

USER EXPERIENCE

NEXT.JS

Nicoly Almeida

Nicoly Almeida



Front-End Software Engineer

Efí Bank



Professora

Uniesp



Embaixadora

Women Techmakers



Organizadora

Google Developers Group



Vice-presidente

Mulher Tech Sim Senhor



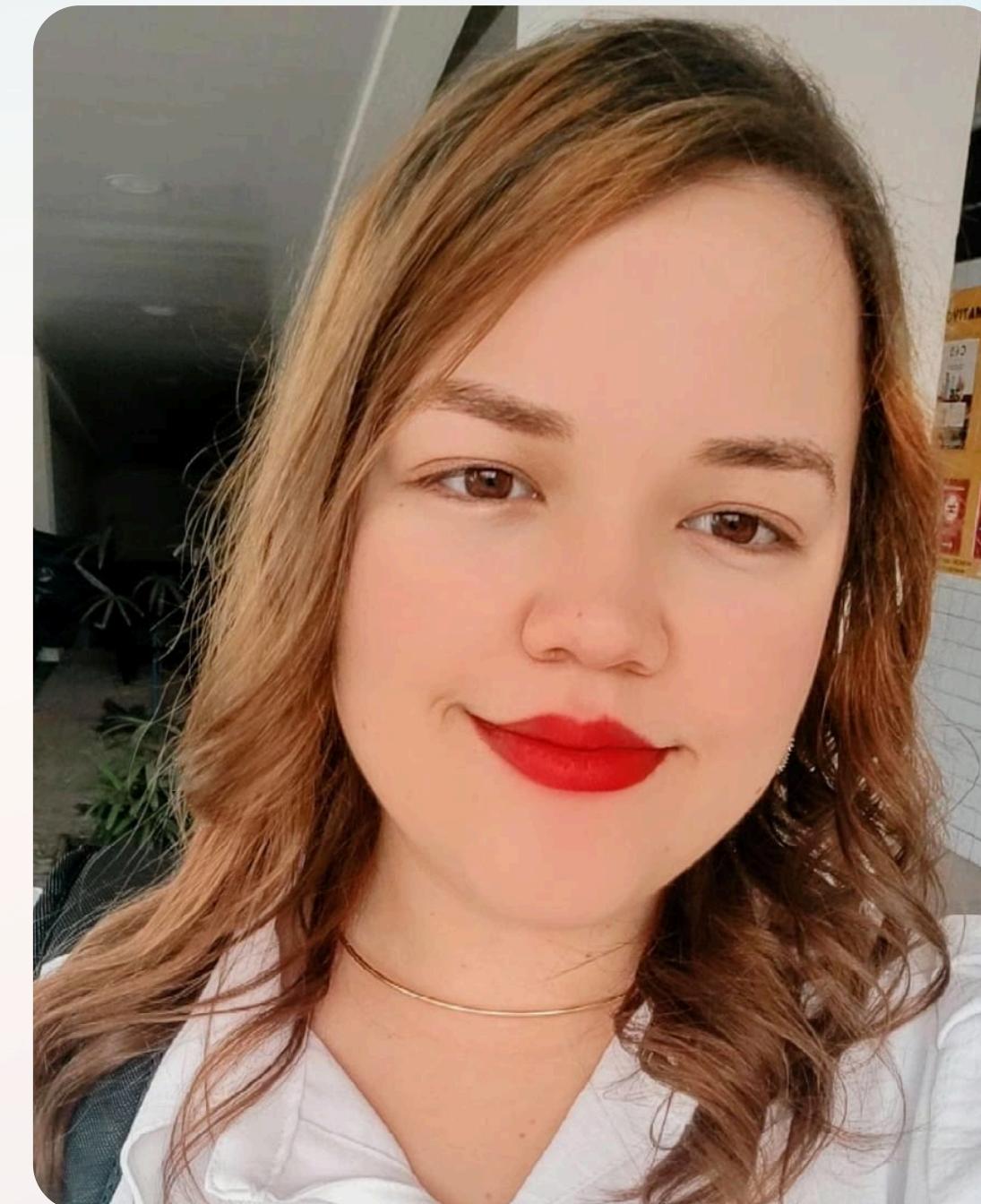
Mestranda

Tecnologia da Informação - IFPB



Especialista

Engenharia de Software - Focus



Vamos entender

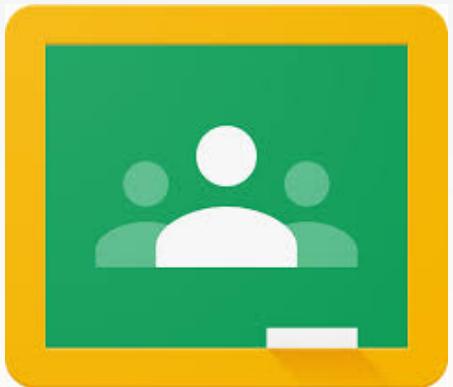
O avanço das aplicações web exige que desenvolvedores frontend:

