Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Análise e Síntese de uma Arquitetura de Referência para Chatbots

Ticiano de Oliveira Fracette

JUIZ DE FORA
JULHO, 2026

Análise e Síntese de uma Arquitetura de Referência para Chatbots

Ticiano de Oliveira Fracette

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: André Luiz de Oliveira
Coorientador: Pedro Henrique Dias Valle

JUIZ DE FORA
JULHO, 2026

ANÁLISE E SÍNTESE DE UMA ARQUITETURA DE REFERÊNCIA PARA CHATBOTS

Ticiano de Oliveira Fracette

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

André Luiz de Oliveira
Doutor em Ciência da Computação pela Universidade de São Paulo

Pedro Henrique Dias Valle
Doutor em Ciência da Computação pela Universidade de São Paulo

Bárbara de Melo Quintela
Doutora em Modelagem Computacional pela UFJF

Igor de Oliveira Knop
Doutor em Modelagem Computacional pela UFJF

JUIZ DE FORA
10 DE JULHO, 2026

À minha família, pelo apoio incondicional em cada etapa desta jornada.

À minha namorada, pela parceria ímpar, pela compreensão e incentivo constantes.

Aos professores que, com paciência e compromisso, tornaram possível não apenas este trabalho, mas também a construção da minha trajetória acadêmica.

Resumo

Contextualização/Motivação: A expansão acelerada dos sistemas conversacionais no contexto tecnológico contemporâneo tornou evidente um problema estrutural recorrente: a maioria dos chatbots é desenvolvida de forma ad-hoc, sem apoio de modelos arquiteturais consolidados, o que resulta em sistemas frágeis, monolíticos, de difícil manutenção e com baixa interoperabilidade. **Objetivo**: Neste trabalho é apresentada a proposta de uma Arquitetura de Referência (AR) para o desenvolvimento de aplicações baseadas em chatbots. **Metodologia**: A arquitetura foi desenvolvida seguindo o processo ProSA-RA (Process for Software Architectures - Reference Architectures).Primeiramente, durante a investigação de fontes de informação, foi conduzida uma revisão sistemática de literatura e análise de sistemas existentes em sete domínios distintos. Posteriormente, na fase de análise arquitetural, os requisitos foram elicitados e categorizados em requisitos funcionais e não funcionais essenciais. Por fim, durante a síntese arquitetural, foram especificadas visões arquiteturais documentadas segundo o modelo "4+1"utilizando diagramas UML. A arquitetura de referência proposta poderá ser utilizada como base (blueprint) conceitual para o desenvolvimento de aplicações baseadas em chatbots. Para avaliar a viabilidade arquitetura proposta, a mesma foi instanciada na especificação do projeto arquitetural de um chatbot para o domínio educacional.

Palavras-chave: Arquitetura de Referência. Chatbots. ProSA-RA.

Agradecimentos

Ao professor André Luiz de Oliveira, pela orientação.

Ao professor Pedro Henrique Dias Valle, pela coorientação.

Aos professores do Departamento de Ciência da Computação da UFJF, cujos ensinamentos ao longo dos anos de graduação formaram a base sobre a qual este trabalho foi construído.

À minha família e amigos, pelo encorajamento constante e pela paciência nos momentos de maior dedicação ao trabalho.

A todos que, de alguma forma, contribuíram para a minha formação acadêmica e pessoal, registro aqui minha sincera gratidão.

"A arquitetura representa as decisões de design significativas que moldam um sistema, onde significativo é medido pelo custo de mudança."

Grady Booch

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AIML	Artificial Intelligence Markup Language
API	Application Programming Interface
AR	Arquitetura de Referência
ATAM	Architecture Trade-off Analysis Method
AVA	Ambiente Virtual de Aprendizagem
BERT	Bidirectional Encoder Representations from Transformers
CRM	Customer Relationship Management
DCC	Departamento de Ciência da Computação
DM	Dialogue Manager (Motor de Diálogo)
ELK	Elasticsearch, Logstash, Kibana
FERA	Framework for Evaluation of Reference Architectures
GDPR	General Data Protection Regulation
GPT	Generative Pre-trained Transformer
IA	Inteligência Artificial
IoT	Internet of Things
IVR	Interactive Voice Response
LGPD	Lei Geral de Proteção de Dados
LLM	Large Language Model
ML	Machine Learning
NLG	Natural Language Generation
NLU	Natural Language Understanding
PLN	Processamento de Linguagem Natural
ProSA-RA	Process for Software Architectures – Reference Architectures
RAG	Retrieval-Augmented Generation
RAModel	Reference Architecture Model
RNF	Requisito Não Funcional
RSL	Revisão Sistemática de Literatura
SAAM	Software Architecture Analysis Method
SOA	Service-Oriented Architecture
SPL	Software Product Line
SysML	Systems Modeling Language
UFJF	Universidade Federal de Juiz de Fora
UML	Unified Modeling Language
WCAG	Web Content Accessibility Guidelines

1 Introdução

1.1 Contextualização

O surgimento e a proliferação de interfaces conversacionais automatizadas representam uma transformação significativa na Interação Humano-Computador na última década. Os chatbots, definidos como aplicações de software capazes de simular diálogos humanos por meio de texto ou voz, deixaram de ser curiosidades experimentais para se tornarem componentes centrais de produtos e serviços em escala global. Plataformas de atendimento ao cliente, assistentes de saúde, tutores educacionais, interfaces bancárias e agentes governamentais incorporam, hoje, alguma forma de sistema conversacional (DALE, 2016).

A expansão de aplicações baseadas em chatbots foi impulsionadapela convergência de três fatores técnicos: o amadurecimento das técnicas de Processamento de Linguagem Natural (PLN), a disponibilidade de infraestrutura computacional elástica em nuvem e, sobretudo, o surgimento dos modelos de linguagem de grande escala (Large Language Models, LLMs) baseados na arquitetura Transformer (VASWANI et al., 2017). Modelos como GPT, BERT e seus sucessores redefiniram o estado da arte em compreensão e geração de linguagem, tornando viável a construção de chatbots com capacidade de resposta contextual sofisticada em tempo real.

Contudo, a velocidade dessa expansão criou um problema estrutural silencioso: a fragmentação arquitetural. Equipes de desenvolvimento, pressionadas por prazos e pela lógica de entrega incremental, frequentemente constroem chatbots de forma ad-hoc, sem apoio de modelos arquiteturais consolidados. O resultado são sistemas monolíticos, acoplados a plataformas específicas, com baixa capacidade de evolução, manutenção custosa e interoperabilidade limitada (NAKAGAWA et al., 2014). Cada nova equipe reinventa soluções para problemas já amplamente conhecidos e estudados pela comunidade de engenharia de software.

Nesse contexto, a adoção de uma Arquitetura de Referência (AR) emerge como

resposta natural. Uma AR não é a arquitetura de um sistema específico, mas um modelo abstrato que captura o conhecimento acumulado sobre como sistemas de um determinado domínio devem ser estruturados, definindo componentes essenciais, suas responsabilidades, interfaces e interações, funcionando como um blueprint que orienta o desenvolvimento de instâncias concretas sem impor rigidez excessiva (BASS; CLEMENTS; KAZMAN, 2012).

1.2 Problema de Pesquisa

O domínio de chatbots carece de uma Arquitetura de Referência formalmente especificada, derivada de um processo sistemático e capaz de orientar o desenvolvimento de sistemas conversacionais com qualidade estrutural garantida desde as fases iniciais do ciclo de vida.

Esse problema se manifesta em três dimensões concretas. Primeiro, na dimensão técnica, onde a ausência de padronização leva ao desperdício de esforço e à repetição de falhas arquiteturais conhecidas. Segundo, na dimensão de qualidade, onde atributos como segurança, escalabilidade, confiabilidade e interoperabilidade são tratados como preocupações tardias, quando deveriam ser requisitos de primeira ordem desde o projeto. Terceiro, na dimensão de conhecimento, onde a experiência acumulada em implementações bem-sucedidas de chatbots não é sistematicamente capturada e disponibilizada para reutilização.

1.3 Justificativa

A justificativa para este trabalho assenta em três pilares complementares. Do ponto de vista científico, a área de arquiteturas de referência para domínios emergentes representa um campo ativo de pesquisa, com demanda por processos sistemáticos e resultados reproduzíveis. A aplicação do ProSA-RA ao domínio de chatbots contribui para a expansão do conhecimento sobre como esse processo se comporta em domínios de software intensivo em inteligência artificial, um território ainda pouco explorado pela literatura especializada.

Do ponto de vista prático, uma AR para chatbots reduz o custo e o tempo de desenvolvimento, diminui o risco técnico e facilita a integração com sistemas legados. Equipes que partem de um modelo arquitetural consolidado concentram esforço na lógica de negócio e na experiência do usuário, em vez de resolver continuamente os mesmos desafios infraestruturais.

Do ponto de vista da qualidade, a existência de uma AR permite que atributos como segurança, escalabilidade e interoperabilidade sejam tratados como propriedades de primeira ordem, endereçadas na fase de projeto e não apenas como correções posteriores.

1.4 Objetivos

O objetivo geral deste trabalho é propor uma Arquitetura de Referência para o domínio de chatbots, documentada segundo o modelo de visões "4+1" e derivada por meio do processo sistemático ProSA-RA.

Os objetivos específicos que sustentam esse propósito são: conduzir, durante a investigação de fontes de informação, uma investigação exaustiva sobre tecnologias, padrões e implementações de chatbots em múltiplos domínios, identificando características comuns e lacunas; posteriormente, na fase de análise arquitetural, elicitar e catalogar os Requisitos Não Funcionais (RNFs) essenciais para chatbots modernos, organizando-os em categorias coerentes; por fim, durante a síntese arquitetural, projetar a estrutura da AR utilizando notação UML e o modelo "4+1", produzindo diagramas detalhados de sequência, componentes, atividades, implantação e casos de uso; especificar as responsabilidades, interfaces e interações dos componentes fundamentais da AR; e, a fim de avaliar a arquitetura proposta, produzir uma instância conceitual da AR sobre um domínio específico, o educacional, demonstrando sua aplicabilidade prática.

1.5 Organização do Trabalho

O restante desta monografia está organizado da seguinte forma. No Capítulo 2 é apresentada a fundamentação teórica. No Capítulo 3 é descrita a revisão sistemática de literatura conduzida. No Capítulo 4 são detalhados os materiais e métodos e são descritas,

respectivamente, as etapas de investigação, análise e síntese arquitetural. No Capítulo 5 é apresentada a instância conceitual no domínio educacional. No Capítulo 6 é apresentada a análise e discussão crítica dos resultados. No Capítulo 7 é encerrado o trabalho.

2 Fundamentação Teórica

Neste capítulo é estabelecida a base conceitual e teórica que sustenta o desenvolvimento e a análise da Arquitetura de Referência para Chatbots. Serão explorados os fundamentos dos sistemas conversacionais, desde sua definição e evolução histórica até as diversas classificações e métodos de funcionamento que os caracterizam no cenário atual da inteligência artificial. Em paralelo, será detalhado o conceito de Arquitetura de Referência, sua importância na engenharia de software para a promoção de reuso e qualidade, e como ela se integra a processos de desenvolvimento estruturados. A compreensão desses pilares é essencial para contextualizar a proposta arquitetural deste trabalho e para justificar as decisões de projeto que visam a construção de chatbots robustos, escaláveis e eficientes.

2.1 Sistemas Conversacionais e Chatbots

Chatbots são aplicações de software que utilizam técnicas de Inteligência Artificial (IA), seja baseadas em regras ou em algoritmos de aprendizado de máquina, para simular interações humanas por meio de texto ou voz (SHAWAR; ATWELL, 2007). O termo abrange um espectro amplo de sistemas, desde interfaces de resposta automática baseadas em correspondência de padrões até agentes conversacionais sofisticados capazes de raciocínio contextual de longa duração.

A definição adotada neste trabalho considera chatbot como qualquer sistema de software capaz de: receber entradas em linguagem natural, seja textual ou vocal; interpretar a intenção comunicativa subjacente a essa entrada; recuperar ou gerar uma resposta adequada ao contexto; e apresentar essa resposta ao usuário de forma compreensível e relevante. Essa cadeia de operações envolve problemas computacionais de considerável complexidade, distribuídos ao longo de múltiplos subsistemas que precisam operar de forma coordenada.

2.1.1 Evolução Histórica

A trajetória histórica dos chatbots é marcada por saltos descontínuos, cada um associado a um avanço técnico fundamental. Compreender essa evolução é essencial para entender as decisões arquiteturais que a AR proposta precisa acomodar.

O marco inaugural é o ELIZA, desenvolvido por Joseph Weizenbaum no MIT entre 1964 e 1966. O sistema simulava um psicoterapeuta rogeriano por meio de correspondência de padrões e transformações de texto, sem qualquer compreensão semântica real. Sua capacidade de provocar no usuário a sensação de interação genuína com um ser humano surpreendeu o próprio criador e lançou as bases para o que posteriormente seria chamado de Efeito ELIZA: a tendência humana de atribuir inteligência e empatia a sistemas que meramente reproduzem padrões linguísticos (WEIZENBAUM, 1966).

Na década de 1970, o PARRY, desenvolvido por Kenneth Colby na Universidade Stanford, introduziu uma inovação conceitual importante: a modelagem de um estado interno. O sistema simulava um paciente paranoico, mantendo um conjunto de atitudes, medos e intenções que influenciavam suas respostas. Pela primeira vez, um chatbot apresentava comportamento contextualmente coerente ao longo de uma conversa.

Os anos 1990 trouxeram o Alice (Artificial Linguistic Internet Computer Entity), baseado na linguagem AIML. O Alice popularizou a ideia de chatbots acessíveis ao público geral pela internet e ganhou múltiplas edições do Prêmio Loebner, uma competição baseada no teste de Turing. Sua arquitetura de base de conhecimento explícita influenciou gerações de sistemas comerciais subsequentes.

O período 2010–2016 foi dominado pelos assistentes pessoais baseados em voz: Siri (Apple, 2011), Google Now (2012), Cortana (Microsoft, 2014) e Alexa (Amazon, 2014). Esses sistemas introduziram o reconhecimento de fala em larga escala, integração com serviços externos via API e aprendizado contínuo a partir de interações reais.

O período pós-2017 é marcado pela revolução dos Transformers. O artigo “Attention is All You Need” (VASWANI et al., 2017) introduziu uma arquitetura de rede neural que substituiu as redes recorrentes como paradigma dominante para tarefas de PLN. Os modelos BERT (DEVLIN et al., 2019) e GPT (RADFORD et al., 2018) demonstraram que o pré-treinamento em grandes corpora textuais produzia resultados sem precedentes

em compreensão e geração de linguagem.

2.1.2 Taxonomia

A classificação dos chatbots pode ser realizada segundo múltiplos critérios, cada um relevante para diferentes decisões arquiteturais. A Figura 2.1 apresenta a taxonomia estruturada adotada neste trabalho.

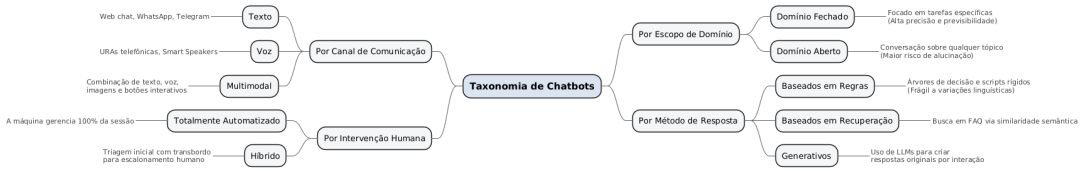

Figura 2.1: Taxonomia dos chatbots por critério de classificação.

Fonte: Autoria própria.

Pelo método de geração de resposta, os chatbots se dividem em três categorias fundamentais. Os sistemas baseados em regras utilizam árvores de decisão determinísticas e bases de conhecimento explícitas para mapear entradas a respostas predefinidas. São previsíveis, auditáveis e adequados para domínios bem delimitados, mas frágeis diante de variações linguísticas não previstas. Os sistemas baseados em recuperação selecionam respostas a partir de repositórios pré-existentes, utilizando métricas de similaridade semântica para identificar a resposta mais adequada. Os sistemas generativos utilizam modelos de aprendizado profundo para produzir respostas originais a cada interação, sem se restringir a um conjunto predefinido.

Pelo escopo de domínio, os chatbots se dividem entre sistemas de domínio fechado, focados em um conjunto específico de tópicos e tarefas, e sistemas de domínio aberto, capazes de conversar sobre qualquer assunto. Pelo nível de intervenção humana, variam de sistemas totalmente automatizados a sistemas híbridos. Pelo canal de comunicação, podem operar via texto em plataformas de mensagem, via voz em dispositivos dedicados, ou via interfaces multimodais.

Independentemente da categoria taxonômica, é possível identificar um conjunto de componentes funcionais presentes em qualquer sistema conversacional. O primeiro componente é a **interface de usuário**, responsável por receber a entrada do usuário e

apresentar a resposta. O segundo é o **módulo de NLU**, que transforma a entrada bruta em uma representação estruturada com intenção, entidades e sentimento. O terceiro é o **motor de diálogo** (DM), o núcleo decisório do sistema. O quarto é a **base de conhecimento**, que armazena as informações utilizadas para responder. O quinto é o **módulo de NLG**, que produz a resposta em linguagem natural. O sexto, presente em sistemas mais sofisticados, é o **módulo de integração externa**, que permite ao chatbot consultar ou modificar dados em sistemas de terceiros via APIs. A Figura 2.2 apresenta uma esquematização dessa arquitetura geral.

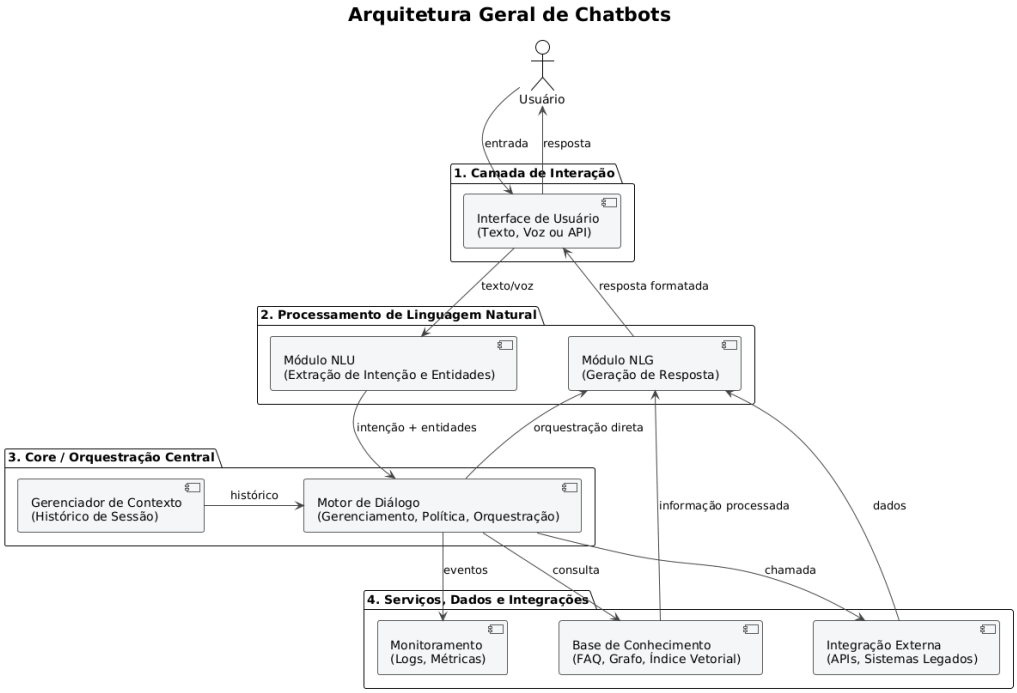

Figura 2.2: Arquitetura geral de chatbots (visão esquemática).

