UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Ambiente de Produção sob tutoria de IA: monitoração, análise e aprimoramento de aplicações

Diógenes Flamarion Pires

Trabalho de Conclusão de Curso
MBA em Inteligência Artificial e Big Data

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Ambiente de Produção sob tutoria de IA: monitoração, análise e aprimoramento de aplicações

Diógenes Flamarion Pires

USP - São Carlos

2024

Diógenes Flamarion Pires

Ambiente de Produção sob tutoria de IA: monitoração, análise e aprimoramento de aplicações

Trabalho de conclusão de curso, apresentado ao Departamento de Ciências de Computação do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo - ICMC/USP, como parte dos requisitos para obtenção do título de Especialista em Inteligência Artificial e Big Data.

Área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Adenilso da Silva Simao

USP - São Carlos

2024

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

P667a

Pires, Diógenes Flamarion
Ambiente de Produção sob tutoria de IA: monitoração, análise e aprimoramento de aplicações / Diógenes Flamarion Pires; orientador Adenilso da Silva Simao; coorientadora Solange de Oliveira Rezende. -- São Carlos, 2024.
50 p.

Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Tutoria de códigos em produção. 2. Eficiência operacional. 3. Qualidade de software. 4. Inovação. I. Simao, Adenilso da Silva , orient. II. Rezende, Solange de Oliveira, coorient. III. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

RESUMO

PIRES, D. F. **Ambiente de Produção sob tutoria de IA**: monitoração, análise e aprimoramento de aplicações. 2024. ?? f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

O desenvolvimento de software tem seu Ciclo de Vida beneficiado pela utilização de Inteligência Artificial em todas as suas etapas, inclusive após a publicação e operação em ambientes de produção. Porém, nessa última etapa referida, os modos de utilização deixam uma lacuna significativa no que tange a análise dos códigos em face dos eventos e ocorrências durante a operação desde considerações sobre desempenho, efetividade, correções, evoluções, atualizações etc. Embora, em muitos casos, tais análises ocorram de fato, elas seguem de modo 'manual', dependendo de especialistas e, até, empiricamente, implicando em falhas, demoras, inconsistências e ineficiência, entre outras consequências indesejáveis e prejuízos. Esta pesquisa investiga a aplicação de inteligência artificial (IA) na tutoria de códigos em ambientes de produção, com o objetivo de melhorar a eficiência operacional e a qualidade dos códigos de aplicações e serviços. O estudo foi conduzido em um banco digital privado, onde modelos e agentes de IA especializados foram implementados para monitorar, analisar e recomendar evoluções, ajustes, correções e atualizações nas aplicações. A metodologia adotada incluiu a coleta de dados de observabilidade e logs do ambiente de produção, que foram utilizados para treinar e validar o modelo central de IA. Os resultados demonstraram que a tutoria de IA não apenas auxilia diretamente na gestão dos códigos das aplicações, mas também impacta positivamente a cultura e os processos de desenvolvimento, especialmente no que tange ao futuro e maneiras inovadoras de utilizar a IA. A IA foi capaz de identificar problemas e recomendar melhorias de forma proativa, resultando em uma redução significativa no tempo de atualização, amadurecimento e evolução dos códigos. Além disso, a análise dos códigos gerados pela IA indicou uma melhoria na qualidade, com a maioria dos códigos atendendo aos padrões estabelecidos. A integração da IA nos processos de desenvolvimento permitiu que os desenvolvedores se concentrassem em tarefas de maior interesse e produtividade, como inovação e desenvolvimento de novas funcionalidades. As conclusões sugerem que a implementação de IA é uma estratégia eficaz para promover a inovação e a eficiência contínua, oferecendo ganhos tangíveis em termos de qualidade do software, estabilidade operacional e satisfação das equipes. Este estudo contribui para o entendimento das potencialidades da IA na gestão de códigos em ambientes de produção, indicando que, com a implementação adequada, a IA pode atuar como um promotor da eficiência operacional e da satisfação de clientes, colaboradores e parceiros.

Palavras-chave: tutoria de códigos em produção, eficiência operacional, qualidade de software, inovação.

ABSTRACT

PIRES, D. F. **Ambiente de Produção sob tutoria de IA**: monitoração, análise e aprimoramento de aplicações. 2024. ?? f. Trabalho de conclusão de curso (MBA em Inteligência Artificial e Big Data) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

Software development has its Life Cycle benefited using Artificial Intelligence in all its stages, including after publication and operation in production environments. However, in this last stage, the methods of use leave a significant gap in terms of code analysis considering events and occurrences during operation, including considerations about performance, effectiveness, corrections, developments, updates etc. Although, in many cases, such analyses do in fact occur, they are carried out in a 'manual' manner, depending on experts and even empirically, resulting in failures, delays, inconsistencies and inefficiency, among other undesirable consequences and losses. This research investigates the application of artificial intelligence (AI) in code tutoring within production environments, aiming to enhance operational efficiency and the quality of application and service codes. The study was conducted at a private digital bank, where specialized AI models and agents were implemented to monitor, analyze, and recommend evolutions, adjustments, corrections, and updates to the applications. The methodology adopted included collecting observability data and logs from the production environment, which were used to train and validate the central AI model. The results demonstrated that AI tutoring not only directly assists in managing application codes but also positively impacts the culture and development processes, especially regarding the future and innovative ways of utilizing AI. The AI was able to proactively identify issues and recommend improvements, leading to a significant reduction in the time required for code updating, maturation, and evolution. Furthermore, the analysis of AI-generated codes indicated an improvement in quality, with most codes meeting established standards. The integration of AI into development processes allowed developers to focus on more engaging and productive tasks, such as innovation and the development of new functionalities. The conclusions suggest that the implementation of AI is an effective strategy for promoting innovation and continuous efficiency, offering tangible gains in terms of software quality, operational stability, and team satisfaction. This study contributes to understanding the potential of AI in code management within production environments, indicating that, with proper implementation, AI can act as a promoter of operational efficiency and the satisfaction of clients, employees, and partners.

Keywords: production code tutoring, operational efficiency, software quality, innovation.

SUMÁRIO

1 INTRODUCÃO.

O Ciclo de Vida do Desenvolvimento de Software (SDLC, *Software Development Life Cycle*) (SHARMA, 2015) pode se valer amplamente da utilização e evolução aceleradas de diversos modelos, ferramentas e Agentes de IA, desde as fases preliminares de desenvolvimento às mais avançadas da operação. Registros e informações de buscas (GOOGLE, 2012), artigos (ETO, 2024), tutoriais, livros e outras produções respectivas à aplicação de recursos e ferramentas de IA abordam a crescente disponibilidade de opções para cada etapa.

No entanto, algumas atividades importantes relacionadas aos códigos publicados e operacionais em ambiente de produção precisam de maior atenção a respeito.

1.1 Contextualização

Desde a ideação, concepção e projeto de softwares, seguindo com o auxílio à codificação (CODECOMPLETE, 2022) (TABNINE, 2018) (GITHUB, 2022), Integração Contínua (CI, *Continuous Integration*) (MICROSOFT, 2024), revisões de códigos (*code reviews*) (VERICODE, 2022), avançando com as gerações, execuções, análises e automações dos diversos tipos de testes e refatorações (REMSOFT, 2023) (VERICODE, 2022) e adiante, promovendo entrega, sustentação e Avaliação Contínua (CE, *Continuous Evaluation*) (VERITY, 2023) (GARTNER, 2024) (INFORCHANNEL, 2023).

Há, especialmente, ferramentas de IA que participam e auxiliam nos processos e práticas que promovem as publicações, como CD (*Continuous Delivery*, Entrega Contínua) e Implantação Contínua (CD, *Continuous Deployment*) (MICROSOFT, 2024), ou nas questões de segurança (VERICODE, 2022).

Após as publicações em produção, as atenções sobre o uso de IA, têm sido, geral e amplamente, determinadas prioridades, como as experiências de usuários (XP, *user e**XP**erience*) (ÅSNE STIGE, 2023, p. 4,5) em suas variadas nuances, monitoração, obtenção de informações e métricas, notificações e alertas sobre o ambiente de produção em si e seu desempenho.

Para tanto, são utilizadas plataformas e ferramentas de monitoração, APM (*Application Performance Management*, Gerenciamento de Desempenho de Aplicações) e observabilidade (IBM, 2023) (BRQ, 2023) (OPSERVICES, 2024). Dois exemplos entre as opções mais conhecidas e usadas, são Dynatrace (**Unsupported source type (DocumentFromInternetSite) for source DYN17.**) e New Relic (NEWRELIC, 2008), com suas respectivas implementações de IA, DAVIS AI (DYNATRACE, 2021), e New Relic AI

(NRAI) (NEWRELIC, 2020). Porém, de fato, essas tecnologias não têm, entre seus pilares, a análise de código em si, nem a indicação de soluções ou aprimoramentos aplicáveis a eles.

Tais instrumentos monitoram e gerenciam disponibilidades e desempenhos das aplicações, dos recursos do ambiente, integrações e eventos com suas características (BRQ, 2023). Referências aos códigos nas observações e registros visam indicar relações com os momentos, interações, implicações e envolvimentos durante suas ocorrências, deixando em outras mãos os códigos em si, suas análises e intervenções.

Não haver análise e tratamento automático e imediato dos códigos fonte nessa etapa do SDLC o que, em tese, pode ser suprido com a tutoria de IA, redunda em perdas de oportunidades e involuções severas e evitáveis.

As informações obtidas por APM e Logs, geralmente, são verificadas em relação ao código, após ocorrerem incidentes ou problemas. As análises e ajustes para evitar ou solucionar as ocorrências a tempo, são comumente tardias, insuficientes, mal-entendidas ou inexistentes. Não raro, a situação chega aos times de sustentação ou desenvolvimento depois de analisada por outras equipes, a menos, ou até que, em alguma perspectiva, o código se torne ‘suspeito’.

Depois de idas, vindas, ‘devidas’ pressões, ‘salas de emergência’ com ‘especialistas’ de ‘mãos cheias’, movidos de suas agendas, *sprints*, turnos ou descanso, chega-se a um paliativo. ‘Alguém’ se responsabiliza pela busca da causa raiz. Porém, prazos pressionam ou se esgotam, impõem-se limitação ou inadequação de capacidade técnica para os níveis e tipos de análise necessários, orçamentos e recursos indisponíveis, outras ‘prioridades’, urgências e novos incidentes. Consolidado o paliativo, os códigos seguem, ‘remendados’, em seus repositórios.

1.2 Objetivo e motivação

A determinação assumida para este empreendimento é estudar a instrumentação de IA (Inteligência Artificial, ou AI, *Artificial Inteligence*) para auxiliar a manter os códigos de aplicações de sistemas corporativos, publicadas em ambiente de produção, continuamente atualizadas, evoluindo e nas suas melhores condições operacionais, de modo contínuo, automático, com a maior brevidade possível ou, pelo menos, viável e compartilhar os resultados dessa experimentação.

