

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

LUCAS PEREIRA MACHADO

**FERRAMENTAS DE DESENVOLVIMENTO WEB: UMA ANÁLISE
COMPARATIVA ENTRE OS PRINCIPAIS NAVEGADORES E SUAS
APLICAÇÕES PRATICAS**

PROPOSTA DE TRABALHO DE CONCLUSÃO DE CURSO

TOLEDO
18 DE NOVEMBRO DE 2025

LUCAS PEREIRA MACHADO

**FERRAMENTAS DE DESENVOLVIMENTO WEB: UMA ANÁLISE
COMPARATIVA ENTRE OS PRINCIPAIS NAVEGADORES E SUAS
APLICAÇÕES PRATICAS**

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
18 DE NOVEMBRO DE 2025

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.

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) ([FIREFOX SOURCE TREE DOCUMENTATION](#)), 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 ([CHROME FOR DEVELOPERS, 2016](#)).

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 [<https://gs.statcounter.com/browser-market-share#monthly-202401-202501>](https://gs.statcounter.com/browser-market-share#monthly-202401-202501)

2 Objetivos

Este capítulo será apresentado o objetivo geral do trabalho e em seguida os objetivos específicos.

2.1 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.

2.2 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.

3 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 uma tarefa desafiadora.

3.1 Relevância do desenvolvimento web

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 ([GLOTZBACH, 2024](#)). 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.

3.2 Lacunas de conhecimento

Apesar da vasta quantidade de informações disponíveis sobre desenvolvimento web, ainda existem lacunas de conhecimento em relação à comparação detalhada entre os principais navegadores e suas aplicações práticas. Dentre essas dificuldades, destaca-se a compreensão limitada sobre as ferramentas de análise de código estático, que possibilitam a identificação de erros antes da execução do código, contribuindo para a detecção precoce de inconsistências e a melhoria da qualidade do software ([ODELL, 2014](#)). Essa lacuna de conhecimento pode levar à escolha de ferramentas subótimas, impactando negativamente a qualidade, o desempenho e a manutenibilidade das aplicações.

3.3 Contribuição com a comunidade

Este trabalho visa contribuir para a comunidade de desenvolvedores web ao fornecer uma análise exploratória dos principais navegadores disponíveis no mercado. Ao identificar as vantagens e desvantagens de cada ferramenta, o estudo poderá auxiliar os desenvolvedores a tomar decisões mais informadas na escolha da ferramenta ideal para seus projetos. Além disso, a pesquisa poderá identificar tendências e direcionar futuras pesquisas na área de desenvolvimento web, impulsionando a evolução da área.

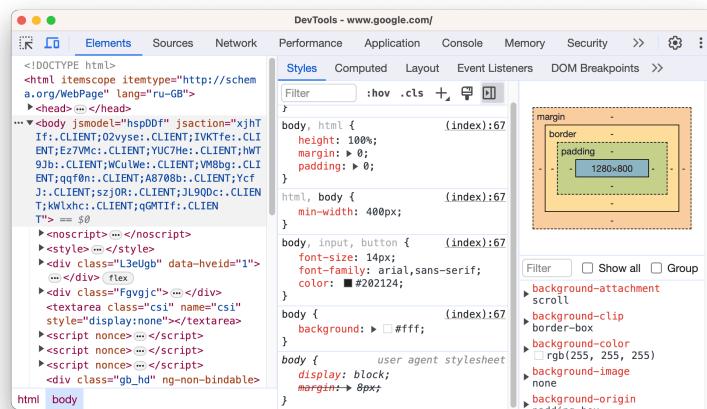
4 Materiais utilizados

As ferramentas de desenvolvimento do navegador, também conhecidas como *DevTools*, são conjuntos de ferramentas incorporadas nos navegadores da web, como Google Chrome e Mozilla Firefox. Essas ferramentas permitem que os desenvolvedores da web inspecionem e depurem o código-fonte da página, além de analisar o desempenho, uso de memória de seus aplicativos da web, entre outros recursos ([APPLE DEVELOPER DOCUMENTATION, 2025](#)). Durante o desenvolvimento deste trabalho foram utilizadas as ferramentas listadas abaixo.

4.1 Chrome DevTools

O Chrome DevTools, Figura 1, representam 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, simular diferentes dispositivos, entre outros recursos. ([CHROME FOR DEVELOPERS, 2016](#)).

Figura 1 – Chrome Devtools

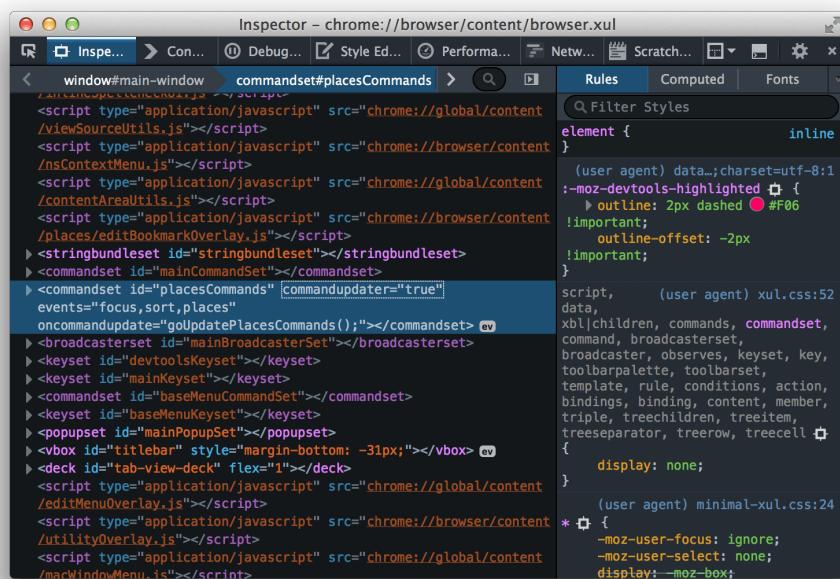


Fonte: Chrome for Developers

4.2 Firefox Developer Tools

O Firefox Developer Tools, ilustrado na Figura 2, oferece uma alternativa robusta aos Chrome DevTools, com um foco em privacidade e personalização. Essas ferramentas proporcionam um ambiente de desenvolvimento completo, permitindo a inspeção de elementos, a análise de redes, o depurador de JavaScript, a visualização de estilos CSS e a otimização do desempenho. Além disso, o Firefox Developer Tools se integra perfeitamente com outras ferramentas do ecossistema Mozilla, como o editor de código Visual Studio Code, facilitando o fluxo de trabalho dos desenvolvedores ([FIREFOX SOURCE TREE DOCUMENTATION](#),).

Figura 2 – Firefox Developer Tools



Fonte: [Firefox Source Tree Documentation](#)

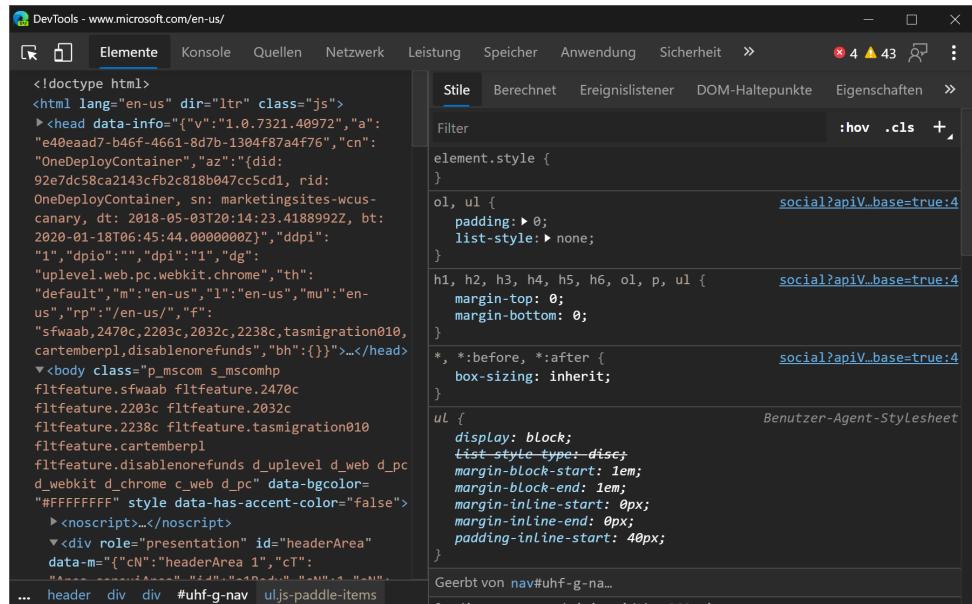
4.3 Edge DevTools

Os Edge DevTools, ilustrado na Figura 3, incorporados ao navegador Microsoft Edge, oferecem uma experiência de desenvolvimento moderna e eficiente. Com uma interface visualmente atraente e funcionalidades similares aos Chrome DevTools, os Edge DevTools permitem inspecionar elementos, depurar JavaScript, analisar o desempenho da página e simular diferentes dispositivos. A integração com o Microsoft Visual Studio Code e outras ferramentas do ecossistema Microsoft torna os Edge DevTools uma opção atraente para desenvolvedores que utilizam a plataforma Windows([MSEdgeTeam, 2023](#)).

4.4 Safari Web Inspector

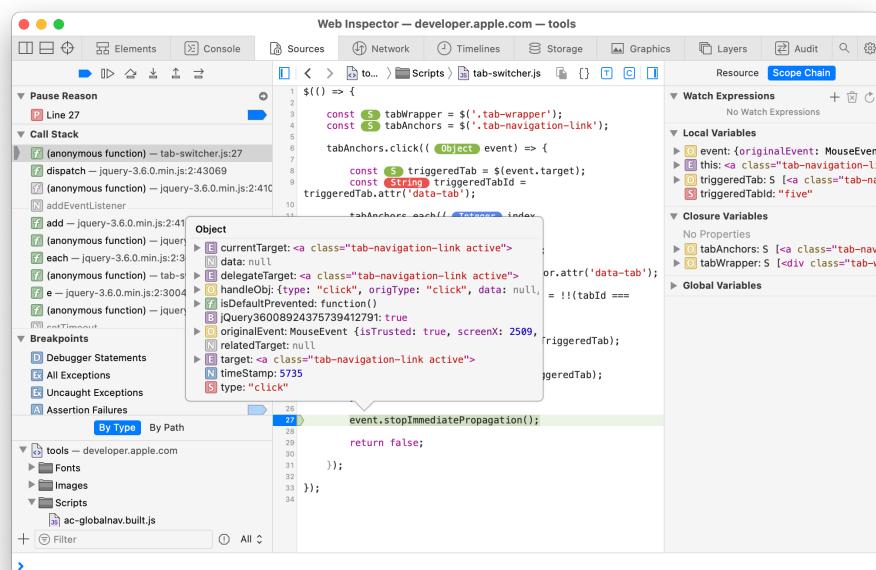
O Safari Web Inspector, ilustrado na Figura 4, é a ferramenta de desenvolvimento integrada ao navegador Apple Safari, projetada para oferecer uma experiência de depuração e desenvolvimento de alta qualidade. Com um foco em desempenho e usabilidade, o Web Inspector permite inspecionar elementos, depurar JavaScript, analisar o desempenho da página e simular diferentes dispositivos([APPLE DEVELOPER DOCUMENTATION, 2025](#)).

Figura 3 – Edge DevTools



Fonte: Microsoft Learn

Figura 4 – Safari Web Inspector



Fonte: Apple Developer Documentation

5 Metodologia

Para o desenvolvimento deste trabalho optou-se por uma abordagem de análise exploratória. Este método se mostra especialmente adequada para este tema. Este método permite um exame detalhado e flexível do tema, proporcionando análises valiosas sobre as funcionalidades e características das ferramentas investigadas, bem como suas aplicações práticas no contexto do desenvolvimento web.

O campo é caracterizado por um rápido avanço tecnológico e pela ampla diversidade de soluções disponíveis. Assim, uma análise exploratória permite identificar padrões, relações e tendências que não seriam facilmente percebidos por meio de abordagens mais restritivas ou estruturadas.

5.1 Analise exploratória

A análise exploratória é uma metodologia qualitativa que busca compreender fenômenos, identificar tendências e levantar hipóteses a partir de dados ou informações iniciais (BEHRENS, 1997). Em contraste com métodos estruturados, que partem de hipóteses previamente definidas, a análise exploratória permite maior flexibilidade e adaptação durante o processo, sendo amplamente utilizada em pesquisas iniciais ou contextos pouco conhecidos.

No contexto deste estudo, a análise exploratória se justifica por permitir uma investigação aberta das ferramentas de desenvolvimento web, explorando desde as funcionalidades nativas até extensões e práticas comuns entre desenvolvedores. Com isso, essa metodologia será utilizada para:

- **Analizar as características das ferramentas:** comparar as funcionalidades, linguagens suportadas, comunidades e ecossistemas de cada ferramenta.
- **Identificar tendências:** observar a evolução das ferramentas ao longo do tempo, a emergência de novas tecnologias e a obsolescência de outras
- **Avaliar a percepção dos desenvolvedores:** por meio de pesquisas, compreender as preferências, necessidades e desafios dos desenvolvedores na escolha de ferramentas.

A aplicação dessa metodologia é realizada em etapas sequenciais, que envolvem desde a coleta de dados até a interpretação dos resultados obtidos.

6 Pesquisa de campo

Entre janeiro e abril de 2025, foi conduzida uma pesquisa de campo para compreender a percepção dos desenvolvedores de software sobre as *devtools*, explorando suas preferências, usos no cotidiano e impactos em diferentes contextos e níveis de experiência. O objetivo foi identificar as funcionalidades mais valorizadas, os desafios enfrentados e a eficiência dessas ferramentas no fluxo de trabalho dos profissionais.

A pesquisa utilizou um formulário estruturado, aplicado por meio de uma plataforma digital, para coletar dados de um público amplo e diversificado. O formulário incluiu questões específicas para filtrar respostas, garantindo que apenas desenvolvedores fossem considerados, excluindo participantes fora do perfil, como não desenvolvedores web, o que assegurou a confiabilidade e validade dos resultados.

Ao todo, foram obtidas um total de 80 respostas, com participantes de diferentes estados brasileiros e diferentes níveis de experiência. Após a aplicação dos critérios de seleção, que filtraram os participantes para incluir apenas desenvolvedores web, 45 respostas foram selecionadas para análise, garantindo a relevância e a confiabilidade dos dados utilizados no estudo.

6.1 Resultados obtidos

Os resultados revelam que as ferramentas de desenvolvimento são essenciais para o desenvolvimento front-end, sendo o Google Chrome o navegador dominante e a inspeção de elementos e o console os recursos mais utilizados. Embora muitos desenvolvedores considerem as DevTools suficientes, especialmente em níveis júnior e intermediário, desenvolvedores seniores frequentemente as complementam com ferramentas ou plugins de IDE, indicando que a complexidade do projeto influencia as preferências de ferramentas.

Um fato interessante é que apenas 27% dos desenvolvedores sabiam que o *Chrome DevTools* oferece mais de 20 ferramentas, apesar de 67% usarem o *DevTools* diariamente. Isso sugere uma lacuna no conhecimento de todos os recursos oferecidos, especialmente entre desenvolvedores juniores e intermediários, que frequentemente utilizam um conjunto limitado de ferramentas, como inspeção de elementos e console.

A dificuldade moderada percebida por desenvolvedores juniores e sua dependência de recursos externos, como tutoriais e colegas, ressaltam a necessidade de melhores recursos educacionais para preencher a lacuna de conhecimento. Essas percepções sugerem que, embora essas ferramentas sejam valiosas, o aumento da divulgação e a oferta de guias e treinamento sobre seus recursos completos podem aprimorar sua eficácia no desenvolvimento web.

7 Ferramentas disponibilizadas pelos navegadores

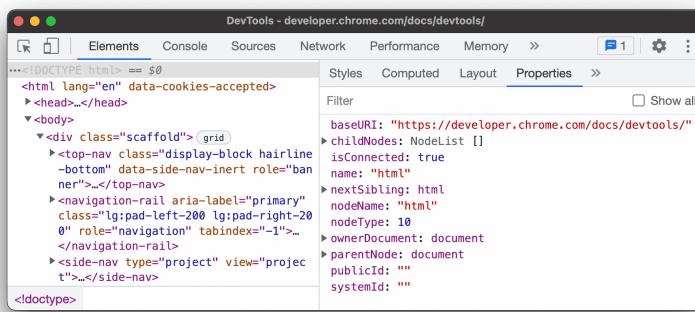
Para compreender melhor as possibilidades oferecidas pelas DevTools e como elas podem ser otimizadas, esta seção apresenta uma lista das ferramentas disponíveis nos principais navegadores do mercado atualmente, explorando suas funcionalidades, onde estão disponíveis e aplicações práticas no desenvolvimento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

7.1 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.

7.1.1 Elementos

Figura 5 – Ferramenta "Elementos" do Chrome Devtools



Fonte: Chrome for Developers

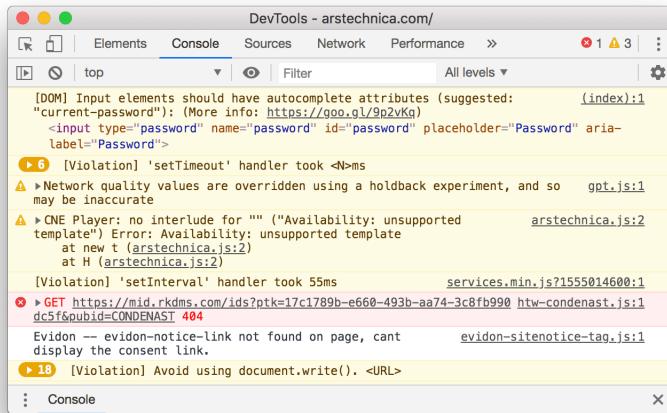
A ferramenta "Elementos" (*Elements*) é utilizada para realizar a inspeção e manipulação em tempo real do *Document Object Model (DOM)*. Ele renderiza a estrutura da página em uma árvore *DOM* hierárquica, como apresentado na Figura 5, permitindo a seleção precisa e a modificação direta de nós. As funcionalidades de edição permitem alterar dinamicamente o conteúdo textual, modificar atributos, alterar o tipo de nó e reordenar elementos estruturalmente (MARTHE, 2024h). 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* (BASQUES; EMELIANOVA, 2019).

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 (BASQUES; EMELIANOVA, 2024a; EMELIANOVA, 2022b). De forma complementar, agrega ainda ferramentas especializadas para

a depuração de *layouts* complexos, como *Grid* e *Flexbox* (YEEN; EMELIANOVA, 2021; YEEN, 2022).

7.1.2 Console

Figura 6 – Ferramenta "Console" do Chrome Devtools



Fonte: Chrome for Developers

O painel "Console" (*Console*) atua como uma interface primária para o diagnóstico e interação com aplicações *web*, como exibido na figura 6, servindo a duas funções principais: a visualização de mensagens registradas e a execução de *JavaScript* (BASQUES, 2024b). Desenvolvedores utilizam a *API* do *Console*, através de métodos, para enviar mensagens de diagnóstico durante 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 (BASQUES, 2024b). 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 (BASQUES, 2024b).

7.1.3 Problemas

Figura 7 – Ferramenta "Problemas" do Chrome Devtools

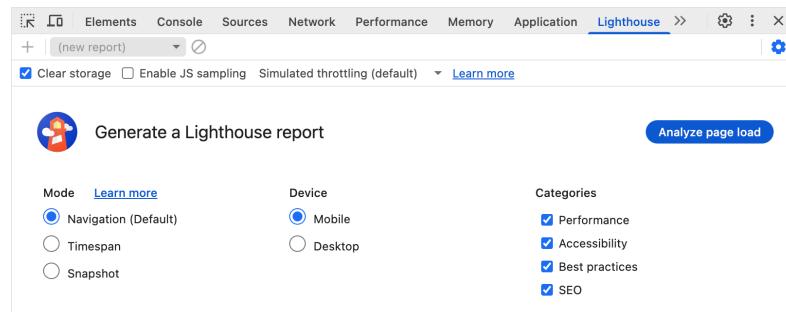
Fonte: Chrome for Developers

O painel "Problemas" (*Issues*), apresentado na figura 7, 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*) (DUTTON; EMELIANOVA, 2020). Sua função é reduzir a verbosidade e a desordem de notificações no "Console" 7.1.2, apresentando os problemas de forma estruturada, agregada e açãoável. 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” 7.1.1. O recurso também permite agrupar os problemas por tipo e filtrar a exibição de problemas relacionados a *cookies* de terceiros (DUTTON; EMELIANOVA, 2020).

