

Especificação de Requisitos de Software - Task Manager

Versão 1.1

Preparado por Thiago Pereira, Julia Dantas, Rafael Nalin,
Aryann Gabriel, Cláudio Antônio

Universidade Federal de Mato Grosso - Campus Araguaia

Novembro de 2025

Baseado em: IEEE Std 830:2023 — *Recommended Practice for Software Requirements Specifications.*

Histórico de Revisões

Versão	Data	Autores	Descrição
1.0	15/11/2025	Thiago Pereira	Rascunho inicial
1.1	23/11/2025	Thiago Pereira	Revisão dos Requisitos
1.2	04/12/2025	Thiago Pereira, Julia Dantas, Rafael Nalin, Aryann Gabriel, Cláudio Antônio	Inclusão do Modelo de Dados, Diagramas de Caso de Uso e Protótipos de Tela

Sumário

Histórico de Revisões	1
Sumário	2
1. Introdução	3
1.1 Propósito	3
1.2 Escopo do Sistema	3
1.3 Definições, Acrônimos e Abreviações	4
1.4 Referências	4
1.5 Visão Geral do Documento	4
2. Descrição Geral	6
2.1 Perspectiva do Produto	6
2.2 Funções do Produto	6
2.3 Usuários e Características dos Usuários	7
2.4 Restrições	7
2.5 Requisitos de Qualidade (atributos do produto)	7
3. Requisitos Específicos	10
3.1 Interfaces Externas	10
3.1.1 Interface com o Usuário (IU)	10
3.1.2 Interfaces de Software e Interfaces de Comunicação	11
3.2 Regras de Negócio	11
3.3 Requisitos Funcionais (RF)	12
3.4 Requisitos Não Funcionais (RNF)	13
3.5 Requisitos de Dados e Banco de Dados	15
Modelos de dados	15
Dicionário de dados	16
Política de backup	17
4. Outros Requisitos	19
4.1 Requisitos Futuros	19
5. Apêndices	20
5.1 Diagramas de Caso de Uso	20
5.2 Modelos de Dados	22
5.3 Protótipos de Interface Gráfica de Usuário	24

1. Introdução

1.1 Propósito

O propósito deste documento de Especificação de Requisitos de Software é detalhar, de forma precisa e estruturada, os requisitos do sistema Task Manager, que será utilizado como estudo de caso na disciplina Tópicos Avançados em Engenharia de Software. O desenvolvimento do sistema adotará práticas e ferramentas alinhadas ao paradigma DevOps, promovendo integração contínua, entrega contínua e colaboração entre as etapas do ciclo de vida do software. A equipe de desenvolvimento é composta pelos discentes matriculados na disciplina, que, ao longo do processo de desenvolvimento, assumem diferentes papéis (*roles*), de modo a vivenciar a dinâmica e as responsabilidades inerentes às diversas funções da engenharia de software.

1.2 Escopo do Sistema

O sistema Task Manager tem como objetivo principal fornecer uma **plataforma web para gerenciamento de tarefas**, permitindo que equipes organizem, acompanhem e controlem suas atividades de forma simples, segura e eficiente. O sistema abrangerá funcionalidades essenciais de cadastro de usuários, autenticação, criação e edição de tarefas, atribuição de responsabilidades e visualização estruturada do trabalho em andamento.

Os objetivos centrais do sistema incluem facilitar o gerenciamento das atividades individuais e em equipe, permitir que gerentes acompanhem o progresso das tarefas atribuídas, oferecer uma interface simples para manipulação e consulta de tarefas e garantir a integridade e segurança das ações por meio de autenticação JWT(*JSON Web Token*). O sistema também visa servir como ambiente de experimentação prática das técnicas estudadas durante a disciplina, especialmente no tocante às práticas DevOps.

O escopo apresenta algumas limitações importantes. O sistema não incluirá funcionalidades avançadas de gestão de projetos, como gráficos de Gantt¹, cronogramas detalhados ou métricas de produtividade. A interface terá caráter funcional, priorizando o aprendizado técnico em detrimento de um design sofisticado. Além disso, o sistema será voltado exclusivamente ao contexto acadêmico da disciplina, não havendo necessidade de escalabilidade para um volume elevado de usuários. Integrações externas, como envio de e-mails ou comunicação com sistemas corporativos, também não fazem parte da versão inicial.

¹ Clark, W. (1923). "The Gantt Chart — A Study in Production Planning." Journal of Business of the University of Chicago.

