



# Guia de Acessibilidade - Nautilus One

## WCAG 2.1 AA Compliance Guide

**Autor:** DeepAgent - Abacus.AI

**Data:** 11 de Dezembro de 2025

**Versão:** FASE 3.2.0



## Índice

1. [Introdução](#)
2. [Princípios Fundamentais](#)
3. [Componentes Acessíveis](#)
4. [Navegação por Teclado](#)
5. [ARIA Labels e Roles](#)
6. [Contraste de Cores](#)
7. [Screen Reader Support](#)
8. [Testes de Acessibilidade](#)
9. [Checklist](#)



## Introdução

Este guia documenta as práticas de acessibilidade implementadas no Nautilus One e fornece diretrizes para desenvolvimento acessível.

### Por que Acessibilidade?

- **15% da população mundial** tem alguma deficiência
- **Requisito legal** em muitos países
- **Melhor UX** para todos os usuários
- **SEO melhor** com HTML semântico
- **Performance** (código mais limpo e eficiente)

### Padrão WCAG 2.1 AA

O Nautilus One segue o padrão **WCAG 2.1 nível AA**, que garante:

- Perceptibilidade (texto alternativo, contraste)
- Operabilidade (navegação por teclado)
- Compreensibilidade (labels claros)
- Robustez (compatibilidade com tecnologias assistivas)

## Princípios Fundamentais

### 1. Semântica HTML

 Evite:

```
<div onClick={handleClick}>Clique aqui</div>
```

 Use:

```
<button onClick={handleClick}>Clique aqui</button>
```

### 2. Sempre use elementos semânticos apropriados

```
// Correto
<header>...</header>
<nav>...</nav>
<main>...</main>
<footer>...</footer>
<button>...</button>
<a href="#">...</a>
```



## Componentes Acessíveis

### Componente `<Clickable>`

Use `<Clickable>` para elementos não-button que precisam ser clicáveis:

```
import { Clickable } from '@components/ui/Clickable';

// ✗ Evite
<div onClick={handleClick}>
  <Icon />
</div>

// ✅ Use
<Clickable onClick={handleClick} aria-label="Abrir menu">
  <Icon />
</Clickable>
```

### Componente `<ClickableCard>`

Para cards clicáveis:

```
import { ClickableCard } from '@/components/ui/clickable';

<ClickableCard
  onClick={handleNavigate}
  aria-label="Ver detalhes do navio"
>
  <h3>Nome do Navio</h3>
  <p>Informações...</p>
</ClickableCard>
```

## Componente `<ClickableIcon>`

Para ícones clicáveis (sempre requer aria-label):

```
import { ClickableIcon } from '@/components/ui/clickable';

<ClickableIcon
  onClick={handleDelete}
  aria-label="Excluir item"
>
  <TrashIcon />
</ClickableIcon>
```

## Navegação por Teclado

### Requisitos Mínimos

Todos os elementos interativos devem ser:

- **Focáveis por Tab** ( `tabIndex={0}` )
- **Ativáveis por Enter ou Espaço**
- **Visíveis quando focados** (outline visível)

### Implementação Manual

Se não puder usar os componentes helper, adicione manualmente:

```
<div
  onClick={handleClick}
  onKeyDown={(e) => {
    if (e.key === 'Enter' || e.key === ' ') {
      e.preventDefault();
      handleClick(e);
    }
  }}
  role="button"
  tabIndex={0}
  aria-label="Descrição"
>
  Conteúdo
</div>
```

## Hook `makeKeyboardAccessible`

Use o hook para props automáticos:

```
import { makeKeyboardAccessible } from '@/utils/accessibility';

const props = makeKeyboardAccessible(handleClick);

<div {...props} aria-label="Descrição">
  Conteúdo
</div>
```

## Atalhos de Teclado Comuns

Tecla	Ação
<b>Tab</b>	Navegar para próximo elemento
<b>Shift + Tab</b>	Navegar para elemento anterior
<b>Enter</b>	Ativar elemento focado
<b>Espaço</b>	Ativar botão focado
<b>Esc</b>	Fechar modal/dropdown
<b>Setas</b>	Navegar em menus, tabs, selects