7.1.4 Lighthouse

Figura 8 – Ferramenta “Lighthouse” do Chrome Devtools



Fonte: Chrome for Developers

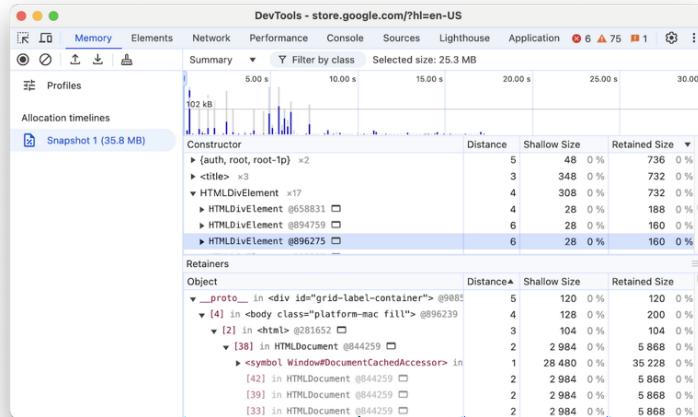
Esta ferramenta é projetada para realizar auditorias abrangentes da qualidade de aplicações *web*, como exibido na figura 8. Sua principal função é gerar um relatório detalhado que avalia o *site* em diversas categorias fundamentais, including Desempenho, Acessibilidade, Práticas Recomendadas e *SEO* (POLLARD, 2025). 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) (POLLARD, 2025). 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*.

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) (CHROME FOR DEVELOPERS, 2019).

7.1.5 Memória

O painel “Memória” (*Memory*), apresentado na figura 9, 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 (MARTHE, 2024g). 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* (MARTHE, 2024g). A análise dos snapshots de *heap* permite a identificação de nós *DOM* separados (*detached DOM nodes*), uma causa comum de vazamentos de memória (KEARNEY; EMELIANOVA, 2024).

Figura 9 – Ferramenta "Memória" do Chrome Devtools

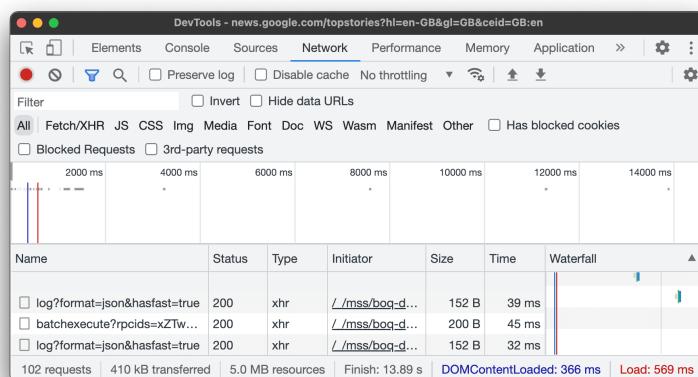


Fonte: Chrome for Developers

7.1.6 Rede

O painel "Rede" (Network) é uma ferramenta de diagnóstico, como exibido na figura 10 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 (MARTHE, 2024e). 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*) (BASQUES, 2024a). A ferramenta também permite a exportação dos dados de atividade de rede em formato *HAR* (*HTTP Archive*) (BASQUES, 2024a).

Figura 10 – Ferramenta "Rede" do Chrome Devtools

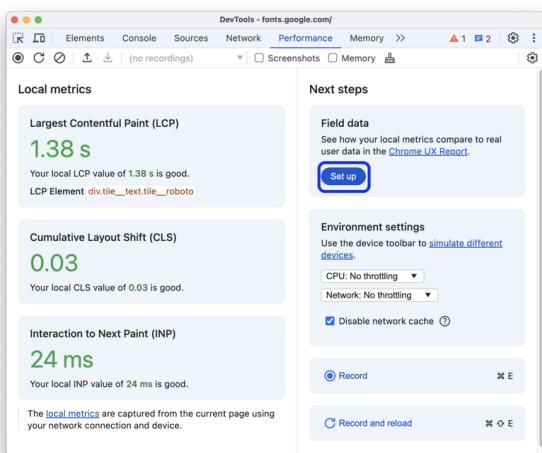


Fonte: Chrome for Developers

7.1.7 Desempenho

O painel “Desempenho” (*Performance*), exibido na figura 11 abaixo, é uma ferramenta de diagnóstico avançada projetada para gravar e analisar perfis de desempenho da *CPU* em aplicações *web* (MARTHE; EMELIANOVA, 2024). 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* (MARTHE; EMELIANOVA, 2024). 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*) (MARTHE; EMELIANOVA, 2024).

Figura 11 – Ferramenta “Desempenho” do Chrome Devtools

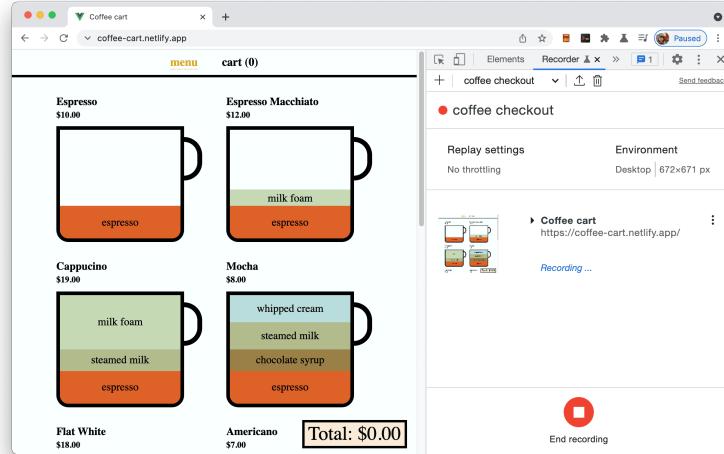


Fonte: Chrome for Developers

7.1.8 Gravador

O painel “Gravador” (*Recorder*), apresentado na figura 12 abaixo, constitui uma ferramenta de diagnóstico avançada, projetada para a análise e depuração de fluxos de interação do usuário em aplicações *web* (MARTHE, 2024c). 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 (MARTHE, 2024c). 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 (MARTHE, 2024c).

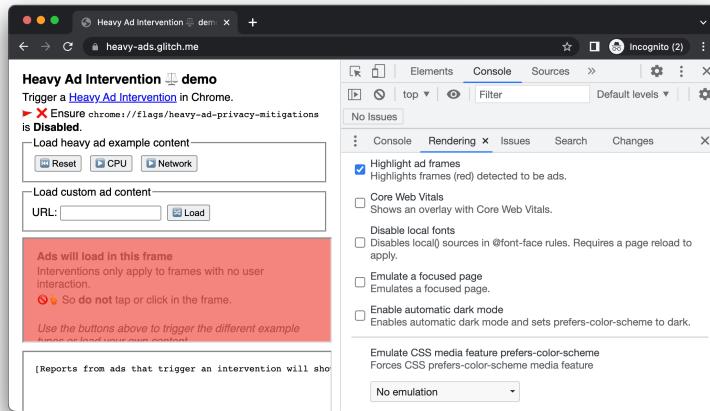
Figura 12 – Ferramenta "Gravador" do Chrome Devtools



Fonte: Chrome for Developers

7.1.9 Renderização

Figura 13 – Ferramenta "Renderização" do Chrome Devtools

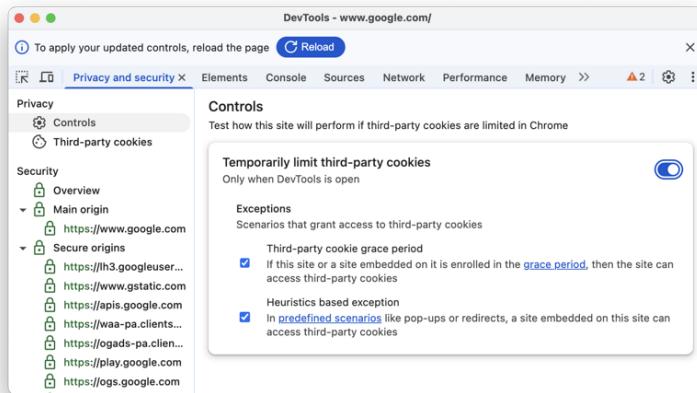


Fonte: Chrome for Developers

O painel "Renderização" (*Rendering*), exibido na figura 13 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* (EMELIANOVA; BASQUES, 2022). 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 (EMELIANOVA, 2022a). 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 (EMELIANOVA, 2022a).

7.1.10 Segurança

Figura 14 – Ferramenta "Segurança" do Chrome Devtools

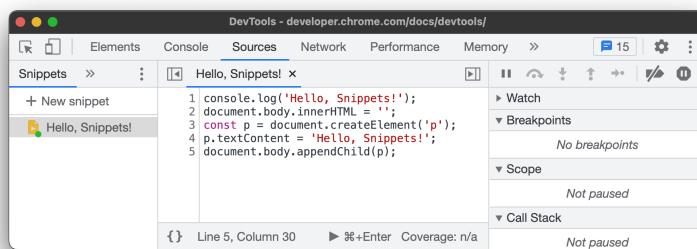


Fonte: Chrome for Developers

O painel "Privacidade e Segurança", apresentado acima na figura 14, 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, exibindo detalhes da conexão, avisos de segurança *HTTP* e informações de certificados (BASQUES; EMELIANOVA, 2025). 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 (BASQUES; EMELIANOVA, 2025).

7.1.11 Fontes

Figura 15 – Ferramenta "Fontes" do Chrome Devtools



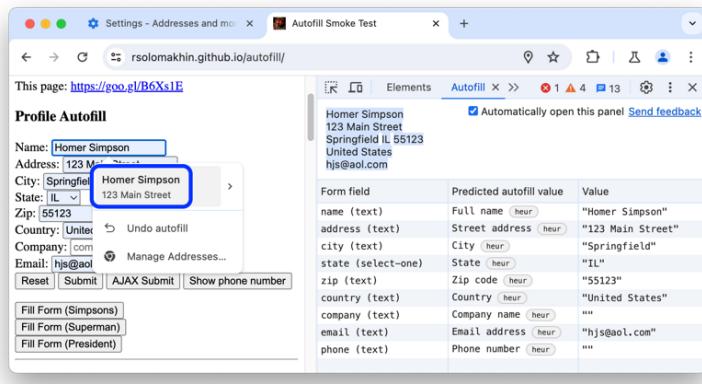
Fonte: Chrome for Developers

O painel "Fontes" (Sources), como pode ser visto na figura 15, 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 (BASQUES; EMELIANOVA, 2024d). 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 (BASQUES; EMELIANOVA, 2024d).

7.1.12 Autofil

Figura 16 – Ferramenta "Autofill"do Chrome Devtools



Fonte: Chrome for Developers

O painel "Autocomplete"(Preenchimento automático), exibido na figura 16, 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 (EMELIANOVA, 2024b). 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 (EMELIANOVA, 2024b). Adicionalmente, ele auxilia na identificação de problemas de implementação, que são relatados no painel "Issues"(Problemas) 7.1.3 para corrigir atributos de preenchimento automático incorretos.

7.1.13 Animações

O painel "Animações"(Animations), ilustrada na figura 17, 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* (BASQUES; EMELIANOVA, 2024b). 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* (BASQUES; EMELIANOVA, 2024b). 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 (BASQUES; EMELIANOVA, 2024b).

Figura 17 – Ferramenta "Animações" do Chrome Devtools

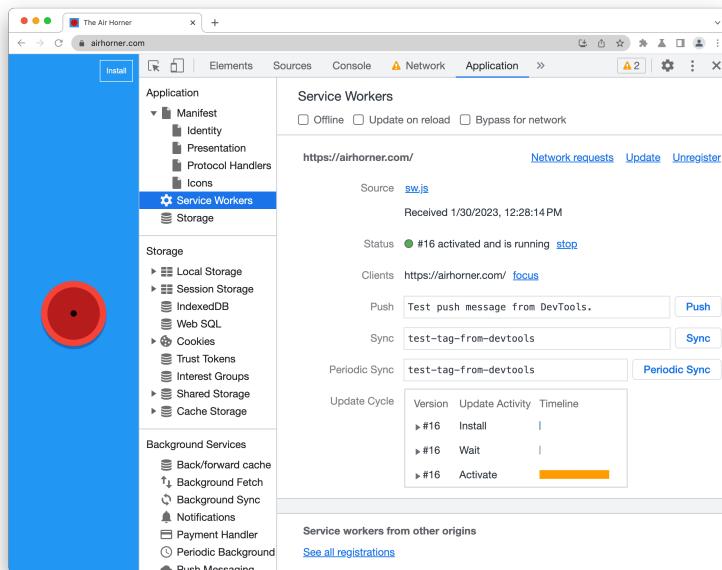


Fonte: Chrome for Developers

7.1.14 Aplicativo

O painel "*Application*"(Aplicativo), exibida na figura 18, é uma ferramenta de diagnóstico abrangente utilizada para inspecionar, modificar e depurar os diversos aspectos 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 (BASQUES; EMELIANOVA, 2016). A ferramenta oferece acesso granular para visualização e edição de múltiplos mecanismos de armazenamento do lado do cliente, incluindo *LocalStorage*, *Session Storage*, *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.

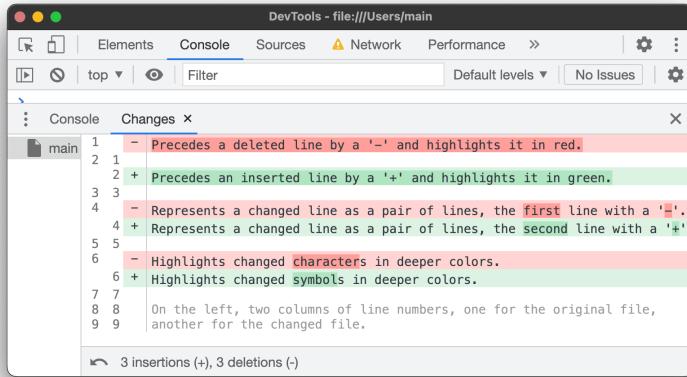
Figura 18 – Ferramenta "Aplicativo" do Chrome Devtools



Fonte: Chrome for Developers

7.1.15 Alterações

Figura 19 – Ferramenta "Alterações" do Chrome Devtools



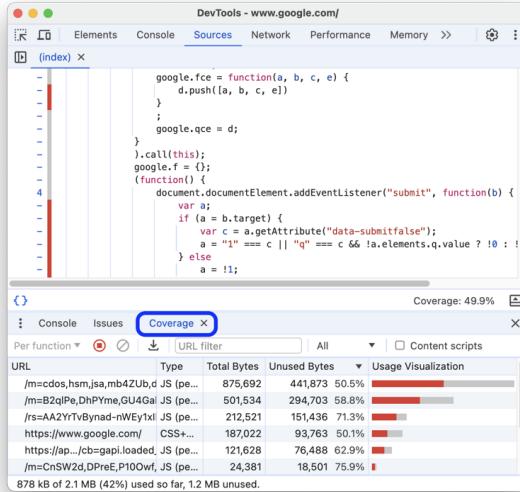
Fonte: Chrome for Developers

O painel "Alterações" (*Changes*), apresentado na figura 19, 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" 7.1.11 (EMELIANOVA, 2024a). O rastreamento de alterações em HTML também é suportado, 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 (EMELIANOVA, 2024a). 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.

7.1.16 Cobertura

O painel "Cobertura" (*Coverage*), exibido na figura 20, 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 (BASQUES; EMELIANOVA, 2024c). 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" 7.1.11, 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 (BASQUES; EMELIANOVA, 2024c).

Figura 20 – Ferramenta "Cobertura" do Chrome Devtools

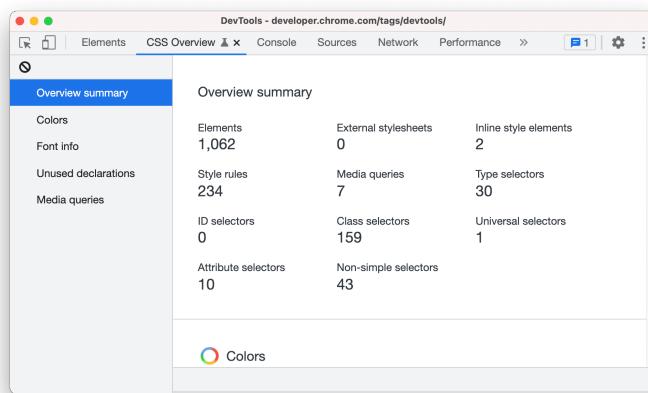


Fonte: Chrome for Developers

7.1.17 Visão geral de CSS

O painel "Visão geral de CSS" (CSS Overview), como ilustrado na figura 21, 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 (YEEN, 2021). 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" (YEEN, 2021). A ferramenta permite a investigação interativa dos elementos afetados, oferecendo funcionalidades para destacá-los na página ou inspecioná-los diretamente no painel "Elementos".

Figura 21 – Ferramenta "Visão geral de CSS" do Chrome Devtools

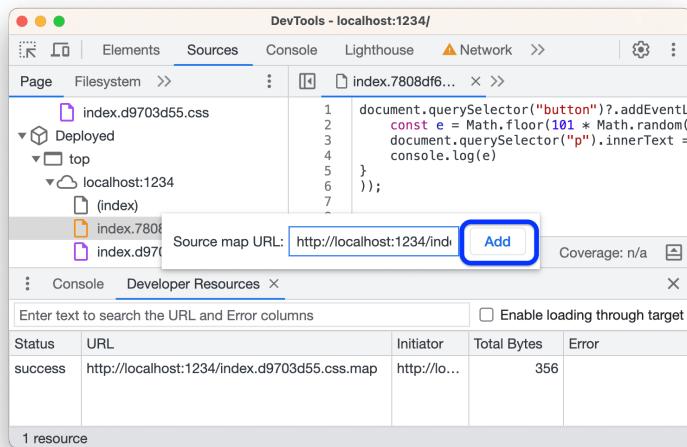


Fonte: Chrome for Developers

7.1.18 Recursos para desenvolvedores

O painel "Recursos para desenvolvedores"(*Developer Resources*), exibida na figura 22, é uma ferramenta de diagnóstico utilizada para verificar o status de carregamento dos mapas de origem (*source maps*) pelo *Chrome DevTools* (EMELIANOVA, 2023). 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* (EMELIANOVA, 2023). 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.

Figura 22 – Ferramenta "Recursos para desenvolvedores" do Chrome Devtools

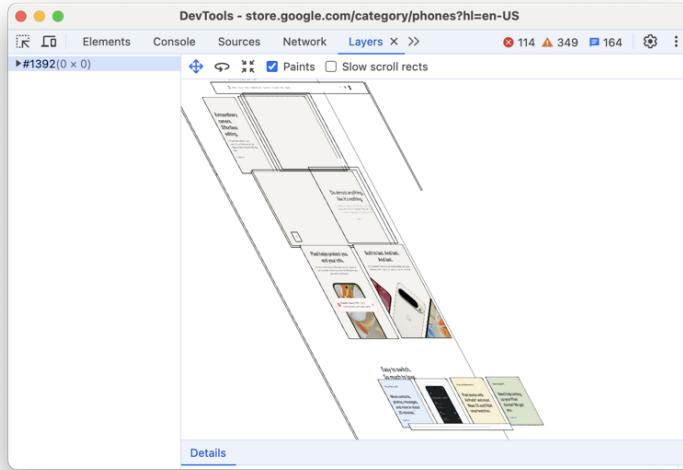


Fonte: Chrome for Developers

7.1.19 Camadas

O painel "Camadas"(*Layers*), apresentado na figura 23, 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 (MARTHE,). 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*) (MARTHE,). Funcionalidades adicionais incluem a capacidade de inspecionar informações do "Paint Profiler" e identificar regiões de rolagem lenta (*Slow scroll rects*).

