Resumo da Transferência do Projeto: Sistema de Controle Agrícola

Visão Geral

O projeto consiste em um protótipo funcional de um sistema web para o gerenciamento completo do trabalho agrícola. A plataforma permite o cadastro e controle de fazendas, serviços, trabalhadores e usuários. Sua funcionalidade central é o registro diário de trabalho, que serve como base para a geração automatizada de relatórios e folhas de pagamento, com opções de exportação para PDF e Excel.

Estado Atual do Projeto

- **Tecnologias Utilizadas:**
    - **Front-end:** HTML, CSS e JavaScript puro (Vanilla JS).
    - **Back-end:** API RESTful desenvolvida em Node.js com o framework Express.
    - **Banco de Dados:** MongoDB (NoSQL) hospedado na nuvem (MongoDB Atlas), com interação via Mongoose.
    - **Hospedagem:** Deploy contínuo (CI/CD) configurado na Vercel, a partir de um repositório no GitHub.
- **Funcionalidades Implementadas:**
    - Sistema de autenticação seguro com tokens (JWT) e senhas criptografadas (bcryptjs).
    - Controle de acesso com dois níveis de permissão: **Admin** e **Operador**.
    - Gerenciamento completo (CRUD) de Fazendas, Serviços, Trabalhadores e Usuários.
    - Módulo para lançamento de produção diária e registro de faltas.
    - Geração de relatórios com filtros por período e exportação para PDF.
    - Módulo avançado de folha de pagamento com cálculos personalizáveis (INSS, Salário Família) e exportação para PDF e Excel.
- **Status do Código:** O sistema está **totalmente funcional** e operacional. O código-fonte, tanto do back-end (`/api`) quanto do front-end (`/public`), está atualizado e reflete todas as funcionalidades descritas.

Plano de Evolução Estratégico (Roadmap)

O plano de desenvolvimento futuro foi dividido em três fases claras:

- **Fase 1 (Curto Prazo): Fortalecer o Back-end**
    - Implementar validação de dados na API com **Zod** para aumentar a segurança e a confiabilidade.
    - Criar um sistema centralizado para tratamento de erros.
    - Adicionar testes automatizados para a API (Jest/Supertest).
- **Fase 2 (Médio Prazo): Modernizar o Front-end**

        - Migrar a interface de JavaScript puro para um framework moderno, como **Next.js (React)**, para melhorar a escalabilidade e a experiência do desenvolvedor.
- **Fase 3 (Longo Prazo): Otimizar a Arquitetura de Dados**
    - Reavaliar o schema do MongoDB e, se necessário, considerar a migração para um banco de dados relacional como o **PostgreSQL** para otimizar consultas complexas.

Próxima Ação Imediata

A tarefa prioritária e imediata é **iniciar a Fase 1**, começando pela **implementação da validação de dados com Zod na API**. Esta ação visa tornar o sistema mais robusto e seguro antes de qualquer outra evolução.