Capítulo 26 – Hackathon do Projeto Final (Arquitetura e Kickoff)

Aléssio Miranda Jr.

Novembro - 2026/2

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

O que você verá neste capítulo

Bem-vindos à fase culminante da nossa jornada! Durante toda a disciplina, você absorveu conhecimento técnico: Redes Neurais, Transformers, APIs de LLM, RAG e Agentes Autônomos. A partir de agora, as aulas expositivas dão lugar a um ambiente imersivo de "Fábrica de Software". Neste capítulo, daremos o pontapé inicial (*Kickoff*) do Hackathon do **Projeto Integrador Capstone**. Exploraremos como estruturar as bases arquiteturais de um sistema produtivo e a divisão de papéis em uma startup, transformando linhas de código em uma plataforma sólida, com valor real e voltada para o mercado de trabalho.

1 Introdução: A Realidade do Mercado de Trabalho

Durante toda a disciplina de Inteligência Artificial 2, nós navegamos pelas fundações matemáticas das Redes Neurais, presenciamos a revolução semântica dos Embeddings, entendemos a complexidade dos Transformers, consumimos APIs massivas (como OpenAI e Gemini), desenvolvemos bancos de dados vetoriais locais (RAG), injetamos ferramentas no modelo através de Agentes Autônomos e, por fim, aprendemos a refinar modelos gigantes através de Fine-Tuning.

Toda essa bagagem, por si só, é formidável. Contudo, a realidade brutal do mercado de trabalho moderno é pragmática: empresas não contratam enge-

nheiros de software para rodar *scripts* Python isolados no terminal do *Jupyter Notebook*. Uma corporação (ou uma startup) contrata profissionais capazes de orquestrar todas essas camadas tecnológicas complexas em um **Produto Funcional Coeso** que resolva um problema crônico de negócios.

As aulas 26 a 30 representam uma simulação intensiva do ambiente corporativo ágil. É o momento de abandonar os exercícios isolados de laboratório e iniciar a construção de um projeto sustentável, utilizando a metodologia baseada em desafios (Hackathon).

2 A Dinâmica do Hackathon e a Formação de Startups

Nos próximos encontros, a dinâmica tradicional de ensino superior (onde o professor fala e o aluno absorve passivamente) desaparece por completo. O laboratório se transforma em uma incubadora.

Os alunos devem se organizar em duplas ou trios que atuarão formalmente como *Startups Tecnológicas Autônomas*. A partir deste momento, o professor sai do palco expositivo e assume o papel consultivo de **Arquiteto Sênior** (ou *Tech Lead*). As interrupções acontecerão apenas sob demanda: a equipe levanta a mão quando se deparar com gargalos arquiteturais severos que exijam destreza para serem desatados, como:

- Erros críticos de concorrência ou conflitos de bibliotecas.
- Falhas conceituais na tubulação de ingestão RAG (ex: *chunking* destruindo a semântica de tabelas de PDF).
- Decisões de segurança de injeção de dependência na arquitetura Spring Boot.

O objetivo central desta reta final é a idealização e a construção colaborativa de uma plataforma inovadora, desenvolvida de ponta a ponta, projetada para impressionar o corpo docente e engatilhar portfólio profissional para os alunos.

3 O Dia 1: Ideação, Estruturação e Setup

Nesta primeira aula oficial do Hackathon, as equipes não devem (e não precisam) escrever a lógica central de inteligência artificial. O Dia 1 é inteiramente dedicado ao planejamento arquitetural e ao *setup* colaborativo. Erros cometidos na raiz do projeto custarão muito caro no Dia 3.

Tabela 1: Cronograma Simplificado do Hackathon Capstone

Encontro	Foco Principal	Entregável do Dia
Aula 26	Kickoff e Arquitetura	Domínio definido, Repositório Git criado e Divisão de papéis pronta.
Aula 27	Backend e Dados	Pipelines de RAG ingeridos e rotas de API (Cérebro) respondendo via Postman.
Aula 28	Interface de Usuário	Front-end conectado à API sem erros de CORS (O Rosto do sistema).
Aula 29	Nuvem e Deploy	Sistema publicado na internet (ex: Render/Vercel) com URL acessível publicamente.
Aula 30	Demoday	Pitch de 10 minutos para a Banca com "Live Demo" da solução.