Figura 23 – Ferramenta "Camadas" do Chrome Devtools

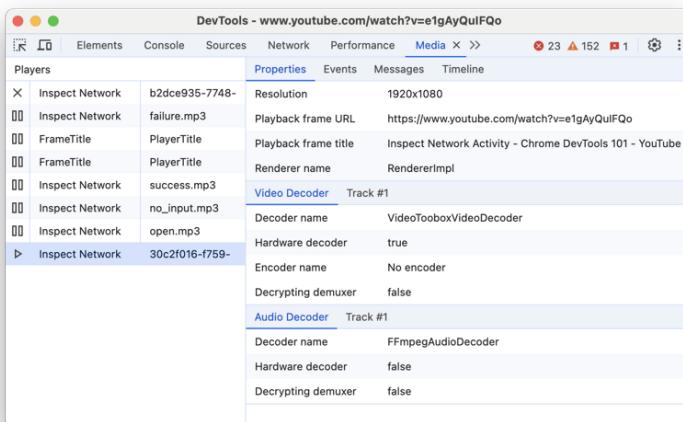


Fonte: Chrome for Developers

7.1.20 Mídia

O painel "Mídia"(*Media*), ilustrado na figura 24, 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* (YEEN; MARTHE, 2024). 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* (YEEN; MARTHE, 2024). O painel também permite a exportação das informações do *player* para análise externa.

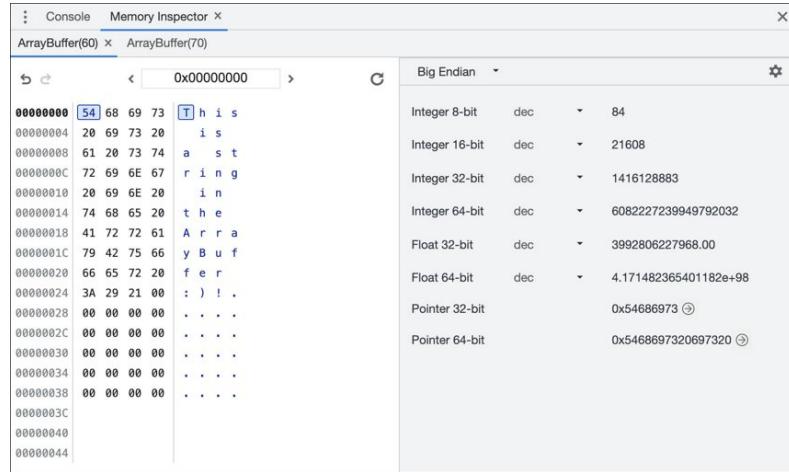
Figura 24 – Ferramenta "Mídia" do Chrome Devtools



Fonte: Chrome for Developers

7.1.21 Inspetor de memória

Figura 25 – Ferramenta "Inspetor de memória" do Chrome Devtools



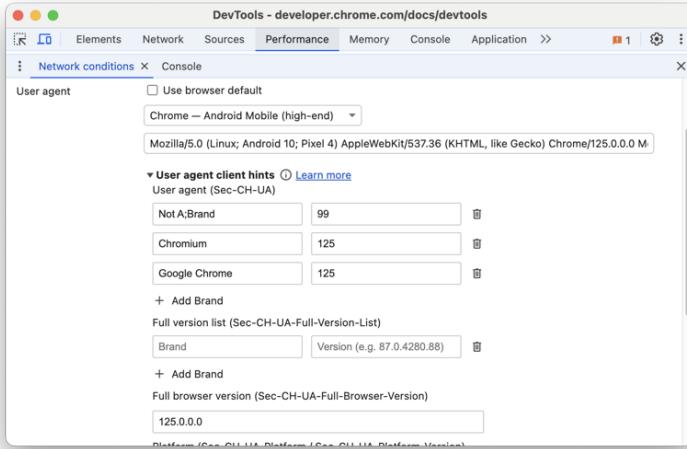
Fonte: Chrome for Developers

O "Inspetor de memória" (*Memory Inspector*), exibido na figura 25, 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++ (YEEN; EMELIANOVA, 2024). A interface exibe o *buffer* de memória 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 (YEEN; EMELIANOVA, 2024). A ferramenta permite a navegação direta para endereços de memória específicos, a alternância de *endianidade* (*endianness*) 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 (YEEN; EMELIANOVA, 2024).

7.1.22 Condições de rede

O painel "Condições de rede" (*Network Conditions*), apresentado na figura 26, 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 (BASQUES; MARTHE, 2024). 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 (BASQUES; MARTHE, 2024).

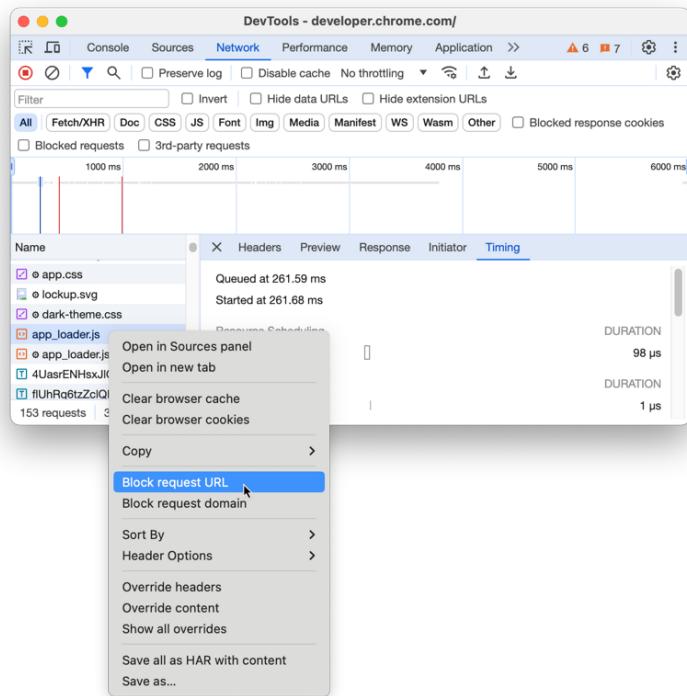
Figura 26 – Ferramenta "Condições de rede" do Chrome Devtools



Fonte: Chrome for Developers

7.1.23 Bloqueio de solicitações de rede

Figura 27 – Ferramenta "Bloqueio de solicitações de rede" do Chrome Devtools



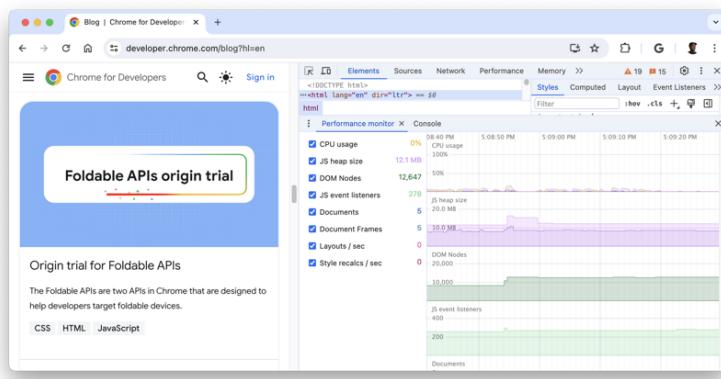
Fonte: Chrome for Developers

O painel "Bloqueio de solicitações de rede" (*Network Request Blocking*), exibido na figura 27, 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 (MARTHE, 2024f). 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 (MARTHE, 2024f). 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" 7.1.6. Uma vez ativado, o painel contabiliza e exibe o número de solicitações interceptadas que correspondem a cada padrão definido (MARTHE, 2024f).

7.1.24 Monitor de Desempenho

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



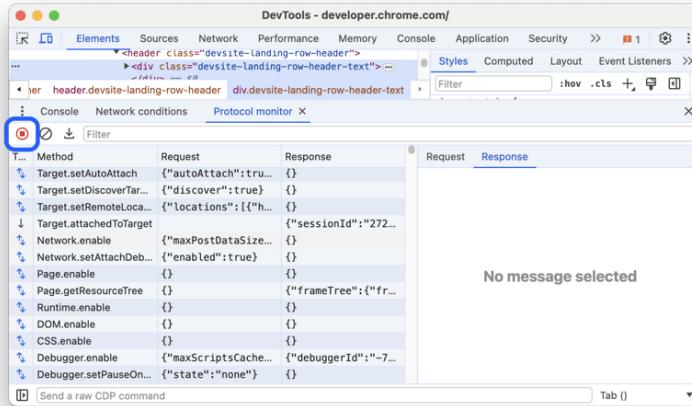
Fonte: Chrome for Developers

O "Monitor de Desempenho" (*Performance Monitor*), como ilustrado na figura 28, 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 (MARTHE, 2024d). 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 (MARTHE, 2024d). 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*.

7.1.25 Monitor de protocolo

O "Monitor de protocolo" (*Protocol Monitor*), apresentado na figura 29, 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 (MARTHE, 2024a). 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 (MARTHE, 2024a). Adicionalmente, o monitor permite o *download* das mensagens registradas em formato *JSON* para análise externa.

Figura 29 – Ferramenta "Monitor de protocolo" do Chrome Devtools

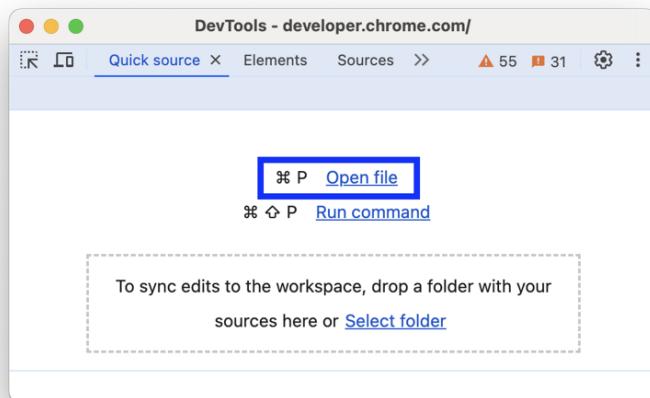


Fonte: Chrome for Developers

7.1.26 Origem rápida

O painel "Origem rápida"(*Quick source*), exibido na figura 30, 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 (MARTHE, 2024b). Esta funcionalidade elimina a necessidade de alternar repetidamente entre a ferramenta "Fontes"(*Sources*) 7.1.11 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 (MARTHE, 2024b).

Figura 30 – Ferramenta "Origem rápida" do Chrome Devtools

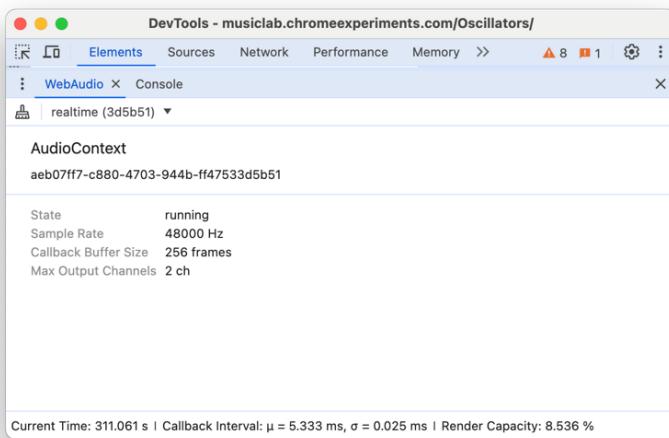


Fonte: Chrome for Developers

7.1.27 WebAudio

O painel "WebAudio", apresentado na figura 31, é uma ferramenta de diagnóstico que exibe métricas de desempenho para instâncias de *AudioContext* em aplicações que utilizam a API *WebAudio* (MARTHE, 2024i). 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*) (MARTHE, 2024i). Adicionalmente, a ferramenta monitora em tempo real o Intervalo de *Callback* (*Callback Interval*) e a Capacidade de Renderização (*Render Capacity*) do processador de áudio.

Figura 31 – Ferramenta "WebAudio" do Chrome Devtools

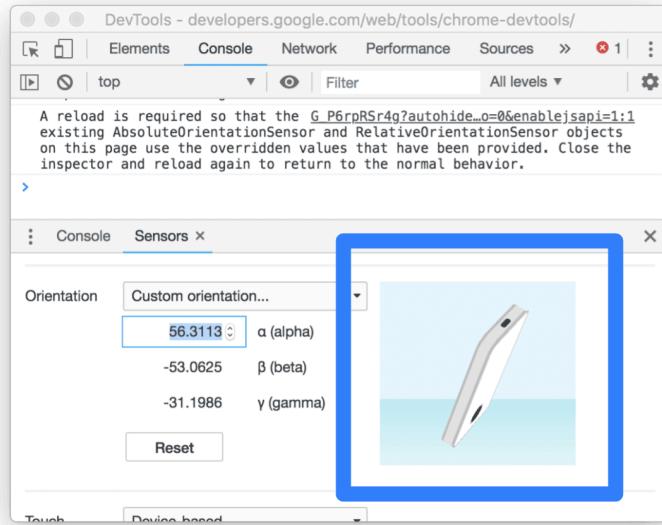


Fonte: Chrome for Developers

7.1.28 Sensores

O painel "Sensores" (*Sensors*), ilustrado na figura 32, 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) (BASQUES; EMELIANOVA, 2020). 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 32 – Ferramenta "Sensores" do Chrome Devtools

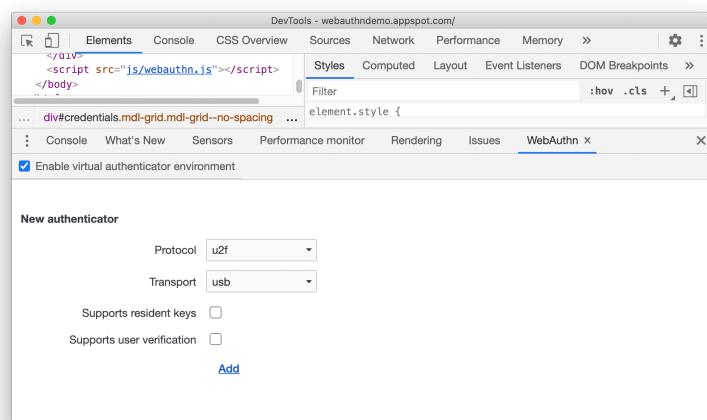


Fonte: Chrome for Developers

7.1.29 WebAuthn

O painel "WebAuthn", apresentado na figura 33, fornece uma interface para a criação e interação com autenticadores virtuais baseados em software, permitindo a depuração da API Web Authentication (MOHAMMAD; YEEN; EMELIANOVA, 2024). 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 (MOHAMMAD; YEEN; EMELIANOVA, 2024).

Figura 33 – Ferramenta "WebAuthn" do Chrome Devtools



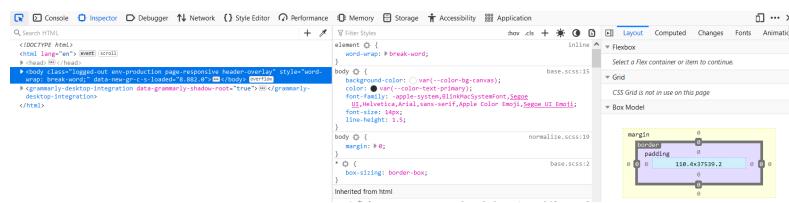
Fonte: Chrome for Developers

7.2 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.

7.2.1 Inspetor

Figura 34 – Ferramenta "Inspetor" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

O Inspetor de Página (Page Inspector), como apresentado na figura 34, 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 (MOZILLA, 2025k). 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 (MOZILLA, 2025k). 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.

7.2.2 Console

Figura 35 – Ferramenta "Console" do Firefox Devtools



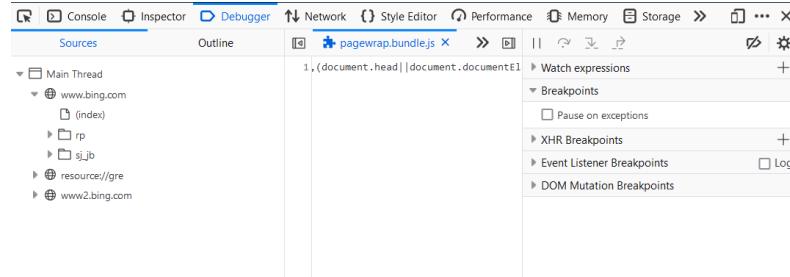
Fonte: Firefox Source Tree Documentation

O Console Web (Web Console), exibido na figura 35, 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 no contexto da página (MOZILLA, 2025n). 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.

7.2.3 Depurador

Figura 36 – Ferramenta "Depurador" do Firefox Devtools

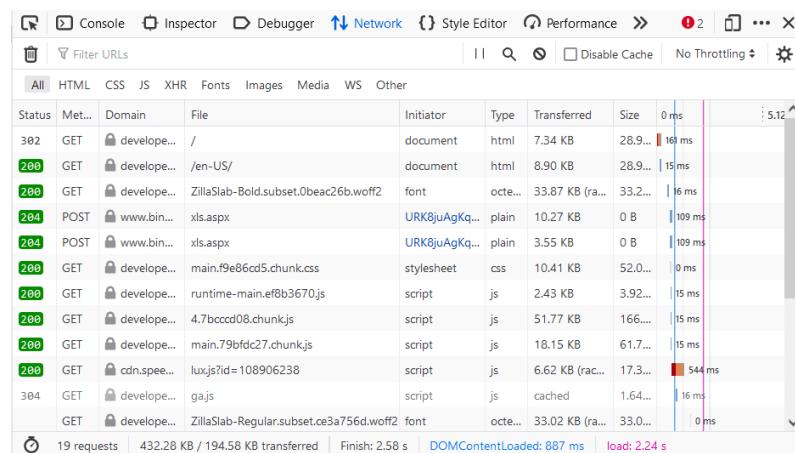


Fonte: Firefox Source Tree Documentation

O Depurador JavaScript (JavaScript Debugger), como ilustrado na figura 36, 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 (MOZILLA, 2025g). 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 (MOZILLA, 2025g). 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.

7.2.4 Rede

Figura 37 – Ferramenta "Rede" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

O Monitor de Rede (Network Monitor), exibido na figura 37, 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

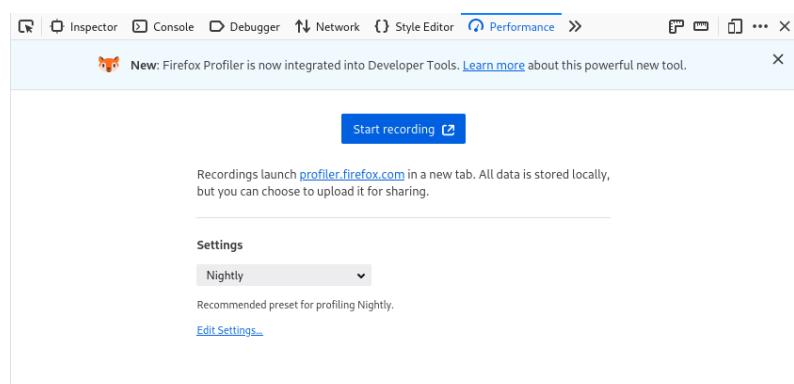
de Ferramentas) estiver aberta, independentemente de o painel de Rede estar ativamente selecionado (MOZILLA, 2025j). Suas capacidades estendem-se à inspeção de Web Sockets, Server-Sent Events (SSE) e à simulação de diferentes condições de rede (throttling) (MOZILLA, 2025j).