Fonte: Autoria própria, adaptado de Cahn (2017) e Huang e CIS (2021).

2.2 Arquitetura de Software

A arquitetura de software refere-se à organização fundamental de um sistema, englobando seus componentes, as relações entre eles, as relações com o ambiente e os princípios que guiam seu design e evolução (BASS; CLEMENTS; KAZMAN, 2012). É o conjunto de decisões de projeto de alto nível que têm impacto sistêmico e custo de mudança elevado, distinguindo-se das decisões de implementação, que são locais e reversíveis.

Há de se levar em consideração os atributos de qualidade, também chamados de requisitos não funcionais(RNFs), que são propriedades do sistema que não descrevem o que ele faz, mas como ele faz. São endereçados primariamente na fase arquitetural, pois decorrem de decisões estruturais que permeiam múltiplos componentes.

Os atributos mais relevantes para o domínio de chatbots incluem: escalabilidade, a capacidade do sistema de lidar com aumento de carga sem degradação proporcional de desempenho; disponibilidade, a proporção do tempo em que o sistema está operacional e acessível; segurança, a capacidade de proteger dados e operações contra acessos não autorizados; interoperabilidade, a facilidade com que o sistema se integra a outros sistemas; manutenibilidade, a facilidade de modificar o sistema para corrigir defeitos ou melhorar funcionalidades; e portabilidade, a capacidade de operar em diferentes ambientes de execução sem modificações substanciais.

Os estilos arquiteturais, por sua vez, são soluções recorrentes para problemas recorrentes no nível arquitetural. Para o domínio de chatbots, os estilos mais relevantes incluem: a arquitetura em camadas (*layered architecture*), que organiza o sistema em níveis de abstração com dependências unidirecionais; a arquitetura orientada a serviços (SOA) e a arquitetura de microsserviços, que decompõem o sistema em unidades funcionais independentes comunicando-se via interfaces padronizadas; o padrão de *pipeline*, que organiza o processamento como uma sequência de estágios transformadores; e o padrão de publicação-assinatura (*publish-subscribe*), adequado para sistemas com múltiplos canais de entrada e saída.

2.2.1 Documentação Arquitetural e o Modelo "4+1"

A documentação arquitetural tem como finalidade registrar, de forma estruturada e comunicável, as principais decisões de arquitetura de um sistema, seus elementos constituintes e as relações entre eles. Em termos práticos, ela funciona como o artefato que permite alinhar entendimento entre diferentes *stakeholders* (por exemplo, equipe de desenvolvimento, manutenção, gestão, segurança e operação), reduzindo ambiguidades sobre responsabilidades de componentes, interfaces e restrições técnicas. Além disso, ao explicitar decisões e racionalizações, a documentação facilita manutenção, evolução e ava-

liação de qualidade arquitetural ao longo do ciclo de vida do software, conforme discutido na literatura de arquitetura de software Bass, Clements e Kazman (2012).

Uma dificuldade recorrente na documentação é que uma única representação raramente é suficiente para descrever um sistema de software real. Há aspectos estruturais (módulos e dependências), comportamentais (interações em tempo de execução), organizacionais (mapeamento em código) e físicos (implantação em infraestrutura) que precisam ser compreendidos sob diferentes perspectivas. Para lidar com isso, Kruchten propôs o modelo de visões "4+1" Kruchten (1995), que organiza a descrição arquitetural em quatro visões principais, complementadas por uma quinta, centrada em cenários. Cada visão atende a preocupações específicas e, em conjunto, elas fornecem uma documentação mais completa e verificável.

No modelo "4+1", a **Visão Lógica** descreve a decomposição funcional do sistema, evidenciando os principais elementos responsáveis pelas funcionalidades percebidas pelo usuário e como eles colaboram. No contexto de chatbots, essa visão torna explícitos os módulos centrais que implementam a conversação, como NLU (compreensão), o motor de diálogo, o gerenciador de contexto, a base de conhecimento, o módulo de NLG (geração) e mecanismos de *fallback*. Diagramas de interação, como diagramas de sequência, são frequentemente utilizados para representar o fluxo de mensagens entre esses elementos, facilitando a compreensão de como uma requisição do usuário se transforma em uma resposta.

A **Visão de Desenvolvimento** (também referida em muitas abordagens como Visão de Implementação) descreve a organização do software do ponto de vista da construção do sistema, isto é, como o código é estruturado em módulos, pacotes, componentes, bibliotecas e camadas, bem como suas dependências. Essa visão é particularmente relevante para orientar decisões de modularidade, separação de responsabilidades e evolução do sistema. Em uma Arquitetura de Referência (AR) para chatbots, essa visão também favorece o reúso: componentes como NLU, NLG, integração externa e observabilidade podem ser desenhados como módulos substituíveis, o que permite instanciar a AR em múltiplos domínios com menor retrabalho. Diagramas de componentes são adequados para documentar essa perspectiva.

A **Visão de Processo** descreve os aspectos dinâmicos do sistema, com ênfase em concorrência, paralelismo, execução assíncrona, escalabilidade e tratamento de falhas em tempo de execução. Para chatbots, essa visão ajuda a justificar decisões como uso de filas, padrões de *circuit breaker* para integrações externas, estratégias de cache e mecanismos de escalonamento horizontal para lidar com picos de tráfego. Diagramas de atividades e descrições de fluxos operacionais tornam evidente como o sistema responde a entradas, toma decisões e coordena múltiplos serviços durante a execução.

A **Visão Física** (ou Visão de Implantação) descreve como os componentes de software mapeiam para a infraestrutura de hardware e rede, incluindo nós de computação, containers, orquestradores, bancos de dados, serviços de mensageria e pontos de borda (por exemplo, API Gateway). Essa visão é essencial para discutir atributos de qualidade como disponibilidade, desempenho, tolerância a falhas e segurança, pois explicita onde residem fronteiras de rede, pontos de observabilidade e políticas de acesso. Em geral, diagramas de implantação são usados para representar essa distribuição.

O elemento "+1" do modelo é a **Visão de Cenários**, normalmente expressa por casos de uso e narrativas de interação. Cenários têm função integradora: eles exercitam as demais visões, permitindo validar se o conjunto de decisões documentadas é suficiente para suportar objetivos reais de usuários e requisitos do domínio. Para um chatbot, cenários típicos incluem: responder perguntas frequentes, recuperar informações em base de conhecimento, executar ações transacionais via integração externa, lidar com baixa confiança de NLU, registrar logs para auditoria e realizar escalonamento para atendimento humano quando necessário.

Neste trabalho, o modelo "4+1" é adotado como estrutura de documentação da Arquitetura de Referência proposta, pois ele oferece uma forma sistemática de apresentar tanto a organização estática quanto o comportamento do sistema e sua implantação, reduzindo lacunas entre o que é projetado e o que é executado. Ao articular as cinco visões, torna-se possível conectar componentes arquiteturais aos requisitos não funcionais do domínio de chatbots, além de favorecer a compreensão, avaliação e instanciação da AR em diferentes contextos de aplicação.

2.3 Arquitetura de Referência

Uma Arquitetura de Referência (AR) pode ser compreendida como uma estrutura arquitetural abstrata, orientada a um domínio, que consolida conhecimento arquitetural recorrente e o expressa por meio de elementos, relações, restrições e decisões típicas, com o objetivo de servir como guia para a construção de sistemas concretos. Em contraste com a arquitetura de um único sistema, a AR não descreve uma implementação específica, mas sim um *blueprint* conceitual que captura soluções arquiteturais comuns, vocabulário do domínio e diretrizes de organização de componentes, permitindo que diferentes sistemas sejam projetados de maneira mais consistente e comparável Bass, Clements e Kazman (2012), Nakagawa, Oquendo e Becker (2012).

Entre as principais características de uma AR, destacam-se: (i) o foco em um domínio de aplicação, isto é, a AR é construída para um conjunto de sistemas que compartilham objetivos, restrições e requisitos de qualidade semelhantes; (ii) a explicitação de pontos de variação, de modo que a arquitetura possa ser instanciada e adaptada a contextos específicos sem perder coerência; e (iii) a incorporação de padrões, estilos e boas práticas reconhecidas, reduzindo a dependência de decisões ad-hoc e favorecendo a rastreabilidade de escolhas arquiteturais ao longo do ciclo de vida do software Nakagawa et al. (2014). Dessa forma, uma AR atua como artefato de reúso em nível arquitetural, apoiando a padronização, a interoperabilidade e a evolução contínua dos sistemas derivados.

No contexto do desenvolvimento de sistemas, a finalidade central de uma AR é elevar a qualidade do produto desde as fases iniciais, orientando o desenho dos componentes e suas interações com base em atributos de qualidade previamente considerados, como segurança, manutenibilidade, escalabilidade e confiabilidade. Ao oferecer uma base comum para múltiplas soluções, a AR também reduz custo e esforço de desenvolvimento, facilita comunicação entre stakeholders e aumenta a previsibilidade do processo, pois decisões arquiteturais essenciais passam a ser documentadas e reutilizadas de forma sistemática, em vez de reconstruídas a cada novo projeto Bass, Clements e Kazman (2012), Nakagawa et al. (2014).

Uma Arquitetura de Referência (AR) eleva o nível de abstração em relação à

arquitetura de um sistema específico. Enquanto a arquitetura de software trata de um único sistema, a AR captura a essência das arquiteturas de uma classe de sistemas em um domínio específico, fornecendo um vocabulário comum, estruturas reutilizáveis e diretrizes de implementação (BASS; CLEMENTS; KAZMAN, 2012). Instâncias concretas da AR são arquiteturas de sistemas específicos, derivadas da AR por seleção, especialização e extensão de seus componentes.

A adoção de uma AR em um domínio de desenvolvimento de software traz benefícios documentados pela literatura (NAKAGAWA et al., 2014): reduz o custo e o tempo de desenvolvimento, pois equipes não precisam tomar decisões arquiteturais fundamentais a cada novo projeto; reduz o risco técnico, pois as decisões arquiteturais codificadas foram testadas e validadas em múltiplos contextos; promove a interoperabilidade entre sistemas desenvolvidos a partir da mesma AR; facilita a manutenção e a evolução; e eleva a qualidade estrutural dos sistemas resultantes.

2.3.1 Trabalhos correlatos

A literatura relacionada a este trabalho pode ser organizada em três eixos complementares: (i) estudos que discutem chatbots sob a ótica de propósito, técnicas e aplicações; (ii) trabalhos fundacionais sobre arquitetura de software e documentação arquitetural; e (iii) pesquisas que propõem modelos e processos para a construção de Arquiteturas de Referência (ARs). A seguir, são detalhados os principais trabalhos correlatos, explicitando o que cada um aborda, o que não é tratado e quais lacunas permanecem, de modo a justificar a relevância e a finalidade desta pesquisa.

Shawar e Atwell (2007)

O estudo de Shawar e Atwell (2007) apresenta uma discussão sobre chatbots enquanto artefatos de interação humano-computador, revisitando conceitos, exemplos e usos recorrentes à época. O trabalho contribui ao sintetizar características gerais de agentes conversacionais e ao situar aplicações típicas, permitindo compreender o papel do chatbot como mediador de informação e serviço.

Entretanto, o foco é predominantemente descritivo e centrado no fenômeno de

uso, sem avançar em uma proposta arquitetural sistemática. Não há uma decomposição explícita em componentes arquiteturais, nem uma preocupação metodológica com atributos de qualidade (por exemplo, escalabilidade, auditabilidade, interoperabilidade) como requisitos arquiteturais deriváveis de evidências empíricas.

A principal lacuna, portanto, é a ausência de uma ponte entre as observações sobre utilidade e características dos chatbots e a definição de uma arquitetura reutilizável, documentada em visões e orientada por requisitos não funcionais. Esta pesquisa avança justamente nesse ponto, ao formalizar componentes e interações por meio de uma AR construída com processo explícito.

Dale (2016)

Em Dale (2016), discute-se o crescimento de sistemas conversacionais, bem como aspectos práticos de adoção, limitações e expectativas de mercado. O texto é relevante por apontar fatores que impulsionam a disseminação de chatbots e por caracterizar desafios recorrentes relacionados a experiência do usuário e ao escopo de domínio. Em especial, o trabalho evidencia que chatbots frequentemente oscilam entre utilidade objetiva (tarefas bem definidas) e frustração quando a compreensão de linguagem e o gerenciamento de contexto não são robustos.

Apesar disso, o trabalho não estabelece um modelo arquitetural de referência, tampouco explicita como transformar tais desafios em requisitos arquiteturais e decisões de projeto. Questões como integração com sistemas corporativos, rastreabilidade de conversas, mecanismos de fallback, observabilidade e evolução modular são mencionadas de forma dispersa, sem culminar em uma estrutura arquitetural generalizável.

A lacuna identificada é a inexistência de um artefato arquitetural que consolide boas práticas em componentes e conectores, com documentação consistente. O presente trabalho endereça essa lacuna ao sintetizar uma AR para chatbots, conectando problemas recorrentes a RNFs e, por fim, a uma estrutura documentada em múltiplas visões.

Huang (2021)

O trabalho de Huang e CIS (2021) descreve chatbots sob a perspectiva de design, arquitetura e aplicações, oferecendo uma visão introdutória das tecnologias associadas, como PLN/NLP e aprendizado de máquina, além de cenários de uso. Seu mérito está em apresentar uma leitura ampla do ecossistema e em reforçar que chatbots modernos dependem de uma combinação de técnicas, dados e infraestrutura.

Contudo, ainda que o texto mencione arquitetura, ele não formaliza uma Arquitetura de Referência no sentido da Engenharia de Software, isto é, um blueprint reutilizável que explicite componentes, interfaces e decisões arquiteturais essenciais para uma família de sistemas Bass, Clements e Kazman (2012). Não há um processo definido para elicitar requisitos e consolidá-los em visões arquiteturais, nem um método de avaliação arquitetural. Além disso, o tratamento de qualidade tende a ser geral, sem propor mecanismos concretos para segurança, auditoria, escalabilidade e interoperabilidade.

A lacuna é a falta de sistematização: falta um processo que conecte evidências do domínio a uma arquitetura representada de modo padronizado e instanciável. Este trabalho preenche essa lacuna ao empregar o ProSA-RA para derivar requisitos e, então, sintetizar a AR com documentação em visões.

Bass et al. (2012)

A obra de Bass, Clements e Kazman (2012) constitui uma referência fundamental para arquitetura de software, ao consolidar conceitos como atributos de qualidade, táticas arquiteturais e o papel da arquitetura como instrumento de comunicação e controle de evolução. O livro oferece o arcabouço conceitual necessário para tratar requisitos não funcionais como direcionadores de decisões arquiteturais, especialmente em sistemas complexos e evolutivos.

Por se tratar de uma obra de caráter geral, o trabalho não se dedica ao domínio de chatbots e não propõe uma AR específica para sistemas conversacionais. Também não define um processo particular para projetar ARs, nem exemplifica uma arquitetura orientada ao fluxo conversacional, à gestão de contexto e à integração com múltiplas fontes de conhecimento.

A lacuna é a ausência de especialização de domínio. O presente trabalho utiliza os princípios e conceitos de Bass, Clements e Kazman (2012) como base para tratar RNFs, mas os materializa em uma AR concreta para chatbots, com componentes e interações explicitamente modelados.

Kruchten (1995)

O modelo "4+1" de Kruchten (1995) propõe uma organização para documentação arquitetural por meio de múltiplas visões, apoiadas por cenários. Sua contribuição central é tornar explícito que uma arquitetura precisa ser comunicada a diferentes stakeholders e que uma única representação é insuficiente para capturar estrutura, comportamento e implantação de sistemas.

O trabalho, entretanto, não trata do domínio de chatbots e não orienta, por si só, como derivar componentes e decisões arquiteturais a partir de evidências empíricas ou de uma revisão de literatura. Além disso, o "4+1" não é um processo de projeto arquitetural; ele é um modelo de documentação e comunicação.

A lacuna, no contexto desta monografia, é a ausência de um mecanismo para ir do domínio e seus requisitos até a arquitetura documentada. Esta pesquisa integra Kruchten (1995) como estrutura de documentação, mas utiliza um processo específico (ProSA-RA) para a análise e síntese da AR, conectando evidências e RNFs às visões arquiteturais resultantes.

Nakagawa et al. (2012): RAModel

Em Nakagawa, Oquendo e Becker (2012), os autores propõem o RAModel como um modelo de referência para caracterizar e organizar elementos de uma Arquitetura de Referência. O RAModel contribui ao oferecer um vocabulário e uma estrutura conceitual para descrever ARs, favorecendo padronização de documentação e comparabilidade entre arquiteturas de diferentes domínios.

O trabalho, contudo, não se destina a um domínio específico, e não entrega uma AR pronta para chatbots. Além disso, o RAModel, por si só, não prescreve um procedimento detalhado de construção da AR baseado em evidências, nem explicita como

conduzir a síntese arquitetural em termos de estilos, padrões e visões UML.

A lacuna é a necessidade de instanciar o RAModel no domínio de chatbots e conectá-lo a um processo operacional de elicitação e síntese. Este trabalho se apoia no RAModel como base conceitual e, a partir dele, adota um processo de construção que culmina em uma AR concreta, documentada e instanciada.

Nakagawa et al. (2014): ProSA-RA

O ProSA-RA, descrito em Nakagawa et al. (2014), define um processo para projeto, representação e avaliação de Arquiteturas de Referência, estruturando atividades de investigação de fontes de informação, análise arquitetural, síntese arquitetural e avaliação. A contribuição do processo é sua sistematização, pois explicita como coletar evidências, derivar requisitos e, então, materializar uma arquitetura representável e avaliável.

Embora o ProSA-RA seja diretamente aplicável ao objetivo desta monografia, o trabalho original não trata de chatbots especificamente e não fornece a decomposição arquitetural do domínio. Além disso, a aplicação do processo depende de decisões sobre como selecionar fontes, como operacionalizar critérios de extração e como documentar as visões resultantes de modo consistente.

A lacuna, portanto, é a ausência de uma aplicação do ProSA-RA ao domínio de chatbots com resultado arquitetural completo e instanciável. Esta monografia justifica sua relevância ao aplicar o ProSA-RA especificamente para chatbots, produzindo uma AR documentada e conectada a RNFs observados no domínio.

Síntese das lacunas e posicionamento deste trabalho

Os trabalhos voltados a chatbots (Shawar e Atwell (2007), Dale (2016), Huang e CIS (2021)) caracterizam aplicações, técnicas e desafios, mas não consolidam uma Arquitetura de Referência sistemática que articule componentes, conectores e decisões orientadas por atributos de qualidade. Por outro lado, os trabalhos de arquitetura de software e de ARs (Bass, Clements e Kazman (2012), Kruchten (1995), Nakagawa, Oquendo e Becker (2012), Nakagawa et al. (2014)) oferecem fundamentos, modelos e processos, mas não instanciam tais contribuições no domínio de chatbots.

Assim, a lacuna central que esta monografia endereça é a falta de uma AR para chatbots construída com base em evidências do domínio, derivada por um processo explícito e documentada segundo um modelo reconhecido de visões. Ao sintetizar e documentar essa AR e ao demonstrar sua instanciação em um domínio específico, neste trabalho se busca reduzir o desenvolvimento ad-hoc de chatbots e fornecer uma base arquitetural reutilizável para construção, manutenção e evolução desses sistemas.

2.4 O Processo ProSA-RA

O ProSA-RA (*Process for Software Architectures - Reference Architectures*) é um processo que sistematiza o design, a representação e a avaliação de arquiteturas de referência (NAKAGAWA et al., 2014). Foi desenvolvido na Universidade de São Paulo, no grupo de pesquisa de Engenharia de Software liderado pela Prof. Elisa Y. Nakagawa, e evoluiu ao longo de múltiplas aplicações em domínios distintos, incluindo mineração visual, teste de software, ambientes de engenharia de software, televisão digital, robótica e sistemas marinhos. A estrutura do processo é apresentada na Figura 2.3.

