

Plano de Tarefas para o MVP

Jullio (Core Backend & APIs)

Objetivo: Finalizar a camada de APIs REST, autenticação e a estrutura administrativa.

Prioridade Tarefa	Módulo	Dependências
Alta		
Finalizar ViewSets de EmailMessage: Implementar os filtros (status, busca por assunto/remetente) e paginação no EmailMessageViewSet.	emails/views.py	emails/models.py
Configurar o Scheduler: Adicionar as configurações do Django-Q Scheduler no settings.py para garantir que o worker fetch_emails de Thales seja executado a cada 5 minutos.	project/settings.py	Nenhuma
Média		
Health Check Endpoint: Criar um endpoint simples (/healthz) para monitoramento de <i>uptime</i> e que o sistema está rodando.	core/views.py	Nenhuma
Média		
Criar Superusuário Admin: Criar o modelo de usuário (User ou CustomUser) se ainda não houver, e garantir que o admin do admin.py Django funcione para gerenciar MailBox, EmailMessage e IntegrationLog.		emails/models.py, integrations/models.py
Baixa		
Finalizar Testes Unitários: Expandir tests/test_api.py para cobrir todos os métodos dos ViewSets (Detail, Reprocess, Filtros).	emails/tests/	Nenhuma

Exportar para as Planilhas

Thales (Workers & Integrações)

Objetivo: Fechar o ciclo do pipeline: buscar o email e orquestrar a extração e as APIs externas, registrando tudo.

Prioridade Tarefa	Módulo	Dependências
Alta	Implementar process_email: Finalizar a lógica no worker para	tasks/tasks.py
		Juliano (schemas.py)

	Prioridade Tarefa	Módulo	Dependências
	chamar a extração de Juliano, salvar o JSON, chamar as integrações de Trello/Telegram e atualizar o IntegrationLog.		
Alta	<p>Integração Trello: Implementar a lógica completa em <code>create_trello_card</code>, incluindo manipulação de exceções e registro de <code>IntegrationLog</code> (SUCESSO/FALHA).</p> <p>Integração Telegram: Implementar a lógica completa em <code>notify_telegram</code> e registro de <code>IntegrationLog</code>.</p> <p>Mocks de Integração: Criar mocks para <code>trello.py</code> e <code>telegram.py</code> para testes unitários, evitando chamadas de API reais durante o CI/CD.</p>	<code>integrations/trello.py</code> <code>integrations/telegram.py</code> <code>integrations/tests/</code>	<code>integrations/models.py</code> <code>integrations/models.py</code> Nenhuma
	Exportar para as Planilhas		

Juliano (IA & Extração)

Objetivo: Finalizar a camada de inteligência e garantir a robustez da extração de dados.

	Prioridade Tarefa	Módulo	Dependências
	<p>Finalizar Schemas Pydantic: Criar/revisar os Schemas Pydantic, garantindo que o Annotated seja usado corretamente para evitar erros de Pylance/tipagem. Implementar os três principais schemas de negócio (ex: <code>ServiceOrder</code>, <code>SupportRequest</code>, <code>Report</code>).</p>	<code>extraction/schemas.py</code>	Nenhuma
Alta	<p>Finalizar ai_wrapper: Concluir a lógica de <code>fallback</code> (re-prompts) e o tratamento de exceções (e.g.,</p>	<code>extraction/ai_wrapper.py</code>	<code>openai</code>

	Prioridade Tarefa	Módulo	Dependências
Média	<p>Timeout, AuthError do OpenAI) na função extract_fields_from_text.</p> <p>Testes de Extração: Criar testes unitários para o ai_wrapper.py usando o mock (sem chamar a API real) para validar se a função lida corretamente com JSONs válidos, inválidos e se o <i>fallback</i> funciona.</p>		
Média	<p>Documentação de Prompts: Revisar e detalhar o docs/prompts.md com os prompts finais e exemplos de <i>few-shot</i> se houver.</p>	extraction/tests/ docs/prompts.md	Nenhuma

Exportar para as Planilhas

Próximo Foco do Time: Integração Crítica

A próxima etapa de integração mais crítica é:

Thales testar o process_email para garantir que ele consiga:

1. Receber o ID do EmailMessage de **Jullio**.
2. Chamar o extract_fields_from_text de **Juliano**.
3. Salvar o JSON retornado no campo extracted_data do modelo de **Jullio**.