A motivação para realizar esta pesquisa é a expectativa de verificar a viabilidade do uso de IA como tutor dos códigos em runtime de produção. Espera-se que a IA (**i**) relacione as informações recebidas de produção com os códigos fonte, (**ii**) analise causas, efeitos, influências e interações entre as aplicações, o ambiente e seus recursos, além de comparar os códigos fonte com outros sob variações de circunstâncias, efeitos, semelhanças, finalidades,

diferenças e funcionalidades, (**iii**) gere e entregue códigos prontos, parciais ou completos, testados e validados, com correções, ajustes e evoluções dos analisados, (**iv**) notifique os times pertinentes e (**v**) envie alertas e avisos a subscritos.

Após esta (**1**) **INTRODUÇÃO** e o (**2**) **REFERENCIAL TEÓRICO** sobre (**2.1**) Entradas, (**2.2**) Processamentos, e (**2.3**) Saídas, são segue a (**3**) **METODOLOGIA** utilizada no experimento e a sua (**4**) **PROPOSTA E DESENVOLVIMENTO**, finalizando com a (**5**) **ANÁLISE DOS RESULTADOS** e as (**6**) **CONCLUSÕES** com considerações pertinentes.

2 REFERENCIAL TEÓRICO

Algumas referências a respeito da utilização de IA de modo geral são pertinentes e necessárias para a realização deste trabalho. Para esse fim, esta sessão aborda, brevemente, algumas, divididas em três blocos: Entradas, Processamentos e Saídas.

2.1 Entradas: dados atualizados continuamente

Entradas referem-se, nesse tópico, às informações obtidas do ambiente de produção, e alguns outros insumos que serão indicados adiante, para serem continua e adequadamente entregues a um Modelo Generativo de IA (Gen AI) (**Unsupported source type (DocumentFromInternetSite) for source Zha23.**), indicando as condições operacionais das aplicações em *runtime*, (TIAGO CARVALHO, 2023) para análise referencial com os códigos fonte relacionados, direta ou indiretamente, às informações obtidas, visando a evolução, refinamento e eventuais ajustes ou correções daqueles códigos.

Como já mencionado, durante o SDLC, estando as aplicações em execução no ambiente de produção, APMs e Log são ferramentas que, atualmente, estão, quase sempre, presentes para acompanhar e auxiliar na saúde e na estabilidade dos sistemas, assim como em eventuais anomalias ou problemas relativos.

APMs, como Dynatrace (**Unsupported source type (DocumentFromInternetSite) for source DYN17.**) e New Relic (NEWRELIC, 2008), com suas respectivas IA, DAVIS (DYNATRACE, 2021), e New Relic AI (NRAI) (NEWRELIC, 2020), são exemplos que têm sido muito bem sucedidos em prover informações e dados sobre os ambientes de produção, como monitoramento de infraestrutura, rastreamento de solicitações, métricas, logs, latências, ineficiências, gargalos, detecção e diagnósticos de falhas, anomalias, problemas, análise de causa raiz, registro de eventos, insights, riscos e ameaças à segurança de recursos e dados, impactos para usuários e processos, entre outras.

Junto às APMs, sistemas especializados nos registros de *Log* (SHEKAR RAMACHANDRAN, 2023) (SONG CHEN, 2022), detalham eventos e atividades que ocorrem com as aplicações e serviços, capturando informações sobre operações, erros, transações e interações dentro do software ou sistema, fornecendo uma visão granular do que acontece durante suas execuções, incluindo informações personalizadas definidas pelos desenvolvedores. Um conjunto de aplicações utilizado para registro de Logs é o ELK Stack (**E**lastic – **L**ogstash – **K**ibana), também conhecido como Elastic Stack (ELK, 2024).

Em (BRQ, 2023), a comparação dos conceitos de Monitoramento e Observabilidade é concluída resumidamente:

> [...] a observabilidade é um conceito mais amplo e estratégico, que envolve a capacidade de entender e compreender o comportamento interno de um sistema ao longo do tempo. Já o monitoramento é uma atividade operacional mais específica, que busca acompanhar o estado atual do sistema em tempo real e detectar problemas imediatos.

Em seguida, são apresentados "Os pilares da observabilidade": Coleta de dados; Armazenamento e processamento; Visualização; Monitoramento proativo; Rastreamento e diagnóstico; e Correlação de dados.

Tanto os documentos técnicos quanto as apresentações comerciais dos produtos de APM e Monitoramento, de modo geral, são concordes com essa abordagem, como se pode constatar em (**Unsupported source type (DocumentFromInternetSite) for source DYN17.**), sobre o Dynatrace, e em (NEWRELIC, 2008), sobre New Relic.

Referencia-se assim a percepção de que, embora as APMs, eventualmente, indiquem diretamente, referências aos códigos ou, mais geralmente, ao que podem 'perceber' externamente a respeito deles, elas, pelo menos ainda, não se ocupam do código em si, de sua análise ou de buscar soluções ou alterações pertinentes. Mesmo com os extensos e relevantes aprimoramentos e avanços proporcionados por IA (DYNATRACE, 2021) (NEWRELIC, 2020), não se remetem a essa perspectiva.

Com efeito similar, resguardadas as particularidades, os sistemas de registro de Log, também não se investem em analisar as aplicações em si, mas refinam seu foco em registrar o que lhes compete, com o mínimo impacto ou interação possível, como é pontuado em (ELK, 2024), ainda que se ocupem, ou possam se ocupar, do registro de informações mais internas ao código, indicações e tratamentos inseridos pelos desenvolvedores e, a depender das implementações, de adicionar referências e eventuais possibilidades para soluções de problemas específicos.

Os dados registrados por APMs e Logs trazem consigo informações relativas às execuções das aplicações ou aos seus contextos, incluindo relações e interações com os demais elementos do sistema e demais aplicações e serviços. Esses dados, relacionados aos códigos fonte das aplicações em execução nos momentos de suas aquisições, possibilitam perspectivas singulares para análises e considerações, especialmente se comparadas entre múltiplas aplicações, contextos e ocorrências.

As informações podem ser compartilhadas e obtidas de formas diferentes, de acordo com cada plataforma. Por exemplo, no caso do Dynatrace, podem ser utilizadas, inclusive, REST API (*Application Programming Interface*: Interface de Programação de Aplicação)

(DYNATRACE, 2024), App Toolkit, que possibilita a criação de componentes especializados ou dedicados (DYNATRACE, 2024), ou os serviços da DAVIS AI (DYNATRACE, 2024).

Esses dados são gerados continuamente durante a operação das aplicações no ambiente de produção e não fazem parte do conhecimento original com que os modelos generativos foram treinados.

Os códigos fonte das aplicações, informações das etapas anteriores do SDLC das aplicações e das execuções dos *pipelines* de CI/CD até a publicação das mesmas aplicações podem ter sido utilizados para ampliar/refinar o treinamento do modelo generativo. Essas 'novas', oriundas do ambiente e execuções das aplicações em produção, podem ser adicionadas às análises feitas pelo modelo generativo por técnicas que ampliam o conhecimento sobre o contexto e dados pertinentes a essas 'novidades'.

Os meios para obter os dados estão disponíveis, mas LLMs (*Large Language Models*) (**Unsupported source type (DocumentFromInternetSite) for source Zha23.**), não têm a capacidade intrínseca de interagir diretamente com processos ou agentes externos (AUFFARTH, 2023, p. 38 - 43), como realizar pesquisas na web, fazer solicitações a APIs externas ou processar entradas de voz. Para realizar essas atividades, o modelo depende de interfaces ou agentes intermediários que realizem as necessárias transduções de suas entradas e saídas.

Uma forma de prover as transduções adequadamente é o uso de Agentes de IA (AUFFARTH, 2023, p. 52, 53) especializados, que possam obter as informações por um dos modos disponíveis na plataforma, preparar, formatar e entregar os dados preparados para a utilização pelo modelo.

Agentes, no entanto, não são meros transdutores, nem, apenas, preparadores de recursos ou informações adicionais para os modelos generativos. Inclusive, o uso da forma plural na referência aos Agentes de IA tem relação imediata com as variedades de tipos, especialidades e objetivos daqueles disponíveis, remetendo à viabilidade de trabalho em conjunto, com ou sem moderação ou orquestração de múltiplos agentes com ações, interações e objetivos igualmente variados, caracterizando um modo de trabalho frequentemente chamado multiagente (NADIRI, 2023) e que, por princípio, potencializa recursos e resultados (AUFFARTH, 2023, p. 52,61,62).

Coletados e preparados adequadamente, os dados oriundos das operações no ambiente de produção podem ser encaminhados para o Modelo Gen AI. Porém, dois outros fluxos de dados podem ser providos para o modelo, a fim de enriquecer as análises e evoluções dos códigos fonte de seus *runtimes* em produção.

De fato, podem ser providas informações de, pelo menos, três origens relevantes:

1. Os dados de observabilidade e Logs dos ambientes de produção, providas através de Agentes de IA especializados;
2. As atualizações dos códigos fonte, a partir de seus repositórios, promovidas pelos desenvolvedores das aplicações, tais que, informadas ao composto de IA pelos *pipelines* de publicação dos códigos, durante os processos de CI/CD, promoverão a ingestão dos dados através de Agentes de IA especializados;
3. As alterações de documentações ou versões, adoções e/ou substituições nas pilhas de desenvolvimento, referências ou instruções adicionais, inclusive de *prompts* disponibilizados para equipes habilitadas para treinamento do composto de IA conforme forem pertinentes, que poderão ter suas entregas ao modelo concretizadas através de agentes especializados.

A essa altura há evidência de que um número elevado de atividades e agentes pode se tornar necessário. A rigor, cada Agente de IA é especializado em uma única responsabilidade. Portanto, facilmente, o número deles pode aumentar, especialmente se levados em conta recursos de monitoração, tratamentos especializados de exceções e erros, comunicações etc.

Outro conjunto de agentes também será necessário para levar aos repositórios os códigos gerados pelo Modelo Gen AI, assim como realizar os alertas e notificações.

Agentes, assim como Ferramentas e outros Recursos, podem ser individualmente desenvolvidos para fins específicos e gerenciados diretamente. Porém, considerando o potencial de multiplicação e as eventuais complexidades envolvidas, inclusive no gerenciamento e otimização dos trabalhos em conjunto deles todos, torna-se interessante a utilização de um facilitador, como um framework ou orquestrador, como o LangChain, desenvolvido em Python (AUFFARTH, 2023) (LANGCHAIN, 2022, p. 46 - 59) ou o AutoGen (MICROSOFT, 2024) (MICROSOFT, 2024), da Microsoft, também desenvolvido em Python, mas disponível, também, para .NET (MICROSOFT, 2024), ambos de código aberto.