• Teoria: title=Os Três Pilares do Kickoff

1. **Ideação de Domínio (Business Case):** Vocês devem definir um problema estreito e real. Não criem "Um assistente que responde qualquer coisa". Criem soluções focadas: "Um assistente que analisa relatórios de radiologia", "Um bot de triagem jurídica trabalhista", ou "Um oráculo de suporte técnico". O valor de um sistema RAG está na sua especialização em um nicho de dados.

2. **A Escolha do Stack Tecnológico:** É imperativo definir a espinha dorsal de infraestrutura.

   - *Back-end:* Python (com FastAPI/LangChain) para times focados em prototipagem rápida, ou Java (com Spring Boot/Spring AI) para times com viés corporativo estruturado?
   - *Bancos de Dados:* O Vector Store rodará localmente na memória (ChromaDB) ou será consumido como um serviço gerenciado na nuvem (Pinecone)?
   - *Front-end:* A interface será desenvolvida rapidamente com Streamlit (Python puro) ou será arquitetada com React/Vue e consumirá APIs REST tradicionais?

3. **Setup de Repositório e Gestão (DevOps 101):** É proibido enviar código por pen-drive. O primeiro passo mecânico é criar o projeto no GitLab, adicionar corretamente o arquivo `.gitignore`, inicializar o arquivo de dependências (`requirements.txt` ou `pom.xml`), e garantir que as chaves de API jamais sejam comitadas no código fonte aberto (uso estrito de `.env`).

▶ Prática: title=O Segredo: Divisão de Responsabilidades

Uma das maiores armadilhas em times iniciantes é a síndrome do "programador duplo": duas pessoas tentando editar o mesmo arquivo de forma síncrona.
A divisão clássica e eficaz:

- **Dev A (Dados e IA):** Responsável pelo scraping, ingestão de PDFs, rotinas de RAG, *chunking*, testes de prompt engineering e geração via LLM.
- **Dev B (Integração):** Responsável por construir as rotas REST (APIs), lidar com autenticação, integrar a interface visual (Front-end) com o Back-end, e garantir que a aplicação não trave (tratamento de erros).

Essa divisão gera *pipelines* paralelos de trabalho e dobra a eficiência da equipe.

Ao final deste primeiro encontro, sua equipe deve ter o repósitorio configurado, as dependências instaladas nos computadores, um diagrama básico de caixas no quadro branco, e a certeza de qual dor vocês irão curar com a IA.

✔ Takeaways

- O Hackathon simula um ambiente de **startup real**: a equipe define um domínio, escolhe o stack tecnológico e divide papéis como Dev A (Dados/IA) e Dev B (Integração).
- A **especialização de domínio** é o diferencial de um RAG: "Assistente que responde qualquer coisa" não tem valor — "Especialista em CLT trabalhista" sim.
- O setup do Dia 1 (Git, dependências, .gitignore, .env) determina o sucesso dos dias seguintes — erros na raiz do projeto custam caro depois.
- Chaves de API **jamais** são comitadas no código. Uso estrito de `.env` + `.gitignore` é obrigatório.
- A divisão de responsabilidades (Dados/IA vs. Integração) gera pipelines paralelos que dobram a eficiência da equipe.

4 Conclusão

O Dia 1 é sobre **planejamento, não código**. Com o domínio definido, o repositório configurado e os papéis divididos, a equipe está pronta para o Dia 2: a construção do **Backend e RAG** — o cérebro da aplicação.

✎ Questões: Autoavaliação

1 **[Direta]**Liste os três entregáveis obrigatórios do Dia 1 do Hackathon e explique por que cada um é pré-requisito para os dias seguintes.