7.2.5 Editor de Estilo

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 (MOZILLA, 2025m). 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), permitindo que desenvolvedores depurem o código-fonte original (ex: Sass, Less) em vez do CSS transpilado ou minificado (MOZILLA, 2025m).

7.2.6 Desempenho

Figura 38 – Ferramenta "Desempenho" do Firefox Devtools



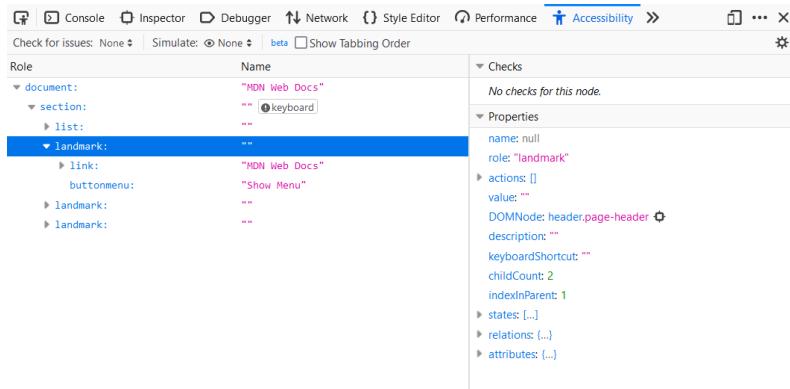
Fonte: Firefox Source Tree Documentation

O painel Desempenho (Performance Panel), como apresentado na figura 38, é a ferramenta designada para a análise da responsividade geral, da execução de JavaScript e do desempenho de layout de uma aplicação web (MOZILLA, 2025f). 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 (MOZILLA, 2025f). Os dados capturados permanecem armazenados localmente, sendo que a ferramenta facilita o upload e compartilhamento desses perfis para análise colaborativa.

7.2.7 Inspetor de acessibilidade

O Inspetor de Acessibilidade (Accessibility Inspector), exibido na figura 39, é um componente das ferramentas de desenvolvedor do Firefox que fornece acesso à árvore de acessibilidade (accessibility tree) da página (MOZILLA, 2025a). 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 (MOZILLA, 2025a).

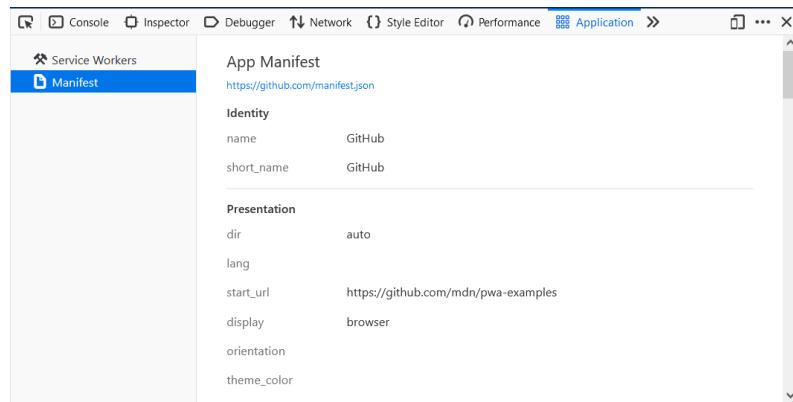
Figura 39 – Ferramenta "Inspetor de Acessibilidade" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

7.2.8 Aplicações

Figura 40 – Ferramenta "Aplicações" do Firefox Devtools



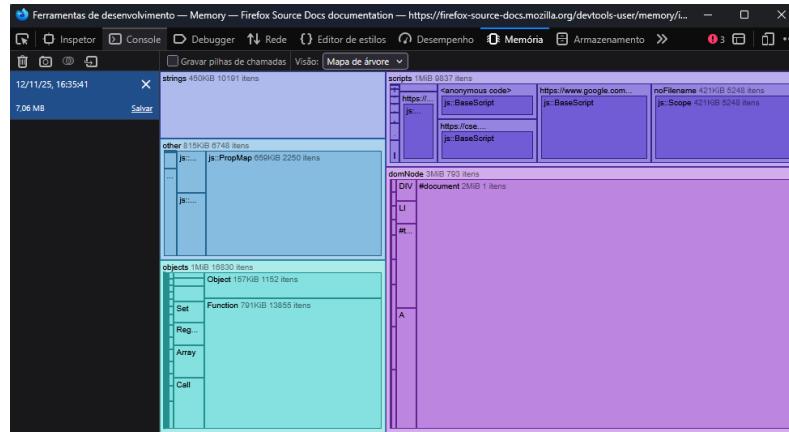
Fonte: Firefox Source Tree Documentation

O painel Aplicação (Application panel), apresentado na figura 40, 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 (PWAs) (MOZILLA, 2025b). Suas capacidades incluem especificamente a inspeção de service workers e de manifestos de aplicações web (web app manifests) (MOZILLA, 2025b).

7.2.9 Memória

A ferramenta Memória (Memory tool), exibida na figura 41, 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) (MOZILLA, 2025i). A visão "Dominators" 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 (MOZILLA, 2025i). Adicionalmente, a ferramenta suporta a comparação entre múltiplos instantâneos e o registro de pilhas de chamadas (call stacks) para rastrear a origem das alocações.

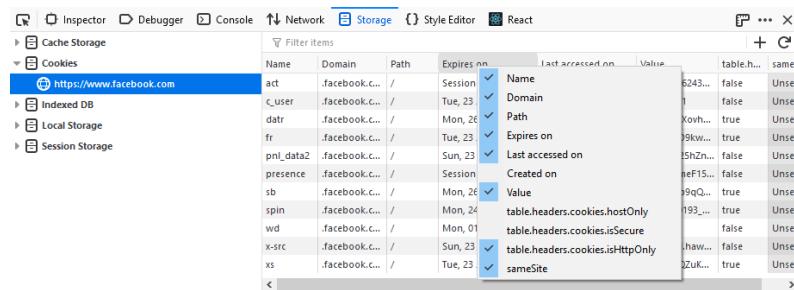
Figura 41 – Ferramenta "Memória" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

7.2.10 Inspetor de armazenamento

Figura 42 – Ferramenta "Inspetor de Armazenamento" do Firefox Devtools



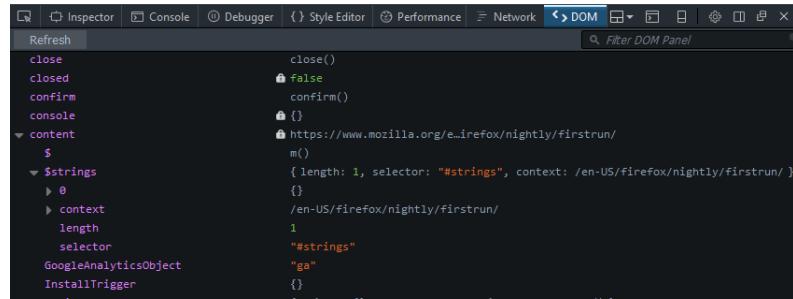
Fonte: Firefox Source Tree Documentation

O Inspetor de Armazenamento (Storage Inspector), ilustrado na figura 42, 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 (MOZILLA, 2025I). 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 (MOZILLA, 2025I).

7.2.11 Visualizador de Propriedades DOM

O Visualizador de Propriedades DOM (DOM Property Viewer), apresentado na figura 43, 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 (MOZILLA, 2025d). 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 (MOZILLA, 2025d). A exibição pode ser atualizada manualmente para refletir alterações no DOM.

Figura 43 – Ferramenta "Visualizador de Propriedades DOM" do Firefox Devtools



```

DOM Panel
Filter DOM Panel

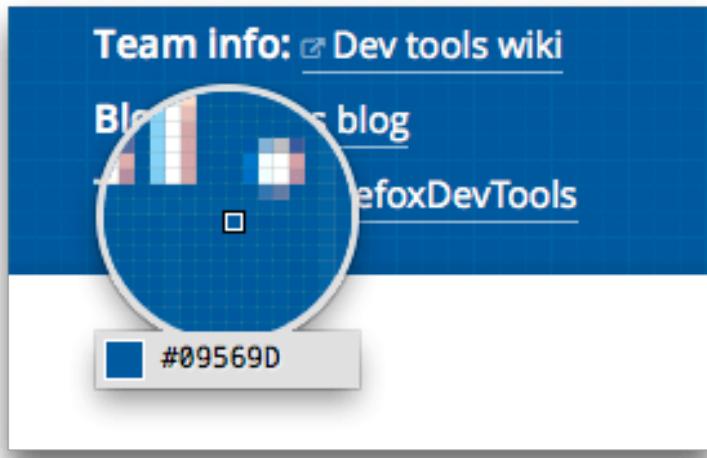
close()
closed
confirm()
console
content
$strings
  0
  context
  length
  selector
GoogleAnalyticsObject
InstallTrigger

close()
false
confirm()
()
https://www.mozilla.org/e.firefox/nightly/firstrun/
m()
{length: 1, selector: "#strings", context: /en-US/firefox/nightly/firstrun/}
()
1
"#strings"
"ga"
()
```

Fonte: Firefox Source Tree Documentation

7.2.12 Conta-gotas

Figura 44 – Ferramenta "Conta-gotas" do Firefox Devtools



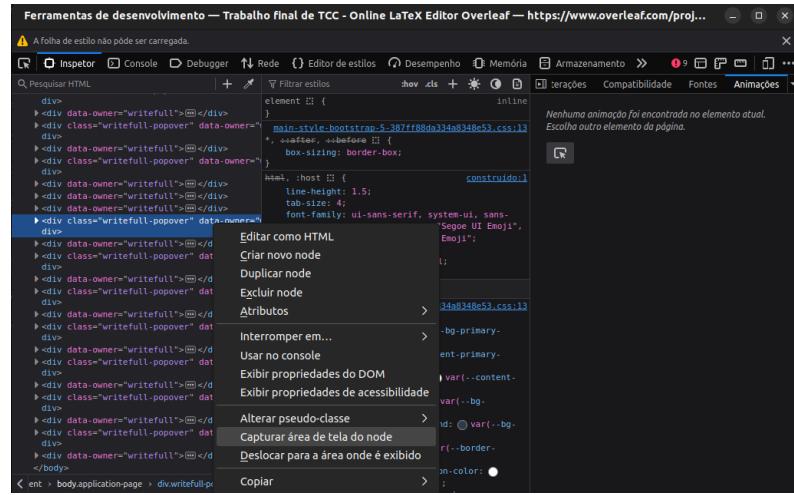
Fonte: Firefox Source Tree Documentation

O Conta-gotas (Eyedropper), exibido na figura 44, é uma ferramenta que permite a seleção de cores da página web atual, operando como uma lupa para seleção com precisão de 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 (MOZILLA, 2025e). 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 (MOZILLA, 2025e).

7.2.13 Captura de Tela

A funcionalidade de Captura de Tela (Screenshot), apresentada na figura 45, 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

Figura 45 – Ferramenta "Captura de Tela" do Firefox Devtools

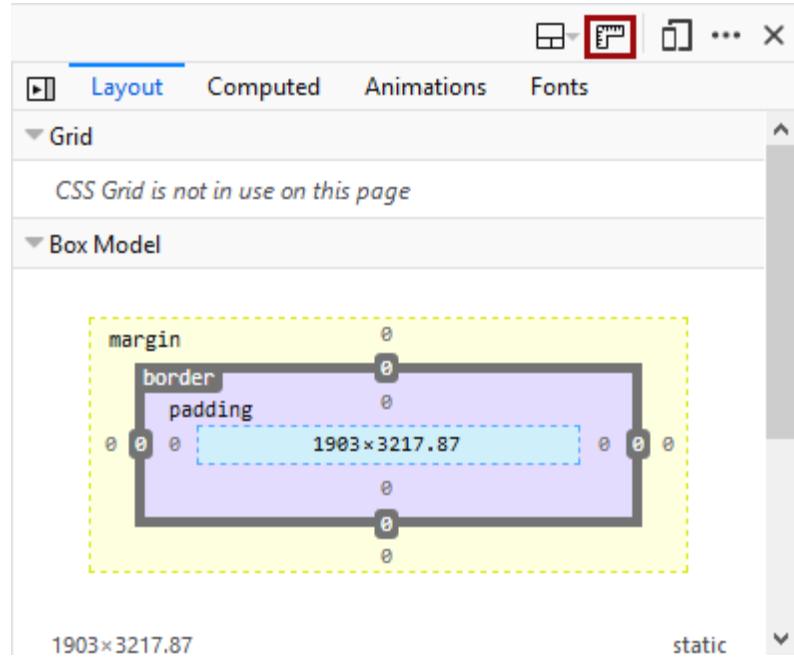


Fonte: Firefox Source Tree Documentation

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 (??).

7.2.14 Régua

Figura 46 – Ferramenta "Régua" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

A ferramenta Régua (Rulers), exibida na figura 46, 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.

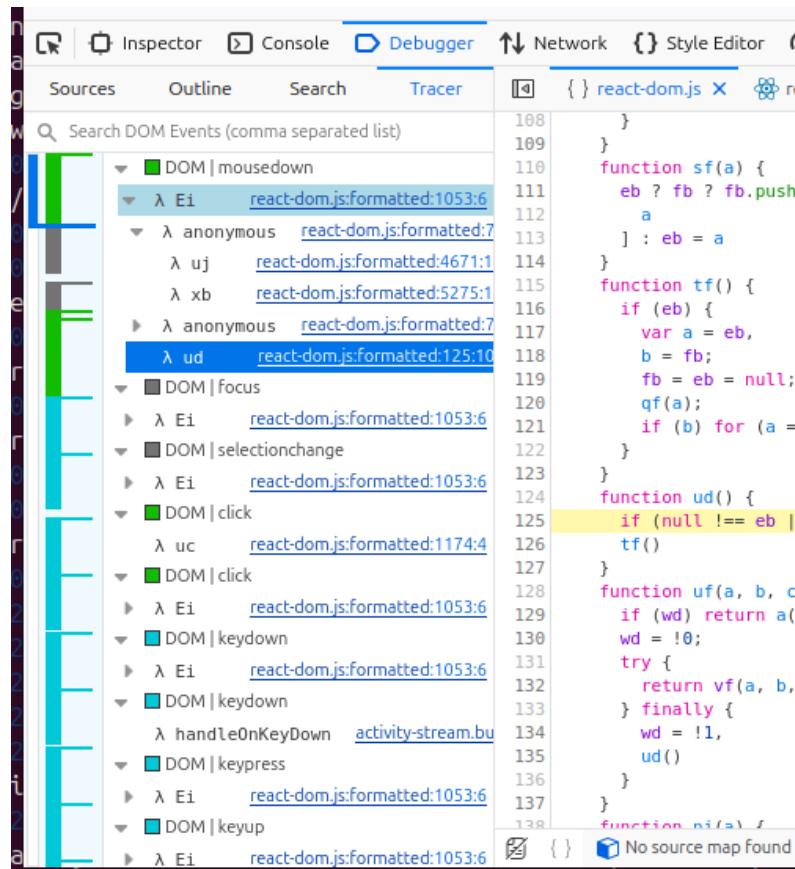
Esta funcionalidade não é persistente, exigindo que o comando seja reativado após cada atualização da página ou navegação (??).

7.2.15 Formatadores personalizados

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 (MOZILLA, 2025c). 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 (MOZILLA, 2025c). Essas funções retornam null ou um array baseado no padrão JsonML para construir a interface HTML customizada, suportando a referênciação de objetos aninhados através de um template de objeto específico.

7.2.16 Traçador JavaScript

Figura 47 – Ferramenta "Traçador JavaScript" do Firefox Devtools



Fonte: Firefox Source Tree Documentation

O Traçador JavaScript (JavaScript Tracer), exibido na figura 47, 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) (MOZILLA, 2025h). Suas opções permitem um início de rastreamento atrasado (delayed start),

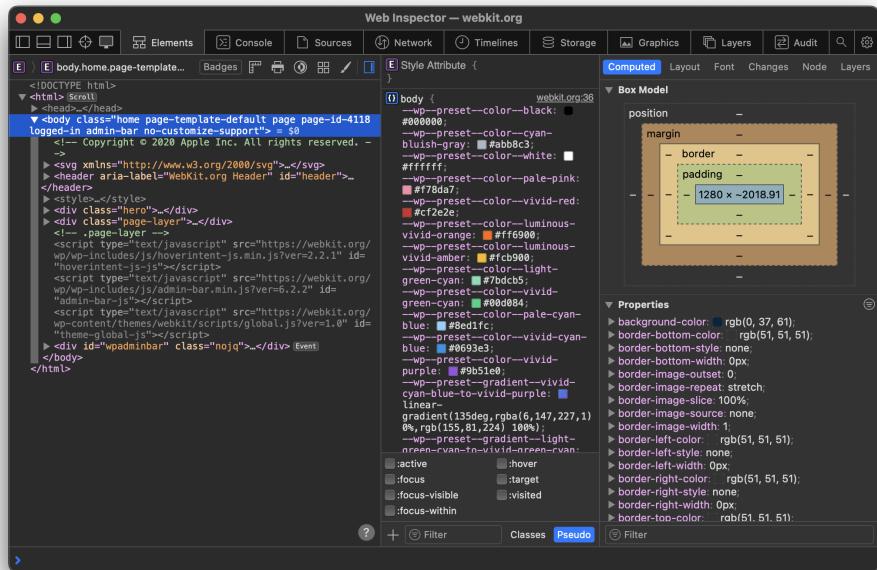
condicionado à interação do usuário ou ao carregamento da página, e pode opcionalmente capturar retornos de função, valores de argumentos, mutações do DOM, bem como operar com limites de profundidade de pilha (depth limit) ou de número de registros (record limit) (MOZILLA, 2025h).

7.3 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.

7.3.1 Elementos

Figura 48 – Ferramenta "Elementos" do Safari Web Inspector



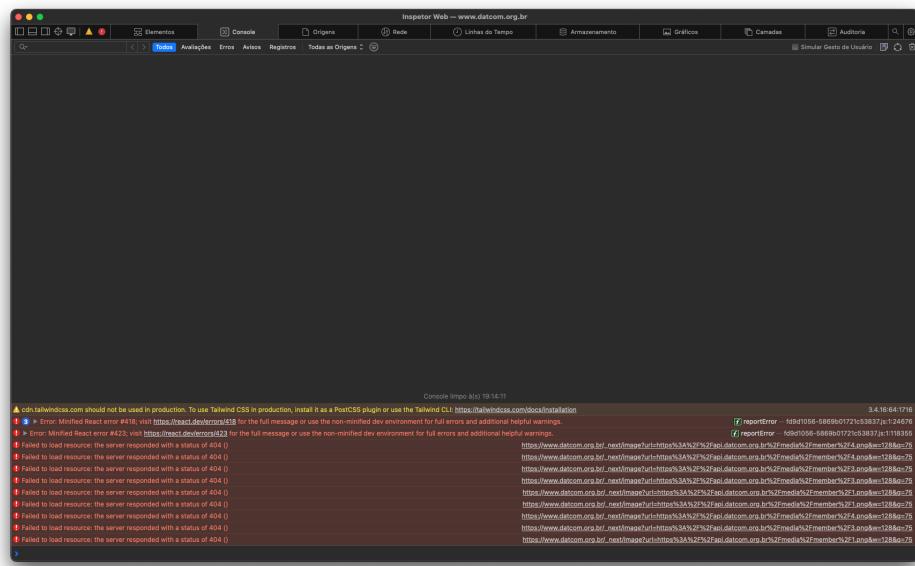
Fonte: Apple Developer Documentation

A aba **Elementos** (*Elements*), apresentada na figura 48, 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*) (ROUSSO, 2021). 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 de acessibilidade (ROUSSO, 2021).

