

Diário de Bordo: Projeto Gerador de Equipes de IA

Este documento registra as etapas de desenvolvimento, desafios encontrados, soluções aplicadas e próximos passos para o seu projeto de Gerador de Equipes de IA com CrewAI.

Fase 0: Preparação do Ambiente e Ferramentas

Problema Inicial (1): ImportError: cannot import name 'tool' from 'crewai_tools' e PydanticDeprecatedSince20 warnings.

Solução Aplicada (1): Corrigido o import da ferramenta tool de crewai_tools para crewai.tools em equipe_mestre/ferramentas.py.


Resultado (1): ImportError resolvido, permitindo o início da execução da Crew.

Fase 1: Interface do Usuário (Streamlit)

Status: A interface Streamlit (app.py) foi planejada e implementada com campos para o prompt do usuário, uploader de arquivo e área de saída.

Observações: A interface tem sido a base para todos os testes e visualização das saídas.


Fase 2: Definição da Equipe-Mestre (5 Agentes)

 Agente 1: Analista de Requisitos

Papel: Analista de Requisitos de IA.

Desafio Persistente: O agente ainda retorna uma Final Answer genérica como "I now can give a great answer." ou "Thought: I now can give a great answer." em vez do conteúdo Markdown estruturado de requisitos. Isso impacta a entrada para os agentes subsequentes.

Status Atual: Definido em equipe_mestre/agentes.py e sua tarefa em equipe_mestre/tarefas.py.

 Agente 2: Especialista em Ferramentas

Papel: Especialista em Ferramentas de Agentes de IA.

Ferramenta: Utiliza Catálogo de Ferramentas da CrewAI.

Desafio Persistente: Similar ao Agente 1, o agente às vezes retorna uma Final Answer genérica ("I should start by listing all the tools..."). Embora o log da ferramenta mostre que ele acessa o catálogo e até formula a resposta correta em seus "Thoughts" (o bloco Markdown das ferramentas), a Final Answer capturada pela Crew não é sempre a esperada.

Status Atual: Definido em equipe_mestre/agentes.py e sua tarefa em equipe_mestre/tarefas.py.

 Agente 3: Designer de Equipes

Papel: Arquiteto de Equipes de IA.

Desafio Persistente: O agente ainda retorna uma Final Answer genérica ("I now can give a great answer.") em vez do objeto JSON do plano de design. O log mostra que ele está pensando na estrutura JSON, mas não a está produzindo como a Final Answer esperada.

Status Atual: Definido em equipe_mestre/agentes.py e sua tarefa em equipe_mestre/tarefas.py.

🚀 Agente 4: Implementador de Equipes Python

Papel: Engenheiro de Software de IA.

Ferramenta: Utiliza Escritor de Código Python.

Sucesso Parcial: O agente conseguiu gerar o código Python em um bloco Markdown, o que é um grande avanço na formatação da saída.

Desafios Atuais:

Erro na Chamada da Ferramenta: No log (s3.txt), houve tentativas falhas de chamar o Escritor de Código Python porque o agente tentou usar um nome de ferramenta incorreto (Criar Código Python) ou passou um input no formato errado (um dicionário encapsulado em string, quando a ferramenta esperava uma string pura para code_string).

Saída no Frontend: A saída exibida no frontend para o "Código Python Gerado" é a descrição da própria tarefa do Agente 4, e não o código Python real. Isso ocorre porque o agente, ao falhar em usar a ferramenta para emitir o código, acaba retornando a si mesmo o prompt que recebeu.

Imports de Ferramentas: O código que o Agente 4 tenta gerar ainda pode ter inconsistências nos imports de ferramentas (e.g., crewai_tools vs. crewai.tools ou imports locais).

Status Atual: Definido em equipe_mestre/agentes.py e sua tarefa em equipe_mestre/tarefas.py.

🚀 Agente 5: Validador de Código (QA)

Papel: Revisor de Qualidade de Código.

Ferramenta: Utiliza Verificador de Sintaxe Python.

Sucesso Parcial: No log do terminal (s3.txt), o Verificador de Sintaxe Python reportou "Sucesso: O código é sintaticamente válido."

