Pipeline de Avaliação de Performance, Custo e Qualidade em Agentes Cognitivos

Anna Júlia de Souza Ferreira, Tech for Humans

I. Introdução

A rápida evolução dos Modelos de Linguagem de Larga Escala (LLMs) redefiniu as possibilidades do atendimento automatizado. No entanto, a transição de um protótipo para uma operação *Enterprise* exige um equilíbrio rigoroso entre custo operacional, latência e eficácia resolutiva.

Este artigo apresenta um pipeline de avaliação automatizada, construído e em operação sobre a plataforma Tech4AI, que investiga de forma comparativa e pragmática os modelos de linguagem adotados por grandes *players* e por seus concorrentes. O foco vai além da avaliação de desempenho bruto: a pipeline mede a viabilidade real de substituição de modelo em ambiente de produção, considerando cotas de API, limites de requisição, custos operacionais e conformidade contratual, sob condições estritamente controladas.

Adicionalmente, o projeto privilegia arquiteturas que reduzam a dependência de modelos sujeitos à rápida depreciação tecnológica, garantindo a possibilidade de substituição por LLMs de desempenho equivalente ou superior, sem comprometer a eficiência e a continuidade de sistemas baseados em *agents*.

A. Premissa da Pesquisa (Base 0)

O alicerce desta avaliação é a denominada **Base 0**: um *dataset* de conversas sintéticas selecionadas como referência de qualidade. Este ponto neutro de qualidade define o critério mínimo de aceitação; qualquer modelo candidato deve, obrigatoriamente, atingir ou superar o nível de qualidade linguística e resolutiva estabelecido por essa base de referência. O alinhamento à Base 0 é definido como equivalência semântica ou melhoria, não identidade literal: paráfrases e variações linguísticas são aceitáveis desde que preservem as informações essenciais e a conclusão resolutiva.

B. Estrutura de Execução

A pesquisa está estruturada em dois horizontes estratégicos:

- **Fase 1: Equivalência e Viabilidade (Baseline)**, implementada e reportada neste artigo. Um motor de *replay* reexecuta as conversas sintéticas da Base 0 sob paridade operacional total com o ambiente de avaliação, isolando a variável do modelo de linguagem (*LLM-swap*) para identificar candidatos que entreguem performance equiparável ou superior, com foco em otimização de custos, redução de latência e ganho de robustez técnica.
- **Fase 2: Otimização e Evolução (Incremental)**, trabalho futuro. Estabelecida a estabilidade operacional, a pesquisa passará a investigar melhorias incrementais, como capacidades avançadas de *Function Calling* e janelas de contexto estendidas, mitigando fricções identificadas no fluxo atual.

C. Contribuições e Objetivo da Pesquisa

Este trabalho contribui com: (i) uma arquitetura de *replay* que isola a variável do modelo de linguagem sob paridade operacional total com o ambiente de referência; (ii) um motor de métricas que combina um critério inteiramente determinístico (regras de negócio), duas métricas híbridas que somam uma base determinística a uma camada de julgamento contextual pontual (TCR e TCA), e critérios não determinísticos (LLM-as-a-Judge em painel de múltiplos juízes, similaridade semântica) em um veredito único e auditável; e (iii) resultados empíricos da avaliação de 8 modelos candidatos (famílias GPT-4.1/5, Gemini 2.5 e modelos open-weight como Kimi K2.5 e GPT-OSS-120B) sobre centenas de execuções da pipeline, cobrindo quatro instâncias sintéticas de atendimento. A seleção do modelo ideal é guiada por uma matriz de decisão que correlaciona capacidade de raciocínio, qualidade linguística em Português (PT-BR), latência total por turno e Custo por Sessão (CPS), garantindo uma operação escalável, segura e competitiva.

II. Trabalhos Relacionados

Avaliação de agentes com LLM-as-a-Judge. A técnica de utilizar um modelo de linguagem como avaliador automatizado de respostas foi consolidada por [6], que demonstrou forte concordância entre juízes LLM fortes (GPT-4) e avaliadores humanos em benchmarks conversacionais (MT-Bench, Chatbot Arena). Mais recentemente, [3] sistematizou o uso de LLM-as-a-Judge especificamente para avaliação de *agents*, distinguindo *code-based graders*, *model-based graders* e avaliadores humanos. A maioria desses trabalhos aplica o juiz a respostas isoladas ou turnos únicos de conversa. Em contraste, este trabalho aplica o LLM-as-a-Judge à **sessão completa** de atendimento, com painel de múltiplos juízes de arquitetura distinta do candidato avaliado, mitigando o *self-preference bias* e permitindo avaliação holística de tarefas transacionais de múltiplos turnos.

Benchmarks de *tool-calling* e *function-calling*. O Berkeley Function Calling Leaderboard (BFCL) [5] estabeleceu um método de avaliação sintática (via *Abstract Syntax Tree*) para validar chamadas de função contra milhares de funções em cenários controlados de benchmark público. Esses trabalhos avaliam a correção da chamada de ferramenta de forma isolada, contra um gabarito sintético. Em contraste, a métrica de *Tool Calling Accuracy* (TCA) deste trabalho compara a

sequência de *tools* invocadas contra o transcript de referência de um atendimento **sintético, selecionado como referência de qualidade** (Base 0), capturando não apenas validade sintática, mas equivalência semântica de sequência em um cenário controlado de avaliação.

Avaliação de agentes conversacionais multi-turno com políticas de domínio. O τ-bench [7] introduziu um benchmark no qual um agente interage com um usuário simulado por LLM e com APIs de domínio (varejo, companhias aéreas), sendo avaliado pelo estado final do banco de dados e pela métrica *pass^k* de consistência entre execuções. Assim como este trabalho, o τ-bench reconhece a necessidade de testar aderência a políticas e consistência em interações de múltiplos turnos. No entanto, o τ-bench depende de um usuário simulado por LM interagindo com bancos de dados sintéticos. Este trabalho utiliza um corpus sintético interno ao reexecutar, via um motor de *replay* com paridade de referência, **conversas sintéticas já concluídas e selecionadas**, isolando a variável de teste exclusivamente na troca do modelo de linguagem (*LLM-swap*).

Em síntese, nenhum dos trabalhos citados combina (i) um corpus sintético de referência selecionado para avaliação, (ii) a troca controlada do modelo de linguagem sob paridade operacional total (*LLM-swap*/*replay*) e (iii) um veredito multi-critério que pondera conjuntamente qualidade, latência e custo, lacuna que este trabalho busca preencher.

III. Arquitetura da Plataforma Tech4AI

A. Visão Geral do Produto

A Tech4AI é uma plataforma de agentes conversacionais de Inteligência Artificial para atendimento automatizado em ambiente corporativo (*Enterprise*). Ela permite configurar diferentes **instâncias** de atendimento de forma isolada, com fluxos, ferramentas e regras próprias para cada cenário de avaliação.

O diferencial de arquitetura é a separação entre *o que o agente deve fazer* e *como o modelo de linguagem decide fazê-lo*: o comportamento esperado para cada instância (escopo do atendimento, tom de voz, passos de um fluxo transacional) é descrito de forma declarativa em uma **Skill**, enquanto integrações com sistemas externos (por exemplo, consulta de cadastro, emissão de protocolo ou confirmação de uma solicitação) são expostas ao modelo como **Actions**, funções que o LLM pode invocar quando a conversa exige uma operação concreta, em vez de apenas texto. Essa separação permite trocar o modelo de linguagem subjacente sem alterar a lógica de negócio de cada instância, o que é a premissa central do experimento de *LLM-swap* deste artigo.

Em termos de funcionamento, cada mensagem do usuário é recebida por um orquestrador (**Agent**) que identifica a Skill correspondente àquele atendimento e conduz um ciclo de decisão: o modelo interpreta a mensagem, decide se pode responder diretamente ou se precisa executar uma ou mais Actions para obter ou alterar uma informação, e repete esse ciclo até que o atendimento seja concluído, por resolução direta ou por transferência a um atendente humano. O estado da conversa (dados do usuário e progresso do atendimento) é mantido por um componente de **Memory**, e toda a comunicação com o canal do usuário final ocorre por meio de uma API de sessões (**API v2**). A Tabela I detalha os pilares centrais do sistema que são mantidos constantes durante o processo de avaliação (*LLM-swap*): apenas o modelo de linguagem por trás do **Agent** é substituído entre execuções.

TABELA I
COMPONENTES PRINCIPAIS DA PLATAFORMA TECH4AI.

Componente	Descrição
Agent	Orquestrador que recebe mensagens, define a *Skill* e executa o ciclo de *completion*.
Skill	Arquivo YAML contendo instruções, tópicos e passos que guiam o LLM no fluxo.
Action	Funções Python executadas via *tool call* para integração com sistemas.
LLMHelper	Interface para chamadas com *structured output* e *tracing* via LangFuse.
Memory	Gerenciador de estado que injeta dados do usuário e progresso como contexto Markdown.
API v2	*Endpoints* REST para abertura de sessões e troca de mensagens via *webhooks*.
System Actions	Ações nativas: `finalizar_conversa` e `transferir_agente_humano` (para solicitações fora do escopo, pedido explícito do usuário ou atendimento especializado).

B. Ciclo de Interação e Decisão

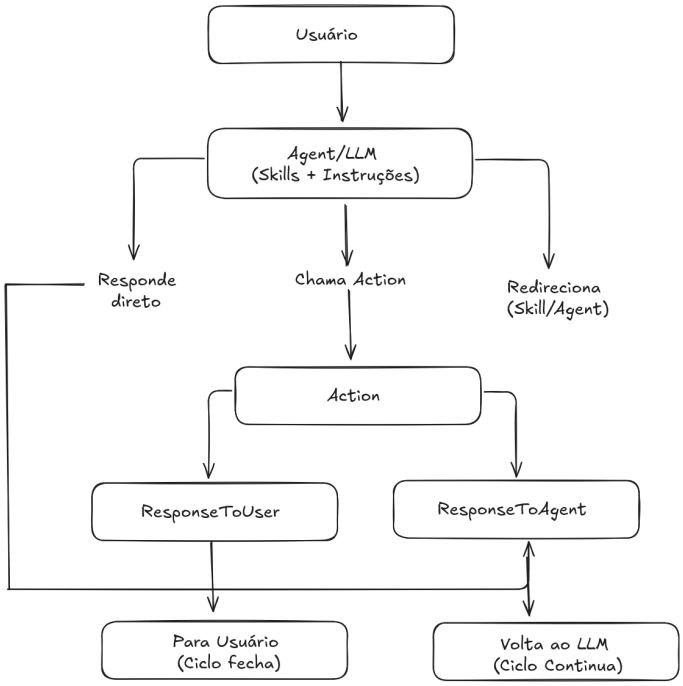
O fluxo de atendimento baseia-se em ciclos de decisão onde o modelo avalia se a solicitação do usuário exige a execução de ferramentas (*Actions*). A plataforma utiliza dois objetos de retorno fundamentais para as *Actions*:

- `ResponseToUser`: Objeto que interrompe o ciclo de raciocínio e entrega a resposta final ao usuário via *webhook*.
- `ResponseToAgent`: Objeto que devolve a saída da *Action* ao LLM para que este realize uma nova decisão (loop de raciocínio).

Cada ciclo de `ResponseToAgent` consome *tokens* adicionais de entrada e saída. Modelos que exigem múltiplos ciclos para resolver tarefas que a Base 0 resolve de forma simples elevam o **Custo por Sessão (CPS)** e o **Tempo Total de Sessão (TTS)**. Portanto, a precisão no acionamento de *Actions* é um critério de sucesso eliminatório.

O escopo de uma conversa não é necessariamente fixo do início ao fim: uma das *Actions* disponíveis ao modelo é o redirecionamento de habilidade, que transfere a condução do atendimento para uma Skill diferente quando a solicitação do usuário sai do escopo da Skill ativa naquele momento. Quando esse redirecionamento ocorre, a plataforma remonta dinamicamente a superfície de *Actions* oferecida ao modelo, restringindo-a às ferramentas declaradas pela nova Skill, e atualiza as instruções do *system prompt* para refletir o novo escopo. Ou seja, o conjunto de ferramentas que o modelo enxerga a cada turno não é estático para toda a sessão: é dinâmico, dependente da Skill ativa naquele ponto da conversa, e esse comportamento é replicado com fidelidade pelo motor de *replay* (Seção de Metodologia).

Fig. 1
CICLO DE INTERAÇÃO TECH4AI



IV. METODOLOGIA

A metodologia deste projeto baseia-se em uma abordagem experimental controlada, utilizando uma pipeline automatizada que reexecuta conversas sintéticas selecionadas na plataforma **Tech4AI** como corpus de referência (Base 0). O objetivo é garantir que a substituição do modelo de linguagem seja avaliada sob condições de paridade com o ambiente de referência, sem depender de tráfego ao vivo.