## ARIA Labels e Roles

### Quando usar aria-label

Use `aria-label` quando o elemento não tem texto visível:

```
// ✅ Correto
<button aria-label="Fechar modal">
  <XIcon />
</button>

// ❌ Incorreto (texto visível não precisa de aria-label)
<button aria-label="Salvar">Salvar</button>
```

## Roles ARIA Comuns

```
// Navegação
<nav role="navigation">...</nav>

// Banner/Header
<header role="banner">...</header>

// Conteúdo principal
<main role="main">...</main>

// Informação adicional
<aside role="complementary">...</aside>

// Footer
<footer role="contentinfo">...</footer>

// Busca
<div role="search">...</div>

// Diálogo/Modal
<div role="dialog" aria-modal="true">...</div>

// Alerta
<div role="alert" aria-live="assertive">...</div>
```

## ARIA States

```
// Expandido/Colapsado
<button aria-expanded={isOpen}>Menu</button>

// Selecionado
<div role="tab" aria-selected={isSelected}>Tab 1</div>

// Desabilitado
<div aria-disabled="true">...</div>

// Oculto
<div aria-hidden="true">...</div>

// Checado
<div role="checkbox" aria-checked={isChecked}>...</div>
```

## Presets ARIA

Use os presets do `accessibility.ts`:

```

import { ariaPresets } from '@/utils/accessibility';

// Button
<div {...ariaPresets.button('Salvar', isPressing)}>
  Salvar
</div>

// Tab
<div {...ariaPresets.tab('Configurações', isSelected, [panel[-1]])}>
  Configurações
</div>

// Dialog
<div {...ariaPresets.dialog('Confirmação', 'dialog-desc')}>
  ...
</div>

```

## Contraste de Cores

### Requisitos WCAG AA

- **Texto normal:** mínimo **4.5:1**
- **Texto grande** (18pt+ ou 14pt+ bold): mínimo **3:1**
- **Componentes UI:** mínimo **3:1**

### Verificar Contraste

Use a função `meetsContrastRequirement`:

```

import { meetsContrastRequirement } from '@/utils/accessibility';

const isAccessible = meetsContrastRequirement(
  '#FFFFFF', // Texto
  '#1E40AF', // Fundo
  false       // false = texto normal
);

if (!isAccessible) {
  console.error('Contraste insuficiente!');
}

```

### Cores Aprovadas (WCAG AA)

Use as variáveis CSS de alto contraste:

```

/* Texto em fundo claro */
color: hsl(var(--hc-text-primary));      /* 16:1 */
color: hsl(var(--hc-text-secondary));    /* 10:1 */

/* Status */
color: hsl(var(--hc-success));          /* 7:1 */
color: hsl(var(--hc-error));           /* 7:1 */
color: hsl(var(--hc-warning));         /* 5.5:1 */

```

# Screen Reader Support

## Classes SR-Only

Para texto visível apenas para screen readers:

```
// CSS
Navegação principal
Pular para conteúdo principal
```

## Live Regions

Para anúncios dinâmicos:

```
import { useLiveRegion } from '@hooks/useAccessibility';

const { announce, regionProps } = useLiveRegion('polite');

// Anunciar mensagem
announce('Item adicionado ao carrinho');

// Renderizar região


{message}


```

## Hook useScreenReaderAnnouncement

```
import { useScreenReaderAnnouncement } from '@hooks/useAccessibility';

const announce = useScreenReaderAnnouncement();

// Anunciar mensagem
announce('Operação concluída com sucesso', 'polite');
announce('Erro crítico!', 'assertive');
```

## Landmarks Semânticos

Sempre use landmarks para estruturar a página:

```

<body>
  /* Skip link */
  <a href="#main" className="skip-link">
    Pular para conteúdo principal
  </a>

  /* Header */
  <header role="banner">
    <nav role="navigation">...</nav>
  </header>

  /* Main content */
  <main id="main" role="main">
    <h1>Título da Página</h1>
    ...
  </main>

  /* Footer */
  <footer role="contentinfo">
    ...
  </footer>
</body>

