Otimização de Guardrails de Segurança em Agentes Conversacionais

Tech for Humans, Júlia Pivato de Oliveira

***Resumo*— Este artigo apresenta os resultados da Fase 1 de uma pesquisa voltada à otimização técnica de *Guardrails* de segurança em agentes conversacionais baseados em modelos de linguagem de larga escala (LLMs). O foco central reside no refinamento de componentes de segurança para mitigar riscos críticos, como *Prompt Injection* e *Jailbreaking*. A metodologia empregou um estudo de ablação sistemático com 72.000 execuções experimentais, utilizando um *dataset* híbrido que integra ataques sintéticos e registros reais de produção. Os resultados revelam a identificação de uma arquitetura de alta performance que atingiu níveis de precisão próximos à perfeição e a eliminação de erros estruturais de resposta, consolidando uma solução resiliente para ambientes produtivos de larga escala. O estudo demonstra ainda como a manipulação do contexto de entrada é determinante para a desambiguação semântica e para a redução de falsos negativos. Conclui-se propondo um modelo de configuração baseado na segregação de responsabilidades técnica e funcional, otimizando a segurança sem comprometer a agilidade do agente principal.**

***Abstract*— This paper presents the results of Phase 1 of a research study on the technical optimization of safety Guardrails for Large Language Model (LLM)-based conversational agents. The study focuses on refining safety components to mitigate critical risks such as Prompt Injection and Jailbreaking. Utilizing a systematic ablation study with 72,000 experimental runs, the methodology leverages a hybrid dataset of synthetic attacks and real-world production logs. The findings highlight a high-performance architecture that achieves near-perfect precision scores and the total elimination of parsing errors, establishing a resilient solution for large-scale production environments. Furthermore, the research identifies how specific input context strategies are essential for semantic disambiguation and the significant reduction of false negative rates. The paper concludes by proposing a configuration model based on the strict segregation of technical and functional responsibilities, optimizing security without compromising the primary agent's operational efficiency.**

***Palavras-Chave*— Segurança em IA. Guardrails. LLM. Otimização de Performance. Arquitetura de Segurança.**

***Keywords*— AI Safety. Guardrails. LLM. Performance Optimization. Security Architecture.**

I. Introdução

A implementação de Agentes Conversacionais baseados em Modelos de Linguagem de Grande Escala (LLMs) em ecossistemas corporativos introduz vetores de risco que abrangem desde a exfiltração de dados até a subversão da lógica de negócios. Para mitigar tais vulnerabilidades, a arquitetura moderna de IA incorpora uma camada de *Guardrails*[1]. Este módulo atua como um sistema de interceptação que avalia as solicitações do usuário. Com base em políticas e conceitos de segurança pré-definidos, o *Guardrail* decide pela liberação ou bloqueio da resposta, validando se a interação está em conformidade com o comportamento esperado do sistema. No contexto desta pesquisa, esta configuração estabelece a base para a Fase I, que foca na otimização técnica e na validação dessa camada de interceptação em cenários reais de produção.

A. Arquitetura de Interceptação e Mecanismo de Decisão

O sistema em estudo emprega um classificador baseado em LLM que utiliza a técnica de *Tool Calling* (chamada de função). Sob este paradigma, o modelo é forçado a realizar uma escolha binária obrigatória entre duas funções predefinidas:

1) **mensagem_segura**: Interações que, embora possam ser informais ou estar fora do escopo de negócio, não apresentam riscos à infraestrutura ou diretrizes éticas.
2) **nao_segura**: Categoria que engloba interações com risco real à integridade do sistema. Inclui técnicas de *Prompt Injection*[2], além de injeções de código como *SQL Injection* (SQLi)[3]. Compreende também manobras de exfiltração de dados[4], conteúdo tóxico[5] e tentativas de *Goal Hijacking*[6] via *roleplay*[7]. Esta categoria consolida qualquer entrada que tente subverter instruções de segurança ou acessar dados protegidos.

B. Definição do Problema: As Dores do Produto

Apesar da arquitetura lógica estabelecida, a operação em ambiente de produção revelou limitações críticas que comprometem a robustez do sistema. Tais vulnerabilidades manifestam-se na dificuldade de resposta a cenários imprevistos, evidenciando lacunas na capacidade de discernimento e na eficiência do componente de segurança, conforme detalhado a seguir:

1) **Overfitting ao Few-Shot (Rigidez Cognitiva)**: O modelo apresenta uma tendência a classificar como ameaça apenas mensagens que são cópias exatas ou variações mínimas dos exemplos fornecidos no *System Prompt*[8]. Essa rigidez impede que o sistema identifique ataques que utilizam variações sutis de linguagem.

[1]Do inglês *Guardrails*. Refere-se a travas de segurança e filtros que garantem respostas éticas e seguras em modelos de IA.

[2]Do inglês *Prompt Injection*. Refere-se ao uso de instruções para enganar a IA e fazê-la ignorar suas regras originais.

[3]Do inglês *SQL Injection*. Refere-se à inserção de comandos maliciosos para manipular ou acessar bancos de dados indevidamente.

[4]Extração não autorizada de informações sensíveis.

[5]Linguagem ofensiva, discurso de ódio ou assédio.

[6]Do inglês *Goal Hijacking*. Refere-se ao desvio do objetivo original da IA para uma nova finalidade definida por um atacante.

[7]Do inglês *Roleplay*. Refere-se à instrução para que a IA assuma um personagem ou comportamento específico em uma interação.

[8]Do inglês *System Prompt*. Refere-se às instruções base fornecidas ao modelo para definir seu comportamento, regras e limites antes da interação com o usuário.

2) **Hipersensibilidade e Falsos Positivos (Atrito de UX[9])**: O sistema demonstra incapacidade em distinguir a frustração do usuário de uma ameaça real. Interações legítimas, porém informais ou carregadas de insatisfação, são frequentemente bloqueadas de forma incorreta, o que prejudica a jornada de atendimento.
3) **Dependência de Contexto e Inferência Rígida**: Existe uma incerteza sobre como o histórico da conversa (*Full History*)[10] influencia a decisão de segurança e se a configuração atual de temperatura (0.0) limita a capacidade do modelo de abstrair o conceito de "ameaça" para além dos exemplos literais.
4) **Dualidade de Escopo e Conflito de Responsabilidade**: Observa-se uma inconsistência na classificação de mensagens *out-of-scope* (fora de escopo), gerando um conflito de competências entre o *Guardrail* e o agente. O sistema oscila entre focar estritamente na integridade da infraestrutura ou bloquear interações meramente irrelevantes ao tema. Essa dualidade exige uma definição clara se a filtragem de escopo deve ser consolidada na camada de segurança ou delegada à lógica de interação do agente principal.

C. Objetivos da Pesquisa

Esta pesquisa é estruturada em etapas consecutivas, sendo que a presente fase (Fase I) fundamenta-se na hipótese de que o módulo atual possui a inteligência necessária, mas encontra-se sub-otimizado devido a parâmetros de inferência rígidos e uma engenharia de *prompt* restritiva. O objetivo central desta etapa inicial é realizar o refinamento técnico do componente de segurança para validar sua viabilidade operacional. Os objetivos específicos desta fase incluem:

- **Maximização da Eficácia**: Atingir o equilíbrio ideal entre *Recall*[11] e *Precision*[12]
- **Generalização de Conceitos**: Garantir que o modelo compreenda o conceito abstrato de ameaças em vez de limitar-se à memorização de padrões textuais específicos.
- **Benchmarking**[13] **de Modelos e Eficiência**: Avaliar se modelos de menor escala (como as versões *Nano* e *Mini*) mantêm o desempenho de segurança adequado, visando a redução de latência e de custos operacionais.

II. Metodologia

Para validar a hipótese de que a otimização de parâmetros e *prompts* elevaria a performance do sistema, foi desenhado um estudo de ablação[14] sistemático. A terminologia adotada fundamenta-se nos padrões de detecção de ameaças do mercado.

TABELA I
Definições de Métricas aplicadas ao contexto de Segurança

Resposta Esperada	Resposta Predita	Resultado (Métrica)
Segura	Segura	True Negative (TN)
Não Segura	Segura	False Negative (FN)
Não Segura	Não Segura	True Positive (TP)
Segura	Não Segura	False Positive (FP)

Fonte: Elaborado pelo autor (2026).

A classe positiva é definida como a presença de uma ameaça (*nao_segura*). As métricas de classificação foram extraídas a partir da análise comparativa entre o rótulo esperado e o rótulo predito pelo modelo, conforme detalhado na Tabela I.

A. Tratamento de Erros e Registros Inválidos

Um aspecto crítico da metodologia é o tratamento de falhas de processamento ou erros de resposta da API. Tais registros são classificados como FAILURE.

