ALEXANDRE CARAMASCHI

FRONTENDS COM VIBECODING

Do briefing ao deploy, com a IA escrevendo e você dirigindo

Guia ilustrado e curso completo - mobile, animação, mídia, acessibilidade e operação

EDIÇÃO 2026

Como usar este guia

Você pode seguir a trilha principal em ordem, usar os dois capítulos especiais como oficinas autônomas ou ir direto ao caderno de consulta. O projeto condutor é um portal de produto hipotético, rotulado como cenário didático, que começa com briefing vago e termina publicado com componentes, estados, mobile, mídia, acessibilidade, testes e reversão.

> Promessa de aprendizagem
> Ao concluir, você terá uma régua para pedir, revisar e publicar interfaces assistidas por IA. O artefato final é um checklist de decisão e qualidade que cabe num pull request.

Tese editorial

A IA diminui o custo marginal de gerar código e aumenta o valor de contexto, revisão e governança. A vantagem não vem de aceitar a primeira tela mais rápido; vem de tornar cada decisão visível, testável e reversível.

Escopo e data

Este e-book adapta o curso Frontends com Vibecoding disponível em alexandrecaramaschi.com, consultado em 15 de agosto de 2026. A página declarava 37 capítulos e 1.267 minutos. Tecnologias e licenças mudam; confirme a documentação oficial antes de adotar dependência ou ativo de mídia.

Sumário

Mapa dos 37 capítulos do curso

A edição consultada em 15 de agosto de 2026 declara 37 capítulos e 1.267 minutos. O mapa abaixo preserva a ordem de decisão usada pela página: compra, linguagem visual, funil, instrumentos, casos, risco, escolha de IA e apêndice prático.

1. O que a empresa compra

Capítulos

#	Capítulo	Duração
1	O que a empresa compra quando compra uma interface	22 min
2	Escrever o prompt que reduz retrabalho	26 min
3	Escolher stacks e frameworks	24 min
4	Comprar biblioteca ou construir	22 min

2. A marca na tela

Capítulos

#	Capítulo	Duração
1	CSS moderno, Tailwind e tokens	32 min
2	Componentes e design systems	34 min
3	Ilustração, ícones e identidade	28 min
4	Animação, imagem e vídeo: as bibliotecas de 2026	42 min

3. A interface como funil

Capítulos

#	Capítulo	Duração
1	Menus e navegação	40 min
2	Layout no celular: a tela pequena manda	38 min
3	Formulários e validação	44 min
4	SEO técnico e GEO	24 min
5	Internacionalização	22 min

4. Instrumentos de decisão

Capítulos

#	Capítulo	Duração
1	Dataviz e dashboards	42 min
2	Tabelas e data grids	58 min
3	Mapas geoespaciais	30 min
4	Diagramas, canvas e grafos	32 min

5. Casos e benchmarks

Capítulos

#	Capítulo	Duração
1	Ler a stack de um site grande	36 min
2	O que faz um site parecer caro	22 min
3	Stripe	40 min
4	Booking	32 min
5	Mercado Livre	32 min
6	LATAM	35 min
7	Apple e Temu	70 min

6. Risco, entrega e condução

Capítulos

#	Capítulo	Duração
1	Acessibilidade, contraste e performance	54 min
2	Segurança e autenticação	26 min
3	Do protótipo ao deploy	38 min
4	Programa de modernização em ondas	24 min
5	Projeto final e antipadrões	44 min

7. Qual IA constrói a interface

Capítulos

#	Capítulo	Duração
1	Claude e OpenAI lado a lado	26 min
2	Gemini e Grok	44 min
3	Custo mensal e governança	30 min

8. Apêndice prático

Capítulos

#	Capítulo	Duração
1	Ambiente Claude Code e MCP	48 min
2	Plugins visuais e imagem	52 min
3	React, Tailwind e CSS na prática	62 min
4	PWA e comportamento em celular	22 min
5	Apresentações e decks web	48 min

1. Dirigir a interface antes de pedir código

Vibecoding encurta a digitação, mas aumenta o peso do briefing, da revisão e do critério de pronto. A interface boa nasce de decisões explícitas sobre público, tarefa, estado e consequência.

O briefing deixa de ser um parágrafo inspiracional e passa a ser o contrato operacional da tela.

O que muda quando a IA escreve a primeira versão

Uma equipe pede uma página de captação na manhã de segunda-feira. Antes do almoço, o agente entrega uma hero, três cartões, formulário e rodapé. O ganho de velocidade é real; o risco também. Sem público, prioridade e critério de aceite, cada bloco fica plausível isoladamente e incoerente como sistema.

A tese deste guia é direta: quem pratica vibecoding precisa dirigir como product manager, diretor de arte e revisor técnico ao mesmo tempo. O agente produz alternativas e executa mudanças; a pessoa define o problema, escolhe o compromisso e responde pelo que foi publicado.

O briefing mínimo que produz uma tela revisável

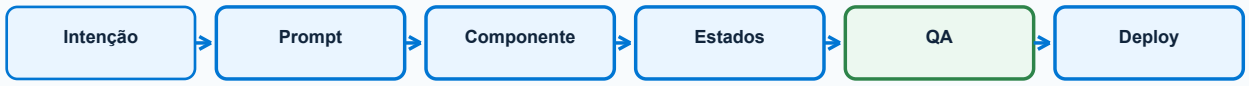
Um briefing útil declara a pessoa que usa a tela, a tarefa que ela precisa concluir, a ordem dos elementos, os estados de carregamento, vazio, erro e sucesso, as restrições de marca, os requisitos de acessibilidade e o limite de performance. O texto também nomeia o que deve permanecer intacto durante a iteração.

A instrução que pede uma página bonita transfere todas as decisões ao modelo. A instrução que pede uma página de cadastro para compradores recorrentes, com o preço e o prazo antes do formulário, erro por campo, confirmação persistente e navegação integral por teclado cria um objeto que pode ser testado.

O loop em quatro tempos

Intenção, geração, revisão e iteração formam o ciclo básico. A revisão compara a tela com o briefing e com o comportamento real no navegador. A iteração muda uma variável por vez: hierarquia, copy, espaçamento, estado ou movimento. Quando tudo muda de uma vez, a equipe perde a causa do ganho e do defeito.

O loop operacional do vibecoding



Checklist do primeiro prompt

- Público e tarefa declarados.
- Hierarquia e estados definidos.
- Restrições de marca, acessibilidade e performance registradas.
- Mudança pequena o bastante para revisar o diff.
- Critério de pronto observável no navegador.

2. Comprar uma interface é assumir três camadas

A tela publicada é apenas a camada visível. Bibliotecas, manutenção, dados pessoais, licença, contraste e atualização entram no contrato no mesmo minuto em que o primeiro componente vai ao ar.

A entrega inclui o que aparece hoje e o que precisará continuar funcionando depois da atualização seguinte.

Tela, manutenção e responsabilidade

A primeira camada é o que o cliente vê e toca. A segunda contém componentes, dependências, pipeline e conhecimento de manutenção. A terceira reúne obrigações: privacidade, acessibilidade, segurança, licença e transparência de persuasão.

A decisão madura registra dono, versão, alternativa de saída e rotina de atualização para cada dependência relevante. Uma biblioteca gratuita pode ter custo total maior do que uma licença comercial quando exige adaptação frequente ou deixa o produto preso a uma API instável.

Design tokens reduzem a conta da mudança

Cor, tipografia, espaço, raio, sombra e movimento viram tokens semânticos. A tela pede `--cor-acao` em vez de repetir um hexadecimal; o componente pede `--espaco-card` em vez de carregar uma sequência arbitrária. A troca de marca acontece na fonte da decisão, e não em dezenas de arquivos.

3. Escolher stack sem transformar tendência em arquitetura

React, Next.js, Tailwind e bibliotecas de componentes resolvem classes de problema. A escolha se torna defensável quando parte do regime de renderização, da equipe, da integração e do custo de saída.

Uma stack saudável é um almoxarifado com peças nomeadas, origem conhecida e política de reposição.

Comece pelo produto, não pelo framework

Uma landing estática, um painel autenticado e um editor colaborativo vivem em regimes diferentes. O primeiro precisa de HTML rápido e publicação simples. O segundo combina estados, permissões e consultas. O terceiro adiciona sincronização, conflito e presença em tempo real. Usar a mesma pilha por hábito desloca complexidade para onde ela custa mais.

O curso trabalha com React e Next.js porque o ecossistema cobre composição, renderização e integração de produto. Tailwind acelera a escrita de estilos quando existe disciplina de tokens; sem ela, as classes viram valores repetidos com outra sintaxe. CSS nativo continua responsável por grid, container queries, cascade layers, variáveis e fallbacks.

Biblioteca ou componente próprio

Use biblioteca quando o problema tem muitas arestas de acessibilidade e comportamento, como diálogo, seletor de data ou grade virtualizada. Construa quando a peça carrega identidade específica, o comportamento é pequeno e a equipe consegue testar todos os estados. A solução híbrida costuma vencer: primitivos acessíveis por baixo e design próprio por cima.

Escolha por regime de produto

Produto	Fundação provável	Decisão crítica
Landing e conteúdo	HTML, CSS e renderização estática	velocidade e manutenção simples
Painel autenticado	React, rotas e consulta de dados	estado, permissão e observabilidade
Editor colaborativo	cliente rico e sincronização	conflito, presença e resiliência
Experiência 3D	canvas/WebGL com fallback	GPU, bateria e acessibilidade

4. Construir linguagem visual como sistema

Identidade não mora na hero. Ela aparece na repetição consistente de tipografia, cor, densidade, borda, estado e movimento em toda decisão do produto.

A paleta ganha função: ação, informação, sucesso, atenção e erro, com contraste testado nos dois temas.

Da referência ao princípio

Copiar a forma de um site famoso produz semelhança superficial e dívida imediata. Extrair o princípio permite adaptação. Da Stripe, por exemplo, vale estudar hierarquia de documentação, confiança e clareza de estado; da Apple, direção de arte, ritmo e art direction; da Booking, densidade, busca e evidência operacional. O princípio sobrevive à troca de conteúdo.

O mini design system começa com tokens e oito componentes de alta repetição: botão, campo, cartão, callout, navegação, diálogo, tabela e estado vazio. Cada um tem variantes, tamanhos, foco, erro, desabilitado e carregamento. A documentação mostra o que usar e o que evitar.

Tema escuro exige desenho próprio

Inverter o fundo e manter as mesmas cores falha em contraste, hierarquia e saturação. O tema escuro reatribui papéis: superfícies se separam por luminosidade, bordas ficam mais presentes, cores de estado perdem saturação e sombras cedem lugar a contornos. A auditoria precisa percorrer os dois temas.

5. Menus e navegação que explicam o produto

A navegação principal é arquitetura de informação tornada visível. Mega menu, dropdown, drawer e command palette têm papéis diferentes e precisam preservar teclado, foco e localização.

A navegação revela o modelo mental do produto antes de o usuário abrir a segunda tela.

Escolher o padrão pela tarefa

Menu simples serve a poucos destinos estáveis. Mega menu organiza famílias extensas quando o agrupamento é parte da decisão. Drawer economiza espaço no celular, mas esconde alternativas e exige uma rota de retorno visível. Command palette atende usuários recorrentes e não substitui a navegação descobrível.

O teste de teclado percorre abertura, setas quando o padrão exige, Escape, retorno de foco, ordem de tabulação e foco visível. A página mantém skip link, heading único e landmarks coerentes. Um menu que funciona com o mouse e prende quem usa teclado está incompleto.

O mobile muda prioridade, não apenas largura

No celular, a primeira tela carrega menos contexto e mais risco de erro de toque. O menu precisa mostrar a ação primária, preservar a posição e evitar sobreposição com barras fixas. O capítulo especial de layout mobile transforma esses princípios numa oficina completa.

6. Formulários, estado e origem do dado

O formulário é o ponto em que copy, validação, acessibilidade, privacidade e receita se encontram. A tela precisa explicar o erro, preservar o que já foi digitado e registrar a origem sem inventar certeza.

O campo tem um contrato

Rótulo persistente, tipo correto, ajuda contextual, mensagem de erro ligada por `aria-describedby`, foco visível e estado de envio formam o contrato mínimo. Placeholder não substitui rótulo. Validação tardia desperdiça tempo; validação agressiva durante a digitação acusa erro antes de a pessoa terminar.

A mensagem diz o que aconteceu e como sair. 'CPF inválido' é diagnóstico incompleto; 'Digite 11 números, sem pontos ou traços' orienta a correção. No envio, o botão impede duplicidade, o estado anuncia progresso e o sucesso confirma o que será feito com o dado.

A atribuição precisa sobreviver ao componente

UTM, referrer, página de origem, consentimento e horário entram no payload de forma auditável. Esconder esses campos apenas no frontend cria confiança falsa; a API valida, normaliza e grava. O relatório distingue desconhecido de orgânico, porque ausência de parâmetro não prova origem.

7. Dashboards, tabelas, mapas e diagramas

Visualização existe para mudar uma decisão. O gráfico certo depende da pergunta, da proveniência e do volume, e sempre ganha alternativa acessível quando a geometria sozinha não basta.

Pergunta antes do gráfico

Comparação pede barras; tendência pede linha; composição pede empilhamento quando a soma importa; distribuição pede histograma ou box plot; relação pede dispersão; processo pede fluxo. Gauge costuma desperdiçar espaço e esconder contexto. Um KPI precisa de valor, período, comparação e definição.

A tabela vence quando o leitor precisa localizar valor exato, comparar muitas dimensões ou exportar. Virtualização entra quando o DOM se torna o gargalo; paginação ou processamento no servidor entra quando o navegador não deveria receber o conjunto inteiro.

Mapas e diagramas têm condição de uso

Mapa vale quando a posição muda a decisão. Ranking por estado não exige mapa se a pergunta é apenas quem vem primeiro. Diagrama explica dependência, fluxo ou responsabilidade; ele falha quando vira uma ilustração sem legenda, leitura alternativa ou relação explícita.

8. Ser encontrado por buscadores e por sistemas de IA

Renderização, semântica, metadados, dados estruturados e conteúdo autossuficiente aproximam SEO técnico e legibilidade para agentes. A página precisa entregar significado no HTML inicial.

A interface começa no documento

Título, descrição, canonical, headings, links descritivos, idioma e dados estruturados precisam concordar. A aplicação cliente pode enriquecer a experiência, mas não deve esconder a resposta principal atrás de uma sequência que o rastreador talvez não execute.

Cada seção abre com resposta completa e depois desenvolve contexto, evidência e exceção. Esse desenho ajuda leitores, snippets e motores generativos sem fragmentar a página artificialmente.

Internacionalização é arquitetura

Texto fora do componente, pluralização, datas, moeda, direção de leitura e tamanho variável entram desde a primeira versão. Traduzir depois exige desfazer concatenações e layouts rígidos. Hreflang e canonical dependem da estratégia de URL, não de um seletor cosmético.

9. Acessibilidade e performance como critérios de aceite

A tela só está pronta quando funciona com teclado, leitor de tela, alto contraste, zoom e rede lenta. Lighthouse e axe ajudam a encontrar defeitos; não substituem o percurso humano.

A auditoria combina semântica, contraste, foco, reflow, mídia alternativa e comportamento real.

Quatro camadas de auditoria

Comece pela semântica: heading, landmarks, rótulos, nomes e estados. Passe ao teclado: ordem, foco, Escape, ausência de armadilha. Verifique percepção: contraste, zoom, espaçamento e alternativas de mídia. Termine com comportamento: anúncios de status, erro, carregamento e preferência de movimento.

WCAG 2.2 exige reflow sem perda de informação em largura equivalente a 320 CSS pixels, com exceções para conteúdo genuinamente bidimensional. O alvo mínimo de 24 por 24 CSS pixels possui exceções documentadas; para controles principais em toque, o curso adota 44 pixels como meta operacional mais confortável.

Performance tem orçamento

Imagem da hero, fontes, JavaScript, vídeo e animação disputam o primeiro segundo. Defina teto por rota, meça em celular intermediário e rede limitada, e trate regressão como falha de build. Largest Contentful Paint, Interaction to Next Paint e Cumulative Layout Shift respondem a problemas diferentes e precisam de diagnóstico separado.

10. Segurança que começa na tela e termina no servidor

O frontend participa de autenticação, autorização percebida, coleta de dado, prevenção de abuso e exposição de segredo. Ocultar um botão nunca substitui permissão verificada no servidor.

Quatro fronteiras

Segredos não entram no bundle. Entrada do usuário é validada no servidor. Cookies de sessão recebem política adequada. A autorização protege cada ação, mesmo quando a interface não mostra o controle. A mensagem de erro não revela detalhes que ajudam enumeração de conta ou ataque.

A proteção também inclui dependências, Content Security Policy, upload, rate limit e registro de eventos. O vibecoding acelera a criação de rotas e amplia a superfície; por isso a revisão de segurança precisa acontecer antes do deploy, não depois do primeiro incidente.

11. Casos de referência: Stripe, Booking e Mercado Livre

Os três casos ensinam compromissos diferentes: confiança e conformidade, busca e persuasão, escala e clareza operacional. A cópia útil preserva o princípio e descarta o ornamento.

O e-commerce expõe a anatomia inteira do frontend: descoberta, comparação, confiança, decisão, pagamento e pós-compra.

Stripe: reduzir dúvida e custo de conformidade

A documentação de duas colunas, os estados claros e a separação entre chave pública e segredo transformam risco técnico em decisão visível. O visual premium vem de tipografia, espaço, gradiente contido e microinteração, e não de uma camada pesada de efeitos.

Booking: densidade com evidência

Busca primeiro, filtros, cartão de propriedade e mensagens contextuais operam num ambiente de comparação rápida. O limite ético é explícito: urgência precisa nascer do inventário real, e nunca de contador fabricado. A tela registra o dado que sustenta o selo.

Mercado Livre: escala continental

Design tokens, componentes repetidos, busca resiliente e estados logísticos claros mantêm coerência num catálogo enorme. A equipe escolhe o que roda no cliente, o que vem pronto do servidor e o que pode ser carregado depois sem interromper a compra.

12. Casos de referência: LATAM, Apple e Temu

Uma venda com data marcada, uma experiência imersiva e uma máquina de engajamento revelam onde animação, densidade e persuasão ajudam ou cobram caro.

LATAM: busca e tarifa como decisão

Calendário de preços, autocomplete, famílias tarifárias e fidelidade precisam manter clareza em múltiplos idiomas, moedas e restrições. A interface não pode esconder condição decisiva até o fim do fluxo. Inventário e preço sustentam a urgência.

Apple: direção de arte com orçamento

Tipografia fluida, mídia dirigida por viewport, seções fixadas e scroll scrubbing criam ritmo quando existe conteúdo preparado para cada enquadramento. A alternativa reduzida e a versão de rede lenta fazem parte do componente. O efeito sem fallback transforma admiração em abandono.

Temu: engajamento sob revisão ética

Recompensa, prova social, urgência e gamificação precisam indicar regra, duração e fonte. O capítulo original usa o contraste entre Apple e Temu para mostrar que atenção não equivale a confiança, e que cada mecanismo de conversão deixa uma obrigação de produto.

13. Do protótipo ao deploy em ondas reversíveis

A entrega profissional nasce de mudanças pequenas, preview por branch, gates automáticos, observabilidade e ensaio de reversão. O projeto final precisa provar qualidade e capacidade de manutenção.

O pipeline transforma revisão em uma sequência repetível, com portões antes de produção e caminho de volta conhecido.

A trilha mínima

Branch curta, commit legível, pull request com escopo, preview, typecheck, lint, teste, acessibilidade e build formam a trilha mínima. O preview permite que quem aprova veja a mudança real sem misturar com produção. O rollback é testado antes da urgência.

Ondas separam fundação, componentes, rotas críticas, casos de borda e otimização. Cada onda tem dono, métrica, risco, critério de aceite e lista de páginas. O agente pode trabalhar em paralelo quando os arquivos não colidem e um orquestrador integra e testa.

O critério de pronto

A tela está pronta quando atende ao briefing, funciona nos estados, mantém navegação e contraste, cumpre orçamento de mídia e JavaScript, passa nos gates, possui analytics verificável e pode ser revertida. Aparência aprovada é um dos itens, não o último argumento.

14. Escolher modelo e governar o custo do vibecoding

Modelos e agentes têm perfis diferentes de contexto, edição, visão, ferramenta e custo. A rota correta usa o modelo mais barato que cumpre o critério, com revisão mais forte conforme o risco.

A matriz que substitui torcida

Compare qualidade de primeira versão, fidelidade a restrições, leitura de repositório, uso de navegador, edição localizada, geração de imagem, latência e custo. Rode a mesma tarefa pequena em dois modelos, preserve o briefing e avalie o diff. Marketing de benchmark não substitui teste no seu código.

O fluxo econômico começa com classificação e rascunho no modelo menor, sobe para o modelo forte quando arquitetura, copy final ou depuração difícil exigem, e fecha com verificação automática. O orçamento registra consumo por tipo de tarefa e evita usar raciocínio premium para renomear uma classe.

Governança mínima do agente

Contexto oficial, arquivos de instrução, plano antes da edição, permissão por ferramenta, branch isolada, teto de custo e revisão de diff formam o conjunto mínimo. O agente não recebe segredo que a tarefa não exige e não publica sem portão explícito.

15. Capítulo especial: layout mobile como produto próprio

Mobile-first significa decidir conteúdo, interação e mídia a partir da menor área útil, e não encolher o desktop. O componente responde ao espaço que recebe, a página reflowa e a ação principal continua alcançável sem cobrir o conteúdo.

A mesma jornada muda de composição quando largura, entrada, rede e contexto de uso mudam.

O cenário que expõe o layout falso

Às 18h, uma pessoa abre o formulário no ônibus, com uma mão, teclado virtual ocupando parte da tela e rede oscilando. A versão de desktop reduzida mantém duas colunas, barra fixa, calendário e vídeo. Tudo cabe em pixels; nada cabe na tarefa.

O layout mobile começa pela ordem de leitura. Conteúdo essencial vem primeiro no DOM, ações ficam próximas da decisão e informação secundária vira expansão, rota ou resumo. A página precisa funcionar em 320 CSS pixels sem rolagem horizontal, conforme o critério de reflow da WCAG 2.2, salvo elementos cuja natureza realmente exige duas dimensões.

Fluxo, grid e container queries

Use fluxo normal antes de posicionamento. Grid e flex resolvem composição; `minmax()`, `auto-fit`, `clamp()` e unidades relativas reduzem a quantidade de breakpoints. Container queries permitem que o cartão responda à coluna que recebeu, útil quando o mesmo componente aparece em sidebar, grade e modal.

Breakpoint nasce do conteúdo quando a linha quebra, o botão perde área de toque ou a comparação deixa de ser legível. Nomes como compacto, confortável e amplo comunicam melhor do que modelos de aparelho. A regra evita layout desenhado para cinco larguras e quebrado nas outras centenas.

Toque, teclado, zoom e área segura

O critério mínimo 2.5.8 da WCAG 2.2 pede alvo de 24 por 24 CSS pixels ou uma das exceções documentadas. Para ações primárias, 44 pixels continua sendo meta operacional confortável. Espaço entre alvos reduz ativação acidental, e o foco visível precisa sobreviver a cabeçalhos e barras fixas.

Meta viewport, `env(safe-area-inset-*)`, zoom, orientação e teclado virtual entram no teste. `100vh` isolado falha em navegadores com barras dinâmicas; unidades `svh`, `lvh` e `dvh` expressam estados diferentes. O modal precisa

caber, rolar internamente e devolver foco.

Mídia dirigida e performance

A imagem de desktop pode perder o assunto ao ser cortada no celular. Art direction usa `<picture>` ou `getImageProps()` para servir composições distintas. Em Next.js, a propriedade `sizes` orienta o navegador a escolher a variante correta do `srcset`; sem ela em imagens responsivas, o navegador pode presumir 100vw e baixar arquivo maior do que o necessário.

Vídeo de fundo no celular só entra quando curto, mudo, pausável e substituível por poster. A preferência de movimento reduzido desliga autoplay e paralaxe. A rota mede LCP, INP, CLS, bytes de imagem, JavaScript e tempo de tarefa em aparelho intermediário.

Oficina mobile em sete passadas

Primeiro, escreva a ordem sem CSS. Segundo, monte uma coluna fluida. Terceiro, transforme repetições em componentes com container queries. Quarto, revise toque, foco e teclado. Quinto, aplique art direction nas imagens. Sexto, teste conteúdo longo, erro e carregamento. Sétimo, rode em 320, 375, 768 e largura ampla, com zoom e rede limitada.

Quatro decisões que tornam um componente realmente responsivo

Matriz mobile: sintoma, causa provável e movimento de saída

Sintoma	Causa provável	Movimento de saída
Rolagem horizontal	largura fixa ou conteúdo sem quebra	grid fluido, min-width:0 e teste em 320 px
CTA cobre o formulário	barra fixa sem reserva de espaço	padding inferior, área segura e foco não obscurecido
Imagem perde o assunto	mesmo corte para toda largura	art direction com picture ou getImageProps
Tabela ilegível	comparação bidimensional comprimida	wrapper horizontal, colunas prioritárias e alternativa resumida
Modal não fecha	altura fixa e foco sem retorno	dvh, rolagem interna, Escape e restauração de foco

Checklist de pronto do capítulo mobile

- A ordem de leitura funciona sem CSS.
- A página não exige rolagem em duas dimensões a 320 CSS pixels, salvo exceção justificável.
- Alvos de toque cumprem o mínimo normativo e ações principais buscam 44 pixels.
- Foco nunca fica escondido por cabeçalho, cookie banner ou barra fixa.

- Imagem usa largura, altura, sizes e corte apropriados.
- Teclado virtual, orientação, zoom e safe area foram testados.
- A rota passa pelo orçamento de LCP, INP, CLS, mídia e JavaScript.

16. Capítulo especial: animações, imagens e vídeos modernos

Movimento bom explica mudança de estado, continuidade e hierarquia. A biblioteca correta depende da interação, e a mídia só entra quando direitos, variantes, fallback e orçamento já têm dono.

A animação começa pela pergunta

O estado mudou? O objeto veio de algum lugar? A pessoa precisa perceber relação entre duas telas? Se a resposta for negativa, a animação é decoração e precisa justificar o custo. Transição de 150 a 250 milissegundos costuma servir a feedback local; sequências narrativas exigem timeline e controle explícito.

Opacidade e transformações preservam o fluxo melhor do que animar largura, altura e posição de layout. `prefers-reduced-motion` troca deslocamento por mudança discreta, remove paralaxe e desliga autoplay. A preferência é funcional, não uma versão empobrecida.

Qual biblioteca usar

CSS cobre hover, foco, expansão simples, skeleton e scroll-driven animation quando o suporte e o fallback foram verificados. Motion para React oferece layout animation, gestos e o hook `useReducedMotion`, adequado a interfaces de produto. GSAP e ScrollTrigger entram quando a timeline, o scrub, o pin e a coordenação justificam uma dependência especializada.

Rive serve a ilustrações interativas controladas por máquina de estados. O runtime pode pausar, parar e assentar quando nenhuma transição precisa avançar, reduzindo trabalho. Lottie reproduz animação vetorial exportada do After Effects, boa para peças lineares e ícones; teste o arquivo real, porque efeito, máscara e expressão nem sempre sobrevivem iguais.

Three.js só entra quando a cena 3D é parte do produto ou da demonstração. O renderer usa WebGL 2, expõe contagem de chamadas e geometria e permite sinal de `powerPreference`; a experiência precisa de fallback estático e limite de pixel ratio em aparelhos móveis.

Bibliotecas de imagens e vídeo

Unsplash oferece licença ampla para imagens, inclusive uso comercial, mas marcas, pessoas reconhecíveis e obras retratadas podem exigir autorização adicional. Pexels cobre fotos e vídeos gratuitos sob licença própria e também proíbe sugerir endosso. A equipe registra fonte, autor, data de download, licença e uso aprovado para cada ativo.

Cloudinary entrega transformações dinâmicas, `srcset`, `sizes`, formatos e qualidade automáticos para imagem e vídeo. Mux Player cuida de streaming adaptativo, controles, legendas e variação de qualidade. Para acervo pequeno, arquivos próprios com `<picture>`, `<video>`, poster, legendas VTT e CDN podem ser suficientes; a plataforma especializada compensa quando volume, múltiplas rendições e analytics viram rotina.

Pipeline de mídia que não quebra no deploy

Origem e direitos vêm antes da edição. O tratamento cria corte, compressão, poster, alt text e legendas. As variantes cobrem largura, densidade e formato. A entrega define prioridade, lazy loading, streaming e cache. A medição fecha com bytes, LCP, taxa de reprodução, abandono e erro.

Nunca use imagem de texto para conteúdo que precisa ser lido, traduzido ou recortado. O texto fica no HTML e a imagem preserva função visual. Vídeo com fala recebe legendas; vídeo essencial também precisa de alternativa quando a informação não está disponível por áudio e legenda apenas.

Oficina de animação e mídia

Escolha uma seção real. Escreva qual mudança o movimento precisa explicar. Faça primeiro em CSS. Suba para Motion ou GSAP apenas se o protótipo provar que a coordenação é necessária. Adicione a versão reduzida, meça long tasks e confirme que a tela continua compreensível parada.

Para a mídia, substitua uma imagem responsiva por art direction com duas composições e `sizes` correto. Troque um MP4 único por poster e rendições adaptativas ou por um serviço especializado. O critério de pronto é a mesma mensagem com menos bytes, sem layout shift e com alternativa acessível.