- Não se limitem à criação de interfaces visuais;
- Garantam experiências intuitivas, fluídas e acessíveis;
- Compreendam o impacto da performance na usabilidade;
- Apliquem boas práticas de UX de forma consciente e sistemática;
- Utilizem frameworks modernos como o Next.js para alinhar tecnologia com experiência.



Nicoly Almeida

Então para começar



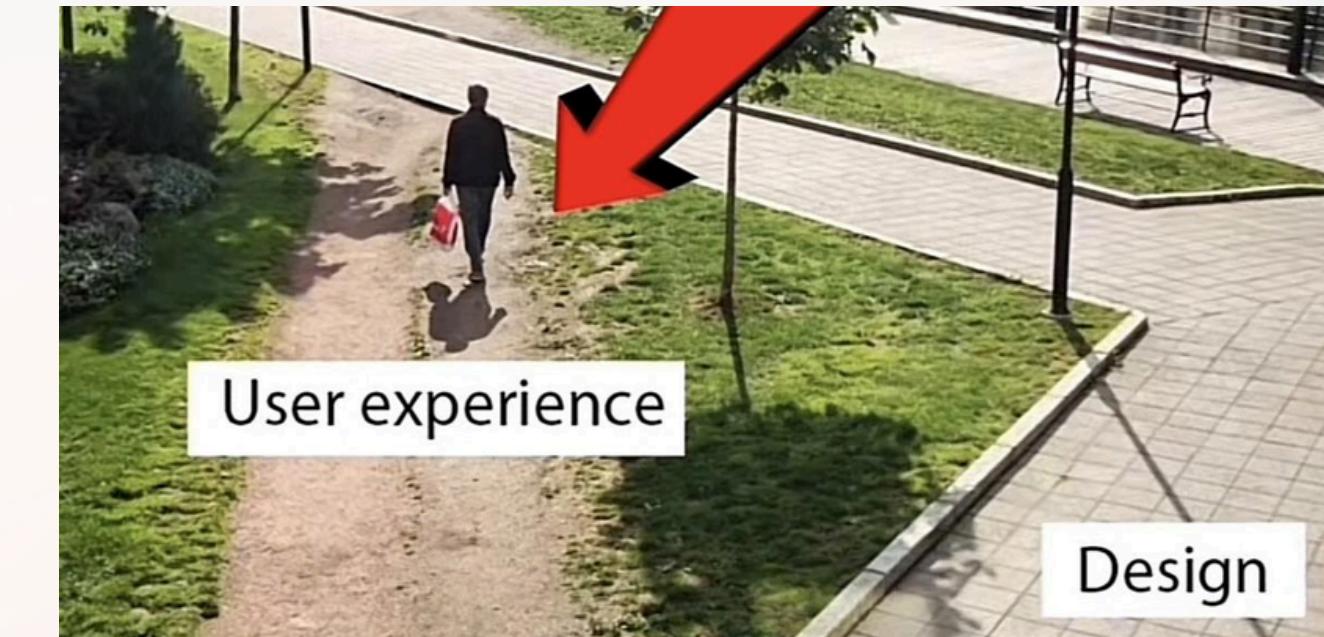
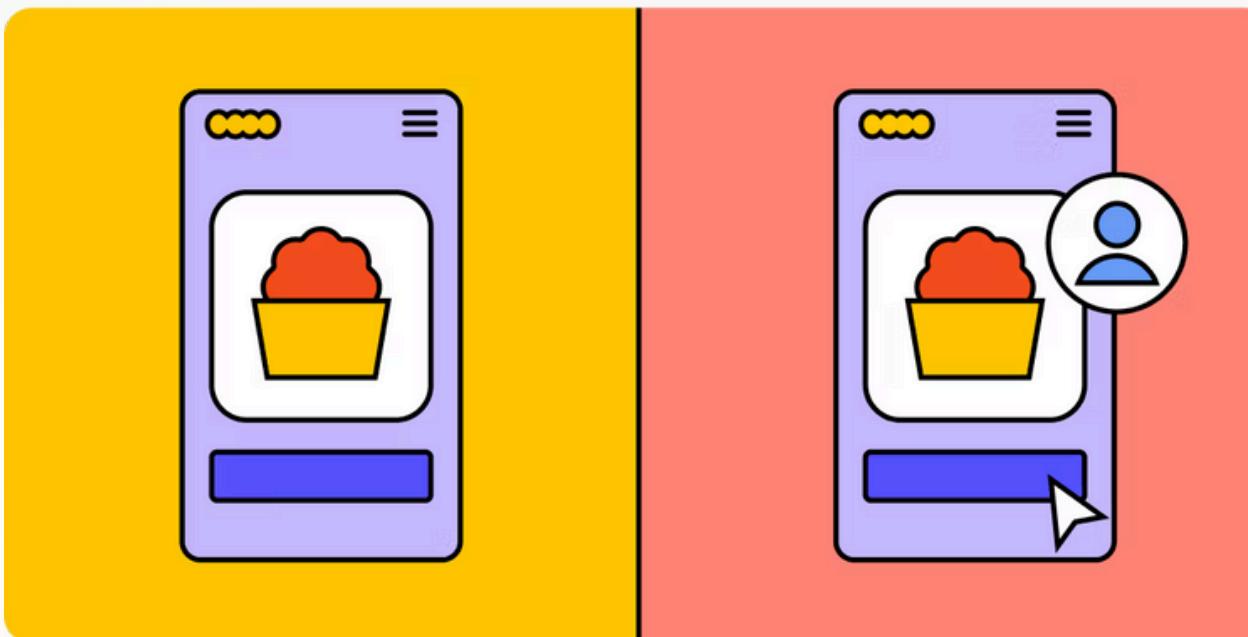
u5g4eeey3