Dentro do contexto de negócio, o Task Manager será desenvolvido e utilizado em ambiente acadêmico, apoiando o aprendizado prático dos estudantes sobre especificação, implementação, testes e entrega contínua. O sistema permitirá simular um cenário realista de desenvolvimento colaborativo, oferecendo aos participantes a oportunidade de vivenciar diferentes papéis dentro de um time DevOps. Além disso, o projeto proporcionará a produção de documentação técnica alinhada a padrões amplamente reconhecidos, como o IEEE 830, contribuindo para a formação profissional e para a compreensão das demandas de um processo de desenvolvimento estruturado.

1.3 Definições, Acrônimos e Abreviações

Task (Tarefa): Atividade registrada no sistema, contendo título, descrição, responsável, status e demais informações necessárias ao acompanhamento.

Usuário: Pessoa cadastrada no sistema, com permissões definidas de acordo com seu perfil.

Gerente: Usuário com permissões ampliadas, capaz de criar tarefas, atribuí-las a outros usuários e acompanhar seu progresso.

JWT (JSON Web Token): Mecanismo de autenticação utilizado para garantir acesso seguro às rotas protegidas.

API (Application Programming Interface): Conjunto de endpoints e regras que permite a comunicação entre cliente e servidor.

CRUD (Create, Read, Update, Delete): Conjunto básico de operações de manipulação de dados oferecidas pelo sistema.

DevOps: Conjunto de práticas que integra desenvolvimento e operações, promovendo automação, colaboração e entrega contínua.

Sprint: Ciclo de desenvolvimento utilizado quando práticas de metodologias ágeis são aplicadas.

1.4 Referências

IEEE Computer Society. **IEEE Std 830:2018** – IEEE Recommended Practice for Software Requirements Specifications.

ISO/IEC/IEEE. **ISO/IEC/IEEE 12207:2017** – Systems and Software Engineering — Software Life Cycle Processes.

ISO/IEC/IEEE. **ISO/IEC/IEEE 29148:2018** - Systems and software engineering — Life cycle processes — Requirements engineering

1.5 Visão Geral do Documento

Seguindo as diretrizes da normativa IEEE 830:2018, o documento está estruturado de maneira lógica, permitindo que diferentes perfis de leitores (*stakeholders*) encontrem

rapidamente as informações relevantes ao seu papel no projeto. A Seção 1 ([Introdução](#)) reúne informações introdutórias essenciais, incluindo o propósito do documento, seu escopo, definições, acrônimos, abreviações e referências. A Seção 2 ([Descrição Geral](#)) apresenta a visão geral do produto, conforme recomendado pela norma, descrevendo as funcionalidades de alto nível, o ambiente operacional, as características dos usuários, as restrições impostas ao sistema e suposições consideradas no projeto. A Seção 3 ([Requisitos Específicos](#)) contém a descrição detalhada dos requisitos funcionais, requisitos não funcionais, requisitos de interface, regras de negócio e demais especificações técnicas necessárias para orientar o desenvolvimento. Por fim, os apêndices incluídos ao final do documento apresentam informações complementares, modelos, exemplos e artefatos que auxiliem na interpretação dos requisitos.

2. Descrição Geral

2.1 Perspectiva do Produto

O sistema Task Manager é concebido como um produto independente, desenvolvido especificamente como estudo de caso da disciplina *Tópicos Avançados em Engenharia de Software*. Ele não é derivado de um sistema anterior, tampouco integra uma suíte de softwares maior. Seu propósito é funcionar como uma aplicação autônoma, com arquitetura simples e modular, adequada ao contexto acadêmico e às práticas DevOps adotadas pela equipe.

Embora seja independente, o sistema apresenta algumas **dependências externas** necessárias ao seu funcionamento. No nível de software, ele depende de um servidor de backend capaz de fornecer **serviços RESTful**, bem como o banco de dados relacional **PostgreSQL** para armazenar informações de usuários, tarefas e permissões. Também utiliza bibliotecas e frameworks para autenticação baseada em **JWT**. Para a documentação da API é utilizado **OpenAPI** e, no frontend, para a construção da interface web (**especificar quais frameworks**).

O sistema depende de ferramentas externas para suportar práticas DevOps, como serviços de controle de versão fornecido pelo **Git**, pipelines de integração contínua e entrega contínua (CI/CD), e ambientes de contêinerização **Docker**. Para testes o sistema dependerá do **Pytest**. Essas dependências, embora não façam parte do produto em si, são fundamentais para o desenvolvimento, implantação e manutenção adequados do sistema.