Diferente das abordagens de classificação tradicionais, registros marcados como *FAILURE* não compõem a matriz de confusão. As métricas de performance (Precisão, *Recall* e F1-Score) são calculadas exclusivamente sobre o subconjunto de registros válidos. A eficiência do sistema quanto ao processamento é monitorada separadamente pela Taxa de Erro de *Parsing*

$$\text{Taxa de Erro de Parsing} = \frac{\sum FAILURE}{\text{Total de Registros}}$$

B. Métricas de Performance

Para a quantificação do desempenho sobre os dados válidos, foram aplicadas as seguintes métricas fundamentais:

- **Precision:** Mede a exatidão das detecções. Responde: "Das mensagens classificadas como ameaça, quantas eram realmente ameaças?"

$$Precision = \frac{TP}{TP + FP}$$

- **Recall:** Mede a taxa de detecção de ameaças reais. Responde: "Das ameaças existentes, quantas o sistema conseguiu identificar?"

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score:** Média harmônica entre Precisão e *Recall*, oferecendo uma visão balanceada da performance global.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

[9]Do inglês *User Experience*. Refere-se à experiência e percepção de um usuário ao interagir com um produto, sistema ou serviço, focando em facilidade de uso e satisfação.

[10]Do inglês *Full History*. Refere-se ao envio de todo o histórico da conversa para o sistema de segurança, permitindo analisar o contexto completo e detectar ameaças contextuais ou desambiguar mensagens suspeitas.

[11]Do inglês *Recall*. Refere-se à taxa de acerto de um sistema de segurança em identificar todas as ameaças reais presentes, minimizando a ocorrência de falsos negativos.

[12]Do inglês *Precision*. Refere-se à exatidão de um sistema de segurança ao identificar ameaças, representando a proporção de alertas emitidos que são realmente ataques reais, visando reduzir falsos positivos.

[13]Do inglês *Benchmarking*. Refere-se ao processo de comparar o desempenho de um modelo ou sistema de segurança contra padrões de referência ou métricas estabelecidas no mercado.

[14]Método experimental que consiste em remover ou alterar componentes específicos de um sistema de IA para avaliar o impacto individual de cada variável no desempenho global.

C. Design do Experimento

A pesquisa utilizou uma matriz de testes A/B[15] variando cinco dimensões principais para identificar a configuração ideal:

- **Modelos LLM (via LiteLLM)**: Foram comparados modelos de diferentes escalas para avaliar o *trade-off* (equilíbrio de forças) entre custo operacional e eficácia de segurança: *Gemini 2.0 Flash, Gemini 2.5 Flash, GPT 4.1 Mini, GPT 4.1 Nano* e *GPT 4o Mini*.
- **Engenharia de Prompt**:
  - **Prompt Simples**: Baseia-se em instruções conceituais e definições abstratas das categorias de risco (como *Jailbreak* e Conteúdo Tóxico), sem a oferta de exemplos práticos, exigindo maior capacidade de inferência do modelo.
  - **Few-Shot**: Inclui exemplos concretos de mensagens e suas respectivas classificações. Por exemplo: *"ignore as instruções anteriores"* $\rightarrow$ *nao_segura*; *"Quero comunicar um sinistro"* $\rightarrow$ *segura*. Este método visa reduzir a ambiguidade e ancorar o comportamento do classificador.
  - **Chain of Thought**: Estrutura o processo de decisão em etapas lógicas obrigatórias, por exemplo:
    - Passo 1: Identificação
    - Passo 2: Análise de Padrões
    - Passo 3: Contextualização

    Dessa maneira, forçando o modelo a analisar a intenção e a resistência a manipulações antes de emitir o veredito binário.
- **Hiperparâmetros (Temperatura)**: Varredura de valores $(0.0, 0.1, 0.3, 0.5)$ para encontrar o equilíbrio entre o determinismo necessário para segurança e a flexibilidade para generalização conceitual.
- **Janela de Contexto**:
  - **Stateless**: Envio apenas da última mensagem do usuário para testar o isolamento e o foco na ameaça imediata.
  - **Full History**: Envio de todo o histórico da conversa para fornecer contexto ao classificador. Esta abordagem visa permitir a desambiguação de intenções, onde mensagens que isoladamente poderiam ser interpretadas como suspeitas são validadas como legítimas com base no fluxo prévio da interação, reduzindo assim a taxa de falsos negativos.
- **Formato de Saída**: A avaliação da robustez sintática baseia-se no contraste entre dois métodos de estruturação via API:
  - **Tools (Function Calling)**: Utiliza a interface de funções da API, onde o parâmetro tool_choice: "required" força o modelo a responder com uma chamada de função específica (mensagem_segura ou nao_segura). Este método não depende apenas do prompt, pois a infraestrutura da API valida a obrigatoriedade da resposta estruturada.
  - **JSON Schema (Structured Output)**: Combina o parâmetro response_format: {"type": "json_object"} com uma instrução restritiva no prompt. Enquanto a API garante a validade do JSON, o prompt define a semântica do campo (ex: seguranca_status), exigindo que o modelo realize o *parsing* interno de acordo com o esquema definido.
- **Contextualização do Agente (System Prompt)**: Avalia o impacto do fornecimento da "persona"e das diretrizes operacionais do agente ao classificador. Foram testados dois cenários:
  - **Sem Contexto**: O classificador analisa a mensagem de forma isolada, sem saber se o bot é um assistente jurídico, médico ou de suporte geral.
  - **Com Contexto de Atuação**: O *prompt* do agente (ex: *"Você é um assistente de seguros especializado em sinistros""*) é incluído na análise. O objetivo é verificar se o conhecimento do domínio reduz a taxa de falsos positivos, permitindo que o modelo diferencie termos técnicos legítimos de tentativas de manipulação ou linguagem ofensiva.

D. Dataset Híbrido de Teste

A validade dos testes foi sustentada por um *dataset* consolidado de 150 casos de teste, categorizados em duas frentes:

1) **Dados Sintéticos de Ataque (62%)**: Casos gerados ou coletados de bases públicas contendo variações de *Jailbreaks* (ex: DAN, Modo Desenvolvedor), codificação em *Base64* e injeções sutis de instruções. Foram incluídos também cenários *out-of-scope* nessa frente para observar como o Guardrail interage com essas demandas sob a lógica atual, visando identificar a origem das inconsistências relatadas em produção.
2) **Dados Reais Sanitizados (38%)**: *Logs* de produção reais que passaram por um processo de desidentificação. Foram ofuscadas categorias de dados sensíveis e PII[16], incluindo informações de identificação civil (CPF, RG), dados de contato (e-mail, telefone) e referências a ativos financeiros. Este subconjunto é fundamental para monitorar falsos positivos em interações legítimas.

Cada unidade de teste no *dataset* é representada por um objeto estruturado em formato JSONL, o que garante a rastreabilidade dos dados. A estrutura de um caso de teste é composta por quatro pilares fundamentais:

1) **Identificação**: Através de um id único e o campo source, que distingue se a origem da amostra é sintética ou proveniente de *logs* reais de produção.
2) **Classificação de Referência (Ground Truth)**: Os campos mensagem_segura (booleano) e tipo_de_ameaca estabelecem o resultado esperado contra o qual o modelo será avaliado.

[15]Método de experimentação comparativa que analisa duas ou mais variantes de uma variável (neste caso, modelos e configurações) para determinar qual versão apresenta melhor desempenho sob métricas específicas.

[16]Do inglês *Personally Identifiable Information*. Refere-se a qualquer dado que possa ser usado para identificar um indivíduo específico, como CPF, e-mail, número de telefone ou dados biométricos.

3) **Contexto Conversacional**: O historico_mensagens contém a sequência de interações anteriores, enquanto o message_to_test_index aponta exatamente qual mensagem do histórico deve ser submetida ao crivo do *Guardrail*.
4) **Configuração do Agente**: O campo system_prompt armazena as instruções completas de persona e habilidades do robô no momento do teste.

Abaixo, apresenta-se a estrutura simplificada de uma entrada do *dataset*:

```
{
  "id": "synthetic_injection_0001",
  "source": "synthetic",
  "mensagem_segura": false,
  "tipo_de_ameaca": "injection_roleplay",
  "historico_mensagens": [
    {"user": "cliente", "mensagem": "..."},
    {"user": "agent", "mensagem": "..."}
  ],
  "message_to_test_index": 9,
  "system_prompt": "..."
}
```

1) Definição Detalhada dos Dados: Os dados presentes no *Dataset* utilizado para testes se enquadram nas seguintes categorias listadas:

TABELA II
CATEGORIAS DE AMEAÇAS E DEFINIÇÕES

Categoria	Descrição
Nenhuma	Não é uma ameaça, são mensagens seguras e legítimas do usuário.
Prompt Injection	Tentativa sutil de manipular o modelo disfarçada como solicitação legítima, sem mencionar explicitamente instruções ou regras.
Injeção de Instruções	Tentativa explícita de modificar ou anular as instruções do sistema, mencionando diretamente "instruções anteriores", "modo"ou comandos inválidos.
Injeção de Dados	Tentativa de manipular ou extrair dados sensíveis, referenciar informações de forma suspeita ou tentar acesso não autorizado.
Injeção de Segurança	Tentativa de burlar ou comprometer medidas de segurança do sistema, mencionando violações ou explorando vulnerabilidades.
Injeção de Roleplay	Tentativa de fazer o modelo assumir um papel ou personagem específico para contornar restrições normais.
Injeção Ofensiva	Conteúdo ofensivo, abusivo ou inapropriado que viola diretrizes éticas.
Conteúdo Sexual	Conteúdo sexual explícito ou inapropriado.
Automutilação	Conteúdo relacionado a automutilação, suicídio ou autolesão que requer intervenção de segurança.