```

## Testes de Acessibilidade

### Testes Automatizados

Execute auditoria com axe-core:

```
npm run test:accessibility
```

### Testes Manuais

#### 1. Navegação por Teclado

- Use apenas Tab, Enter, Espaço, Setas
- Verifique se todos os elementos são acessíveis
- Verifique se o foco é visível

#### 2. Screen Reader

- Windows: NVDA (gratuito)
- macOS: VoiceOver (nativo)
- Verifique se todos os elementos são anunciados corretamente

#### 3. Zoom

- Teste com zoom de 200%
- Verifique se o layout não quebra
- Verifique se todo o conteúdo é visível

#### 4. Contraste

- Use DevTools do Chrome
- Verifique Lighthouse Accessibility Score

## Playwright Accessibility Tests

```
import { test, expect } from '@playwright/test';
import AxeBuilder from '@axe-core/playwright';

test('should not have accessibility violations', async ({ page }) => {
  await page.goto('/');

  const results = await new AxeBuilder({ page })
    .withTags(['wcag2a', 'wcag2aa', 'wcag21a', 'wcag21aa'])
    .analyze();

  expect(results.violations).toEqual([]);
});
```

## ✓ Checklist de Acessibilidade

### Para cada componente novo:

- [ ] Usa elementos HTML semânticos apropriados
- [ ] Todas as imagens têm `alt` `text`
- [ ] Botões e links têm labels descritivos
- [ ] Elementos clicáveis são acessíveis por teclado
- [ ] Foco é visível (outline)
- [ ] Contraste de cores  $\geq 4.5:1$  (texto normal)
- [ ] Inputs têm `<label>` associados
- [ ] Modais têm focus trap e fecham com Esc
- [ ] Mensagens de erro têm `role="alert"`
- [ ] Formulários têm validação acessível
- [ ] Testado com navegação por teclado
- [ ] Testado com screen reader (opcional mas recomendado)

### Para cada página:

- [ ] Tem um `<h1>` único e descritivo
- [ ] Hierarquia de headings correta (`h1 → h2 → h3`)
- [ ] Tem landmarks semânticos (`header`, `nav`, `main`, `footer`)
- [ ] Tem skip link para conteúdo principal
- [ ] Todas as funcionalidades são acessíveis por teclado
- [ ] Score Lighthouse Accessibility  $\geq 90$

## Recursos Adicionais

### Documentação Oficial

- [WCAG 2.1 Guidelines](https://www.w3.org/WAI/WCAG21/quickref/) (<https://www.w3.org/WAI/WCAG21/quickref/>)
- [MDN Accessibility](https://developer.mozilla.org/en-US/docs/Web/Accessibility) (<https://developer.mozilla.org/en-US/docs/Web/Accessibility>)
- [WAI-ARIA Authoring Practices](https://www.w3.org/WAI/ARIA/apg/) (<https://www.w3.org/WAI/ARIA/apg/>)

## Ferramentas

- [axe DevTools](https://www.deque.com/axe/devtools/) (<https://www.deque.com/axe/devtools/>)
- [Lighthouse](https://developers.google.com/web/tools/lighthouse) (<https://developers.google.com/web/tools/lighthouse>)
- [WAVE](https://wave.webaim.org/) (<https://wave.webaim.org/>)
- [Color Contrast Analyzer](https://www.tpgi.com/color-contrast-checker/) (<https://www.tpgi.com/color-contrast-checker/>)

## Screen Readers

- **NVDA** (Windows, gratuito): <https://www.nvaccess.org/>
  - **JAWS** (Windows, pago): <https://www.freedomscientific.com/products/software/jaws/>
  - **VoiceOver** (macOS/iOS, nativo)
  - **TalkBack** (Android, nativo)
- 



## Próximos Passos

1. **Revisar todos os componentes** com este guia
  2. **Adicionar testes** de acessibilidade no CI/CD
  3. **Treinar equipe** em práticas de acessibilidade
  4. **Fazer auditoria trimestral** com usuários reais
- 

**Mantido por:** DeepAgent - Abacus.AI

**Última atualização:** 11 de Dezembro de 2025

**Versão:** FASE 3.2.0