2.2 Funções do Produto

O sistema **Task Manager** oferece um conjunto de funções essenciais para apoiar o gerenciamento de tarefas em equipes de trabalho tais como:

- **Gerenciamento de Usuários:** Permitir o cadastro de usuários (colaborador e gerente), diferenciando perfis de acesso que representam funções/ações diferentes no sistema.
- **Gerenciamento de Tarefas:** Criar, editar, visualizar e organizar tarefas de forma simples e estruturada.
- **Atribuição de Tarefas:** Permitir que gerentes atribuam tarefas a usuários e acompanhem seu progresso.
- **Interface Web:** Oferecer uma interface web funcional para interação com as funcionalidades do sistema.
- **API REST:** Disponibilizar endpoints para comunicação entre frontend e backend, seguindo padrões de interoperabilidade e documentação (Swagger/OpenAPI).

- **Integridade e Rastreabilidade:** Registrar ações do sistema, garantindo consistência, segurança e rastreabilidade dos dados de usuários e tarefas.

2.3 Usuários e Características dos Usuários

Papel	Descrição	Necessidades Principais
Gestor	Cria projetos e atribui tarefas aos colaboradores.	Acompanhar andamento, monitorar prazos e organizar fluxos de trabalho.
Colaborador	Recebe tarefas atribuídas e atualiza o progresso.	Registrar status, informar andamento e manter tarefas atualizadas.
Administrador	Gerencia contas de gestores e colaboradores, além das configurações do sistema.	Garantir o funcionamento, a disponibilidade e a segurança da aplicação.
Desenvolvedor	Responsável pelo desenvolvimento, manutenção e evolução do sistema.	Desenvolver funcionalidades, corrigir problemas e integrar serviços.

2.4 Restrições

O sistema Task Manager será desenvolvido como uma aplicação web independente, utilizando **Python** com **Flask** no backend, tecnologias web padrão no frontend, **Docker** para containerização, **Git** para controle de versão e **Pytest** para testes automatizados. O produto dependerá de um banco de dados **PostgreSQL** para armazenamento de usuários e tarefas. O sistema deve prover **logs (biblioteca logging)** e mecanismos de **backup**. O projeto seguirá as recomendações da IEEE 830 e boas práticas de desenvolvimento seguro, com restrições de prazo alinhadas ao calendário da disciplina, com entrega programada para fevereiro de 2026.

2.5 Requisitos de Qualidade (atributos do produto)

Atributo	Descrição	Prioridade	Métricas / Critério Quantitativo
Capacidade / Escalabilidade	O sistema deve suportar múltiplos	Baixa	Até 20 usuários simultâneos; até 1.000

	usuários simultâneos e crescimento do volume de tarefas.		tarefas cadastradas sem degradação perceptível.
Usabilidade	Interface intuitiva, permitindo que novos usuários realizem tarefas sem treinamento extensivo.	Baixa	$\geq 90\%$ de sucesso em tarefas em testes de usabilidade; ≤ 3 cliques para ações principais.
Desempenho / Tempo de Resposta	Sistema rápido e responsivo, garantindo boa experiência do usuário.	Alta	Tempo médio de resposta $\leq 2s$ para consultas; $\leq 2s$ para criação/edição de tarefas.
Segurança	Controle de acesso por perfil, autenticação JWT e proteção de dados sensíveis.	Alta	Nenhuma violação crítica em testes de segurança; criptografia de dados sensíveis.
Manutenibilidade	Código modular, documentado e testável, facilitando correções e melhorias futuras.	Média	Cobertura mínima de testes unitários de 70%; documentação atualizada e padronizada.
Portabilidade	Execução em múltiplos ambientes com mínima configuração.	Média	Compatível com Linux, Windows e contêiner Docker padrão.
Auditabilidade / Rastreabilidade	Todas as ações de usuários e mudanças críticas devem ser registradas para auditoria.	Média	Logs completos para $\geq 95\%$ das operações críticas ; registros armazenados por no mínimo 6 meses.
Compatibilidade	O sistema deve funcionar corretamente em diferentes navegadores e dispositivos.	Média	Testado em Chrome, Firefox e Edge; interface responsiva mínima.
Recuperabilidade	O sistema deve ser capaz de se recuperar rapidamente de falhas	Alta	Backup automático a cada 24h; recuperação completa de dados ≤ 10 minutos.

	ou quedas inesperadas.		
--	------------------------	--	--

3. Requisitos Específicos

Esta seção da ERS descreve cada requisito funcional do sistema, mantendo um identificador único para rastreabilidade.

