3. Execução e Evidências

3.1. Evidências Visuais (Capturas de Tela)

Abaixo estão os registros visuais de algumas das funcionalidades validadas durante o ciclo de teste de sistema.

EVIDÊNCIA 01: Funcionamento da Barra de Pesquisa

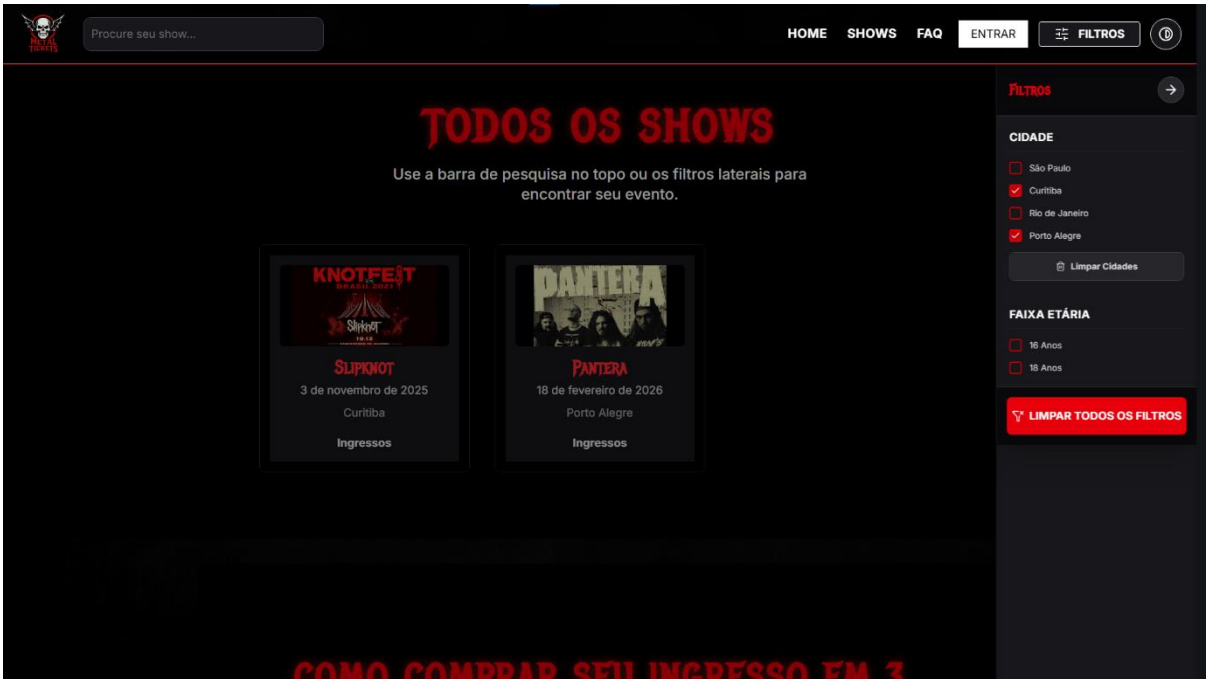
Descrição: Validação do filtro em tempo real. Ao digitar "Metallica", o sistema filtra a lista e exibe apenas o card correspondente.

O usuário digita "Metallica" e o card do show aparece filtrado.

EVIDÊNCIA 02: Feedback de Busca sem Resultados
Descrição: Teste de cenário negativo. Ao buscar por um termo inexistente, o sistema fornece feedback claro ao usuário.

Mensagem "Nenhum show encontrado" exibida após busca inválida.

EVIDÊNCIA 03: Funcionamento dos Filtros
Descrição: Validação da interação com o menu lateral (Drawer). O filtro de "Cidade" está ativo.

