UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

Pedro Ribeiro Fernandes

UFG
UNIVERSIDADE FEDERAL DE GOIÁS

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

PEDRO RIBEIRO FERNANDES

Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): PEDRO RIBEIRO FERNANDES

Título do trabalho: Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Pedro Ribeiro Fernandes**, **Discente**, em 04/02/2026, às 16:46, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:43, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

PEDRO RIBEIRO FERNANDES

Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.
Orientador: Prof. Dr. Fernando Marques Federson

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

FERNANDES, PEDRO RIBEIRO
Large Language Model Operations [manuscrito]: Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção / PEDRO RIBEIRO FERNANDES. - 2025.
93 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Large Language Models. 3. Llmops.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

PEDRO RIBEIRO FERNANDES

Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

PEDRO RIBEIRO FERNANDES

Large Language Model Operations

Framework e Plataforma Didática para Operacionalização de Arquiteturas em Produção

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **LLMOps**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.

Palavras-chave: Inteligência artificial; Large language models; LLMOps.

ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **LLMOps**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.

Keywords: Artificial intelligence; Large language models; LLMOps.

Goiânia

2025

Minha Jornada

Pedro Ribeiro Fernandes

Especialista em: LLMOps

MINHA JORNADA

Nome: **Pedro Ribeiro Fernandes**

Especialidade: **LLMOps**

Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

Minha Jornada

A minha jornada para me especializar em LLMOps (*Large Language Model Operations*) começou com uma percepção pessoal clara: havia uma escassez de materiais basilares e organizados sobre a área. Por isso, nas **Semanas 1 e 2**, dediquei-me a construir essa fundação. Revisei cursos da DeepLearningAI, estudei a distinção entre FMOps (*Foundation Model Operations*) e LLMOps, entendendo que todo LLM é um modelo de fundação, mas nem todo modelo de fundação é um LLM. Estudei artigos, pesquisei em blogs, consumi conteúdo e, por fim, comecei a desenhar um roadmap pessoal e a definir os "Níveis de Organização" da área de LLMOps, tentando mapear como decisões e ações em cada nível podem ser impactantes para o restante do sistema como um todo. Os materiais desenvolvidos estão reunidos no **Apêndice 1**.

Na **Semana 3**, senti a necessidade de aprofundar a teoria e, para isso, utilizei o livro "*LLMOps: Managing Large Language Models in Production*" para entender melhor os casos de uso e o que já existia de conhecimento consolidado por parte das grandes empresas e

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

profissionais. Foi um momento de conectar os pontos e, para isso, melhorei o dicionário de termos técnicos para organizar a "sopa de letrinhas" da área e comecei a ver as relações de causa e efeito no *pipeline* como, por exemplo, como a quantização de um modelo lá na ponta afeta diretamente o custo e a latência da infraestrutura. Os materiais desenvolvidos nesta **Semana** estão reunidos no **Apêndice 2**.

A virada para uma visão mais estrutural (e arquitetural) aconteceu nas **Semanas 4 e 5**. Mudei um pouco o planejamento original para incorporar o estudo de ontologias em função de observações sobre a condução das atividades até este momento. Usei ferramentas como o Protégé para criar uma ontologia visual de LLMOps, definindo classes e hierarquias. Isso me deu clareza para desenvolver um framework estratégico próprio, onde adicionei uma camada de "Automação de Operações" às etapas tradicionais que eu via nos artigos para desenvolvimento de sistemas em LLMOps. Foi nessa fase que diagramei as principais arquiteturas e técnicas (como RAG e Fine-tuning) e defini a *stack* de ferramentas que eu usaria na prática, contendo soluções da GCP, AWS e open-source. Os resultados dos estudos realizados estão reunidos no **Apêndice 3**.

Quando cheguei nas **Semanas 6 e 7**, havia começado o meu maior desafio pessoal, ou seja, de fato colocar a “mão na massa” e fazer um sistema baseado em LLM funcional de ponta a ponta. E para suportar o processo, defini que usaria a API da Brapi, sobre dados financeiros, para dar vida aos meus exemplos. Planejei dois *pipelines* principais: (1) um de RAG para relatórios financeiros e (2) um de Fine-tuning focado em *tool-use*. Como eu não tinha muita experiência com desenvolvimento web, precisei pedir ajuda a colegas e usar o apoio de modelos de IA para estruturar a *Learning LLMOps Platform*, uma plataforma criada por mim para unificação do conhecimento, utilizando React e Tailwind. Os resultados obtidos nesta etapa estão apresentados no **Apêndice 4**.

O trabalho pesado de “hands-on” ocorreu nas **Semanas 8 e 9**. Configurei todo o ambiente na nuvem da Google (GCP), usando Cloud Run e Cloud SQL e no Firebase, para autenticação de usuários da plataforma criada, armazenamento de conteúdos e dados de interações do usuário (como os formulários de maturidade da plataforma). Tive algumas

dificuldades iniciais com o *deploy* (que foram relatadas e explicadas para pessoas que no futuro vão, possivelmente, se deparar com essas dificuldades). Consegui colocar “em produção” um crawler para coletar os dados financeiros. O próximo passo foi realizar o *deploy* e tornar a plataforma acessível publicamente no domínio llmops.com.br - o que foi uma grande conquista pessoal. Mais do que uma página na web, consegui nessas **Semanas** implementar a arquitetura de "API Blackbox" em produção com funcionalidades reais de segurança (guardrails), mascaramento de dados sensíveis e monitoramento de custos. Além disso, também avancei na criação e deploy da Arquitetura Agentes que tinha como objetivo consultar meu banco de dados usando linguagem natural (Text2SQL). As observações, referências e resultados estão reunidos no **Apêndice 5**.

A **Semana 10** foi dedicada para “lapidar” tudo que foi realizado. Meu foco se voltou para unificar o conteúdo, garantindo que a plataforma criada não fosse apenas um repositório de código, mas um guia útil para pesquisadores e profissionais, com testes “em produção” e o *deploy* final de diversos exemplos que construí. Olhando para trás, o Processo foi muito além de aprender a usar ferramentas, me possibilitando entender como arquitetar, desenvolver e operacionalizar sistemas de IA do zero (**Apêndice 6**).

Em função de tudo que foi vivido ao longo dessa jornada, gostaria de deixar registrado que todo esse Processo serviu não só para a minha construção profissional, mas também para mudar minha visão a respeito de mim mesmo e da minha capacidade de mergulhar em um tema e me tornar especialista. Com certeza enxergo que ainda existem muitas coisas dentro da área que eu preciso aprofundar, mas essa jornada me ofereceu o caminho e a mentalidade necessária para que isso!

APÊNDICE 1

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 3 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Revisão de fundamentos de LLMOps em curso da DeepLearningAI
Link para o curso: https://learn.deeplearning.ai/courses/llmops

Estudo aprofundado da história da área de LLMOps (áreas precursoras a tendências futuras)
Link do documento: Cronologia da área - LLMOps

Estudo dos principais termos e terminologias que guiam a área
Link do documento: Termos e terminologias - LLMOps

Compreensão dos níveis de organização da área de LLMOps
Link do documento: Níveis de organização - LLMOps

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Aprofundar o estudo sobre os níveis de organização, trazendo uma explicação mais aprofundada de cada nível e suas interconexões

Estudar um paper que solicitei acesso no ResearchGate (link)

Estudar e explorar ferramentas relevantes à área de LLMOps, seguindo como referência o documento de termos e terminologias

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("gate") de aprovação: 11 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA: requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa segunda Semana eu:

Busquei mais artigos e artigos mais atualizados a respeito de LLMOps, transição e diferenças entre MLOps e LLMOps e fundamentos da área para solidificar os arquivos desenvolvidos na Semana anterior (foi notada uma grande escassez de materiais basilares da área)

Descobri e estudei superficialmente sobre uma nova área que precede LLMOps: FMOps (Foundation Model Operations) e busquei entender como ela se encaixa no "meio" de (entre) MLOps e LLMOps. Entre as principais diferenças estão:

- Foundation Models se diferem de LLMs, pois são os modelos-base, mais genéricos, enquanto LLMs são mais específicos (Ex.: LLaMA (Foundation model), CodeLLaMA (LLM) especializado
- Todo LLM é um FM, mas nem todo FM é um LLM
- LLMOps consiste, em termos simples, criar um pipeline para 'transformar" um Foundation Model (FM) em um LLM especializado em um caso de uso específico

Estudei o artigo LLMOps: Definitions, Framework and Best Practices (link) mencionado no planejamento da Semana anterior e extraí informações importantes a respeito de:

- Camadas de uma aplicação baseada em LLM
- Framework das etapas do processo de LLMOps
- Metodologias para desenvolvimento de um pipeline LLMOps confiável
- Métricas para avaliação de LLMs (dificuldade pessoal)

Utilizei o artigo acima como base para melhorar o documento sobre os níveis de organização da área e enriquecer as interconexão entre os níveis, adicionando pontos como por exemplo:

- Quais implicações o modelo-base (foundation model) tem sobre o pipeline LLMOps
- Qual a importância dos tokens para a governança da empresa que implementa soluções baseadas em LLMs
- Como a esfera tecnológica afeta a produção e a stack de ferramentas no pipeline LLMOps

Defini e discorri sobre três categorias de relações entre os níveis de organização de LLMOps, sendo elas:

- Relações entre níveis fundamentais (base teórica) e níveis de aplicação (implementação prática)
- Relações entre níveis fundamentais e níveis de organizacionais (impacto sistêmico)

- Relações entre níveis de aplicação e níveis organizacionais

Documento atualizado: Níveis de organização - LLMOps

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Iniciar desenvolvimento de um roadmap para se tornar especialista em LLMOps, seguindo o exemplo do Roadmap de Estudos para se tornar um Engenheiro(a) de Dados

Iniciar exploração e planejar leitura fracionada do livro "LLMOps: Managing Large Language Models in Production" (link para o livro) adquirido essa Semana (com prazo de entrega até dia 19/09)

Continuar estudo de papers sobre Responsible LLMOps* e fundamentos de MLOps

Estudar e explorar ferramentas relevantes à área de LLMOps, porém focado na subárea de AgentOps

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

*O paper Responsible LLMOps foi o primeiro suficientemente relevante a citar o termo LLMOps, dando início a área como uma disciplina

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

História e cronologia da área de LLMOps

Introdução

A área de LLMOps (Large Language Model Operations) surgiu como uma evolução natural de práticas anteriores em tecnologia, arquitetura de software e operações, tornando-se essencial diante da popularização e expansão dos LLMs. E nesse documento busquei estabelecer um panorama histórico que vai me guiar nos estudos a respeito do que áreas precursoras e tendências do LLMOps, reforçando a postura de especialista.

Origem e Marcos Históricos

1. Engenharia de Software e Operações (Décadas de 1980-2000)

- Engenharia de Software trouxe práticas de versionamento, testes, modularização e integração contínua, fundamentais para a construção de sistemas confiáveis em escala.
- **DevOps** surgiu no final dos anos 2000 como resposta à necessidade de integração entre desenvolvimento e operações, promovendo automação de deploy, infraestrutura como código e colaboração entre times.

2. O Surgimento e a Consolidação do MLOps (2015-2021)

- MLOps (Machine Learning Operations) apareceu para responder à crescente demanda pelo ciclo de vida completo de modelos de aprendizado de máquina, adaptando e aprimorando práticas de DevOps para necessidades específicas como retraining, monitoramento, rastreabilidade e gestão de dados.
- Ferramentas como Kubeflow, MLflow, TensorFlow Extended, SageMaker, entre outras, estabeleceram padrões para automação de pipelines, deployment e governança de modelos.

3. A Revolução dos Modelos de Linguagem de Grande Porte (2020-2023)

- Com o lançamento de modelos como GPT-3, BERT e PaLM, surgiram desafios tecnicamente inéditos: modelos gigantescos, consumo de recursos massivo, necessidade de orquestração entre agentes automatizados e interações em

linguagem natural.
- Empresas viram a necessidade de expandir o conceito de MLOps para cobrir as peculiaridades dos LLMs, como prompt engineering, observabilidade voltada para modelos generativos, gestão de uso de APIs, e integração de componentes baseados em RAG (Retrieval Augmented Generation).
- ChatGPT (2022) foi um marco público, popularizando o acesso massivo a IA generativa e acelerando a demanda por soluções robustas de produção e governança específica para LLMs.

4. O Surgimento da LLMOps como Disciplina (2022–2025)

- LLMOps se consolidou como disciplina própria, abrangendo:
  - Observabilidade e rastreabilidade de respostas,
  - Otimização de prompts,
  - Controle de pipelines envolvendo agentes inteligentes,
  - Integração com cloud providers (GCP, AWS, Azure).
- Ferramentas como LangChain, LlamaIndex, LangSmith, Helicone, Vellum e Arize AI começaram a compor o ecossistema dedicado de governança e observabilidade.
- Surge com essa terminologia a partir do paper Responsible LLMOps de 2022, que foi o primeiro suficientemente relevante a citar a área com essa terminologia

Correlacionando Áreas: Como Cada Momento Construiu LLMOps

Área Predecessora	Contribuições Principais para LLMOps	Exemplos de Continuidade
Engenharia de Software	Versionamento, modularização, testes, CI/CD	Versionamento de modelos e prompts; testes de prompts
DevOps	Infraestrutura como código (IaC), automação, deploy	Infra-estrutura escalável; automação de pipelines
MLOps	Retraining, ML pipelines, monitoramento, automação	Pipelines de LLM, monitoramento de saídas

AI Research	Modelos fundacionais, técnicas de treinamento massivo	LLMs, fine-tuning, prompt engineering
DataOps	Governança, catálogos e qualidade de dados	Rastreabilidade de contexto nos dados de entrada
Ferramentas Cloud	Elasticidade, escalabilidade, serverless, integração APIs	Orquestração de agentes na nuvem

Tendências Atuais e Futuras

- Agentic LLMOps e Multi-Agents: Espera-se uma explosão de sistemas autônomos compostos por múltiplos agentes interativos, exigindo orquestração mais avançada e novas práticas de rastreabilidade.
- Observabilidade avançada: Foco em detecção automática de deriva comportamental, bias, toxicidade e detecção de falhas em tempo real.
- Governança e Compliance: Crescente preocupação com regulação, explicabilidade, privacidade de dados e ética em produção.[i]
- Automação e Feedback em Tempo Real: Pipelines inteligentes que reagem a dados contextuais, feedback dos usuários e métricas de performance.
- Infraestrutura Híbrida: Orquestração combinada entre clouds públicas, privadas e edge computing, visando escalabilidade massiva e latência mínima.

Conclusão

A área de LLMOps é resultado orgânico da maturação das práticas de engenharia de software, DevOps, MLOps e das demandas inéditas trazidas pelos modelos de linguagem. Hoje, está se consolidando por meio de práticas que ampliam automação, governança, rastreabilidade, segurança e eficiência para modelos cada vez mais complexos e presentes em todas as áreas – especialmente em educação, saúde (área em que eu atuo), finanças e setores criativos.

Referências

- DevOpsSchool: evolução do MLOps
- Coralogix: MLOps Process Maturity Path

- Google Cloud: LLMOps aplicações práticas
- TrueFoundry: Arquitetura de LLMOps
- Cloudera: Adaptação de MLOps para GenAI

Termos e terminologias de LLMOps

Introdução

Esse arquivo tem o objetivo de conhecer, explorar e solidificar o conhecimento a respeito de termos e terminologias que são base para a área de LLMOps. Com isso espera-se obter uma compreensão mais abrangente sobre os pontos que guiam a área e trazem uma visão mais clara a respeito dos processos

Divisão de termos

Com o objetivo de organizar os termos, busquei uma divisão que fizesse sentido, então aqui estão as categorias de termos que eu escolhi, usando como referência, mas não se limitando, aos níveis de organização estabelecidos no decorrer do estudo:

- Fundamentos e Termos Precedentes
- Terminologias Técnicas Fundamentais
- Prompt Engineering
- Pipeline de LLMOps
- Deploy e Infraestrutura Cloud
- Monitoramento e Observabilidade
- Otimização
- Agentes e Aplicações Avançadas

A partir dessa organização, ficou mais fácil entender e buscar estudar de forma mais eficiente os termos mais importantes para a área.

Dicionário dos termos

Foram usadas ferramentas para a busca de termos como Perplexity e NotebookLM, focando sempre em fontes confiáveis, focando sempre nos termos em inglês para me acostumar com o que sempre aparece no mercado a fora.

Agora, estão listados abaixo todos os termos que busquei e pesquisei e que fazem parte do ecossistema de LLMOps, seguindo os níveis de organização e o nível de

prioridade de cada um deles:

1. Fundamentos e Termos Precedentes

Herança das Disciplinas Anteriores

- MLOps (Machine Learning Operations): Base metodológica que LLMOps estende para necessidades específicas de LLMs. Práticas como versionamento de modelos, pipelines de treino e monitoramento contínuo foram adaptadas para lidar com a escala e complexidade únicos dos LLMs.
- DevOps (Development Operations): Cultura operacional de automação e colaboração essencial para LLMOps. Conceitos como CI/CD, infraestrutura como código e monitoramento foram especializados para o contexto de modelos de linguagem.
- Foundation Model: Modelo pré-treinado de grande escala que serve como ponto de partida, evitando treino *from scratch* e reduzindo custos drasticamente. Exemplos incluem GPT-4, LLaMA, Claude e Gemini.