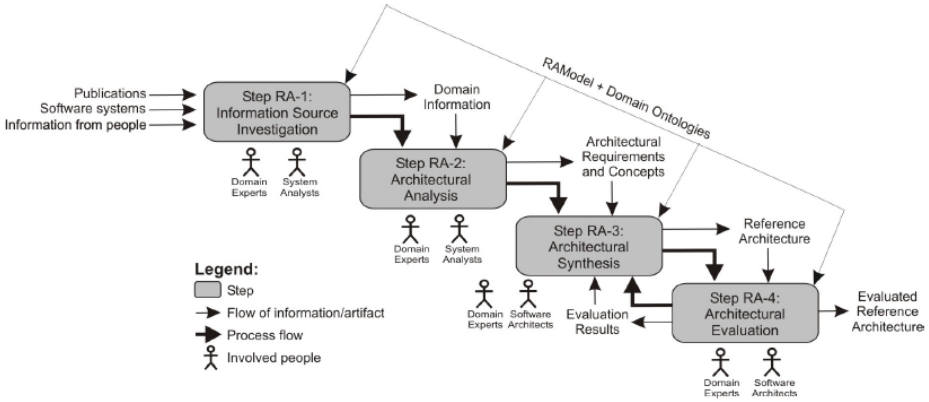

Figura 2.3: Estrutura do ProSA-RA segundo Nakagawa et al. (2014).

Fonte: Adaptado de Nakagawa et al. (2014).

A primeira etapa tem como objetivo coletar e organizar o conhecimento existente sobre o domínio alvo. O ProSA-RA identifica cinco categorias principais de fontes de informação: pessoas com conhecimento especializado no domínio; sistemas de software existentes no domínio; publicações técnicas e científicas, para as quais recomenda o uso

de Revisão Sistemática de Literatura (RSL); modelos de referência e ARs existentes em domínios correlatos; e ontologias de domínio.

A segunda etapa transforma o conhecimento coletado em requisitos arquiteturais concretos por meio de três atividades principais: identificação dos requisitos dos sistemas; identificação dos requisitos da AR; e identificação dos conceitos do domínio. Esse mapeamento é a ponte entre o mundo dos requisitos e o mundo da estrutura arquitetural.

A terceira etapa produz a descrição arquitetural da AR. Com base nos requisitos e conceitos identificados, estilos e padrões arquiteturais são selecionados, e os conceitos são organizados em componentes e conectores. A descrição arquitetural é produzida na forma de visões arquiteturais.

A quarta etapa realiza a avaliação da AR produzida mediante o framework FERA (*Framework for Evaluation of Reference Architectures*). FERA é composto por 93 questões de múltipla escolha que guiam revisores na detecção de defeitos na descrição arquitetural. Esta etapa não está contemplada no escopo deste trabalho.

Em paralelo, o RAModel (*Reference Architecture Model*) é um modelo de referência para arquiteturas de referência, proposto por Nakagawa, Oquendo e Becker (2012). A Figura 2.4 ilustra sua estrutura. Ele fornece um catálogo de elementos organizados em quatro grupos: Domínio, Aplicação, Infraestrutura e Elementos Transversais. O RAModel cumpre um papel duplo no contexto do ProSA-RA: na etapa de investigação, guia a identificação de fontes de informação; nas etapas de análise e síntese, serve como checklist para verificar se todos os elementos relevantes estão sendo considerados.

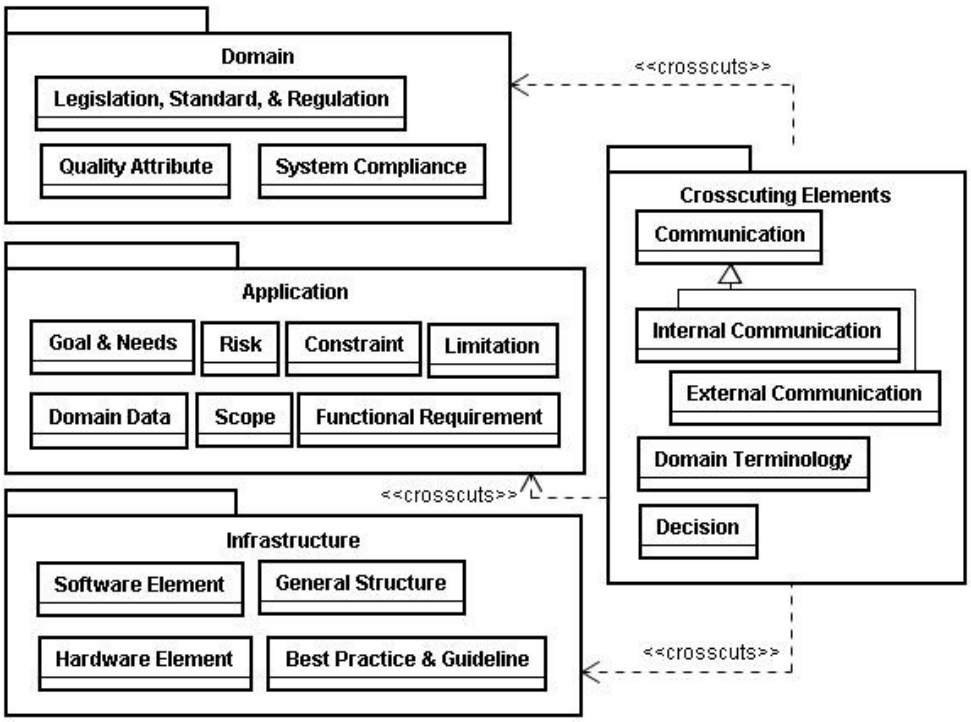

Figura 2.4: RAModel: modelo de referência para arquiteturas de referência.

Fonte: Adaptado de Nakagawa, Oquendo e Becker (2012).

3 Revisão Sistemática de Literatura

A Revisão Sistemática de Literatura (RSL) é uma metodologia de pesquisa que visa identificar, avaliar e sintetizar o conjunto de estudos relevantes sobre uma questão de pesquisa específica, por meio de um protocolo explícito, rigoroso e reproduzível (KITCHENHAM, 2004). Ao contrário das revisões narrativas tradicionais, que dependem do julgamento subjetivo do revisor na seleção e análise das fontes, a RSL impõe transparência metodológica: qualquer pesquisador que execute o mesmo protocolo deve chegar a um conjunto similar de estudos e conclusões.

A adoção da RSL neste trabalho é motivada por dois fatores complementares. O primeiro é a recomendação explícita do ProSA-RA de utilizar revisão sistemática como mecanismo para levantamento de conhecimento publicado sobre o domínio alvo (NAKAGAWA et al., 2014). O segundo é a necessidade de construir a fundamentação da Arquitetura de Referência sobre uma base auditável: cada componente proposto deve ser rastreável a evidências da literatura, não apenas ao julgamento do autor.

Para fins de reporte, adotou-se o protocolo **PRISMA 2020** (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) (PAGE et al., 2021), que estabelece diretrizes para condução e relato de revisões sistemáticas, amplamente adotado em pesquisas de engenharia de software (KITCHENHAM; CHARTERS, 2007).

A questão de pesquisa principal que guia a RSL é:

> **QP:** Quais são as características arquiteturais, os componentes essenciais e os requisitos de qualidade recorrentes em chatbots modernos, conforme reportados na literatura técnica e científica entre 2010 e 2024?

Três questões secundárias complementam a questão principal:

- **QS1:** Quais arquiteturas de referência ou modelos arquiteturais para chatbots foram propostos na literatura?
- **QS2:** Quais atributos de qualidade são mais frequentemente identificados como críticos para chatbots?

- **QS3:** Quais tecnologias e padrões são mais amplamente adotados no desenvolvimento de chatbots em diferentes domínios de aplicação?

3.1 Protocolo de Revisão

3.1.1 Fontes de Busca

As bases de dados selecionadas para a busca foram IEEE Xplore, ACM Digital Library, Springer Link e Google Scholar. A escolha é justificada pela cobertura abrangente das principais conferências e periódicos de ciência da computação e engenharia de software, incluindo publicações centrais ao tema desta pesquisa, como IEEE Software, ACM Computing Surveys, Empirical Software Engineering, e os anais das conferências ICSE, WICSA e ECSA. O período de cobertura foi definido entre janeiro de 2010 e dezembro de 2024, delimitação que coincide com o surgimento dos primeiros assistentes pessoais baseados em aprendizado de máquina e alcança o estado da arte em sistemas baseados em modelos de linguagem de grande escala.

3.1.2 String de Busca

A string de busca foi construída iterativamente, por meio de três ciclos de refinamento, para equilibrar abrangência (evitar omissão de estudos relevantes) e precisão (evitar excesso de resultados irrelevantes). A versão final utilizada foi:

```
("chatbot"OR "conversational agent"OR "dialogue system"OR "virtual
assistant"OR "conversational AI") AND ("architecture"OR "reference
architecture"OR "software architecture"OR "system design"OR "architectural
pattern") AND ("quality attributes"OR "non-functional requirements"OR
"scalability"OR "reliability"OR "security"OR "interoperability"OR
"maintainability")
```

Adaptações menores da string foram aplicadas em cada base para acomodar diferenças de sintaxe de busca e vocabulário controlado de indexação.

3.1.3 Critérios de Inclusão e Exclusão

Os critérios de inclusão foram definidos como:

1. Estudos publicados entre 2010 e 2024, em inglês ou português;
2. Estudos que abordem aspectos arquiteturais de chatbots ou sistemas conversacionais, incluindo componentes, integrações e decisões de design;
3. Estudos que discutam requisitos de qualidade para chatbots, sejam eles requisitos não funcionais gerais ou específicos de domínio;
4. Estudos que apresentem implementações documentadas de chatbots em contextos de aplicação real ou experimental, com descrição suficiente para análise arquitetural;
5. Estudos publicados em periódicos revisados por pares, anais de conferências ou como relatórios técnicos de instituições reconhecidas.

Os critérios de exclusão foram definidos como:

1. Estudos puramente linguísticos, focados em aspectos fonológicos, sintáticos ou semânticos de linguagem natural, sem foco em engenharia de software;
2. Estudos sobre reconhecimento de fala isolado, sem discussão de arquitetura de sistema conversacional;
3. Artigos de opinião, posicionamento ou editoriais sem base empírica ou metodológica;
4. Estudos cujo texto completo não estivesse disponível nas plataformas acessadas;
5. Estudos com foco exclusivo em aspectos de negócio ou experiência do usuário, sem contribuição técnica em arquitetura ou qualidade de software.

3.1.4 Processo de Seleção dos Estudos

O processo de seleção foi conduzido em três rodadas sequenciais, descritas a seguir.

Rodada 1 (triagem por título e resumo): os títulos e resumos de todos os registros retornados pela busca foram avaliados individualmente com base nos critérios de inclusão e exclusão. Registros que não atendessem a pelo menos um critério de inclusão, ou que satisfizessem qualquer critério de exclusão, foram descartados nesta fase. Em caso

de dúvida sobre a relevância com base apenas no título e no resumo, o estudo foi mantido para avaliação na rodada seguinte.

Rodada 2 (avaliação do texto completo): os estudos que passaram pela primeira rodada tiveram seus textos completos lidos e avaliados em maior profundidade. Cada estudo foi verificado quanto à sua contribuição efetiva para as questões de pesquisa definidas. Estudos excluídos nesta fase tiveram a razão da exclusão registrada.

Rodada 3 (*backward snowballing*): a lista de referências dos estudos incluídos na rodada anterior foi verificada para identificar estudos relevantes não capturados pela busca inicial. Os estudos identificados foram submetidos ao mesmo processo de avaliação das rodadas anteriores.

3.1.5 Extração de Dados

Para cada estudo incluído, os seguintes elementos foram extraídos sistematicamente e registrados em uma planilha estruturada: identificação (título, autores, ano, veículo de publicação); tipo de estudo (artigo de pesquisa, relato de experiência, revisão, relatório técnico); domínio do chatbot reportado; componentes arquiteturais identificados ou discutidos; atributos de qualidade abordados; tecnologias e frameworks mencionados; e síntese da contribuição principal.

A extração foi realizada pelo autor da monografia, com revisão por amostragem pelo orientador para garantir consistência. Discrepâncias foram resolvidas por consenso.

3.1.6 Avaliação da Qualidade dos Estudos

A qualidade metodológica dos estudos incluídos foi avaliada por um conjunto de quatro critérios, baseados em adaptações do checklist de avaliação de qualidade para estudos de engenharia de software proposto por Kitchenham e Charters (2007):

1. **Clareza do contexto:** o estudo descreve claramente o domínio de aplicação e o contexto de desenvolvimento do chatbot?

2. **Descrição arquitetural:** o estudo descreve componentes, interfaces ou decisões arquiteturais de forma suficientemente detalhada para extração?

3. **Discussão de qualidade:** o estudo aborda atributos de qualidade de forma explícita, com critérios ou métricas observáveis?

4. **Validade dos resultados:** os resultados são apoiados por evidências empíricas, implementações reais ou validações formais?

Cada critério foi pontuado como totalmente satisfeito (2), parcialmente satisfeito (1) ou não satisfeito (0), resultando em uma pontuação máxima de 8 por estudo. Estudos com pontuação inferior a 3 foram excluídos, pois contribuiriam com evidências de qualidade insuficiente para sustentar decisões arquiteturais. Nenhum estudo que atingiu a fase de elegibilidade apresentou pontuação inferior a esse limiar.

3.2 Resultados da Revisão

O processo de busca e seleção está representado no fluxograma PRISMA apresentado na Figura 3.1.

A busca inicial retornou 847 registros nas quatro bases consultadas: IEEE Xplore (312), ACM Digital Library (198), Springer Link (221) e Google Scholar (116). Após remoção de 235 duplicatas, 612 estudos únicos foram submetidos à primeira rodada de triagem por título e resumo. Desses, 523 foram excluídos nesta fase: 301 por estarem fora do escopo temático (foco em PLN puro, sem discussão arquitetural), 89 por período de publicação fora do corte, 45 por idioma inadequado, e 88 por tipo de publicação inelegível (apresentações de workshops sem revisão formal, por exemplo).

Os 89 estudos remanescentes foram avaliados por texto completo. Desses, 55 foram excluídos com justificativa registrada: 23 por ausência de foco em engenharia de software, 18 por não apresentar abordagem arquitetural identificável, e 14 por indisponibilidade do texto completo nas plataformas acessadas. Ao final desta fase, 34 estudos foram incluídos.

O processo de *backward snowballing* sobre as referências dos 34 estudos incluídos identificou 18 candidatos adicionais, dos quais 11 atenderam aos critérios de elegibilidade e foram incluídos. O total final de estudos na síntese é de **45 estudos primários**.

Os 45 estudos incluídos cobrem o período de 2011 a 2024, com concentração

crescente a partir de 2017, coincidente com o impacto da arquitetura *Transformer* no campo. Do total, 28 são artigos publicados em conferências ou periódicos revisados por pares, 9 são relatórios técnicos de instituições acadêmicas ou empresas de tecnologia, e 8 foram adicionados pelo processo de snowballing, sendo 6 publicações anteriores a 2017 de relevância histórica para a área.

Em termos de domínio de aplicação, os estudos cobrem: atendimento ao cliente (18 estudos), saúde (9), educação (7), finanças (5), governo (3) e entretenimento (3). A diversidade de domínios é necessária para garantir que a AR resultante não seja enviesada pelas particularidades de um único contexto.

A análise dos estudos incluídos revelou que a grande maioria dos sistemas conversacionais documentados na literatura apresenta uma estrutura arquitetural convergente, variando principalmente na implementação dos componentes, não em sua organização fundamental. O *pipeline* de processamento (recepção, compreensão, gerenciamento de diálogo, geração e entrega) aparece, sob variações de nomenclatura, em 38 dos 45 estudos analisados.

Os trabalhos de Cahn (2017) e Huang e CIS (2021) oferecem as descrições arquiteturais mais detalhadas, identificando explicitamente os componentes de Compreensão de Linguagem Natural (NLU), Gerenciamento de Diálogo (DM) e Geração de Linguagem Natural (NLG) como os três pilares estruturais. Essa organização tripartite é coerente com os sistemas comerciais mais amplamente documentados, como Rasa Framework e Microsoft Bot Framework.

Nenhum dos 45 estudos propõe uma Arquitetura de Referência formalmente especificada por meio de um processo sistemático. A lacuna confirmada por este resultado constitui a principal motivação e contribuição do presente trabalho.

3.2.1 Requisitos de Qualidade e Tecnologias Recorrentes

A análise dos estudos permitiu identificar e quantificar os requisitos de qualidade mais frequentemente citados como críticos para chatbots, conforme apresentado na Tabela 3.1.

Escalabilidade lidera com 41 ocorrências em 45 estudos, refletindo o caráter ine-

Tabela 3.1: Frequência de atributos de qualidade nos estudos incluídos na RSL (n = 45).

Atributo de Qualidade	**Freq.**	**Justificativa Recorrente**
Escalabilidade	41	Variabilidade inerente da carga em produção
Segurança e Privacidade	39	Conformidade com GDPR, LGPD, HIPAA
Confiabilidade	37	Disponibilidade e consistência das respostas
Interoperabilidade	35	Integração com sistemas legados e APIs
Manutenibilidade	31	Atualização de modelos e bases de conhecimento
Desempenho e Latência	29	Impacto na experiência do usuário
Portabilidade	24	Migração entre ambientes de nuvem
Extensibilidade	21	Adição de novos domínios e canais
Auditabilidade	18	Rastreabilidade em domínios regulados
Acessibilidade	14	Conformidade WCAG em contextos públicos
Explicabilidade	11	Transparência em domínios de alto risco
Personalização	9	Adaptação ao perfil do usuário

Fonte: Autoria própria, com base nos estudos incluídos na RSL.

rentemente variável da carga em sistemas conversacionais: uma plataforma de atendimento pode passar de centenas para milhões de interações simultâneas em eventos pontuais. Segurança e privacidade aparecem em 39 estudos, com ênfase crescente após a vigência do GDPR (2018) e da LGPD (2020), que impuseram obrigações legais explícitas a sistemas que processam dados pessoais em diálogo, como os chatbots necessariamente fazem. Explicabilidade, o atributo de menor frequência entre os relevantes, concentra-se nos estudos de domínio de saúde e governo, onde a rastreabilidade das decisões do sistema é mandatória.

Já em termos de tecnologias, os estudos evidenciam uma convergência em torno de três famílias de ferramentas. Os *frameworks* de desenvolvimento de chatbots (Rasa, Microsoft Bot Framework, Dialogflow, Watson Assistant) aparecem em 27 estudos como base para implementações, todos organizando o processamento na separação NLU-DM. Os modelos de linguagem pré-treinados (BERT, GPT e suas variantes) aparecem em 31 estudos publicados a partir de 2019, com crescimento acentuado pós-2022. Arquiteturas de

Retrieval-Augmented Generation (RAG) (LEWIS et al., 2020) aparecem em 12 estudos publicados entre 2021 e 2024, como resposta ao problema de alucinação em sistemas generativos puros.

Em termos de padrões arquiteturais, o padrão de *Circuit Breaker* para resiliência de integração externa aparece em 19 estudos; o padrão de filas de mensagens para desacoplamento de carga aparece em 23; e o padrão de *Observer* para monitoramento não intrusivo aparece em 16. Esses padrões foram diretamente incorporados à AR proposta.

3.2.2 Lacunas e Limitações Identificadas

A RSL identificou quatro lacunas que direcionam a contribuição deste trabalho.

A primeira e mais significativa é a **ausência de ARs formalmente especificadas** para o domínio de chatbots. Os trabalhos mais próximos dessa contribuição são descritivos e informais, sem o rigor de um processo sistemático como o ProSA-RA. Essa lacuna é o ponto de partida direto deste trabalho.