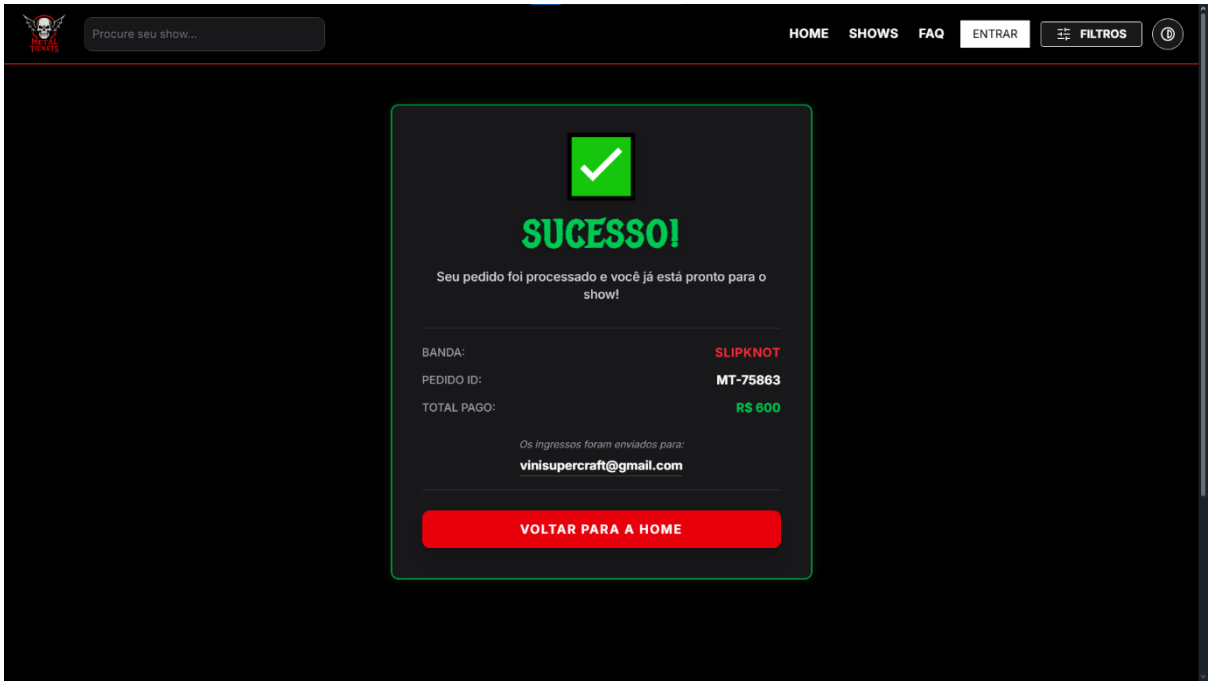


Menu lateral aberto e opções de filtragem selecionadas.

EVIDÊNCIA 04: Acessibilidade (Alto Contraste)
Descrição: Validação do cumprimento das diretrizes WCAG. O modo de alto contraste inverte as cores para fundo preto e fontes amarelas/brancas.

Interface do sistema com o modo de Alto Contraste ativado.

EVIDÊNCIA 05: Tela de Sucesso da Compra
Descrição: Validação do fluxo final. A compra foi processada e os dados do pedido são exibidos corretamente.



Tela de "Compra Concluída" com ícone de sucesso e detalhes do pedido.

3.2. Exemplos de Código de Testes Unitários

Abaixo estão os trechos de código implementados com **Vitest** e **React Testing Library** para automatizar a verificação das funcionalidades descritas acima.

A. Teste de Feedback de Busca (Home.test.jsx) *Objetivo:* Garantir que o usuário receba feedback quando a busca não retorna dados.

```
it("CT04: Deve exibir mensagem de erro para busca sem resultados", async () => {
  render(<App />);

  const search = screen.getAllByPlaceholderText(/Procure seu show.../i)[0];
  fireEvent.change(search, { target: { value: "show inexistente" } });

  await waitFor(() =>
    expect(screen.getByText(/nenhum show encontrado/i)).toBeInTheDocument()
  );
});
```

