Documentação: Sistema de Acompanhamento de Tarefas

O sistema de acompanhamento de tarefas foi desenvolvido para permitir que funcionários, supervisores e gerentes interajam com um sistema centralizado de gerenciamento de tarefas, cada um com suas permissões e funcionalidades específicas.

1. Equipe de Desenvolvimento

Nome Completo	**Matrícula**
Carlos Cesar Passos	*12724115774*
Felipe Costa	*12724120050*
Jeferson Santos	*12724124361*
Lucas Santana	*12724125417*
Edgard	*12724118869*
Guilherme Alves	*12724129198*

2. Requerimentos de Software

Para a correta compilação e execução da aplicação, os seguintes softwares e tecnologias são necessários:

- **Ambiente de Execução:**
  - **Node.js:** Versão 18.x ou superior.
  - **Gerenciador de Pacotes:** npm (versão 9.x ou superior) ou yarn.
- **Linguagens de Programação:**
  - **TypeScript:** Utilizado em todo o frontend para garantir um código mais robusto, legível e com tipagem estática.
  - **JavaScript:** Linguagem base para a execução da aplicação no navegador e no ambiente Node.js.
- **Tecnologias de Frontend:**
  - **React (v18.2.0):** Biblioteca principal para a construção da interface de usuário reativa e baseada em componentes.
  - **Vite:** Ferramenta de build moderna e servidor de desenvolvimento local, que oferece uma experiência de desenvolvimento rápida.
  - **Tailwind CSS:** Framework CSS *utility-first* para estilização rápida e consistente da interface.

    - **react-router-dom:** Biblioteca para gerenciamento de rotas e navegação entre as diferentes páginas da aplicação.
- **Tecnologias de Backend e Banco de Dados:**
    - **Supabase:** Plataforma de código aberto utilizada como Backend as a Service (BaaS). Ela oferece:
        - **Banco de Dados:** Uma instância de **PostgreSQL**, um banco de dados relacional, para o armazenamento persistente de todos os dados da aplicação (usuários, tarefas, etc.).
        - **Autenticação:** Sistema integrado para gerenciar o cadastro, login e as sessões dos usuários.
        - **APIs:** Geração automática de APIs para interagir com o banco de dados de forma segura.
- **Navegador Web:**
    - Qualquer navegador moderno com suporte completo a JavaScript, como Google Chrome, Mozilla Firefox, Safari ou Microsoft Edge.

3. Instruções para Instalação e Execução

- Passo 1: Instalar o Pré-requisito (Node.js)
    - Acesse: https://nodejs.org/
    - Baixe a versão LTS.
    - Instale o Node.js com as opções padrão.
- Passo 2: Abrir o Terminal
    - Windows:
        1. Pressione Windows + R.
        2. Digite `cmd` e pressione Enter.
    - macOS:
        1. Vá em Aplicativos > Utilitários > Terminal.
        2. Ou use Command + Barra de espaço, pesquise por "Terminal" e pressione Enter.
    - Linux:
        1. Geralmente Ctrl + Alt + T.
        2. Ou procure por "Terminal" no menu de aplicativos.
- Passo 3: Navegar até a Pasta do Projeto
    - No terminal, digite `cd ` (com um espaço).
    - Arraste a pasta do projeto para o terminal.
    - Pressione Enter.
    - Exemplo: `cd C:\\Users\\SeuUsuario\\Downloads\\projeto-loja-de-roupas-17`
- Passo 4: Executar os Comandos

    - Instalar as dependências: Digite `npm install` e pressione Enter.
    - Aguarde a conclusão.
    - Iniciar o site: Digite `npm run dev` e pressione Enter.
- Passo 5: Acessar o Site
    - Após `npm run dev`, copie o endereço local (ex: `http://localhost:8080/`).
    - Abra o navegador.
    - Cole o endereço e pressione Enter.
    - O terminal precisa ficar aberto. Use Ctrl + C para desligar o site

Configuração Inicial do Banco de Dados

O projeto utiliza o Supabase para o banco de dados. Ja existe usuarios ja criados:

Email: gerente@loja.com
Senha: 123456

Email: supervisor@loja.com
Senha: 123456

Email: funcionario1@loja.com
Senha: 123456

Email: funcionario2@loja.com
Senha: 123456

Email: funcionario3@loja.com
Senha: 123456

4. Justificativa para a Abordagem de Comunicação Escolhida

A comunicação entre o cliente (frontend em React) e o servidor (backend Supabase) foi implementada utilizando a biblioteca **@supabase/supabase-js**, que funciona como um cliente para a **API** gerada pelo Supabase. A escolha desta abordagem em detrimento de implementações de mais baixo nível, como Sockets puros ou RPC, foi estratégica e baseada nos seguintes pilares:

- **Produtividade e Abstração:** A principal vantagem é a alta produtividade. A biblioteca cliente do Supabase abstrai a complexidade da comunicação de rede. Em vez de construir requisições HTTP manualmente ou gerenciar conexões de Sockets, o desenvolvimento se concentra na lógica de negócio, utilizando

métodos diretos e intuitivos (.from('tabela').select(), .insert(), etc.) que são traduzidos em chamadas de API seguras e otimizadas.

- **Segurança Integrada (Row Level Security):** O Supabase utiliza a segurança a nível de linha (RLS) do PostgreSQL como um pilar central. Isso permite que as regras de acesso (ex: "um funcionário só pode ver suas próprias tarefas") sejam definidas diretamente no banco de dados. A API gerada respeita essas políticas automaticamente, garantindo que a comunicação seja segura por padrão e evitando a exposição acidental de dados. Implementar essa camada de segurança manualmente seria complexo e propenso a erros.
- **Escalabilidade e Manutenção:** Utilizar uma API RESTful padronizada, gerenciada pelo Supabase, garante que a arquitetura é escalável e de fácil manutenção. Se, no futuro, for necessário criar outros clientes (como um aplicativo móvel), eles poderão consumir a mesma API, reutilizando toda a lógica de negócio e segurança já implementada no backend.
- **Funcionalidades em Tempo Real:** Embora a comunicação principal seja via API (requisição-resposta), o Supabase também oferece suporte a subscrições em tempo real (*realtime subscriptions*) sobre a mesma API. Isso permite que o cliente "escute" mudanças no banco de dados (novas tarefas, atualizações de status) e atualize a interface do usuário instantaneamente, sem a necessidade de implementar uma complexa infraestrutura de WebSockets do zero.

Em suma, a abordagem de comunicação via **API do Supabase** foi escolhida por oferecer um equilíbrio ideal entre simplicidade de desenvolvimento, segurança robusta, escalabilidade e funcionalidades avançadas, alinhando-se perfeitamente aos requisitos do projeto e às práticas modernas de desenvolvimento de software.