Católica de Joinville

Sistema de Gestão de Tarefas e Projetos

Humberto Peres da Rocha Filho

Curso: Engenharia de Software.

Data de Entrega: a definir

Conteúdo

Resumo

Neste documento é apresentado o desenvolvimento de um sistema de gestão de tarefas e projetos, abordando suas funcionalidades essenciais, arquitetura, requisitos e tecnologias utilizadas. O objetivo é fornecer um fluxo organizado para a criação, acompanhamento e finalização de tarefas, promovendo maior produtividade e rastreabilidade em projetos.

1. Introdução

Contexto: A gestão eficiente de tarefas é um dos pilares de projetos bem-sucedidos. Com o aumento da complexidade e da quantidade de demandas em ambientes empresariais, é essencial contar com ferramentas especializadas para organizar e monitorar atividades.

Justificativa: Equipes de desenvolvimento de software, gestão de projetos e operações precisam de uma solução eficaz para atribuir, acompanhar e concluir tarefas de forma colaborativa. O sistema proposto visa preencher essa necessidade ao proporcionar uma interface intuitiva e integração com fluxos de trabalho modernos.

Objetivos: O objetivo principal deste projeto é criar um sistema web que permita a gestão de tarefas por meio de cartões organizados em quadros. Como objetivos secundários, busca-se:

- Permitir a criação e atribuição de tarefas com status customizáveis.
- Facilitar a colaboração entre membros de uma equipe.
- Implementar dashboards para acompanhamento do progresso.

Processo de Desenvolvimento: O desenvolvimento do sistema seguirá práticas ágeis para garantir entregas incrementais e feedback contínuo. Inicialmente, será criado um protótipo para validação da interface e dos fluxos do sistema. Em seguida, o desenvolvimento será iniciado de forma iterativa, com as funcionalidades sendo divididas em tarefas claras e organizadas. Serão empregados testes unitários para assegurar a qualidade do código, além da utilização de ferramentas de lint para manter o código limpo, legível e consistente. O versionamento será gerenciado com Git, permitindo rastreabilidade e colaboração entre os desenvolvedores. Práticas de integração contínua (CI/CD) também serão aplicadas para automatizar testes e deploys, garantindo que as atualizações sejam entregues de forma confiável e eficiente.

Funcionalidades adicionais: O sistema contará com dashboards configuráveis para visualização do progresso das tarefas, permitindo ao usuário personalizar as métricas exibidas, como

distribuição de tarefas por status, por membro da equipe, por prioridade, entre outras visualizações. Essa flexibilidade auxilia no acompanhamento de projetos de forma clara e intuitiva, adaptando-se às necessidades específicas de cada equipe ou workspace.

2. Descrição do Projeto

Tema do Projeto: Desenvolvimento de um sistema de gestão de tarefas e projetos, focado na eficiência e colaboração entre equipes.
Problemas a Resolver:

- Falta de organização em tarefas e prazos.
- Dificuldade na atribuição e monitoramento de atividades.
- Falta de visibilidade sobre o andamento de projetos.

Limitações do Projeto: Este sistema não contempla, inicialmente, funcionalidades de upload de arquivos ou anexos nas tarefas, nem chat ou comunicação interna entre usuários. O sistema terá limites operacionais de até 5.000 tarefas por workspace, 20 workspaces por usuário e 100 usuários por workspace, podendo ser ajustados conforme a evolução e demanda do projeto.

3. Especificação Técnica

3.1. Requisitos de Processo

Requisitos Funcionais:

- RF01 - O usuário deve poder criar uma conta com suas informações pessoais.
- RF02 - O usuário deve poder realizar login no sistema.
- RF03 - O usuário deve poder editar suas informações pessoais.
- RF04 - O usuário deve poder criar, editar e excluir tarefas e épicos.
- RF05 - O usuário deve poder personalizar os status das tarefas.
- RF06 - O usuário deve poder definir responsáveis e prazos para as tarefas.
- RF07 - O usuário deve poder criar quadros de tarefas.
- RF08 - Caso o quadro seja Scrum, o usuário deve poder criar e gerenciar Sprints.