Nicoly Almeida

UX ≠ UI

Exemplo prático: Um formulário com um layout visual elegante (UI) mas que não mostra mensagens de erro em tempo real e demora para responder (UX ruim).



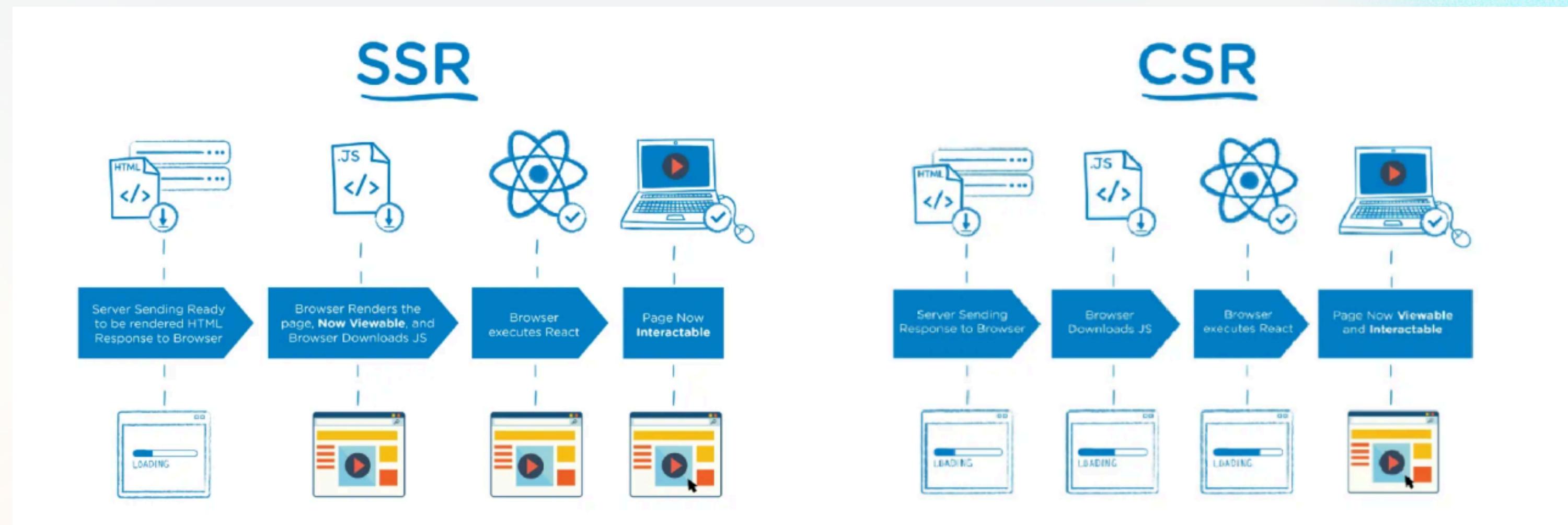
Leitura recomendada: Krug, S. *Don't Make Me Think: A Common Sense Approach to Web Usability*. New Riders, 2014.

