Especificação de Requisitos de Software: Agenda Online

1. Introdução

1.1 Objetivo

Desenvolver uma aplicação web para gerenciamento de tarefas pessoais, com foco em agenda, notificações e histórico. A aplicação deve ser intuitiva, segura e responsiva.

1.2 Escopo

- Funcionalidades: Cadastro/login, adição/gerenciamento de tarefas, visualização de calendário, alarmes e histórico.
- Exclusões: Integração com e-mails externos (exceto 2FA), suporte a múltiplos usuários colaborativos ou exportação de dados.

1.3 Glossário

- Partições: Seções expansíveis na UI (ex: modais, accordions ou drawers) para exibir conteúdo sem navegação para novas abas.
- Autenticação Dupla (2FA): Verificação adicional via código enviado por e-mail após login.
- Tarefa: Item agendado com nome, tipo (estudo/azul, atividade física/vermelho, trabalho/verde, outro/amarelo), data e horário.

2. Requisitos Funcionais

2.1 Gestão de Usuários

- RF01: Tela de login com e-mail e senha. Validação de credenciais no back-end.

- RF02: Se a conta não existir, opção de cadastro com nome completo, e-mail único e senha (mín. 8 caracteres, com hash bcrypt). Envio de e-mail de confirmação e ativação de 2FA.
- RF03: Após login bem-sucedido, redirecionar para tela principal.

- **2.2 Tela Principal**
  - RF04: Barra horizontal superior com ícones: Calendário (visualização mensal), Alarme (lista de notificações), Histórico (tarefas concluídas) e botão "+" para adicionar tarefas.
  - RF05: Adicionar tarefa via partição (modal expansível): Campos obrigatórios (nome, tipo com cor associada, data, horário). Validações: Data futura, sem conflitos de horário. Armazenar no BD.
  - RF06: Listagem de tarefas pendentes na tela principal, filtradas por data/horário atual. Suporte a edição (atualizar campos) e exclusão (confirmação via modal).
- **2.3 Funcionalidades Adicionais**
  - RF07: Calendário mensal: Visualização em grade (biblioteca como react-calendar). Clique em dia e exiba mini-bloco com tarefas desse dia (nome, tipo, horário).
  - RF08: Alarme: Verificação automática de tarefas próximas (1 dia antes e 1 hora antes do horário). Exibir notificação push no navegador (usando Notification API) e badge no ícone de alarme.
  - RF09: Histórico: Lista paginada de tarefas concluídas (marcar como concluída via checkbox na listagem). Filtros por data/tipo. Tarefas pendentes permanecem na listagem principal.
  - RF10: Logout: Encerra sessão e redireciona para tela de login.
- **2.4 Fluxo Básico do Usuário**
  - Usuário acessa a página inicial (login/cadastro).
  - Se novo, cadastre-se (validação e 2FA).
  - Login → Validação → Acessa o painel principal.
  - No painel: Adicionar/lista/edita/exclui tarefas; visualiza calendário/alarmes/histórico.

- Logout para encerrar a sessão.
- Tratamento de erros: Mensagens amigáveis (ex: "E-mail inválido") e retry options.

3. Requisitos Não Funcionais

3.1 Usabilidade

- Interface simples e intuitiva (UX/UI seguindo princípios de Material Design). Tempo médio para adicionar tarefa: < 30s. Suporte a acessibilidade (WCAG 2.1 AA: alt texts, navegação por teclado).

3.2 Segurança

- Senhas: Armazenadas hashed (bcrypt) no BD. HTTPS obrigatório. Proteção contra brute-force (rate limiting: 5 tentativas/5min). Apenas usuários logados acessam dados (JWT tokens para autenticação). 2FA via e-mail (biblioteca como pyotp no back-end).

3.3 Desempenho e Compatibilidade

- Tempo de resposta: < 2s para operações CRUD. Responsivo para mobile/desktop (media queries no CSS).
- Navegadores: Chrome 90+, Firefox 88+, Edge 90+. Sem suporte a IE.

3.4 Manutenibilidade e Escalabilidade

- Código modular (ex: componentes React reutilizáveis). Documentação inline (JSDoc). Cobertura de testes: > 80% (unitários e integração).
- Escalabilidade: Suporte a 100 usuários simultâneos iniciais (otimizar queries SQL).

3.5 Outros

- Idiomas: Português (BR) como padrão.
- Métricas de Sucesso: Taxa de retenção de usuários > 70% após 1 mês (via analytics opcionais).