- RF09 - O usuário deve poder configurar propriedades iniciais do quadro, como Prioridade, Etapa, Equipe e Workspace.
- RF10 - O usuário deve poder acessar os workspaces em que esteja incluído como membro da equipe.
- RF11 - O sistema deve criar todas as novas tarefas na coluna inicial definida pelo usuário (por exemplo, "To Do").
- RF12 - O sistema deve permitir a visualização detalhada de uma tarefa.
- RF13 - O usuário administrador deve poder criar, editar e excluir usuários.
- RF14 - O sistema deve disponibilizar dashboards para visualização do progresso de tarefas por diferentes metricas que podem ser escolhidas pelo usuário.

Requisitos Não-Funcionais:

- RNF01 - O sistema deve ser responsivo.
- RNF02 - O sistema não deve permitir o registro de dois usuários com o mesmo nome de usuário (username).
- RNF03 - O sistema deve implementar autenticação segura, garantindo o armazenamento seguro dos dados dos usuários.
- RNF04 - O sistema deve impedir a criação de workspaces com identificadores duplicados.
- RNF05 - O sistema deve permitir que o usuário gerencie quadros de tarefas nos formatos Kanban e Scrum.
- RNF06 - O sistema deve possibilitar a integração com API de CEP para preenchimento automático de endereço durante o cadastro de usuários.
- RNF07 - O sistema não oferecerá suporte para funcionamento offline, sendo necessário acesso à internet para utilização.

3.2. Casos de Uso (UML):

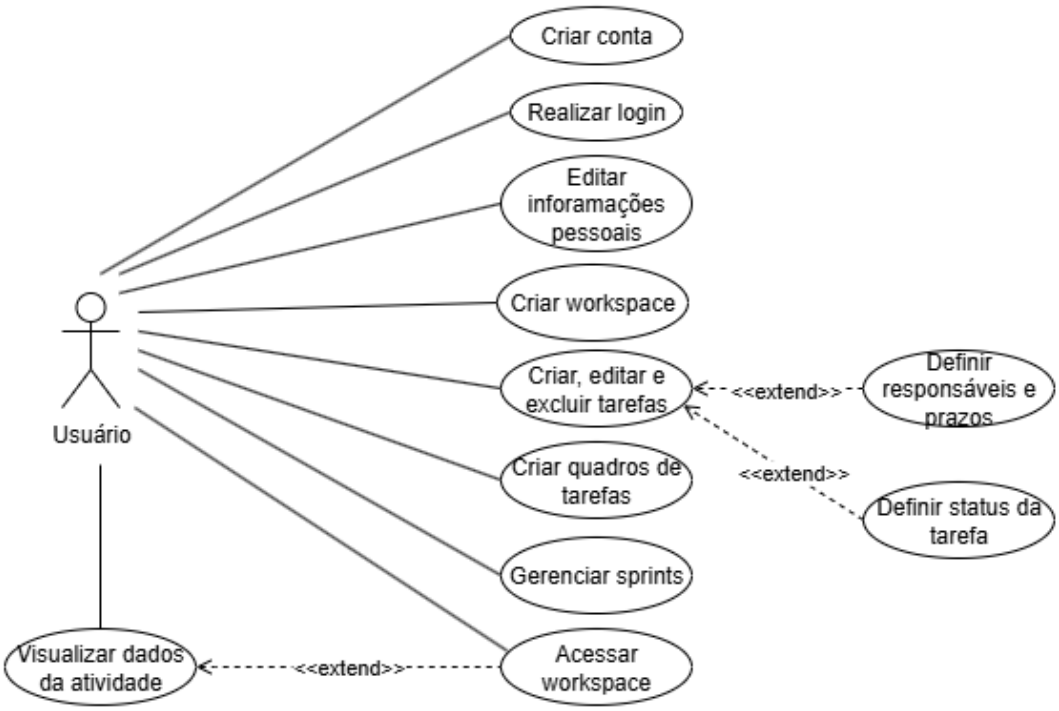


Figura 1: Diagrama de Casos de Uso (UML)

