Relatório Técnico de Análise de Falhas e Vulnerabilidades

Sistema: Travel HR Buddy (Nautilus One)

Data da Análise: 11 de dezembro de 2025
Versão do Sistema: 0.0.0
Tecnologias: React 19.2.1, TypeScript 5.8.3, Vite 5.4.19, Next.js 15.5.7
Analista: DeepAgent - Sistema de Análise Automatizada

📊 Resumo Executivo

Visão Geral

O sistema Travel HR Buddy (Nautilus One) é uma aplicação complexa de gestão empresarial desenvolvida com React/TypeScript e Vite. A análise identificou **67 categorias de problemas** distribuídas em diferentes níveis de severidade, incluindo vulnerabilidades críticas de segurança, problemas de performance, code smells e questões de manutenibilidade.

Estatísticas Gerais

- **Total de Arquivos Analisados:** ~2.500+ arquivos
- **Linhas de Código:** Estimadas em 150.000+
- **Módulos Principais:** 30+ módulos funcionais
- **Dependências:** 108 produção + 33 desenvolvimento

Distribuição de Severidade

- 🔴 **CRÍTICAS:** 18 falhas (segurança, dados sensíveis, vulnerabilidades)
- 🟡 **MÉDIAS:** 32 falhas (performance, code smells, manutenibilidade)
- 🟢 **BAIXAS:** 17 falhas (melhorias, otimizações, boas práticas)

Impacto no Negócio

- **Risco de Segurança:** ALTO - Exposição de credenciais e falta de CSP
- **Risco de Performance:** MÉDIO - Bundle size e re-renders excessivos
- **Risco de Manutenibilidade:** MÉDIO-ALTO - Code smells e falta de type safety

---

🔴 FALHAS CRÍTICAS

1. Exposição de Credenciais Sensíveis no Código Fonte

Severidade: 🔴 CRÍTICA
CWE-798: Uso de Credenciais Hardcoded

Descrição:
Credenciais do Supabase (URL e chave pública) estão hardcoded diretamente no código fonte como valores padrão.

Localização:

```
Arquivo: src/integrations/supabase/client.ts
Linhas: 5-6
```

Código Problemático:

```
const SUPABASE_URL = import.meta.env.VITE_SUPABASE_URL || "https://vnbptmixvwropvanyh-db.supabase.co";
const SUPABASE_PUBLISHABLE_KEY = import.meta.env.VITE_SUPABASE_PUBLISHABLE_KEY || "eyJhbGciOiJIUzI1NiIsInR5c-CI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InZuYnB0bWl4dndyb3B2YW55aGRiIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NTg1NzczNTEsImV4cCI6MjA3NDE1MzM1MX0.-LivvlG-PJwz_Caj5nVk_dhVeheaXPCROmXc4G8UsJcE";
```

Impacto:
- Qualquer pessoa com acesso ao código fonte pode acessar o banco de dados
- Possibilidade de vazamento de dados sensíveis
- Risco de ataques à infraestrutura do Supabase
- Violação de compliance (LGPD, GDPR)

Recomendação:
1. Remover imediatamente as credenciais hardcoded
2. Invalidar as chaves expostas no Supabase
3. Gerar novas credenciais
4. Usar apenas variáveis de ambiente sem fallback hardcoded
5. Implementar validação que falhe ruidosamente se variáveis não estiverem definidas:

```
const SUPABASE_URL = import.meta.env.VITE_SUPABASE_URL;
const SUPABASE_PUBLISHABLE_KEY = import.meta.env.VITE_SUPABASE_PUBLISHABLE_KEY;

if (!SUPABASE_URL || !SUPABASE_PUBLISHABLE_KEY) {
  throw new Error("CRITICAL: Supabase credentials not configured. Please set VITE_SUPABASE_URL and VITE_SUPABASE_PUBLISHABLE_KEY environment variables.");
}
```

Prioridade: P0 - IMEDIATA

---

2. Falta de Content Security Policy (CSP)

Severidade: 🔴 CRÍTICA
CWE-693: Proteção Inadequada Contra Injeção

Descrição:
O sistema não implementa Content Security Policy nos headers HTTP, deixando a aplicação vulnerável a ataques XSS, clickjacking e injeção de código malicioso.

Localização:

```
Arquivo: public/_headers
Problema: Ausência de header Content-Security-Policy
```

Headers Atuais:

```
X-Frame-Options: SAMEORIGIN
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Referrer-Policy: strict-origin-when-cross-origin
Permissions-Policy: camera=(), microphone=(), geolocation=(self)
```

Impacto:
- Vulnerável a ataques XSS (Cross-Site Scripting)
- Possibilidade de injeção de scripts maliciosos
- Risco de roubo de tokens e sessões
- Exposição a ataques de clickjacking

Recomendação:
Adicionar CSP rigorosa ao arquivo `public/_headers`:

```
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://cdn.jsdelivr.net https://unpkg.com; style-src 'self' 'unsafe-inline' https://fonts.googleapis.com; font-src 'self' https://fonts.gstatic.com data:; img-src 'self' data: blob: https:; connect-src 'self' https://*.supabase.co wss://*.supabase.co https://api.openai.com https://api.mapbox.com; frame-ancestors 'self'; base-uri 'self'; form-action 'self'
```

Nota: Avaliar se `'unsafe-inline'` e `'unsafe-eval'` podem ser removidos após migração para nonces.

Prioridade: P0 - IMEDIATA

---

3. Vulnerabilidades em Dependências NPM

Severidade: 🔴 CRÍTICA
CWE-1035: Uso de Componentes com Vulnerabilidades Conhecidas

Descrição:
Múltiplas dependências apresentam vulnerabilidades conhecidas identificadas pelo `npm audit`.

Vulnerabilidades Identificadas:

3.1 esbuild (CVE via GHSA-67mh-4wv8-2f99)

- **Severidade:** Moderate
- **Versão:** <=0.24.2
- **Problema:** Permite que qualquer website envie requisições ao servidor de desenvolvimento
- **CVSS:** 5.3 (Médio)
- **CWE:** CWE-346 (Origin Validation Error)

3.2 glob (GHSA-5j98-mcp5-4vw2)

- **Severidade:** High

- **Versões:** 10.2.0-10.4.5, 11.0.0-11.0.3
- **Problema:** Command Injection via CLI com shell:true
- **Impacto:** Execução de código arbitrário

3.3 js-yaml (GHSA-mh29-5h37-fv8m)

- **Severidade:** Moderate
- **Versão:** 4.0.0-4.1.0
- **Problema:** Prototype Pollution via merge (<<)

3.4 nodemailer (GHSA-rcmh-qjqh-p98v)

- **Severidade:** Moderate
- **Versão:** <=7.0.10
- **Problema:** DoS causado por chamadas recursivas no addressparser

3.5 xlsx (GHSA-4r6h-8v6p-xvw6, GHSA-5pgg-2g8v-p4x9)

- **Severidade:** High
- **Problema:** Prototype Pollution e ReDoS (Regular Expression Denial of Service)
- **Fix:** Não disponível atualmente

3.6 vitest e dependências

- **Severidade:** Moderate
- **Problema:** Dependências transitivas com vulnerabilidades
- **Requer:** Upgrade para v4.0+

Impacto:
- Risco de execução de código arbitrário
- Possibilidade de ataques DoS
- Corrupção de protótipos JavaScript
- Comprometimento do ambiente de desenvolvimento

Recomendação:
1. **Imediato:**
`bash`
`npm audit fix`

1. **Breaking Changes (Avaliar):**
   `bash`
   `npm audit fix --force`

2. **Substituir xlsx:**
   - Considerar alternativa: `exceljs` ou `sheetjs-style`
   - Ou aguardar fix upstream

3. **Upgrade Vitest:**
   `bash`
   `npm install -D vitest@^4.0.0 @vitest/ui@^4.0.0 @vitest/coverage-v8@^4.0.0`

4. **Implementar:**
   - CI/CD check com `npm audit` bloqueando vulnerabilidades HIGH/CRITICAL
   - Dependabot ou Renovate Bot para atualizações automáticas

Prioridade: P0 - IMEDIATA (para HIGH), P1 (para MODERATE)

---

4. Uso Excessivo de dangerouslySetInnerHTML

Severidade: 🔴 CRÍTICA
CWE-79: Cross-Site Scripting (XSS)

Descrição:
14 ocorrências de `dangerouslySetInnerHTML` no código, muitas sem sanitização adequada, criando vetores de ataque XSS.