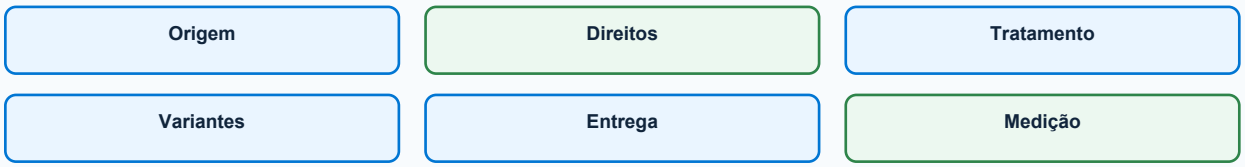
Escada de complexidade: use a primeira camada que resolve

- **CSS** estado simples
- **Motion** produto React
- **GSAP** timeline complexa
- **Rive / Lottie** arte animada
- **Three.js** 3D necessário

Matriz de bibliotecas e serviços

Ferramenta	Use quando	Evite quando
CSS / Web Animations	estado simples, reveal, transição e scroll nativo	a timeline exige orquestração extensa
Motion	produto React, gestos, presença e layout animation	o efeito é resolvido por duas regras CSS
GSAP	timeline, scrub, pin e coordenação de cenas	a equipe não manterá o roteiro nem o fallback
Rive	ilustração interativa com máquina de estados	a arte é linear ou a dependência WASM não se paga
Lottie	ícone e narrativa vetorial linear exportada	há expressão não compatível ou interação complexa
Three.js	3D é conteúdo ou demonstração central	é apenas fundo decorativo de uma landing
Cloudinary	muitas variantes e transformações de mídia	o acervo é pequeno e estável
Mux	vídeo adaptativo, legenda, player e analytics	um clipe curto e local resolve com poster

Pipeline de mídia: a otimização começa antes do arquivo



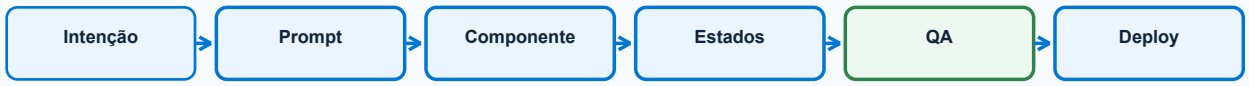
Checklist de pronto de animação e mídia

- Cada movimento responde a uma mudança de estado ou relação espacial.
- A versão reduzida é desenhada e testada, sem autoplay ou paralaxe.
- A animação preserva teclado, foco e leitura do conteúdo parado.
- Imagem e vídeo têm origem, licença e autorização registradas.
- Imagem responsiva usa art direction quando o assunto muda de enquadramento.
- Vídeo possui poster, controles quando informativo e legenda VTT.
- Bytes, LCP, long tasks, abandono de reprodução e erro entram na medição.

Instrumentos de decisão

Estas matrizes transformam o curso em uma ferramenta de revisão. Use-as antes do primeiro prompt, no fim de cada onda e no pull request que pede publicação.

O pipeline que liga intenção a deploy



Régua de escolha do componente

Pergunta	Se sim	Se não
Há comportamento acessível complexo?	use primitivo testado	componente próprio pode bastar
A peça repete em três rotas?	promova ao design system	mantenha local até estabilizar
A identidade exige forma própria?	estilize sobre primitivo	use variante padrão
Existe dono de atualização?	registre versão e rotina	reduza dependência
A saída é reversível?	documente a migração	evite lock-in

Gate por etapa

Etapa	Prova	Falha que bloqueia
Briefing	público, tarefa, estados e aceite	pedido aberto a interpretação
Design	tokens e componentes	valor visual repetido sem fonte
Implementação	diff pequeno e estados completos	segredo, erro silencioso ou dependência inventada
QA	teclado, mobile, contraste e rede	overflow, foco oculto ou LCP fora do teto
Deploy	preview, observabilidade e rollback	produção sem caminho de volta

Caderno de consulta do curso

Esta parte recompõe a cobertura dos arquivos que publicam os 37 capítulos do curso. Os trechos foram selecionados ao longo de cada módulo, preservando começo, meio e fechamento, para manter prompts, comparações, exercícios e critérios de decisão sem repetir integralmente cada bloco visual da página.

Caderno 1. Fundamentos e capítulos incorporados à página principal

Fonte interna: `src/app/educacao/frontends-com-vibecoding/page.tsx`

As cinco partes que toda loja online repete:

Abra a loja onde você comprou da última vez e olhe a página inicial. Ela tem cinco partes, e as mesmas cinco aparecem no Mercado Livre, na Amazon e na Shopee: um cabeçalho com logo, campo de busca e carrinho; uma faixa de destaque com a oferta do dia; uma grade de produtos; o cartão de produto que se repete dentro dessa grade; um rodapé com contato e política de privacidade.

Construir frontend é montar essas partes, ligar cada uma aos dados que vêm do servidor e decidir o que acontece quando o cliente clica. O cartão de produto é a peça que mais ensina o método: você desenha um só, com foto, nome, preço e botão, e a página repete esse mesmo cartão dezenas de vezes com dados diferentes.

Faça agora, em três minutos: abra o site que você mais usa e escreva as cinco partes dele numa folha. Esse rascunho é a lista de peças que você vai pedir à IA no próximo módulo.

Por que trocar uma cor às vezes custa uma tarde inteira:

Guarde cada decisão visual num lugar só. O azul da marca, o espaçamento padrão e o raio de canto dos botões viram variáveis com nome, e cada tela lê a variável em vez de repetir o valor.

Veja a diferença na prática. Com o valor `#1d4ed8` copiado em trinta arquivos, trocar o azul da marca é trinta edições, e a que você esquecer vai aparecer numa página perdida do site. Com uma variável `--cor-marca` lida por todas as telas, é uma edição.

O mesmo raciocínio vale para as bibliotecas que você instala. Cada uma traz outras bibliotecas junto, e a maior parte do código que roda no navegador do seu cliente foi escrita por gente que você nunca vai conhecer. Rode `npm outdated` uma vez por mês para ver quais dessas peças ficaram para trás.

Quatro atalhos que economizam hoje e cobram depois

Cinco critérios decidem a stack. Responda os cinco antes de instalar qualquer coisa.

1. Interatividade: diga se a tela reage ao clique o tempo todo, com filtro, arrastar e atualização ao vivo, ou se o visitante apenas lê. 2. Descoberta: diga se o conteúdo precisa aparecer no Google e ser citado por assistentes de inteligência artificial. 3. Simplicidade: diga se a peça precisa abrir de um arquivo, sem servidor e sem instalação. 4. Manutenção: diga quem cuida disso daqui a um ano e com qual tecnologia essa pessoa já trabalha. 5. Desempenho: diga quanto JavaScript pode chegar ao celular do seu cliente antes de a tela travar.

A regra que sai dos cinco cabe numa linha: escolha a stack mais simples que ainda atende à interatividade de que você precisa. Subir um Next.js completo para servir um relatório de uma página é comprar build, dependência e atualização que o problema não pediu.

O Astro na prática, do zero ao site no ar:

O Astro é um gerador de sites que entrega HTML pronto e só manda JavaScript para o navegador nos pedaços que você marcar.

O que vem dentro dele: rotas criadas a partir da pasta `src/pages`, suporte a Markdown para escrever conteúdo, coleções de conteúdo que conferem os campos de cada post, otimização automática de imagem e as ilhas, que são componentes React, Vue ou Svelte ligados só onde você pedir.

Quando usar: blog, documentação, site institucional, página de campanha, qualquer site em que o conteúdo é o produto. Quando não usar: aplicativo com login, painel com dado que muda a cada segundo, tela cheia de

formulário conectado ao servidor. Nesses casos, vá de Next.js.

O que mudou em agosto de 2026, depois do retrato de julho:

A linha 16.3 do Next.js saiu como estável em 3 de agosto de 2026, com redução de até 90% da memória no servidor de desenvolvimento e a suíte Instant Navigations. Desde 20 de julho de 2026 o framework passou a ter lançamento mensal de segurança.

O React não teve versão menor nova no trimestre. A linha estável segue em 19.x com correções, entre elas a 19.2.8, de 21 de julho de 2026.

O que fazer com isso: assine o aviso de lançamento do framework que você usa e reserve uma hora por mês para rodar `npm outdated` e aplicar a correção de segurança. Saber a versão deixou de bastar, porque agora existe um patch com cadência e alguém precisa aplicá-lo.

Onde o estado do React encontra a classe do Tailwind:

Junte os dois assim: o React guarda o estado numa variável e você escolhe a classe a partir dela. Um botão de alternar tema guarda `escuro` como verdadeiro ou falso, e a classe do cartão vira `escuro ? bg-slate-900 : bg-white`.

A regra que evita bagunça: mantenha a lista de classes curta e previsível. Quando a expressão de classe passar de duas condições, mova a decisão para uma função `variantes` fora do JSX, ou use a biblioteca `cva`, que existe justamente para isso.

O comportamento é a parte que você não improvisa. Um diálogo acessível precisa prender o foco dentro dele, devolver o foco ao botão que o abriu quando fecha, fechar com `Esc`, declarar `role=dialog` e `aria-modal` e esconder o resto da página do leitor de tela. Um agente improvisando com `div` e `onClick` erra quase todos esses pontos, e o defeito passa despercebido para quem testa só com o mouse.

Guarde a divisão que organiza o capítulo inteiro: a acessibilidade vem da primitiva, o visual vem de você.

Desde julho de 2026 o shadcn/ui abre projetos novos com o Base UI, mantido pela equipe do MUI, e o Radix continua disponível como opção explícita.

Use o nome que o resto do mundo usa:

Antes de pedir uma peça, confira o nome dela num catálogo público. O component.gallery reúne 60 componentes, 95 design systems e 2.671 exemplos reais, e é ali que as distinções finas ficam claras: popover abre no clique e aceita interação dentro do painel; tooltip abre ao passar o mouse e só mostra texto; e o que um projeto chama de Accordion outro publica como Collapse, Disclosure ou Expander.

O ganho é direto. Pedindo pelo nome certo, você recebe o comportamento convencional junto, incluindo teclado e leitor de tela. Peça inventada com nome próprio custa explicação toda vez que alguém novo abrir o código.

Anatomia de um componente shadcn/ui: o Button como exemplo

Escolha um estilo e uma forma, e fique neles:

O Hugeicons, com mais de 59.000 ícones, oferece cinco estilos (Stroke, Solid, Duotone, Twotone e Bulk) em três formas (Rounded, Standard e Sharp). São quinze combinações possíveis, e o seu produto inteiro mora numa só.

Mistura de estilos é o defeito visual mais fácil de enxergar de longe: dois ícones lado a lado, um de traço e um preenchido, denunciam a tela antes de qualquer análise.

O erro de pedido nasce daí. Pedir o ícone de salvar num prompt e o da lixeira em outro, meia hora depois, termina com traços e tamanhos que não conversam, porque cada pedido isolado é uma nova chance de o agente escolher outro padrão. Peça o conjunto de uma vez, com a espessura de traço escrita e a cor vindo do token.

Quando usar cada recurso visual

O estado base é sempre visível. O `opacity: 0` mora somente dentro do `@keyframes from`; na regra base do elemento ele apaga o conteúdo de quem não recebe a animação. Com `animation-fill-mode: both`, se a animação não rodar, o conteúdo permanece visível. Esse é o antídoto do bug 'carrega e some', e a mesma garantia serve a quem pediu menos movimento ao próprio aparelho: sem animação, a interface continua inteira.

Checklist de animação: rodar antes de mergear

Comece pelo rótulo:

Troque todo rótulo genérico por um que descreve a ação. Em vez de Continuar, escreva Criar conta grátis. Em vez de Enviar, escreva Baixar o PDF. O Nielsen Norman Group chama isso de aroma de informação: a propriedade de um link dizer, antes do clique, o que a pessoa encontra do outro lado.

Menu mal resolvido é a forma mais rápida de perder o visitante. Ele não acha o que procura, perde a noção de onde está e sai sem reclamar com ninguém.

Antes de escolher a forma do menu, defina o propósito. Navegar entre páginas pede com uma lista de links. Executar um comando na própria tela pede menu de ação. Mostrar e esconder um bloco de conteúdo pede accordion ou popover. Escolher a forma antes do propósito é o que produz menu de site se comportando como menu de aplicativo de computador.

Navegação de qualidade é um sistema em camadas:

O navbar.gallery, galeria pública dedicada a navegação, cataloga nove tipos: barra estática ou fixa, dropdown e flyout, mega menu, sidebar, busca, faixa de aviso (announcement bar), navegação de tela cheia, breadcrumbs e abas, cada um com a sua faixa de uso. O padrão dos destaques da galeria é combinar vários tipos em camadas deliberadas, em vez de apostar tudo num só. Duas regras práticas saem direto do catálogo: acima de cerca de seis destinos no mesmo item, o mega menu é a resposta padrão, porque dropdown comprido vira lista de rolagem; e a faixa de aviso vive numa camada separada ACIMA da navbar, nunca competindo com os itens de navegação pelo mesmo espaço. Cloudflare e Asana são bons repertórios de exemplo do arranjo em camadas.

Uma nota de plataforma que barateia tudo isso: o posicionamento por âncora (anchor positioning) entrou no Baseline em janeiro de 2026, com a chegada do Firefox 147, o que viabiliza popover ancorado ao gatilho sem biblioteca de posicionamento.

Checklist de navegação por teclado: rodar em todo PR

Mapa de tipos de navegação: quando usar cada padrão

O Recharts na prática, em cinco passos:

O Recharts é uma biblioteca de gráficos para React que desenha em SVG e monta cada gráfico com componentes.

O que vem dentro dela: os tipos de gráfico prontos (`BarChart`, `LineChart`, `AreaChart`, `PieChart`), as peças que você encaixa dentro deles (`XAxis`, `YAxis`, `Tooltip`, `Legend`, `CartesianGrid`), o `ResponsiveContainer` que ajusta a largura sozinho e o `accessibilityLayer`, que liga a navegação do gráfico pelo teclado.

Quando usar: painel em React com séries de tamanho normal. Quando não usar: dezenas de milhares de pontos, porque o SVG cria um elemento por ponto e a tela engasga; nesses casos vá de uPlot ou ECharts.

1. Rode `npm install recharts`. 2. Importe as peças: `import { ResponsiveContainer, BarChart, Bar, XAxis, YAxis, Tooltip } from "recharts";`. 3. Envolva o gráfico num . Resultado esperado: o gráfico acompanha a largura da coluna ao redimensionar a janela. 4. Passe a cor pelo seu token, com `fill="var(--color-primary)"` na . Resultado esperado: o gráfico muda junto com o tema escuro. 5. Ligue o `accessibilityLayer` no e escreva abaixo a mesma série numa com a classe `sr-only`. Resultado esperado: quem usa leitor de tela ouve os números, e quem usa teclado percorre as barras.

O axe na prática, em quatro passos:

O axe é o verificador de acessibilidade que roda dentro do navegador e lista as falhas com o elemento exato e a regra quebrada.

O que vem dentro dele: as regras da WCAG que dá para conferir por máquina, o cálculo de contraste com a cor final composta, a extensão para Chrome e Firefox, e a versão de linha de comando para rodar no seu processo de publicação.

Quando usar: em toda tela, antes de publicar. Quando não usar como prova: para declarar conformidade, porque metade dos critérios só se confere à mão.

1. Instale a extensão axe DevTools no navegador e abra a sua página. 2. Abra as ferramentas do desenvolvedor com F12, vá até a aba axe DevTools e clique em analisar. Resultado esperado: uma lista de problemas, cada um com o elemento destacado na tela. 3. Troque o tema para escuro e analise de novo. Resultado esperado: as falhas de contraste do tema escuro aparecem, e elas quase nunca são as mesmas do tema claro. 4. Para rodar sem abrir o navegador, use `npx @axe-core/cli http://localhost:3000`. Resultado esperado: o mesmo relatório no terminal, pronto para virar um portão antes de publicar.

Roteiro de auditoria WCAG 2.2: quatro camadas em sequência

Os portões antes de publicar:

Encadeie as verificações num comando só e rode esse comando antes de todo push. O que a IA escreveu chega com arestas previsíveis: tipo frouxo, aviso de lint ignorado, nenhum teste, imagem pesada e nenhuma marcação para buscador.

1. `tsc --noEmit` confere os tipos e pega o parâmetro que mudou de nome. 2. O lint, com Biome ou ESLint, pega variável não usada e regra de acessibilidade quebrada no JSX. 3. Os testes pegam a regressão, que é a coisa que funcionava e parou de funcionar. 4. O `build` pega o import que só existia na sua máquina. 5. O Lighthouse CI pega a página que ficou lenta desde a semana passada.

Junte tudo num script chamado `verify`, encadeado com `&&`, para o primeiro que falhar impedir os próximos de rodar. No `package.json`, escreva algo como `tsc --noEmit && biome check . && vitest run`.

Mire dez segundos nesse comando. Portão lento é portão que alguém pula, e portão pulado não protege ninguém.

A medida que importa depois de publicar é o tempo de voltar atrás. Publicar rápido impressiona; reverter em dois minutos é o que salva a madrugada.

Git flow para feature: do vibecoding ao merge

Dashboard do CI/CD: o que cada gate mostra

A hora de juntar tudo:

Construa um painel de uma página em três pedidos, revisando cada um antes de pedir o próximo. Você já dirigiu o agente, montou componentes, cuidou de ícone, contraste, velocidade e publicação; falta integrar as peças num projeto do começo ao fim.

Pedir tudo de uma vez devolve um bloco grande demais para você julgar, e o que não dá para julgar acaba aceito no escuro.

A sequência tem três movimentos: executar os três pedidos, reconhecer os anti-padrões pelo sintoma na tela e revisar o resultado contra o checklist final.

Uma regra atravessa o capítulo inteiro: você não terceiriza o entendimento. Cada trecho aceito sem leitura é uma responsabilidade que você assumiu sem saber qual. Leia, teste, explique em voz alta com as suas palavras, e só então siga em frente.

CÓDIGO E DEPENDÊNCIAS ├── [ ] Li e entendi cada diff; não há código "mágico" sem explicação └── [ ] Toda dep existe no npm na versão citada (npm view)

ACESSIBILIDADE E CONTRASTE ├── [ ] Contraste AA em todo texto nos dois temas (axe-core) ├── [ ] Operável 100% por teclado (Tab / Esc / Enter / foco visível) ├── [ ] Nenhum estado comunica só por cor └── [ ] Modal: botão fechar ≥ 44px + Esc + foco preso

VISUAL E MOTION ├── [ ] Ícones Lucide coerentes (size/strokeWidth/token); zero emojis ├── [ ] Todo SVG tem fill próprio; contêiner com largura definida ├── [ ] Estado base sempre visível (sem opacity:0 fora de keyframes) └── [ ] prefers-reduced-motion: reduce → animações suprimidas

PERFORMANCE E MOBILE ├── [ ] Testado em 390px: nada vaza, tabelas com scroll-x ├── [ ] LCP < 2,5s e CLS < 0,1 no Lighthouse └── [ ] npm run verify (tsc + lint + testes) passa sem erro

DESCOBERTA ├── [ ] JSON-LD válido, sitemap só com URLs 200 ├── [ ] canonical = og:url = URL do sitemap (mesma forma) └── [ ] IndexNow disparado após deploy

Dashboard de projeto final: três iterações de prompt

Maturidade do projeto que você está construindo

A oficina tem quatro estações:

1. O agente que escreve, o Claude Code rodando no terminal, dentro da pasta do projeto. 2. O projeto no ar, com `npm run dev` ligado em outra janela do terminal. 3. O navegador, que o agente abre para ler o console e medir contraste. 4. O git, que guarda cada tela que funcionou.

A diferença entre um chat comum e um agente de codificação é que o segundo escreve arquivos no seu disco. Ele lê a pasta inteira, cria arquivo novo, edita o existente e roda comando, o que muda o seu trabalho de copiar e colar para descrever e revisar.

Com a máquina digitando, o gargalo vira descrever. Existe um teste barato para saber se a sua intenção está clara: termine em voz alta a frase esta tela deu certo se. Se você não consegue completar, o agente também não vai conseguir.

Commite a cada tela que funciona, com uma mensagem curta. É isso que deixa você pedir mudanças ousadas sem medo, porque voltar custa um comando.

Setup do ambiente Claude Code em quatro passos

O Slidev na prática, em cinco passos:

1. Rode `npm init slidev@latest`. Ele cria a pasta com um `slides.md` de exemplo. 2. Rode `npm run dev` dentro dela. Resultado esperado: a apresentação abre no navegador e cada `---` do arquivo virou um slide. 3. Edite o `slides.md` num editor de texto. Resultado esperado: o slide aberto se atualiza enquanto você digita. 4. Aperte a tecla `p` no navegador para abrir o modo apresentador. Resultado esperado: você vê o próximo slide, as suas notas e um cronômetro; a plateia vê só o slide. 5. Rode `npx slidev export`. Resultado esperado: um PDF com o texto selecionável, pronto para anexar num e-mail.

O erro que todo iniciante comete: colar no slide o parágrafo inteiro que ia falar. O texto longo vai para a nota do apresentador, num comentário HTML, e no slide fica a frase que a plateia precisa ler.

Slide eficaz: anatomia de oito slides de um deck profissional

O que a empresa compra quando compra uma interface

O funil: ser encontrado, ser entendido e converter

Seis empresas e dois benchmarks sob a lupa

Risco, entrega e condução

Apêndice: para quem vai pôr a mão

Curso gratuito de interface para quem aprova, contrata e responde pelo resultado. [valor dinâmico] módulos em sete etapas de decisão mais um apêndice opcional: o que a empresa assume ao aprovar uma interface, o briefing que evita retrabalho, escolha de tecnologia lida como escolha de fornecedor com custo de saída, governança de design system, a interface como funil de descoberta e conversão, os cinco instrumentos de decisão que chegam à mesa da diretoria (painel, tabela, mapa, análise e diagrama), seis empresas e dois capítulos de benchmark de stack sob a lupa (Stripe, Booking, Mercado Livre, LATAM, Apple e Temu), acessibilidade e desempenho como exposição jurídica e receita, segurança de interface, portões de qualidade, condução de um programa de modernização em ondas, o checklist de revisão do entregável e a escolha do fornecedor de inteligência artificial que gera as telas, com Claude, OpenAI, Grok e Gemini comparados por preço, benchmark independente e política de roteamento por tarefa. A trilha executiva não passa por terminal, por agente nem por código. Em português.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 2. A11y extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/a11y-extra.ts`

O relatório verde do axe confere presença, e só isso. Atributo `alt` que falta, rótulo sem campo correspondente, contraste calculável abaixo do mínimo, atributo ARIA inválido, título fora de ordem: tudo isso é conferência de existência no código, e é o piso do trabalho. Circula no setor a estimativa de que a verificação automática cubra cerca de um terço dos problemas reais, sem fonte primária que sustente o número com método aberto.

A fração exata importa pouco, porque a natureza do que sobra é conhecida e nada disso se descobre lendo marcação. Se a ordem de leitura faz sentido. Se o anúncio é compreensível. Se o foco para onde deveria. Se a região de anúncio fala na hora certa. E a pergunta que resume as outras: se dá para concluir a tarefa. A máquina responde se o atributo existe; só alguém ouvindo a página responde se ela funciona.

Esse alguém é você, e custa dez minutos por fluxo. Ligue o leitor de tela, feche os olhos e tente comprar alguma coisa. Na primeira tentativa aparecem defeitos que nenhum scanner reporta: o botão anunciado duas vezes, o ícone anunciado como `imagem`, a janela sobreposta que abre sem levar o foco junto e o link que diz apenas `clique aqui`, sem dizer para onde leva.

Grave a tela e o áudio da tentativa e guarde com a data. A gravação é a linha de base contra a qual você compara a versão seguinte, e é a única prova que sobrevive à sua própria memória de como estava antes.

Os seis passos do teste de ouvido, com o tempo de cada um: ligar o leitor, ouvir a página inteira, percorrer com Tab, saltar títulos, abrir a lista de links e concluir a tarefa de olhos fechados.

Desenhe o contêiner desde o início e altere apenas o texto depois, peça que a frase seja lida inteira em vez de palavra por palavra e mantenha o contêiner audível sem ocupar espaço visível na tela.

Os dois mecanismos competem entre si quando disparam juntos, e essa é a razão pela qual a correção completa às vezes entrega menos que a correção parcial. Mover o foco e anunciar uma mudança no mesmo instante coloca o leitor de tela diante de dois pedidos, e ele atende o primeiro: o foco recém-movido tem prioridade e o texto da região de anúncio é engolido sem ser lido. A regra que resolve é uma estratégia por transição. Troca de tela pede foco no título principal, porque o título já informa onde a pessoa chegou. Mudança parcial, como filtro aplicado ou item salvo, pede região de anúncio, porque aí o foco precisa continuar onde o cliente o deixou. Fazer as duas coisas ao mesmo tempo entrega menos que fazer uma só.

Trilha praticante, para quem vai abrir o componente. Três regras pegam a maior parte dos defeitos de ordem de foco em menu, campo de busca com sugestões, janela sobreposta e abas. A ordem do teclado acompanha a ordem visual, e valor de prioridade positivo é proibido, sem exceção, porque um único número positivo reordena a página inteira para quem navega por teclado. A janela sobreposta prende o foco dentro dela enquanto está aberta e o devolve ao elemento que a abriu ao fechar. Em grupos de itens equivalentes, apenas um é alcançável pela tecla de tabulação e as setas movem o foco dentro do grupo, o que faz a tabulação pular o conjunto inteiro em vez de percorrer item por item.

Medir antes de cortar, e medir onde o cliente está:

Meça antes de cortar. O maior responsável pela demora em responder ao toque, e por parte da demora em mostrar o conteúdo, é a quantidade de código enviada de uma vez. A cura é conhecida: quebrar o pacote por página, carregar o que é pesado apenas quando ele for usado e importar somente as funções necessárias em vez do pacote inteiro. O que a maioria erra é a ordem: cortar por palpite retira o que não pesava e mantém o que pesava. Abra primeiro o relatório de composição do pacote e veja quais dependências ocupam o espaço, porque a resposta raramente é a que você apostaria antes de olhar.

A biblioteca web-vitals, a ficha completa. O que é: uma biblioteca oficial do Google que mede, no navegador de cada visitante, as três medidas públicas de experiência da página. O que vem dentro: as funções `onLCP`, `onINP` e `onCLS`, mais o valor de cada medida com o detalhe do elemento que causou o pior caso. Quando usar: em produção,

sempre que você quiser saber o que os seus visitantes realmente sentem. Quando não usar: no lugar de uma auditoria de laboratório durante o desenvolvimento, porque as duas medem coisas diferentes.

Guia passo a passo.

1. Instale com `npm install web-vitals`. 2. Crie um arquivo que rode apenas no cliente e importe as três funções: `import { onLCP, onINP, onCLS } from "web-vitals"`. 3. Chame cada uma passando a mesma função de envio, do tipo `onLCP(enviar)`. 4. Dentro de `enviar`, mande o nome da medida e o valor para onde você guarda dados, com `navigator.sendBeacon`. 5. Espere alguns dias de tráfego real e compare os seus números com os limiares publicados.

Resultado esperado: três números vindos dos aparelhos dos seus visitantes, que costumam divergir do resultado bonito da auditoria local. O erro que todo iniciante comete é medir uma vez, no próprio computador, com a rede do escritório, e tratar isso como o desempenho do produto.

A figura abaixo separa os dois métodos e traz os três limiares que entram no requisito, sempre no percentil 75.

Adiantar a próxima página no cursor ou no foco do link derruba o tempo do clique para milissegundos; adiantar a lista inteira gasta os dados e o processador do cliente. A segunda fatura cai na sua medição, porque adiantar executa o código da página e a análise conta visita que não houve.

Acessibilidade e medição: o que já venceu e o que está agendado

As duas pontas do calendário: a regra interina americana empurrou os prazos do Título II para 26/04/2027 e 26/04/2028, mantendo o padrão WCAG 2.1 AA, enquanto a liminar de 07/07/2026 deu 180 dias à União sob multa diária de R$ 10 mil. O registro que protege você é o mesmo dos dois lados.

Duas anotações fecham o seu registro de cada tela que você publica. A primeira é a data em que você percorreu os três fluxos principais com leitor de tela, escrita ao lado da gravação. A nota de uma varredura automatizada não substitui essa linha, e foi exatamente essa troca que o tribunal de Caen recusou. A segunda são as três medidas de experiência no percentil 75, ao lado das mesmas medidas dos dois sites com que você compete. Esses números são públicos e você levanta os três no relatório de experiência de página do Google em poucos minutos.

O relatório verde do começo continua valendo pelo que ele é: prova de que os atributos existem no código. A pergunta que ele nunca respondeu é se a pessoa cega termina a compra, e responder isso custa dez minutos de escuta por fluxo. Gastar esses dez minutos antes de publicar transforma o defeito em correção de uma tarde. Publicar sem gastar transforma o mesmo defeito no que o Carrefour France encontrou: nome de empresa, percentual de painel e valor de multa por dia.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 3. Agente nativo extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/agente-nativo-extra.ts`

Houve um período em que o agente improvisava a interface de programação de uma biblioteca a partir do que estava no treinamento, e errava sempre que a biblioteca tinha mudado. O ecossistema respondeu publicando contexto oficial para agentes: uma conexão pelo protocolo de contexto, abreviado como MCP na documentação e que é o padrão pelo qual o agente conversa com ferramentas e sistemas externos, uma skill ou um arquivo de instruções lido pelo agente. A primeira pergunta antes de qualquer pedido sobre uma biblioteca de interface extensa deixou de ser como escrever o pedido: passou a ser se o fornecedor publicou contexto oficial desta biblioteca. Quando publicou, carregar esse contexto elimina a invenção de interface de programação na origem, o que é diferente de corrigir o erro depois na revisão.

Três palavras aparecem daqui em diante. Contexto oficial é a documentação que o próprio fornecedor publica em formato que uma máquina lê. Conexão é o canal pelo qual o agente consulta esse contexto em tempo de trabalho. Invenção de interface de programação é o erro em que o modelo afirma que existe um comando que nunca existiu, e ele só aparece quando alguém tenta rodar o código.

Ao lado do contexto por biblioteca, circulam listas de regras objetivas de interface, escritas no formato de obrigação, recomendação e proibição, cobrindo interação, formulário, tema escuro, área segura da tela e anti-padrões conhecidos. Elas chegam em dois formatos: um arquivo de instruções que o agente aplica durante a geração e um comando de auditoria que devolve os achados com arquivo e linha. Confira o endereço do repositório e o nome do comando dessas listas na fonte antes de adotá-las, porque lista de regras citada de memória tem o mesmo defeito que ela pretende corrigir. O padrão de uso é estável: regra objetiva aplicada durante a geração pega desvio estrutural, e auditoria da tela rodando pega o que só existe depois de renderizar. Uma não substitui a outra.