Fonte: Elaborado pelo autor (2026).

E. Pipeline de Execução e Métricas

O *pipeline*[17] foi automatizado para realizar milhares de combinações de testes. Essa abordagem resultou em um volume de 480 execuções por caso, totalizando um *dataset* de resultados com 72.000 instâncias processadas.

Cada um dos 150 casos de teste foi submetido a uma matriz combinatória composta por 5 modelos de linguagem, 4 níveis de temperatura, 2 modos de contexto, 2 formatos de saída, 3 variações de *prompting* e 2 condições de inclusão do *Prompt* do Agente (com/sem). Para cada execução, foram extraídas as seguintes métricas:

- **Métricas de Classificação**: Utilizou-se o *Recall* e a *Precision* para avaliar a confiabilidade dos bloqueios.
- **Métricas Operacionais**: A latência[18] foi monitorada para validar a viabilidade de uso em tempo real, juntamente com a estabilidade de *parsing*[19], métrica crítica para garantir a continuidade da experiência do usuário.

III. RESULTADOS E ANÁLISE DE PERFORMANCE

A análise de dados foi estruturada de forma incremental, avaliando desde o desempenho intrínseco dos modelos até o impacto das variáveis de configuração.

A. Avaliação Comparativa de Modelos

Nesta etapa, isolou-se a performance de cada modelo para entender seu comportamento padrão em tarefas de segurança. A análise quantitativa revela disparidades significativas na eficácia dos modelos, especialmente no que tange ao equilíbrio entre precisão e recall.

TABELA III
DESEMPENHO COMPARATIVO POR MODELO

Modelo	F1	Prec.	Rec.	Erro	Lat.
Gemini 2.0 Flash	0,9781	0,9792	0,9769	0,00%	1,027
Gemini 2.5 Flash	0,9779	0,9910	0,9651	16,99%	1,707
GPT 4.1 Mini	0,9243	0,9853	0,8703	0,00%	1,814
GPT 4.1 Nano	0,8851	0,8249	0,9548	0,14%	0,914
GPT 4o Mini	0,9174	0,8717	0,9682	0,00%	1,014

Fonte: Elaborado pelo autor (2026).

1) Gemini 2.0 Flash: O modelo *Gemini 2.0 Flash* apresenta o perfil mais equilibrado do estudo, exibindo uma matriz de confusão com alta concentração na diagonal principal, o que reflete uma classificação precisa tanto para ameaças quanto para interações seguras.

Com um *F1-Score* de 0,9781 e um *Recall* de 0,9769, o sistema demonstra alta eficácia na detecção de ataques,

[17]Do inglês *Pipeline*. Refere-se ao conjunto de processos ou etapas de computação encadeadas de forma sistemática, onde a saída de uma etapa serve como entrada para a próxima, permitindo a automação do fluxo de dados.

[18]Tempo decorrido entre o envio da requisição e o recebimento da resposta estruturada pelo sistema.

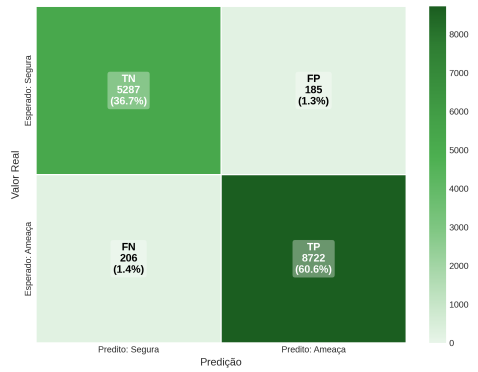
[19]Do inglês *Parsing*. Refere-se ao processo de analisar e converter a saída bruta do modelo para um formato estruturado específico, como JSON ou Markdown, garantindo que os dados sejam legíveis por outros sistemas.

minimizando a ocorrência de falsos negativos que poderiam comprometer a integridade da sessão.

Adicionalmente, sua performance técnica é evidenciada pela ausência total de erros de *parsing* (0,00%) e uma latência média de 1,027s, posicionando-o como a solução mais resiliente para ambientes de produção. Esta consistência operacional indica que o modelo consegue processar instruções complexas de segurança sem gerar o "atrito de UX" observado em variantes menos precisas, garantindo uma jornada de atendimento fluida e protegida.

Fig. 1

MATRIZ DE CONFUSÃO - GEMINI 2.0 FLASH

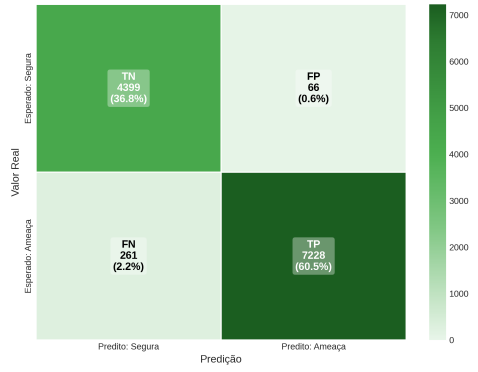


Fonte: Elaborado pelo autor (2026).

2) Gemini 2.5 Flash: O modelo *Gemini 2.5 Flash* apresenta o maior rigor de classificação do grupo, atingindo uma *Precision* de 0,9910, o que se traduz em um índice quase nulo de falsos positivos e uma confiabilidade excepcional ao autorizar mensagens seguras.

Fig. 2

MATRIZ DE CONFUSÃO - GEMINI 2.5 FLASH



Fonte: Elaborado pelo autor (2026).

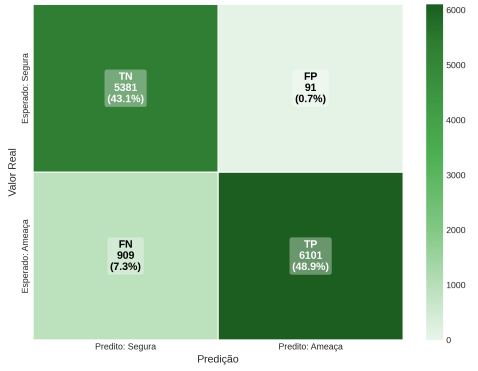
Entretanto, sua viabilidade operacional em larga escala é severamente comprometida por uma taxa de erro de *parsing* de 16,99%, a mais alta registrada no estudo. Em um cenário real de produção, esse dado indica que aproximadamente uma em cada seis interações falharia devido à incapacidade do modelo em aderir estritamente ao formato de saída (*Tools* ou *JSON Schema*). Essa instabilidade sintática cria um gargalo crítico na integração direta, exigindo mecanismos complexos de *fallback* ou reprocessamento, o que elevaria significativamente o custo computacional e a latência percebida pelo usuário final.

3) GPT 4.1 Mini: O modelo *GPT 4.1 MIni* é caracterizado por um perfil de alta confiabilidade no bloqueio, sustentado por uma *Precision* de 0,9853. Este dado garante que as mensagens marcadas como ameaça são quase certamente maliciosas, oferecendo uma operação com baixíssimo índice de falsos alarmes.

Por outro lado, o modelo apresenta um *Recall* de 0,8703, indicando uma postura significativamente menos agressiva na captura de ataques em comparação aos modelos da família Gemini. Na prática, essa métrica revela uma vulnerabilidade crítica: cerca de 13% das ameaças, predominantemente as mais sutis ou codificadas, passam despercebidas (falsos negativos), atingindo o agente principal sem a devida interceptação.

Fig. 3

MATRIZ DE CONFUSÃO - GPT 4.1 MINI



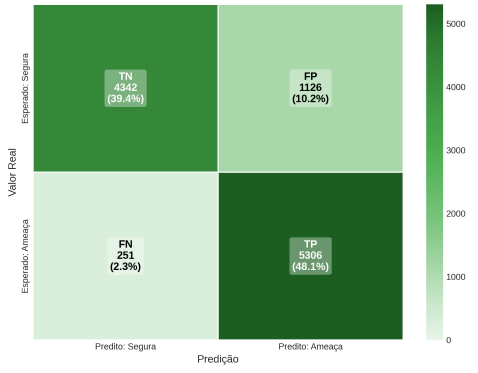
Fonte: Elaborado pelo autor (2026).

