MASTER AI ENGINEER

Conviértete en AI Engineer

con Java y Spring AI

12 SEMANAS **30** SECCIONES **160+** LECCIONES **5** PROYECTOS

El único programa en español que te enseña a construir
sistemas de IA production-ready sobre tu stack Java.
Desde los fundamentos de LLMs hasta el despliegue en cloud.

CONVERTIRSE EN AI ENGINEER

https://www.codeja.dev/masters/master-ai-engineer-java-langchain-spring/sales

1 SEMANA 1

Fundamentos de LLMs y entorno de trabajo

◆ SPRING AI, PROVEEDORES Y PRIMER CHATCLIENT

SEC. 1 FUNDAMENTOS DE LLMS Y EL ECOSISTEMA JAVA AI

1.1 Qué es un token? La unidad de medida de los LLMs

1.2 Temperatura, top-p y top-k: controlar la creatividad del modelo

1.3 Context window: el límite que define tu arquitectura

1.4 Spring AI vs LangChain4j: cuándo elegir cada uno

1.5 Crear el proyecto Spring AI desde Spring Initializr

1.6 Tu primer ChatClient: hola mundo con streaming reactivo

SEC. 2 CONECTAR CON OPENAI, ANTHROPIC Y OLLAMA

2.1 Comparativa de proveedores: GPT-4o vs Claude 3.5 vs Llama 3

2.2 Configurar OpenAI en Spring AI con Spring profiles

2.3 Configurar Anthropic Claude en Spring AI

2.4 Ollama local: instalar y conectar modelos open-source

2.5 Abstracción ChatModel: intercambiar proveedores sin cambiar código

2.6 Laboratorio: router que asigna cada tarea al modelo óptimo

2 SEMANA 2

Prompt Engineering y Outputs estructurados

◆ TÉCNICAS AVANZADAS, PARSERS Y MEMORIA DE CONVERSACIÓN

SEC. 3 PROMPT ENGINEERING AVANZADO

3.1 Anatomía de un prompt profesional: system, user y assistant

3.2 Zero-shot vs Few-shot: elegir la técnica correcta

3.3 Chain-of-Thought: hacer que el modelo piense en voz alta

3.4 Prompt injection: el ataque que debes conocer y cómo prevenirlo

3.5 PromptTemplate en Spring AI: parametrizar y versionar prompts

3.6 BeanOutputConverter: de texto libre a objetos Java tipados

3.7 Structured outputs de OpenAI: JSON garantizado sin parseo frágil

3.8 Laboratorio: extractor de datos de facturas con prompts robustos

SEC. 4 GESTIÓN DE CONVERSACIÓN Y MEMORIA DE CONTEXTO

4.1 El problema del contexto: por qué los chats olvidan

4.2 Estrategias de historial: sliding window, summarization y buffer

4.3 InMemoryChatMemory en Spring AI: memoria para desarrollo

4.4 ChatMemory persistente con Redis: memoria entre reinicios

4.5 ChatMemory persistente con JPA: memoria en base de datos relacional

4.6 Memoria semántica: recordar hechos del usuario, no mensajes

4.7 Laboratorio: chatbot de soporte con memoria multi-sesión en Redis

3 SEMANA 3

Multimodalidad y Proyecto Módulo 1

◆ IMÁGENES, AUDIO, DOCUMENTOS Y PRIMERA ENTREGA

SEC. 5 MULTIMODALIDAD: IMÁGENES, AUDIO Y DOCUMENTOS

5.1 Modelos multimodales: qué pueden ver, escuchar y leer

5.2 Enviar imágenes al LLM: base64 y URL en Spring AI

5.3 Leer PDFs con DocumentReader de Spring AI

5.4 Procesar Word, Excel y HTML como documentos

5.5 Transcripción de audio con Whisper API desde Java

5.6 Laboratorio: pipeline de extracción automática de contratos PDF

SEC. 6 PROYECTO MÓDULO 1: API DE ANÁLISIS INTELIGENTE DE FEEDBACK

6.1 Diseño del servicio: arquitectura, endpoints y flujo de datos

6.2 Implementación completa: prompts, parsers, tests y documentación

6.3 Code review guiado: patrones, errores comunes y refactoring

4 SEMANA 4

Tool Calling y Arquitectura de Agentes

◆ @TOOL ANNOTATION, REACT PATTERN Y GUARDRAILS

SEC. 7 TOOL CALLING: DARLE HERRAMIENTAS AL LLM

7.1 El ciclo tool calling: cómo decide el LLM qué herramienta usar

7.2 JSON Schema de herramientas: cómo describir una tool para el LLM

7.3 @Tool annotation en Spring AI: la forma idiomática de definir herramientas