3.3. Considerações de Design

O sistema é projetado com uma arquitetura modular e escalável. Para organizar a estrutura foi adotado o padrão **MVC** (Model-View-Controller), permitindo uma distinção clara entre a interface do usuário, a lógica de negócios e a camada de dados. A aplicação terá design responsivo e poderá utilizar técnicas de compressão de dados e carregamento sob demanda de componentes para otimizar o desempenho em dispositivos móveis.

Visão Inicial da Arquitetura: A aplicação será composta por dois componentes principais:

- **Front-end:** Desenvolvido com ReactJS, responsável pela interface gráfica e experiência do usuário.
- **Back-end:** Construído em NodeJS com Express, servindo como intermediário entre o banco de dados e o front-end por meio de uma API RESTful.

Padrões de Arquitetura: A adoção do padrão **MVC** visa estruturar o código de forma organizada, facilitando a manutenção e a escalabilidade do projeto. Além disso, a autenticação será implementada utilizando JWT, garantindo maior segurança no controle de acesso.

Escalabilidade e Tolerância a Falhas: O sistema será projetado para suportar múltiplos usuários simultâneos, com a expectativa inicial de atender entre 20 e 50 usuários ativos simultâneos em pequenos ambientes de equipe e até 200 usuários cadastrados no total. Será possível implementar tolerância a falhas utilizando práticas de backup periódico e estrutura modular para reinicialização de serviços de forma isolada em caso de falhas. Para otimização futura de consultas e escalabilidade, o sistema poderá integrar mecanismos de caching, como o uso de Redis, para reduzir a latência em consultas frequentes e melhorar a performance geral do sistema.

Modelos C4: A arquitetura do sistema será documentada em quatro camadas:

- **Contexto:** Definição dos principais elementos do sistema, como usuários, interface e API.

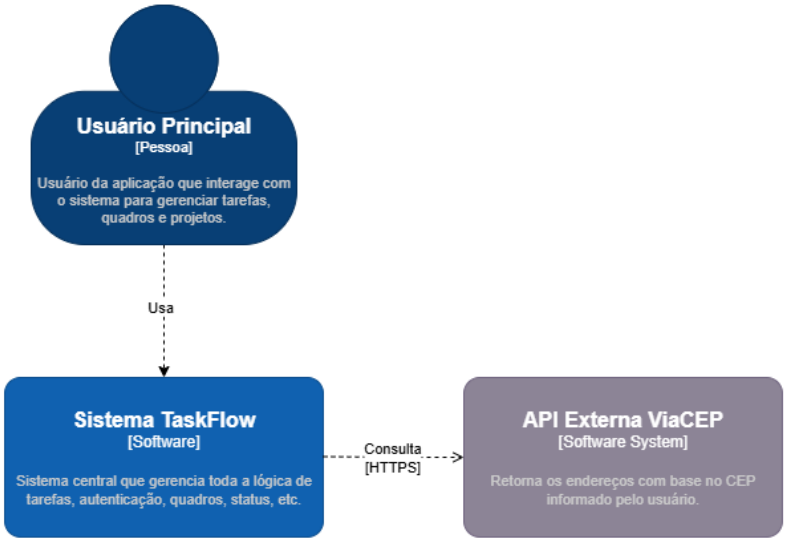


Figura 2: Diagrama de Contexto

- **Contêineres:** Representação dos serviços principais, incluindo a API back-end e o front-end hospedado separadamente.