4. Arquitetura Técnica

4.1 Front-End

- Framework: React (com hooks e context para estado global).

- Estilos: TailwindCSS v4 + ShadCN (priorizar sobre Material UI para evitar conflitos; usar MUI apenas para componentes específicos como DatePicker).
- Linguagem: TypeScript (tipagem estrita para props e APIs).
- Build: Vite (para desenvolvimento rápido e hot-reload).
- Outros: Axios para chamadas HTTP; react-router para navegação interna (SPA).

4.2 Back-End

- Framework: Django (Python) para API RESTful (CRUD de tarefas e autenticação).
- Autenticação: Django REST Framework com JWT; 2FA via e-mail (usando django-otp). Armazenamento de usuários e autenticação totalmente integrados ao Django ORM.
- Banco de Dados: PostgreSQL como banco único (relacional) para todas as entidades, incluindo usuários e tarefas. Migrações e gerenciamento via Django ORM (makemigrations/migrate).
- Linguagem: Python (Django); JavaScript opcional para scripts utilitários (ex: validações client-side, mas não no core back-end).
- Outros: CORS para integração front-back; Docker para conteinerização e ambiente padronizado de desenvolvimento e deploy.

4.3 Integração

- API Endpoints: `/auth/login`, `/auth/register`, `/tasks/` (GET/POST/PUT/DELETE), `/calendar/`, `/alarms/`, `/history/`.
- Deploy: Front em Vercel/Netlify; Back em Heroku/Railway com Postgres gerenciado (ex: Supabase).

5. Riscos e Prioridades

5.1 Riscos

- Falhas de autenticação ou envio de 2FA por e-mail – Mitigação: teste unitários e ambiente de e-mail configurado com fallback.
- Erros de migração no banco de dados (PostgreSQL) – Mitigação: uso de migrations controladas pelo Django ORM e backups automáticos.

- Problemas de CORS na integração front-and/back-and – Mitigação: configuração adequada dos domínios permitidos no Django e no front-and.
- Dependência de serviços externos (ex: deploy no Heroku/Railway, e-mail SMTP) – Mitigação: definir alternativas de contingência (ex: outro provedor de deploy).

- **5.2 Prioridades (MoSCoW)**

- Must: Login, adicionar tarefas.
- Should: Calendário, alarme.
- Could: Filtros avançados.
- Won't: App mobile nativo.

6. Detalhamento dos Casos de Uso

6.1 UC01: Cadastro de Novo Usuário

- Atores: Usuário Anônimo.
- Pré-condições: Usuário acessou a tela inicial e não possui conta.
- Fluxo Principal:
  - Usuário clica em "Criar Conta".
  - Sistema exibe formulário: nome completo, e-mail, senha (confirmação).
  - Usuário submete; sistema valida (e-mail único, senha forte) e armazena hashed no MongoDB.
  - O sistema envia e-mail de confirmação com link de ativação.
  - Usuário clica no link; sistema ativa 2FA (código por e-mail).
  - Usuário insere código; conta é ativada.
- Fluxo Alternativo:
  - 2a: E-mail já existe → Mensagem de erro e opção de login.
- Fluxo de Exceção:
  - 3a: Senha fraca → Erro e prompt para correção.
  - 4a: Falha no e-mail → Retry ou notificação de erro.
- Pós-condições: Conta criada e ativada; usuário redirecionado para login.

6.2 UC02: Login do Usuário

- Atores: Usuário Anônimo (com conta existente).
- Pré-condições: Conta ativada no sistema.
- Fluxo Principal:
  - Usuário insere e-mail e senha na tela de login.
  - Sistema valida credenciais (hash no MongoDB) e gera JWT token.