A. Estratégia de Avaliação: LLM-Swap

O diferencial deste *design* é o isolamento da variável de teste através do conceito de *LLM-swap*. Nesta configuração:

- A **Skill** (instruções YAML), as **Actions** (funções Python), o **Memory Manager** e o *prompt* permanecem inalterados: é literalmente o mesmo *prompt* validado para o ambiente de avaliação, não uma versão reescrita ou simplificada para fins de pesquisa. Isso garante que o comportamento observado no *replay* reflita as mesmas instruções que orientam o atendimento de referência, preservando a consistência do cenário sintético do experimento.
- A conversa sintética selecionada não é enviada a um agente ao vivo: um **motor de *replay*** reconstrói o contexto exato em que cada turno ocorreu e reexecuta apenas a decisão do LLM sob teste, sob paridade operacional com o comportamento de referência do produto. A paridade inclui: preenchimento dos blocos de memória e hora no *system prompt* (`update_memory_and_time_in_system_prompt`) com os valores sintéticos daquele momento, as *tools* e `initial_messages` do baseline antes da primeira mensagem do usuário, o remonte dinâmico da superfície de *tools* a cada troca de Skill ativa (Seção de Arquitetura da Plataforma Tech4AI), filtro do histórico por `agent_id` após um *handoff*, e a regra de encerramento alinhada ao último passo do baseline.
- Esse preenchimento de contexto não deve ser confundido com *prompt injection*: em nenhum momento o roteiro de referência ou a resposta esperada da Base 0 é inserida no *prompt* como instrução para guiar a decisão do modelo candidato. O LLM sob teste recebe apenas o mesmo contexto informativo do cenário sintético (dados do usuário, histórico da conversa, ferramentas disponíveis) e decide de forma independente como agir. Nenhuma técnica de indução ou manipulação do *prompt* foi empregada para aproximar artificialmente o comportamento do candidato do *baseline*.
- A única variável modificada é o modelo de linguagem (LLM) que decide cada turno.

Dessa forma, garante-se que qualquer variação de performance, latência ou custo seja estritamente atribuível à troca do modelo, e não a diferenças no contexto reconstruído ou a algum viés introduzido no *prompt*.

B. Estrutura do Repositório

A organização modular do código permite a rastreabilidade total do experimento:

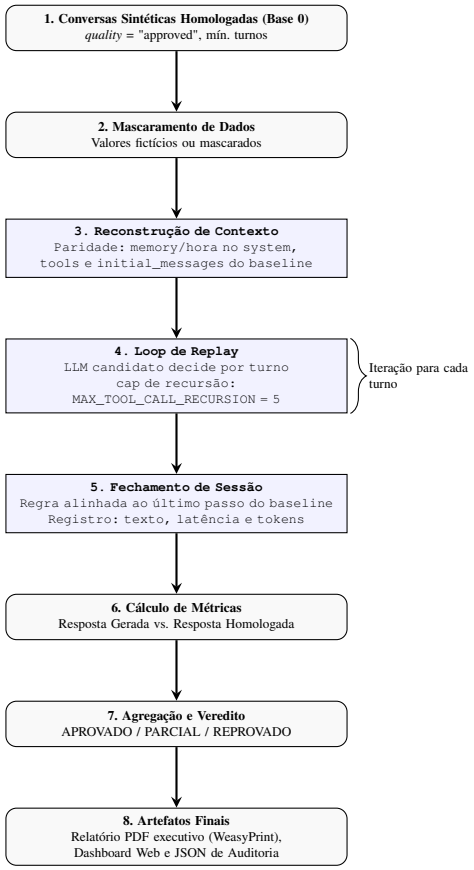
```
agent_eval/
├── evaluation/ ..... Métricas: TCR/TCA, score e Judge
│   ├── acceptance_metrics.py ... Aceitação: TCR/TCA/regras
│   ├── acceptance_operational.py .... Pass Rate operacional
│   ├── scoring.py  Fórmulas de TCR/TCA e similaridade narrativa
│   ├── composite_score.py ..... Score Final (Q/L/C)
│   ├── judge_policy.py ..... Política do painel de juízes
│   ├── judge_runtime.py ..... Execução do painel de juízes
│   ├── execution_quality.py  Qualidade de execução por passo
│   └── tool_adherence_diagnosis.py ..... Diagnóstico de
│       aderência de tools
├── replay/ ..... Motor de replay com paridade de referência
│   ├── engine.py ..... Orquestrador do replay + cap de recursão
│   ├── prod_parity.py ..... Paridade com o runtime Tech4AI
│   ├── memory_synth.py ..... Síntese de memória/contexto
│   └── tool_history.py ..... Formatação de histórico de tools
├── api/ ..... Backend FastAPI
│   └── app.py ..... Aplicação e rotas
├── reporting/ ..... Artefatos de auditoria
│   ├── reports_store.py ..... Relatório consolidado por sessão
│   └── audit_consolidated_pdf.py ..... PDF executivo
│       (WeasyPrint)
├── dataset.py ..... Carga e mascaramento da Base 0
├── pipeline.py ..... Orquestração ponta a ponta
└── config.py ..... Thresholds e pesos
frontend/ ..... Dashboard Web (React/TypeScript)
```

C. Fluxo de Execução da Pipeline

O processo de avaliação é estruturado como um ciclo fechado ilustrado na Figura 2.

Fig. 2

FLUXO DETALHADO DA PIPELINE DE AVALIAÇÃO DE MODELOS



D. Execução em Lote e Concorrência

Para viabilizar a execução de mais de cem rodadas de avaliação em tempo hábil de desenvolvimento, a pipeline processa múltiplas combinações de sessão e modelo candidato **simultaneamente**, em vez de uma de cada vez. Essa execução paralela é feita por meio de múltiplas linhas de processamento (*threads*) reais, em vez de um mecanismo de concorrência cooperativa (*asyncio*), já que o motor de *replay* é executado de forma síncrona e threads reais se mostraram mais confiáveis nesse contexto.

Cada provedor de modelos de linguagem (no caso, a Microsoft Azure, que hospeda os modelos avaliados neste artigo) impõe um limite de quantas requisições (e quantos *tokens* processados) podem ser enviadas por minuto para cada modelo. Se a pipeline enviar requisições simultâneas acima desse limite, o provedor passa a recusar as excedentes com um erro de "limite de taxa excedido"(HTTP 429), obrigando a pipeline a tentar novamente mais tarde, o que, paradoxalmente, torna a execução mais lenta quanto mais agressivamente ela tenta paralelizar.

O limite de requisições por minuto não é o mesmo para todos os modelos: famílias de modelos mais recentes e mais sofisticadas (como a família GPT-5, avaliada neste trabalho) operam sob cotas sensivelmente mais restritas do que modelos mais antigos. Para lidar com essa diferença sem exigir ajuste manual a cada rodada, a pipeline implementa um **teto de concorrência adaptativo**: antes de iniciar o lote, ela verifica quais modelos serão testados e, caso a família GPT-5 esteja entre eles, reduz automaticamente o número de execuções simultâneas apenas para aquela rodada (por padrão, no máximo 4 em paralelo), preservando um paralelismo maior para os demais modelos. Esse ajuste evita o desperdício de tempo com tentativas que falhariam de qualquer forma, tornando a execução, na prática, mais rápida e mais estável do que uma estratégia de paralelismo fixo e máximo.

Complementarmente, cada unidade de trabalho (uma combinação de sessão e modelo) que falha por instabilidade de rede, tempo-limite ou erro HTTP transitório é reexecutada automaticamente, com um intervalo de espera crescente entre as tentativas (*retry* com *backoff* exponencial). As conexões de rede com a Azure também são reaproveitadas entre chamadas de uma mesma linha de processamento, evitando o custo de reabrir uma conexão segura a cada requisição. Em conjunto, esses mecanismos permitiram executar as centenas de rodadas de avaliação realizadas neste trabalho, totalizando milhares de avaliações de sessão, dentro de um ciclo de desenvolvimento iterativo.

E. Metrics Engine: Aquisição e Cálculo das Métricas

O módulo `evaluation/` centraliza a aquisição e o cálculo das métricas que comparam a resposta gerada pelo modelo candidato com a referência selecionada da Base 0. A comparação segue uma hierarquia de avaliação: (1) métricas por turno, entre elas a TCR e a TCA (que combinam uma base determinística com uma camada de julgamento contextual quando a tool chamada diverge do baseline, detalhado a seguir), além das regras de negócio; (2) agregação dos resultados; (3) LLM-as-a-Judge para as dimensões qualitativas da **sessão como um todo** que não admitem automação determinística. O Judge é aplicado majoritariamente à avaliação holística da sessão, não a cada turno isoladamente, com a exceção do juiz de competência de ferramentas usado por turno dentro da própria TCR e da TCA. As duas camadas a seguir compõem essa comparação:

1) **LLM-as-a-Judge (avaliação qualitativa da sessão):** Um ou mais modelos juízes recebem como entrada a sessão completa (ou agregado representativo), incluindo a pergunta do usuário, a resposta gerada pelo candidato e a resposta de referência da Base 0 (*ground truth*). A mesma sessão pode ser avaliada por n juízes distintos em um mesmo painel. A variação entre múltiplas gerações do candidato para a mesma sessão não é obtida agrupando tentativas na mesma chamada de julgamento, e sim executando a pipeline de forma independente e repetida (Seção de Resultados, "Consistência do Veredito entre Execuções"), cada execução julgada uma única vez pelo painel. A estrutura do *prompt*, as rubricas, a escala (Likert de 4 pontos) e a regra de agregação do painel estão detalhadas na Seção de LLM-as-a-Judge deste documento.
2) **Similaridade Semântica (apoio pontual à TCA):** para a maioria dos argumentos de ferramenta (IDs, datas, valores, números de protocolo), a comparação é literal: uma redação diferente da baseline conta como erro.

A única exceção é uma lista restrita de parâmetros narrativos de texto livre (por exemplo, o motivo de uma transferência de habilidade, ou a mensagem de despedida em *tools* de encerramento da conversa), onde, e só aí, uma redação diferente não deveria penalizar o modelo. Nesses casos, a pipeline tenta duas camadas em sequência: primeiro uma sobreposição de tokens (índice de Jaccard), leve e determinística; se essa falhar, calcula a similaridade de cosseno entre embeddings (`text-embedding-3-small` da Azure) do argumento gerado e do argumento de referência, e, se essa segunda chamada falhar por qualquer motivo (ex.: indisponibilidade momentânea da API), o argumento é tratado como não equivalente, sem interromper a avaliação. Essa restrição de escopo é proposital: generalizar a similaridade textual para qualquer argumento diluiria ainda mais a base **determinística e reproduzível** da TCA (Seção de Critérios do Projeto), que já convive com uma camada própria de julgamento contextual, delimitada a casos de divergência do baseline (Seção de Discussão). Não há, hoje, uma comparação de embeddings entre a resposta completa da sessão e a resposta de referência como sinal independente de apoio ao LLM-as-a-Judge.

O veredito final resulta da combinação ponderada da avaliação qualitativa do Judge com as métricas de base determinística (TCR e TCA, ambas já incorporando o juiz de competência de ferramentas e a similaridade semântica de argumentos, e regras de negócio), garantindo que a substituição do modelo seja avaliada sob múltiplas perspectivas.

F. Avaliações Determinísticas e Não Determinísticas

O cálculo de métricas (Etapa 6 da *pipeline*) combina duas estratégias complementares de avaliação, permitindo cobrir tanto aspectos técnicos objetivos quanto aspectos qualitativos e semânticos.

1) Avaliações Determinísticas: As métricas de base determinística são aquelas cujo componente técnico é **reproduzível e invariante** entre execuções, independentemente de aleatoriedade do modelo; a TCR e a TCA contínuas, usadas na composição final, incorporam também uma camada pontual de julgamento contextual, detalhada a seguir. Incluem:

- **Tool Calling Accuracy (TCA):** sua base é determinística: (1) validade sintática do JSON de invocação; (2) existência da *action* no catálogo disponível; (3) presença e tipagem dos parâmetros obrigatórios; (4) equivalência da sequência de tools invocadas em relação ao transcript de referência da Base 0, mediante regras de normalização definidas. Quando uma tool chamada diverge do passo esperado do baseline, essa base determinística é complementada por um juiz de competência de ferramentas (LLM-as-a-Judge), cuja nota de pertinência contextual compõe parte do escore contínuo de TCA (Seção de Discussão).
- **Task Completion Rate (TCR):** não é uma checagem binária. O TCR é uma nota contínua (entre 0 e 1), calculada como uma média ponderada de três componentes: conclusão suave dos passos do fluxo transacional (peso de 45%), eficiência do caminho percorrido em relação ao roteiro de referência (peso de 30%) e satisfação do fechamento da sessão (peso de 25%); o componente de conclusão soma o maior valor entre o match determinístico por passo e a nota do mesmo juiz de competência de ferramentas usado pela TCA, quando a tool diverge do baseline. Um indicador binário de aprovação (`tcr_pass`), esse sim inteiramente determinístico, continua existindo separadamente, reservado para o *gate* eliminatório de *Pass Rate* (Seção de Critérios do Projeto), mas a nota de TCR usada na composição do Score Final é sempre a versão contínua.
- **Regras de Negócio:** Validações baseadas em expressões regulares ou lógica fixa (ex.: formato de protocolo, presença de campos obrigatórios na resposta).

A base determinística dessas métricas (regras de negócio e os indicadores binários estritos de TCR e TCA) oferece reprodutibilidade total, independente da temperatura ou da variância do modelo; suas versões contínuas, como já descrito, têm reprodutibilidade parcial pela camada de julgamento contextual, mas seguem sendo auditáveis e aplicadas de forma consistente aos oito candidatos.