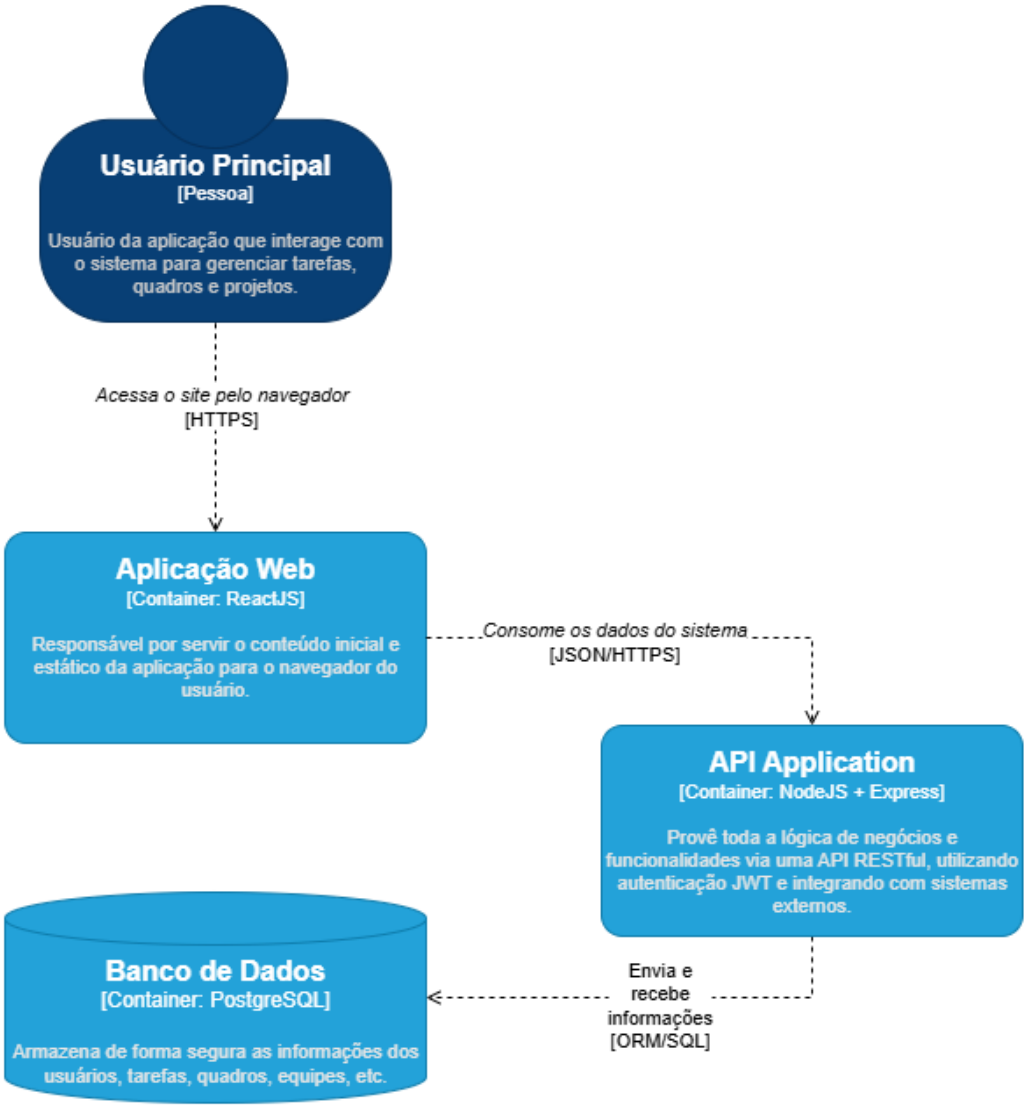


Figura 3: Diagrama de Container

- **Componentes:** Detalhamento dos módulos internos, como gerenciamento de autenticação e operações com banco de dados.

