



UI'S ESCALÁVEIS COM TAILWIND CSS NO NEXT.JS

Nicoly Almeida

Objetivo do treinamento

- Dominar a criação de interfaces performáticas e escaláveis com Tailwind CSS no Next.js, aplicando boas práticas de arquitetura, componentização e padronização de estilos em projetos de grande porte.
- Aplicar técnicas avançadas de integração e otimização, incluindo SSR/SSG, acessibilidade e temas.
- Estruturar e evoluir design systems internos para garantir consistência e escalabilidade em times grandes.

Nicoly Almeida

Introdução e Contexto

Utility-first CSS: Diferente de BEM ou OOCSS, onde criamos nomes de classes semânticas (**btn-primary**, **header__title**), no Tailwind trabalhamos com utilitários (**px-4 py-2 bg-blue-600**). Isso permite prototipagem rápida e consistência.

Por que funciona bem no Next.js:

- **SSR:** Next renderiza HTML no servidor, e como Tailwind gera classes pré-compiladas, o CSS crítico já chega pronto.
- **Purge automático:** Tailwind remove classes não usadas, mantendo o bundle leve.

Impacto prático: Redução de CSS de centenas de KB para 10–20 KB em produção.

Nicoly Almeida

Introdução e Contexto

<!-- BEM -->

```
<button class="btn btn--primary">  
  <span class="btn__icon">★</span>  
  Comprar  
</button>
```

<!-- Utility-first (Tailwind) -->

```
<button class="inline-flex items-center gap-2 px-4 py-2 rounded-md bg-  
blue-600 text-white hover:bg-blue-700">  
  <span>★</span>  
  Comprar  
</button>
```

Nicoly Almeida

Setup Avançado do Tailwind

- Configuração mínima: *npm install tailwindcss postcss autoprefixer*.
- Enterprise setup:
 - Criar tailwind.config.ts modular.
 - Usar presets para compartilhar temas em múltiplos projetos/monorepos.
- Exemplo: um preset compartilhado entre 3 apps diferentes, mantendo identidade visual.

Setup Avançado do Tailwind

apps/
web/ (Next)
admin/ (Next)
packages/
ui/ ← componentes + tailwind-variants
tailwind/ ← preset com tokens

postcss.config.js

```
module.exports = {  
  plugins: {  
    tailwindcss: {},  
    autoprefixer: {},  
  },  
};
```

Instalação

```
npm i -D tailwindcss postcss autoprefixer  
npx tailwindcss init -p
```

globals.css (em app/)

```
@tailwind base;  
@tailwind components;  
@tailwind utilities;  
  
@layer base {  
  :root {  
    --radius: 12px;  
  }  
}
```

Nicoly Almeida

Setup Avançado do Tailwind

- Server Components (Next 13+):
 - Garantir que o CSS do Tailwind não quebre quando usado em app/.
 - Evitar imports de CSS globais desnecessários.
- Tokens: Centralizar design tokens (cores, espaçamento, tipografia) no tailwind.config.ts – pode até consumir JSON exportado do Figma.

Arquitetura de UI

- Problema comum: classes enormes e não reutilizáveis.
- Solução: abstrair usando ferramentas como clsx ou tailwind-variants (cva).
- Exemplo prático: criar um botão com variantes (primary, secondary, danger) sem duplicar código.
- Headless UI/Radix UI: fornecem acessibilidade pronta (menus, diálogos, tooltips) e você estiliza com Tailwind. Isso evita reimplementar comportamento complexo de acessibilidade.

Performance e SSR/SSG

Como Next.js trata o CSS no SSR:

- Tailwind gera classes estáticas.
- CSS crítico é incluído direto no HTML inicial.

Problema comum: inflar classes (mt-1 mt-2 mt-3) e gerar bundles grandes.

Boas práticas:

- Usar @layer para criar utilitários internos.
- Evitar @apply em excesso, mas útil para padrões repetidos.

Benchmark:

- Styled-components injeta estilos via JS no runtime (piora TTFB).
- Tailwind gera CSS estático (melhor LCP).

Escalabilidade

- Problema: em times grandes, Tailwind pode virar “spaghetti” de classes.
- Soluções:
 - Criar UI Primitives (<Button>, <Card>, <Input>).
 - Usar tailwind-variants para variantes de UI.
 - Estruturar monorepos com configs compartilhadas (presets).
- Versionamento: criar um design system interno versionado como um pacote npm.

Integrações Avançadas

- Dark Mode / Temas:
 - Com next-themes, usar class strategy (dark:bg-gray-900).
 - Permite múltiplos temas além de dark/light (ex: brand themes).
- Framer Motion + Tailwind:
 - Tailwind define layout, Motion cuida da animação.
 - Exemplo: transição de sidebar responsiva.
- Storybook + Tailwind:
 - Documentar componentes visuais.
 - Configuração simples com postcss-loader.
- Design Tokens Sync:
 - Plugins que exportam tokens direto do Figma para tailwind.config.ts.
 - Evita divergência entre design e dev.



VAMOS PRATICAR?

Nicoly Almeida