7.3.2 Console

Figura 49 – Ferramenta “Console” do Safari Web Inspector



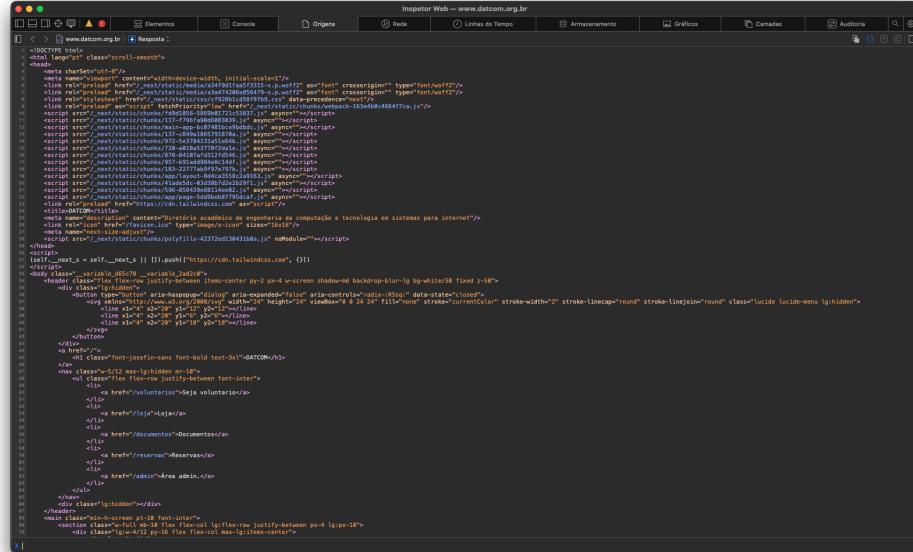
Fonte: Apple Developer Documentation

A aba **Console** (*Console*), exibida na figura 49, 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 (BURG et al., 2020a). 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 (BURG et al., 2020a). 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 (BURG et al., 2020a).

7.3.3 Fontes

A aba **Fontes** (*Sources*), ilustrada na figura 50, centraliza a inspeção de recursos e a depuração de JavaScript. Ela cataloga todos os recursos da página, incluindo requisições dinâmicas, permitindo organização (por tipo ou caminho) e filtragem. A ferramenta fornece visualizações de conteúdo de recursos, como dados de requisição/resposta, e representações formatadas para JSON e XML/HTML, além de utilitários de análise de código, como *code coverage* (cobertura de código) e *type profiling* (perfilagem de tipos) (ROUSSO, 2020b). Seu componente principal é o depurador de JavaScript, que oferece controle total da execução

Figura 50 – Ferramenta "Fontes" do Safari Web Inspector

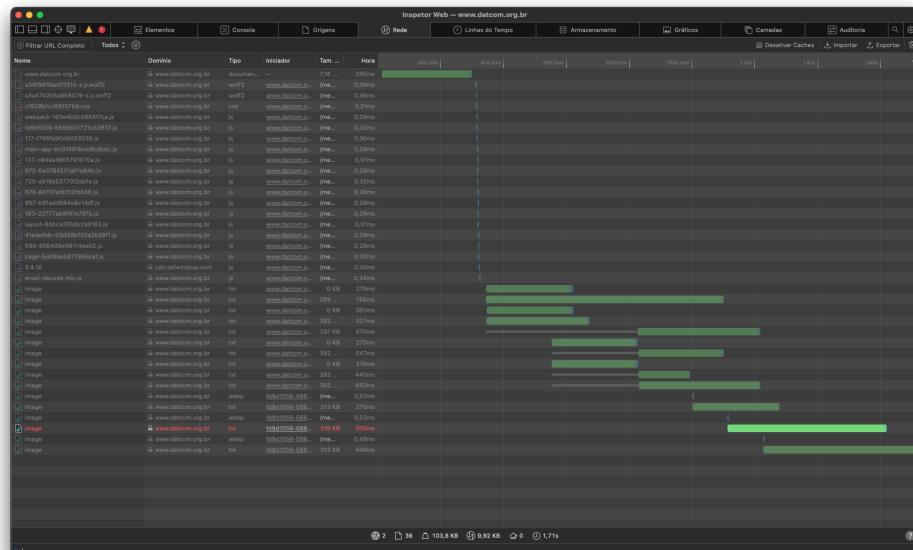


Fonte: Apple Developer Documentation

(pausa, continuação, *step over*, *step in*, *step out*) e gerenciamento de múltiplos tipos de breakpoints (DOM, Evento, URL). Quando a execução é pausada, o inspetor exibe a Pilha de Chamadas (*Call Stack*), que inclui rastreamento de chamadas assíncronas, e a Cadeia de Escopo (*Scope Chain*) para análise de variáveis (ROUSSO, 2020b). A aba também suporta a criação de *Local Overrides* para substituir respostas de rede e *Console Snippets*.

7.3.4 Rede

Figura 51 – Ferramenta "Rede" do Safari Web Inspector



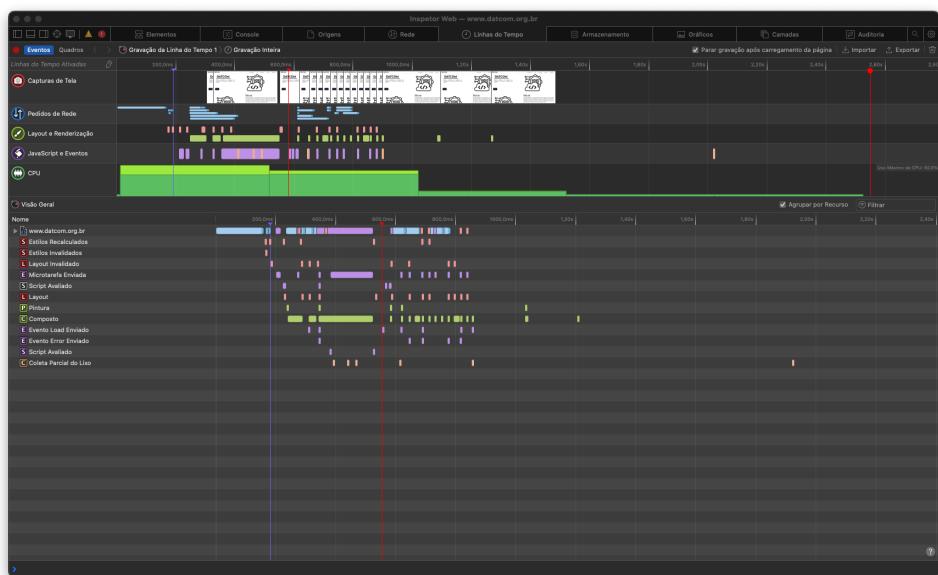
Fonte: Apple Developer Documentation

A aba **Rede** (*Network*), apresentada na figura 51, fornece uma tabulação exaustiva

de todos os recursos requisitados, abrangendo desde ativos estáticos (CSS, JS, HTML) até requisições de API, como XMLHttpRequest (XHR), *fetch* e WebSocket. A interface permite a filtragem de recursos por URL ou tipo, a persistência de logs entre navegações (“Preserve Log”) e a desabilitação do cache de recursos (“Ignore Cache”) (BURG et al., 2020b). A seleção de um recurso individual revela painéis dedicados para inspeção de cabeçalhos (*Headers*) de requisição e resposta, cookies, detalhes de payload (*Sizes*) e um detalhamento gráfico das fases de temporização (*Timing*) da requisição. A ferramenta ainda exibe estatísticas agregadas, como o tempo total de carregamento (*Load Time*) e o total de bytes transferidos, e suporta a exportação e importação de dados no formato HTTP Archive (HAR) (BURG et al., 2020b).

7.3.5 Linha do tempo

Figura 52 – Ferramenta “Linha do Tempo” do Safari Web Inspector



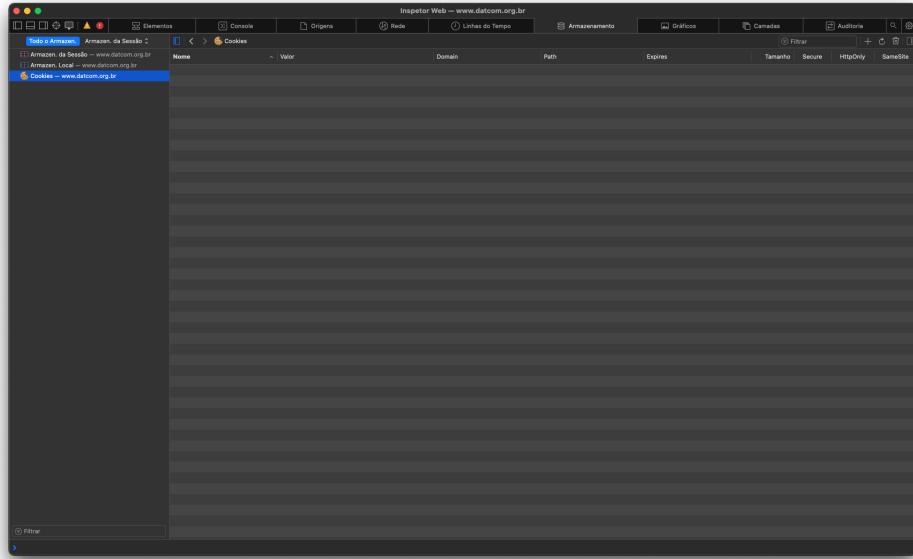
Fonte: Apple Developer Documentation

A aba **Linhas do Tempo** (*Timelines*), apresentada na figura 52, constitui o conjunto de ferramentas de profiling de performance e introspecção do Web Inspector. Ela opera através 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) (BURG et al., 2020c). 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 (BURG et al., 2020c).

7.3.6 Armazenamento

A aba **Armazenamento** (*Storage*), apresentada na figura 53, fornece uma inspeção detalhada e capacidades de gerenciamento para os diversos mecanismos de armazenamento de

Figura 53 – Ferramenta "Armazenamento" do Safari Web Inspector



Fonte: Apple Developer Documentation

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 (??).

7.3.7 Gráficos

A aba **Gráficos** (*Graphics*), exibida na figura 54, 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 (??).

7.3.8 Camadas

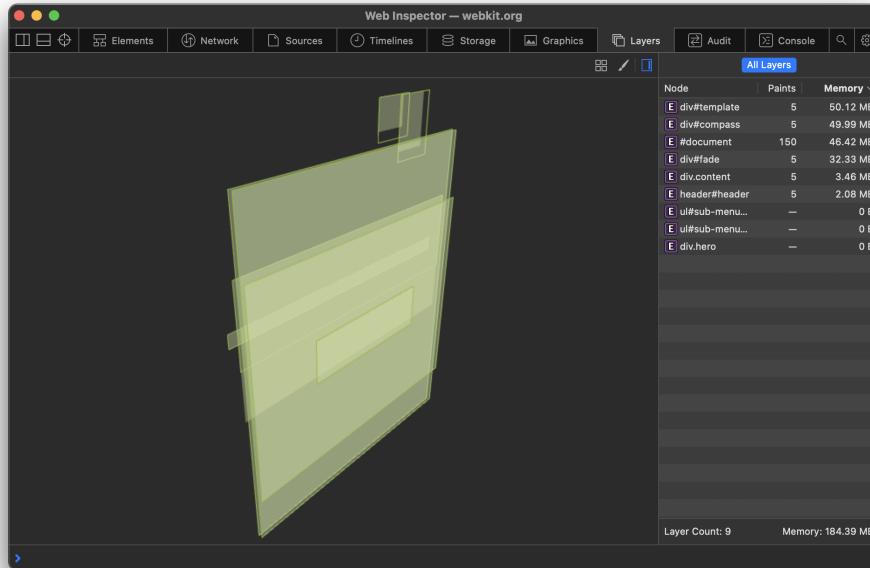
A aba **Camadas** (*Layers*), apresentada na figura 55, fornece uma interface para a análise da performance de renderização, visualizando o processo de *compositing* (composição). Ela exibe como os nós DOM, após o cálculo de layout, são desenhados em superfícies distintas (camadas) que são subsequentemente 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 (ROUSSO; KIRSLING;

Figura 54 – Ferramenta "Gráficos" do Safari Web Inspector



Fonte: Apple Developer Documentation

Figura 55 – Ferramenta "Camadas" do Safari Web Inspector

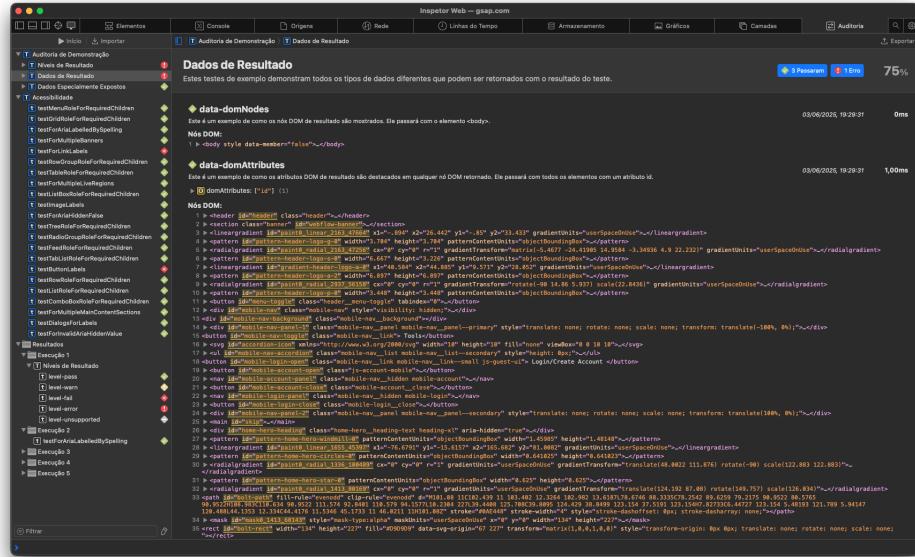


Fonte: Apple Developer Documentation

FRASER, 2020). 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 (ROUSSO; KIRSLING; FRASER, 2020). Adicionalmente, inclui diagnósticos visuais, como "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 (ROUSSO; KIRSLING; FRASER, 2020).

7.3.9 Auditoria

Figura 56 – Ferramenta "Auditoria" do Safari Web Inspector



Fonte: Apple Developer Documentation

A aba **Auditoria** (*Audit*), apresentada na figura 56, 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* (ROUSSO, 2020a). 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 como *Pass* (Aprovado), *Warning* (Aviso), *Failed* (Falha), *Error* (Erro) ou *Unsupported* (Não Suportado) (ROUSSO, 2020a). 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 (ROUSSO, 2020a).

7.3.10 WebDriver

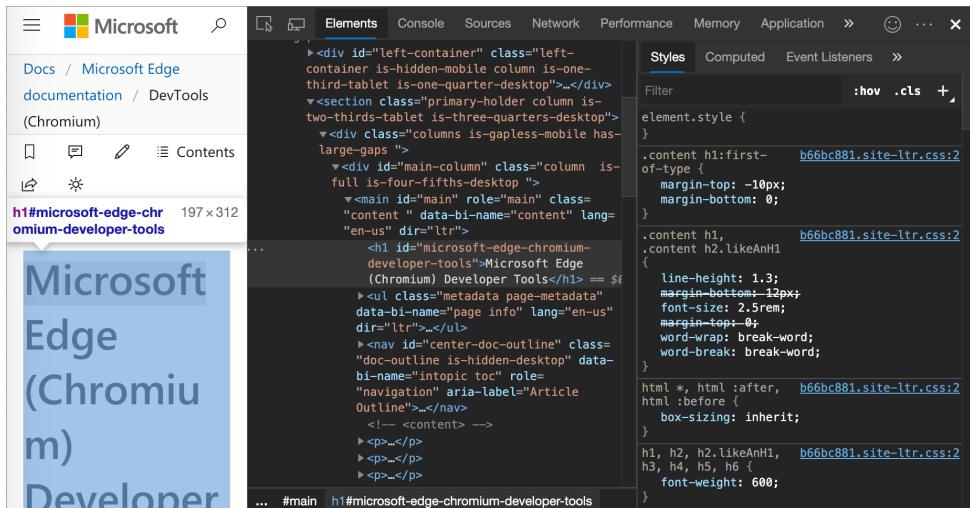
O **WebDriver** para Safari fornece a implementação da Apple do protocolo W3C WebDriver, permitindo a automação de testes de ponta a ponta para o conteúdo web. A arquitetura é mediada pelo executável `safaridriver`, que recebe comandos REST API de bibliotecas de cliente, como o Selenium, e os encaminha para a instância do navegador (APPLE DEVELOPER DOCUMENTATION,). 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 (APPLE DEVELOPER DOCUMENTATION,). 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 ([APPLE DEVELOPER DOCUMENTATION](#),).

7.4 Microsoft Edge

7.4.1 Elementos

Figura 57 – Ferramenta "Elementos" do Microsoft Edge Devtools



Fonte: Microsoft Learn

O painel **Elementos** (Elements) 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).

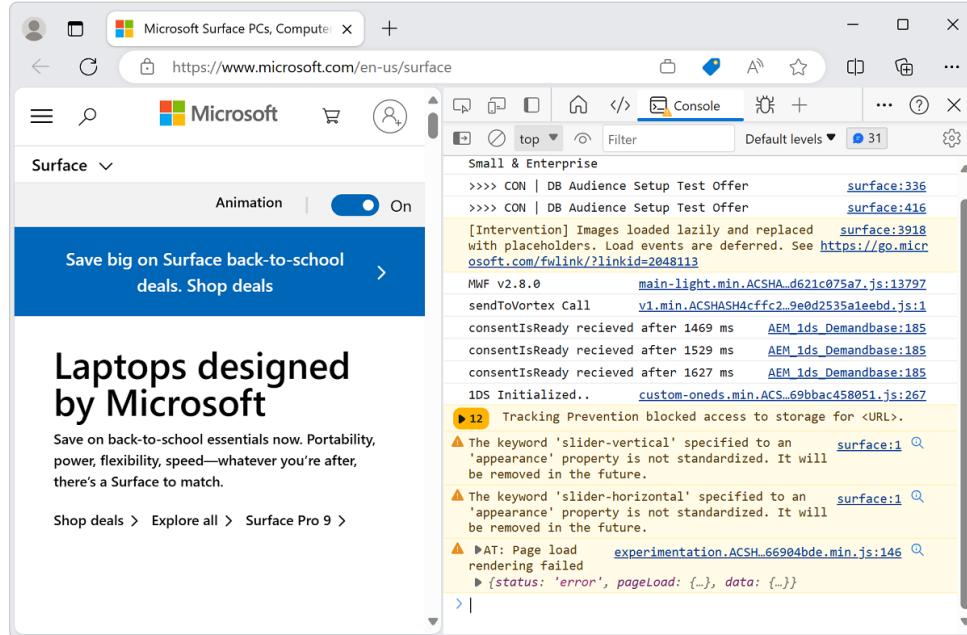
7.4.2 Console

O **Console** (Console) 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.