7.4 Tool que consulta una API REST externa con WebClient

7.5 Tool que consulta una base de datos JPA

7.6 Seguridad en tools: contexto de usuario y auditoría

7.7 Laboratorio: asistente empresarial con 5 herramientas reales

SEC. 8 ARQUITECTURA DE AGENTES: REACT Y PLANIFICACIÓN

8.1 ReAct pattern: el bucle Think - Act - Observe

8.2 Plan-and-Execute: planificar antes de actuar para tareas largas

8.3 Guardrails de agentes: max iterations, budget y human-in-the-loop

8.4 Implementar el bucle ReAct manualmente en Spring AI

8.5 Laboratorio: agente de investigación con búsqueda web real

5 SEMANA 5

MCP y Sistemas Multi-Agente

◆ MODEL CONTEXT PROTOCOL, ORQUESTACIÓN Y MENSAJERÍA

SEC. 9 MCP (MODEL CONTEXT PROTOCOL) CON SPRING AI

9.1 Qué es MCP y por qué es el USB de las herramientas de IA?

9.2 Conceptos MCP: Resources, Tools, Prompts y Sampling

9.3 Crear un servidor MCP en Spring Boot con spring-ai-mcp-server

9.4 Cliente MCP: consumir servidores externos desde Spring AI

9.5 Transporte stdio: servidor MCP como proceso hijo

9.6 Laboratorio: servidor MCP que expone un ERP como herramientas IA

SEC. 10 SISTEMAS MULTI-AGENTE

10.1 Cuándo necesitas más de un agente: límites del agente único

10.2 Patrón Supervisor-Worker: orquestador y sub-agentes especializados

10.3 Implementar el patrón Supervisor en Spring AI

10.4 Comunicación asíncrona entre agentes con Spring Events

10.5 Agentes en microservicios: comunicación vía RabbitMQ

10.6 Laboratorio: pipeline multi-agente de generación de informes

6 SEMANA 6

Workflows, Agentes de Datos y Proyecto Módulo 2

◆ STATE MACHINE, VIRTUAL THREADS, TEXT-TO-SQL Y ENTREGA

SEC. 11 WORKFLOWS Y ORQUESTACIÓN DE TAREAS COMPLEJAS

11.1 Workflows deterministas vs agenticos: el tradeoff control-flexibilidad

11.2 Modelar un workflow como Spring State Machine

11.3 Paralelismo con Virtual Threads (Java 21): tareas del agente en paralelo

11.4 Human-in-the-loop: pausar el agente y esperar confirmación

11.5 Laboratorio: workflow de onboarding de empleados con aprobación humana

SEC. 12 AGENTES DE DATOS: SQL GENERADO POR IA

12.1 Text-to-SQL: hacer que el LLM escriba consultas de base de datos

12.2 Implementar Text-to-SQL con Spring JDBC: de pregunta a ResultSet

12.3 Sanitización y validación del SQL generado: nunca ejecutar a ciegas

12.4 Laboratorio: analista de datos conversacional sobre una BD empresarial

SEC. 13 PROYECTO MÓDULO 2: AGENTE DE SOPORTE TÉCNICO AUTÓNOMO

13.1 Diseño del agente: herramientas, flujo de decisión y casos límite

13.2 Implementación completa: agente, tools, guardrails y tests

13.3 Demo y métricas: tasa de resolución, tiempo medio y coste por ticket

7 SEMANA 7

Embeddings y Bases de Datos Vectoriales

◆ SEMÁNTICA, ANN, PGVECTOR Y REDIS STACK

SEC. 14 EMBEDDINGS Y BÚSQUEDA SEMÁNTICA DESDE CERO

14.1 Qué es un embedding? De palabras a vectores de números

14.2 Modelos de embedding: text-embedding-3, nomic-embed y e5-multilingual

14.3 Arquitectura RAG completa: indexación y consulta paso a paso

14.4 EmbeddingModel en Spring AI: OpenAI, Ollama y modelos locales

14.5 Laboratorio: motor de búsqueda semántica vs palabras clave

SEC. 15 BASES DE DATOS VECTORIALES EN PROFUNDIDAD

15.1 ANN: cómo buscar en millones de vectores en milisegundos

15.2 Cuándo usar pgvector, Redis Stack, Weaviate o Chroma

15.3 pgvector con Spring AI: setup, indexado y consultas

15.4 Filtrado por metadata: buscar solo en los documentos correctos

15.5 Redis Stack como vector store: latencia sub-milisegundo

15.6 Laboratorio: benchmark comparativo de vector stores con 100k documentos