A segunda é a **fragmentação de atributos de qualidade**: cada estudo tende a focar em um subconjunto dos atributos relevantes, sem uma visão integrada que mostre como os diferentes atributos se relacionam e como os *trade-offs* entre eles devem ser gerenciados na fase de projeto.

A terceira é a **ausência de instâncias de domínio** derivadas de um modelo arquitetural comum. Os estudos de aplicação descrevem arquiteturas específicas de domínio sem referência a um modelo base, tornando impossível identificar o que é estruturalmente necessário para qualquer chatbot e o que é específico de cada contexto.

A quarta é a **cobertura limitada de domínios emergentes**, com escassez de estudos sobre chatbots jurídicos, agentes em realidade aumentada e sistemas conversacionais multiagente, que introduzem requisitos arquiteturais ainda não sistematizados.

Essas lacunas, em conjunto, confirmam tanto a relevância da contribuição proposta quanto as limitações que a AR resultante precisará reconhecer explicitamente. Seguindo as diretrizes do protocolo PRISMA, são relatadas as principais ameaças à validade desta RSL.

Viés de publicação: estudos com resultados positivos têm maior probabilidade

de serem publicados e indexados nas bases consultadas. Chatbots com características arquiteturais problemáticas podem estar sub-representados na literatura, levando a uma visão otimista dos padrões encontrados.

Viés de seleção: a triagem por título e resumo foi realizada por um único revisor, o que pode ter introduzido erros de julgamento. Para mitigar esse risco, critérios de inclusão e exclusão foram definidos antes da execução da busca e aplicados de forma consistente.

Limitação de cobertura: quatro bases de dados cobrem a maior parte da literatura relevante, mas trabalhos publicados exclusivamente em repositórios regionais ou anais não indexados podem ter sido omitidos. O processo de backward snowballing mitiga parcialmente essa limitação, mas não a elimina.

Evolução acelerada do domínio: o campo de chatbots baseados em LLM evolui em ritmo superior ao ciclo de publicação científica. Desenvolvimentos técnicos relevantes de 2023-2024 podem estar documentados apenas em literatura cinza (relatórios técnicos, blogs de pesquisa, pré-impressões) não capturada pela estratégia de busca adotada.

Essas ameaças não invalidam os resultados da RSL, mas estabelecem os limites dentro dos quais as conclusões devem ser interpretadas.

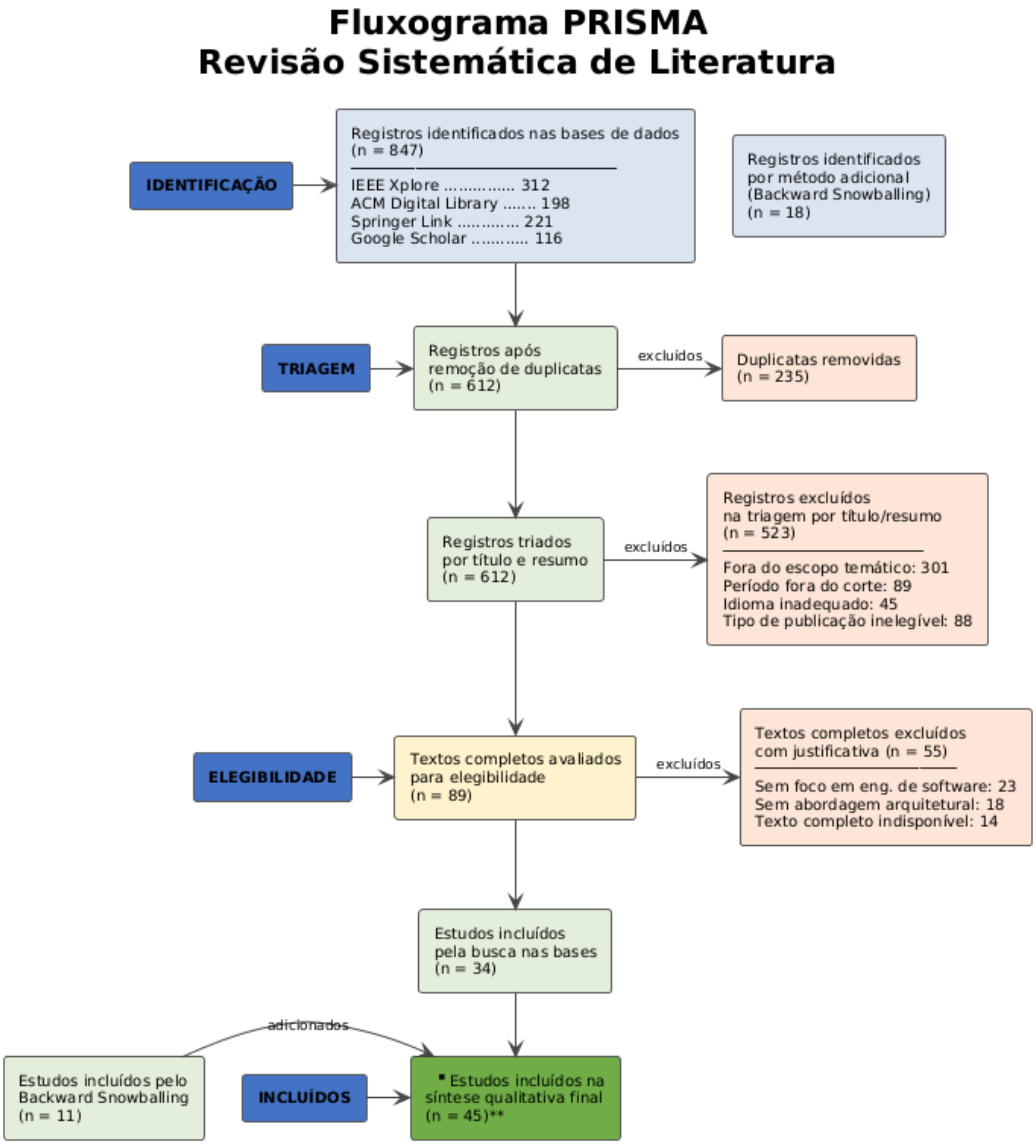

Figura 3.1: Fluxograma PRISMA do processo de seleção de estudos da Revisão Sistemática de Literatura.

Fonte: Autoria própria, adaptado de Page et al. (2021).

4 Material e Métodos

Neste trabalho é adotado o ProSA-RA como método principal para o desenvolvimento da AR. A escolha é motivada por três fatores: o ProSA-RA é um processo sistemático e documentado, com histórico de aplicação bem-sucedida em múltiplos domínios; ele é suficientemente flexível para acomodar as particularidades do domínio de chatbots; e a literatura de origem é acessível e detalhada o suficiente para guiar a aplicação sem ambiguidades.

O presente trabalho cobre as três primeiras das quatro etapas do ProSA-RA. A etapa de avaliação arquitetural (RA-4) está fora do escopo e é indicada como trabalho futuro.

A etapa de investigação de fontes de informação (RA-1) foi conduzida por meio de três atividades principais: a RSL descrita no Capítulo 3; a análise direta de sete chatbots representativos de domínios distintos, cobrindo saúde (Ada Health), varejo (Amazon Alexa), bancário (Bradesco BIA), educacional (Duolingo), atendimento ao cliente (Zendesk Answer Bot), entretenimento (Character.AI) e governo (Portal Gov.br); e a consulta a documentações técnicas públicas de frameworks de desenvolvimento de chatbots (Rasa, Microsoft Bot Framework, Google Dialogflow, IBM Watson Assistant).

A etapa de análise arquitetural (RA-2) foi conduzida a partir dos dados coletados, com organização dos RNFs em tabela estruturada e mapeamento dos conceitos do domínio a partir dos componentes funcionais identificados.

A etapa de síntese arquitetural (RA-3) produziu as cinco visões arquiteturais do modelo “4+1”, documentadas por meio de diagramas UML.

A documentação arquitetural foi produzida utilizando ferramentas de modelagem UML, com foco na precisão semântica dos diagramas. Os diagramas foram construídos iterativamente, com revisões sucessivas para garantir consistência entre visões.

4.1 RA-1: Investigação das Fontes de Informação

A seleção dos chatbots para análise seguiu o critério de maximizar a cobertura de domínios distintos, garantindo que a AR resultante seja suficientemente abrangente. A Tabela 4.1 apresenta os sete sistemas analisados com suas principais características.

Tabela 4.1: Chatbots analisados por domínio e principais características.

Chatbot	**Domínio**	**Características Arquiteturais**	**RNFs Prioritários**
Ada Health	Saúde	Motor de inferência clínica, base de conhecimento médico estruturado	Confiabilidade, Segurança, Explicabilidade
Amazon Alexa	Varejo / Assistência	Modelo de Skills, reconhecimento de voz como serviço independente	Escalabilidade, Extensibilidade, Baixa latência
Bradesco BIA	Bancário	Integração com core bancário legado, autenticação multifatorial	Segurança, Confiabilidade, Auditabilidade
Duolingo	Educacional	Modelagem do usuário, feedback pedagógico, gamificação integrada	Personalização, Manutenibilidade
Zendesk Answer Bot	Atendimento	Modelo de triagem escalável, integração com CRM e ticketing	Escalabilidade, Rastreabilidade
Character.AI	Entretenimento	Gestão de memória de longo prazo, consistência de persona	Criatividade, Coerência de persona
Portal Gov.br	Governo	Acessibilidade WCAG, multilinguismo, Lei de Acesso à Informação	Acessibilidade, Interoperabilidade

Fonte: Autoria própria.

O Ada Health é um assistente de saúde que guia usuários por meio de questionários clínicos adaptativos para produzir avaliações de risco de condições médicas. Seu diferencial arquitetural é o motor de inferência clínica, construído sobre uma base de conhecimento médico estruturado e validado por médicos. Do ponto de vista arquitetural, o Ada Health evidencia a importância crítica de três atributos: confiabilidade, pois erros em avaliações de saúde têm consequências potencialmente severas; segurança e privacidade, dado o caráter altamente sensível dos dados; e explicabilidade, pois os usuários precisam compreender a base das avaliações para confiar no sistema.

A Alexa opera por meio de um modelo de Skills, onde cada skill é uma aplicação

independente invocada em resposta a uma intenção identificada. As características arquiteturais mais relevantes são: a separação clara entre a plataforma de compreensão de linguagem e as habilidades de domínio, que permite extensibilidade sem modificação do núcleo; o reconhecimento de voz como serviço separado do motor de diálogo; e a arquitetura orientada a eventos.

A BIA (Bradesco Inteligência Artificial) evidencia a necessidade de um componente de integração robusta com sistemas legados (core bancário), tipicamente sistemas de décadas de idade com APIs limitadas. A segurança assume papel absolutamente central, com autenticação multifatorial, criptografia de ponta a ponta, auditoria de transações e conformidade regulatória com o Banco Central do Brasil.

As características arquiteturais relevantes do Duolingo incluem: adaptação do nível de dificuldade com base no desempenho do usuário, exigindo um componente de modelagem do usuário; feedback imediato e pedagogicamente estruturado; e gamificação integrada, que requer coordenação entre o motor de diálogo e o sistema de recompensas.

O Answer Bot evidencia o padrão de triagem escalável, com capacidade de escalamento gracioso para atendimento humano quando o chatbot não consegue resolver a demanda, integração com sistemas de ticketing e CRM, e rastreabilidade completa das interações.

O Character.AI evidencia a importância de um componente de gestão de memória de longo prazo e de consistência de persona, arquiteturalmente muito diferente dos componentes de base de conhecimento dos sistemas de domínio fechado.

O chatbot do Portal Gov.br impõe requisitos de acessibilidade (conformidade com WCAG), multilinguismo, conformidade com a Lei de Acesso à Informação e capacidade de lidar com vocabulário variado, dado o espectro amplo do público-alvo.

Em paralelo, a análise dos frameworks Rasa, Microsoft Bot Framework, Dialogflow e Watson Assistant revelou convergências arquiteturais significativas. Todos os quatro organizam o processamento em dois subsistemas claramente separados: o módulo de NLU, responsável por classificar intenções e extrair entidades, e o motor de diálogo (DM), responsável por gerenciar o estado da conversa e decidir a próxima ação. Essa separação permite que os dois subsistemas evoluam independentemente e que diferentes

modelos de NLU sejam trocados sem afetar a lógica de diálogo.

4.2 RA-2: Análise Arquitetural

A partir da análise dos chatbots selecionados e dos frameworks investigados, foram identificados os requisitos de sistemas que servirão de matéria-prima para a derivação dos requisitos arquiteturais.

Os principais requisitos funcionais identificados incluem: receber e processar entradas em linguagem natural (texto e/ou voz); identificar a intenção comunicativa do usuário; extrair entidades relevantes da entrada; manter e gerenciar o estado da conversa ao longo de múltiplas trocas; recuperar informações relevantes de bases de conhecimento internas ou externas; gerar respostas em linguagem natural coerentes com o contexto; integrar-se com sistemas externos via APIs; escalar para atendimento humano quando necessário; e registrar interações para fins de auditoria e melhoria contínua.

Os requisitos não funcionais foram derivados da análise dos chatbots e da RSL, organizados em categorias conforme apresentado na Tabela 4.2.

Tabela 4.2: Requisitos não funcionais da Arquitetura de Referência.

Categoria	**Requisito**	**Descrição**	**Domínios Prioritários**
Desempenho	Tempo de resposta	O sistema deve responder em menos de 2 segundos em condições normais de carga	Todos
Escalabilidade	Escalabilidade horizontal	O sistema deve suportar aumento de carga mediante adição de instâncias sem modificações arquiteturais	Todos

Categoria	**Requisito**	**Descrição**	**Domínios Prioritários**
Segurança	Confidencialidade	Dados dos usuários devem ser criptografados em trânsito e em repouso	Saúde, Bancário, Gov
Segurança	Autenticação	O sistema deve suportar mecanismos de autenticação robustos	Bancário, Gov
Confiabilidade	Disponibilidade	O sistema deve apresentar disponibilidade mínima de 99,5%	Todos
Confiabilidade	Tolerância a falhas	Falhas em componentes individuais não devem derrubar o sistema	Todos
Interoperabilidade	Integração via APIs	O sistema deve expor e consumir APIs padronizadas	Todos
Manutenibilidade	Modularidade	Componentes devem ser substituíveis sem impacto em outros módulos	Todos
Portabilidade	Independência de plataforma	O sistema deve poder ser implantado em diferentes ambientes de nuvem	Todos
Acessibilidade	Suporte multilíngue	O sistema deve suportar múltiplos idiomas de forma configurável	Educacional, Gov
Auditabilidade	Rastreabilidade	Todas as interações devem ser registradas e rastreáveis	Bancário, Gov, Saúde

Categoria	**Requisito**	**Descrição**	**Domínios Prioritários**
Extensibilidade	Adição de domínios	Novos domínios de conhecimento devem poder ser adicionados sem modificação do núcleo	Todos

Fonte: Autoria própria.

O mapeamento dos requisitos arquiteturais para conceitos do domínio produziu o vocabulário fundamental da AR. Os conceitos identificados são: **Interface de Entrada e Saída**, ponto de contato com o mundo externo; **Módulo de NLU**, responsável pelo processamento semântico da entrada; **Motor de Diálogo (DM)**, núcleo decisório do sistema; **Gerenciador de Contexto**, que mantém o contexto da conversa ao longo de múltiplas trocas; **Base de Conhecimento**, repositório das informações de resposta; **Módulo de NLG**, que transforma a resposta estruturada em texto natural; **Módulo de Integração Externa**, que gerencia a comunicação com sistemas de terceiros; **Módulo de Autenticação e Segurança**, que gerencia identidade, acesso e criptografia; e **Módulo de Monitoramento e Logs**, que coleta métricas e registra interações. Esse conjunto constitui o vocabulário da AR e é o insumo direto para a síntese.

4.3 RA-3: Síntese Arquitetural

A síntese arquitetural deve ser guiada pelos RNFs elicitados e pelos estilos arquiteturais que melhor os endereçam. Para a AR proposta, três estilos foram adotados de forma combinada.

O primeiro é o estilo em **camadas**, adotado para organizar a separação de responsabilidades entre a interface de usuário, a lógica de processamento conversacional e a infraestrutura de persistência e integração. Esse estilo é justificado pela necessidade de manutenibilidade e portabilidade: quando cada camada se comunica apenas com a camada imediatamente abaixo, modificações em uma não propagam efeitos colaterais nas demais.

O segundo é o estilo de **pipeline**, adotado para organizar o fluxo de processamento de uma mensagem do usuário: recepção, normalização, análise semântica, gerenciamento de diálogo, recuperação de conhecimento e geração de resposta. Esse estilo é justificado pela clareza do fluxo e pela facilidade de substituir ou reconfigurar estágios individuais, sendo o padrão adotado pelos principais frameworks comerciais.

O terceiro é o estilo de **publicação-assinatura**, adotado para a integração com sistemas externos e para o desacoplamento entre o motor de diálogo e os módulos de monitoramento. Esse estilo permite que novos integradores sejam adicionados sem modificar os componentes existentes.

4.3.1 Visão Lógica

A Visão Lógica tem como finalidade expor a funcionalidade do sistema por meio das interações entre seus elementos constituintes, tornando explícito como uma entrada do usuário se transforma em uma resposta. Para esse propósito, o Diagrama de Sequência da UML oferece a representação mais adequada porque preserva a dimensão temporal das trocas de mensagens e torna visíveis as dependências entre componentes no nível da execução, diferentemente do Diagrama de Classes, que descreveria estrutura estática. A Figura 4.1 apresenta o diagrama resultante.

A decisão mais importante da Visão Lógica não é o fluxo em si, mas a quebra de sincronismo entre recepção e processamento. Sem a fila, a Interface precisaria aguardar a conclusão do pipeline completo (NLU + DM + recuperação de conhecimento + NLG) antes de liberar o canal de comunicação. Esse acoplamento temporal torna o sistema frágil sob carga: um pico de requisições causa degradação imediata na experiência do usuário porque cada nova mensagem disputa o mesmo conjunto de recursos que está processando as mensagens anteriores.