3.1 Interfaces Externas

3.1.1 Interface com o Usuário (IU)

O sistema TaskManager será acessado por meio de uma interface web responsiva, desenvolvida em HTML, CSS (Bootstrap) e JavaScript. A interface deve atender aos princípios de usabilidade, consistência e acessibilidade. Às principais telas do sistema são enumeradas a seguir. Acrescenta-se que a prototipação das telas está presente no Apêndice A – Protótipo das Telas do TaskManager a este documento:

- **Tela de Login**
 - Campos: e-mail, senha.
 - Ação: autenticação via JWT.
 - Feedback em caso de erro e link para saiba mais
- **Dashboard**
 - Exibição de tarefas atribuídas ao usuário atualmente logado no sistema.
 - Indicadores de status (Concluída, Em andamento, Atrasada).
- **Gerenciamento de Tarefas**
 - Listagem paginada e filtrável.
 - Ações: criar, editar, remover, visualizar detalhes.
 - Filtros por status, prioridade, equipe, usuário responsável.
- **Tela de Criação/Edição de Tarefas**
 - Campos: título, descrição, prioridade, status, prazo, responsável.
 - Para gerentes: permitir atribuição a membros da equipe.
- **Tela de Administração (apenas gestores)**
 - Cadastro/edição de usuários.
 - Controle de equipes.
 - Permissões e perfis.
- **Tela de configurações**
 - Logout.
 - Alterar/editar informações do usuário.

3.1.2 Interfaces de Software e Interfaces de Comunicação

O TaskManager se integrará a diversas interfaces de software internas e externas usando os formatos de troca JSON e padrão arquitetural RestFul.

- **API REST do TaskManager**
 - Padrão: HTTP/HTTPS
 - Formatos: JSON
 - Autenticação: JWT (Bearer Token)
 - Endpoints:
 - /auth/login
 - /tasks
 - /users
 - /teams
- **Banco de Dados**
 - PostgreSQL
- **Formatos de Troca**
 - JSON será o formato padrão para todas as entradas e saídas da API.
 - Logs devem seguir JSON estruturado.

3.2 Regras de Negócio

ID	Regra
RN01	Apenas gestores podem criar , editar ou excluir projetos, equipes e tarefas de qualquer membro da equipe.
RN02	Apenas administradores podem criar ou excluir perfis de gestores e colaboradores no sistema.
RN03	Uma tarefa não pode ser assinalada com completa se não tiver 100% no campo indicador de percentual trabalhado.
RN04	Uma tarefa não pode ser iniciada após sua data deadline (data limite).
RN05	Um projeto só pode ser encerrado se todas as tarefas forem finalizadas
RN06	Cada tarefa deve ter um e apenas um responsável ativo no sistema.
RN07	Gestores e colaboradores só podem visualizar projetos e tarefas das equipes em que participam.
RN08	Todo projeto deve estar vinculado a uma equipe válida e ativa.
RN09	A remoção de um usuário só pode ser realizada por administradores. Usuários vinculados como responsáveis por tarefas ativas não podem ser removidos.

RN10	Comentários nas tarefas podem ser feitos por qualquer membro da equipe.
-------------	--