Dentro do projeto, as conexões ficam declaradas num arquivo versionado, e quem clona o repositório herda as mesmas permissões de acesso. É aí que a conveniência vira decisão de segurança. Construir uma conexão própria compensa quando o agente precisa de um dado que só existe num sistema seu, como a sua tabela de cores: em vez de você colar esses valores a cada sessão, o agente busca na fonte. O critério de parada é curto: conexão nova entra quando o dado muda, vive num sistema seu e seria repetitivo passar à mão. Fora disso, cada conexão é mais uma porta aberta para manter. Servidor remoto entra pelo endereço publicado na documentação:

Faça você mesmo: medir o ganho do contexto oficial em uma tarde

Carregado o catálogo, falta a oficina que o mantém em uso. O agente recebe duas classes de instrução, e a diferença entre elas decide quanto trabalho você refaz. A primeira você digita no pedido do dia, e ela se perde quando a sessão fecha. A segunda mora em arquivos dentro do projeto e é recarregada em toda tarefa, sem você digitar de novo. O conjunto desses arquivos aparece na documentação sob o nome de harness: memória do projeto, conexões, atalhos de fluxo, guardas automáticas e trabalho em paralelo.

Em frontend o ganho é maior do que em qualquer outra frente, porque as exigências que mais importam se repetem em todo componente: a cor sai de um token, o texto passa no contraste mínimo, o foco de teclado fica visível, toda lista trata o estado vazio e todo pedido de dado trata carregamento e erro.

A memória do projeto, a ficha completa. O que é: o arquivo `CLAUDE.md` na raiz do repositório, que o agente relê antes de cada tarefa. O que vem dentro: a stack real, os comandos do projeto, as convenções de pasta e nome e as regras de frontend que você não quer repetir.

Guia passo a passo.

1. Crie o arquivo `CLAUDE.md` na raiz do repositório. 2. Escreva a seção Stack, com framework, versões e sistema de estilo que o projeto usa de verdade. 3. Escreva a seção Comandos, copiando os scripts que já existem no `package.json`. 4. Escreva a seção Regras de frontend com afirmações conferíveis. `Escreva bom código` não muda nada na saída; `texto secundário usa text-slate-600, nunca text-slate-400` muda cada geração,

porque descreve um estado que você confere olhando o arquivo pronto. 5. Mantenha o arquivo entre 40 e 60 linhas, porque cada linha é cobrada como contexto em todo pedido. 6. Abra uma sessão nova, peça um componente qualquer e confira se as regras foram respeitadas sem você repetir nenhuma.

Resultado esperado: o próximo componente já sai com os seus tokens e o seu contraste. O erro que todo iniciante comete é encher o arquivo de frases genéricas sobre qualidade, que ocupam contexto pago e não mudam uma linha do resultado.

O arquivo da raiz vale para o projeto inteiro, uma subpasta pode ter o seu próprio, e existe ainda um arquivo global de usuário para as suas preferências pessoais. Comece pelo da raiz, que rende mais por linha escrita.

Prompt pronto: o agente escreve o retrato do projeto, você revisa

O calendário de manutenção da oficina: o que já venceu, o que está correndo

invocado por /auditar-tela --> --- description: Audita a tela aberta no browser MCP (contraste, console, teclado) e corrige --- Abra http://localhost:3000$ARGUMENTS no browser via MCP e:

1. Leia o console; aponte erros e warnings. 2. Meça o contraste de cada texto contra o fundo; cite valores abaixo de WCAG AA. 3. Teste navegação por teclado; confirme foco visível. 4. Corrija no código o que reprovar, reabra a tela e confirme que passou.

Não altere o layout; ajuste só cores e o necessário para acessibilidade.

Prompt pronto: escrever e auditar ao mesmo tempo

Faça estas duas coisas EM PARALELO, cada uma num subagente próprio:

Subagente 1 (escrever): crie o componente em React + Tailwind, com props { titulo, preco, destaque }, seguindo os tokens e regras do CLAUDE.md.

Subagente 2 (auditar): abra a tela /precos no browser via MCP e audite contraste WCAG AA, foco por teclado e erros de console em TODOS os cards já existentes; liste cada reprovação com o valor medido e o seletor.

Quando ambos terminarem, consolide: me mostre o novo componente e a lista de reprovações da auditoria, separadamente.

Resultado esperado: duas saídas independentes na mesma janela de tempo, e nenhuma delas contaminada pelo caminho que a outra percorreu.

O modo de planejamento: o agente investiga sem tocar em arquivo, o plano mostra alcance, casos sem equivalente e ordem das etapas, e a correção acontece nele, barata, antes da aprovação que libera a edição.

Prompt pronto: o plano antes da primeira edição

As nove peças da oficina: o problema de cada uma e quando instalá-la

Custo e risco por decisão de governança do assistente

Faça você mesmo: montar a oficina do seu projeto, na ordem em que as peças rendem

Cinco gestos fecham a oficina, e os cinco cabem numa hora. Primeiro, abra o seu `CLAUDE.md` e confira se toda regra que você já repetiu duas vezes numa conversa está escrita lá. Segundo, instale o hook que barra o commit quando o verificador reprova, e teste forçando um erro. Terceiro, abra o `.mcp.json` e confirme que nenhuma credencial está escrita em texto puro dentro dele. Quarto, escreva na configuração o teto de auxiliares simultâneos que você aceita, porque ficar no padrão de fábrica também é uma escolha. Quinto, marque no calendário a data em que você relê a tabela de preços dos modelos, que muda mais rápido que a sua configuração.

Critério de uso
Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 4. Animacao midia moderna

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/animacao-midia-moderna.ts`

Animação, imagem e vídeo: as bibliotecas de 2026

Você vai aprender a ordem de gasto do movimento e da mídia: primeiro o que o navegador já faz sozinho, depois Motion e GSAP quando a tela pedir, e depois imagem, ícone e vídeo servidos no formato e no tamanho certos. Cada peça vem com passo a passo, exemplo real e o erro que o iniciante comete na estreia.

Acima do CSS de sempre existe um degrau que ainda quase ninguém usa, e ele é a melhor troca entre esforço e efeito da lista.

A transição entre telas anima a troca de página fazendo um elemento comum às duas continuar de uma para a outra. Numa loja, a foto do produto que estava na vitrine cresce até o lugar dela na página de detalhe, em vez de a tela inteira piscar em branco. O trabalho é dar o mesmo nome ao elemento nas duas telas, com `view-transition-name`, e envolver a atualização em `document.startViewTransition`. A troca dentro da mesma página entrou em Baseline newly available em outubro de 2025, com o Firefox 144; a troca entre páginas diferentes, declarada por `@view-transition`, ainda não é Baseline segundo a MDN, então trate a segunda como melhoria e teste sem ela.

A rolagem também virou linha do tempo. Com `animation-timeline: view()`, o bloco revela conforme entra na tela, sem observador escrito à mão e sem biblioteca. O recurso segue sem interoperabilidade em julho de 2026 e consta como área de foco do Interop 2026, com o Safari entregando correções ao longo de 2026 nas versões 26.4 e 26.5 do WebKit. A consequência prática é uma só: embrulhe em `@supports` e garanta o estado final visível, para que nada suma onde a linha do tempo ainda não existe.

// Degrau 4: GSAP, linha do tempo amarrada à rolagem. npm i gsap import gsap from "gsap"; import { ScrollTrigger } from "gsap/ScrollTrigger";

gsap.registerPlugin(ScrollTrigger);

const mediaQuery = window.matchMedia("(prefers-reduced-motion: reduce)"); if (!mediaQuery.matches) { const linha = gsap.timeline({ scrollTrigger: { trigger: ".secao-cena", start: "top 80%", end: "bottom 40%", scrub: true, // o progresso da cena segue o dedo na rolagem }, });

linha .to(".cena-titulo", { y: -20, opacity: 1, duration: 0.4 }) .to(".cena-produto", { scale: 1.05, duration: 0.6 }, "<0.1") .to(".cena-legenda", { opacity: 1, duration: 0.4 }); } // Sem o if, quem pediu menos movimento recebe a cena inteira mesmo assim.

A animação parece perfeita na sua máquina e engasga no celular do cliente. Três causas explicam quase todos os casos. A primeira é animar propriedade que muda o layout, como `height` numa sanfona; a saída é animar a escala ou usar a sanfona nativa. A segunda é espalhar `will-change` por dezenas de elementos, o que reserva memória de vídeo à toa; use em um elemento por vez e remova depois. A terceira é rodar cálculo dentro do evento de rolagem, que dispara muitas vezes por segundo; deixe a linha do tempo do CSS ou o observador de visibilidade fazerem esse trabalho. Meça no aparelho real, com a limitação de rede e de processador ligada no painel de desenvolvedor.

// Certo: cada ícone pedido pelo nome. import { Play, Pause, Volume2 } from "lucide-react";

// Errado: arrasta a coleção inteira para a conta. // import * as Icones from "lucide-react";

export function ControlesVideo({ tocando }: { tocando: boolean }) { return (

{tocando ? : }

); } // O rótulo vive no botão. O desenho é decorativo e fica oculto para o leitor de tela.

Peso é o assunto que amarra animação, imagem e vídeo, e ele se mede em três minutos. Abra o painel de desenvolvedor do navegador, vá à aba de rede, marque a caixa que desativa o cache e recarregue a página. A barra de baixo mostra quantas requisições saíram e quantos kilobytes desceram, e o filtro por tipo separa imagem, script e mídia.

O alvo público continua o mesmo: até 2,5 segundos para o conteúdo principal aparecer, medido no percentil 75 de visitantes reais. Esse número vale mais que a impressão colhida na sua máquina, porque a sua máquina tem a rede da sua casa e o processador que você escolheu.

A partir daí, o teto por página é uma escolha sua, escrita antes de a primeira imagem entrar no repositório. Um teto que funciona na prática: a imagem de destaque abaixo de 150 kilobytes no espaço real onde ela aparece, o pacote de ícones cobrando só os desenhos usados, e nenhum vídeo descendo antes do play. Repita a medição com a limitação de rede ligada, porque é ela que aproxima o teste do celular de quem compra.

Cada peça de movimento e de mídia, o que ela custa e como perceber o erro

Prompt: pedir movimento e mídia com orçamento declarado

Faça você mesmo: uma seção animada e uma mídia leve, em 25 minutos

O mecanismo por trás de todo este capítulo é um só. Movimento e mídia são as duas coisas que a pessoa recebe antes de entender o que o seu produto faz, e as duas são pagas na conta de dados e na paciência dela.

A disciplina que sobra depois de tirar a técnica é curta. Comece no que o navegador já faz, porque isso não acrescenta peso nem dependência. Suba de degrau quando a tela pedir, com um sinal que você consiga nomear. Sirva cada imagem no tamanho do espaço onde ela aparece, cada ícone pelo nome e cada vídeo com o quadro parado à frente. Confirme tudo com a preferência por menos movimento ligada e com a rede limitada.

Anime uma seção da sua página e prepare a mídia dela dentro de um teto escrito antes de começar.

Critérios de sucesso: -- A animação usa apenas deslocamento e opacidade, e desliga sob a preferência do sistema. -- O conteúdo continua visível e completo quando o script não roda. -- A imagem de destaque desce no formato moderno e no tamanho do espaço real, com largura e altura declaradas. -- O vídeo, se existir, só começa a baixar depois do toque no play. -- Você sabe dizer, em kilobytes, quanto a página baixa na primeira visita.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 5. Benchmark premium

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/benchmark-premium.ts`

O que faz um site parecer caro

Vinte e seis sites de alto padrão de design lidos pelo painel de rede do navegador: onde um site revela sozinho a tecnologia que o gera, em que três famílias essas escolhas se dividem e qual delas serve ao seu caso. Fecha com os cinco traços que aparecem em quase todos eles, quatro dos quais você copia hoje sem gastar nada.

Ferramenta desta aula: o painel de desenvolvedor do navegador. É a única que você precisa, ela já está instalada e não custa nada.

O que vem dentro: uma aba Rede, que lista todo arquivo que a página baixou; uma aba Elementos, que mostra o HTML montado; uma aba Console, para mensagens de erro; e um seletor de velocidade de rede, para simular celular em conexão ruim.

Quando usar: sempre que quiser saber como um site foi feito, quanto ele pesa ou por que a sua própria página demora. Quando não usar: para julgar desempenho geral do site, porque a sua máquina e a sua rede não são as de quem visita.

Passo a passo para tirar o laudo de qualquer site em seis minutos:

1. Abra o site no Chrome ou no Edge e tecle F12. Resultado esperado: o painel abre na lateral ou embaixo.

2. Clique na aba Rede, marque a caixa Desativar cache e recarregue a página com F5. Resultado esperado: a lista de arquivos aparece do zero, sem cópia guardada.

3. Clique na primeira linha da lista, que é o documento HTML, e procure o bloco Cabeçalhos da resposta. Resultado esperado: você lê linhas como Server, Via, CF-Ray e cache-control, e já sabe quem serve o site.

4. Volte à lista e leia os caminhos. Ver /_next/ diz Next.js; ver cdn.prod.website-files.com diz Webflow; ver framerusercontent.com diz Framer; ver /wp-content/ diz WordPress. Resultado esperado: uma frase do tipo "é Next.js, servido pela Vercel, com conteúdo no Sanity".

5. Filtre por Fonte e olhe o domínio da linha terminada em woff2. Resultado esperado: se o arquivo vem do próprio domínio do site, a fonte é comercial e paga; se vem de fonts.gstatic.com, é o serviço gratuito do Google.

6. Leia o rodapé do painel, que soma o peso total e o número de arquivos. Resultado esperado: dois números que você anota e usa como teto na sua própria página.

O erro que todo iniciante comete: esquecer de marcar Desativar cache. Sem isso, o navegador serve arquivos guardados da visita anterior, a lista fica curta e o peso total aparece muito menor do que é para quem chega pela primeira vez.

Sinal que você vê no painel, o que ele prova e com que confiança

Peça assim: o laudo de um site

Audite a stack de frontend deste site e me devolva um laudo com NÍVEL DE CONFIANÇA por campo, separando o que é evidência direta do que é inferência. URL: .

Colete e cruze estes sinais (não invente o que não viu): - Cabeçalhos HTTP da resposta principal (Server, Via, x-wf-region, CF-Ray, X-Amz-Cf-Pop) e de alguns arquivos. - Caminhos de arquivo: procure /_next/, cdn.prod.website-files.com, framerusercontent.com, images.ctfassets.net, cdn.sanity.io, a.storyblok.com, /wp-content/, /cdn-cgi/image/, imagedelivery.net. - Atributos no HTML entregue: __NEXT_DATA__, data-wf-page/site, data-framer-*, meta generator. - Cookies (ex.: grav-site) e parâmetro de URL de arquivo (ex.:

?dpl_ da Vercel). - Entrega de fonte (woff2 no próprio domínio contra Google Fonts) e biblioteca de animação (procure gsap, lottie, rive, framer-motion no HTML e no JS).

Classifique cada campo (framework, CSS, fontes, movimento, hospedagem, conteúdo) como EVIDÊNCIA (cite o sinal exato) ou INFERÊNCIA (diga em que se apoia). Ao final, diga a qual das três famílias (construtor, Next.js composável ou artesanal) o site pertence, e por quê.

Para o seu primeiro site de produto, comece pela família C e suba depois. Escreva em Astro com HTML e CSS próprios, publique na Vercel ou na Cloudflare Pages, e guarde o texto em arquivos Markdown dentro do próprio repositório. Custa zero, serve por borda de rede sem configuração e já cumpre três dos cinco denominadores. Quando outra pessoa passar a escrever o texto, e só então, acrescente o Sanity, que tem camada gratuita e conecta ao Astro com uma biblioteca oficial. Vá para o Webflow ou o Framer apenas se você não pretende escrever código nenhum, sabendo que sair de lá significa reconstruir a página.

As três famílias, dimensão a dimensão

A escada de adoção em cinco degraus: variável nomeada para todo valor (hoje, sem custo), página estática (uma tarde, sem custo), borda de rede (dez minutos, já embutida em Vercel, Netlify e Cloudflare Pages), corte da biblioteca de animação (uma hora, sem custo) e, no topo, o único degrau pago, licenciar uma fonte comercial e servi-la do próprio domínio.

Peça assim: a landing no padrão das referências

Construa a landing do meu produto no padrão técnico das referências de alto padrão, arranjo de código próprio, sem clonar nenhum site específico. Aplique os cinco denominadores:

- Renderização: pré-render estático servido por borda de rede; nada de montagem integral no navegador. O conteúdo precisa estar visível com o JavaScript desligado.
- Tipografia: fontes servidas pelo próprio domínio, com font-display: swap, subconjunto de caracteres e preload apenas do peso do título. Um par de display e corpo; se eu não fixar a família, sugira duas com licença aberta e me diga a licença de cada.
- Cor: paleta em OKLCH como custom properties em :root; derive hover e bordas com color-mix(in oklch, ...). Um único acento de marca, neutro quente no fundo e texto em quase-preto.
- Tokens: nenhum valor de cor, espaçamento ou raio solto nos componentes; tudo por token nomeado.
- Movimento: contido, só hover com transform/opacity e revelação discreta ao entrar na viewport, tudo sob prefers-reduced-motion.
- Conteúdo: se ele for mudar toda semana, proponha um repositório desacoplado e explique em uma frase quando ele compensa contra manter o texto em arquivos Markdown no próprio repositório.

Metas: tempo até o conteúdo principal abaixo de 2,5 s, deslocamento de layout perto de zero, contraste no nível AA. Justifique cada decisão em uma frase.

O que cada escolha compra e o que ela cobra de você

Exercício desta aula, que leva uma hora.

Monte, com o agente, um laudo curto de dois sites de alto padrão que você admira, e use o resultado numa decisão sua.

Entregue: 1. Para cada site, um laudo preenchido pelo agente com o pedido de auditoria deste módulo, cobrindo framework, estilo, fontes, movimento, distribuição e conteúdo, com cada campo marcado como evidência, citando o sinal exato, ou como inferência. 2. A classificação de cada site numa das três famílias, justificada pela decisão de onde mora o conteúdo e por mais um sinal. 3. Para o seu próprio projeto, qual família você escolhe, ancorado na pergunta de quem vai mexer no site depois.

Critérios de sucesso: -- Cada campo do laudo cita o sinal que o sustenta, e o que não tem sinal aparece marcado como inferência. -- Você conferiu pelo menos três dos sinais com os próprios olhos no painel de rede, sem aceitar a palavra do agente. -- A família escolhida se justifica por quem vai mexer no site, e você consegue dizer qual dos cinco denominadores vai adotar primeiro. -- Nenhuma decisão sua se justifica pelo fato de um site conhecido usar aquilo.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 6. Benchmark stacks

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/benchmark-stacks.ts`

Ler a stack de um site grande sem chutar

Como quatro empresas de escala montam a interface, separando o que elas declararam do que se deduz, e o que dessa leitura cabe no seu projeto de uma pessoa. Traz a ficha completa das duas escolhas de framework que você tem hoje, a recomendação direta de stack para começar e o caderno de princípios que serve de preâmbulo em todo pedido ao agente.

Política de movimento reduzido do design system (modelo replicável)

A estrada pavimentada do Mercado Livre: design system, runtime com React e renderização no servidor, segurança embutida no caminho padrão e plataforma de publicação produzem uma experiência padrão entregue sem custo a centenas de equipes, que mantêm autonomia no domínio de negócio.

Peça assim

Apple: cada tipo de tela pede uma ferramenta:

DECLARADO (fonte pendente de conferência) por descrições de vaga oficiais: a engenharia de varejo e comunicação de marketing trabalha com uma pilha em React, enquanto a loja online é descrita como plataforma global de comércio eletrônico separada, operando em dezenas de países. Produto, marketing e finalização de compra não compartilham o mesmo ambiente de execução. No produto web, a engenharia reversa pública do serviço de música mostra uma aplicação de página única num framework diferente, composta por dezenas de componentes encapsulados, e as páginas de marketing são construídas com melhoria progressiva. A tecnologia se organiza pelo tipo de experiência, e o que atravessa todas é identidade, dado e contrato.

Vale listar o que a evidência disponível não sustenta sobre a Apple, por mais que circule: que o site inteiro roda uma base só, que a loja usa a mesma pilha do marketing, que existe um design system web único e global ou que as páginas formam uma aplicação de página única. Cada vaga comprova que uma equipe usa uma tecnologia, e para de provar aí. Confiança alta para a existência de plataformas separadas; confiança baixa para qualquer alegação de pilha única.

O que interessa a você é o mecanismo. Telas diferentes podem usar tecnologias diferentes e ainda compartilhar o mesmo componente de interface, porque a peça é encapsulada e nenhum estilo vaza para dentro nem para fora. É a forma mais durável de um design system: o mesmo sobrevive se o seu projeto trocar de framework no ano que vem. A Apple adotou Web Components nativos compilados com o Stencil, que gera Custom Elements com Shadow DOM a partir de TypeScript e JSX, importáveis em qualquer stack. O Shadow DOM é a subárvore privada onde o componente guarda o próprio CSS, e daí vem a resistência visual, porque cápsula não quebra quando a página ao redor muda.

A lição vira uma regra que você aplica hoje: escolha a ferramenta pelo tipo de tela. Página de marca pede conteúdo estático e velocidade; fluxo de cadastro e pagamento pede estado, validação e resistência a erro. Isso significa pedidos diferentes por tipo de tela, com o mesmo arquivo de tokens por baixo dos dois, que é o que impede a sua identidade de rachar entre as páginas.

A matriz comparativa estrutural: princípio dominante, renderização, design system e performance nas quatro empresas. A mesma palavra 'frontend' descreve quatro filosofias: padronização (ML), latência (Amazon), separação de jornadas (Apple) e interatividade (Duolingo).

Ficha do Next.js.

O que é: o framework que transforma React num site inteiro, com endereços, montagem no servidor e publicação prontos.

O que vem dentro: roteamento por pasta, em que cada subpasta de app vira uma URL; componentes de servidor, que rodam antes de chegar ao navegador e cortam JavaScript; um componente de imagem que gera versões por tamanho sozinho; um componente de fonte que baixa e serve a família do seu domínio; e cache com regras que você declara por página.

Quando usar: qualquer coisa com login, painel, carrinho, formulário de vários passos ou dado que muda a cada minuto. Quando não usar: um site de cinco páginas de texto, onde ele entrega mais máquina do que você precisa.

Passo a passo até a primeira tela no ar:

1. Rode npx create-next-app@latest e aceite TypeScript, Tailwind e a pasta app. Resultado esperado: uma pasta nova com tudo instalado.

2. Rode npm run dev e abra localhost:3000. Resultado esperado: a página inicial de exemplo aparece no navegador e recarrega sozinha quando você salva um arquivo.

3. Abra app/globals.css e escreva ali os seus tokens de cor, espaço e tipografia. Resultado esperado: um arquivo só com toda a identidade.

4. Crie a pasta app/precos com um arquivo page.tsx dentro. Resultado esperado: o endereço localhost:3000/precos passa a existir, sem você configurar rota nenhuma.

5. Rode npm run build e depois npm start. Resultado esperado: o terminal lista cada rota e diz se ela é estática ou dinâmica, e é essa lista que você confere antes de publicar.

6. Publique com npx vercel. Resultado esperado: uma URL pública servida por borda de rede, sem configuração.

O erro que todo iniciante comete: escrever "use client" no topo de todo arquivo para calar um aviso do editor. Essa linha manda o componente inteiro para o navegador e joga fora o principal ganho do Next.js. Coloque-a só no componente que de fato precisa de clique ou de estado.

O que copiar de cada empresa, e o que não copiar

Cinco lições costuram as leituras deste módulo, e a última engloba as quatro primeiras. A estrada vem antes das telas, com o arquivo de tokens pronto antes do primeiro componente. Velocidade se embute e se mede desde o começo, com esqueleto de carregamento e teto de peso por tela, porque acabamento é o primeiro item que você corta quando está com pressa de publicar. A ferramenta se escolhe pelo tipo de tela, tratando página de marca e fluxo de pagamento como problemas diferentes. Estado e animação entram onde resolvem alguma coisa. A quinta lição é a mais cara de ignorar: adotar tecnologia porque uma empresa grande usa entra pela porta do pedido, no momento em que você escreve ao agente que ele faça igual a um site conhecido. Escreva outra coisa. Descreva o seu problema, com tipo de tela, volume de dados, quem usa e o que a pessoa precisa fazer ali, e deixe a arquitetura sair disso.

Camada 2, uso real: entrega, estado, testes, medição e arquitetura

Camada 3, identidade: design system, estilo e movimento

Resta o mecanismo por trás dessa leitura. Toda interface publicada vaza sinais sobre como foi construída: o formato do endereço, os arquivos que o navegador baixa, os cabeçalhos que o servidor devolve, a estrutura da página montada. Quem sabe ler esses sinais reconstrói boa parte da arquitetura sem acesso ao código, e o capítulo seguinte ensina o passo a passo dessa leitura no painel de rede.

O limite é preciso. Os sinais provam o que aquela página faz naquele momento, sem provar que a empresa inteira faz assim, nem que fará amanhã, nem por que escolheu assim. A distância entre o que o sinal mostra e a conclusão que você tira dele é exatamente o espaço da inferência. Quando alguém apresentar a arquitetura de um gigante como argumento, pergunte se aquilo foi declarado por alguém ou deduzido a partir de algum sinal.

O que cada jeito de copiar uma referência compra e cobra

Faça você mesmo: as suas sete decisões numa página

Critério de uso
Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 7. Booking

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/booking.ts`

Booking: a fronteira entre persuasão que converte e passivo que se acumula

O que a maior loja de viagem do mundo faz na tela para acelerar a decisão de compra, e como reproduzir o que é honesto: o card de resultado, a busca com filtros, as fontes do sistema, o selo de escassez ancorado em dado real e o seu primeiro teste A/B sem plataforma nenhuma.

Anatomia do card de propriedade: cada parte com um papel único

A página de resultados expõe cerca de 24 categorias de filtro no desktop (econsultancy, 2023), e o número assusta até você notar que ninguém usa 24. A pessoa toca nos dois ou três filtros que importam para ela, e o resto fica disponível e silencioso. O valor está na combinação, com filtros independentes que recalculam o resultado na hora, cada um reduzindo o conjunto sem apagar o caminho de volta.

Quatro detalhes sustentam uma busca longa sem cansar quem busca. A coluna de filtros gruda no topo ao rolar, com `position: sticky`. O mapa alternável entrega leitura espacial no momento em que "perto da praia" passa a valer mais que qualquer estrela. A barra inferior no celular destaca "Salvos" e "Ofertas", os dois motivos pelos quais alguém volta ao app. E a busca por voz cobre a digitação difícil na rua (uxmag, 2023).

Antes de escrever a primeira linha desse conjunto, decida onde o estado dos filtros vai morar. A resposta certa é a URL, e a figura abaixo mostra as três coisas que você ganha de graça com essa escolha.

Com IA, escrever a variante leva segundos, então o trabalho de valor migrou para escolher a hipótese e definir como medir. Seis variantes prontas e nenhuma métrica declarada é muito trabalho e nenhuma decisão.

Decida a hipótese e a métrica antes da variante. "Mover o botão para cima aumenta o clique em reservar" é testável com métrica clara. "Deixar a página mais bonita" consome tráfego e não responde nada. Os seis passos da figura abaixo rodam o primeiro teste sem plataforma nenhuma, e dois deles concentram os erros de estreante: a atribuição da variante precisa ser determinística, por hash do identificador, e a leitura precisa aceitar que a maioria dos testes empata.

Os cinco gatilhos, a copy histórica e a única versão que você pode gerar hoje

A estratificação sistêmica: escassez, prova social, atividade ao vivo, aversão à perda e ancoragem disparam em sincronia na mesma tela e se reforçam. Auditar componente por componente nunca examina esse conjunto: a unidade de auditoria é a tela montada.

A linha passa pela veracidade do dado. Escassez HONESTA: o contador reflete o estoque real, e "resta 1" significa que resta 1. Escassez ENGANOSA: o estoque é apresentado como mais limitado do que de fato é, ou o relógio reinicia sozinho. Mesmo componente de UI, mesma animação, mesma cor. O que muda é a fonte de dados, e é só isso que um fiscal ou um cliente desconfiado vai conseguir conferir.

Faça você mesmo: auditar os selos de uma tela em 3 minutos

Os cinco prompts a seguir reconstroem os padrões centrais da Booking, com o card, a busca facetada, as system fonts, a escassez honesta e a experimentação, cada um com a acessibilidade já no enunciado. Quatro detalhes decidem se o resultado presta.

O badge de score expõe um aria-label completo ("Avaliação 8,4 de 10, Excelente"), porque o leitor de tela não infere a escala a partir de uma caixa colorida. O font stack pede a propriedade font-family isolada, com system-ui no topo, porque o shorthand `font:` reseta line-height, weight e variant para os valores iniciais e mata a herança em silêncio.

No selo de escassez, o atributo data-fonte no markup é a trilha de auditoria que prova a origem do número, e a regra que ele materializa vale sem exceção: se o número não vem da API, o selo não existe. No teste A/B, o hash determinístico resolve o erro mais comum de quem monta o experimento em casa, já que uma variante sorteada a cada render faz o botão piscar entre versões e transforma o seu dado em ruído; com um hash simples sobre o id (FNV-1a, por exemplo), a atribuição fica estável entre sessões.

Prompt: Busca search-first + navegação facetada

Prompt: Escassez honesta ancorada no dado real

Copiar da Booking vs Evitar

Engajamento de marketing ético no frontend cabe em três disciplinas. A primeira é medir com experimento em vez de chutar o que converte. A segunda é reter com um programa que entregue algo de fato, e não com uma saída difícil de encontrar. A terceira é sustentar cada aviso de escassez com o estoque que ele diz representar. A fronteira tem lastro legal: padrões escuros de urgência e opt-in pré-marcado caem sob DMA e DMCCA na Europa e no Reino Unido, e sob a atuação da FTC nos EUA contra dark patterns. O selo fabricado que parecia esperto chega ao balanço como passivo.

Faça você mesmo: página de resultados ética por construção

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 8. Cicd extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/cicd-extra.ts`

Rodar os testes na sua máquina é frágil por dois motivos que costumam aparecer no mesmo dia ruim. Você esquece de rodar, e o que passa no seu notebook quebra em outro, onde a versão do Node é diferente. A automação move a verificação para um lugar neutro: a cada proposta de alteração, um servidor limpo instala tudo do zero, roda a checagem de tipos, o lint, os testes e o build, e recusa a entrada quando algo falha.

