📊 Tecnologias dos Dashboards - Atende+

🎯 Stack Tecnológico Completo

Frontend Framework

- **React 18.3.1** - Biblioteca principal para construção da interface
- **TypeScript** - Tipagem estática para maior segurança e manutenibilidade
- **Vite 6.3.5** - Build tool moderna e rápida

📈 Biblioteca de Gráficos: **Recharts 2.15.2**

Recharts é a biblioteca principal usada para criar todos os gráficos e visualizações de dados no dashboard.

Características:

- ✅ Baseada em React e D3.js
- ✅ Componentes declarativos e fáceis de usar
- ✅ Responsiva e otimizada para performance
- ✅ Suporte a animações suaves
- ✅ Altamente customizável

Gráficos Implementados:

1. Gráfico de Barras (BarChart)

```
import { BarChart, Bar, XAxis, YAxis, CartesianGrid, Tooltip, Respon

<BarChart data={senhasPorTipo}>
  <CartesianGrid strokeDasharray="3 3" vertical={false} stroke="#e2e
  <XAxis dataKey="name" angle={-45} textAnchor="end" height={60} />
  <YAxis />
  <Tooltip />
  <Bar dataKey="value" fill="#6366f1" radius={[6, 6, 0, 0]} barSize=
</BarChart>
```

Uso: Mostra a demanda por tipo de serviço

2. Gráfico de Pizza (PieChart)

```
import { PieChart, Pie, Cell } from 'recharts';

<PieChart>
  <Pie
    data={senhasPorStatus}
    cx="50%"
    cy="50%"
    innerRadius={70}
    outerRadius={100}
    paddingAngle={4}
    dataKey="value"
  >
    {senhasPorStatus.map((entry, index) => (
      <Cell key={`cell-${index}`} fill={entry.color} strokeWidth={0}
    ))}
  </Pie>
  <Tooltip />
</PieChart>
```

Uso: Visualiza a distribuição de status das senhas (aguardando, atendendo, concluída, etc.)

🎨 UI Components & Styling

Radix UI (Componentes Headless)

Biblioteca de componentes acessíveis e não estilizados:

```
"@radix-ui/react-dialog": "^1.1.6",
"@radix-ui/react-dropdown-menu": "^2.1.6",
"@radix-ui/react-select": "^2.1.6",
"@radix-ui/react-tabs": "^1.1.3",
"@radix-ui/react-tooltip": "^1.1.8",
// ... e mais 20+ componentes
```

Vantagens:

- ✅ Totalmente acessível (ARIA compliant)
- ✅ Sem estilos pré-definidos (total controle)
- ✅ Composição flexível
- ✅ Suporte a teclado e screen readers

TailwindCSS 3.4.17

Framework CSS utility-first para estilização:

```
<div className="bg-white rounded-2xl shadow-soft border border-secon
  {/* Conteúdo */}
</div>
```

Recursos usados:

- ✅ Classes utilitárias responsivas
- ✅ Sistema de cores customizado
- ✅ Animações e transições
- ✅ Grid e Flexbox layouts

🔄 Comunicação em Tempo Real

Socket.IO 4.8.2

Biblioteca para comunicação WebSocket bidirecional:

```
import { io } from 'socket.io-client';

const socket = io();

socket.on('stateUpdated', (payload) => {
  // Atualiza dashboard em tempo real
  setSenhas(payload.senhas);
  setSenhaAtual(payload.senhaAtual);
});
```

Recursos:

- ✅ Atualizações em tempo real
- ✅ Reconexão automática
- ✅ Broadcasting para múltiplos clientes
- ✅ Eventos customizados

🎯 Gerenciamento de Estado

React Context API

```
import { createContext, useContext, useState, useEffect } from 'reac

const SenhasContext = createContext<SenhasContextType | undefined>(u

export const useSenhas = () => {
  const context = useContext(SenhasContext);
  return context;
};
```

Dados gerenciados:

- Senhas (fila completa)
- Usuários online
- Senha atual sendo chamada
- Últimas senhas atendidas
- Serviços disponíveis

🎨 Ícones

Lucide React 0.487.0

Biblioteca de ícones moderna e leve:

```
import { Clock, Users, TrendingUp, Activity, BarChart2, PieChart } f

<Clock className="w-5 h-5 text-orange-600" />
```

Características:

- ✅ 1000+ ícones
- ✅ Totalmente customizáveis
- ✅ Tree-shakeable (apenas ícones usados são incluídos)
- ✅ SVG otimizados

