Épico 1 - MVP Funcionalidades básicas

Objetivo: Criar as funcionalidades básicas para que o usuário gerencie suas tarefas de forma organizada.

História 1.1 — CRUD de Tarefas

- ➔ **Como Usuário, quero criar, editar e excluir atividades para organizar meu trabalho**
  - ◆ **Critérios de Aceitação:**
    - O usuário consegue criar uma nova atividade com título e descrição.
    - O usuário consegue editar o título e a descrição de uma atividade existente.
    - O usuário consegue excluir uma atividade.
    - O sistema exibe feedback visual claro para cada ação (criação, edição, exclusão).
  - ◆ **Subtarefas:**
    - Desenvolver telas de formulário para criar, editar e excluir de atividades.
    - Implementar feedback visual das ações (erro ou confirmação)
    - Implementar a lógica de back-end no banco de dados (criar,editar e excluir).
    - Validar dados inseridos nos formulários (ex: campos obrigatórios, formato).
    - Desenvolver integração da interface com o back-end para operações de CRUD

História 1.2 — Anexos em Tarefas

- ➔ **Como Usuário, quero anexar arquivos em tarefas ao concluir para garantir a entrega**
  - ◆ **Critérios de Aceitação:**
    - O usuário consegue anexar um arquivo a uma tarefa concluída ou em edição.
    - O usuário consegue visualizar os arquivos anexados à tarefa.
    - O usuário consegue baixar os arquivos anexados.
  - ◆ **Subtarefas:**
    - Desenvolver interface para um campo de upload de arquivos na tela de conclusão/edição da tarefa.
    - Implementar lógica de back-end para receber e armazenar os arquivos anexados
    - Implementar funcionalidade para permitir ao usuário visualizar os arquivos anexados à tarefa.
    - Implementar funcionalidade para permitir ao usuário baixar os arquivos anexados.

História 1.3 — Atribuição a membros

- ➔ **Como usuário, quero atribuir atividades a membros da equipe para distribuir as responsabilidades**
  - ◆ **Critérios de Aceitação:**

- O usuário consegue selecionar um membro da equipe e atribuí-lo a uma atividade.
- O sistema exibe claramente o membro da equipe atribuído a cada atividade.

◆ **Subtarefas:**

- Desenvolver interface para exibir a lista de membros da equipe (ex: dropdown de seleção ou lista de cards).
- Desenvolver interface para permitir ao usuário selecionar um membro da equipe e associá-lo a uma atividade (no momento de criar ou editar a atividade).
- Implementar lógica de back-end para armazenar a informação de quem está atribuído a cada atividade.
- Implementar lógica de back-end para permitir a filtragem de atividades por membro da equipe.
- Implementar feedback visual na interface para mostrar claramente a quem uma atividade está atribuída.

História 1.4 — Autenticação Básica

➔ **Como usuário, quero acessar a plataforma com Login próprio para maior autonomia e segurança**

◆ **Critérios de Aceitação:**

- O usuário consegue registrar uma nova conta.
- O usuário consegue fazer login com suas credenciais.
- O sistema valida as credenciais de login e oferece feedback em caso de falha.

◆ **Subtarefas:**

- Desenvolver interface de login (campos de usuário/email e senha).
- Implementar funcionalidade de registro de novos usuários.
- Implementar lógica de back-end para autenticação e autorização de usuários
- Desenvolver telas de feedback para login bem-sucedido ou falha (ex: mensagens de erro).

História 1.5 — Log de Auditoria

➔ **Como usuário, quero visualizar um log de auditoria para ter melhor controle das alterações da equipe em relação às tarefas**

◆ **Critérios de Aceitação:**

- O sistema registra automaticamente criações, edições em campos chave da tarefa e a exclusão lógica, incluindo usuário, data/hora, tipo de alteração e dados relevantes (antigos/novos para edições).
- O administrador acessa uma interface que exibe o log de auditoria de forma legível, mostrando quem, quando e o que foi alterado em cada registro.
- Os registros no log de auditoria são imutáveis após a criação.

