise e depuração de fluxos de interação do usuário em aplicações *web* (??). Sua funcionalidade central permite a captura e reprodução de sequências de eventos do usuário, facilitando a identificação de anomalias em caminhos de conversão e a medição da *performance*. A ferramenta oferece capacidades de edição interativa das etapas gravadas, permitindo a inserção de pontos de interrupção (*breakpoints*) e a execução passo a passo para uma depuração detalhada (??). Adicionalmente, os fluxos de usuário capturados podem ser exportados em formatos estruturados, como *JSON*, ou como *scripts* de automação para bibliotecas externas, como o *Puppeteer*, permitindo a integração com processos de teste automatizados (??).

Figura 10 – Ferramenta "Desempenho" do Chrome Devtools



Fonte: Chrome for Developers

Figura 11 – Ferramenta "Gravador" do Chrome Devtools

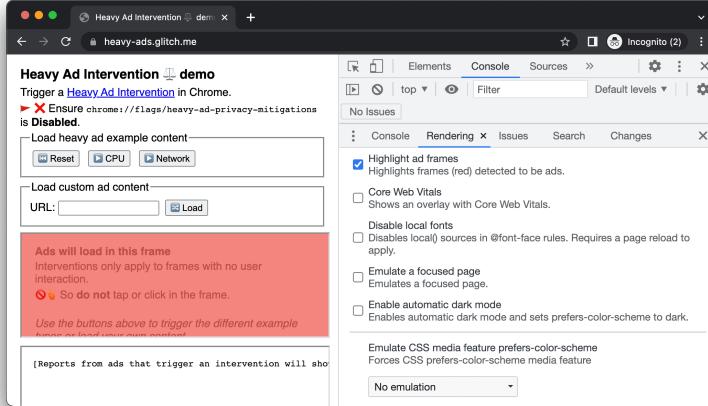


Fonte: Chrome for Developers

O painel "Renderização" (*Rendering*), exibido na figura 12 é um conjunto de ferramentas de diagnóstico focado na análise de aspectos visuais e de *performance* da renderização de conteúdo web. Suas funcionalidades primárias permitem a identificação de problemas de desempenho, como repinturas (*Paint Flashing*), mudanças de layout (*Layout Shifts*), problemas de rolagem, e a visualização de estatísticas de renderização e métricas das *Core Web Vitals* (??). A ferramenta também oferece capacidades avançadas de emulação de recursos de mídia CSS, permitindo testar como a página é renderizada sob diferentes condições, como esquemas de cores preferidos ou redução de movimento, sem necessidade de alteração no código-fonte (??). Adicionalmente, disponibiliza efeitos para depuração, incluindo a emulação de foco na página, a ativação de um modo escuro automático e a simulação de diversas deficiências visuais para análise de acessibilidade (??).

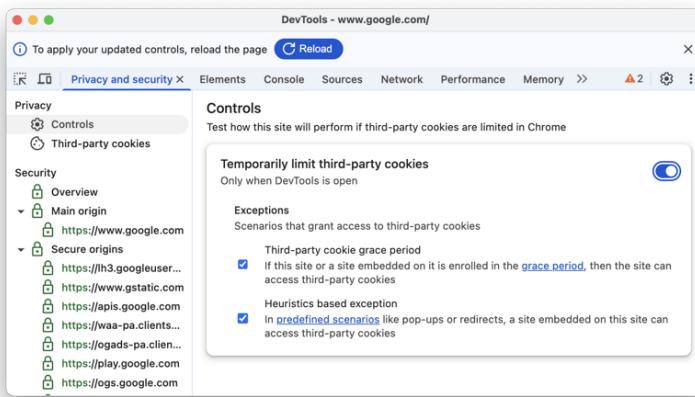
O painel "Privacidade e Segurança", apresentado acima na figura 13, é uma ferramenta de diagnóstico que permite a inspeção e depuração de protocolos de segurança e o gerenciamento da privacidade da página. Esta funcionalidade possibilita a análise das origens dos recursos,

Figura 12 – Ferramenta "Renderização" do Chrome Devtools



Fonte: Chrome for Developers

Figura 13 – Ferramenta "Segurança" do Chrome Devtools

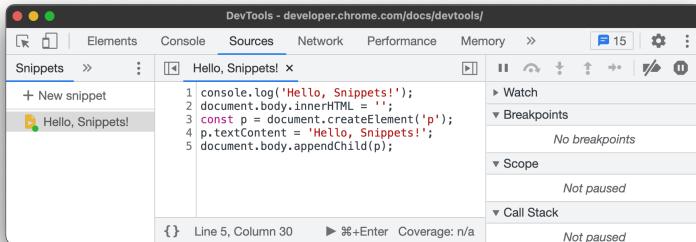


Fonte: Chrome for Developers

exibindo detalhes da conexão, avisos de segurança *HTTP* e informações de certificados (??). O painel identifica problemas críticos, como origens principais não seguras (solicitadas via *HTTP*), configurações de *HTTPS* corrompido (por exemplo, certificados inválidos) e a ocorrência de conteúdo misto, onde recursos inseguros são requisitados em uma página segura. Adicionalmente, a seção de "Privacidade" oferece mecanismos para inspecionar e simular a limitação temporária de *cookies* de terceiros, permitindo a verificação do comportamento do *site* sob restrições de rastreamento (??).

O painel "Fontes"(Sources), como pode ser visto na figura 14, constitui um ambiente integrado para a inspeção, edição e depuração dos recursos de uma aplicação *web*. Ele permite o acesso e a visualização da árvore de arquivos carregados, incluindo *scripts JavaScript*, folhas de estilo *CSS* e imagens. Sua funcionalidade principal é a depuração de *JavaScript*, permitindo ao desenvolvedor definir pontos de interrupção (*breakpoints*) para pausar intencionalmente a execução do código e analisar o fluxo de controle linha por linha (??). Adicionalmente, o painel funciona como um editor de código-fonte para *CSS* e *JavaScript*, possibilita a criação e execução de "Snippets"(fragmentos de *script* reutilizáveis) e suporta a configuração de "Workspaces"para persistir as modificações feitas no navegador diretamente no sistema de arquivos local (??).

Figura 14 – Ferramenta "Fontes" do Chrome Devtools



Fonte: Chrome for Developers

Figura 15 – Ferramenta "Autofill" do Chrome Devtools

Form field	Predicted autofill value	Value
name (text)	Full name (heur.)	"Homer Simpson"
address (text)	Street address (heur.)	"123 Main Street"
city (text)	City (heur.)	"Springfield"
state (select-one)	State (heur.)	"IL"
zip (text)	Zip code (heur.)	"55123"
country (text)	Country (heur.)	"United States"
company (text)	Company name (heur.)	""
email (text)	Email address (heur.)	"hjs@aol.com"
phone (text)	Phone number (heur.)	""

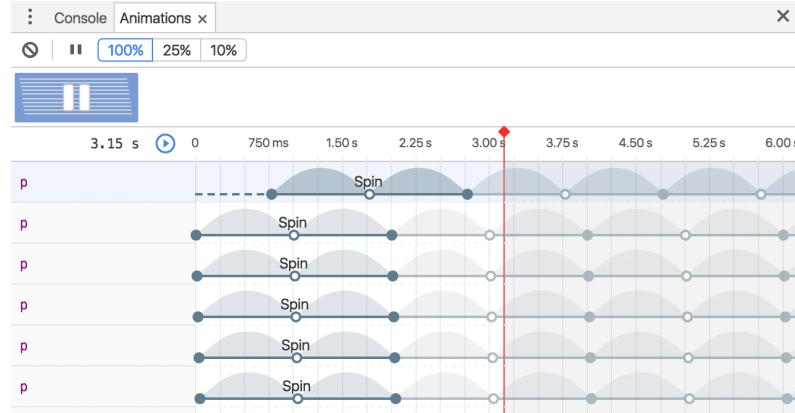
Fonte: Chrome for Developers

O painel "Autofill"(Preenchimento automático), exibido na figura 15, é uma ferramenta de diagnóstico utilizada para inspecionar e depurar o preenchimento de informações de endereço em formulários web. Esta funcionalidade permite a análise do mapeamento entre os campos de formulário detectados, os valores previstos determinados heuristicamente pelo navegador e os dados de endereço reais salvos pelo usuário que são inseridos (??). O painel apresenta uma visualização tabular detalhada dessa correspondência e oferece a capacidade de injetar dados de endereço de teste para validar o comportamento do formulário (??). Adicionalmente, ele auxilia na identificação de problemas de implementação, que são relatados no painel "Issues"(Problemas) para corrigir atributos de preenchimento automático incorretos.

O painel "Animações"(Animations), ilustrada na figura 16, é uma ferramenta de diagnóstico para a inspeção e modificação de sequências de animação. Ele captura e agrupa automaticamente animações CSS, transições CSS, animações da Web (Web Animations API) e a API View Transitions, embora não ofereça suporte a animações baseadas em *requestAnimationFrame* (??). A ferramenta permite a inspeção detalhada de grupos de animação, possibilitando ao desenvolvedor diminuir a velocidade, repetir ou analisar o código-fonte associado. As funcionalidades de modificação permitem o ajuste interativo de parâmetros temporais, como duração, atraso (*delay*) e os deslocamentos de *keyframes* (??). Adicionalmente, o painel facilita a depuração da API View Transitions, permitindo a edição de seus pseudoelementos CSS (ex: ::view-transition) enquanto a execução da animação está pausada (??).

O painel "Application"(Aplicativo), exibida na figura 17, é uma ferramenta de diagnóstico abrangente utilizada para inspecionar, modificar e depurar os diversos aspectos

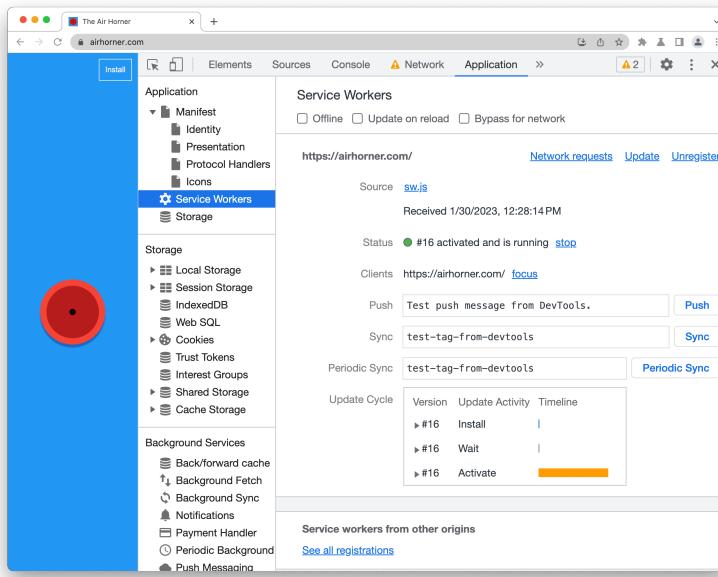
Figura 16 – Ferramenta "Animações" do Chrome Devtools



Fonte: Chrome for Developers

de armazenamento, manifesto e execução de uma aplicação *web*. Ele permite a análise do *manifest.json*, a inspeção e depuração de *service workers*, including a emulação de eventos *push*, e o gerenciamento de dados do *site*, como a simulação de cotas de armazenamento (??). A ferramenta oferece acesso granular para visualização e edição de múltiplos mecanismos de armazenamento do lado do cliente, incluindo *LocalStorage*, *SessionStorage*, *IndexedDB*, *Cookies*, *Shared Storage* e *Cache Storage* (??). Adicionalmente, o painel facilita o teste de serviços em segundo plano, como *Background Fetch*, *Background Sync* e *Speculative Loads*, e a inspeção da estrutura de *frames* da página e suas políticas de segurança associadas.

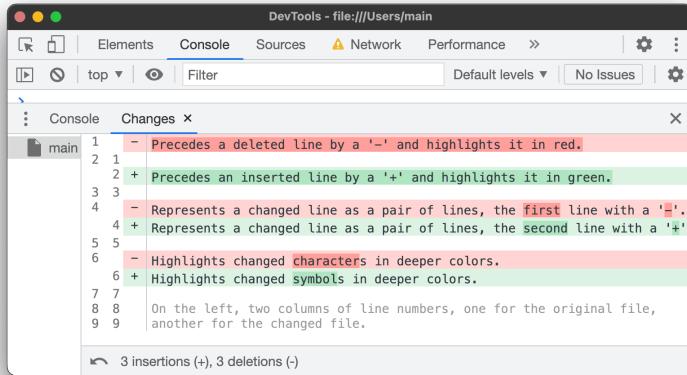
Figura 17 – Ferramenta "Aplicativo" do Chrome Devtools



Fonte: Chrome for Developers

O painel "Alterações"(*Changes*), apresentado na figura 18, opera como uma ferramenta de monitoramento em tempo real para modificações de código-fonte executadas diretamente no ambiente de inspeção do navegador. Esta funcionalidade rastreia edições efetuadas em arquivos CSS, seja no painel "Elementos > Estilos" ou em "Fontes", e em arquivos JavaScript dentro da ferramenta "Fontes"(??). O rastreamento de alterações em HTML também é suportado,

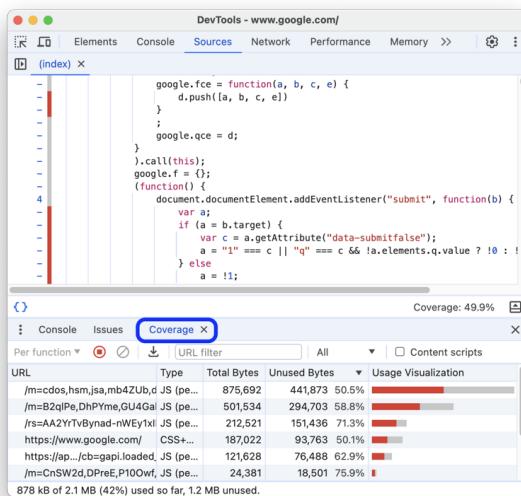
Figura 18 – Ferramenta "Alterações"do Chrome Devtools



Fonte: Chrome for Developers

contanto que o recurso "Substituições locais"(*Local Overrides*) esteja habilitado. A ferramenta apresenta uma visualização *diff* formatada automaticamente, que destaca inserções e exclusões de código (??). Funcionalidades adicionais incluem a capacidade de copiar todas as alterações de CSS acumuladas e a opção de reverter todas as modificações aplicadas a um arquivo específico.

Figura 19 – Ferramenta "Cobertura"do Chrome Devtools



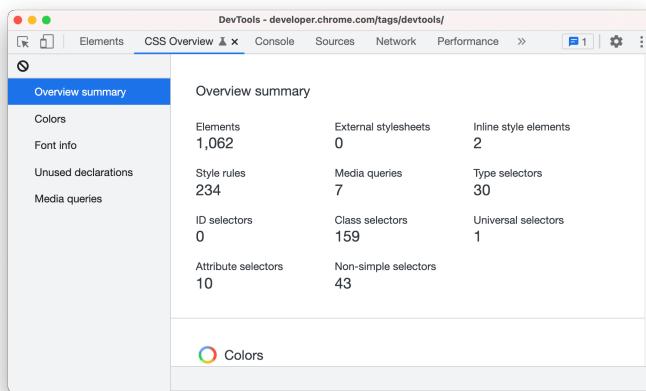
Fonte: Chrome for Developers

O painel "Cobertura"(*Coverage*), exibido na figura 19, é uma ferramenta de diagnóstico projetada para identificar código *JavaScript* e *CSS* não executado. Esta funcionalidade opera através da instrumentação do código durante uma sessão de gravação, que pode ser iniciada com um recarregamento da página, para monitorar quais partes do código são de fato utilizadas durante o carregamento e a interação do usuário (??). Ao concluir a gravação, a ferramenta gera um relatório que quantifica, para cada recurso analisado, o total de *bytes* e o volume exato de "bytes não usados". Adicionalmente, o painel "Cobertura" oferece uma visualização detalhada linha por linha na ferramenta "Fontes", demarcando o código não utilizado. A análise

pode ser configurada por escopo ("Por função"ou "Por bloco"), permitindo que desenvolvedores isolem e removam código supérfluo com o objetivo de otimizar o desempenho de carregamento da página e reduzir o consumo de dados da rede (??).

O painel "Visão geral de CSS"(*CSS Overview*), como ilustrado na figura 20, é uma ferramenta de diagnóstico que, mediante uma captura iniciada pelo usuário, gera um relatório estatístico abrangente sobre a utilização de *CSS* em uma página *web*, com o objetivo de identificar potenciais otimizações (??). Este relatório coleta dados de todas as ocorrências de *CSS*, incluindo declarações não utilizadas, e estrutura a informação em seções principais: "Resumo da visão geral", "Cores", que também identifica problemas de baixo contraste, "Informações da fonte", "Declarações não utilizadas"e "*Media queries*"(??). A ferramenta permite a investigação interativa dos elementos afetados, oferecendo funcionalidades para destacá-los na página ou inspecioná-los diretamente no painel "Elementos".

Figura 20 – Ferramenta "Visão geral de CSS"do Chrome Devtools

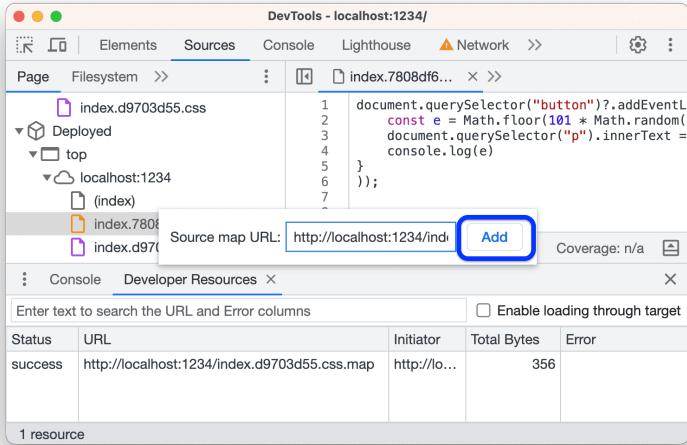


Fonte: Chrome for Developers

O painel "Recursos para desenvolvedores"(*Developer Resources*), exibida na figura 21, é uma ferramenta de diagnóstico utilizada para verificar o status de carregamento dos mapas de origem (*source maps*) pelo *Chrome DevTools* (??). Por padrão, o *DevTools* tenta carregar automaticamente os mapas de origem ao ser aberto, e este painel exibe o "Status"e eventuais mensagens de "Erro"resultantes dessas tentativas. A ferramenta permite a filtragem dos recursos por *URL* ou mensagem de erro e oferece opções para solucionar falhas de carregamento, como problemas de *cross-origin*, ao permitir que os mapas sejam requisitados através do próprio *site* (??). Adicionalmente, caso os mapas de origem não estejam presentes no ambiente, como em produção, o painel viabiliza o carregamento manual mediante o fornecimento de uma *URL*, permitindo assim a depuração do código-fonte original.

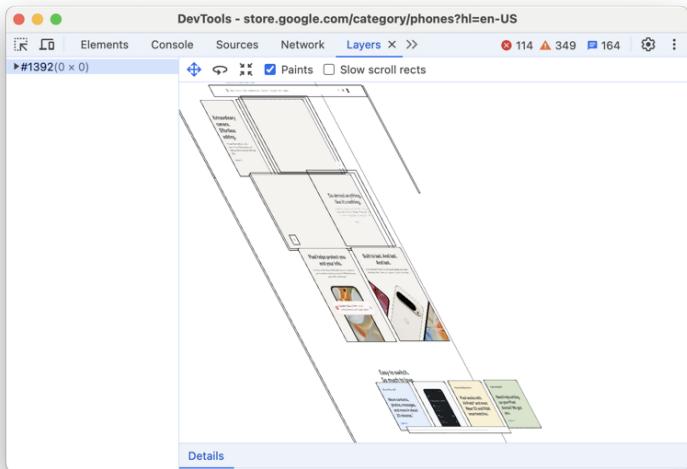
O painel "Camadas"(*Layers*), apresentado na figura 22, é uma ferramenta de diagnóstico que permite a análise da composição de renderização de um *website*. Ele apresenta um diagrama 3D interativo que ilustra como o navegador organiza o conteúdo em distintas camadas (??). Esta visualização auxilia na identificação de problemas de renderização, na otimização de animações e na depuração de *performance* de rolagem. A ferramenta lista todas as camadas renderizadas em uma árvore expansível e fornece detalhes como tamanho, contagem de pintura (*Paint Count*) e motivos para a composição (*Compositing Reasons*) (??). Funcionalidades adicionais incluem a capacidade de inspecionar informações do "Paint Profiler"e identificar regiões de rolagem lenta (*Slow scroll rects*).

Figura 21 – Ferramenta "Recursos para desenvolvedores" do Chrome Devtools



Fonte: Chrome for Developers

Figura 22 – Ferramenta "Camadas" do Chrome Devtools

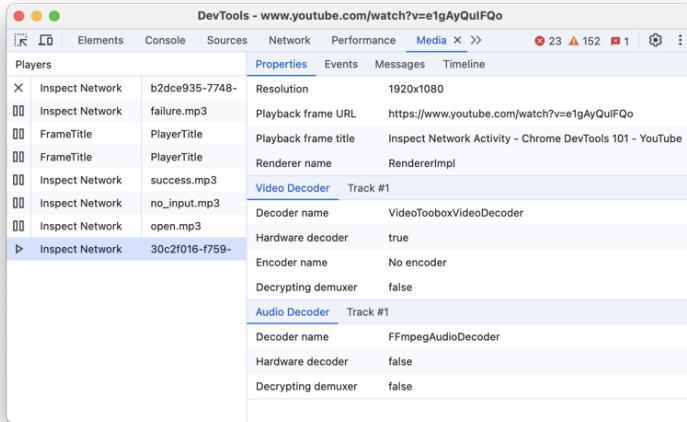


Fonte: Chrome for Developers

O painel "Mídia"(*Media*), ilustrado na figura 23, é a ferramenta primária para a inspeção e depuração de *players* de mídia incorporados em uma página *web*. Ele identifica e lista todas as fontes de áudio e vídeo ativas, permitindo a análise detalhada de cada *player* (??). A ferramenta é segmentada em abas que expõem dados técnicos: "*Properties*"(Propriedades) exibe informações como resolução e *codecs*; "*Events*"(Eventos) registra todos os eventos emitidos pelo *player*; "*Messages*"(Mensagens) apresenta *logs* de diagnóstico filtráveis; e "*Timeline*"(Linha do tempo) visualiza em tempo real os estados de reprodução e *buffer* (??). O painel também permite a exportação das informações do *player* para análise externa.