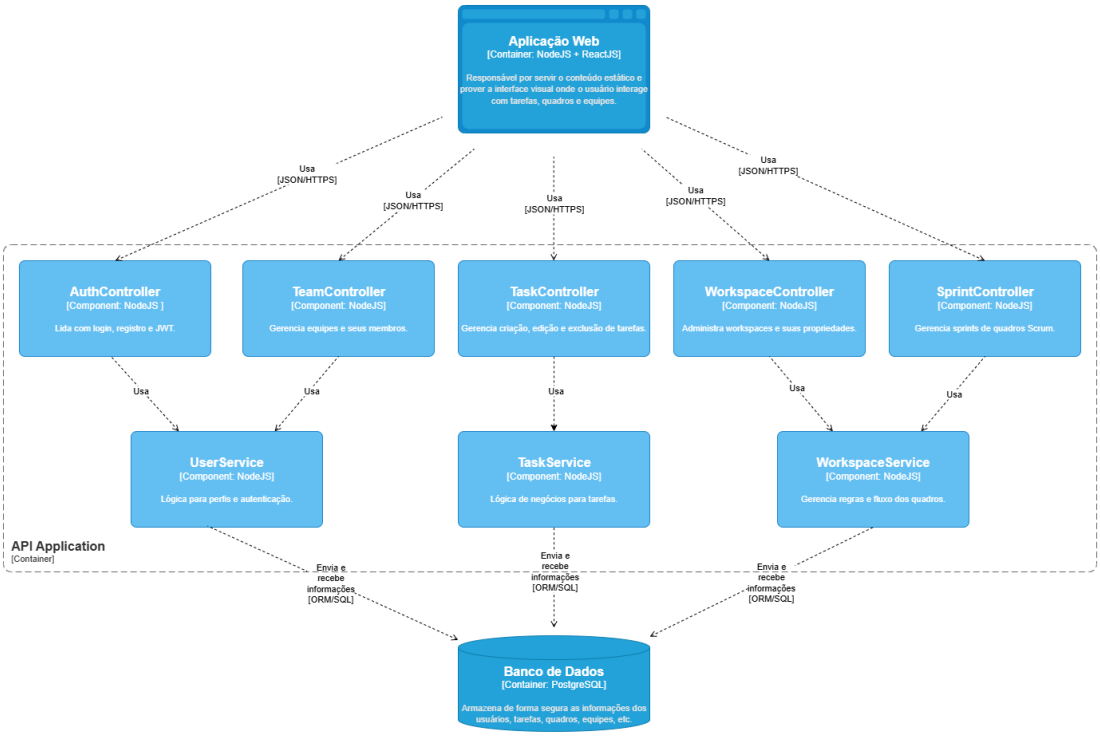


Figura 4: Diagrama de Componente

- **Código:** Estruturação do código-fonte em módulos independentes, promovendo reutilização e organização.