📊 Componentes do Dashboard

1. KPI Cards (Métricas Principais)

```
// 4 cards principais
- Senhas em Espera (Clock icon)
- Atendimentos Concluídos Hoje (TrendingUp icon)
- Tempo Médio de Atendimento (Activity icon)
- Guichês Ativos (Users icon)
```

2. Gráficos de Visualização

```
// 2 gráficos lado a lado
- Gráfico de Barras: Demanda por Serviço
- Gráfico de Pizza: Status Geral das Senhas
```

3. Listas em Tempo Real

```
// 2 listas lado a lado
- Fila de Espera (com tempo de espera dinâmico)
- Últimos Atendimentos (últimas 10 senhas concluídas)
```

⚡ Features Especiais

1. Tempo de Espera Dinâmico

```
const TempoEspera = ({ inicio }: { inicio: Date }) => {
  const [espera, setEspera] = React.useState('');
  const [isLate, setIsLate] = React.useState(false);

  React.useEffect(() => {
    const update = () => {
      const diff = Math.floor((new Date().getTime() - new Date(inici
      setEspera(diff + ' min');
      setIsLate(diff >= 15); // Alerta após 15 minutos
    };
    update();
    const interval = setInterval(update, 60000); // Atualiza a cada
    return () => clearInterval(interval);
  }, [inicio]);

  return (
    <div className={isLate ? 'bg-red-100 text-red-700' : 'bg-seconda
      <Clock className="w-3 h-3" />
      {espera}
    </div>
  );
};
```

Funcionalidade:

- ✅ Atualização automática a cada minuto
- ✅ Alerta visual quando tempo > 15 minutos
- ✅ Cálculo em tempo real

2. Indicador "Ao Vivo"

```
<div className="flex items-center gap-2 text-primary-600 bg-primary-
  <Activity className="w-5 h-5" />
  <span className="text-sm font-bold uppercase tracking-wide">Ao Viv
</div>
```

3. Animações Suaves

```
className="animate-in fade-in duration-500"
className="hover:shadow-lg transition-all"
className="hover:-translate-y-0.5"
```

🗂️ Estrutura de Arquivos

```
src/
├── components/
│   ├── admin/
│   │   └── LiveDashboard.tsx     ← Dashboard principal
│   ├── Administrador.tsx          ← Painel admin
│   └── Atendente.tsx              ← Painel atendente
├── context/
│   └── SenhasContext.tsx          ← Estado global
└── styles/
    └── index.css                  ← Estilos globais
```

📦 Dependências Principais

```
{
  "react": "^18.3.1",
  "recharts": "^2.15.2",              // ← Gráficos
  "socket.io-client": "^4.8.2",       // ← Tempo real
  "lucide-react": "^0.487.0",         // ← Ícones
  "tailwindcss": "^3.4.17",           // ← Styling
  "@radix-ui/*": "^1.x.x"             // ← UI Components
}
```

🎨 Paleta de Cores do Dashboard

```
const colors = {
  orange: '#f59e0b',    // Senhas aguardando
  green: '#10b981',     // Concluídas
  blue: '#3b82f6',      // Chamadas
  cyan: '#06b6d4',      // Atendendo
  red: '#ef4444',       // Canceladas
  purple: '#a855f7',    // Guichês ativos
  indigo: '#6366f1'     // Gráficos
};
```

🚀 Performance

Otimizações Implementadas:

- ✅ **ResponsiveContainer**: Gráficos responsivos sem re-renders desnecessários
- ✅ **React.memo**: Componentes memorizados
- ✅ **useEffect com cleanup**: Previne memory leaks
- ✅ **Lazy loading**: Componentes carregados sob demanda
- ✅ **Socket.IO**: Apenas dados alterados são transmitidos

📱 Responsividade

```
// Grid responsivo
className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6"

// Breakpoints TailwindCSS:
// sm: 640px
// md: 768px
// lg: 1024px
// xl: 1280px
```

🔗 Referências

- **Recharts**: https://recharts.org/
- **Radix UI**: https://www.radix-ui.com/
- **TailwindCSS**: https://tailwindcss.com/
- **Socket.IO**: https://socket.io/
- **Lucide Icons**: https://lucide.dev/

💡 Exemplo de Uso

Para visualizar o dashboard:

1. Acesse: http://localhost:3001
2. Faça login como **Administrador**
3. O dashboard será exibido automaticamente com dados em tempo real

Credenciais: `admin` / `admin`
