Histórico de Versões

Data	Versão	Descrição	Autor	Revisor	Aprovado por
17/12/2024	1.0	Definição de escopo do projeto	Isabel Santos	Isabel Santos	Isabel Santos
09/07/2025	2.0	Atualização das funcionalidades	Isabel Santos	Isabel Santos	Isabel Santos

Responsáveis

1. Equipe

Coordenador

Isabel Lima dos Santos

Desenvolvimento

Isabel Lima dos Santos

2. Cliente

Gestor do Sistema

Usuários interessados em gerenciar as necessidades de acompanhar e atualizar tarefas de qualquer local.

Documento de Visão de Projeto

1. Objetivo

O propósito deste documento é coletar, analisar e definir as necessidades de alto-nível e características do projeto de software, focando nas potencialidades requeridas pelos afetados e usuários-alvo, e como estes requisitos serão abordados no projeto de software.

A visão do projeto documenta o ambiente geral de processos a ser desenvolvido para o sistema durante o projeto, fornecendo a todos os envolvidos uma descrição compreensível deste e de suas macro-funcionalidades.

O Documento de Visão de Projeto documenta apenas as necessidades e funcionalidades do sistema que vão estar sendo atendidas no projeto de software.

2. Descrição do Projeto

O projeto consiste no desenvolvimento de uma API RESTful robusta e segura para um sistema de gerenciamento de tarefas, utilizando Flask como framework principal. As funcionalidades essenciais incluem cadastro de usuários, autenticação segura com JWT e Autenticação de Dois Fatores (2FA), criação, edição, exclusão e compartilhamento de tarefas, e sincronização de dados entre dispositivos. A arquitetura integra um banco de dados relacional (PostgreSQL) para dados de usuários e tarefas, e um banco não relacional (MongoDB) para logs e auditoria.

3. Envolvimento

3.1. Abrangência

O sistema será utilizado de forma externa, permitindo acesso remoto para os usuários via API, garantindo alta disponibilidade e sincronização de dados. Ele atenderá a usuários que necessitam de mobilidade para gerenciar suas tarefas a partir de diferentes dispositivos (desktops, mobile, etc.) através de um cliente frontend que consumirá esta API.

3.2. Papel das Partes Interessadas

3.2.1 - Cliente

Descrição	□ Responsável pelos requisitos funcionais e não funcionais do sistema. □ Responsável pelos testes do sistema para homologação do projeto.
Papel no desenvolvimento	□ Atuar como facilitador e especificador dos requisitos sistêmicos perante a equipe de desenvolvimento. □ Garantir que as regras de negócio sejam suportadas pela base legal. □ Validar e aprovar os requisitos. □ Fornecer os parâmetros operacionais do sistema. □ Acompanhar o desenvolvimento do sistema. □ Decidir sobre a realização de reuniões com os colaboradores. □ Participar da homologação das decisões relacionadas aos sistemas. □ Participar da Homologação do produto final.
Insumos ao projeto de software	□ Requisitos Funcionais. □ Requisitos Não-Funcionais. □ Casos de testes para homologação do sistema.

	Consultas diversas para validação do sistema. Homologação das aplicações.
Representante	Isabel Lima dos Santos

3.2.2 - Equipe de Desenvolvimento TaskSync

Descrição	Responsável pela identificação dos requisitos do software e pelo desenvolvimento dos modelos estáticos e dinâmicos do projeto
Papel no desenvolvimento	Identificar e descrever as necessidades do usuário, especificando as funcionalidades do software que irão atendê-las. Levantar os requisitos funcionais e não funcionais do projeto. Definir o que irá interagir com o sistema. Identificar dentro da visão lógica do sistema, a melhor forma de acomodar as necessidades do usuário, e o impacto da solução adotada sobre os requisitos do sistema. Em suma, responsável pela geração de um produto que atenda aos requisitos que foram identificados junto ao usuário.
Insumos ao projeto de software	Documento de visão e demais documentos de requisitos de software. Aplicação desenvolvida de acordo com o especificado (artefatos do projeto).
Representante	Isabel Lima dos Santos

3.3. Papel dos Atores

Descrição	**Usuário Geral, que cria e gerencia suas tarefas**
Papel	Criar editar e excluir tarefas, acompanhar o status das tarefas e visualizar o histórico de alterações
Insumos ao sistema	Dados de tarefas (título, descrição, prioridade, status).
Representante	A definir.

4. Interfaces do Sistema (Frontend React)

4.1 Descrição da Interface

O projeto inclui uma interface de frontend moderna e reativa, desenvolvida com React, que consome a API RESTful do TaskSync. O objetivo é fornecer uma experiência de usuário fluida e intuitiva para o gerenciamento de tarefas, com um design limpo e responsivo, implementado com o framework Tailwind CSS. A aplicação é uma *Single Page Application* (SPA), onde a navegação entre as diferentes seções ocorre sem a necessidade de recarregar a página, graças ao uso do React Router.