3.3 Requisitos Funcionais (RF)

ID	Descrição do Requisito Funcional	Papel
RF01	O sistema deve permitir que o gestor crie tarefas para membros da equipe, informando título, descrição, responsável, datas (início e fim), status e prioridade.	Gestor
RF02	O sistema deve permitir que o gestor crie e participe de múltiplas equipes, sem limite mínimo ou máximo imposto pelo sistema.	Gestor
RF03	O sistema deve permitir que um colaborador altere exclusivamente o campo <i>status</i> das tarefas em que ele é o responsável, registrando a alteração nos logs de auditoria de modo a preservar o histórico.	Colaborador
RF04	O sistema deve garantir que gestores só possam visualizar e gerenciar tarefas e projetos pertencentes às equipes das quais participam.	Gestor
RF05	O sistema deve permitir que o gestor cadastre, edite e exclua projetos, incluindo nome, descrição, datas, equipe, prioridade e status.	Gestor
RF06	O sistema deve permitir que o administrador crie, edite e exclua usuários e projetos.	Administrador
RF07	O sistema deve permitir login com autenticação JWT, retornando um token válido para acesso às rotas protegidas.	Colaborador
RF08	O sistema deve permitir que gestores visualizem, editem e excluam tarefas de qualquer membro da sua equipe.	Gestor
RF09	O sistema deve permitir que o colaborador visualize todos os projetos dos quais participa e as tarefas associadas.	Colaborador
RF10	O sistema deve permitir que colaboradores adicionem registros de progresso nas tarefas, incluindo comentários e percentual concluído.	Colaborador
RF11	O sistema deve permitir buscar e filtrar tarefas por status, responsável, projeto, prazo ou palavra-chave.	Gestor / Administrador
RF12	O sistema deve exibir indicadores de andamento, como percentual concluído, tarefas atrasadas e produtividade da equipe.	Gestor
RF13	O sistema deve disponibilizar a documentação OpenAPI/Swagger da API para consulta e teste.	Desenvolvedor

RF14	O sistema deve incluir testes automatizados de autenticação, CRUDs e permissões de acesso.	Desenvolvedor
RF15	O sistema deve permitir ao gestor comparar o tempo estimado e o tempo real trabalhado em cada tarefa.	Gestor
RF16	O sistema deve permitir ao gestor visualizar relatórios com o total de tarefas e total de horas associadas a cada colaborador.	Gestor
RF17	O sistema deve permitir ao gestor visualizar relatórios da quantidade de tarefas e horas associadas a cada projeto em que participa.	Gestor
RF18	O sistema deve invalidar tokens JWT expirados ou revogados, recusando acesso a rotas protegidas.	Sistema
RF19	O sistema deve registrar todas as ações de login e logout.	Sistema
RF20	O sistema deve permitir anexar arquivos a tarefas, validando tipo e tamanho permitido.	Colaborador
RF21	O sistema deve permitir adicionar e remover etiquetas (tags) em tarefas para organização semântica. Etiquetas não podem conter caracteres especiais além de hífen ou sublinhado e devem ter no máximo 30 caracteres.	Colaborador/Gestor
RF22	O sistema deve registrar histórico de alterações de tarefas, incluindo quem alterou, o que alterou e quando.	Sistema
RF23	A API deve disponibilizar endpoints REST para equipes, usuários, tarefas e projetos.	Desenvolvedor
RF24	O sistema deve permitir que administradores consultem logs por período, usuário, ação ou entidade.	Administrador
RF25	O sistema deve impedir exclusão de tarefas vinculadas a projetos finalizados, exigindo reabertura do projeto.	Sistema
RF26	O sistema deve permitir que gestores visualizem a carga de trabalho acumulada por colaborador em tempo real.	Gestor
RF27	O sistema deve gerar relatório de produtividade por período (diário, semanal, mensal).	Gestor
RF28	Campos de texto (título, descrição, comentários) devem respeitar limites definidos: <ul style="list-style-type: none"> título \leq 100 caracteres descrição \leq 2000 caracteres comentário \leq 1000 caracteres 	Sistema

3.4 Requisitos Não Funcionais (RNF)

ID	Descrição do RNF	Categoria
----	------------------	-----------

RNF01	O backend deve adotar arquitetura em camadas, separando responsabilidades em Controllers, Services, Repositories e Models, implementando API RESTful com versionamento de endpoints (ex.: <i>/api/v1</i>).	Arquitetura
RNF02	O backend deve utilizar PostgreSQL como banco relacional principal, incluindo suporte a migrations, índices, constraints e transações ACID.	Persistência
RNF03	O backend deve utilizar SQLAlchemy ² como ORM(<i>Object-Relational Mapping</i>) para mapeamento objeto-relacional, garantindo mapeamento objeto-relacional e manutenção estruturada do schema.	Persistência
RNF04	O backend deve utilizar Flask como framework principal, adotando Blueprints para modularização e Middlewares para autenticação e validações.	Framework
RNF05	O frontend deve ser desenvolvido em HTML/CSS/JS, podendo utilizar Jinja2 ³ ou frameworks SPA (<i>Single Page Applications</i>) como React/Vue, desde que a aplicação seja responsiva e compatível com navegadores modernos.	Interface Web
RNF06	A cobertura de testes unitários deve atingir no mínimo 70%.	Qualidade / Testes
RNF07	O sistema deve utilizar Git para versionamento, seguindo o padrão GitHub Flow.	Versionamento
RNF08	Os commits devem seguir o padrão <i>Conventional Commits</i> . ⁴	Versionamento
RNF09	Todos os membros da equipe devem trabalhar em todas as partes do projeto.	Time de Desenvolvimento
RNF10	Toda a aplicação (backend, frontend e banco de dados) deve ser integralmente containerizada com Docker, utilizando Docker Compose para orquestração local.	Deploy / Containerização
RNF11	Deve haver pipeline CI/CD no GitHub Actions contemplando build, testes, linting e deploy automático.	CI/CD
RNF12	A aplicação deve utilizar arquivos <i>.env</i> separados por ambiente (dev, test, prod), garantindo segurança e isolamento de credenciais.	Configuração
RNF13	O sistema deve monitorar todos os processos relacionados a backups e restaurações, registrando sucesso ou falhas e gerando alertas para o administrador em caso de erro. As informações salvas devem seguir a política de backup.	Backup/Monitoramento

