

## Jullio (Core Backend & Segurança da Infra)

**Objetivo:** Finalizar a camada de APIs REST, autenticação, estrutura administrativa, e **aplicar *hardening*** de segurança em todo o *framework* Django.

Prioridade	Tarefa	Módulo(s)	Critério de Aceite (Segurança)
SI - Alta	Adotar <b>django-environ</b> e <b>django-cors-headers</b>	requirements.txt	Instalar bibliotecas de segurança e atualizar settings.py para usar env.db(), env.list(), etc.
SI - Alta	Hardening do settings.py	project/settings.py	Definir <b>X_FRAME_OPTIONS = 'DENY'</b> , <b>SECURE_BROWSER_XSS_FILTER = True</b> , e configurar listas <b>estritas</b> de CORS_ALLOWED_ORIGINS e CSRF_TRUSTED_ORIGINS.
SI - Alta	Atualizar .env.sample	.env.sample	Incluir novas variáveis de segurança (ALLOWED_HOSTS, CORS_ALLOWED_ORIGINS, etc.) forçando a configuração externa.
Alta	Configuração do Scheduler (Django-Q)	project/settings.py	Configurar o Q_CLUSTER e agendar a chamada periódica para a função <i>wrapper</i> de <i>fetch</i> de emails.
Média	Finalizar ViewSets e Filtros	emails/views.py	Implementar os filtros (status, busca por q=) e paginação no EmailMessageViewSet.
Média	Health Check Endpoint	core/views.py	Garantir que o endpoint /healthz verifique o <i>uptime</i> do serviço e a conectividade do banco de dados (DB).
Baixa	Testes Finais de API:	emails/tests/	Expandir a cobertura do <b>Pytest</b> para garantir que os filtros e o endpoint de <i>reprocess</i> funcionem corretamente.

Exportar para as Planilhas

---

## Thales (Workers, Integrações & SI)

**Objetivo:** Implementar o pipeline assíncrono (Django-Q) e as integrações externas, garantindo o

registro de logs e a segurança no tratamento de dados.

Prioridade Tarefa	Módulo(s)	Critério de Aceite (Segurança/Funcional)
SI - Alta	<b>Segurança de Secrets (MailBox)</b> tasks/tasks.py	Garantir que as credenciais do MailBox sejam usadas apenas para a conexão IMAP e <b>NUNCA</b> sejam salvas em <i>logs</i> ou <i>exceptions</i> de forma <i>plaintext</i> .
Alta	<b>Implementar process_email (Orquestrador)</b> tasks/tasks.py	Finalizar o <i>worker</i> para salvar o JSON de Juliano, chamar as integrações de Trello/Telegram, e atualizar o IntegrationLog (SUCESSO/FALHA).
Alta	<b>Integração Trello com Log</b> integrations/trello.py	Criar a função <i>create_trello_card</i> que registra o <i>payload</i> (exceto <i>secrets</i> ) e o <i>status</i> (SUCESSO/FAILED) no IntegrationLog.
Alta	<b>Integração Telegram com Log</b> integrations/telegram.py	Criar a função <i>notify_telegram</i> com registro de log.
Média	<b>Implementação Mocks de Integração</b> integrations/tests/	Criar <i>mocks</i> para Trello/Telegram para testes unitários, garantindo que o <i>process_email</i> possa ser testado sem chaves de API.

Exportar para as Planilhas

---

### Juliano (IA, Extração & SI)

**Objetivo:** Implementar a lógica de extração da IA, validação de dados (Pydantic) e construir os mecanismos de segurança contra uso indevido da API (custo).

Prioridade Tarefa	Módulo(s)	Critério de Aceite (Segurança/Funcional)
SI - Alta	<b>Controle de Custo (Denial of Wallet)</b> extraction/ai_wrapper.py	Implementar lógica de <b>limitação</b> de tokens/tamanho do texto no <i>wrapper</i> e, idealmente, um contador simples para prevenir <i>loops</i> infinitos de re-prompt.
Alta	<b>Finalizar</b> extraction/ai_wrapper.py	Concluir o tratamento de erros (exceções

Prioridade	Tarefa	Módulo(s)	Critério de Aceite (Segurança/Funcional)
	<b>ai_wrapper (Re-prompts)</b>		da API e ValidationError do Pydantic) com a estratégia de <i>re-prompts</i> e <i>fallback</i> final (return None).
<b>Alta</b>	<b>Finalizar Schemas Pydantic</b>	extraction/schemas.py	Garantir que os schemas sejam válidos e usem Annotated[int, conint(...)] para as restrições, definindo a estrutura de dados para o Trello.
<b>Média</b>	<b>Sanitização de Entrada de IA</b>	extraction/ai_wrapper.py	Implementar uma sanitização simples do body_text para remover ou mascarar PII (Informações Pessoais Identificáveis) antes de enviar à API externa.
<b>Média</b>	<b>Testes de Extração:</b>	extraction/tests/	Criar testes para o <i>wrapper</i> (usando <i>mock</i> ) para validar as regras de <b>JSON válido/inválido</b> e o acionamento do <i>fallback</i> .

Exportar para as Planilhas

---

Com este plano, o **Julio** estabiliza o lado da segurança, e **Thales** e **Juliano** têm tarefas claras para a lógica de negócio, aderindo às novas regras de SI