2 **[Direta]**Compare as vantagens de usar Python (FastAPI/LangChain) versus Java (Spring Boot/Spring AI) como stack de backend para o projeto. Que fatores da equipe devem guiar essa escolha?

3 **[Direta]**Explique por que o arquivo `.env` deve estar no `.gitignore` e descreva o fluxo correto para um colega novo da equipe configurar suas próprias chaves localmente.

4 **[Reflexiva]**Uma equipe escolheu o domínio “Assistente que responde qualquer coisa sobre saúde”. Critique essa escolha sob a perspectiva de: qualidade do RAG, viabilidade técnica em 5 dias, e riscos éticos/legais.

5 **[Reflexiva]**A “síndrome do programador duplo” (duas pessoas editando o mesmo arquivo) é um anti-padrão em Hackathons. Proponha uma estratégia de Git branching e divisão de tarefas que elimine esse problema.

Lab. 26: Configuração da Arquitetura Base

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

⋆ Objetivo do Laboratório

Kickoff oficial do Hackathon. O objetivo da atividade prática de hoje é estruturar o projeto de forma profissional no Git, estabelecer o escopo do que será desenvolvido e realizar os primeiros testes de integração de bibliotecas para garantir que a arquitetura escolhida compila e funciona para todos do grupo.

5 Atividade Prática

1. Organização do Time

Forme sua dupla ou trio. Discuta com a equipe a ideia de Produto e levante os requisitos mínimos necessários para a prova de conceito (*MVP - Minimum Viable Product*).

2. Configuração do Repositório

1. Um membro da equipe deve criar o repositório central no GitLab institucional e convidar os outros como *Maintainers/Developers*.
2. Adicione um arquivo `.gitignore` correto para o ecossistema escolhido (Python ou Java), evitando o *commit* de pastas densas como `node_modules`, `venv`, `target` e variáveis de ambiente (`.env`).
3. Crie o arquivo de leitura principal (`README.md`) contendo:
   - Nome do Projeto e Resumo do que a IA fará.
   - Ferramentas escolhidas (LLM, Framework de IA, Linguagem, Banco Vetorial).

3. O Primeiro “Esqueleto”

Independentemente da linguagem escolhida, a equipe deve *commitar* a versão “Hello World” da API para provar que a infraestrutura está montada.

No Python (FastAPI):

```
from fastapi import FastAPI
app = FastAPI()

@app.get(‘‘/’’)
def health_check():
    return {‘‘status’’: ‘‘IA Server Online’’}
```

Ou no Java (Spring Boot):

```
@RestController
public class HealthController {
    @GetMapping(‘‘/’’)
    public String healthCheck() {
        return ‘‘IA Server Online’’;
    }
}
```

4. Entrega e Divisão de Tarefas

Todos os membros do grupo deverão clonar o repositório recém-criado, garantir que os dependentes (`pip install -r requirements.txt` ou `mvn clean install`) finalizem sem erros. Utilizem o quadro de *Issues* ou *Kanban* do próprio repositório para listar as tarefas a serem executadas nos próximos laboratórios (Aulas 27 e 28).

6 Critério de Avaliação

O laboratório 26 será considerado concluído pela constatação do repositório remoto contendo os *commits* iniciais de mais de um integrante da equipe, presença do esqueleto do projeto principal rodando localmente sem erros e o `README.md` bem documentado.

✔ Takeaways

- O repositório está criado, clonado por todos e com o esqueleto “Hello World” rodando — a infraestrutura base está pronta.
- O `.gitignore` e o `.env` estão configurados — nenhuma chave será comitada acidentalmente.
- O `README.md` documenta o escopo, as tecnologias e a divisão de responsabilidades da equipe.
- O quadro de Issues/Kanban lista as tarefas dos próximos 4 dias — planejamento é a alma do Hackathon.

◇ Informação

Gancho para Amanhã: O esqueleto está montado, mas vazio. Amanhã vocês construirão o **cérebro** do produto: ingestão de documentos, VectorStore, pipeline RAG e endpoints REST. O objetivo é que ao final do Lab 27, o backend responda perguntas reais via Postman.