² <https://www.sqlalchemy.org/>

³ <https://jinja.palletsprojects.com/en/stable/intro/>

⁴ <https://www.conventionalcommits.org/en/v1.0.0/>

RNF14	O sistema deve usar autenticação JWT com renovação de tokens e expiração programada.	Segurança
RNF15	Credenciais de banco e chaves JWT não devem ser armazenadas no código-fonte; elas devem ser carregadas exclusivamente por variáveis de ambiente.	Segurança
RNF16	Todas as comunicações entre cliente e servidor devem usar HTTPS em produção; a conexão com o banco deve ser criptografada quando suportado.	Segurança
RNF17	O projeto deve incluir documentação técnica completa: README, guia de instalação, arquitetura, endpoints da API, modelos de dados e guia para desenvolvedores.	Qualidade / Documentação
RNF18	O código deve ser modular, coeso e seguir princípios SOLID, permitindo extensão e manutenção sem alto acoplamento.	Qualidade / Manutenibilidade
RNF19	A interface web deve ser responsiva e funcionar em dispositivos móveis e desktop.	Usabilidade
RNF20	A aplicação deve registrar logs estruturados em JSON, contendo data, usuário autenticado, entidade manipulada, dados relevantes e status da operação	Monitoramento / Logs
RNF21	A aplicação deve expor métricas de saúde (healthcheck endpoint) indicando estado do banco, disponibilidade da API e versão implantada.	Monitoramento
RNF22	O código Python deve seguir PEP8 e ser analisado por Flake8 ⁵ /Black no pipeline CI.	Qualidade / Manutenibilidade
RNF23	O sistema deve garantir que, em caso de falha crítica, indisponibilidade do banco de dados, corrupção de dados ou perda parcial do ambiente, o ambiente completo (backend, banco de dados e arquivos associados) seja restaurado e colocado em operação dentro de um tempo máximo de 5 minutos.	Backup/Continuidade

3.5 Requisitos de Dados e Banco de Dados

Modelos de dados

Entidade	Atributos principais	Observações
----------	----------------------	-------------

⁵ <https://flake8.pycqa.org/en/latest/>

Usuário	id, nome, e-mail, senha (hash), perfil (Colaborador/Gestor/Admin), equipe_id, data_criacao, data_atualizacao	A senha deve ser armazenada com hash seguro (bcrypt/Argon2).
Equipe	id, nome, gestor_id, data_criacao, data_atualizacao, descricao, colaborador_id	Cada equipe possui apenas um gestor responsável.
Projeto	id, nome, descricao, equipe_id, data_inicio, data_fim, prioridade, status, data_criacao, data_atualizacao	Um projeto está vinculado a uma equipe.
Tarefa	id, titulo, descricao, projeto_id, responsavel_id, status, prioridade, data_inicio, data_fim, percentual_concluido, time_tracking, data_criacao, data_atualizacao	Cada tarefa pertence a um projeto e pode ter comentários e registros de progresso.
Comentario	id, tarefa_id, usuario_id, texto, data_criacao	Relaciona usuário e tarefa, para histórico de comunicação. Somente membros da equipe podem comentar nas tarefas.
Registro de Progresso	id, tarefa_id, usuario_id, percentual, descricao, data_criacao	Permite acompanhar a evolução de cada tarefa.
Log de Auditoria	id, usuario_id, acao, entidade, data_hora	Registro imutável de operações críticas.