2) Avaliações Não Determinísticas: As métricas não determinísticas capturam aspectos que exigem interpretação semântica ou julgamento qualitativo, onde uma comparação literal (ex.: *string matching*) seria inadequada. Incluem:

- **LLM-as-a-Judge:** O modelo juiz avalia a adequação da **sessão completa** em relação à intenção do usuário e à referência selecionada da Base 0. O Judge é reservado para dimensões que exigem interpretação semântica e não admitem automação determinística (ex.: tom, completude resolutiva, alinhamento à Base 0). A nota atribuída pode variar levemente entre execuções: diferente do que uma calibração de pesquisa poderia sugerir, a temperatura usada nas chamadas dos juízes não é reduzida para forçar maior determinismo. Ela é definida por uma configuração própria e dedicada aos juízes (`JUDGE_LLM_TEMPERATURE`, *default* 1), independente da temperatura que o candidato de fato usou naquela sessão, não uma réplica dinâmica dela; para a família GPT-5, a Azure fixa a temperatura em 1 de qualquer forma (Seção de Provedores), o que apenas coincide com esse *default* dos juízes, sem relação causal entre as duas configurações.
- **Similaridade Semântica:** comparação de sobreposição de tokens e, em caso de falha desta, de embeddings, entre um argumento narrativo gerado e o argumento de referência, restrita a uma lista específica de parâmetros e usada como *fallback* dentro da TCA quando a comparação literal de texto livre falha, permitindo capturar equivalências parafrásticas que um *match* exato ignoraria.
- **Avaliação de Consistência (múltiplas execuções):** Para cenários em que se deseja medir a estabilidade do modelo, são realizadas múltiplas rodadas com a mesma entrada, e analisa-se a variância das respostas para o mesmo *input*.

A combinação das duas abordagens garante que o veredito final considere tanto a precisão técnica (determinística) quanto a qualidade linguística e resolutiva (não determinística), alinhada aos critérios estabelecidos na Seção de Critérios do Projeto.

G. Evidências no Transcript e Artefatos de Auditoria

Os artefatos de auditoria incluem **evidências explícitas** por turno e por critério, indicando se a resposta foi considerada adequada e quais aspectos justificam a decisão. Para cada critério (TCR, TCA, alinhamento semântico, ausência de alucinação), registra-se um indicador binário (pass/fail) e, quando aplicável, trechos ou justificativas que permitam rastreabilidade. O feedback do LLM-as-a-Judge é armazenado como evidência qualitativa da avaliação holística. O JSON de *run* completo (`resultado_*.json`) preserva os dados brutos para reprocessamento; um relatório consolidado mais leve (por sessão, com veredito operacional e funil de falhas) alimenta o Dashboard Web sem exigir nova execução do replay. Essas evidências permitem auditoria e debugging sem necessidade de re-execução da *pipeline*.

V. Dataset

O *dataset* utilizado na avaliação é composto pela **Base 0**: conversas sintéticas selecionadas como referência de qualidade na plataforma Tech4AI. Estes registros constituem o corpus de referência (*ground truth*) contra o qual as respostas dos modelos candidatos são comparadas. A construção da Base 0 é, por natureza, um processo **qualitativo**: cada sessão é validada por revisão humana antes de ser admitida como referência, e não apenas por filtros automáticos.

A. Critérios de Amostragem

O ponto de partida é um conjunto de sessões sintéticas já mascaradas e marcadas com `quality = "approved"` como referência interna de qualidade, com número mínimo de turnos definido em `config.py` para garantir complexidade representativa. Este conjunto inicial reúne 106 sessões, distribuídas em diversas **instâncias de atendimento** de diferentes domínios de atendimento. Uma instância de atendimento corresponde, nesta pesquisa, ao conjunto de sessões de uma mesma instância sintética de avaliação, atendida por uma configuração de agente própria (*Skill*, *Actions* e tom de voz específicos, Seção de Arquitetura da Plataforma Tech4AI).

B. Curadoria Qualitativa da Base 0

A marcação automática `quality = "approved"` é necessária, mas não suficiente: nem toda sessão candidata demonstra, na prática, aderência estrita ao comportamento esperado do agente. Por isso, cada uma das 106 sessões do conjunto inicial foi **revisada manualmente, turno a turno**, por um analista humano, contra um roteiro estruturado que cobre: roteamento de *skills* (*steps*/tópicos), aderência às instruções do *system prompt*, sequência de mensagens usuário/assistente, corretude da sequência, nomes e argumentos das *tool calls*, e conformidade com regras de *Anti-Overlap* (não sobreposição de escopo entre *skills*). Cada sessão recebeu um veredito explícito de seleção, com evidência de turno/mensagem quando aplicável. Divergências atribuíveis ao mascaramento de dados (por exemplo, inconsistências em identificadores, contatos, endereços ou datas mascaradas) foram deliberadamente desconsideradas nessa revisão, por não refletirem falhas do agente no cenário avaliado.

Esse processo de curadoria qualitativa resultou em uma **Base 0 final de 20 sessões selecionadas**, refinada a partir do conjunto inicial de 106, distribuídas em quatro instâncias de atendimento (Tabela II).

TABELA II
COMPOSIÇÃO DA BASE 0 FINAL (20 SESSÕES SELECIONADAS), POR INSTÂNCIA DE ATENDIMENTO.

Instância	Sessões aprovadas	Participação
A	2	10%
B	9	45%
C	8	40%
D	1	5%

Nas 20 sessões da Base 0 final, o número de turnos de usuário por sessão tem média de 7,85 (mediana 5,5; mínimo 2, máximo 33), com média de 10,5 chamadas de ferramenta por sessão (máximo 32). O sentimento do usuário sintético ao final do atendimento é predominantemente neutro ou positivo (19 de 20 sessões) e nenhuma das 20 sessões foi transferida a atendimento humano, o que é coerente com o critério de curadoria, que privilegia exemplos de resolução autônoma bem-sucedida como referência de qualidade.

C. Dados Fictícios e Mascaramento

Antes de qualquer processamento ou envio às APIs dos provedores, o *dataset* passa obrigatoriamente por uma etapa de **mascaramento e geração de dados fictícios**, executada uma única vez sobre o conjunto selecionado, antes de qualquer sessão entrar no ciclo de avaliação. Esta etapa é crítica por três motivos fundamentais:

- **Isolamento do corpus:** Garantir que o corpus de avaliação use apenas valores fictícios ou mascarados antes de qualquer chamada externa.
- **Reprodutibilidade:** O mascaramento permite que o mesmo *dataset* seja reutilizado em múltiplas rodadas de avaliação mantendo apenas informações fictícias ou mascaradas, mantendo a integridade do experimento.
- **Neutralidade da Avaliação:** A substituição por dados fictícios evita que o modelo juiz ou os candidatos sejam influenciados por identificadores específicos do cenário sintético, preservando o foco na qualidade da resposta.

O mecanismo de mascaramento vai além de uma substituição simples: cada campo previsto pelas regras de mascaramento é substituído por um valor **fictício, porém válido no formato**. Quando aplicável, os valores fictícios preservam a estrutura esperada para validações de formato feitas pelo agente ou pela ferramenta. A substituição é determinística e consistente dentro de cada sessão: o mesmo identificador sempre recebe o mesmo valor fictício, inclusive quando aparece de

formas diferentes na mesma conversa, preservando a coerência da narrativa sem jamais reintroduzir qualquer valor original. Os dados enviados às APIs dos provedores são sempre a versão mascarada; essa etapa é executada de forma automática antes de qualquer sessão entrar no fluxo de avaliação.

VI. Critérios do Projeto: Avaliação de Modelos em ambiente de Atendimento

Esta seção estabelece as métricas de sucesso fundamentais para determinar a viabilidade operacional de modelos de linguagem em ambiente *Enterprise*. O objetivo central é assegurar que a solução selecionada atenda aos requisitos de eficiência de custo, conformidade técnica e eficácia na resolução de tarefas complexas.

A. *Eficiência de Custos e Viabilidade Operacional*

Esta dimensão estabelece os parâmetros para a análise da sustentabilidade financeira e da conformidade contratual da solução em escala corporativa.

1) ***Custo por Sessão (CPS)****:* Diferente da mensuração isolada de custo bruto por tokens, o *Cost Per Session* (CPS) projeta o valor real de uma interação completa, englobando todos os ciclos de *input* e *output* necessários para a resolução de um chamado.

- **Metodologia de Cálculo:** A "Base 0"é utilizada como referencial estatístico para determinar o volume médio de tokens processados por atendimento. O cálculo do CPS considera não apenas a mensagem final ao usuário (`ResponseToUser`), mas também todos os ciclos intermediários de raciocínio e execução de ferramentas (`ResponseToAgent`), que consomem *tokens* adicionais
- **Objetivo:** Identificar modelos que resolvam as solicitações com o menor número de turnos de pensamento, evitando que múltiplos ciclos de execução de *Actions* elevem o custo operacional e o tempo total da sessão.

2) ***Tierização de Modelos (Budget, Mid-range)****:* Os candidatos são categorizados de acordo com seu posicionamento de custo e capacidade de raciocínio no mercado

- **Categorias:** Os modelos são estratificados em tiers para facilitar a comparação de custo-benefício:
  - ***Budget:*** Modelos focados em alta eficiência e baixo custo (ex.: GPT-4.1 nano, Gemini 2.5 Flash).
  - ***Mid-Range:*** Modelos com equilíbrio entre raciocínio complexo e custo moderado (ex.: GPT-5 mini, GPT-5.4 mini).
- **Critério de Seleção:** O foco da pesquisa é localizar o *sweet spot* operacional: o modelo de menor custo marginal que consiga replicar ou superar o nível de qualidade estabelecido pela Base 0.

3) ***SLA e Disponibilidade Enterprise****:* Consiste na análise das garantias contratuais de cada provedor para garantir a continuidade da operação em larga escala.

- **Escalabilidade:** Avaliação rigorosa dos limites de requisições por minuto (RPM) e cotas de API disponíveis em planos corporativos.
- **Segurança Operacional:** O critério de conformidade exige garantias contratuais de proteção dos dados trafegados e de não utilização do conteúdo para treinamento de modelos públicos ou proprietários dos provedores.
- **Resiliência Tecnológica:** Preferência por provedores e modelos que apresentem menor risco de depreciação tecnológica rápida, garantindo a sustentabilidade da arquitetura de agents a longo prazo.

B. *Performance Técnica e Latência*

A agilidade na resposta em tempo real é tratada como um fator crítico para a experiência do usuário final e para a fluidez da interação humano-IA. Esta dimensão avalia se o modelo sustenta o dinamismo necessário para diálogos naturais.

1) ***Latência por Turno****:* As chamadas da *pipeline* às APIs dos provedores não usam modo *streaming*: a resposta completa de cada chamada ao LLM (texto e eventuais *tool calls*) chega de uma só vez, sem eventos incrementais por *token*. Isso significa que o *Time to First Token* (TTFT) e o *throughput* em *tokens* por segundo, que dependeriam de registrar o instante de chegada de cada *token* individualmente, não são metricamente distinguíveis do tempo total da chamada nesta arquitetura: do ponto de vista da *pipeline*, o "primeiro *token*"e o "último *token*"chegam no mesmo instante.

O que a *pipeline* de fato mede, e registra ao final de cada sessão (`eval_telemetry`), é a **latência total por turno**: o tempo decorrido entre o envio de uma requisição ao LLM e o recebimento completo da resposta, somado ao longo de todas as chamadas de um turno quando o agente decide executar uma ou mais *Actions* antes de finalizar (Seção de Arquitetura da Plataforma Tech4AI). Como o agente pode encadear múltiplas chamadas de ferramenta por turno, essa latência acumulada tende a ser bem maior do que a de uma única chamada isolada ao modelo, e é essa latência acumulada, não uma estimativa de TTFT ou de *throughput*, que compõe a nota L_{norm} do Score Final.

- **Impacto no Usuário:** Em ambientes de atendimento síncrono, a latência percebida é o principal indicador de "percepção de inteligência"; atrasos elevados geram fricção e a sensação de sistema inoperante.
- **Critério de Aceitação:** O teto de latência por turno utilizado na normalização da nota L_{norm} do Score Final (Seção de Critério de Seleção Final) é de 30 segundos, valor calibrado a partir de turnos reais que podem envolver múltiplas chamadas de ferramenta. Turnos mais rápidos recebem pontuação de latência proporcionalmente maior.
- **Volume de *tokens* como proxy complementar:** na ausência de *throughput* medido por *token*, o volume total de *tokens* processados por sessão (Seção de Resultados) funciona como um proxy adicional de esforço computacional e, indiretamente, de tempo de processamento, sem isolar uma taxa de geração por segundo.

C. *Capacidade Resolutiva e Sucesso Operacional*

Esta dimensão avalia a eficácia técnica do modelo em interagir com ferramentas externas e manter a coerência lógica durante a execução de tarefas complexas. É o pilar que garante que o agente não apenas "fale bem", mas "resolva problemas".