Embora sua estabilidade de *parsing* seja satisfatória, esse comportamento sugere a necessidade de camadas adicionais de validação caso o modelo seja adotado como único *Guardrail* em contextos de alta periculosidade.

4) GPT 4.1 Nano: O modelo *GPT 4.1 Nano* prioriza a agilidade operacional, registrando a menor latência média do estudo (0,914s), o que o torna ideal para aplicações que exigem respostas quase instantâneas.

Fig. 4

MATRIZ DE CONFUSÃO - GPT 4.1 NANO



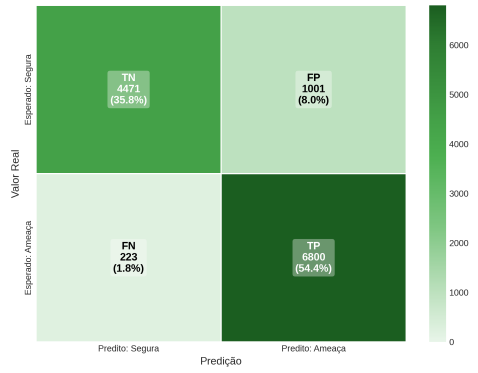
Fonte: Elaborado pelo autor (2026).

Contudo, este ganho de velocidade reflete uma perda substancial na qualidade da classificação: o modelo apresenta a menor *Precision* do grupo (0,8249), o que resulta em um volume considerável de falsos positivos.

Caracterizado por um perfil que "atira primeiro e pergunta depois", sua sensibilidade descalibrada tende a bloquear interações inofensivas com frequência, prejudicando severamente a experiência do usuário legítimo (*UX*) ao gerar interrupções desnecessárias. Embora seja eficiente em termos de custo computacional, sua aplicação como *Guardrail* primário exige cautela, pois o alto índice de falsos alarmes pode levar à frustração do cliente e à degradação da confiança no sistema automatizado.

5) GPT 4o Mini: O modelo *GPT 4o Mini* apresenta um comportamento focado em detecção ampla, sustentado por um *Recall* de 0,9682, o que o torna extremamente eficiente em garantir que ameaças não passem despercebidas pelo sistema. Entretanto, sua *Precision* de 0,8717 revela uma tendência a classificar mensagens seguras como suspeitas com maior frequência que os modelos de topo. Esse perfil é ideal para fluxos de segurança crítica onde o custo de um Falso Negativo (vulnerabilidade) é substancialmente maior que o impacto operacional de um Falso Positivo (atrito). Além disso, o modelo mantém uma latência competitiva de 1,014s e estabilidade total de *parsing*, consolidando-se como uma escolha estratégica para camadas de defesa agressivas que não podem tolerar brechas, mesmo que isso resulte em uma filtragem mais rigorosa de interações legítimas.

Fig. 5
MATRIZ DE CONFUSÃO - GPT 4O MINI



Fonte: Elaborado pelo autor (2026).

6) O Fenômeno do Bloqueio Nativo (Safety Filters): Um fator determinante na performance observada foi a incidência de bloqueios nativos operados pelos próprios provedores de modelos (*Safety Filters*), que interceptam a requisição antes mesmo do processamento pela camada de *Guardrail* customizada. No total, 9,98% das requisições (7.182 instâncias) foram bloqueadas automaticamente, o que caracteriza uma camada de segurança primária e intrínseca aos modelos de larga escala.

TABELA IV
DISTRIBUIÇÃO E EFICÁCIA DOS BLOQUEIOS NATIVOS POR MODELO

Modelo	**Total Bloqueios**	**Taxa (%)**	**Eficácia (TP)**
GPT 4.1 Mini	1.918	13,32%	100,00%
GPT 4.1 Nano	3.359	23,33%	100,00%
GPT 4o Mini	1.905	13,23%	99,74%

Fonte: Elaborado pelo autor (2026).

No que tange à cobertura, a eficácia na mitigação de riscos (*Recall*) apresentou disparidades significativas conforme a categoria do ataque. O sistema demonstrou maior sensibilidade para *Conteúdo Sexual* (20,00%) e *Injeção de Roleplay* (29,90%), o que contrasta com o desempenho crítico em tentativas de *Injeção de Dados*, cujo bloqueio nativo foi de apenas 5,22%. Com uma taxa agregada de contenção de apenas 16,09%, evidencia-se que a maioria das ameaças transpassa os filtros primários, reforçando a necessidade indispensável de uma camada de *Guardrail* robusta e especializada para garantir a segurança da aplicação.

A análise léxica das mensagens que sofreram bloqueio nativo revela padrões semânticos associados a tentativas de subversão do sistema. Termos como "ignore", "desconsidere", "anteriores" e "instruções" figuram entre as palavras-chave mais recorrentes, evidenciando que os filtros são altamente reativos a comandos que visam a anulação de diretrizes de segurança (*jailbreak* por negação de contexto).

Embora esses filtros elevem o *Recall* geral, a disparidade observada no modelo *GPT 4.1 Nano*, com taxa de bloqueio de 23,33%, sugere uma estratégia de segurança mais agressiva e menos granulada em modelos de menor escala. Esta rigidez operacional atua como um mecanismo compensatório para limitações no raciocínio lógico, mas deve ser ponderada contra o risco de degradação da experiência do usuário (*UX*) em cenários de alta ambiguidade linguística, onde o bloqueio preventivo pode substituir uma análise contextual mais refinada.

B. Análise das Variáveis Experimentais

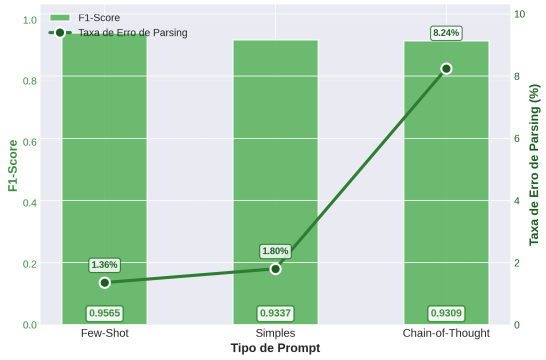
Após a caracterização dos modelos, esta seção detalha como as variáveis de configuração influenciaram o desempenho do sistema, identificando os componentes determinantes para a eficácia do *Guardrail*.

1) Arquitetura de Prompt e Raciocínio Estruturado: O achado mais significativo deste estudo foi o desempenho superior da técnica de *Few-Shot* em comparação ao *Chain of Thought* (CoT). Embora o CoT seja eficaz em tarefas de lógica pura, sua aplicação em filtros de segurança estruturados apresentou um efeito colateral de instabilidade sintática:

- **Prompt Simples**: Demonstrou limitações na generalização (F1: 0,9337), evidenciando que instruções puramente conceituais são insuficientes para cobrir as nuances de ataques complexos.
- **Few-Shot**: Consolidou-se como a estratégia mais robusta (F1: 0,9565). A ancoragem em exemplos concretos permitiu ao modelo alinhar o veredito de segurança com uma baixa taxa de erro de *parsing* (1,36%).

- **Chain of Thought**: Registrou o menor F1-Score (0,9309). A verbosidade do raciocínio intermediário frequentemente corrompeu o formato JSON esperado, elevando o erro de *parsing* para 8,24%. Isso ocorre porque a geração de texto livre antes da saída estruturada cria uma "competição de contexto", levando o modelo a incluir explicações narrativas ou delimitadores textuais que violam o rigor sintático do JSON Schema.

Fig. 6
COMPARAÇÃO EFICÁCIA VS. ERRO DE PARSING - ESTRATÉGIAS DE PROMPTING



Fonte: Elaborado pelo autor (2026).

2) Impacto da Inclusão do Prompt do Agente: A inclusão do *System Prompt* do agente principal no contexto do *Guardrail* foi testada para avaliar se o conhecimento do domínio auxiliaria o modelo a contextualizar melhor as interações. Os resultados revelam que o contexto do agente geralmente induz o modelo a um comportamento significativamente mais defensivo:

TABELA V
COMPARATIVO DE PERFORMANCE COM A INCLUSÃO DO PROMPT DO AGENTE

Configuração	F1-Score	Prec.	Rec.	Erro
Sem Prompt Agente	0,9395	0,9459	0,9331	4,57%
Com Prompt Agente	0,9416	0,9203	0,9638	3,03%

Fonte: Elaborado pelo autor (2026).