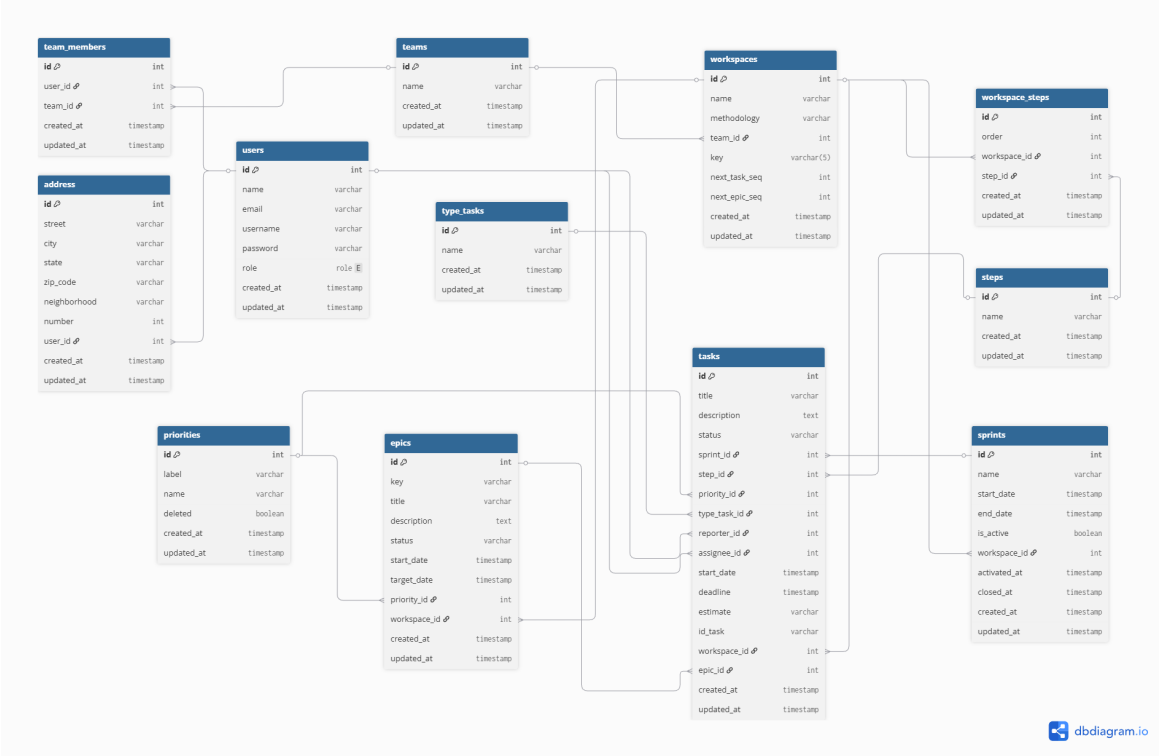


Figura 5: Diagrama de Classe

3.4. Stack Tecnológica

Linguagens de Programação:

- **Front-end:** ReactJS para a construção da interface.
- **Back-end:** NodeJS com Express para processamento das requisições.

Frameworks e Bibliotecas Utilizadas:

- ReactJS para o desenvolvimento do front-end.
- Express para a criação da API REST.
- Prisma ORM para modelagem, migrations e acesso ao PostgreSQL.
- JWT para autenticação e segurança.

Banco de Dados: O PostgreSQL será utilizado como sistema gerenciador de banco de dados devido à sua robustez, suporte a transações, consistência, escalabilidade e facilidade de manutenção. Além disso, sua estrutura relacional facilita a modelagem de entidades como tarefas, usuários, equipes e workspaces de forma clara e normalizada, garantindo integridade dos dados. O PostgreSQL é adequado para o volume estimado de dados do projeto, considerando a gestão de até 200 usuários cadastrados e milhares de tarefas distribuídas em múltiplos workspaces, atendendo com segurança as necessidades do sistema em pequenos e médios ambientes de equipe.

Ferramentas e Tecnologias de Suporte:

- **GitHub:** Utilizado para versionamento de código, controle de mudanças e colaboração entre desenvolvedores.
- **Docker:** Responsável pela criação, gerenciamento e execução de contêineres, garantindo um ambiente de desenvolvimento padronizado.
- **Jest:** Utilizado para testes unitários no back-end.
- **Vitest:** Utilizado para testes unitários no front-end.
- **Vite:** Ferramenta de build e desenvolvimento para o front-end, oferecendo hot reload e otimização de performance.

3.5. Considerações de Segurança

A segurança da aplicação será reforçada por diferentes práticas e medidas, sendo desenvolvida em conformidade com a **Lei Geral de Proteção de Dados (LGPD)**, garantindo o tratamento adequado dos dados pessoais dos usuários, permitindo solicitações de exclusão ou alteração de dados, e garantindo a coleta apenas de informações essenciais. O sistema adotará JWT para autenticação, garantindo que apenas usuários autorizados acessem determinadas funcionalidades. Para prevenir ataques como injeção SQL e cross-site scripting (XSS), serão implementadas validações rigorosas nas entradas de dados. Além disso, informações sensíveis, como senhas, serão armazenadas de forma criptografada para garantir sua integridade e confidencialidade.