*1) **Pass Rate (Taxa de Sucesso Operacional)**:* O *Pass Rate* é a métrica soberana de aprovação binária da pesquisa. Ele determina se o modelo candidato é capaz de atingir ou superar o padrão de qualidade da Base 0 em termos de completude resolutiva e precisão factual. Um atendimento é contabilizado como "Sucesso"(*Pass*) apenas quando o modelo cumpre cumulativamente:

- **Task Completion Rate (TCR):** O modelo deve atingir o estado final do fluxo transacional (ex: confirmação de cancelamento ou alteração cadastral) utilizando as ferramentas disponíveis, sem abandono prematuro do protocolo. Um indicador binário estrito (`tcr_pass`) é puramente determinístico, mas a nota contínua de TCR usada no *gate* de Pass Rate pondera a conclusão dos passos, a eficiência do caminho percorrido e a satisfação do fechamento (Seção de Metodologia), e seu componente de conclusão soma o maior valor entre o match determinístico e a nota do juiz de competência de ferramentas quando a tool diverge do baseline.
- **Tool Calling Accuracy (TCA):** combina uma camada determinística (validade sintática do JSON, existência da *action* no catálogo, parâmetros obrigatórios e equivalência da sequência de tools invocadas em relação ao transcript de referência da Base 0) com uma camada de julgamento contextual: quando a tool chamada diverge do passo esperado do baseline, o mesmo juiz de competência de ferramentas (LLM-as-a-Judge) avalia se a ação é pertinente ao contexto da conversa, e essa nota compõe parte da TCA contínua usada no *gate* de Pass Rate (Seção de Metodologia).
- **Integridade Factual:** A resposta final não deve conter informações inventadas ou desprovidas de fundamentação no contexto fornecido (alucinação). A validação é realizada via LLM-as-a-Judge (avaliação holística da sessão) e regras de negócio.

*2) **Retenção de Memória ao Longo da Sessão**:* Diferente de um teste sintético de degradação por tamanho de contexto (*lost-in-the-middle*), a mitigação deste risco na plataforma Tech4AI é **arquitetural**, não uma métrica isolada calculada pela *pipeline*: o componente de **Memory** (Seção de Arquitetura da Plataforma Tech4AI) reinjeta os dados críticos do atendimento (identificação do usuário, progresso da jornada, informações já coletadas) diretamente no *system prompt* a cada turno, por meio de `update_memory_and_time_in_system_prompt` (Seção de Metodologia). Isso significa que a continuidade dos fatos operacionalmente relevantes não depende exclusivamente da capacidade nativa do modelo de atender corretamente a informações posicionadas no meio de um contexto longo: mesmo que um candidato tivesse esse tipo de limitação, os dados essenciais estão sempre presentes na porção mais recente do *prompt*, não apenas implícitos em turnos anteriores da conversa.

- **O que a *pipeline* de fato avalia:** a paridade de referência do motor de *replay* garante que cada modelo candidato receba exatamente a mesma injeção de memória que o baseline recebeu (Seção de Metodologia); qualquer falha de continuidade observada nos resultados é, portanto, atribuível a como o modelo candidato usa (ou não) essa informação já fornecida, não a uma falha da arquitetura de retenção em si.
- **Limite reconhecido:** esta pipeline não inclui um teste sintético dedicado de degradação por tamanho bruto de contexto (sessões deliberadamente alongadas além do observado na Base 0); a sessão mais longa da Base 0 tem 33 turnos de usuário (Seção de Dataset), o que delimita a faixa de duração de atendimento coberta por esta avaliação.

D. *Qualidade Comportamental e Governança*

Esta dimensão foca na experiência do usuário e na segurança da interação, garantindo que o modelo não seja apenas funcional, mas também aderente à identidade da marca.

*1) **Experiência do Cliente e Tom de Voz**:* Consiste na validação do desempenho do modelo especificamente no idioma Português do Brasil (PT-BR).

- **Regionalização:** O modelo é avaliado quanto à compreensão de nuances linguísticas, gírias e regionalismos presentes na Base 0
- **Persona:** Verifica-se se a resposta gerada mantém a aderência cultural, a empatia e o tom de voz institucional esperado.

*2) **Hallucination Rate (Crítico e Geral)**:* O *Hallucination Rate* é mensurado em duas dimensões para permitir granularidade na análise de risco:

- **Hallucination Rate$_{\text{crítico}}$:** Percentual de respostas que contêm alucinações com impacto direto na decisão do usuário (ex.: valores incorretos em confirmações, datas ou protocolos inventados). Requisito eliminatório: deve ser 0%.
- **Hallucination Rate$_{\text{geral}}$:** Percentual de respostas com qualquer tipo de informação inventada ou desprovida de fundamentação no contexto, incluindo imprecisões menores. O limite aceitável é $\leq 2\%$.

VII. Avaliação Automatizada: LLM-as-a-Judge

Para viabilizar a análise em escala sem comprometer a profundidade qualitativa, o projeto adota a metodologia *LLM-as-a-Judge*, classificada tecnicamente como *Model-Based Evaluation* (Avaliação Baseada em Modelo) ou Avaliação de Referência Semântica. Esta abordagem preenche a lacuna entre as métricas determinísticas tradicionais e a inspeção humana, permitindo uma análise heurística da intenção e resolução do atendimento. O LLM-as-a-Judge é aplicado à avaliação **holística da sessão**, considerando o fluxo completo do atendimento. Avaliações unitárias (por turno ou por ação) são preferencialmente realizadas por métricas de base determinística, garantindo reprodutibilidade; as regras de negócio são inteiramente determinísticas, enquanto o TCR e a TCA contínuos são a exceção parcial, combinando essa mesma base determinística com uma nota de um juiz de competência de ferramentas quando a chamada diverge do baseline (Seção de Metodologia). Fora desse caso pontual, o Judge complementa

as dimensões que exigem interpretação semântica e não admitem automação determinística, sobretudo na avaliação holística da sessão.

A. Fundamentação e Protocolo de Julgamento

A eficácia da metodologia *LLM-as-a-Judge* baseia-se na superioridade cognitiva do avaliador em relação ao modelo sob teste. Para garantir a isenção e a precisão do veredito, o protocolo exige que o modelo juiz possua capacidades de raciocínio superiores (*Reasoning*) e seja, obrigatoriamente, de uma arquitetura ou família distinta dos modelos avaliados. Esta estratégia visa mitigar o *self-preference bias* (viés de autopreferência), onde um modelo tende a atribuir notas maiores a respostas que mimetizam seu próprio padrão sintático.

A temperatura usada nas chamadas ao painel de juízes não recebe um ajuste especial de pesquisa para reduzir aleatoriedade, e é definida por uma configuração própria e independente da usada pelo candidato em cada sessão (Seção de Metodologia).

B. Múltiplos Juízes e Tentativas

Para aumentar a robustez e reduzir a variância do julgamento, o projeto adota uma estratégia de **múltiplos juízes**: a mesma **sessão** é avaliada por dois ou mais modelos juízes distintos, cada um de arquitetura ou família diferente dos modelos avaliados. A combinação das notas desses juízes segue duas regras distintas, dependendo do que está sendo medido:

- **Nota de qualidade e veredito da sessão:** calcula-se a **mediana** entre os juízes: o valor "do meio"da distribuição, que evita que uma avaliação isoladamente muito alta ou muito baixa distorça o resultado final.
- **Detecção de alucinação:** a regra é deliberadamente mais rígida. Em vez de uma média ou de uma votação por maioria, prevalece a classificação **mais grave** encontrada entre os juízes (nenhuma alucinação $<$ alucinação geral $<$ alucinação crítica). Ou seja, basta que **um único juiz** identifique uma alucinação crítica para que a sessão inteira seja marcada como tal, mesmo que os demais juízes não tenham percebido o problema. Essa escolha prioriza a segurança sobre a conveniência estatística: para um agente de atendimento em produção, é preferível investigar um alarme falso ocasional a deixar passar uma alucinação real por diluição na média.

Adicionalmente, quando se deseja medir a estabilidade do candidato, são realizadas n tentativas de geração com a mesma entrada; cada resposta gerada passa pelo mesmo painel de juízes. Essa abordagem permite triangulação do veredito e maior confiança na decisão final de aprovação ou reprovação do modelo.

C. Estrutura do Prompt Avaliador

O *prompt* de julgamento efetivamente usado pela *pipeline* para a avaliação holística de sessão é estruturado sobre quatro elementos, garantindo que o modelo juiz atue de forma consistente e auditável:

- **Dados quantitativos da sessão**: notas por passo já calculadas nas etapas anteriores de avaliação (*assembly_score*, majoritariamente determinístico, com comparação literal ao registro; *context_score*, a nota do juiz de competência de ferramentas sobre a adequação ao contexto da conversa, Seção de Metodologia) e o desfecho da sessão (se foi encerrada e por qual ferramenta). O prompt instrui explicitamente o juiz a não tratar um *assembly_score* baixo como sinônimo de mau atendimento quando o *context_score* e o desfecho para o usuário forem adequados, já que o candidato pode ter resolvido o pedido por um caminho de *tools* diferente do registro de referência.
- **Regra de veredito (*golden rule*)**: o *status* mais baixo (*poor*) é reservado para falha grave (pedido ignorado, dado perigoso, conversa incompleta quando deveria fechar); se o agente resolveu o pedido e encerrou adequadamente, o veredito mínimo já é *adequate*, com *good*/*excellent* reservados a casos de contexto e desfecho fortes.
- **Tratamento de artefatos de mascaramento**: o prompt instrui explicitamente o juiz a reconhecer trechos mascarados em campos numéricos (documentos, contatos, endereços) como artefato do processo de mascaramento (Seção de Dataset), e não como falha do agente, dado sintético ausente ou motivo de alucinação.
- **Formato de saída estruturado**: a resposta do juiz é sempre um JSON, nunca texto livre, com um veredito qualitativo, uma justificativa textual de nível de sessão (nunca um resumo turno a turno) e um bloco de métricas detalhadas com nota Likert de 1 a 4.

D. Definição do Prompt de Julgamento

O *System Prompt* padronizado utilizado na avaliação holística de sessão instrui o modelo juiz a devolver um JSON estruturado contendo um veredito qualitativo (*poor*/*adequate*/*good*/*excellent*) e, dentro de `detailed_metrics`, uma nota Likert de 1 a 4 para qualidade em relação à Base 0 (`quality_vs_base0`) e para tom/clareza (`tone_clarity`), além do tipo de alucinação identificado (`none`/`general`/`critical`) e trechos curtos de evidência (resposta de referência vs. resposta do candidato) que sustentam o veredito de alinhamento. A nota `quality_vs_base0`, de 1 a 4, é a que efetivamente existe na saída do juiz; para compor o Score Final (Seção de Critério de Seleção Final), ela é normalizada linearmente para o intervalo 0 a 1.

Para reduzir ambiguidade e garantir auditabilidade, o prompt exige uma justificativa textual associada a cada nota, não apenas o valor numérico isolado. A Listagem 1 apresenta uma versão condensada do prompt efetivamente utilizado, preservando as regras centrais e o formato de saída; trechos de estilo linguístico (recomendações de português do Brasil) e exemplos ilustrativos de máscara de dados foram omitidos por brevidade e por não pertencerem às sessões sintéticas avaliadas.

Listagem 1: Estrutura do Judge Prompt (avaliação holística de sessão)

```
SESSION_REPLAY_JUDGE_SYSTEM = """
Voce e avaliador senior de agentes conversacionais.
Recebe o resultado de um REPLAY de uma sessao completa.

Escreva todos os campos textuais em portugues do Brasil,
com tom profissional e acolhedor.

Dados da sessao: tool_steps traz, por passo, assembly_score
(comparacao literal com o registro de referencia) e
context_score (adequacao ao contexto da conversa); nao trate
um assembly_score baixo como sinonimo de mau atendimento se o
context_score e o desfecho para o usuario forem adequados, o
candidato pode ter resolvido por um caminho de tools diferente
do registro. session_outcome indica se a sessao foi encerrada
e por qual ferramenta.

Regra de ouro: "poor" e reservado para falha grave (pedido
ignorado, dado perigoso, conversa incompleta quando deveria
fechar). Se o agente resolveu o pedido e encerrou
adequadamente, o veredito minimo ja e "adequate"; use "good"
ou "excellent" quando contexto e desfecho forem fortes, mesmo
com divergencias pontuais de tool vs. registro.

Anonimizacao do dataset: trechos mascarados em campos
numericos (documentos, contatos, enderecos) sao artefato do
pipeline de mascaramento, nao falha do agente. Nao
penalize tom ou clareza, e nao marque como alucinacao, por
causa desses placeholders.

Texto principal: escreva UM comentario continuo sobre o
contexto geral da sessao (arco do pedido, desfecho, tom
global); nunca enumere ou resuma turno a turno.

Responda SOMENTE com JSON valido:

{
  "score": <float 0.0-1.0>,
  "verdict": "<poor|adequate|good|excellent>",
  "reasoning": "<2 a 5 frases, visao global da sessao>",
  "tools_assessment": "<1 a 3 frases sobre o uso de tools>",
  "conversation_assessment": "<1 a 4 frases sobre o dialogo>",
  "detailed_metrics": {
    "quality_vs_base0": <int 1-4>,
    "tone_clarity": <int 1-4>,
    "hallucination_type": "<none|general|critical>",
    "quality_rationale": "<2 a 4 frases>",
    "tone_rationale": "<1 a 3 frases>",
    "alignment_evidence_reference": "<trecho curto>",
    "alignment_evidence_system": "<trecho curto>",
    "alignment_verdict": "<match|partial|mismatch>"
  }
}
"""
```

VIII. Provedores

Os modelos candidatos foram acessados por meio de duas infraestruturas de nuvem. A grande maioria foi servida pela **Azure AI**: toda a família OpenAI (GPT-4o, GPT-4.1 e GPT-5) e os modelos de peso aberto de terceiros (GPT-OSS-120B e Kimi-K2.5), cada um configurado como um *deployment* nomeado dentro do catálogo de modelos da Azure AI Foundry. A família **Gemini** foi a única servida diretamente pelo **Google Cloud Platform**, via Vertex AI.