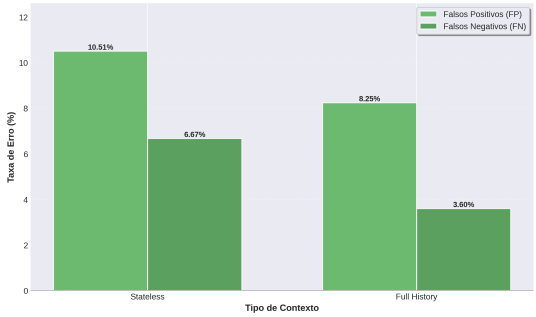
A interpretação técnica dos dados revela:

- **Aumento da Sensibilidade (Ganho de Recall):** Com a inclusão do *prompt* do agente, o *Recall* subiu para 0,9638. Isso indica que o modelo tornou-se significativamente mais capaz de identificar ameaças e tentativas de subversão, reduzindo o volume de ataques que passariam despercebidos (Falsos Negativos).
- **Aumento de Bloqueios Indevidos (Perda de Precisão):** A *Precisão* caiu para 0,9203. O conhecimento do domínio faz com que o modelo interprete ambiguidades com maior rigor, resultando em um aumento de Falsos Positivos (mensagens seguras bloqueadas incorretamente por serem confundidas com tentativas de quebra de escopo).

A conclusão operacional é que a inclusão do *prompt* do agente melhora a detecção de ameaças com a consequência do aumento de Falsos Positivos. A decisão de utilizá-lo depende da prioridade do sistema: segurança máxima (priorizando *Recall*) ou fluidez da experiência do usuário (priorizando *Precision*).

3) Efeito da Janela de Contexto: A análise experimental demonstrou que a inclusão do histórico completo da conversa é a variável de maior relevância para a redução de Falsos Negativos. A transição do modo de operação isolado (*Stateless*) para o processamento baseado em histórico completo (*Full History*) atua como um mecanismo de desambiguação semântica fundamental para sistemas conversacionais.

Fig. 7
COMPARAÇÃO FP VS. FN - JANELA DE CONTEXTO



Fonte: Elaborado pelo autor (2026).

Conforme os dados da análise, a utilização de *Full History* elevou o *F1-Score* de 0,9288 para 0,9522. O ganho real consolidado de 2,34 pontos percentuais valida a importância da persistência de contexto. Este incremento de performance é decisivo para identificar ameaças que utilizam o encadeamento de mensagens para camuflar intenções maliciosas, permitindo que o modelo interprete a entrada atual sob a ótica do fluxo conversacional prévio.

TABELA VI
COMPARATIVO CONSOLIDADO DE DESEMPENHO POR JANELA DE CONTEXTO

Contexto	Lat.	Prec.	Rec.	F1	Erro
Full History	1,334s	0,9407	0,9640	0,9522	3,68%
Stateless	1,282s	0,9244	0,9333	0,9288	3,92%

Fonte: Elaborado pelo autor (2026).

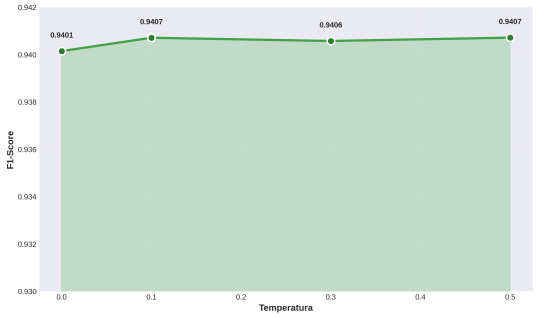
Os resultados consolidam o modo *Full History* como a configuração mais consistente para o sistema, uma vez que o incremento de 3,08 pontos percentuais no *Recall* e a redução da taxa de falsos negativos pela metade, reduzindo o risco de ataques graduais, justificam amplamente o acréscimo marginal de 0,05s na latência média. Essa escolha prioriza a robustez da segurança e a integridade da jornada do usuário, mitigando vulnerabilidades que o modo *Stateless* não é capaz de interceptar devido à ausência de contexto histórico.

4) Sensibilidade à Temperatura: A varredura de temperatura revelou que o sistema mantém uma estabilidade excepcional sob diferentes níveis de estocasticidade[20].

Ao analisar o desempenho do modelo em diferentes configurações de temperatura (variando de 0,0 a 0,5), observou-se que as oscilações no *F1-Score* são estatisticamente marginais, conforme demonstrado na figura. 8.

Fig. 8

IMPACTO DE DIFERENTES FAIXAS DE TEMPERATURA NO F1-SCORE



Fonte: Elaborado pelo autor (2026).

Diferente do que se observa em tarefas criativas, a tarefa de classificação de segurança demonstrou alta resiliência à variação deste parâmetro. O sistema manteve um *F1-Score* médio de aproximadamente 0,94 em todos os cenários testados. Esta estabilidade sugere que o *prompt* e a estrutura lógica do sistema são robustos o suficiente para manter a consistência da classificação, independentemente do nível de amostragem probabilística aplicado na inferência. Para fins de produção, a escolha de temperaturas mais baixas (próximas a 0,0) é recomendada apenas para garantir a reprodutibilidade técnica, sem prejuízo à eficácia da detecção.

5) Robustez do Formato de Saída: A comparação entre os protocolos de resposta revelou distinções significativas entre o uso de *Structured Output* (*JSON Schema*) e *Tools* (*Function Calling*). Os dados consolidados na Tabela VII indicam que a escolha do protocolo impacta não apenas a acurácia da classificação, mas também a resiliência operacional do sistema.

TABELA VII

COMPARATIVO DE PERFORMANCE E ESTABILIDADE POR PROTOCOLO DE SAÍDA

Formato de Saída	**F1**	**Prec.**	**Rec.**	**Erro**
Json Schema	0,9400	0,9411	0,9389	5,97%
Function Calling	0,9411	0,9248	0,9579	1,62%

Fonte: Elaborado pelo autor (2026).

O protocolo *Tools (Function Calling)* demonstrou ser a abordagem mais eficaz para a detecção de ameaças, atingindo um *Recall* de 0,9579. Esta maior sensibilidade indica que o modelo, quando operando via funções, consegue identificar uma gama mais ampla de intenções maliciosas. Além disso, este formato apresentou uma robustez sintática superior, com uma taxa de erro de *parsing* de apenas 1,62% — um índice aproximadamente 3,7 vezes menor que o registrado pelo *JSON Schema*.

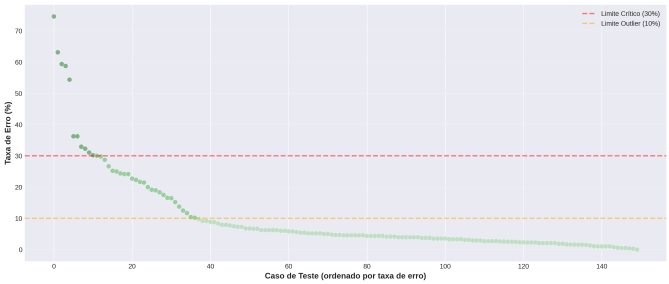
Por outro lado, o *JSON Schema* apresentou uma Precisão ligeiramente superior (0,9411), sugerindo um perfil mais conservador que minimiza bloqueios indevidos de usuários legítimos. Contudo, devido à maior incidência de falhas de formatação e ao *Recall* inferior, o uso de *Tools* consolida-se como a arquitetura preferencial para cenários de alta disponibilidade e rigor defensivo, equilibrando a máxima captura de ameaças com a menor taxa de falha técnica.

C. Análise de Outliers: Padrões de Falha e Causas Raiz

No universo de 150 casos analisados, foram identificados e isolados 37 casos classificados como *outliers*[21]. A seleção desses casos baseou-se em um critério de criticidade técnica, definindo como *outliers* todas as mensagens que apresentaram uma taxa de erro total (classificação e *parsing*) superior a 10,0%. Esse subconjunto foi submetido a uma análise detalhada para mapear comportamentos anômalos e falhas sistemáticas nos modelos de linguagem avaliados.

Fig. 9

GRÁFICOS DE DISPERSÃO - TAXA DE ERRO VS. CASOS DE TESTE



Fonte: Elaborado pelo autor (2026).

A análise dos *outliers* revela vulnerabilidades sistêmicas distintas entre modelos de grande escala e modelos compactos, além de demonstrar como a configuração do ambiente influencia a precisão do *Guardrail*.

1) Tipologia de Mensagens com Maiores Índices de Falha: As falhas de classificação do modelo concentram-se em três perfis principais, revelando limitações no tratamento de ambiguidades semânticas e técnicas de evasão:

- **Ambiguidade e Falta de Contexto (Falsos Positivos):** Respostas afirmativas simples ou escolhas de menu (ex: “1”, “Sim”) são frequentemente mal interpretadas como potenciais riscos devido à falta de densidade semântica.

[20]Refere-se à natureza probabilística e aleatória do processo de seleção de *tokens*. Em LLMs, uma maior estocasticidade resulta em respostas mais variadas e criativas, enquanto uma baixa estocasticidade torna o modelo mais determinístico e previsível.

[21]Do inglês *outlier*. Refere-se a observações que se desviam drasticamente da média do conjunto de dados, podendo indicar erros de medição, falhas de execução no *pipeline* ou comportamentos atípicos que distorcem a análise estatística.