O AutoGen é um framework projetado para definir, configurar e compor uma infraestrutura multiagente e oferece uma interface de usuário de baixa codificação chamada AutoGen Studio, que facilita a criação dos fluxos de trabalho.

O LangChain é um framework capaz de construir aplicativos compatíveis com LLMs, integrando módulos reutilizáveis que agregam funcionalidades e recursos, inclusive o

gerenciamento de Agentes de IA, além de, também, contar com uma extensa e ativa comunidade e um grande número e variedade de módulos prontos para uso.

Um conceito central do LangChain é o de criação e combinações de cadeias, inclusive de agentes e da colaboração entre eles com base em diversos paradigmas, como a interação entre eles com base em objetivos (AUFFARTH, 2023, p. 52), por exemplo, e mantém o foco em ferramentas de programação e plugins.

O LangChain tem uma comunidade maior e mais plugins disponíveis, enquanto o AutoGen é mais recente e impulsionado pela Microsoft com uma comunidade em formação e evolução.

Essa implementação, baseada na ingestão, tratamento e contextualização de um LLM por agentes orquestrados, teoricamente, possibilita que um modelo pré-treinado (**Unsupported source type (DocumentFromInternetSite) for source Zho23.**), com os códigos fonte das aplicações em *runtime* de produção, possa analisá-los em face das informações e referências continuamente atualizadas, em tempo muito próximo do tempo real, resultando em elevação significativa da possibilidade de serem mantidos atualizados e evoluindo.

2.2 Processamentos: LLM de IA Generativa contextualizada

Modelos de Processamento de Linguagem, como os baseados em PLN (Processamento de Linguagem Natural, ou NLP, *Natural Language Processor*), podem ser desenvolvidos do início. No entanto, considerados todos os aspectos, como as especializações, recursos tecnológicos e habilidades imprescindíveis para um empreendimento dessa envergadura, resultados satisfatórios demandariam custos e prazos elevados.

Como consequência, provavelmente, ao alcançar maturidade, e apenas para o caso da maioria das organizações suficientemente habilitadas, o produto alcançado, provavelmente já estará desatualizado em relação à evolução das tecnologias pertinentes e, quase certamente, em relação aos pares que optaram por outros modelos de implementação.

Adicionalmente, o desenvolvimento particular incorre invariavelmente em dados de treinamento limitados, salvaguardas e ferramentas restritas e visões ou referências menos amplas do que o desejável para garantir o mínimo espectro fundacional indispensável.

Modelos fundacionais diversos e suficientemente bem estruturados e testados, de modo geral, serão opções mais eficazes do que o desenvolvimento desde o início, especialmente se, consideradas as características e necessidades do caso de uso, modelos pré-treinados com parâmetros e testes/validações suficientemente amplos e qualificados estiverem disponíveis.

Sobre os LLM de IA Generativa, em (AUFFARTH, 2023, p. 38, 39) são indicadas limitações conhecidas que afetam negativamente o processamento, resumidas a seguir em tradução livre:

- **Conhecimento desatualizado**: Confiam apenas em seus dados de treinamento. Sem integração externa, não podem fornecer informações recentes do mundo real;
- **Incapacidade de agir**: Não podem realizar ações interativas como pesquisas ou cálculos. Isso limita severamente as funcionalidades disponíveis nativamente;
- **Riscos de alucinação**: O conhecimento/treinamento insuficiente pode levar à geração de conteúdo incorreto ou sem sentido se não for devidamente fundamentado;
- **Vieses e discriminação**: Dados de treinamento tendenciosos podem produzir vieses de natureza religiosa, ideológica ou política;
- **Falta de transparência**: O comportamento de modelos grandes e complexos pode ser opaco e de difícil interpretação, causando desalinhamento com os valores humanos;
- **Falta de contexto**: Perda de contextos de *prompts*, conversas anteriores ou detalhes mencionados anteriormente, insuficiência de informações adicionais relevantes com o *prompt* fornecido.

Ainda sobre as limitações dos modelos Gen AI, de (ARSANJANI, 2023), a adaptação seguinte indica causas comuns para ocorrência de Alucinação e ajustes aplicáveis para promover sua redução:

- ***Prompts* vagos ou excessivamente amplos**: *Prompts* sem especificidade podem dificultar o entendimento do contexto e gerar respostas irrelevantes ou imprecisas;
- **Conhecimento limitado do domínio**: Treinamentos em conjuntos de dados de uso geral podem não ter referências suficientes para precisão em domínios específicos;
- **Dados de treinamento insuficientes**: Dados de Treinamento carentes de qualidade, podem comprometer a compreensão dos padrões e relacionamentos linguísticos;
- **Incerteza na linguagem**: Linguagem ambígua pode dificultar interpretação de nuances sutis gerando saídas desalinhadas com o significado original.

Ajustes para reduzir ocorrências de alucinações:

- Enriquecer o contexto com informações adicionais ou restrições;
- Treinar com grande volume de dados específicos do domínio para que forneçam melhor compreensão dos padrões e relações relevantes para o contexto;
- Realizar Ajuste Fino (*fine-tuning*) ou Ajuste Fino Eficiente de Parâmetros (*Parameter-Efficient Fine-Tuning*, PEFT) para tarefas ou domínio. O ajuste fino é uma técnica que envolve o treinamento de um LLM em um conjunto de dados menor de dados especificamente adaptado à tarefa em questão ou ao domínio específico;
- Utilizar RAG, uma técnica que amplia o *prompt* com informações adicionais, podendo ter origem em bancos vetoriais de texto ou código, ampliando o contexto com dados quase em tempo real;
- Usar Raciocínio e Consulta Iterativa, técnicas que podem ajudar na geração das respostas. Por exemplo, pedido fornecimento das evidências para as alegações ou geração de explicações alternativas;
- Aumentar a especificidade e clareza dos *prompts*;
- Utilizar exemplos, aprendizagem em contexto;
- Dividir tarefas complexas em etapas mais simples;
- Cadeia de pensamento (CoT): Solicitar explicação dos passos que levaram à resposta;
- Diversificar as fontes de informação utilizadas para fundamentação dos fatos.

A escolha do Modelo Generativo para qualquer finalidade e contexto deve considerar diversos aspectos. A IBM, na publicação (IBM; ARMAND RUIZ; VIVEK BHARATHI, 2024), em seu blog institucional, apresenta um artigo informando que "[...] investigamos por que as escolhas do modelo básico são importantes e como elas capacitam as empresas a escalar a geração de IA com confiança." [1]. A empresa inicia o artigo indicado propondo que a escolha de modelos múltiplos de IA Generativa é importante porque "No mundo dinâmico da IA Generativa, abordagens únicas são inadequadas. À medida que as empresas se esforçam para aproveitar o poder da IA, é necessário ter um espectro de opções de modelos à sua disposição" [1], seguindo com a indicação de sete razões que tornam necessárias tanto a multiplicidade de modelos quanto o exercício da escolha:

- Estimular a inovação;
- Personalizar para obter vantagem competitiva;
- Reduzir do tempo de lançamento no mercado (*time to market*);
- Manter flexibilidade diante de mudanças;
- Otimizar custos em todos os casos de uso;
- Mitigar Riscos;
- Conformidade a Regulatórios.

Na continuidade do artigo, são sugeridas seis etapas para lidar com a escolha de um modelo para um caso de uso específico, considerando a opção de utilizar modelos específicos para casos de usos diversos:

1. Identificar claramente o caso de uso: determinar as necessidades e requisitos específicos para a aplicação para o negócio envolve a elaboração de *prompts* detalhados que considerem sutilezas inerentes para ajudar a garantir que o modelo esteja alinhado com os objetivos;
2. Listar todas as opções viáveis de modelo: avaliar vários modelos com base no tamanho, precisão, latência e riscos associados, incluindo entender os pontos fortes e fracos de cada modelo, como as compensações entre precisão, latência e taxa de transferência;
3. Avaliar os atributos do modelo: relevando a adequação do tamanho do modelo em relação às necessidades, considerando como a escala do modelo pode afetar o desempenho e os riscos envolvidos. Esta etapa se concentra no dimensionamento

[1] As citações referentes ao artigo (IBM; ARMAND RUIZ; VIVEK BHARATHI, 2024) são traduções e adaptações livres do autor.

correto do modelo para se ajustar ao caso de uso da melhor forma, pois maior não é, necessariamente, melhor. Modelos menores podem superar os maiores em domínios e casos de uso direcionados;

4. Opções do modelo de teste: Realize testes para ver se o modelo funciona conforme o esperado em condições que imitam cenários do mundo real. Isso envolve o uso de benchmarks acadêmicos e conjuntos de dados específicos de domínio para avaliar a qualidade da saída e ajustar o modelo, por exemplo, por meio de engenharia de *prompt* ou *fine-tuning*;
5. Refinar as seleções com base no custo e nas necessidades de implantação: após o teste, considerar fatores como ROI, custo-benefício e os aspectos práticos da implantação do modelo nos sistemas e infraestrutura existentes e outros benefícios, como menor latência ou maior transparência;
6. Escolher o modelo que oferecer mais valor: selecionar o modelo que melhor se adapte às demandas específicas do caso de uso com o melhor equilíbrio entre desempenho, custo e riscos associados.

Considerações semelhantes e com ênfases diversas podem ser encontradas em diversos contextos e com múltiplas abordagens similares ou distintas. De modo geral, as abordagens essenciais são encontradas na maioria dos casos e alguns são destacadas considerações especialmente relevantes para casos específicos ou culturas específicas.

Em todo caso, é relevante que a escolha do modelo seja criteriosa e clara em relação à sua aplicação, inclusive no quesito estratégico de usar múltiplos modelos, e sua conformidade às questões de custos e cultura da organização.

No bojo dos pontos mais elementares a considerar, dois aspectos proeminentes são (a) a finalidade para implementação da solução e (b) as características específicas do modelo.

Quanto às características específicas (b), a atenção recai sobre as implementações inerentes ao modelo, por exemplo, sobre conceitos, arquitetura, operação e controles de configuração, inclusive à luz das medidas que precisem ser adotadas para evitar as limitações acima e quaisquer outras inerentes ao modelo em consideração.

Sobre a finalidade para a solução (a), diferentes LLMs Generativos, são utilizados para aplicações distintas, como, por exemplo, documentação, tradução, assistência a escrita, pesquisa, conversação etc.

Enquanto este trabalho é escrito, um ponto de grande concorrência entre alguns dos mais avançados LLMs é a “troca de códigos” (*code-switching*), ou “mistura de códigos” (*code-mixing*) (HU, ZHANG e AL., 2023) (HIMASHI RATHNAYAKE, 2024), que implicam em

formas e habilidades do modelo em alternar, simultaneamente, entradas em uma língua, ou mais do que uma, e saídas em outra, ou outras, diferentes, sem perdas significativas de desempenho.

Cada uma das possíveis aplicações terá suas próprias necessidades e peculiaridades, portanto, potencialmente, demandará características e especialidades específicas do modelo considerado.