A. Roteamento Único e Adaptações por Provedor

O código de integração segue uma regra de roteamento simples: se o nome do modelo começa com "gemini"e há credenciais do Google configuradas, a chamada é enviada ao Vertex AI; caso contrário, ela é enviada à Azure, usando o próprio nome do modelo como identificador do *deployment* no endpoint. Essa arquitetura de **roteamento único** é o que torna o experimento de *LLM-swap* operacionalmente simples: na maioria dos casos, trocar de modelo significa apenas trocar um nome de configuração, sem reescrever a integração.

Essa simplicidade na superfície não elimina a necessidade de ajustes pontuais por trás da API unificada, feitos ao longo do desenvolvimento à medida que cada modelo revelava um comportamento diferente do esperado:

- **Google Cloud (Vertex AI):** o Vertex utiliza um formato próprio de mensagens e de definição de ferramentas (*function calling*), diferente do padrão adotado pela OpenAI e usado como referência no restante da pipeline. Foi necessária uma camada de tradução nos dois sentidos: as mensagens e as definições de *tools* são convertidas do formato OpenAI para o formato Vertex antes do envio, e a resposta do Vertex é convertida de volta para o formato OpenAI antes de seguir para o motor de avaliação: assim, o restante da pipeline nunca precisa saber qual provedor de fato respondeu.
- **Azure, família GPT-5:** os *deployments* GPT-5* na Azure recusam o parâmetro de temperatura com qualquer valor diferente de 1 (o provedor devolve um erro HTTP explícito) e exigem um nome de parâmetro diferente para o limite de *tokens* de saída. A pipeline reconhece esses *deployments* pelo nome e ajusta a requisição automaticamente, sem exigir configuração manual por rodada.
- **Azure, GPT-OSS-120B:** diferente dos demais modelos deste estudo, o GPT-OSS-120B não responde no formato de *chat completions* usado como referência pelo restante da pipeline. Ele utiliza nativamente o formato **Harmony**, o *template* de conversação próprio da família de modelos abertos *gpt-oss* da OpenAI, que estrutura a resposta em canais distintos (por exemplo, raciocínio interno e resposta final) em vez do formato único de mensagem esperado por um *endpoint* de *chat completions* convencional. Essa diferença de formato exigiu, durante o período em que esse modelo foi testado, um tratamento específico na etapa que extrai a decisão do modelo (se ele quis responder em texto ou acionar uma ferramenta). Ainda assim, como detalhado na próxima seção, esse foi o modelo com a maior taxa de turnos em que a pipeline não identificou nem uma chamada de ferramenta nem uma resposta em texto.

IX. Modelos Candidatos

A Tabela III apresenta os 8 modelos candidatos efetivamente avaliados na pipeline, categorizados por *tier* de custo e provedor de acesso.

TABELA III
MODELOS CANDIDATOS CATEGORIZADOS POR TIER E PROVEDOR.

Modelo	Provedor	Tier	Observação
Gemini 2.5 Flash	GCP (Vertex AI)	Budget	Tende a responder em texto em vez de acionar a ferramenta esperada.
GPT-4.1 nano	Azure	Budget	Alta taxa de escolha de ferramenta incorreta.
GPT-5 mini	Azure	Mid-range	Único candidato com alucinação crítica registrada.
GPT-5 nano	Azure	Budget	Menor *Pass Rate* entre os modelos de alto volume testados.
GPT-5.4 mini	Azure	Mid-range	Melhor *Pass Rate* observado entre os modelos testados.
GPT-5.4 nano	Azure	Budget	Menor custo e latência entre os candidatos; aprovado, com a maior base de evidência entre os aprovados.
GPT-OSS-120B	Azure	Mid-range	Peso aberto; menor *Pass Rate* observado, mesmo com tratamento específico de formato durante o período em que foi testado.
Kimi-K2.5	Azure	Mid-range	Peso aberto; maior incidência de limitação de taxa do provedor (HTTP 429).

A. Nota sobre Modelos Excluídos da Comparação

Três modelos aparecem em algumas execuções da pipeline sem, no entanto, integrarem a comparação de candidatos deste artigo:

- **GPT-4o mini:** não é tratado como candidato porque a Base 0 (Seção de Dataset) é composta por conversas sintéticas de referência geradas pelo próprio GPT-4o mini, o que tornaria uma comparação direta contra essa referência circular. Por isso, o GPT-4o mini esteve presente na grande maioria das execuções da pipeline não como um concorrente, mas como **controle de validação**: sua presença confirma que o motor de *replay* reproduz fielmente o comportamento de referência antes de submeter os demais candidatos às mesmas condições.
- **GPT-4o e GPT-4.1:** não foram, de fato, testados como candidatos a substituição. Seu papel na pipeline é o de **modelos juízes** do painel de LLM-as-a-Judge (Seção de LLM-as-a-Judge).

As Seções de Resultados, Discussão e Veredito Final tratam exclusivamente dos oito modelos candidatos genuínos.

B. Desempenho Observado por Modelo

Cada sessão da Base 0 pode ser reavaliada em múltiplas rodadas da pipeline; a Tabela IV agrega, por modelo, a *Pass Rate* (percentual de sessões que atingiram cumulativamente TCR e TCA, Seção de Critérios do Projeto) e o motivo de falha mais frequente identificado pela pipeline.

TABELA IV
DESEMPENHO POR MODELO: PASS RATE E PRINCIPAL MOTIVO DE FALHA.

Modelo	Pass Rate	Falha mais frequente
Gemini 2.5 Flash	78,5%	Respondeu em texto, sem acionar ferramenta
GPT-4.1 nano	68,7%	Ferramenta incorreta
GPT-5 mini	62,9%	Ferramenta incorreta
GPT-5 nano	57,3%	Ferramenta incorreta
GPT-5.4 mini	92,5%	Ferramenta incorreta (baixa incidência)
GPT-5.4 nano	73,3%	Ferramenta incorreta
GPT-OSS-120B	10,6%	Nenhuma ferramenta nem texto (decisão não reconhecida)
Kimi-K2.5	77,0%	Limitação de taxa do provedor (HTTP 429)

Os dados revelam que os modelos avaliados falham de formas qualitativamente diferentes, e nem sempre pelo motivo mais intuitivo (falta de inteligência do modelo). Quatro padrões se destacam:

1) **Escolha de ferramenta incorreta (mais comum):** para a maioria dos modelos da família GPT, quando algo dá errado, é porque o modelo decidiu acionar uma ferramenta plausível, mas que não era a esperada para aquele passo do atendimento, por exemplo, consultar uma dúvida geral quando o fluxo esperava o registro formal de uma ocorrência. É um erro de julgamento sobre *qual* ação tomar, não de capacidade de seguir o formato exigido.
2) **Resposta em texto em vez de ação (Gemini):** o Gemini 2.5 Flash apresenta um padrão de falha diferente do restante: em vez de escolher a ferramenta errada, ele tende a simplesmente responder ao usuário em linguagem natural quando o fluxo esperava que uma ferramenta fosse acionada. Isso sugere uma diferença de calibração entre "quando agir"e "quando apenas conversar"em relação aos modelos da família GPT, mesmo com prompts e ferramentas disponíveis idênticos.
3) **Limitação de taxa do provedor, não de raciocínio (Kimi-K2.5):** para o Kimi-K2.5, a causa de falha mais comum não é um erro de julgamento do modelo, e também não é uma instabilidade genérica de rede: das falhas técnicas registradas, 100% são o mesmo erro específico, HTTP 429 (*Too Many Requests*), devolvido pela Azure quando a cota de requisições por minuto do *deployment* é excedida; não há nenhuma ocorrência de *timeout* ou erro 5xx entre elas. A pipeline já tenta novamente automaticamente antes de desistir (*backoff* exponencial, Seção de Metodologia); os cerca de um quarto das sessões avaliadas que ainda assim encontram esse problema são aquelas em que o limite de requisições foi excedido de forma persistente o suficiente para esgotar as tentativas. Isso indica que a cota configurada para esse *deployment* na Azure é, na prática, mais restritiva do que a dos modelos nativos da OpenAI, e que parte do desempenho aparentemente pior desse modelo é, na verdade, limitação de taxa do provedor, não incapacidade do modelo em si.
4) **Decisão não reconhecida pela pipeline (GPT-OSS-120B):** o GPT-OSS-120B se destaca negativamente por um motivo estrutural e identificado: em 87% das sessões em que participou, a pipeline não conseguiu identificar nem uma chamada de ferramenta nem uma resposta em texto: o turno terminou sem uma decisão reconhecível. A causa é o formato nativo de saída desse modelo (*Harmony*, Seção de Provedores), que não segue o padrão de *chat completions* usado como referência por todo o restante da pipeline. Enquanto o GPT-OSS-120B esteve em teste, a pipeline contou com um tratamento específico para lidar com essa diferença, mas mesmo assim a taxa residual de turnos sem decisão reconhecível permaneceu alta, evidência de que aquele tratamento não cobria a totalidade dos casos da Base 0, o que é retomado na Seção de Limitações como trabalho futuro.

Vale destacar ainda que o GPT-5 mini foi o único modelo, entre os 8 candidatos avaliados, com ocorrência de alucinação classificada como crítica pelo painel de juízes, um resultado pontual (2 ocorrências), mas relevante por violar o requisito eliminatório de Hallucination $\text{Rate}_{\text{crítico}} = 0\%$ definido na Seção de Critérios do Projeto, e que impacta diretamente seu veredito final.

X. CRITÉRIO DE SELEÇÃO FINAL

Após a execução da *pipeline* de testes, os modelos candidatos são submetidos a uma matriz de decisão multidimensional que determina seu *status* de aprovação. Para viabilizar a comparação entre métricas de naturezas distintas (percentuais de acerto, milissegundos e dólares), cada indicador é convertido, a partir dos dados coletados nos experimentos, para uma escala linear de 1 a 4, na qual 4 representa o cenário ideal e 1 o pior desempenho observado no grupo.

O cálculo do Score Final (S) é definido pela média ponderada das notas normalizadas, em escala de 1 a 4:

$$S = 50\% \cdot Q + 20\% \cdot L_{norm} + 30\% \cdot C_{norm} \quad (1)$$

Nesta equação, Q, L_{norm} e C_{norm} são as notas na escala original de 1 a 4. A versão normalizada para a escala 0 a 1 usada internamente pela pipeline (e exibida na Seção de Resultados) é obtida pela mesma transformação linear $(nota-1)/3$; o Score Final em porcentagem ($S_{\%}$) é obtido a partir do S em escala de 1 a 4 por essa mesma fórmula, $(S-1)/3\times100$ (Seção de Veredito Final).

A composição dos pesos reflete a priorização da excelência técnica, sem negligenciar a viabilidade operacional:

- **Qualidade (Q):** a nota Likert de 1 a 4 atribuída pelo painel de juízes (Seção de LLM-as-a-Judge) é normalizada linearmente para a escala 0 a 1 usada internamente pela pipeline, pela fórmula $(nota-1)/3$: a nota 1 (inadequada) corresponde a $Q_{norm} = 0$, e a nota 4 (excelente) corresponde a $Q_{norm} = 1$.
- **Latência (L_{norm}):** Nota de 1 a 4 inversamente proporcional à latência observada por turno (Seção de Critérios do Projeto). Modelos mais rápidos recebem pontuações maiores.
- **Custo (C_{norm}):** Nota de 1 a 4 inversamente proporcional ao Custo por Sessão (CPS), favorecendo modelos com menor impacto financeiro no plano Enterprise.

Essa abordagem garante que o modelo selecionado para produção não seja apenas o mais inteligente em termos linguísticos, mas também o mais sustentável para a infraestrutura de *DevOps* e para a experiência do usuário final.

XI. Resultados

Esta seção apresenta os resultados agregados da execução da pipeline sobre os 8 modelos candidatos genuínos (Seção de Modelos Candidatos), consolidando o Score Final (S), a *Pass Rate*, as taxas de alucinação e os proxies de custo e latência observados ao longo de todas as avaliações de sessão realizadas. O GPT-4o mini (controle de validação do motor de *replay*) e o GPT-4o e o GPT-4.1 (usados como modelos juízes, não como candidatos) não integram esta comparação, pelos motivos detalhados na Seção de Modelos Candidatos.

A. Nota Metodológica sobre o Custo

As execuções registradas neste trabalho não dispõem de telemetria de custo em dólares por chamada (`cost_usd`). Por essa razão, a nota de Custo (C_{norm}) usada na composição do Score Final é calculada a partir do volume total de *tokens* processados por sessão, normalizado contra um teto de referência: funcionando como um **proxy** do custo computacional, e não como um valor monetário de Custo por Sessão (CPS) real; essa é a métrica usada de forma consistente para os 8 candidatos na matriz de decisão. Como complemento, a subseção "Estimativa de Custo Real (USD)"mais adiante nesta seção converte o mesmo consumo de *tokens* para dólares usando os preços públicos de tabela de cada provedor. A conversão para valores de custo contratados diretamente com cada provedor, quando os contratos comerciais estiverem consolidados, permanece indicada como trabalho futuro na Seção de Limitações.

B. Score Final, Pass Rate e Alucinação por Modelo

A Tabela V apresenta, para cada modelo, o Score Final médio ($S_{\%}$), suas três componentes normalizadas (Q, L, C, na escala 0 a 1 utilizada internamente pela pipeline, equivalente à escala de 1 a 4 da Seção de Critério de Seleção Final), a *Pass Rate*, e as duas taxas de alucinação (Hallucination $\text{Rate}_{\text{crítico}}$ e Hallucination $\text{Rate}_{\text{geral}}$, esta última já incorporando as ocorrências críticas).