Um portão só merece o nome quando bloqueia. Enquanto a verificação apenas relata, ela é um painel, e painel se ignora quando o prazo aperta. A figura a seguir mostra o ciclo inteiro, e as duas seguintes trazem a stack recomendada e a anatomia do arquivo que você vai escrever.

GitHub Actions: a esteira que roda a verificação sozinha.

O que é: um serviço que já vem junto do repositório no GitHub e executa comandos numa máquina limpa toda vez que algo acontece no projeto. Você o configura com um arquivo YAML dentro de `.github/workflows/`.

O que vem dentro: as ações prontas `actions/checkout` para baixar o código e `actions/setup-node` para instalar o Node com cache, mais os gatilhos `pull_request` e `push`, os segredos do repositório lidos por referência e a chave `concurrency` para cancelar execução obsoleta.

Quando usar: em qualquer projeto que more no GitHub. Quando não usar: se o repositório está no GitLab, use o GitLab CI, que faz a mesma coisa com sintaxe parecida.

Passo a passo:

1. Crie a pasta `.github/workflows` e o arquivo `ci.yml` dentro dela. 2. Escreva o gatilho `on: [pull_request]` e um job `build` rodando em `ubuntu-latest`. 3. Ponha os passos na ordem: `actions/checkout@v4`, `actions/setup-node@v4` com `cache: npm`, `npm ci`, `npm run verify`, `npm run build`. Resultado esperado: abra uma proposta e veja a aba Checks ficar verde. 4. Nas configurações do repositório, vá em Branches, crie uma regra para `main` e marque o check `build` como obrigatório. Resultado esperado: com o check vermelho, o botão de incorporar fica cinza. 5. Adicione `concurrency: { group: ${{ github.ref }}, cancel-in-progress: true }`. Resultado esperado: um segundo envio na mesma proposta cancela a execução anterior.

O erro que todo iniciante comete: escrever o arquivo e parar aí. Sem a proteção do ramo principal no passo 4, o fluxo roda, avisa e continua sendo ignorado. O infográfico da anatomia abaixo marca essa peça em destaque.

O infográfico a seguir repete os seis passos na ordem, para você conferir o arquivo que o agente devolveu. Trate o tempo total da esteira como requisito de mesmo peso, porque depois que você abre outra tarefa a revisão volta a acontecer sem contexto fresco, e é aí que defeito passa.

Segredos, e o que fazer quando um vaza:

Chave de integração, endereço de monitoramento e token de publicação são segredos, e o arquivo local que os contém nunca entra no repositório. Na esteira, eles ficam no cofre da plataforma e são lidos por referência. Variável marcada como pública vai para o pacote enviado ao navegador, então chave secreta jamais entra nela.

Segredo que já foi enviado ao repositório não se conserta apagando o arquivo: ele permanece no histórico, e histórico público é permanente. O único conserto real é trocar a chave no provedor e invalidar a exposta. Daí a preferência, na esteira, por credencial efêmera emitida a cada execução. O ecossistema de pacotes caminhou nessa direção: em dezembro de 2025 os tokens clássicos de longa duração foram revogados de forma permanente, o login passou a abrir sessão de duas horas, o token granular de escrita ganhou prazo máximo de 90 dias e a publicação por identidade confiável virou o caminho recomendado.

A máquina que roda a esteira virou alvo, e isso muda a conversa de acesso. Num levantamento da GitGuardian sobre a propagação de um verme na cadeia de pacotes (State of Secrets Sprawl 2026), 59% das máquinas comprometidas eram executores de integração contínua, e o mesmo relatório contabilizou 33.185 segredos únicos

expostos no conjunto. A esteira concentra credenciais de todos os ambientes, e por isso merece o cuidado de acesso do servidor de produção.

O mecanismo da reversão: o domínio volta ao artefato imutável anterior em segundos e o incidente é estancado, mas os dados que a versão nova gravou e o contrato de dado alterado não voltam; o conserto de raiz sai por proposta normal, pelos mesmos portões.

Gere o workflow de CI/CD do GitHub Actions para um frontend Vite + React 19 + TypeScript. Requisitos do job de CI (roda em pull_request e push na main): - checkout, setup-node com node-version: lts/* e cache de npm (não fixe Node 20: ele atingiu fim de vida em 30/04/2026, pelo calendário oficial de releases em https://github.com/nodejs/release, e CI em runtime sem suporte de segurança é passivo, não infraestrutura). - npm ci (instalação reprodutível pelo lockfile). - rode o script `verify` (tsc --noEmit && biome check && vitest run) e depois `npm run build`. - Lighthouse CI com budget de LCP < 2,5s, falhando o job se estourar. - use concurrency com cancel-in-progress para não pagar runs obsoletos. Requisitos do job de deploy (só na main, só se a CI passou; use `needs`): - deploy na Vercel lendo o token de ${{ secrets.VERCEL_TOKEN }}, nunca hardcoded. - envie os source maps ao Sentry usando ${{ secrets.SENTRY_AUTH_TOKEN }} e NÃO publique os source maps no site. Explique cada `env:` e me diga quais secrets devo cadastrar no GitHub e como faço rollback se o deploy quebrar.

Faça você mesmo: montar o portão e ensaiar a reversão em 20 minutos

Monte os cinco controles nesta ordem, do mais barato ao mais caro. A verificação local roda a cada alteração e custa um script no `package.json`. Na esteira, a mesma verificação protege o ramo principal. A pré-visualização por proposta deixa você conferir a tela no celular antes de publicar. A medição de campo mostra o que nenhum laboratório enxerga. E a reversão responde à pergunta que fecha o assunto: quanto tempo você leva para voltar à versão de ontem.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 9. Comparativo claude gemini

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/comparativo-claude-gemini.ts`

Gemini e Grok: o barato que cobra por fora

As duas ofertas mais baratas por token, lidas pelo que elas cobram além dele: o preço que dobra numa data já publicada, o envio em lote que corta a conta pela metade, a imagem que custa vinte vezes o texto no mesmo modelo e as buscas que o agente dispara sozinho. Traz a ficha completa do envio em lote, o passo a passo para medir o seu custo por tarefa e a lista de desligamentos com data. Todos os valores foram lidos em 15/08/2026.

A ficha do envio em lote em cinco quadros: o que é (o pedido entra numa fila e a resposta volta depois), o que vem dentro (arquivo com muitos pedidos, identificador de trabalho e arquivo de resposta), quando usar (classificar acervo, gerar texto em massa, reprocessar arquivo), quando não usar (qualquer coisa com você esperando) e o preço, que é metade do padrão nos dois períodos da tabela.

Preço, gatilho e o que fica fora da tabela de tokens (aferido em 15/08/2026)

A conta de um agente tem duas variáveis, e a tabela de preços mostra uma. A primeira é o preço por milhão de tokens. A segunda é quanto o modelo consome até concluir, medida em turnos e em tokens. As duas se multiplicam, e é por isso que um modelo mais caro por token pode sair mais barato por tarefa concluída.

A SpaceXAI lançou o Grok 4.6 em 12 de agosto de 2026 com foco declarado em agentes de longa duração. Nas medições independentes da Artificial Analysis, apuradas em 15 de agosto de 2026, ele pontua 61 no Intelligence Index da casa, atrás do Claude Opus 5 com 63 e do Claude Fable 5 com 62, marca Elo 1753 no GDPval-AA v2, 88,4% no Terminal-Bench v2.1 e 50,7% no τ³-Banking. O preço de US$ 2 e US$ 6 fica mais de 60% abaixo do Claude Opus 5, a US$ 5 e US$ 25, e do GPT-5.6 Sol, a US$ 5 e US$ 30.

O dado que muda a conta está no consumo. No AA-Briefcase, benchmark PRIVADO da mesma casa para trabalho longo de agente, o Grok 4.6 resolve tarefas em cerca de 53 turnos e cerca de 0,5 bilhão de tokens de entrada em média, contra cerca de 103 turnos e cerca de 2,0 bilhões de tokens de entrada do Claude Opus 5 no modo de esforço máximo. Três ressalvas viajam com esse número. A comparação vale contra um modo de esforço específico, e outro modo pode não repetir a mesma razão. O teste é privado, então você não consegue rodá-lo nem julgar quanto ele se parece com o seu trabalho. A medida é média, e a média silencia sobre a tarefa que der errado e consumir o triplo.

Faça você mesmo: o custo por tarefa concluída, em 20 minutos e sem escrever código

A medição independente conta uma história mais favorável ao ângulo de custo. A Artificial Analysis compra os tokens e roda os testes por conta própria: coloca o Gemini 3.7 Flash em esforço alto com índice 56, custo combinado de US$ 0,58 por milhão de tokens, 340,1 tokens de saída por segundo, 9,83 segundos até o primeiro token e 1,7 minuto de tempo médio por tarefa, com velocidade de saída perto de três vezes a do GPT-5.6 Terra e 60% em pass^5 contra 54% do Claude Opus 5 em esforço máximo. O índice 57 atribuído ao GPT-5.6 Terra e ao Muse Spark 1.2 está na tabela do Google, e a página da Artificial Analysis sequer traz esses dois modelos, então somar as duas réguas produz um ranking que nenhuma delas publica.

Duas conferências finais evitam erro bobo. O AutomationBench-AA da Artificial Analysis registra 62,7% para o Gemini 3.7 Flash e o AutomationBench de conjunto privado da tabela do Google registra 30,4% para o mesmo modelo, com nomes quase iguais e medições distintas. Ser forte em automação de terminal também é coisa diferente de ser forte em interface julgada por gente: na WebDev Arena, que compara respostas às cegas com julgamento humano, o grok-4.6-high aparece em quinto lugar com 1631 pontos na leitura de 15 de agosto de 2026, atrás de claude-opus-5-max com 1692, kimi-k3-max com 1674, qwen3.8-max com 1667 e claude-opus-5-high com

1663. Para escolher a ferramenta que constrói as suas telas, olhe a preferência humana e teste nas suas próprias páginas.

Quem mediu o quê, e o que cada linha permite concluir (15/08/2026)

As três linhas da fatura de um agente autônomo na apuração de 15/08/2026: tokens na faixa normal, tokens na faixa longa com reprecificação do pedido inteiro a partir de 200 mil tokens, e ferramentas cobradas por chamada fora do token. A terceira linha é a única sem teto natural, porque quem decide o número de chamadas é o agente.

Na SpaceXAI, quem assina do outro lado mudou. A SpaceX adquiriu a xAI em 2 de fevereiro de 2026 em transação integralmente em ações; em maio de 2026 veio o anúncio de que a xAI deixaria de existir como empresa separada; em julho de 2026 o nome passou a SpaceXAI. A empresa que aparece nos termos de uso hoje é outra. Vale o alerta sobre a fonte deste parágrafo: a cronologia veio de fonte secundária consolidada, boa para datar marcos e insuficiente para conclusão jurídica.

O risco existe dos dois lados, com origens diferentes. O acesso ao modelo mais capaz da Anthropic foi suspenso em todo o mundo em 12 de junho de 2026 por controle de exportação do governo dos Estados Unidos, os controles caíram em 30 de junho e o acesso voltou em 1º de julho de 2026. A defesa é a mesma nas duas origens: tenha um segundo modelo já configurado no seu projeto e teste a troca uma vez por mês, para descobrir os erros num dia calmo.

O calendário que decide o cronograma: desligamentos, cotas e ritmo de versão

Prompt pronto: a tabela de custo por tarefa que o agente monta para você

Para trabalho de volume que pode esperar, use Gemini 3.7 Flash em envio em lote: US$ 0,375 de entrada e US$ 1,875 de saída por milhão até 31/12/2026, metade do padrão, sem trocar nada de lugar. Para agente que trabalha por horas, comece pelo Grok 4.6 com duas travas ligadas desde o primeiro dia, um teto de chamadas de ferramenta e um alarme quando o pedido passar de 180 mil tokens. Para gerar imagem, orce à parte pelo valor por peça, US$ 0,0336 por imagem de 1K no lote, porque a saída em imagem custa vinte vezes a saída em texto no mesmo modelo. E se você for planejar 2027, use a coluna da direita da tabela do Google, que já está publicada.

As oito checagens que você faz sozinho, no navegador e no código

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 10. Comparativo claude openai

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/comparativo-claude-openai.ts`

Claude e OpenAI lado a lado: qual deles escreve a sua tela

Qual modelo você chama para cada tipo de tela, quanto isso custa por mês e como conferir sozinho um percentual de benchmark antes de acreditar nele. Traz a ficha completa da verificação visual automática do Claude Code, o par antes e depois do pedido sem direção de arte e um fluxograma de três perguntas para escolher o modelo. Todos os números foram lidos em 15 de agosto de 2026 e vencem em semanas.

Fluxograma de três perguntas com resposta sim ou não para decidir entre assinatura e pagamento por consumo: hora seguida de trabalho esbarra na janela de cinco horas e no teto semanal, uso diário do topo de linha esbarra no limite semanal próprio do Opus, e uso irregular pede API. Sem nenhuma dessas condições, comece no plano barato, onde a janela do Plus rende de 10 a 100 mensagens com o Sol e de 250 a 2.000 com o Luna.

Preço de tabela por 1 milhão de tokens, lido em 15/08/2026

Linha do tempo dos lançamentos e dos cortes de preço deste capítulo: o artigo de suporte da Anthropic sobre o limite semanal do Opus em 02/06/2026, a abertura da família GPT-5.6 em 09/07/2026, o lançamento do Claude Opus 5 em 24/07/2026, o corte de 20% no Terra e de 80% no Luna em 30/07/2026, a leitura de todos os números em 15/08/2026 e o aumento cancelado do Sonnet 5, que valeria em 01/09/2026.

O SWE-Bench Pro mostra o efeito contrário. Na tabela publicada pela própria OpenAI, o GPT-5.6 Sol resolve 64,6% dos casos contra 80,3% do Claude Mythos 5 e 80% do Claude Fable 5, uma diferença de mais de quinze pontos a favor do concorrente, divulgada pelo fornecedor que perde. Parar a leitura nessa linha inverte o sentido da mesma tabela: nela, o Sol ganha em DeepSWE v1.1, com 72,7% contra 69,7% do Fable 5, e em Terminal-Bench 2.1, com 88,8% contra 83,1%. Quem cita uma linha só está fazendo o mesmo recorte que critica.

O índice geral de inteligência da mesma avaliadora mostra a velocidade com que esses números envelhecem. Em 15 de agosto de 2026, o Claude Opus 5 marca 63 pontos e ocupa a posição 1 de 188 modelos avaliados; o GPT-5.6 Sol marca 61 pontos na posição 5 de 188. No lançamento, em 9 de julho de 2026, esse mesmo Sol marcava 59 pontos e ficava 1 ponto abaixo do Claude Fable 5, a cerca de um terço do custo. O custo por tarefa do Sol nesse índice foi medido em US$ 1,04 em 9 de julho e revisado para US$ 1,23 em 15 de agosto, uma alta de 18% em cinco semanas sem que o preço de tabela mudasse. Cinco semanas bastaram para mover posição, pontuação e custo do mesmo modelo na mesma avaliadora.

O que cada régua diz sobre os dois fornecedores

As duas arenas da figura são também as ferramentas mais úteis do seu dia a dia, e você usa as duas de graça pelo navegador. A Design Arena mostra duas telas geradas por modelos que você não sabe quais são, e pede que você escolha a melhor; cada voto entra no Elo. A WebDev Arena faz o mesmo com aplicações web pequenas. Use-as assim: pense na tela que você precisa construir, digite esse pedido na arena, veja as duas respostas lado a lado e repare em quais decisões visuais você preferiu. O resultado esperado é que você aprenda a nomear o que gosta, e é esse vocabulário que entra no seu próximo pedido.

Duas capacidades separam os dois fornecedores no trabalho de interface. A primeira é a verificação visual automática. A documentação oficial do Claude Code descreve o agente conferindo sozinho o que produziu, com a frase "takes screenshots, inspects the DOM, clicks elements, fills forms, and fixes issues it finds", e registra que a verificação vem ligada por padrão, com desligamento por projeto na chave autoVerify de um arquivo de configuração. Do lado da OpenAI, o anúncio de 9 de julho de 2026 afirma que o GPT-5.6 traz "uma mudança significativa em discernimento de design" e que as capacidades mais fortes de uso do computador permitem a ele inspecionar e refinar o resultado renderizado. A primeira frase descreve comportamento de produto, com nome de chave de configuração e padrão declarado; a segunda é alegação de marketing sem número associado. Nenhuma

avaliação independente aberta mede discernimento de design.

As quatro linhas de direção de arte que entram antes do pedido: tipografia com família nomeada por você, um acento de cor só aplicado em ação principal e destaque de estado, escala de espaçamento de 4, 8, 12, 16, 24, 32 e 48 pixels, e a lista de proibições, com sombra em cartão, borda acima de 12 pixels, emoji e componente de catálogo.

Peça assim: as quatro linhas de direção de arte

Dez checagens fecham o capítulo. Faça cada uma sozinho, no navegador, antes de fixar uma escolha de modelo para o seu projeto.

1. Qual a data de leitura de cada número que você anotou, e algum deles já mudou desde então?

2. Quem mediu cada percentual: o próprio fornecedor, o leaderboard oficial ou uma avaliadora independente?

3. Qual arnês e qual versão exata do benchmark produziram o número?

4. Existe submissão conferida no leaderboard oficial do teste citado, ou só a tabela do fornecedor?

5. A tabela que você está lendo mostra todas as linhas, incluindo aquelas em que o modelo perde?

6. Qual o preço na faixa de contexto longo, e o seu pedido típico chega perto desse tamanho?

7. Você abriu o painel de consumo depois do primeiro dia de uso e viu o gasto real por tarefa?

8. A verificação visual da tela está ligada na sua configuração, e você já abriu a tela no seu navegador para conferir o que o agente disse?

9. As quatro linhas de direção de arte estão no início do seu pedido, ou você está aceitando o centro estatístico da web?

10. Se você for de assinatura, qual o limite semanal do plano, e o que você faz no dia em que ele estourar?

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 11. Css extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/css-extra.ts`

Aqui você aprende a segunda camada do CSS, a que o Tailwind não entrega em classe utilitária: a ordem em que os seus estilos se aplicam, a derivação de cores, o alinhamento de grades aninhadas, a animação do que antes não animava e os campos que crescem sozinhos. A meta é uma só: cada valor visual da sua marca vive num lugar único e todo o resto sai dele por cálculo. Cada cor repetida à mão é uma tela que vai ficar fora do padrão na próxima troca de identidade.

O Tailwind v4 é um motor de CSS que gera as classes utilitárias que você usa no HTML, direto do arquivo de estilos, sem arquivo de configuração em JavaScript.

Dentro dele vêm o `@import "tailwindcss"`, que traz o motor inteiro; o bloco `@theme`, onde você declara os seus tokens de cor, espaço, fonte e raio; a diretiva `@utility`, para criar utilitária própria; e a variante `@variant`, para condições como tema escuro e passar o mouse.

Use em qualquer projeto React ou Next.js novo. Em página estática de uma tela só, o CSS puro já resolve e você evita um passo de build.

Passo a passo para colocar de pé num projeto Next.js:

1. Instale com `npm i tailwindcss @tailwindcss/postcss` e crie `postcss.config.mjs` com `export default { plugins: { "@tailwindcss/postcss": {} } }`. 2. No `src/app/globals.css`, escreva `@import "tailwindcss";` na primeira linha. Resultado esperado: uma classe como `text-3xl` já funciona na página. 3. Logo abaixo, abra o bloco de tema: `@theme { --color-marca: oklch(0.62 0.19 255); --radius-card: 12px; }`. Resultado esperado: as classes `bg-marca`, `text-marca` e `rounded-card` passam a existir sozinhas. 4. Use os tokens no JSX: . 5. Confira o resultado com as ferramentas do navegador. Resultado esperado: no CSS gerado, `bg-marca` aponta para `var(--color-marca)`, e trocar o valor no bloco de tema muda todos os botões de uma vez.

O erro que todo iniciante comete é procurar o `tailwind.config.js` da versão 3 e criar um. Na versão 4 os tokens moram no bloco `@theme` dentro do CSS, e o arquivo de configuração em JavaScript só existe para casos de compatibilidade.

A segunda regra de sistema é a derivação de cor. Em vez de cravar mais hexadecimal, que sai de sincronia na primeira troca de marca, misture: `color-mix(in oklch, var(--marca) 88%, white)` resolve o tom de passar o mouse, as bordas e os véus a partir de um valor de origem só. O `in oklch` mantém a transição suave, sem o cinza que a mistura em sRGB produz no meio do caminho.

A conta é fácil de ver. Um projeto com oito matizes e quatro estados por matiz precisa de 32 valores escritos e mantidos à mão, ou de oito valores e uma função. Quem mantém a lista curta troca a identidade visual num commit.

Um cuidado que quase ninguém toma: tom derivado por cálculo pode cair abaixo do contraste mínimo sem você ter escolhido aquela cor. Meça cada variação gerada como se fosse cor nova, com o painel de contraste das ferramentas do navegador.

O tema claro e escuro segue a mesma lógica de valor único. Quando bastar seguir a preferência do sistema, declare `color-scheme: light dark` no `:root` e passe os dois valores numa linha com `light-dark(#fafafa, #101014)`: o bloco `[data-theme="dark"]` inteiro desaparece do arquivo. Use `light-dark()` quando o site só obedecer ao sistema operacional, e `[data-theme]` quando existir um botão de tema na tela, porque a função nativa não tem como saber que a pessoa clicou. Mantenha um valor comum na linha anterior como plano B, porque a função ainda não está em toda base instalada.

Combine os três: gere a paleta em OKLCH, derive os estados com `color-mix(in oklch, ...)` e resolva claro e escuro com `light-dark()`. O resultado é um arquivo de tokens minúsculo, com uma cor base por matiz e todo o resto derivado, que o agente entende e estende sem inventar hexadecimal novo. Ao pedir ao Claude Code, escreva a frase inteira: "derive hover e bordas com color-mix a partir dos tokens, sem cravar hex novos".

Quatro trocas de improviso por propriedade: o subgrid substitui altura fixa e medição por JavaScript, o @property faz a variável registrada interpolar em vez de saltar, o field-sizing: content aposenta o ajuste por scrollHeight, e o scroll-snap devolve o carrossel a um scroll de verdade.

Peça assim

Aplique a baseline CSS de 2026 nesta tela, sempre como progressive enhancement: - Estilize o estado aberto de dialog/details com :open (sem checar atributo nem classe). - Derive a cor de texto com contrast-color() embrulhado em @supports, mantendo o token AA como fallback. - Onde houver clip-path/offset-path complexo, use shape() em vez da string do path(). Embrulhe cada recurso em @supports, mantenha fallback e respeite prefers-reduced-motion.

A anatomia visual do alto padrão, medida em 26 referências.

Uma auditoria de 26 sites reconhecidos por design mediu o layout real de cada um, lendo o CSS publicado em vez de olhar a captura de tela. O que sobrou é uma lista curta de convergências: valores que a maioria desses sites de fato usa, e que costumam ser os mesmos que um projeto novo acaba escolhendo por tentativa e erro ao longo de meses. Transformados em token, eles entregam o ritmo visual do premium sem copiar tela nenhuma, porque o que se copia aqui é a medida. Uma ressalva de método vem antes dos números: o conjunto foi curado por um diretório de referências, e não sorteado, então ele descreve o que essa vitrine faz, sem autoridade para descrever o que a web faz.

Os primeiros valores convergentes são os de largura, de base de espaçamento e de ritmo vertical:

Tabela do módulo

Token	Valor convergente	Evidência no corpus
Largura máxima de conteúdo	1200px	25 dos 26; exceções em 1280 e 1400
Unidade-base do espaçamento	4px	cerca de 17 sites; segundo grupo em 8px
Gap entre seções	80px	o mais convergente (14 sites)
Padding de card	24px	a moda; 32px e 40px nos espaçosos
Gap entre elementos	16px	a moda; 8px nos densos, 24px nos amplos
Coluna de leitura longa	480 a 560px	medida editorial dentro do container

A anatomia de card mais recorrente é `padding: 24px` com `gap: 16px` entre os filhos, e a repetição desse par em quase todo o conjunto autoriza tratá-lo como ponto de partida. Um segundo detalhe se repete e passa despercebido: a navegação costuma ser mais larga que o miolo do conteúdo, o que dá ao topo da página um ar de moldura e mantém a coluna de leitura estreita o bastante para o olho não perder a linha ao voltar.

O raio de canto é assinatura de marca, e os sites do conjunto se dividem em três regimes. O primeiro é a pílula, de 99 a 9999 pixels, que devolve a silhueta da marca em botões e tags; um dos sites usa só 8 ou 100, sem nada entre os dois, porque valor intermediário ali diluiria a assinatura. O segundo é o médio, de 8 a 40 pixels, o visual de aplicativo moderno, com casos de 12 e de 26 pixels aplicados uniformemente em tudo. O terceiro é o reto, de 0 a 4 pixels, que persegue a estética de documento impresso, com um site em 0 em tudo e outro proibindo qualquer valor acima de 12. A assinatura mais elaborada nasce do contraste entre dois regimes: pílula em tudo, exceto os campos de formulário, que ficam em 0, e é esse contraste que faz a pílula ser notada.

Monte meus tokens de layout e cor no padrão técnico dos sites de alto padrão, num único arquivo de CSS, mantendo a base de cor em OKLCH: - Container: --container-max: 1200px, centralizado sobre canvas full-bleed. - Espaçamento em base 4 (4/8/12/16/24/32) e --section-gap: 80px; padding de card 24px com gap interno de 16px; coluna de leitura de texto longo limitada a ~560px. - Raio: escolha UM regime entre pílula (assinatura de marca), médio 8-16px (app) e reto 0-4px (editorial), justifique a escolha e aplique consistente; se usar contraste de raio,

documente a exceção (ex.: inputs em 0). - Cor: um único acento de marca, usado SÓ em CTA/ações (nunca decorativo); canvas neutro quente (near-white, jamais #fff puro); texto em quase-preto (jamais #000); par semântico sucesso/erro com AA. - Elevação: sem box-shadow pesada, com profundidade por empilhamento de superfícies e hairline de 1px; se houver sombra, tingida da marca e com 1-5% de opacidade. Regra: nenhum valor solto nos componentes; tudo de token nomeado. Mantenha contraste AA nos dois temas.

Critério de uso
Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 12. Diagramas canvas

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/diagramas-canvas.ts`

Explicar arquitetura, fluxo e rede numa figura que ninguém redesenha à mão

Três famílias de figura resolvem problemas diferentes: a que você descreve em texto e o build desenha, a que o usuário do seu produto edita arrastando peças e a que representa uma rede de milhares de conexões. Você aprende a reconhecer qual delas o seu caso pede, instalar a biblioteca certa passo a passo e deixar o texto da figura legível nos dois temas.

Árvore de decisão. A pergunta que separa as famílias: o diagrama é fixo e versionado (diagram-as-code), o usuário edita os nós (canvas node-based) ou a informação é a própria rede com muitas conexões (grafos). Cada família tem seu formato de entrega.

Mermaid, a ficha completa. O que é: uma biblioteca que lê um texto de poucas linhas e devolve o desenho pronto, calculando sozinha onde cada caixa fica. O que vem dentro: fluxograma, diagrama de sequência, cronograma de Gantt, diagrama de entidade e relacionamento, mapa mental e diagrama de estados. Quando usar: fluxo simples, sequência e cronograma, até algo perto de uma dúzia de caixas. Quando não usar: arquitetura com muitas camadas dentro de camadas, porque o arranjo automático embaralha.

Guia passo a passo, do zero à figura na sua página.

1. No terminal, dentro do projeto, rode `npm install mermaid`. 2. Crie um arquivo `arquitetura.mmd` com quatro linhas descrevendo o desenho. Comece por `flowchart TD` e escreva uma ligação por linha. 3. Instale o compilador de linha de comando com `npm install -D @mermaid-js/mermaid-cli`. 4. Gere a imagem com `npx mmdc -i arquitetura.mmd -o public/arquitetura.svg`. 5. Abra o arquivo gerado e confira se cada tem cor própria declarada. Onde faltar, aplique o tema do Mermaid com as cores do seu projeto. 6. Coloque o comando do passo 4 dentro do script de build no `package.json`, para a figura se refazer sozinha a cada publicação.

Resultado esperado: um arquivo SVG na pasta pública que ninguém precisa redesenhar, e que o navegador de quem lê exibe sem carregar biblioteca alguma. O erro que todo iniciante comete é embutir o Mermaid na página e deixá-lo desenhar no navegador do visitante, o que carrega centenas de quilobytes de código para produzir uma imagem que já poderia estar pronta.

D2, quando o desenho cresce. O que é: a mesma ideia do Mermaid, com uma diferença que decide a escolha. O D2 trata agrupamento como conceito de primeira classe, então você escreve `servidor: { api; cache }` e ele desenha a caixa em volta. O que vem dentro: contêineres aninhados, vários motores de arranjo, temas prontos e exportação para SVG, PNG e PDF. Quando usar: arquitetura com camadas, do tipo cliente, servidor, dados e integrações externas. Quando não usar: um fluxo de cinco caixas, onde o aprendizado da sintaxe nova não se paga.

Recomendação direta: para até uma dúzia de elementos, use Mermaid; acima disso, ou quando existirem camadas dentro de camadas, troque por D2. O motivo cabe numa linha: o D2 desenha o agrupamento e o Mermaid espalha as caixas.

Prompt: diagrama de arquitetura em D2 compilado no build

Descreva a arquitetura deste repositório e gere o diagrama em D2 com contêineres por camada (cliente, servidor, dados, integrações externas).

Requisitos: - Compile o D2 para SVG no passo de build (não no navegador do leitor). - Injete os tokens de cor do projeto (var(--accent), var(--text), var(--border)) no SVG gerado; nada de cores hardcoded. - Todo com fill EXPLÍCITO e font-size >= 14px. - Use layout ELK; agrupe por camada com contêineres aninhados.

Critérios de aceite: 1. O SVG final não referencia nenhuma lib em runtime. 2. Rodei o auditor de contraste (Chrome MCP) e todo texto passa AA. 3. Se o sistema tiver mais de 12 nós, você justificou D2 sobre Mermaid. 4. As setas têm rótulo quando o tipo de chamada não é óbvio.