No caso de tratamento de códigos de computadores, fatores como as linguagens suportadas (C, C#, Java, Python, Go, Dart, ...), as tarefas a executar (como geração por instrução em linguagem natural, com ou sem código pré-existente, sugestão de continuação de código, correção, explicação, ajustes finos, conversões entre linguagens, comparações entre linguagens, algoritmos ou implementações, entre linguagens iguais ou diferentes, entre outras), assertividade sintática e lógica, complexidades diversas e modos de atualização e aprendizado, são exemplos de considerações relevantes e, até, mandatórias na escolha do modelo.

Apesar de, conforme a visão do senso comum, estarmos vivendo algo como a "infância da IA" (**Unsupported source type (DocumentFromInternetSite) for source Kar23.**), especialmente na geração e tratamento de códigos fonte de computadores, talvez, e até mesmo por isso, atualmente haja grande diversidade de modelos propostos para processar esses códigos, pelo menos em algumas das versões de determinados Modelos Base.

Entre os modelos dessa categoria são bem conhecidos GPT, Mistral, Gemma, Orca, Phi, StableLM e Vicuna, por exemplo. Cada um com suas particularidades, especialidades e limitações. Entre as características distintivas de alguns desses modelos, ser de código aberto e dispor de licenciamento gratuito para fins específicos ou amplos são diferenciais de importante consideração nas escolhas para implementações.

A disponibilidade de LLM Gen AI está evoluindo muito rapidamente, tanto no sentido de evoluções e versões dos que já são conhecidos e presentes há algum tempo quanto no lançamento de novos modelos. Esse comportamento tende a continuar por algum tempo, com, até mesmo, a possibilidade de ser incrementado. Entre outras implicações, é notável que as análises e conclusões deste, ou qualquer outro trabalho semelhante, precisa ser revisado em pouco tempo e as decisões tomadas precisam ser acompanhadas de políticas de atualização ou adequação claras e bem estruturadas. Mesmo enquanto essa pesquisa é realizada, novidades e perspectivas geram grandes volumes de informação (e desinformação), desafiando o foco.

Dos LLM de código aberto e licenciamento permissivo, com cláusulas de responsabilidade e restrições para evitar abusos, dois deles se destacam por suas próprias razões nesse final do primeiro semestre de 2024, o **Qwen** (**Unsupported source type (DocumentFromInternetSite) for source Ali24.**) e **Llama** (META, 2024).

Os dois modelos se destacam por atingirem desempenho e resultados, em regra, superiores aos demais na mesma categoria e muito próximos ou, eventualmente, superiores aos de opções pagas ou comercialmente restritivas. Corroborando as considerações sobre a evolução rápida dos modelos, note-se que a última versão do Llama foi lançada em 18 de abril de 2024 e o Qwen teve sua versão mais recente lançada em 06 de junho de 2024.

O modelo **Qwen**, é um modelo base ou de fundação (*Foundation Model*) desenvolvido pelo Alibaba Group. A versão **Qwen2** (ALIBABA GROUP - QWEN TIME, 2024) é uma evolução significativa da versão anterior (Qwen1.5), que impressiona por entrar em concorrência direta pelo título de SOTA[2] (Estado da Arte) com o modelo da Meta/Microsoft e os demais concorrentes ao título até então.

Com versões de tamanhos variados, para atender necessidades e finalidades diversas, a família de modelos Qwen, em sua configuração atual, apresenta as variações Qwen2-0.5B (com ½ bilhão de parâmetros), Qwen2-1.5B (com 1,5 bilhão de parâmetros), Qwen2-7B (com 7 bilhões de parâmetros), Qwen2-57B-A14B (com 57 bilhões de parâmetros), and Qwen2-72B (com 72 bilhões de parâmetros), incluindo algumas variações com designação *Instruct*, que foram refinadas para o tratamento de códigos de computadores, alcançando indicadores de desempenho semelhante aos dos mais eficientes modelos atuais.

A nova versão passou da compreensão de duas línguas (chinês e inglês) à compressão de mais 27 línguas e, em suas versões Qwen2-7B-Instruct and Qwen2-72B-Instruct, oferece suporte estendido para até 128 mil tokens de comprimento de contexto.

Outra importante evolução, relativa à arquitetura do modelo nessa nova versão, é a implementação da Atenção Agrupada de Consultas (GQA, *Group Query Attention*) (AINSLIE, LEE-THORP, *et al.*, 2023) em todas as suas variações, obtendo, em consequência, o aumento das velocidades de inferência e redução do consumo de memória, de modo que se torna mais acessível em termos de hardware e interessante para mais aplicações.

O modelo **Llama**, também um modelo de fundação, é desenvolvido em conjunto pelas empresas Meta e Microsoft. Em sua versão **Llama 3** (META, 2023), evolução da versão 2 lançada em 2023, com variações inicialmente pré-treinadas com diferentes números de parâmetros, 7 bilhões (7B), 13 bilhões (13B), 34 bilhões (34B) e 70 bilhões (70B). O modelo

[2] Acrônimo para **Estado da Arte** (**SOTA**: *State-Of-The-Art*). No contexto da IA, indica os modelos mais capazes para alcançar os resultados de uma tarefa, seja no Aprendizado de Máquina, Redes Neurais Profundas Processamento de Linguagem Natural ou de realização de tarefas genéricas. Para mais informações: (SRIVASTAVA, 2022).

base recebeu ajustes finos (*fine tuning*) (AUFFARTH, 2023, p. 225-256) para tratamento de códigos de computadores e essa especialização foi nomeada **Code Llama 3** (META, 2023).

O Code Llama 3 ainda foi adicionalmente treinado para refinar o tratamento de linguagem natural combinada com as tarefas relacionadas a códigos, evoluindo sua precisão e qualidade na geração de resultados mais alinhado com o que se espera com base nas instruções fornecidas, resultando em uma versão mais robusta e precisa que recebeu a designação distintiva de **Code Llama 3 - Instruct**.

Os modelos Llama 3 foram, desde o início, criados sob a abordagem de código aberto com flexibilidade acentuada com uso do método de *transformer* autorregressivo (**Unsupported source type (DocumentFromInternetSite) for source Bap23.**) (BASKARAN, 2023) sobre uma arquitetura de apenas decodificador (*decoder-only*) (**Unsupported source type (DocumentFromInternetSite) for source Zha23.**) (META, 2024), pré-treinado em um extenso corpus de dados auto supervisionados, usando a Aprendizagem por Reforço com *Feedback* Humano (RLHF) (AUFFARTH, 2023, p. 228, 229) (**Unsupported source type (DocumentFromInternetSite) for source Zha23.**), e a versão Code Llama Instruct pode ser baixada, refinada, retreinada, implantada através de diversos recursos e ambientes, além de possibilitar, se necessário ou útil, a personalização granular do próprio código fonte (META, 2023) (META, 2023), possibilitando maior liberdade e demandando controle fino e personalizado sobre questões como responsabilidade, aderência à cultura e negócio e flexibilidade de adaptação e evolução.

A proximidade dos indicadores de desempenho entre as versões atuais dos modelos Qwen e Llama, especialmente as de tamanho maior, é visível na mais recente avaliação apresentada pelo Hugging Face em seu *Open LLM Leaderboard* (FOURRIER, HABIB, *et al.*, 2024), indicada na Figura 1.

Figura 1 - Comparação de desempenho das maiores versões dos modelos Qwen2 e Llama3

T	Model	Average ↑	IFEval	BBH	MATH Lvl 5	GPQA	MUSR	MMLU-PRO
◆	Qwen/Qwen2-72B-Instruct	43.02	79.89	57.48	35.12	16.33	17.17	48.92
○	meta-llama/Meta-Llama-3-70B-Instruct	36.67	80.99	50.19	23.34	4.92	10.92	46.74

Fonte: (FOURRIER, HABIB, *et al.*, 2024)

A escolha entre Qwen2 e Llama3 precisa considerar detalhes de implementação específicos, entre eles o ‘tamanho’ do modelo, sendo referência à quantidade de parâmetros. No caso das maiores variações de cada um, Qwen2-72B e Llama3-70B, considerando a utilização em tratamentos de código de computadores há um detalhe distintivo que pode pender em favor do modelo da Meta.

Ocorre que, embora sejam modelos de código aberto, por decisão estratégica da empresa, o Qwen2-72B mantém o uso da licença Qianwen original, enquanto as demais variações do modelo adotaram a Licença Apache 2.02. A Qianwen é uma licença mais restritiva do que a Apache, exigindo, por exemplo, que o código fonte alterado permaneça publicamente disponível ainda que possa utilizar uma licença diferente. Então é importante avaliar os impactos dessas restrições ao considerar a opção pelo modelo Qwen2, na versão 72B, inclusive nas variações *Instruct*.

O modelo selecionado pode gerar os novos códigos com base em seu pré-treinamento, em retreinamentos (ajuste fino, *fine-tuning*), nos dados de entrada providos pelos agentes e nas instruções (*prompts*) adicionadas para orientar os tratamentos que o modelo deve providenciar para as informações recebidas. Em seguida será necessário armazenar os resultados.

2.3 Saídas: resultados

Ao utilizar Modelos ou Agentes Conversacionais em linguagem natural, como um chat, por exemplo, é esperado, normalmente, que o resultado, ou saída, do modelo seja parte de um diálogo. Portanto, um interlocutor, por princípio, é assumido. Por outro lado, em uma aplicação de busca ou pesquisa, essa mesma postura não é igualmente assumida, já que, nesse caso, tanto pode estar ocorrendo uma sessão interativa quanto pode se tratar de uma operação em lote ou agendada, em consonância com os requisitos e modos de uso da aplicação.

O caso específico de uso precisa estar claro desde o princípio da preparação de um serviço, especialmente no que tange à escolha do modelo, ou modelos quando for o caso. As etapas seguintes também manterão em vista os requisitos e parâmetros necessários ao caso de uso. Porém é notável que as fases finais e as saídas (resultados) do(s) modelo(s) são, normalmente, aspectos mandatórios em todo o ciclo, desde o início. Assim, o fim (resultado), pode-se dizer, define o princípio. Afinal, o objetivo é alcançar resultados de qualidade em todos os sentidos possíveis.

Com isso não se considera que qualquer etapa seja menos importante, nem que seja a etapa final que deva receber maior atenção. De fato, um conceito amplamente lembrado em contextos de tratamento e análise de dados é que “se entra lixo, sai lixo” (muito provavelmente). Assim como um processamento inadequado ou com parâmetros insuficientes ou de baixa qualidade tende a resultar em saídas indesejáveis.

Dito isso, de um modelo de IA que analisa códigos, especialmente à luz de eventos e relações externas ao próprio código, para gerar códigos que sejam ajustes, correções e

evoluções dos analisados, espera-se que o resultado seja código de qualidade minimamente superior ao original, ainda que a superioridade possa ser circunstancial ou contextual.