7.4.3 Fontes

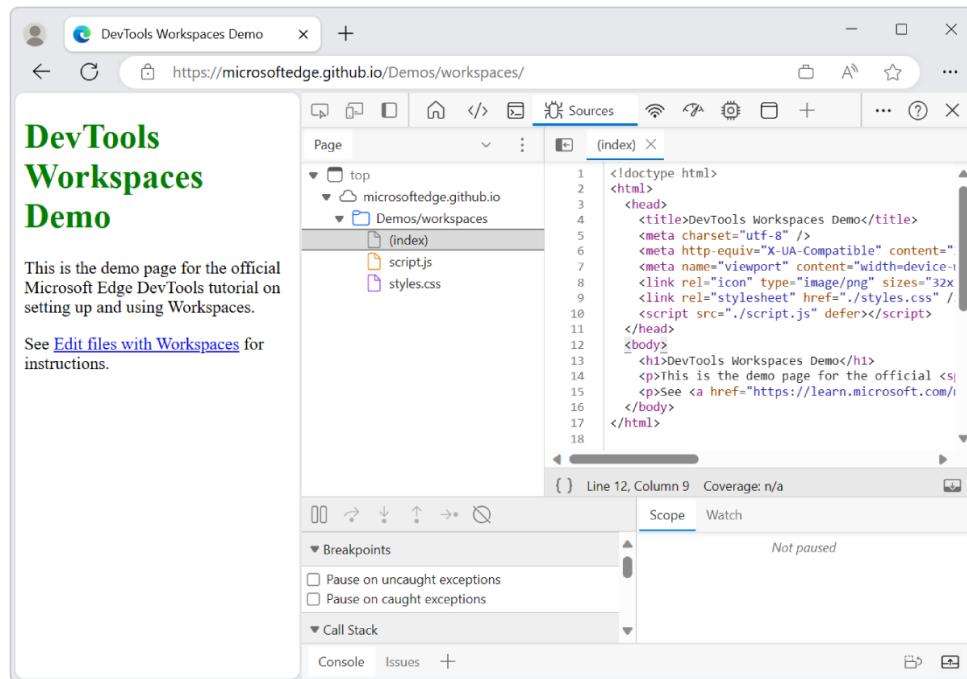
O painel **Fontes** (Sources) funciona como um ambiente de depuração de código-fonte integrado. É projetado para a análise, edição e depuração de JavaScript front-end. A ferramenta permite a navegação pelos recursos da página, a edição de scripts e a utilização de um depurador de JavaScript. Este último suporta funcionalidades essenciais como a definição de pontos de interrupção (*breakpoints*), a execução passo a passo (*step-through*) do código, a inspeção da pilha de chamadas (*call stack*) e a monitorização de variáveis em escopo.

Figura 58 – Ferramenta "Console" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 59 – Ferramenta "Fontes" do Microsoft Edge Devtools

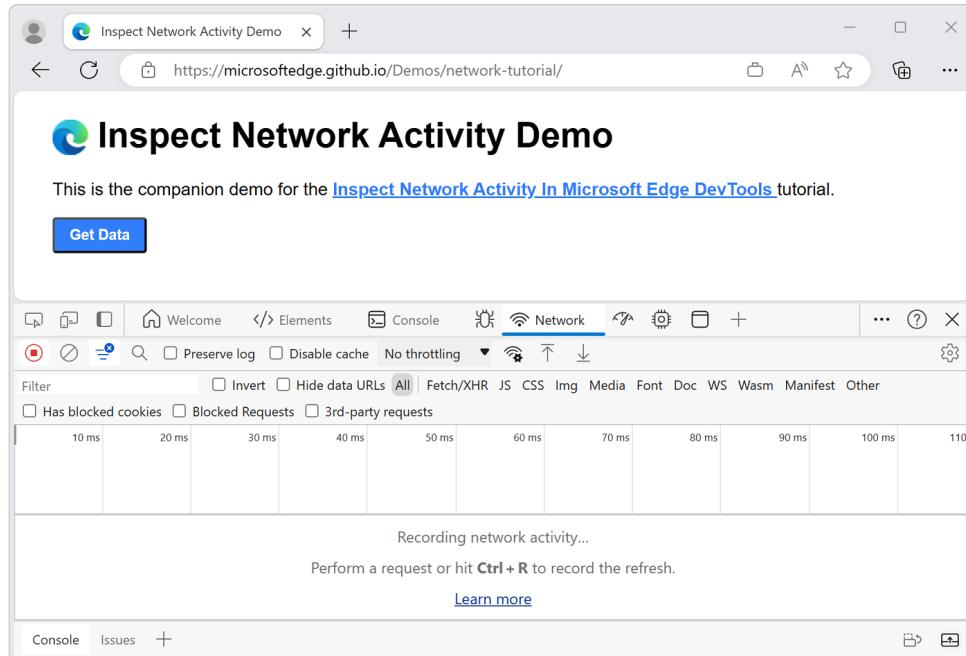


Fonte: Microsoft Learn

7.4.4 Rede

A ferramenta **Rede** (Network) 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

Figura 60 – Ferramenta "Rede" do Microsoft Edge Devtools

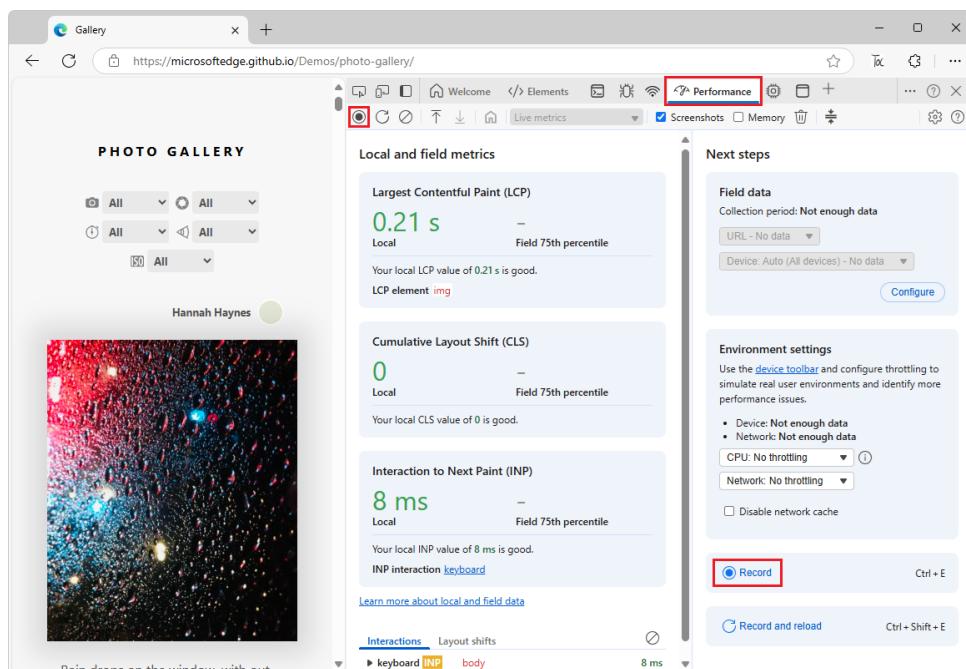


Fonte: Microsoft Learn

ainda a simulação de condições de rede adversas (*throttling*).

7.4.5 Desempenho

Figura 61 – Ferramenta "Desempenho" do Microsoft Edge Devtools



Fonte: Microsoft Learn

O painel **Desempenho** (Performance) 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.

7.4.6 Aplicação

Figura 62 – Ferramenta "Aplicação" do Microsoft Edge Devtools

The screenshot shows the Microsoft Edge DevTools interface with the 'Application' tab selected. The left sidebar lists storage types: Manifest, Service Workers, Storage, and Cookies. Under Cookies, a list of items for the domain https://www.bing.com is shown. The table has columns for Name, Value, Do..., Path, Ex..., Size, Htt..., Sec..., Sa..., Par..., and P... (partially visible). A red box highlights the 'Filter' button at the top of the table. The table contains several rows of cookie data, including DEVTOOLS!, MUID, and various session and user ID cookies.

Name	Value	Do...	Path	Ex...	Size	Htt...	Sec...	Sa...	Par...	P...
DEVTOOLS!	1F2DCDF249CD681E1413DFC...	.ms...	/	20...	41	✓	No...			High
MUID	1F2DCDF249CD681E1413DFC...	.bi...	/	20...	36	✓	No...			High
_SS	SID=00	.ms...	/	20...	9					Me...
_C_ETH	1	.ms...	/	20...	7	✓	✓			Me...
USRLOC		.ms...	/	20...	6	✓	✓	No...		Me...
ai_session	gxIXe0Z1Z765duyANBIZaU 16...	ww...	/	20...	60	✓	No...			Me...
msaoptout	0	ww...	/	20...	10					Me...
MicrosoftApplicati...	6febcc34-edc8-4d53-9338-48...	ww...	/	20...	74	✓	No...			Me...
SRCHUSR	DOB=20220510&P0EX=O	.bi...	/	20...	26	✓	No...			Me...
MUIDB	1F2DCDF249CD681E1413DFC...	ww...	/	20...	37	✓				Me...
ACL	AhDcoey39w4xEkbCQceut39j6...	.bi...	/	20...	91	✓	✓	No...		Me...
OIDR	ghBeYmE6WBjKXXCr0QYHFPX...	.bi...	/	20...	1159	✓	✓	No...		Me...
ipv6	hit=1689214253620&t=4	.bi...	/	Ses...	25	✓	No...			Me...
BFB	AhDXFORrDnG-j8ILLGKQHq...	.bi...	/	20...	198	✓	✓	No...		Me...
USRLOC	HS=1&ELOC=LAT=47.675807...	.bi...	/	20...	94	✓	✓	No...		Me...
...

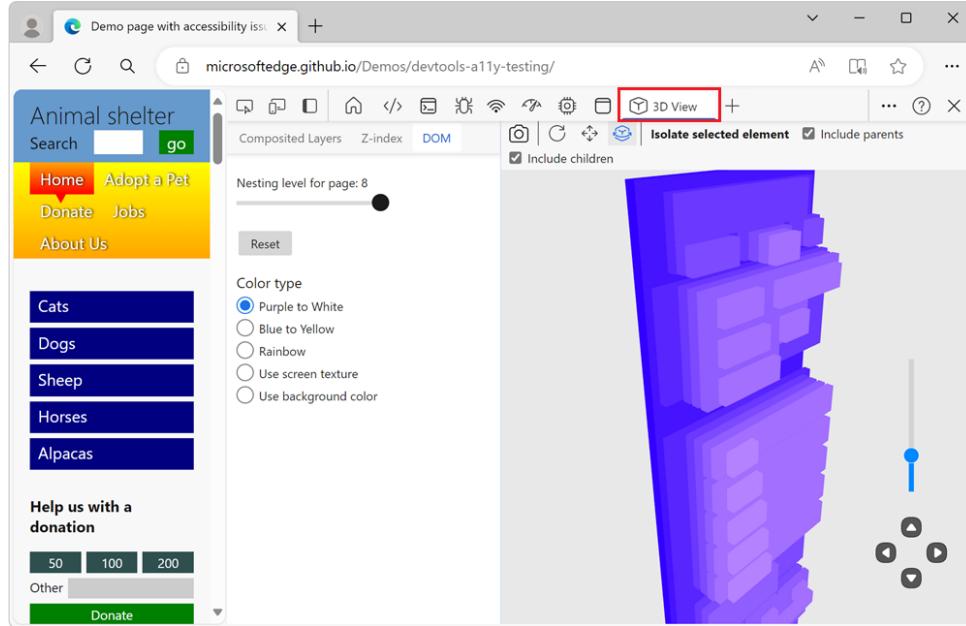
Fonte: Microsoft Learn

A ferramenta **Aplicação** (Application) 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, é um componente vital 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*).

7.4.7 Exibição 3D

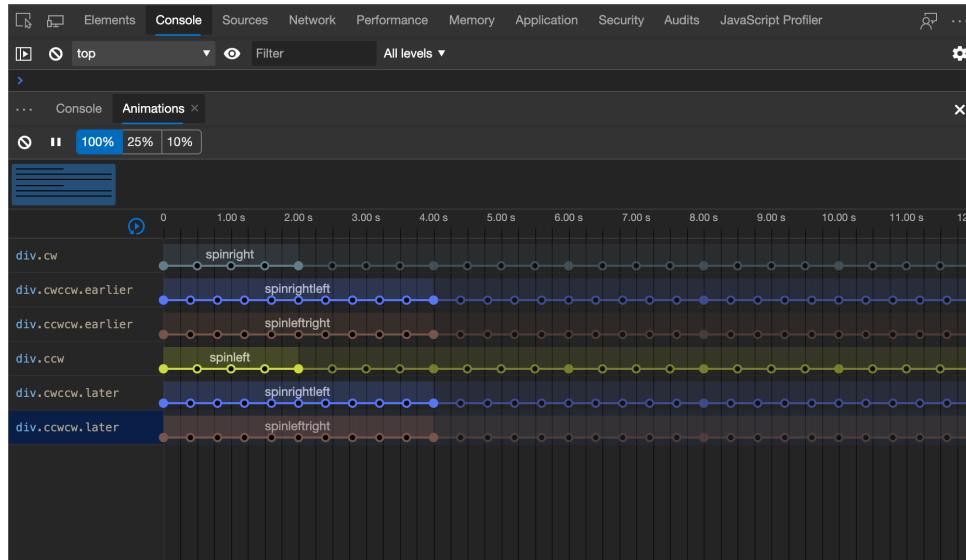
A **Exibição 3D** (3D View) é um recurso de visualização avançado para a depuração de contextos de renderização complexos. Ao modelar a página em um espaço tridimensional, a ferramenta facilita a identificação de problemas de sobreposição de elementos (z-index), a análise da hierarquia do DOM e a inspeção das camadas de composição (*composed 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.

Figura 63 – Ferramenta "Exibição 3D" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 64 – Ferramenta "Animações" do Microsoft Edge Devtools

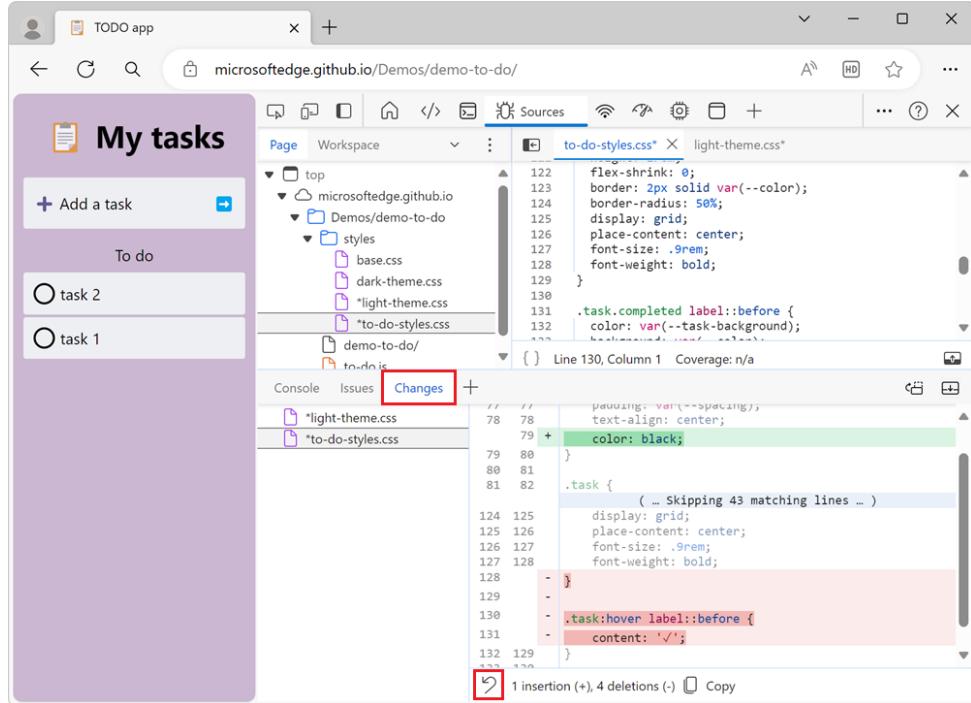


Fonte: Microsoft Learn

7.4.8 Animações

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

Figura 65 – Ferramenta "Alterações" do Microsoft Edge Devtools



Fonte: Microsoft Learn

7.4.9 Alterações

A ferramenta **Alterações** (Changes) monitora modificações efetuadas nos arquivos-fonte (CSS, JavaScript, HTML) diretamente no ambiente de desenvolvimento. 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 locais. Isso facilita a revisão das alterações antes de sua persistência no sistema de arquivos, permitindo também a reversão das modificações.

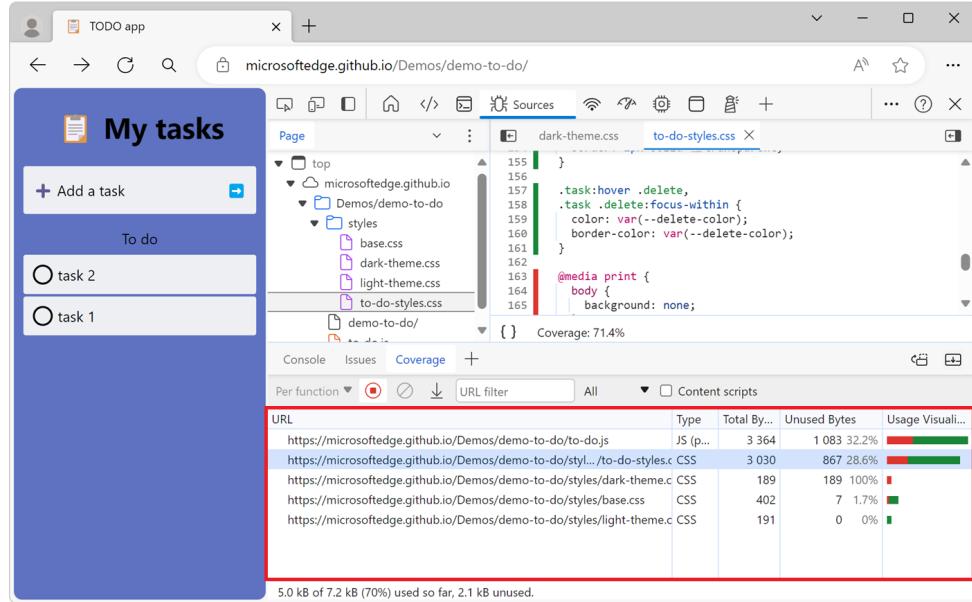
7.4.10 Cobertura

O painel **Cobertura** (Coverage) 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 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.

7.4.11 Analisador de Falhas

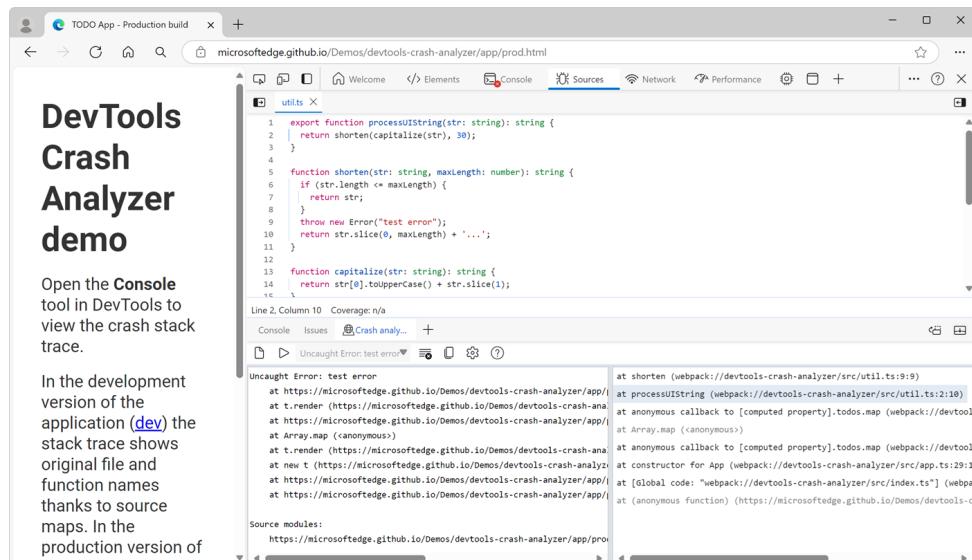
O **Analisador de Falhas** (Crash Analyzer) é um utilitário de diagnóstico pós-morte concebido para analisar falhas de aplicações em produção. Sua principal função é processar *stack traces* (pilhas de chamadas) de JavaScript minificados, que são frequentemente ofuscados. Ao aplicar os *source maps* 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.