A anatomia de um editor React Flow: o contêiner com altura declarada, a lista de nós, a lista de arestas, os pontos de encaixe, o minimapa e o componente de nó que você mesmo escreve. Sem altura no contêiner, a tela nasce com zero pixel.

Prompt: editor de fluxo com React Flow e nós customizados

Monte um editor React Flow com TRÊS tipos de nó customizado: gatilho, condição e ação. Cada tipo é um componente React próprio estilizado com os tokens do projeto (var(--accent) para gatilho, var(--success) para ação, var(--border) nas bordas; texto var(--text)).

Requisitos: - Minimap, snap ao soltar e navegação por teclado ativos. - Persista o grafo (nós + arestas) em JSON no localStorage e recarregue no mount. - Handles de conexão visíveis e com alvo de toque >= 24px (WCAG 2.2).

Critérios de aceite: 1. Recarreguei a página e o fluxo voltou idêntico (persistência ok). 2. Consigo criar, conectar e apagar nós só pelo teclado. 3. Os três tipos de nó são visualmente distintos e usam os tokens. 4. Contraste do rótulo de cada nó passa AA (auditor no Chrome MCP).

Cada caminho, o que ele te dá e o que ele te cobra

Feche o módulo representando o seu próprio projeto nas três frentes, com o auditor de contraste rodado nos dois temas no fim.

Faça você mesmo: as três famílias no seu projeto, em 20 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 13. Ds 2026 extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/ds-2026-extra.ts`

O mesmo arquivo de tokens vale para design e código: a ferramenta de design exporta em formato aberto; Terrazzo ou Style Dictionary o transformam em variáveis CSS por tema, que alimentam o bloco de tema do Tailwind.

Três coisas se moveram desde a redação original deste módulo, e vale atualizar a sua stack.

A primeira é o formato do arquivo de tokens. Existe hoje uma convenção aberta, o DTCG, que descreve cor, espaço, tipografia, apelidos e temas num arquivo neutro, com suporte a espaços de cor modernos, e que as ferramentas de design e de pipeline já leem. O ganho para você é o custo de saída: a sua marca passa a viver num arquivo que não pertence a nenhum fornecedor, então trocar de ferramenta de design mais adiante deixa de significar reconstruir a paleta. No prompt, nomeie a convenção em vez de deixar o agente inventar formato próprio.

O pipeline que transforma esse arquivo em variáveis de CSS ganhou alternativa mais enxuta. O Terrazzo consome o arquivo de tokens e emite variáveis de estilo, tipos e folha pré-processada, com configuração menor que a do Style Dictionary. A diferença entre os dois é de tamanho de configuração, e não de capacidade, o que torna a comparação barata: peça as duas ao agente e olhe o resultado. Do lado do desenho, o Penpot é a alternativa de código aberto ao modo de desenvolvedor do Figma e também expõe os tokens na convenção aberta, então o mesmo arquivo viaja de uma ferramenta à outra.

Há um ganho estético embutido nessa disciplina. O tema padrão que acompanha o gerador de componentes é reconhecível de longe, e produto que parece template perde percepção de preço na primeira tela. Abra um editor visual de tema com cor em espaço perceptual, gere o bloco de variáveis e peça ao agente que o integre ao tema do projeto medindo o contraste dos pares de texto sobre fundo nos dois temas. A identidade aparece sem sair da disciplina de token, e sobrevive à próxima atualização do gerador.

A árvore atualizada: projeto React novo recebe a Base UI, que é a base padrão do gerador; acessibilidade contratual, vários idiomas ou leitura da direita para a esquerda pedem React Aria; Svelte ou Vue pedem Ark UI; base que já roda em Radix segue nele, disponível por opção explícita.

Quando você depende de uma biblioteca de terceiro, a pergunta que separa decisão de palpite é objetiva: com que frequência esse projeto publica correção. A Base UI chegou à versão estável em 11 de dezembro de 2025 com 35 componentes e manteve cadência mensal, chegando à linha 1.6 em junho de 2026. O Radix publicou três vezes entre junho e julho de 2026. Na outra ponta, o Headless UI tem como última versão estável a 2.2.10, de 7 de abril de 2026, com apenas versões de canal interno depois disso, o que dá um intervalo de cerca de três meses e meio sem liberação estável. Nenhum desses números condena uma biblioteca sozinho. Todos eles são o que transforma a escolha da base em decisão que você consegue explicar.

A cadência de publicação das três bases, de dezembro de 2025 a julho de 2026: Base UI estável em 11/12/2025 com 35 componentes e cadência mensal até a linha 1.6, Radix com três publicações entre junho e julho, e Headless UI parado na 2.2.10 de 07/04/2026.

Custo e risco por decisão de base de componentes

Vou converter e distribuir nosso design system. Faça em quatro entregas:

1. Converta os tokens atuais do projeto para o formato DTCG em tokens/tokens.json ($value/$type), com cor em OKLCH, aliases semânticos (accent, danger, success) e dois temas (light e dark). 2. Configure o Terrazzo para emitir custom properties escopadas por [data-theme="light"] e [data-theme="dark"] a partir desse tokens.json. NÃO edite os arquivos gerados; trate a pasta de saída como build. 3. Ligue o bloco de tema do Tailwind às custom properties geradas, sem nenhum hex solto no código. 4. Crie um registry.json na raiz expondo nosso KPI card e as convenções de projeto como itens instaláveis pela CLI do shadcn.

Critérios de aceite: o contraste de texto sobre fundo passa em WCAG 2.2 AA nos DOIS temas (me mostre os pares medidos); nenhuma cor hexadecimal aparece fora do tokens.json; e a instalação do nosso KPI card num projeto limpo termina sem erro.

A nota que muda a conversa sobre acessibilidade.

O alvo de conformidade que hoje tem valor jurídico é a norma de acessibilidade na versão 2.2, nível AA. Na Europa, a diretiva de acessibilidade de produtos e serviços se aplica desde 28 de junho de 2025 a comércio eletrônico e serviços bancários ao consumidor, sem multa em nível europeu e com sanções definidas por cada país. No Brasil, o instrumento chegou por via judicial, com prazo e multa diária fixados em julho de 2026. Nenhuma dessas normas referencia a versão 3.0, que segue como rascunho de trabalho.

Escreva esse limite na memória do projeto, no `CLAUDE.md`: o alvo é a norma 2.2 no nível AA, e o agente recusa sugestões de mirar a versão 3.0 ou de otimizar contraste por algoritmo alternativo. A versão 3.0 segue como rascunho, com o próprio documento afirmando que ainda tem vários anos de trabalho pela frente e que é inadequado citá-lo como algo além de trabalho em progresso; o algoritmo alternativo de contraste foi retirado do rascunho em 2023 e o método segue indefinido. Sem esse limite escrito, o modelo mira o padrão de que mais se falou na internet, e você entrega contraste que reprova na auditoria que vale.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 14. Ds extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/ds-extra.ts`

UI/Botao

Uma página de referência com os componentes desenhados à mão resolve um produto. Ela empaca quando aparecem o segundo produto, o segundo aplicativo e a segunda plataforma: os estados passam a ser desenhados a olho, o componente só existe dentro da tela em que nasceu, e nada é conferido automaticamente. Quatro ferramentas mudam esse patamar, e você opera as quatro com o agente desde que o pedido nomeie a saída esperada.

A ordem de adoção é esta: Storybook para desenvolver a peça isolada, Style Dictionary para gerar os tokens por plataforma, tailwind-variants para descrever as variantes e Changesets para versionar e publicar. Suba um degrau de cada vez, e só suba quando a dor do degrau anterior aparecer.

O pipeline que escala: uma fonte única de tokens vira CSS, JS, iOS e Android via Style Dictionary; os componentes nascem isolados no Storybook (testados com Chromatic) e saem versionados via Changesets.

Anatomia de um arquivo de stories: a exportação padrão nomeia o componente, cada exportação nomeada é um estado, a função play clica sozinha e o parâmetro de acessibilidade transforma violação em teste reprovado.

O Style Dictionary é uma ferramenta que lê os seus tokens num arquivo só e gera as versões para cada plataforma: CSS para a web, TypeScript para o código, Swift para iPhone e XML para Android.

Dentro dele vêm o formato de token em JSON, as conversões de valor e de nome por plataforma, os formatadores que montam cada arquivo de saída e a configuração que amarra tudo.

Use quando a mesma marca precisar aparecer em mais de uma plataforma, ou quando você quiser trocar a identidade visual num arquivo só. Num site que existe apenas na web, o bloco `@theme` do Tailwind já cumpre esse papel.

Passo a passo:

1. Instale com `npm i -D style-dictionary`. 2. Crie `tokens/color.json` no formato aberto: `{ "cor": { "acento": { "$value": "#0070ca", "$type": "color" } } }`. 3. Crie `style-dictionary.config.js` apontando a origem para `tokens/**/*.json` e declarando as plataformas `css`, `js`, `ios-swift` e `android`. 4. Rode `npx style-dictionary build`. Resultado esperado: a pasta `build/` aparece com um arquivo por plataforma, e o de CSS traz `--cor-acento: #0070ca;`. 5. Importe só o arquivo gerado de CSS no seu `globals.css` e apague os valores hexadecimais soltos dos componentes. Resultado esperado: trocar o valor em `tokens/color.json` e rodar o build muda a cor em todo o produto. 6. Acrescente `build/` ao `.gitignore` ou regenere o conteúdo na esteira.

O erro que todo iniciante comete é editar o arquivo gerado porque era mais rápido. Isso recria exatamente a divergência entre plataformas que a ferramenta existe para eliminar. Escreva no `CLAUDE.md` do projeto que a pasta de saída é resultado de compilação, senão o agente vai corrigir o arquivo errado com a melhor das intenções.

Custo e risco por degrau de maturidade do design system

O mecanismo comum às quatro peças: reduzir o número de lugares onde a mesma decisão visual está escrita: muitas telas viram uma peça isolada, trezentos arquivos viram um token, a cópia por tela vira variante declarada e o recado vira número de versão.

Estou escalando nosso design system. Para o componente Field (label + input + mensagem de erro):

1. Crie a fonte de tokens em tokens/color.json no formato DTCG ($value/$type), com accent e danger, e um style-dictionary.config.ts que gere saídas para CSS, JS, iOS (Swift) e Android (XML). NÃO edite os arquivos gerados em build/. 2. Implemente src/components/ui/field.tsx com tailwind-variants (tv): use slots (base, label, input, erro) e

uma variante `invalido` que altere AO MESMO TEMPO o input e o label. Consuma os tokens via var(--...), zero hex solto. 3. Gere field.stories.tsx com uma story por estado (normal, com erro, desabilitado), uma play function que digite no campo e verifique aria-invalid, e parameters.a11y.test = "error" para que violações de acessibilidade quebrem o teste. 4. Adicione um changeset (minor) descrevendo a adição do Field.

Critérios: contraste AA; foco de teclado visível via focus-visible; o painel de acessibilidade do Storybook deve passar sem violações.

O mapa das ferramentas por dor de escala:

Cada ferramenta ataca um ponto específico onde o design system caseiro racha ao crescer. A tabela abaixo é critério de escolha, e não lista de compras.

Tabela do módulo

Ferramenta	Papel no design system	O que ela substitui
Storybook	Desenvolver a peça isolada, com cada estado testável	Página de referência com estados à mão
Verificador de acessibilidade	Falhar a integração na violação, e não na auditoria	Conferência manual que ninguém repete
Chromatic	Comparar a aparência aprovada com a atual	Olhar a tela e achar que está igual
Style Dictionary	Gerar as saídas por plataforma de um arquivo só	Valor de cor repetido em três lugares
tailwind-variants	Tratar componente de várias partes numa definição	Um objeto por parte, escrito à mão
Changesets	Versionar, registrar e distribuir para outros times	Avisar no canal de mensagens do time

O ponto de virada é fácil de reconhecer. Enquanto a biblioteca serve um produto, a página de referência com tokens em arquivo basta. Quando ela passa a servir mais de um produto ou mais de uma plataforma, o degrau seguinte compra previsibilidade de prazo, e é aí que a próxima troca de identidade visual volta a caber numa semana em vez de ocupar um trimestre.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 15. Escolher modelo governanca

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/escolher-modelo-governanca.ts`

Quanto custa o seu mês de vibecoding

Como medir o que você gasta de verdade, que é o custo de terminar uma tarefa, e não o preço por token. Traz o botão que corta a sua conta quase pela metade sem trocar de modelo, o passo a passo para medir na sua própria carga e a lista de datas em que um modelo que você usa deixa de existir. Todos os números carregam a data em que foram lidos.

Quatro linhas correm fora da tarifa e precisam entrar na sua conta. A faixa longa de contexto: na documentação de preços da OpenAI, apurada em 15/08/2026, o GPT-5.6 Sol passa de US$ 5,00 para US$ 10,00 por milhão de tokens de entrada e de US$ 30,00 para US$ 45,00 na saída quando o pedido entra na faixa longa. Na documentação da SpaceXAI, pedidos acima de 200 mil tokens reprecificam a requisição inteira, e o Grok 4.6 vai de US$ 2,00 e US$ 6,00 para US$ 4,00 e US$ 12,00. As ferramentas por chamada: busca na web, busca no X e execução de código custam US$ 5,00 por mil chamadas cada, e a busca em coleções custa US$ 2,50 por mil, cobradas fora do token, com o agente decidindo sozinho quantas chamadas fazer. O tempo de sessão: agentes gerenciados da Anthropic cobram US$ 0,08 por hora de sessão em execução, e o exemplo oficial de uma sessão de uma hora no Opus 5 fecha em US$ 0,705 somando entrada, saída e tempo. E o custo de simplesmente avaliar: a Artificial Analysis gastou US$ 3.836,05 para rodar o Claude Opus 5 em esforço máximo no Intelligence Index e US$ 2.823,25 para rodar o GPT-5.6 Sol em configuração máxima no mesmo índice. Comparar modelos a sério custa dinheiro, e é por isso que você vai medir só no que faz todo dia.

Medir o seu custo por tarefa leva uma tarde e não exige código. Faça assim:

1. Liste as cinco tarefas que você mais repete com o agente. Cada linha precisa de um verbo e de um entregável, do tipo "gerar uma página de listagem com filtros" ou "converter três telas para tema escuro". Resultado esperado: uma lista curta e concreta.

2. Marque cada tarefa como síncrona, quando você fica esperando na tela, ou assíncrona, quando pode buscar depois. Resultado esperado: as assíncronas viram candidatas a envio em lote, que sai por metade do preço no Google.

3. Rode cada tarefa dez vezes, alternando os dois modelos candidatos. Resultado esperado: vinte execuções registradas, e não uma impressão.

4. Anote, para cada execução, o custo mostrado no painel de consumo, os minutos até terminar e quantas vezes você precisou pedir correção. Resultado esperado: três colunas por linha.

5. Some à parte as chamadas de ferramenta e as imagens geradas. Resultado esperado: essas duas linhas ficam separadas da linha de tokens, porque elas crescem por conta própria.

6. Escreva o pior caso ao lado da média. Resultado esperado: você enxerga a tarefa que consome o triplo, que é justamente a que estoura o mês.

O erro que todo iniciante comete: comparar pela média de três execuções. Três é pouco para uma distribuição com cauda longa, e a média de três esconde exatamente o caso caro que você precisa conhecer.

Deixe o esforço alto como padrão do seu dia e reserve o máximo para a tarefa que você não conseguiu resolver de outro jeito. Na medição independente de 15/08/2026, essa única mudança leva o custo por tarefa de US$ 17,79 para US$ 10,41 no mesmo Claude Opus 5, com 25,7 minutos em vez de 36,2. Ligue o máximo quando a tarefa envolver decisão de arquitetura ou migração grande, e desligue de volta assim que terminar. O ajuste mora nas configurações do agente, e é o botão mais barato de mexer no curso inteiro.

O ajuste de esforço, medido dentro do mesmo modelo

A promessa de janela grande é medida em janela pequena. A tabela oficial do Google publica o resultado de recuperação em contexto longo (GDM-MRCR v2, oito agulhas) como média em 128 mil tokens, com 97,0% para o Gemini 3.7 Flash, 93,5% para o GPT-5.6 Terra, 91,8% para o Gemini 3.6 Flash e 81,5% para o Claude Sonnet 5, ainda que a janela anunciada seja de 1 milhão de tokens. Se você pretende colar um projeto inteiro no pedido, teste com o seu volume real antes de confiar.

O prazo de validade do número fecha o argumento. Em 09/07/2026 a Artificial Analysis publicou o GPT-5.6 Sol em 59 pontos e segundo lugar no Intelligence Index; em 15/08/2026 a página da mesma avaliadora marca 61 pontos e a posição 5 entre 188 modelos. O custo por tarefa do índice, na mesma fonte, passou de US$ 1,04 em 09/07/2026 para US$ 1,23 em 15/08/2026, alta de 18% em cinco semanas. Anote sempre a data ao lado do número que você guardar.

A mesma prova, três réguas: Terminal-Bench lido por quem mede

Desligamento com data. A OpenAI anunciou em 31/07/2026 que o GPT-5.4 e o GPT-5.4 mini deixam de estar disponíveis no Codex para quem entra com conta ChatGPT em 31 de agosto de 2026, com substitutos nomeados e com instrução expressa de revisar padrões de espaço de trabalho, agentes personalizados e tarefas agendadas antes do corte. O Google desliga o Imagen 4 em 17 de agosto de 2026, com substituto indicado. Se você tem um script que chama um modelo pelo nome, essa data é a sua.

Quem assina do outro lado. A xAI passou a subsidiária integral da SpaceX em 2 de fevereiro de 2026, o fim como empresa separada foi anunciado em maio de 2026 e o novo nome, SpaceXAI, veio em julho de 2026. A cronologia está consolidada em fonte secundária, o que basta para você datar os marcos e não basta para conclusão jurídica.

O preço fecha a lista, porque ele se move rápido. A OpenAI cortou o preço do Luna em 80% e o do Terra em 20% vinte e um dias depois do lançamento, em 30/07/2026, mantendo o Sol inalterado. O Gemini 3.7 Flash tem preço introdutório de US$ 0,75 na entrada e US$ 3,75 na saída por milhão de tokens até 31/12/2026, e passa a US$ 1,50 e US$ 7,50 em 1º de janeiro de 2027. E o aumento do Claude Sonnet 5 previsto para 1º de setembro de 2026 foi cancelado, com os US$ 2 e US$ 10 virando preço padrão.

As datas que mexem no seu projeto, entre junho de 2026 e janeiro de 2027 (apuração de 15/08/2026)

A tabelinha de roteamento que você guarda no próprio projeto

Quatro hábitos resolvem o seu mês. Primeiro, meça o custo por tarefa concluída na sua própria carga antes de acreditar em qualquer comparação de preço por token. Segundo, deixe o esforço alto como padrão e ligue o máximo só na tarefa difícil, porque a diferença entre as duas configurações do mesmo modelo chega a 71% do custo por tarefa na medição independente de 15/08/2026. Terceiro, mantenha um segundo modelo configurado e teste a troca uma vez por mês, porque dezenove dias de indisponibilidade já aconteceram em junho de 2026. Quarto, abra o painel de consumo toda sexta-feira nas primeiras semanas, até você conseguir prever a fatura sem olhar.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 16. Forms

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/forms.ts`

Formulário e origem do dado: onde a receita vaza e por que a tela mente

Você vai construir um formulário que não perde cliente e uma tela que não mente sobre o número que exibe. O caminho é o do dado: o que você pede no cadastro, onde esse dado é conferido, de onde vem o valor que a tela mostra depois e o que aparece quando a fonte falha. A stack é Zod, react-hook-form e TanStack Query, com guia passo a passo de cada uma.

Para formulário em React em 2026, use Zod para descrever os dados, react-hook-form para controlar os campos e @hookform/resolvers para ligar os dois. Se o peso do pacote enviado ao celular for restrição dura, troque o Zod pelo Valibot, que faz o mesmo com menos bytes. Se o formulário for grande e cheio de campos que dependem uns dos outros, troque o react-hook-form pelo TanStack Form. Nos outros casos, fique no par padrão, que tem mais exemplos publicados e por isso rende respostas melhores quando você pede código ao agente.

Anatomia de um campo acessível, com cada atributo apontado: `htmlFor` liga o rótulo ao campo, `aria-invalid` marca o erro, `aria-describedby` amarra a mensagem e `role="alert"` faz o leitor de tela anunciar assim que o texto aparece.

Faça você mesmo: prove que o erro do seu formulário é acessível (10 minutos)

Qual biblioteca de formulário escolher

Um pedido por seção. Quando vários botões disputam o mesmo peso visual, quem olha não sabe qual é o próximo passo e hesita, e a hesitação aparece na taxa de conclusão. No formulário, a tradução é literal: o botão que envia é o único elemento com peso máximo, e o link de ajuda, a opção de cancelar e o aviso de política ficam em hierarquia visual mais baixa, sempre alcançáveis pelo teclado. Repare na tela de pagamento da Booking: o botão de reservar é o único preenchido em cor sólida, e todo o resto da coluna é texto ou link.

A mecânica do pedido se copia entre setores. As galerias de referência de chamada de ação classificam os exemplos por mecânica, como botão, formulário, janela sobreposta e página de preço, e não por indústria. Isso significa que a solução de um banco serve a uma escola sem tradução.

Os quatro estados de toda tela de dados, com critério de aprovação

Os dois números da New Relic em 2026, com 200 tomadores de decisão nos Estados Unidos: 82% relataram falha em produção ligada a código gerado por IA e 78% relataram aumento de incidentes. O efeito prático é a chance maior de a sua tela encontrar a fonte fora do ar.

Onde buscar o dado

Há uma escolha anterior à biblioteca: o estilo do contrato de dado. No REST, cada recurso tem seu endereço e a tela junta as partes que precisa. No GraphQL, existe um endereço só e a tela descreve os campos que quer. O REST é mais simples de manter e mais fácil de guardar em cache na própria infraestrutura; o GraphQL economiza viagem em tela com muitos blocos e cobra em complexidade de servidor. Para a maior parte dos produtos, comece em REST. Nunca misture os dois estilos para o mesmo dado, porque os caches divergem e a tela passa a mostrar dois valores para a mesma pergunta.

Decidido onde buscar, a tela precisa se defender da rede. A função `fetch` do navegador só rejeita a chamada por falha de rede, o que significa que uma resposta com erro 500 chega ao seu código como se tivesse dado certo. Confira `response.ok` à mão, sempre. Sobre esse detalhe entram três defesas, nesta ordem. O tempo limite de

alguns segundos existe para a tela não ficar presa numa fonte que já morreu, e é ele que transforma espera indeterminada em erro tratável; no `fetch`, isso é `AbortSignal.timeout(8000)`. A repetição com recuo espaça as tentativas e resolve falha momentânea, e só faz sentido depois de existir tempo limite; mal calibrada, ela transforma uma queda pequena do fornecedor em ataque da sua tela contra o seu próprio servidor. A mensagem específica fecha a sequência, separando sem conexão, erro do servidor e recurso não encontrado, porque é essa distinção que diz à pessoa se ela tenta de novo, espera ou liga para o atendimento.

No caminho até o painel, o fluxo é sempre o mesmo: buscar, conferir a resposta na borda, tratar os quatro caminhos e passar o dado já limpo ao gráfico ou ao cartão de indicador. O gráfico nunca busca dado sozinho. Essa separação permite trocar a fonte sem reescrever a visualização e mantém num lugar único a checagem de que o número existe, é número e está na unidade certa.

O cache existe para não repetir a mesma pergunta. Ele compra velocidade e cobra em frescor, e o parâmetro que decide quanto tempo o valor guardado continua valendo é uma escolha de produto disfarçada de configuração. Saldo de conta e estoque toleram segundos; catálogo tolera horas. No TanStack Query esse número é o `staleTime`, e escrevê-lo com intenção evita a reclamação clássica de que a tela mostra valor diferente do extrato.

Custo e risco por decisão, do campo preenchido à fonte do número

Faça você mesmo: um cadastro que sobrevive à fonte lenta (20 minutos)

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 17. I18n

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/i18n.ts`

Entrar em outro país sem refazer o produto

Você vai preparar o produto para rodar em outro idioma sem reescrever tela nenhuma. São dois trabalhos separados: tirar os textos de dentro do código, que se faz uma vez, e traduzir e adaptar preço, data e plural, que se repete a cada mercado. A stack é next-intl com a API Intl do navegador, com guia passo a passo.

A estrutura de pastas de um projeto com idiomas: um catálogo por idioma em `messages/`, o idioma no endereço por `src/app/[locale]/`, o middleware que redireciona a raiz e o arquivo que entrega as mensagens ao componente de servidor.

O mesmo componente com direção da esquerda para a direita e o inverso. Como o espaçamento é escrito em termos de início da linha, o layout espelha sozinho e o CSS não é reescrito por idioma.

Concatenar congela a ordem do português dentro do código e quebra nos idiomas em que a ordem muda; a chave única com placeholder devolve a frase inteira ao catálogo, e o tradutor controla a ordem e a posição da variável em cada idioma.

O fluxograma da escolha: com peso do pacote como restrição dura, Paraglide; com alguém corrigindo texto sem esperar publicação, i18next; nos outros casos, next-intl, que é o padrão do App Router.

Internacionalize o componente a seguir (React 19 + TypeScript + Tailwind v4), SEM mudar a aparência nem a acessibilidade.

Regras obrigatórias: 1. Extraia TODO texto visível para chaves de tradução; nada de literal no JSX. 2. Use placeholders nas frases (ex.: "último acesso {data}"); NUNCA concatene. 3. Plural e gênero via ICU MessageFormat dentro da string traduzível. 4. Formate datas, números e moeda com a API Intl no locale ativo (nunca à mão). 5. Troque todo CSS físico por lógico (ms-/me-/ps-/pe-/text-start) para que o layout funcione em RTL. 6. Mantenha contraste AA, foco visível e os atributos ARIA existentes. 7. Entregue também o catálogo pt-BR.json e en-US.json com as chaves criadas. Apresente: (a) o componente internacionalizado, (b) os dois catálogos, (c) uma lista das decisões de i18n que você tomou e por que.

Custo e risco por decisão de expansão de idioma

Aplique a localização a um cartão de produto para português do Brasil e para árabe, com preço, data e plural formatados pelo navegador.

Critérios de sucesso: -- Nenhuma frase montada aos pedaços: tudo por marcador dentro da frase traduzível, com plural e ordem corretos nos dois idiomas. -- Preço e data formatados pelo navegador, e não escritos à mão. -- O layout espelha sozinho no árabe, sem regra presa a um lado físico. -- Trocar o idioma não quebra nenhum texto nem o alinhamento.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 18. Ident extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/ident-extra.ts`

Use imagem gerada por modelo para textura e cena de fundo que você vai editar depois. Para o que representa a sua marca, o que inclui logotipo, ícone de sistema e diagrama, use vetor desenhado sob medida, recurso de código aberto com licença clara ou foto licenciada.

Quatro cuidados decidem se a imagem gerada pode ir ao ar, e vale segui-los nesta ordem.

1. Registre. Anote a plataforma, a data, a instrução que você escreveu, as edições que você fez e os termos de licença comercial. O que protege é a contribuição humana registrada, porque a geração pura não é registrável. 2. Converta e redimensione. A saída chega com vários megabytes e derruba o tempo até o conteúdo principal. Converta para AVIF ou WebP e gere o arquivo no tamanho do espaço real onde a imagem aparece. 3. Escreva a descrição. Objetiva quando a imagem carrega informação, vazia quando ela só enfeita. O modelo não escreve isso por você, e a falta aparece na auditoria de acessibilidade. 4. Fuja do genérico. Sem direção de arte, o resultado converge para a estética de banco de imagem. Escreva o propósito, o público e as proibições antes de gerar.

O ponto jurídico circula deformado e merece precisão. Nos Estados Unidos, a Suprema Corte negou seguimento ao caso Thaler contra Perlmutter em 2 de março de 2026, mantendo a decisão do circuito federal do Distrito de Columbia e a posição do escritório de direitos autorais de que obra puramente gerada por máquina não é registrável. Obras com contribuição humana relevante seguem protegíveis, e o limiar dessa contribuição não foi definido. Na prática, quem documenta a edição humana tem argumento e quem não documenta fica sem.

Cada escolha de ativo visual, o que você ganha e o que você paga

Faça você mesmo: a ficha de licença dos seus ativos visuais (15 minutos)

A esteira do vetor: o export inchado da ferramenta de design passa pelo SVGO na esteira de publicação (múltiplas passadas, precisão de duas casas, viewBox preservado) e chega à pasta pública com metade do tamanho; com muitos ícones, o arquivo único de símbolos entrega todos numa requisição, com cor herdada e rótulo por uso.

Crie uma rota de OG image DINÂMICA no Next.js (App Router) usando @vercel/og. Caminho: src/app/og/route.tsx, rodando no edge runtime.

Requisitos: - Tamanho 1200x630 (padrão Open Graph). - Receba ?title=... da query string e imprima esse título na imagem. - Trate o caso sem title (texto padrão) e limite o tamanho do texto. - Layout em flexbox (Satori só entende flex), fundo com a cor da marca, título grande, e um rodapé com o domínio do site. - Use uma fonte custom carregada via fetch de arquivo .ttf (woff não serve no Satori). - Adicione um script que rode o SVGO em múltiplas passadas, com precisão de duas casas e viewBox preservado, lendo de src/assets/icons e gravando em public/icons. Depois mostre como apontar a metatag og:image para essa rota em generateMetadata.

Ícone de aplicativo: os cinco arquivos que bastam

Ativo visual, origem e peso que ele pode custar

A rolagem também virou linha do tempo de animação. Dá para nomear um contêiner de rolagem ou a visibilidade de um elemento e fazer outro elemento consumir esse nome, de modo que uma barra de progresso reaja à rolagem do artigo abaixo dela sem uma linha de programação. O risco acompanha o ganho, porque o recurso ainda não se comporta igual em todo navegador. As animações dirigidas por rolagem não são interoperáveis em julho de 2026: constam como área de foco do Interop 2026, o Safari 26.4, de 24 de março de 2026, passou a executá-las na thread do compositor, e o Safari 26.5, de 11 de maio de 2026, publicou mais quatro correções de linha do tempo e de progresso. Trate o efeito como melhoria progressiva, com o estado final estático garantido para quem não o receber, porque requisito de aceite que depende do navegador do cliente transforma a homologação numa loteria.