◆ **Subtarefas:**

- Criar modelo de dados para armazenar informações de auditoria (usuário, data/hora, tipo de alteração, dados antigos/novos, ID da tarefa).

- Implementar lógica para registrar alterações em todos os campos da tarefa (nome, descrição, tema, status, responsável, data de entrega e anexo) no log de auditoria.
- Implementar lógica para registrar a exclusão lógica (soft delete) da tarefa no log de auditoria.
- Criar uma interface de usuário para exibir o log de auditoria, acessível apenas por usuários com perfil de administrador.
- Integrar o log de auditoria com as funcionalidades de criação, edição e exclusão de tarefas (História 1.1).

História 1.6 — Visualização de Tarefas

➔ **Como usuário, quero visualizar tanto as tarefas gerais da equipe quanto às minhas por meio de um perfil, para melhor gerenciamento**

◆ **Critérios de Aceitação:**

- Para usuários com perfil de administrador:
  - O sistema exibe todas as tarefas da equipe.
  - O sistema permite alternar para visualizar apenas as tarefas atribuídas ao próprio administrador.
  - O sistema indica claramente qual visualização está ativa (todas as tarefas ou minhas tarefas).
- Para usuários comuns:
  - O sistema exibe apenas as tarefas atribuídas a ele.

◆ **Subtarefas:**

- Desenvolver e implementar a lógica de perfis de usuário (administrador e comum).
- Implementar funcionalidade de filtro na interface para que o administrador possa alternar entre "Minhas Tarefas" e "Todas as Tarefas da Equipe".
- Implementar lógica de back-end para filtrar as tarefas atribuídas ao administrador logado e para exibir todas as tarefas da equipe.
- Implementar feedback visual claro na interface para indicar qual visualização (minhas tarefas ou tarefas da equipe) está ativa.
- Desenvolver interface para exibir exclusivamente a lista de tarefas atribuídas ao usuário comum logado, sem acesso a tarefas de outros usuários ou da equipe geral.
- Implementar lógica de back-end para filtrar as tarefas garantindo que apenas as atribuídas ao usuário comum logado sejam retornadas, impedindo o acesso a dados de tarefas não relacionadas a ele.

História 1.7 - Notificações de Tarefas

➔ **Como usuário, quero receber notificações das minhas tarefas na interface para assim ter melhor autonomia do meu tempo.**

◆ **Critérios de Aceitação:**

- O sistema exibe notificações automáticas na interface web quando ocorrem eventos relevantes relacionados às tarefas do usuário (ex:

tarefa criada, editada, excluída, prazo próximo, tarefa atrasada, tarefa atribuída).
    - As notificações contém informações relevantes sobre a tarefa (ex: título, prazo, quem atribuiu/editou/excluiu).
    - O usuário consegue acessar a tarefa diretamente através da notificação na interface.
    - O sistema exibe um histórico de notificações recebidas na interface.
- **Subtarefas:**
    - Desenvolver interface para exibir notificações (Componente de notificação com dropdown).
    - Implementar lógica de back-end para disparar notificações na interface para tarefas relacionadas ao usuário logado, com base em eventos de CRUD (criação, edição, exclusão, prazo, atribuição, etc.).
    - Desenvolver templates para os diferentes tipos de notificações a serem exibidas no sistema.
    - Implementar funcionalidade para que, ao clicarem nas notificações, os usuários sejam direcionados às tarefas.
    - Implementar funcionalidade para exibir o histórico de notificações no componente de notificação com dropdown.

História 1.8 - Recuperação de Senha via E-mail

➔ **Como usuário, quero recuperar minha senha via e-mail para garantir maior segurança da minha conta.**

- **Critérios de Aceitação:**
    - O usuário consegue solicitar a recuperação de senha informando seu e-mail cadastrado.
    - O sistema envia um e-mail com um link para redefinição de senha para o e-mail do usuário.
    - O link de redefinição de senha possui uma validade limitada (30 minutos).
    - O usuário consegue redefinir sua senha através do link.
    - O sistema valida a nova senha de acordo com os requisitos de segurança (ex: mínimo de caracteres, complexidade).
    - Após a redefinição, a senha antiga não é mais válida.
    - O sistema exibe feedback visual claro sobre o sucesso ou falha da solicitação e da redefinição de senha.
- **Subtarefas:**
    - Desenvolver interface para solicitação de recuperação de senha (componente na tela de login e campo de e-mail e botão de envio).
    - Implementar lógica de back-end para gerar e enviar e-mails com links temporários de redefinição de senha.
    - Desenvolver interface para redefinição de senha (campos para nova senha e confirmação).