Desafio Crítico: Há uma contradição grave. No frontend, o "Status da Validação do Código" mostrou "Erro de validação: O código fornecido possui erros de sintaxe." Isso indica que a Final Answer do Agente 5 (que é o que a Crew retorna) não está refletindo o resultado da ferramenta, ou o script_python que ele recebeu para validar não era o mesmo que o Agente 4 de fato gerou (devido aos problemas de encadeamento).

Status Atual: Definido em equipe_mestre/agentes.py e sua tarefa em equipe_mestre/tarefas.py.

Pontos Chave na Orquestração (app.py)

Sucesso: O erro AttributeError: 'Task' object has no attribute 'execute' foi corrigido removendo as chamadas .execute() ao passar os outputs das tarefas.

Sucesso Parcial: O erro expected string or bytes-like object, got 'TaskOutput' desapareceu da mensagem de erro principal no frontend, o que é um grande avanço na manipulação de outputs.

Desafio Persistente: A causa raiz dos problemas de exibição e inconsistência de validação ainda reside no fato de que os agentes 1, 2 e 3 não estão produzindo suas Final Answers estritamente no formato esperado (Markdown/JSON puro), o que afeta a cadeia de inputs.

Próximos Passos (Prioridades)

Forçar Saída Estrita dos Agentes 1, 2 e 3:

Ação: Revisar as descrições das tarefas `analisar_requisitos`, `identificar_ferramentas` e `projetar_equipe` em `equipe_mestre/tarefas.py`. Precisamos ser ainda mais imperativos sobre a Final Answer conter apenas o bloco Markdown ou JSON, sem qualquer texto adicional de introdução ou conclusão. Adicionar frases como "SUA RESPOSTA FINAL DEVE COMEÇAR COM markdown` e TERMINAR COM , SEM NADA MAIS." ou "SUA RESPOSTA FINAL DEVE SER APENAS O OBJETO JSON, SEM QUALQUER TEXTO ADICIONAL."

Objetivo: Eliminar completamente as frases genéricas (I now can give a great answer.) e garantir que o `.output` de cada tarefa contenha exatamente o formato esperado.

Corrigir a Chamada da Ferramenta do Agente 4:

Ação: Na tarefa `implementar_equipe_python` em `equipe_mestre/tarefas.py`, garantir que o agente use o nome EXATO da ferramenta (Escritor de Código Python) e que o Action Input seja uma string pura do código, não um dicionário encapsulado.

Guia de Imports para o Engenheiro (Agente 4): Reforçar no prompt da tarefa `implementar_equipe_python` uma seção explícita para o Agente 4 sobre os imports corretos:

Guia de Imports para o Código Python Gerado:

- Para ferramentas da biblioteca `crewai_tools` (ex: `DuckDuckGoSearchTool`, `WebsiteSearchTool`, etc.), importe assim: `from crewai_tools import NomeDaFerramenta1, NomeDaFerramenta2`.
- Certifique-se de que `Agent`, `Task`, `Crew`, `Process` vêm de `crewai`.
- `ChatOpenAI` de `langchain_openai`.
- `load_dotenv` e `os` de seus respectivos módulos.

Objetivo: Permitir que o Agente 4 chame a ferramenta corretamente e gere um script Python com imports válidos.

Investigar Inconsistência do Agente 5 e Exibição do Frontend:

Ação: Uma vez que os outputs dos agentes anteriores estejam limpos, reavaliar a Final Answer do Agente 5 e como ela é exibida. A contradição pode ser resolvida quando o input para o validador estiver sempre limpo e no formato esperado.

Objetivo: Garantir que o status de validação no frontend seja consistente com o log da ferramenta.

Implementar Loop de Feedback (Validador -> Implementador):

Ação: Este será o próximo grande passo após a consistência das saídas ser estabelecida. Isso provavelmente envolverá o uso de `max_retries` e `retry_agent` nas tarefas da CrewAI, ou uma orquestração mais complexa com um Agente Gerente.

Objetivo: Permitir que o Agente 4 receba feedback do Agente 5 e tente corrigir o código automaticamente.