FEEDBACK ESTRUTURADO

- **Avaliadores:** Felipe, Fernando
- **Data:** 30/06/2025
- **Grupo:** Grupo 3
- **Área:** Computação

---

1. Visão Geral

- **Objetivo do feedback:** Ajudar os trainees a desenvolverem melhor os pontos fracos identificados. Um breve review sobre o Projeto Trainee

---

2. Critérios de Avaliação

Métrica	Peso (%)	Aval. 1	Aval. 2	Média	Contribuição Ponderada
Planejamento e Gestão de Tarefas	10	8	8	8.0	0.80
Funcionalidade	10	8	9	8.5	0.85
Qualidade do Código	20	9	6	7.5	1.50
Prototipação e UI/UX	15	8	7	7.5	1.13
Documentação	15	6	7	6.5	0.98
Testes e QA	5	0	0	0.0	0.00
Colaboração e Trabalho em Equipe	10	8	8	8.0	0.80
Cumprimento de Prazos	5	8	9	8.5	0.43
Apresentação	10	7	6	6.5	0.65
Total / Nota Geral	100				7.14

Nota Geral: 7,14 / 10 (71,4%)

3. Pontos Fortes

- **Funcionalidade**: O CRUD de tarefas está bem implementado, e os filtros funcionam como esperado. O sistema de autenticação com JWT é um diferencial positivo.

- **Colaboração e Trabalho em Equipe:** A equipe tem um bom fluxo de commits e contribuições regulares. Todos os membros do grupo estão ativamente envolvidos no desenvolvimento do projeto.

---

4. Oportunidades de Melhoria

- **Testes e QA:** A falta de testes automatizados pode prejudicar a evolução do projeto. A implementação de testes unitários e de integração ajudaria a garantir a estabilidade da aplicação.

---

5. Detalhamento dos Critérios e Observações

5.1 Planejamento e Gestão de Tarefas

Observação: O gerenciamento de tarefas é feito de maneira informal, sem o uso de ferramentas como Issues, Projects ou milestones do GitHub. Isso dificulta o acompanhamento mais estruturado do progresso e das pendências do projeto.

Impacto: Dificuldade em rastrear o progresso individual das tarefas, identificar gargalos e estimar o tempo necessário para concluir o projeto.

Recomendações:

- Utilizar o GitHub Issues para gerenciar as tarefas e bugs.

- Adotar uma board Kanban para visualização das etapas do desenvolvimento.

5.2 Funcionalidade

Pontos fortes:

- O CRUD de tarefas está bem implementado e funcional.

- O sistema de autenticação com JWT, registro e login de usuários está bem estruturado.

Melhorias:

- Implementar validações mais robustas no front-end para evitar que dados inválidos (e-mails mal formatados, campos vazios) sejam enviados para o back-end. Exemplo: Usar bibliotecas como Yup ou Formik para simplificar a validação.

- As mensagens de erro atuais são genéricas e pouco informativas. Torná-las mais claras e direcionadas para o usuário. Exemplo: "O e-mail fornecido não é válido" em vez de "Erro ao cadastrar".

5.3 Qualidade do Código

Back-end:

- A modularização está boa, porém subiu o arquivo .env no repositório (não pode em produção)

Front-end:

- O uso de TypeScript e React é adequado, mas arquivos TSX em linha única tornam o código difícil de ler. Faltou modularizar melhor as telas de /src em pastas (Atomic Design).
- Arquivos .css, .tsx estão misturados na mesma pasta! Isso dificulta a manutenção e desenvolvimento.

Recomendações:

- Refatorar o código para deixar mais legível
- Modularizar melhor o projeto em pastas distintas /frontend e /backend (isso facilita a manutenção e permite a independencia da aplicação)

5.4 Prototipação e UI/UX

Pontos fortes:

- A navegação entre telas e funcionalidades (login, registro, tarefas) está clara e intuitiva.

Melhorias:

- A responsividade pode ser melhorada para diferentes tamanhos de tela, especialmente para dispositivos móveis.

- Realizar testes de usabilidade seria útil para garantir que a interface seja de fácil navegação.

5.5 Documentação

Observação:

- Tem que ter um README.md tanto para o frontend quanto para o backend (já que os comandos e dependências são diferentes).
- Faltou detalhar melhor os comandos e dependencias.

Impacto:

- A ausência de documentação adequada pode resultar em problemas de integração e dificultar a contribuição de novos membros para o projeto.

Recomendações:

- Criar um README detalhado com:
    - Descrição do projeto
    - Tecnologias utilizadas
    - Passos para instalação e execução
    - Exemplos de chamadas à API
    - Variáveis de ambiente necessárias

5.6 Testes e QA

Observação:

- O projeto não possui testes automatizados implementados. Isso é uma lacuna importante, especialmente para garantir que o código não quebre com novas alterações.

Recomendações:

- Implementar testes unitários no back-end (por exemplo, usando Jest ou Mocha) para os endpoints principais.
- Adicionar testes de integração e de componente no front-end.
- Integrar os testes ao pipeline de CI, para garantir que o código está sempre testado antes de ser integrado.

5.7 Colaboração e Trabalho em Equipe

Pontos fortes:

- A equipe se mostrou bastante colaborativa, com commits frequentes e trabalho em conjunto.

Melhorias:

- Utilizar Pull Requests para que o código seja revisado antes de ser integrado, garantindo maior qualidade e evitando conflitos.

5.8 Cumprimento de Prazos

Pontos fortes:

- O projeto seguiu um cronograma razoável, com entrega de funcionalidades dentro do prazo.

Recomendações:

- Vincular os commits às issues para que o progresso seja mais bem monitorado e o acompanhamento dos prazos seja mais eficiente.

5.9 Apresentação

Pontos fortes:

- A apresentação do projeto foi clara e bastante explicativa.

Melhorias:

- A apresentação poderia ser mais formal, utilizando slides para uma estrutura mais organizada.

- Incluir uma demonstração ao vivo do sistema (logo no início), abordando as funcionalidades principais durante a apresentação.

Bom Trabalho!