---

2. Terminologias Técnicas Fundamentais

Processamento de Linguagem

- Token: Unidade básica de processamento de texto em LLMs (palavra, subpalavra, caractere). A compreensão da tokenização é fundamental para entender como LLMs processam informação.
- Embedding: Representação vetorial numérica de tokens ou conceitos, formando a base matemática para busca semântica e similaridade.
- Transformer: Arquitetura neural dominante em LLMs modernos, baseada em *attention mechanisms* que permitem processamento eficiente de sequências longas.
- Context Window: Limitação técnica que define a quantidade máxima de tokens que o modelo pode processar simultaneamente, impactando diretamente o design de aplicações.

Operações do Modelo

- Parameters: Valores numéricos aprendidos durante o treinamento que determinam o comportamento do modelo. LLMs modernos possuem bilhões de parâmetros.
- Inference: Processo operacional principal em produção - gerar saídas do

modelo usando dados de entrada via API calls.

---

3. Prompt Engineering

- Prompt: Interface principal entre usuário e modelo - entrada textual fornecida para gerar uma resposta específica.
- Prompt Template: Estrutura pré-definida reutilizável para tarefas específicas, permitindo padronização e otimização de interações.
- Few-shot Learning: Técnica essencial onde poucos exemplos são fornecidos no prompt para guiar o modelo, permitindo personalização sem fine-tuning.
- Chain-of-Thought: Método que inclui raciocínio passo-a-passo nas respostas, melhorando significativamente a performance em tarefas de raciocínio complexo.

---

4. Pipeline de LLMOps (AGEMC, CRISP-DM adaptado)

- Data Preparation: Processo de coleta, limpeza e preparação de dados para treinamento/fine-tuning. A qualidade dos dados determina diretamente a qualidade das saídas do sistema.
- Model Selection: Decisão estratégica de escolher o modelo foundation adequado, impactando performance, custo e latência do sistema final.
- Fine-tuning: Ajuste fino do modelo para tarefas ou domínios específicos, crucial para personalização sem treino *from scratch*.
- Evaluation: Processo de teste e validação que vai além de métricas tradicionais, incluindo avaliação de qualidade pedagógica, *bias* e adequação contextual.
- Deployment: Colocação do modelo em produção, envolvendo desafios únicos de escala, latência e integração.
- Monitoring: Acompanhamento contínuo que inclui detecção de deriva comportamental, *bias* e qualidade das saídas.

---

5. Deploy e Infra em Cloud

Containerização e APIs

- Containerization: Empacotamento do modelo em containers para

portabilidade e consistência entre ambientes de desenvolvimento e produção (principalmente com Docker)
- API Endpoint: Interface padrão que permite acesso ao modelo via HTTP/REST, facilitando integração com aplicações existentes.
- Load Balancing: Distribuição de requisições entre múltiplas instâncias para garantir performance e alta disponibilidade.

Escalabilidade e Recursos

- Auto Scaling: Aumento/diminuição automática de recursos baseado na demanda, essencial para gestão eficiente de custos operacionais.
- GPU Allocation: Designação otimizada de recursos GPU caros e críticos para processar requisições do modelo eficientemente.

---

6. Monitoramento e Observabilidade

Qualidade e Performance

- Observability: Capacidade de entender o comportamento interno e performance do sistema, fundamental para transparência operacional.
- Model Drift: Mudança gradual na performance devido a alterações nos dados de entrada ao longo do tempo.
- Hallucination: Problema crítico específico de LLMs - geração de informações factualmente incorretas mas plausíveis.
- Latency Monitoring: Acompanhamento do tempo de resposta para garantir SLA e experiência adequada do usuário.

---

7. Otimização

Técnicas de Melhoria

- RLHF (Reinforcement Learning from Human Feedback): Treinamento baseado em feedback humano para alinhamento com valores e redução de *bias*.
- LoRA (Low-Rank Adaptation): Técnica eficiente de fine-tuning que modifica apenas uma fração dos parâmetros originais.
- RAG (Retrieval Augmented Generation): Sistema híbrido que enriquece o conhecimento do modelo com busca externa, sem necessidade de retreinamento.

- Vector Database: Infraestrutura especializada para armazenar e buscar embeddings vetoriais, fundamental para sistemas RAG.

---

8. Agentes e Aplicações

Sistemas Agênticos

- AI Agent: Sistema autônomo que usa LLMs para executar tarefas complexas, representando a evolução de modelos passivos para sistemas ativos.
- MCP Server (Model Context Protocol Server): Padrão emergente que fornece ferramentas e interfaces para agentes acessarem sistemas externos.
- Tool Calling: Capacidade do modelo de invocar funções/ferramentas externas, estendendo suas capacidades além da geração de texto.
- Agentic Workflow: Fluxo de trabalho onde múltiplos agentes colaboram para completar tarefas complexas, representando o futuro da automação.

Conclusão

Esse arquivo busca organizar um dicionário da área de LMOps, considerando que eu não preciso dominar todos os termos, porém para ser um especialista eu preciso minimamente conhecer cada um deles e ter noção de como cada um pode ser usado no contexto profissional e acadêmico.

Referências

https://lakefs.io/blog/llmops
https://tulsipatro29.medium.com/llmops-part-1-the-introduction-81fa77999f44
https://arxiv.org/html/2503.15577v1
https://ieeexplore.ieee.org/document/10698359

[Este documento foi substituído posteriormente por uma planilha]

Níveis de organização da área de LLMOps

Introdução e inspiração

Nessa Semana, o meu objetivo foi entender como a área de LLMOps pode ser "quebrada" em outros níveis, buscando entender os principais objetos de estudo que, como especialista, eu precisaria dominar e quais eu precisaria ter um conhecimento minimamente relevante para otimizar minha compreensão geral da área escolhida.

Inspirado pela Biologia e seus níveis de organização, busquei essa Semana explorar e descobrir como poderia quebrar em 12 níveis a minha área de interesse

Nesse sentido, busquei entender qual seria a menor unidade de estudo (plausível) que a área poderia chegar e, pensando em LLMs, logo surgiu a primeira: os tokens.

Níveis de organização

Os tokens são as menores unidades de texto usadas para treinamento, precificação, aplicação de janela de contexto, dentre vários outros aspectos do LLM. Portanto, achei razoável trazer o token como o meu primeiro nível de estudo para a minha organização da "pirâmide do LLMOps".

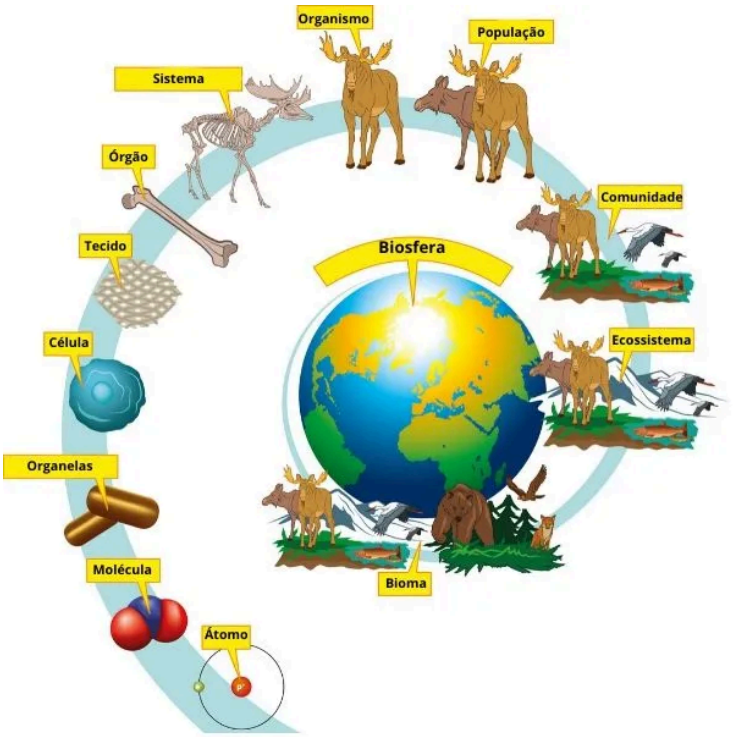


Imagem usada como referência para a minha organização

Mas, pensando sobre o primeiro nível, me propus estabelecer o que seria o meu último nível, considerando o contexto inteiro de uma aplicação (mas sem me prender a ela). E refletindo sobre isso, cheguei à conclusão de que seria cabível trazer a Esfera Tecnológica (na figura do Estado regulatório) como o último nível da minha organização, uma vez que, considerando os aspectos sociais, regulatórios,

éticos e econômicos, é relevante que eu como especialista, tenha conhecimento relevante a respeito dessa área e como ela impacta meu trabalho, mesmo que abrangente, complexa e burocrática.

A partir disso, busquei entender como poderia “subir” do token à Esfera Tecnológica e comecei a estabelecer alguns parâmetros e algumas ideias, chegando a algo com a seguinte estrutura:

- **L1: Token - O "Átomo" dos LLMs:**

  É a unidade fundamental de texto que o modelo processa, servindo como a base para medir o custo, a latência e o tamanho da entrada e saída do sistema.

- **L2: Embedding/Representação - A "Molécula" Semântica:**

  Traduz tokens em vetores numéricos, formando uma representação matemática que captura o significado e as relações semânticas entre as palavras.

- **L3: Camada/Layer - As "Organelas" Neurais:**

  Cada camada da rede neural realiza uma transformação específica nos embeddings, permitindo que o modelo aprenda padrões progressivamente mais complexos nos dados.

- **L4: Modelo Base - A "Célula" Foundation:**

  É a rede neural pré-treinada cujo comportamento e capacidade de generalização são definidos por seus bilhões de parâmetros internos, que podem ser ajustados ou quantizados.

- **L5: Prompt/Interface - O "Tecido" de Interação:**

  Estrutura a interação com o modelo, combinando instruções, exemplos e contexto externo (via RAG e bases vetoriais) para guiar a geração de respostas relevantes.

- **L6: Agente/Aplicação - O "Órgão" Inteligente:**

  Utiliza o modelo como um motor de raciocínio para executar tarefas complexas, tomar decisões e interagir com ferramentas externas de forma autônoma para cumprir um objetivo.

- **L7: Pipeline MLOps - O "Sistema" de Produção:**

  Automatiza o ciclo de vida técnico do modelo, gerenciando desde a preparação de dados e versionamento até a avaliação contínua, o

treinamento e o deploy (CI/CD).

- **L8: Sistema LLMOps - O "Organismo" Completo:**

  Orquestra a operação em tempo real, gerenciando o roteamento entre modelos, o controle de custos, a aplicação de guardrails de segurança e a auditoria do sistema em produção.

- **L9: Infraestrutura - A "População" Tecnológica:**

  Engloba a plataforma de hardware (cloud/on-prem, GPUs), a camada de execução que serve os modelos com autoescalonamento e a camada de dados que gerencia data lakes e bases de embeddings.

- **L10: Organização/Empresa - A "Comunidade" Corporativa:**

  Representa o conjunto de pessoas, processos e estratégias de negócio que direcionam o desenvolvimento, a governança e a aplicação da tecnologia de LLMs para atingir seus objetivos.

- **L11: Ecossistema - A "Biosfera" de IA:**

  Compreende a interação dinâmica entre empresas, comunidades open-source, provedores de modelos e instituições de pesquisa que impulsionam a inovação e a competição no campo.

- **L12: Esfera Tecnológica - O "Planeta" Digital:**

  Define o ambiente global onde a IA opera, sendo moldada por leis, normas técnicas e expectativas sociais que impõem requisitos verificáveis de segurança, privacidade e ética.

OBS.: O uso de LX é para facilitar a visualização e referenciamento aos níveis em outras partes do documento, como por exemplo na área de interconexões.

Classificação dos níveis

Com essa organização, comecei a pensar em uma classificação que poderia me ajudar a dividir ainda mais esses níveis de organização, de forma que eu pudesse organizar melhor meu tempo de estudo, guiado de acordo com essa classificação e com a prioridade que cada parte tem, sendo mais próximo do “organismo” (Sistema LLMOps) mais importante e mais distante, menos relevante. Assim cheguei na classificação a seguir:

- Níveis Fundamentais (1-4): A Base Técnica

Nesse nível, o foco das áreas é reunir os blocos estruturais dos LLMs, como tokens, embeddings, camadas e modelos base. São os elementos que sustentam a

engenharia de modelos antes de qualquer aplicação prática.

- Níveis de Aplicação (5-8): Implementação Prática

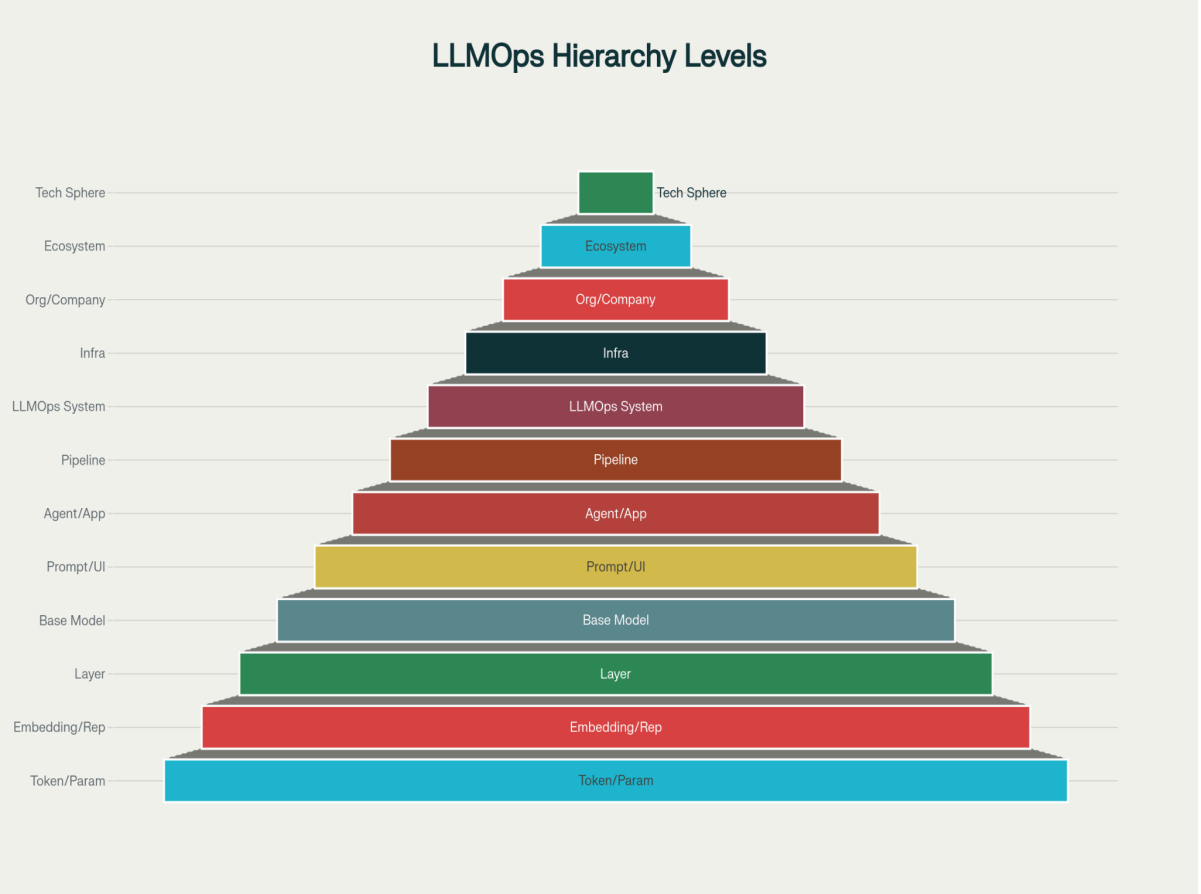
Aqui, os níveis abrangem as formas de interação e utilização dos LLMs, desde prompts até agentes autônomos, pipelines de MLOps e sistemas completos de LLMOps. Aqui a teoria se transforma em prática operacional.

- Níveis Organizacionais (9-12): Impacto Sistêmico

Nesse estágio os níveis refletem como a infraestrutura, as empresas, os ecossistemas e a esfera tecnológica moldam a adoção de LLMOps. Este grupo conecta a operação técnica ao impacto estratégico, regulatório e social.

Visualização dos níveis de organização

Agora, visualmente, assim ficou minha pirâmide dos níveis de organização hierárquico de LLMOps, considerando os níveis apresentados:



Interconexões entre os níveis

Esse tópico foca em apresentar, nível por nível, quais são as relações mais essenciais entre os níveis de organização, ou seja, é feita uma análise top-down e bottom-up a respeito de como cada nível afeta outros (top-down) e como ele é afetado pelos outros (bottom-up). Além disso, foram adicionados também campos para análise de métricas e riscos iminentes para cada um dos níveis, considerando as interconexões apresentadas.

L1. Token — “o átomo”

- Afeta: L5 Prompt (custo/latência por token, *truncation*), L2 Embeddings (qualidade da tokenização → semântica), L6 Agente (janelas de contexto úteis ou não).
- É afetado por: L4 Modelo (tokenizer), L12 Esfera Tec. (regras sobre PII/redação de logs), L10 Organização (políticas de retenção).
- Métricas/controles: custo por 1k tokens, *context utilization*, taxa de truncamento; políticas de mascaramento/redação.
- Riscos: custo descontrolado, vazamento de PII em logs, perda de signal por má tokenização em PT-BR.