Chegado a esse ponto, a questão a avaliar é se o código gerado, além de sua qualidade nos termos abordados, acrescenta valor suficiente ao substituir o anterior de modo que ‘valham a pena’ os esforços e custos dessa alteração no contexto e momento em que se tornam disponíveis ou se, eventualmente, a despeito dos avanços inerentes, venha a ser mais oportuno aguardar melhores ou maiores evoluções para se decidir pela atualização.

Para que essas análises sejam efetuadas e as decisões sejam assertivamente tomadas, os códigos, com as devidas indicações das alterações e suas pertinências, devem ser disponibilizados em repositórios definidos e preparados para esses fins e configurados de maneira a possibilitar tanto a sua eventual publicação quanto seu arquivamento, sendo preterido por futuras opções mais aderentes às visões e intenções dos responsáveis por tais decisões, mesmo que elas sejam, futuramente, automatizadas.

Como, eventualmente, algumas recomendações serão publicadas e outras não serão, ao persistir as recomendações de códigos nos repositórios, porém questões de versionamento devem ser tratadas adequadamente ao serem disponibilizadas, exigindo tratamentos específicos e especializados para cada aplicação e para cada caso ou estado anterior ou previsto.

Em todo caso, para que esses novos códigos se tornem disponíveis nos repositórios em que devam ficar, o uso de Agentes de IA especificamente especializados e habilitados para tais tarefas possibilita a realização controlada e eficiente desse objetivo, assim como do envio de notificações e avisos nos casos em que sejam necessários.

Ao fazer referência a que os agentes sejam especializados, possivelmente seja claro o sentido em que cada agente tenha os conhecimentos necessários para realizar sua tarefa específica.

Quanto à referência a serem habilitados, vale esclarecer que, para executarem suas funções, os agentes também precisam ser reconhecidos e ter quaisquer ‘credenciais’ necessárias tanto para as autenticações quanto para as autorizações pertinentes às suas execuções e integrações internas e/ou externas (chaves, tokens, senhas, IP etc.), consideradas as respectivas questões de segurança, compliance, políticas, armazenamento e utilizações, entre outras.

Sobre essas referências, especializações e habilitações, deve-se ter em mente que se aplicam a todos os agentes em seus próprios contextos e funcionalidades, tanto esses que atuem com as saídas/resultados, quanto com aqueles anteriormente citados no contexto de entradas e quaisquer outros em suas condições particulares.

3 METODOLOGIA

Para a realização do trabalho proposto, algumas das definições metodológicas e conceituais adotadas são esclarecidas nesta sessão, para propiciar uma visão suficientemente clara do processo e eventuais ajustes necessários.

3.1 Tipo de pesquisa e momento das definições

Com a percepção da possibilidade de usar a IA como tutor dos códigos das aplicações de sistemas corporativos, publicadas em ambiente de produção, continuamente atualizadas, evoluindo e nas suas melhores condições operacionais, de modo contínuo, automático e no tempo mais breve possível e viável, este trabalho assume as características de uma pesquisa experimental.

As definições seguintes foram pautadas sobre os dados e ambiente acessíveis para implementar o ambiente funcional para realizar os experimentos e nas estimativas plausíveis para os prazos, disponibilidades e custos considerados exequíveis a partir de junho de 2024, momento em que se completa o planejamento das atividades.

3.2 Localização espacial e temporal do ambiente e dados iniciais

O estudo será realizado sobre o ambiente de produção de um banco digital privado, especializado em investimentos, em São Paulo, capital, que passa a ser referido como “Banco”.

Entre os códigos das aplicações e serviços em operação no ambiente de produção do Banco serão selecionados como objeto desse estudo 1.200 aplicações/serviços, entre APIs, *Background Services/Workers*, *Hubs* de *Web Sockets* (SignalR), Funções sem Servidor etc., incluindo todas as suas dependências, componentes, bibliotecas etc., os quais serão a base principal de refinamento (*fine-tuning*) para o Modelo Central Gen AI do projeto e formarão o contexto para as experimentações e suas conclusões.

Esses códigos estão escritos em diversas linguagens, usando diferentes frameworks, componentes e tecnologias variadas são executadas, principalmente, em Clouds (Nuvens), majoritariamente na Amazon Cloud, diversas instâncias em Microsoft Azure e GCP (Google Cloud Platform) e outras poucas instâncias *on-premisse*.

Atualmente, há aplicações/serviços, e bases de dados, considerados legados, que mantém arquiteturas, tecnologias, frameworks, componentes e versões, especialmente de linguagens de programação mais antigas, tanto em função de fusões e aquisições quanto de

desenvolvimentos de integrações. Considera-se que, proporcionalmente, esses casos correspondam a 45% da base operacional.

Conforme registros do Dynatrace, nos últimos 12 meses, esse contexto demanda, aproximadamente, as seguintes médias mensais dos principais grupos de processamento:

Tabela 1 - Quantidades médias de Grupos de Processamento das principais tecnologias relacionadas aos códigos selecionados

Tecnologia	**Quantidade de Grupos de Processos**
.NET	100.000
ASP.NET	230
Kubernetes	76.000
GO	61.000
Python	70
Windows	190
Linux	13.800
Nginx	3.600
Apache Tomcat	1.600
Apache HTTP Server	100
IIS	180
Node.js	3.100
Java	1.900
Erlang	70
RabbitMQ	6.800
Amazon SQS	700
Apache Kafka	160
Redis	140
MongoDB	600
Docker	500
Amazon S3	220
Amazon API Gateway	120
Oracle Database	40
MS SQL Server	20
PostgreSQL	40
MySQL	10
Netty	250
Jetty	400

Fonte: **Dynatrace** – dados coletados pelo autor.

Os códigos selecionados incluem aplicações/serviços de *frontend* (tradicionais e micro *frontends*), *mobile* e *backend*, incluindo microsserviços, monolitos, funções sem servidor, BFF (*Backend for Frontend*) e outras arquiteturas e estruturas, formando a plataforma do banco que

atende clientes, Pessoas Físicas (PF) e Pessoas Jurídicas (PJ), tais como Correntistas, Investidores, Corretoras de Investimentos, Assessores Internos e Externos, entre outros.

A Plataforma do Banco registra variações de alguns milhares a dezenas de milhares de acessos únicos diários, que ocorrem em patamares sazonais e pontuais com médias ascendentes, especialmente nos últimos dois anos.

3.3 Fluxo e cronograma

Em uma visão panorâmica, as etapas seguintes resumem a jornada planejada para o experimento:

1. Preparação do ambiente e recursos iniciais;
2. Disponibilização do Modelo Central pré treinado no Microsoft Azure;
3. Obtenção dos códigos nos repositórios Azure DevOps originais;
4. Geração dos *forks* respectivos também no Azure;
5. Vetorização (*embeddings*) dos códigos do *forks*;
6. Disponibilização do Banco de Dados Vetorial no Azure;
7. Armazenamento dos códigos na base vetorial;
8. Submissão da base vetorial para conhecimento do Modelo Central;
9. Obtenção dos particionamentos de dados para treino e validação do modelo;
10. Execução do treino e validação para refinamento do modelo;
11. Execução dos ciclos de testes e ajustes do modelo refinado;
12. Disponibilização dos Agentes de Entrada e Saída;
13. Avaliações e ajustes do processo previsto;
14. Análise e síntese dos resultados e conclusões;
15. O Encerramento do estudo.

As etapas 1 a 11 serão realizadas, principalmente, no ambiente Azure Machine learning (AML), com utilização de seus recursos de programação e execução, especialmente os notebooks ali disponíveis. Embora seja o conjunto com maior número de etapas, os códigos e insumos preparados e experimentados com antecedência deverão propiciar a execução objetiva em um menor período nas 9 primeiras etapas e maior na 10ª e 11ª.

A etapa 12 disponibilizará os agentes no ambiente Azure, orquestrados pelo LangChain, com base nos códigos já experimentados com scripts e notebooks nas etapas anteriores, automatizando o processo para sua sequência em regime operacional.

As etapas 13 a 15 concluirão o experimento com as análises pertinentes e conclusões.

O cronograma previsto deverá seguir o seguinte esquema geral:

- Etapas 1 a 9: dias 1 a 3;
- Etapas 10 e 11: dias 4 a 29;
- Etapa 12: dias 30 e 31;
- Etapas 13 a 15: dias 32 a 35.

3.4 Dados de entrada e Modelo Central

Um *fork* (réplica integral independente) dos repositórios dos códigos das 1.200 aplicações/serviços selecionados será disponibilizado em um novo contexto de projeto com acesso protegido e limitado à IA, seus agentes e os desenvolvedores e administradores do composto de IA, para receber as recomendações e atualizações recomendadas pelo Modelo Central, através de seus agentes, em *branches* cronologicamente identificados e criados a partir do *branch* principal (*main*) do *fork*.

A partir do *fork*, os códigos e demais informações disponíveis nos repositórios serão armazenadas em uma base de dados vetorial para o refinamento inicial e atualizações do Modelo Central.

Para armazenamento dos dados vetoriais, será utilizada uma instância do Banco de Dados Vetorial Weaviate (WEAVIATE, 2024) (WEAVIATE, 2024), criado nativamente para IA, de código aberto e utilização gratuita para estudos e pesquisa.

Os scripts necessários para as execuções dessas atividades, credenciais inerentes e recursos de apoio foram preparados e testados em contextos e simulações locais durante o processo de planejamento, restando editar as referências e endereços que somente estarão disponíveis a partir da etapa 1. Essas mesmas preparações foram realizadas a respeito de todas as etapas de 1 a 11.

Após o refinamento do Modelo Central e seus agentes, a ser concluído com os testes da etapa 11, alterações que ocorrerem nos repositórios originais serão atualizados no conhecimento do Modelo Central, pelos agentes no *branch* principal (*main*) do *fork*, que será o único *branch* considerado atual e válido, para a IA, embora, para fins de aprendizado e referência, todos os *branches* originais, com todos os seus *commits* registrados, serão incluídos no *fine-tuning* inicial do modelo.

O conhecimento do Modelo Central a respeito dos códigos será enriquecido e ajustado sempre que necessário por *prompts* de instruções, que esclarecerão os aspectos evolutivo e progressivo dos *branches*, *commits*, versões e suas cronologias, com especial ênfase nas

indicações dos *pipelines*, *builds*, *rollbacks* etc., conforme registrados nas ocorrências dos seus eventos.

Juntamente com os códigos das aplicações e serviços, providos inicialmente na base vetorial, os dados históricos de observabilidade e logs, obtidos sobre as execuções das aplicações/serviços do contexto e suas relações com seus ambientes de execução e outras aplicações no contexto e/ou externas a ele, serão utilizados no processo de *fine-tuning* inicial do Modelo Central para refinar, validar e testar o entendimento do processo de análise a partir desses dados.