Dicionário de dados

Campo	Tipo	Restrição	Descrição
id	integer	PK, auto-increment	Identificador único da entidade
nome	varchar(255)	NOT NULL	Nome do usuário, equipe ou projeto
email	varchar(255)	NOT NULL, UNIQUE	E-mail do usuário
senha	varchar(255)	NOT NULL	Hash da senha
perfil	enum('Colaborador','Gestor','Admin')	NOT NULL	Perfil de acesso do usuário

status	enum('Pendente','Em andamento','Concluída','Atrasada')	NOT NULL	Estado da tarefa
prioridade	enum('Baixa','Média','Alta')	NOT NULL	Prioridade da tarefa/projeto
data_inicio	timestamp	NOT NULL	Data inicial da tarefa ou projeto
data_fim	timestamp	NOT NULL	Data final prevista
data_criacao	timestamp	NOT NULL, default NOW()	Data de criação do registro
data_atualizacao	timestamp	NOT NULL, default NOW(), auto-update	Data da última modificação
descricao	varchar(2000)	NOT NULL	Descrição da Equipe ou Tarefa
titulo	varchar(100)	NOT NULL	Título da Tarefa
texto	varchar(1000)	NOT NULL	Texto do comentário
percentual_concluido	integer(0-100)	NOT NULL	Percentual da Tarefa
timetracking	timestamp	NOT NULL	Timetracking da Tarefa
percentual	integer(0-100)	NOT NULL	Percentual registrado pelo usuário na tarefa
acao	varchar(100)	NOT NULL	Ação realizada para registro no Log
entidade	varchar(100)	NOT NULL	Entidade alvo da ação

Política de backup

A política de backup visa garantir a preservação dos dados armazenados no banco PostgreSQL e a capacidade de recuperação em caso de falhas, corrupção de dados ou eventos inesperados. São considerados críticos dados sobre usuários, tarefas, projetos, equipes, registros de progresso, comentários e logs de auditoria. Desta forma, devem ser incluídos nos backups:

- Todo o banco de dados PostgreSQL;
- Arquivos de configuração essenciais
 - variáveis de ambiente
 - scripts de inicialização

Os dados críticos devem ter seus backups de forma que o sistema possa ser restaurado a um estado consistente em caso de falha.

Os tipos de backup devem ser:

- Backup Completo (Full): diário, a cada 24h.
- Backup Incremental: a cada 6h.
- Backup de Arquivos/Anexos: diário, com compressão ZIP ou TAR.

O processo de backup deve ser automatizado via script ou ferramenta de CI/CD e falhas no backup devem gerar alerta para o administrador.

4. Outros Requisitos

4.1 Requisitos Futuros

ID	Descrição	Categoria / Observação
RF-F01	Integração com Google Calendar para sincronização automática de tarefas e eventos.	Facilita visualização de prazos e compromissos externos.
RF-F02	Implementação de visualização Kanban com suporte a drag and drop de tarefas entre colunas.	Melhora a experiência de gestão visual e organização das tarefas.
RF-F03	Geração automática de relatórios em PDF, consolidando projetos, tarefas e progresso da equipe.	Permite exportação de informações para análise e compartilhamento.
RF-F04	O sistema deve registrar métricas de performance via Prometheus (ex.: tempo médio de resposta, throughput, falhas).	Monitoramento
RF-F05	A aplicação deve suportar cache de dados para melhorar performance em consultas frequentes (Redis ou similares).	Desempenho
RF-F06	Nas tarefas podem ser vinculados anexos.	Sistemas
RF-F07	Realizar a sugestão automática de etiquetas/tags considerando a análise semântica da descrição da tarefa	Sistema

5. Apêndices

5.1 Diagramas de Caso de Uso

- Diagramas de caso de uso (TODO) - Claudio e Aryann

Nome do caso de Uso	Cadastro de Usuário
Atores	Usuário
Pré-condições	Dados do Usuário: Nome (Char 255 caracteres), E-mail (Char 255 caracteres), Idade (Int 3 caracteres), Sexo Biológico (Tipo Específico), Tipo Sanguíneo (Tipo Específico)
Fluxo normal - Descrição	Após o usuário repassar seus dados, o sistema irá cadastrá-lo na base de banco de dados
Fluxos Alternativos	Caso o usuário já tenha cadastro ele será redirecionado para o Login.