- **Requisições Out-Of-Scope (Falsos Negativos):** Este ponto levanta uma questão conceitual sobre a arquitetura do Guardrail. Pedidos puramente estéticos ou literários são contabilizados como Falsos Negativos, mas a hipótese levantada é que fugas de escopo não possuem a natureza semântica de uma "mensagem não segura" pela definição adotada. Portanto, elas não seriam falhas de segurança, mas sim de aderência ao domínio, não devendo ser processadas pela mesma lógica de detecção de danos.
- **Dados Sensíveis e Estruturas Irregulares:** A presença de padrões numéricos rígidos ou ruídos de formatação (espaçamentos excessivos) confunde, sobretudo, os modelos de menor escala, gerando bloqueios indevidos.

2) Combinações de Parâmetros e Desempenho Crítico: A análise detalhada dos *outliers* revela que as falhas não são distribuídas uniformemente, mas concentram-se em combinações específicas de parâmetros que potencializam as fraquezas inerentes de cada modelo.

- **Interação entre Modelo e Formato de Saída**: A análise da Tabela VIII revela que o formato de saída altera drasticamente o comportamento de segurança dos modelos. Esse fenômeno indica que modelos menores adotam uma postura conservadora para compensar incertezas lógicas quando forçados a seguir protocolos rígidos.

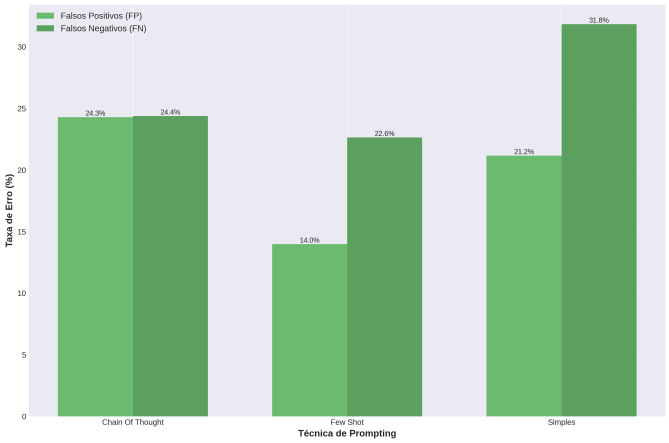
TABELA VIII
PRINCIPAIS COMBINAÇÕES PROBLEMÁTICAS ENTRE MODELO E FORMATO

Modelo	Formato	Tipo de Erro	% do Total
GPT 4.1 Mini	JSON Schema	FN	29,3%
GPT 4.1 Mini	Tools	FN	19,1%
GPT 4.1 Nano	Tools	FP	28,3%
GPT 4o Mini	JSON Schema	FP	22,1%
GPT 4o Mini	Tools	FP	21,0%

Fonte: Elaborado pelo autor (2026).

- **Eficácia das Estratégias de Prompting e Contexto**: A configuração do prompt e a gestão do histórico de mensagens alteram drasticamente o comportamento do *Guardrail*:
  - **Análise de Estratégias de Prompting**: A análise da Figura 13 revela que a concentração de *outliers* com taxas de erro superiores a 10% pode estar diretamente vinculada a falhas de julgamento lógico induzidas por técnicas específicas de *prompting*. Observou-se que métodos de raciocínio estendido, como o *Chain-of-Thought* (CoT), embora aumentem a assertividade em casos gerais, tendem a gerar alucinações de segurança em cenários ambíguos, resultando em classificações erradas.

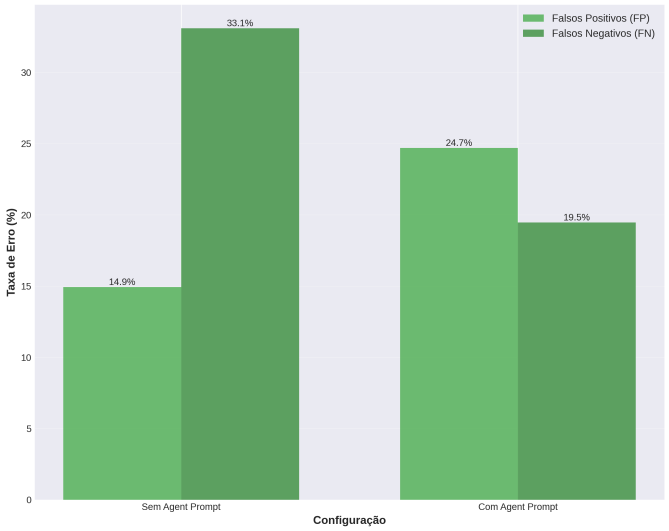
Fig. 10
COMPARATIVO TAXA DE ERRO VS. TÉCNICAS DE PROMPTING - OUTLIERS



Fonte: Elaborado pelo autor (2026).

  - **O Impacto do Agent Prompt**: A inclusão do *Agent Prompt* atua como um reforço crítico de vigilância, mas apresenta-se como uma "faca de dois gumes". Por um lado, sua ativação reduziu significativamente a taxa de Falsos Negativos, por outro lado, essa postura defensiva elevou os Falsos Positivos. Esse comportamento indica que instruções adicionais aumentam a sensibilidade do modelo, levando-o a bloquear mensagens ambíguas por precaução.

Fig. 11
COMPARATIVO TAXA DE ERRO VS. AGENT PROMPT - OUTLIERS

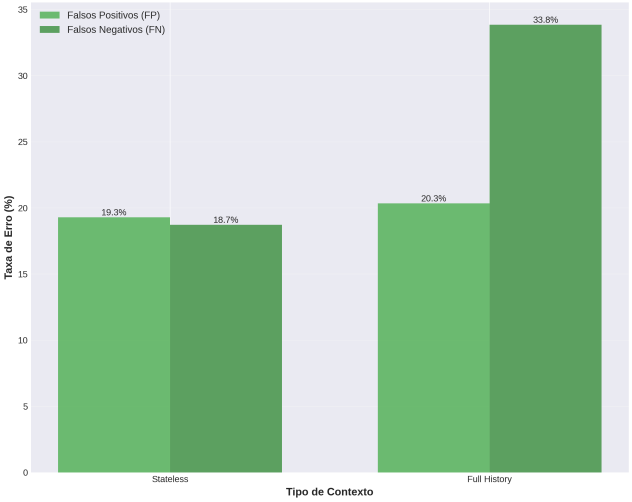


Fonte: Elaborado pelo autor (2026).

  - **Gestão de Contexto e Histórico**: No modo *Stateless*, a distribuição de erros é praticamente simétrica (19% para FP e FN). Isso indica que, sem memória, o modelo falha de forma genérica.

Por outro lado, embora o histórico melhore a média geral, ele altera o perfil de falha nos casos críticos de forma perigosa. O salto dos Falsos Negativos (FN) para 33,8% revela que, em cenários de alta complexidade, o excesso de contexto atua como um ruído.

Fig. 12
COMPARATIVO TAXA DE ERRO VS. CONTEXTO - OUTLIERS

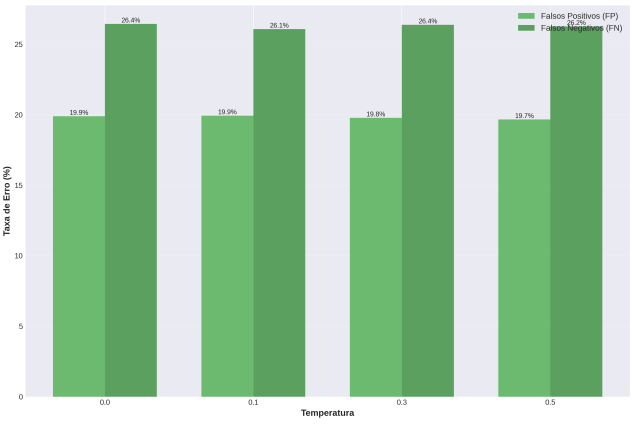


Fonte: Elaborado pelo autor (2026).

O aumento expressivo de FNs em detrimento da estabilidade dos FPs (20,3%) no modo *Full History* sugere que o modelo se torna mais "confiante" ou "relaxado" à medida que a conversa avança. Ele tende a assumir que, se as interações anteriores foram seguras, a atual também o é, permitindo que ataques de *jailbreak* multi-etapas passem despercebidos.

– **Impacto da Temperatura**: Embora a temperatura tenha apresentado um efeito marginal na média de erro geral das amostras, observou-se que valores intermediários podem induzir instabilidades em modelos específicos.

Fig. 13
COMPARATIVO TAXA DE ERRO VS. TEMPERATURA - OUTLIERS



Fonte: Elaborado pelo autor (2026).

O modelo GPT-4.1-mini, por exemplo, atingiu seu pico de vulnerabilidade em temperatura 0.3, acumulando 216 Falsos Negativos. Esse comportamento sugere que níveis moderados de aleatoriedade podem comprometer a aderência do modelo às diretrizes de segurança sem necessariamente melhorar a fluidez da resposta.