Esses dados foram registrados no primeiro semestre de 2024 e serão particionados em:

- Dados de treinamento: referentes ao primeiro quadrimestre; e
- Dados de validação: referentes ao terceiro bimestre.

Após a geração e tratamento inicial dos *forks*, nas etapas 4 e 5 e a conclusão do refinamento propriamente dito, conforme abordado nas etapas 8 a 10, durante as quatro semanas seguintes, diariamente, serão obtidos os dados de observabilidade e logs pertinentes, que serão usados para realização, no dia seguinte, de testes com o modelo (Dados de teste), com ajustes de hiperparâmetros, prompts e demais necessidades percebidas, de modo que, ao final de cada semana, 7 dias corridos, sejam realizadas avaliações, revisões e eventuais realinhamentos do processo e/ou do modelo.

O aspecto da cronologia dos registros de observabilidade e logs é especialmente significativo, assim como a relação de cada registro/evento com cada aplicação específica e os trechos de código especificamente em execução durante as ocorrências.

Portanto, as informações entregues ao Modelo Central para análise serão estruturadas com base nesses aspectos, de modo a terem as relações devidamente vinculadas com as execuções do código e encadeados logicamente para preservação da integridade relacional, inclusive, de causa e efeito, considerando cada aplicação/serviço individualmente e suas interações com o ambiente (contêiners, *clusters*, memória, processador, rede etc.) e com outras aplicações/serviços, seja como consumidores, seja como servidores, seja como pares e/ou dependências

A cronologia, como aspecto relevante das informações de observabilidade e logs, implica, especialmente, no particionamento dos dados para refinamento do Modelo Central, indicando como critério mais apropriado uma divisão temporal do que uma aleatória, de modo que os conjuntos de Treino, Validação e Testes sejam escolhidos, respectivamente, com base na sua ordem e momento de geração e referenciados adequadamente a cada aplicação/serviço.

O modelo Llama3.1-70B-Instruct, versão mais recente do modelo Llama a essa altura do trabalho, foi escolhido como Modelo Central, para ser executado no ambiente Microsoft Azure por sua proximidade com a infraestrutura em uso no Banco, especialmente o Azure DevOps.

Os dados relativos às operações das aplicações em ambiente de produção, e sobre o ambiente de produção em si, serão coletados e disponibilizados para a IA com proteção por anonimização e ofuscação, no que tange a informações sensíveis ou internas. É relevante pontuar que os dados e informações, em nenhum momento, serão usados ou apresentados em qualquer ambiente externo aos do Banco.

Os dados de treino, validação e testes serão buscados no Dynatrace e no Kibana pelos códigos dos notebooks, e, posteriormente, pelos agentes orquestrados por LangChain, com base nos namespaces das aplicações/serviços e pelas designações a eles atribuídas em seus *pipelines* de publicação, trazendo, também os registros/eventos relacionados às suas execuções e às chamadas recebidas ou efetuadas, de e para os outras aplicações/serviços com que se relacionem, conforme sejam registradas no código e nas informações de observabilidade e logs do Dynatrace e do Kibana.

Sendo conhecidas e reconhecidas pelos códigos dos agentes, essas referências poderão ser vinculadas trechos específicos dos registros das execuções dos códigos vetorizados como conhecimentos a partir dos repositórios dos seus *forks* e, a partir daí, relacionados para análises que evidenciem os ajustes, correções, evoluções e aperfeiçoamentos.

3.5 Dados de Saída e resultados

O Modelo Central, com base na análise dos códigos reconhecidos e aprendidos a partir da base vetorial a que foram adicionados advindos dos *forks* dos repositórios originais, e, posteriormente, de suas atualizações, em comparação com as informações de observabilidade e logs, recebidas do ambiente de produção pelo Dynatrace e Kibana, deverá gerar códigos completos, que serão adicionadas aos mesmos repositórios respectivos, através dos agentes especializados para esse fim, na forma de novos *branches* criados a partir dos *branches main* de cada repositório dos códigos selecionados e, a partir desses novos branches, os agentes deverão, também, gerar os respectivos *pull-requests* (PR) para sua promoção à branch *main* pertinente.

Depois de gerar recomendações/código para um repositório, o Modelo Central deverá voltar a gerar novas recomendações para o mesmo repositório apenas se houver diferenças substanciais não indicadas anteriormente ou se for instruída a ignorar as recomendações anteriores.

Os agentes deverão, também, enviar notificações e alertas aos responsáveis pelos repositórios em que ocorrerem alterações.

Os testes buscarão nos códigos gerados por faltas de pertinência e/ou valor nas recomendações e códigos gerados, assim como inconsistências, incompletudes, alucinações, imprecisões, omissões e/ou erros efetivos de lógica, sintaxe ou padrões reconhecíveis, além daqueles perceptíveis através da execução de testes unitários, de integração e/ou de regressão ou estresse, conforme existam previamente, sejam evoluídos e/ou adicionados/criados como parte dos processos criativos da própria IA.

Com base nos resultados analisados nos testes será tomada a decisão de indeferir a recomendação, ajustar os códigos ou aprovar os PR gerados pelos agentes do Modelo Central, após a geração dos *branches* respectivos, serão tomadas, de modo análogo ao que será feito, posterior e eventualmente, no caso de se prosseguir com a utilização do composto de IA.

Adicionalmente, quando novas versões das linguagens, frameworks, componentes e demais tecnologias forem publicadas, o modelo deverá ser atualizado com o conhecimento dessas novidades, através de agentes especializados para essas tarefas, assim como deverá ocorrer, também, para os casos de adoções de novas tecnologias nas alterações dos códigos adidos ao contexto de conhecimento da IA.

4 PROPOSTA E DESENVOLVIMENTO

Nesta sessão é apresentado um resumo da execução de cada etapa do trabalho com comentários pertinentes para o entendimento da experiência.

4.1 Etapas 1 a 9

Nas etapas iniciais foi liberado o acesso ao ambiente AML, tendo os scripts e códigos disponíveis para a preparação do ambiente, os procedimentos partiram do aprovisionamento de:

- Duas máquinas virtuais (VM: *Virtual Machine*):
  - 1 VM Standard_NC6:
    - 6 vCPUs (Virtual Central Process Unity);
    - 56 GB RAM (Giga Bytes de Random Access Memory);
    - 1 GPU NVIDIA Tesla K80 (GPU: Graphic Process Unity);
    - Para utilização nos refinamentos do Modelo Central;
  - 1 VM Standard_D4_v3:
    - 4 vCPUs;
    - 16 GB RAM;
    - Para utilização em processamentos e tarefas gerais;
- Uma unidade SSD (Solid Disk State) de 1 TB (Tera Bytes), para armazenamento de arquivos, recursos e transferências de dados.

Um espaço de trabalho (*workspace*) foi criado no AML para execução dos procedimentos necessários e o modelo Llama-3.1-70B-Instruct foi registrado no *workspace*.

Um novo Projeto foi criado no Azure DevOps e os forks dos repositórios originais selecionados foram adicionados a esse novo Projeto, usando uma lista dos repositórios selecionados e a um script Python executando subprocessos com comandos do Azure CLI. Esta atividade demandou quase 10 horas.

O Banco de Dados Vetorial Weaviate foi instalado no Azure Kubernetes Service (AKS), a partir do GitHub (https://weaviate.github.io/helm-charts/), usando o Helm Charts.

A classe ‘CodeDoc’ foi criada para estruturar os códigos e informações adicionais vetorizadas no Weaviate.

Foi iniciado o script Python escrito para (a) vetorizar os códigos e adicionar os *embeddings* à base Weaviate e (b) carregá-los no *workspace* AML para utilização pelo Modelo Central. São previstas 42 horas para a sua conclusão, executando assíncrona e paralelamente as duas atividades (a e b) com (b) sendo iniciada 6 horas após (a).

A monitoração dos processos de vetorização e disponibilização dos embeddings receberam o foco até serem concluídos com sucesso, sendo seguidos pelo início do treinamento do Modelo Central com os *embeddings* dos códigos paralelamente com o começo da obtenção das informações de observabilidade e logs para a segunda etapa do refinamento do modelo, com o treinamento e validação usando essas informações.

Com a conclusão dessas duas últimas atividades iniciadas, encerraram-se as etapas de 1 a 9 e foi iniciada a segunda parte do refinamento do Modelo Central, com previsão de encerramento em 24 horas.

4.2 Etapa 10

Concluída pouco depois do previsto, a validação com o particionamento do terceiro bimestre de dados de observabilidade e logs, encerrando a etapa 10 da jornada desse estudo. Observados os resultados dessa etapa, foram registrados os seguintes parâmetros:

Desempenho	Acurácia Geral: 82% Precisão: 80% Recall: 83% F1-Score: 81,5%
Correlação de Dados	O modelo registrou correlação forte entre os logs de erros seções de código, identificando corretamente padrões de falhas em 83% dos casos analisados.
Consistência	A validação mostrou que o modelo mantém consistência na análise de dados, mesmo quando exposto a variações nos padrões de logs.
Robustez	Até onde foi possível perceber, pequenas variações nos dados de entrada não afetaram significativamente o desempenho do Modelo Central, indicando uma robustez adequada para operações em ambientes de produção.

Com base nos resultados observados, o modelo aparenta estar em condições para iniciar a fase de testes diários com dados coletados do ambiente de produção.

4.3 Etapa 11

No início da Etapa 11 foi executado o script que obtém as informações relativas ao dia anterior e os disponibiliza para o Modelo Central realizar as análises.

Os testes são analisados por amostragem, aleatoriamente em termos individuais. Durante essa etapa de teste, o Modelo Central deverá gerar recomendações a cada 5 dias para as aplicações/serviços em que os critérios previamente estabelecidos sejam atendidos, gerando os códigos que serão adicionados aos repositórios pertinentes como novos branches com os PR respectivos, ambos providenciados, posteriormente, por agentes especializados e, nesta etapa, por scripts e notebooks com execução agendada.

Durante os testes e, especialmente, após cada geração de códigos, ajustes e correções poderão ser efetuados visando o aperfeiçoamento do Modelo Central, dos agentes e do processo.

Nos dias de teste sem pontos notáveis apenas uma nota "os testes foram bem-sucedidos" será registrada, deixando espaço para anotações mais significativas sempre que útil ou necessário. Nos dias em que houver geração de códigos haverá maiores informações.

Nesse primeiro dia nenhum destaque pareceu necessário, então, os testes foram bem-sucedidos.

Analisando os dados dos **primeiros 6 dias** os testes foram bem-sucedidos.

Na análise dos dados do **sétimo dia** foram verificados tempos aumentados nas obtenções das informações dos dados de observabilidade e logs. Uma análise mais detalhada deu a perceber que um aumento significativo do tempo das respostas às APIs do Dynatrace e do Kibana e as ocorrências de diversos eventos e ações de resiliência causados por timeouts frequentes, mas não foram encontradas evidências que indicassem problemas com relação ao processo de coletas de informação ou análises.