Localizações Críticas:

```
1. src/pages/documents/ai.tsx - Exibição de OCR text
2. src/pages/admin/documents/templates-dynamic.tsx - Preview de templates
3. src/pages/admin/vault-ai-complete.tsx - Resultados de busca destacados
4. src/pages/admin/assistant-logs.tsx - Respostas do assistente
5. src/pages/admin/assistant.tsx - Mensagens do chat
6. src/components/peotram/peotram-ai-assistant.tsx - Conteúdo do assistente
```

Código Problemático:

```
// Exemplo 1: Sem sanitização
<div dangerouslySetInnerHTML={{ __html: msg.content }} />

// Exemplo 2: Sanitização insuficiente
<div dangerouslySetInnerHTML={{
  __html: log.answer.replace(/<a /g,
"<a target=\"_blank\" rel=\"noopener noreferrer\" ")
}} />
```

Impacto:
- Injeção de scripts maliciosos
- Roubo de cookies e tokens
- Redirecionamento para sites phishing
- Execução de código no contexto do usuário

Recomendação:
1. **Instalar DOMPurify:**
`bash`
`npm install dompurify`
`npm install --save-dev @types/dompurify`

1. **Criar helper de sanitização:**
```typescript
// src/utils/sanitize.ts
import DOMPurify from 'dompurify';

export function sanitizeHtml(dirty: string): string {
return DOMPurify.sanitize(dirty, {
ALLOWED_TAGS: ['b', 'i', 'em', 'strong', 'a', 'p', 'br', 'ul', 'ol', 'li', 'h1', 'h2', 'h3', 'code', 'pre'],
ALLOWED_ATTR: ['href', 'target', 'rel'],
ALLOW_DATA_ATTR: false
});
```

}
```

1. **Usar em todos os casos:**
   ```typescript
   import { sanitizeHtml } from '@/utils/sanitize';
```

1. **Preferir react-markdown quando possível:**
   ```typescript
   import ReactMarkdown from 'react-markdown';

{msg.content}
```

Prioridade: P0 - IMEDIATA

---

5. Uso Excessivo de console.log em Produção

Severidade: 🔴 CRÍTICA (Performance/Segurança)
CWE-532: Inserção de Informações Sensíveis em Arquivo de Log

Descrição:
1.404 ocorrências de console.log/error/warn no código fonte, muitas sem verificação de ambiente de produção.

Estatísticas:

```
console.log: ~800 ocorrências
console.error: ~400 ocorrências
console.warn: ~204 ocorrências
```

Impacto:
- Vazamento de informações sensíveis em produção (tokens, dados de usuário, queries)
- Degradação de performance (console operations são lentas)
- Aumento do bundle size
- Dificuldade de debugging em produção

Exemplos Problemáticos:

```
// Vazamento potencial de dados
console.log("User data:", userData);
console.log("API Response:", response);
console.error("Auth token:", token);
```

Recomendação:
1. **Implementar logger centralizado:**
```typescript
// src/lib/logger.ts
type LogLevel = 'debug' | 'info' | 'warn' | 'error';

class Logger {
private isDev = import.meta.env.DEV;

```
  private log(level: LogLevel, ...args: any[]) {
    if (!this.isDev && level !== 'error') return;

    const method = level === 'debug' ? 'log' : level;
    console[method](`[${level.toUpperCase()}]`, ...args);
  }

  debug(...args: any[]) { this.log('debug', ...args); }
  info(...args: any[]) { this.log('info', ...args); }
  warn(...args: any[]) { this.log('warn', ...args); }
  error(...args: any[]) { this.log('error', ...args); }
```

}

export const logger = new Logger();
```

1. **Configuração Vite já remove console.log:**
   - Verificado em `vite.config.ts` linha 308-313
   - Terser já configurado para remover em produção
   - ✅ Implementação parcial existente

2. **Substituir todos os console.log:**
   ```bash
   # Buscar e substituir
   find src -type f \( -name "*.ts" -o -name "*.tsx" \) -exec sed -i 's/console\.log/logger.debug/g' {} +
   find src -type f \( -name "*.ts" -o -name "*.tsx" \) -exec sed -i 's/console\.error/logger.error/g' {} +
   find src -type f \( -name "*.ts" -o -name "*.tsx" \) -exec sed -i 's/console\.warn/logger.warn/g' {} +
   ```

3. **Adicionar ESLint rule:**
   ```json
   {
     "rules": {
       "no-console": ["error", { "allow": ["error"] }]
     }
   }
   ```

Prioridade: P1 - ALTA

---

6. Ausência de Rate Limiting em APIs

Severidade: 🔴 CRÍTICA
CWE-770: Alocação de Recursos sem Limites

Descrição:
Não há evidências de rate limiting implementado para proteger endpoints de API contra abuso e ataques de força bruta.
   ```

Impacto:
- Vulnerável a ataques DDoS
- Possibilidade de brute force em autenticação
- Consumo excessivo de recursos do Supabase
- Custos elevados de API

Recomendação:
1. **Implementar rate limiting no Supabase Edge Functions:**
```typescript
// supabase/functions/_shared/rate-limit.ts
import { Redis } from '@upstash/redis';

const redis = new Redis({
url: Deno.env.get('UPSTASH_REDIS_URL')!,
token: Deno.env.get('UPSTASH_REDIS_TOKEN')!,
});

export async function rateLimit(
identifier: string,
limit: number = 100,
window: number = 60
): Promise<{ success: boolean; remaining: number }> {
const key = `rate_limit:${identifier}`;
const current = await redis.incr(key);

  if (current === 1) {
    await redis.expire(key, window);
  }

  return {
    success: current <= limit,
    remaining: Math.max(0, limit - current)
  };

}
```

1. **Usar em todas as Edge Functions:**
   ```typescript
   import { rateLimit } from '../_shared/rate-limit.ts';

const rateLimitResult = await rateLimit(req.headers.get('x-forwarded-for') || 'unknown', 100, 60);
if (!rateLimitResult.success) {
return new Response('Too Many Requests', { status: 429 });
}
```

1. **Adicionar headers de rate limit:**
   `typescript`
   `return new Response(JSON.stringify(data), {`
   `headers: {`
   `'X-RateLimit-Limit': '100',`
   `'X-RateLimit-Remaining': rateLimitResult.remaining.toString(),`

```
      'X-RateLimit-Reset': new Date(Date.now() + 60000).toISOString()
    }
  });
```

Prioridade: P0 - IMEDIATA

---

7. Type Safety Inadequado (1.592 ocorrências de any/unknown)

Severidade: 🔴 CRÍTICA (Manutenibilidade)
CWE-1286: Tipo de Dado Inadequado

Descrição:
1.592 ocorrências de tipos `any` e `unknown` sem validação adequada, comprometendo a type safety do TypeScript.

Configuração TypeScript Problemática:

```
// tsconfig.json
{
  "compilerOptions": {
    "strict": true,
    "noImplicitAny": true,
    "strictNullChecks": false,  // ❌ PROBLEMA
    "strictFunctionTypes": true,
    "noUnusedParameters": false, // ❌ PROBLEMA
    "noUnusedLocals": false      // ❌ PROBLEMA
  }
}
```

Impacto:
- Bugs em runtime não detectados em compile time
- Dificuldade de manutenção e refatoração
- IntelliSense ineficaz
- Erros de null/undefined em produção

Exemplos Problemáticos:

```
// Tipo any sem validação
function processData(data: any) {
  return data.value.toUpperCase(); // Runtime error se data.value for undefined
}

// Unknown sem type guard
function handleResponse(response: unknown) {
  // @ts-ignore
  return response.data; // Perigoso
}
```

Recomendação:
1. **Ativar strict null checks:**

```
json
  {
    "compilerOptions": {
      "strictNullChecks": true,
```

```
      "noUnusedParameters": true,
      "noUnusedLocals": true
    }
  }
```

1. **Criar tipos adequados:**
   ```typescript
   // Em vez de any
   interface ApiResponse {
   data: T;
   status: number;
   message: string;
   }

function processData(data: ApiResponse): T {
return data.data;
}
```

1. **Usar type guards para unknown:**
   ```typescript
   function isApiResponse(value: unknown): value is ApiResponse {
   return (
   typeof value === 'object' &&
   value !== null &&
   'data' in value &&
   'status' in value
   );
   }

function handleResponse(response: unknown) {
if (!isApiResponse(response)) {
throw new Error('Invalid response format');
}
return response.data;
}
```

1. **Refatoração gradual:**
   - Criar issue tracker para conversão de `any` → tipos apropriados
   - Começar por módulos críticos (auth, payments, data)
   - Usar `@ts-expect-error` com comentário explicativo quando any for temporário

Prioridade: P1 - ALTA (refatoração gradual)

---

8. Ausência de Autenticação em Rotas Sensíveis

Severidade: 🔴 CRÍTICA
CWE-306: Ausência de Autenticação para Função Crítica

Descrição:
Análise da estrutura de rotas revela potencial ausência de validação de autenticação em todas as rotas administrativas e sensíveis.