A fila desacopla esses dois ritmos. A Interface opera com latência próxima de zero (apenas enfileira a mensagem e retorna confirmação de recebimento), enquanto o Motor de Diálogo consome a fila no ritmo que a infraestrutura permite. Em cenários de alta demanda, múltiplas instâncias do Motor de Diálogo podem consumir a mesma fila em paralelo sem que a Interface precise conhecer essa topologia, satisfazendo simultaneamente

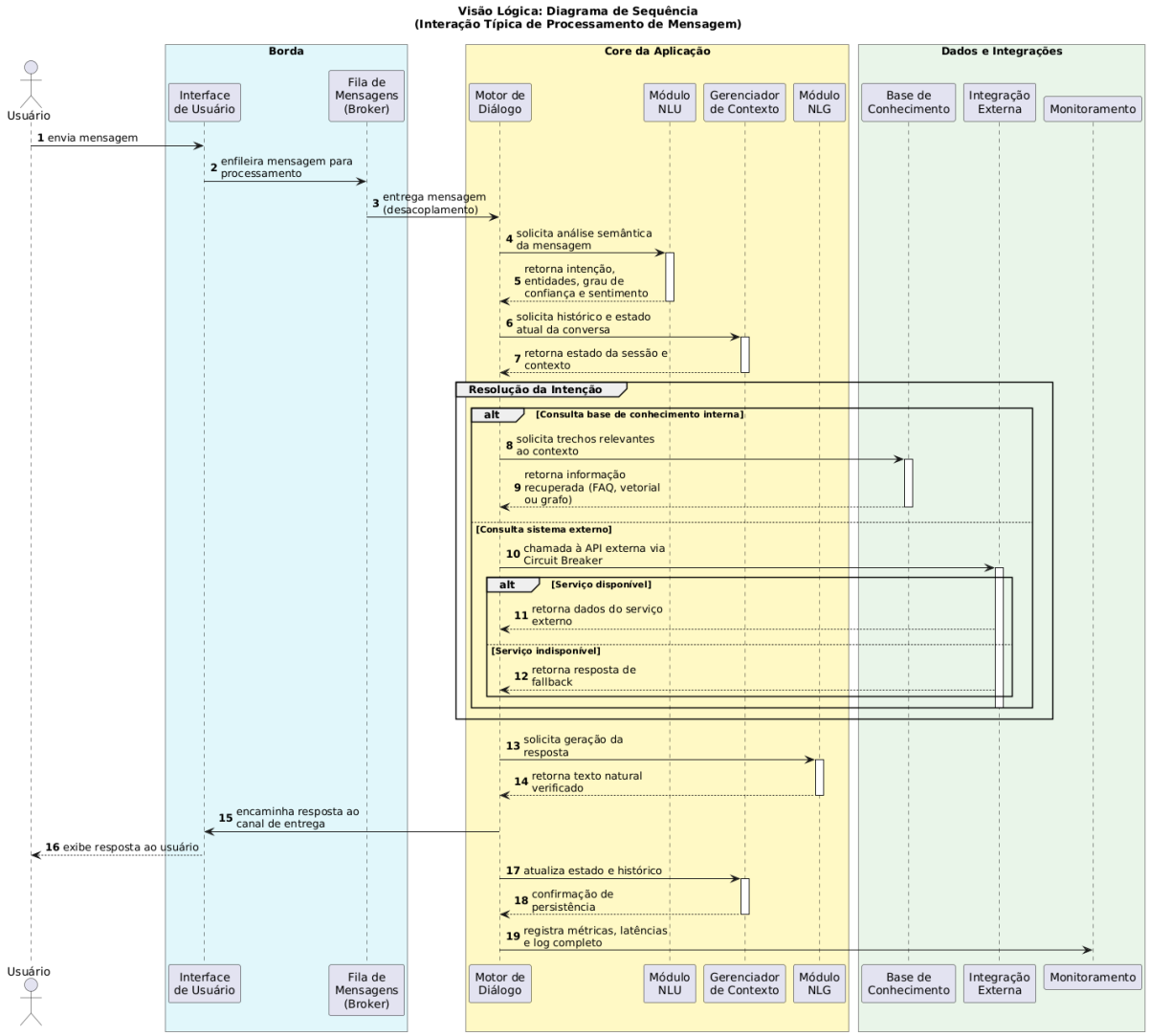

Figura 4.1: Visão Lógica: Diagrama de Sequência representando o fluxo completo de processamento de uma mensagem.

Fonte: Autoria própria.

os requisitos de escalabilidade horizontal e de disponibilidade.

A alternativa considerada e descartada foi o modelo de requisição-resposta síncrona com *timeout* configurável. Embora mais simples de implementar, esse modelo impõe um teto fixo de *throughput* determinado pela capacidade de uma única instância do Motor de Diálogo, e falha de forma abrupta (*timeout* visível ao usuário) quando esse teto é atingido. A fila falha de forma gradual (aumento de latência, não de erro), o que é preferível do ponto de vista da experiência do usuário e da observabilidade operacional.

O fluxo representado na Figura 4.1 percorre oito etapas funcionalmente distintas, cada uma com implicações de qualidade que merecem detalhamento. Na etapa de recepção

e enfileiramento, a Interface recebe a entrada do usuário (texto ou voz, dependendo do canal), aplica normalização superficial (remoção de caracteres de controle e truncamento de mensagens excessivamente longas para prevenir ataques de *prompt injection*) e publica a mensagem na fila com metadados de contexto: identificador de sessão, canal de origem, *timestamp* e identificador do usuário autenticado. O identificador de sessão é o mecanismo que permite ao Motor de Diálogo recuperar o estado de conversas anteriores do mesmo usuário sem precisar transmiti-lo integralmente na mensagem.

Na etapa de consumo e despacho para NLU, o Motor de Diálogo retira a mensagem da fila e a encaminha ao Módulo de NLU. Essa chamada é feita com o identificador de sessão, mas não com o histórico completo da conversa: o NLU opera sobre a mensagem atual isolada, pois sua função é classificar a intenção da entrada presente, não interpretar sequências conversacionais. Isso é uma decisão deliberada de separação de responsabilidades, uma vez que a interpretação de contexto conversacional é responsabilidade do Motor de Diálogo.

Na etapa de análise semântica, o Módulo de NLU retorna uma estrutura de dados padronizada contendo quatro campos: a intenção classificada (por exemplo, *consulta_conceitual*), as entidades extraídas (por exemplo, {"conceito": "árvore AVL", "comparação": "árvore binária"}), o grau de confiança da classificação (valor entre 0 e 1) e o sentimento detectado (positivo, neutro ou negativo, com intensidade). O retorno em estrutura padronizada é fundamental: o Motor de Diálogo não precisa conhecer o mecanismo interno do NLU para interpretar seu resultado, e essa interface estável é o que torna o Componente de NLU substituível sem modificação do núcleo da AR.

Na etapa de consulta de contexto, o Motor de Diálogo consulta o Gerenciador de Contexto com o identificador de sessão. O Gerenciador retorna o estado atual da conversa: intenções classificadas nas interações anteriores, entidades já confirmadas pelo usuário, ações já executadas e o ponto do fluxo de diálogo em que a conversa se encontra. Essa consulta é o que torna o chatbot ciente de que o usuário está no meio de uma tarefa multietapa (por exemplo, um processo de agendamento que requer múltiplas informações) em vez de tratar cada mensagem como isolada.

Na etapa de seleção de ação, o Motor de Diálogo aplica a política de decisão: dado

o estado da conversa e a intenção recém-classificada, qual é a ação mais adequada? As ações possíveis formam um conjunto finito, composto por responder diretamente com base no conhecimento interno, consultar a Base de Conhecimento, invocar um sistema externo via Módulo de Integração, solicitar clarificação ao usuário (quando o grau de confiança do NLU está abaixo do limiar configurado), ou escalar para atendimento humano. A política pode ser implementada como máquina de estados finita, como modelo de aprendizado por reforço, ou como regras enriquecidas por um LLM. A AR não prescreve a implementação, apenas exige que ela seja encapsulada dentro do Motor de Diálogo e exposta por uma interface uniforme.

Na etapa de recuperação de informação, quando a ação selecionada requer dados externos, o Motor de Diálogo formula uma consulta estruturada para a Base de Conhecimento ou para o Módulo de Integração Externa. A consulta estruturada (e não a mensagem bruta do usuário) é o que garante precisão na recuperação: em vez de buscar por "quando usar AVL em vez de ABB", o sistema busca por {"conceitos": ["AVL", "ABB"], "relação": "comparativa", "aspecto": "aplicabilidade"}. Essa estruturação é responsabilidade do Motor de Diálogo e é o resultado direto da análise semântica feita pelo NLU.

Na etapa de geração de resposta, o Módulo de NLG recebe a ação selecionada e o conteúdo recuperado e produz a resposta em linguagem natural. Um subcomponente de pós-processamento verifica a resposta quanto a comprimento, adequação de tom e ausência de conteúdo potencialmente inadequado antes de liberá-la para entrega. Em implementações baseadas em LLM, esse pós-processamento pode incluir verificação de consistência factual contra a base de conhecimento, o que é especialmente importante nos domínios educacional e de saúde.

Na etapa de registro, simultânea e assíncrona em relação à entrega da resposta, o Módulo de Monitoramento registra a interação completa: intenção identificada, grau de confiança, ação selecionada, fontes consultadas, resposta gerada, tempo de processamento de cada etapa e eventuais exceções. Esse registro não é apenas para fins de auditoria: é a fonte de dados para o processo de melhoria contínua do sistema, que inclui retreinamento de modelos de NLU, identificação de lacunas na Base de Conhecimento e ajuste de limiares

de confiança.

A decisão de separar NLU e Motor de Diálogo em componentes independentes, em vez de implementá-los como um único módulo de processamento conversacional, gera um trade-off entre modularidade e latência. Cada fronteira entre componentes introduz uma chamada de rede (em implementações distribuídas) ou uma serialização e deserialização de dados (em implementações monolíticas modulares), o que aumenta a latência total do pipeline. Em sistemas que adotam um LLM como motor central (onde NLU, gerenciamento de diálogo e NLG são funções do mesmo modelo), essa separação pode parecer artificial. A justificativa da AR para mantê-la é dupla: ela preserva a testabilidade independente de cada componente, essencial para sistemas em domínios regulados onde cada decisão precisa ser auditável, e permite que componentes diferentes evoluam em ritmos diferentes, o que reduz o risco técnico de atualizações.

4.3.2 Visão de Implementação

A Visão de Implementação descreve a organização estática do software em componentes, módulos e dependências, revelando como o sistema está estruturado do ponto de vista da construção. Para representá-la, o Diagrama de Componentes da UML é adotado porque torna explícitas as interfaces que cada componente expõe e consome, permitindo verificar se as dependências são unidirecionais (como exige o estilo em camadas adotado) e se cada componente pode ser substituído sem efeitos colaterais nos demais. A Figura 4.2 apresenta o diagrama resultante.

O critério central que guiou a decomposição em componentes foi a substituibilidade: cada componente deve poder ser trocado por uma implementação alternativa sem que nenhum outro componente precise ser modificado. Esse critério é mais restritivo do que a simples separação de responsabilidades, porque exige que as interfaces entre componentes sejam suficientemente estáveis para sobreviver a mudanças de implementação. A consequência prática é que a AR não especifica tecnologias, mas contratos de interface, isto é, o que entra em cada componente e o que sai, independentemente de como o processamento interno ocorre.

Esse princípio tem implicação direta sobre a testabilidade da AR: se cada com-

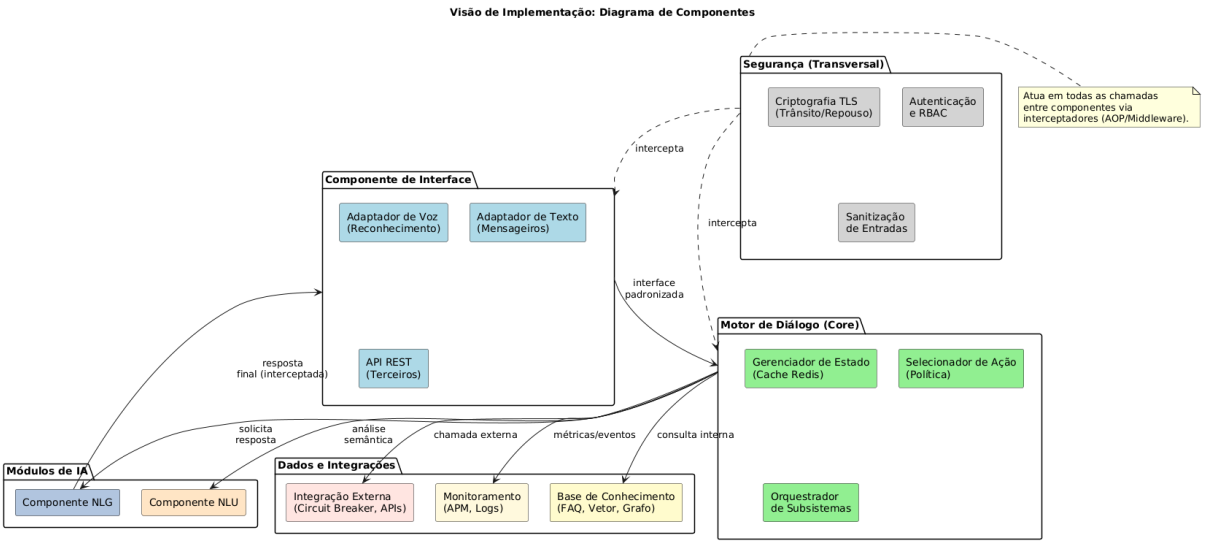

Figura 4.2: Visão de Implementação: Diagrama de Componentes com subcomponentes, dependências e papel transversal do componente de segurança.

Fonte: Autoria própria.

ponente é substituível por interface, ele também é substituível por um dublê de teste (*mock* ou *stub*), o que torna o sistema completamente testável em isolamento. Em sistemas conversacionais modernos, onde o comportamento dos modelos de linguagem é não determinístico, a capacidade de substituir o Módulo de NLU por um dublê determinístico durante os testes é essencial para a reprodutibilidade dos testes de integração.

O **Componente de Interface** encapsula toda a variabilidade relacionada aos canais de comunicação. Sua responsabilidade central é isolar o Motor de Diálogo das particularidades de cada canal: uma mensagem recebida via WhatsApp, via *widget* web ou via chamada de voz deve ser entregue ao Motor de Diálogo na mesma estrutura de dados padronizada, independentemente do formato original. Esse isolamento é implementado por meio de adaptadores de canal, um por tipo de canal suportado. A interface que o Componente de Interface expõe ao Motor de Diálogo é uma única operação: `processar_mensagem(session_id, content, metadata)`, onde `metadata` contém o canal de origem, o formato da entrada e informações de autenticação já validadas. A interface que o Motor de Diálogo expõe ao Componente de Interface é igualmente simples: `entregar_resposta(session_id, content, format_hint)`, onde `format_hint` permite que o NLG sinalize preferências de formatação específicas do canal (texto simples para SMS, *markdown* para aplicativos de mensagem, SSML para síntese de voz).

A decisão de incluir uma API REST como subcomponente do Componente de Interface (e não como componente separado) reflete o entendimento de que a API REST é simplesmente mais um adaptador de canal, desta vez direcionado a sistemas de terceiros que desejam integrar o chatbot como serviço. Expô-la como componente separado criaria uma assimetria arquitetural sem justificativa funcional.

O **Componente de NLU** encapsula o processamento semântico da entrada. Sua responsabilidade mais crítica não é a classificação em si, mas a produção de um grau de confiança calibrado: um NLU que retorna sempre confiança alta, independentemente da ambiguidade real da entrada, é mais prejudicial do que um NLU com precisão moderada mas com incerteza bem calibrada, porque remove do Motor de Diálogo a capacidade de acionar o fluxo de clarificação quando necessário. A interface que o NLU expõe é a operação `analisar(text, language_hint)` → {`intent, entities, confidence, sentiment`}. O campo `language_hint` permite que o Motor de Diálogo informe o idioma detectado na sessão, melhorando a precisão da classificação em sistemas multilíngues.

O subcomponente de análise de sentimento, embora opcional em chatbots de domínio fechado simples, é obrigatório na AR proposta porque é o mecanismo que alimenta um dos critérios de escalamento: a detecção de sentimento negativo intenso pode disparar a transferência para atendimento humano mesmo quando a intenção identificada é tecnicamente tratável pelo chatbot. Em domínios de saúde e suporte emocional, essa capacidade não é opcional.

O **Motor de Diálogo** é o componente de maior complexidade estrutural da AR, porque concentra as responsabilidades de gerenciamento de estado, tomada de decisão e orquestração de subsistemas. Internamente, ele é composto por três subcomponentes com responsabilidades distintas. O gerenciador de estado mantém o contexto ativo da conversa em memória de acesso rápido (cache de sessão), com um mecanismo de expiração configurável: sessões inativas por mais de um período definido são arquivadas na Zona de Dados e o cache é liberado. Essa política de expiração é uma decisão de *trade-off* entre consumo de memória e continuidade da experiência do usuário. O selecionador de ação implementa a política de decisão, que deve respeitar dois invariantes: toda ação selecionada deve ser rastreável (a razão pela qual foi selecionada deve ser registrável para

fins de auditoria) e toda ação deve ter um *fallback* definido (se a ação principal falhar, o sistema deve saber o que fazer em seguida). O orquestrador de subsistemas gerencia as chamadas aos demais componentes com controle de *timeout* e tratamento de exceção, sendo cada chamada com um *timeout* configurável independente.

O **Componente de Base de Conhecimento** abstrai diferentes tipos de repositório de informação sob uma interface uniforme: `consultar(query_structure, top_k)` → `[{content, source, relevance_score}]`. Essa uniformidade é crítica para a extensibilidade: adicionar uma nova fonte de informação requer apenas a implementação de um novo adaptador de repositório dentro do componente, sem modificação da interface exposta ao Motor de Diálogo. Internamente, o componente mantém dois tipos de repositório com características complementares: o repositório estruturado, adequado para consultas por correspondência exata, e o repositório vetorial, que suporta busca por similaridade semântica. A decisão de qual repositório consultar é feita internamente ao componente com base na estrutura da consulta recebida.

O **Componente de Segurança** é o único componente da AR com papel transversal: ele não se insere no pipeline de processamento de forma sequencial, mas atua como interceptador nas fronteiras entre todos os demais componentes. Sua implementação como conjunto de interceptadores (em vez de um serviço centralizado chamado explicitamente por cada componente) garante que nenhum componente precise conhecer ou invocar explicitamente os mecanismos de segurança, eliminando o risco de um componente omitir a validação de autorização antes de executar uma ação. A desvantagem é que os interceptadores introduzem latência adicional em cada fronteira entre componentes e seu comportamento é menos visível no diagrama de sequência. Para fins de auditoria, o Componente de Segurança registra no Módulo de Monitoramento todos os eventos de autenticação, autorização e rejeição de entrada, com metadados suficientes para reconstituir qualquer incidente a partir dos logs.

O principal *trade-off* desta visão é entre granularidade de componentes e *overhead* de comunicação. A decomposição proposta em oito componentes principais maximiza a substituibilidade e a testabilidade em isolamento, mas cada fronteira entre componentes tem custo: serialização de dados, chamadas de rede (em implementações distribuídas),

ou chamadas de função com *overhead* de validação de interface (em implementações monolíticas modulares). Para chatbots de baixo volume ou domínios com requisitos de latência extremamente restritivos (abaixo de 500ms de ponta a ponta), pode ser adequado consolidar NLU e Motor de Diálogo em um único componente, aceitando menor substituibilidade em troca de menor latência. A AR documenta essa possibilidade como ponto de variação: a fusão de NLU e Motor de Diálogo é uma especialização válida da AR.

4.3.3 Visão de Processo

A Visão de Processo descreve os aspectos dinâmicos do sistema em execução, com ênfase nos pontos de decisão, nos fluxos alternativos e nos mecanismos de resiliência. O Diagrama de Atividades da UML com partições (raias) por camada é adotado porque torna visível qual responsabilidade pertence a qual camada do sistema em cada etapa do fluxo, permitindo verificar se as fronteiras de responsabilidade estão sendo respeitadas durante a execução. A Figura 4.3 apresenta o diagrama resultante.

O Diagrama de Atividades da AR apresenta três pontos de decisão estruturais, cada um com critérios explícitos que determinam qual caminho o fluxo percorre. A explicitação desses critérios é o que transforma um diagrama descritivo em uma especificação funcional utilizável por uma equipe de implementação.

O primeiro ponto de decisão ocorre após a classificação de intenção pelo NLU: o grau de confiança retornado é suficiente para prosseguir? O limiar de confiança é um parâmetro configurável da AR (valor padrão sugerido: 0,7, ajustável por domínio). Quando o grau de confiança está acima do limiar, o fluxo prossegue normalmente. Quando está abaixo, o sistema entra em fluxo de clarificação.