C. Veredito Aplicado

O veredito de cada modelo é obtido aplicando, uma única vez, a matriz de decisão da Seção de Veredito Final (Score médio, *Pass Rate* e as duas taxas de alucinação, de forma cumulativa) sobre os valores já apresentados na Tabela V. É importante destacar que esses valores não são uma média simples entre execuções independentes: são calculados ponderando pelo volume de sessões avaliadas, de modo que um modelo testado em centenas de sessões ao longo de várias execuções não é comparado em pé de igualdade com um modelo testado em poucas dezenas de sessões concentradas em execuções curtas. Essa ponderação por volume evita que uma execução pequena, favorável ou desfavorável, pese o mesmo que uma execução grande na definição do veredito final.

Sob esse critério, o **GPT-5.4 mini**, o **GPT-5.4 nano** e o **Kimi-K2.5** atingem simultaneamente Score $\geq$ 75%, Pass Rate $\geq$ 70% e ambas as taxas de alucinação dentro do limite, recebendo o *status* de **APROVADO**. O **GPT-5 nano** recebe **PARCIAL**: Score ou Pass Rate insuficientes para aprovação plena, mas dentro dos limites de alucinação. Os demais quatro modelos (Gemini 2.5 Flash, GPT-4.1 nano, GPT-5 mini e GPT-OSS-120B) são **REPROVADOS**, na maioria dos casos por Hallucination $\text{Rate}_{\text{geral}}$ acima de 2% (ou Hallucination $\text{Rate}_{\text{crítico}}$ acima de 0% no caso do GPT-5 mini), mesmo com Score e Pass Rate competitivos.

O resultado mais notável desta seção é que a taxa de alucinação (não o Score Final nem a *Pass Rate*) é o fator decisivo na maioria dos vereditos de reprovação. Três dos oito modelos candidatos (Gemini 2.5 Flash, GPT-4.1 nano e GPT-5 mini) apresentam Score Final acima de 79% e *Pass Rate* competitiva, mas são reprovados por excederem o limite de 2% de Hallucination $\text{Rate}_{\text{geral}}$ definido neste estudo, mesmo sem nenhuma ocorrência de alucinação crítica. O **Kimi-K2.5** é o único candidato a registrar Hallucination $\text{Rate}_{\text{geral}}$ de 0% em toda a amostra, embora, como discutido mais adiante, essa aprovação venha acompanhada de uma ressalva prática relevante. O **GPT-5.4 nano**, também aprovado, tem a Hallucination $\text{Rate}_{\text{geral}}$ mais próxima do limite eliminatório entre os três aprovados (cerca de 2% contra um teto de 2%), uma margem de segurança estreita discutida adiante nesta seção.

D. Consistência do Veredito entre Execuções

O veredito da subseção anterior é calculado sobre o conjunto agregado, ponderado por volume de sessões, de todas as

TABELA V

SCORE FINAL, COMPONENTES NORMALIZADAS, PASS RATE E HALLUCINATION RATE POR MODELO.

Modelo	$S_{\%}$	Q	L	C	**Pass**	$H_{\textbf{crít}}$	$H_{\textbf{geral}}$
Gemini 2.5 Flash	79,7	0,79	0,85	0,77	78,5%	0,0%	2,9%
GPT-4.1 nano	82,9	0,82	0,92	0,79	68,7%	0,0%	2,4%
GPT-5 mini	79,2	0,74	0,81	0,86	62,9%	1,6%	3,2%
GPT-5 nano	74,1	0,76	0,68	0,74	57,3%	0,0%	1,2%
GPT-5.4 mini	86,4	0,82	0,92	0,89	92,5%	0,0%	0,9%
GPT-5.4 nano	83,1	0,75	0,92	0,91	73,3%	0,0%	2,0%
GPT-OSS-120B	64,9	0,44	0,92	0,81	10,6%	0,0%	2,1%
Kimi-K2.5	80,0	0,79	0,69	0,89	77,0%	0,0%	0,0%

avaliações de cada modelo. Modelos de linguagem são, por natureza, probabilísticos: o mesmo modelo pode produzir um resultado diferente a cada execução independente da pipeline, de modo que uma reprovação isolada, ou mesmo um agregado formado por poucas execuções, não deveria ser lida como uma característica estável e definitiva do modelo. Para investigar isso, sem substituir o veredito ponderado por volume já apresentado, a mesma matriz de decisão foi recalculada de forma independente para **cada execução** da pipeline, permitindo observar, execução por execução, com que frequência cada modelo atinge cada *status* (Tabela VI). Essa checagem funciona como verificação adicional de robustez: ela mostra se a reprovação (ou aprovação) de um modelo se repete de forma consistente entre execuções independentes, ou se é dependente da composição específica de sessões observada em cada uma.

TABELA VI

DISTRIBUIÇÃO DE VEREDITO POR EXECUÇÃO INDEPENDENTE DA PIPELINE, POR MODELO.

Modelo	**APROVADO**	**PARCIAL**	**REPROVADO**
Gemini 2.5 Flash	47,2%	8,3%	44,4%
GPT-4.1 nano	41,2%	31,4%	27,5%
GPT-5 mini	33,3%	33,3%	33,3%
GPT-5 nano	23,1%	53,8%	23,1%
GPT-5.4 mini	84,6%	0,0%	15,4%
GPT-5.4 nano	50,9%	23,6%	25,5%
GPT-OSS-120B	0,0%	33,3%	66,7%
Kimi-K2.5	80,0%	20,0%	0,0%

Essa análise revela um padrão importante: alguns modelos com veredito agregado **REPROVADO** estão, na verdade, longe de ser consistentemente ruins. O caso mais notável é o **Gemini 2.5 Flash** (47,2% aprovado *contra* 44,4% reprovado, praticamente empatado) e, em seguida, o **GPT-5 mini** (33,3% em cada categoria). Para esses dois modelos, a reprovação agregada parece mais dependente da composição específica de sessões observadas do que uma característica estrutural do modelo, merecendo mais execuções antes de uma conclusão definitiva, e não um descarte definitivo.

O contraponto necessário é o **GPT-OSS-120B**: em nenhuma de suas execuções ele atingiu o *status* de APROVADO. Diferente dos casos anteriores, essa consistência **reforça**, em vez de contradizer, o veredito agregado: aqui, o desempenho fraco não parece ser fruto de variância estatística, mas de um padrão estrutural, coerente com a causa já identificada na Seção de Modelos Candidatos (formato de resposta nativo *Harmony*, incompatível com o padrão de *chat completions*). Já o **Kimi-K2.5** e o **GPT-5.4 mini**, mantêm essa aprovação de forma consistente (80,0% e 84,6% das execuções, respectivamente), reforçando, e não apenas assumindo, a confiança no veredito agregado para ambos. O **GPT-5.4 nano**, também aprovado no agregado, é o caso menos consolidado dos três: aprovado em apenas 50,9% de suas execuções, com 25,5% delas reprovadas, um padrão mais próximo da volatilidade observada no Gemini e no GPT-5 mini (do lado dos reprovados) do que da consistência do Kimi-K2.5 e do GPT-5.4 mini.

E. Custo Computacional e Latência

A Tabela VII detalha os proxies de custo computacional (*tokens* totais e número de chamadas ao LLM por sessão) e a latência total por sessão observados durante o *replay*. Como o número de vezes que cada sessão da Base 0 foi reavaliada varia (Seção de Modelos Candidatos), a média por sessão não é calculada diretamente sobre todas as avaliações somadas: primeiro calcula-se a média de cada sessão da Base 0 individualmente entre suas repetições, e só então a média dessas médias é tirada por modelo. Isso evita que uma sessão reavaliada com muito mais frequência do que as demais (por ter entrado em mais rodadas de teste) tenha peso desproporcional no resultado final.

TABELA VII

PROXIES DE CUSTO COMPUTACIONAL E LATÊNCIA POR SESSÃO, POR MODELO.

Modelo	**Tokens/sessão**	**Chamadas LLM/sessão**	**Latência total/sessão**
Gemini 2.5 Flash	180.869	40,7	169,1 s
GPT-4.1 nano	151.315	34,2	80,8 s
GPT-5 mini	93.797	25,9	148,2 s
GPT-5 nano	190.151	36,9	366,8 s
GPT-5.4 mini	112.257	27,7	57,7 s
GPT-5.4 nano	75.026	21,6	44,1 s
GPT-OSS-120B	154.542	47,0	129,5 s
Kimi-K2.5	70.285	16,6	167,8 s

Dois pontos chamam a atenção nesta tabela. Primeiro, o **GPT-5.4 nano** tem a menor latência total por sessão entre os candidatos (44,1 segundos), e o **Kimi-K2.5** o menor volume de *tokens* (70.285); ambos aprovados, essa combinação de custo e velocidade reforça que os dois modelos aprovados de menor porte são competitivos mesmo frente ao GPT-5.4 mini. Segundo, o **GPT-5 nano** apresenta a maior latência total por sessão (366,8 segundos) entre todos os modelos, mais de

oito vezes a latência do GPT-5.4 nano, uma diferença que não é explicada apenas pelo volume de *tokens*, e que penaliza diretamente sua nota de latência ($L = 0,68$, a mais baixa observada) na composição do Score Final.

F. Estimativa de Custo Real (USD)

A Tabela VIII converte o mesmo consumo médio de *tokens* de entrada e saída por sessão (normalizado em duas etapas, como na Tabela VII, mas agora separando *tokens* de entrada e de saída, cobrados a preços distintos por cada provedor) em uma estimativa de Custo por Sessão (CPS) em dólares. Os preços usados são os de tabela pública (*pay-as-you-go*, sem desconto contratual) vigentes em cada provedor no momento da consulta [1], [2], [8]. Ressalva importante: são os preços públicos de lista, não necessariamente os valores efetivamente contratados pela Tech4AI junto a cada provedor, que podem incluir descontos comerciais não refletidos aqui; a substituição por telemetria de custo contratado é indicada como trabalho futuro na Seção de Limitações.

A tabela também traz o preço de *input* cacheado de tabela de cada provedor, quando disponível. O **GPT-OSS-120B** e o **Kimi-K2.5** não têm um preço de *input* cacheado publicado: o primeiro não tem essa informação divulgada na tabela de preços, e o segundo, segundo a própria documentação da Azure, não oferece suporte a *prompt caching*. Como a pipeline não registra a taxa de acerto de cache de fato observada durante o *replay* (não há telemetria de *cached_tokens* nas chamadas do experimento), o CPS estimado na última coluna assume o cenário mais conservador de **nenhum acerto de cache**, ou seja, todo *input* cobrado ao preço cheio; o preço cacheado é informado apenas como referência de quanto o CPS poderia cair caso a Tech4AI viesse a operar esses modelos em produção com reaproveitamento de contexto.

O resultado mais notável desta estimativa é que ela não reproduz a ordenação da nota de Custo normalizada por *tokens* (C_{norm}) usada no Score Final: o **GPT-5.4 mini**, que recebe $C_{norm} = 0,89$ (Tabela V), tem, na prática, o maior CPS estimado entre os oito candidatos (cerca de $0,093 por sessão), aproximadamente 5,3 vezes o custo do GPT-5.4 nano ($0,017) e 5,8 vezes o do GPT-4.1 nano ($0,016, o mais barato da amostra). A causa não é o volume de *tokens* processados (Tabela VII), mas o preço de tabela por *token* de saída do GPT-5.4 mini ($4,50/milhão), bem mais alto do que o do GPT-5.4 nano ($1,25/milhão) na mesma família de modelos. Esse achado, retomado como argumento metodológico mais amplo na Seção de Discussão, mostra que o proxy de *tokens* pode inverter a ordem de custo real entre modelos quando o preço por *token* varia de forma expressiva entre eles, e não deve ser confundido com uma previsão de Custo por Sessão (CPS) em dinheiro.

G. Leitura Qualitativa: Pontos Fortes e Fracos por Modelo

O veredito da matriz de decisão resume cada modelo em uma única categoria, mas a decisão prática de qual modelo adotar em produção depende de ponderar pontos fortes e fracos específicos que a tabela, isoladamente, não comunica, sobretudo quando um modelo aprovado carrega uma ressalva operacional relevante, ou quando um modelo reprovado ainda tem méritos que justificam acompanhamento futuro. Esta subseção resume essa leitura, modelo a modelo.