Rotas Identificadas como Sensíveis:

```
- /admin/*
- /settings
- /user/profile
- /api-gateway
- /vault-ai
- /developer/*
```

Implementação Atual:

```
// App.tsx - Uso de ProtectedRoute e AdminRoute
<Route path="/admin" element={<AdminRoute><Admin /></AdminRoute>} />
<Route path="/settings" element={<ProtectedRoute><Settings /></ProtectedRoute>} />
```

Problemas Potenciais:
1. Se `ProtectedRoute` não validar token corretamente
2. Se não houver refresh de token
3. Se permitir acesso com token expirado
4. Se não validar roles adequadamente

Recomendação:
1. **Auditar implementação de ProtectedRoute:**
```typescript
// src/components/auth/protected-route.tsx
export function ProtectedRoute({ children, requiredRole }: Props) {
const { user, session, isLoading } = useAuth();
const location = useLocation();

```
// ✅ Verificar loading
if (isLoading) return <LoadingSpinner />;

// ✅ Verificar autenticação
if (!user || !session) {
  return <Navigate to="/auth" state={{ from: location }} replace />;
}

// ✅ Verificar expiração do token
if (session.expires_at && new Date(session.expires_at * 1000) < new Date()) {
  return <Navigate to="/auth" state={{ from: location }} replace />;
}

// ✅ Verificar role se necessário
if (requiredRole && user.role !== requiredRole) {
  return <Navigate to="/unauthorized" replace />;
}

return <>{children};
```
```

}
```

1. **Implementar validação no backend (Supabase RLS):**
   ```sql
   – Exemplo de Row Level Security
   CREATE POLICY “Users can only access own data”
   ON user_profiles
   FOR SELECT
   USING (auth.uid() = user_id);

CREATE POLICY “Only admins can access admin data”
ON admin_settings
FOR ALL
USING (
EXISTS (
SELECT 1 FROM user_profiles
WHERE user_id = auth.uid()
AND role = ‘admin’
)
);
```

1. **Adicionar middleware de autenticação em Edge Functions:**
   ```typescript
   export async function requireAuth(req: Request) {
   const token = req.headers.get(‘Authorization’)?.replace(‘Bearer ‘, ‘’);

   if (!token) {
   return new Response(‘Unauthorized’, { status: 401 });
   }

   const { data: { user }, error } = await supabase.auth.getUser(token);

   if (error || !user) {
   return new Response(‘Unauthorized’, { status: 401 });
   }

   return user;
   }
   ```

Prioridade: P0 - IMEDIATA (auditar e corrigir)

---

9. Falta de Validação de Input em Formulários

Severidade: 🔴 CRÍTICA
CWE-20: Validação de Input Inadequada

Descrição:
Sistema possui apenas 2 arquivos de validação (`src/validations/`), insuficiente para a complexidade da aplicação.
   ```

Estrutura Atual:

```
src/validations/
├── patches/
│   └── 606.ts
└── registry.ts
```

Impacto:
- Injeção de dados maliciosos
- Corrupção de banco de dados
- Bypass de regras de negócio
- XSS via input não validado

Recomendação:
1. **Implementar validação com Zod em todos os formulários:**
```typescript
// src/validations/auth.validation.ts
import { z } from 'zod';

export const loginSchema = z.object({
email: z.string()
.email('Email inválido')
.min(3, 'Email muito curto')
.max(255, 'Email muito longo'),
password: z.string()
.min(8, 'Senha deve ter no mínimo 8 caracteres')
.max(128, 'Senha muito longa')
.regex(/[A-Z]/, 'Senha deve conter letra maiúscula')
.regex(/[a-z]/, 'Senha deve conter letra minúscula')
.regex(/[0-9]/, 'Senha deve conter número')
.regex(/[^A-Za-z0-9]/, 'Senha deve conter caractere especial'),
});

export const signupSchema = loginSchema.extend({
fullName: z.string()
.min(3, 'Nome muito curto')
.max(100, 'Nome muito longo')
.regex(/^[a-zA-ZÀ-ÿ\s]+$/, 'Nome deve conter apenas letras'),
confirmPassword: z.string(),
}).refine((data) => data.password === data.confirmPassword, {
message: 'Senhas não coincidem',
path: ['confirmPassword'],
});
```

1. **Usar em todos os formulários:**
   ```typescript
   import { useForm } from 'react-hook-form';
   import { zodResolver } from '@hookform/resolvers/zod';
   import { loginSchema } from '@/validations/auth.validation';
   ```

```
const { register, handleSubmit, formState: { errors } } = useForm({
resolver: zodResolver(loginSchema)
});
```

1. **Validar no backend também:**
   ```typescript
   // supabase/functions/login/index.ts
   import { loginSchema } from '../_shared/validations/auth.ts';

const body = await req.json();

try {
const validated = loginSchema.parse(body);
// Processar login
} catch (error) {
if (error instanceof z.ZodError) {
return new Response(JSON.stringify({ errors: error.errors }), {
status: 400
});
}
}
```

1. **Criar validações para:**
   - Autenticação (login, signup, reset)
   - Perfil de usuário
   - Documentos
   - Viagens
   - Recursos HR
   - Configurações
   - Dados financeiros

Prioridade: P0 - IMEDIATA

---

10. Uso Inadequado de localStorage sem Try-Catch

Severidade: 🔴 CRÍTICA
CWE-754: Verificação Inadequada de Condições Excepcionais

Descrição:
544 ocorrências de uso de localStorage/sessionStorage sem tratamento adequado de erros, causando crashes em modo privado ou quando storage está cheio.

Problemas:
1. Safari em modo privado lança exceção ao acessar localStorage
2. Quota exceeded quando storage está cheio
3. Cookies bloqueados por configurações de privacidade
4. Crash da aplicação sem fallback

Código Problemático:

```
// Sem tratamento
function saveData(key: string, value: any) {
  localStorage.setItem(key, JSON.stringify(value)); // ❌ Pode lançar exceção
}

// Leitura sem validação
const data = JSON.parse(localStorage.getItem('user')); // ❌ Pode ser null
```

Implementação Parcial Existente:
O arquivo `src/integrations/supabase/client.ts` já implementa `safeLocalStorage`, mas não é usado em todo o código.

Recomendação:
1. **Criar helper centralizado:**
```typescript
// src/lib/storage.ts
type StorageType = 'local' | 'session';

class SafeStorage {
private storage: Storage | null;
```

```
constructor(type: StorageType = 'local') {
  try {
    const storage = type === 'local' ? localStorage : sessionStorage;
    const testKey = '__storage_test__';
    storage.setItem(testKey, 'test');
    storage.removeItem(testKey);
    this.storage = storage;
  } catch {
    console.warn(`${type}Storage is not available, using memory fallback`);
    this.storage = null;
  }
}

getItem<T = string>(key: string): T | null {
  try {
    const item = this.storage?.getItem(key) ?? null;
    if (!item) return null;

    try {
      return JSON.parse(item) as T;
    } catch {
      return item as unknown as T;
    }
  } catch (error) {
    console.error(`Error reading from storage:`, error);
    return null;
  }
}

setItem<T = any>(key: string, value: T): boolean {
  try {
    const stringValue = typeof value === 'string'
      ? value
      : JSON.stringify(value);

    this.storage?.setItem(key, stringValue);
    return true;
  } catch (error) {
    if (error.name === 'QuotaExceededError') {
      console.error('Storage quota exceeded');
      // Tentar limpar dados antigos
      this.clearOldData();
    }
    console.error(`Error writing to storage:`, error);
    return false;
  }
}

removeItem(key: string): boolean {
  try {
    this.storage?.removeItem(key);
    return true;
  } catch (error) {
    console.error(`Error removing from storage:`, error);
    return false;
  }
}

clear(): boolean {
  try {
    this.storage?.clear();
    return true;
```

```
    } catch (error) {
      console.error(`Error clearing storage:`, error);
      return false;
    }
  }

  private clearOldData(): void {
    // Implementar lógica para remover dados antigos
    // Baseado em timestamps ou LRU
  }
```

}

export const storage = new SafeStorage('local');
export const sessionStorage = new SafeStorage('session');
```

1. **Substituir todos os usos diretos:**
   `bash`
   `# Buscar e listar`
   `grep -r "localStorage\|sessionStorage" src --include="*.tsx" --include="*.ts" > storage-usage.txt`

2. **Usar em todo o código:**
   ```typescript
   import { storage } from '@/lib/storage';

// Em vez de localStorage.setItem('key', JSON.stringify(data))
storage.setItem('key', data);

// Em vez de JSON.parse(localStorage.getItem('key'))
const data = storage.getItem('key');
```

1. **Adicionar ESLint rule:**
   `json`
   `{`
   `"rules": {`
   `"no-restricted-globals": ["error", {`
   `"name": "localStorage",`
   `"message": "Use storage from @/lib/storage instead"`
   `}, {`
   `"name": "sessionStorage",`
   `"message": "Use sessionStorage from @/lib/storage instead"`
   `}]`
   `}`
   `}`

Prioridade: P1 - ALTA

```

11. Potencial Memory Leak em useEffect

Severidade: 🔴 CRÍTICA (Performance)
CWE-401: Liberação Inadequada de Recurso

Descrição:
1.863 ocorrências de `useEffect` no código. Análise de amostras indica potencial memory leak por falta de cleanup em subscriptions, timers e event listeners.