Nicoly Almeida

Estratégias de Renderização

- **SSG (Static Site Generation)**: entrega páginas geradas no build time; ótimo para conteúdo estático.
- **SSR (Server-Side Rendering)**: entrega páginas a cada requisição; útil para conteúdo dinâmico.
- **ISR (Incremental Static Regeneration)**: combina SSG com atualizações pontuais em segundo plano.
- **CSR (Client-Side Rendering)**: conteúdo renderizado diretamente no navegador, ideal para interatividade alta.

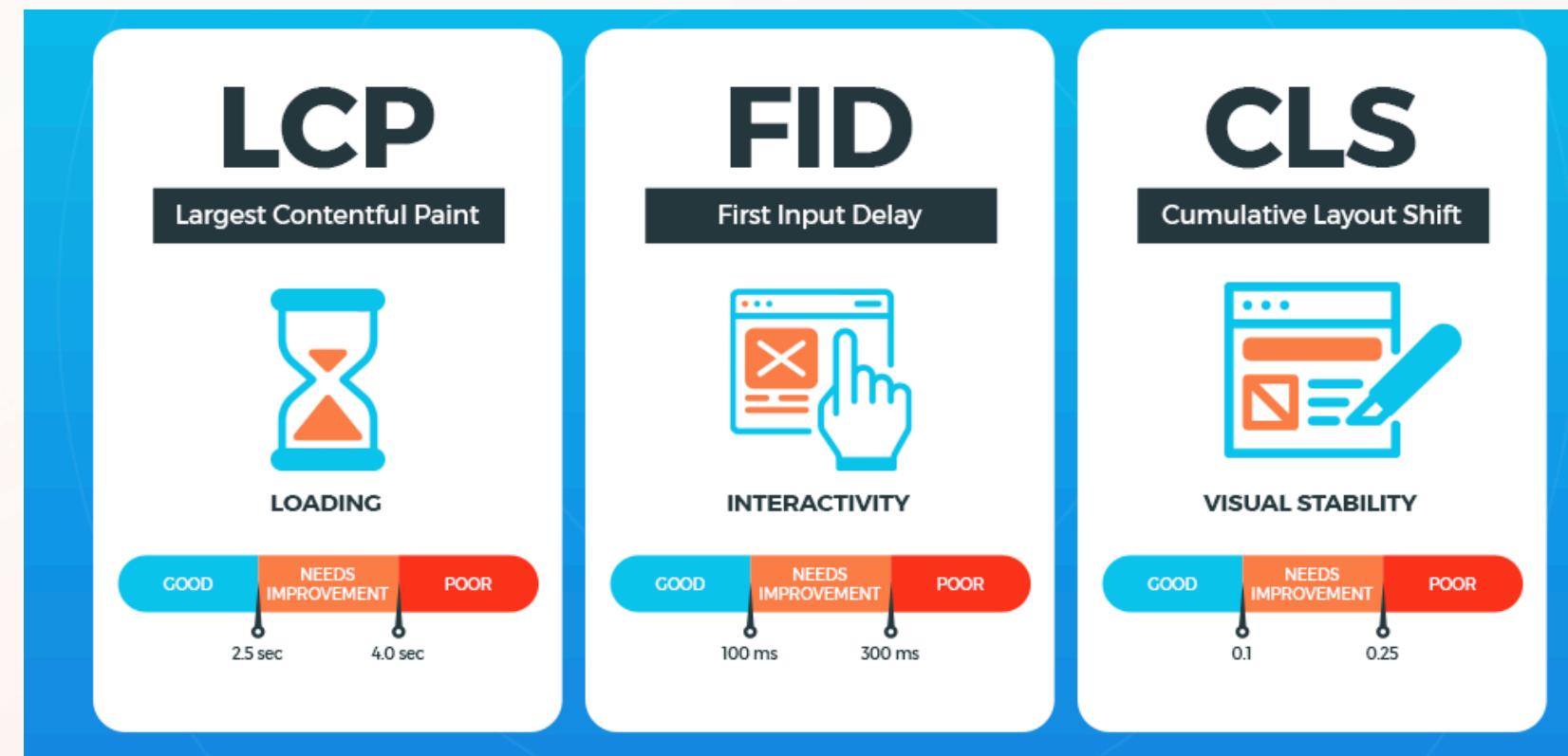
Estratégias de Renderização



Nicoly Almeida

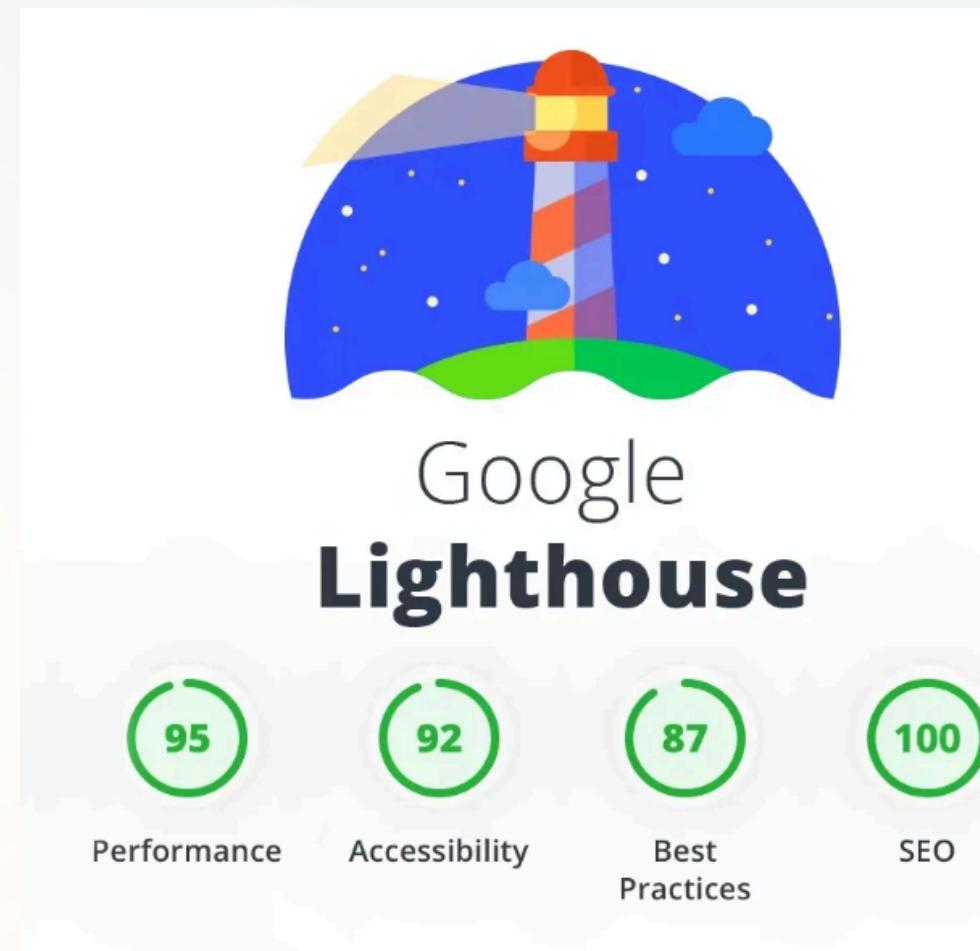
Métricas Core Web Vitals

- **LCP (Largest Contentful Paint)**: tempo para carregar o maior elemento visível.
- **CLS (Cumulative Layout Shift)**: medida de instabilidade visual.
- **FID (First Input Delay) / INP**: tempo até a resposta à primeira interação do usuário.
- **Hydration Time**: importante para aplicações CSR/SSR com React.