8 SEMANA 8

Chunking, Ingestion y RAG Avanzado

◆ BÚSQUEDA HÍBRIDA, RERANKING Y EVALUACIÓN DE CALIDAD

SEC. 16 CHUNKING ESTRATÉGICO Y PIPELINE DE INGESTION

16.1 Por qué el tamaño del chunk define la calidad del RAG

16.2 Estrategias de chunking: fixed-size, recursive, semántico y por secciones

16.3 TokenTextSplitter y custom DocumentTransformer en Spring AI

16.4 Parent Document Retriever: indexar chunks y devolver secciones completas

16.5 Pipeline de ingestion robusto con Spring Batch

16.6 Laboratorio: ingestar 200 manuales técnicos PDF con pipeline paralelo

SEC. 17 RETRIEVAL AVANZADO: BÚSQUEDA HÍBRIDA Y RERANKING

17.1 El problema del RAG naive: cuándo la búsqueda vectorial falla

17.2 Búsqueda híbrida: combinar vectorial y BM25 con Reciprocal Rank Fusion

17.3 Implementar búsqueda híbrida en pgvector con RRF

17.4 Reranking con Cohere Rerank: reordenar resultados con precisión

17.5 Query expansion: generar múltiples variantes de la pregunta

17.6 Laboratorio: comparar RAG naive vs híbrido vs híbrido con rerank

9 SEMANA 9

Agentic RAG y Proyecto Módulo 3

◆ CITACIÓN, EVALUACIÓN RAGAS Y KNOWLEDGE BASE EMPRESARIAL

SEC. 18 GENERACIÓN CON CITACIÓN Y EVALUACIÓN DE CALIDAD RAG

18.1 Alucinaciones en RAG: cuándo el modelo inventa aunque tenga los documentos

18.2 QuestionAnswerAdvisor en Spring AI con citación de fuentes

18.3 Prompts para respuestas fundamentadas: forzar al modelo a citar

18.4 LLM-as-judge: evaluar calidad de respuestas automáticamente

18.5 Laboratorio: suite de evaluación automatizada para tu RAG

SEC. 19 AGENTIC RAG: EL AGENTE QUE DECIDE CUÁNDO BUSCAR

19.1 RAG pasivo vs Agentic RAG: el agente como orquestador del retrieval

19.2 Envolver el vector store como herramienta del agente

19.3 Self-RAG: el modelo decide si necesita buscar o ya sabe la respuesta

19.4 Multi-hop retrieval: responder preguntas que requieren varios pasos

19.5 Laboratorio: asistente legal con RAG sobre normativa RGPD y laboral

SEC. 20 PROYECTO MÓDULO 3: KNOWLEDGE BASE EMPRESARIAL COMPLETA

20.1 Diseño del sistema RAG: fuentes, pipeline y stack tecnológico

20.2 Implementación y evaluación con RAGAS

20.3 Demo, métricas finales y presentación de resultados

10 SEMANA 10

Observabilidad: Métricas, Trazas y LangSmith

◆ MICROMETER, PROMETHEUS, GRAFANA, OPENTELEMETRY Y JAEGER

SEC. 21 MÉTRICAS Y MONITORIZACIÓN CON MICROMETER Y PROMETHEUS

21.1 Métricas específicas de LLM: qué medir que no existe en APIs clásicas

21.2 Métricas automáticas de Spring AI: gen_ai.* out of the box

21.3 Métricas custom con Micrometer: coste real y KPIs de negocio

21.4 Exportar métricas a Prometheus y crear dashboard en Grafana

21.5 Alertas sobre coste y rendimiento en Grafana

21.6 Laboratorio: stack completo Prometheus + Grafana + alertas

SEC. 22 TRAZAS DISTRIBUIDAS CON OPENTELEMETRY Y JAEGER

22.1 Distributed tracing: ver la cadena de una request de principio a fin

22.2 Spring AI + Micrometer Tracing + OpenTelemetry: configuración automática

22.3 Spans custom: trazar cada paso del agente y cada tool call

22.4 Analizar trazas en Jaeger: encontrar el cuello de botella

22.5 Laboratorio: trazar end-to-end una consulta RAG y un agente

SEC. 23 LANGSMITH: DEBUGGING Y TRAZABILIDAD DE PROMPTS

23.1 Qué ofrece LangSmith que Jaeger no tiene

23.2 Integrar LangSmith con Spring AI via OpenTelemetry bridge

23.3 Navegar la UI de LangSmith: filtrar, buscar y comparar runs

23.4 Datasets y experimentos en LangSmith: evaluar cambios de prompt

23.5 Laboratorio: encontrar 3 bugs en un agente roto usando solo LangSmith

11 SEMANA 11

Testing, Seguridad y Costes Enterprise