Figura 66 – Ferramenta "Cobertura" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 67 – Ferramenta "Analizador de Falhas" do Microsoft Edge Devtools

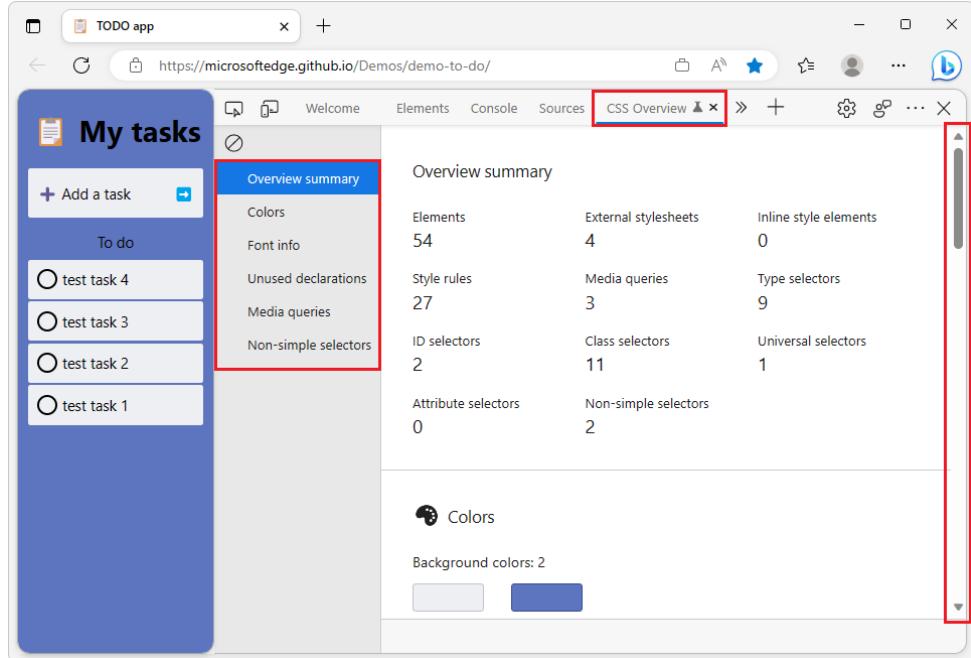


Fonte: Microsoft Learn

7.4.12 Descrição Geral de CSS

A ferramenta **Descrição Geral de CSS** (CSS Overview) realiza uma auditoria estática do CSS de uma página. Ela captura um instantâneo do estado do CSS e 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.

Figura 68 – Ferramenta "Descrição Geral de CSS" do Microsoft Edge Devtools



Fonte: Microsoft Learn

7.4.13 Problemas

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.

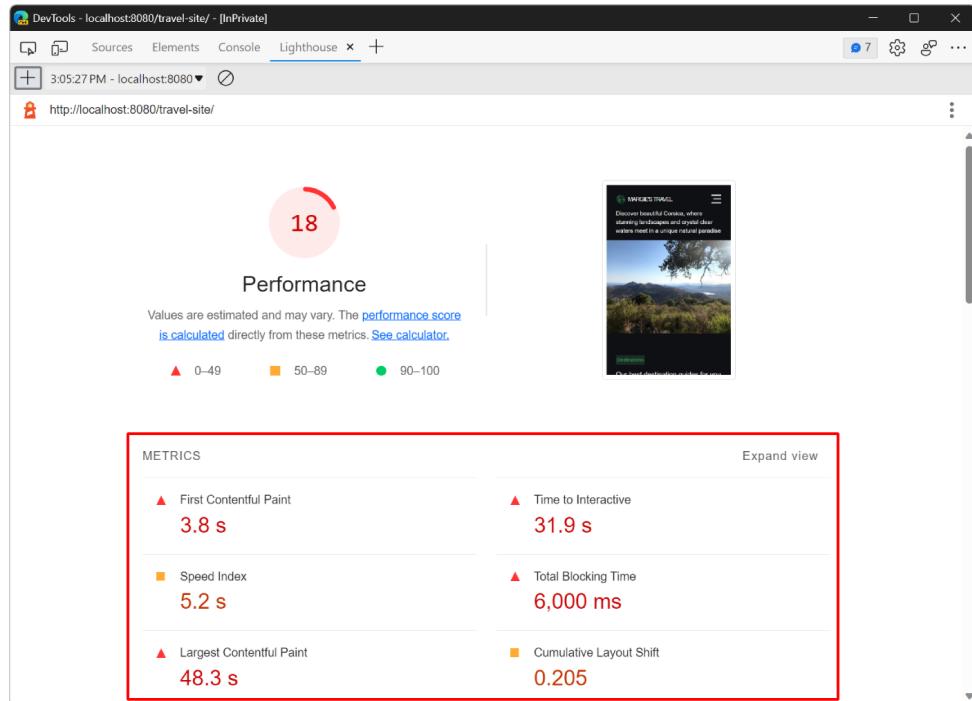
7.4.14 Farol

A ferramenta **Lighthouse** (anteriormente "Audits") 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.

7.4.15 Mídia

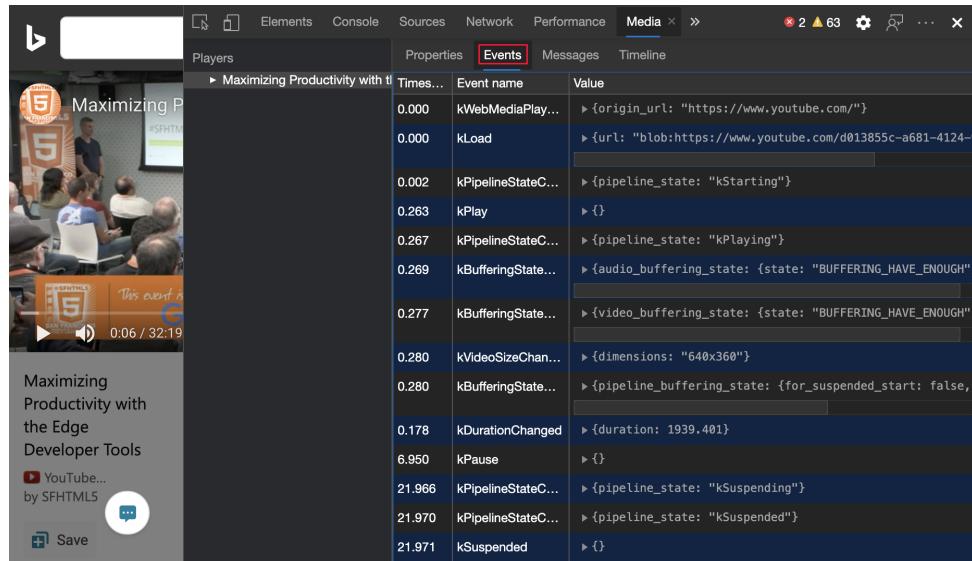
O painel **Mídia** (Media) é um utilitário de depuração específico para a inspeção de reprodutores de mídia (`<video>` e `<audio>`) em uma página. A ferramenta monitora e exibe propriedades do reproduutor, eventos do ciclo de vida da mídia (ex: *play*, *pause*, *seek*) e mensagens de log. Facilita a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

Figura 69 – Ferramenta "Farol" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 70 – Ferramenta "Mídia" do Microsoft Edge Devtools

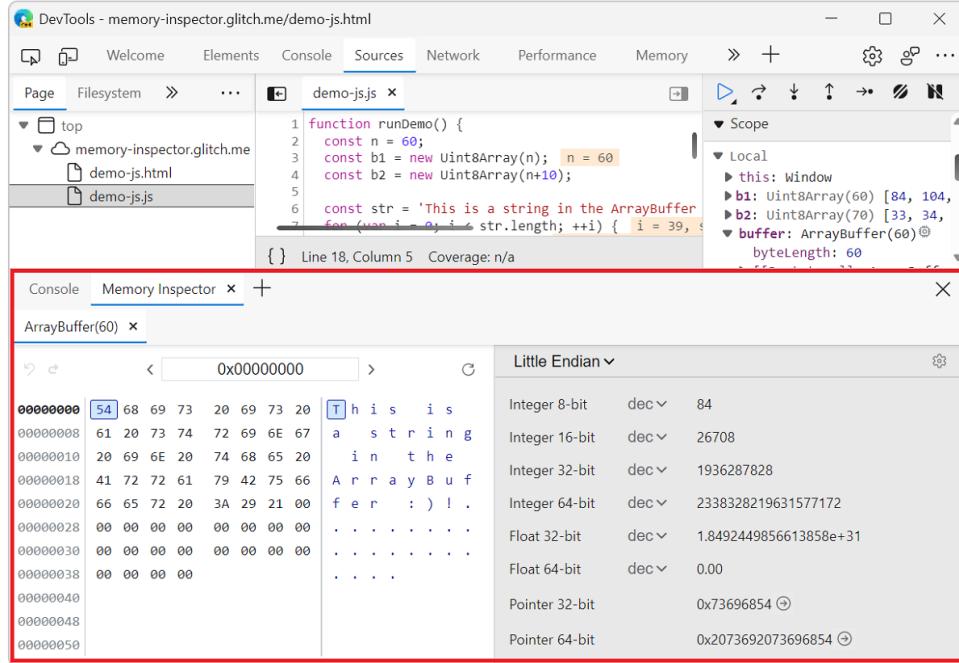


Fonte: Microsoft Learn

7.4.16 Inspetor de Memória

O **Inspetor de Memória** (Memory Inspector) 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*.

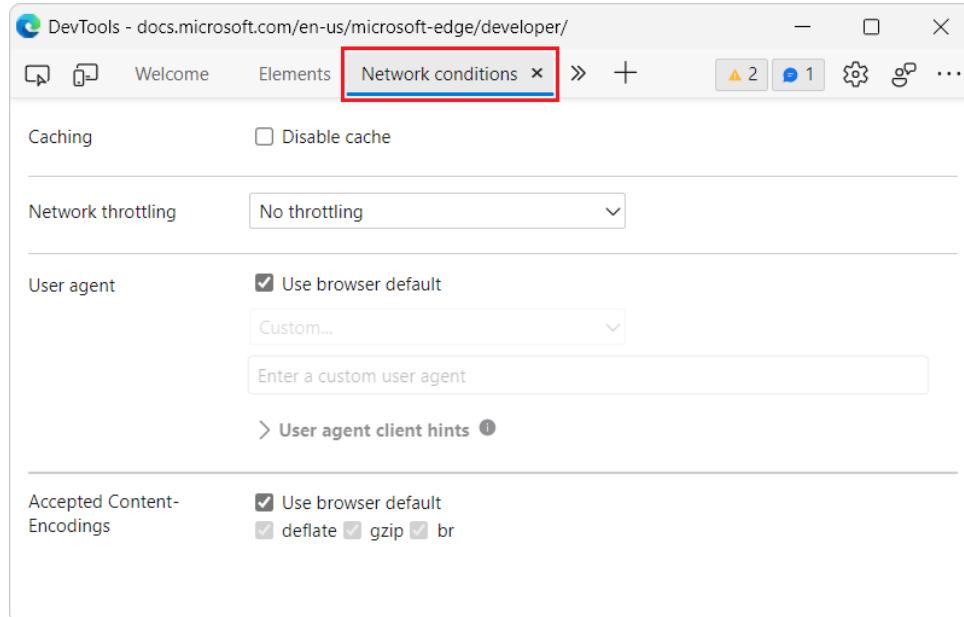
Figura 71 – Ferramenta "Inspetor de Memória" do Microsoft Edge Devtools



Fonte: Microsoft Learn

7.4.17 Condições de Rede

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



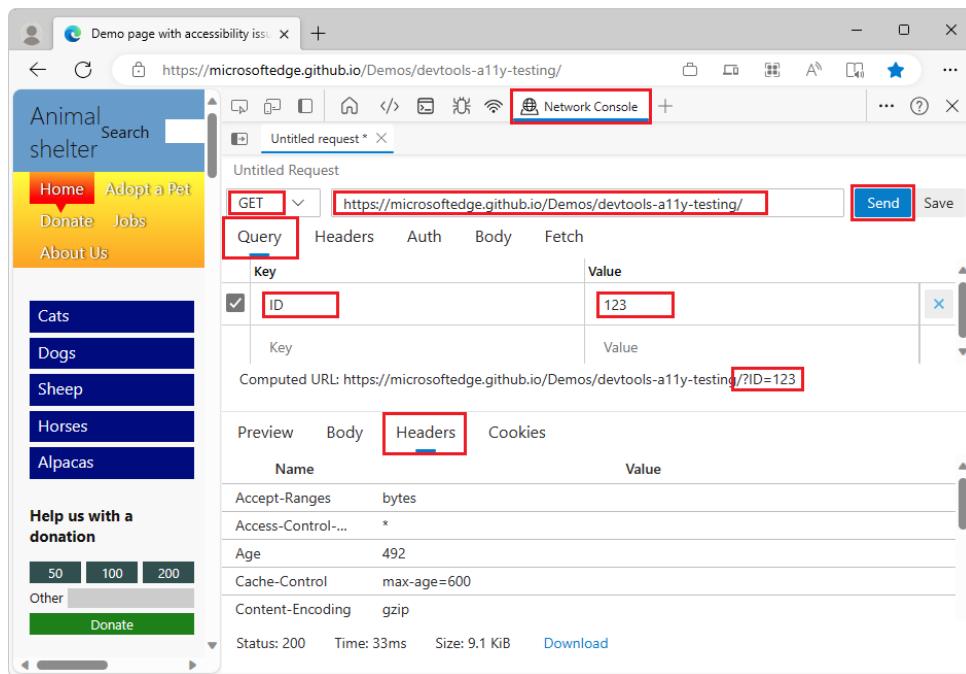
Fonte: Microsoft Learn

A ferramenta **Condições de Rede** (Network Conditions) 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 sobreescriver 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.

7.4.18 Console de Rede

Figura 73 – Ferramenta "Console de Rede" do Microsoft Edge Devtools



Fonte: Microsoft Learn

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

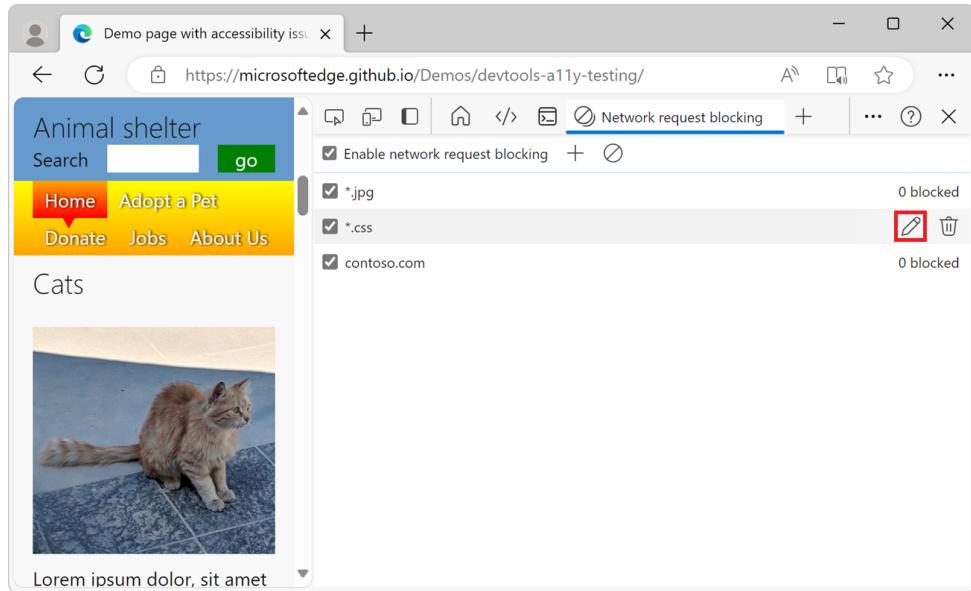
7.4.19 Bloqueio de Solicitações de Rede

Este utilitário 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 resiliência da aplicação, o comportamento de *fallback* e a renderização da página em cenários de indisponibilidade de recursos críticos.

7.4.20 Monitoramento de Desempenho

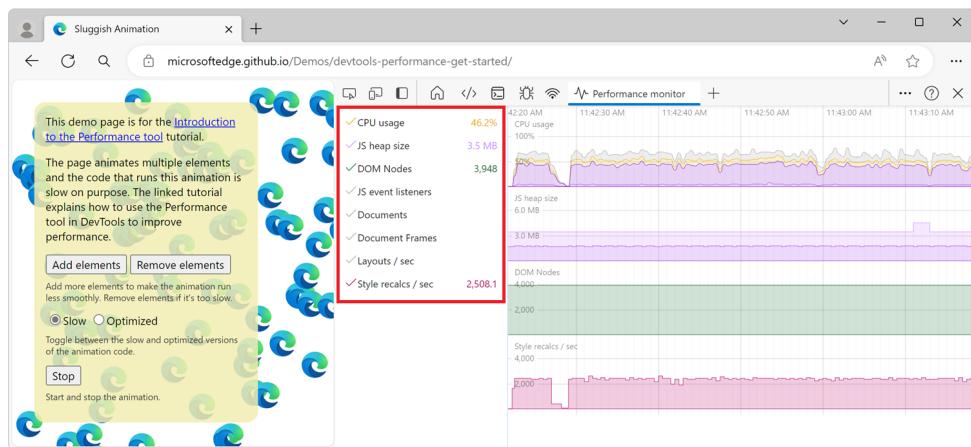
O **Monitoramento de Desempenho** (Performance Monitor) oferece uma visualização em tempo real de métricas de desempenho de tempo de execução. Diferente do perfilamento detalhado do painel "Desempenho", este monitor exibe gráficos contínuos de indicadores-chave, como uso da CPU, tamanho do *heap* de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

Figura 74 – Ferramenta "Bloqueio de Solicitações de Rede" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 75 – Ferramenta "Monitoramento de Desempenho" do Microsoft Edge Devtools



Fonte: Microsoft Learn

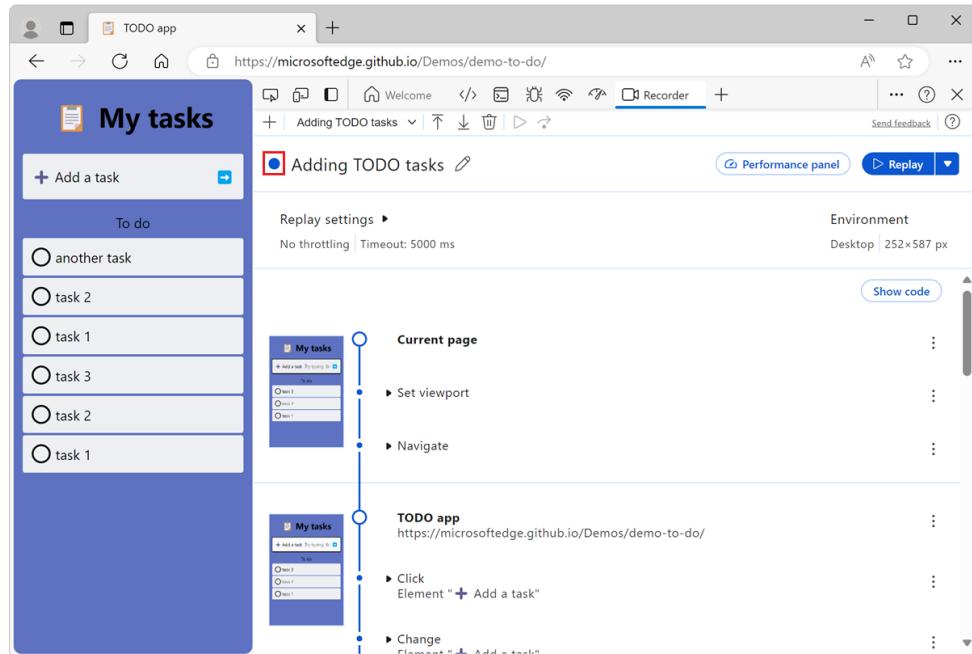
7.4.21 Gravador

A ferramenta **Gravador** (Recorder) permite a captura e 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.