Os frameworks passaram a embrulhar a interface de transição de visualização do navegador. No Next, uma opção de configuração liga as transições do roteador de aplicação e os elementos com continuidade recebem um nome de transição; no Astro, o componente de roteamento no cliente cumpre o mesmo papel. Nos dois casos a transição fica declarativa e desliga sozinha sob preferência por menos movimento. O detalhe de compatibilidade divide o recurso em duas metades de risco diferente. Dentro do mesmo documento, a transição é linha de base desde o Firefox 144, de outubro de 2025, e cabe num requisito. Entre documentos diferentes ela não é linha de base segundo a documentação de referência, e os tipos de transição chegaram ao Firefox 147 apenas para aplicação de página única, o que a coloca como polimento para quem tem o navegador atualizado.

O orçamento de movimento pronto para colar no seu arquivo de instruções: faixas de duração, mola calibrada, as garantias de acessibilidade e o teto de resposta ao toque. Estourou o teto, a animação sai.

Faça você mesmo: carimbe o orçamento de movimento num pedido real (20 minutos)

Custo e risco por decisão de movimento

Duas coisas de nome parecido se confundem em quase toda conversa sobre animação. Uma é o construtor de sites chamado Framer, que gera a página inteira e injeta o próprio motor de movimento no resultado. A outra é o pacote que um projeto React instala, batizado originalmente com nome semelhante e hoje chamado apenas Motion. Em nenhum dos sites React do corpus há prova do pacote no código entregue: o que aparece é sempre o motor nativo do construtor, identificável por um atributo próprio no HTML. Ao pedir animação ao agente, escreva o nome do pacote e o caminho de importação.

Peça assim

Aplique movimento no padrão de contenção das referências de alto padrão: - Camada base em CSS puro: reação ao hover só com transform/opacity, revelação ao entrar na viewport por IntersectionObserver (estado base visível), sanfona nativa. - Para o caráter (entrada de cards, hero), prefira curva de mola via linear() no CSS, sem biblioteca; só suba para a lib Motion (motion/react) se precisar herdar velocidade de gesto (arrastar e soltar). - NÃO instale GSAP nem WebGL a menos que animação seja a tese do produto; se propuser, justifique em uma frase. - Adicione transição de rota pela opção de View Transitions do framework, com view-transition-name nos elementos contínuos, e trate a transição entre páginas como melhoria progressiva. - Tudo sob prefers-reduced-motion: reduce; anime só transform e opacity; não piore a medida de resposta à interação. Se eu escrever 'Framer Motion', leia 'a lib Motion (motion/react)', nunca o builder Framer.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 19. Layout mobile

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/layout-mobile.ts`

Layout no celular: a tela pequena manda

Você vai desenhar do celular para cima e resolver, um a um, os defeitos que só aparecem na tela pequena. São oito: a barra do navegador que come a altura, o recorte da câmera que engole o botão, a grade que precisa se adaptar sem lista de pontos de quebra, o alvo pequeno demais para o dedo, o menu que ignora o botão voltar, a tabela que estoura, o campo que fica embaixo do teclado e a foto de desktop baixada no 4G.

/* Errado: mede a tela com a barra do navegador escondida. No celular o bloco nasce alto demais e o botão cai fora. */ .hero { height: 100vh; }

/* Certo: acompanha a barra que aparece e some. */ .hero { min-height: 100dvh; /* dinâmica: segue o movimento */ }

/* Para o que PRECISA estar visível na abertura, use o pior caso. */ .checkout-acao { min-height: 100svh; /* pequena: conta com a barra visível */ }

Os dois passos da área segura, na ordem

/* 1. A grade conta as colunas. Nenhuma media query envolvida. */ .vitrine { display: grid; gap: 16px; grid-template-columns: repeat(auto-fit, minmax(220px, 1fr)); }

/* 2. Título fluido entre um piso e um teto. */ .titulo { font-size: clamp(1.5rem, 4vw, 3rem); line-height: 1.15; }

/* 3. O componente decide pelo espaço que recebeu, não pela janela. */ .slot-do-cartao { container-type: inline-size; }

.cartao { display: grid; gap: 8px; }

@container (min-width: 400px) { .cartao { grid-template-columns: 120px 1fr; /* foto ao lado do texto */ align-items: center; } }

Os dois tamanhos que importam e o papel do espaçamento: 24 por 24 é o piso do critério 2.5.8 do WCAG 2.2 no nível AA, 44 por 44 é o alvo confortável do critério 2.5.5 no nível AAA, e alvos colados erram o toque mesmo passando no piso.

Área clicável generosa e gestos que se comportam

"use client"; import { useEffect } from "react";

export function useFecharNoVoltar(aberta: boolean, fechar: () => void) { useEffect(() => { if (!aberta) return;

// 1. Ao abrir, empilha uma entrada no histórico. window.history.pushState({ gaveta: true }, "");

// 2. O botão voltar do aparelho dispara popstate: aqui só fechamos. const aoVoltar = () => fechar(); window.addEventListener("popstate", aoVoltar);

return () => { window.removeEventListener("popstate", aoVoltar); // 3. Se a gaveta fechou pelo X ou pelo Esc, desempilha a entrada // para não deixar um voltar "morto" no histórico. if (window.history.state?.gaveta) window.history.back(); }; }, [aberta, fechar]); }

Os dois caminhos da tabela no celular

Campo por campo: o tipo, o teclado e o preenchimento

Falta a imagem, que é onde o celular paga a conta mais cara sem ninguém perceber.

Uma foto de banner exportada para monitor grande tem largura suficiente para preencher um desktop. Servida como está para um celular numa rede 4G, ela baixa inteira, ocupa dados de quem está com o pacote no fim e trava a primeira pintura da tela. A tela mostra uma imagem de 390 pixels de largura, e o navegador baixou muito mais que isso.

A correção tem duas partes, e a segunda é a que o pessoal esquece. O `srcset` lista as versões disponíveis com a largura real de cada arquivo. O `sizes` diz quanto espaço a imagem vai ocupar em cada faixa de tela. Sem o `sizes`, o navegador supõe que a imagem ocupa a largura inteira da janela e escolhe um arquivo maior que o necessário, então o `srcset` sozinho entrega metade do ganho.

Dois atributos completam o conjunto. `loading="lazy"` adia o download do que está abaixo da dobra, e `width` com `height` reserva o espaço antes de a imagem chegar, evitando o pulo de conteúdo que faz a pessoa tocar no lugar errado.

No Next.js, o componente `next/image` faz o `srcset`, o formato moderno, o carregamento adiado e a reserva de espaço sozinho, a partir do `width` e do `height` que você informa. Ainda assim você precisa escrever o `sizes` quando a imagem não ocupa a largura inteira, porque essa informação vem do seu layout e o componente não tem como adivinhá-la.

A foto certa para cada tela, com as duas partes

Custo e risco por decisão de layout em celular

Exercício: peça ao agente a passada de celular, com critérios conferíveis

Layout de celular quebra em três lugares que o computador nunca mostra: na altura, porque a barra do navegador aparece e some; nas bordas, porque o recorte da câmera e a barra de gestos ocupam espaço que o sistema informa e você precisa devolver; e no dedo, porque o alvo que o mouse acerta o toque erra. Resolva esses três antes de qualquer refinamento visual, e teste com o teclado aberto num aparelho de verdade. O modo dispositivo do navegador simula tamanho de tela com fidelidade, e simula mal justamente as três coisas que quebram.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 20. Mapas geo

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/mapas-geo.ts`

Quando a resposta é um mapa e quando é uma tabela ordenada

Você aprende a decidir se o caso pede mapa ou uma lista ordenada, a escolher entre Leaflet, MapLibre e deck.gl com um critério só, a montar um mapa do Brasil sem pagar por chave de API e a não pintar o mapa de população achando que pintou o mapa de vendas. Cada biblioteca vem com o guia de instalação passo a passo e o erro que derruba a primeira tentativa.

Os três degraus da stack de mapa

Leaflet, a ficha completa. O que é: a biblioteca de mapa mais antiga e mais simples da web, feita para colocar marcadores sobre um mapa comum. O que vem dentro: mapa deslizante, marcador com balão de informação, camadas, controles de zoom e uma coleção enorme de plugins da comunidade. Quando usar: página de contato com as lojas, mapa de eventos, localizador de unidade. Quando não usar: quando você precisa colorir áreas por um número ou girar e inclinar o mapa.

Guia passo a passo do primeiro mapa com marcadores.

1. Instale com `npm install leaflet` e, se você usa TypeScript, `npm install -D @types/leaflet`. 2. Importe a folha de estilo da biblioteca no seu componente. Sem ela, o mapa aparece em pedaços desalinhados. 3. Crie uma `div` com altura declarada, do tipo `h-[400px]`. Sem altura, o mapa não inicializa e você vê um espaço cinza. 4. Chame `L.map(elemento).setView([-15.79, -47.88], 4)`, que centra o mapa no Brasil. 5. Acrescente o fundo com `L.tileLayer` apontando para um provedor de peças e escreva a atribuição exigida por ele no campo `attribution`. 6. Para cada loja, chame `L.marker([lat, lng]).addTo(mapa).bindPopup("Nome da loja")`.

Resultado esperado: um mapa que arrasta, aproxima e mostra o balão ao clicar no marcador. Os dois erros de estreia são a altura esquecida no passo 3 e a folha de estilo esquecida no passo 2, e juntos eles respondem pela maior parte das perguntas de iniciante sobre Leaflet na internet.

MapLibre com PMTiles, a ficha completa. O que é: o MapLibre desenha o mapa a partir de dados vetoriais em vez de imagens prontas, o que permite girar, inclinar e trocar a cor de cada camada. O PMTiles é um formato que guarda o mapa inteiro num arquivo só. O que vem dentro: estilo por camada declarado em JSON, giro e inclinação, globo tridimensional, marcadores, balões e leitura de fontes de dado externas. Quando usar: painel por região, mapa com identidade visual própria, qualquer caso em que você quer parar de pagar por carregamento. Quando não usar: cinco marcadores numa página institucional.

A diferença de custo está no jeito de servir as peças do mapa. No modelo tradicional, cada pedacinho quadrado vem de um serviço que conta os pedidos e cobra por mil carregamentos, então o sucesso da campanha vira aumento de fatura. Com o PMTiles, todos os pedaços vivem dentro de um arquivo hospedado no seu armazenamento, e o navegador pede só o trecho que precisa, como quem lê duas páginas de um livro sem comprar o volume inteiro. Você troca fatura variável por uma linha fixa de armazenamento, e assume a tarefa de atualizar o arquivo de tempos em tempos, porque o mapa desatualizado custa exatamente o mesmo do mapa correto.

Guia passo a passo do mapa vetorial do Brasil sem chave de API.

1. Baixe o extrato do planeta em PMTiles no site do projeto Protomaps. 2. Instale a ferramenta de linha de comando `pmtiles` e recorte só o Brasil com `pmtiles extract`, passando a caixa de coordenadas do país. O arquivo cai de dezenas de gigabytes para poucos. 3. Suba o arquivo no seu armazenamento, seja Cloudflare R2, Amazon S3 ou equivalente, e confirme que ele responde a pedidos parciais e permite acesso do seu domínio. 4. No projeto, instale `npm install maplibre-gl pmtiles`. 5. Registre o protocolo antes de criar o mapa: `const p = new pmtiles.Protocol(); maplibregl.addProtocol("pmtiles", p.tile)`. 6. Crie o mapa apontando a origem para `pmtiles://https://seu-armazenamento/brasil.pmtiles` e aplique um estilo básico legível. 7. Abra a aba

de rede do navegador e confirme que nenhuma requisição sai para um serviço de mapa pago.

Resultado esperado: um mapa vetorial do Brasil que gira, inclina e usa as suas cores, com zero chave de API no projeto. O erro que todo iniciante comete é esquecer o passo 5: sem registrar o protocolo, o MapLibre trata `pmtiles://` como endereço desconhecido e o erro chega travestido de falha de rede, o que manda você caçar problema de permissão que não existe.

A arquitetura tradicional pede um servidor de tiles, uma chave de API e cobra por mil carregamentos, com lock-in no fornecedor. O PMTiles substitui tudo isso por um arquivo único servido com range requests em storage estático: custo de storage, zero chave, zero lock-in.

Exercício: medir quanto o contexto oficial reduz a invenção

Este exercício pede terminal e um agente instalado.

1. Verifique se o projeto da biblioteca de mapa que você usa publica # contexto para agentes (servidor de contexto, pacote de instruções ou # documentação em formato legível por máquina). Anote o que encontrou # e a data da consulta.

2. SEM esse contexto carregado, peça ao agente: "Monte um mapa com meus dados de expansão por país (arquivo paises.geojson com uma propriedade 'clientes' por país), lendo as peças do mapa de um arquivo único hospedado em armazenamento estático. Colora por taxa."

3. Guarde o resultado. Carregue o contexto oficial e repita o MESMO pedido.

4. Compare os dois códigos lado a lado e conte: # - quantos métodos citados não existem na versão instalada # - quantos nomes de formato ou de protocolo estão errados # - qual das duas versões roda sem ajuste

Critério de aceite: você registra a diferença em número, não em impressão, # e declara a versão da biblioteca em que a comparação foi feita.

Exercício: mapa vetorial sem chave e sem servidor de mapa

deck.gl, a ficha completa. O que é: uma camada que desenha os seus dados na placa de vídeo, por cima de um mapa de base. O que vem dentro: camadas prontas de pontos dispersos, arcos, hexágonos agregados, grade, calor, caminho e polígono, todas configuradas por objeto de propriedades. Quando usar: quando o volume de pontos passa do que a tela consegue desenhar um a um. Quando não usar: abaixo de alguns milhares de pontos, onde o MapLibre sozinho já resolve.

A ordem de grandeza que uso como heurística: até cerca de mil elementos, o desenho vetorial comum resolve; alguns milhares ainda cabem em desenho por pixel; centenas de milhares de pontos pedem a placa de vídeo. Dois milhões de pedidos de um aplicativo de entrega, plotados um a um ou agregados em hexágonos, ficam na terceira faixa.

Guia passo a passo da primeira camada.

1. Instale com `npm install deck.gl` sobre um projeto que já tenha MapLibre funcionando. 2. Importe `MapboxOverlay` do pacote de compatibilidade e adicione ao mapa como controle. 3. Crie uma `ScatterplotLayer` passando o array de pontos, a função que devolve a posição de cada um e o raio em metros. 4. Troque a mesma fonte de dados para uma `HexagonLayer` com `radius` de mil metros e ligue as duas com um botão, para quem lê alternar. 5. Meça: abra o painel de desempenho do navegador e observe a taxa de quadros ao arrastar o mapa.

Resultado esperado: a rolagem continua fluida com centenas de milhares de pontos, e você tem o número medido em vez da sensação. O erro que todo iniciante comete é colocar rótulo de texto dentro da camada de dado da placa de vídeo, o que derruba o desempenho que motivou a troca. Rótulo fica no andar de cima, e a figura seguinte mostra os quatro andares.

Três formas de mapa temático, e o que cada uma exige

O erro clássico atravessa as três formas: não dividir por nada. Vendas, clientes, cliques e chamados crescem junto com a população, então o mapa colorido pelo número bruto vira uma cópia do mapa demográfico com cara de análise. Quase todo mapa do Brasil com o Sudeste em vermelho é isso.

Antes de escolher a cor, escolha o denominador e escreva o nome dele na legenda: habitantes, domicílios, base instalada ou produto interno bruto da região. A legenda importa porque a figura circula sozinha, e o que só você sabe não protege ninguém.

Duas escolhas decidem se a figura ainda distorce depois de normalizada: quantas faixas de cor a escala tem e onde ficam os cortes entre elas. Cinco faixas de igual amplitude e cinco faixas por quantil produzem mapas visivelmente diferentes a partir do mesmo dado, e a segunda sempre parece mais equilibrada, porque por construção ela distribui as regiões em partes iguais. As duas são defensáveis. Escreva ao lado da legenda qual você usou, junto do denominador, e qualquer pessoa consegue reconstruir a sua figura.

Exercício: dois milhões de pontos e o teto de desenho

Faça você mesmo: auditar o mapa pela borda, em 15 minutos

Custo e risco por decisão de mapa

Fecha o módulo um exercício de ponta a ponta: construa um mapa de penetração por unidade federativa do Brasil, dirigindo o agente e conferindo cada item da lista abaixo.

Critérios de sucesso: -- O mapa de base usa um arquivo único hospedado em armazenamento próprio, sem nenhuma chamada a serviço de mapa pago, comprovado na lista de requisições de rede. -- Nenhuma variável de ambiente contém chave de mapa. -- A camada de área colore por taxa normalizada, nunca por contagem bruta, e a legenda declara o denominador. -- Existe uma tabela alternativa com unidade federativa e valor, percorrível por teclado e legível por leitor de tela. -- Você auditou contraste e navegação por teclado e corrigiu o que falhou, anexando o relatório. -- Você justifica em uma frase por que este caso pede mapa, e não a lista ordenada.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 21. Ondas melhoria

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/ondas-melhoria.ts`

Conduzir um programa de modernização em ondas

Como pegar um projeto que já está no ar e modernizá-lo em cinco rodadas, com um portão de verificação entre uma e a seguinte. Você começa medindo, sem tocar em arquivo, e termina com o número antes e depois na mesma tabela.

git worktree: duas frentes escrevendo ao mesmo tempo sem se atropelar.

O que é: um comando do próprio git que cria uma segunda pasta de trabalho ligada ao mesmo repositório, cada uma num ramo diferente. Você abre dois editores, roda dois agentes e nenhum dos dois vê o arquivo do outro.

O que vem dentro: `git worktree add`, `git worktree list` e `git worktree remove`. As pastas compartilham o histórico e os objetos do git, então nada de código é duplicado.

Quando usar: sempre que mais de uma frente escrever no mesmo projeto ao mesmo tempo. Quando não usar: quando você trabalha sozinho e em série, porque aí `git checkout -b` já basta.

Passo a passo:

1. Na raiz do projeto, rode `git checkout -b modernizacao`. 2. Crie a segunda pasta com `git worktree add ../wt-tokens -b onda3-tokens`. Resultado esperado: a pasta `wt-tokens` aparece ao lado do projeto, com o código inteiro. 3. Crie a terceira com `git worktree add ../wt-icones -b onda3-icones`. 4. Rode `npm install` dentro de cada pasta nova, porque a `node_modules` não é compartilhada. Resultado esperado: cada pasta roda o projeto sozinha. 5. Ao fim da rodada, volte à pasta principal e incorpore os dois ramos, um de cada vez. Resultado esperado: nenhum conflito, porque os escopos não se cruzaram.

O erro que todo iniciante comete: dar aos dois agentes o mesmo arquivo para editar, mesmo com worktree separado. A separação de pasta protege contra sobrescrita, e a separação de escopo protege contra conflito na hora de juntar. Você precisa das duas.

O pedido-mãe abaixo dispara as cinco ondas. Cole-o uma vez, com o repositório anexado ou o endereço do site no ar, e o agente conduz rodada a rodada, esperando o seu OK entre elas.

Prompt-mãe: uma demanda, cinco ondas (cole e adapte)

A ferramenta que fecha cada rodada: Lighthouse e axe-core na primeira, teclado e inventário de estados na segunda, busca por cor solta na terceira, aba Performance na quarta e Lighthouse CI com orçamento na quinta.

Faça você mesmo: a linha de base em 20 minutos

Prompt-mãe da Onda 2 (cole e adapte)

A terceira rodada unifica o sistema visual e a identidade. Sobre um fluxo já claro, as frentes normalizam tokens de cor com tema escuro de verdade, conjunto de ícones coerente, tipografia servida pelo próprio domínio, ilustração vetorial e otimização de imagem, tudo amarrado a um único arquivo de tokens. É a rodada com o maior efeito sobre a percepção de qualidade por hora investida, e a que mais depende de a anterior ter fechado.

O ganho mais rentável aqui é esse arquivo único. Com toda cor saindo de uma variável, trocar o tema inteiro ou a identidade da marca vira edição num lugar só, e o tema escuro para de exibir texto preto sumindo no fundo escuro. Na maioria dos produtos antigos acontece o contrário: a mesma cor de marca escrita à mão em arquivo após arquivo, cada ocorrência com um dígito diferente, e nenhuma forma de saber quantas telas quebram quando alguém decide clarear o azul. O teste de que a rodada fechou cabe em um comando: uma busca por valor de cor solto no repositório volta vazia.

Prompt-mãe da Onda 3 (cole e adapte)

A quarta rodada cuida do movimento e das microinterações. Com o visual coeso, entra o movimento a serviço da clareza: revelação conforme a rolagem e transição entre telas com recurso nativo do navegador, microinterações de resposta à ação, tridimensional leve onde ele agrega e vetor animado para elementos com estado, sempre com a preferência do sistema por movimento reduzido respeitada e o estado base visível. Cada animação precisa de função declarada. Animação sem função é peso de página com aparência de capricho.

O defeito mais perigoso da rodada é o carrega e some: conteúdo com `opacity:0` no estado base, esperando um gatilho de JavaScript que falha. O bloco fica invisível para o usuário e para o buscador ao mesmo tempo, e numa página secundária isso leva semanas até alguém reclamar. A regra que fecha essa porta é curta: o `opacity:0` vive só dentro do `@keyframes`, com `animation-fill-mode: both`. Vale a mesma disciplina para o tridimensional, que sem teto de polígono, de peso e de quadros por segundo trava o celular de quem está comprando, e o prejuízo aparece no funil de compra antes de aparecer em relatório de erro.

Prompt-mãe da Onda 4 (cole e adapte)

O ciclo seguro de cada onda: auditar em read-only, propor com antes/depois, aplicar em branch, verificar nos quality gates e só então fazer merge. Quando um portão reprova, o ciclo volta para a correção.

Um levantamento da New Relic com 200 tomadores de decisão nos Estados Unidos completa o quadro: 82% tiveram falha em produção ligada a código escrito com inteligência artificial nos últimos seis meses, e 74% dizem que ao menos um quarto desse código precisa de retrabalho. Num projeto que já está no ar o efeito é amplificado, porque cada frente escreve sobre código que já tem usuário do outro lado.

O segundo risco é ficar sem o modelo no meio do trabalho. Em 12 de junho de 2026, o governo dos Estados Unidos aplicou controles de exportação sobre o Claude Fable 5 e o Claude Mythos 5, e a Anthropic suspendeu o acesso para todos os clientes; os controles foram levantados em 30 de junho e o acesso reaberto em 1º de julho de 2026. Em 26 de junho, a OpenAI anunciou o GPT-5.6 já limitando o acesso inicial a parceiros homologados. Foram dezenove dias sem o modelo topo de linha.

Escreva no `README` do projeto qual é o modelo alternativo, e teste o pedido-mãe nele pelo menos uma vez. Descobrir a dependência no dia da suspensão custa a semana inteira.

As cinco decisões de condução e o que cada uma faz com o seu projeto

Faça você mesmo: o programa de cinco rodadas de ponta a ponta

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 22. Plugins visuais

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/plugins-visuais.ts`

Apêndice: plugins visuais e geração de imagem

O que entra no seu agente e o que sai dele em forma de imagem. Na entrada, como instalar plugin sem cair no pacote de nome inventado, o que ler antes de confirmar e qual escopo escolher. Na saída, os cinco caminhos para criar uma imagem, o guia de otimização com sharp e o componente de imagem do Next, com o passo a passo de cada um e o erro de estreia.

As duas vias pelas quais um pacote entra no assistente: a descritiva injeta texto cobrado por turno e vira orçamento recorrente; a executiva registra comandos e conexões que rodam com o privilégio de quem instalou e equivale a admitir um fornecedor. Três defesas se reforçam em ordem: catálogo autorizado, leitura do que o pacote adiciona e escopo restrito.

As quatro abas do painel `/plugin`, e o que procurar em cada uma

As cinco dimensões que a skill `frontend-design` governa

Prompt: pedir a tela com direção estética declarada

Use a skill frontend-design para a landing de um ateliê de joias autorais.

Propósito: transmitir exclusividade e feito à mão; público: clientes de alto padrão. Direção estética: editorial, tipografia serifada de display marcante, muito espaço negativo, uma única cor dominante quente com acento sóbrio. Proibido: gradiente roxo, fontes Inter/Roboto/Arial, layout centralizado simétrico. Elemento memorável: um título serifado grande alinhado à borda esquerda, com a foto do produto sangrando para fora da grade.

Depois de definir a direção, implemente em React + Tailwind seguindo os tokens e o piso de acessibilidade do CLAUDE.md (contraste WCAG AA, foco visível, estados de loading e erro). Antes de finalizar, liste as escolhas estéticas que fez e por que.

O mecanismo formal de extensões do protocolo de contexto é onde a terceira porta mora: o servidor conectado devolve uma interface interativa completa, renderizada pelo aplicativo do usuário dentro de uma moldura isolada, e as ferramentas declaram antes quais telas podem entregar, o que permite carregar com antecedência, guardar em cache e submeter cada uma a revisão de segurança. A declaração prévia é o detalhe que interessa a quem responde por risco, porque ela faz a lista de interfaces que um fornecedor pode desenhar dentro da sua ferramenta ser conhecida antes do primeiro uso.

Qual peça atende qual etapa, na ordem em que elas entram

Prompt: gerar um conjunto coerente de ilustrações

Preciso de um conjunto de 4 ilustrações para as seções da landing page de [produto], com estilo COERENTE entre si: - traço/linha limpo, paleta com a cor de destaque #[hex] e tons neutros; - proporção 4:3, fundo transparente ou claro; - temas: [seção 1], [seção 2], [seção 3], [seção 4].

Gere um prompt-base reutilizável que eu possa variar só no tema, mantendo o mesmo estilo. Antes de gerar, confirme a licença de uso comercial do modelo e quem detém os direitos da imagem resultante. Entregue as imagens em alta e liste o passo de otimização (converter para AVIF/WebP e gerar o tamanho certo).

A esteira em quatro etapas, com o peso caindo a cada uma: original de 4,2 MB, redimensionado para a largura real, convertido para formato moderno e entregue com largura, altura, descrição e imagem provisória. As duas etapas do meio são as que somem do processo.

Cada escolha, da porta de entrada à de saída

Duas listas e um número, em dez minutos. Primeira lista: os catálogos de onde você aceita instalar plugin, escrita no arquivo de projeto. Segunda lista: as origens das imagens que o seu site usa, com a licença de cada uma anotada e uma marca vermelha nas que você não sabe responder. O número é o limite de espera que a sua tela precisa respeitar, escrito ao lado do aparelho e da rede em que você vai medir. Se as duas listas e o número não saírem em dez minutos, você acabou de encontrar a próxima tarefa do projeto.

Faça você mesmo: bancada visual e uma imagem de destaque, em 20 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 23. Prompt extra

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/prompt-extra.ts`

Anexe uma referência antes de escrever o pedido. Você tem quatro à mão: uma captura de tela, o endereço de um quadro no Figma, um componente que já existe no seu repositório e o print da tela com defeito. Cada uma troca adjetivo por evidência. Escrever `um card bonito e moderno` deixa o agente escolher por você; anexar a captura do card que você já aprovou tira a escolha da mão dele.

Três palavras aparecem daqui em diante e vale definir agora. Modo de desenvolvedor é a aba do Figma que mostra medidas e nomes reais em vez da aparência. Variável de design é uma cor ou um espaçamento guardado com nome próprio, do tipo `--color-brand-600`, que permite trocar a identidade inteira mexendo num lugar só. Code Connect é o registro de que determinado desenho no Figma corresponde a determinado componente que você já escreveu, o que impede o agente de recriar o mesmo botão pela quinta vez.

A ordem da figura é a ordem de força, e vale seguir de baixo para cima. Se você não tem nenhuma referência, arranje uma antes de pedir. Um site que você admira, aberto no navegador e capturado com a tecla de print, já é âncora suficiente para o primeiro pedido. A loja da Apple, a tela de pagamento da Stripe e a listagem de hotéis do Booking servem bem porque cada uma resolve um problema visual diferente e você reconhece a solução na hora.

O mesmo card pedido de dois jeitos: o pedido vago devolve cor inventada, fonte de sistema e nenhum estado de erro; o pedido ancorado nomeia os tokens, lista os estados que faltam na imagem e diz o que ignorar na captura.

[imagem anexada: screenshot da tela de listagem de pedidos]

OBJETIVO: Reconstrua FIELMENTE a tabela de pedidos da imagem anexa como um componente React 19 + TypeScript + Tailwind v4. Reproduza layout, espaçamento, hierarquia tipográfica e o selo de status colorido.

O QUE É DADO (vira prop, não hardcode): número do pedido, cliente, valor, data e status. Os textos da imagem são exemplo.

TOKENS (NÃO INVENTE CORES): use exatamente estes, que são os do nosso sistema: -- superfície #ffffff, texto principal #0f172a, texto secundário #475569 -- status: pago = oklch(0.65 0.15 150), pendente = oklch(0.75 0.15 80), cancelado #64748b Se alguma cor da imagem não casar com a lista, use a mais próxima da lista e me avise; não crie um hex novo.

ESTADOS (a imagem não mostra): linha em hover com leve realce; foco de teclado visível na linha; estado vazio ("Nenhum pedido ainda") e estado de carregamento (skeleton de 5 linhas).

ACESSIBILIDADE: semântica com ; o status não depende SÓ da cor (texto + bolinha); contraste AA em todo texto.

IGNORE na captura: a barra do navegador, o cursor e qualquer sombra de janela do sistema.

[Figma Dev Mode MCP conectado; frame selecionado: "Card / Produto / Default"]

Implemente o frame selecionado como componente React + Tailwind v4.

USE OS METADADOS DO MCP, NÃO ESTIME: -- Cores: gere a partir das VARIÁVEIS do Figma (ex.: --color-surface, --color-brand-600). Não converta para hex chutado. -- Espaçamento e tamanhos: use as medidas exatas do Auto Layout (gap, padding) que o MCP retornar. -- Tipografia: respeite os estilos de texto nomeados (família, peso, tamanho, altura de linha). -- Se houver mapeamento Code Connect para um componente nosso, REUTILIZE-O em vez de recriar.