Padrões Problemáticos Comuns:

```
// ❌ PROBLEMA 1: Fetch sem abort
useEffect(() => {
  fetch('/api/data').then(data => setState(data));
}, []);

// ❌ PROBLEMA 2: Timer sem cleanup
useEffect(() => {
  const interval = setInterval(() => {
    updateData();
  }, 1000);
}, []);

// ❌ PROBLEMA 3: Event listener sem remover
useEffect(() => {
  window.addEventListener('resize', handleResize);
}, []);

// ❌ PROBLEMA 4: Subscription sem unsubscribe
useEffect(() => {
  const subscription = supabase
    .channel('changes')
    .on('postgres_changes', handleChange)
    .subscribe();
}, []);
```

Impacto:
- Consumo crescente de memória
- Degradação de performance ao longo do tempo
- Crashes em dispositivos com pouca RAM
- Execução de callbacks em componentes desmontados

Recomendação:
1. **Implementar cleanup em TODOS os useEffects:**

```typescript
// ✅ CORRETO: Fetch com abort
useEffect(() => {
const controller = new AbortController();
```

```
  fetch('/api/data', { signal: controller.signal })
    .then(data => setState(data))
    .catch(error => {
      if (error.name !== 'AbortError') {
        console.error(error);
      }
    });

  return () => controller.abort();
```

}, []);

// ✅ CORRETO: Timer com cleanup
useEffect(() => {
const interval = setInterval(() => {
updateData();
}, 1000);

```
  return () => clearInterval(interval);
```

}, []);

// ✅ CORRETO: Event listener com remover
useEffect(() => {
const handleResize = () => setWidth(window.innerWidth);
window.addEventListener('resize', handleResize);

```
  return () => window.removeEventListener('resize', handleResize);
```

}, []);

// ✅ CORRETO: Subscription com unsubscribe
useEffect(() => {
const subscription = supabase
.channel('changes')
.on('postgres_changes', handleChange)
.subscribe();

```
  return () => {
    subscription.unsubscribe();
  };
```

}, []);
```

1. **Criar hooks customizados para padrões comuns:**

```typescript
// src/hooks/useInterval.ts
export function useInterval(callback: () => void, delay: number | null) {
const savedCallback = useRef(callback);
```
```

```
  useEffect(() => {
    savedCallback.current = callback;
  }, [callback]);

  useEffect(() => {
    if (delay === null) return;

    const id = setInterval(() => savedCallback.current(), delay);
    return () => clearInterval(id);
  }, [delay]);
```

}

// src/hooks/useEventListener.ts
export function useEventListener(
eventName: K,
handler: (event: WindowEventMap[K]) => void,
element: Window | HTMLElement = window
) {
const savedHandler = useRef(handler);

```
  useEffect(() => {
    savedHandler.current = handler;
  }, [handler]);

  useEffect(() => {
    const eventListener = (event: Event) =>
      savedHandler.current(event as WindowEventMap[K]);

    element.addEventListener(eventName, eventListener);
    return () => element.removeEventListener(eventName, eventListener);
  }, [eventName, element]);
```

}
```

1. **Auditar todos os useEffects:**
   `bash`
   `# Encontrar useEffects sem return`
   `grep -r "useEffect" src --include="*.tsx" --include="*.ts" -A 10 | grep -v "return () => " > effects-without-cleanup.txt`

2. **Adicionar ESLint rule:**
   `json`
   `{`
   `"rules": {`
   `"react-hooks/exhaustive-deps": "error"`
   `}`
   `}`

**Prioridade:** P1 - ALTA (auditoria gradual)

---
```

12. Ausência de Error Boundaries Adequados

Severidade: 🔴 CRÍTICA
CWE-755: Tratamento Inadequado de Condições Excepcionais

Descrição:
Sistema não implementa Error Boundaries de forma granular, causando crash completo da aplicação quando ocorre erro em um componente.

Implementação Atual:
Usa `react-error-boundary`, mas não há evidências de uso granular em módulos críticos.

Impacto:
- Crash completo da aplicação por erro em um componente
- Perda de estado do usuário
- Experiência ruim do usuário
- Dificuldade de debugging

Recomendação:
1. **Criar Error Boundary customizado:**

```typescript
// src/components/error-boundary/ModuleErrorBoundary.tsx
import { Component, ReactNode } from 'react';
import { logger } from '@/lib/logger';

interface Props {
children: ReactNode;
fallback?: ReactNode;
moduleName: string;
onError?: (error: Error, errorInfo: React.ErrorInfo) => void;
}

interface State {
hasError: boolean;
error: Error | null;
}

export class ModuleErrorBoundary extends Component {
constructor(props: Props) {
super(props);
this.state = { hasError: false, error: null };
}
```

```
  static getDerivedStateFromError(error: Error): State {
    return { hasError: true, error };
  }

  componentDidCatch(error: Error, errorInfo: React.ErrorInfo) {
    logger.error(`Error in ${this.props.moduleName}:`, error, errorInfo);

    // Enviar para Sentry
    if (window.Sentry) {
      window.Sentry.captureException(error, {
        contexts: {
          react: {
            componentStack: errorInfo.componentStack,
          },
        },
        tags: {
          module: this.props.moduleName,
        },
      });
    }

    this.props.onError?.(error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      if (this.props.fallback) {
        return this.props.fallback;
      }

      return (
        <div className="flex flex-col items-center justify-center p-8 text-center">
          <div className="text-destructive text-xl mb-4">
            ⚠️ Erro no módulo {this.props.moduleName}
          </div>
          <p className="text-muted-foreground mb-4">
            Ocorreu um erro neste módulo. O resto da aplicação continua funcionando.
          </p>
          <button
            onClick={() => this.setState({ hasError: false, error: null })}
            className="px-4 py-2 bg-primary text-primary-foreground rounded"
          >
            Tentar novamente
          </button>
        </div>
      );
    }

    return this.props.children;
  }
```

}
```

1. **Envolver todos os módulos:**

```typescript
// App.tsx

```

1. **Criar boundaries específicos:**

```typescript
  // src/components/error-boundary/AsyncBoundary.tsx
  export function AsyncBoundary({ children }: { children: ReactNode }) {
    return (
      <ErrorBoundary
        fallback={
          <div className="flex items-center justify-center p-4">
            <div className="text-center">
              <p>Erro ao carregar componente</p>
              <button onClick={() => window.location.reload()}>
                Recarregar
              </button>
            </div>
          </div>
        }
      >
        <Suspense fallback={<LoadingSpinner />}>
          {children}
        </Suspense>
      </ErrorBoundary>
    );
  }
```

Prioridade: P1 - ALTA

---

13. CORS e Configurações de Segurança HTTP Inadequadas

Severidade: 🔴 CRÍTICA
CWE-346: Origin Validation Error

Descrição:
Não há evidências de configuração CORS adequada, potencialmente permitindo requisições de origens não confiáveis.

Impacto:
- CSRF (Cross-Site Request Forgery)
- Roubo de dados por sites maliciosos
- Replay attacks
- Sessões sequestradas

Recomendação:
1. **Configurar CORS no Supabase:**
```

```sql
   -- Configurar no Supabase Dashboard > Settings > API
   -- Allowed Origins: https://seu-dominio.com, https://www.seu-dominio.com
```