L2. Embedding/Representação — “a molécula semântica”

- Afeta: L5 Prompt (seleção de contexto), L6 Agente (recall em RAG), L7 Pipeline (jobs de indexação, *drift* semântico), L9 Infra (custo de armazenamento/consulta).
- É afetado por: L1 Token (tokenizer), L4 Modelo (dimensão, *encoder*), L12 (ex.: restrições sobre dados pessoais).
- Métricas/controles: *recall/precision@k*, *MMR diversity*, latência de busca, *freshness* do índice, *embedding drift*.
- Riscos: *retrieval* pobre → alucinação; custo alto de *vector store*; *privacy leakage* em índices.

L3. Camada/Layer — “organelas”

- Afeta: L4 Modelo (profundidade/arquitetura), L7 Pipeline (treino/serving mais caros), L9 Infra (memória, paralelismo).
- É afetado por: L12 (padrões/limites energéticos/privacidade → preferir distilação/quantização), L10 (budget/SLAs).
- Métricas/controles: FLOPs, *throughput*, *activation sparsity*, quantização, *kv-cache* hit rate.
- Riscos: latência/sobrecusto; inviabilizar *on-prem/edge*; degradar qualidade ao quantizar.

L4. Modelo Base — “a célula”

- Afeta: L5 Prompt (capacidade de seguir instruções), L6 Agente (funcall/tool-use), L7 Pipeline (avaliação, *fine-tuning*, *routing*), L9 Infra (tipos de acelerador).
- É afetado por: L3 (arquitetura), L12 (licenças, conformidade), L10 (*fine-tune* vs *prompt-engineering*).
- Métricas/controles: *task metrics* (ex.: exact match, faithfulness), *toxicity*/*safety*, custo/req. energéticos.
- Riscos: *model risk* (viés, segurança), *vendor lock-in*, incompatibilidade com jurisdições.

L5. Prompt/Interface — "tecido de interação"

- Afeta: L6 Agente (orquestra a ação), L7 Pipeline (*prompt lineage*/versionamento), L8 Sistema LLMOps (roteamento e políticas), L1 (custo).
- É afetado por: L2 (RAG/memória), L4 (capacidades/formatos), L10 (políticas de discurso, *brand voice*), L12 (requisitos de disclosure).
- Métricas/controles: *prompt success rate*, *self-consistency*, custo/latência por fluxo, *guardrail violations*.
- Riscos: *prompt injection*/*jailbreak*, fuga de segredos, respostas fora de tom/marca.

L6. Agente/Aplicação — "o órgão inteligente"

- Afeta: L7 (telemetria, *feedback loops*), L8 (governança runtime), L10 (processos e produtividade), L9 (padrões de uso).
- É afetado por: L5 (protocolos estruturados), L4 (func-calling/toolformer), L2 (recuperação), L12/L10 (limites de autonomia).
- Métricas/controles: *task completion*, *tool-use success*, taxa de escalonamento humano, *unsafe action rate*.
- Riscos: ações incorretas (finanças/saúde/jurídico), *over-automation*, fragilidade a dados externos.

L7. Pipeline MLOps — "o sistema de produção"

- Afeta: L8 (confiabilidade do ecossistema), L9 (demanda por compute/armazenamento), L10 (capex/opex), L4 (ciclo de versões).
- É afetado por: L6 (telemetria/feedback), L5 (mudanças em prompts), L12 (requisitos de avaliação/auditoria), L10 (governança).
- Métricas/controles: tempo de ciclo (ideia→prod), *evals* contínuas, *drift*, *rollback MTTR*, cobertura de testes.
- Riscos: mudanças sem controle → regressões; falta de *evals*; auditoria fraca.

L8. Sistema LLMOps — "o organismo"

- Afeta: L5–L7 (políticas, roteamento, *rate-limits*), L9 (alocação de recursos), L10 (visibilidade de custos/risco).

- É afetado por: L12 (normas/padrões), L10 (apetite a risco e metas de negócio).
- Métricas/controles: SLOs (qualidade, latência, custo), *policy enforcement*, observabilidade ponta-a-ponta.
- Riscos: *policy gaps*, ausência de trilhas de auditoria, custos imprevisíveis.

L9. Infraestrutura — "a população tecnológica"

- Afeta: L4 (viabilidade de modelos), L7 (janelas de lote, escalar/auto-scale), L6 (latência para o usuário).
- É afetado por: L8 (orquestração), L10 (budget/segurança), L12 (residência de dados).
- Métricas/controles: utilização de GPU/CPU, *tail latency (p99)*, custo por chamada, *egress*, disponibilidade.
- Riscos: *under/over-provisioning*, custos explosivos, *single region* → risco regulatório.

L10. Organização/Empresa — "a comunidade"

- Afeta: L4 (seleção de modelo e licenças), L8 (políticas de uso), L9 (estratégia de plataforma), L5–L7 (prioridades).
- É afetado por: L11 (concorrência/parcerias), L12 (leis/regulamentos), resultados de L6 (impacto no negócio).
- Métricas/controles: ROI, NPS/CSAT, *value at risk*, compliance, capacitação de times.
- Riscos: descompasso estratégia↔capabilidade; *shadow AI*; risco reputacional.

L11. Ecossistema — "a biosfera"

- Afeta: L4 (oferta de modelos/ferramentas), L9 (provedores), L8 (padrões de fato), L10 (talentos e *benchmarks*).
- É afetado por: L12 (marcos regulatórios), dinâmica de mercado e pesquisa.
- Métricas/controles: diversidade de fornecedores, maturidade de frameworks, interoperabilidade.
- Riscos: *vendor lock-in* sistêmico, obsolescência rápida, dependência de *APIs*.

L12. Esfera Tecnológica (regulatória/social) — "o planeta"

- Afeta (top-down): todos os níveis, de L10 (governança) a L1 (tokenização que preserve privacidade/logs), sobretudo L8 (políticas), L7 (auditorias/evals), L9 (soberania de dados).
- É afetado por: L11 (pressão do mercado/associações), incidentes públicos (L6) que realimentam normas.
- Métricas/controles: conformidade (auditorias), *risk register*, tempo de adequação a novos requisitos.

- Riscos: multas, bloqueios de operação, perda de confiança pública.

Fluxos de causa-e-efeito

Essa seção tem como objetivo apresentar de maneira prática e didática relação de causa-efeito que ocorrem entre os níveis quando algo “muda” ou acontece em algum nível, trazendo as consequências dessa mudança ou acontecimento para níveis interconectados.

Para facilitar a compreensão dos exemplos, serão apresentados dois exemplos, sendo um top-down (de um nível superior na pirâmide para um nível inferior) e outro bottom-up (de um nível inferior da pirâmide para um nível superior)

Exemplo top-down
Nova regra de privacidade sancionada pelo Governo:
Nova exigência (L12) proíbe retenção de logs com PII → a **organização (L10)** define política de redação → o sistema LLMOps (L8) aplica filters e PII-redaction na borda → **pipelines (L7)** passam a avaliar leakage → **agentes (L6)** evitam ecoar PII → **prompts (L5)** exigem schemas mais restritivos → **revisões no tokenizer/logging (L1)** e no armazenamento de **embeddings (L2)**.

Exemplo bottom-up
Otimizar custo sem perder qualidade com quantização de LLM:
Quantizar (L3) o **modelo base (L4)** de 16-bit→8-bit reduz latência e custo na **infra (L9)** → permite ampliar **janelas de contexto (L1)** mantendo **SLO (L8)** → com **prompts melhores (L5)** + **RAG calibrado** (L2), o agente (L6) mantém task completion. Checklist: medir **evals antes/depois (L7)** e verificar **conformidade (L12)**.

A partir desses exemplos, é possível visualizar de forma prática como a compreensão dos níveis de organização e suas interconexões são essenciais para que um especialista em LLMOps saiba lidar com mudanças e propor otimizações para cada situação dentro de um pipeline completo.

Referências

- https://www.biologianet.com/ecologia/niveis-de-organizacao-em-biologia.html
- https://www.databricks.com/glossary/llmops
- https://www.researchgate.net/publication/384745667_LLMOps_Definitions_Framework_and_Best_Practices

APÊNDICE 2

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 18 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa terceira Semana eu:

Fiz adições no documento dos níveis de organização, incluindo pontos que corroboram o entendimento a respeito da área de LLMOps de uma forma mais interconectada e holística. As adições incluem:

- Relações e conexões entre os níveis mais aprofundadas (além de mais relações)
    - Exemplo: L1. **Token** — "o átomo"
        - **Afeta**: L5 Prompt (custo/latência por token, truncation), L2 Embeddings (qualidade da tokenização → semântica), L6 Agente (janelas de contexto úteis ou não).
        - **É afetado por**: L4 Modelo (tokenizer), L12 Esfera Tec. (regras sobre PII/redação de logs), L10 Organização (políticas de retenção).
- Análise bottom-up e top-down sobre relações de causa-efeito entre os níveis
    - Exemplo: A **quantização (L3)** do **modelo base (L4)** reduz a latência e o custo da infraestrutura (L9), permitindo ampliar as **janelas de contexto (L1)** e manter o **SLO (L8)**. Com **prompts melhores (L5)** e **RAG (L2)** calibrado, **o agente (L6)** mantém a conclusão da tarefa. É importante medir as **avaliações (L7)** antes e depois, e verificar a **conformidade (L12)**.
- Tabelas e matrizes para visualização de relações entre níveis

Enriqueci e melhorei o dicionário de termos e terminologias que um especialista em LLMOps precisa ter conhecimento, adicionando tudo em uma planilha organizada, com informações como:

- Tipo de entidade (se o termo é uma categoria, um estágio do pipeline ou um termo simples)
- Descrição resumida do termo para entendimento básico
- Importância do termo para a área de LLMOps
- Relações entre o termo apresentado e outros termos do dicionário
- Relevância do termo em relação à área de LLMOps (o critério utilizado foi a influência direta do termo na execução do pipeline de LLMOps
- Fase do pipeline de LLMOps a qual o termo pertence (seguindo o framework do artigo inicial com 5 etapas, sendo elas: 1. Requirement Definition; 2. Data Preprocesing; 3. Model Engineering (Training); 4. Model Engineering (Inference); 5. Monitoring/Observability)

Link para a planilha com o dicionário: Dicionário LLMOps

Adicionei os primeiros pontos do Roadmap, os Fundamentos:
- Conceitos base dela IA Generativa
- Linguagem comum de LLMOps
- Linux/Cloud/DevOps
- Arquiteturas de LLMs

Planejei a leitura do livro, focando na parte de use-cases, estabelecendo a leitura dos capítulos 2 e 3:
- Cap.2. Introduction to LLMOps (revisar conceitos e enriquecer materiais anteriores)
- Cap.3. LLM-Based Applications (buscar aplicações, ferramentas e trilhas a serem seguidas)

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Construir uma stack de desenvolvimento do pipeline de LLMOps com as ferramentas mais adequadas ao meu conhecimento para cada etapa apresentada no framework do paper

Continuar o desenvolvimento do roadmap, agora adicionando a seção de Ferramentas, Plataformas e definir as trilhas que podem ser seguidas

Iniciar estudo das arquiteturas das principais plataformas cloud para o pipeline de LLMOps (GCP, AWS, Azue)

Estudar artigo BPMN LLMOps e artigo do Sávio para começar a estudar arquiteturas

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Roadmap de um especialista em LLMOps

Objetivo do documento

Esse documento tem como objetivo apresentar de forma direta, didática e textual, um roadmap dos fundamentos necessários para um profissional que quer se tornar especialista na área de LLMOps

1. Engenharia de Dados

Esta seção foca na preparação e gestão da informação antes que ela chegue aos modelos.

- Pré-processamento de texto: Técnicas de limpeza como remoção de stopwords, lemmatização e *stemming*.
- Pipelines de dados:
  - Processos de ETL & ELT.
  - Arquiteturas de dados: Medallion, Lambda e Kappa.
  - Desafios principais: Data Drift (mudança no padrão dos dados) e Dados enviesados.
- Vector databases (Bancos de dados vetoriais): Cruciais para LLMs, envolvem *Chunking* (quebra de texto), Vetorização e Indexação.

2. Arquiteturas de LLM

Define as estruturas básicas de redes neurais usadas em modelos de linguagem:

- Encoder-only (apenas codificador, ex: BERT).
- Decoder-only (apenas decodificador, ex: GPT).
- Encoder-decoder (ambos, ex: T5).

3. IA Generativa

Cobre os conceitos teóricos fundamentais de como a IA gera conteúdo:

- Tokenização: Como o texto é quebrado em unidades menores.
- Janela de contexto: O limite de informação que o modelo processa de uma vez.
- Mecanismo de atenção: O "coração" dos Transformers.
- Embeddings: Representação numérica de palavras/conceitos.

- Leituras base: Referências teóricas.

4. Machine Learning

Foca na avaliação e categorização dos modelos:

- Tipos de modelos: Diferenciação entre LLMs open-source, open-weight e closed-source (proprietários).
- Métricas de avaliação: Acurácia, Recall, F1-Score e Precisão.
- Tipos de aprendizado: Supervisionado, Não-supervisionado e por Reforço.

5. Linux/Cloud/DevOps

Trata da infraestrutura e operacionalização (LLMOps):

- Git: Versionamento não só de código, mas também de modelos e prompts.
- Docker: Conteinerização de modelos, orquestração e *Load Balancing*.
- Custos em nuvem: FinOps específico para LLMs (gestão de custos de computação).
- Pipelines CI/CD: Automação com ferramentas como Jenkins, GitLab CI e Github Actions.

[Esse documento é uma representação textual do diagrama visual, disponível no link a seguir: https://whimsical.com/roadmap-llmops-Vq3Ts6Zp21Gjt3Qfkdd1FP

APÊNDICE 3

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 25 de set. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Recapitulando, nas três primeiras Semanas foi desenvolvido um trabalho com foco em construir e melhorar materiais que serviriam como base de estudo para a área de LLMOps, trazendo uma abordagem base com os níveis de organização, um dicionário de termos relevantes para a área e um breve estudo cronológico da área, que está emergindo agora.

Nessa quarta Semana, mudando um pouco o planejamento, estudei a respeito de ontologias e busquei aplicar esse conhecimento de forma visual, principalmente para cumprir parte do objetivo do meu processo, que é produzir um conteúdo que seja relevante para quem quiser se tornar um especialista em LLMOps.

Durante o estudo de ontologia, conheci e utilizei conceitos e ferramentas como Entidades, Classes, Instâncias e Protégé

Para conter todos os estudos e desenvolvimentos em relação a ontologias em LLMOps, foi criado um documento que reuniu, dentre outras coisas:

- Classes da ontologia (no meu caso os níveis de organização)
- Relações entre as classes
- Atributos de cada classe
- Hierarquias intra classes (entre indivíduos da classe, como um tipo de agente ou prompt)
- Métricas e riscos por classe (ou nível)

Link para o documento: Ontologia - LLMOps

Como parte do processo de afunilamento do conhecimento, usei essa Semana também para me aprofundar em pontos mais específicos, focando nos níveis de organização mais relevantes ao meu tema, que seriam os níveis de implementação, sendo eles: L6 - Agente; L7 - Pipeline MLOps; L8 - Sistema LLMOps; L9 - Infraestrutura, baseando-me nos artigos citados no planejamento da Semana anterior

Por isso, busquei estudar e me aprofundar nas principais arquiteturas de sistemas LLM-Based, para que eu pudesse desenvolver um material didático para visualização e implementação de diferentes arquiteturas, baseando-se em critérios e aspectos mais práticos e realistas. Diante disso, encontrei na

literatura as principais arquiteturas atuais para pipelines de LLMOps, incluindo, mas não se limitando a:

- Arquiteturas clássicas MLOps (pipeline, microsserviços e híbrida) (Fonte)
- Arquiteturas consolidadas de LLMOps (API-Blackbox, RAG, Fine-tuning)
- Arquiteturas avançadas de LLMOps (Model-routing, LLM-streaming, Data sovereignty)

Nesse esforço ainda, busquei trazer uma visão mais abrangente sobre as arquiteturas, oferecendo juntamente com a explicação de cada, uma visão de:

- Quando usar
- Exemplo de caso de uso
- Métricas principais da arquitetura
- Principais riscos
- Níveis organizacionais mais críticos para cada uma.

Link para o documento: Arquiteturas - LLMOps

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Aprofundar o estudo das arquiteturas e diagramar cada uma da forma mais simples, porém completa possível, para facilitar o entendimento de cada uma

Construir uma stack de desenvolvimento do pipeline de LLMOps com as ferramentas mais adequadas ao meu conhecimento para cada etapa das arquiteturas

Continuar o desenvolvimento do roadmap, agora adicionando a seção de Ferramentas, Plataformas e definir as trilhas que podem ser seguidas

Começar o planejamento de desenvolvimento de exemplos-guia para as arquiteturas principais, ou seja, planejar protótipos de aplicações das arquiteturas com base em um dataset comum

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

Como o meu planejamento mudou devido à recomendação do professor Cedric para o estudo de ontologias e recomendações do professor Sávio, prorroguei o planejamento da Semana passada para construir a stack de desenvolvimento e o desenvolvimento das seções de Ferramentas, Plataformas e Trilhas do roadmap para essa Semana.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Ontologia de LLMOps

Esse é o Documento da Ontologia de LLMOps, um material pré-modelagem feito para ser didático, visual e baseado em exemplos.
Ele se ancora nos 12 níveis de organização da área de LLMOps (L1–L12), com descrições, interconexões, métricas e riscos já apresentados nesse material (dos níveis de organização), que é usado aqui como espinha dorsal.

