

# Gerenciador de Conexões RDP/VNC/SSH Enterprise

Versão: 3.1.1

Data da Última Atualização: 16 de Setembro de 2025

## 1. Visão Geral do Projeto

O Gerenciador de Conexões Enterprise é uma aplicação desktop multiplataforma, construída com **Electron** e **React**, projetada para centralizar, simplificar e monitorizar o gerenciamento de múltiplas conexões de acesso remoto. O objetivo principal é fornecer uma interface de usuário (UI) moderna, intuitiva e unificada para que profissionais de TI possam organizar, aceder e verificar o estado das suas conexões de forma eficiente e segura.

### Principais Tecnologias

- **Electron:** Para a criação da aplicação desktop multiplataforma.
- **React:** Para a construção de uma interface de utilizador reativa e baseada em componentes.
- **Electron Builder:** Para empacotamento e criação de instaladores.
- **Electron Store:** Para persistência de dados e configurações do utilizador.

## 2. Arquitetura da Aplicação

A aplicação segue uma arquitetura robusta dividida em três camadas principais, garantindo segurança e separação de responsabilidades.

### 2.1. Backend (Processo Principal - Electron)

- **Arquivo Principal:** public/electron.js
- **Responsabilidades:**
  - Criação e gerenciamento da janela principal da aplicação (BrowserWindow).
  - Comunicação com o sistema operacional (criação de menus, diálogos de erro/arquivo, etc.).
  - Execução de processos externos (child\_process) para iniciar as conexões RDP (mstsc.exe), SSH (putty.exe) e VNC (tvnviewer.exe).
  - Persistência de dados através da biblioteca electron-store, que salva as configurações num arquivo JSON local.
  - Gerenciamento de toda a lógica de testes de conectividade (ping, porta, etc.) através do ConnectivityTester.

### 2.2. Frontend (Processo de Renderização - React)

- **Ponto de Entrada:** src/index.js

- **Componente Raiz:** src/App.js
- **Responsabilidades:**
  - Renderização de toda a interface do utilizador.
  - Gerenciamento centralizado do estado da aplicação (listas de grupos, conexões, estado da UI) no componente App.js.
  - Coleta de interações do utilizador (cliques, preenchimento de formulários).
  - Envio de solicitações para o Backend (via preload.js) para realizar ações que o navegador não pode.

### 2.3. Ponte (Preload Script)

- **Arquivo Principal:** public/preload.js
- **Responsabilidades:**
  - Atua como uma ponte segura entre o Frontend e o Backend.
  - Utiliza o contextBridge do Electron para expor seletivamente APIs do Backend (como iniciar uma conexão ou salvar um arquivo) para o Frontend de forma segura, através do objeto global window.api.

## 3. Principais Funcionalidades

- **Gerenciamento Multi-Protocolo:** Suporte para organizar e lançar conexões RDP, SSH e VNC (via cliente externo TightVNC/RealVNC).
- **Dashboard de Monitoramento:** Uma tela centralizada que exibe o estado (Online, Offline, etc.) e a latência de todos os servidores RDP/SSH, com a funcionalidade de "Testar Todos" para uma verificação em massa.
- **Sistema de Conectividade:** Testes individuais ou em lote para verificar a disponibilidade de servidores antes de conectar, incluindo verificação de porta e ping (ICMP).
- **Interface Reativa e Moderna:** UI construída com componentes React, incluindo modais para criação de grupos/servidores e um tema escuro consistente com barra de rolagem customizada.
- **Persistência de Dados:** Todas as configurações de grupos e servidores são salvas localmente e carregadas a cada inicialização.
- **Importação e Exportação:** Funcionalidade para criar backups (exportar) e restaurar (importar) todas as configurações de conexão através de um arquivo JSON.
- **Empacotamento para Instalação:** O projeto está configurado com electron-builder para gerar um instalador .exe para fácil distribuição em ambientes Windows.

## 4. Como Começar (Ambiente de Desenvolvimento)

Siga os passos abaixo para configurar e executar o projeto num ambiente de desenvolvimento.

1. **Clonar o Repositório**

```
git clone <URL_DO_SEU_REPOSITORIO>
cd <NOME_DA_PASTA>
```

## 2. Instalar as Dependências

npm install

## 3. Executar a Aplicação

O projeto usa concurrently para iniciar o servidor de desenvolvimento do React e o Electron ao mesmo tempo.

npm run electron:start

Isto abrirá a aplicação em modo de desenvolvimento com acesso às ferramentas de desenvolvedor.

## 5. Scripts Disponíveis

- npm start: Inicia apenas o servidor de desenvolvimento do React.
- npm run electron:start: Inicia a aplicação completa (React + Electron) para desenvolvimento.
- npm run build:react: Compila e otimiza o código React para produção.
- npm run build: Cria a versão de produção do React e, em seguida, usa o electron-builder para gerar o instalador na pasta dist/.