A análise dos dados do **oitavo dia** foi a primeira em que ocorreu a geração de códigos. Das 1.200 aplicações/serviços em observação, 63 novos *branches* foram gerados com as seguintes considerações:

- No caso de 3 desses novos branches, havia falhas graves, como trechos faltantes, erros de má formação de textos e dois deles não puderam ser compilados.
- Em 14 branches as alterações foram apresentadas exclusivamente críticas sobre práticas pouco recomendáveis (*code smell*), de modo geral, relevantes, mas que, aparentemente, não significavam problemas sérios.

- Em 22 dos 63 novos *branches* indicavam alterações relevantes, porém em aplicações legadas, que apresentavam elevada complexidade para alterações e já tinham previsão de substituição em médio prazo.
- Outros 15 branches identificaram questões de componentes com vulnerabilidades documentadas, de fato, precisam de atenção em curto prazo, embora representassem riscos menores para impactos eventuais.
- Nas 9 restantes foram identificadas questões pertinentes a respeito de melhorias de desempenho ou estabilidade, destacando-se como as contribuições mais importantes entre os 63 casos.

Para aprimorar as análises do Modelo Central um arquivo com esclarecimentos e orientações relativos aos pontos levantados foi elaborado e disponibilizado para o Modelo através da base vetorial, a fim de servir de referência para alinhar os comportamentos menos desejados e algumas indicações foram adicionadas aos prompts de instrução.

Com os dados do **nono ao décimo segundo dias** os testes foram bem-sucedidos, embora possa ser pontuado que o nono dia se tratou de um feriado estadual em São Paulo, mas não se registrou qualquer variação respectiva.

No dia **décimo terceiro dia** ocorreu a segunda geração de códigos. Nessa ocasião 3 novos branches foram adicionados aos repositórios pertinentes. Todos os casos foram significativos, sendo que em um deles se observou elevação irregular e de retenção de sockets sem utilização por implementação inadequada no uso de clientes do protocolo HTTP.

Em retorno para o Modelo Central, a questão foi marcada como de alta criticidade para manter sob atenção outros casos semelhantes.

Com os dados do **décimo quarto ao décimo sétimo dia** os testes foram bem-sucedidos, porém com percepção de lentidão muito acima do normal nas execuções dos procedimentos.

Com os dados do **décimo oitavo dia**, quando deveria ter ocorrido a terceira geração de códigos no período, não foi gerada qualquer recomendação/*branch*. Após verificações e análises dos registros de execução, constatou-se que alguns ajustes nos prompts de instruções, aparentemente, criaram situações inadequadas para as análises, portanto demandaram ajustes que foram implementados.

As análises dos dados do **décimo nono ao vigésimo segundo dia** fluíram de modo que os testes foram bem-sucedidos.

Com os dados do **vigésimo terceiro dia**, ocasião da quarta geração de códigos prevista, 11 branches foram geradas com sucesso e, embora, pequenas e objetivas, as mudanças indicadas foram todas significativas e um aspecto que chamou a atenção é que, dos 11 casos, 9 foram

relacionados a serviços e os problemas assinalados eram relacionados a baixa eficiência causada pelo tratamento de encadeamentos de chamadas síncronas e assíncronas. Diante do desempenho verificado, nenhuma alteração pareceu necessária a essa altura.

Com os dados do período do **vigésimo quarto ao vigésimo sétimo dia** os testes foram bem-sucedidos.

Os dados do **vigésimo oitavo dia** foram os últimos da etapa de testes previstos para gerar códigos. Na ocasião, apenas 2 branches foram gerados e, embora as indicações fossem relevantes, não representaram questões a ter em alta consideração por tratarem de questões relativas a formatos de escrita de código que entraram em desuso nas versões mais recentes e as aplicações em foco se encaixam entre os legados previstos para atualizações.

O **vigésimo nono dia** foi dedicado para revisões dos resultados dos testes anteriores e preparações para as próximas etapas.

4.4 Etapa 12

Os scripts e notebooks em utilização nas etapas anteriores cumpriam as funções que, efetivamente, devem ser devem ser executadas por agentes especializados sob orquestração do LangChain, conforme definições para o estudo em sua completude. Sendo assim, os dois dias seguintes (trigésimo e trigésimo primeiro) foram dedicados a converter aqueles instrumentos provisórios, que propiciavam tanto acompanhamento granular das execuções quanto evoluções pontuais e controladas dos códigos, mas que, em sua forma prevista ganham em desempenho, autonomia e com as características da orquestração mais elaborada.

O processo em si foi simples e rápido pela aderência dos padrões utilizados na codificação e pela simplicidade oferecida pelo orquestrador. Realizadas as conversões e publicadas as novas ferramentas, nos dias seguintes (etapas 13 e 14) tanto os novos agentes quanto o Modelo Central estiveram sob acompanhamento e avaliações visando a conclusão apropriada do experimento.

4.5 Etapa 13 a 15

Nos quatro dias das Etapas 13 a 15 poucos ajustes foram realizados quanto a orquestração dos novos agentes. No demais, transcorreram sem novidades.

5 ANÁLISES DE RESULTADOS

Revendo e analisando os resultados obtidos a partir da implementação da inteligência artificial no ambiente de produção de um banco digital privado, vale estruturar a análise em torno de três eixos principais: eficiência operacional, qualidade do código gerado e impacto nos processos de desenvolvimento. Cada um desses aspectos é crucial para entender o valor agregado pela IA ao contexto.

5.1 Eficiência operacional

A implementação de agentes de IA especializados resultou em melhorias significativas na eficiência operacional, uma vez que tem capacidade para reduzir, significativamente, o tempo de resposta às necessidades de ajustes, correções, evoluções, atualizações e adequações dos códigos em operação no ambiente de produção, o que pode ser atribuído à capacidade da IA de monitorar continuamente o ambiente e identificar essas necessidades, recomendando as alterações em formato de código pronto ou muito próximo disso.

5.2 Qualidade do código gerado

De modo geral, embora ainda seja necessário manter bastante atenção aos códigos gerados pela IA, as peças fornecidas, minimamente, aceleram e facilitam a localização e a implementação de soluções, além de apontarem os problemas em si ou sua iminência e eventuais consequências.

Os desenvolvedores são beneficiados porque a IA não apenas produziu códigos funcionais, mas também ajudou a identificar e corrigir vulnerabilidades e ineficiências que poderiam ter passado despercebidas em outras formas de revisões.

5.3 Impacto nos processos de desenvolvimento

A análise indica que a IA proporciona suporte aos desenvolvedores, automatizando tarefas repetitivas e permitindo que o foco fosse direcionado a atividades mais estratégicas.

Os desenvolvedores podem, assim, dedicar mais atenção à inovação e ao desenvolvimento de novas funcionalidades, em lugar de se concentrarem em tarefas de manutenção e buscas de problemas nem sempre evidentes ou fáceis de localizar.

6 CONCLUSÕES

As conclusões deste estudo indicam o impacto positivo da IA na gestão de códigos em ambiente de produção, evidenciando melhorias significativas em eficiência operacional e na qualidade do software.

A capacidade da IA de detectar e corrigir problemas de forma autônoma resulta em uma operação mais estável e confiável. A análise sugere que a integração contínua de IA em processos operacionais pode se traduzir em economias significativas, elevação de eficiência e melhor alocação de recursos, implicando em melhoria da eficiência operacional.

Sobre a qualidade do código e segurança, a capacidade da IA de identificar vulnerabilidades e sugerir melhorias é extremamente vantajosa em um contexto em que a segurança e a conformidade são prioridades. Recomenda-se o aprimoramento do uso da IA e a conformidade com padrões que facilitem e expandam a sua participação no SDLC, especialmente nos ambientes de produção.

Consideradas as experiências vivenciadas durante a execução deste trabalho, para maximizar os benefícios alcançáveis com a tutoria da IA sobre os códigos em operação no ambiente de produção, é recomendável investir em treinamento para as equipes, propiciando aos desenvolvedores melhores possibilidades de capacitação para trabalhar em conjunto com tecnologias de IA. Além disso, a criação de uma cultura de inovação e experimentação pode facilitar a adoção de novas ferramentas e processos, promovendo um ciclo de melhoria contínua.

Por fim, este estudo demonstrou que a IA pode ser proficiente em evoluir a eficiência, a qualidade e a satisfação das equipes internas, clientes e parceiros. As observações sugerem que, com a implementação adequada, a IA pode facilitar o futuro do desenvolvimento e a manutenção das aplicações/serviços em ambientes produtivos.

REFERÊNCIAS

AINSLIE, Joshua *et al.* GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. **Google Research**, 23 dez. 2023.

ALIBABA GROUP - QWEN TEAM. Welcome to Qwen! **Read The Docs**, fev. 2024. Disponível em: https://qwen.readthedocs.io/en/latest/. Acesso em: 29 nov. 2024.

ALIBABA GROUP - QWEN TIME. Hello Qwen2. **GitHug - Qwen2**, 2024. Disponível em: https://qwenlm.github.io/blog/qwen2/. Acesso em: 29 nov. 2024.

ARSANJANI, Ali. Navigating the Challenges of Hallucinations in LLM Applications: Strategies and Techniques for Enhanced Accuracy. **Medium**, 2023. Disponível em: https://dr-arsanjani.medium.com/navigating-the-challenges-of-hallucinations-in-llm-applications-strategies-and-techniques-for-ab2b5ddc4a63. Acesso em: 22 jun. 2024.

ÅSNE STIGE, Yuzhen Z. P. M. E. D. Z. Artificial intelligence (AI) for user experience (UX) design: a systematic literature review and future research agenda. **Information Technology & People**, Sheffield, 29 ago. 2023.

AUFFARTH, Ben. **Generative AI with LangChain:** Build large language model (LLM) apps with Python, ChatGPT and other LLMs. Birmingham: Packt Publishing, 2023.

AXELTON, Karen. STATE OF ARTIFICIAL: With AI In Its Infancy, B2B Orgs Begin To Prioritize Early Adoption. **Demand Gen Report (DGR)**, abr. 2023. Disponível em: https://www.demandgenreport.com/resources/state-of-artificial-intelligence-with-ai-in-its-infancy-b2b-orgs-begin-to-prioritize-early-adoption/7834/. Acesso em: 22 jun. 2024.

BASKARAN, Saravanan H. A Comparison of Transformer and Autoregressive LLM Designs. **International Journal of Research Publication and Reviews, Vol 4, no 11**, nov. 2023. 19-26.

BRQ. Observabilidade: o que é, desafios e ferramentas, dez. 2023. Disponível em: https://blog.brq.com/observabilidade. Acesso em: 29 nov. 2024.

CODECOMPLETE. AI-Powered DevTools for Enterprise, fev. 2022. Disponível em: https://codecomplete.ai. Acesso em: 29 nov. 2024.