- Implementar lógica de back-end para validar e atualizar a senha do usuário.
- Definir e implementar a validade do link temporário de redefinição de senha.
- Implementar validação da nova senha no front-end e back-end.
- Desenvolver telas de feedback para sucesso e erro na recuperação e redefinição de senha.

História 1.9 - Gerenciamento de Equipes

➔ **Como usuário, quero criar e gerenciar equipes e seus membros para organizar a colaboração e a atribuição de tarefas.**

◆ **Critérios de Aceitação:**

- O administrador pode criar, editar (nome/descrição) e excluir equipes, tornando-se automaticamente o administrador inicial.
- O administrador pode adicionar e remover membros (por e-mail) e designá-los como administradores de equipe.
- O usuário pode visualizar todas as equipes das quais participa e seus membros.
- O sistema exibe feedback visual claro para todas as ações de gerenciamento de equipes.
- A aba "Equipes" integra-se dinamicamente com outras páginas, atualizando as informações com base na equipe selecionada; um aviso é exibido se nenhuma equipe estiver ativa.
- O usuário recebe notificações de convites (interface e e-mail) para equipes, podendo aceitar ou recusar através de um modal.

◆ **Subtarefas:**

- Desenvolver a interface da aba de equipes, incluindo formulários para criar, editar e excluir equipes, além da exibição da lista de equipes e seus membros.
- Implementar a funcionalidade na interface para o usuário selecionar uma equipe ativa, incluindo a exibição clara da equipe selecionada e a navegação entre as equipes.
- Criar a interface para o botão de notificação de convite e o modal que exibe os convites recentes, com opções para aceitar ou recusar.
- Implementar a lógica de front-end para que, ao selecionar uma equipe, todas as outras páginas sejam atualizadas dinamicamente, exibindo informações da equipe selecionada e a exibição de avisos quando nenhuma equipe estiver ativa.
- Implementar o CRUD de equipes no back-end, incluindo gerenciamento de membros e administradores de equipe, a associação automática do criador como administrador inicial e controle de nome.

- Implementar a lógica de back-end para o disparo de notificações de convite, o registro e as ações de aceitar/recusar convites, assim como um e-mail de notificação com link de direcionamento para a página equipe.
- Implementar a lógica de back-end para filtrar e apresentar dados nas outras páginas com base na equipe selecionada, garantindo que apenas as informações relevantes para a equipe ativa sejam exibidas.

História 2.0 — Calendário

➔ **Como usuário, quero gerenciar minhas tarefas em um calendário integrado na plataforma, com a opção de sincronizá-las com o Google Calendar, para otimizar meu planejamento pessoal e de equipe.**

◆ **Critérios de Aceitação:**

- **Calendário Integrado:**
  - O usuário consegue visualizar todas as suas tarefas e seus respectivos prazos em um calendário exibido diretamente na plataforma.
  - O calendário integrado deve permitir a visualização em diferentes formatos (ex: dia, mês e ano).
  - O usuário consegue interagir com as tarefas diretamente no calendário (ex: clicar para ver detalhes, redirecionar para a tarefa).
- **Calendário Externo (Google Calendar):**
  - O usuário consegue conectar sua conta do Google Calendar ao sistema.
  - As atividades com prazo de conclusão devem ser automaticamente sincronizadas com eventos no Google Calendar.
  - Alterações nos prazos das atividades no sistema devem ser refletidas nos eventos do Google Calendar.
  - O usuário pode desativar a sincronização com o Google Calendar a qualquer momento.
- **Feedback de Status da Integração:**
  - O sistema fornece feedback visual sobre o status da integração com o Google Calendar (conectado/desconectado, erros de sincronização).