Objetivo do documento

Entregar um plano claro (com tabelas, listas e exemplos) de como será a ontologia de LLMOps:

- O que ela cobre (entidades), como organiza (classes = 12 níveis), como liga as coisas (propriedades), o que medir (métricas), o que vigiar (riscos) e exemplos concretos (indivíduos) — tudo em linguagem simples.
- Serve como roteiro de construção (o próximo passo é transformar isso em OWL/SKOS/SHACL).

---

1) Visão geral do que a ontologia vai cobrir (inventário de entidades)

Pense em "peças do jogo" (entidades) e "setas" (relações) que compõem LLMOps de ponta a ponta.

Categoria (agrupamento)	Descrição simples	Exemplos de indivíduos (nomes de exemplo)	Nível principal
Token / Tokenizador	Unidade básica de texto / como o texto vira token	`Token("paciente")`, `Tokenizer_BPE_PTBR_v3`	L1
Embedding / Índice Vetorial	Vetor semântico e seus índices	`Embedding_PT_768_OpenText_v1`, `Index_Clientes_2025Q1`	L2

Camada (Layer)	Blocos internos do modelo (attention, MLP, MoE etc.)	`Layer_Attn_23`, `Layer_MoE_7`	L3
Modelo base (Foundation)	O modelo em si (pré-treinado/fine-tuned/quantizado)	`Modelo_Llama3_8B`, `Modelo_XYZ_13B_Q8`	L4
Prompt / Interface	Como falamos com o modelo (instrução, exemplos, RAG)	`Prompt_SAC_Amigável_v2`, `Prompt_Extracao_JSON_Fatura_v1`, `Prompt_RAG_Academico_v3`	L5
Agente / Aplicação	Lógica que usa o modelo + ferramentas para cumprir objetivos	`Agente_Suporte_EduVest`, `Agente_Roteirizador_LogX`	L6
Pipeline (MLOps)	Processos de treino, avaliação, deploy e rollback	`Pipeline_Treino_2025Q1`, `Pipeline_Evals_Contínuas`	L7
Sistema LLMOps	Orquestração em produção (roteamento, políticas, custo)	`Plataforma_LLMOps_Ask4Me`	L8
Infraestrutura	Cloud/on-prem, aceleradores, rede, data lake, vector store	`Cluster_GPU_BR-1`, `VectorStore_X`	L9
Organização/Empresa	Pessoas, processos, objetivos, políticas internas	`EduVest (fictícia)`, `Banco Áureo (fictício)`	L10
Ecossistema	Comunidades, fornecedores, hubs, parceiros	`Hub_OpenSource_X`, `Fornecedor_API_Y`	L11

Esfera Tecnológica/Regulatória	Leis, normas técnicas, expectativas sociais	`LGPD_BR`, `AI_Act_UE`, `Política_Interna_Dados_v1`	L12
Métrica	O que medimos (qualidade, custo, latência, segurança)	`Latência_p95_ms`, `Custo_1k_tokens`, `Faithfulness`	Transversal
Risco	O que pode dar errado	`PII_Leakage`, `Prompt_Injection`, `Bias`	Transversal
Política/Controle	Regras a seguir (lei/padrão/política)	`Controle_Redação_Logs`, `Política_Autonomia_Agentes`	Transversal

Obs. As descrições dos 12 níveis e interconexões estão baseadas no seu documento de referência (L1→L12), que detalha “o que cada nível afeta” e “por quem é afetado”, além de métricas e riscos por nível.

---

2) Classes da ontologia (os 12 níveis, explicados em linguagem simples)

Como lembrar: do micro (token) ao macro (leis e sociedade).

Classe (nível)	“Para que serve?” (1 frase)	Exemplos de indivíduos
L1 Token	Medir custo, latência e tamanho das entradas/saídas.	`Token("contrato")`, `Tokenizer_BPE_PTBR_v3`.
L2 Embedding	Representar significado e buscar contexto relevante (RAG).	`Embedding_PT_768_OpenText_v1`, `Index_Clientes_2025Q1`.
L3 Camada	Definir a “arquitetura interna” que viabiliza o modelo.	`Layer_Attn_23`, `Layer_MoE_7`.

L4 Modelo Base	Ser o “motor” de linguagem e raciocínio.	`Modelo_Llama3_8B`, `Modelo_XYZ_13B_Q8`.
L5 Prompt/Interface	Guiar o modelo com instruções, exemplos e esquema de saída.	`Prompt_SAC_Amigável_v2`, `Prompt_Extração_JSON_Fatura_v1`.
L6 Agente/Aplicação	Executar tarefas com o modelo e ferramentas (objetivo/fluxo).	`Agente_Suporte_EduVest`, `Agente_Roteirizador_LogX`.
L7 Pipeline MLOps	Automatizar treino, avaliação, deploy, rollback.	`Pipeline_Treino_2025Q1`, `Pipeline_Evals_Contínuas`.
L8 Sistema LLMOps	Orquestrar tudo em produção (roteamento, custo, políticas).	`Plataforma_LLMOps_Ask4Me`.
L9 Infraestrutura	Disponibilizar compute, rede, armazenamento, vector stores.	`Cluster_GPU_BR-1`, `VectorStore_X`.
L10 Organização	Dar direção (metas, governança, políticas internas).	`EduVest (fictícia)`, `Banco Áureo (fictício)`.
L11 Ecossistema	Conectar com fornecedores, open source e pesquisa.	`Hub_OpenSource_X`, `Fornecedor_API_Y`.
L12 Esfera Tecnológica	Estabelecer leis/normas e expectativas sociais.	`LGPD_BR`, `AI_Act_UE`, `Política_Interna_Dados_v1`.

---

3) Relações (as “setas” entre as peças)

Principais relações e como aplicar:

Relação (verbo)	De → Para	Quando usar	Exemplo simples

afeta / é_afetado_por	Qualquer nível → outro nível	Para mapear causa-e-efeito entre níveis.	`L12 Esfera Tec. afeta L8 Sistema (novas regras)`; `L1 Token afeta L5 Prompt (custo/latência)`.
usa	Agente/Prompt/Pipeline/Sistema → Modelo/Embedding/Ferramenta	Para declarar dependência operacional.	`Agente usa Modelo_Llama3_8B`.
produz	Pipeline/Sistema/Agente → Métrica/Relatório/Artefato	Para ligar execução a resultados.	`Pipeline produz Métrica Latência_p95_ms`.
temMétrica	Qualquer classe → Métrica	Para anexar o que será medido.	`Prompt temMétrica Custo_1k_tokens`.
temRisco	Qualquer classe → Risco	Para gestão de risco por ativo.	`Agente temRisco Prompt_Injection`.
dependeDe	Um ativo → Outro	Para dependências estáticas (build/deploy).	`Pipeline dependeDe VectorStore_X`.
éParteDe	Item → Conjunto/Sistema	Para composição hierárquica.	`Prompt_SAC_Amigável_v2 éParteDe Agente_Suporte`.
implementadoEm	Agente/Sistema/Modelo → Infra	Onde roda.	`Sistema implementadoEm Cluster_GPU_BR-1`.
treinadoCom	Modelo → Dataset	Vínculo de treino/fine-tune.	`Modelo_XYZ_13B treinadoCom Dataset_Contratos_Anon`.

indexa	Embedding/Índice → Dataset/Documento	Índices vetoriais e suas fontes.	`Index_Clientes_2025Q1 indexa Dataset_Clientes`.
consultaDe	Prompt/Agente → Fonte de contexto	Para RAG e memórias.	`Prompt_RAG consultaDe Index_Clientes`.
governadoPor	Ativo → Política/Lei	Para compliance e auditoria.	`Logs governadoPor Política_Interna_Dados_v`

---

4) Atributos (campos que descrevem cada coisa)

Regra prática: campos curtos, inequívocos, que ajudem a responder quem/onde/quando/quanto/com-o-quê.

Classe	Atributos-exemplo (tipo)	Exemplo de valor
L1 Token	`custo_por_1M_tokens (número)`, `janela_tokens (número)`, `truncation_rate (%)`	`R$ 0,003`, `128k`, `2,1%`
L2 Embedding	`dimensao (número)`, `modelo_embedding (texto)`, `recall_k (número)`, `latencia_busca_ms (número)`	`768`, `OpenText-v1`, `0,63@k=20`, `38`
L3 Camada	`tipo (texto)`, `bits (número)`, `parametros (número)`	`Self-Attention`, `8`, `42e6`
L4 Modelo	`nome (texto)`, `versao (texto)`, `contexto_max_tokens (número)`, `licenca (texto)`	`Llama3`, `8B`, `8192`, `open`
L5 Prompt	`formato_saida (texto)`, `tam_tokens (número)`, `usa_rag (bool)`	`JSON schema`, `850`, `true`

L6 Agente	`grau_autonomia (enum)`, `ferramentas (lista de tools)`, `escopo (texto)`	`média`, `[CRM, NavegadorSeguro]`, `Suporte nível 1`
L7 Pipeline	`tipo (enum)`, `agendamento (texto)`, `tempo_ciclo_min (número)`	`eval contínua`, `diário 01:00`, `45`
L8 Sistema	`slo_latencia_ms (número)`, `slo_custo (número)`, `politica_roteamento (texto)`	`150`, `R$0,02/call`, `Exploração 10%`
L9 Infra	`provedor (texto)`, `regiao (texto)`, `acelerador (texto)`, `custo_hora (número)`	`CloudX`, `BR-1`, `A100`, `R$12,50`
L10 Organização	`setor (texto)`, `apetite_risco (enum)`, `política_privacidade (texto)`	`Educação`, `médio`, `v1-2025`
L11 Ecossistema	`tipo_parceiro (enum)`, `dependencia (texto)`	`fornecedor`, `API embeddings`
L12 Esfera Tec.	`jurisdicao (texto)`, `obrigatoriedade (enum)`, `vigencia (data)`	`UE`, `alta`, `2025-06-01`
Métrica	`categoria (enum)`, `formula (texto)`, `periodicidade (texto)`	`latência`, `p95`, `por release`
Risco	`severidade (enum)`, `probabilidade (enum)`, `tratamento (texto)`	`alta`, `média`, `redação/PII`
Política/Controle	`tipo (enum)`, `nível_aplicação (texto)`, `responsável (texto)`	`lei`, `logs`, `Compliance`

5) Exemplos de indivíduos (prontos para exercícios e testes)

Já vêm "etiquetados" pelo nível e ajudam a montar demos e consultas.

Prompts (L5)

- `Prompt_SAC_Amigável_v2` — objetivo: atendimento ao aluno; saída JSON com campos `[resumo, próxima_ação]`.
- `Prompt_Extracao_JSON_Fatura_v1` — extrai valores de fatura (chaves exigidas).
- `Prompt_RAG_Academico_v3` — consulta teses/artigos (RAG) e retorna referências.

Agentes (L6)
- `Agente_Suporte_EduVest` — ferramentas: `CRM`, `NavegadorSeguro`, `Calendário`.
- `Agente_Roteirizador_LogX` — ferramentas: `CalcFreteAPI`, `Maps`, `ERP`.

Modelos (L4)
- `Modelo_Llama3_8B` (open) • `Modelo_XYZ_13B_Q8` (quantizado).

Embeddings/Índices (L2)
- `Embedding_PT_768_OpenText_v1` • `Index_Clientes_2025Q1` (fonte: `Dataset_Clientes`).

Pipelines (L7)
- `Pipeline_Evals_Contínuas` (avalia `faithfulness`, `toxicity`, `latência_p95`).
- `Pipeline_Treino_2025Q1` (gera `Modelo_XYZ_13B_Q8` e relatório).

Sistema LLMOps (L8)
- `Plataforma_LLMOps_Ask4Me` — roteia por custo/latência, aplica guardrails, registra auditoria.

Infra (L9)
- `Cluster_GPU_BR-1` (A100), `VectorStore_X` (região BR-1).

Organizações (L10) *(fictícias)*
- `EduVest` (edtech) • `Banco Áureo` (financeiro) • `PorkLog` (logística suinícola).

Esfera Tecnológica/Políticas (L12/transversal)
- `LGPD_BR` • `AI_Act_UE` • `Política_Interna_Dados_v1` (redação de logs, PII).

Métricas
- `Latência_p95_ms` • `Custo_1k_tokens` • `Faithfulness` • `ToxicityScore`.

Riscos
- `PII_Leakage` • `Prompt_Injection` • `Bias` • `Vendor_LockIn`.

---

6) Hierarquias entre indivíduos (árvores simples)

Árvores “do geral para o específico” — úteis para ensino e navegação.

Prompt
- Prompt de Sistema (regras/voz)
- Prompt de Tarefa (instrução principal)
- Prompt com Exemplos (few-shot)
- Prompt com Ferramentas (function/tool use)
- Prompt com RAG (consulta fontes)

Agente
- Conversacional (SAC, FAQ)
- Executor de Tarefas (extração, preencher CRM)
- Orquestrador (multiagentes/múltiplas ferramentas)
- Supervisor (revisão humana na linha)

Métrica
- Qualidade (exact match, faithfulness)
- Segurança (toxicity, jailbreak success)
- Performance (latência média, p95)
- Custo (R$/1M tokens, GPU-h)

Risco
- Privacidade (PII leakage)
- Segurança (prompt injection, exfiltração)
- Justiça/Sesgo (bias)
- Operacional (custo imprevisível, disponibilidade)

---

7) Métricas e Riscos por nível (resumo rápido)

Extraído da análise de interconexões L1–L12

Nível	Métricas (exemplos)	Riscos (exemplos)
L1 Token	custo/1M tokens, utilização de contexto, truncation	custo descontrolado, PII em logs
L2 Embedding	recall/precision@k, latência de busca, frescor do índice	alucinação por recall baixo, vazamento em índices

L3 Camada	FLOPs, throughput, sparsity, kv-cache hit	latência, degradação por quantização
L4 Modelo	qualidade de tarefa, segurança, custo/energia	viés/segurança, lock-in
L5 Prompt	taxa de sucesso, consistência, custo por fluxo	injection/jailbreak, fuga de segredos e PIIs
L6 Agente	task completion, tool-use success, escalonamento humano	ação insegura/indevida, over-automation
L7 Pipeline	tempo de ciclo, cobertura de evals, MTTR	regressões sem controle, auditoria fraca
L8 Sistema	SLOs (latência, custo), policy enforcement	lacunas de política, custos imprevisíveis
L9 Infra	utilização GPU/CPU, p99, custo/call, TTFT (Time-To-First-Token)	over/under-provisioning, risco regulatório
L10 Org	ROI, NPS/CSAT, compliance	shadow AI, risco reputacional
L11 Ecossistema	diversidade fornecedores, interoperabilidade	lock-in sistêmico, obsolescência
L12 Esfera Tec.	conformidade, tempo de adequação	multas, bloqueios, perda de confiança

8) Dois fluxos de causa-e-efeito (para mostrar “as setas”)

Úteis para explicar por que relações importam na ontologia.

Top-down (lei nova → tudo muda):
L12 *(nova regra de privacidade)* → L10 *(política interna de redação)* → L8 *(enforcement)* → L7 *(evals de leakage)* → L6 *(agente evita ecoar PII)* → L5 *(prompts com schemas mais restritos)* → L1/L2 *(token/log & índice segura)*.

Bottom-up (otimização técnica → efeito em cadeia):

Quantização em L3 → Modelo (L4) mais rápido → Infra (L9) mais barata → SLO (L8) mantido → Prompts (L5) e RAG (L2) ajustados → Agente (L6) mantém task completion; medir antes/depois (L7) e checar conformidade (L12).