- **GPT-5.4 mini:** melhor *Pass Rate* observada (92,5%) e leitura equilibrada em custo, latência e alucinação, sem um ponto fraco evidente nos dados coletados.
- **Kimi-K2.5 (com ressalva prática):** único modelo com 0% de Hallucination $\text{Rate}_{\text{geral}}$ em toda a amostra e boa *Pass Rate* (77%): respondeu bem quando respondeu. Porém, sua latência total por sessão (167,8 segundos) é a terceira mais alta do estudo, e cerca de 27% de suas sessões avaliadas encontraram limitação de taxa do provedor (HTTP 429 na Azure, não erro do modelo). Na prática, essa combinação de latência alta com sensibilidade à cota de requisições o torna um candidato arriscado para atendimento síncrono em tempo real, mesmo estando formalmente aprovado pelos critérios de qualidade: a aprovação reflete a qualidade das respostas, não a prontidão operacional do *deployment*.
- **GPT-5.4 nano (com ressalva de margem):** menor latência total por sessão do estudo e segundo menor volume de *tokens* (atrás apenas do Kimi-K2.5), com a maior base de evidência entre os três aprovados. A ressalva já discutida nesta seção (Hallucination $\text{Rate}_{\text{geral}}$ próxima do teto e veredito por execução menos consolidado dos três) concentra-se em um pequeno número de sessões específicas da Base 0 que se repetem entre execuções (Seção de Discussão).
- **GPT-5 nano:** uma das menores taxas de alucinação geral (1,2%), mas a pior latência total por sessão do estudo (366,8 segundos): um ponto fraco que, na prática, pode inviabilizar seu uso em atendimento em tempo real independentemente do veredito de qualidade.
- **Gemini 2.5 Flash (indefinido por execução):** mantém *Pass Rate* de 78,5% e Score Final de 79,7, mas ultrapassa o limite de Hallucination $\text{Rate}_{\text{geral}}$ no agregado ponderado (2,9%). Por execução, porém, o resultado é quase um empate entre aprovação (47,2%) e reprovação (44,4%): o veredito definitivo exige mais execuções. Adicionalmente, o Gemini tende a responder em texto em vez de acionar a ferramenta esperada (Seção de Modelos Candidatos), um padrão de comportamento a corrigir antes de uma nova rodada.
- **GPT-4.1 nano (competitivo em qualidade):** mantém *Pass Rate* de 68,7% e Score Final de 82,9, mas ultrapassa o limite de Hallucination $\text{Rate}_{\text{geral}}$ no agregado ponderado (2,4%); por execução, atinge o *status* pleno em 41,2% dos casos: resultado intermediário, nem tão volátil quanto o Gemini nem tão consistente quanto o GPT-OSS-120B.
- **GPT-5 mini (indefinido por execução):** nota de custo competitiva (0,86), mas é o único modelo do estudo com ocorrência de alucinação crítica (1,6% das sessões): o requisito eliminatório mais rígido da pesquisa. Por execução, o resultado se divide igualmente entre os três *status* possíveis (33,3% cada), o que, somado ao volume ainda moderado de avaliações, recomenda cautela antes de descartar o modelo.

TABELA VIII
ESTIMATIVA DE CUSTO POR SESSÃO (CPS) EM USD, A PARTIR DE PREÇOS PÚBLICOS DE TABELA.

Modelo	Input (USD/1M)	Input cacheado (USD/1M)	Output (USD/1M)	CPS estimado (USD)
Gemini 2.5 Flash	0,30	0,03	2,50	0,0586
GPT-4.1 nano	0,10	0,025	0,40	0,0158
GPT-5 mini	0,25	0,025	2,00	0,0466
GPT-5 nano	0,05	0,005	0,40	0,0222
GPT-5.4 mini	0,75	0,075	4,50	0,0925
GPT-5.4 nano	0,20	0,02	1,25	0,0173
GPT-OSS-120B	0,15	n/d	0,60	0,0267
Kimi-K2.5	0,60	n/d	3,00	0,0520

- **GPT-OSS-120B (padrão consistente entre execuções):** nota de latência competitiva (0,92) nas sessões em que de fato produz uma resposta reconhecível, mas a menor *Pass Rate* do estudo (10,6%), majoritariamente por sessões em que a pipeline não identifica nem chamada de ferramenta nem texto (Seção de Modelos Candidatos). Diferente dos demais modelos com desempenho fraco, este é o único que nunca atingiu o *status* pleno em nenhuma de suas execuções: o resultado aqui é o mais bem fundamentado estatisticamente do estudo, coerente com uma causa estrutural já identificada, e não com variância de amostra.

XII. DISCUSSÃO

Os resultados apresentados na seção anterior só têm o peso que têm porque foram obtidos sob uma premissa que orientou todas as decisões de desenho deste trabalho: reproduzir um ambiente de referência com a maior fidelidade possível, sem depender de tráfego ao vivo. Esta seção discute essa escolha, o que ela permitiu descobrir, os limites que ela mesma impõe, e como os achados deste estudo se relacionam com a literatura de avaliação de agentes.

A. Fidelidade ao Ambiente de Referência como Filosofia Central

Diferente de um benchmark genérico, o motor de *replay* deste trabalho reconstrói o contexto exato de cada turno selecionado sob paridade com o comportamento de referência do produto: os mesmos blocos de memória e hora injetados no *system prompt*, as mesmas *tools* e mensagens iniciais do baseline antes da primeira mensagem do usuário, o mesmo filtro de histórico após uma transferência entre agentes, a mesma regra de encerramento e o mesmo limite de cinco rodadas de chamada de ferramenta por turno (Seção de Metodologia). Essa é uma diferença qualitativa em relação às abordagens discutidas na Seção de Trabalhos Relacionados: o τ-bench depende de um usuário simulado por outro modelo de linguagem interagindo com bancos de dados sintéticos, e o BFCL avalia chamadas de função contra um gabarito construído para o benchmark. Aqui, o corpus é composto por conversas sintéticas selecionadas, reexecutadas sob regras consistentes de avaliação.

Essa fidelidade, no entanto, tem um limite deliberado: a execução das ferramentas (*Actions*) não é ao vivo. Para manter segurança e reprodutibilidade, ações transacionais são tratadas como respostas controladas no ambiente de avaliação, sem efeito externo. Por isso, o motor de *replay* trata a resposta de uma ferramenta de duas formas: quando a chamada do modelo candidato corresponde ao passo esperado do baseline (nome certo, ou equivalência semântica aceita), a pipeline injeta a resposta de referência prevista para aquele passo; quando a chamada diverge do baseline (nome errado ou uma ação fora do roteiro de referência), a pipeline recorre a uma resposta simulada de falha. Essa resposta simulada não afirma sucesso: ela informa ao modelo que a ferramenta não foi executada com sucesso e que o atendimento deve continuar com as informações disponíveis, desincentivando deliberadamente uma nova tentativa da mesma ação como se ela tivesse funcionado. Essa escolha de design, combinada com o teto de recursão de *tool-calls* por turno (Seção de Metodologia) e uma classificação específica que distingue uma repetição idêntica da mesma *tool* sem motivo aparente de uma nova tentativa plausível após um sinal de falha, é o que impede que uma chamada de ferramenta divergente do baseline degenere em um loop de repetições dentro do mesmo turno.

Esse é o único ponto em que a fidelidade ao ambiente de referência é conscientemente limitada, e vale destacar que o desenho da TCA amortece parcialmente o efeito colateral dessa limitação, combinando duas camadas em vez de isolá-las. A base da *Tool Calling Accuracy* (TCA) de fato compara a chamada do modelo contra a sequência do baseline (Seção de Critérios do Projeto) de forma determinística e rígida por definição. Mas quando a chamada diverge desse passo esperado, um juiz de competência de ferramentas (LLM-as-a-Judge) é instruído a avaliar se a ação é **adequada ao contexto da conversa**, com o registro de referência explicitamente marcado como informativo, não obrigatório de ser igualado, e a nota desse juiz passa a compor a própria TCA contínua (Seção de Metodologia), não uma métrica à parte. Ou seja: um modelo que resolve o problema por um caminho diferente, porém sensato, não é simplesmente penalizado pela rigidez da comparação determinística; a camada de julgamento contextual eleva sua nota dentro da própria TCA quando reconhece competência, o que é o oposto de duas camadas independentes, é uma métrica única que combina rigor determinístico com avaliação contextual pontual.

B. O que a Fidelidade ao Ambiente de Referência Torna Possível

A consequência prática dessa escolha metodológica é que os resultados da Seção de Resultados não são números de benchmark descontextualizados: são leituras acionáveis para uma decisão controlada de substituição de modelo. Quando este estudo aponta que o GPT-4.1 nano falha no critério de Hallucination $\text{Rate}_{\text{geral}}$, essa não é uma inferência a partir de um proxy: é uma observação sobre como esse modelo se comporta especificamente nas instâncias sintéticas de atendimento, nos fluxos transacionais e no tom de voz definidos para a Base 0. Um benchmark público não teria como antecipar esse resultado, porque não carrega o mesmo contexto de referência selecionado para esta avaliação.

C. O Gate de Alucinação como Achado Metodológico

Um dos achados mais relevantes deste trabalho não é sobre um modelo específico, mas sobre o próprio desenho do veredito. Se a decisão de aprovação dependesse apenas do Score Final (uma média ponderada de qualidade, latência e custo), pelo menos três modelos com Score acima de 79% teriam sido recomendados para adoção mesmo excedendo o limite de alucinação aceitável. Foi apenas por o critério de aceitação tratar a Hallucination Rate como um **gate eliminatório**, aplicado depois e independentemente da média ponderada, que esse risco não passou despercebido. Isso sustenta um argumento metodológico mais amplo: em avaliação de agentes que tomam decisões em fluxos críticos, uma média ponderada de qualidade não deveria, sozinha, decidir a aprovação: critérios de segurança precisam operar como filtros independentes, não como mais um componente diluído na média.

D. Variância entre Execuções: Por que um Veredito Agregado Não Basta

A Seção de Resultados recalculou o veredito execução por execução, e não apenas sobre o agregado, exatamente por reconhecer essa natureza probabilística dos modelos de linguagem. Dois dos quatro modelos classificados como REPROVADO no agregado, Gemini 2.5 Flash e GPT-5 mini, oscilam entre aprovação e reprovação de forma quase equilibrada, dependendo apenas de qual conjunto específico de sessões compôs aquela execução. O mesmo tipo de sensibilidade à amostra aparece, de forma diferente, do lado dos modelos aprovados: o GPT-5.4 nano atinge o *status* de APROVADO no agregado, mas seu veredito por execução é o menos consolidado entre os três aprovados, com pouco mais da metade das execuções confirmando a aprovação.

Essa constatação tem uma implicação metodológica direta: um único veredito REPROVADO, isolado, não deveria ser suficiente para descartar um modelo candidato: é necessário observar se a reprovação se repete de forma consistente ao longo de múltiplas execuções independentes, ou se é dependente da amostra específica observada. O caso do **GPT-OSS-120B** ilustra o contraste: sua reprovação se repetiu em todas as suas execuções, sem exceção, o que é uma evidência estatística qualitativamente diferente e mais forte do que uma reprovação agregada isolada. Em outras palavras, nem toda reprovação tem o mesmo peso probatório, e um processo de decisão responsável precisa diferenciar reprovações estruturais (repetidas, coerentes com uma causa identificável) de reprovações dependentes da amostra (que pedem mais dados antes de qualquer conclusão).

E. Funcionamento do LLM-Swap

A arquitetura de roteamento único descrita na Seção de Provedores (em que trocar de modelo significa, na maioria dos casos, apenas trocar um nome de configuração) é o que torna o experimento de *LLM-swap* operacionalmente viável. Mas essa simplicidade na superfície não se estende ao comportamento observado: os quatro padrões de falha identificados na Seção de Modelos Candidatos (escolha de ferramenta incorreta, resposta em texto em vez de ação, limitação de taxa do provedor, e decisão não reconhecida pela pipeline) mostram que cada família de modelo tende a falhar de um jeito diferente e específico, mesmo estando atrás da mesma interface. Na prática, isso significa que adotar uma arquitetura de *LLM-swap* não elimina a necessidade de monitoramento contínuo por família de modelo: apenas torna a substituição tecnicamente mais simples de executar, não de validar.

F. Limites do Atendimento Homologado como Critério de Qualidade

A curadoria qualitativa da Base 0 (Seção de Dataset) revelou um ponto que vai além deste estudo específico: a aprovação de um atendimento sintético de referência não é, sozinha, garantia de que o agente seguiu corretamente o roteamento de *skills* e a sequência de ferramentas esperada. Das 106 sessões inicialmente candidatas, apenas 20 resistiram à revisão manual turno a turno. Isso é uma lição que extrapola este artigo: qualquer organização que pretenda usar conversas sintéticas selecionadas como referência de qualidade para avaliar ou treinar agentes de IA precisa considerar que a satisfação percebida no cenário sintético e a correção técnica do processo são coisas diferentes, e que a segunda exige revisão humana dedicada, não pode ser inferida apenas da primeira.

Uma limitação semelhante de desenho aparece na leitura de custo: a nota de Custo (C_{norm}) usada no Score Final é normalizada por volume de *tokens*, não por valor monetário real, e esse proxy pode inverter a ordem de custo real entre modelos quando o preço por *token* varia de forma expressiva entre provedores, como visto no caso do **GPT-5.4 mini** (Seção de Resultados); por isso, a leitura baseada em *tokens* deve ser complementada por uma leitura em dólares sempre que a decisão final de adoção de um modelo estiver em jogo.

XIII. Veredito Final e Critérios de Aceitação

A classificação final de cada modelo candidato é determinada por um conjunto de regras de decisão que ponderam a performance técnica, a segurança operacional e a eficiência de custo. O Score médio é expresso em porcentagem conforme a fórmula $(S-1)/3 \times 100$ (ver Seção de Critério de Seleção Final). O teto de latência de 30 segundos por turno (Seção de

Critérios do Projeto) é um critério de aceitação recomendado, integrado à nota L_{norm}; não é eliminatório isoladamente. As condições são avaliadas em ordem de prioridade: primeiro os gates de alucinação (que reprovam o modelo independentemente das demais notas), depois o critério de APROVADO, e só então o de PARCIAL; REPROVADO é o caso residual, quando nenhuma das condições anteriores é atingida. A Tabela IX resume as condições para cada status de aprovação, já aplicada aos 8 modelos candidatos na Seção de Resultados.