◆ **Subtarefas:**

- Desenvolver a interface do calendário integrado (visualização de dia, mês e ano).
- Implementar a lógica de back-end para gerenciar tarefas no calendário integrado (ex: buscar, exibir, interagir com tarefas).
- Desenvolver interface para as configurações de integração com calendários (botões para conectar/desconectar Google Calendar ou outro calendário).

- Implementar a lógica de back-end para autenticação e autorização via APIs do Google Calendar ou outro calendário.
- Implementar a lógica de back-end para criar, atualizar e excluir eventos nos calendários externos com base nos prazos das atividades.
- Garantir a independência na sincronização de eventos para evitar duplicações ou inconsistências.
- Desenvolver feedback visual para o status da integração com o calendário (conectado/desconectado, erros de sincronização).

História 2.1 — Modal de Tarefa

➔ **Como usuário, quero clicar em uma tarefa e visualizar um modal com seu resumo, histórico de alterações (para administradores) e a sinalização visual de prazos vencidos, para ter acesso rápido às informações importantes.**

◆ **Critérios de Aceitação:**

- O usuário consegue clicar em qualquer tarefa na lista e um modal é exibido com os detalhes da tarefa (título, descrição, prazo, responsável, status, anexos, etc.).
- O modal de visualização é somente leitura, não permitindo edições diretas.
- O modal pode ser fechado facilmente (ex: clicando fora ou em um botão "Fechar").
- O modal exibe todos os anexos da tarefa, permitindo a visualização ou download.
- **Para usuários com perfil de administrador:** O modal exibe um histórico de todas as alterações feitas naquela tarefa.
- O histórico de alterações mostra quem fez a alteração, a data/hora, o tipo de alteração e os dados relevantes (antigos/novos para edições de campos).
- O log de auditoria exibido é filtrado automaticamente para mostrar apenas as alterações da tarefa que está sendo visualizada.
- O sistema exibe um ícone de alerta (ex: triângulo vermelho) nos cards de tarefas cujo prazo de conclusão já passou, tanto na lista principal quanto dentro do modal de visualização.
- O ícone de alerta é claramente visível e se destaca no card da tarefa e no modal.
- O ícone de alerta desaparece automaticamente assim que a tarefa é concluída ou seu prazo é atualizado para uma data futura.
- Ao passar o mouse sobre o ícone de alerta (tanto no card quanto no modal), uma pequena dica de ferramenta (tooltip) é exibida, informando que a tarefa está atrasada.

◆ **Subtarefas:**

- Desenvolver a interface do modal de visualização de tarefas.
- Implementar a lógica de front-end para exibir o modal ao clicar em uma tarefa.

- Implementar a lógica de back-end para buscar todos os detalhes da tarefa ao ser solicitada para o modal.
- Desenvolver a interface para exibir o log de alterações de forma clara e organizada dentro do modal.
- Implementar a lógica de back-end para filtrar o log de auditoria por ID da tarefa para exibição no modal.
- Garantir que apenas administradores possam visualizar este log específico no modal.
- Implementar a lógica de back-end para identificar tarefas com prazo de conclusão vencido.
- Implementar a lógica de front-end para exibir o ícone de alerta nos cards das tarefas atrasadas e no modal de visualização.
- Implementar a lógica para remover o ícone de alerta quando a tarefa for concluída ou o prazo for atualizado.

Épico 2 - Interface Intuitiva e Multiplataforma

Objetivo: Proporcionar uma experiência de usuário responsiva e intuitiva em diferentes dispositivos, com foco na autonomia do usuário e na entrega de informações importantes.

História 2.2 — Filtragem do Log

➔ **Como Administrador, quero filtrar o log de auditoria por data, usuário e tipo de alteração, para encontrar informações específicas sobre as modificações das tarefas da equipe.**

- ◆ **Critérios de Aceitação:**
  - O administrador consegue filtrar o log de auditoria por um período de datas específico (data específica ou início e fim).
  - O administrador consegue filtrar o log de auditoria por um usuário que realizou a ação.
  - O administrador consegue filtrar o log de auditoria por tipo de evento (criação, edição, exclusão).
  - O sistema permite combinar múltiplos filtros (ex: por data E por usuário).
  - O sistema exibe os resultados do filtro de forma clara e organizada.