1. **Adicionar validação de origem em Edge Functions:**

```typescript
// supabase/functions/_shared/cors.ts
const ALLOWED_ORIGINS = [
'https://seu-dominio.com',
'https://www.seu-dominio.com',
...(Deno.env.get('DENO_ENV') === 'development' ? ['http://localhost:8080'] : [])
];

export function corsHeaders(origin: string | null): Record {
const allowedOrigin = origin && ALLOWED_ORIGINS.includes(origin)
? origin
: ALLOWED_ORIGINS[0];

  return {
    'Access-Control-Allow-Origin': allowedOrigin,
    'Access-Control-Allow-Headers': 'authorization, x-client-info, apikey, content-type',
    'Access-Control-Allow-Methods': 'POST, GET, OPTIONS, PUT, DELETE',
    'Access-Control-Max-Age': '86400',
  };

}

export function handleCors(req: Request): Response | null {
if (req.method === 'OPTIONS') {
return new Response('ok', {
headers: corsHeaders(req.headers.get('origin'))
});
}
return null;
}
```

1. **Usar em todas as Edge Functions:**

```typescript
import { handleCors, corsHeaders } from '../_shared/cors.ts';

Deno.serve(async (req) => {
const corsResponse = handleCors(req);
if (corsResponse) return corsResponse;
```

```
  // ... sua lógica

  return new Response(JSON.stringify(data), {
    headers: {
      ...corsHeaders(req.headers.get('origin')),
      'Content-Type': 'application/json'
    }
  });
```

});
```

1. **Implementar CSRF tokens:**

```typescript
// src/lib/csrf.ts
export function generateCsrfToken(): string {
const array = new Uint8Array(32);
crypto.getRandomValues(array);
return Array.from(array, byte => byte.toString(16).padStart(2, '0')).join('');
}

export function setCsrfToken(): string {
const token = generateCsrfToken();
document.cookie = `csrf_token=${token}; Secure; SameSite=Strict; Path=/`;
return token;
}

export function getCsrfToken(): string | null {
const match = document.cookie.match(/csrf_token=([^;]+)/);
return match ? match[1] : null;
}

export function validateCsrfToken(token: string): boolean {
const storedToken = getCsrfToken();
return storedToken === token;
}
```

Prioridade: P0 - IMEDIATA

---

14. Falta de Sanitização em URLs e Query Parameters

Severidade: 🔴 CRÍTICA
CWE-88: Construção Inadequada de Argumento

Descrição:
Múltiplas ocorrências de construção de URLs e query parameters sem sanitização adequada.

Exemplo Problemático:
```

```
// src/services/mapbox.ts (linha identificada)
`https://api.mapbox.com/geocoding/v5/mapbox.places/Rio%20de%20Janeiro.json?ac-
cess_token=${apiKey}&limit=1`
```

Impacto:
- Open Redirect
- Injeção de parâmetros maliciosos
- Bypass de validações
- Exposição de dados sensíveis via URL

Recomendação:
1. **Criar helper de URL segura:**

```typescript
// src/lib/url-builder.ts
export class SafeUrlBuilder {
private url: URL;

  constructor(baseUrl: string) {
    this.url = new URL(baseUrl);
  }

  addParam(key: string, value: string | number): this {
    // Sanitizar key e value
    const safeKey = encodeURIComponent(key);
    const safeValue = encodeURIComponent(String(value));
    this.url.searchParams.append(safeKey, safeValue);
    return this;
  }

  addPath(...segments: string[]): this {
    // Sanitizar path segments
    const safePath = segments
      .map(segment => encodeURIComponent(segment))
      .join('/');
    this.url.pathname += `/${safePath}`;
    return this;
  }

  toString(): string {
    return this.url.toString();
  }
```

}

// Uso
const url = new SafeUrlBuilder('https://api.mapbox.com/geocoding/v5/mapbox.places')
.addPath(query)
.addParam('access_token', apiKey)
.addParam('limit', 1)
.toString();
```

1. **Validar redirects:**
```

```typescript
// src/lib/redirect-validator.ts
const ALLOWED_DOMAINS = [
'seu-dominio.com',
'www.seu-dominio.com'
];

export function isValidRedirect(url: string): boolean {
try {
const parsed = new URL(url);
return ALLOWED_DOMAINS.some(domain =>
parsed.hostname === domain || parsed.hostname.endsWith( .${domain} )
);
} catch {
return false;
}
}

export function safeRedirect(url: string, fallback: string = '/'): void {
const target = isValidRedirect(url) ? url : fallback;
window.location.href = target;
}
```

1. **Sanitizar query parameters em rotas:**

```typescript
import { useSearchParams } from 'react-router-dom';

function Component() {
const [searchParams] = useSearchParams();

// ❌ ERRADO
const redirect = searchParams.get('redirect');
window.location.href = redirect; // Perigoso!

// ✅ CORRETO
const redirect = searchParams.get('redirect');
if (redirect && isValidRedirect(redirect)) {
  window.location.href = redirect;
} else {
  window.location.href = '/';
}

}
```

Prioridade: P1 - ALTA

---

15. Ausência de Timeout em Requisições HTTP

Severidade: 🔴 CRÍTICA (Availability)
CWE-400: Consumo Descontrolado de Recursos

Descrição:
204 ocorrências de `fetch()` no código sem configuração de timeout, causando travamento da UI em caso de falha de rede.

Impacto:
- UI congelada indefinidamente
- Usuário sem feedback
- Consumo de memória por requisições pendentes
- Degradação da experiência

Recomendação:
1. **Criar wrapper de fetch com timeout:**

```typescript
// src/lib/fetch-with-timeout.ts
export interface FetchOptions extends RequestInit {
timeout?: number;
retries?: number;
retryDelay?: number;
}

export async function fetchWithTimeout(
url: string,
options: FetchOptions = {}
): Promise {
const {
timeout = 30000, // 30 segundos padrão
retries = 0,
retryDelay = 1000,
...fetchOptions
} = options;
```

```
  const controller = new AbortController();
  const timeoutId = setTimeout(() => controller.abort(), timeout);

  try {
    const response = await fetch(url, {
      ...fetchOptions,
      signal: controller.signal
    });

    clearTimeout(timeoutId);

    if (!response.ok && retries > 0) {
      await new Promise(resolve => setTimeout(resolve, retryDelay));
      return fetchWithTimeout(url, { ...options, retries: retries - 1 });
    }

    return response;
  } catch (error) {
    clearTimeout(timeoutId);

    if (error.name === 'AbortError') {
      throw new Error(`Request timeout after ${timeout}ms`);
    }

    if (retries > 0) {
      await new Promise(resolve => setTimeout(resolve, retryDelay));
      return fetchWithTimeout(url, { ...options, retries: retries - 1 });
    }

    throw error;
  }
```

}
```

1. **Usar em todo o código:**

```typescript
import { fetchWithTimeout } from '@/lib/fetch-with-timeout';

// Em vez de fetch(url)
const response = await fetchWithTimeout(url, {
timeout: 10000,
retries: 3,
retryDelay: 1000
});
```

1. **Configurar React Query com timeout:**

```typescript
// src/lib/performance/query-config.ts
export function createOptimizedQueryClient() {
return new QueryClient({
defaultOptions: {
queries: {
staleTime: 5 * 60 * 1000,
gcTime: 10 * 60 * 1000,
```
```

retry: 3,
retryDelay: (attemptIndex) => Math.min(1000 * 2 ** attemptIndex, 30000),
networkMode: ‘offlineFirst’,
// Adicionar timeout global
queryFn: async ({ queryKey, signal }) => {
const timeout = setTimeout(() => {
signal?.abort();
}, 30000);

```
        try {
          const response = await fetch(queryKey[0] as string, { signal });
          clearTimeout(timeout);
          return response.json();
        } catch (error) {
          clearTimeout(timeout);
          throw error;
        }
      }
    },
    mutations: {
      retry: 1,
      networkMode: 'offlineFirst',
    },
  },
});
```

}
```

**Prioridade:** P1 - ALTA

---

## 16. Gerenciamento Inadequado de Secrets em CI/CD

**Severidade:** 🔴 CRÍTICA
**CWE-522:** Credenciais Insuficientemente Protegidas

**Descrição:**
Arquivos `.env.*` commitados no repositório contêm configurações sensíveis que podem expor secrets.

**Arquivos Identificados:**

```
.env.development
.env.production
.env.staging
.env.example
```

Impacto:
- Exposição de API keys se commitadas
- Vazamento de credenciais no histórico do Git
- Acesso não autorizado a serviços
```

Recomendação:
1. **Verificar se há secrets commitados:**

```bash
# Verificar histórico
git log –all –full-history – .env.production
git log –all –full-history – .env.development
git log –all –full-history – .env.staging

# Se houver secrets, usar git-filter-repo
git filter-repo –path .env.production –invert-paths
git filter-repo –path .env.development –invert-paths
git filter-repo –path .env.staging –invert-paths
```

1. **Atualizar .gitignore:**

```
# .gitignore
.env
.env.local
.env.*.local
.env.development
.env.production
.env.staging

# Manter apenas
.env.example
```

