

# Teste Técnico — Engenheiro(a) de IA | Blis AI

## Contexto

A Blis desenvolve agentes de IA para o setor de turismo. Neste teste, você vai construir um mini-sistema de agentes que auxilia agências de viagem a responder dúvidas e sugerir opções para seus clientes.

---

## O que você vai construir

Uma **API REST com arquitetura multi-agent** usando LangGraph, onde dois agentes colaboram para responder perguntas sobre viagens.

## Arquitetura esperada

```
Cliente (HTTP) → FastAPI → Orchestrator Agent
                        |
                        ├── FAQ Agent (RAG)
                        └── Search Agent (Web Search / Tool)
```

**FAQ Agent** — responde perguntas frequentes sobre políticas de bagagem, check-in, documentação para viagem etc., usando RAG sobre uma base de documentos que você mesmo vai popular (pode ser um conjunto de PDFs ou textos que você criar/baixar).

**Search Agent** — usa uma tool de busca (Tavily, SerpAPI ou similar) para buscar informações em tempo real, como preços aproximados, conexões, ou novidades de companhias aéreas.

**Orchestrator** — recebe a pergunta do usuário, decide qual agente (ou ambos) deve responder, e consolida a resposta final.

---

## Requisitos técnicos

### Obrigatórios:

- Python 3.11+ com FastAPI
- LangGraph para orquestração dos agentes

- Pelo menos um endpoint REST: `POST /chat` recebendo `{ "session_id": "...", "message": "..." }`
- Persistência de estado das sessões usando o checkpointer nativo do LangGraph com **Redis** ( `langgraph-checkpoint-redis` ). O `docker-compose` deve subir o Redis junto da aplicação. O uso de `MemorySaver` será aceito apenas como fallback de desenvolvimento, mas não como solução principal
- RAG funcional no FAQ Agent (qualquer vector store: Chroma, FAISS, etc.)
- Web search funcional no Search Agent
- `README.md` claro com instruções para rodar localmente

### Diferenciais avaliados:

- Streaming da resposta via SSE ou WebSocket
- Testes (mesmo que básicos)
- Logging estruturado
- Uso de tipagem forte (Pydantic models)

---

## Requisito obrigatório de processo

**Você deve usar um AI coding agent durante o desenvolvimento** — Claude Code, Cursor, Windsurf ou similar.

No README, dedique uma seção chamada **“Como usei IA no desenvolvimento”** explicando:

- Qual ferramenta usou (ex: Cursor + Claude Sonnet)
- Quais MCPs configurou e para quê (ex: MCP de filesystem, MCP de busca, MCP do próprio projeto)
- Exemplos reais de como a IA te ajudou: geração de código, debug, refactor, análise de logs
- O que funcionou bem e o que você teve que corrigir manualmente

Isso **não é opcional**. Avaliaremos tanto o produto final quanto a maturidade no uso de ferramentas de IA no ciclo de desenvolvimento.

---

# Critérios de avaliação

Critério	Peso
Qualidade e clareza do código Python	Alto
Funcionamento correto dos agentes (RAG + Search)	Alto
Orquestração com LangGraph	Alto
Maturidade no uso de AI coding tools + MCPs	Alto
Documentação e README	Médio
Diferenciais (streaming, testes, logging)	Médio
Histórico de commits (clareza, frequência e evolução do desenvolvimento)	Médio
Tempo total de desenvolvimento (eficiência na entrega)	Médio

## Entrega

- Repositório público no GitHub
- No README: instruções de setup, variáveis de ambiente necessárias, e a seção sobre uso de IA

## Dica de tema para a base RAG

Para não deixar o candidato perdido, sugerimos que o FAQ Agent seja especialista em **políticas de bagagem e documentação de viagem** — algo que qualquer agência de viagens precisa responder diariamente. Você pode usar textos públicos de companhias como LATAM, Gol ou Azul, ou criar documentos fictícios coerentes.

A ideia por trás desse teste é simples: queremos ver como você pensa uma solução, como você usa as ferramentas disponíveis hoje, e se você tem o perfil de quem constrói rápido sem abrir mão de qualidade. Boa sorte!