Serão implementadas rotinas de **backup periódico** do banco de dados e logs de atividades dos usuários para rastreabilidade. A autenticação via JWT incluirá expiração de tokens e renovação controlada para maior segurança de sessão. Poderá ser implementado monitoramento de eventos suspeitos, como múltiplas tentativas de login incorreto, bloqueando ou alertando administradores em caso de comportamentos anômalos.

4. Próximos Passos

Com a conclusão da fase de desenvolvimento e implantação inicial do sistema, o projeto encontra-se em pleno funcionamento em ambiente de produção na AWS. As próximas etapas visam expandir funcionalidades essenciais para aprimorar a colaboração e produtividade das equipes.

Os principais objetivos incluem a implementação de um sistema de notificações em tempo real via e-mail e push notifications, mantendo os usuários informados sobre atualizações relevantes nas tarefas. Será adicionado suporte para upload e anexo de arquivos nas tarefas para armazenamento seguro de documentos, imagens e outros arquivos relacionados aos projetos.

A integração com GitHub permitirá sincronização bidirecional de issues e commits, vinculando automaticamente o código desenvolvido às tarefas do sistema e facilitando o rastreamento do progresso técnico. Complementarmente, será implementado um histórico completo de alterações para cada tarefa, registrando todas as modificações realizadas, seus autores e timestamps, proporcionando rastreabilidade e transparência nas ações da equipe.

Por fim, melhorias na infraestrutura incluem a implementação de backup automatizado do banco de dados e otimizações de performance para suportar o crescimento da base de usuários.

5. Referências

BROWN, Simon. **C4 Model for visualising software architecture**. 2023. Disponível em: https://c4model.com. Acesso em: 15 abr. 2025.

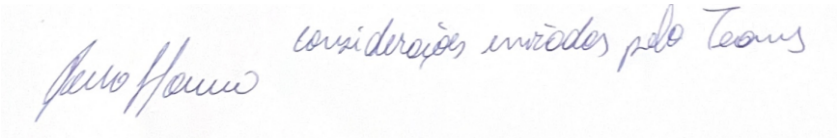
React – A JavaScript library for building user interfaces. 2024. Disponível em: https://reactjs.org. Acesso em: 15 abr. 2025.

Node.js Documentation. 2024. Disponível em: https://nodejs.org. Acesso em: 15 abr. 2025.

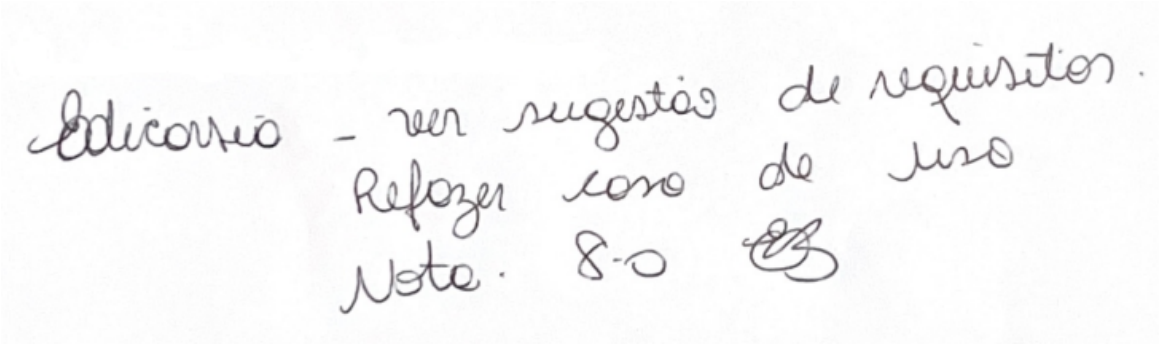
Docker Documentation. 2024. Disponível em: https://docs.docker.com. Acesso em: 15 abr. 2025.

6. Avaliações de Professores

- Assinatura 1.



- Assinatura 2.



- Assinatura 3.