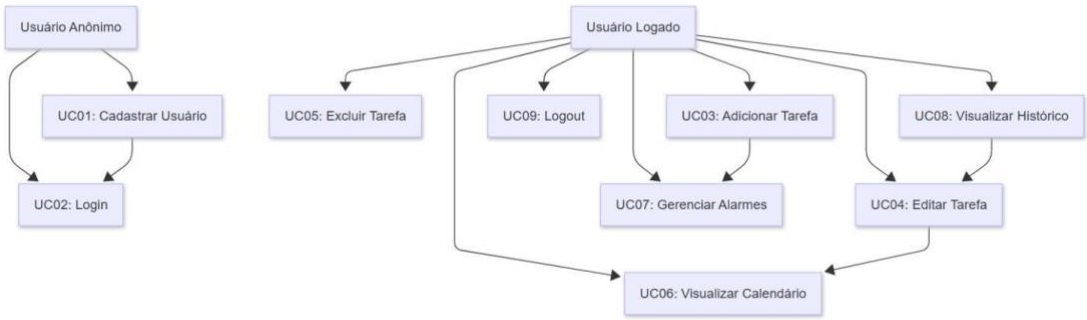
        - Sistema solicita 2FA (envia código por e-mail).
        - Usuário insere código; sistema valida.
        - Sessão iniciada; redireciona para tela principal.
    - Fluxo Alternativo:
        - 1a: "Lembrar-me" → Armazena cookie seguro por 7 dias.
    - Fluxo de Exceção:
        - 2a: Credenciais inválidas → Erro após 5 tentativas (bloqueio temporário).
        - 4a: Código 2FA expirado → Reenvio automático.
    - Pós-condições: Usuário logado; acesso a dados pessoais liberado.
- **6.3 UC03: Adicionar Tarefa**
    - Atores: Usuário Logado.
    - Pré-condições: Usuário na tela principal, logado.
    - Fluxo Principal:
        - Usuário clica no botão "+" na barra superior.
        - Modal expansível abre: Campos para nome, tipo (dropdown com cores), data (date picker), horário (time picker).
        - Usuário submete; sistema valida (data futura, sem conflitos) e salva no PostgreSQL (associado ao user_id).
        - Modal fecha; tarefa aparece na listagem pendente.
    - Fluxo Alternativo:
        - 2a: Seleção rápida de tipo → Aplica cor automática na UI.
    - Fluxo de Exceção:
        - 3a: Conflito de horário → Alerta e sugestão de ajuste.
        - 3b: Campos obrigatórios vazios → Validação em tempo real (vermelho no campo).
    - Pós-condições: Tarefa adicionada; listagem atualizada.
- **6.4 UC04: Editar ou Excluir Tarefa**
    - Atores: Usuário Logado.
    - Pré-condições: Tarefa existe na listagem pendente.
    - Fluxo Principal (Editar):

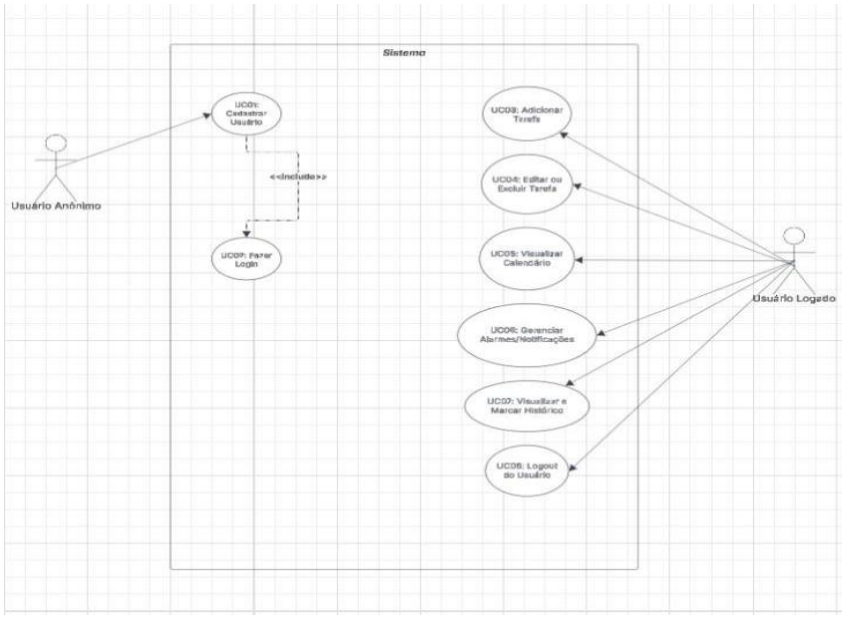
        - Usuário clica em ícone de edição na tarefa.
        - Modal abre com campos pré-preenchidos.
        - Usuário altera e submete; sistema atualiza no BD via PUT.
    - Fluxo Principal (Excluir):
        - Usuário clica em ícone de lixeira.
        - Modal de confirmação: "Confirmar exclusão?".
        - Usuário confirma; sistema deleta via DELETE.
    - Fluxo de Exceção:
        - 3a (Editar): Versão desatualizada (conflito) → Merge ou erro.
    - Pós-condições: Tarefa editada/excluída; listagem atualizada.
- **6.5 UC05: Visualizar Calendário**
    - Atores: Usuário Logado.
    - Pré-condições: Usuário na tela principal.
    - Fluxo Principal:
        - Usuário clica no ícone de Calendário.
        - Partição expansível mostra grade mensal (dias destacados com tarefas).
        - Usuário clica em um dia; mini-bloco exibe tarefas (nome, tipo, horário).
    - Fluxo Alternativo:
        - 2a: Navegação por mês (setas) → Carrega tarefas via API.
    - Fluxo de Exceção:
        - 3a: Dia sem tarefas → Mensagem "Nenhuma tarefa".
    - Pós-condições: Calendário visualizado; usuário pode clicar em detalhes.
- **6.6 UC06: Gerenciar Alarmes/Notificações**
    - Atores: Usuário Logado.
    - Pré-condições: Tarefas agendadas próximas (1 dia/1h antes).
    - Fluxo Principal:
        - Sistema verifica periodicamente (via cron job no back-end ou setInterval no front).