4.2 Funcionalidades da Interface

As funcionalidades da interface foram projetadas para espelhar e dar acesso a todas as capacidades da API de backend:

- **Autenticação de Usuário:**
  - Telas de Cadastro e Login: Formulários dedicados para registro de novos usuários e para autenticação, com validações de campos em tempo real para garantir que os dados (e-mail, nome de usuário, senha) estejam no formato correto antes do envio.
  - Fluxo de Autenticação em Duas Etapas (2FA): Após o login, o usuário é direcionado para uma tela para inserir um código de verificação, proporcionando uma camada adicional de segurança.
  - Recuperação de Senha: Interface completa para que o usuário possa solicitar a redefinição de sua senha e, posteriormente, criar uma nova através de um link seguro.
- **Dashboard e Gerenciamento de Tarefas:**
  - Visualização em Kanban: A página principal após o login é um dashboard no estilo Kanban, que organiza as tarefas em colunas de status ("Pendente", "A fazer", "Em andamento", "Concluído"), permitindo uma visão clara do fluxo de trabalho.
  - Criação e Edição de Tarefas: Um modal dedicado permite que os usuários criem novas tarefas ou editem tarefas existentes, preenchendo campos como título, descrição, prioridade e status.
  - Interatividade: Os cartões de tarefas (`TaskCard`) permitem a exclusão e a mudança de status diretamente do dashboard, oferecendo uma experiência de usuário ágil e interativa.

4.3 Arquitetura da Interface e Componentes Principais

A aplicação segue uma arquitetura baseada em componentes, promovendo a reutilização de código e a manutenibilidade.

- `App.js`: Componente raiz que gerencia as rotas da aplicação com `React Router`.
- `Layout.jsx`: Componente que define a estrutura visual principal da aplicação, como a presença ou ausência da barra de navegação dependendo da rota.
- `PrivateRoute.jsx`: HOC (High-Order Component) que protege rotas, garantindo que apenas usuários autenticados possam acessá-las.
- `api.js`: Módulo central para a comunicação com a API, contendo a configuração do `axios` e interceptors para tratamento de tokens e erros de autenticação.
- `useTasks.js`: Hook personalizado que encapsula a lógica de estado para carregar, adicionar, editar e excluir tarefas, simplificando a gestão de dados nos componentes.

4.4 Restrições Tecnológicas (Frontend)

- Framework Principal: React (versão 18.0.0 ou superior).
- Roteamento: React Router (versão 7.2.0 ou superior) para a navegação da SPA.
- Estilização: Tailwind CSS (versão 3.0.0 ou superior) para um desenvolvimento de UI rápido e consistente.
- Requisições HTTP: Axios é utilizado para todas as comunicações com a API de backend.
- Ícones: A biblioteca `react-icons` é utilizada para a inclusão de ícones na interface.

4.

5. Necessidades e Funcionalidades

Necessidade 1	**Benefício**
Gerenciar Tarefas	**<Médio>**
Id Func.	**Descrição das Funcionalidades/atores envolvidos**
F1.1	Criar Tarefa
	Usuário Geral
F1.2	Editar Tarefa
	Usuário Geral
F1.3	Excluir Tarefa
	Usuário Geral
F1.4	Acompanhar Status da Tarefa
	Usuário Geral
F1.5	Visualizar Histórico de Alterações
	Usuário Geral

Necessidade 2	**Benefício**
Sincronização de Dados	**<Crítico>**
Id Func.	**Descrição das Funcionalidades/atores envolvidos**
F2.1	Sincronização via API REST
	Sistema
F2.2	Otimização de consultas com cache local
	Sistema

Necessidade 3	**Benefício**
Serviço e Autenticação de Usuários	**<Crítico>**
Id Func.	**Descrição das Funcionalidades/atores envolvidos**
F3.1	Cadastro de Usuário com validação de dados
	Usuário Geral
F3.2	Login com E-mail e Senha
	Usuário Geral
F3.3	Autenticação de Dois Fatores (2FA) via E-mail
	Usuário Geral
F3.4	Geração de Tokens de Acesso (JWT)
	Sistema
F3.5	Renovação de sessão com Refresh Tokens
	Sistema
F3.6	Recuperação de Senha
	Usuário Geral
F3.7	Autenticação via Conta Google (OAuth2)
	Usuário Geral

6. Restrições Tecnológicas

- A API será desenvolvida em **Python** com o framework **Flask**.
- O acesso à API deve ser realizado via internet através de **HTTPS**, com segurança garantida por certificados **SSL/TLS**.
- O banco de dados principal para dados de usuários e tarefas será o **PostgreSQL**.
- O **MongoDB** será utilizado para o armazenamento de logs de auditoria.
- A autenticação de sessão será baseada em **JSON Web Tokens (JWT)**.
- A segurança das senhas será garantida com o algoritmo de hashing **bcrypt**.

- A API será protegida contra ataques de força bruta através de **Rate Limiting**.
- Não será implementada autenticação via Certificado Digital nesta versão.

7. Critérios de aceitação do Sistema

O sistema será considerado aceito quando todas as funcionalidades descritas nos requisitos (Seção 4) estiverem implementadas, testadas e validadas em conjunto pela equipe de desenvolvimento e pelo cliente.

8. Escopo Não Incluído no Sistema

- Comunicação em tempo real via WebSockets.
- Monitoramento avançado de falhas e performance.
- Cache distribuído com Redis não será implementado nesta versão (cache local em memória será utilizado).
- Interface administrativa completa.

9. Premissas e Restrições

Premissas	Restrições
• O sistema deverá ser acessível de dispositivos móveis e desktops. • A API usará o padrão REST para a sincronização de dados. • A arquitetura será modular, facilitando futuras manutenções e expansões.	• O sistema não será desenvolvido utilizando contêineres (Docker, Kubernetes) nesta fase. • O sistema não suportará mais de 50 usuários simultâneos nesta versão inicial. • O cache de dados será local (em memória), não utilizando uma solução distribuída como Redis nesta versão.

10. Ligações com Outros Projetos

Não existem interligações com outros projetos ou sistemas previstos para esta versão do sistema.