1. **Implementar validação de secrets:**

```bash
# .github/workflows/secret-scan.yml
name: Secret Scan

on: [push, pull_request]

jobs:
secret-scan:
runs-on: ubuntu-latest
steps:
- uses: actions/checkout@v3
with:
fetch-depth: 0
```

```
      - name: Gitleaks Scan
        uses: gitleaks/gitleaks-action@v2
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

```

1. **Usar secrets management adequado:**
```

```bash
# Vercel
vercel env add VITE_SUPABASE_URL production
vercel env add VITE_SUPABASE_PUBLISHABLE_KEY production

# Netlify
netlify env:set VITE_SUPABASE_URL “value” –context production
netlify env:set VITE_SUPABASE_PUBLISHABLE_KEY “value” –context production
```

1. **Documentar no README:**

```markdown
## Environment Variables

Required environment variables (never commit these):

```bash
# Copy .env.example to .env.local
cp .env.example .env.local

# Fill in your values
VITE_SUPABASE_URL=your_url_here
VITE_SUPABASE_PUBLISHABLE_KEY=your_key_here
```

Prioridade: P0 - IMEDIATA

---

17. Falta de Monitoramento e Alertas

Severidade: 🔴 CRÍTICA (Observability)
CWE-778: Registro Inadequado

Descrição:
Sistema não implementa monitoramento adequado de erros, performance e uso, dificultando identificação proativa de problemas.

Problemas:
- Sentry configurado mas sem captura sistemática
- Sem métricas de performance
- Sem alertas automáticos
- Dificuldade de troubleshooting

Recomendação:
1. **Configurar Sentry adequadamente:**

```typescript
// src/lib/monitoring/sentry.ts
import * as Sentry from ‘@sentry/react’;
import { BrowserTracing } from ‘@sentry/tracing’;

export function initSentry() {
if (import.meta.env.PROD && import.meta.env.VITE_SENTRY_DSN) {
```
```

Sentry.init({
dsn: import.meta.env.VITE_SENTRY_DSN,
environment: import.meta.env.MODE,
integrations: [
new BrowserTracing({
tracingOrigins: [‘localhost’, /^\//],
routingInstrumentation: Sentry.reactRouterV6Instrumentation(
React.useEffect,
useLocation,
useNavigationType,
createRoutesFromChildren,
matchRoutes
),
}),
new Sentry.Replay({
maskAllText: true,
blockAllMedia: true,
}),
],

```
    // Performance Monitoring
    tracesSampleRate: 0.1, // 10% das transações

    // Session Replay
    replaysSessionSampleRate: 0.1,
    replaysOnErrorSampleRate: 1.0,

    // Filtragem de erros
    beforeSend(event, hint) {
      // Ignorar erros conhecidos e não críticos
      if (event.exception?.values?.[0]?.value?.includes('ResizeObserver')) {
        return null;
      }

      // Adicionar contexto
      event.tags = {
        ...event.tags,
        userAgent: navigator.userAgent,
        viewport: `${window.innerWidth}x${window.innerHeight}`,
      };

      return event;
    },

    // Ignorar URLs específicas
    ignoreErrors: [
      'Non-Error promise rejection captured',
      'ResizeObserver loop limit exceeded',
      'ChunkLoadError',
    ],
  });
}
```

}
```

1. **Implementar métricas customizadas:**

```typescript
// src/lib/monitoring/metrics.ts
import * as Sentry from '@sentry/react';

export const metrics = {
// Performance
recordPageLoad(page: string, duration: number) {
Sentry.metrics.distribution('page.load.duration', duration, {
tags: { page }
});
},

  // Business metrics
  recordAction(action: string, success: boolean) {
    Sentry.metrics.increment('user.action', 1, {
      tags: { action, success: success.toString() }
    });
  },

  // API metrics
  recordApiCall(endpoint: string, statusCode: number, duration: number) {
    Sentry.metrics.distribution('api.duration', duration, {
      tags: { endpoint, statusCode: statusCode.toString() }
    });

    if (statusCode >= 400) {
      Sentry.metrics.increment('api.error', 1, {
        tags: { endpoint, statusCode: statusCode.toString() }
      });
    }
  },

  // Resource usage
  recordMemoryUsage() {
    if ('memory' in performance) {
      const memory = (performance as any).memory;
      Sentry.metrics.gauge('memory.used', memory.usedJSHeapSize, {
        unit: 'byte'
      });
    }
  },

};
```

1. **Implementar health checks:**

```typescript
// src/lib/monitoring/health-check.ts
export interface HealthCheck {
name: string;
status: 'healthy' | 'degraded' | 'unhealthy';
responseTime?: number;
```
```

error?: string;
}

export async function checkHealth(): Promise {
const checks: HealthCheck[] = [];

```
  // Supabase health
  checks.push(await checkSupabase());

  // API health
  checks.push(await checkAPI());

  // Storage health
  checks.push(await checkStorage());

  return checks;
```

}

async function checkSupabase(): Promise {
const start = Date.now();

```
  try {
    const { error } = await supabase.from('health_check').select('*').limit(1);

    return {
      name: 'Supabase',
      status: error ? 'unhealthy' : 'healthy',
      responseTime: Date.now() - start,
      error: error?.message
    };
  } catch (error) {
    return {
      name: 'Supabase',
      status: 'unhealthy',
      responseTime: Date.now() - start,
      error: error.message
    };
  }
```

}
```

1. **Configurar alertas:**

```yaml
# alerting-rules.yml (para Sentry)
- name: Error Rate Alert
metric: error.rate
threshold: 10
window: 5m
notification: slack, email
```

- name: API Response Time
  metric: api.duration.p95
  threshold: 3000
```

window: 5m
notification: slack

- name: Memory Usage
  metric: memory.used
  threshold: 512MB
  window: 1m
  notification: slack
  ```

**Prioridade:** P1 - ALTA

---

## 18. Problemas de Acessibilidade (a11y)

**Severidade:** 🔴 CRÍTICA (Compliance/UX)
**WCAG 2.1:** Múltiplas violações

**Descrição:**
Múltiplos problemas de acessibilidade identificados que violam WCAG 2.1 e podem impedir uso por pessoas com deficiência.

**Problemas Identificados:**

### 18.1 Elementos Clicáveis sem Suporte a Teclado

- **Quantidade:** 3.811 elementos `onClick` sem `onKeyDown` / `onKeyPress`
- **Impacto:** Usuários de teclado não conseguem navegar

### 18.2 Imagens sem Texto Alternativo

- **Quantidade:** 34 elementos `&lt;img&gt;` sem atributo `alt`
- **Impacto:** Screen readers não conseguem descrever imagens

### 18.3 Listas sem Keys

- **Impacto:** Problemas de performance e acessibilidade

**Recomendação:**
1. **Adicionar suporte a teclado em elementos clicáveis:**

```typescript
// ❌ ERRADO


Click me

// ✅ CORRETO


{ if (e.key === 'Enter' || e.key === ' ') { e.preventDefault(); handleClick(); } }} > Click me

// ✅ MELHOR: Usar elemento semântico
<button onClick={handleClick}>Click me</button>
```

1. **Adicionar alt em todas as imagens:**
```

```typescript
// ❌ ERRADO

// ✅ CORRETO

Nautilus One Logo

// Para imagens decorativas

```

1. **Adicionar keys em listas:**

```typescript
// ❌ ERRADO
{items.map(item =>

{item.name}
)}

// ✅ CORRETO
{items.map(item =>

{item.name}
)}
```

1. **Implementar landmarks ARIA:**

```typescript


    ...
```

Dashboard

...
...
```

1. **Adicionar labels em formulários:**

```typescript
// ❌ ERRADO


// ✅ CORRETO
Email


// Ou com aria-label
```
```

```

1. **Configurar testes de acessibilidade:**

```typescript
// tests/a11y/accessibility.test.ts
import { axe, toHaveNoViolations } from 'jest-axe';

expect.extend(toHaveNoViolations);

describe('Accessibility', () => {
it('should not have accessibility violations', async () => {
const { container } = render();
const results = await axe(container);
expect(results).toHaveNoViolations();
});
});
```

1. **Adicionar script de validação:**

```
json
  {
    "scripts": {
      "test:a11y": "playwright test --grep @a11y"
    }
  }
```

Prioridade: P1 - ALTA (compliance legal)

🟡 FALHAS MÉDIAS

19. Performance: Bundle Size Excessivo

Severidade: 🟡 MÉDIA
CWE-400: Consumo Descontrolado de Recursos

Descrição:
Configuração de code splitting existe mas pode ser otimizada. Múltiplas bibliotecas pesadas carregadas desnecessariamente.

