Resumo - Frontend React

Visão Geral

Interface moderna e responsiva desenvolvida em **React com TypeScript**, oferecendo busca unificada em dados governamentais brasileiros com design sóbrio e rolagem dinâmica.

Tecnologias Principais

- **React 18** - Framework frontend moderno
- **TypeScript** - Tipagem estática para JavaScript
- **Vite** - Build tool rápido e moderno
- **CSS Modules** - Estilos componentizados
- **Axios** - Cliente HTTP para APIs

Arquitetura

Estrutura de Diretórios

```
frontend/src/
├── components/          # Componentes reutilizáveis
│   ├── common/         # Button, Input, Modal, Loading
│   ├── layout/         # Header, Footer, Sidebar
│   └── search/         # SearchBar, Filters, Results
├── pages/              # Home, Search, About
├── hooks/              # Custom hooks (useSearch, useInfiniteScroll)
├── services/           # API services e tipos
├── context/            # Context API (Search, Theme)
├── utils/              # Utilitários e helpers
└── styles/             # Estilos globais e variáveis
```

Componentes Principais

1. SearchBar - Barra de Busca Inteligente

```
interface SearchBarProps {
  onSearch: (query: string) => void;
  suggestions?: string[];
  loading?: boolean;
}
```

Funcionalidades: - Autocomplete com sugestões - Debounce para otimização - Histórico de buscas - Validação em tempo real

2. SearchFilters - Filtros Avançados

```
interface FilterOptions {
  apis: string[];
  categories: string[];
  dateRange: { start: Date | null; end: Date | null };
}
```

Funcionalidades: - Filtros por API/fonte - Filtros por categoria - Seleção de período - Filtros persistentes

3. InfiniteScroll - Rolagem Dinâmica

```
const InfiniteScroll: React.FC<{
  hasMore: boolean;
  loading: boolean;
  onLoadMore: () => void;
  children: React.ReactNode;
}>
```

Funcionalidades: - Carregamento automático - Skeleton loading - Indicador de fim - Performance otimizada

4. ResultCard - Card de Resultado

```
interface SearchResult {
  id: string;
  source: string;
  category: string;
  title: string;
  description: string;
  data: Record<string, any>;
  relevance: number;
  timestamp: string;
}
```

Funcionalidades: - Design responsivo - Badges de fonte/categoria - Ações contextuais - Expansão de detalhes

Custom Hooks

useSearch - Gerenciamento de Busca

```
interface UseSearchReturn {
  results: SearchResult[];
  loading: boolean;
  error: string | null;
  hasMore: boolean;
  search: (query: string, filters?: FilterOptions) => void;
  loadMore: () => void;
  clearResults: () => void;
}
```

useInfiniteScroll - Rolagem Infinita

```
const useInfiniteScroll = (callback: () => void) => {
  // Detecta scroll no final da página
  // Chama callback para carregar mais dados
  // Gerencia estado de carregamento
}
```

useDebounce - Otimização de Performance

```
const useDebounce = (value: string, delay: number) => {
  // Atrasa execução de funções
  // Reduz chamadas desnecessárias à API
  // Melhora experiência do usuário
}
```

Design System

Paleta de Cores

```
:root {
  --primary-color: #1e40af;      /* Azul governo */
  --secondary-color: #64748b;    /* Cinza neutro */
  --accent-color: #059669;       /* Verde destaque */
  --background-color: #f8fafc;   /* Fundo claro */
  --surface-color: #ffffff;      /* Superfícies */
  --text-primary: #1e293b;       /* Texto principal */
  --text-secondary: #64748b;     /* Texto secundário */
}
```

Tipografia

- **Fonte**: Inter (moderna e legível)
- **Escalas**: 0.75rem a 1.875rem
- **Pesos**: 400 (regular), 500 (medium), 600 (semibold)
- **Hierarquia**: Clara e consistente

Espaçamento

- **Sistema**: Múltiplos de 0.25rem (4px)
- **Escala**: 0.25rem a 3rem
- **Consistência**: Grid de 8px
- **Responsivo**: Adapta-se ao dispositivo