O QUE O FIGMA NÃO DEFINE, eu defino: estados de hover/foco/carregando, comportamento responsivo abaixo de 640px e os textos, que viram props.

[imagem anexada: print do card de produto no mobile (390px), com o preço vazando para fora da borda direita]

O bug está na imagem: no mobile (390px) o preço vaza para fora da borda direita do card, como mostrado.

Corrija SÓ isso, sem mexer no layout do desktop. Provável causa: falta de min-w-0 / truncamento no container do texto, ou largura fixa onde deveria ser flexível.

Depois, AUDITE no browser via MCP em 390px e me confirme que o conteúdo cabe dentro da borda em ambos os temas.

Qual âncora usar, como usar e o erro que cada uma esconde:

Tabela do módulo

Âncora	Como usar	O erro que todo iniciante comete
Captura de tela	Para o arranjo e a proporção	Anexar sem dizer o que ignorar na imagem
Figma conectado	Pelas variáveis e medidas reais	Conectar e continuar pedindo sem citar as variáveis
Código do repositório	Apontando um componente pelo caminho do arquivo	Descrever o padrão em palavras em vez de apontar o arquivo
Print do defeito	Na largura exata em que o defeito aparece	Descrever o sintoma por texto e receber outra correção

Antes do próximo pedido, responda uma pergunta: qual dessas quatro você tem agora. Qualquer uma já muda o resultado, porque cada âncora concreta é um pedaço a menos que o agente preenche com a média da internet.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 24. Pwa

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/pwa.ts`

Apêndice: app instalável e comportamento em celular

Você vai transformar um site num aplicativo que instala na tela inicial e abre sem rede, sem passar por loja e sem manter dois projetos. O apêndice cobre as três peças que fazem isso funcionar, a ficha do Workbox com guia passo a passo, e os três detalhes de celular que separam aplicativo de site encolhido.

Instalar pela web resolve o caso mais comum. Atende ao produto que ainda valida uma ideia, ao que precisa ser encontrado na busca e ao que ganha com correção publicada em minutos, e atende, acima de tudo, a quem quer manter um código só para web, Android e iPhone. O Starbucks e o Spotify usam a versão instalável da web ao lado do aplicativo de loja, e o Twitter serviu por anos a maior parte dos usuários por esse caminho.

Construir nativo passa a valer com três gatilhos: o produto depende de recurso pesado do aparelho, como processamento de câmera, localização contínua, sensores, conexão de curta distância ou biometria; estar na loja faz parte do posicionamento comercial; ou a exigência gráfica passa do que o navegador entrega.

Entre os dois extremos existe um caminho intermediário, o React Native com Expo, que reaproveita o seu conhecimento de React para gerar aplicativo de loja sem montar o ambiente completo de cada plataforma. Ele não deve ser a escolha padrão. Produto feito de conteúdo, catálogo, painel ou vitrine é melhor servido pela instalação via web, porque a ponte cobra o preço da loja sem devolver capacidade que esse produto vá usar. Antecipar essa migração troca publicação em segundos por aprovação de terceiro e dois pacotes a manter.

Tabela do módulo

Estratégia	Como funciona	Quando usar
Cache-first	Cache; rede só se faltar	Fontes, ícones, JS/CSS versionado (hash)
Network-first	Rede; cache se a rede falhar	API, feed, dados que mudam e precisam ser frescos
Stale-while-revalidate	Cache agora + atualiza no fundo	Avatares, capas, conteúdo "quase atual" tolera 1 visita
Network-only	Só rede, nunca cacheia	Pagamento, login, checkout (nunca offline)
Cache-only	Só cache, nunca rede	App shell pré-cacheado no build (precache)

Funcionar sem rede tem dois níveis, e a distância de custo entre eles é grande.

No primeiro, abrir sem rede, a estrutura da tela fica guardada na instalação e a pessoa vê uma página explicando a ausência de conexão em vez da tela de erro do navegador. É barato e entrega a maior parte do valor percebido, porque o que irrita é o aplicativo que não abre.

No segundo, operar sem rede, os dados ficam no aparelho, a interface lê do local e uma fila sincroniza quando a conexão volta. É a lógica do Google Agenda e do bloco de notas, e traz junto o problema de resolver conflito quando duas alterações competem pelo mesmo registro. Suba do primeiro para o segundo quando existir gente usando o produto em lugar sem sinal, e não pela vontade de anunciar o recurso.

O convite de instalação é diferente em cada plataforma. Em Android e em computador, o navegador dispara o evento `beforeinstallprompt` quando o site cumpre os critérios; guarde esse evento, mostre o botão de instalar no momento certo, e não na primeira visita, e chame `prompt()` no clique. No iPhone esse evento não existe: a instalação segue manual, pelo menu de compartilhamento, então detecte o sistema e mostre a instrução no lugar do botão. Quem entrega o mesmo botão para as duas plataformas entrega, no iPhone, um botão que não faz nada.

Antes e depois da área segura: sem `viewport-fit=cover` e sem `env(safe-area-inset-bottom)`, a barra de gestos cobre o botão de pagar; com os dois, o recuo do sistema abre espaço e o botão volta a ser tocável.

Transforme meu projeto Next.js (App Router, React 19, Tailwind v4) em uma PWA instalável, usando Workbox. Entregue, com explicação curta de cada arquivo:

1. public/manifest.webmanifest com nome, short_name, display standalone, theme_color e background_color coerentes com meu tema, e ícones 192/512 + um maskable 512. Liste os arquivos de ícone que devo gerar. 2. Um service worker com estratégias por tipo de recurso: - imagens: stale-while-revalidate, teto de 60 entradas / 30 dias; - rotas /api/: network-first com timeout de 3s, fallback no cache; - fontes e assets versionados: cache-first; - precache do app shell + página /offline.html como fallback de navegação. 3. O registro do service worker no cliente, só em produção e só se 'serviceWorker' existir em navigator. 4. Um componente React que escuta beforeinstallprompt, guarda o evento e só aparece quando dá para instalar; no iOS (sem o evento), mostra a dica de "Adicionar à Tela de Início". 5. A meta viewport com viewport-fit=cover e exemplos de safe-area-inset na barra superior e na barra de ação inferior.

Critérios de aceite: o app instala no Android/desktop; abre offline mostrando /offline.html; não serve cache velho após deploy (cache versionado); passa na auditoria PWA do Lighthouse. Use min-height: 100dvh nos containers de tela cheia. Aponte qualquer limitação no iOS.

Custo e risco por decisão de distribuição em celular

Transforme uma página comum num aplicativo instalável com funcionamento sem rede.

Critérios de sucesso: -- Ficha de instalação com nome e ícones, instalável de verdade no aparelho. -- Processo de segundo plano com estratégia adequada por tipo de recurso. -- Existe uma página servida quando a rede falha, no lugar da tela de erro do navegador. -- O bloco de abertura usa a altura dinâmica e respeita a área segura da tela.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 25. React tailwind css

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/react-tailwind-css.ts`

Apêndice: React, Tailwind e o CSS que o agente ainda não conhece

Os cinco padrões de React e Tailwind que aparecem em todo componente que o agente devolve, a regra que diz onde cada tipo de estado deve morar e o catálogo do que o navegador já entrega pronto sem você instalar nada. Cada biblioteca vem com o guia passo a passo e o erro de estreia. Fecha com a política de adoção que você escreve uma vez e cola no arquivo de instruções do agente.

Padrão de componente React + Tailwind prontos para produção

O Tailwind parte dos utilitários: em vez de criar uma classe de CSS por componente, você compõe a aparência com classes pequenas e previsíveis direto no JSX. O primeiro hábito a instalar é pensar da tela pequena para a grande, porque classe sem prefixo vale para todas as larguras e os prefixos `sm:`, `md:` e `lg:` aplicam a partir daquele ponto de quebra, o que significa estilizar o celular primeiro e acrescentar apenas o que muda no computador. Quando nenhuma classe pronta serve, existem os valores arbitrários entre colchetes, do tipo `w-[37px]`, `bg-[#0176d3]` ou `grid-cols-[200px_1fr]`, e a disciplina aqui é tratá-los como exceção, preferindo os tokens do seu `@theme`, na forma `bg-brand-500`, porque é o que mantém a consistência quando a marca mudar.

Um detalhe cobra atenção no componente reutilizável que aceita `className` de fora. Se alguém passar `px-6` e a base já tiver `px-4`, quem decide é a ordem no CSS final, e o resultado às vezes surpreende. A dupla `clsx` e `tailwind-merge` resolve o conflito, porque o `twMerge` mantém apenas o último utilitário de cada grupo. Aqui ficamos no `cn` puro para não sair de React e Tailwind, e quando o componente vira peça de biblioteca essa dupla é o padrão de mercado.

Componente de UI: todos os estados obrigatórios

Qual ferramenta para qual estado

TanStack Query, a ficha completa. O que é: a camada que cuida do dado que vem do servidor. Ela guarda em cache, junta chamadas repetidas numa só, revalida quando você volta para a aba e devolve os estados de carregando e de erro prontos. O que vem dentro: `useQuery` para ler, `useMutation` para escrever, invalidação por chave, tentativa automática em caso de falha e paginação com rolagem infinita.

Quando usar: lista que muda, consulta periódica, qualquer dado que pertence ao servidor. Quando não usar: dentro de um Server Component do Next, onde a busca acontece no servidor e a biblioteca fica sobrando.

Guia passo a passo.

1. Instale com `npm install @tanstack/react-query`. 2. Crie um `QueryClient` e envolva a aplicação no `QueryClientProvider`, num componente marcado como cliente. 3. Substitua o par `useState` mais `useEffect` por `const { data, isPending, isError } = useQuery({ queryKey: ["pedidos"], queryFn: buscarPedidos })`. 4. Trate os três estados na ordem: carregando, erro e sucesso. 5. Para gravar, use `useMutation` e chame `queryClient.invalidateQueries({ queryKey: ["pedidos"] })` no sucesso, para a lista se atualizar sozinha.

Resultado esperado: o `useEffect` de busca some do componente, e voltar para a aba já traz o dado fresco. O erro que todo iniciante comete é usar a mesma `queryKey` para consultas diferentes, o que faz uma tela mostrar o dado da outra sem nenhum aviso.

nuqs, a ficha completa. O que é: uma biblioteca que faz os parâmetros do endereço se comportarem como estado do React. O que vem dentro: `useQueryState` e `useQueryStates`, com conversão de tipo para número, booleano e data. Quando usar: filtro, aba, ordenação, busca e página. Quando não usar: dado sensível, que não deve aparecer na barra de endereço.

Guia passo a passo. Um, instale com `npm install nuqs`. Dois, envolva a aplicação no `NuqsAdapter` do seu framework. Três, troque `const [busca, setBusca] = useState("")` por `const [busca, setBusca] = useQueryState("busca")`. Quatro, para número, use `useQueryState("pagina", parseAsInteger.withDefault(1))`.

Resultado esperado: recarregar a página mantém o filtro, o botão Voltar funciona e o link com o filtro aplicado pode ser mandado para outra pessoa. O erro que todo iniciante comete é esquecer o adaptador na raiz, o que faz o valor nunca chegar ao endereço.

O que mudou de faixa entre dezembro de 2025 e julho de 2026

A linha do tempo fecha em junho, e o trimestre seguiu andando. Em junho de 2026 o `field-sizing` virou Baseline, o que autoriza campos de formulário que crescem com o conteúdo sem uma linha de JavaScript, e as funções `rect()` e `xywh()` em `shape-outside` fecharam suporte no mesmo lote, liberando texto que contorna recortes retangulares sem imagem auxiliar. No lado do que estreia num navegador só, o Chrome 149 trouxe as gap decorations, que desenham linhas nos vãos de grade e caixa flexível, e o Chrome 150 trouxe a propriedade `text-fit`, que ajusta o corpo da fonte à largura da caixa. Em julho, o Chrome 151 adicionou `ruby-overhang`. Nenhuma dessas cinco linhas apareceria numa geração de código feita hoje sem que alguém as citasse pelo nome.

O ritmo dessa lista muda em setembro, e a mudança atinge o prazo de revisão da política. O Chrome 151 é o estável corrente em 15 de agosto de 2026, publicado em 28 de julho, e ele encerrou o suporte ao macOS 12, o que joga para o orçamento de parque de máquinas quem ainda roda essa versão, sob pena de ficar com navegador sem correção de segurança. A partir de 8 de setembro de 2026, com o Chrome 153, o navegador passa a publicar versão nova a cada duas semanas em vez de quatro, mudança anunciada em 3 de março de 2026, com os canais de teste inalterados e o canal estendido mantido em oito semanas. O efeito de gestão é aritmético: a plataforma passa a mudar de estado no dobro da frequência, e uma política revista de trimestre em trimestre acumula seis versões de atraso entre uma leitura e a seguinte.

Prompt: refazer o carrossel Embla como carrossel 100% CSS

Este componente usa Embla (uma biblioteca de carrossel em JavaScript). Refaça-o como carrossel 100% CSS dentro de @supports, mantendo o Embla como fallback quando o navegador não for compatível com os pseudo-elementos novos.

Use: scroll-snap para alinhar os slides, ::scroll-button(left)/(right) para os botões anterior e próximo, ::scroll-marker() para os pontos indicadores e :target-current para marcar o slide ativo.

Critérios de aceite (auditáveis via Chrome MCP): - navegação completa por Tab e setas do teclado, sem armadilha de foco; - os pontos e botões são anunciados por leitor de tela (semântica tablist); - o botão "anterior" fica desabilitado no primeiro slide e o "próximo" no último; - onde o suporte não existe, o Embla assume e a UX não quebra; - me diga quantos KB de JavaScript deixaram de ser carregados no caminho CSS.

/* Polimentos de plataforma como enhancement: sem suporte, degradam limpos */ .btn { border-radius: 12px; corner-shape: squircle; /* sem suporte -> arredondado normal */ text-box: trim-both cap alphabetic; /* sem suporte -> texto normal */ }

.grid { display: grid; gap: 1rem; column-rule: 1px solid var(--border); /* divisória no gap, sem border hack */ row-rule: 1px solid var(--border); }

/* Ordem de foco segue a ordem visual reordenada por 'order' */ .toolbar { display: flex; reading-flow: flex-visual; }

Folha de consulta: cada recurso, a faixa em julho de 2026 e como pedir ao agente

Faça você mesmo: três melhorias autoguardadas e a política, em 20 minutos

Duas listas e uma data, em dez minutos. Primeira lista: os fluxos do seu projeto que têm teste automático e os que não têm, escritos lado a lado sem constrangimento. Segunda lista: as bibliotecas de interface que você instalou, com a data do último release de cada uma, copiada do registro npm. A data é a da próxima vez em que você relê a política de adoção, marcada no calendário antes de fechar o editor. Sem as duas listas e a data, a qualidade do seu projeto depende da sua memória.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 26. Seguranca

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/seguranca.ts`

Segurança de interface: as falhas que nascem na tela

As quatro falhas que o código gerado por IA mais reproduz no frontend, e como fechar cada uma. Você vai guardar a sessão do jeito certo, limpar o texto que vem de fora, tirar toda chave do navegador e conferir permissão na camada que toca o banco.

A escada de adoção, do degrau mais barato ao mais caro: sessão em cookie httpOnly, Zod nas duas pontas, DOMPurify na gravação, permissão revalidada na camada que toca o dado e política de origem de conteúdo.

Os dois momentos de limpar HTML vindo de fora: na gravação existe um ponto único, por onde todo texto passa, e toda tela recebe o conteúdo já limpo; na exibição cada tela precisa lembrar, e a tela nova escrita no ano seguinte é justamente a que não lembra.

Os dois golpes lado a lado: na injeção, o código do estranho roda dentro da página e lê a sessão; na falsificação de requisição, o código nem entra: o navegador é induzido a usar a credencial que já carrega. Cada golpe tem as próprias defesas, e resolver um deixa o outro inteiro.

gitleaks e npm audit: as duas varreduras que rodam antes de enviar.

O que é: `gitleaks` é um programa único, escrito em Go, que lê o repositório inteiro procurando padrões de chave e token. `npm audit` já vem com o npm e compara as suas dependências com o banco público de vulnerabilidades.

O que vem dentro do gitleaks: mais de 160 regras prontas para chaves da AWS, Stripe, OpenAI, GitHub e outros serviços, um modo que varre o histórico inteiro e um arquivo `.gitleaksignore` para marcar o falso positivo.

Quando usar: os dois, em toda alteração, antes de enviar. Quando não usar: nunca deixe de usar, porque os dois terminam em segundos num projeto de frontend.

Passo a passo:

1. Instale o gitleaks pelo gerenciador do seu sistema, com `winget install gitleaks` no Windows ou `brew install gitleaks` no Mac. 2. Rode `gitleaks detect --source .` na raiz do projeto. Resultado esperado: uma lista vazia, ou o arquivo e a linha exata onde a chave está. 3. Rode `npm audit --omit=dev`. Resultado esperado: a contagem de vulnerabilidades por severidade. 4. Corrija o que der com `npm audit fix` e leia o restante caso a caso, porque nem toda vulnerabilidade em dependência de build afeta o pacote publicado. 5. Coloque os dois num gancho de pré-envio com `husky`, para que eles rodem sozinhos. Resultado esperado: o envio é recusado quando o gitleaks encontra alguma coisa.

O erro que todo iniciante comete: rodar o gitleaks só na alteração atual. A chave que você apagou ontem continua no histórico, e é lá que ela é encontrada. Use `gitleaks detect` sobre o repositório inteiro pelo menos uma vez.

Defesa em profundidade: a tela cuida da experiência, a decisão de acesso é revalidada na camada que toca o dado, e o próprio banco reforça quem é dono do registro. Quebrar a camada da tela não dá acesso ao dado.

As cinco checagens de segurança em código gerado, com o critério de aprovação

Implemente um fluxo de login por e-mail e senha em Next.js 16 (App Router) usando uma biblioteca de autenticação estabelecida. Requisitos de segurança, obrigatórios:

- Sessão em cookie httpOnly + Secure + SameSite=Lax. NUNCA use localStorage para token de sessão. - Nenhum segredo no cliente: chave de assinatura e credenciais ficam em variáveis de servidor (sem prefixo NEXT_PUBLIC_). - Proteja /painel revalidando a sessão no Server Component E na camada que lê o dado. Não confie apenas na

camada de entrada. - Confira a entrada no servidor, reaproveitando a mesma descrição de dados usada no formulário. - Política de origem de conteúdo, nosniff e HSTS na configuração central.

Depois de escrever o código, faça uma REVISÃO DE SEGURANÇA do próprio resultado: liste cada risco de injeção, falsificação de requisição, vazamento de segredo e desvio de autorização que possa ter restado, e como você o mitigou. Se algo não puder ser garantido, diga explicitamente em vez de assumir que está seguro.

As quatro decisões de segurança e o que cada uma faz com o seu projeto

Faça você mesmo: auditar e corrigir um componente vulnerável em 20 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 27. Seogeo

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/seogeo.ts`

Ser encontrado por buscador e por IA

Como fazer a sua página aparecer no Google e ser citada por ChatGPT, Gemini e Perplexity. Você vai escolher a forma de montar a página, escrever os metadados, o sitemap e os dados estruturados, e medir o tempo de carregamento no celular real de quem visita.

Três caminhos entregam o conteúdo pronto, e a escolha é feita rota por rota. A geração estática grava o arquivo na hora do build e serve o mesmo HTML para todo mundo, que é o certo para blog, página de curso e ficha de produto que muda pouco. A montagem no servidor refaz o HTML a cada pedido, que é o certo para preço de passagem na LATAM ou estoque no Mercado Livre. Componentes de servidor rodam no servidor por padrão e mandam ao navegador só o que precisa de interação, que é o comportamento nativo do Next.js App Router.

O fluxograma abaixo transforma essa escolha em duas perguntas.

A stack de descoberta recomendada: Next.js App Router, generateMetadata, os arquivos sitemap.ts e robots.ts nativos, JSON-LD escrito à mão e o pacote web-vitals. Quatro das cinco peças já vêm dentro do framework.

Antes e depois de preencher o openGraph: sem ele o link colado numa conversa vira uma linha cinza com o endereço cru; com ele, vira cartão com imagem de 1200 por 630, título e uma linha de descrição.

A anatomia de uma página legível por máquina: quatro linhas no cabeçalho (título, descrição, canônico e JSON-LD) e três partes no corpo (um h1, a cápsula de resposta logo abaixo e os h2 em ordem).

Ser citado por assistente depende de quatro coisas, e nenhuma delas é atalho. A primeira é afirmação verificável, porque número com fonte é citado e adjetivo não. A segunda é a cápsula de resposta, que abre cada seção respondendo direto à pergunta do título. A terceira é autoria verificável, com pessoa real, credencial e organização declaradas na página e repetidas no JSON-LD de artigo. A quarta é a desconfortável: estar bem posicionado na busca orgânica costuma ser pré-condição para ser citado, porque é de lá que vem a maior parte das fontes usadas nas respostas geradas, segundo o levantamento da seoClarity de 2025.

A cápsula de resposta é a que rende mais por hora investida. Em todo conteúdo importante, abra a seção com a pergunta real como título e responda na primeira frase, em texto simples que se sustente sozinho se for arrancado da página. O Google monta a apresentação rica a partir dela e o assistente extrai exatamente aquele trecho como citação. Você escreve uma vez e ganha nos dois canais.

Exemplo do dia a dia. Um restaurante que troca o título "Nosso cardápio" por "Quanto custa um rodízio de pizza na Vila Madalena?" e responde na primeira linha "O rodízio custa R$ 89 por pessoa de terça a quinta e R$ 109 na sexta e no sábado" acabou de escrever uma cápsula. O mesmo vale para uma página de banco que responde "Qual é o prazo para contestar uma compra no cartão?" antes de contar a história da instituição.

Nunca use acento no endereço de uma página. `/categoria/joias-prata` funciona; `/categoria/jóias-prata` quebra em silêncio. O acento vira uma sequência codificada, o endereço oficial deixa de bater com o servido, links externos chegam corrompidos e a página entra no índice como cópia. O texto que a pessoa lê é sempre acentuado; o endereço, o identificador e a chave são sempre em letras simples. Gere o endereço uma vez, no servidor, a partir do título, com uma função de slug.

As quatro decisões de descoberta e o que cada uma faz com o seu projeto

Faça você mesmo: a camada de descoberta de um artigo, em 25 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 28. Stacks complementares

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/stacks-complementares.ts`

Comprar biblioteca ou construir: a régua de decisão por camada

A recomendação direta de stack, camada por camada: gráfico, movimento, ícone, biblioteca de componente, motor de estilo e base do projeto. Em cada uma você encontra a peça padrão, o motivo de trocar por outra, o guia de instalação passo a passo e o erro que derruba a primeira tentativa. É o módulo que os demais citam quando o assunto reaparece.

A stack recomendada em uma tela: Recharts para gráfico, CSS puro para movimento, Lucide para ícone, shadcn sobre Radix para componente, Tailwind v4 para estilo e Next.js ou Astro para base, conforme o projeto seja aplicação ou conteúdo.

Camada de gráficos: o que autoriza sair do padrão

A regra de renderização vale para qualquer biblioteca: SVG até cerca de mil elementos (nítido e acessível), Canvas para milhares (tempo real) e WebGL/WebGPU acima de cem mil pontos.

Camada 3, ícones. Recomendação: use Lucide e mantenha só ele. O que é: uma coleção de mais de mil ícones desenhados no mesmo traço, distribuída como componentes React. O que vem dentro: os ícones, o controle de espessura de linha, o tamanho por propriedade e a cor herdada do texto ao redor. Quando usar: praticamente sempre. Quando não usar: quando a sua marca já tem uma família de ícones desenhada sob medida.

Guia passo a passo.

1. Instale com `npm install lucide-react`. 2. Importe pelo nome do ícone: `import { ShoppingCart } from "lucide-react"`. 3. Use como componente, com . 4. Deixe a cor por conta do texto ao redor, porque o ícone herda `currentColor` sozinho. 5. Quando o ícone é decorativo ao lado de um texto, acrescente `aria-hidden="true"`. Quando ele é o botão inteiro, escreva `aria-label` no botão.

Resultado esperado: ícones consistentes que trocam de cor junto com o tema, sem nenhuma imagem no repositório. O erro que todo iniciante comete é misturar duas famílias, pegando um ícone do Lucide e outro do Font Awesome porque o segundo tinha o desenho que faltava. A interface passa a parecer montada por duas pessoas diferentes, e ninguém sabe apontar o motivo.

Existem agregadores como o Iconify, que reúnem centenas de milhares de ícones numa chamada só. Use para prototipar e evite no que vai ao ar, justamente porque eles desfazem a consistência que a escolha única garante. Em ilustração e avatar, confira a licença antes de baixar: o caso mais comum é o conjunto gratuito para uso pessoal e cobrado no uso comercial, e a descoberta chega tarde. O regime completo de licença de ícone, ilustração e fonte tem dono próprio no módulo "Vestir a marca na interface: ícone, ilustração, tipografia e licença".

Camada 5, motor de estilo. Recomendação: use Tailwind v4. O que decide essa camada é onde o custo do estilo é pago. O modelo antigo calcula o estilo no navegador de quem visita, e cobra do celular dele. O modelo atual gera o arquivo de estilo no momento de publicar, e cobra uma vez só, na sua máquina.

O Tailwind v4 é o padrão desse segundo grupo. Ao lado dele convivem o UnoCSS, que é um motor atômico configurável, o Open Props, que é um conjunto de tokens prontos adotável aos poucos, e as opções tipadas vanilla-extract e StyleX, que derrubam o build quando você erra o nome de um token. Recomendação direta: comece com Tailwind v4; se o seu projeto exige que errar o nome de um token quebre a compilação, troque por vanilla-extract. Trocar de motor no meio de um projeto que já roda bem costuma custar mais do que rende. O catálogo do que o navegador entrega sem motor nenhum está no módulo "Apêndice: o CSS que chegou depois do treinamento do agente".

Camada 6, a base do projeto. Recomendação: use Next.js para aplicação e Astro para site de conteúdo. É a única camada em que trocar significa reescrever. No mundo React existem ainda o React Router, forte em formulário e carregamento de dado, e o TanStack Start, com tipagem de ponta a ponta. Fora do React há SvelteKit, Nuxt para Vue e duas apostas de desempenho extremo. Antes de olhar gosto pessoal, olhe continuidade: confira se a linha que você vai adotar ainda recebe correção de segurança e quem publica essa correção. Base sem correção é problema com data marcada, mesmo funcionando hoje.

A tabela abaixo traz o estado das versões em 15 de agosto de 2026, para você saber o que instalar hoje.

Camada de base: o estado das versões em 15 de agosto de 2026

Cada escolha, o que você ganha e o que você paga

Um detalhe muda a conta de tudo o que você instala. A biblioteca escolhida quase nunca chega sozinha: ela traz as próprias dependências, que trazem outras, e a árvore que o seu projeto carrega fica ordens de grandeza maior que a lista de peças que você escolheu. Rode `npm ls --all | wc -l` num projeto React comum e veja o número. Nada nesse processo pede sua aprovação, porque a instalação é um comando de uma linha.

Dois efeitos saem daí. O primeiro é de licença: uma peça com licença permissiva arrasta outra com termos restritivos, e você descobre quando alguém audita o projeto. O segundo é de segurança: cada peça da árvore é uma porta, e boa parte dos incidentes recentes de cadeia de suprimento entrou por um pacote periférico que ninguém escolheu de propósito.

O hábito que resolve cabe em dois comandos. Rode `npm audit` de vez em quando e leia o resultado. E, antes de aceitar qualquer pacote sugerido pelo agente, abra a página dele no registro npm e confira que existe. Modelos inventam nome de pacote com confiança total, e existe uma classe de ataque que registra justamente os nomes que os modelos costumam inventar, esperando quem instale sem olhar. A frase que fecha essa porta está no prompt abaixo.

Prompt: faça a IA escolher a stack justificando o trade-off

Faça você mesmo: montar a stack das seis camadas, em 20 minutos

Abra o `package.json` do seu projeto e leia a lista de dependências com três marcadores na mão. Marque de vermelho toda peça cuja página no registro npm não tem release nos últimos doze meses. Marque de amarelo toda peça com licença comercial. Marque de verde toda peça cuja função o navegador já cobre hoje, do tipo carrossel, acordeão, posicionamento de menu e animação de altura. Os vermelhos são o que você vai manter sozinho quando quebrar. Os amarelos são a conta anual. Os verdes são o que sai na próxima limpeza. Três números numa linha, dez minutos, e você sabe o estado do seu projeto.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 29. Stripe

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/stripe.ts`

Stripe: quando a interface reduz o custo de conformidade

Como montar um checkout em que o número do cartão nunca toca o seu código, e como reproduzir os padrões visuais e de documentação da Stripe sem copiar marca nem código. Sai daqui com o formulário funcionando e o gradiente em CSS puro.

Antes e depois da Promise do loadStripe: dentro do componente, três renderizações baixam o script três vezes; no topo do arquivo, a promessa nasce uma vez e o script é baixado uma vez.

Faça você mesmo: o checkout seguro por padrão em 15 minutos

Tipografia, e a autoridade pela restrição:

No redesign de julho de 2020, a Stripe trocou a Camphor pela Söhne (Kris Sowersby, Klim Type Foundry, 2019; Fonts In Use, 2020). O brief é "a memória da Akzidenz-Grotesk filtrada pela Helvetica", uma neogrotesca que quer ser infraestrutura neutra e clara. A Söhne Collection levou o Red Dot Best of the Best em 2020 (red-dot.org, projeto 49017), o que dá a medida do ativo comprado, junto com o custo de licença.

Uma única família variável (sohne-var) cobre toda a interface, e as escolhas-chave aparecem no CSS público: títulos em peso 300 com tracking negativo (-0,020em ou mais apertado), corpo em peso 400 a 16px e valores monetários em monospace. O peso 300 é contraintuitivo num setor que associa solidez a peso gráfico. A Stripe sussurra em light onde as outras gritam em bold, e parece mais segura por isso: "autoridade que vem da restrição" (DESIGN.md, 2024).