TABELA IX
MATRIZ DE DECISÃO PARA APROVAÇÃO DE MODELOS.

Veredito	Requisitos Técnicos
APROVADO	Score médio $\geq$ 75% **e** Pass Rate $\geq$ 70% **e** Hallucination Rate$_{\text{crítico}} = 0\%$ **e** Hallucination Rate$_{\text{geral}} \leq 2\%$
PARCIAL	(Score médio $\geq 63,75\%$ **ou** Pass Rate $\geq 50\%$) **e** Hallucination Rate$_{\text{crítico}} = 0\%$ **e** Hallucination Rate$_{\text{geral}} \leq 2\%$
REPROVADO	Hallucination Rate$_{\text{crítico}} > 0\%$ **ou** Hallucination Rate$_{\text{geral}} > 2\%$; **ou**, quando ambos os *gates* de alucinação são atendidos, Score médio $< 63,75\%$ **e** Pass Rate $< 50\%$ (caso residual, quando nem APROVADO nem PARCIAL se aplicam)

A. Ponto de Decisão: Base 0

É importante ressaltar que o desempenho na **Base 0** é o critério soberano. Benchmarks públicos (como τ-bench e BFCL) servem apenas como triagem inicial; a aprovação final exige que o modelo atinja ou supere a qualidade dos atendimentos sintéticos selecionados e curados como referência de qualidade na plataforma Tech4AI, em termos de equivalência semântica ou melhoria (não identidade literal).

B. Artefatos de Auditoria

Para cada rodada de avaliação, a *pipeline* gera três artefatos que sustentam a governança do projeto:

- **Relatório PDF executivo:** gerado a partir do relatório consolidado, reúne a matriz executiva de decisão, gráficos comparativos, análise de causa-raiz e a visão agregada por sessão para todos os modelos de uma rodada.
- **Relatório consolidado por sessão:** uma derivação leve do *run* completo (veredito operacional por sessão, funil de falhas traduzido em linguagem de negócio e o índice de sessões com evidências) é o artefato que alimenta o Dashboard Web sem exigir nova execução do *replay*.
- **JSON de *run* completo:** preserva os dados brutos de entrada e saída, evidências de correspondência por turno e critério, e o feedback do Judge como evidência qualitativa, permitindo o rastreamento de regressões e a reavaliação de métricas sem a necessidade de novas chamadas de API.

a) Escolha do Modelo Ideal: Dos 8 modelos candidatos avaliados, três atingiram o *status* de **APROVADO**: GPT-5.4 mini, GPT-5.4 nano e Kimi-K2.5 (Seção de Resultados). O **GPT-5.4 mini** apresenta o maior Score Final entre os aprovados (86,4%) e a melhor *Pass Rate* de todo o estudo (92,5%), sem nenhuma ressalva prática identificada, sendo, portanto, a recomendação deste trabalho para adoção em produção. O **GPT-5.4 nano**, também aprovado, combina o menor custo e a menor latência do estudo com a maior base de evidência entre os três aprovados, mas sua Hallucination Rate$_{\text{geral}}$ fica próxima do teto eliminatório de 2%, e seu veredito por execução é o menos consolidado dos três aprovados (Seção de Resultados): um candidato secundário sólido, sobretudo em cenários críticos de custo e latência, mas cujo veredito merece acompanhamento à medida que mais execuções forem acumuladas. O **Kimi-K2.5**, apesar de tecnicamente aprovado (Score Final de 80,0% e *Pass Rate* de 77,0%), deve ser tratado como um candidato secundário condicionado à resolução prévia da limitação de taxa do seu *deployment* na Azure e de sua latência elevada (Seção de Modelos Candidatos), e não como uma alternativa pronta para atendimento síncrono em tempo real. Nem o GPT-4o mini (controle de validação do motor de *replay*) nem o GPT-4o e o GPT-4.1 (usados como modelos juízes) integram esta comparação, pelos motivos detalhados na Seção de Modelos Candidatos.

XIV. LIMITAÇÕES E TRABALHOS FUTUROS

A. Limitações

- **Falta de acesso corporativo a diferentes provedores:** o desenho original da pesquisa previa candidatos de cinco provedores (Groq, Cloudflare, Azure, Mistral AI e GCP). Na prática, o acesso corporativo (contratação, cotas e aprovação de compras junto a cada provedor) esteve disponível, durante o período desta pesquisa, majoritariamente para Azure e GCP, o que concentrou os testes nesses dois provedores e não permitiu, ainda, comparar o custo-benefício de infraestrutura frente aos demais.
- **Veredito do GPT-5.4 nano com margem estreita:** apesar de aprovado no agregado, sua Hallucination Rate$_{\text{geral}}$ está muito próxima do teto eliminatório de 2% (Seção de Discussão), e boa parte das ocorrências de alucinação geral identificadas se concentra em um pequeno número de sessões específicas da Base 0 que se repetem entre execuções. Por execução, é o veredito menos consolidado entre os três aprovados: menos da metade das suas execuções confirma o *status* de APROVADO.
- **Custo como proxy de *tokens*, não valor monetário:** como discutido na Seção de Discussão, a ausência de telemetria de custo contratado em dólares nesta rodada de experimentos obriga a nota de Custo (C_{norm}) usada no Score Final a ser uma aproximação por volume de *tokens*. A Seção de Resultados apresenta uma estimativa complementar em dólares a partir de preços públicos de tabela de cada provedor, mas essa estimativa não reflete eventuais descontos comerciais contratados pela Tech4AI, e não substitui o Custo por Sessão (CPS) real.
- **Execução de ferramentas simulada fora do caminho de referência:** quando um modelo candidato toma um caminho válido, porém diferente do baseline, a resposta da ferramenta que ele recebe é genérica, não real (Seção de Discussão). Isso pode não refletir fielmente o resultado de um caminho alternativo genuinamente correto.

- **Extração incompleta do formato nativo do GPT-OSS-120B:** a alta taxa de sessões sem chamada de ferramenta nem texto reconhecível decorre do formato *Harmony* desse modelo (Seção de Modelos Candidatos). O tratamento implementado durante o período em que o modelo foi testado não cobria a totalidade dos casos observados na Base 0.

B. Trabalhos Futuros

- **Fase 2 da pesquisa (Seção de Introdução):** com a estabilidade operacional da Fase 1 estabelecida por este artigo, o próximo horizonte é investigar melhorias incrementais (*Function Calling* aprimorado e janelas de contexto estendidas) que possam elevar ainda mais o nível de serviço.
- **Ampliar a Base 0:** aumentar o volume de sessões avaliadas.
- **Resolver a limitação de taxa do provedor para o *deployment* do Kimi-K2.5** (aumento de cota junto à Azure ou paralelismo mais conservador para esse modelo, análogo ao já aplicado à família GPT-5) antes de considerá-lo apto para atendimento síncrono em tempo real, dado o percentual relevante de sessões com erro HTTP 429 observado (Seção de Modelos Candidatos).
- **Caso o GPT-OSS-120B volte a ser incluído nos testes, completar a camada de extração dedicada ao seu formato nativo (*Harmony*)**, cobrindo os cenários da Base 0 em que o tratamento usado durante as avaliações originais não reconhecia corretamente a decisão do modelo (chamada de ferramenta ou resposta em texto).
- **Substituir a estimativa por preços públicos de tabela (Seção de Resultados) por telemetria de custo efetivamente contratado em dólares**, assim que os descontos comerciais de cada provedor estiverem consolidados junto à equipe de DevOps.
- **Expandir a cobertura de provedores** (Groq, Cloudflare, Mistral AI) prevista no desenho original da pesquisa, permitindo uma comparação de custo-benefício de infraestrutura mais completa.
- **Aprofundar a investigação da alucinação crítica do GPT-5 mini**, hoje observada em apenas duas sessões, para determinar se representa um padrão estável do modelo ou uma ocorrência isolada.
- **Acumular mais execuções independentes para os modelos com veredito agregado volátil** (Gemini 2.5 Flash e GPT-5 mini), de modo a diferenciar com mais confiança reprovações estruturais de reprovações dependentes da amostra observada.

XV. Conclusão

Este artigo apresentou um pipeline de avaliação automatizada que investiga, sob paridade operacional com o ambiente de referência, a viabilidade de substituir o modelo de linguagem de um agente conversacional Enterprise sem comprometer a qualidade, a latência ou o custo do atendimento. Diferente de benchmarks públicos genéricos, o motor de *replay* construído para esta pesquisa reexecuta conversas sintéticas, já selecionadas e curadas qualitativamente, sob as mesmas regras que o sistema de referência aplicaria: memória e tempo injetados no *system prompt*, mensagens e ferramentas do baseline preservadas, e o mesmo limite de raciocínio por turno.

Dos 8 modelos candidatos avaliados, três, GPT-5.4 mini, GPT-5.4 nano e Kimi-K2.5, atingiram o *status* de **APROVADO** na matriz de decisão multi-critério proposta, considerando o agregado de todas as avaliações realizadas. O achado mais relevante deste trabalho não está na identidade desses três modelos, mas no motivo pelo qual os demais cinco foram excluídos: na maioria dos casos, não foi a qualidade das respostas nem a taxa de conclusão de tarefas que reprovou um modelo, mas sim a taxa de alucinação ultrapassando o limite eliminatório definido. Isso confirma, com dados de cenários sintéticos controlados, um princípio metodológico que este artigo defende: em agentes que tomam decisões sobre usuários e fluxos críticos, critérios de segurança precisam operar como filtros independentes, não como um componente a mais diluído em uma média ponderada.

Como os modelos de linguagem são sistemas probabilísticos, este trabalho também recalculou o veredito execução por execução, não apenas sobre o agregado, e essa checagem revelou uma distinção importante entre os modelos reprovados: dois deles, Gemini 2.5 Flash e GPT-5 mini, oscilam entre aprovação e reprovação de forma quase equilibrada entre execuções independentes, o que sugere que sua reprovação agregada é dependente da amostra observada, e não uma característica estrutural do modelo. O contraponto é o GPT-OSS-120B, que nunca atingiu aprovação em nenhuma de suas execuções: uma reprovação consistente e estatisticamente mais robusta, coerente com a causa estrutural identificada (formato de resposta nativo *Harmony*, incompatível com o padrão de *chat completions* do restante da pipeline). A lição geral é que um único veredito agregado não deveria, sozinho, embasar o descarte definitivo de um modelo: é a consistência da reprovação ao longo de múltiplas execuções que distingue um problema estrutural de uma variação estatística esperada.

A recomendação final deste trabalho é a adoção do **GPT-5.4 mini**, sem nenhuma ressalva operacional identificada. O **GPT-5.4 nano** é a alternativa de menor custo e latência, com margem de segurança mais estreita e veredito por execução menos consolidado; o **Kimi-K2.5**, apesar de tecnicamente aprovado, permanece candidato secundário até a resolução de sua limitação de taxa do provedor e de sua latência elevada (Seção de Veredito Final).

Este estudo também deixa contribuições que extrapolam a escolha de um modelo específico: evidenciou que a seleção de um atendimento sintético de referência não substitui a revisão técnica humana como critério de curadoria de dados de referência; que uma arquitetura de *LLM-swap* tecnicamente simples não elimina a necessidade de monitoramento específico por família de modelo; e que decisões de substituição de modelo em ambiente Enterprise se beneficiam mais de uma avaliação sob paridade com o ambiente de referência do que de benchmarks públicos de propósito geral. Com a estabilidade operacional da Fase 1 estabelecida por este trabalho, a Fase 2 da pesquisa (voltada a capacidades incrementais como

Function Calling aprimorado e janelas de contexto estendidas) e as demais direções indicadas na Seção de Limitações seguem como continuidade natural deste projeto.

REFERÊNCIAS

[1] Microsoft Azure, "Azure OpenAI Service: Pricing," 2026. [Online]. Disponível em: https://azure.microsoft.com/en-us/pricing/details/azure-openai/. Acesso em: 13 jul. 2026.

[2] Microsoft Azure, "Foundry Models Pricing: Kimi," 2026. [Online]. Disponível em: https://azure.microsoft.com/en-us/pricing/details/ai-foundry-models/kimi/. Acesso em: 13 jul. 2026.

[3] M. Grace, J. Hadfield, R. Olivares, e J. De Jonghe, "Demystifying Evals for AI Agents," *Anthropic Engineering Blog*, jan. 2026. [Online]. Disponível em: https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents

[4] M. Finio e A. Downie, "What is AI Agent Orchestration?," *IBM Think*, 2025. [Online]. Disponível em: https://www.ibm.com/think/topics/ai-agent-orchestration

[5] S. G. Patil *et al.*, "The Berkeley Function Calling Leaderboard (BFCL): From Tool Use to Agentic Evaluation of Large Language Models," em *Forty-second International Conference on Machine Learning*, 2025.

[6] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, e I. Stoica, "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena," em *Advances in Neural Information Processing Systems (NeurIPS: Datasets and Benchmarks Track)*, 2023. [Online]. Disponível em: https://arxiv.org/abs/2306.05685

[7] S. Yao, N. Shinn, P. Razavi, e K. Narasimhan, "τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains," *arXiv preprint arXiv:2406.12045*, 2024. [Online]. Disponível em: https://arxiv.org/abs/2406.12045

[8] Google, "Gemini Developer API Pricing," 2026. [Online]. Disponível em: https://ai.google.dev/gemini-api/docs/pricing?hl=pt-br. Acesso em: 13 jul. 2026.