- O sistema mantém a ordenação padrão do log (mais recente primeiro) após a aplicação dos filtros.

◆ **Subtarefas:**

- Desenvolver interface para campos de filtro de data, seleção de usuário (dropdown ou campo de busca) e seleção de tipo de evento (dropdown).
- Implementar lógica de back-end para aplicar os filtros de data, usuário e tipo de evento na consulta ao log de auditoria.
- Garantir que a lógica de back-end suporte a combinação de múltiplos filtros.
- Atualizar a interface para exibir os resultados filtrados dinamicamente.
- Implementar feedback visual claro na interface para indicar quais filtros estão ativos.
- Garantir que a ordenação do log por data seja mantida após a aplicação dos filtros.

História 2.3 — Filtro de Tarefas

➔ **Como usuário, quero filtrar minhas tarefas por meio de uma barra de pesquisa e um botão de filtro para encontrar e gerenciar minhas atividades de forma eficiente.**

◆ **Critérios de Aceitação:**

- O usuário consegue visualizar uma barra de pesquisa para buscar tarefas específicas por título, descrição ou outros campos relevantes.
- O usuário consegue visualizar um botão de filtro que, ao ser acionado, exibe opções para filtrar tarefas por:
  - Categoria
  - Nome
  - Título
  - Descrição
  - Prazo
  - Usuário
  - Status (Não Iniciada, Em Andamento, Concluída, Atrasada)
- O sistema permite a combinação de múltiplos critérios de filtro e pesquisa.
- O sistema exibe os resultados da pesquisa e da filtragem de forma clara e organizada.
- A aplicação dos filtros e da pesquisa deve ser responsiva e apresentar os resultados rapidamente.

◆ **Subtarefas:**

- Desenvolver a interface da barra de pesquisa de tarefas para busca direta.
- Desenvolver a interface para um único botão de filtro que, ao ser clicado, abre um modal com as opções de filtro para todos os campos de tarefa (categoria, nome, título, descrição, prazo, usuário e status).
- Implementar a lógica de back-end para buscar tarefas com base no texto da barra de pesquisa, considerando todos os campos relevantes.
- Implementar a lógica de back-end para filtrar tarefas com base em todos

os critérios selecionados pelo usuário por meio do botão de filtro (categoria, nome, título, descrição, prazo, usuário e status).
- Atualizar a interface para exibir os resultados filtrados e pesquisados dinamicamente.
- Atualizar a interface para a tarefa pesquisada na barra de pesquisa
- Implementar feedback visual claro na interface para indicar quais filtros estão ativos e se há resultados para a pesquisa.

História 2.4 — Arrastar e Soltar Tarefas

➔ **Como usuário, quero arrastar e soltar tarefas entre diferentes colunas de status para gerenciar meu trabalho de forma intuitiva.**

◆ **Critérios de Aceitação:**

- O usuário consegue visualizar todas as suas tarefas em um quadro com colunas distintas para cada status (ex: "Não Iniciada", "Em Andamento", "Concluída").
- O usuário consegue arrastar uma tarefa de uma coluna de status para outra (ex: de "Não Iniciada" para "Em Andamento").
- Ao arrastar uma tarefa para uma nova coluna, o status da tarefa é automaticamente atualizado no sistema.
- O sistema exibe feedback visual claro durante a ação de arrastar e soltar (ex: realce da área de destino, animação da tarefa).
- O usuário consegue clicar em uma tarefa no quadro para visualizar seus detalhes.
- O quadro de tarefas é responsivo e funciona bem em diferentes tamanhos de tela.

◆ **Subtarefas:**

- Implementar a funcionalidade de arrastar e soltar (drag-and-drop) para as tarefas no front-end.
- Desenvolver a lógica de back-end para atualizar o status da tarefa no banco de dados quando ela é movida entre colunas.
- Implementar feedback visual para as ações de arrastar e soltar.