Responsividade

Breakpoints

Dispositivo	Largura	Layout
Mobile	320px - 768px	Stack vertical, menu hambúrguer
Tablet	768px - 1024px	Layout híbrido, sidebar colapsável
Desktop	1024px+	Layout completo, sidebar fixa

Adaptações Mobile

- **Menu hambúrguer** para navegação
- **Filtros em modal** para economizar espaço
- **Cards empilhados** verticalmente
- **Touch gestures** para interação
- **Botões maiores** para facilitar toque

Gerenciamento de Estado

Context API

```
interface SearchContextType {
  searchState: SearchState;
  search: (query: string, filters?: FilterOptions) => void;
  loadMore: () => void;
  setFilters: (filters: FilterOptions) => void;
  clearResults: () => void;
}
```

Estado Local

- **useState** para estado simples
- **useReducer** para estado complexo
- **useCallback** para otimização

- **useMemo** para valores computados

Performance

Otimizações Implementadas

1. **Code Splitting** - Carregamento sob demanda
2. **Lazy Loading** - Componentes e imagens
3. **Memoização** - React.memo e useMemo
4. **Debounce** - Redução de chamadas API
5. **Virtual Scrolling** - Para listas grandes
6. **Service Worker** - Cache offline

Métricas Esperadas

- **First Contentful Paint**: < 1.5s
- **Largest Contentful Paint**: < 2.5s
- **Time to Interactive**: < 3.5s
- **Cumulative Layout Shift**: < 0.1

Acessibilidade

Recursos Implementados

- **Navegação por teclado** completa
- **Screen reader** compatível
- **Contraste** WCAG AA
- **Focus management** adequado
- **ARIA labels** descritivos
- **Semantic HTML** estruturado

Testes

Estratégia de Testes

```
// Testes unitários com Jest
test('should perform search when enter is pressed', async () => {
  render(<SearchBar onSearch={mockSearch} />);
  const input = screen.getByPlaceholderText('Buscar dados oficiais...');
  fireEvent.change(input, { target: { value: 'despesas' } });
  fireEvent.keyPress(input, { key: 'Enter' });
  await waitFor(() => {
    expect(mockSearch).toHaveBeenCalledWith('despesas');
  });
});
```

Tipos de Testes

- **Unitários** - Componentes isolados
- **Integração** - Fluxos completos
- **E2E** - Cenários de usuário
- **Acessibilidade** - Conformidade WCAG

Build e Deploy

Configuração Vite

```
export default defineConfig({
  plugins: [react()],
  build: {
    outDir: 'dist',
    sourcemap: true,
    rollupOptions: {
      output: {
        manualChunks: {
          vendor: ['react', 'react-dom'],
          utils: ['axios', 'date-fns']
        }
      }
    }
  }
});
```

Otimizações de Build

- **Tree shaking** automático
- **Minificação** de código
- **Compressão** de assets
- **Chunking** inteligente
- **Source maps** para debug

Funcionalidades Avançadas

1. Busca Inteligente

- **Autocomplete** com sugestões contextuais
- **Correção automática** de termos
- **Histórico** de buscas recentes
- **Filtros inteligentes** baseados no contexto

2. Visualização de Dados

- **Gráficos interativos** com Chart.js
- **Tabelas responsivas** com ordenação
- **Exportação** em múltiplos formatos
- **Compartilhamento** de resultados

3. Experiência do Usuário

- **Loading states** informativos
- **Error boundaries** para recuperação
- **Feedback visual** em todas as ações
- **Animações suaves** e naturais

Vantagens da Arquitetura

1. **Modularidade** - Componentes reutilizáveis e bem organizados
2. **TypeScript** - Tipagem forte reduz bugs e melhora DX
3. **Performance** - Otimizações modernas implementadas
4. **Responsividade** - Design mobile-first adaptativo
5. **Acessibilidade** - Conformidade com padrões web
6. **Testabilidade** - Arquitetura facilita testes automatizados
7. **Manutenibilidade** - Código limpo e bem documentado

Próximos Passos

1. Implementar PWA (Progressive Web App)
2. Adicionar modo escuro/claro
3. Implementar notificações push
4. Adicionar analytics de uso
5. Otimizar para SEO