7.4.22 Renderização

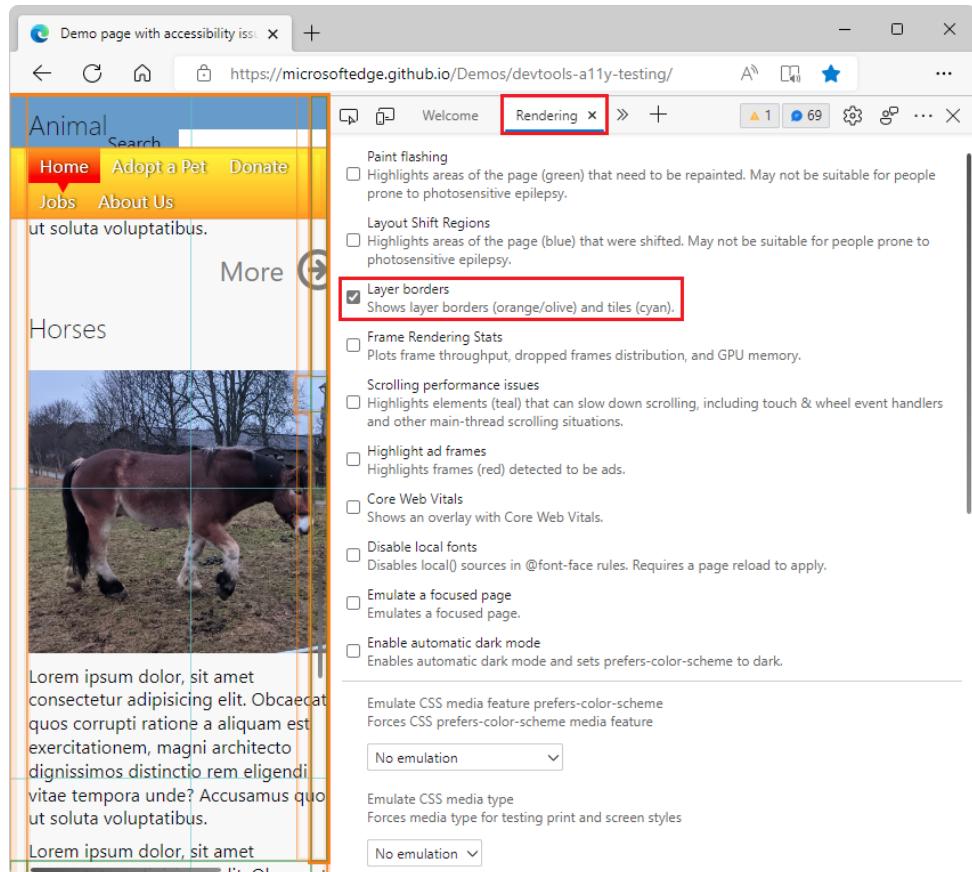
O painel **Renderização** (Rendering) é um conjunto de utilitários de diagnóstico focados na visualização da saída do motor de renderização. Suas funcionalidades incluem a emulação de recursos de mídia CSS (como `prefers-color-scheme` e modo de impressão), a simulação de deficiências visuais (ex: daltonismo,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

Figura 76 – Ferramenta "Gravador" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 77 – Ferramenta "Renderização" do Microsoft Edge Devtools



Fonte: Microsoft Learn

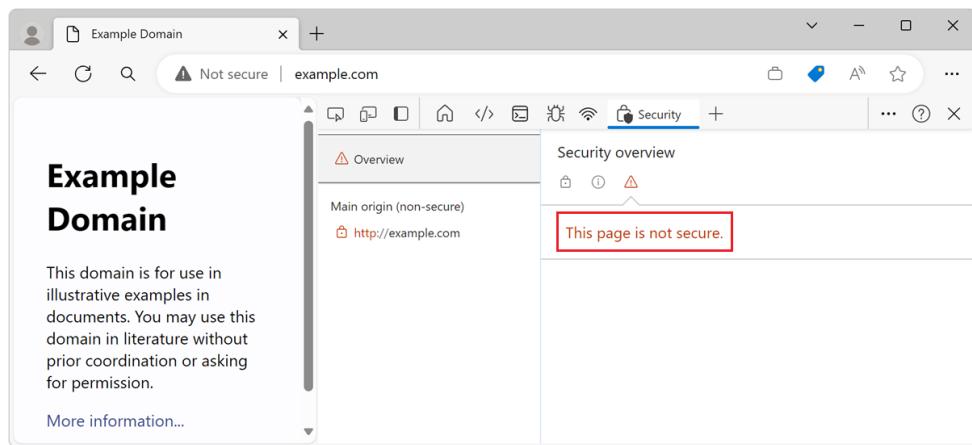
depuração (ex: *paint flashing*, *layout shift regions*).

7.4.23 Pesquisa

A ferramenta **Pesquisa** (Search) 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 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".

7.4.24 Segurança

Figura 78 – Ferramenta "Segurança" do Microsoft Edge Devtools



Fonte: Microsoft Learn

O painel **Segurança** (Security) 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.

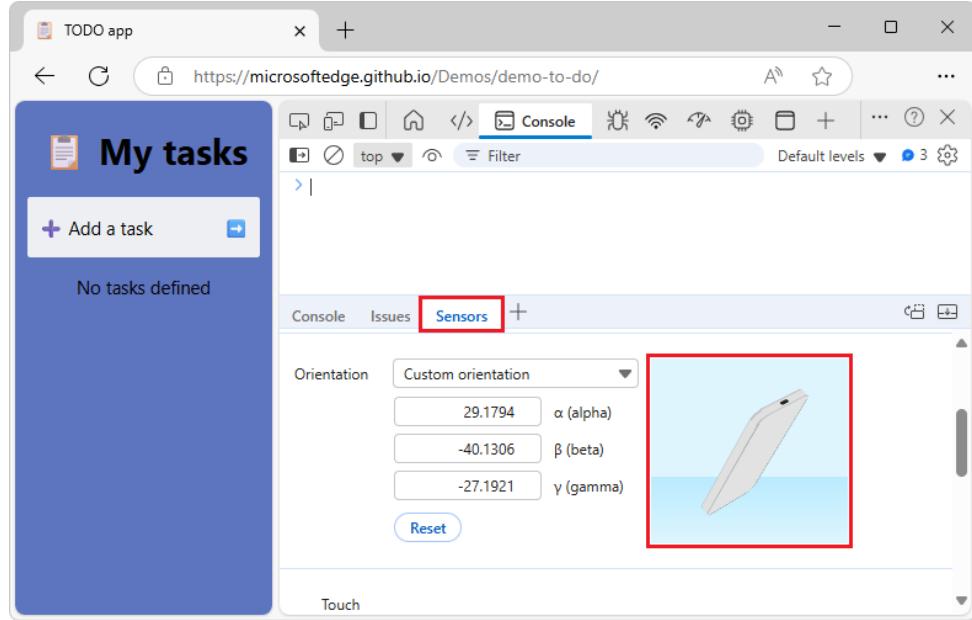
7.4.25 Sensores

A ferramenta **Sensores** (Sensors) 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.

7.4.26 Monitor de Mapas de Origem

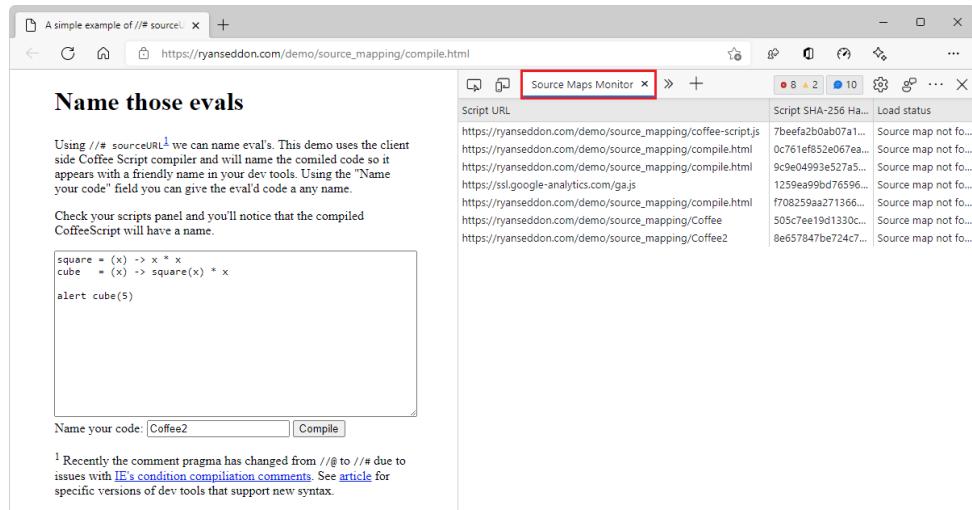
Este painel monitora especificamente a requisição e o status de carregamento de mapas de origem (*source maps*). Ele fornece um log de quais arquivos *source map* foram solicitados, se foram carregados com sucesso ou se falharam. É um utilitário de diagnóstico crucial para garantir que a depuração de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

Figura 79 – Ferramenta "Sensores" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 80 – Ferramenta "Monitor de Mapas de Origem" do Microsoft Edge Devtools



Fonte: Microsoft Learn

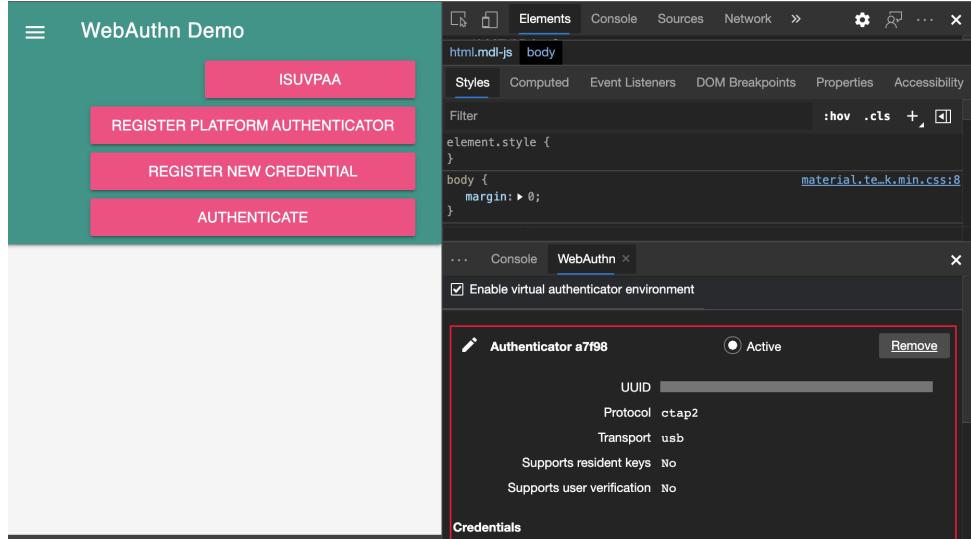
7.4.27 WebAuthn

A ferramenta **WebAuthn** 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.

7.4.28 WebAudio

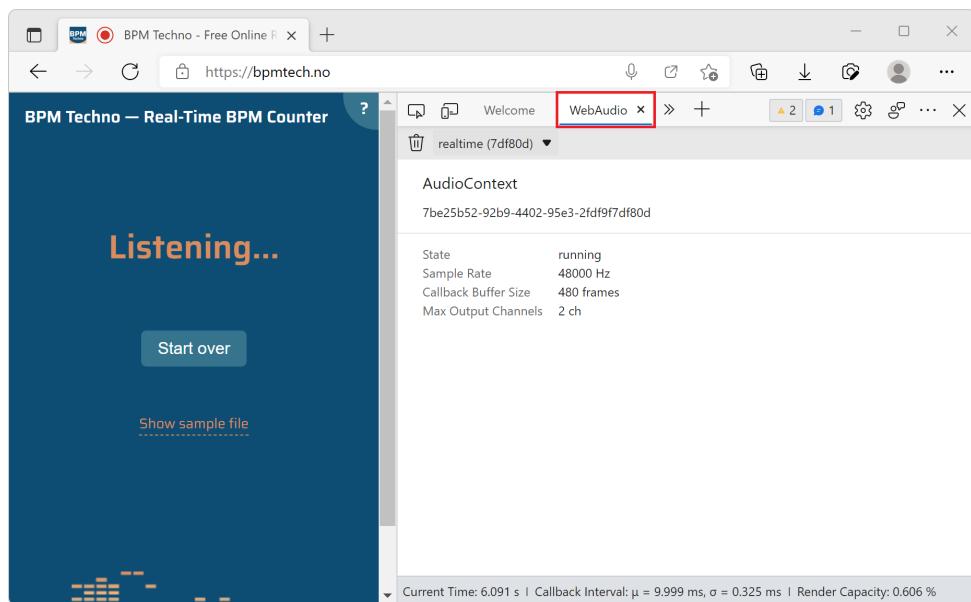
O painel **WebAudio** é um utilitário de diagnóstico para aplicações que utilizam a API WebAudio. Ele renderiza um grafo visual interativo de todos os nós de áudio (*AudioNodes*) instanciados, mostrando suas conexões e estado. Esta visualização é essencial para depurar o

Figura 81 – Ferramenta "WebAuthn" do Microsoft Edge Devtools



Fonte: Microsoft Learn

Figura 82 – Ferramenta "WebAudio" do Microsoft Edge Devtools



Fonte: Microsoft Learn

fluxo de processamento de áudio, inspecionar os parâmetros dos nós e monitorar o *AudioContext* em tempo real.

Funcionalidades / Navegadores	Google Chrome	Mozilla Firefox	Microsoft Edge
Inspeção de elementos	Elementos	Inspetor de Página	Elementos
Console	Console	Console	Console
Depurador JavaScript	Origens	Depurador JavaScript	Origens
Monitoramento de Rede	Rede	Monitor de Rede	Rede
Análise de desempenho	Desempenho	Painel de desempenho	Linha do tempo
Análise de Memória	Memória	Memória	Linha do tempo
Armazenamento de Aplicação	Aplicativos	Inspetor de Armazenamento	Aplicativo
Edição e Manipulação de estilos	Elementos	Editor de estilos	Elementos
Inspeção de segurança	Segurança	–	Segurança
Emulação de dispositivos	Modo Dispositivo	Responsive Design Mode	Modo Dispositivo
Inspeção e edição de animações	Animações	Inspetor de Página	Animações
Análise de Acessibilidade	Lighthouse	Inspetor de Acessibilidade	Lighthouse
Auditoria de qualidade e desempenho	Lighthouse	–	Lighthouse
Cobertura de código	Cobertura	–	Cobertura
Gravar fluxo do usuário	Gravador	–	Gravador

Referências

APPLE DEVELOPER DOCUMENTATION. **WebDriver: automação de testes no Safari**. Disponível em: <<https://developer.apple.com/documentation/safari-developer-tools/webdriver>>. Citado 2 vezes nas páginas 42 e 43.

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 4 e 5.

BASQUES, K. **Inspecionar atividade de rede**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/network?hl=pt-br>>. Citado na página 12.

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

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 17.

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 9.

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 26.

BASQUES, K.; EMELIANOVA, S. **Acessar e alterar CSS**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/css?hl=pt-br>>. Citado na página 9.

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 16.

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 na página 18.

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 16.

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 15.

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 22.

BEHRENS, J. Princípios e procedimentos de análise exploratória de dados. **Psychological Methods**, v. 2, p. 131–160, 1997. Citado na página 7.

- 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 na página 37.
- BURG, B. et al. **Network Tab: guia da aba de rede**. 2020. Disponível em: <<https://webkit.org/web-inspector/network-tab/>>. Citado na página 39.
- BURG, B. et al. **Timelines Tab: guia da aba de linhas do tempo**. 2020. Disponível em: <<https://webkit.org/web-inspector/timelines-tab/>>. Citado na página 39.
- CHROME FOR DEVELOPERS. **Visão geral | Chrome DevTools**. 2016. Disponível em: <<https://developer.chrome.com/docs/devtools/overview?hl=pt-br>>. Citado 2 vezes nas páginas 1 e 4.
- CHROME FOR DEVELOPERS. **Lighthouse performance scoring**. 2019. Disponível em: <<https://developer.chrome.com/docs/lighthouse/performance/performance-scoring?hl=en>>. Citado na página 11.
- DUTTON, S.; EMELIANOVA, S. **Problemas: encontrar e corrigir problemas**. 2020. Disponível em: <<https://developer.chrome.com/docs/devtools/issues?hl=pt-br>>. Citado 2 vezes nas páginas 10 e 11.
- 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 14.
- 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 9.
- 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 20.
- 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 na página 18.
- 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 16.
- 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 14.
- FIREFOX SOURCE TREE DOCUMENTATION. **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 5.
- GLOTZBACH, R. J. Tecnologias atuais e prevalentes no currículo da web. **Education and New Developments 2024 – Volume 1**, 2024. Citado na página 3.

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 11.

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 20.

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 na página 24.

MARTHE, D. S. **Painel de fonte rápida**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/quick-source?hl=pt-br>>. Citado na página 25.

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 13.

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 24.

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 12.

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 23 e 24.

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 11.

MARTHE, D. S. **Visão geral do painel Elementos**. 2024. Disponível em: <<https://developer.chrome.com/docs/devtools/elements?hl=pt-br>>. Citado na página 9.

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 26.

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 13.

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 27.

MOZILLA. **Accessibility Inspector: inspetor de acessibilidade**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/accessibility_inspector/>. Citado na página 30.

MOZILLA. **Application: painel de aplicação**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/application/>>. Citado na página 31.

MOZILLA. **Custom Formatters: formatadores personalizados**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/custom_formatters/index.html>. Citado na página 35.

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 32.

MOZILLA. **Eyedropper: conta-gotas**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/eyedropper/index.html>>. Citado na página 33.

MOZILLA. **Firefox Profiler: documentação da ferramenta de perfil**. 2025. Disponível em: <<https://profiler.firefox.com/docs/#/>>. Citado na página 30.

MOZILLA. **JavaScript Debugger: depurador de JavaScript**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/debugger/>>. Citado na página 29.

MOZILLA. **JavaScript Tracer: rastreador de JavaScript**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/javascript_tracer/index.html>. Citado 2 vezes nas páginas 35 e 36.

MOZILLA. **Memory: ferramenta de memória**. 2025. Disponível em: <<https://firefox-source-docs.mozilla.org/devtools-user/memory/index.html>>. Citado na página 31.

MOZILLA. **Network Monitor: monitor de rede**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/network_monitor/>. Citado na página 30.

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 28.

MOZILLA. **Storage Inspector: inspetor de armazenamento**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/storage_inspector/index.html>. Citado na página 32.

MOZILLA. **Style Editor: editor de estilos**. 2025. Disponível em: <https://firefox-source-docs.mozilla.org/devtools-user/style_editor/index.html>. Citado na página 30.

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 28.

MSEGETEAM. **Overview of DevTools - Microsoft Edge Development**. 2023. Disponível em: <<https://learn.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/overview>>. Citado na página 5.

ODELL, D. **Pro JavaScript Development Coding, Capabilities, and Tooling**. [S.I.]: Berkeley, Ca Apress, 2014. 423-437 p. Citado na página 3.

POLLARD, B. **Lighthouse: otimize seu site**. 2025. Disponível em: <<https://developer.chrome.com/docs/devtools/lighthouse?hl=pt-br>>. Citado na página 11.

ROUSSO, D. **Audit Tab: guia da aba de auditoria**. 2020. Disponível em: <<https://webkit.org/web-inspector/audit-tab/>>. Citado na página 42.

- ROUSSO, D. **Sources Tab: guia da aba de fontes e recursos**. 2020. Disponível em: <<https://webkit.org/web-inspector/sources-tab/>>. Citado 2 vezes nas páginas 37 e 38.
- ROUSSO, D. **Elements Tab: guia da aba de elementos**. 2021. Disponível em: <<https://webkit.org/web-inspector/elements-tab/>>. Citado 2 vezes nas páginas 36 e 37.
- ROUSSO, D.; KIRSLING, R.; FRASER, S. **Layers Tab: guia da aba de camadas**. 2020. Disponível em: <<https://webkit.org/web-inspector/layers-tab/>>. Citado na página 41.
- 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 19.
- 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 10.
- 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 10.
- 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 na página 22.
- 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 21.