---

9) Tabelas de “pronto para modelar” (resumo para construir)

9.1 Mínimos de documentação por classe

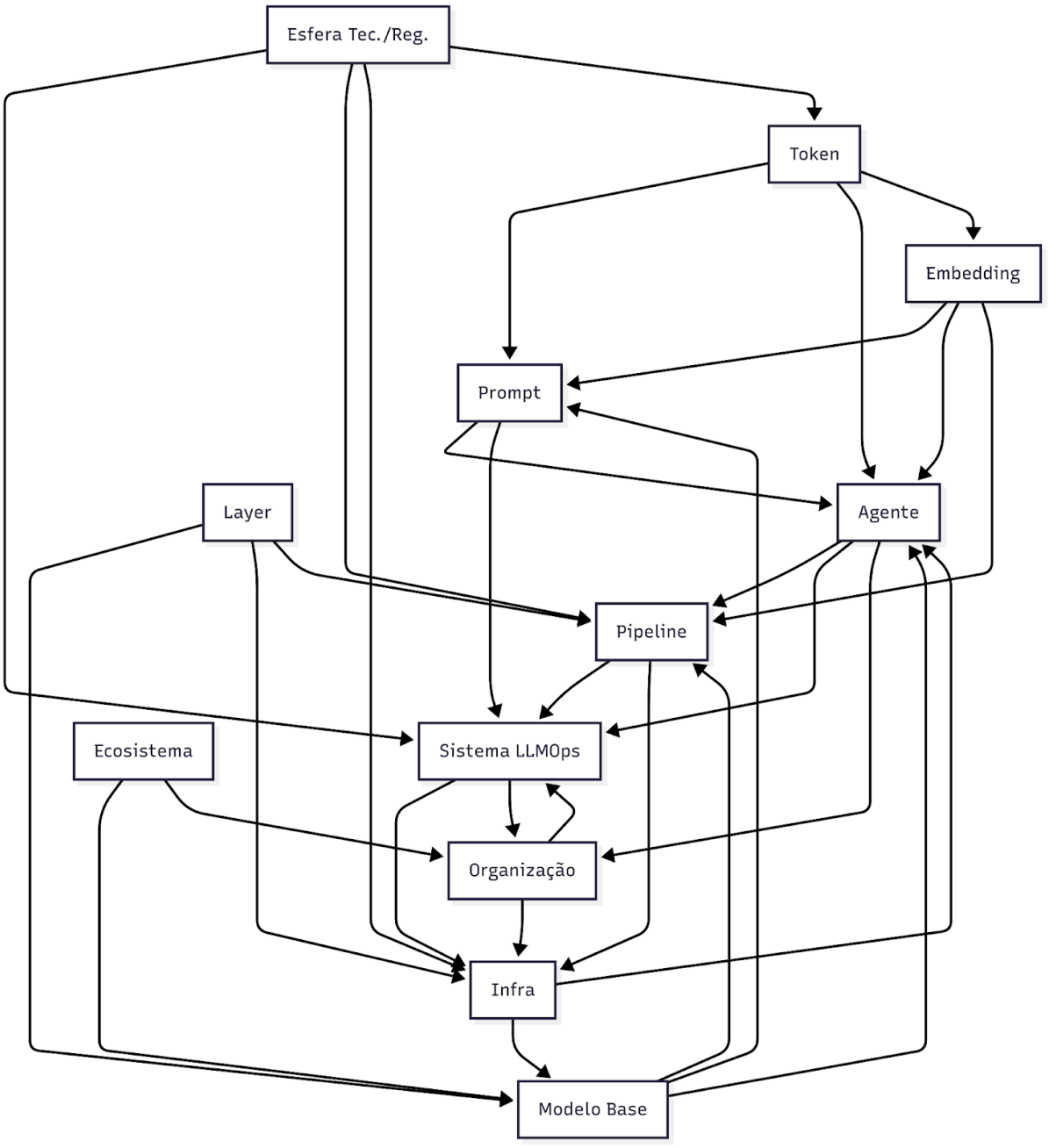
Classe	Campos mínimos recomendados
L1–L4	nome, versão, fornecedor/licença (se aplicável), indicadores de custo/latência
L5	objetivo, formato de saída, exemplos, dependências (RAG/ferramentas), riscos
L6	objetivo de negócio, ferramentas, limites de autonomia, métricas de sucesso
L7	tipo, gatilhos, artefatos produzidos, rollback, avaliação
L8	políticas ativas, roteamento, observabilidade, custo orçado/real
L9	provedor, região, acelerador, custo/hora, disponibilidade
L10	setor, objetivos, políticas internas, responsáveis
L11	parceiros, dependências externas, SLAs
L12	normas aplicáveis, status, responsáveis internos

9.2 “Sete relações” que resolvem 80% dos casos

`afeta`, `usa`, `produz`, `temMétrica`, `temRisco`, `dependeDe`, `governadoPor`.

---

10) Visual prático



Essa imagem mostra visualmente uma diagramação de todos os níveis de organização que envolvem LLMOps e quais são as interações principais entre cada um deles.

Conclusão

Com este documento, você tem todas as peças: o que existe (entidades), como agrupar (classes), como conectar (relações), como descrever (atributos), o que medir (métricas) e o que temer (riscos) — além de indivíduos-exemplo para treino e ensino.
O próximo passo é só materializar (abrir os arquivos seed no Protégé e ir preenchendo com seus exemplos). Esse roteiro se apoia integralmente na sua pirâmide de LLMOps (L1–L12) e nos fluxos de interconexão já definidos.

Guia de Arquiteturas de LLMOps

Sumário

Objetivo do documento

O objetivo desse documento é apresentar de maneira simples e didática as principais arquiteturas usadas para a construção de pipelines em LLMOps, quando usar cada uma delas (com exemplos), quais são os principais componentes, riscos e métricas e cada um e, por fim, como implementar cada uma a partir de visualização clara dos passos e ferramentas necessárias para isso.

Este documento consolida um panorama sobre arquiteturas de *pipelines* para LLMOps, conectando decisões de arquitetura a objetivos de negócio e aos Níveis de Organização de LLMOps (L1–L12). Partimos de padrões consolidados em MLOps (pipeline, microsserviços e híbrido) e os traduzimos para o universo de LLMs (RAG, *prompt engineering*, *fine-tuning*, agentes e governança), estruturando um guia de escolha e um checklist de perguntas que facilitam a convergência entre time estratégico e time técnico. O documento se ancora em dois eixos referenciais:

- Arquiteturas de MLOps e seus trade-offs (pipeline, microsserviços, híbrido; automação, CI/CD, monitoramento, contêineres e orquestração). Ver o diagrama geral do ciclo de ML e a tabela de prós/cons nas *páginas 5–6* do artigo-base.
- Processo BPMN para LLMOps com duas raias (estratégica e operacional), *handoffs* claros e infraestrutura tratada como *black box*; também a representação em laço infinito do ciclo LLMOps (descoberta→desenvolvimento→entrega→operar→avaliar). Ver Figura 3 (p.6) e Figura 4 (p.8) do artigo-base.

Como fio condutor, usamos o documento “Níveis de Organização – LLMOps (L1–L12)”, que estabelece interconexões top-down e bottom-up (ex.: impacto de regulações em logs/PII e efeito de quantização na infra/SLOs) e que aqui vinculamos diretamente às decisões de arquitetura.

---

1) Níveis de organização (L1–L12): o que mudam na arquitetura

A seguir, uma leitura rápida de como cada nível influencia escolhas de arquitetura do pipeline. Onde relevante, indicamos exemplos de implicações diretas em componentes LLMOps.

- L1 Token → Define custo/latência por contexto; pressiona *prompt design* e políticas de truncamento; exige *logging* com *redaction* de PII.
- L2 Embeddings → Viabiliza RAG (qualidade de *retrieval*, *freshness*, *drift* semântico); decide *vector store*, *recall@k* e *reranking*.
- L3 Camada/Layer → Arquitetura/quantização/kv-cache ditam viabilidade de *serving* (on-prem vs cloud), e custos.
- L4 Modelo base → Capacidade do modelo (instrução, *function calling*) impulsiona a necessidade de RAG e *prompt engineering* e determina ciclo de *fine-tuning* e avaliação.
- L5 Prompt/Interface → Exige *prompt lineage*, guardrails, avaliação de *jailbreak*; influencia versão e canary de *templates*.
- L6 Agente/Aplicação → Define necessidade de *tool use*, orquestração e telemetria; impacta *SLOs* de completude de tarefa.
- L7 Pipeline MLOps → Automatização (CI/CD, *model registry*, *evals* contínuas) e *rollbacks*; maior governança e auditoria. (Ver automações de dados/treino/deploy/monitoramento nas *páginas 6–8* do artigo de MLOps.)
- L8 Sistema LLMOps → Roteamento por custo/qualidade, *rate-limits*, políticas; observabilidade ponta-a-ponta.
- L9 Infraestrutura → A plataforma escolhida (cloud/on-prem/híbrida) determina que etapas do pipeline integram melhor entre si (ex.: *managed vector DB* + *serverless*

inference vs *GPU on-prem* com *K8s*). (O papel de contêineres/Kubernetes e a integração com ferramentas está sintetizado no artigo de MLOps.)

- L10 Organização → Define padrões, orçamento (CapEx/OpEx), risco e metas; orienta políticas L5–L8.
- L11 Ecossistema → Oferta de modelos/fornecedores e padrões *de facto* (influencia L4, L8–L9).
- L12 Esfera Tecnológica/Regulatória → Requisitos de privacidade/segurança/ética *top-down* afetam todos os níveis e principalmente L8 (políticas), L7 (avaliações/auditorias) e L9 (soberania de dados).

---

2) Do MLOps ao LLMOps: padrões, o que permanece e o que muda

- Padrões de MLOps — As arquiteturas pipeline, microsserviços e híbridas permanecem válidas em LLMOps, com os mesmos *trade-offs*: pipeline (simples de entender, porém menos flexível), microsserviços (flexível e escalável, porém mais complexo de operar), híbrida (equilíbrio pragmático, mantendo facilidade com flexibilidade).

- Automação, CI/CD e *Observability* — As recomendações clássicas (ETL/Airflow, CI/CD, contêineres/Kubernetes, *model registry*, *monitoring*, *drift detection*) são cruciais para LLMOps — *principalmente* porque RAG, *prompts* e *guardrails* mudam frequentemente.

- Tradução para LLMOps — O artigo BPMN [2] propõe raias estratégica e operacional, com *handoffs* claros, infraestrutura como *black box* (para manter o modelo geral aplicável) e ciclo em laço infinito (descobrir→desenhar→desenvolver→avaliar→operar→reavaliar).

---

3) Arquiteturas de referência (LLMOps) — quando usar, componentes, riscos

Parte 1: Arquiteturas Fundamentais de MLOps

a) Arquitetura em Pipeline

- Quando usar: Ideal para fluxos lineares e didáticos, especialmente para times iniciando ou com baixa necessidade de reordenar estágios.
- Exemplo real: Uma linha de forecasting com estágios sequenciais: preparação de dados → treino de modelo → deploy em produção.
- Níveis chave: L7 (ciclo, evals, rollback), L9 (janelas de batch), L10 (governança).
- Métricas prioritárias: Tempo de ciclo (ideia → produção), cobertura de testes/avaliações contínuas, detecção de drift, MTTR (Tempo Médio para Reparo) de rollback.

b) Arquitetura de Microserviços

- Quando usar: Quando flexibilidade, escalabilidade e equipes independentes são críticas. Cada etapa do ciclo de vida é um serviço autônomo.
- Exemplo real: Serviços separados para coleta de dados, treino de modelo e forecasting, orquestrados via mensageria ou filas.
- Níveis chave: L7 (telemetria por serviço), L8 (roteamento/políticas), L9 (auto-scale, latência p99).
- Métricas prioritárias: SLOs por serviço (latência p95/p99), custo por chamada, disponibilidade, taxa de erro inter-serviços.

c) Arquitetura Híbrida

- Quando usar: Para combinar a simplicidade de um pipeline para etapas principais com a flexibilidade de microserviços para partes especializadas (ex: feature store).
- Exemplo real: Pipeline principal de treino e deploy, integrado a um microserviço de feature store e outro para serving de baixa latência.
- Níveis chave: L7 (automação ponta-a-ponta), L8 (políticas), L9 (containerização/orquestração).
- Métricas prioritárias: Throughput do pipeline, latência p99 do serviço de serving, estabilidade dos deploys.

Parte 2: Arquiteturas Específicas de LLMOps

Estes são os padrões mais comuns e específicos para o desenvolvimento de aplicações com LLMs.

a) API Black-Box / Prompt-Only

- Quando usar: Para pilotos, assistentes de baixo risco e tarefas como sumarização, tradução e geração de texto, onde o foco é o *time-to-value* e o *prompting* é o mecanismo central.
- Exemplo real: Um chatbot no estilo ChatGPT para responder a perguntas do atendimento interno.

- Componentes: Templates de prompt (L5), políticas de uso e segurança (L8), avaliações leves de qualidade (L7) e a gestão da API do provedor (L9).
- Prós e Contras: Baixo esforço de implementação vs. alto *vendor lock-in* e dificuldade na governança de custos por token.
- Métricas prioritárias: Taxa de sucesso do prompt, violações de *guardrails*, custo por 1k tokens, latência p95/p99 e *inter-token latency*.
- Níveis críticos: L5, L8, L9, L12 (políticas de uso/dados pessoais).

b) Retrieval-Augmented Generation (RAG)

- Quando usar: Para "aterrar" as respostas do LLM em uma base de conhecimento proprietária (documentos, políticas internas), reduzindo alucinações e garantindo conformidade sem a necessidade de fine-tuning.
- Exemplo real: Um sistema de Q&A sobre políticas internas que usa busca vetorial para encontrar trechos relevantes e os injeta no prompt.
- Componentes: Geração de *embeddings* (L2), banco de dados vetorial (Vector DB), *retriever* (recuperador), *reranker* (reclassificador), prompts estruturados (L5) e avaliações de *retrieval* e *faithfulness* (fidelidade) (L7).
- Prós e Contras: Redução significativa de alucinações vs. exigência de operação e manutenção de índices de vetores (garantindo *freshness*).
- Métricas prioritárias: Recall/Precision@k, diversidade (MMR), latência de busca, *freshness* do índice, *embedding drift* e taxa de alucinação.
- Níveis críticos: L2, L5, L7, L8, L9.

c) RAG+ (Avançado ou agêntico)

- Quando usar: Para tarefas que exigem raciocínio em múltiplos passos, processamento de documentos longos ou o uso de ferramentas externas.
- Componentes extras: Um sistema de *planner* → *solver*, memória de curto e longo prazo para o agente e integração com *tools* (ex: busca na web, consulta a CRM, execução de SQL).
- Trade-offs: Maior latência e custo operacional; exige observabilidade detalhada (L8) e telemetria do agente (L6).
- Níveis críticos: L2, L6, L8.

d) Fine-Tuning / SLMs de Domínio

- Quando usar: Para especializar modelos em domínios fechados, adaptar o estilo de escrita à marca, ou criar `function-calling` robusto. Requer acesso aos pesos do modelo ou o uso de modelos open-source.
- Exemplo real: O projeto Alpaca, que demonstrou o fine-tuning econômico do LLaMA para seguir instruções.
- Componentes: Pipeline de dados e rotulagem (L7), *trainer* (treinador), registro de modelos, deploy em *canary* e *gates* de segurança.

- Riscos: Vazamento de dados (*data leakage*), sobreajuste (*overfitting*) e alto custo de *serving* do modelo (L9).
- Métricas prioritárias: Métricas de tarefa (*exact match*, *faithfulness*), toxicidade/segurança, custo/energia por *epoch*, *throughput* e tempo para treinar.
- Níveis críticos: L4, L7, L8, L9, L10 (licenças), L12 (compliance).

e) Agentes Orquestrados

- Quando usar: Para automações complexas e copilotos que interagem com múltiplos sistemas, compondo tarefas dinamicamente com LLMs e ferramentas externas (ex: LangChain).
- Exemplo real: Um agente no estilo AutoGPT que planeja uma viagem, decompondo a tarefa em subtarefas (buscar voos, reservar hotel) e utilizando as ferramentas adequadas.
- Componentes: Orquestrador (L6), protocolos de comunicação estruturados (JSON/SCHEMA) (L5), adaptadores de ferramentas (*tool adapters*) e políticas de governança em tempo de execução (L8).
- Riscos: Execução de ações incorretas e automação excessiva (*over-automation*), exigindo *guardrails* fortes e um mecanismo de *human-in-the-loop*.
- Métricas prioritárias: Taxa de conclusão da tarefa, sucesso no uso de ferramentas, taxa de escalonamento para humanos e *unsafe action rate*.
- Níveis críticos: L5, L6, L8, L12 (limites de autonomia).

Parte 3: Arquiteturas Avançadas e Especializadas

a) Model Routing

- Quando usar: Para balancear custo, qualidade e latência, roteando a requisição para diferentes LLMs (ex: um modelo pequeno e barato para tarefas simples, um grande e caro para tarefas complexas) com base em políticas.
- Componentes: Roteador* (L8), políticas de *fallback* e *canary*, e telemetria para testes A/B.
- Prós e Contras: Ótima relação custo/qualidade vs. aumento da complexidade operacional.
- Níveis críticos: L7, L8, L9.

* Existem ferramentas Open-source que realizam esse roteamento, como o RouteLLM, que usa Paretto para selecionar o modelo ideal

b) LLMs Embutidos em SaaS/Plataformas

- Quando usar: Para aproveitar recursos de IA generativa nativos em plataformas como Salesforce, SAP ou ServiceNow, acelerando o desenvolvimento.

- Exemplo real: Usar recursos como Einstein (Salesforce) ou Now Assist (ServiceNow) para produtividade.
- Atenção: É crucial analisar a titularidade dos dados, cláusulas de responsabilidade e governança.
- Métricas prioritárias: ROI, NPS/CSAT, taxa de adoção, incidentes de compliance.
- Níveis críticos: L8, L9, L10 (governança/risco).

c) Privacidade & Soberania (On-prem/Edge)

- Quando usar: Quando há dados sensíveis, requisitos regulatórios ou necessidade de residência de dados, exigindo uma implantação local (*on-premise*) ou na borda (*edge*).
- Componentes: Orquestração de contêineres (Kubernetes), servidores de inferência, gerenciamento de segredos e *proxy* para anonimização de dados (PII) (L8).
- Riscos: Alto custo de capital (CapEx), complexidade de infraestrutura e necessidade de uma plataforma MLOps madura (CI/CD, monitoramento).
- Níveis críticos: L7, L8, L9, L12.

d) Tempo Real / Streaming

- Quando usar: Em aplicações como chatbots com resposta em streaming, assistentes de voz ou copilotos integrados em aplicações que exigem baixa latência.
- Componentes: Tecnologias como *server-push*, decodificação parcial (*partial decode*), compartilhamento de KV-cache, filas de processamento e controle de fluxo (*backpressure*).
- Riscos: Latência de cauda longa (*tail latency* p99) e esgotamento de recursos (*resource thrashing*).
- Níveis críticos: L3, L4, L8, L9.