O fluxo de clarificação suporta três estratégias selecionáveis por configuração de domínio. A primeira é a pergunta aberta: “Não entendi completamente sua mensagem. Pode reformular?”. A segunda é a pergunta fechada com opções: “Você está perguntando sobre X ou sobre Y?”, que é mais eficiente quando o NLU identifica duas intenções candidatas com confiança similar. A terceira é a hipótese explícita: “Entendi que você quer saber sobre X. Estou certo?”, preferível quando o NLU tem uma hipótese dominante

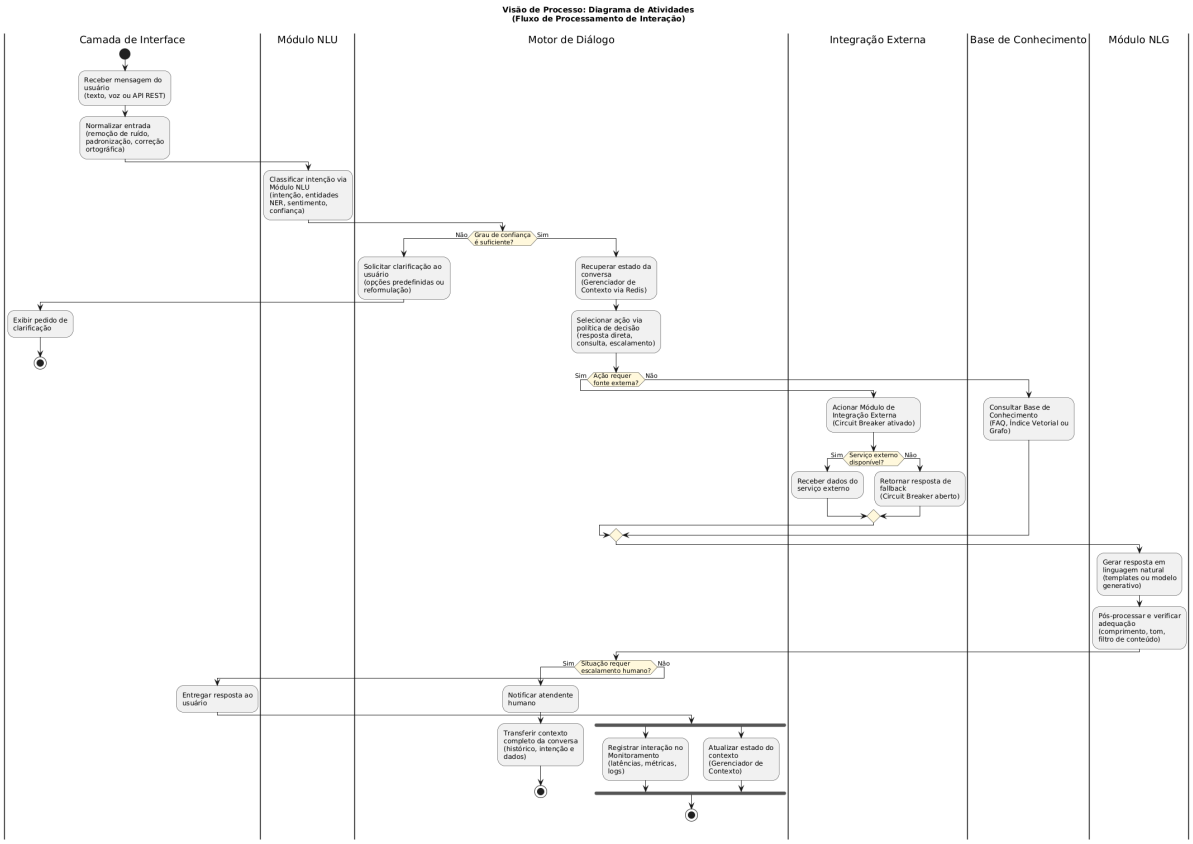

Figura 4.3: Visão de Processo: Diagrama de Atividades com raias por camada, fluxo de clarificação, *circuit breaker* e escalamento humano.

Fonte: Autoria própria.

mas com confiança insuficiente. A AR especifica um limite máximo de iterações de clarificação (valor padrão sugerido: duas iterações). Após esse limite, independentemente do grau de confiança, o fluxo escala para atendimento humano, prevenindo o cenário em que o sistema entra em um loop de clarificação frustrante para o usuário.

O segundo ponto de decisão ocorre após a seleção de ação pelo Motor de Diálogo: a ação requer consulta a fonte externa? Para essas ações, o Módulo de Integração Externa é acionado com o mecanismo de *Circuit Breaker*. O funcionamento do *Circuit Breaker* neste contexto é o seguinte: o circuito começa fechado (requisições passam normalmente); quando o número de falhas consecutivas ao mesmo serviço externo ultrapassa um limiar configurável (por exemplo, cinco falhas em trinta segundos), o circuito abre e requisições subsequentes retornam imediatamente com a resposta de *fallback* configurada; após um período de espera configurável (por exemplo, sessenta segundos), o circuito passa para o estado semiaberto, onde uma única requisição de teste é executada, e o resultado deter-

mina se o circuito fecha novamente ou retorna ao estado aberto. Essa política garante que a falha de um sistema externo não propague degradação para toda a experiência do chatbot.

O terceiro ponto de decisão é o escalamento para atendimento humano, tratado pela AR como mecanismo de primeira classe. Os critérios que disparam o escalamento são configuráveis por domínio, mas a AR define um conjunto mínimo obrigatório: número máximo de iterações de clarificação atingido, detecção de sentimento negativo com intensidade acima de um limiar configurável, solicitação explícita do usuário por atendente humano (essa solicitação nunca deve ser ignorada, independentemente de qualquer outro critério), e identificação de intenção fora do escopo do chatbot. Quando o escalamento é ativado, o sistema notifica o atendente humano disponível com o contexto completo da conversa, transfere o canal de comunicação e registra o escalamento com todos os metadados relevantes. A transferência de contexto é o detalhe mais crítico: sem ela, o usuário precisa repetir ao atendente humano tudo o que já comunicou ao chatbot.

O processamento de uma mensagem não é inteiramente sequencial: o registro no Módulo de Monitoramento é assíncrono e não bloqueia a entrega da resposta ao usuário. Da mesma forma, quando a ação selecionada requer tanto uma consulta à Base de Conhecimento quanto uma chamada a um sistema externo, essas duas consultas podem ser executadas em paralelo, com o Motor de Diálogo aguardando o resultado de ambas antes de invocar o NLG. Em um fluxo sequencial estrito, o tempo total é a soma dos tempos de cada etapa; com paralelismo, o tempo total é determinado pelo componente mais lento dentre os que executam em paralelo. Para chatbots com meta de resposta abaixo de 2 segundos, a introdução de paralelismo entre consultas independentes pode ser determinante para atingir esse requisito.

O principal *trade-off* desta visão é entre resiliência e complexidade operacional. Cada mecanismo de resiliência introduzido, seja o *Circuit Breaker*, as filas, os fluxos de clarificação ou os limites de escalamento, adiciona estados possíveis ao sistema, aumentando a superfície de testes necessária para garantir comportamento correto em todos os cenários. Um sistema sem *Circuit Breaker* é mais simples de testar, mas menos resiliente em produção. A AR aceita essa complexidade adicional porque os sistemas conversacio-

nais em domínios como saúde, financeiro e governo operam em condições onde a resiliência não é opcional.

4.3.4 Visão de Implantação

A Visão de Implantação descreve como os componentes de software são distribuídos pela infraestrutura física ou virtual, tornando explícitas as fronteiras de rede, os pontos de observabilidade e as políticas de acesso que afetam diretamente atributos de qualidade como disponibilidade, desempenho e segurança. Para representá-la, utiliza-se o Diagrama de Implantação da UML, apresentado na Figura 4.4.

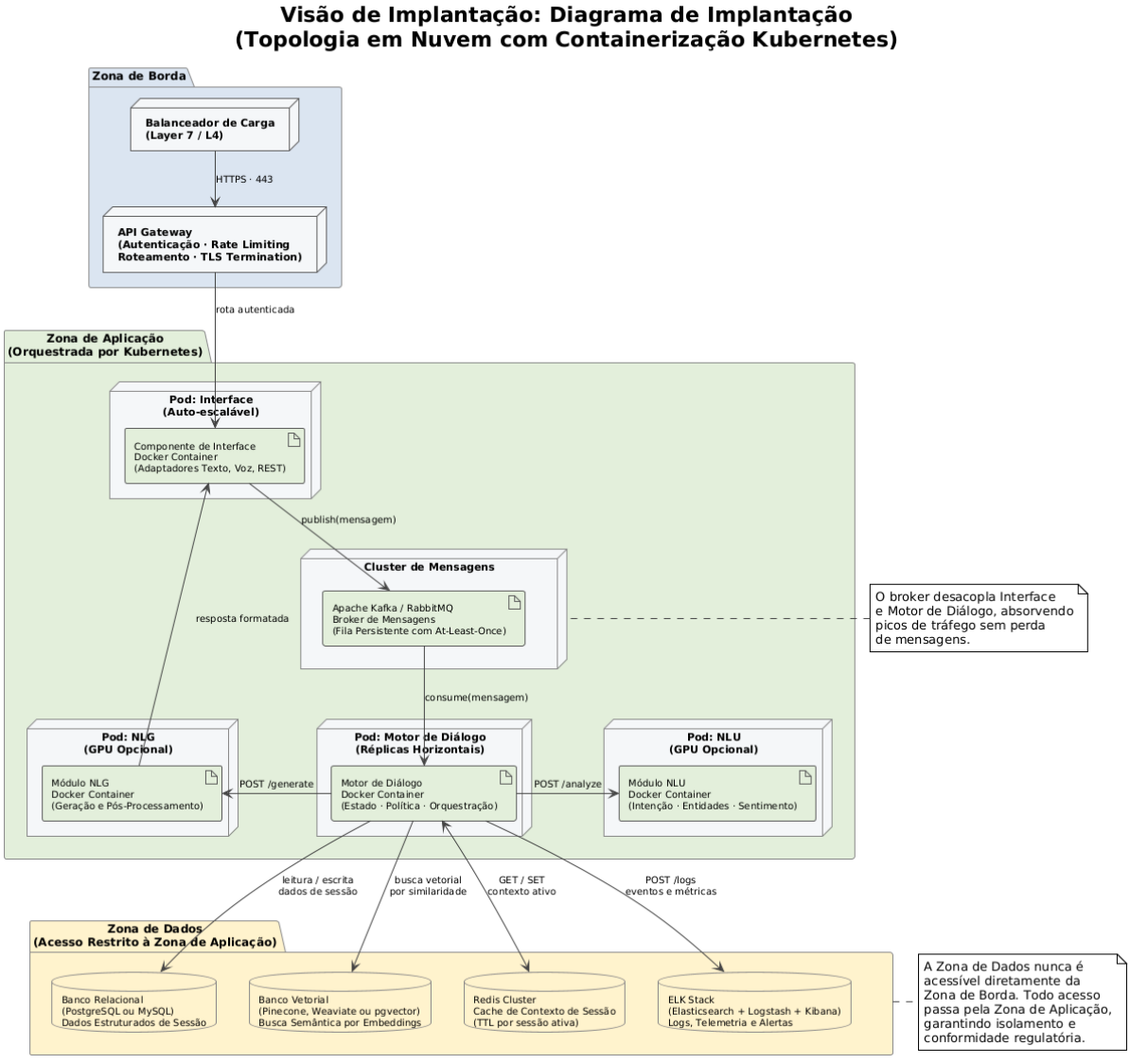

Figura 4.4: Visão de Implantação: distribuição dos componentes em zonas de borda, aplicação e dados, com containerização Docker e orquestração Kubernetes.

Fonte: Autoria própria.

A alternativa de implantação considerada e descartada foi a topologia monolítica em servidor único. Embora de menor custo operacional inicial, essa topologia inviabiliza o escalonamento horizontal (requisito identificado em 41 dos 45 estudos da RSL) e introduz um único ponto de falha que compromete o requisito de disponibilidade mínima de 99,5%. A topologia em zonas com containerização, adotada pela AR, aceita maior custo operacional em troca de escalabilidade, portabilidade e resiliência estrutural.

A AR propõe a divisão da infraestrutura em três zonas com graus decrescentes de exposição ao mundo externo. Essa organização não é apenas administrativa: ela é a materialização física dos requisitos de segurança na topologia de rede, garantindo que dados sensíveis nunca sejam acessíveis a partir de caminhos não controlados.

A **Zona de Borda** é o único ponto de entrada do sistema a partir da internet pública. Ela contém dois elementos estruturais. O primeiro é o balanceador de carga, responsável por distribuir as requisições de entrada entre múltiplas instâncias do Componente de Interface, usando algoritmos de distribuição baseados em saúde das instâncias (excluindo do pool instâncias que falhem nos *health checks*) e em afinidade de sessão quando o protocolo de canal exigir. O segundo é o API Gateway, que centraliza três funções de segurança: autenticação inicial por meio de validação de tokens OAuth 2.0, limitação de taxa de requisições por usuário e por endereço IP (proteção contra ataques de força bruta e de negação de serviço), e roteamento semântico, que direciona requisições de canais distintos para adaptadores correspondentes no Componente de Interface.

A separação entre balanceador de carga e API Gateway é uma decisão deliberada. Um balanceador de carga puro opera na camada de transporte e distribui pacotes; o API Gateway opera na camada de aplicação e inspeciona conteúdo. Colapsá-los em um único componente aumentaria o acoplamento e tornaria mais difícil escalar cada função independentemente, pois o volume de autenticação e o volume de tráfego bruto crescem em taxas diferentes.

A **Zona de Aplicação** contém os componentes de processamento conversacional: Motor de Diálogo, Módulo de NLU, Módulo de NLG e Módulo de Integração Externa. Cada um é implantado como um serviço containerizado com Docker, gerenciado por um orquestrador Kubernetes. A escolha do Kubernetes como orquestrador é justificada por

três capacidades que satisfazem diretamente requisitos da AR: escalonamento automático horizontal, que cria novas instâncias de um componente quando métricas de CPU ou de fila atingem limiares configurados; auto-recuperação, que reinicia instâncias com falha sem intervenção manual; e implantação sem tempo de inatividade (*rolling updates*), que permite atualizar componentes em produção sem interromper o serviço.

O *broker* de mensagens, posicionado na fronteira entre a Zona de Borda e a Zona de Aplicação, implementa a fila descrita na Visão Lógica. Em termos de implantação, ele é o componente que absorve os picos de tráfego: quando a taxa de chegada de mensagens supera a capacidade momentânea dos consumidores na Zona de Aplicação, as mensagens acumulam na fila sem serem descartadas, e são processadas à medida que os consumidores recuperam capacidade. A profundidade da fila em tempo real é a métrica mais direta para acionar o escalonamento automático: quando a profundidade supera um limiar configurado, o Kubernetes instancia novas réplicas do Motor de Diálogo.

A **Zona de Dados** contém os componentes de persistência: a Base de Conhecimento, o armazenamento de contexto de sessão e o sistema de logs e telemetria. A separação desta zona das demais tem implicação de segurança direta: nenhum componente da Zona de Borda tem permissão de rede para se comunicar diretamente com a Zona de Dados. Todo acesso aos dados passa obrigatoriamente pela Zona de Aplicação, que implementa os controles de autorização pertinentes. Esse isolamento é o mecanismo arquitetural que satisfaz requisitos regulatórios de proteção de dados como LGPD e GDPR, que exigem que dados pessoais sejam acessíveis apenas a componentes que possuam necessidade funcional comprovada.

A Base de Conhecimento utiliza dois tipos de armazenamento com propósitos complementares: um banco de dados relacional para dados estruturados com esquema definido (informações de usuários, histórico de sessões, configurações de domínio) e um banco vetorial para busca semântica. A separação entre esses dois tipos não é tecnológica por conveniência: ela reflete naturezas de acesso radicalmente distintas. Consultas relacionais são determinísticas e baseadas em chaves; consultas vetoriais são probabilísticas e baseadas em similaridade semântica. Usar um único sistema para ambas as funções implicaria sacrificar desempenho em uma delas.

O armazenamento de contexto de sessão é implementado em memória (Redis ou equivalente) e não em banco de dados persistente. Essa decisão endereça o requisito de tempo de resposta: o contexto da conversa é o dado mais acessado no fluxo de processamento, consultado a cada interação. A latência de acesso a banco em disco (tipicamente entre 5ms e 20ms) seria acumulada em todas as interações, enquanto a latência de Redis é inferior a 1ms. O custo dessa decisão é a volatilidade: em caso de falha do Redis, o contexto das sessões ativas é perdido, e o chatbot retomará as conversas sem histórico. A AR aceita esse trade-off por considerar que a perda de contexto de sessão, embora ruim para a experiência do usuário, é substancialmente menos grave do que a degradação de latência em todas as interações.

O sistema de logs e telemetria coleta eventos de todos os componentes das três zonas por meio de agentes de coleta instalados em cada nó do cluster Kubernetes. Esses agentes não dependem de chamadas explícitas dos componentes de aplicação: eles interceptam saídas padrão dos processos e eventos da infraestrutura de forma transparente. Isso satisfaz o requisito de observabilidade sem introduzir acoplamento entre os componentes de aplicação e o sistema de monitoramento. Um componente que não conhece o sistema de monitoramento pode ser substituído sem afetar a capacidade de observação do sistema.

4.3.5 Visão de Cenário

A Visão de Cenário, o "+1" do modelo de Kruchten (1995), cumpre uma função distinta das quatro visões anteriores. Enquanto as demais visões descrevem o sistema sob perspectivas especializadas (estrutura lógica, organização de componentes, dinâmica de execução, topologia de infraestrutura), a Visão de Cenário exercita o sistema como um todo a partir da perspectiva de quem o usa, verificando se o conjunto de decisões documentado é suficiente para suportar os objetivos reais dos atores envolvidos.

Para representá-la, utiliza-se o Diagrama de Casos de Uso da UML, apresentado na Figura 4.5.

A AR identifica três atores com perfis e necessidades distintos. O **Usuário Final** é o ator primário: qualquer pessoa que interage com o chatbot por meio de um canal

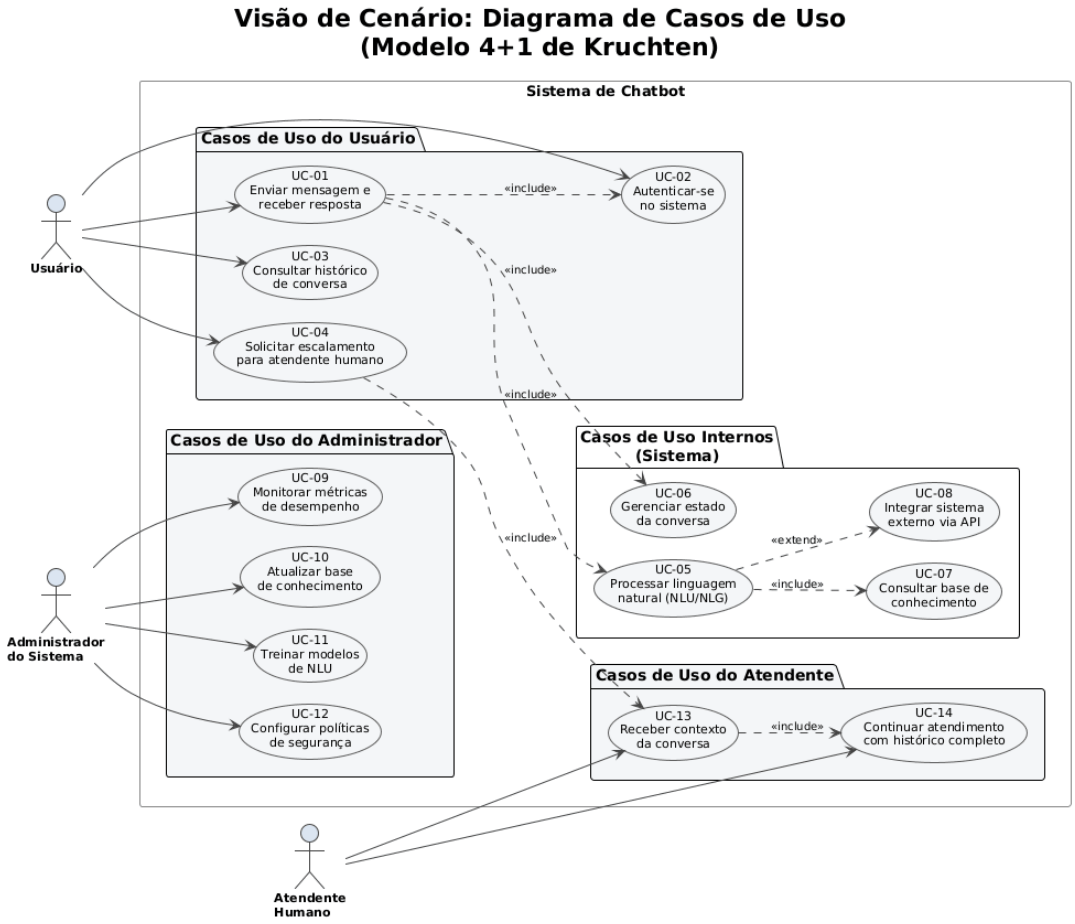

Figura 4.5: Visão de Cenário: Diagrama de Casos de Uso com os três atores principais, casos de uso internos e relações de inclusão e extensão.

