

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 |
| Alta | Configurar o Scheduler: Adicionar as configurações do Django-Q Scheduler no settings.py para garantir que o <i>worker</i> 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 <i>admin</i> do Django funcione para gerenciar MailBox, EmailMessage e IntegrationLog. | admin.py | 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 <i>worker</i> 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. | | |
| | Integração Trello: Implementar a lógica completa em | | |
| Alta | create_trello_card, incluindo a manipulação de exceções e registro de IntegrationLog (SUCESSO/FALHA). | integrations/trello.py | integrations/models.py |
| | Integração Telegram: Implementar a lógica completa em | | |
| Alta | notify_telegram e registro de IntegrationLog. | integrations/telegram.py | integrations/models.py |
| | Mocks de Integração: Criar <i>mocks</i> para trello.py e telegram.py para testes unitários, evitando chamadas de API reais durante o CI/CD. | | |
| Média | | integrations/tests/ | 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 |
|-------------|---|--------------------------|----------------|
| | 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: ServiceOrder, SupportRequest, Report). | | |
| Alta | | extraction/schemas.py | Nenhuma |
| | Finalizar ai_wrapper: Concluir a lógica de <i>fallback</i> (re-prompts) e o tratamento de exceções (e.g., | | |
| Alta | | extraction/ai_wrapper.py | openai |

| Prioridade | Tarefa | Módulo | Dependências |
|------------|---|-------------------|--------------|
| | Timeout, AuthError do OpenAI) na função <code>extract_fields_from_text</code> . | | |
| Média | Testes de Extração: Criar testes unitários para o <code>ai_wrapper.py</code> 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. | extraction/tests/ | Nenhuma |
| Média | Documentação de Prompts: Revisar e detalhar o <code>docs/prompts.md</code> com os prompts finais e exemplos de <i>few-shot</i> se houver. | 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**.