Parte 4: Métricas Essenciais por Área ("Lanes")

- Estratégia (L10): ROI, NPS/CSAT, *value at risk*, prontidão para compliance.
- Dados (L2): Recall/Precision@k, latência de busca, *freshness* do índice, *embedding drift*.
- LLM (L4/L5): Métricas de tarefa (*exact match/faithfulness*), taxa de sucesso do prompt, violações de *guardrails*.
- Dev/CI-CD (L7): Tempo de ciclo (ideia → produção), cobertura de testes/avaliações, MTTR.
- Ops/Produção (L8/L9): SLOs de qualidade, latência (p99) e custo por chamada, disponibilidade, TTFT (Time-To-First-Token)

- Compliance (L12): Trilhas de auditoria, aderência a políticas, tempo de adequação a novos requisitos.

---

4) Guia: “Como escolher a arquitetura certa para o seu caso”

A. Decida pelos Níveis Organizacionais (L9–L12, L10):

1. Regulação & dados sensíveis (L12) → necessidade de residência/mascaramento? Se sim, favoreça on-prem/híbrido, proxies de PII e *guardrails* no *edge*.
2. Orçamento e TCO (L10) → CAPEX x OPEX (Servidor próprio ou Cloud)? *Burst* imprevisível? *Routing* de modelos ajuda a controlar custo/qualidade.
3. Plataforma (L9) → *managed services* integram melhor etapas (RAG + inferência) do que *do-it-yourself*, mas elevam *lock-in*. (Ver notas sobre modularidade e integração com ferramentas no artigo de MLOps.)

B. Escolha o motor (L4) e a estratégia de *grounding*:

- Se o modelo base não domina o domínio → RAG primeiro; *fine-tuning* só quando há *dataset* robusto e necessidade persistente.
- Se há estilo/voz e func-calling complexo → considere SFT/DPO (Reforço) + *evals* rigorosas e *canary*.

C. Operação & governança (L7–L8):

- Priorize *pipelines* automatizados (dados→evals→deploy) e monitoramento contínuo (qualidade, *drift*, custo, *guardrail violations*). (Automação e *monitoring* são pontos centrais das melhorias propostas no artigo de MLOps.)
- Modele o processo em BPMN com duas raias para clarear *handoffs* entre estratégia (objetivos, *gates*) e operação (execução, telemetria). (Ver Figura 4, p.8.)

D. Forma canônica de decisão (texto-árvore):

- Preciso de conhecimento proprietário? → RAG.
- Respostas com estilo/fluxo estruturado? → *Prompt templates* + *schemas*; se insuficiente → SFT/DPO.
- Custo/latência críticos? → Routing + quantização (L3) + *KV-cache*.
- Risco regulatório alto? → On-prem/híbrido, *PII proxy*, *audit trails*, *human-in-the-loop*.

Guia Rápido: Quando Cada Arquitetura é Indicada

- Pipeline (MLOps): Equipes iniciantes e cadência previsível.
- Microserviços (MLOps): Necessidade de escala, flexibilidade e equipes autônomas.
- APIs Black-Box: Rápido *time-to-value*; foco em prompting, custos e latência.

- LLMs Embutidos: Aceleração de projetos com foco em governança de dados e ROI.
- RAG: Quando "aterrar" respostas em dados atualizados é mais crítico que retreinar.
- Fine-Tuning/SLMs: Quando precisão e contexto de domínio são mandatórios.
- Agentes: Para compor fluxos complexos com múltiplas ferramentas e supervisão humana
- Agentes MCP: Quando for necessário um agente apenas (ou um agente manager) com muitas ferramentas distintas , sendo necessário unificar em um protocolo.
- LLMs multimodais: Soluções que exigem interpretabilidade para diferentes fontes de dados, como áudio, imagem e vídeos.

---

5) Perguntas-guia (para Discovery e desenho do pipeline)

1. Dados & Governança (L12/L10): Há PII/sigilo? Requisitos de residência? Auditorias periódicas?
2. Objetivo de negócio & SLOs (L10/L8): Qual métrica de sucesso (CSAT/ROI/tempo de tarefa)? Latência p95? *Budget* por 1M tokens?
3. Estratégia de conhecimento (L2/L5/L6): O que é conhecimento estável vs volátil? RAG basta ou requer *fine-tuning*?
4. Ciclo de mudança (L7): Frequência de mudança em *prompts*, coleções RAG e *políticas*? Há *evals* automatizadas? (Automação/CI/CD recomendadas em MLOps.)
5. Plataforma (L9): Cloud *managed* (integração rápida) vs on-prem (controle/soberania)? Haverá *autoscaling* e isolamento?
6. Riscos & Controles (L8/L12): *Guardrails*, *rate-limits*, *redaction*, *human-in-the-loop*, trilhas de auditoria?

---

6) Módulos exportáveis (pipelines específicos reutilizáveis entre arquiteturas)

> ***Ideia-chave:*** *cada arquitetura tem um “pedaço específico” (pipeline) que pode ser modularizado e plugado em outras — promovendo interoperabilidade (MLOps), clareza de handoffs (BPMN) e governança por níveis (Lx).*

M1) RAG Pipeline (da Arquitetura RAG)

- O que é: Ingestão - cleaning - embeddings - index vector - retriever→prompt builder. Pode ser enxertado em APIs black-box, Agentes, Multimodal e Fine-tuning (como grounding de validação).
- Benefícios: Reduz alucinação; mantém freshness; evita overfitting de fine-tune para conhecimento volátil.
- Complexidades: Orquestrar reindexações; drift semântico; latência de busca.

M2) Prompt Engineering + Prompt Registry (da Arquitetura API Black-box)

- O que é. Catálogo versionado de templates, A/B testing, prompt lineage, evals contínuas. Reutilizável em todas as arquiteturas (Agentes, RAG, Multimodal, Fine-tune).
- Benefícios. Controle de qualidade/estilo; rollbacks rápidos; governança de custo por token.
- Complexidades. Jailbreak/injection; dependência do modelo.

M3) Fine-tuning/SLM Pipeline (da Arquitetura Fine-tuning/SLMs)

- O que é. Curadoria/rotulagem→datasets→treino (SFT/DPO/RLHF)→registry→canary→rollout. Pode ser plugado a RAG/Agentes/Multimodal para estilo/robustez.
- Benefícios. Personalização profunda; function-calling robusto.
- Complexidades. Licenças/IP; data leakage; custo de serve.

M4) Planner/Executor & Tool Adapters (da Arquitetura de Agentes)

- O que é. Planner (decomposição), executor (roteamento entre LLMs e ferramentas), adapters para APIs. Exportável para APIs black-box, MCP (padrão), RAG (retrieval como tool) e Multimodal.
- Benefícios. Automação multi-passo; composição.
- Complexidades. Controle de autonomia; erros de execução.

M5) Router de Modelos & Políticas de Custo/Latência (da Arquitetura Model Routing)

- O que é. Roteamento por política (qualidade/custo/latência), canary/fallback.
- Benefícios. Otimiza TCO/SLOs; abstrai provedores.
- Complexidades. Observabilidade fina; A/B.

M6) Guardrails, Redação de PII & Auditoria (lanes Compliance/OPs)

- O que é. Filtros IO, policies as code, retenção/mascaramento, trilhas de auditoria. Reaplicável em todas as arquiteturas.
- Benefícios. Redução de risco regulatório e reputacional.
- Complexidades. Cobertura e bypass; governança multi-fornecedor.

M7) CI/CD & Evals Contínuas (das arquiteturas clássicas de MLOps)

- O que é. Automação de dados→treino→deploy→monitoramento, com gates de evals. Enxertável em qualquer stack LLMOps.
- Benefícios. Time-to-value, rollbacks previsíveis.
- Complexidades. Orquestração multi-equipes; data lineage.

M8) Streaming/Tempo Real (derivado de OPs/Infra)

- O que é. Server-push, partial decode, KV-cache, backpressure. Modular para chat live, voz e copilotos in-app.
- Benefícios. UX responsiva; produtividade.
- Complexidades. Tail latency; throttling.

M9) MCP Server + Catálogo de Tools/Recursos (da Arquitetura MCP)

- O que é. Servidor MCP que expõe ferramentas/recursos/prompts com autenticação, rate limits e contratos; plug-and-play para agentes de qualquer provedor. Pode substituir/adaptar tool adapters de agentes legados.
- Benefícios. Padroniza integrações; reduz glue code; promove reuso corporativo.
- Complexidades. Segurança (consent/escopos), versão de tools e compatibilidade.

M10) Gestão Multimodal & RAG Cross-modal (da Arquitetura Multimodal)

- O que é. Ingestão/OCR/ASR/frames→embeddings por modalidade→multi-vector store→retriever cross-modal→construção de contexto → VLM/LLM.
- Benefícios. Grounding rico; cobertura de cenários reais (voz, imagens, PDFs).
- Complexidades. Custos de featureization; schemas heterogêneos; data rights por mídia.

---

7) Três playbooks de arquitetura

A) Assistente de conhecimento interno (Compliance Alto)

- Arquitetura: RAG Clássico + *prompt templates* estritos + *guardrails* + *PII proxy* + *human-in-the-loop*.
- Medições: *faithfulness*, *retrieval recall@k*, violações de *guardrail*; latência p95; custo/1k tokens.
- Foco de níveis: L2/L5/L7–L9/L12. (Automação/monitoramento para *drift* e *freshness*; residência de dados.)

B) Copiloto de produtividade para times internos

- Arquitetura: Routing (modelo pequeno para *drafts*; grande para *critical turns*) + *memory curta* + *tool use* (calendário, documentos).
- Medições: *task completion*, *human takeover rate*, NPS.
- Foco de níveis: L5–L6–L8; controle de custos (L8/L9) e políticas de uso (L12).

C) FAQ público com low-touch

- Arquitetura: Prompt-Only + *caching* + *A/B* de *prompts*.
- Medições: CSAT, latência, custo/consulta.
- Foco: L5, L8, L9 (serviços gerenciados), com *gates* estratégicos para escalar.

---

8) Anti-patterns (o que evitar) e trade-offs recorrentes

- "RAG sem governança": índice desatualizado/sem *evals* → *drift* semântico e alucinação. (A importância de monitoramento e automação é enfatizada no artigo de MLOps.)
- "Tudo via *fine-tuning*": quando *prompting/RAG* resolveriam com menor custo e risco.
- "Observabilidade fraca": falta de *lineage* de *prompts*, de *telemetria* do agente e de *rollbacks* aumenta MTTR (Mean Time to Repair).
- "Arquitetura presa a uma nuvem": integração ótima, porém *lock-in* alto; mitigue com contratos e camadas de abstração. (O artigo de MLOps comenta desafios de integração multi-ferramentas e custos.)

---

9) Como documentar e governar (passos mínimos)

1. Mapa BPMN do fluxo (duas raias) incluindo *gates* e *handoffs [2]* (ver Figura 4, p.8).
2. Catálogo de *prompts* com versionamento/lineage (L5/L7) e *evals* automatizadas por *template*.
3. Políticas LLMOps em L8: roteamento, *rate-limits*, *guardrails*, *redaction*, retenção de logs.
4. Observabilidade: *dashboards* de custo/token, p95, *faithfulness*, *retrieval recall*, violações de segurança.
5. Ciclos de revisão: *business value assessment* periódico e lifecycle assessment para ajustar/aposentar artefatos (ver BPMN).

---

11) Conclusão

Projetar a arquitetura correta de pipeline em LLMOps é ato estratégico e técnico. Use os Níveis L1–L12 para explicitar causas/efeitos e *trade-offs*; adote padrões de MLOps (automação, CI/CD, monitoramento, contêineres/orquestração) como base operacional; formalize a colaboração Estratégia↔Operação com BPMN e *gates* claros. Isso reduz riscos de qualidade, custo e conformidade, e acelera a captura de valor. (Ver o diagrama geral e tabela de *trade-offs* em MLOps, e o processo BPMN para LLMOps com raias e laço infinito.)

Apêndice A — Métricas sugeridas por camada (exemplos)

- RAG: *recall/precision@k*, *faithfulness*, *MMR diversity*, *freshness* do índice, custo/consulta.
- Agentes: *task completion*, sucesso de *tool use*, *unsafe action rate*, *handoff* humano.
- Operação: p95/p99, custo/1M tokens, *SLO* de qualidade/latência/custo, taxa de *rollback* e MTTR.

Apêndice B — Papéis e handoffs (resumo BPMN)

- Estratégia: escopo, *business reqs*, aprovação de arquitetura, *business value assessment*.
- Operação/Dev: requisitos funcionais, arquitetura do artefato, *data prep*, seleção/fine-tuning, *build*, *evals*, *deploy*, integração, monitoramento/manutenção, *lifecycle assessment*. (Ver Figura 4, p.8.)

Referências

- [1] Arquiteturas de MLOps – definição de padrões, automação, CI/CD, monitoramento, contêineres e *trade-offs* pipeline/microsserviços/híbrido; Figura do ciclo geral (p.5) e Tabela de prós/cons (p.5).
- [2] BPMN para LLMOps – processo padronizado, duas raias (estratégica e operacional), laço infinito (Figura 3, p.6) e mapa BPMN completo (Figura 4, p.8); *black-box* de infraestrutura.
- [3] Níveis de Organização L1–L12 – pirâmide de níveis, interconexões top-down/bottom-up, métricas e riscos por nível; exemplos de causa-efeito (privacidade/PII; quantização).

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 2 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa Semana, corroborando o processo de construção de um material de direcionamento e especialização em LLMOps, desenvolvi um trabalho focado em destrinchar as etapas principais do processo de LLMOps, definidas no artigo inicial (LLMOps: Definitions, Framework and Best Practices), atribuindo a cada uma delas uma análise detalhada dos principais pontos a serem levados em consideração antes de começar cada uma delas, ou seja, no planejamento.

As 5 macro etapas definidas no artigo citadas, e que serão usadas como referência no decorrer da Residência são:

- Definição dos requisitos (Estratégia e Compliance)
- Pré-processamento dos dados (Dados)
- Engenharia de modelo para treinamento (Modelo)
- Engenharia de modelo para inferência (Modelo e Operações)
- Monitoramento e observabilidade (Operações)

Porém, arbitrariamente, adicionei uma nova macro etapa que "junta" todas que é a camada de "Automação de Operações" (Dev), uma vez que essa é uma etapa que pode levantar requisitos próprios e possui uma complexidade comparável às demais etapas, fechando assim 6 macro etapas do pipeline LLMOps.

Com isso, estabeleci a organização que usaria para separar as arquiteturas em partes claras e que fizessem sentido para fechar o pensamento a respeito de cada arquitetura de forma visual

Link da guia do documento com as arquiteturas: Resumo geral de arquiteturas - LLMOps

A partir daí, com a visão adquirida a partir das macro etapas, arquiteturas e níveis de organização, desenvolvi um **framework estratégico para sistemas LLMOps** com auxílio do ChatGPT Pro, que conta com:

- Guia das macro-etapas para decisões
- Árvores de decisões arquiteturais
- Métricas de referência para cada macro-etapa
- Mapa para os níveis de organização
- Entregáveis de planejamento por etapa

- Exemplos práticos

Dentro desse documento, com base no documento das arquiteturas de LLMOps e no livro LLMOps: Managing Large Language Models in Production, levantei pontos e perguntas importantes para a parte estratégica (ou time estratégico) responder antes de cada uma das macro etapas do pipeline LLMOps como um todo, o que inclui também definir a arquitetura LLMOps mais adequada. Dentre esses pontos e perguntas, estão por exemplo:

- Sua solução precisa mesmo de um LLM?
- Qual é o usuário final da minha aplicação LLM-based? (cliente, colaborador, outro sistema interno da empresa, …)
- Como escolher a melhor arquitetura de LLM (ou FM) para o meu caso? (Encoder-only, Decoder-only, Encoder-Decoder)
- Qual infraestrutura usar no projeto? (Infra/servidor local, Full-Cloud, Híbrido)
- Qual o nível de automação necessário e onde essa automação é mais importante? (Baixo, focado apenas no pipline de dados; Alto, no decorrer do sistema completo; …)

Link do documento: Framework estratégico - LLMOps

Além disso, realizei a diagramação das arquiteturas principais, com o objetivo de conseguir enriquecer o material da Guia das arquiteturas e, para isso, criei uma nova guia no documento focado apenas nas diferentes arquiteturas LLMOps, trazendo o diagrama visual dela, considerando que as informações principais a respeito de cada uma estão na guia principal.

Para ajudar na visualização e organização dos diagramas, seguindo as macro-etapas do pipeline de LLMOps estabelecidas anteriormente, estabeleci que todos teriam 6 divisões principais inspiradas nas macro-etapas do pipelines de LLMOps, sendo elas: Estratégia, Compliance, Dados, Modelo, Dev (automação e testes) e Operações (Endpoint, monitoramento, etc). Dessa forma, pude explorar bem os elementos de cada arquitetura, criando uma visão modular de como gerenciar cada parte das arquiteturas existentes e consolidadas para criar novas, mais adaptadas para cada caso de uso (Seção 6 do documento - Módulos exportáveis).

