Capítulo 22 – Projeto Prático RAG com Spring (Dia de Trabalho do TP2)

Aléssio Miranda Jr.

Outubro - 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

Esta aula é um **laboratório imersivo de construção**, não uma aula expositiva. As duplas terão os 50 minutos integrais para avançar no **Trabalho Prático 2 (TP2)**: o sistema RAG corporativo com Spring AI. O professor atua como *Tech Lead* consultivo — desbloqueando gargalos arquiteturais, revisando código e tirando dúvidas pontuais. O objetivo é que, ao final desta sessão, as duplas tenham seu sistema RAG **funcionando de ponta a ponta**: ingestão de PDFs → Vector Store → API REST → interface consumindo a API.

1 Introdução: O Dia de Trabalho Intensivo

A Aula 21 formalizou os padrões de resiliência (Retry, Circuit Breaker, Rate Limiting) e definiu os critérios de avaliação do TP2. Esta sessão é a oportunidade prática de integrar **todos** os componentes estudados desde a Aula 15 em um produto coeso:

- Integrando o Vector Store no Spring AI
- Criando os Endpoints de Chat com RAG
- Aplicando padrões de arquitetura limpa (separação de camadas)
- Testando os Endpoints da API com Postman/Insomnia

2 Checklist de Integração

As duplas devem usar este checklist como guia de progresso durante a sessão:

▶ Prática: title=Pipeline de Desenvolvimento do TP2

1. **Ingestão de Dados:** Os PDFs do domínio escolhido foram processados? O `TokenTextSplitter` está configurado com overlap adequado?
2. **Vector Store:** O ChromaDB (ou PGVector) está populado com os chunks? A busca por similaridade retorna resultados coerentes?
3. **API REST:** O endpoint `POST /perguntar` recebe uma pergunta e retorna uma resposta fundamentada nos documentos?
4. **Resiliência:** O `@Retryable` e o fallback estão implementados? O sistema sobrevive a um erro 500 simulado?
5. **Interface:** O front-end (Streamlit ou HTML+JS) consome a API e exibe a resposta formatada?

3 Dicas de Arquitetura Limpa

• Teoria: title=Separação de Responsabilidades

O código do TP2 deve seguir a separação clássica MVC:

- **Controller:** Recebe as requisições HTTP e delega para o Service.
- **Service:** Contém a lógica de negócio (chamada ao ChatClient, montagem do RAG).
- **Configuration:** Configura Beans do Spring (VectorStore, ChatClient, Advisors).

Misturar lógica de RAG dentro do Controller é um anti-padrão que dificulta testes e manutenção.

4 Conclusão

Com o TP2 em estágio avançado de desenvolvimento, as duplas consolidam na prática **todo** o Arco 3: APIs de LLM, Prompt Engineering, RAG, Spring AI, front-end e resiliência. A entrega final do TP2 encerra o Arco de Integração Corporativa e prepara o terreno para o Arco 4: **Agentes Autônomos e Projeto Final**.

✔ Takeaways

- Esta aula é **prática pura**: 50 minutos de desenvolvimento imersivo com suporte do Tech Lead.
- O checklist de integração (Ingestão → Vector Store → API → Resiliência → Interface) guia o progresso das duplas.
- A separação MVC (Controller → Service → Configuration) é obrigatória para código limpo e testável.
- O TP2 é a síntese prática de tudo desde a Aula 15: quem domina o TP2, domina o Arco 3 inteiro.
- A Regra dos 20 Minutos continua valendo: bloqueios técnicos devem ser escalados ao professor imediatamente.

✎ Questões: Autoavaliação

1 **[Direta]** Descreva a separação de responsabilidades MVC aplicada a um projeto Spring AI com RAG. Que código vai no Controller, no Service e na Configuration?

2 **[Direta]** Liste os 5 componentes do checklist de integração do TP2 e explique por que a ordem (Ingestão → Vector Store → API → Resiliência → Interface) é importante.

3 **[Direta]** Qual a vantagem de usar overlap no `TokenTextSplitter`? O que acontece com a qualidade das respostas do RAG se o overlap for zero?

4 **[Reflexiva]** Uma dupla chegou ao final desta aula com o backend funcionando, mas sem front-end. Eles têm 50 minutos na próxima aula (que é a última do TP2). Proponha um plano de ação priorizando o que é essencial para a nota.

5 **[Reflexiva]** Compare a experiência de desenvolver o TP2 (sistema RAG corporativo) com os laboratórios individuais das aulas anteriores. O que a integração de ponta a ponta ensina que os labs isolados não ensinam?