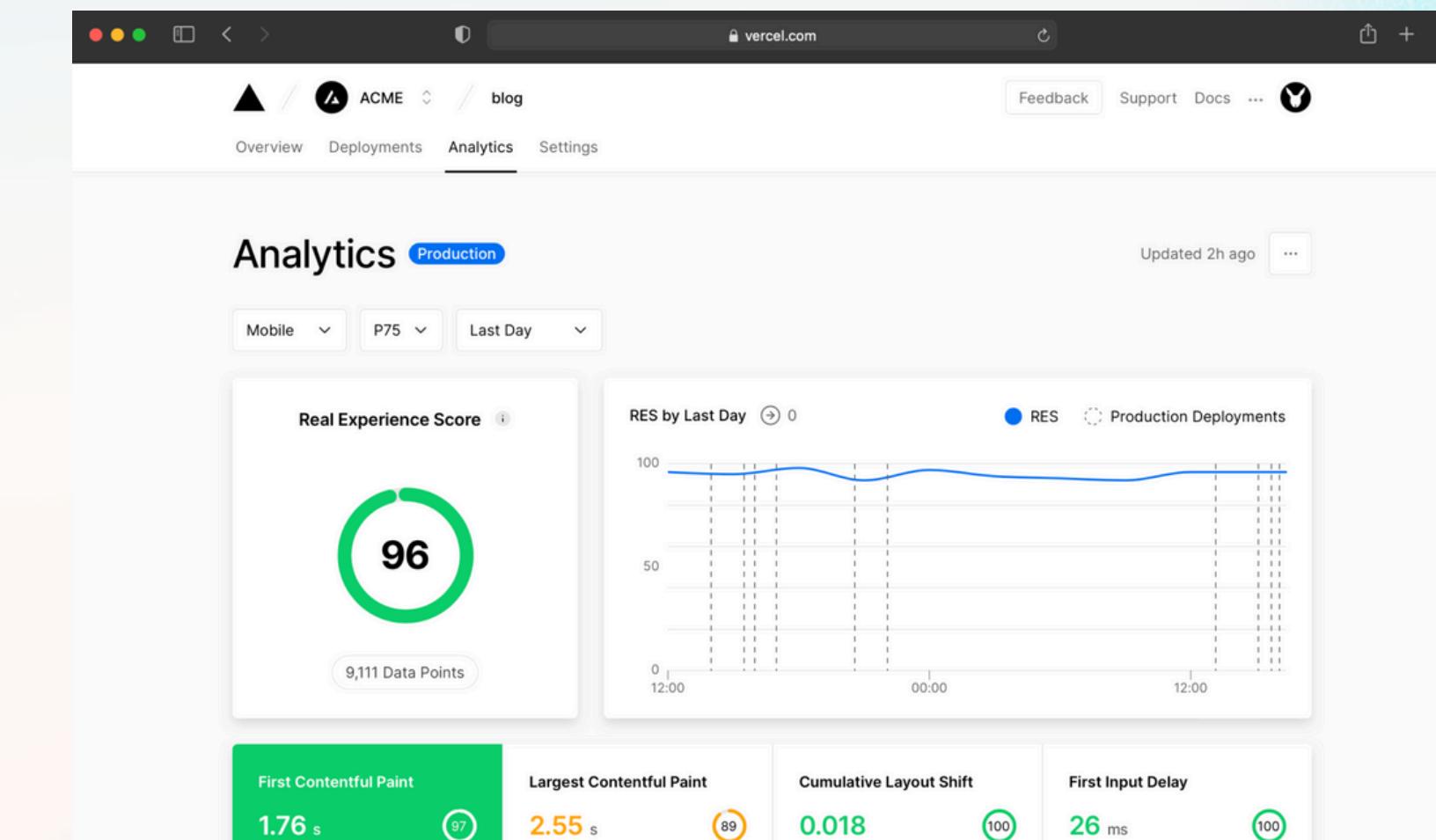


Nicoly Almeida

Ferramentas para monitoramento



[Google Lighthouse](#)



[Vercel Analytics](#)

Referências: [Next.js on Vercel](#)

Nicoly Almeida

Padrões de UX em Next.js

Scroll Restoration: evitar que o usuário volte ao topo ao navegar entre páginas.