Link do documento: Guia de arquiteturas - LLMOps

Por fim, como planejado, estabeleci uma stack inicial de desenvolvimento seguindo a organização das macro-etapas e a divisão de módulos exportáveis. Essa stack conta com quatro categorias de ferramentas que são apresentadas primeiro dentro de cada macro-etapa do pipeline LLMOps, sendo elas:

- Ferramentas GCP
- Ferramentas AWS
- Ferramentas Azure
- Ferramentas open-source

E, segundo, com ferramentas mais específicas para etapas que podem ser utilizadas em praticamente qualquer arquitetura, como mostrado na seção 6 do documento de Resumo geral das arquiteturas, na guia de Guia de arquiteturas, como por exemplo ferramentas para:

- Ferramentas para pipelines de RAG
- Ferramentas para Fine-tuning
- Ferramentas para Model Routing
- Ferramentas para MCP Servers

Link da guia do documento: Resumo geral de arquiteturas - LLMOps

Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Criar repositório base para a criação dos pipelines de cada arquitetura e planejar organização

Encontrar e/ou escolher elementos base para o desenvolvimento dos exemplos-guia, sendo eles:
- Base de dados para recuperação (Módulo de RAG)
- Base de dados para Fine-tuning
- LLM base para fine tuning no HuggingFace
- LLM closed-source para pipelines simples

Planejamento e desenvolvimento inicial de plataforma para unificação do conhecimento, direcionamento de estudos e anamnese de maturidade de pessoas e empresas em LLMOps

Finalizar o planejamento de desenvolvimento de exemplos-guia para as arquiteturas principais, ou seja, planejar protótipos de aplicações das arquiteturas com base em um dataset comum

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Framework estratégico para sistemas LLMOps

Sumário

Objetivo do documento

A seguir está um framework estratégico de LLMOps (v1.0) que transforma as 6 macro-etapas do pipeline LLMOps em perguntas orientadoras, decisões, métricas e entregáveis. O material é enxuto, prático e apoiado por referências sólidas, além de mapear cada etapa aos níveis de organização do LLMOps para ajudar no alinhamento tático-estratégico.

Como usar este framework

1. Leia a etapa correspondente ao seu momento (descoberta, POC, rollout, operação).
2. Responda às perguntas (checklist).
3. Defina métricas/meta (SLOs/OKRs) e artefatos de saída de cada etapa (o que precisa existir para avançar).
4. Tome decisões explícitas (trade-offs) e registre em um ADR (Architecture Decision Record).
5. Aplique governança e riscos com base em NIST AI RMF/ISO 42001 e OWASP LLM Top-10. (NIST)

Guia das macro-etapas do pipeline

1) Definição dos requisitos (Estratégia & Compliance)

Objetivo: validar se *precisa mesmo* de LLM, qual problema de negócio resolve, quem usa, sob quais regras e com qual apetite a risco/custos.

Perguntas-chave

- Precisa de LLM? Alternativas simples (busca BM25, regras, forms, FAQ, workflow) resolvem?
- Quem é o usuário final? (cliente, colaborador, outro sistema) e qual "momento de uso"?
- Qual tarefa alvo? (ex.: responder dúvidas reguladas, resumir documentos, apoiar decisão)
- Qual resultado esperado? (tempo economizado, redução de erros, receita, CSAT/NPS)
- Qual tolerância a erro/alucinação e exposição? (alto risco? exige revisão humana?)
- Que dados entram/saem? Há PII ou dados sensíveis? Há bases internas para RAG?
- Regulação aplicável? LGPD (Brasil), requisitos setoriais, EU AI Act caso haja UE.
- Modelo e arquitetura de LLM: Encoder-only (classificar/buscar), Decoder-only (gerar texto), Encoder-Decoder (seq2seq).
- Build vs. buy: API proprietária, open-source self-hosted, fine-tune ou apenas RAG?
- Infraestrutura: on-prem, cloud ou híbrido? Restrições de residência de dados?
- Nível de automação necessário (p.ex., só pipeline de dados X sistema completo com agentes)?
- Métricas de sucesso e SLOs (qualidade, custo e tempo)?
- Governança: quem aprova releases; quem é DPO/Encarregado; como auditar?
- ***PERGUNTA FINAL: Qual arquitetura de sistema LLMOps é a mais adequada?***

Decisões (exemplos de trade-offs)

- RAG x Fine-tuning: RAG reduz custo/treino e favorece atualização; fine-tuning melhora estilo/consistência mas exige dados curados e TEVV.
- API de terceiros x self-host: API acelera *time-to-value*, self-host dá controle de custo/latência/dados.

Métricas/SLOs complementares

- Time-to-First-Value (semanas) e porcentagem de tarefas assistidas.
- Qualidade: *task success rate*, *hallucination rate* (≤ X%), *guardrail violations* (tendendo a zero).

- Custo: R$ por tarefa concluída; R$ / 1M tokens; teto mensal.
- Risco/Compliance: % fluxos com PII redigida; RIPD/DPIA concluído; auditorias passadas.

Entregáveis

- PRD (Product Requirements Document) de IA com as perguntas respondidas (problema, usuários, hipóteses de valor).
- Mapa de riscos com NIST AI RMF & Perfil de GAI (Gerative AI Profile). (NIST)
- Políticas (retenção, PII, *zero-data-retention*, *human-in-the-loop*).
- ADR: modelo/arquitetura/infra escolhidos. (para isso usar o documento Guia de arquiteturas)

Boas práticas / cases

- Morgan Stanley escalou assistente interno (RAG sobre base proprietária) com 98% de adoção ao priorizar *evals* e controles desde o início. (OpenAI)
- Databricks recomenda centralizar *model hub*, vetores e *human feedback* como pilares de LLMOps. (Databricks Documentation)

Gate de saída: problema, métricas e políticas definidos; risco/regulatório mapeados; ADR aprovado.

---

2) Pré-processamento dos dados (Dados)

Objetivo: preparar o “combustível” do sistema (dados e contexto), garantindo qualidade, segurança e *retrieval* eficaz.

Perguntas-chave

- Quais fontes confiáveis? (repositórios, wikis, PDFs, tickets) e quem é dono de cada uma?
- Há PII/segredo? Como anonimizar/redigir? (LGPD) (Serviços e Informações do Brasil)
- Como chunkar e etiquetar? Tamanho de *chunk*, sobreposição, metadados (autor, data, tipo).
- Qual estratégia de versão/freshness? Reindexação, *SLA de atualização*, arquivamento.
- Base vetorial: qual (e por quê)? Relevância de filtros (por campo), MMR/diversidade.
- Dados de avaliação: *golden set* realista (perguntas, contexto, respostas corretas) para possível fine-tuning.
- Política de logs/retention do índice e consultas (privacidade/forense).
- Idioma/variante (PT-BR) e termos técnicos: há *drift* semântico?

Métricas

- Retrieval: *precision/recall@k*, *latência de busca* e *context utilization*. (Ragas)
- Qualidade do conteúdo: taxa de duplicidade, *metadata fill-rate*, % documentos vencidos.
- Segurança: % PII redigida, incidentes de *leakage* em índices.

Entregáveis

- Data Map + Catálogo de contexto (com donos e SLAs).
- Esquema de chunk & metadados versionado.
- Índice vetorial com *política de atualização*.
- Conjunto de avaliação rotulado para RAG (dataset baseline).

Boas práticas / referências

- Métricas específicas de RAG (faithfulness, answer relevancy, context precision/recall). (Ragas)
- Diagnóstico e *tracing* de RAG em produção com Arize Phoenix; guias de avaliação da Google Cloud. (Phoenix)

Gate de saída: índice com *recall@k* ≥ meta, PII tratada, *golden dataset* pronto.

---

3) Engenharia de modelo para treinamento (Modelo)

Objetivo: decidir se há *fine-tuning* (como, quando e por quê), com testes controlados e guarda-corpos.

Perguntas-chave

- Somente RAG basta? Quando *SFT/LoRA* agrega (estilo, formato, ferramentas, multilíngue)?
- Modelo base: proprietário via API, *open source* self-host, *size* e licença compatível?
- Dados de treino: curadoria, balanceamento, deduplicação, *safety* e representatividade.
- Critérios de sucesso offline: quais *evals* passar antes do *deploy*?
- Custo/infra: GPUs, *FLOPs*, janela de contexto, quantização aceitável?
- Riscos: viés, *jailbreak*, uso indevido; quem audita e aprova?

Métricas

- Qualidade específica da tarefa (ex.: *exact match*, *factuality/faithfulness*).
- Segurança/toxicidade e guardrail violations (OWASP LLM Top-10). (OWASP)
- Custo e latência por 1k tokens (treino e *inference-like*).
- Energia/pegada estimada (para reporte ESG, quando aplicável).

Entregáveis

- Plano de treino (dados, scripts, hiperparâmetros, orçamento, *fallback*).
- Relatório de *evals* (pass/fail) com *golden set*.
- Model Card + Data Card.

Boas práticas / referências

- Databricks: fluxo LLMOps com *model hub*, vetores e *human feedback* integrados ao ciclo. (Databricks Documentation)
- NIST AI RMF & GAI Profile para *TEVV* (teste, avaliação, verificação e validação). (NIST)

Gate de saída: *evals* batendo *baseline* + *safety* aprovado; ADR de *fine-tuning* assinado.

---

4) Engenharia de modelo para inferência (Modelo & Operações)

Objetivo: servir com qualidade, previsibilidade de custo e segurança, suportando picos e *fallbacks*.

Perguntas-chave

- Como servir? API externa, *gateway* próprio, *router* multi-modelos, *function calling*?
- SLOs de latência (p50/p95/p99) e concurrency esperada?
- Custos e *FinOps*: *rate limits*, *budget caps*, *circuit breakers* e cache (KV-cache).
- Segurança: OWASP LLM (injeção de prompt, saída insegura, *excessive agency*). (OWASP)
- RAG em produção: filtros de acesso, versão do índice, *time-to-freshness*.
- Observabilidade (tracing por *span* de cada chamada, *prompt lineage*).
- Fallbacks e *degradations*: *retry*, *smaller model*, *template alternativo*, *human handoff*.

Métricas

- Qualidade online: *answer quality score* (E2E), *handoff* humano ≤ X%.
- Plataforma: erro (% 4xx/5xx), latência (p50/p95/p99), custo por chamada.
- RAG: *precision/recall@k* em tempo real; *context utilization*; *stale rate* do índice.

Entregáveis

- Runbooks de incidentes (latência alta, custo fora do teto, *guardrail violation*).
- Políticas de roteamento (modelo, *cost cap*, fallback).
- Testes de carga e *chaos drills*.

Boas práticas / referências

- Arquiteturas de referência de LLMOps para RAG e *model serving* (3ºs APIs vs self-host). (Databricks Documentation)

Gate de saída: SLOs sustentados em *load test* + *runbooks* validados.

---

5) Monitoramento & Observabilidade (Operações)

Objetivo: medir o que importa (qualidade, custo, risco), detectar *drift* e fechar o ciclo de melhoria contínua.

Perguntas-chave

- O que medir continuamente? (qualidade de resposta, segurança, custo, latência, uso)
- Como avaliar qualidade em produção? *A/B*, *eval suites*, amostragem com revisão humana.
- Quais *dashboards* e alertas? Quem atende; MTTR alvo?
- Telemetria e privacidade: retenção/anonimização de logs (LGPD). (Serviços e Informações do Brasil)
- Como re-treinar/ajustar? Quais gatilhos para atualizar índice, *prompts* ou modelo?

Métricas

- Qualidade: *faithfulness*, *answer relevancy*, *context precision/recall*, *toxicity*. (Ragas)
- Uso: tarefas assistidas/dia, *adoption rate*.
- Risco: taxa de *prompt injection* bloqueada, *unsafe action rate*.
- Custo: R$ / tarefa, % acima do orçamento.

Entregáveis

- Painéis operacionais + painel executivo (valor/custo/risco).
- Catálogo de *evals* (offline/online) versionado.
- Plano de resposta a incidentes de GenAI (com OWASP LLM). (OWASP)

Boas práticas / referências

- *Tracing* + *evals* para RAG/Agentes (Arize Phoenix, LangSmith/Align Evals). (Phoenix)
- Guia Google para evitar “falhas silenciosas” em RAG com avaliação automatizada. (Google Cloud)

Gate de saída: *dashboards* operacionais ativos; *alerts* e *playbooks* testados; *eval loop* rodando.

---

6) Automação de Operações (Dev)

Objetivo: tornar reprodutível e seguro *prompts/índices/modelos*, com CI/CD + avaliações como *gates*.

Perguntas-chave

- O que versionar? *Prompts*, *indexers*, modelo, *evals*, políticas, *guardrails*.
- Como liberar mudanças? PR + CI, *eval suite* → *staging* → canário → produção.
- Feature flags para *prompts/agents*?
- Rollback: quando, como e para onde?
- Automação de compliance: trilhas de auditoria, *model/data cards*, *policy as code* (PII).
- FinOps automatizado: alertas de custo, *budget caps* por serviço/time.

Métricas

- Tempo de ciclo (ideia → prod), MTTR, taxa de rollback; cobertura de testes/evals.
- Conformidade: % mudanças com *policy checks*; *audit pass rate*.

Entregáveis

- Pipeline CI/CD com *gates* de avaliação.
- Prompts registry e policy as code (PII, *blocked terms*, *safety rules*).
- Catálogo de jobs (indexação, re-treino, *sweeps* de *prompts*).

Boas práticas / referências

- Fluxos LLMOps com ambientes (dev/stage/prod), *human review* e *model serving*. (Databricks Documentation)
- OWASP GenAI para *hardening* contínuo (supply chain, saída insegura, agência excessiva). (OWASP)

Gate de saída: *pipeline* operando com *gates*; *feature flags* e *rollbacks* prontos.

Decisões arquiteturais — árvores rápidas

1) Preciso mesmo de um LLM?

- Se tarefa é determinística/curta e documentação é estável → regras/busca.
- Se precisa gerar texto estruturado a partir de documentos internos → RAG leve.
- Se precisa estilo/voz específicos ou ferramentas → RAG + *fine-tuning*.
- Se há requisitos extremos de latência/privacidade → self-host + quantização (avaliar TCO).

2) Escolha de arquitetura (FM/LLM)

- Encoder-only: busca/embeddings/classificação.
- Decoder-only: geração, chat, agentes.
- Encoder-Decoder: tradução/sumário de alta fidelidade.

Métricas de referência (coloque metas)

- Qualidade: *task success*, *faithfulness* (RAG), *guardrail violations*. (Ragas)
- Produto: *adoption rate* (usuários ativos), TTFV, CSAT/NPS.
- Operação: p50/p95/p99, custo por tarefa.
- Risco/Compliance: % PII redigida, auditorias NIST/ISO/AI Act concluídas.

Exemplo de impacto mensurável: em experimento controlado, devs com GitHub Copilot completaram tarefa 55% mais rápido — um bom *proxy* para definir metas de produtividade quando o seu caso de uso é semelhante (assistência à escrita). (The GitHub Blog)

Governança e Compliance (o que não pode faltar)

- RIPD/DPIA para fluxos com PII (LGPD) e políticas claras de retenção/anonimização.
- NIST AI RMF + Perfil de GenAI: use para mapear riscos e ações por função (Govern, Map, Measure, Manage). (NIST)
- ISO/IEC 42001: considere adotar o AIMS (AI Management System) para processos e auditorias. (ISO)
- OWASP LLM Top-10: trate injeção de prompt, saída insegura, agência excessiva e cadeia de suprimentos desde o design. (OWASP)
- Brasil – Marco de IA: acompanhe o PL 2338/2023 (aprovado no Senado em 10/12/2024 e encaminhado à Câmara em 17/03/2025). (The Library of Congress)

Mapa para os Níveis de organização do LLMOps

- Etapa 1 toca L8–L12 (sistema, infra, organização, ecossistema e esfera regulatória).
- Etapa 2 foca L1–L2 (token/embedding) e L9 (infra de dados).
- Etapa 3 cobre L3–L4 (camadas/modelo), com impacto em L7 (pipeline).
- Etapa 4 integra L5–L6 (prompt/agente) com L8–L9 (orquestração/serving).

- Etapa 5–6 consolidam L7–L8 (pipeline e organismo LLMOps), retroalimentando todos os níveis.

Entregáveis por etapa (checklist resumido)

Etapa	Entregáveis mínimos
1	PRD de IA; Mapa regulatório (LGPD/AI Act); Matriz de risco (NIST/ISO); ADR de arquitetura/infra
2	Data Map; Esquema de chunk/metadata; Índice vetorial com *SLA*; *Golden set*
3	Plano de treino; *Eval report* (qualidade+safety); *Model/Data Cards*; ADR de *fine-tuning*
4	SLOs e *runbooks*; Políticas de roteamento/fallback; Teste de carga; Plano de *degradations*
5	Painéis operacionais/executivos; Catálogo de *evals*; Playbook de incidentes; Processo de revisão humana
6	CI/CD com *gates* de *eval*; *Prompts registry*; *Policy as code*; Feature flags; Plano de rollback

---

Exemplos práticos para inspirar metas internas

- Assistente interno (Wealth Management): RAG em base proprietária + *evals* rigorosos → adoção maciça (~98% das equipes). Métricas: *time-to-answer*, precisão percebida, conformidade. (OpenAI)
- Assistência a desenvolvedores: ganhos de produtividade (ex.: 55% mais rápido em tarefa controlada) — útil como parâmetro de redução de tempo para tarefas repetitivas. (The GitHub Blog)

Observações finais e próximos passos

- Comece pequeno, meça cedo, automatize depois: POC com RAG leve e *evals* básicos; promova só quando as métricas “fecharem a conta”.
- Trate custo como SLO (não apenas orçamento): *gates* automáticos, *caps* e *alerts*.