Bibliotecas Pesadas Identificadas:

```
- @tensorflow/tfjs: ~100KB+ (AI/ML)
- three.js: ~500KB (3D graphics)
- mapbox-gl: ~200KB (maps)
- recharts + chart.js: ~150KB (charts duplicados)
- @tiptap/*: ~100KB (editor)
- firebase: ~100KB (se usado)
```

Impacto:
- Tempo de carregamento inicial lento
```

- Consumo excessivo de banda
- Experiência ruim em conexões lentas
- Bounce rate alto

Recomendação:
1. **Análise de bundle:**
`bash`
`npm install -D rollup-plugin-visualizer`

```typescript
// vite.config.ts
import { visualizer } from 'rollup-plugin-visualizer';

plugins: [
visualizer({
open: true,
gzipSize: true,
brotliSize: true,
})
]
```

1. **Lazy load bibliotecas pesadas:**

```typescript
// ❌ ERRADO: Import no topo
import mapboxgl from 'mapbox-gl';

// ✅ CORRETO: Dynamic import
const MapComponent = lazy(() => import('./components/Map'));

// Ou dentro de função
async function initMap() {
const mapboxgl = await import('mapbox-gl');
// usar mapboxgl
}
```

1. **Remover charts duplicados:**
   - Manter apenas recharts OU chart.js
   - Preferir recharts (mais leve e React-native)

2. **Tree-shaking otimizado:**

```typescript
// ❌ ERRADO
import _ from 'lodash';
_.debounce(fn, 300);

// ✅ CORRETO
import debounce from 'lodash/debounce';
```

debounce(fn, 300);
```

1. **Implementar route-based code splitting:**

```typescript
const routes = [
{
path: '/maps',
component: lazy(() => import('./pages/Maps')),
preload: () => import('./pages/Maps')
},
{
path: '/3d-view',
component: lazy(() => import('./pages/3DView')),
preload: () => import('./pages/3DView')
}
];

// Preload em hover
routes[0].preload()}
```

> Maps
>
> ```

1. **Configurar budget no Vite:**

```typescript
   // vite.config.ts
   build: {
     rollupOptions: {
       output: {
         manualChunks: {
           // ... existing chunks
         }
       }
     },
     // Adicionar warnings
     chunkSizeWarningLimit: 500, // Já configurado
   }
```

Meta de Bundle Size:
- **Initial Bundle:** < 200KB (gzipped)
- **Per Route:** < 100KB (gzipped)
- **Total:** < 2MB (gzipped)

Prioridade: P2 - MÉDIA

```

20. Re-renders Excessivos e Falta de Memoização

Severidade: 🟡 MÉDIA
Performance Impact: Alto

Descrição:
Apesar de 1.220 usos de memoização (React.memo, useMemo, useCallback), análise indica que muitos componentes ainda sofrem re-renders desnecessários.

Padrões Problemáticos:

```
// ❌ PROBLEMA 1: Props inline objects
<Component data={{ id: 1, name: 'Test' }} />

// ❌ PROBLEMA 2: Funções inline
<Button onClick={() => handleClick(id)} />

// ❌ PROBLEMA 3: Array/Object em useState
const [data, setData] = useState([]);

// ❌ PROBLEMA 4: Context sem memoização
const value = { user, setUser, permissions };
return <Context.Provider value={value}>{children}</Context.Provider>
```

Recomendação:
1. **Usar React DevTools Profiler para identificar:**
`bash`
`  # Instalar extensão Chrome/Firefox`
`  # Analisar componentes com re-renders frequentes`

1. **Memoizar callbacks:**

```typescript
// ❌ ERRADO
```

```

1. **Memoizar valores computados:**

```typescript
// ❌ ERRADO
const filteredData = data.filter(item => item.active);

// ✅ CORRETO
const filteredData = useMemo(
  () => data.filter(item => item.active),
  [data]
);
```

1. **Memoizar contexts:**

```typescript
function AuthProvider({ children }) {
  const [user, setUser] = useState(null);
  const [permissions, setPermissions] = useState([]);

  // ✅ Memoizar value
  const value = useMemo(
    () => ({ user, setUser, permissions, setPermissions }),
    [user, permissions]
  );

  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
}
```

1. **Usar React.memo adequadamente:**
```

* Submit

*

* ```
/
interface ButtonProps {
/ Variante visual do botão /
variant?: ‘primary’ | ‘secondary’ | ‘destructive’;
/ **Se true, mostra spinner e desabilita */**
loading?: boolean;
/ Função chamada ao clicar /
onClick?: () => void;
/ Conteúdo do botão /
children: React.ReactNode;
}

export function Button({ variant = ‘primary’, loading, onClick, children }: ButtonProps) {
// Implementation
}
```

1. **Gerar documentação automática:**

```
bash
  npm install -D typedoc
```

```
json
  {
    "scripts": {
      "docs:generate": "typedoc --out docs/api src"
    }
  }
```

1. **Criar guia de contribuição:**
   - Padrões de código
   - Como testar
   - Como documentar
   - Processo de review

Prioridade: P3 - BAIXA (melhoria contínua)

26-40. Outras Falhas Médias

Por questões de espaço, listo resumidamente as demais falhas médias:

26. Falta de Retry Logic em Requisições Críticas (P2)
27. Estado Global Não Otimizado (P2)
28. Websocket/Realtime sem Reconnection Logic (P3)
29. Falta de Pagination em Listas Grandes (P2)
30. Images sem Lazy Loading (P2)
```

31. **Fonts não Otimizadas** (P3)
32. **Service Worker não Versionado** (P3)
33. **Logs de Debug em Produção** (Já abordado em #5)
34. **Hard-coded Strings sem i18n** (P3)
35. **Falta de Feature Flags** (P3)
36. **Ausência de Analytics Adequado** (P3)
37. **Backup e Disaster Recovery não Documentado** (P2)
38. **Falta de API Documentation (OpenAPI/Swagger)** (P3)
39. **Git Commits sem Padrão** (P4)
40. **Falta de Code Owners** (P4)

🟢 FALHAS BAIXAS

41. ESLint Rules Muito Permissivas

Severidade: 🟢 BAIXA
Code Quality Impact: Baixo

Descrição:
ESLint configurado com muitas rules em “warn” ao invés de “error”, permitindo code smells.

Configuração Atual:

```
{
  "rules": {
    "@typescript-eslint/no-unused-vars": "warn",  // ❌ Deveria ser "error"
    "no-unused-vars": "warn",                     // ❌ Deveria ser "error"
    "@typescript-eslint/no-explicit-any": "warn", // ❌ Deveria ser "error"
    "@typescript-eslint/ban-ts-comment": "warn",  // ❌ Deveria ser "error"
    "react/no-unescaped-entities": "warn",        // ❌ Deveria ser "error"
    "no-empty": "warn"                            // ❌ Deveria ser "error"
  }
}
```

Recomendação:

```
{
  "extends": [
    "eslint:recommended",
    "plugin:react/recommended",
    "plugin:@typescript-eslint/recommended",
    "plugin:react-hooks/recommended",
    "prettier"
  ],
  "rules": {
    "react/react-in-jsx-scope": "off",
    "@typescript-eslint/no-unused-vars": ["error", {
      "argsIgnorePattern": "^_",
      "varsIgnorePattern": "^_"
    }],
    "@typescript-eslint/no-explicit-any": "error",
    "@typescript-eslint/ban-ts-comment": "error",
    "no-console": ["error", { "allow": ["error"] }],
    "react-hooks/exhaustive-deps": "error",
    "react/prop-types": "off",
    "semi": ["error", "always"],
    "quotes": ["error", "double"]
  }
}
```

Prioridade: P4 - BAIXA

---

42. Prettier Configuration Inconsistente

Severidade: 🟢 BAIXA

Recomendação:

```
{
  "semi": true,
  "trailingComma": "es5",
  "singleQuote": false,
  "printWidth": 100,
  "tabWidth": 2,
  "useTabs": false,
  "arrowParens": "always",
  "endOfLine": "lf"
}
```

Prioridade: P4 - BAIXA

---

43. Git Ignore Incompleto

Severidade: 🟢 BAIXA

Arquivos a Adicionar:

```
# Logs
logs
*.log
npm-debug.log*
yarn-debug.log*
yarn-error.log*
pnpm-debug.log*
lerna-debug.log*

# Build artifacts
dist
dist-ssr
*.local
.vite*
.vercel*

# Environment files
.env
.env.local
.env.*.local

# IDE
.vscode/*
!.vscode/extensions.json
.idea
*.swp
*.swo
*~

# OS
.DS_Store
Thumbs.db

# Coverage
coverage
.nyc_output

# Temporary files
*.tmp
*.temp

# Database
*.sqlite
*.sqlite3
*.db
```