As cinco escolhas que produzem o tom inteiro, e o que cada uma custa

Prompt: hero com gradiente aurora em CSS puro

O degrau seguinte é mais radical. Quando o leitor está autenticado, as amostras vêm preenchidas com a sua chave de teste real e o código copia-e-roda sem editar uma linha; o visitante anônimo vê placeholder com instrução de login. A assimetria elimina dois atritos de uma vez, que são abrir o Dashboard em outra aba e errar a chave ao colar (docs.stripe.com/development/get-started; apidog.com, 2024-2026). Acima disso ainda existe o Stripe Shell, que roda a chamada na própria página, de modo que a escada do anônimo até o primeiro pagamento de teste tem quatro degraus e nenhum deles exige sair da documentação.

O princípio se aplica mesmo sem personalização por usuário: ofereça o exemplo na linguagem certa, com click-to-copy, e deixe claro qual é o único valor que o leitor precisa trocar. Cada edição manual exigida é uma chance de ele desistir.

Crie um campo de pagamento React acessível com quatro estados visuais e semânticos: vazio, digitando, erro e sucesso.

Contrato obrigatório: - Label associado ao input (for/id ou aria-label); nunca dependa só do placeholder. - Foco visível no estado de digitação: outline com offset, jamais outline:none sem substituto. - aria-invalid="true" no estado de erro, removido nos demais. - aria-describedby ligando o input à mensagem de erro. - Região aria-live="polite" que anuncia a mensagem de erro sem roubar o foco; role="alert" só entra no estado de erro.

Regras: - Estado visual e estado semântico andam juntos: a borda vermelha SEMPRE vem acompanhada de aria-invalid e mensagem anunciada. - Cores por token (var(--border), var(--accent), var(--card), var(--text)) e tons semânticos de status; contraste AA. - Não use cor sozinha para indicar erro: combine com ícone ou texto.

O ciclo que governa os cinco prompts: você especifica o padrão e a versão ética, o agente gera em segundos, você audita; e o que reprova volta como regra nova do enunciado. As guardas viajam dentro do prompt porque regra que mora só na cabeça do autor não sobrevive ao segundo sprint.

Prompt: Documentação de duas colunas com copiar acessível

Prompt: Hero premium calma com gradiente leve

Prompt: Command palette acessível

Síntese final:

O custo de produzir código despencou, e o seu trabalho é o que a IA não faz: especificar o padrão certo, fixar tokens, exigir acessibilidade (contraste AA, foco visível, prefers-reduced-motion) e traçar a linha ética. A paleta de comandos ilustra por que a especificação vale mais que a geração. Ela parece simples e é o componente que mais erra acessibilidade, porque o foco do DOM precisa ficar no input enquanto a seleção anda pela lista via aria-activedescendant, o Esc precisa sempre fechar e devolver o foco ao gatilho, e o item destacado precisa combinar cor com peso ou borda. Acertar esse padrão é o que separa a sensação de produto premium de uma armadilha de teclado.

Faça você mesmo: a página de documentação estilo Stripe, auditada

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 30. Tabelas grids

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/tabelas-grids.ts`

Tabela pesada e análise no navegador

Você aprende a escolher entre os quatro degraus de uma tela de dados, do simples ao desenho na placa de vídeo, com o guia de instalação de cada biblioteca e o teste que diz em qual degrau o seu caso está. Fecha com a análise que roda dentro do navegador de quem lê, sem servidor de dados, e com os quatro itens que todo número exibido precisa carregar.

TanStack Table com TanStack Virtual, a ficha completa. O que é: a dupla que sustenta o segundo degrau. O TanStack Table cuida da lógica da tabela, sem desenhar nada; o TanStack Virtual desenha apenas as linhas visíveis. O que vem dentro do primeiro: definição de colunas, ordenação, filtro, agrupamento, paginação, seleção de linha e coluna fixa. O que vem dentro do segundo: o cálculo de quais linhas cabem na janela e a altura falsa que mantém a barra de rolagem no tamanho certo.

Quando usar: qualquer tabela acima de alguns milhares de linhas em que você quer aparência própria. Quando não usar: cinquenta linhas, onde o puro resolve e nasce acessível.

Guia passo a passo.

1. Instale com `npm install @tanstack/react-table @tanstack/react-virtual`. 2. Defina as colunas num array, cada uma com `accessorKey` e `header`. 3. Crie a tabela com `useReactTable({ data, columns, getCoreRowModel: getCoreRowModel() })`. Sem esse último item, a tabela sai vazia e sem erro. 4. Guarde `data` e `columns` fora do componente ou dentro de `useMemo`. Um array novo a cada render faz a tabela remontar sem parar. 5. Crie um contêiner rolável com altura fixa e guarde a referência dele. 6. Chame `useVirtualizer({ count, getScrollElement, estimateSize: () => 44 })` e renderize apenas as linhas devolvidas por `getVirtualItems()`. 7. Desloque cada linha com `transform: translateY(...)` a partir do `start` de cada item virtual.

Resultado esperado: cem mil linhas com rolagem fluida e a semântica de tabela preservada. Dois erros derrubam a estreia: o passo 4, que produz uma tabela piscando sem parar, e esquecer a altura fixa do contêiner no passo 5, que faz o virtualizador calcular zero linhas visíveis.

Um efeito colateral merece aviso: a busca do próprio navegador, o `Ctrl+F`, deixa de encontrar o que não foi desenhado. Se a sua tela depende disso, ofereça um campo de busca próprio.

Faça você mesmo: o teto real da sua tela, em 20 minutos

Exercício: registre só o que a tela usa e prove o ganho de peso

Antes de escrever qualquer código: consulte a documentação e o registro de pacotes para descobrir qual é a linha ESTÁVEL corrente da biblioteca de tabela deste projeto e qual linha está em versão de teste. Cite os números que você encontrou, com a data da consulta.

Depois, nesta tela: 1. Registre explicitamente apenas os recursos que ela usa (ordenação e paginação). Não importe filtro, agrupamento nem tabela dinâmica. 2. Rode o analisador de pacote antes e depois e me mostre, em kilobytes, quanto o corte economizou no arquivo desta rota. 3. Extraia a configuração comum para uma base reutilizável, de onde as outras tabelas do produto herdem o mesmo comportamento.

Critérios de aceite: - Você declarou a linha estável e a linha de teste com a data da consulta, em vez de assumir de memória. - O relatório de pacote traz o número antes e depois, não uma estimativa. - O comportamento visível (ordenar coluna, paginar) fica idêntico ao anterior. - Se a versão que você precisaria usar estiver em teste, diga isso em vez de adotá-la em silêncio.

Exercício: a mesma tabela nos dois caminhos, auditadas lado a lado

Os dois caminhos da configuração escrita por agente sobre uma biblioteca de superfície extensa. De memória, a lacuna vira invenção plausível que só falha na execução; consultando a documentação da versão instalada (ou o contexto do fornecedor), a propriedade é conferida contra a interface real, e o autocompletar fecha a dúvida em segundos.

DuckDB-Wasm com Mosaic, a ficha completa. O que é: o DuckDB é um banco analítico que roda por inteiro dentro do navegador, na versão compilada para WebAssembly. O Mosaic é a camada que coordena vários elementos visuais em cima dele. O que vem dentro: SQL completo, leitura de arquivo Parquet por partes, filtro cruzado entre gráficos e a coordenação que transforma um arraste no histograma em consulta única.

Quando usar: exploração de conjunto grande, painel que precisa filtrar rápido e caso em que o dado não deve sair da máquina de quem lê. Quando não usar: uma tabela de duzentas linhas, onde carregar o motor pesa mais que o dado.

Guia passo a passo.

1. Converta o seu CSV para Parquet, que é o formato organizado por coluna e comprimido. Um comando do próprio DuckDB faz isso. 2. Hospede o arquivo Parquet num armazenamento estático que aceite pedido parcial. 3. No projeto, instale `npm install @duckdb/duckdb-wasm`. 4. Inicialize o motor e mostre um estado de carregamento visível, porque a inicialização leva alguns segundos. 5. Registre o arquivo remoto e faça a primeira consulta com `SELECT count(*) FROM 'https://.../dados.parquet'`. 6. Confira na aba de rede que o navegador baixou apenas alguns pedaços, e não o arquivo inteiro.

Resultado esperado: um filtro que atualiza três gráficos em poucos milissegundos, sobre milhões de linhas, sem nenhum servidor de dados. O erro que todo iniciante comete é carregar o Parquet inteiro na memória com `fetch` antes de entregar ao motor, o que joga fora justamente a leitura por partes que motivou a escolha.

O gargalo muda de lugar quando você faz isso: deixa de ser o desenho da tela e passa a ser a consulta. Melhorias de desempenho passam a depender de como o dado foi organizado antes, e trocar de biblioteca de gráfico não move o número.

O calendário do motor traz um prazo. O DuckDB 1.5.0, de codinome Variegata, saiu em 9 de março de 2026 com o tipo VARIANT, para dado semiestruturado sem achatamento prévio, e geometria nativa no núcleo. Em agosto de 2026 o cliente que roda no navegador está estável na 1.5.5, construído sobre o DuckDB 1.5.4. A linha anterior, a 1.4 de codinome Andium, é a de suporte estendido e recebe atualizações até setembro de 2026. Se o seu projeto está nela, migre agora, com teste de regressão das consultas, enquanto ainda chega correção.

Exercício: explorador no navegador, sem servidor de dados

Exercício: relatório calculado antes de publicar

O espectro de quando o dado é processado. À esquerda, antes de publicar, com retrato estático: o pacote de conselho, leve e imprimível. No meio, a cada visita, consultando a fonte: marketing diário e filtro ao vivo. À direita, em fluxo contínuo: cotação e telemetria. Cada projeto ocupa um ponto do espectro, e a pergunta é qual.

Exercício: relatório por cliente com separação de dados

Os quatro itens que sustentam um número de painel executivo: origem, data de apuração, responsável e definição, acessíveis em até dois cliques. Faltando qualquer um, o número vira afirmação sem lastro e contamina a leitura do resto do relatório.

Faça você mesmo: prove a proveniência de três números, em 15 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 31. Usecase apple

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/usecase-apple.ts`

Apple e Temu: os dois extremos do engajamento

Duas marcas ocupam as pontas do mesmo eixo: uma vende pela contenção, a outra pelo estímulo. Traz o limite entre copiar o método de uma marca admirada e copiar o que pertence a ela, a ficha completa das ferramentas de animação com recomendação direta, e o teste de três perguntas que separa urgência verdadeira de padrão enganoso.

A ficha do Lucide em cinco quadros: o que é (mais de mil ícones em traço, licença aberta), o que vem dentro (um componente por ícone e as propriedades size, strokeWidth e color), quando usar (em qualquer projeto, no lugar do ícone proprietário), quando não usar (ícone de várias cores pede SVG próprio) e o detalhe que separa o caprichado do amador, que é o alinhamento de tamanho e espessura com o texto ao lado.

Faça você mesmo: audite uma página imersiva pela rede, em 10 minutos

Demonstração viva do reveal ao rolar: três cartões sobem em sequência escalonada, como animation-timeline: view() faz ao entrar na viewport. Sem suporte ou com movimento reduzido, já aparecem prontos.

Prompt: Micro-interações que respeitam o usuário

Crie um conjunto de micro-interações CSS para cards e botões, estilo Apple (sutis, com profundidade), que rodem na GPU e respeitem a preferência de movimento.

Comportamento: - Hover/focus de card: leve elevação (translateY de -4px) e sombra mais suave; transição apenas de transform e box-shadow/opacity (compostas na GPU), nunca de width/height/top. - Botão: estado :hover, :active e :focus-visible com foco SEMPRE visível (anel de foco com contraste AA); o :active dá um leve scale(0.98) para sensação tátil. - Duração curta (150-250ms) e easing suave (ease-out na entrada).

Acessibilidade (obrigatória): - Tudo dentro de @media (prefers-reduced-motion: no-preference), ou seja, as transições só existem se o usuário NÃO pediu redução; sob reduce, estados mudam sem movimento. - :focus-visible nunca é removido; contraste do anel >= 3:1.

Tokens de cor; sem asset/fonte proprietária; contraste AA. Comente por que animar transform/opacity é barato e width/top é caro.

Prompt: Pipeline de mídia responsiva e performática

O storytelling converte porque tem funil atrás dele, e a página de produto é a dobradiça entre o bloco que constrói marca, com teaser e keynote, e o bloco que converte, com trade-in e financiamento tirando a fricção do preço antes de o cliente procurar o valor em outro lugar. Some a isso a consistência cross-superfície, com site, loja, embalagem e keynote na mesma estética, e cada detalhe reforça a percepção premium.

Os mecanismos que produzem esse engajamento são reproduzíveis sem um único asset proprietário: o whitespace estratégico que faz o olho tratar o produto como assunto, com uma ideia por tela; a micro-interação de hover que devolve profundidade e confirma que o elemento respondeu; a copy de benefício com tipografia gigante como elemento gráfico; e a especificação virando herói numa tela inteira. Mostrar antes de contar resume o conjunto, e é o contrário do que o agente entrega por padrão quando você pede uma página de produto sem roteiro. No meio do funil fica o comparador lado a lado, no padrão good-better-best, em que o realce do topo é estético e nenhuma coluna esconde o que entrega.

Custo e risco por decisão de inspiração visual

Quatro mecânicas centrais fazem o trabalho e se encaixam em cadeia. A roleta entra primeiro, porque a incerteza é o gancho mais barato de instalar e o único que dispensa qualquer benefício real de partida. O presente surpresa estende esse gancho no tempo, com uma caixa que revela prêmio diferente a cada visita e mantém a expectativa viva sem prometer nada específico. O cronômetro de oferta converte a expectativa em decisão, porque a contagem regressiva transforma vontade difusa em pressa. O check-in diário fecha o circuito junto com a indicação de amigos, premiando quem abre o aplicativo todo dia e quem traz alguém, o que transfere ao próprio usuário parte do custo de aquisição do cliente seguinte.

O efeito das quatro juntas é maior que a soma delas, e isso pesa na hora de escolher: a roleta sozinha rende a visita de hoje, e combinada com a sequência diária ela passa a cobrar a visita de amanhã, porque a segunda mecânica cria custo em abandonar o que a primeira entregou. Ligar a recompensa surpresa e recusar a sequência que pune a falta é escolher deliberadamente um efeito menor, e é uma escolha defensável desde que você a escreva no arquivo de decisões do projeto.

A disciplina do curso vale aqui com mais razão que em qualquer outro trecho: animação só sob `prefers-reduced-motion: no-preference`, com o estado base sempre visível, porque o resultado da roleta precisa existir com ou sem o giro; anime só `transform` e `opacity`, que rodam no compositor; e trate a microinteração como confirmação do que acabou de acontecer, em vez de laço que mantém a pessoa no ciclo. Movimento que existe só para prender o usuário é padrão enganoso com outro nome, e o teste é verificável: desligue a animação e veja se a informação continua lá. Cada efeito de cassino da Temu é uma microinteração comum, e a tabela abaixo mostra qual técnica produz cada um e qual condição o torna verdadeiro. É a terceira coluna que decide se o item entra no produto.

Peça assim: prova social verdadeira

Uma armadilha aparece na primeira tentativa de fazer isso com agente. Pedir à IA que deixe a tela "mais persuasiva como a Temu", sem guardrail, devolve cronômetro que reinicia, "resta 1" fixo e confirmshaming por padrão, porque é isso que o modelo viu na média da web, e o resultado chega com aparência de trabalho pronto. O conserto é carimbar as condições no próprio pedido: números reais, sem reinício, sem confirmshaming, cancelamento fácil, e animação sob `prefers-reduced-motion` com estado base visível. Sem esse carimbo, quem revisa a entrega gasta mais tempo removendo padrão enganoso do que teria gasto especificando o pedido.

Custo e risco por mecanismo de engajamento

Faça você mesmo: audite e corrija uma página de oferta, em 30 minutos

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 32. Usecase latam

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/usecase-latam.ts`

LATAM: a interface de uma companhia aérea, da borda da rede ao programa de fidelidade

Como ler a tecnologia de um site pela evidência que ele publica sem querer, o que a interface de uma companhia aérea ensina sobre preço vivo, marca em tokens e pós-venda, e onde a persuasão esbarra na lei de defesa do consumidor no Brasil. Traz a ficha completa do i18n e da formatação de números em português, com passo a passo numerado, e o fluxograma que decide entre página montada na hora e página guardada.

A paleta conta a fusão: azul da LAN somado ao vermelho da TAM resulta no índigo institucional; o coral traz paixão; o gradiente índigo-coral atravessa faixas de contraste muito diferentes.

Escolhido o dia no calendário, a decisão seguinte estreita mais o funil, e é ali que o upsell aparece mostrando o que a tarifa barata deixa de fora. A Economy é vendida em quatro tarifas: Basic (só item pessoal), Light (mão mais milhas), Plus (assento mais bagagem) e Top (flexível, com mudança de voo); acima delas ficam Premium Comfort e Premium Business (LATAM Help Center, 2026). A apresentação é uma matriz comparativa good-better-best, e a decisão editorial que a torna defensável é mostrar abertamente, linha por linha, o que você não leva no Basic, porque comparar é direito do usuário e a tabela existe para revelar.

A mesma matriz vira dark pattern com uma mudança pequena: basta a tarifa barata omitir a restrição e o passageiro descobrir no balcão que a mala de mão nunca esteve incluída. Good-better-best sustenta preço quando cada coluna declara com clareza o que entrega e o que tira, e a diferença entre vender mais e enganar cabe na franqueza da própria tabela, antes de qualquer botão maior na coluna mais cara.

Ficha do next-intl, a biblioteca que resolve idioma em projeto Next.js.

O que é: uma biblioteca que guarda as suas frases em arquivos JSON, um por idioma, e entrega a frase certa conforme o endereço que a pessoa abriu.

O que vem dentro: um arquivo de mensagens por idioma; a função useTranslations, que busca a frase pela chave; o roteamento por prefixo de idioma, que cria /pt e /en sozinho; e a integração com o Intl para número, moeda e data.

Quando usar: assim que existir um segundo idioma, ou quando você já sabe que vai existir. Quando não usar: num site de uma página só em português, onde o Intl nativo resolve os formatos sem biblioteca nenhuma.

Passo a passo até a primeira frase traduzida:

1. Instale com npm install next-intl. Resultado esperado: a instalação termina sem tocar em nenhuma outra dependência.

2. Crie messages/pt.json e messages/en.json, com a mesma estrutura de chaves nos dois. Resultado esperado: dois arquivos espelhados, do tipo {"busca": {"botao": "Buscar"}}.

3. Configure o middleware de idioma com os locales pt e en, e o padrão em pt. Resultado esperado: abrir a raiz do site leva para /pt automaticamente.

4. No componente, escreva const t = useTranslations("busca") e use t("botao") no lugar do texto. Resultado esperado: nenhuma frase escrita direto no JSX.

5. Formate valores com o Intl, e nunca à mão: new Intl.NumberFormat("pt-BR", {style: "currency", currency: "BRL"}).format(624.87) devolve R$ 624,87 com vírgula decimal. Resultado esperado: o mesmo código serve para dólar trocando duas palavras.

6. Escreva um teste que percorre as duas árvores de mensagens e falha se uma chave existir num idioma e faltar no outro. Resultado esperado: a frase órfã em espanhol nunca chega à sua página.

O erro que todo iniciante comete: traduzir o texto visível e esquecer as frases que só o leitor de tela lê, como o aria-label do botão e o anúncio de aria-live. Foi exatamente esse o defeito flagrado na auditoria da LATAM.

Os três testes: o aviso de assentos só existe quando o número vem do inventário real, a caixa do seguro nasce desmarcada, porque a pré-marcada é o padrão que rendeu multa do Senacon, e a taxa aparece antes do último passo, para a pessoa poder comparar.

Prompt: Busca search-first com autocomplete acessível e i18n

Prompt: Calendário de preços acessível

Prompt: Matriz de famílias tarifárias (good-better-best verdadeiro)

Configure os tokens de marca de uma identidade estilo LATAM em CSS puro, com a disciplina de contraste do gradiente.

1) Tokens em :root: --brand-indigo (institucional), --brand-coral (acento), --brand-gradient (linear-gradient índigo->coral), além de surface, text, text-muted e o token de destaque "menor tarifa" (verde com AA). Defina também [data-theme="dark"] reatribuindo os tokens com contraste AA. - Cada token de marca leva um comentário com seu SIGNIFICADO (índigo = interseção LAN+TAM) e a ressalva de que o hex é aproximação a validar no manual oficial.

2) Separe fonte de MARCA (display, peso forte, só em títulos) de fonte de PRODUTO (system fonts legíveis para corpo). Use font-family isolada (nunca o shorthand font:).

3) Texto sobre o gradiente: como o coral é claro, texto branco quebra o AA. Entregue uma solução: um overlay sólido semitransparente atrás do texto (ou fixar o contraste sobre o ponto mais claro). Comente por que conferir a olho não resolve aqui.

Entregue o bloco :root, o [data-theme="dark"], a regra de tipografia e o padrão de texto sobre gradiente.

Prompt: Barra de progresso de fidelidade ancorada em dado real

Prompt: Escassez de assento verdadeira (ancorada no inventário, CDC)

Copiar da LATAM vs Evitar

Custo e risco por decisão numa venda com data marcada

Exercício prático:

CRIAR, por vibecoding, uma tela de busca e seleção de voo estilo LATAM que seja ética por construção. Dirija o Claude Code com prompts explícitos, sem escrever o código à mão, e entregue quatro peças.

-- Uma busca search-first (FlightSearchBox) com autocomplete acessível que desambigua aeroporto/cidade e anuncia a contagem via aria-live, com TODAS as strings vindas de um dicionário de locale (nenhuma string órfã de outro idioma). -- Um calendário de preços (PriceCalendar) com botão por dia, aria-label completo em pt-BR (vírgula decimal), destaque "Menor tarifa" por cor E texto e dias sem voo desabilitados. -- Uma matriz de tarifas (Basic/Light/Plus/Top) em semântica que revela o que cada tarifa inclui e o que não inclui. -- UM aviso de escassez de assento que só aparece quando o dado mock disser assentos_restantes <= 4, com a fonte do número visível no código (data-fonte) e o texto concordando em número.

Cinco critérios fecham o exercício, e todos são verificáveis na própria tela. 1. Contraste AA confirmado nos dois temas (claro e escuro): teste o verde de "Menor tarifa" e o texto sobre o gradiente de marca. 2. A tela inteira é navegável por teclado, com foco visível em todos os interativos. 3. Nenhuma string fora do locale ativo (rode um teste que falhe se vazar "opciones" ou similar numa render pt-BR). 4. Nenhuma escassez fabricada: remover assentos_restantes <= 4 do mock faz o aviso desaparecer; nenhum opt-out pré-marcado em adicionais. 5. Você consegue explicar, padrão por padrão, por que cada técnica é verdadeira, ancorada em dado real e em

consentimento explícito, sem pressão inventada.

Quando os cinco critérios passarem, você terá provado que sabe auditar uma stack, tratar marca como sistema e separar engajamento verdadeiro de manipulação, no contexto legal brasileiro, onde essa diferença tem nome e número de lei.

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Caderno 33. Usecase mercadolivre

Fonte interna: `src/app/educacao/frontends-com-vibecoding/modulos/usecase-mercadolivre.ts`

Mercado Livre: o funil de um e-commerce em escala continental

Como a maior loja da região organiza a interface, e o que dessa organização você copia sozinho: tokens em três camadas, card de produto que responde a uma objeção por elemento, buy box que reúne a decisão num bloco só e um teto de desempenho que barra o enfeite antes dele entrar. Traz a ficha completa do Tailwind CSS v4 e do pipeline de imagem, com passo a passo numerado.

As três camadas de token em cascata: o token de componente consulta a paleta semântica, que consulta o valor bruto, e o componente na tela nunca escreve o hexadecimal. Para o tema escuro, redefina só a camada do meio; para trocar a marca, edite a de baixo. O erro comum é redefinir a camada de cima, o que obriga a refazer o trabalho a cada tema novo.

Ficha do Tailwind CSS v4.

O que é: uma biblioteca de classes pequenas que você escreve direto no HTML, cada uma fazendo uma coisa só, como p-4 para espaçamento interno ou text-lg para tamanho de letra.

O que vem dentro: um bloco @theme onde você declara os seus tokens e o Tailwind gera as classes correspondentes; variantes de estado como hover:, focus-visible: e dark:; um motor que só inclui no arquivo final as classes que você de fato usou; e suporte a container queries, para o componente reagir ao tamanho do espaço em que ele está, em vez do tamanho da janela.

Quando usar: qualquer projeto em que você escreve as telas, sozinho ou com agente, porque o agente acerta muito mais com classes utilitárias do que inventando nomes de classe. Quando não usar: quando você precisa distribuir um componente para páginas que não têm Tailwind, caso em que o Web Component com Shadow DOM resolve melhor.

Passo a passo até a primeira classe funcionando:

1. Instale com npm install tailwindcss @tailwindcss/vite e adicione o plugin no arquivo de configuração do Vite. Resultado esperado: a instalação termina sem pedir arquivo de configuração separado.

2. No topo do seu CSS, escreva @import "tailwindcss". Resultado esperado: as classes utilitárias passam a funcionar imediatamente.

3. Declare os seus tokens dentro de um bloco @theme, com nomes como --color-acao e --spacing-secao. Resultado esperado: o Tailwind cria sozinho as classes bg-acao, text-acao e py-secao.

4. Escreva uma tela usando só essas classes. Resultado esperado: nenhum valor de cor solto no arquivo, e a identidade toda vindo do bloco @theme.

5. Crie o tema escuro com @media (prefers-color-scheme: dark) redefinindo as variáveis do @theme. Resultado esperado: a tela troca de tema sem você tocar em nenhum componente.

6. Rode o build e abra o CSS final. Resultado esperado: um arquivo pequeno, com apenas as classes usadas.

O erro que todo iniciante comete: escrever a cor direta na classe, do tipo bg-[#3483fa]. Isso funciona e destrói o ganho inteiro, porque a cor volta a morar espalhada nos componentes. Declare no @theme e use o nome.

O mesmo card com e sem ordem de prioridade no pedido: à esquerda, dez elementos com o mesmo peso visual, que é o que o agente devolve quando você não diz a ordem; à direita, imagem no topo, título pequeno, preço atual em corpo grande com o valor cheio riscado acima, parcelamento abaixo e a linha de frete fechando em verde com texto.

Ficha do pipeline de imagem, que é o item da lista acima que mais devolve resultado.

O que é: a combinação de formato moderno, tamanho certo por tela e carregamento adiado que faz a mesma foto pesar um terço do que pesava.

O que vem dentro: a tag , que oferece AVIF, depois WebP, depois JPEG, para o navegador escolher o primeiro que ele entende; o atributo srcset, que lista várias larguras do mesmo arquivo; o atributo sizes, que diz quanto da tela a imagem vai ocupar; e os atributos loading e fetchpriority, que decidem a ordem de download.

Quando usar: em toda imagem de conteúdo. Quando não usar: em ícone e em traço, que pedem SVG, porque SVG escala sem perder nitidez e costuma pesar menos que qualquer foto.

Passo a passo, com uma ferramenta gratuita:

1. Rode npx @squoosh/cli --avif auto --webp auto foto.jpg. Resultado esperado: três arquivos, um por formato, e você compara os tamanhos.

2. Gere as larguras que você precisa, por exemplo 400, 800 e 1200 pixels. Resultado esperado: nove arquivos ao todo, três larguras em três formatos.

3. Escreva a tag com um por formato e um de JPEG no fim. Resultado esperado: navegador novo baixa AVIF, navegador antigo baixa JPEG, e nenhum quebra.

4. Coloque width e height na tag . Resultado esperado: o espaço fica reservado e a página para de pular quando a foto chega.

5. Marque loading="lazy" em tudo que está abaixo da dobra e fetchpriority="high" só na imagem principal. Resultado esperado: a primeira tela carrega antes, porque o resto espera a vez.

6. Abra o painel de rede, recarregue e olhe o peso total. Resultado esperado: um número menor que o de antes, que você anota como o seu novo teto.

O erro que todo iniciante comete: esquecer width e height. Sem eles a página pula quando cada foto chega, o deslocamento de layout estoura, e a pessoa clica no botão errado porque ele se mexeu debaixo do dedo.

A anatomia da tag de imagem, atributo por atributo: um `source` por formato, `srcset` com as larguras, `sizes` dizendo quanto da tela a foto ocupa, `width` e `height` reservando o espaço e `loading` decidindo quem espera a vez.

O que cada decisão de arquitetura de loja compra e cobra

> Critério de uso
> Volte a este caderno quando precisar recuperar um prompt, um caso ou uma decisão específica. Para executar o curso na ordem pedagógica e usar os componentes interativos, abra a edição online.

Referências verificadas

Documentações consultadas em 15 de agosto de 2026. Links de software, licença e norma precisam ser conferidos novamente quando a decisão virar contratação ou publicação.

1. Curso Frontends com Vibecoding, edição consultada em 15/08/2026
https://alexandrecaramaschi.com/educacao/frontends-com-vibecoding

2. Next.js: Image component
https://nextjs.org/docs/app/api-reference/components/image

3. W3C: Web Content Accessibility Guidelines 2.2
https://www.w3.org/TR/WCAG22/

4. W3C: Target Size (Minimum)
https://www.w3.org/WAI/WCAG22/Understanding/target-size-minimum.html

5. W3C: Reflow
https://www.w3.org/WAI/WCAG21/Understanding/reflow

6. web.dev: Container queries
https://web.dev/learn/css/container-queries/

7. MDN: CSS scroll-driven animations
https://developer.mozilla.org/en-US/docs/Web/CSS/Guides/Scroll-driven_animations

8. Motion: useReducedMotion
https://motion.dev/docs/react-use-reduced-motion

9. GSAP: ScrollTrigger
https://gsap.com/docs/v3/Plugins/ScrollTrigger/

10. GSAP: matchMedia
https://gsap.com/docs/v3/GSAP/gsap.matchMedia()/

11. Rive: State machine playback
https://rive.app/docs/runtimes/web/state-machines

12. Three.js: WebGLRenderer
https://threejs.org/docs/pages/WebGLRenderer.html

13. Cloudinary: Responsive images
https://cloudinary.com/documentation/responsive_html

14. Mux: vídeo no design do site
https://www.mux.com/docs/examples/use-videos-in-your-website-design

15. Unsplash License
https://unsplash.com/license

16. Pexels License
https://www.pexels.com/license/