O "Inspetor de memória"(*Memory Inspector*), exibido na figura 24, é uma ferramenta de diagnóstico projetada para a inspeção de *buffers* de memória binária, como *ArrayBuffer*, *TypedArray* e *DataView* em *JavaScript*, além da *WebAssembly.Memory* de aplicações *WebAssembly* (*Wasm*) compiladas a partir de C++ (??). A interface exibe o *buffer* de memória

Figura 23 – Ferramenta "Mídia" do Chrome Devtools



Fonte: Chrome for Developers

Figura 24 – Ferramenta "Inspetor de memória" do Chrome Devtools

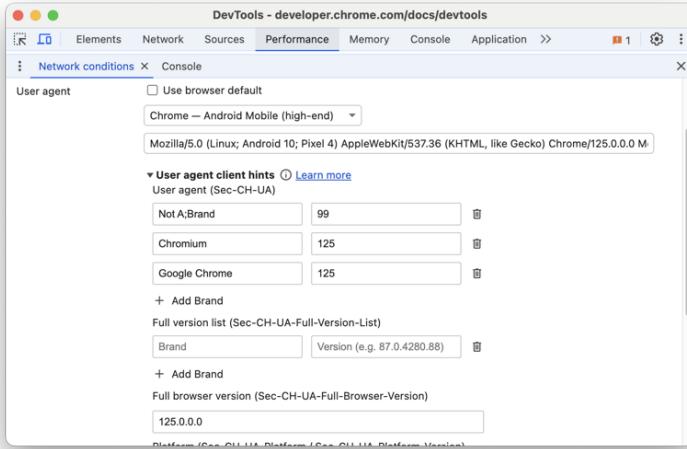
Memory Inspector	
Console	Memory Inspector X
ArrayBuffer(60) X	ArrayBuffer(70)
0x00000000 < 0x00000000 >	Big Endian
00000000 54 68 69 73 T h i s	Integer 8-bit dec 84
00000004 20 69 73 20 i s	Integer 16-bit dec 21608
00000008 61 20 73 74 a s t	Integer 32-bit dec 1416128883
0000000C 72 69 6E 67 r i n g	Integer 64-bit dec 6082227239949792032
00000010 20 69 6E 20 i n	Float 32-bit dec 3992806227968.00
00000014 74 68 65 20 t h e	Float 64-bit dec 4.171482365401182e+98
00000018 41 72 72 61 A r r a	Pointer 32-bit 0x5468697320697320 ⊕
0000001C 79 42 75 66 y B u f	Pointer 64-bit 0x5468697320697320 ⊕
00000020 66 65 72 20 f e r	
00000024 3A 29 21 00 :) .	
00000028 00 00 00 00	
0000002C 00 00 00 00	
00000030 00 00 00 00	
00000034 00 00 00 00	
00000038 00 00 00 00	
0000003C	
00000040	
00000044	

Fonte: Chrome for Developers

apresentando os endereços de *byte* e seus valores em formato hexadecimal, uma representação *ASCII* adjacente e um "Inspetor de valor" que decodifica os *bytes* selecionados em múltiplos formatos (por exemplo, ponto flutuante, inteiro) e codificações (??). A ferramenta permite a navegação direta para endereços de memória específicos, a alternância de *endianidade* (*endian-ness*) e pode ser iniciada dinamicamente durante a depuração a partir do painel "Escopo" para analisar o estado da memória de um objeto em um ponto de interrupção (??).

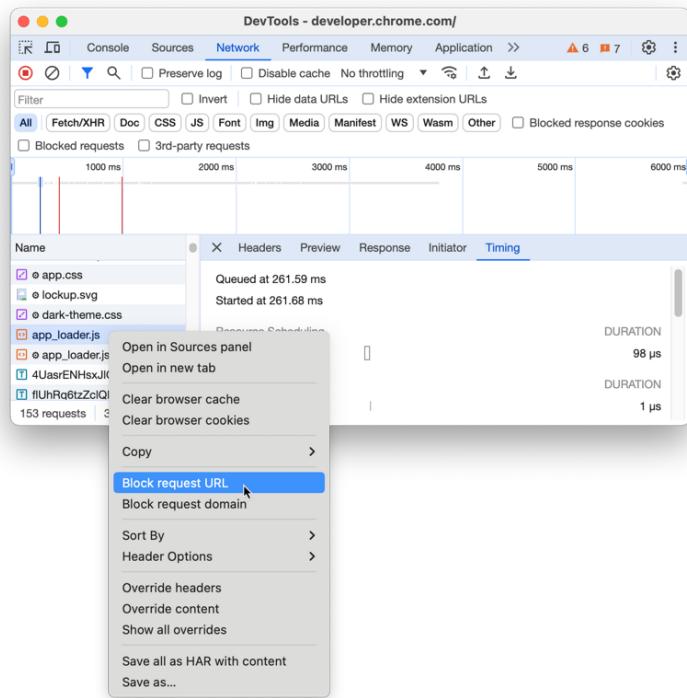
O painel "Condições de rede" (*Network Conditions*), apresentado na figura 25, é uma ferramenta de emulação que permite ao desenvolvedor substituir a *string* do *user agent* e simular diferentes velocidades de rede. A substituição da *string* do *user agent* altera a forma como o navegador se identifica para os servidores *web*, o que é utilizado para testar *design* responsivo, compatibilidade e detecção de recursos ao simular navegadores distintos ou versões anteriores (??). A ferramenta também permite a personalização das Dicas de Cliente *HTTP* (*User-Agent Client Hints*). A funcionalidade de limitação de rede (*throttling*) possibilita a simulação de conexões de rede variadas, como 3G rápida, 3G lenta ou *offline*, auxiliando na análise do comportamento da aplicação sob diferentes condições de conectividade (??).

Figura 25 – Ferramenta "Condições de rede" do Chrome Devtools



Fonte: Chrome for Developers

Figura 26 – Ferramenta "Bloqueio de solicitações de rede" do Chrome Devtools

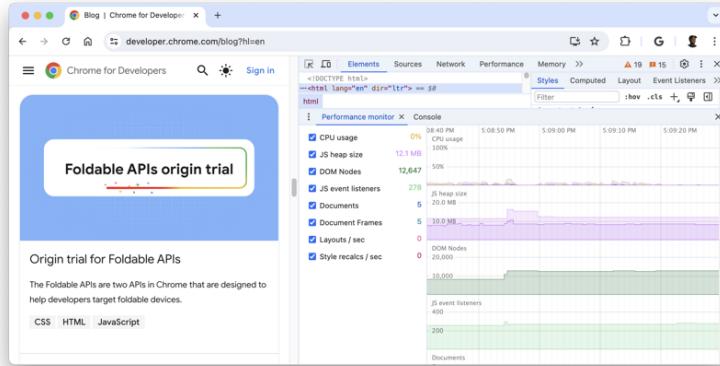


Fonte: Chrome for Developers

O painel "Bloqueio de solicitações de rede"(*Network Request Blocking*), exibido na figura 26, é uma funcionalidade de diagnóstico utilizada para testar o comportamento de uma página web sob a condição de falha no carregamento de recursos específicos, como imagens ou folhas de estilo (??). A ferramenta permite a definição de múltiplos "padrões" de bloqueio, os quais podem ser URLs completos, URLs parciais com correspondência de curinga (*), ou nomes de domínio (??). O desenvolvedor pode adicionar, editar, remover e alternar o estado (ativo/inativo) desses padrões. As regras de bloqueio também podem ser iniciadas contextualmente a partir do painel "Rede". Uma vez ativado, o painel contabiliza e exibe o

número de solicitações interceptadas que correspondem a cada padrão definido (??).

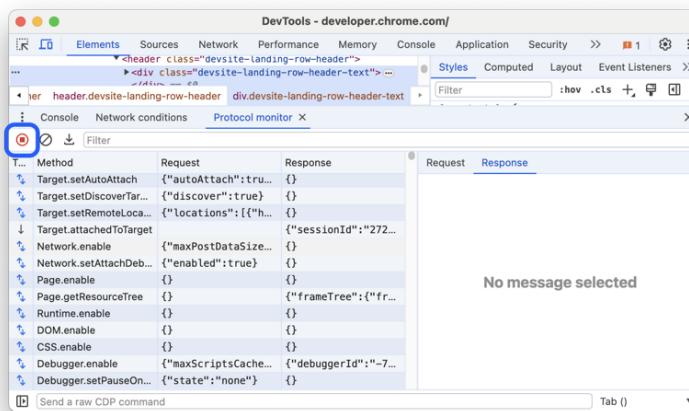
Figura 27 – Ferramenta "Monitor de Desempenho" do Chrome Devtools



Fonte: Chrome for Developers

O "Monitor de Desempenho" (*Performance Monitor*), como ilustrado na figura 27, é uma ferramenta de diagnóstico que fornece uma visualização em tempo real do desempenho de carregamento e execução de uma aplicação *web*. A ferramenta apresenta uma linha do tempo que plota graficamente diversas métricas, permitindo que estas sejam ativadas ou desativadas para análise (??). As métricas rastreadas incluem: uso da CPU, tamanho do *heap JavaScript*, o número total de nós *DOM*, *listeners* de eventos *JavaScript*, documentos e *frames*, bem como a frequência de *layouts* e recálculos de estilo por segundo (??). Além disso, o monitor persiste seus dados durante a navegação entre páginas, auxiliando na identificação de padrões de uso de recursos que podem indicar ineficiências de código ou problemas como *layout thrashing*.

Figura 28 – Ferramenta "Monitor de protocolo" do Chrome Devtools



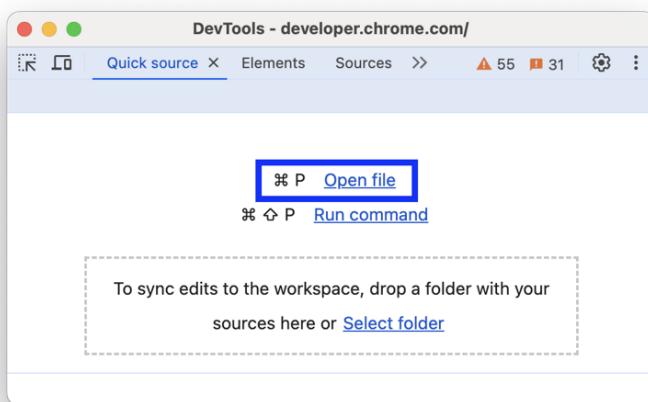
Fonte: Chrome for Developers

O "Monitor de protocolo" (*Protocol Monitor*), apresentado na figura 28, é uma ferramenta que permite a inspeção de todas as solicitações e respostas do Protocolo *Chrome DevTools* (CDP). Esta funcionalidade registra as mensagens do CDP e permite a análise detalhada dos dados de solicitação e resposta (??). A ferramenta viabiliza o envio direto de comandos brutos do CDP, suportando tanto comandos simples sem parâmetros quanto comandos complexos com parâmetros estruturados em *JSON*, auxiliado por um editor de

comandos dedicado (??). Adicionalmente, o monitor permite o *download* das mensagens registradas em formato *JSON* para análise externa.

O painel "Origem rápida"(*Quick source*), exibido na figura 29, opera como uma interface suplementar para visualização e edição de arquivos de origem. Sua principal utilidade reside na sua integração na "gaveta"(*drawer*), por padrão na parte inferior da janela do *DevTools*, o que permite ao desenvolvedor inspecionar e modificar o código-fonte enquanto mantém acesso simultâneo a outros painéis (??). Esta funcionalidade elimina a necessidade de alternar repetidamente entre a ferramenta "Fontes"(*Sources*) principal e outras ferramentas. O painel "Origem rápida" abre automaticamente o último arquivo editado na ferramenta "Fontes" e permite a abertura de outros arquivos através de atalhos de teclado (*Command+P* ou *Ctrl+P*), que são contextualmente redirecionados para este painel quando ele está em foco (??).

Figura 29 – Ferramenta "Origem rápida"do Chrome Devtools



Fonte: Chrome for Developers

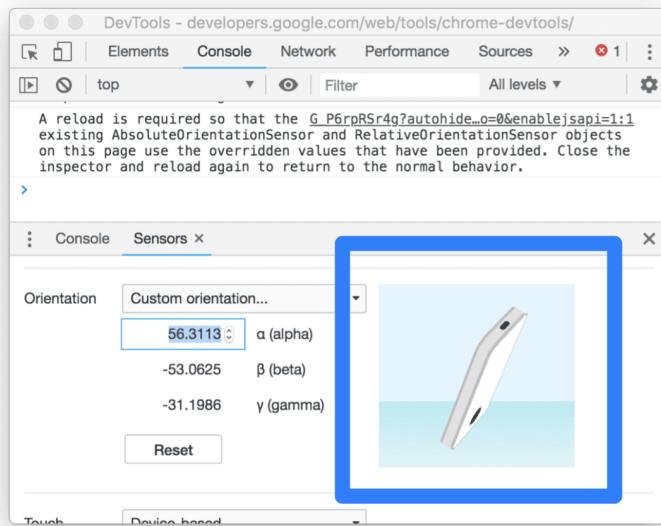
O painel "WebAudio", apresentado na figura 30, é uma ferramenta de diagnóstico que exibe métricas de desempenho para instâncias de *AudioContext* em aplicações que utilizam a API *WebAudio* (??). Após a seleção de um contexto de áudio específico, o painel apresenta métricas chave, incluindo o Estado (*State*) operacional (por exemplo, *running*, *suspended*), a Taxa de Amostragem (*Sample Rate*) em Hz, o Tamanho do Buffer de Callback (*Callback Buffer Size*) em quadros e o Número Máximo de Canais de Saída (*Max Output Channel Count*) (??). Adicionalmente, a ferramenta monitora em tempo real o Intervalo de Callback (*Callback Interval*) e a Capacidade de Renderização (*Render Capacity*) do processador de áudio. O painel "Sensores"(*Sensors*), ilustrado na figura 31, é uma ferramenta de emulação projetada para simular diversas entradas de *hardware* e estados do dispositivo. Suas principais funcionalidades incluem a substituição de dados de geolocalização, a simulação de orientação do dispositivo, a forçagem de eventos de toque e a emulação de estados da API *Idle Detection* (detector inativo) (??). Adicionalmente, a ferramenta viabiliza a substituição do valor de simultaneidade de *hardware* (*navigator.hardwareConcurrency*) e a simulação de estados da API *Compute Pressure* (pressão da *CPU*).

Figura 30 – Ferramenta "WebAudio" do Chrome Devtools



Fonte: Chrome for Developers

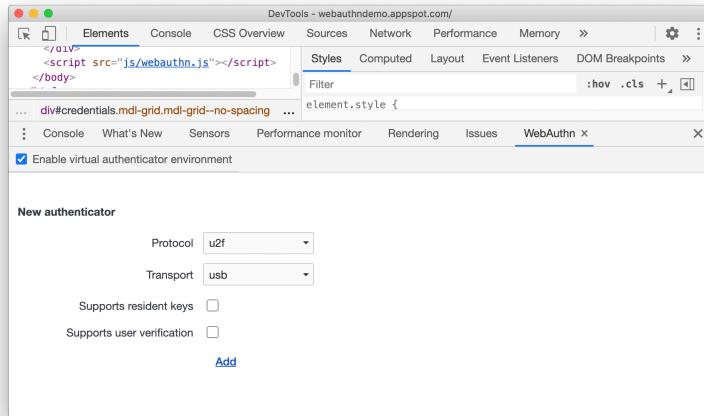
Figura 31 – Ferramenta "Sensores" do Chrome Devtools



Fonte: Chrome for Developers

O painel "WebAuthn", apresentado na figura 32, fornece uma interface para a criação e interação com autenticadores virtuais baseados em software, permitindo a depuração da API Web Authentication (??). A ferramenta possibilita a ativação de um ambiente de autenticador virtual, no qual desenvolvedores podem adicionar, renomear e remover autenticadores. É possível configurar o protocolo (por exemplo, ctap2, u2f) e o transporte (por exemplo, USB, NFC) de cada autenticador, além de registrar credenciais e monitorar seus IDs e contagens de assinaturas durante os testes (??).

Figura 32 – Ferramenta "WebAuthn"do Chrome Devtools

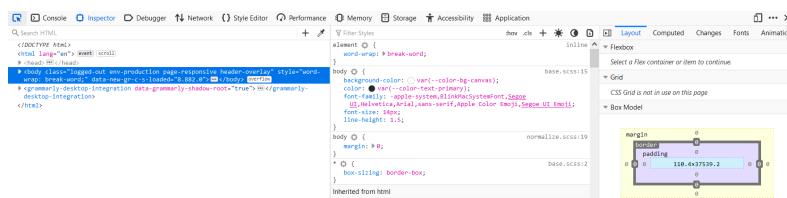


Fonte: Chrome for Developers

2.2 Mozilla Firefox

O navegador Mozilla Firefox integra um conjunto abrangente de ferramentas de desenvolvimento web, centralizadas na interface denominada "Toolbox"(Caixa de Ferramentas). Esta suíte é projetada para permitir que desenvolvedores inspecionem, depurem, monitorem e modifiquem aplicações web diretamente no navegador. A "Toolbox" agrupa a maioria das ferramentas de desenvolvedor incorporadas e pode ser acessada através do menu de contexto (Inspecionar), do menu principal do navegador ou por atalhos de teclado. As subseções seguintes detalham os componentes individuais desta suíte de ferramentas.

Figura 33 – Ferramenta "Inspetor"do Firefox Devtools

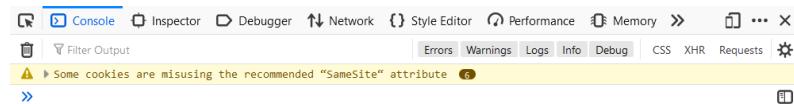


Fonte: Firefox Source Tree Documentation

O Inspetor de Página (Page Inspector), como apresentado na figura 33, é um componente central das ferramentas de desenvolvedor do Firefox, concebido para a inspeção e modificação em tempo real da estrutura HTML e das folhas de estilo (CSS) de um documento web (??). A ferramenta permite a análise e manipulação direta do Document Object Model (DOM), a edição de propriedades CSS, a avaliação do modelo de caixa (box model), e a depuração de layouts complexos, incluindo CSS Grid e Flexbox (??). Suas funcionalidades estendem-se à inspeção de manipuladores de eventos (event listeners), análise de animações, edição de fontes e seleção de cores, operando tanto em instâncias locais do navegador quanto em alvos de depuração remotos.

O Console Web (Web Console), exibido na figura 34, é um componente das ferramentas de desenvolvedor do Firefox que opera como uma interface interativa para registro e depuração. Sua principal função é registrar informações detalhadas associadas a uma página web, incluindo requisições de rede, erros e advertências de JavaScript, CSS e segurança, além de mensagens de erro, aviso e informacionais explicitamente registradas pelo código JavaScript executado

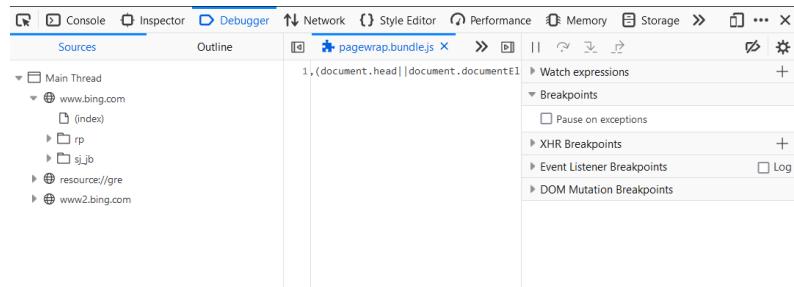
Figura 34 – Ferramenta "Console" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

no contexto da página (??). Adicionalmente, a ferramenta viabiliza a interação direta com o documento, permitindo a execução de expressões JavaScript no contexto da página para fins de análise e manipulação dinâmica.

Figura 35 – Ferramenta "Depurador" do Firefox Devtools



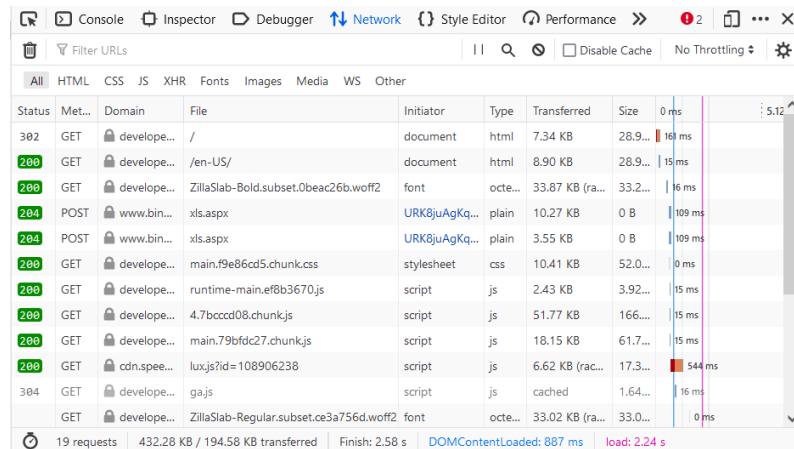
Fonte: Firefox Source Tree Documentation

O Depurador JavaScript (JavaScript Debugger), como ilustrado na figura 35, é uma ferramenta de diagnóstico que permite a análise e retificação de código JavaScript, possibilitando a execução passo a passo (step-through) e a inspeção ou modificação do estado do script para facilitar a identificação de bugs (??). A ferramenta é capaz de depurar código executado localmente no Firefox ou em alvos remotos, como o Firefox for Android. Suas funcionalidades centrais incluem a habilidade de pausar a execução em pontos específicos através de breakpoints, breakpoints condicionais, pontos de interrupção em XHR, listeners de eventos, exceções ou mutações do DOM (??). Adicionalmente, oferece controle sobre o fluxo de execução, inspeção de valores, suporte a source maps, formatação de código minificado (pretty-printing) e depuração de worker threads.