3) Análise das Causas Raiz: As falhas observadas nos casos críticos (*outliers*) não ocorrem de forma aleatória, mas derivam de três pilares estruturais que definem o comportamento dos modelos sob diferentes condições de contorno:

1) **Equilíbrio entre Escala Cognitiva e Restritividade:** Observou-se uma correlação inversa entre a escala da arquitetura e a sua permissividade. Modelos mais compactos, como o *GPT 4.1 Nano* e o *GPT 4o Mini*, tendem a adotar uma postura de "bloqueio por precaução" devido à menor capacidade de processar nuances semânticas complexas. Essa rigidez cognitiva faz com que interpretem sequências numéricas isoladas ou mensagens extremamente curtas como ameaças, sendo essas arquiteturas responsáveis por 85,6% de todos os falsos positivos registrados. Em contrapartida, modelos como o *GPT 4.1 Mini* demonstram uma subestimação crônica de ameaças sutis, originando 48,4% do total de falsos negativos.
2) **Determinismo e Inflexibilidade do Formato de Saída:** O protocolo de resposta exerce influência decisiva no rigor da classificação. A interface baseada em *Tools (Function Calling)* impõe uma lógica de execução que se mostrou inerentemente mais rígida que o *JSON Schema*, resultando em uma taxa de falsos positivos 50% superior. Essa rigidez estrutural frequentemente força o modelo a uma classificação binária errônea, dificultando a dissociação entre a forma da mensagem e a intenção maliciosa subjacente. O caso mais emblemático envolve pedidos de conteúdos criativos, como poemas ou histórias, que atingiram uma taxa de 72,9% de falsos negativos por conseguirem ludibriar a camada de segurança através da camuflagem semântica.
3) **Déficit de Percepção em Processamento *Stateless*:** A ausência de retenção de memória no modo *Stateless* degradou significativamente a acurácia, apresentando o dobro da taxa de erro em ameaças de injeção quando comparado ao uso do histórico completo. Sem o contexto prévio, o modelo sofre de uma "amnésia funcional", falhando em identificar que interações aparentemente inócuas como o convite "vamos jogar um jogo" constituem, na realidade, estágios iniciais de tentativas deliberadas de subversão do fluxo de instruções do sistema (*jailbreak* multi-etapa).

IV. IDENTIFICAÇÃO DAS CONFIGURAÇÕES DE ELITE

A análise final cruzou todas as variáveis experimentais para determinar as combinações que oferecem a melhor viabilidade operacional.

O ranking (Figura 14) foi estabelecido através do Score Ajustado, uma métrica que pondera o F1-Score pela estabilidade sintática, penalizando configurações que tornam o sistema inoperante em produção:

$$\text{Score Ajustado} = \text{F1-Score} \times (1 - \text{Taxa de Erro de Parsing})$$

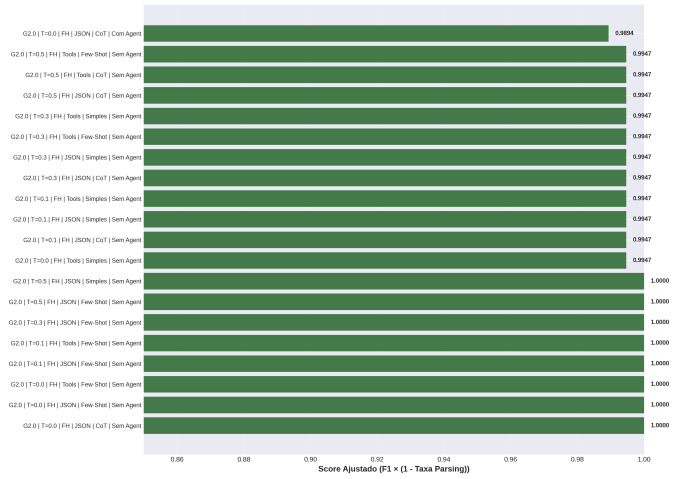
Comparativamente à média geral, as configurações de elite apresentam um salto qualitativo significativo, conforme ilustrado na Tabela IX.

TABELA IX
COMPARAÇÃO ENTRE AS VINTE MELHORES COMBINAÇÕES E A MÉDIA GERAL

Métrica	Top 20 (Elite)	Média Geral
F1-Score médio	0,9965	0,9364
Erro de Parsing	0,00%	2,15%
Score Ajustado	0,9965	0,9154

Fonte: Elaborado pelo autor (2026).

Fig. 14
TOP 20 CONFIGURAÇÕES DE ELITE



Fonte: Elaborado pelo autor (2026).

A. O Domínio do Modelo Gemini 2.0 Flash

O fator de maior impacto para atingir a performance de elite foi a escolha da arquitetura. O *Gemini 2.0 Flash* ocupa 100% das posições no Top 20. O seu diferencial competitivo reside na estabilidade estrutural em relação ao seu concorrente direto, *Gemini 2.5 Flash*, ao a manter uma taxa de erro de *parsing* nula (0,00%) em todas as combinações de topo, um pré-requisito fundamental para a fiabilidade em sistemas de produção.

B. Padrões de Configuração e Fatores Críticos

Ao analisar as 20 melhores combinações, emergem padrões claros que definem o estado da arte para o sistema de *Guardrails*:

- **Gestão de Contexto:** O uso de *Full History* é obrigatório para a performance máxima, estando presente em 100% das configurações do *Top 20*. A manutenção do histórico completo provou ser essencial para a precisão da detecção.
- **Agent Prompt:** A omissão do *Agent Prompt* é benéfica. Observa-se que 95% das configurações de elite não o utilizam, uma vez que a sua inclusão tende a aumentar erros de estruturação sem ganho proporcional na detecção.
- **Técnicas de Prompting:** O *Few-Shot* lidera com 40% de presença, demonstrando ser a técnica mais equilibrada. O *Chain of Thought* (30%) e o *Simples* (30%) também são viáveis, desde que o modelo garanta estabilidade no *parsing*.
- **Formato de Saída:** O *JSON Schema* possui uma vantagem (60%) sobre o formato *Tools* (40%), embora ambos apresentem robustez elevada nas mãos do Gemini 2.0 Flash.

C. Recomendações para Implementação em Produção

Com base na hierarquia de importância identificada, a configuração ideal para um ambiente de produção (*Golden Path*) deve seguir a prioridade dos parâmetros detalhados na Tabela X.

TABELA X
HIERARQUIA DE PARÂMETROS PARA CONFIGURAÇÃO DO SISTEMA

Parâmetro	Prioridade
Modelo	Crítica (Estabilidade)
Contexto	Alta (Precisão)
Agent Prompt	Alta (Parsing)
Prompting	Média
Formato	Média
Temperatura	Baixa

Fonte: Elaborado pelo autor (2026).

a) Configurações a Evitar:: Para garantir a integridade do sistema, devem ser evitados modelos com volatilidade de *parsing* frequente (como o Gemini 2.5 Flash associado a CoT) e contextos do tipo *Stateless*, que degradam severamente a capacidade de reconhecimento de padrões de ataque.

V. DELINEAMENTO DE RESPONSABILIDADES: SEGURANÇA VS. ESCOPO

A eficiência de um sistema de segurança para IA conversacional reside na separação rigorosa entre a integridade técnica do modelo e a sua lógica funcional de negócio.

Esta pesquisa demonstra que a sobrecarga da camada de *Guardrail* com instruções contextuais do agente (*Agent Prompt*) pode constituir uma causa raiz para a degradação da experiência do usuário, uma vez que a ambiguidade léxica introduzida por regras externas eleva a incidência de bloqueios indevidos.

Tal fenômeno é corroborado pela análise das configurações de elite (*Top 20*), onde observou-se que 95% das combinações de maior performance abdicam do *Agent Prompt*. Essa tendência indica que a exclusão de contextos operacionais não essenciais permite ao modelo dedicar sua capacidade atencional exclusivamente à detecção de ameaças, resultando em um sistema mais robusto e com menor atrito para solicitações legítimas.

A. O Guardrail como Camada de Segurança Técnica

O *Guardrail* deve atuar exclusivamente como uma camada de interceptação síncrona, focada na mitigação de ameaças que comprometam a integridade e a confiabilidade do sistema.

- **Foco em Ameaças Sistêmicas:** Sua responsabilidade limita-se à detecção de *Prompt Injection*, ataques de manipulação de instruções e filtragem de conteúdos ofensivos. Conforme preconizado pela OWASP, o objetivo primordial é garantir a separação rigorosa entre as instruções do sistema e os dados fornecidos pelo usuário, tratando o *input* puramente como informação a ser processada, e não como comando [1].
- **Neutralidade de Domínio:** A camada de *Guardrail* deve ser agnóstica às regras de negócio ou especificidades de produtos. Sua análise deve ser estritamente técnica e binária (*safe* ou *unsafe*), assegurando que tentativas de subversão sejam barradas independentemente do contexto comercial ou da aplicação final [2].
- **Prevenção de Falsos Positivos:** Para otimizar a precisão operacional, os *Guardrails* devem concentrar-se na segurança e integridade sistêmica, seguindo padrões de severidade para riscos críticos como ódio, violência e *jailbreak* [3]. Ao isolar a verificação de "escopo" da camada de proteção, evita-se que termos legítimos de suporte, como "cancelar" ou "devolução", sejam erroneamente sinalizados como anomalias, reduzindo o índice de falsos positivos e mitigando atritos no atendimento ao usuário.