DYNATRACE. The world needs software to work perfectly, ago. 2017. Disponível em: https://www.dynatrace.com/company. Acesso em: 29 nov. 2024.

DYNATRACE. Meet Davis, our powerful AI-engine, ago. 2021. Disponível em: https://www.dynatrace.com/platform/artificial-intelligence. Acesso em: 29 nov. 2024.
DYNATRACE. API versions. **Developer Dynatrace**, 2024. Disponível em: https://developer.dynatrace.com/platform-services/general/versioning/. Acesso em: 29 nov. 2024.

DYNATRACE. Davis AI service. **Davis AI service**, 2024. Disponível em: https://developer.dynatrace.com/platform-services/services/davis-analyzers/. Acesso em: 29 nov. 2024.

DYNATRACE. Dynatrace App-Toolkit. **Dynatrace App-Toolkit**, 2024. Disponível em: https://developer.dynatrace.com/reference/app-toolkit/. Acesso em: 29 nov. 2024.
ELK. Elastic Stack. **Elastic Stack**, 2024. Disponível em: https://www.elastic.co/pt/elastic-stack. Acesso em: 29 nov. 2024.

ERBEL, J., G. J. Scientific workflow execution in the cloud using a dynamic runtime model. **University of Goettingen - Softw Syst Model, 23**, Goettingen, Germany, 2024. 163–193.
ETO. Emerging Technology Observatory. **Center for Security and Emerging Technology - Georgetown University**, jan. 2024. Disponível em: https://sciencemap.eto.tech/?ai_pred=10%2C100&all_subjects=Artificial+intelligence&x_growth_pred=True&mode=summary&cols=cluster_id%2Cclass_arts%2Ccset_extracted_phrase%2CNP%2Cgrowth_3yr_p_rank%2Cai_pred. Acesso em: 29 nov. 2024.

FOURRIER, Clémentine *et al.* Open LLM Leaderboard V2. **Hugging Face**, 2024. Disponível em: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard. Acesso em: 22 jun. 2024.

GARTNER. Automated Software Testing Adoption and Trends, fev. 2024. Disponível em: https://www.gartner.com/peer-community/oneminuteinsights/automated-software-testing-adoption-trends-7d6. Acesso em: 29 nov. 2024.

GITHUB. Início Rápido para o GitHub Copilot, jun. 2022. Disponível em: https://docs.github.com/pt/copilot/quickstart. Acesso em: 29 nov. 2024.

GOOGLE. Artificial Inteligence - Interest over time, set. 2012. Disponível em: https://trends.google.com/trends/explore?cat=12&date=2022-01-01%202024-06-22&geo=BR&q=%2Fm%2F0mkz&hl=en. Acesso em: 29 nov. 2024.

HIMASHI RATHNAYAKE, Janani S. R. R. S. R. AdapterFusion-based multi-task learning for code-mixed and code-switched text classification. **Engineering Applications of Artificial Intelligence, Volume 127, Part A**, jan. 2024.

HU, Ke; ZHANG, Yu; AL., Du N. E. **Massively Multilingual Shallow Fusion with Large Language Models**. CASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing. Rhodes Island, Greece: IEEE. 2023. p. 1 - 5.
IBM. O que é observabilidade e por que é importante?, set. 2023. Disponível em: https://www.ibm.com/br-pt/resources/automate/observability-basics. Acesso em: 29 nov. 2024.

IBM; ARMAND RUIZ; VIVEK BHARATHI. Scaling generative AI with flexible model choices. **IBM.COM**, maio 2024. Disponível em: https://www.ibm.com/blog/scaling-generative-ai-with-flexible-model-choices/. Acesso em: 29 nov. 2024.

INFORCHANNEL. Empresas usam IA para garantir qualidade no desenvolvimento de softwares, dez. 2023. Disponível em: https://inforchannel.com.br/2023/12/07/empresas-usam-ia-para-garantir-qualidade-no-desenvolvimento-de-softwares. Acesso em: 29 nov. 2024.

KONSTANTINOS FILIPPOU, George E. T. G. A. G. A. P. Structure Learning and Hyperparameter Optimization Using an Automated Machine Learning (AutoML) Pipeline. **International Hellenic University**, 09 abr. 2023.

LANGCHAIN. LangChain. **LangChain**, 2022. Disponível em: https://python.langchain.com/v0.2/docs/introduction/. Acesso em: 29 nov. 2024.

MAHMUD MH, Nayan M. A. D. K. M. Software Risk Prediction: Systematic Literature Review on Machine Learning Techniques. **Applied Sciences**, 2022.

META. Introducing Code Llama, a state-of-the-art large language model for coding, ago. 2023. Disponível em: https://ai.meta.com/blog/code-llama-large-language-model-coding. Acesso em: 29 nov. 2024.

META. Introducing Llama, jul. 2023. Disponível em: https://ai.meta.com/llama. Acesso em: 29 nov. 2024.

META. Introducing Meta Llama 3: The most capable openly available LLM to date. **Meta - Llama3**, abr. 2024. Disponível em: https://ai.meta.com/blog/meta-llama-3/. Acesso em: 29 nov. 2024.

MICROSOFT. AutoGen. **Microsoft - Github**, 2024. Disponível em: https://microsoft.github.io/autogen/. Acesso em: 29 nov. 2024.

MICROSOFT. Get start with AutoGen for dotnet. **Microsoft - Github**, 2024. Disponível em: https://microsoft.github.io/autogen-for-net/. Acesso em: 29 nov. 2024.

MICROSOFT. Microsoft - Autogen. **Github - AutoGen**, 2024. Disponível em: https://github.com/microsoft/autogen?formCode=MG0AV3. Acesso em: 29 nov. 2024.

MICROSOFT. what Is Continuous Delivery (CD)? **Learn Microsoft**, 2024. Disponível em: https://learn.microsoft.com/pt-br/devops/deliver/what-is-continuous-delivery. Acesso em: 29 nov. 2024.

MICROSOFT. what is continuous integration (CI)? **Lear Microsoft**, mar. 2024. Disponível em: https://learn.microsoft.com/pt-br/devops/develop/what-is-continuous-integration. Acesso em: 29 nov. 2024.

NADIRI, Yashar T. A. A. Multi-Agent Collaboration: Harnessing the Power of Intelligent LLM Agents, 05 jun. 2023.

NEWRELIC. All-in-one observability, maio 2008. Disponível em: https://newrelic.com. Acesso em: 29 nov. 2024.

NEWRELIC. Meet New Relic AI, your observability assistant, out. 2020. Disponível em: https://docs.newrelic.com/docs/new-relic-solutions/new-relic-one/core-concepts/new-relic-ai. Acesso em: 29 nov. 2024.

OPSERVICES. A IA aplicada a observabilidade e monitoramento de TI, abr. 2024. Disponível em: https://www.opservices.com.br/a-ia-aplicada-a-observabilidade-e-monitoramento-de-ti. Acesso em: 29 nov. 2024.

REBECKA C. ÅNGSTRÖM, Michael B. L. D. M. M. M. W. W. Getting AI Implementation Right: InsIghts from a global survey. **California Management Review 66 (1)**, California, 30 ago. 2023. 5 - 22.

REMSOFT. Inteligência Artificial: O Futuro Dos Testes De Software, ago. 2023. Disponível em: https://remsoft.com.br/blog/tecnologias/ia-o-futuro-dos-testes-de-softwares. Acesso em: 29 nov. 2024.

ROZIÈRE, Baptiste; GEHRING, Jonas; GLOECKLE, Fabian E. A. Code Llama: Open Foundation Models for Code. **Meta - Code Llama**, ago. 2023. Disponível em: https://ai.meta.com/research/publications/code-llama-open-foundation-models-for-code/. Acesso em: 22 jun. 2024.

SHARMA, Shreta A. P. S. K. Integrating AI Techniques In SDLC: Design Phase Perspective. Kochi: Association for Computing Machinery, 2015. p. 383–387.

SHEKAR RAMACHANDRAN, Rupali A. P. M. H. B. G. L. A. K. Automated Log Classification Using Deep Learning. **Procedia Computer Science**, 31 jan. 2023.

SONG CHEN, Hai L. BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. **Qingdao University of Technology - Applied Artificial Intelligence, 36**, 17 nov. 2022.

SRIVASTAVA, Niharika. What is SOTA in Artificial Intelligence? **E2E Networks**, 2022. Disponível em: https://www.e2enetworks.com/blog/what-is-sota-in-artificial-intelligence. Acesso em: 22 jun. 2024.

TABNINE. The AI coding assistant that you control, nov. 2018. Disponível em: https://www.tabnine.com. Acesso em: 29 nov. 2024.

TIAGO CARVALHO, João B. P. P. J. M. P. C. A DSL-based runtime adaptivity framework for Java, Porto, Portugal, 23 ago. 2023.

VERICODE. Desenvolvimento ágil, seguro e eficiente, maio 2022. Disponível em: https://vericode.com.br/servicos/devsecops. Acesso em: 29 nov. 2024.

VERICODE. Os testes automatizados são um imperativo dos negócios digitais, maio 2022. Disponível em: https://vericode.com.br/servicos/testes-automatizados. Acesso em: 29 nov. 2024.

VERICODE. Seu código fonte está sendo revisado automaticamente?, maio 2022. Disponível em: https://vericode.com.br/servicos/analise-de-codigo-fonte. Acesso em: 29 nov. 2024.

VERITY. Como a IA pode ajudar um QA (Quality Assurance)?, jun. 2023. Disponível em: https://www.verity.com.br/post/como-a-ia-pode-ajudar-um-qa-quality-assurance. Acesso em: 29 nov. 2024.

WEAVIATE. The AI-Native, Open Source Vector Database. **Weaviate Platform**, 2024. Disponível em: https://weaviate.io/platform. Acesso em: 29 nov. 2024.

WEAVIATE. Weaviate. **GitHub**, 2024. Disponível em: https://github.com/weaviate/weaviate. Acesso em: 29 nov. 2024.

XI, Zhiheng; CHEN, Wenxiang; GUO, Xin E. A. The Rise and Potential of Large Language Model Based Agents: A Survey, 14 set. 2023.

ZAHRAA SADDI KADHIM, Khalil I. G. H. S. A. Artificial Neural Network Hyperparameters Optimization: A Survey. **International Journal of Online and Biomedical Engineering**, 06 dez. 2022.

ZHAO, Wayne X.; ZHOU, Kun; AL., Li J. E. A Survey of Large Language Models. **Arxiv**, 24 nov. 2023. Disponível em: https://arxiv.org/abs/2303.18223. Acesso em: 22 jun. 2024.

ZHOU, Ce; QIAN, Li; AL., Li C. E. A Comprehensive Survey on Pretrained Foundation Models: A History from BERT to ChatGPT. **Arxiv**, fev. 2023. Disponível em: https://arxiv.org/abs/2302.09419. Acesso em: 22 jun. 2024.