O Monitor de Rede (Network Monitor), exibido na figura 36, é a ferramenta designada para exibir todas as requisições HTTP que o Firefox executa, incluindo carregamentos de página e XMLHttpRequests. Esta funcionalidade permite a análise da duração e dos detalhes de cada transação. A ferramenta registra a atividade de rede continuamente enquanto a 'Toolbox' (Caixa de Ferramentas) estiver aberta, independentemente de o painel de Rede estar ativamente selecionado (??). Suas capacidades estendem-se à inspeção de Web Sockets, Server-Sent Events (SSE) e à simulação de diferentes condições de rede (throttling) (??).

O Editor de Estilo (Style Editor) é um componente das ferramentas de desenvolvedor do Firefox que facilita a inspeção e manipulação das folhas de estilo (stylesheets) associadas a um documento web. A ferramenta permite a edição em tempo real do CSS existente, com alterações aplicadas imediatamente à página, bem como a criação de novos stylesheets ou a importação de folhas de estilo externas (??). Sua interface inclui um painel de editor de código e uma barra lateral dedicada à navegação de At-rules, como @media, @supports, @layer e @container. Além disso, o Editor de Estilo oferece suporte a "source maps" (mapas de origem),

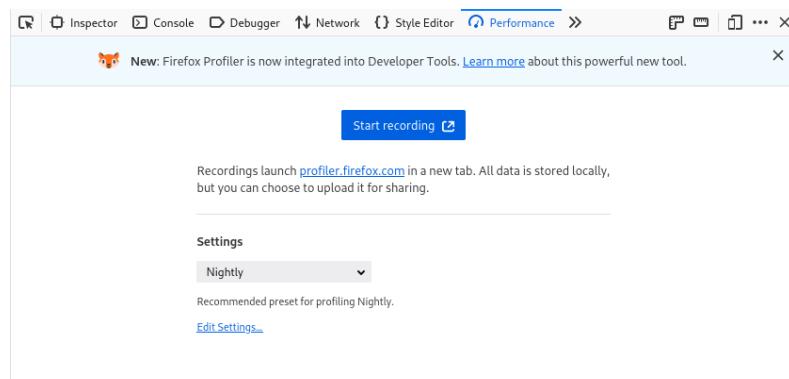
Figura 36 – Ferramenta "Rede" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

permitindo que desenvolvedores depurem o código-fonte original (ex: Sass, Less) em vez do CSS transpilado ou minificado (??).

Figura 37 – Ferramenta "Desempenho" do Firefox Devtools



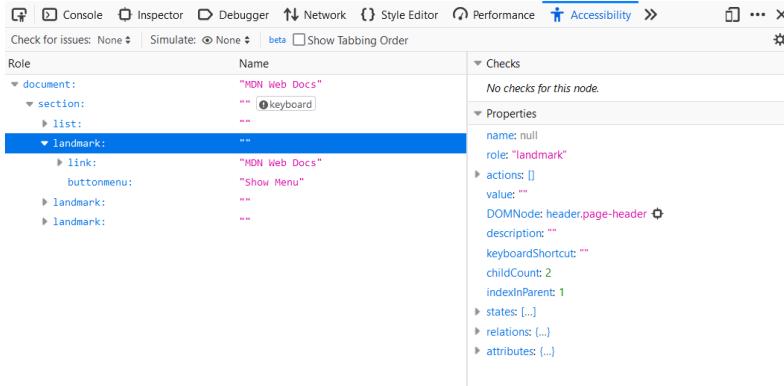
Fonte: Firefox Source Tree Documentation

O painel Desempenho (Performance Panel), como apresentado na figura 37, é a ferramenta designada para a análise da responsividade geral, da execução de JavaScript e do desempenho de layout de uma aplicação web (??). Sua operação baseia-se na captura de um perfil de desempenho, que é subsequentemente carregado na interface de usuário do Firefox Profiler, uma aplicação web externa, para inspeção detalhada (??). Os dados capturados permanecem armazenados localmente, sendo que a ferramenta facilita o upload e compartilhamento desses perfis para análise colaborativa.

O Inspetor de Acessibilidade (Accessibility Inspector), exibido na figura 38, é um componente das ferramentas de desenvolvedor do Firefox que fornece acesso à árvore de acessibilidade (accessibility tree) da página (??). Sua finalidade é permitir a inspeção da estrutura semântica exposta às tecnologias assistivas, possibilitando a verificação de atributos, a identificação de elementos ausentes ou a avaliação de componentes que necessitem de atenção para garantir a conformidade (??).

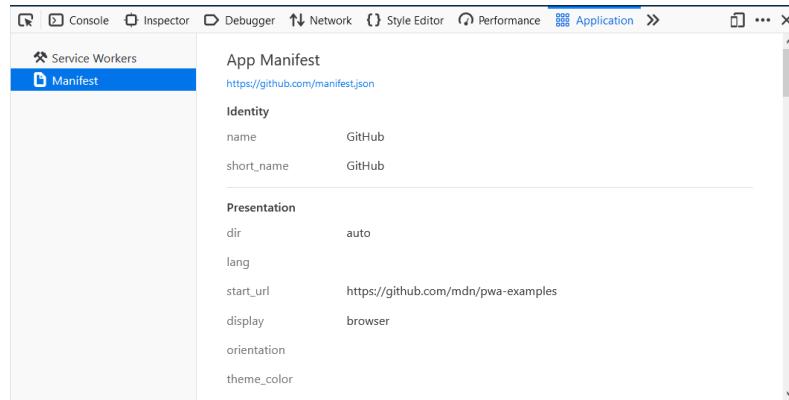
O painel Aplicação (Application panel), apresentado na figura 39, é um componente das ferramentas de desenvolvedor do Firefox que fornece instrumentos para a inspeção e depuração de aplicações web modernas, comumente designadas como Progressive Web Apps

Figura 38 – Ferramenta "Inspetor de Acessibilidade" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

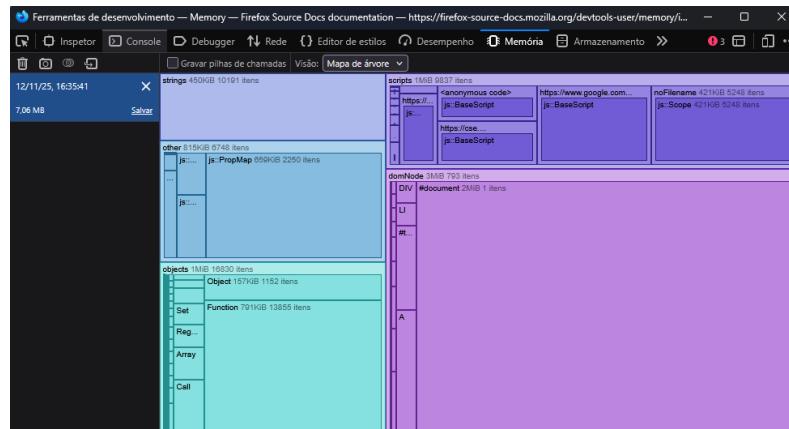
Figura 39 – Ferramenta "Aplicações" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

(PWAs) (??). Suas capacidades incluem especificamente a inspeção de service workers e de manifestos de aplicações web (web app manifests) (??).

Figura 40 – Ferramenta "Memória" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

A ferramenta Memória (Memory tool), exibida na figura 40, destina-se à análise do uso de memória, permitindo a captura de um instantâneo (snapshot) do heap de memória do separador atual. Esta funcionalidade fornece múltiplas visualizações do heap, que demonstram quais objetos são responsáveis pelo consumo de memória e os locais exatos no código

onde as alocações estão ocorrendo. As visualizações disponíveis incluem "Tree map"(mapa de árvore), "Aggregate"(agregada) — que apresenta uma tabela de tipos alocados — e "Dominators"(dominadores) (??). A visão "Dominator" é notável por exibir o "tamanho retido"(retained size) dos objetos, definido como o tamanho do próprio objeto acrescido do tamanho de outros objetos que ele mantém vivos através de referências (??). Adicionalmente, a ferramenta suporta a comparação entre múltiplos instantâneos e o registro de pilhas de chamadas (call stacks) para rastrear a origem das alocações.

Figura 41 – Ferramenta "Inspetor de Armazenamento"do Firefox Devtools

The screenshot shows the Firefox DevTools Storage Inspector for the domain https://www.facebook.com. The table lists various cookies with columns for Name, Domain, Path, Expires on, Last accessed on, and Value. Key entries include 'act' (Facebook.com), 'c_user' (Facebook.com), 'datr' (Facebook.com), 'fr' (Facebook.com), 'pnt_data2' (Facebook.com), 'presence' (Facebook.com), 'sb' (Facebook.com), 'spin' (Facebook.com), 'wd' (Facebook.com), 'x-src' (Facebook.com), and 'xs' (Facebook.com). The 'Expires on' column shows dates like Tue, 23 Mon, 26 Sun, 23 etc. The 'Last accessed on' column shows times like 0243... Mon, 26 193... Mon, 01 12:59... Sun, 23 etc. The 'Value' column contains JSON-like strings representing cookie data.

Name	Domain	Path	Expires on	Last accessed on	Value
act	Facebook.com	/	Session	0243...	false
c_user	Facebook.com	/	Tue, 23	1	false
datr	Facebook.com	/	Mon, 26	Kovh...	true
fr	Facebook.com	/	Tue, 23	y9kvw...	true
pnt_data2	Facebook.com	/	Sun, 23	zShZ...	false
presence	Facebook.com	/	Session	beF15...	false
sb	Facebook.com	/	Mon, 26	Value	true
spin	Facebook.com	/	Mon, 24	table.headers.cookies.hostOnly	193...
wd	Facebook.com	/	Mon, 01	table.headers.cookies.isSecure	false
x-src	Facebook.com	/	Sun, 23	table.headers.cookies.isHttpOnly	Jhaw...
xs	Facebook.com	/	Tue, 23	sameSite	ZUk...

Fonte: Firefox Source Tree Documentation

O Inspetor de Armazenamento (Storage Inspector), ilustrado na figura 41, é a ferramenta designada para a inspeção dos diversos tipos de armazenamento que uma página web pode utilizar. Atualmente, o inspetor suporta a análise de Cache Storage (caches DOM criados através da Cache API), Cookies (criados pela página ou por iframes nela contidos), bancos de dados IndexedDB (incluindo seus Object Stores e os itens neles armazenados), Local Storage e Session Storage (??). A interface organiza os objetos de armazenamento hierarquicamente por origem e, no presente momento, fornece uma visualização estritamente de leitura (read-only) dos dados armazenados (??).

Figura 42 – Ferramenta "Visualizador de Propriedades DOM"do Firefox Devtools

The screenshot shows the Firefox DevTools DOM Property Viewer. The left sidebar lists properties of the window object, including close, closed, confirm, console, content, \$, \$strings, \$0, context, length, selector, GoogleAnalyticsObject, and InstallTrigger. The right pane displays the values for these properties, such as close() for close, false for closed, and a confirmation dialog for confirm(). The content property is expanded to show its length and selector.

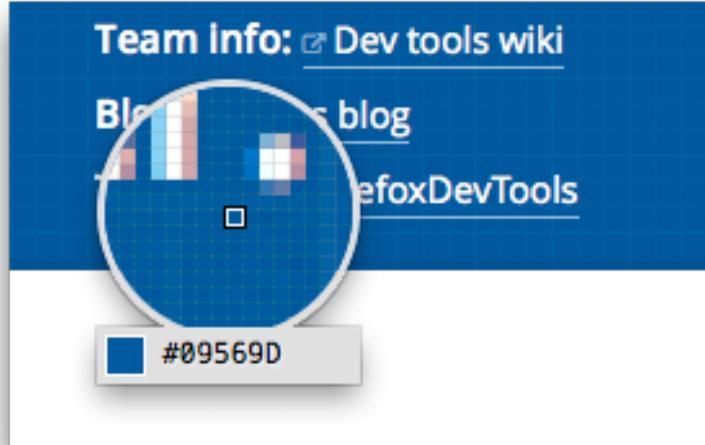
Property	Value
close	close()
closed	false
confirm	confirm()
console	{}
content	<pre>m() { length: 1, selector: "#strings", context: /en-US/firefox/nightly/firstrun/ } {} /en-US/firefox/nightly/firstrun/ 1 "#strings" "ga" {}</pre>
\$	m()
\$strings	{ length: 1, selector: "#strings", context: /en-US/firefox/nightly/firstrun/ }
\$0	{}
context	/en-US/firefox/nightly/firstrun/
length	1
selector	"#strings"
GoogleAnalyticsObject	"ga"
InstallTrigger	{}

Fonte: Firefox Source Tree Documentation

O Visualizador de Propriedades DOM (DOM Property Viewer), apresentado na figura 42, é uma ferramenta de inspeção que facilita a análise das propriedades do Document Object Model (DOM). A ferramenta apresenta essas propriedades em uma estrutura de árvore expansível, iniciando a inspeção a partir do objeto window global da página corrente ou de um iframe selecionado (??). A interface exibe o nome de cada propriedade e seu valor correspondente, indicando visualmente propriedades não graváveis (non-writable) e permitindo a filtragem da lista para localizar itens específicos (??). A exibição pode ser atualizada manualmente para refletir alterações no DOM.

O Conta-gotas (Eyedropper), exibido na figura 43, é uma ferramenta que permite a seleção de cores da página web atual, operando como uma lupa para seleção com precisão de

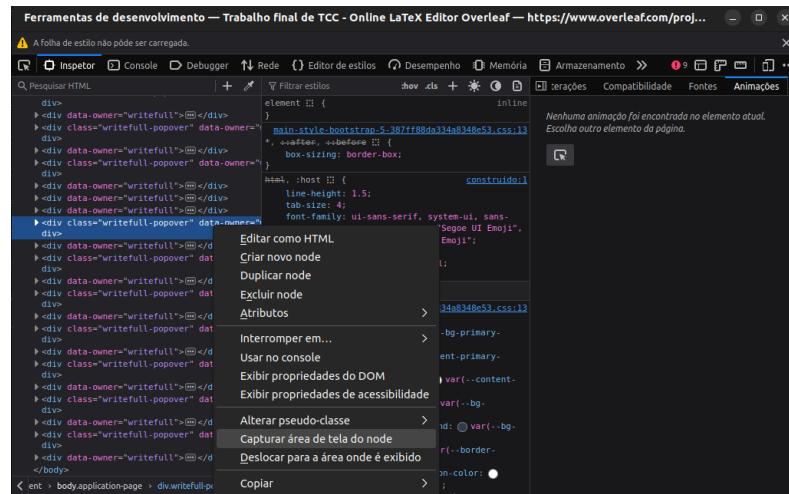
Figura 43 – Ferramenta "Conta-gotas" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

pixel e exibindo o valor de cor do pixel corrente. A ferramenta possui duas utilizações principais: selecionar uma cor da página para copiá-la para a área de transferência, ou modificar um valor de cor diretamente na visualização de Regras (Rules view) do Inspetor (??). Esta segunda funcionalidade é acessada através do seletor de cores (color picker) da visualização de Regras, onde a ativação do ícone do conta-gotas permite que a cor selecionada na página atualize a propriedade CSS correspondente (??).

Figura 44 – Ferramenta "Captura de Tela" do Firefox Devtools

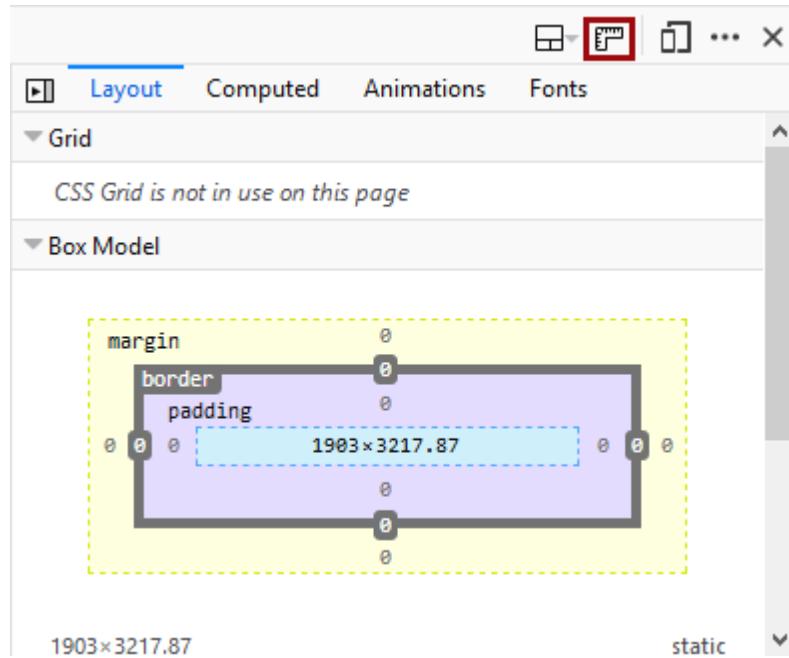


Fonte: Firefox Source Tree Documentation

A funcionalidade de Captura de Tela (Screenshot), apresentada na figura 44, permite a geração de imagens da página web renderizada, abrangendo tanto a captura da página inteira (full-page screenshot) quanto a de um nó (node) específico do DOM. A ferramenta é invocada através de um ícone opcional na barra de ferramentas, pelo menu de contexto de um elemento no Inspetor de HTML, ou programaticamente pelo Console Web (??). A partir da versão 62 do Firefox, a função auxiliar :screenshot do console facilita controle granular, permitindo a definição de atrasos (delay), a especificação de um seletor CSS para o alvo, a alteração da razão de pixels do dispositivo (DPR) e a nomeação do arquivo de saída. Por padrão, as imagens

são salvas no diretório de "Downloads", podendo alternativamente ser copiadas para a área de transferência (??).

Figura 45 – Ferramenta "Régua" do Firefox Devtools



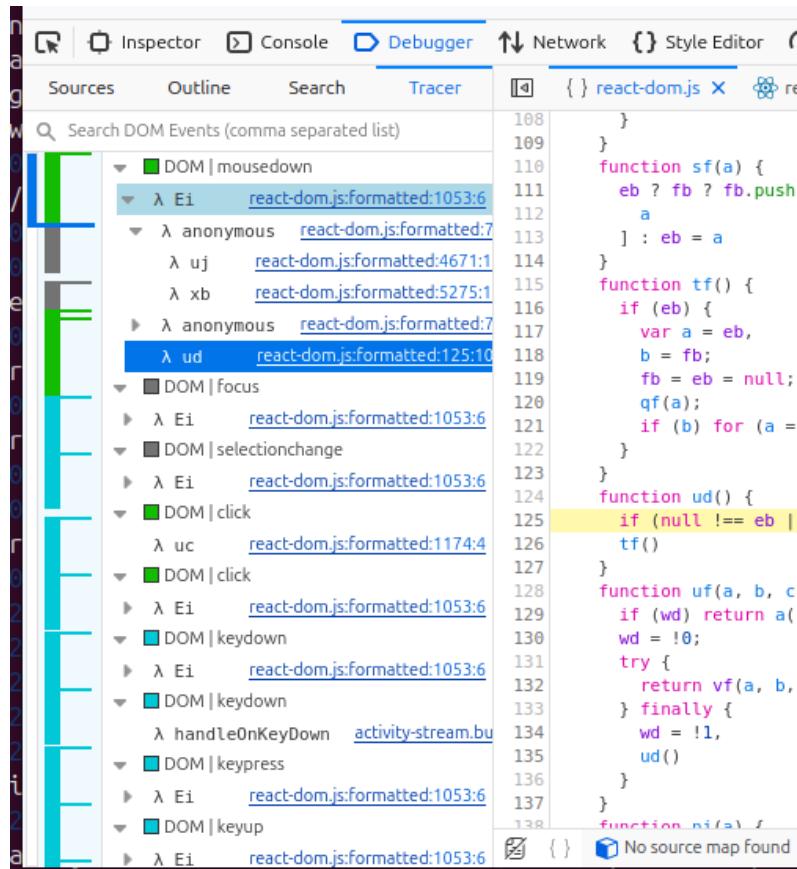
Fonte: Firefox Source Tree Documentation

A ferramenta Régulas (Rulers), exibida na figura 45, proporciona a sobreposição de guias dimensionais, especificamente réguas horizontais e verticais, diretamente sobre a página web renderizada (??). As unidades de medida utilizadas são pixels. Adicionalmente, a ferramenta exibe as dimensões atuais do viewport (área de visualização) próximo ao canto superior direito. Esta funcionalidade não é persistente, exigindo que o comando seja reativado após cada atualização da página ou navegação (??).

Os Formatadores Personalizados (Custom Formatters), ilustrado na figura ??, são uma funcionalidade que permite a um website controlar a renderização de variáveis JavaScript dentro do Console Web e do Depurador, visando aprimorar o processo de depuração ao fornecer uma representação mais intuitiva de objetos complexos (??). A implementação é realizada através da definição de um array global denominado devtoolsFormatters, cujos elementos são objetos formatadores. Cada formatador deve conter uma função header e, opcionalmente, funções hasBody e body (??). Essas funções retornam null ou um array baseado no padrão JsonML para construir a interface HTML customizada, suportando a referência de objetos aninhados através de um template de objeto específico.

O Traçador JavaScript (JavaScript Tracer), exibido na figura 46, é uma ferramenta experimental do Firefox, ativada através da preferência devtools.debugger.features.javascript-tracing, concebida para registrar todas as chamadas de função JavaScript. A ferramenta permite a configuração de múltiplos destinos de saída (logging output), incluindo o Console Web, uma barra lateral no Depurador, um registro no Firefox Profiler ou a saída padrão (stdout) (??). Suas opções permitem um início de rastreamento atrasado (delayed start), condicionado à interação do usuário ou ao carregamento da página, e pode opcionalmente capturar retornos de função, valores de argumentos, mutações do DOM, bem como operar com limites de profundidade de pilha (depth limit) ou de número de registros (record limit) (??).