Fonte: Autoria própria.

disponível. Suas necessidades fundamentais são receber respostas relevantes em tempo aceitável e ter sua privacidade protegida. O **Administrador do Sistema** é o ator responsável por configurar, monitorar e evoluir o chatbot. Suas necessidades incluem visibilidade sobre o desempenho do sistema, capacidade de atualizar a base de conhecimento sem interrupção do serviço e controle sobre políticas de segurança e acesso. O **Atendente Humano** é o ator que recebe o escalamento de conversas que o chatbot não conseguiu resolver autonomamente. Sua necessidade fundamental é receber o contexto completo da conversa no momento do escalamento, para não obrigar o usuário a repetir informações.

Casos de uso do Usuário Final

O caso de uso central é **Enviar mensagem e receber resposta**, que encapsula o fluxo primário descrito na Visão Lógica. Esse caso de uso inclui, por relação de inclusão

obrigatória, o caso de uso **Autenticar-se no sistema**: todo usuário precisa ser identificado antes de interagir, ainda que a identificação possa ser implícita (por exemplo, um token de sessão gerado automaticamente pelo canal de comunicação) ou explícita (login com credenciais institucionais).

O caso de uso **Solicitar escalamento para atendente humano** representa o fluxo de exceção mais importante da perspectiva do usuário. Ele pode ser ativado de duas formas: por iniciativa do usuário (quando o usuário digita explicitamente que deseja falar com um atendente) ou por decisão automática do sistema (quando o Motor de Diálogo detecta condições que tornam o escalamento necessário, como múltiplas tentativas malsucedidas de identificar a intenção ou detecção de sentimento negativo intenso). Em ambas as formas, o fluxo subsequente é idêntico: o contexto da conversa é transferido ao Atendente Humano e o canal de comunicação é redirecionado.

O caso de uso **Consultar histórico de conversa** permite ao usuário recuperar interações anteriores. Ele ativa o Gerenciador de Contexto e o sistema de logs, e é relevante para domínios onde a continuidade entre sessões é importante, como saúde (onde o usuário pode retomar um processo de avaliação iniciado em sessão anterior) e educação (onde o docente pode revisar as dúvidas levantadas por um estudante ao longo do semestre).

Casos de uso do Administrador

O caso de uso **Monitorar métricas de desempenho** ativa o Componente de Monitoramento e os painéis de telemetria. As métricas de interesse para um administrador de chatbot incluem: taxa de resolução autônoma (proporção de interações resolvidas sem escalamento humano), distribuição de intenções identificadas (que revela quais tópicos geram mais demanda), tempo médio de resposta por componente (que permite identificar gargalos de desempenho), taxa de classificação com baixa confiança (que indica necessidade de retreinamento do modelo de NLU) e volume de escalamentos por categoria (que indica lacunas da base de conhecimento).

O caso de uso **Atualizar a base de conhecimento** é um dos mais críticos para a manutenibilidade do sistema. A AR especifica que esse caso de uso deve ser realizável sem interrupção do serviço em produção: a base de conhecimento é atualizada de forma

incremental, e o novo conteúdo se torna disponível para consulta sem que uma janela de manutenção seja necessária. Isso é viabilizado pela estrutura do Componente de Base de Conhecimento, que expõe uma interface de escrita separada da interface de leitura utilizada pelo Motor de Diálogo.

O caso de uso **Treinar novo modelo de NLU** ativa o processo de atualização do classificador de intenções. Na AR, esse caso de uso é especificado como uma operação assíncrona: o treinamento ocorre em ambiente isolado e o novo modelo é promovido a produção apenas após validação automática contra um conjunto de testes de regressão. Essa especificação evita que um modelo mal treinado seja implantado inadvertidamente em produção, o que causaria degradação da qualidade de classificação perceptível pelos usuários.

O caso de uso **Configurar políticas de segurança** permite ao administrador ajustar parâmetros do Componente de Segurança sem modificar código: limites de taxa de requisição por usuário, algoritmos de autenticação aceitos, categorias de conteúdo bloqueadas pelo pós-processador do NLG e políticas de retenção de logs de auditoria.

Verificação de consistência entre visões

A função de verificação da Visão de Cenário é exercitada cruzando cada caso de uso com os componentes das visões anteriores. O resultado dessa verificação é apresentado a seguir.

O caso de uso **Enviar mensagem e receber resposta** ativa, em sequência: Componente de Interface, fila de mensagens, Motor de Diálogo, Módulo de NLU, Gerenciador de Contexto, Base de Conhecimento ou Módulo de Integração Externa (dependendo da ação selecionada), Módulo de NLG e Módulo de Monitoramento. Todos esses componentes estão presentes na Visão de Implementação, e seu fluxo de ativação é coerente com a Visão Lógica e a Visão de Processo. A verificação passa.

O caso de uso **Solicitar escalamento** ativa: Motor de Diálogo (para detecção automática ou recepção da solicitação explícita), notificação ao Atendente Humano (via Módulo de Integração Externa ou canal dedicado de notificação) e transferência de contexto (via Gerenciador de Contexto). A Visão de Implementação deve contemplar o

mecanismo de notificação ao atendente: se esse mecanismo não estiver explicitamente representado, há uma inconsistência a resolver. Na AR proposta, esse mecanismo é implementado como um conector de saída do Módulo de Integração Externa, configurável por domínio (e-mail, sistema de ticketing, plataforma de atendimento).

O caso de uso **Treinar novo modelo de NLU** ativa componentes que não aparecem no fluxo de processamento principal: um ambiente de treinamento, um repositório de conjuntos de dados e um pipeline de validação automática. Esses componentes são endereçados na Visão de Implantação como componentes da Zona de Aplicação que operam fora do caminho crítico de atendimento, mas precisam ter acesso de leitura à Base de Conhecimento e acesso de escrita ao Componente de NLU. Essa verificação revelou a necessidade de especificar explicitamente as permissões de rede entre o ambiente de treinamento e as demais zonas, o que constitui um refinamento da Visão de Implantação decorrente da análise da Visão de Cenário, ilustrando precisamente a função integradora que Kruchten (1995) atribui a essa visão.

4.4 Mapeamento entre Requisitos Não Funcionais e Componentes

A Tabela 4.3 consolida o relacionamento entre os doze requisitos não funcionais elicitados e os componentes e mecanismos arquiteturais que os endereçam. O mapeamento serve a dois propósitos complementares: verificar a cobertura (todo RNF tem pelo menos um componente responsável) e tornar explícitos os trade-offs (um mecanismo que satisfaz um RNF pode penalizar outro). A análise dos trade-offs é o aspecto mais relevante desta seção para uma equipe que instancia a AR em um domínio específico.

Tabela 4.3: Mapeamento entre Requisitos Não Funcionais e Componentes da AR.

Requisito Não Funcional	**Componente(s) Responsável(is)**	**Mecanismo Arquitetural**
Escalabilidade	Motor de Diálogo, Zona de Aplicação	Fila de mensagens, escalonamento horizontal via Kubernetes
Disponibilidade	Todos	Redundância de instâncias, *Circuit Breaker*, *fallback*
Confidencialidade	Componente de Segurança, Zona de Dados	Criptografia TLS, criptografia em repouso, pseudonimização de logs
Autenticação	Componente de Segurança, API Gateway	OAuth 2.0, autenticação multifatorial
Tempo de resposta	Módulo de NLU, Motor de Diálogo	Cache de sessão em Redis, pipeline com paralelismo de consultas
Interoperabilidade	Componente de Interface, Integração Externa	Adaptadores por canal, API REST padronizada
Manutenibilidade	Todos	Containerização, separação em camadas, interfaces padronizadas
Portabilidade	Zona de Implantação	Docker, Kubernetes, independência de provedor de nuvem
Auditabilidade	Módulo de Monitoramento	Logging estruturado, rastreabilidade de interações

Requisito Não Funcional	**Componente(s) Responsável(is)**	**Mecanismo Arquitetural**
Extensibilidade	Componente de NLU, Base de Conhecimento	Interfaces substituíveis, adaptadores de repositório
Suporte multilíngue	Módulo de NLU, Módulo de NLG	Modelos de linguagem parametrizáveis por idioma, *language_hint* na interface do NLU
Tolerância a falhas	Integração Externa, Motor de Diálogo	*Circuit Breaker*, respostas de *fallback*, limite de iterações de clarificação

Fonte: Autoria própria.

A análise dos trade-offs derivados desse mapeamento revela quatro tensões estruturais entre requisitos que qualquer instância da AR precisará resolver por configuração ou por especialização de componente.

Trade-off T1: Escalabilidade versus Tempo de Resposta. O mecanismo de fila que satisfaz o requisito de Escalabilidade introduz latência adicional no fluxo de processamento. Em condições normais de carga, a latência da fila é negligenciável, uma vez que as mensagens são consumidas tão rapidamente quanto são produzidas. Em condições de carga elevada, quando múltiplas instâncias do Motor de Diálogo são necessárias, o tempo de espera antes do consumo cresce com a profundidade da fila. A resposta arquitetural a esse problema é o provisionamento antecipado (aumentar o número de instâncias antes que a fila cresça) em vez do provisionamento reativo (aumentar após a fila crescer), o que requer integração com sistemas de monitoramento de métricas preditivas.

Trade-off T2: Auditabilidade versus Confidencialidade. O requisito de Auditabilidade exige que todas as interações sejam registradas com detalhamento suficiente para reconstituir qualquer incidente. O requisito de Confidencialidade exige que dados dos usuários sejam protegidos. Esses dois requisitos entram em conflito quando as

interações contêm dados pessoais sensíveis, como ocorre sistematicamente em chatbots de saúde e financeiro. A resolução arquitetural adotada pela AR é a pseudonimização dos logs: o identificador de usuário armazenado nos logs de auditoria é um token pseudônimo gerado por função *hash* com sal secreto a partir do identificador real, e a tabela de correspondência entre identificadores reais e pseudônimos é armazenada separadamente com controle de acesso mais restritivo. Isso permite auditoria técnica sem que os logs de auditoria, por si sós, constituam exposição de dados pessoais.

Trade-off T3: Extensibilidade versus Consistência. O mecanismo de interfaces substituíveis que satisfaz o requisito de Extensibilidade introduz risco de inconsistência quando múltiplos repositórios de conhecimento são ativados simultaneamente. Se dois repositórios contêm informações contraditórias sobre o mesmo tema (especialmente provável quando os ciclos de atualização são diferentes), o Motor de Diálogo precisa de uma política de resolução de conflito. A AR especifica que essa política é responsabilidade do Componente de Base de Conhecimento, que deve retornar os resultados com metadados de fonte e *timestamp* de última atualização, permitindo que o Motor de Diálogo implemente a política adequada ao domínio (por exemplo, priorizar sempre a fonte mais recente, ou priorizar a fonte com maior pontuação de relevância, ou sinalizar ao usuário a existência de informações divergentes).

Trade-off T4: Manutenibilidade versus Desempenho. A containerização e a separação em camadas que satisfazem o requisito de Manutenibilidade introduzem *overhead* de comunicação em relação a uma implementação monolítica altamente otimizada. Em termos práticos, uma chamada de rede entre o Motor de Diálogo e o Módulo de NLU tem latência tipicamente entre 5ms e 50ms (dependendo da infraestrutura), enquanto uma chamada de função local teria latência abaixo de 1ms. Em um pipeline com seis etapas sequenciais, esse *overhead* pode acumular entre 30ms e 300ms, representando entre 1,5% e 15% do orçamento de latência total de 2 segundos. Para domínios com requisitos de latência extremamente restritivos, a AR recomenda colocar NLU e Motor de Diálogo na mesma zona de rede ou na mesma instância de processo, sacrificando parcialmente a substituibilidade independente em troca de latência reduzida.

5 Instância Conceitual no Domínio Educacional

O domínio educacional foi escolhido como contexto de instanciação conceitual da AR por três razões complementares. Primeiro, é um domínio com demanda real e crescente por chatbots, especialmente no ensino superior, onde a proporção de alunos por professor torna inviável o atendimento individualizado em larga escala. Segundo, é um domínio com características técnicas ricas, que exercitam múltiplas dimensões da AR proposta: necessidade de personalização, integração com sistemas acadêmicos, suporte a múltiplos idiomas e tratamento de conteúdo técnico complexo. Terceiro, está alinhado ao contexto acadêmico deste trabalho, onde uma instância de chatbot de suporte a disciplinas de computação é especialmente pertinente.

A instância conceitual proposta é um assistente conversacional voltado ao suporte de alunos de graduação em disciplinas de ciência da computação. Seu escopo primário abrange: resposta a dúvidas conceituais sobre conteúdo programático das disciplinas; orientação sobre atividades e avaliações; fornecimento de referências bibliográficas e materiais complementares; e registro de dúvidas recorrentes para alimentar o planejamento pedagógico do docente.

5.1 Requisitos Específicos do Domínio

A instanciação sobre o domínio educacional introduz requisitos que se somam aos requisitos gerais da AR e que guiam as escolhas de personalização.

Em termos de requisitos funcionais específicos, a instância deve ser capaz de identificar o nível de proficiência do aluno com base nas dúvidas formuladas e calibrar a complexidade e o estilo das respostas; deve suportar consultas sobre múltiplas disciplinas com bases de conhecimento distintas, sem confundir conteúdo entre elas; deve integrar-se ao sistema acadêmico da instituição para verificar o calendário de avaliações, prazos de

entrega e ementa das disciplinas; e deve ser capaz de detectar quando uma dúvida excede o escopo de um chatbot e recomendar o atendimento presencial com o docente ou monitor.

Em termos de requisitos não funcionais específicos, o domínio educacional adiciona ênfase ao requisito de precisão: respostas imprecisas em contexto educacional não apenas frustram o aluno, mas potencialmente induzem erros de aprendizagem. Isso impõe sobre o Módulo de NLG um critério de qualidade mais exigente, com mecanismos de verificação de consistência factual contra a base de conhecimento antes de entregar a resposta. O requisito de privacidade também assume caráter específico, pois dados de desempenho e padrão de dúvidas de estudantes são dados sensíveis sujeitos à LGPD e a regulações educacionais internacionais como a FERPA (*Family Educational Rights and Privacy Act*). A instância deve, portanto, implementar política de retenção mínima de dados: logs de conversa são anonimizados após 90 dias, e o modelo do estudante é acessível apenas ao próprio estudante e ao docente responsável pela disciplina.

5.2 Mapeamento da AR para a Instância

A instanciação de uma AR consiste em selecionar, especializar e, quando necessário, estender os componentes da AR para atender aos requisitos específicos do domínio de aplicação. A Figura 5.1 apresenta o diagrama conceitual da instância educacional.

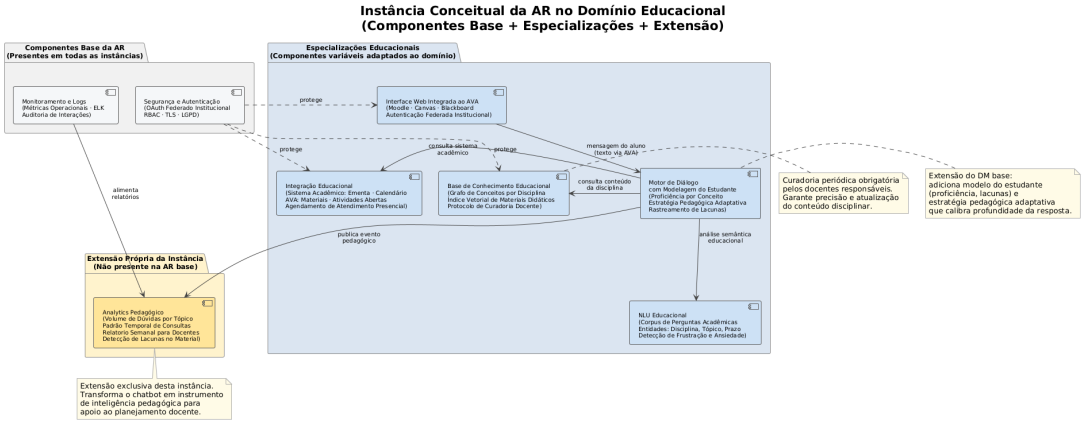

Figura 5.1: Instância conceitual da AR no domínio educacional: componentes base em cinza, especializações educacionais em azul e extensão de Analytics Pedagógico em amarelo.

Fonte: Autoria própria.

A distinção entre o diagrama da AR base (Capítulo 4) e o diagrama da instância revela três categorias de decisão de instanciação. Os componentes em cinza são adotados sem modificação estrutural, apenas com configuração de parâmetros. Os componentes em azul são especializações: mantêm a interface da AR base, mas têm sua implementação interna substituída ou estendida para atender requisitos educacionais. O componente em amarelo é uma extensão pura: não existia na AR base e foi acrescido para cobrir uma funcionalidade sem equivalente no modelo geral.

A Tabela 5.1 detalha, para cada componente da AR base, os valores e especializações adotados na instância educacional. Esse nível de especificação é o que diferencia uma instância conceitual de uma descrição de projeto suficientemente detalhada para guiar implementação.

Tabela 5.1: Parametrização dos componentes da AR base na instância educacional.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Componente de Interface	Configuração de canal	Canal primário: interface web integrada ao AVA institucional (Moodle/Canvas). Autenticação federada com login institucional via OAuth 2.0. Canal secundário: API REST para integração com aplicativos móveis da instituição.
Componente de Interface	Adaptadores ativos	Adaptador de texto ativado. Adaptador de voz desativado na versão inicial. API REST ativada para acesso programático por docentes.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Módulo de NLU	Corpus de treinamento	Perguntas de disciplinas de Estruturas de Dados, Algoritmos, Programação Orientada a Objetos e Banco de Dados, coletadas em formulários aplicados em semestres anteriores na UFJF.
Módulo de NLU	Intenções definidas	Seis categorias: dúvida conceitual, dúvida comparativa, pedido de exemplo, solicitação de referência bibliográfica, consulta sobre avaliação/prazo e pedido de exercício prático.
Módulo de NLU	Limiar de confiança	0,75. Abaixo desse valor, o sistema solicita clarificação antes de agir.
Módulo de NLU	Detecção de sentimento	Três estados relevantes ao contexto educacional: frustração com dificuldade (dispara resposta com encorajamento pedagógico), ansiedade pré-avaliação (dispara orientação sobre recursos de apoio) e desengajamento (notifica docente via Analytics Pedagógico).