Tabela 2 - Diagrama de Caso de uso - Cadastro de Usuário

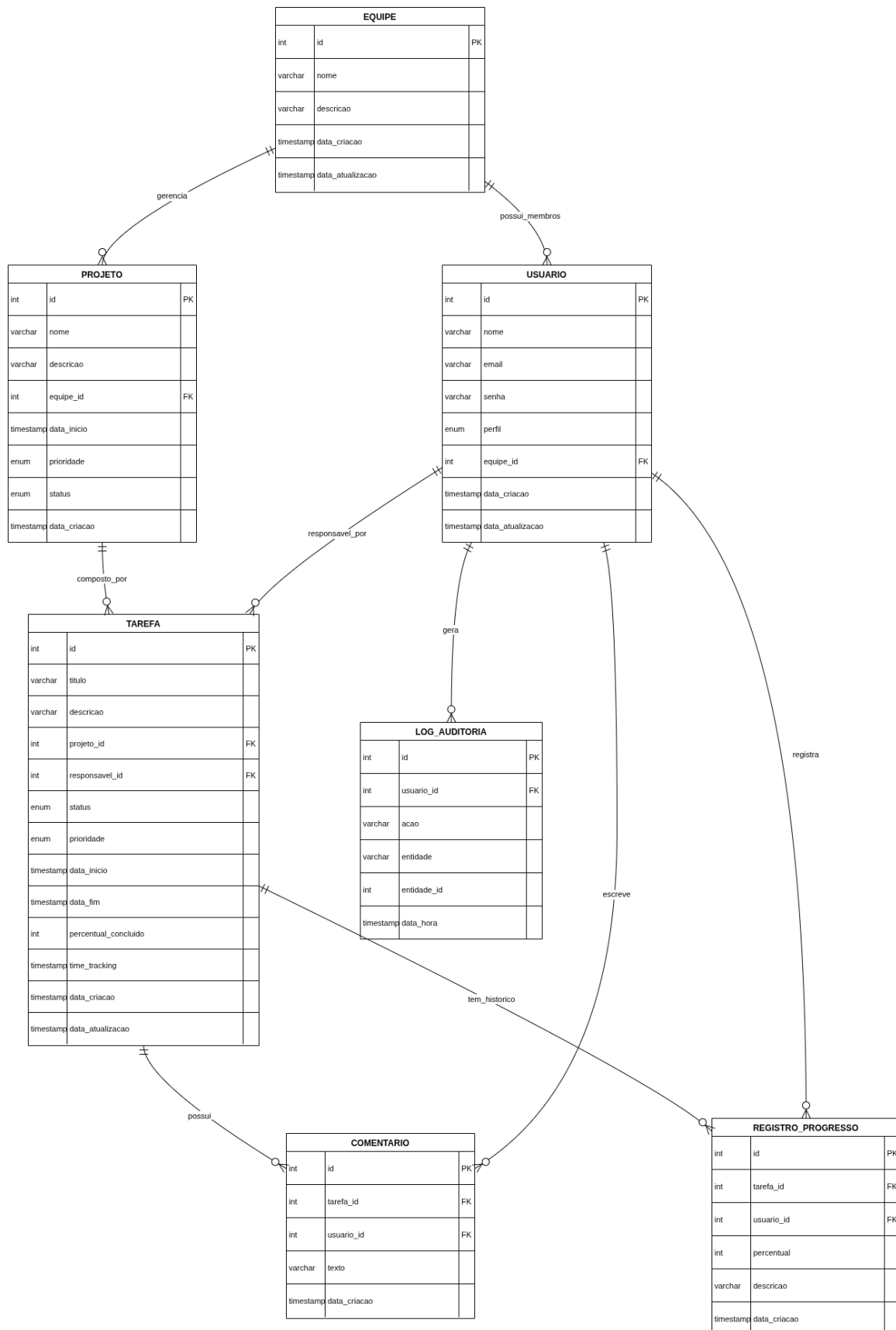
Fonte: Elaborado pela autora

5.2 Modelos de Dados

Adicionar um relacionamento entre tarefa e equipe.

Verificar a entidade *identidade_id*

PostgreSQL



5.3 Protótipos de Interface Gráfica de Usuário

<https://www.figma.com/site/LBKLWGKyffAc36vi3soZP8/Sem-t%C3%ADulo?node-id=0-1&t=svR2DTSH5lYxkLao-1>