Figura 46 – Ferramenta "Traçador JavaScript" do Firefox Devtools



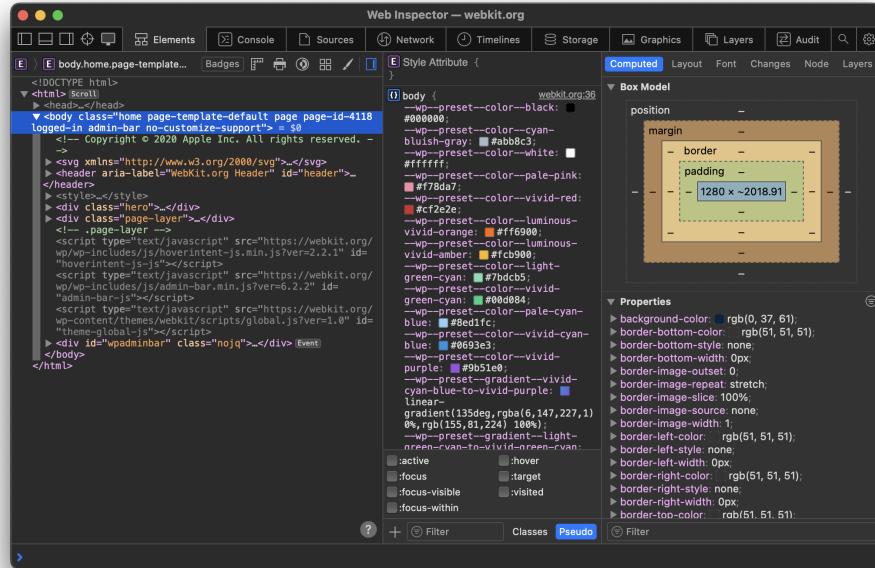
Fonte: Firefox Source Tree Documentation

2.3 Apple Safari

O navegador Apple Safari, baseado no engine WebKit, fornece um conjunto de ferramentas de desenvolvimento integradas, essenciais para a depuração, profiling de performance e automação de aplicações web. O componente central dessa suíte é o Web Inspector, uma ferramenta de diagnóstico que oferece introspecção granular em tempo real do estado da página. As subseções a seguir detalham as funcionalidades de cada aba primária do Web Inspector — abrangendo desde a manipulação do DOM ("Elementos") e execução de scripts ("Console"), até a depuração de código ("Fontes"), análise de tráfego ("Rede"), profiling ("Linhas do Tempo"), inspeção de dados ("Armazenamento"), análise de renderização ("Gráficos" e "Camadas") e testes de conformidade ("Auditoria"). A seção conclui com uma análise do "WebDriver", o framework da Apple que implementa o protocolo W3C para a automação de testes de ponta a ponta.

A aba **Elementos** (*Elements*), apresentada na figura 47, constitui a interface primária para a inspeção e manipulação em tempo real do Document Object Model (DOM) de uma página. Ela apresenta uma visualização interativa da árvore DOM, permitindo a edição direta de nós, atributos e conteúdo textual, ao mesmo tempo que identifica nós não renderizados e destaca elementos com layouts específicos, como CSS Grid ou Flexbox, através de emblemas (*badges*) (??). Uma barra lateral de detalhes complementa a árvore, oferecendo ferramentas granulares para depuração: o painel "Styles" (Estilos) permite a análise e modificação de regras CSS, organizadas por especificidade e herança, com editores especializados; o painel "Computed" (Computado) exibe os valores CSS finais aplicados e visualiza o box model; e painéis adicionais facilitam a inspeção de layout (Grid/Flexbox), tipografia (incluindo fontes variáveis), e dados

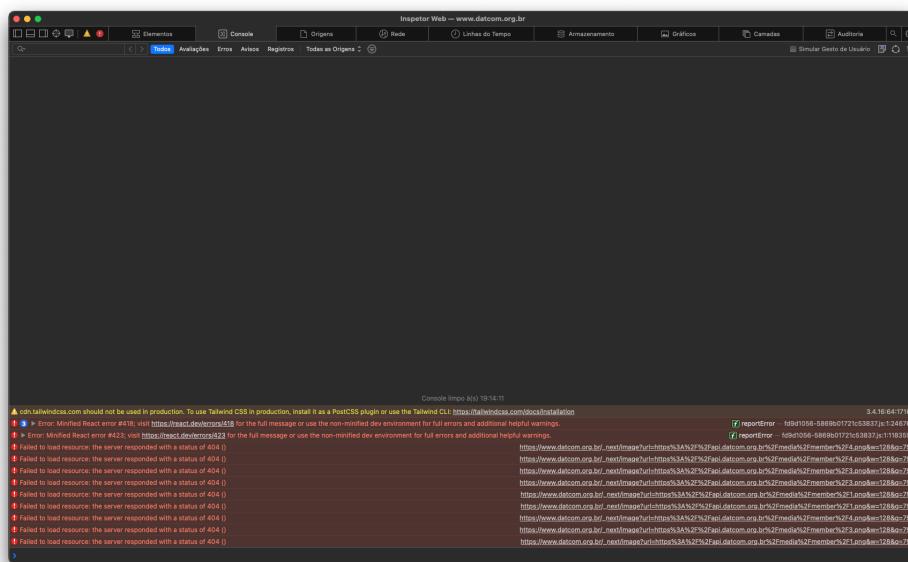
Figura 47 – Ferramenta "Elementos" do Safari Web Inspector



Fonte: Apple Developer Documentation

de acessibilidade (??).

Figura 48 – Ferramenta "Console" do Safari Web Inspector



Fonte: Apple Developer Documentation

A aba **Console** (*Console*), exibida na figura 48, opera como um hub de diagnóstico multifacetado, funcionando simultaneamente como uma interface de linha de comando (CLI) para a execução interativa de JavaScript e como um sistema de registro (*logging*) abrangente. Sua utilidade primária reside na capacidade de interagir em tempo real com o ambiente JavaScript da página, permitindo a depuração interativa, a modificação dinâmica de variáveis de tempo de execução e a recuperação imediata de informações de estado (??). Além da execução

de código, o Console é crítico para o relatório de erros, avisos e mensagens informacionais, oferecendo um fluxo de trabalho eficiente onde a seleção de um erro pode navegar diretamente para o código-fonte correspondente (??). Além disso, a ferramenta oferece funcionalidades de monitoramento de rede para rastrear atividades e durações de requisições e a inspeção de Event Listeners, fornecendo, assim, uma visão holística da saúde da página (??).

Figura 49 – Ferramenta "Fontes" do Safari Web Inspector

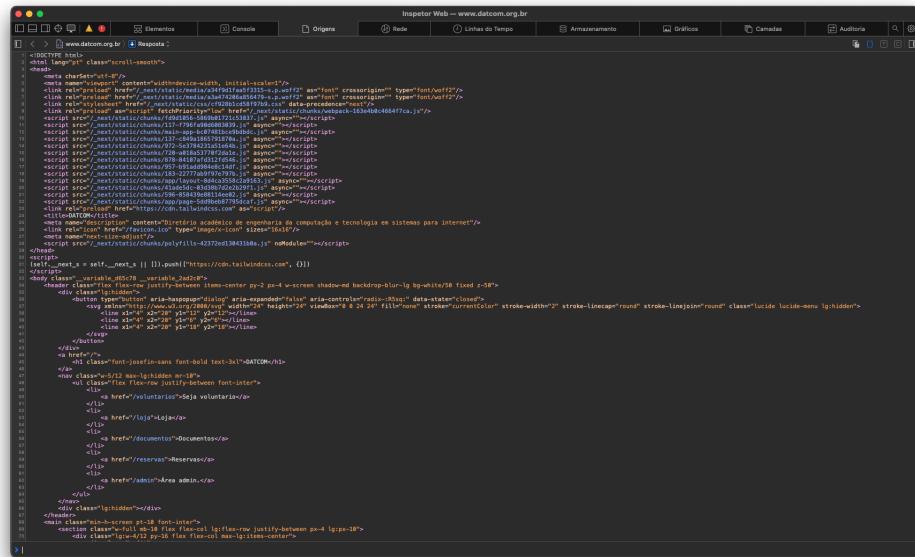
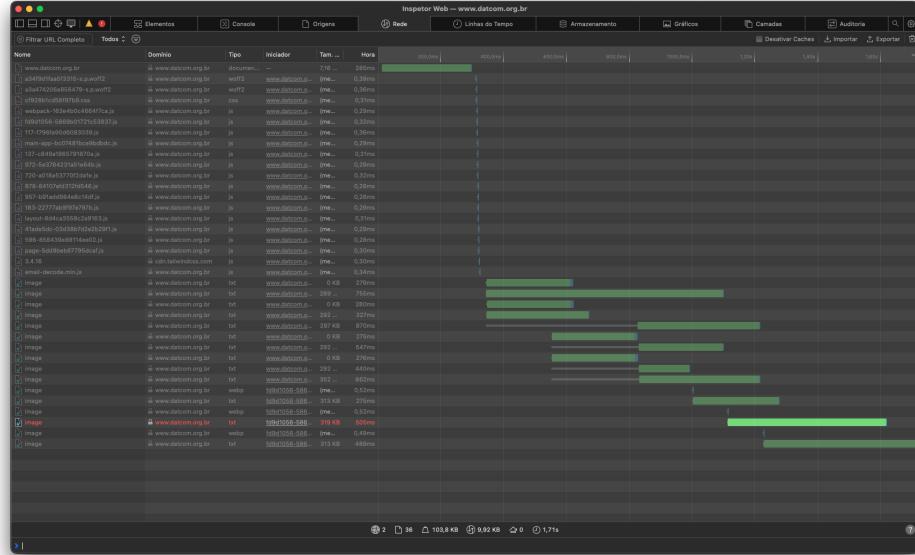
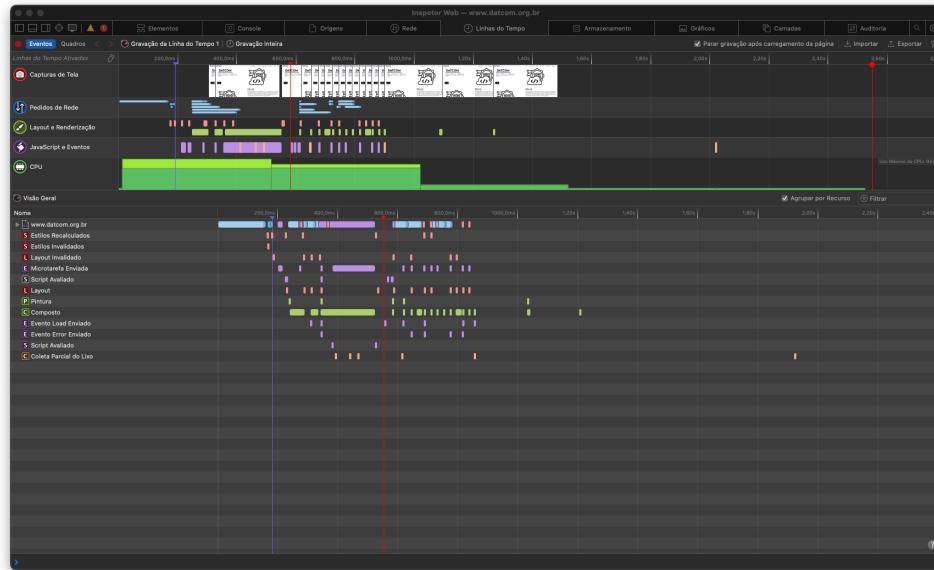


Figura 50 – Ferramenta "Rede" do Safari Web Inspector



Fonte: Apple Developer Documentation

Figura 51 – Ferramenta "Linha do Tempo" do Safari Web Inspector

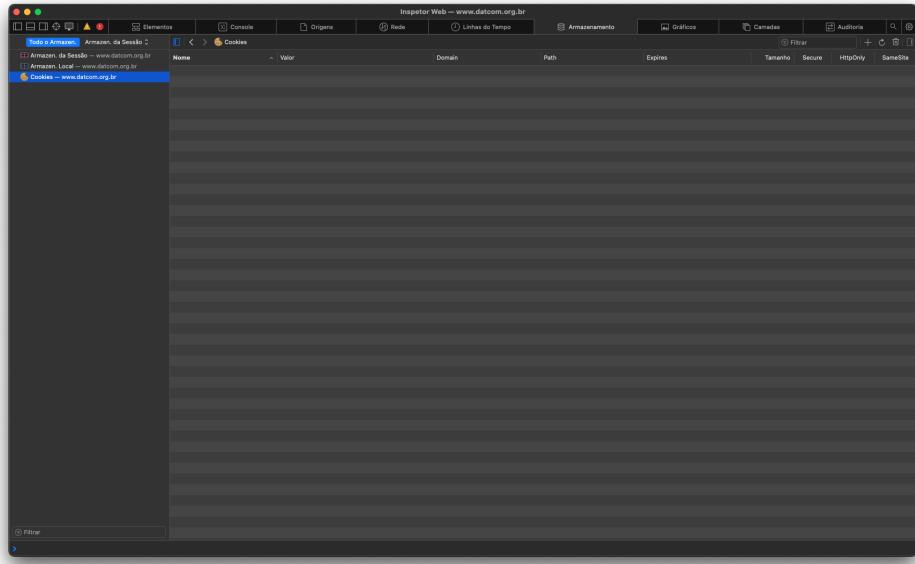


Fonte: Apple Developer Documentation

da gravação de atividades da página, desabilitando temporariamente recursos de depuração, como otimizações JIT e breakpoints, para assegurar medições realistas do desempenho. Os dados capturados são organizados principalmente na “Events View” (Visualização de Eventos) e na “Frames View” (Visualização de Quadros) (??). A “Events View” plota a atividade em um gráfico geral e a segmenta em linhas do tempo especializadas, como “Network Requests”, “Layout & Rendering” (para recálculos de estilo, layout e *paint*), “JavaScript & Events” (que gera árvores de chamadas ou *call trees*), “CPU” e “Memory” (incluindo a captura de *heap snapshots* via “JavaScript Allocations”). Em contrapartida, a “Frames View” agrupa todas as atividades pelo frame de renderização em que ocorreram, analisando o tempo de execução de

cada quadro e comparando-o com os benchmarks de 30 e 60 FPS (??).

Figura 52 – Ferramenta "Armazenamento" do Safari Web Inspector



Fonte: Apple Developer Documentation

A aba **Armazenamento** (*Storage*), apresentada na figura 52, fornece uma inspeção detalhada e capacidades de gerenciamento para os diversos mecanismos de armazenamento de dados do lado do cliente. Ela cataloga dados de Application Cache, cookies, bancos de dados (como Web SQL e IndexedDB), *local storage* e *session storage* (??). A ferramenta permite não apenas a visualização de valores, metadados de cookies (como domínio e expiração) e o uso de espaço, mas também a modificação e exclusão ativas de entradas. Essa funcionalidade é crucial para a depuração de problemas de persistência de dados, como conteúdo obsoleto (*stale content*), gerenciamento de estados de usuário (transitórios e persistentes), e a verificação de aplicações *stateful* ou com capacidades offline (??).

A aba **Gráficos** (*Graphics*), exibida na figura 53, é uma ferramenta especializada, projetada para fluxos de trabalho técnicos e criativos, que oferece funcionalidades para a inspeção, pré-visualização e manipulação de elementos gráficos, animações e conteúdo do elemento HTML5 `<canvas>` (??). Ela provê capacidades de análise de assets de imagem, detalhando dimensões, tamanho de arquivo e formato para fins de otimização, e permite a inspeção profunda e edição direta da estrutura e atributos de Scalable Vector Graphics (SVG) (??). Adicionalmente, a ferramenta suporta a inspeção de formatos de imagem modernos como WebP e AVIF, a modificação de atributos do `<canvas>` (ex: largura, altura), e a pré-visualização de keyframes de animações ("Animation Preview") oriundas de CSS, JavaScript ou canvas (??).

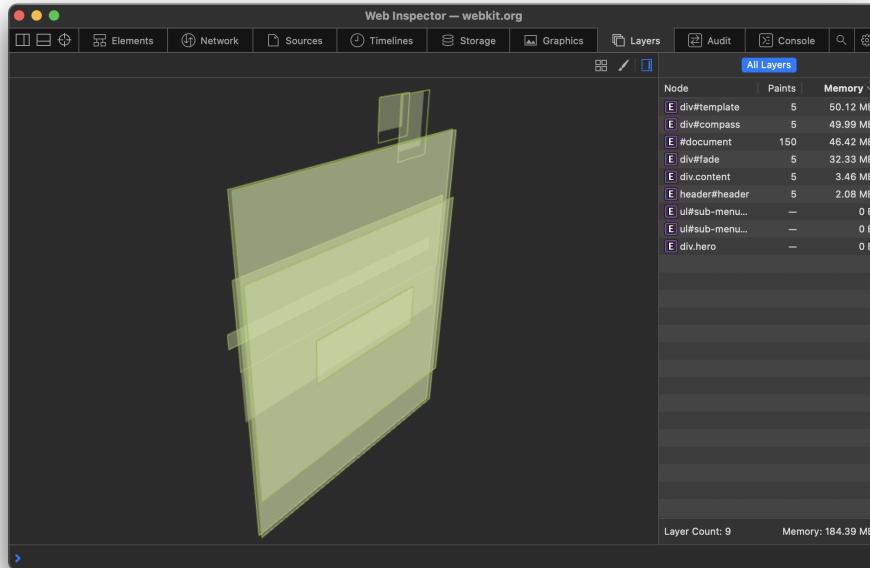
A aba **Camadas** (*Layers*), apresentada na figura 54, fornece uma interface para a análise da performance de renderização, visualizando o processo de *compositing* (composição). Ela exibe como os nós DOM, após o cálculo de layout, são desenhados em superfícies distintas (camadas) que são subsequentemente compostas para formar a visualização final da página. Esse mecanismo é crítico, pois o isolamento de um elemento em sua própria camada permite animações através de simples recomposição, em vez de exigir um *repaint* (repintura) completo da página, embora essa otimização resulte em um custo de memória (??). A ferramenta apresenta uma visualização 3D interativa da árvore de camadas e um painel lateral ("All Layers") que detalha o custo de memória, a contagem de *paints* e as razões específicas que justificaram a criação de cada camada (??). Adicionalmente, inclui diagnósticos visuais, como

Figura 53 – Ferramenta "Gráficos" do Safari Web Inspector



Fonte: Apple Developer Documentation

Figura 54 – Ferramenta "Camadas" do Safari Web Inspector

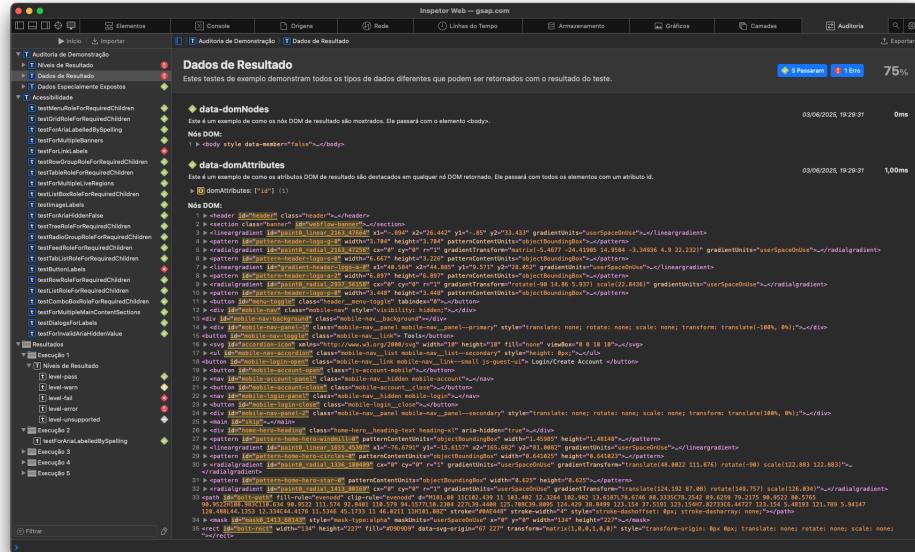


Fonte: Apple Developer Documentation

"Show compositing borders" (Exibir bordas de composição) e "Enable paint flashing" (Habilitar flash de pintura), para identificar atividade de *repaint* e os limites das camadas (??).

A aba **Auditória** (*Audit*), apresentada na figura 55, fornece um framework para a execução de coleções de testes automatizados contra a página inspecionada, projetados para avaliar a estrutura do DOM, a conformidade de atributos de acessibilidade (conforme especificações como WAI-ARIA) e a aderência a regras de *design system* (??). Cada auditoria é definida como um grupo de testes ou um caso de teste individual, consistindo em um snippet de JavaScript executado no contexto da página. Após a execução, os resultados são classificados

Figura 55 – Ferramenta "Auditoria" do Safari Web Inspector



Fonte: Apple Developer Documentation

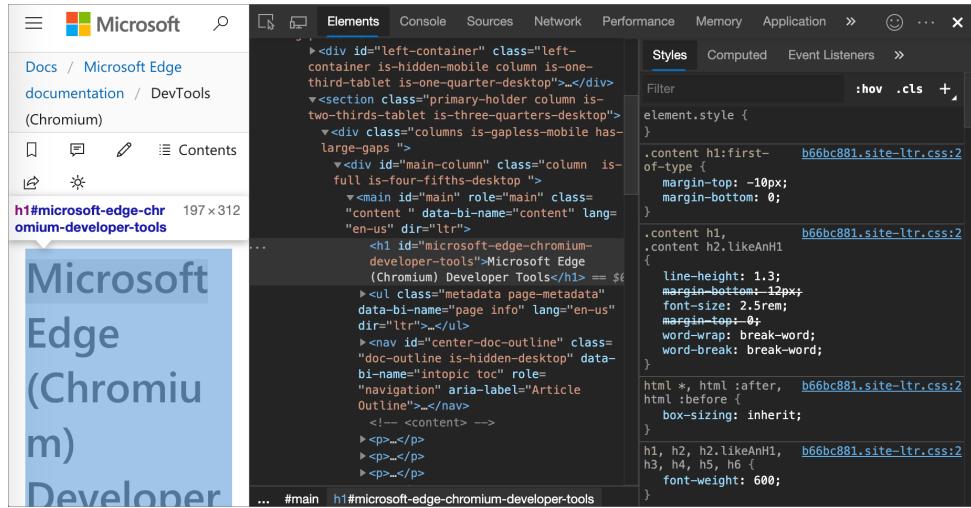
como *Pass* (Aprovado), *Warning* (Aviso), *Failed* (Falha), *Error* (Erro) ou *Unsupported* (Não Suportado) (??). A ferramenta permite a criação e modificação de auditorias personalizadas através de uma estrutura JSON, suportando funções de teste assíncronas que podem retornar objetos de resultado complexos, incluindo referências a `domNodes`. Notavelmente, as funções de teste recebem acesso a um objeto `WebInspectorAudit` especial, que expõe APIs para consultar recursos da página e dados da árvore de acessibilidade que não são acessíveis ao JavaScript padrão (??).