Lab. 22: Dia de Trabalho — Integração do TP2

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

Esta é uma **sessão de desenvolvimento intensivo** (50 minutos). As duplas devem usar este tempo para integrar todos os componentes do TP2: ingestão de PDFs reais no VectorStore, endpoints RAG funcionais, resiliência (`@Retryable`) e front-end consumindo a API. O professor atua como *Tech Lead* consultivo — use o checklist abaixo para guiar o progresso.

5 Checklist de Integração do TP2

Use esta tabela para acompanhar o progresso da dupla. Marque cada item conforme for validando:

#	Componente	Status
1	Ingestão de PDFs: `DocumentReader` + `TokenTextSplitter`	☐
2	VectorStore populado: busca por similaridade retorna resultados	☐
3	Endpoint `POST /api/rag`: recebe pergunta, retorna resposta fundamentada	☐
4	`@Retryable` + `@Recover`: sistema sobrevive a erro simulado	☐
5	Front-end (HTML/Streamlit): consome a API e exibe resposta formatada	☐
6	CORS configurado: front-end se comunica com back-end sem bloqueio	☐
7	`.env` / variáveis de ambiente: chaves não aparecem no Git	☐

6 Roteiro de Priorização

Se o tempo estiver apertado, siga esta ordem de prioridade (do mais ao menos crítico):

Prioridade Alta (sem isso, o TP2 não funciona)

1. **Ingestão + VectorStore:** Sem dados no banco vetorial, não existe RAG. Valide com uma query manual no código.

2 **Endpoint RAG funcional:** Teste via Postman/Insomnia com pelo menos 3 perguntas: uma dentro do domínio, uma fora do domínio, e uma ambígua.

Prioridade Média (diferencia uma nota boa de uma excelente)

1 **Resiliência:** `@Retryable` com fallback amigável (Lab 21). Simule erro com chave inválida.

2 **Front-end conectado:** HTML ou Streamlit consumindo a API. CORS resolvido.

Prioridade Bônus (polimento)

1 Streaming de resposta (SSE) para experiência “tipo ChatGPT”.

2 Exibição das fontes (documentos do VectorStore) na interface.

3 README.md bem documentado com instruções de execução.

7 Dicas Rápidas de Debugging

⚠ Importante: Problemas Mais Comuns no TP2

- **VectorStore vazio:** O pipeline de ingestão precisa rodar **antes** de iniciar o servidor. Use um `@Bean` com `CommandLineRunner` para popular automaticamente no startup.
- **CORS bloqueado:** A anotação `@CrossOrigin` deve estar no Controller, não no Service. Alternativa global: `WebMvcConfigurer.addCorsMappings()`.
- **Chave de API não encontrada:** Verifique se a variável de ambiente está exportada no terminal correto (`export OPENAI_API_KEY=...`).
- **Chunking destrutivo:** Se as respostas são incoerentes, aumente o `overlap` do `TokenTextSplitter` para 10–20% do tamanho do chunk.

8 Entrega

1 Ao final da sessão, faça `git add .` e `git commit -m "TP2: Integração [componente implementado]"`.

2 Envie para o GitLab: `git push origin main`.

3 A **entrega final do TP2** segue o prazo definido na Aula 21. Use as horas restantes fora de sala para polir o projeto.

9 Critério de Avaliação

O Lab 22 não tem entrega unitária separada — ele faz parte do **TP2**. A avaliação será pelo andamento observado pelo professor durante a sessão e pelo estado do repositório no GitLab ao final do prazo.

Conclusão: O que Vocês Estão Construindo

✔ Takeaways

- Esta sessão é **prática pura**: 50 minutos de desenvolvimento imersivo com suporte do Tech Lead (professor).
- O checklist de 7 itens guia o progresso: Ingestão → VectorStore → Endpoint → Resiliência → Front-end → CORS → Segurança.
- A **Regra dos 20 Minutos** vale: se um bug não foi resolvido em 20 min, escale imediatamente ao professor.
- O TP2 é a síntese de tudo desde a Aula 15: quem domina o TP2, domina o Arco III inteiro.

◇ Informação

Gancho para a Parte IV: Com o TP2 entregue, vocês dominam a cadeia completa de IA Generativa Corporativa: API → Prompt Engineering → RAG → Spring AI → Front-end → Resiliência. Na próxima parte do curso, entraremos no universo dos **Agentes Autônomos e Fine-Tuning** — onde o LLM deixa de ser um oráculo passivo para se tornar um ator que planeja e executa ações no mundo real.