◆ LLM-AS-JUDGE, OWASP TOP 10, GUARDRAILS Y MULTI-TENANCY

SEC. 24 TESTING DE SISTEMAS DE IA: DE UNITARIO A EVALUACIÓN CONTINUA

24.1 La pirámide de tests para LLMs: qué testear en cada capa

24.2 Tests unitarios con MockChatModel: lógica Java sin llamar al LLM

24.3 Tests de integración con @SpringBootTest y Testcontainers

24.4 Implementar EvaluationService con LLM-as-judge en JUnit 5

24.5 Dataset-driven testing: probar contra casos de prueba en JSON

24.6 Laboratorio: pipeline de evaluación continua en GitHub Actions

SEC. 25 COSTES, SEGURIDAD Y GUARDRAILS ENTERPRISE

25.1 OWASP Top 10 para LLM Applications: los riesgos que debes conocer

25.2 Semantic cache: responder sin llamar al LLM cuando ya tienes la respuesta

25.3 Rate limiting por usuario y tenant con Bucket4j

25.4 Guardrails de entrada y salida: filtrar contenido dañino

25.5 Multi-tenancy en RAG: que el tenant A no vea documentos del tenant B

25.6 Laboratorio: red team de un sistema RAG multi-tenant

12 SEMANA 12

Despliegue, CI/CD y Proyecto Final

◆ DOCKER, KUBERNETES, GRAALVM, LLMOPS Y PORTFOLIO

SEC. 26 CONTAINERIZACIÓN Y DESPLIEGUE CON DOCKER Y KUBERNETES

26.1 Dockerfile multi-stage optimizado para Spring Boot con JRE 21

26.2 Spring Boot Buildpacks: imagen sin Dockerfile con un comando

26.3 Kubernetes: Deployment, Service y ConfigMap para la app Spring AI

26.4 HPA y escalado automático para sistemas LLM

26.5 GraalVM Native Image: arranque en milisegundos para serverless

26.6 Laboratorio: desplegar el sistema RAG completo en Kubernetes local (kind)

SEC. 27 CI/CD Y LLMOPS: EL CICLO DE VIDA DE UN SISTEMA DE IA

27.1 LLMOps vs MLOps: las diferencias que cambian el pipeline

27.2 Pipeline GitHub Actions completo: build - test - eval - deploy

27.3 Versionado de prompts en Git: prompts como código

27.4 Feature flags para A/B testing de prompts en producción

27.5 Laboratorio: pipeline que detecta una regresión de prompt

SEC. 28 RESILIENCIA, STREAMING Y OPTIMIZACIÓN EN PRODUCCIÓN

28.1 Circuit breaker con Resilience4j: cuándo el proveedor LLM falla

28.2 Streaming reactivo end-to-end: del LLM al navegador con SSE

28.3 Model cascading: intentar primero el modelo barato, escalar si falla

28.4 Laboratorio: simular un outage de OpenAI y ver el fallback en acción

13 SEMANA 12 (Cont.)

SEC. 29 PROYECTO FINAL: DISEÑO, IMPLEMENTACIÓN E INTEGRACIÓN

29.1 Definición del proyecto final: AI Engineer Assistant completo

29.2 Diagrama de arquitectura C4: sistema, contenedores y componentes

29.3 ADRs: documentar las decisiones de arquitectura

29.4 Sesión de implementación guiada: integrar todos los módulos

29.5 Validación: suite de tests y métricas de calidad antes del despliegue

SEC. 30 DESPLIEGUE, PORTFOLIO Y SIGUIENTE NIVEL

30.1 Despliegue en cloud: Railway o Render con pipeline automatizado

30.2 README y documentación de portfolio: cómo presentarlo en una entrevista

30.3 El AI Engineer en 2026: fine-tuning, modelos de razonamiento y computer use

30.4 Presentación final: demo en vivo, Q&A técnico y feedback

---

SOBRE EL PROGRAMA

El Master AI Engineer con Java y Spring AI es un programa de 12 semanas para desarrolladores Java con experiencia en Spring Boot que quieren convertirse en AI Engineers sin abandonar su stack. Cubre desde los fundamentos de los LLMs hasta el despliegue en cloud de sistemas de IA completos: agentes autónomos, RAG empresarial, observabilidad y LLMOps.

30 secciones · **160+ lecciones** · **5 proyectos** · Acceso vitalicio · Comunidad privada

CONVERTIRSE EN AI ENGINEER

https://www.codeja.dev/masters/master-ai-engineer-java-langchain-spring/sales