O **WebDriver** para Safari fornece a implementação da Apple do protocolo W3C WebDriver, permitindo a automação de testes de ponta a ponta para o conteúdo web. A arquitetura é mediada pelo executável `safaridriver`, que recebe comandos REST API de bibliotecas de cliente, como o Selenium, e os encaminha para a instância do navegador (??). Para garantir a integridade do teste e prevenir contaminação de estado, a execução é confinada a “Janelas de Automação Isoladas” (*Isolated Automation Windows*), que operam em um modo efêmero, similar à navegação privada, isolado do histórico, cookies e preferências do usuário (??). Além disso, um painel transparente (“Glass Pane”) é instalado sobre a janela para interceptar e neutralizar interações de usuário (mouse ou teclado) que poderiam comprometer a execução do teste. Notavelmente, o WebDriver mantém integração total com o Web Inspector, permitindo que ferramentas de depuração, como o Console e o depurador de scripts, sejam utilizadas simultaneamente à execução dos testes automatizados (??).

2.4 Microsoft Edge

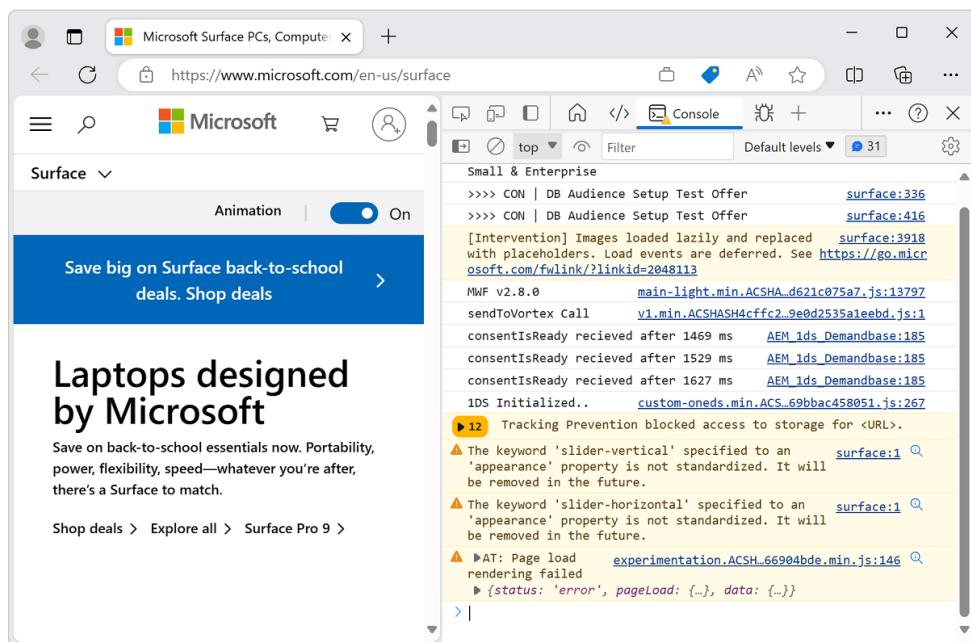
O painel **Elementos** (Elements), exibido na figura 56, oferece uma interface robusta para a inspeção e manipulação do Document Object Model (DOM) e das Folhas de Estilo em Cascata (CSS) em tempo real. Ele proporciona uma representação interativa da árvore DOM, refletindo o estado dinâmico da aplicação, que pode divergir do HTML fonte original devido a manipulações via JavaScript (??). A ferramenta facilita a depuração de layout através da edição direta de atributos HTML, conteúdo textual e propriedades CSS, permitindo ainda a simulação de pseudo-estados (ex: `:hover`) e a análise do modelo de caixa (box model) (??).

Figura 56 – Ferramenta "Elementos" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 57 – Ferramenta "Console" do Microsoft Edge Devtools

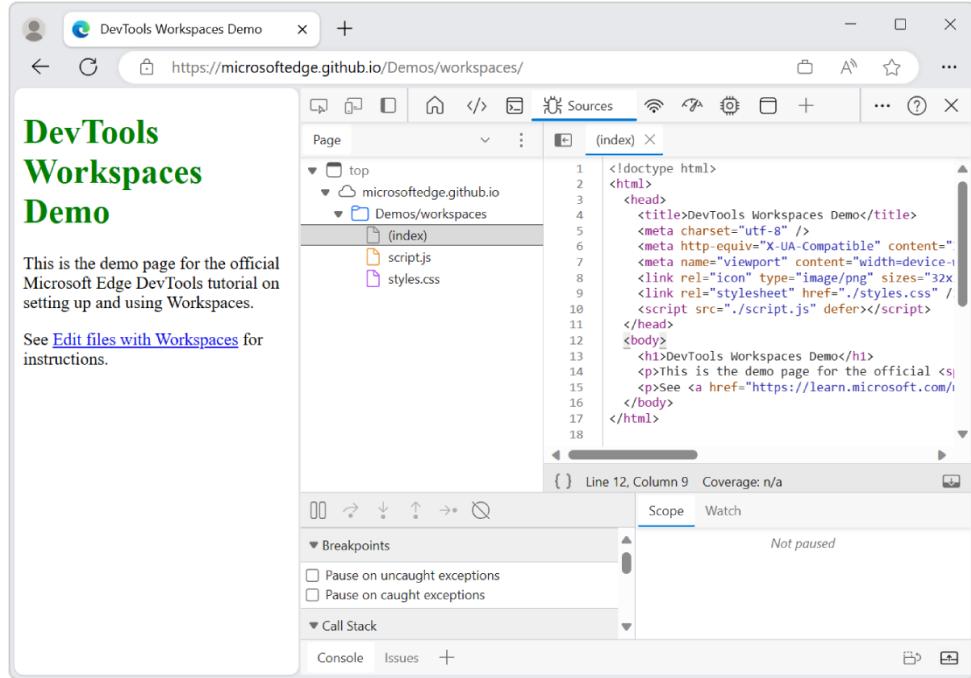


Fonte: Microsoft Learn

O **Console** (Console), apresentado na figura 57, atua como um ambiente interativo de Read-Eval-Print Loop (REPL) para a execução de scripts JavaScript. Sua função primária é o diagnóstico de tempo de execução, servindo como o principal canal para o registro (*logging*) de erros, advertências e saídas diagnósticas (ex: `console.log()` (??)). Permite a interação direta com o contexto da página, viabilizando a inspeção e manipulação programática do DOM e do objeto `Window`, além do teste de expressões e a utilização de funções utilitárias de depuração (??).

O painel **Fontes** (Sources), ilustrado na figura 58, funciona como um ambiente de depuração de código-fonte integrado. É projetado para a análise, edição e depuração de JavaScript front-end. A ferramenta permite a navegação pelos recursos da página, a edição de scripts e a utilização de um depurador de JavaScript (??). Este último suporta funcionalidades essenciais como a definição de pontos de interrupção (*breakpoints*), a execução passo a passo

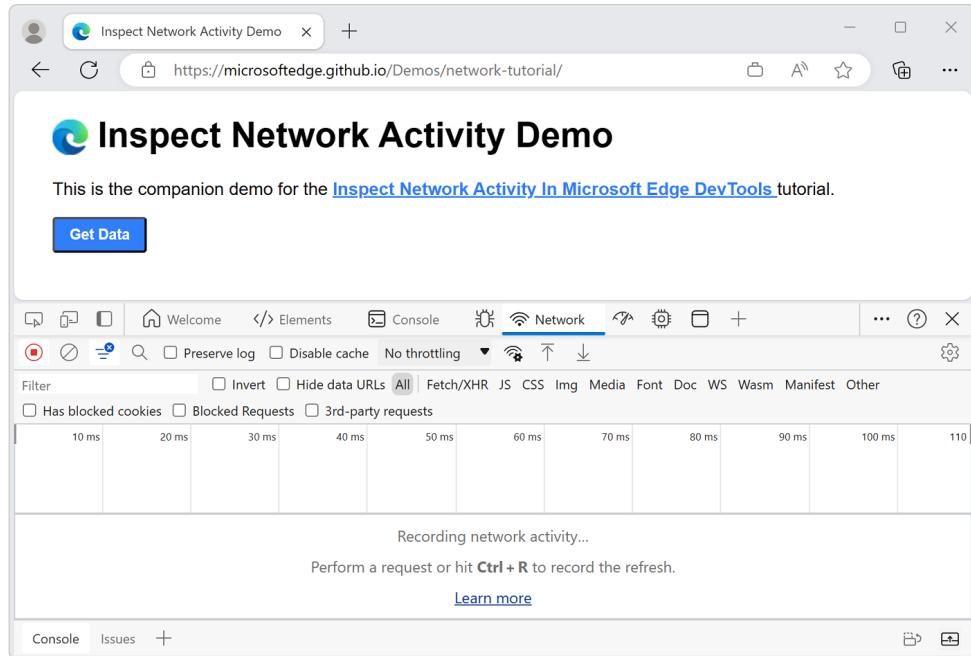
Figura 58 – Ferramenta "Fontes" do Microsoft Edge Devtools



Fonte: Microsoft Learn

(*step-through*) do código, a inspeção da pilha de chamadas (*call stack*) e a monitorização de variáveis em escopo (??).

Figura 59 – Ferramenta "Rede" do Microsoft Edge Devtools

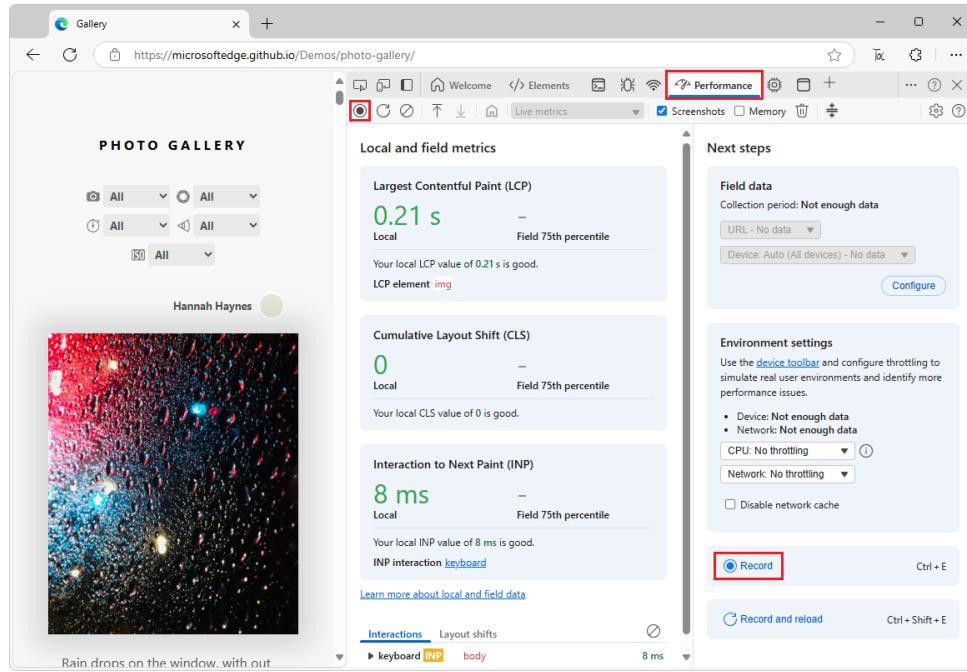


Fonte: Microsoft Learn

A ferramenta **Rede** (Network), ilustrada na figura 59, é um componente de diagnóstico focado na análise do tráfego de rede. Ela monitora e registra todas as solicitações (requests) e respostas (responses) HTTP/S, incluindo documentos, scripts, folhas de estilo, mídias e chamadas de API (XHR/Fetch) (??). A ferramenta oferece uma análise detalhada de cabeçalhos,

códigos de status, conteúdo e temporização (através de um gráfico de *Waterfall*), sendo crucial para a identificação de gargalos de latência, falhas de requisição e otimização de *payloads* (??). Permite ainda a simulação de condições de rede adversas (*throttling*) (??).

Figura 60 – Ferramenta "Desempenho" do Microsoft Edge Devtools



Fonte: Microsoft Learn

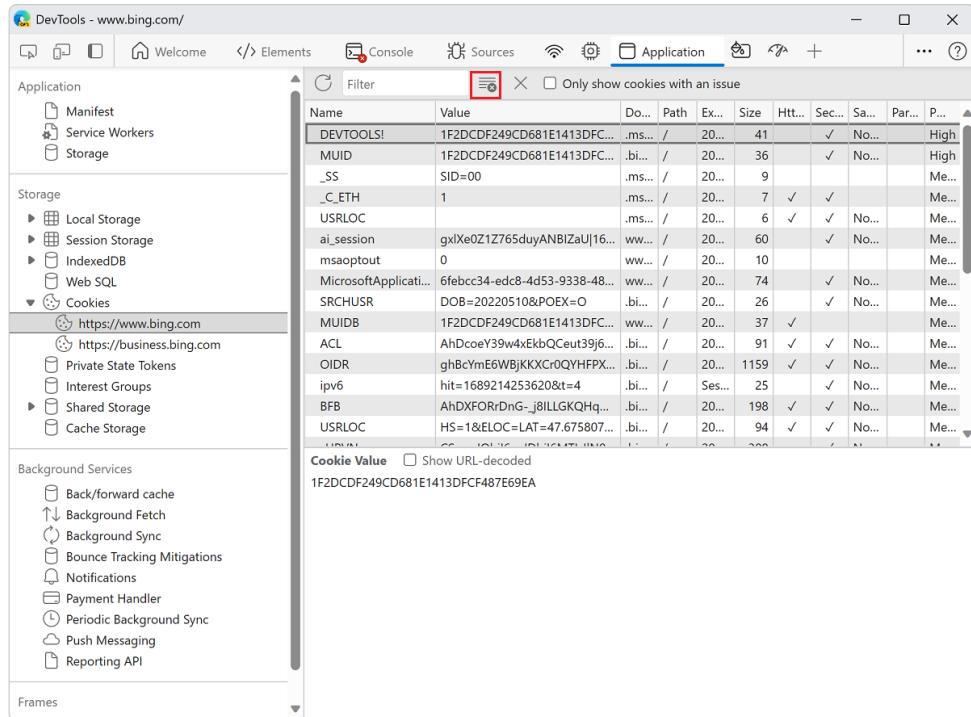
O painel **Desempenho** (Performance), como exibido na figura 60, é um utilitário de perfilamento (*profiling*) avançado para a análise da performance de tempo de execução. A ferramenta opera em duas modalidades: um monitoramento em tempo real dos Core Web Vitals (LCP, CLS, INP) e a gravação de um perfil detalhado (??). Este perfil captura uma linha do tempo da atividade da *thread* principal, incluindo a execução de JavaScript, operações de renderização (*layout* e *paint*), e eventos de interação (??). A análise deste traço permite a identificação precisa de gargalos de CPU e operações de longa duração que afetam a responsividade da aplicação .

A ferramenta **Aplicação** (Application), apresentado na figura 61, proporciona uma interface centralizada para a inspeção e gerenciamento do armazenamento no lado do cliente. Ela permite a análise e manipulação de mecanismos de armazenamento como Local Storage, Session Storage, IndexedDB, Cookies e Cache Storage (??). Adicionalmente, pode ser utilizado para a depuração de Progressive Web Apps (PWAs), oferecendo controle sobre o ciclo de vida dos *Service Workers*, a validação do Manifesto da Aplicação e a inspeção de serviços em segundo plano (*Background Services*) (??).

A **Exibição 3D** (3D View), ilustrada na figura 62, é um recurso de visualização avançado para a depuração de contextos de renderização complexos. Ao modelar a página em um espaço tridimensional, a ferramenta facilita a identificação de problemas de sobreposição de elementos (*z-index*), a análise da hierarquia do DOM e a inspeção das camadas de composição (*composited layers*) criadas pelo motor de renderização (??). A ferramenta segmenta a visualização em abas dedicadas para Camadas Compostas, Z-index e DOM, auxiliando na compreensão da estrutura de renderização (??).

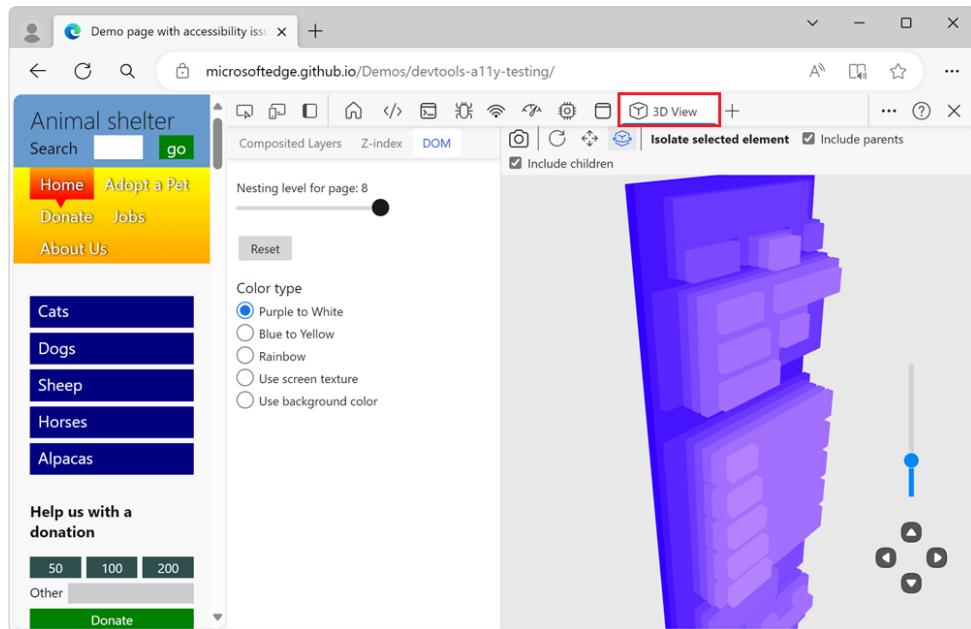
O inspetor de **Animações** (Animation Inspector), exibido na figura 63, é um utilitário focado na inspeção e depuração de animações declarativas. A ferramenta captura sequências

Figura 61 – Ferramenta "Aplicação" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 62 – Ferramenta "Exibição 3D" do Microsoft Edge Devtools

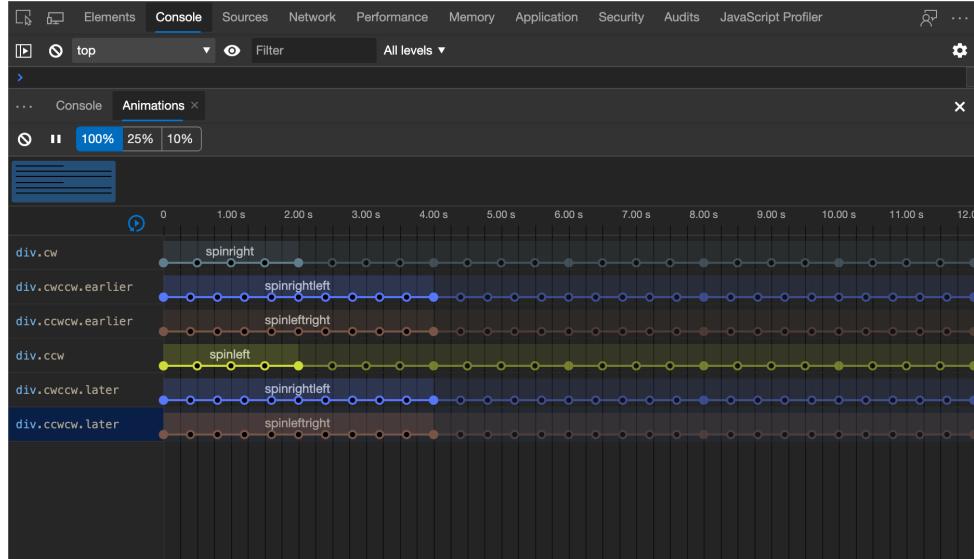


Fonte: Microsoft Learn

de animação (CSS Animations, CSS Transitions e Web Animations), agrupando-as com base em seu tempo de início (??). Permite a desaceleração, repetição e análise do código-fonte das animações, além de possibilitar a modificação interativa de parâmetros como duração, atraso (*delay*) e temporização de *keyframes* (??).