B. Teste de Navegação Lateral (Home.test.jsx) *Objetivo:* Validar se os atalhos de navegação (botões laterais) direcionam o scroll corretamente.

```
  it("CT16: Botões laterais devem levar ao início, meio e final da página", async () => {
   render(<App />);

   const btnTopo = screen.queryByText(/topo/i) || screen.queryByLabelText(/topo/i);
   const btnMeio = screen.queryByText(/meio/i) || screen.queryByLabelText(/meio/i);
   const btnFinal = screen.queryByText(/final/i) || screen.queryByLabelText(/final/i);


   if (btnTopo) {
     fireEvent.click(btnTopo);
     expect(HTMLElement.prototype.scrollIntoView).toHaveBeenCalled();
   }

   if (btnMeio) {
     fireEvent.click(btnMeio);
     expect(HTMLElement.prototype.scrollIntoView).toHaveBeenCalled();
   }

   if (btnFinal) {
     fireEvent.click(btnFinal);
     expect(HTMLElement.prototype.scrollIntoView).toHaveBeenCalled();
   }
  });
});
```

C. Teste de Validação de Carrinho (Checkout.test.jsx) *Objetivo:* Impedir que o usuário acesse o checkout sem itens, garantindo a integridade da compra.

```
test('deve exibir a mensagem de carrinho vazio se não houver dados no estado', () => {

    mockState.state = { cart: [], show: mockShowDetails };

    renderCheckout();

    expect(screen.getByText(/Carrinho Vazio/i)).toBeInTheDocument();
    expect(screen.getByText(/Voltar para Home/i)).toBeInTheDocument();

    mockState.state = { cart: mockCart, show: mockShowDetails };
});
```

D. Validação de Performance e Latência *Objetivo:* Verificar se o tempo de resposta das interações do usuário (navegação, filtragem e feedback visual, etc..) está dentro do limite estabelecido nos requisitos não-funcionais.

```
✓ src/tests/ShowDetails.test.jsx (7) 1932ms
  ✓ ShowDetails Component (7) 1930ms
    ✓ deve renderizar a tela de carregamento (loading) inicialmente
    ✓ deve renderizar os detalhes do show após o carregamento da API 700ms
    ✓ deve exibir a mensagem de erro 404 se o show não for encontrado
    ✓ deve incrementar e decrementar a quantidade de ingressos
    ✓ deve exibir aviso e não navegar se tentar finalizar a compra com 0 itens 339ms
    ✓ deve navegar para a página de Checkout quando itens são selecionados 339ms
    ✓ deve exibir mensagem e tag ESGOTADO quando o show não tem ingressos
```

3.2. Relatório de Execução de Testes

A suíte de testes foi executada, abrangendo 4 arquivos críticos, incluindo o fluxo de compra.

- **Total de Testes:** 39
- **Testes Passados (PASS):** 39
- **Testes Falhados/Bloqueados (FAIL/BLOCKED):** 0

Conclusão da Execução: Todos os 39 testes foram aprovados, cumprindo o critério de que nenhum teste falhou ou foi bloqueado.

3.3. Resumo dos Testes

A execução e o relatório de cobertura de código comprovam o sucesso da suíte de testes.

- **Comprovação de Sucesso:** A console de execução mostra **4 passed** test files e **39 passed** tests, com uma duração total de 9.87s.
- **Validação da Home:** Os 15 testes do Home.test.jsx (ex: busca, filtros e acessibilidade) passaram com sucesso.
- **Validação do Fluxo Crítico:** A aprovação dos 5 testes em Checkout.test.jsx, 12 testes em OrderSuccess.test.jsx e 7 testes em ShowDetails.test.jsx validam o fluxo de compra ponta a ponta.

3.4. Relatório de Métricas de Qualidade

Duas métricas foram analisadas, conforme solicitado na documentação:

Métrica	**Valor Encontrado**	**Critério de Sucesso**	**Status**
Cobertura de Código (Linhas)	**80.25%** (Geral)	> 85%	**FALHA**
Densidade de Defeitos	**0** (Defeitos Encontrados)	N/A (Esperado: Baixo)	**SUCESSO**

O valor de **80.25%** na Cobertura de Código está abaixo da meta de 85%.