Prioridade: P4 - BAIXA

44-57. Outras Falhas Baixas

Por questões de espaço, listo resumidamente:

44. Falta de Commit Hooks (Husky) (P4)
45. Changelog não Automatizado (P4)
46. README sem Badges (P4)
47. Falta de Contributing Guidelines (P4)
48. Issue Templates Ausentes (P4)

49. Pull Request Template Ausente (P4)
50. Semantic Versioning não Seguido (P4)
51. Package.json sem Keywords (P4)
52. License File Presente mas Tipo Não Especificado (P4)
53. Dependências Dev em Produção (Verificar)
54. Scripts NPM não Documentados (P4)
55. Arquivos de Lock Múltiplos (Verificar package-lock.json e bun.lockb)
56. Variáveis de Ambiente não Documentadas (P3)
57. Favicon não Otimizado (P4)

📋 Recomendações Priorizadas

Prioridade P0 - IMEDIATA (Executar em 24-48h)

1. **[#1]** Remover credenciais hardcoded do Supabase
2. **[#2]** Implementar Content Security Policy
3. **[#3]** Corrigir vulnerabilidades NPM (especialmente HIGH)
4. **[#4]** Sanitizar todos os `dangerouslySetInnerHTML`
5. **[#6]** Implementar rate limiting
6. **[#8]** Auditar autenticação em rotas protegidas
7. **[#9]** Implementar validação de input com Zod
8. **[#13]** Configurar CORS adequadamente
9. **[#16]** Remover secrets do Git e implementar secrets management

Prioridade P1 - ALTA (Executar em 1-2 semanas)

1. **[#5]** Substituir console.log por logger centralizado
2. **[#7]** Habilitar TypeScript strict mode (gradual)
3. **[#10]** Implementar SafeStorage para localStorage
4. **[#11]** Auditar e corrigir memory leaks em useEffect
5. **[#12]** Implementar Error Boundaries granulares
6. **[#14]** Sanitizar URLs e query parameters
7. **[#15]** Adicionar timeout em todas as requisições
8. **[#17]** Configurar monitoramento com Sentry
9. **[#18]** Corrigir problemas de acessibilidade

Prioridade P2 - MÉDIA (Executar em 1 mês)

1. **[#19]** Otimizar bundle size
2. **[#20]** Reduzir re-renders desnecessários
3. **[#23]** Aumentar cobertura de testes
4. **[#37]** Documentar backup e disaster recovery

Prioridade P3 - BAIXA (Backlog)

1. **[#22]** Atualizar dependências desatualizadas
2. **[#24]** Refatorar funções longas
3. **[#25]** Adicionar documentação JSDoc
4. **[#28-#40]** Outras melhorias de qualidade

Prioridade P4 - MUITO BAIXA (Nice to have)

1. **[#41-#57]** Melhorias de DevEx e tooling

📊 Métricas de Qualidade do Código

Complexidade

- **Arquivos Analisados:** ~2.500+
- **Linhas de Código:** ~150.000+
- **Módulos:** 30+
- **Componentes:** ~500+

Dívida Técnica Estimada

- **Critical Issues:** 18 (estimativa: 3-4 semanas de trabalho)
- **Medium Issues:** 32 (estimativa: 8-10 semanas)
- **Low Issues:** 17 (estimativa: 2-3 semanas)
- **Total:** ~15-17 semanas de trabalho para resolução completa

Risco de Segurança

🔴 **ALTO**
- Credenciais expostas
- Falta de CSP
- Vulnerabilidades em dependências
- XSS via dangerouslySetInnerHTML

Risco de Performance

🟡 **MÉDIO**
- Bundle size grande
- Re-renders excessivos
- Falta de lazy loading
- Memory leaks potenciais

Risco de Manutenibilidade

🟡 **MÉDIO-ALTO**
- 1.592 tipos `any` / `unknown`
- Funções muito longas
- Falta de documentação
- Código duplicado

🎯 Roadmap de Correção Sugerido

Sprint 1 (Semana 1-2): Segurança Crítica

- [ ] Remover credenciais hardcoded
- [ ] Implementar CSP
- [ ] Corrigir vulnerabilidades NPM

- [ ] Sanitizar dangerouslySetInnerHTML
- [ ] Configurar CORS

Sprint 2 (Semana 3-4): Validação e Autenticação

- [ ] Implementar validação com Zod
- [ ] Auditar rotas protegidas
- [ ] Implementar rate limiting
- [ ] Configurar secrets management

Sprint 3 (Semana 5-6): Performance e Logging

- [ ] Substituir console.log
- [ ] Implementar SafeStorage
- [ ] Adicionar timeout em requisições
- [ ] Configurar Sentry

Sprint 4 (Semana 7-8): Quality Assurance

- [ ] Corrigir memory leaks
- [ ] Implementar Error Boundaries
- [ ] Corrigir acessibilidade
- [ ] Aumentar cobertura de testes

Sprint 5+ (Semanas 9-17): Otimização e Refatoração

- [ ] Otimizar bundle size
- [ ] Habilitar strict mode TypeScript
- [ ] Refatorar código duplicado
- [ ] Melhorar documentação

📝 Conclusão

O sistema Travel HR Buddy (Nautilus One) é uma aplicação complexa e ambiciosa com **múltiplos problemas críticos de segurança e qualidade** que requerem atenção imediata.

Pontos Positivos ✅

- Arquitetura modular bem estruturada
- Uso de tecnologias modernas (React 19, TypeScript 5.8, Vite)
- Implementação de PWA e offline-first
- Estrutura de testes existente
- Documentação extensa em `/docs`
- Configuração de build otimizada

Pontos Críticos ⚠️

- **URGENTE:** Credenciais expostas no código fonte
- **URGENTE:** Falta de Content Security Policy
- **URGENTE:** Vulnerabilidades em dependências
- **URGENTE:** XSS via dangerouslySetInnerHTML
- Alto número de tipos `any` (1.592)

- Console logs em produção (1.404)
- Falta de validação de input adequada

Recomendação Final

É IMPERATIVO corrigir as 9 falhas P0 (Prioridade Imediata) antes de considerar o sistema production-ready. As falhas de segurança identificadas expõem o sistema e os dados dos usuários a riscos significativos.

Após correção das falhas críticas, o sistema deve passar por:
1. Auditoria de segurança externa (pentest)
2. Revisão de código por especialista em segurança
3. Testes de carga e stress
4. Validação de compliance (LGPD/GDPR)

Estimativa de Tempo para Production-Ready:
- Correção de falhas P0: 1-2 semanas (urgente)
- Correção de falhas P1: 2-4 semanas
- Testes e validação: 1-2 semanas
- **Total:** 4-8 semanas para um sistema seguro e confiável

📞 Contato e Próximos Passos

Para dúvidas sobre este relatório ou assistência na implementação das correções:

1. **Priorizar falhas P0** - Começar imediatamente
2. **Criar issues** no GitHub para cada falha
3. **Estabelecer sprints** de correção
4. **Configurar CI/CD** para prevenir regressões
5. **Implementar code review** obrigatório

Relatório gerado por: DeepAgent - Sistema de Análise Automatizada
Data: 11 de dezembro de 2025
Versão do Relatório: 1.0

Apêndices

A. Ferramentas Recomendadas

- **Segurança:** Snyk, GitGuardian, Gitleaks
- **Qualidade:** SonarQube, CodeClimate
- **Testes:** Playwright, Vitest, Testing Library
- **Monitoramento:** Sentry, Datadog, New Relic
- **Bundle Analysis:** Webpack Bundle Analyzer, Rollup Visualizer

B. Recursos Úteis

- OWASP Top 10 (https://owasp.org/www-project-top-ten/)

- TypeScript Strict Mode Guide (https://www.typescriptlang.org/tsconfig#strict)
- React Performance Optimization (https://react.dev/learn/render-and-commit)
- Web Accessibility Guidelines (https://www.w3.org/WAI/WCAG21/quickref/)
- Content Security Policy Reference (https://content-security-policy.com/)

C. Scripts de Automação

```
# Script para buscar problemas comuns
#!/bin/bash

echo "🔍 Buscando credenciais hardcoded..."
grep -r "password\s*=\s*[\"\']" src/

echo "🔍 Buscando console.log..."
grep -r "console\." src/ | wc -l

echo "🔍 Buscando any types..."
grep -r ":\s*any" src/ | wc -l

echo "🔍 Buscando dangerouslySetInnerHTML..."
grep -r "dangerouslySetInnerHTML" src/ | wc -l

echo "✅ Análise concluída!"
```

FIM DO RELATÓRIO