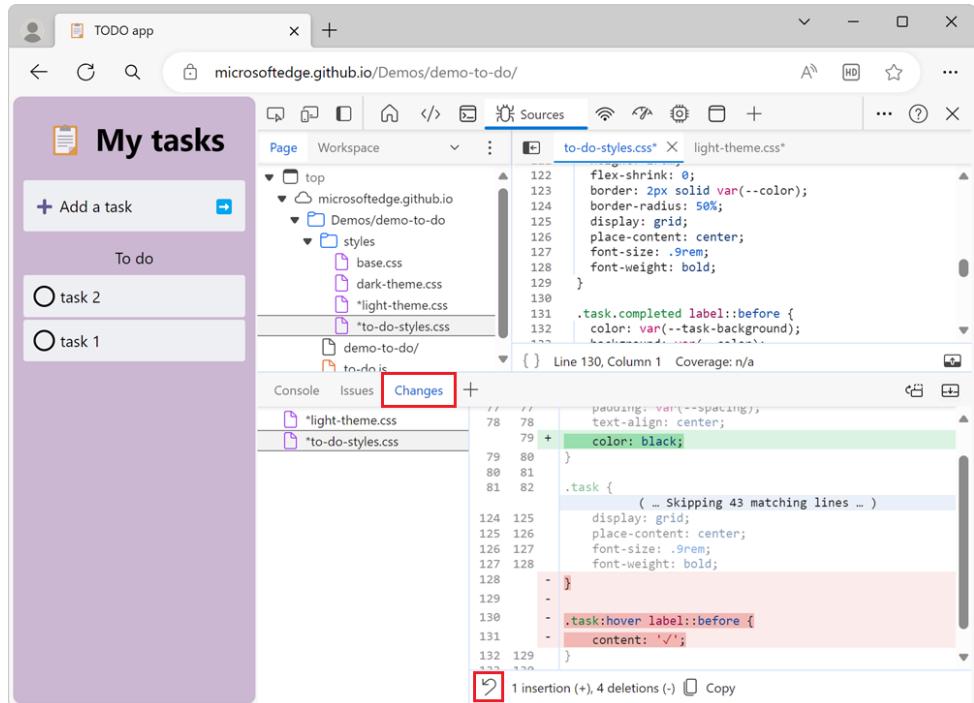
A ferramenta **Alterações** (Changes), apresentada na figura 64, monitora modificações efetuadas nos arquivos-fonte (CSS, JavaScript, HTML) diretamente no ambiente de desen-

Figura 63 – Ferramenta "Animações" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 64 – Ferramenta "Alterações" do Microsoft Edge Devtools

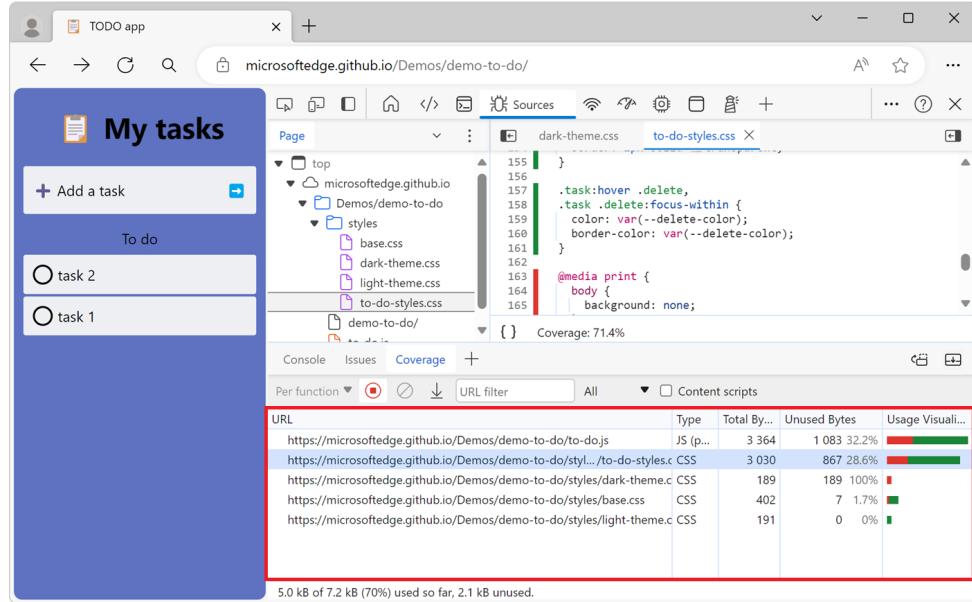


Fonte: Microsoft Learn

volvimento. Sua função primária é apresentar um comparativo visual (*diff*) que detalha as discrepâncias (adições e remoções de linhas) entre o estado original do recurso e as edições (??). Isso facilita a revisão das alterações antes de sua persistência no sistema de arquivos, permitindo também a reversão das modificações (??).

O painel **Cobertura** (Coverage), exibido na figura 65, é um utilitário analítico que quantifica a utilização de código JavaScript e CSS durante o carregamento e a interação com a página. A ferramenta identifica segmentos de código não executados, fornecendo métricas de bytes (utilizados vs. não utilizados) e um detalhamento visual linha a linha (??). Esta análise é

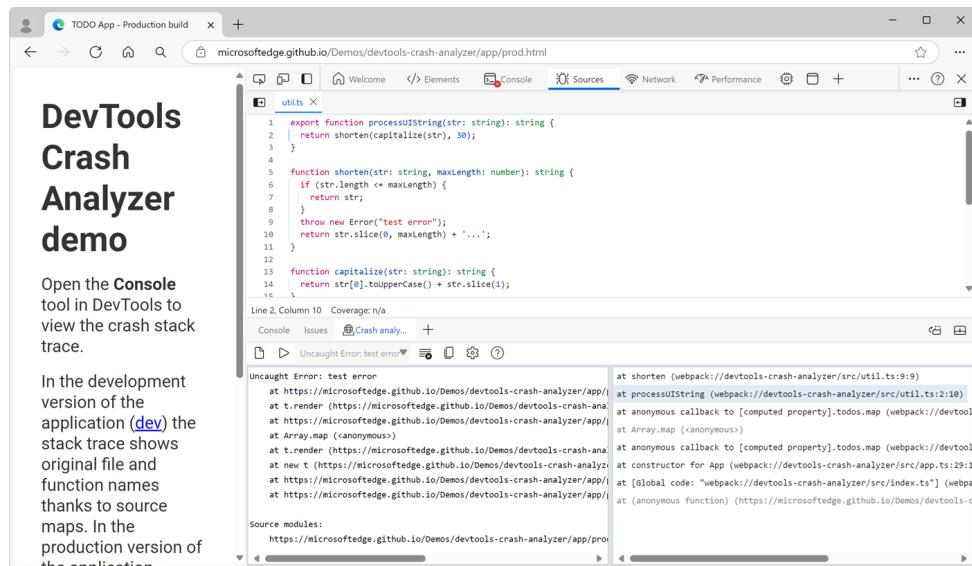
Figura 65 – Ferramenta "Cobertura" do Microsoft Edge Devtools



Fonte: Microsoft Learn

fundamental para otimizações de *tree-shaking* e remoção de código morto (*dead code*), visando a redução do *payload* da aplicação e a melhoria do tempo de carregamento (??).

Figura 66 – Ferramenta "Analizador de Falhas" do Microsoft Edge Devtools

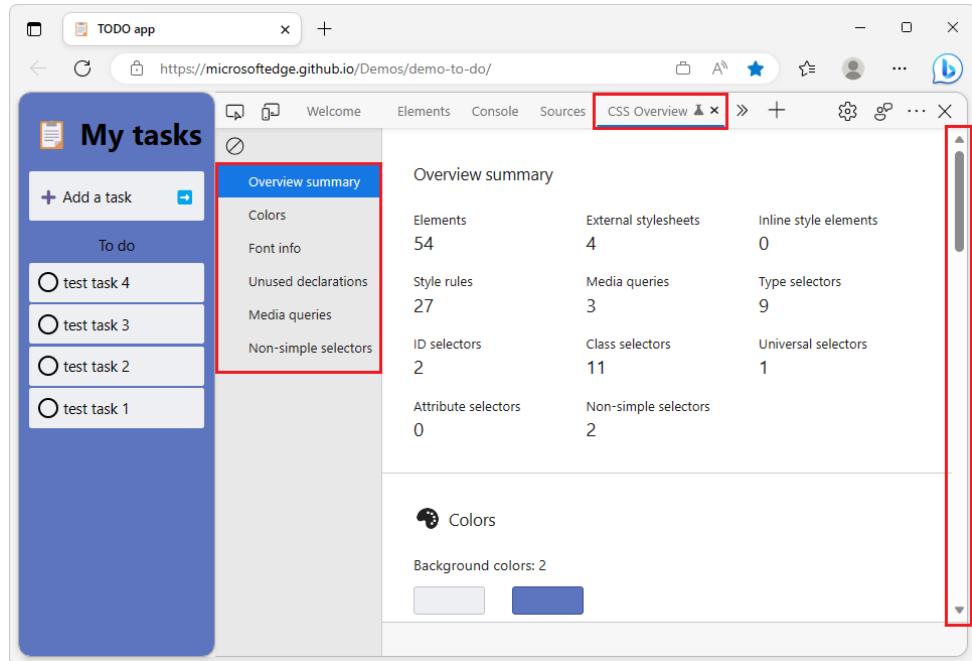


Fonte: Microsoft Learn

O **Analisador de Falhas** (Crash Analyzer), ilustrado na figura 66, é um utilitário de diagnóstico pós-morte concebido para analisar falhas de aplicações em produção. Sua principal função é processar *stack traces* (pilhas de chamadas) de JavaScript minificados, que são frequentemente ofuscados (??). Ao aplicar os *source maps* correspondentes, a ferramenta reverte o código ao seu estado original não minificado, permitindo a identificação da causa raiz da falha ao mapear o erro de volta aos nomes de funções e linhas de código legíveis (??).

A ferramenta **Descrição Geral de CSS** (CSS Overview), apresentada na figura 67, realiza uma auditoria estática do CSS de uma página. Ela captura um instantâneo do estado do

Figura 67 – Ferramenta "Descrição Geral de CSS" do Microsoft Edge Devtools



Fonte: Microsoft Learn

CSS gera um relatório que cataloga o uso de cores, tipografia (*fonts*) e *media queries* (??). A ferramenta é projetada para identificar inconsistências no design system, como cores duplicadas ou estilos de fonte não padronizados, e destaca problemas de acessibilidade, notadamente questões de contraste de cor insuficientes (??).

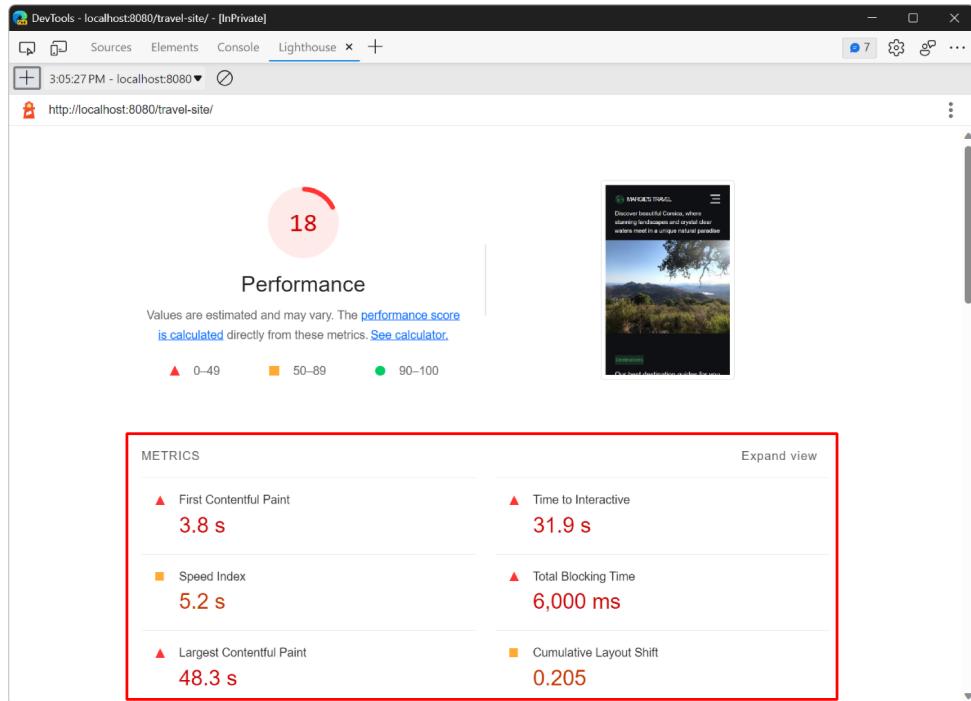
O painel **Problemas** (Issues) funciona como um agregador proativo de diagnósticos. Ele analisa continuamente a página e consolida problemas detectados em categorias como acessibilidade, compatibilidade entre navegadores (*cross-browser compatibility*), desempenho, segurança e conformidade com PWA (??). A ferramenta fornece descrições contextuais de cada problema e links diretos para a documentação ou para o recurso afetado, centralizando o feedback de múltiplas fontes de auditoria (??).

A ferramenta **Lighthouse** (anteriormente "Audits"), exibida na figura 68, é um utilitário de auditoria automatizada para a avaliação da qualidade de páginas web. Ela executa uma série de testes e gera relatórios de diagnóstico abrangentes, pontuando a página em métricas de Desempenho, Acessibilidade, Melhores Práticas, SEO e Progressive Web App (PWA) (??). Os relatórios fornecem uma linha de base quantificável e recomendações açãoáveis para a otimização de cada uma dessas categorias (??).

O painel **Mídia** (Media), apresentado na figura 69, é um utilitário de depuração específico para a inspeção de reprodutores de mídia (`<video>` e `<audio>`) em uma página. A ferramenta monitora e exibe propriedades do reproduutor, eventos do ciclo de vida da mídia (ex: *play*, *pause*, *seek*) e mensagens de log (??). Facilita a depuração de problemas de reprodução, *buffering* e eventos de mídia (??).

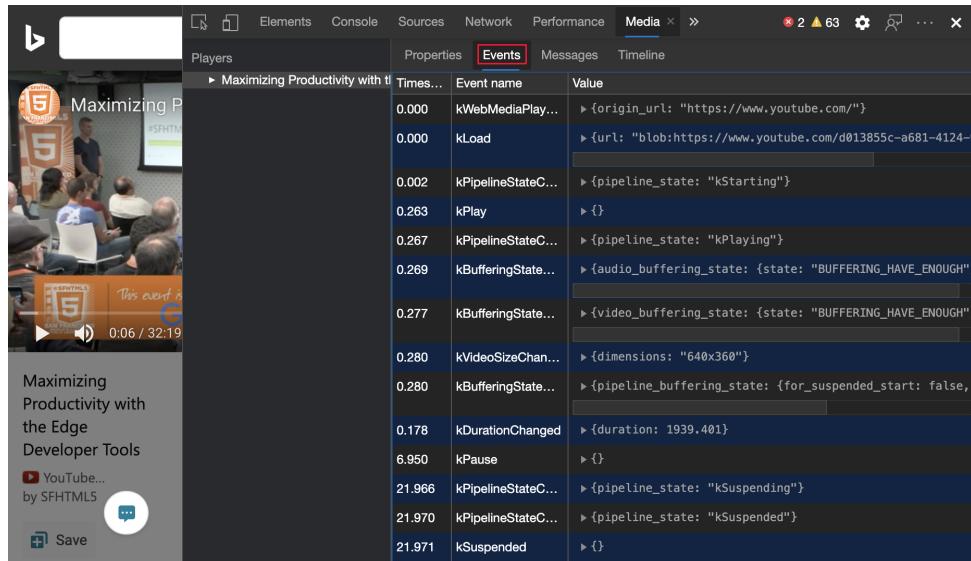
O **Inspetor de Memória** (Memory Inspector), ilustrado na figura 70, é um utilitário para a inspeção de baixo nível de buffers de memória binária. Ele é projetado para analisar *ArrayBuffer*, *TypedArray*, *DataView* e, crucialmente, a memória linear de WebAssembly (Wasm) (??). A ferramenta exibe os bytes brutos em formato hexadecimal, uma representação ASCII adjacente e um inspetor de valores que interpreta os dados em múltiplos formatos (ex: inteiros de 32 bits, *floats*), suportando a alternância entre *big-endian* e *little-endian* (??).

Figura 68 – Ferramenta "Farol" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 69 – Ferramenta "Mídia" do Microsoft Edge Devtools

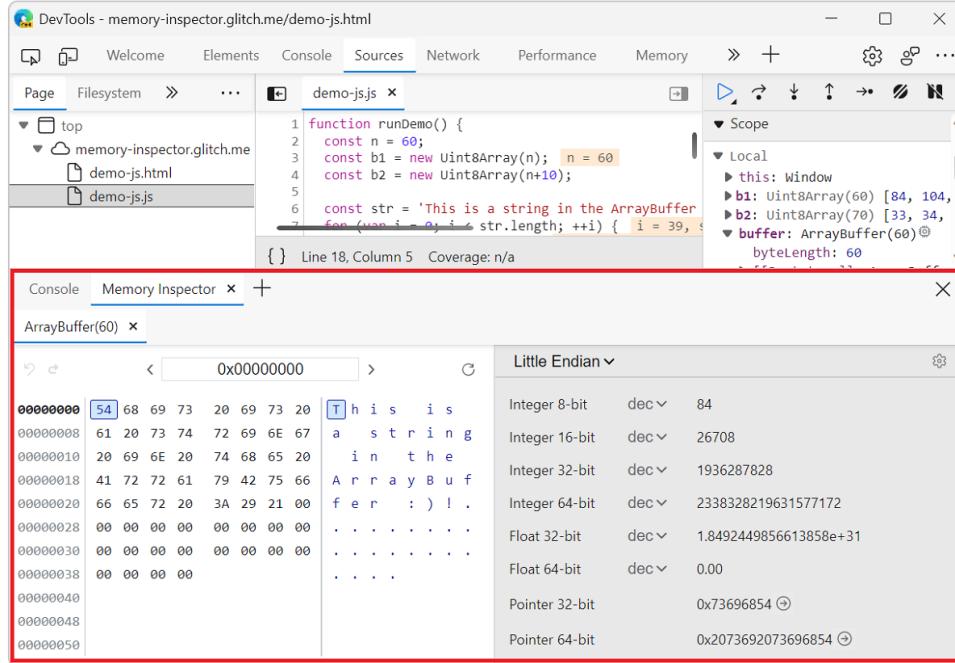


Fonte: Microsoft Learn

A ferramenta **Condições de Rede** (Network Conditions), apresentada na figura 71, é um utilitário de simulação de ambiente. Suas funções primárias incluem a desativação do cache do navegador, a aplicação de limitação de banda (*throttling*) para emular diferentes velocidades de conexão (ex: 3G lento) e a capacidade de sobrescrever a *string* de agente do usuário (*user agent*) (??). Adicionalmente, permite a configuração das codificações de conteúdo (Content-Encodings) aceitas para testar o processamento de respostas comprimidas (??).

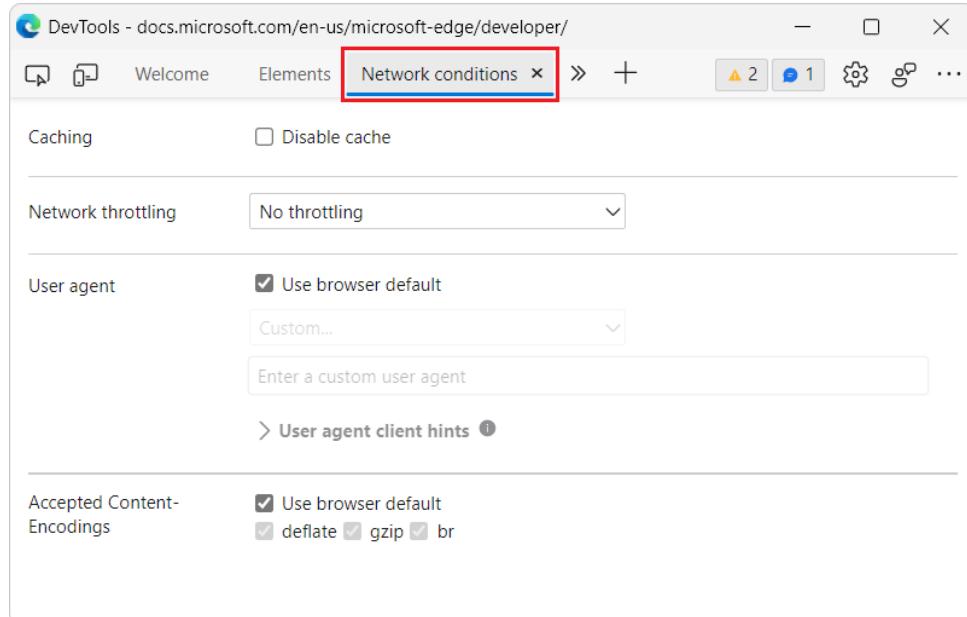
O **Console de Rede** (Network Console), exibido na figura 72, é um cliente de API integrado, projetado para a composição, envio e teste de solicitações HTTP. Permite

Figura 70 – Ferramenta "Inspetor de Memória" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 71 – Ferramenta "Condições de Rede" do Microsoft Edge Devtools

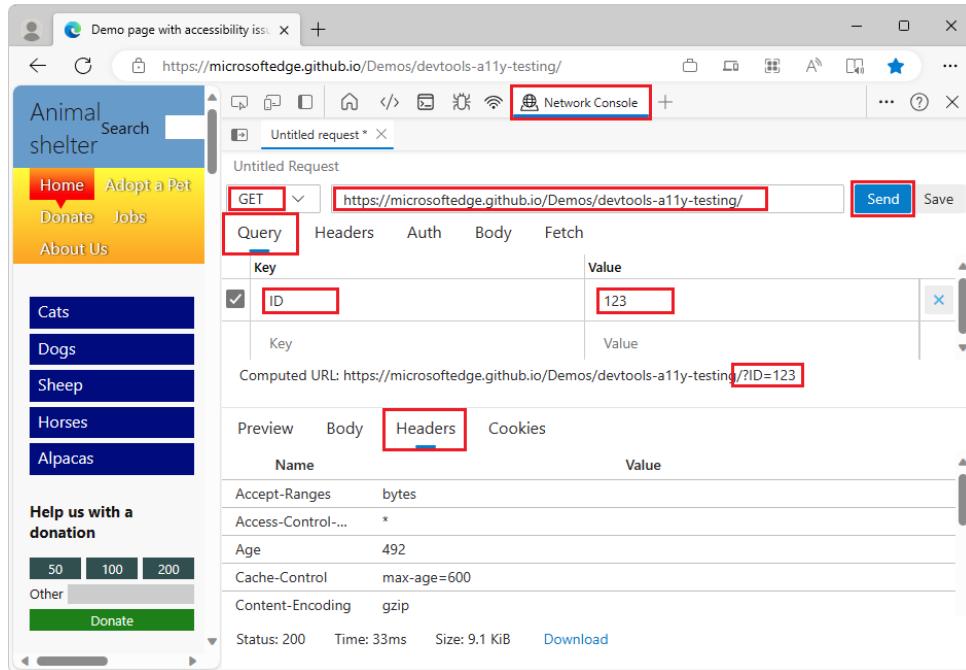


Fonte: Microsoft Learn

a configuração detalhada de requisições, especificando o método (GET, POST, etc.), URL, cabeçalhos e corpo da solicitação (??). É usado primariamente para a depuração de APIs Web (REST/OpenAPI) e é compatível com a importação e exportação de coleções (ex: formato Postman) (??).

