estamos enfrentando um problema que precisa ser sanado:

quando estamos preenchendo um novo formulario observamos que se não anexarmos nenhum arquivo ele funciona e salva normalmente, mas se por acaso incluirmos um arquivo ele não salva.

Estou trabalhando em um sistema de solicitação de ajustes para aplicativos contratados e gostaria de implementar [DESCREVA A MELHORIA/NOVA FUNCIONALIDADE AQUI].

Com base nos arquivos fornecidos (frontend, backend e controllers), por favor:

1. Analise detalhadamente todos os componentes do sistema que precisarão ser modificados

Atente-se para que a alteração não comprometa nenhum outro processo já em perfeito funcionamento, compare tudo para que não haja inconsistência de dados.

Mesmo que demore um pouco para responder, faça uma análise minuciosa nos arquivos do frontend e backend em todas ligações de variáveis, nomes de campos, nomes de tabelas entre outros e observe nesta alteração que será feita quais precisarão ser mexida.

2. Descreva as alterações necessárias em cada arquivo, incluindo:

- Novos campos/componentes no frontend (HTML/CSS/JS)

- Modificações nos controllers existentes ou necessidade de novos controllers

- Alterações no modelo de dados (se aplicável)

- Impacto na API existente

-Apontamento de necessidade de bibliotecas ou arquivos a confirmar se já instalados no pacote.

3. Forneça exemplos de código para as principais mudanças, não apenas mostrando mas informando detalhadamente onde incluir a string bem como o nome do arquivo a ser alterado.

4. Discuta quaisquer considerações de desempenho ou segurança relevantes

5. Sugira uma ordem de implementação recomendada

6. Identifique possíveis desafios e como mitigá-los

O sistema atual consiste em:

- Páginas HTML com formulários interconectados

- JavaScript frontend para manipulação de formulários e chamadas API

- Backend Node.js com controllers para formulários, fornecedores e projetos

- Frontend hospedado no Vercel

- Banco de dados MySQL hospedado no Railway

Por favor, forneça uma análise completa considerando esta arquitetura existente.  
informe a necessidade de outros arquivos para analise se necessário.

git add .

git commit -m "230420251136"

git push

Contexto para o Projeto de Solicitação de Ajustes

Visão Geral

Este projeto é um sistema de solicitação de ajustes e melhorias para aplicativos contratados, composto por:

* Frontend: HTML, CSS e JavaScript (arquivos index.html e script.js)
* Backend: Node.js com controllers (formularioController.js, fornecedoresController.js, projetosController.js)
* Banco de dados: MySQL
* Infraestrutura: Hospedado na Vercel (frontend) e Railway (backend/banco de dados)

Funcionalidades Atuais

1. **Formulário de Ajuste**: Permite registrar solicitações de ajustes em aplicativos
2. **Cadastro de Fornecedores**: Gerencia empresas responsáveis pelos aplicativos
3. **Cadastro de Projetos**: Gerencia os projetos/aplicativos contratados

Arquitetura

* Frontend simples com navegação entre formulários
* API RESTful no backend
* Upload de mídia (imagens/vídeos) para documentar problemas
* Comunicação entre frontend e backend via fetch API

Tecnologias Utilizadas

* **Frontend**: HTML5, CSS3, JavaScript puro (sem frameworks)
* **Backend**: Node.js com módulos nativos
* **Banco de Dados**: MySQL
* **Hospedagem**: Vercel (frontend), Railway (backend + MySQL)
* **Gerenciamento de Uploads**: Multer

**Arquivos Essenciais para Anexar**

**1. Frontend (HTML/CSS/JS):**

* index.html → Estrutura principal do formulário e navegação.
* script.js → Lógica de interação, chamadas à API e manipulação de dados.
* style.css (se existir) → Estilos específicos (como não está anexado, seu CSS está inline no HTML).

**2. Backend (Node.js):**

* formularioController.js → Lógica do formulário de ajustes (CRUD + uploads).
* fornecedoresController.js → Lógica de cadastro de fornecedores.
* projetosController.js → Lógica de cadastro de projetos.
* **Modelos (se existirem)** → Arquivos como formularioModel.js, fornecedoresModel.js, etc. (não anexados, mas importantes se houver).

**3. Rotas (API):**

* Arquivos como routes.js ou api.js (não anexados, mas críticos se existirem).

**4. Configurações:**

* package.json → Lista de dependências e scripts do projeto.
* Configurações do banco de dados (ex: database.js).

**5. Estrutura do Banco de Dados (opcional, mas útil):**

* Scripts SQL ou diagramas das tabelas (ex: schema.sql).