        - Para tarefas próximas, exibe notificação push (título: "Tarefa próxima: [Nome]").
        - Usuário clica no ícone de Alarme; lista de notificações ativas.
        - Usuário descarta notificação (marca como lida).
    - Fluxo Alternativo:
        - 1a: Permissões de notificação negadas → Prompt para ativar.
    - Fluxo de Exceção:
        - 2a: Erro no push → Fallback para badge na UI.
    - Pós-condições: Usuário notificado; alarmes gerenciados.
- **6.7 UC07: Visualizar e Marcar Histórico**
    - Atores: Usuário Logado.
    - Pré-condições: Tarefas concluídas existem.
    - Fluxo Principal:
        - Usuário clica no ícone de Histórico.
        - Partição mostra lista paginada (tarefas marcadas como concluídas, com data de conclusão).
        - Para tarefa pendente na listagem principal: Usuário marca checkbox "Concluída"; sistema atualiza status no BD.
        - Tarefa move para histórico.
    - Fluxo Alternativo:
        - 2a: Filtros (data/tipo) → API query com parâmetros.
    - Fluxo de Exceção:
        - 3a: Falha na marcação → Retry ou erro de rede.
    - Pós-condições: Histórico atualizado; tarefa transferida.
- **6.8 UC08: Logout do Usuário**
    - Atores: Usuário Logado.
    - Pré-condições: Usuário na tela principal.
    - Fluxo Principal:
        - Usuário clica em botão de logout (na barra superior ou menu).
        - Sistema invalida JWT token e limpa cookies/sessão.
        - Redireciona para tela de login.

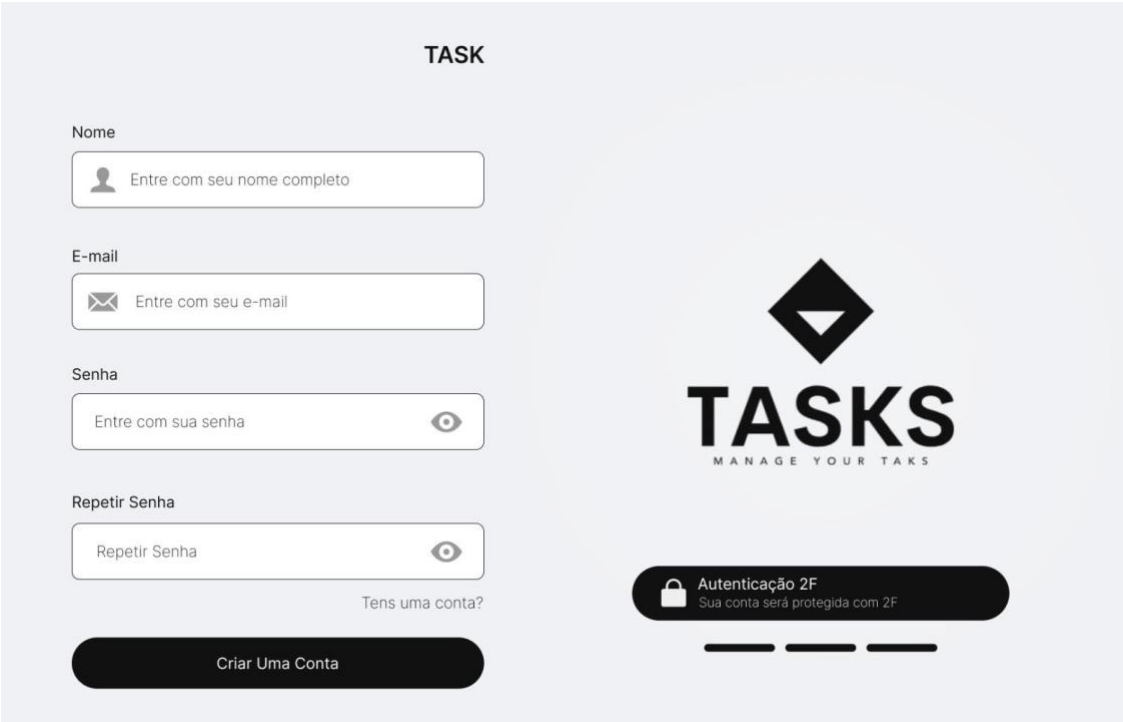
- Fluxo Alternativo:
  - 1a: Confirmação modal → "Deseja sair?".
- Fluxo de Exceção:
  - 2a: Erro na invalidação → Logout forçado via front-end.
- Pós-condições: Sessão encerrada; acesso bloqueado.