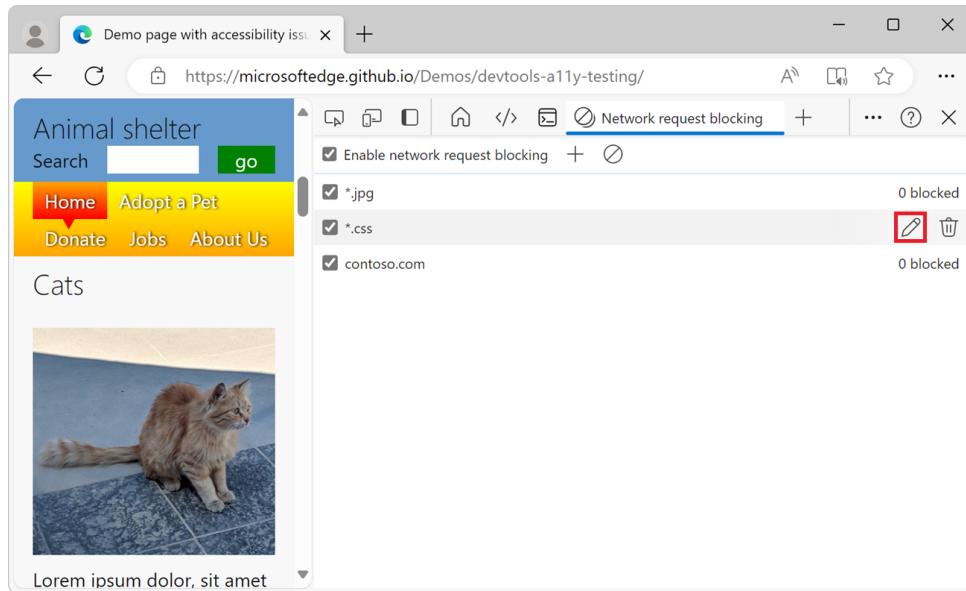
Este utilitário, apresentado na figura 73, permite a simulação de falhas de rede ao bloquear o carregamento de recursos específicos. Os desenvolvedores podem definir padrões (incluindo wildcards) para impedir que solicitações de rede para URLs, domínios ou tipos de arquivos específicos sejam concluídas (??). Esta funcionalidade é essencial para testar a

Figura 72 – Ferramenta "Console de Rede" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 73 – Ferramenta "Bloqueio de Solicitações de Rede" do Microsoft Edge Devtools



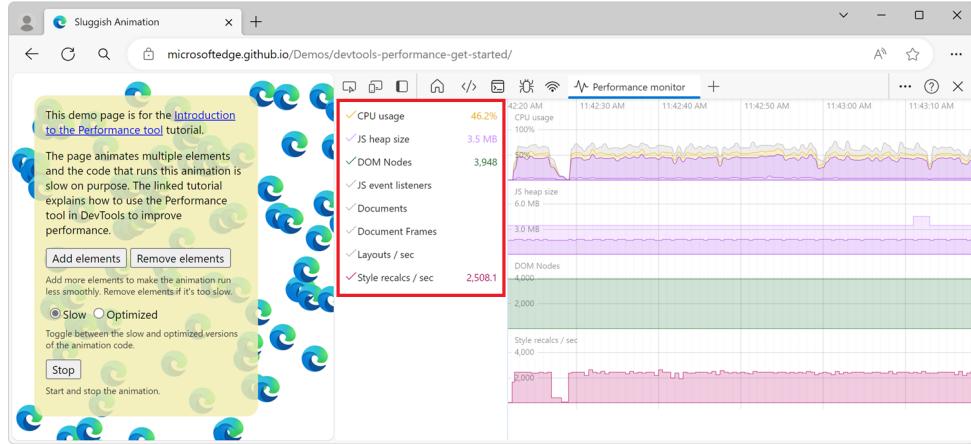
Fonte: Microsoft Learn

resiliência da aplicação, o comportamento de *fallback* e a renderização da página em cenários de indisponibilidade de recursos críticos (??).

O **Monitoramento de Desempenho** (Performance Monitor), ilustrado na figura 74, oferece uma visualização em tempo real de métricas de desempenho de tempo de execução. Diferente do perfilamento detalhado do painel "Desempenho", este monitor exibe gráficos contínuos de indicadores-chave, como uso da CPU, tamanho do *heap* de JavaScript, contagem de nós DOM e a frequência de recálculos de layout e estilo por segundo, auxiliando na identificação de consumo excessivo de recursos (??).

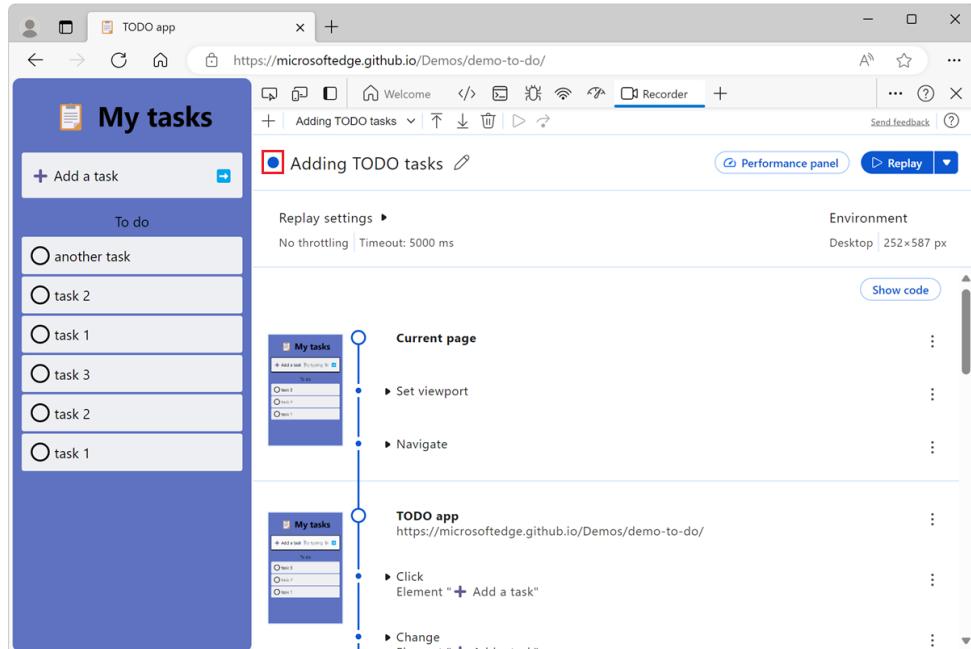
A ferramenta **Gravador** (Recorder), apresentado na figura 75, permite a captura e

Figura 74 – Ferramenta "Monitoramento de Desempenho" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 75 – Ferramenta "Gravador" do Microsoft Edge Devtools



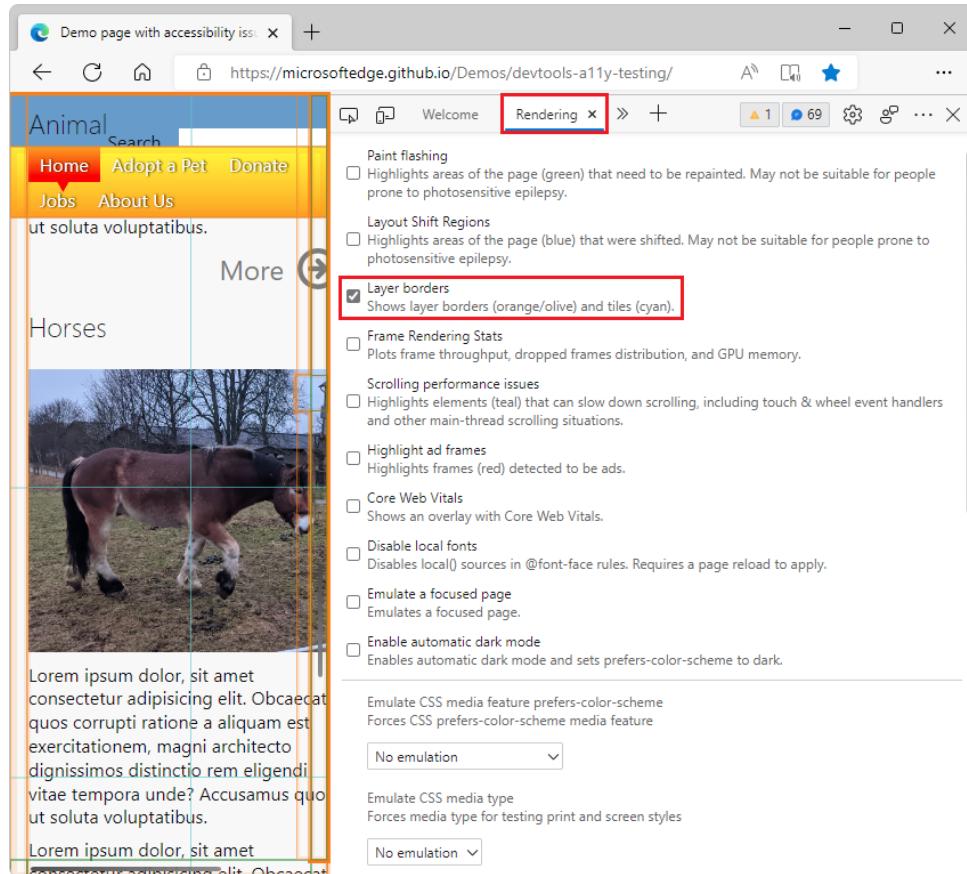
Fonte: Microsoft Learn

reprodução de fluxos de interação do usuário. Ela registra ações como cliques, entradas de teclado e eventos de navegação, permitindo que a sequência seja executada automaticamente (??). Este utilitário é usado para automatizar testes de regressão, analisar fluxos de usuário complexos e medir o desempenho de tempo de execução durante a reprodução do fluxo, identificando gargalos de performance em interações específicas (??).

O painel **Renderização** (Rendering), exibido na figura 76, é um conjunto de utilitários de diagnóstico focados na visualização da saída do motor de renderização. Suas funcionalidades incluem a emulação de recursos de mídia CSS (como `prefers-color-scheme` e modo de impressão), a simulação de deficiências visuais (ex: daltonismo, visão turva) e a ativação de sobreposições de depuração (ex: `paint flashing`, `layout shift regions`) (??).

A ferramenta **Pesquisa** (Search), ilustrada na figura ??, oferece uma funcionalidade de busca global que abrange todos os arquivos de recursos carregados pela página (HTML, CSS, JS). Ela suporta a localização de sequências de texto literais e expressões regulares, com opções

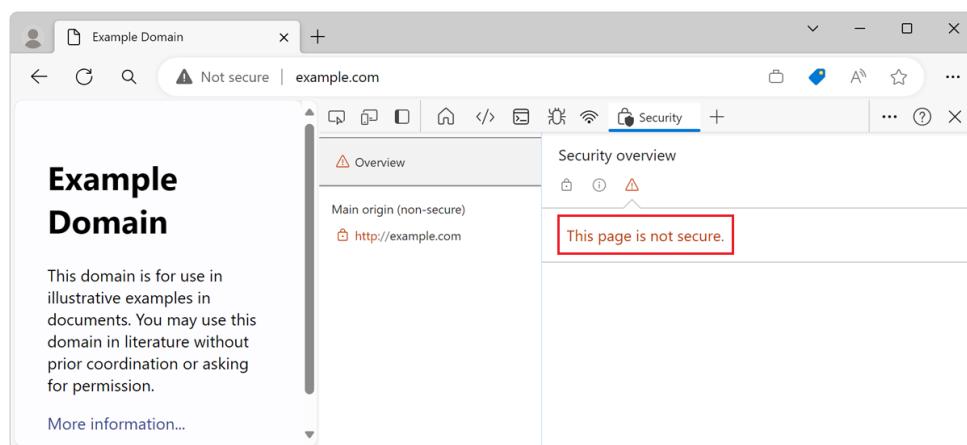
Figura 76 – Ferramenta "Renderização" do Microsoft Edge Devtools



Fonte: Microsoft Learn

de sensibilidade a maiúsculas e minúsculas (??). Os resultados são listados por arquivo, e a seleção de um resultado direciona o usuário para a linha correspondente no painel "Fontes"(??).

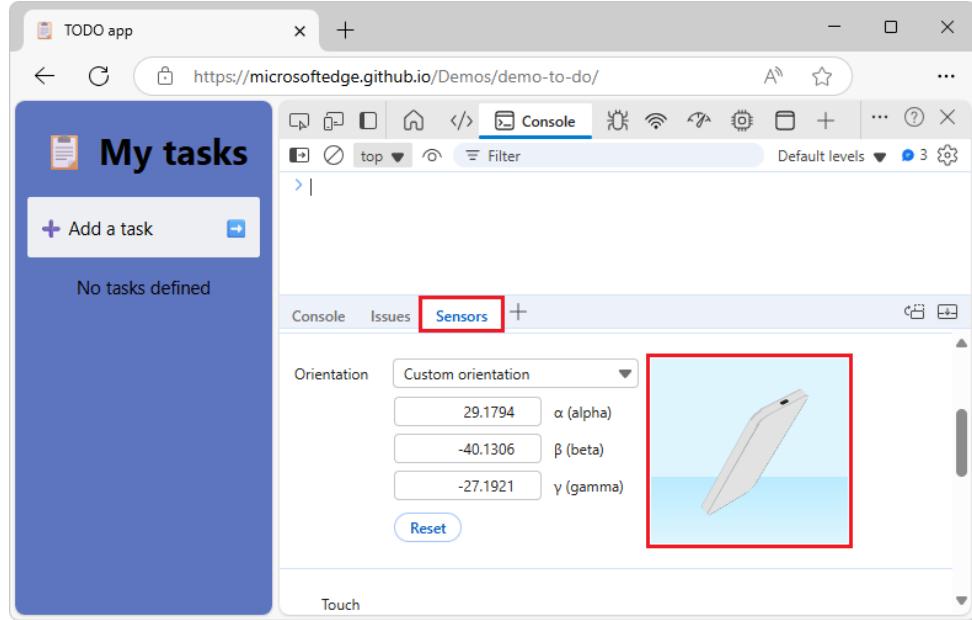
Figura 77 – Ferramenta "Segurança" do Microsoft Edge Devtools



Fonte: Microsoft Learn

O painel **Segurança** (Security), apresentado na figura 77, analisa o status de segurança da conexão de uma página. Ele verifica a validade do certificado HTTPS da origem principal e identifica a ocorrência de "conteúdo misto"(*mixed content*), que ocorre quando uma página segura (HTTPS) carrega sub-recursos (como imagens ou scripts) de origens inseguras (HTTP) (??). A ferramenta detalha o status do certificado e da conexão para cada origem (??).

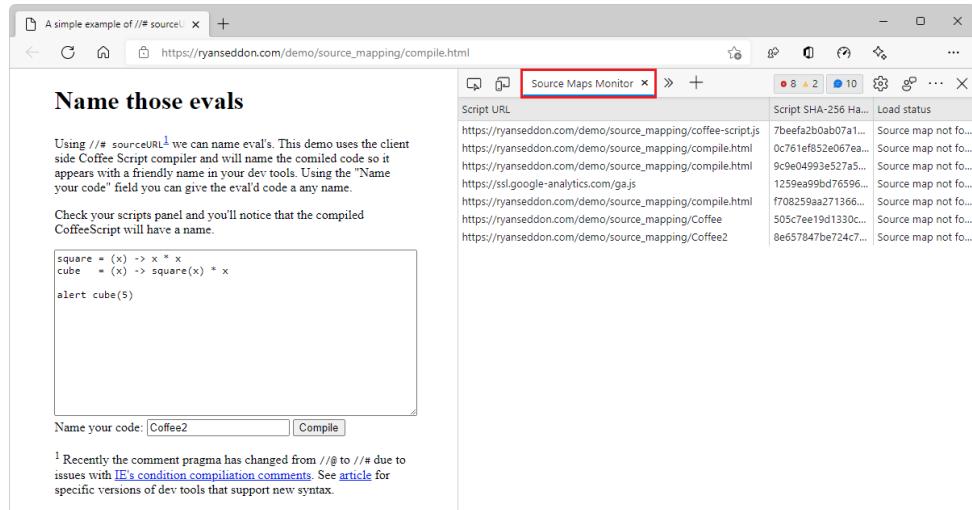
Figura 78 – Ferramenta "Sensores" do Microsoft Edge Devtools



Fonte: Microsoft Learn

A ferramenta **Sensores** (Sensors), exibida na figura 78, é um utilitário de emulação de hardware e estados do sistema. Ela permite sobreescriver a geolocalização (latitude/longitude), simular a orientação física do dispositivo (dados de acelerômetro e giroscópio) e forçar eventos de toque (??). Capacidades mais recentes incluem a emulação de estados do *Idle Detector*, a simulação da concorrência de hardware (`navigator.hardwareConcurrency`) e a emulação da pressão da CPU (??).

Figura 79 – Ferramenta "Monitor de Mapas de Origem" do Microsoft Edge Devtools

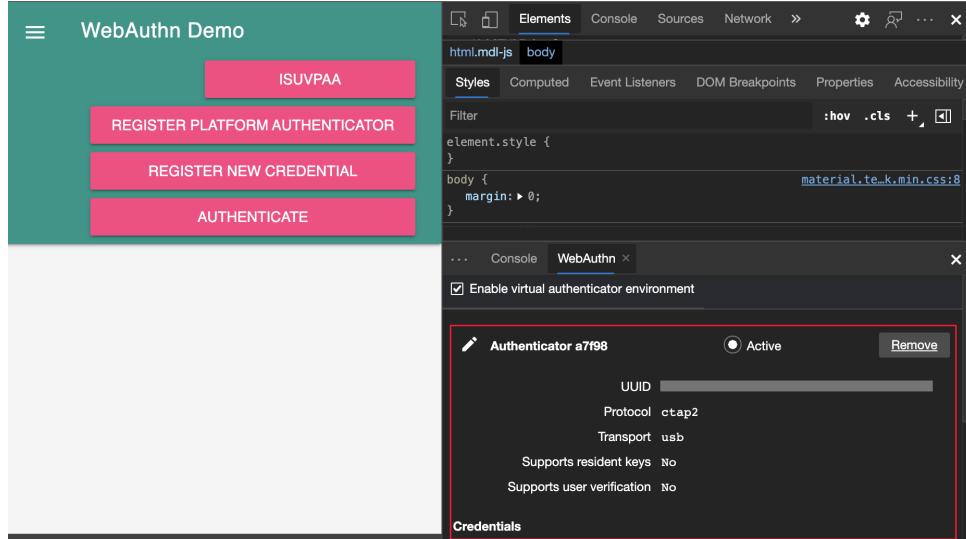


Fonte: Microsoft Learn

Este painel, apresentado na figura 79, monitora especificamente a requisição e o status de carregamento de mapas de origem (*source maps*). Ele fornece um log de quais arquivos *source map* foram solicitados, se foram carregados com sucesso ou se falharam (??). É um utilitário de diagnóstico crucial para garantir que a depuração de código minificado ou transpilado (como TypeScript) funcione corretamente, mapeando o código executado de volta

ao código-fonte original (??).

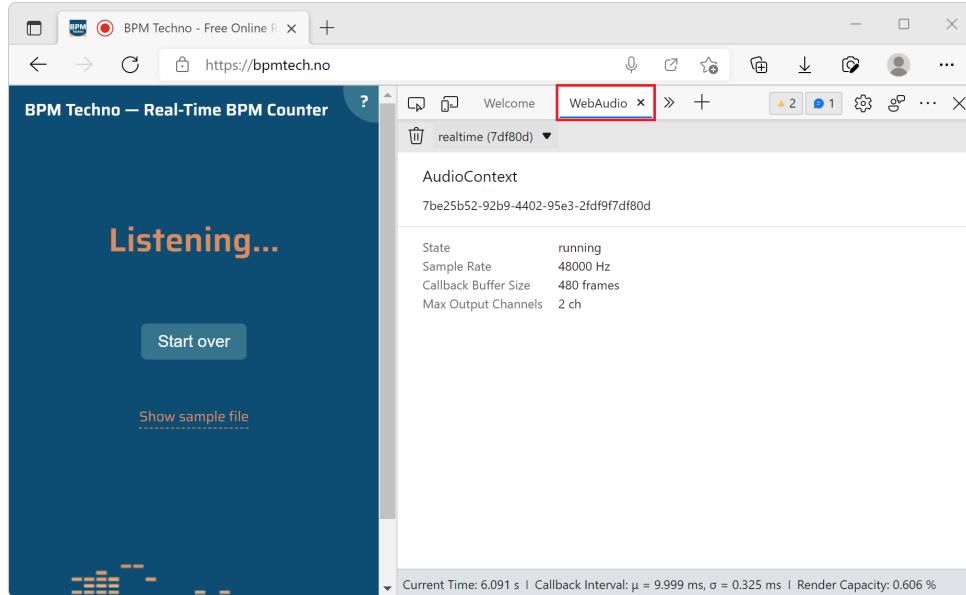
Figura 80 – Ferramenta "WebAuthn" do Microsoft Edge Devtools



Fonte: Microsoft Learn

A ferramenta **WebAuthn**, ilustrada na figura 80, é um utilitário de depuração para a API de Autenticação da Web. Ela permite a criação e gerenciamento de autenticadores virtuais baseados em software, eliminando a necessidade de dispositivos físicos (como chaves de segurança USB) durante o desenvolvimento (??). O painel permite a configuração de atributos do autenticador, como protocolo (CTAP2, U2F), transporte e suporte a chaves residentes (??).

Figura 81 – Ferramenta "WebAudio" do Microsoft Edge Devtools



Fonte: Microsoft Learn

O painel **WebAudio**, apresentado na figura 81, é um utilitário de diagnóstico para aplicações que utilizam a API WebAudio. Ele renderiza um grafo visual interativo de todos os nós de áudio (*AudioNodes*) instanciados, mostrando suas conexões e estado (??). Esta visualização é essencial para depurar o fluxo de processamento de áudio, inspecionar os parâmetros dos nós e monitorar o *AudioContext* em tempo real (??).

2.5 Tabela comparativa das ferramentas nativas dos navegadores

Diante das diversas funcionalidades oferecidas pelas devtools dos principais navegadores, as tabelas apresenta uma comparação das ferramentas nativas disponíveis no Google Chrome, Mozilla Firefox, Microsoft Edge e Apple Safari. Esta comparação visa auxiliar desenvolvedores na escolha do navegador mais adequado às suas necessidades específicas de desenvolvimento e depuração web.

