Católica de Joinville

Sistema de Gestão de Tarefas e Projetos

Curso: Engenharia de Software.

Data de Entrega: a definir

Conteúdo

Resumo

Este documento apresenta 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 de forma visual e flexível. 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.
- Integrar com outras ferramentas via APIs.

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.

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.
- 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 nos formatos Kanban ou Scrum.
- 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 usuário deve poder aplicar filtros para buscar tarefas, quadros ou workspaces.
- RF13 - O sistema deve permitir a visualização detalhada de uma tarefa.

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.

3.2. Casos de Uso (UML):

Figura 1: Diagrama de Casos de Uso (UML)

3.3. Considerações de Design

O sistema foi 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.

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

- **Front-end:** Desenvolvido com React, responsável pela interface gráfica e experiência do usuário.

- **Back-end:** Construído em Node.js 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.

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

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

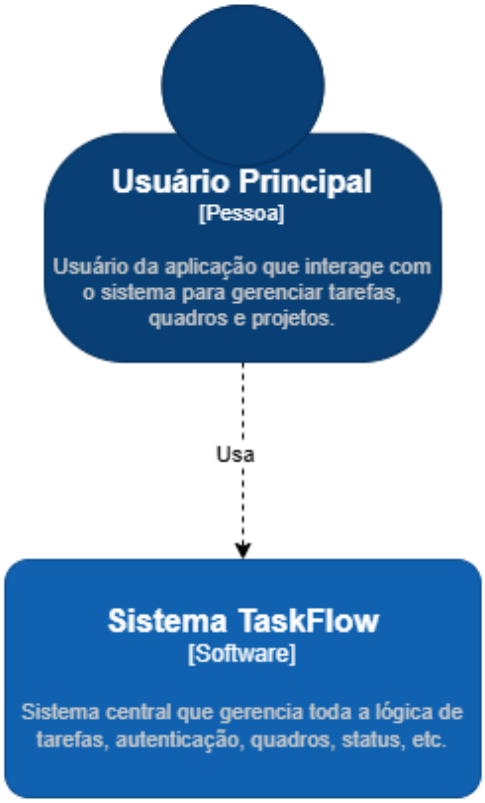

Figura 2: Diagrama de Contexto

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

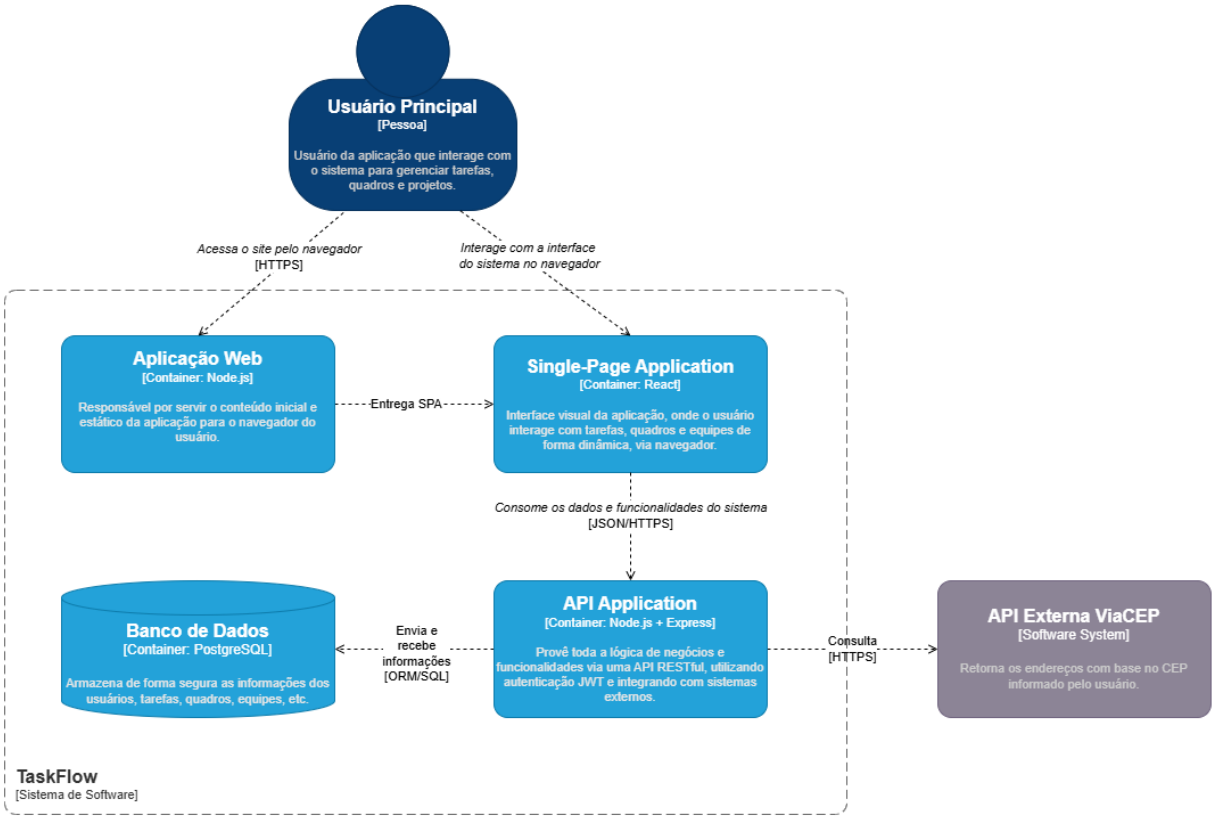

Figura 3: Diagrama de Container

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

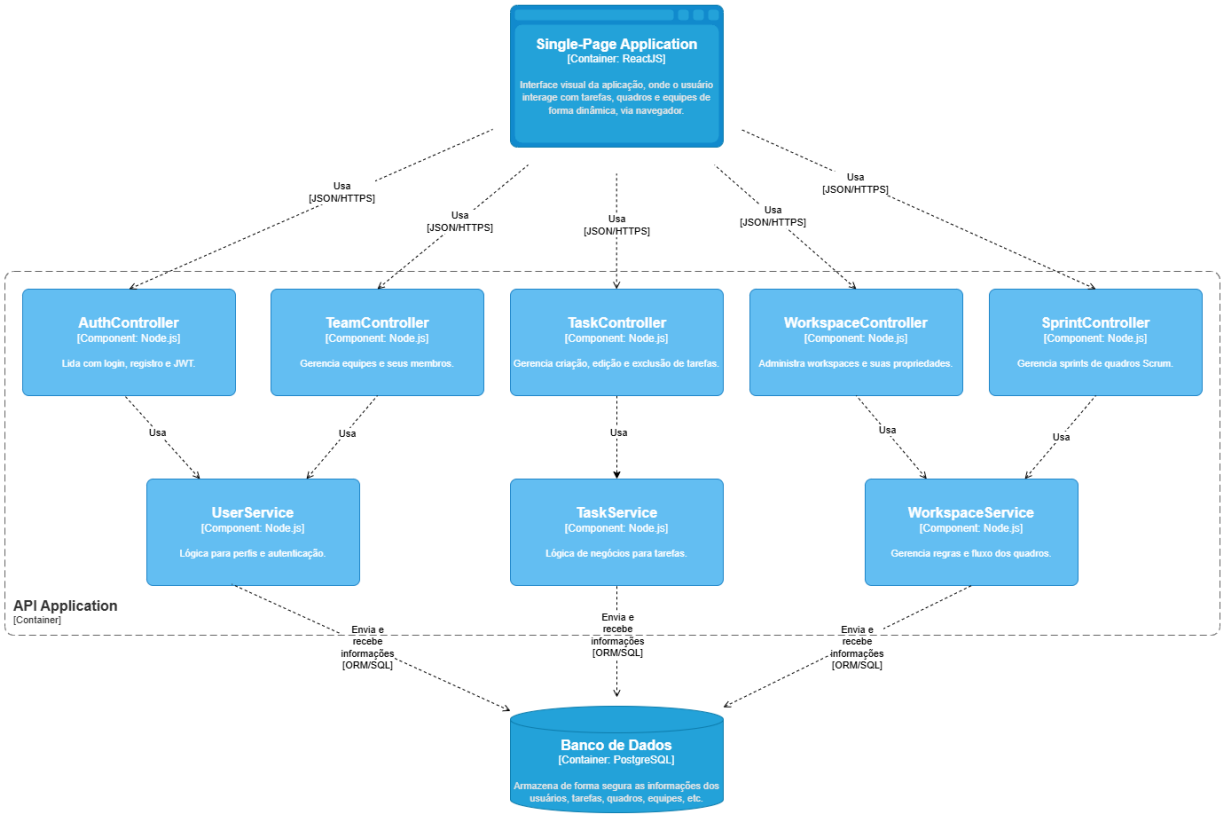

Figura 4: Diagrama de Componente

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

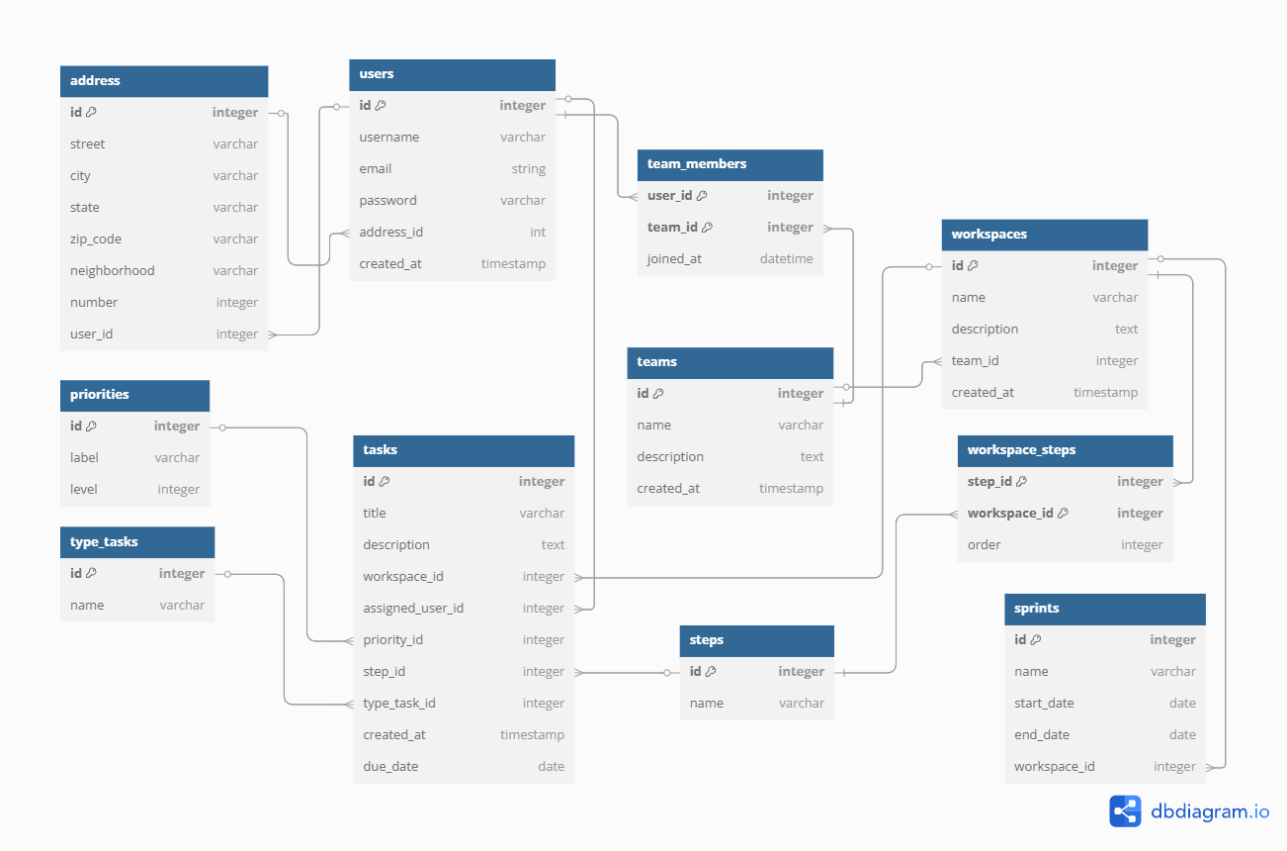

Figura 5: Diagrama de Classe

3.4. Stack Tecnológica

Linguagens de Programação:

- **Front-end:** React para a construção da interface.
- **Back-end:** Node.js com Express para processamento das requisições.

Frameworks e Bibliotecas Utilizadas:

- React para o desenvolvimento do front-end.
- Express para a criação da API REST.
- Sequelize para gerenciamento do banco de dados.
- JWT para autenticação e segurança.

Ferramentas e Tecnologias de Suporte:

- **GitHub:** Utilizado para versionamento de código, controle de mudanças e colaboração entre desenvolvedores.
- **Jira:** Empregado para planejamento, organização e acompanhamento das tarefas do projeto.
- **Docker:** Responsável pela criação, gerenciamento e execução de contêineres, garantindo um ambiente de desenvolvimento padronizado.

3.5. Considerações de Segurança

A segurança da aplicação será reforçada por diferentes práticas e medidas. 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.

4. Próximos Passos

Com a finalização do documento, o projeto avançará para a fase de desenvolvimento conforme o cronograma estabelecido. Os próximos passos incluem a criação dos protótipos da interface, implementação dos módulos principais e testes rigorosos para validar a funcionalidade e segurança do sistema antes do lançamento.

5. Referências

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

DOCKER Documentation. Acesso em: 15 abr. 2025. 2024. Disponível em: <https://docs.docker.com>.

NODE.JS Documentation. Acesso em: 15 abr. 2025. 2024. Disponível em: <https://nodejs.org>.

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

6. Avaliações de Professores