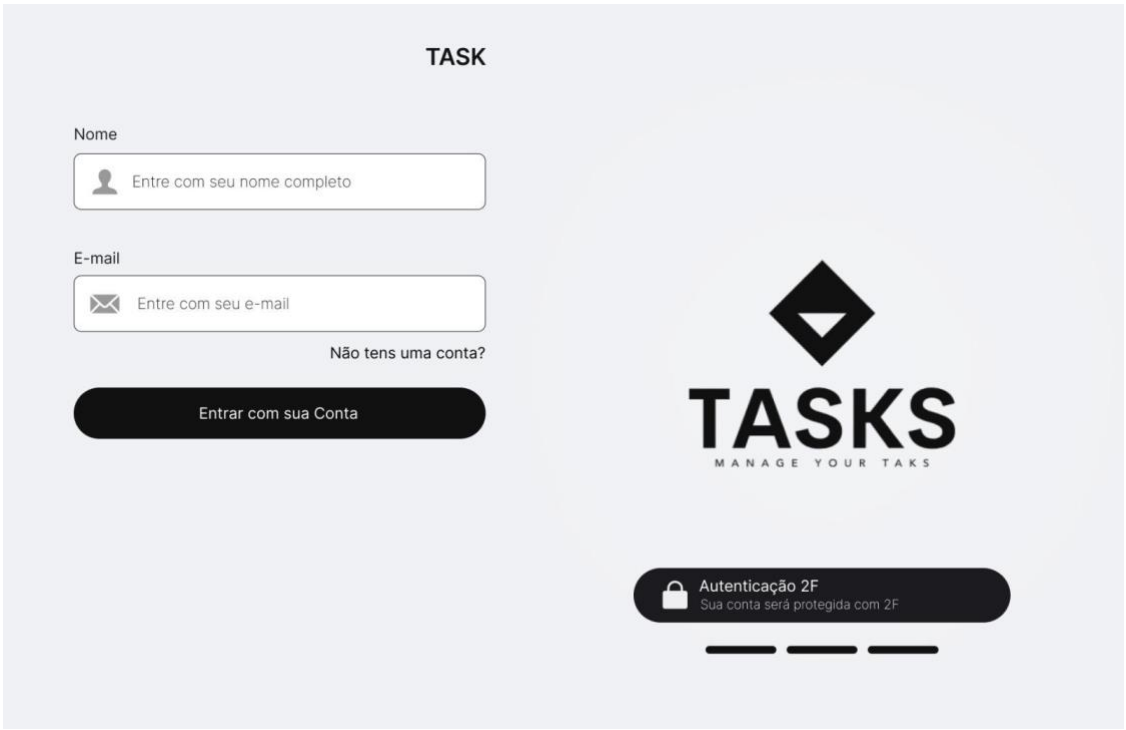
6.9 Diagrama de Caso de Uso





- **7. Layout do Sistema de Login (Figma)**




TASK
Nome
Entre com seu nome completo
E-mail
Entre com seu e-mail
Não tens uma conta?
Entrar com sua Conta
TASKS
MANAGE YOUR TAKS
Autenticação 2F
Sua conta será protegida com 2F