Tabela 1 – Comparação das ferramentas nativas dos navegadores - Parte 1

Funcionalidades / Navegadores	Google Chrome	Mozilla Firefox
Inspecionar, editar e depurar HTML e CSS	Elements	Inspector
Verificar registros de erros	Console	Web Console
Executar código JavaScript	Console	Web Console
Verificar código-fonte	Sources	Debugger
Verificar registros de rede	Network	Network Monitor
Analizar desempenho	Performance	Performance
Monitorar uso de memória	Memory	Memory
Gerenciar armazenamento da aplicação web	Application	Storage
Entender a hierarquia do DOM	Elements	Inspector
Inspecionar e modificar animações CSS	Animations	-
Verificar mudanças em arquivos	Changes	-
Encontrar trechos de código não utilizados	Coverage	-
Diagnosticar falhas da aplicação web	-	-
Relatório de estilos utilizados	CSS Overview	-
Visualizar URL de recursos	Network / Sources	Network Monitor
Encontrar e corrigir problemas	Issues	-
Inspecionar players de mídia	Media	-
Visualizar objetos em memória	Memory Inspector	-
Simular condições de rede	Network conditions	-
Realizar requisições HTTP	-	-
Bloquear requisições de rede	Network request blocking	-
Medir o desempenho da aplicação	Performance	Performance
Gravar interações do usuário	Recorder	-
Emular deficiências visuais	Rendering	Accessibility
Encontrar recursos específicos	Search	-
Inspecionar a segurança da aplicação	Security	-
Emular sensores de dispositivos	Sensors	-
Monitorar carregamento de arquivos	-	-
Monitorar o gráfico de nós WebAudio	WebAudio	-
Emular autenticação	WebAuthn	-

Fonte: Elaborado pelo autor.

Tabela 2 – Comparação das ferramentas nativas dos navegadores - Parte 2

Funcionalidades / Navegadores	Microsoft Edge	Apple Safari
Inspecionar, editar e depurar HTML e CSS	Elements	Elements
Verificar registros de erros	Console	Console
Executar código JavaScript	Console	Console
Verificar código-fonte	Sources	Sources
Verificar registros de rede	Network	Network
Analisar desempenho	Performance	Timelines
Monitorar uso de memória	Memory	Timelines
Gerenciar armazenamento da aplicação web	Application	Storage
Entender a hierarquia do DOM	Elements	Elements
Inspecionar e modificar animações CSS	Animations	Timelines
Verificar mudanças em arquivos	Changes	Changes
Encontrar trechos de código não utilizados	Coverage	-
Diagnosticar falhas da aplicação web	Crash analyzer	-
Relatório de estilos utilizados	CSS Overview	-
Visualizar URL de recursos	Network / Sources	Sources / Network
Encontrar e corrigir problemas	Issues	Audit
Inspecionar players de mídia	Media	Timelines
Visualizar objetos em memória	Memory Inspector	-
Simular condições de rede	Network conditions	-
Realizar requisições HTTP	Network Console	-
Bloquear requisições de rede	Network request blocking	-
Medir o desempenho da aplicação	Performance	Timelines
Gravar interações do usuário	Recorder	-
Emular deficiências visuais	Rendering	-
Encontrar recursos específicos	Search	-
Inspecionar a segurança da aplicação	Security	-
Emular sensores de dispositivos	Sensors	-
Monitorar carregamento de arquivos	Source Maps Monitor	-
Monitorar o gráfico de nós WebAudio	WebAudio	-
Emular autenticação	WebAuthn	-

Fonte: Elaborado pelo autor.

3 Resultados

A análise realizada neste estudo apresentou um panorama sobre as ferramentas de desenvolvimento nativas dos principais navegadores web modernos. Com base na revisão técnica exploratória, é possível perceber que, apesar da metade dos navegadores analisados oferecerem um conjunto robusto de funcionalidades, há uma variação significativa na adoção e utilização dessas ferramentas pelos desenvolvedores web.

Também foi observado que, embora há a lista de recursos possa ser extensas, todos os navegadores analisados oferecem algumas funcionalidades em comum, que inclui a inspeção e edição da árvore DOM e CSS, um console JavaScript interativo, o monitoramento detalhado de requisições de rede e ferramentas de perfilamento de desempenho. No entanto, a profundidade e a usabilidade dessas ferramentas variam consideravelmente entre os navegadores.

A análise técnica das ferramentas de perfilamento revelou abordagens distintas na arquitetura de diagnóstico de desempenho e uso de recursos entre os navegadores. No Google Chrome, observou-se uma segregação funcional clara, onde o painel 'Desempenho' é dedicado exclusivamente à análise de CPU e monitoramento de métricas vitais (Core Web Vitals) através de gráficos de chamas (flame graphs) e árvores de chamadas. Paralelamente, o Chrome oferece um painel de 'Memória' independente, focado em diagnósticos granulares como snapshots de heap, amostragem de alocação e identificação de elementos DOM desconectados para isolar vazamentos de memória.

Em contraste, o Safari Web Inspector adota uma filosofia de visualização holística através da aba 'Linhas do Tempo' (Timelines). Esta ferramenta unifica a captura de dados, agregando atividades de rede, layout, JavaScript, CPU e alocações de memória em uma única interface cronológica. Diferente da abordagem do Chrome, o Safari estrutura a análise em duas perspectivas principais: a 'Visualização de Eventos' e a 'Visualização de Quadros' (Frames View), sendo esta última projetada especificamente para analisar o tempo de execução de cada quadro de renderização em comparação com benchmarks de 30 e 60 FPS. Essa divergência indica que, enquanto o Chrome prioriza a profundidade de diagnóstico em ferramentas isoladas, o Safari privilegia a correlação de eventos para a análise de fluidez visual.

Além disso, foi notório que ainda existe uma grande espaço para novos trabalhos que explorem mais a fundo as necessidades dos desenvolvedores web, visto que grande parte do conteúdo apresentado tem como base a documentação oficial dos navegadores, que carece de informações para determinados recursos, e não há atualmente muitos estudos que investiguem e aprofundem o uso prático dessas ferramentas no dia a dia dos desenvolvedores.

4 Considerações Finais

Este estudo cumpriu seu objetivo de analisar, de forma exploratória, as ferramentas de desenvolvimento (devtools) integradas aos principais navegadores do mercado, trazendo um panorama geral dessas ferramentas. Por meio deste trabalho, ficou claro que, embora todas as plataformas compartilhem um conjunto de recursos para a manipulação do DOM e depuração de scripts, cada navegador apresenta particularidades distintas, adaptando-se a diferentes demandas do fluxo de trabalho em desenvolvimento web.

Na análise comparativa das funcionalidades, o Firefox distinguiu-se pelos recursos focados em privacidade e design (como CSS Grid e Flexbox), enquanto o Microsoft Edge demonstrou robustez na integração com o ecossistema Windows e o VS Code. Por sua vez, o Safari Web Inspector, mesmo restrito ao ambiente Apple, provou-se indispensável para a depuração e auditoria de performance em dispositivos iOS e macOS.

Os dados obtidos na pesquisa de campo revelaram um cenário relevante sobre as práticas de utilização. Verificou-se que, a despeito da sofisticação das devtools modernas, existe uma subutilização de seus recursos avançados. O fato de apenas 27% dos desenvolvedores dominarem a totalidade das ferramentas do Chrome DevTools, restringindo-se majoritariamente ao uso do "Inspetor de Elementos" e do "Console", aponta para uma lacuna significativa na capacitação técnica dos profissionais.

Em relação à produtividade e à qualidade de software, conclui-se que as devtools são vitais para a detecção precoce de erros e a otimização de performance (via Lighthouse e painéis de desempenho). Contudo, a eficácia desses recursos é limitada pelo nível de conhecimento do usuário. Observou-se, ainda, que desenvolvedores seniores tendem a complementar o uso do navegador com plugins de IDE, sugerindo que, para arquiteturas complexas, as ferramentas nativas, isoladamente, ainda não suprem todas as necessidades de desenvolvimento.

Por fim, este estudo sugere que a evolução do desenvolvimento web não depende apenas de novas funcionalidades nos navegadores, mas sim da disseminação do conhecimento sobre as ferramentas já existentes. Recomenda-se, para trabalhos futuros, a criação de materiais didáticos focados nas ferramentas de "Memória", "Desempenho" e "Segurança", áreas que demonstraram maior carência de domínio técnico, visando elevar o padrão de qualidade e eficiência na construção de aplicações web.

Referências

APPLE DEVELOPER DOCUMENTATION. **WebDriver: automação de testes no Safari**. Disponível em: <<https://developer.apple.com/documentation/safari-developer-tools/webdriver>>. Citado na página 36.

APPLE DEVELOPER DOCUMENTATION. **Web Inspector | Apple Developer Documentation**. 2025. Disponível em: <<https://developer.apple.com/documentation/safari-developer-tools/web-inspector>>. Citado 2 vezes nas páginas 2 e 4.

BASQUES, K. **Inspecionar atividade de rede**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/network?hl=pt-br>>. Citado na página 8.

BASQUES, K. **Visão geral do console**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/console?hl=pt-br>>. Citado 2 vezes nas páginas 5 e 6.

BASQUES, K.; EMELIANOVA, S. **Depurar Progressive Web Apps**. 2016. Disponível em: <<https://developer.chrome.com/docs/devtools/progressive-web-apps?hl=pt-br>>. Citado na página 12.

BASQUES, K.; EMELIANOVA, S. **Introdução à visualização e alteração do DOM**. 2019. Disponível em: <<https://developer.chrome.com/docs/devtools/dom?hl=pt-br>>. Citado na página 5.

BASQUES, K.; EMELIANOVA, S. **Sensores: emular sensores do dispositivo**. 2020. Disponível em: <<https://developer.chrome.com/docs/devtools/sensors?hl=pt-br>>. Citado na página 21.

BASQUES, K.; EMELIANOVA, S. **Acessar e alterar CSS**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/css?hl=pt-br>>. Citado na página 5.

BASQUES, K.; EMELIANOVA, S. **Animações: inspecionar e modificar efeitos de animação CSS**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/css/animations?hl=pt-br>>. Citado na página 11.

BASQUES, K.; EMELIANOVA, S. **Cobertura: encontre JavaScript e CSS não utilizados**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/coverage?hl=pt-br>>. Citado 2 vezes nas páginas 13 e 14.

BASQUES, K.; EMELIANOVA, S. **Visão geral do painel Origens**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/sources?hl=pt-br>>. Citado na página 10.

BASQUES, K.; EMELIANOVA, S. **Painel de privacidade e segurança**. 2025. Disponível em: <<https://developer.chrome.com/docs/devtools/security?hl=pt-br>>. Citado na página 10.

BASQUES, K.; MARTHE, D. S. **Condições de rede: substituir a string do user agent**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/device-mode/override-user-agent?hl=pt-br>>. Citado na página 17.

BURG, B. et al. **Console Command Line API: API de linha de comando do console**. 2020. Disponível em: <<https://webkit.org/web-inspector/console-command-line-api/>>. Citado 2 vezes nas páginas 31 e 32.

- BURG, B. et al. **Network Tab: guia da aba de rede**. 2020. Disponível em: <<https://webkit.org/web-inspector/network-tab/>>. Citado na página 32.
- BURG, B. et al. **Timelines Tab: guia da aba de linhas do tempo**. 2020. Disponível em: <<https://webkit.org/web-inspector/timelines-tab/>>. Citado 2 vezes nas páginas 33 e 34.
- DUTTON, S.; EMELIANOVA, S. **Problemas: encontrar e corrigir problemas**. 2020. Disponível em: <<https://developer.chrome.com/docs/devtools/issues?hl=pt-br>>. Citado na página 6.
- EMELIANOVA, S. **Emular recursos de mídia CSS**. 2022. Disponível em: <<https://developer.chrome.com/docs/devtools/rendering/emulate-css?hl=pt-br>>. Citado na página 9.
- EMELIANOVA, S. **Encontrar CSS inválido, modificado, inativo e outros**. 2022. Disponível em: <<https://developer.chrome.com/docs/devtools/css/issues?hl=pt-br>>. Citado na página 5.
- EMELIANOVA, S. **Recursos para desenvolvedores: visualizar e carregar manualmente mapas de origem**. 2023. Disponível em: <<https://developer.chrome.com/docs/devtools/developer-resources?hl=pt-br>>. Citado na página 15.
- EMELIANOVA, S. **Alterações: acompanhe suas alterações de HTML, CSS e JavaScript**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/changes?hl=pt-br>>. Citado 2 vezes nas páginas 12 e 13.
- EMELIANOVA, S. **Preenchimento automático: inspecionar e depurar endereços salvos**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/autofill?hl=pt-br>>. Citado na página 11.
- EMELIANOVA, S.; BASQUES, K. **Descobrir problemas com o desempenho da renderização**. 2022. Disponível em: <<https://developer.chrome.com/docs/devtools/rendering/performance?hl=pt-br>>. Citado na página 9.
- GLOTZBACH, R. J. Tecnologias atuais e prevalentes no currículo da web. **Education and New Developments 2024 – Volume 1**, 2024. Citado na página 2.
- GOOGLE. **Visão geral | Chrome DevTools**. 2016. Disponível em: <<https://developer.chrome.com/docs/devtools/overview?hl=pt-br>>. Citado 2 vezes nas páginas 1 e 2.
- GOOGLE. **Lighthouse performance scoring**. 2019. Disponível em: <<https://developer.chrome.com/docs/lighthouse/performance/performance-scoring?hl=en>>. Citado na página 7.
- KEARNEY, M.; EMELIANOVA, S. **Gravar snapshots de heap**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/memory-problems/heap-snapshots?hl=pt-br>>. Citado na página 7.
- MARTHE, D. S. **Painel “Camadas”: explore as camadas do seu site**. Disponível em: <<https://developer.chrome.com/docs/devtools/layers?hl=pt-br>>. Citado na página 15.
- MARTHE, D. S. **Monitor de protocolo: visualizar e enviar solicitações de CDP**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/protocol-monitor?hl=pt-br>>. Citado 2 vezes nas páginas 19 e 20.

MARTHE, D. S. **Painel de fonte rápida**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/quick-source?hl=pt-br>>. Citado na página 21.

MARTHE, D. S. **Painel do Gravador: registre e meça o fluxo do usuário**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/recorder/overview?hl=pt-br>>. Citado na página 8.

MARTHE, D. S. **Painel do monitor de desempenho**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/performance-monitor?hl=pt-br>>. Citado na página 19.

MARTHE, D. S. **Painel Network: analisar a carga e os recursos da rede**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/network/overview?hl=pt-br>>. Citado na página 8.

MARTHE, D. S. **Solicitações de rede: teste seu site bloqueando solicitações de rede**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/network-request-blocking?hl=pt-br>>. Citado 2 vezes nas páginas 18 e 19.

MARTHE, D. S. **Visão geral do painel de memória**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/memory?hl=pt-br>>. Citado na página 7.

MARTHE, D. S. **Visão geral do painel Elementos**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/elements?hl=pt-br>>. Citado na página 5.

MARTHE, D. S. **WebAudio: visualizar métricas da API WebAudio**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/webaudio?hl=pt-br>>. Citado na página 21.

MARTHE, D. S.; EMELIANOVA, S. **Painel de desempenho: analise o desempenho do seu site**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/performance/overview?hl=pt-br>>. Citado na página 8.

MICROSOFT LEARN. **Analyze pages using the Inspect tool**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/css/inspect>>. Citado na página 36.

MICROSOFT LEARN. **Application tool, to manage storage**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/storage/application-tool>>. Citado na página 39.

MICROSOFT LEARN. **Compose and send web API requests using the Network Console tool**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/network-console/network-console-tool>>. Citado na página 45.

MICROSOFT LEARN. **Console overview**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/console/>>. Citado na página 37.

MICROSOFT LEARN. **Crash analyzer tool**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/crash-analyzer>>. Citado na página 42.

Microsoft Learn. **Emulate authenticators and debug WebAuthn**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/webauthn>>. Citado na página 50.

MICROSOFT LEARN. **Emulate device sensors**. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/sensors>>. Citado na página 49.

MICROSOFT LEARN. **Find and fix problems using the Issues tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/issues/>>. Citado na página 43.

MICROSOFT LEARN. **Find source files for a page using the Search tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/search/search-tool>>. Citado na página 48.

MICROSOFT LEARN. **Find unused JavaScript and CSS code with the Coverage tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/coverage/>>. Citado 2 vezes nas páginas 41 e 42.

MICROSOFT LEARN. **Inspect a JavaScript ArrayBuffer with the Memory inspector tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/memory-inspector/memory-inspector-tool>>. Citado na página 43.

MICROSOFT LEARN. **Inspect and modify CSS animation effects.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/inspect-styles/animations>>. Citado na página 40.

MICROSOFT LEARN. **Inspect network activity.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/network>>. Citado 2 vezes nas páginas 38 e 39.

MICROSOFT LEARN. **Lighthouse tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/lighthouse/lighthouse-tool>>. Citado na página 43.

MICROSOFT LEARN. **Measure runtime performance of a page using the Performance monitor tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/performance-monitor/performance-monitor-tool>>. Citado na página 46.

MICROSOFT LEARN. **Navigate webpage layers, z-index, and DOM using the 3D View tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/3d-view>>. Citado na página 39.

MICROSOFT LEARN. **Network conditions tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/network-conditions/network-conditions-tool>>. Citado na página 44.

MICROSOFT LEARN. **Network request blocking tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/network-request-blocking/network-request-blocking-tool>>. Citado 2 vezes nas páginas 45 e 46.

MICROSOFT LEARN. **Optimize CSS styles with the CSS overview tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/css/css-overview-tool>>. Citado na página 43.

MICROSOFT LEARN. **Performance tool: Analyze your website's performance.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/performance/overview>>. Citado na página 39.

MICROSOFT LEARN. **Record and replay user flows and measure performance.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/recorder>>. Citado na página 47.

MICROSOFT LEARN. **Rendering tool, to see what a webpage looks like with different display options or vision deficiencies.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/rendering-tools/rendering-tool>>. Citado na página 47.

MICROSOFT LEARN. **Source Maps Monitor tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/source-maps-monitor/source-maps-monitor-tool>>. Citado 2 vezes nas páginas 49 e 50.

MICROSOFT LEARN. **Sources tool overview.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/sources>> . Citado 2 vezes nas páginas 37 e 38.

MICROSOFT LEARN. **Track changes to files using the Changes tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/changes/changes-tool>>. Citado na página 41.

MICROSOFT LEARN. **Understand security issues using the Security tool.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/security>> . Citado na página 48.

MICROSOFT LEARN. **View and debug media players information.** Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/media-panel>> . Citado na página 43.

Microsoft Learn. **WebAudio tool.** 2025. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools/webaudio/webaudio-tool>> . Citado na página 50.

MOHAMMAD, F.; YEEN, J.; EMELIANOVA, S. **WebAuthn: emular autenticadores.** 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/webauthn?hl=pt-br>> . Citado na página 22.

MOZILLA. **Firefox DevTools User Docs — Firefox Source Docs documentation.** Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/>> . Citado 2 vezes nas páginas 1 e 2.

MOZILLA. **Accessibility Inspector: inspetor de acessibilidade.** 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/accessibility_inspector> . Citado na página 25.

MOZILLA. **Application: painel de aplicação.** 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/application>> . Citado na página 26.

MOZILLA. **Custom Formatters: formatadores personalizados.** 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/custom_formatters/index.html> . Citado na página 29.

MOZILLA. **DOM Property Viewer: visualizador de propriedades do DOM.** 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/dom_property_viewer/index.html> . Citado na página 27.

MOZILLA. **Eyedropper: conta-gotas.** 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/eyedropper/index.html>> . Citado na página 28.

MOZILLA. **Firefox Profiler: documentação da ferramenta de perfil.** 2025. Disponível em: <https://profiler.firefox.com/docs/#> . Citado na página 25.

MOZILLA. **JavaScript Debugger: depurador de JavaScript**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/debugger/>>. Citado na página 24.

MOZILLA. **JavaScript Tracer: rastreador de JavaScript**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/javascript_tracer/index.html>. Citado na página 29.

MOZILLA. **Memory: ferramenta de memória**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/memory/index.html>>. Citado na página 27.

MOZILLA. **Network Monitor: monitor de rede**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/>. Citado na página 24.

MOZILLA. **Page Inspector: tour pela interface do usuário**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/page_inspector/ui_tour/index.html>. Citado na página 23.

MOZILLA. **Storage Inspector: inspetor de armazenamento**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/storage_inspector/index.html>. Citado na página 27.

MOZILLA. **Style Editor: editor de estilos**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/style_editor/index.html>. Citado 2 vezes nas páginas 24 e 25.

MOZILLA. **Web Console: tour pela interface do usuário**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/web_console/ui_tour/index.html>. Citado na página 24.

MSEDEGETEAM. **Overview of DevTools - Microsoft Edge Development**. 2023. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/overview>>. Citado na página 4.

POLLARD, B. **Lighthouse: otimize seu site**. 2025. Disponível em: <<https://developer.chrome.com/docs/devtools/lighthouse?hl=pt-br>>. Citado na página 6.

ROUSSO, D. **Audit Tab: guia da aba de auditoria**. 2020. Disponível em: <<https://webkit.org/web-inspector/audit-tab/>>. Citado 2 vezes nas páginas 35 e 36.

ROUSSO, D. **Sources Tab: guia da aba de fontes e recursos**. 2020. Disponível em: <<https://webkit.org/web-inspector/sources-tab/>>. Citado na página 32.

ROUSSO, D. **Elements Tab: guia da aba de elementos**. 2021. Disponível em: <<https://webkit.org/web-inspector/elements-tab/>>. Citado 2 vezes nas páginas 30 e 31.

ROUSSO, D.; KIRSLING, R.; FRASER, S. **Layers Tab: guia da aba de camadas**. 2020. Disponível em: <<https://webkit.org/web-inspector/layers-tab/>>. Citado 2 vezes nas páginas 34 e 35.

YEEN, J. **Visão geral do CSS: identifique possíveis melhorias no CSS**. 2021. Disponível em: <<https://developer.chrome.com/docs/devtools/css-overview?hl=pt-br>>. Citado na página 15.

YEEN, J. **Inspecionar e depurar layouts Flexbox CSS**. 2022. Disponível em: <<https://developer.chrome.com/docs/devtools/css/flexbox?hl=pt-br>>. Citado na página 5.

- YEEN, J.; EMELIANOVA, S. **Inspecionar layouts de grade CSS**. 2021. Disponível em: <<https://developer.chrome.com/docs/devtools/css/grid?hl=pt-br>>. Citado na página 5.
- YEEN, J.; EMELIANOVA, S. **Memory Inspector: inspecionar ArrayBuffer, TypedArray, DataView e Wasm Memory**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/memory-inspector?hl=pt-br>>. Citado 2 vezes nas páginas 16 e 17.
- YEEN, J.; MARTHE, D. S. **Mídia: visualizar e depurar informações dos players de mídia**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/media-panel?hl=pt-br>>. Citado na página 16.