- Documente tudo: decisões (ADR), políticas, *runbooks*, *cards* — isso encurta auditorias e acelera evolução do sistema.

Referências

- https://github.blog/news-insights/research/research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness
- https://openai.com/index/morgan-stanley
- https://docs.ragas.io/en/v0.1.21/concepts/metrics/?utm_source=chatgpt.com
- https://docs.databricks.com/aws/en/machine-learning/mlops/llmops
- https://cloud.google.com/blog/products/ai-machine-learning/optimizing-rag-retrieval?utm_source=chatgpt.com
- https://www.nist.gov/publications/artificial-intelligence-risk-management-framework-ai-rmf-10?utm_source=chatgpt.com
- https://www.researchgate.net/profile/Debmalya-Biswas/publication/381469112_Responsible_LLMOps_Integrating_Responsible_AI_practices_into_LLMOps/links/666f3017b769e7691938a6cf/Responsible-LLMOps-Integrating-Responsible-AI-practices-into-LLMOps.pdf

APÊNDICE 4

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 8 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa sexta Semana, continuando os estudos e partindo um pouco para a prática da área de LLMOps, tive diversos imprevistos relacionados a trabalho que me impediram de dedicar o tempo que gostaria para o Processo, porém ainda consegui desenvolver o planejamento da Semana anterior.

Dito isso, realizei o desenvolvimento do planejamento dos exemplos-guia, montando um passo-a-passo completo para a construção do pipeline LLMOps seguindo cinco arquiteturas principais, sendo elas:

- Arquitetura API Blackbox
- Arquitetura RAG
- Arquitetura Finetuning
- Arquitetura Agentes
- Arquitetura Agentes + MCP

Nesse planejamento foram definidos alguns pontos cruciais para determinar quais seriam as partes fixas do pipeline, ou seja, que seriam iguais para todas as arquiteturas. Dentre essas definições, estão:

- Todas as bases de dados serão armazenadas no GCS (Google Cloud Storage)
- Todos os modelos serão hospedados via vLLM e implementados em um endpoint do Cloud Run

Link do documento: Planejamento de exemplos-guia

Encontrei a API que vai servir como base para o meu crawler de coleta de dados, chamada Brapi, que é uma API aberta de coleta de dados do mercado financeiro
Link da API: brapi.dev

A partir dessa API, defini quais seriam os requisitos para os pipelines de RAG e finetuning:

- O pipeline de RAG seria voltado para geração enriquecida com recuperação de dados financeiros de empresas em relatórios construídos a partir dos dados coletados pela API
- O pipeline de finetuning vai ser focado em **tool-use tuning**, ou seja, o treinamento vai ser focado em otimizar o uso das tools da API no modelo

Com isso, pude desenvolver um planejamento para a plataforma para unificação do conhecimento dentro do repositório base para essa plataforma, levantando os requisitos e contando com o ChatGPT para estruturar as partes técnicas.

Link do documento: Planejamento de desenvolvimento da plataforma
Link do repositório: Learning LLMOps

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Continuar desenvolvimento inicial de plataforma para unificação do conhecimento, direcionamento de estudos e anamnese de maturidade de pessoas e empresas em LLMOps

Finalizar o planejamento de desenvolvimento de exemplos-guia para as arquiteturas principais, ou seja, planejar protótipos de aplicações das arquiteturas com base em um dataset comum

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 15 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa sexta Semana, indo para a pratica da minha área de estudo, LLMOps, com algumas limitações de tempo de desenvolvimento, foquei em desenvolver a base para a plataforma de unificação de conhecimento e dos exemplos guia.

O principal desenvolvimento da Semana foi na parte de arquitetação dos exemplos-guia, que antes iriam ser desenvolvidos separadamente, porém agora, com base nos dados da Brapi API, separei a implementação em dois pipelines que se retroalimentam, formando o que pode ser uma aplicação completa se usada na sequência apresentada, com as partes específicas de cada arquitetura. Esses dois pipelines são descritos por:

- Pipeline 1: Arquitetura API Black box gera um dataset especializado em tool-use com prompt tuning; dataset gerado é usado para realizar o fine tuning da Arquitetura Fine tuning; Modelo fine tunado nessa arquitetura vai ser usado na Arquitetura Agentes+MCP com o modelo especializado em tool-use com as rotas da BrapiAPI
- Pipeline 2: Os dados coletados pela BrapiAPI serão armazenados no CloudSQL da GCP. A partir disso, a Arquitetura Agentes vai receber uma tool de Text2SQL que vai ser responsável por realizar queries SQL no banco de dados para retornar os valores corretos para uma gama de perguntas pré definidas, gerando assim documentos com relatórios e analises de dados financeiros que serão devidamente armazenados em um Data Warehouse (nesse caso o bigquery da GCP) e, posteriormente, esses documentos serão submetidos a um pipeline RAG simples para que a Arquitetura RAG ofereça respostas diretas bom base nos documentos gerados

Com isso definido, realizei a configuração inicial do repositório dos exemplos-guia, adicionando:

- Estrutura de pastas do repositório
- Migrations do banco de dados
- Configurações iniciais do repositório (Makefile, .env.example, README, etc)
- Scripts para execuções de ações na GCP
- Arquivos de utilidades (caching, logging)

Além disso, desenvolvi os primeiros elementos da plataforma, que vai contar com uma stack de desenvolvimento focada em:

- React e TypeScript (para a engine)
- Tailwind CSS, React Flow e Framer Motion (para UI e elementos gráficos)
- Firebase (para autenticação)
- Zustand (para backend e gestão de estado da plataforma)

Como não tenho muita experiência com desenvolvimento web, pedi auxílio para colegas de Ciência da Computação e Sistemas de Informação que me ajudaram a entender melhor como gerenciar essas tecnologias. Além disso, utilizei o Cursor Pro com o modo Plan para que, com base nos requisitos que eu levantei, desenvolvesse um plano de desenvolvimento completo para a plataforma.

A partir disso, desenvolvi as funcionalidades inicias da plataforma, incluindo:

- Estrutura inicial do frontend (landing page, páginas, botões, etc.)
- Banco de dados inicial para receber os dados dos usuários
- Ferramenta de diagramação de arquiteturas com ReactFlow

Link do repositório de exemplos-guia: Learning LLMOps
Link do repositório da plataforma: Learning LLMOps Platform

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Realizar testes iterativos nas funcionalidades implementadas da plataforma, principalmente dos formulários

para coleta de dados e diagramas com ReactFlow

Implementar a primeira arquitetura de API Black box dos exemplos-guia em um endpoint para inferência e teste

Observação: [caso precise fazer alguma observação, de qualquer "natureza"]

O objetivo dos pipelines não é, necessariamente, oferecer uma solução mercadológica. Mas, demonstrar de forma dinâmica a capacidade de construção de pipelines de LLMOps com diferentes abordagens, mesmo tendo, a priori, dados que não são textuais.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 5

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 22 de out. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa Semana, foquei em desenvolver principalmente a Plataforma:

Como o ambiente base para o desenvolvimento dos exemplos-guia e de plataforma de aprendizado de LLMOps, realizei a configuração do ambiente da GCP, passando por ferramentas como:

- Cloud Run para deploy de containers
- Cloud SQL para armazenamento dos dados relacionais
- Cloud Scheduler para trigger do Crawler diariamente
- Secret Manager para gestão de credenciais
- Cloud DNS para gestão do domínio da plataforma

Para consolidar a base de dados, realizei o desenvolvimento do crawler para a coleta dos dados da Brapi API e fiz o deploy dele no Cloud Run a partir de um script.

Realizei o desenvolvimento inicial da primeira arquitetura (API Blackbox), porém ainda não realizei o deploy por dificuldades com o Cloud Run.

Agora, sobre a plataforma de aprendizado, realizei a implementação e teste da estrutura básica dela, trazendo páginas que possam conter os materiais que estou desenvolvendo no decorrer do Processo, que incluem:

- Blog: página para apresentação de conteúdos curtos sobre LLMOps, com o objetivo de conscientização
- Guia: página dedicada à apresentação do Guia das Macro-etapas de LLMOps, desenvolvido durante o Processo
- Podcasts: página contendo podcasts gerados por IA (via NotebookLM) sobre temas relevantes e baseados nos materiais desenvolvidos
- Ferramentas: página para apresentação da stack de ferramentas desenvolvida e apresentada durante o Processo
- Quem sou eu: página de portfólio pessoal
- Laboratório: conjunto de páginas que estão em desenvolvimento para apresentação dos formulários de anamnese de maturidade, arquiteturas diagramadas e laboratório para montagem de arquiteturas

Além disso, realizei algumas implementações pontuais de elementos para melhorar UX e UI, contando, por exemplo, a funcionalidade de login com o Google, adição de abas para direcionamento do leitor (para quem é esse conteúdo; após consumir você vai conseguir fazer/compreender…).

Link do repositório de exemplos-guia: Learning LLMOps
Link do repositório da plataforma: Learning LLMOps Platform

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Testar e validar pipeline de dados simples (ELT, ainda sem Transformações) com crawler, Cloud Scheduler e Cloud SQL

Realizar o deploy da plataforma de aprendizado no meu domínio llmops.com.br, tornando-a disponível para o acompanhamento de desenvolvimento

Implementar a primeira arquitetura de API Black box dos exemplos-guia em um endpoint para inferência e teste

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

A plataforma de aprendizado tem por objetivo ajudar pessoas a realizarem o planejamento e direcionamento de desenvolvimento de forma unificada e visualmente agradável. Isso inclui, claramente, ajudar a mim mesmo nos meus estudos e implementações futuras

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 5 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa Semana, o foco principal foi desenvolver de forma concreta tudo que foi explorado no decorrer das Semanas sobre o tema LLMOps, começando pela otimização da plataforma, porém com ênfase nos exemplos-guia.

Realizei o deploy da plataforma de aprendizado na versão inicial, ainda sem algumas funcionalidades, porém já disponibilizada para acesso global no link llmops.com.br

Desenvolvi um pipeline simples de CI/CD, via Github Actions, com funcionalidades como:
- Deploy automatizado na GCP
- Atribuição de Secrets em Cloud
- Testes de carga
- Testes unitários

Além desses ajustes, como a plataforma já se encontra em produção e disponível para acesso público, foquei no final da Semana em:
- Ajuntar o conteúdo desenvolvido no decorrer do Processo (textos, podcasts, links, vídeos, etc)
- Lapidar pontos importantes da estrutura da plataforma
- Adaptar a linguagem dos conteúdos do Processo para a plataforma
- Corrigir erros e informações equivocadas
- Adaptação da página de portfólio

Ainda não finalizei as implementações da plataforma, por isso a finalização desses pontos será feito na última Semana

Realizei o deploy da primeira arquitetura em produção, a Arquitetura API Blackbox e, com isso, implementei também algumas funcionalidades complementares ao serviço, como:
- Biblioteca de prompts para versionamento e configuração otimizada com PromptTemplates do Langchain
- Esquema de observabilidade de LLM com log de entrada, saída, custo e latência
- Mascaramento de PII usando regex
- Sistema de guardrails para Prompt Injection, Toxicidade entre outros
- Limitador de custo de acordo com os preços das APIs
- Roteamento de modelos com OpenRouter e RouteLLM

Nessa arquitetura, como o objetivo é desenvolver um dataset para fine tuning de LLMs para tool-use, foram construídas duas rotas, sendo:
- /chat: funcionalidade de chat (focado no mercado financeiro e com o uso dos guardrails de tópicos para impedir que o assistente fale de outros assuntos)
- /dataset-generation: rota para a geração do dataset, focada em coletar perguntas e apresentar as tools a serem usadas para coletar as informações para responder as perguntas no formato adequado

Para facilitar os testes locais e em produção, construí uma pasta no Postman para executar testes de cada funcionalidade individualmente, separando as requisições por arquitetura em pastas.

Link do Workspace do Postman: Postman Workspace

Além da arquitetura API Blackbox, também fiz a implementação da Arquitetura Agentes, que conta com uma ferramenta de Text2SQL, responsável por realizar consultas no meu banco de dados

relacional a partir de queries criadas pelo LLM. Os pontos principais desenvolvidos dessa arquitetura nessa Semana foram:

- Desenvolvimento do fluxo de agentes com Lang Graph
- Definição da abordagem de Text2SQL a ser usada (agêntico ou serviço)
- Desenvolvimento da ferramenta Text2SQL
- Atribuição e teste local do fluxo completo com o uso da ferramenta
- Validação e armazenamento dos resultados das análises no bucket da GCS

Diante do desenvolvimento dos meus exemplos-guia, achei relevante citar uma referência que uso há muito tempo e que me inspirou na organização do meu repositório, que é o CrewAI-examples: um repositório feito pela equipe da CrewAI focado em mostrar exemplos de implementação das funcionalidades da ferramenta

Link do repositório CrewAI-examples: https://github.com/crewAIInc/crewAI-examples

Link do repositório de exemplos-guia: Learning LLMOps
Link do repositório da plataforma: Learning LLMOps Platform

Link para acessar o conteúdo da plataforma: llmops.com.br

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Finalizar desenvolvimento da arquitetura RAG e implementar em produção

Realizar implementação básica da Arquitetura Fine-tuning

Realizar implementação de servidor MCP na GPS para a arquitetura Agentes+MCP

Lapidar por completo a plataforma e adicionar todos os conteúdos desenvolvidos no decorrer da Residência

Adição de funcionalidades para melhoria da UX da plataforma

Observação: **[caso precise fazer alguma observação, de qualquer "natureza"]**

1. O objetivo do deploy da plataforma em um domínio público, mesmo antes da adição completa das funcionalidades e do conteúdo é: possibilitar o acompanhamento do desenvolvimento da plataforma e promover a colaboração entre pessoas que queiram propor melhorias e oferecer conteúdos para serem publicados nela
2. A escolha arquitetural de microsserviços no desenvolvimento dos exemplos-guia foi motivada pela flexibilidade
3. Provavelmente, a Arquitetura Fine Tuning e a Arquitetura Agentes+MCP não serão colocadas em produção neste momento.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!

APÊNDICE 6

Termo de Aceite de Entrega

Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

Data da Reunião ("Gate") de aprovação: 12 de nov. de 2025

Participantes da Entrega [matriculados em Residência em IA]:

Pedro Ribeiro Fernandes

Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Nessa última Semana, o foco principal foi lapidar tudo que foi explorado no decorrer das Semanas sobre o tema LLMOps e implementar mais arquiteturas.

Durante todas as Semanas, foram desenvolvidos:
1-3 Semana: Pesquisa, desenvolvimento e consolidação de um material base para estudo de LLMOps
4-5 Semana: Estudo e aprofundamento na literatura sobre Arquiteturas LLMOps e aplicações práticas
6-7 Semana: Organização e desenvolvimento da base para recebimento dos conteúdos (repositórios e plataforma)
8-9 Semana: Desenvolvimento e implementação em produção dos exemplos-guia das arquiteturas de sistemas LLMOps e da plataforma de unificação de conhecimento da área
10 Semana: Lapidação da plataforma, testes em produção e deploy final dos exemplos-guia

Realizei o deploy do pipeline de vetorização, conectado com o meu bucket do Google Cloud Storage, que realiza o processo de:

1. Busca no bucket arquivos que não foram vetorizados, uma vez por dia
2. Realiza uma estratégia de chunking híbrido (fixo e dinâmico com LLM)
3. Vetoriza os chunks
4. Salva os chunks vetorizados e seus metadados na tabela SQL do CloudSQL

Realizei o deploy da arquitetura RAG que realiza a busca vetorial na tabela SQL e retorna os chunks com maior similaridade para incluir no prompt como contexto. Foram definidas práticas como:

- PromptTemplates do Langchain para incluir informações complementares e o contexto adicional nos prompts do LLM
- Treshold (limiar de similaridade) de 0.7, para garantir que apenas chunks com alto score de similaridade sejam incluídos no contexto

Além de implementações técnicas, desenvolvi e lapidei os READMEs de cada arquitetura no repositório de exemplos-guia, focando em oferecer uma visão detalhada e prática de cada ponto de cada arquitetura, com o objetivo de facilitar a compreensão do usuário sobre elas

Com o objetivo de lapidar e melhorar a plataforma, com base em feedbacks e sugestões implementei funcionalidades e modificações como:

- Adição de componente flutuante para explicação de termos técnicos
- Adição de seção de “Materiais Externos” para algumas páginas
- Modificação da aba de Guia, agora Guias, que possui vários guias práticos para diversas área de LLMOps, como RAG, Engenharia de dados e MLOps (ainda a serem desenvolvidos)

Adicionei também, na pasta no Postman, as rotas e requisições de testes para executar testes de cada funcionalidade de cada arquitetura em produção individualmente, separando as requisições por arquitetura em pastas.

Link do Workspace do Postman: Postman Workspace

Link do repositório de exemplos-guia: Learning LLMOps
Link do repositório da plataforma: Learning LLMOps Platform

Link para acessar o conteúdo da plataforma: llmops.com.br

Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Observação: **[caso precise fazer alguma observação, de qualquer “natureza”]**

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go!