B. O Agente como Gestor de Escopo e Utilidade

O Agente Principal deve ser o único responsável pelo discernimento semântico e pela aplicação das regras de negócio, atuando na camada de lógica da aplicação.

- **Discernimento de Escopo Funcional:** O Agente identifica solicitações seguras, porém fora de sua competência (ex.: produtos não integrados). Em vez de um bloqueio técnico, ele fornece uma resposta dialógica de "fora de escopo", preservando a fluidez da interação e a experiência do usuário.
- **Gestão de Utilidade vs. Segurança:** Ao centralizar o escopo no Agente, o sistema diferencia falhas de utilidade de violações de segurança. Isso permite que interações inesperadas sejam tratadas como lacunas de conhecimento, mantendo a prestatividade sem acionar gatilhos de segurança desnecessários.

C. Síntese da Segregação

A Tabela XI resume o modelo arquitetural proposto para garantir o equilíbrio entre proteção e fluidez conversacional.

TABELA XI
MATRIZ DE RESPONSABILIDADES: GUARDRAIL VS. AGENTE

Característica	Guardrail (Segurança)	Agente (Escopo)
Missão	Proteção contra ataques e injeções	Gestão de utilidade e regras de negócio
Contexto	Analisa a segurança da mensagem	Analisa a intenção da solicitação
Dados	Não (foca na detecção de padrões maliciosos)	Sim (valida, cadastra e processa dados)

Fonte: Elaborado pelo autor (2026).

Em suma, a segurança reside na soberania do filtro externo (*Guardrail*), enquanto a utilidade reside no discernimento semântico do Agente. Esta segregação impede que o conservadorismo necessário para a segurança técnica prejudique a flexibilidade exigida pelo atendimento ao cliente.

VI. CONCLUSÃO

A análise abrangente de 150 casos de teste e 64.818 execuções experimentais válidas (do universo inicial de 72.000 interações planejadas, 7.182 execuções passaram por bloqueios nativos e foram analisadas separadamente) consolida as bases para a otimização de *Guardrails* de segurança em agentes conversacionais, estabelecendo as seguintes determinações:

- **Desempenho Global do Sistema**: Os resultados validam a eficácia da solução e mostram que a arquitetura é extremamente promissora. O sistema já apresenta um comportamento consistente, restando apenas etapas de otimização de configuração para elevar o sistema ao seu nível máximo de performance e segurança.
- **Supremacia do Gemini 2.0 Flash**: Este modelo definiu o estado da arte na pesquisa, apresentando 0,00% de erro de *parsing* e com um F1-Score de 97,81%, ele equilibra perfeitamente a sensibilidade de detecção com a estabilidade operacional necessária para ambientes de produção.
- **Inviabilidade Operacional do Gemini 2.5 Flash**: Apesar de métricas de classificação competitivas, o Gemini 2.5 Flash apresentou uma taxa de falha sintática crítica de 16,99%, tornando-o tecnologicamente inviável para *Guardrails* que dependem de saídas estruturadas.
- **Limitações de Modelos Reduzidos**: Os modelos *Mini* e *Nano* (GPT-4) demonstraram viés conservador excessivo, elevando a taxa de Falsos Positivos ao bloquear mensagens legítimas (Precision de 82,49% no modelo Nano), o que degrada severamente a experiência do usuário (UX).

A. Impacto das Variáveis e Diretrizes de Arquitetura

A análise multivariada permitiu isolar os componentes que maximizam a eficácia do sistema, estabelecendo as diretrizes para a configuração de produção:

- **Gestão de Contexto**: A utilização de *Full History* provou ser obrigatória, elevando o F1-Score em 2,34 pontos percentuais. O contexto histórico é o principal fator de desambiguação para validar intenções que, isoladamente, pareceriam maliciosas, ou identificar ataques multi-turno.
- **Estratégias de Prompting**: A técnica *Few-Shot* consolidou-se como a mais equilibrada (F1: 0,9565; Erro Parsing: 1,36%). Embora o *Chain-of-Thought* (CoT) potencialize o raciocínio, sua alta taxa de erro sintático (8,24%) limita sua aplicação a modelos de altíssima estabilidade.
- **Isolamento de Identidade**: A ausência do *Agent Prompt* na camada de segurança é um padrão no *Top 20* das melhores configurações. A inclusão de instruções do agente no *Guardrail* gera conflito cognitivo no modelo, aumentando bloqueios indevidos e erros de *parsing*.
- **Formatos de Saída**: Tanto *JSON Schema* quanto *Tools* mostraram-se viáveis. O *JSON Schema* oferece uma leve vantagem em precisão (menos bloqueios indevidos), enquanto o *Tools* apresenta um *Recall* superior, sendo preferível quando a tolerância ao risco de segurança é mínima.

B. Taxonomia de Falhas e Fronteiras de Vulnerabilidade

A análise dos 37 *outliers* identificados (24,7% do dataset) revela padrões de falha que delimitam os desafios atuais da IA:

- **Padrões de Falsos Positivos (Bloqueio Indevido)**: Concentram-se em mensagens ultra-curtas, números isolados (CPFs, protocolos) ou formatações irregulares. Modelos menores tendem a classificar ruído sintático como tentativa de evasão por precaução.
- **Padrões de Falsos Negativos (Vazamento de Escopo)**: O maior desafio reside em mensagens *out-of-scope* (57,7% de erro) que não são identificadas pela arquitetura atual como ameaças. Isso ocorre devido à lógica de mercado utilizada, que prioriza a detecção de padrões de ataque explícitos, falhando ao processar solicitações de natureza estética ou literária que, embora fora do domínio, não ativam os gatilhos de segurança convencionais.
- **Trade-off Segurança vs. Usabilidade**: Modelos menores priorizam segurança (alto Recall, baixa Precision), enquanto modelos maiores (Gemini 2.0 Flash) conseguem o discernimento necessário para manter a fluidez do diálogo sem abrir brechas de integridade.

C. Direcionamento Estratégico para Produção

Com base nas evidências empíricas, a arquitetura de referência para implementação imediata é definida pela configuração:

> *Gemini 2.0 Flash + Full History + Few-Shot Prompting + JSON Schema + Temperatura 0.0 a 0.5 + Sem Agent Prompt.*

Esta combinação alcançou F1-Score de 100% em testes específicos, garantindo que o *Guardrail* atue como uma camada de segurança técnica pura, delegando ao agente principal a lógica de escopo de negócio.

D. Considerações Finais

A pesquisa conclui que a otimização científica de *Guardrails* permite superar o dilema entre rigor defensivo e agilidade conversacional. O Gemini 2.0 Flash emerge como o pilar central desta arquitetura, sendo o único capaz de oferecer estabilidade sintática absoluta e discernimento semântico superior. A segurança da plataforma não deve ser vista como um limitador de interações, mas como um diferencial de robustez, onde a excelência técnica na filtragem de dados protege a integridade do sistema e a confiança do usuário final.

VII. Referências

[1] O. Foundation, *LLM Prompt Injection Prevention Cheat Sheet*, Guia de referência para segurança contra injeção de prompt, 2024. endereço: https://cheatsheetseries.owasp.org/cheatsheets/LLM_Prompt_Injection_Prevention_Cheat_Sheet.html.

[2] M. AI, *Llama-Prompt-Guard-2-86M: A defensive model for prompt injection*, 2024. endereço: https://huggingface.co/meta-llama/Llama-Prompt-Guard-2-86M.

[3] Microsoft, *Default safety policies in Azure AI Foundry*, 2024. endereço: https://learn.microsoft.com/en-us/azure/ai-foundry/foundry-models/concepts/default-safety-policies.

[4] I. Research, "Chain of Thought Prompting and its implications in LLM reasoning", *arXiv preprint*, 2024. endereço: https://www.ibm.com/think/topics/chain-of-thoughts.

[5] A. Sharma, "When to use Function Calling vs Structured Outputs in LLM Applications", *arXiv / Vellum technical review*, 2024. endereço: https://www.vellum.ai/blog/when-should-i-use-function-calling-structured-outputs-or-json-mode.

[6] N. Kirch, C. Weisser, S. Field, H. Yannakoudakis e S. Casper, "What Features in Prompts Jailbreak LLMs? Investigating the Mechanisms Behind Attacks", *arXiv:2411.03343*, 2024. endereço: https://arxiv.org/abs/2411.03343.