Exemplo:



```
<Link href="/produtos" scroll={false}>Produtos</Link>
```

Transições de Página com Animação: usar bibliotecas como Framer Motion para suavizar a navegação. Exemplo:



```
<motion.div exit={{ opacity: 0 }} animate={{ opacity: 1 }} />
```

****Prefetch Inteligente com **``.** Exemplo:



```
<Link href="/sobre" prefetch>Sobre</Link>
```

Rotas Dinâmicas Otimizadas: usar **getStaticProps** e **getStaticPaths** para gerar páginas com performance e UX elevadas.

Nicoly Almeida

Responsividade e Adaptabilidade

Mobile First com Tailwind CSS

Tailwind facilita a criação de layouts responsivos com classes específicas por breakpoint:



```
<div className="p-4 md:p-8 lg:p-12">Conteúdo</div>
```

Mobile First com Tailwind CSS

Tailwind facilita a criação de layouts responsivos com classes específicas por breakpoint:



```
const isMobile = useMediaQuery('max-width: 768px');
```

Progressive Enhancement

Design progressivo significa garantir que funcionalidades essenciais funcionem mesmo sem JavaScript. Isso melhora a acessibilidade e compatibilidade em dispositivos antigos.

Nicoly Almeida

Referência: https://developer.mozilla.org/en-US/docs/Glossary/Progressive_enhancement

Feedback Imediato ao Usuário

Técnicas recomendadas

- Skeleton screens:



```
<div className="animate-pulse bg-gray-300 h-6 w-3/4" />
```

- Toasts:



```
import toast from 'react-hot-toast';
toast.success('Salvo com sucesso!');
```

- Transições



```
const [isPending, startTransition] = useTransition();
startTransition(() => router.push('/dashboard'));
```

Nicoly Almeida

Microinterações

Pequenas animações e reações aumentam a percepção de fluidez e qualidade:



```
<motion.button whileTap={{ scale: 0.95 }}>Enviar</motion.button>
```

Exemplos:

- Botões que indicam cliques
- Inputs que validam em tempo real
- Animações de carregamento inline

Leitura recomendada:

- Tidwell, J. Designing Interfaces. O'Reilly, 2020.
- Norman, D. The Design of Everyday Things. Basic Books, 2013.

Estudo de Caso Real

Problema identificado:

- Alto tempo de LCP e CLS elevado, prejudicando experiência de navegação e SEO

Soluções implementadas:

- Troca de por next/image
- Skeleton screens para carregar listas
- Lazy hydration em partes menos críticas
- Prefetch manual para links principais

Resultados:

- 42% de melhoria em LCP
- 18% de aumento em conversões no funil

Nicoly Almeida

Design Pattern – Container/Presentational

Problema identificado:

- Alto tempo de LCP e CLS elevado, prejudicando experiência de navegação e SEO

Soluções implementadas:

- Troca de por next/image
- Skeleton screens para carregar listas
- Lazy hydration em partes menos críticas
- Prefetch manual para links principais

Resultados:

- 42% de melhoria em LCP
- 18% de aumento em conversões no funil

Nicoly Almeida

Design Pattern – Container/Presentational

O padrão Container/Presentational separa responsabilidades entre dois tipos de componentes:

- **Container Components (Smart):**

- Lidam com lógica de negócios, estado, chamadas à API
- Conectam-se com contextos, Redux, hooks personalizados

- **Presentational Components (Dumb):**

- Recebem dados por props
- Focam na renderização e estilo
- Sem conhecimento de onde os dados vieram

Benefícios:

- Reaproveitamento de componentes visuais
- Testes unitários facilitados
- Redução de acoplamento entre UI e lógica de aplicação



VAMOS PRATICAR?

Nicoly Almeida