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Motor de Diálogo	Política de decisão	Máquina de estados finita, correspondentes aos fluxos de dúvida conceitual, dúvida comparativa, consulta administrativa, clarificação, escalamento e encerramento. Escolhida em detrimento de aprendizado por reforço pela auditabilidade exigida no domínio educacional.
Motor de Diálogo	Extensão: Modelo do Estudante	Representação dinâmica de proficiência por tópico (escala ordinal de três níveis: iniciante, intermediário, avançado), atualizada a cada interação. Parâmetro de decaimento temporal: tópico sem interação há mais de 30 dias tem proficiência reduzida em um nível.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Motor de Diálogo	Escalamento para humano	Critérios de disparo: três tentativas consecutivas de clarificação sem sucesso; sentimento de ansiedade intensa detectado pelo NLU; solicitação explícita do usuário; intenção classificada fora do conjunto de seis categorias treinadas. Destino do escalamento: sistema de agendamento de atendimento com o monitor da disciplina.
Gerenciador de Contexto	Janela de contexto	Dez turnos de conversa mantidos em memória de curto prazo (Redis, TTL de 30 minutos). Resumo das sessões anteriores persistido em banco relacional por até 90 dias, após os quais é anonimizado.
Base de Conhecimento	Estrutura	Grafo de conhecimento por disciplina: nós são conceitos com metadados de dificuldade (escala 1-5) e pré-requisitos; arestas representam relações de dependência conceitual. Materiais indexados em banco vetorial.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Base de Conhecimento	Fontes indexadas	Livros-texto adotados nas disciplinas; slides das aulas (formato PDF); listas de exercícios com gabarito comentado; ementas e planos de ensino. Atualização sem interrupção de serviço mediante interface de escrita segregada da interface de leitura.
Base de Conhecimento	Política de conflito	Prioridade para a fonte com *timestamp* mais recente em caso de conteúdo contraditório entre semestres. Sinalização ao usuário quando a divergência é identificada entre fontes de mesma data.
Módulo de NLG	Estratégia de geração	Resposta em dois estágios: introdução do pré-requisito (quando o Modelo do Estudante indica lacuna no pré-requisito), seguida do conceito solicitado. *Templates* parametrizáveis por nível de proficiência: iniciante (analogias e exemplos concretos), intermediário (definição formal e exemplo) e avançado (definição formal e comparação com alternativas).

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Módulo de NLG	Pós-processamento	Verificação factual automática: cada afirmação da resposta é comparada contra a base de conhecimento antes da entrega. Respostas com afirmações sem respaldo na base são sinalizadas ao usuário com indicação de incerteza. Limiar de comprimento: respostas superiores a um determinado número X de palavras são resumidas com referência ao material completo.
Componente de Integração Externa	Conectores ativos	Sistema acadêmico institucional (consulta de calendário, ementa e notas, somente leitura via API REST); AVA institucional (referência a materiais publicados e atividades abertas); sistema de agendamento de atendimento com monitores.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Componente de Segurança	Política de dados	Pseudonimização de logs após 90 dias. Acesso ao Modelo do Estudante restrito ao próprio estudante e ao docente da disciplina, controlado por papel no sistema de autenticação institucional. Conformidade com LGPD e FERPA documentada em política de privacidade acessível no AVA.
Módulo de Monitoramento	Métricas operacionais	Latência por estágio do pipeline; taxa de clarificação por disciplina; taxa de escalamento por disciplina; distribuição de intenções por semana.

Componente da AR Base	**Tipo de Decisão**	**Valor ou Especialização na Instância Educacional**
Módulo de Monitoramento	Extensão: Analytics Pedagógico	Componente adicional, sem equivalente na AR base. Agrega interações em relatórios semanais por disciplina, identificando: tópicos com maior volume de dúvidas; padrões temporais de dúvidas em relação ao calendário de avaliações; perguntas recorrentes sem resposta satisfatória na base de conhecimento (indicadores de lacuna no material didático). Relatórios entregues ao docente via painel no AVA.

Fonte: Autoria própria.

A tabela evidencia que a maior parte das decisões de instanciação incide sobre três componentes: o Módulo de NLU (corpus, intenções e limiar), o Motor de Diálogo (política de decisão e Modelo do Estudante) e a Base de Conhecimento (estrutura em grafo e política de conflito). Esses três componentes concentram o que é genuinamente específico do domínio educacional. Os demais componentes são parametrizados por configuração, sem necessidade de especialização de implementação, o que confirma a hipótese de que a AR base é suficientemente abstrata para acomodar instâncias de domínios distintos com baixo retrabalho estrutural.

5.3 Fluxo de Interação Típico na Instância Educacional

Para ilustrar concretamente como a instância opera, apresenta-se um fluxo de interação típico. Um estudante do terceiro período, matriculado em Estruturas de Dados, acessa o chatbot pelo AVA e digita: "Não entendi quando usar árvore AVL em vez de árvore binária normal."

O Módulo de NLU classifica a intenção como "dúvida comparativa" e extrai as entidades "árvore AVL" e "árvore binária de busca". O grau de confiança é 0,94, superior ao limiar de 0,75 definido na Tabela 5.1, portanto o sistema não solicita clarificação.

O Motor de Diálogo consulta o Gerenciador de Contexto e verifica que é a primeira mensagem desta sessão. Consulta o Modelo do Estudante e identifica que este aluno tem histórico de três interações anteriores sobre árvores binárias, com perguntas sobre inserção e remoção, mas nenhuma sobre balanceamento. O nível de proficiência registrado para o tópico "balanceamento em árvores" é iniciante. Conforme a estratégia de geração definida para esse nível, a resposta adotará analogias e exemplos concretos antes da definição formal.

O Motor de Diálogo formula uma consulta para a Base de Conhecimento: "distinção entre árvore AVL e ABB, aplicabilidade, conceito de balanceamento". A Base retorna trechos relevantes do livro-texto da disciplina e do slide da aula mais recente sobre o tema, com metadados de fonte e data.

O Módulo de NLG gera uma resposta em dois estágios: primeiro introduz o conceito de desbalanceamento em árvores binárias de busca com um exemplo numérico concreto, dado que o Modelo do Estudante indica que esse pré-requisito apresenta lacuna; em seguida, apresenta a invariante da AVL como solução ao problema identificado e conclui com uma orientação prática sobre quando cada estrutura é preferível.

O pós-processador verifica factualmente a resposta contra a base de conhecimento. A resposta é aprovada e entregue ao estudante. O Gerenciador de Contexto registra a interação na janela de dez turnos. O Modelo do Estudante atualiza o nível de proficiência em "balanceamento em árvores" de iniciante para intermediário, dado que o tópico foi

abordado com resposta satisfatória. O Analytics Pedagógico incrementa o contador de dúvidas sobre AVL para o relatório semanal do docente da disciplina de Estruturas de Dados.

6 Análise dos Resultados

A aplicação do ProSA-RA ao domínio de chatbots produziu resultados coerentes com o objetivo estabelecido: uma AR documentada em cinco visões arquiteturais, derivada de fontes sistemáticas e capaz de ser instanciada em domínios específicos com adaptações rastreáveis.

A hipótese de que chatbots de domínios distintos compartilham uma estrutura arquitetural comum foi confirmada pela investigação. O pipeline NLU-DM-NLG aparece, com variações de nomenclatura, em todos os sete sistemas analisados e em todos os quatro frameworks estudados. Essa convergência sugere que o problema arquitetural do processamento conversacional tem uma solução de fato dominante ao menos no nível estrutural, o que valida a utilidade de uma AR para capturar e disseminar esse padrão.

Por outro lado, a diversidade de requisitos não funcionais entre domínios foi maior do que o esperado. A distância arquitetural entre um chatbot de saúde (Ada Health, com foco em confiabilidade e explicabilidade) e um chatbot de entretenimento (Character.AI, com foco em criatividade e coerência de persona) é significativa o suficiente para sugerir que, em domínios muito especializados, uma AR de segundo nível pode ser necessária. Essa hierarquia de ARs é um conceito reconhecido na literatura (NAKAGAWA et al., 2014) e constitui uma extensão natural deste trabalho.

A AR apresenta quatro pontos fortes que merecem destaque. O primeiro é a **fundamentação sistemática**, que é consequência direta da adoção do ProSA-RA como processo condutor e da RSL como instrumento principal da etapa RA-1. A decisão de conduzir uma RSL com protocolo PRISMA documentado, em vez de utilizar revisão narrativa, implicou um esforço metodológico adicional, mas produziu um resultado que não seria possível de outra forma: cada componente da AR pode ser rastreado às fontes que o justificam. O Módulo de NLU é justificado pela presença do pipeline NLU-DM-NLG em 38 dos 45 estudos. O padrão *Circuit Breaker* é justificado pelos 19 estudos que o identificam como mecanismo de resiliência para integrações externas. A fila de mensagens entre Interface e Motor de Diálogo é justificada pelos 23 estudos que a identificam como

mecanismo de desacoplamento de carga. Sem a RSL, essas decisões existiriam, mas sem rastreabilidade, tornando impossível avaliar se derivam de evidências ou do julgamento isolado do autor.

O segundo é a **cobertura de atributos de qualidade**, que é consequência direta da etapa RA-2 e do mapeamento consolidado na Tabela 4.3. A decisão de organizar a análise arquitetural em torno dos doze RNFs, antes de definir qualquer componente, inverteu a ordem habitual do desenvolvimento ad-hoc, onde atributos de qualidade são incorporados reativamente após a estrutura estar definida. Essa inversão tem consequências concretas: o Componente de Segurança surgiu como elemento transversal porque o requisito de confidencialidade, identificado em 39 estudos, não pode ser satisfeito por um componente isolado; ele precisa atuar como interceptador em todas as chamadas entre componentes. Se a análise tivesse começado pelos componentes e não pelos RNFs, o Componente de Segurança provavelmente teria sido modelado como um módulo de autenticação pontual, com cobertura incompleta.

O terceiro é a **flexibilidade de instanciação**, que é consequência direta da distinção, feita na etapa RA-3, entre estilos arquiteturais fixos e componentes com pontos de variação. A decisão de adotar o estilo de *pipeline* combinado com interfaces padronizadas entre componentes permitiu que a instância educacional adicionasse o Módulo de Analytics Pedagógico e especializasse o Motor de Diálogo com a Modelagem do Estudante sem nenhuma modificação nos componentes restantes da AR base. Isso não é uma propriedade acidental: é o resultado direto de uma decisão arquitetural que prioriza a substituibilidade sobre a otimização local. Um Motor de Diálogo monolítico seria mais eficiente em termos de latência, mas tornaria essa especialização impossível sem reescrita.

O quarto é a **organização em cinco visões complementares**, que é consequência direta da adoção do modelo "4+1" de Kruchten (1995). A decisão de documentar a AR em múltiplas visões, em vez de produzir um único diagrama de componentes, revelou inconsistências que não seriam visíveis de outra forma. A Visão de Cenário, ao exercitar o caso de uso de treinamento de novo modelo de NLU sobre as demais visões, revelou que o ambiente de treinamento não estava representado na Visão de Implantação. Essa inconsistência foi corrigida antes da versão final. Em um processo de documentação

com visão única, ela teria permanecido oculta.

Quanto às limitações e pontos de melhoria, a primeira limitação é a **ausência de avaliação formal com o framework FERA**, que deixa sem resposta a pergunta sobre quais defeitos de documentação a AR ainda apresenta. A experiência de Nakagawa et al. (2014) com a AR SiMuS é ilustrativa: mesmo após um processo cuidadoso, a avaliação com FERA identificou ausência de informações sobre limitações das visões, padrões internacionais utilizados e pontos de variabilidade explicitamente documentados. A AR proposta neste trabalho certamente apresenta problemas análogos. A ausência do FERA não invalida a AR, mas estabelece um limite claro de confiança: pode-se afirmar que as decisões foram tomadas com base em evidências sistemáticas, mas não que a documentação resultante está livre de lacunas detectáveis por revisão externa.

A segunda limitação é a **ausência de validação empírica por instanciação implementada**. A instância educacional do Capítulo 5 demonstra que a AR é instanciável conceitualmente, mas não que as decisões arquiteturais resistem a condições reais de uso. Em particular, três decisões merecem validação empírica antes de serem adotadas em produção: o modelo do estudante com atualização incremental por interação pode gerar perfis imprecisos se o aluno fizer perguntas fora do seu nível real para testar o sistema; a política de conflito na base de conhecimento educacional (priorizar a fonte mais recente) pode ser inadequada quando materiais didáticos mais recentes ainda não foram revisados pelo docente; e a integração com o AVA institucional depende de APIs que variam entre instituições, tornando o Módulo de Integração Educacional potencialmente mais complexo de implementar do que sua descrição conceitual sugere.

A terceira limitação é a **cobertura incompleta de domínios emergentes**. Os sete sistemas analisados na etapa RA-1 cobrem bem os domínios estabelecidos de chatbots, mas três categorias emergentes foram analisadas apenas tangencialmente: chatbots jurídicos, que introduzem o requisito de raciocínio baseado em precedentes; agentes conversacionais em realidade aumentada, que introduzem requisitos de percepção de contexto espacial; e sistemas multiagente, onde múltiplos chatbots colaboram em uma tarefa com requisitos de coordenação não contemplados na AR base.

A quarta limitação é a **cobertura parcial de chatbots baseados em LLM**.

A AR é abstrata o suficiente para acomodar LLMs como implementação do Módulo de NLG ou do Motor de Diálogo, mas não oferece orientação sobre os desafios arquiteturais próprios desses modelos. O controle de alucinação mediante arquiteturas RAG (LEWIS et al., 2020), o gerenciamento de custo de inferência em escala e os mecanismos de moderação de conteúdo introduzem decisões arquiteturais que a AR atual simplesmente não cobre, porque nenhum dos 45 estudos da RSL os tratava com profundidade suficiente para sustentar uma decisão rastreável.

6.1 Rastreabilidade entre RSL, RNFs e Decisões Arquiteturais

A rastreabilidade entre as etapas do ProSA-RA é, em si mesma, um resultado verificável. O quadro a seguir sintetiza a cadeia de evidência para as quatro decisões arquiteturais de maior impacto na AR proposta.

A **decisão D1** (fila de mensagens entre Interface e Motor de Diálogo) foi motivada pelo RNF de Escalabilidade, identificado como o mais frequente na RSL (41 dos 45 estudos), e pelo padrão de fila identificado em 23 estudos como o mecanismo dominante de desacoplamento de carga. Na Visão Lógica, ela se manifesta como a quebra de sincronismo entre recepção e processamento. Na Visão de Implantação, ela se materializa no *broker* de mensagens (Apache Kafka ou RabbitMQ) posicionado entre a Zona de Borda e a Zona de Aplicação.

A **decisão D2** (organização em três zonas com isolamento progressivo de acesso) foi motivada pelos RNFs de Confidencialidade e Autenticação, identificados em 39 estudos como críticos após a vigência do GDPR e da LGPD. Na Visão de Implantação, ela se manifesta na separação explícita entre Zona de Borda, Zona de Aplicação e Zona de Dados, com a regra de que a Zona de Dados nunca é acessível diretamente da Zona de Borda.

A **decisão D3** (padrão *Circuit Breaker* no Componente de Integração Externa) foi motivada pelo RNF de Tolerância a Falhas e pelo padrão identificado em 19 estudos como mecanismo de resiliência para integração com serviços de terceiros. Na Visão de

Processo, ela se manifesta no fluxo alternativo que é ativado quando uma integração externa falha repetidamente, retornando uma resposta de *fallback* em vez de propagar o erro para o usuário.

A **decisão D4** (interface padronizada do Módulo de NLU) foi motivada pelo RNF de Manutenibilidade e pela observação, presente em todos os quatro *frameworks* analisados, de que a separação entre NLU e Motor de Diálogo com interface bem definida é o que permite que diferentes modelos de linguagem sejam trocados sem modificação da lógica de diálogo. Na Visão de Implementação, ela se manifesta na especificação da interface do NLU como um contrato com quatro campos fixos: intenção, entidades, grau de confiança e sentimento.

Essa cadeia de evidência, RNF e decisão é o que distingue uma AR sistematicamente derivada de uma proposta de design ad-hoc, e é o critério pelo qual a qualidade deste trabalho deve ser avaliada.

A Tabela 6.1 apresenta a comparação entre a AR proposta e ARs correlatas da literatura, evidenciando sua posição em relação ao estado da arte.

Tabela 6.1: Comparação entre a AR proposta e ARs correlatas.

Critério	**AR Proposta**	**JAUS**	**SiMuS**	**Continua**
Processo de criação	ProSA-RA	Ad-hoc	ProSA-RA	Consórcio industrial
Domínio-alvo	Chatbots (geral)	Robótica móvel	Multi-robótica de serviço	Dispositivos de saúde
Cobertura de RNFs	12 RNFs explícitos	Foco em tempo real	Foco em modularidade	Foco em interoperabilidade
Visões documentadas	“4+1” (5 visões)	Não padronizado	5 visões (SysML)	Não padronizado
Avaliação formal	Fora do escopo	Não reportada	FERA (93 questões)	Validação industrial
Instanciação	Sim (conceitual)	Sim (implementada)	Sim (implementada)	Sim (implementada)

Fonte: Autoria própria, com base em Nakagawa et al. (2014).

A comparação evidencia que a AR proposta está em posição competitiva com ARs do estado da arte em termos de rigor do processo, cobertura de RNFs e rastreabilidade. Em relação à SiMuS, a AR com aplicação mais próxima do ProSA-RA na literatura, a

principal diferença favorável é a rastreabilidade explícita entre RSL, RNFs e decisões, que a SiMuS documenta apenas parcialmente. A principal diferença desfavorável é a ausência de avaliação com FERA e de instanciação implementada, que na SiMuS foram conduzidas e produziram refinamentos documentados.

A aplicação do ProSA-RA ao domínio de chatbots confirmou ainda a observação de Nakagawa et al. (2014) sobre a principal dificuldade do processo: a etapa de generalização, que consiste em identificar, a partir de requisitos específicos de sistemas concretos, os requisitos arquiteturais que descrevem adequadamente uma classe de sistemas. No caso dos chatbots, essa dificuldade foi amplificada pela heterogeneidade dos sistemas analisados: a distância entre os requisitos do Ada Health (confiabilidade e explicabilidade em contexto médico) e os do Character.AI (coerência de persona e memória de longo prazo em entretenimento) é suficientemente grande para sugerir que uma hierarquia de ARs, com uma AR base e ARs de segundo nível por domínio, pode ser mais adequada do que uma única AR generalista, o que constitui a extensão mais relevante indicada como trabalho futuro.

7 Considerações Finais

Neste trabalho foi apresentada a análise e síntese de uma Arquitetura de Referência para o domínio de chatbots, conduzida por meio do processo ProSA-RA de Nakagawa et al. (2014). As contribuições concretas do trabalho são as seguintes.

A primeira contribuição é a aplicação documentada do ProSA-RA ao domínio de chatbots, um domínio de software intensivo em inteligência artificial ainda pouco explorado pela literatura de arquitetura de referência. Essa aplicação expande o repertório de domínios sobre os quais o processo foi experimentado e gera lições sobre suas particularidades nesse contexto, especialmente no que diz respeito à etapa de generalização, onde a heterogeneidade dos sistemas analisados tornou mais complexa a identificação dos requisitos verdadeiramente transversais ao domínio.

A segunda contribuição é a revisão sistemática de literatura com protocolo documentado segundo o PRISMA, que produziu uma síntese de 45 estudos primários sobre arquitetura e qualidade de chatbots, identificou os doze requisitos não funcionais mais recorrentes no domínio e confirmou a lacuna central que este trabalho endereça: a ausência de ARs formalmente especificadas para sistemas conversacionais na literatura de engenharia de software.

A terceira contribuição é a AR propriamente dita, documentada em cinco visões arquiteturais segundo o modelo "4+1" de Kruchten (1995). A AR define oito componentes fundamentais, seus mecanismos de interação, os estilos arquiteturais recomendados e o mapeamento explícito entre doze requisitos não funcionais e os componentes e mecanismos que os endereçam, tornando visíveis os quatro trade-offs estruturais que qualquer instância da AR precisará resolver: escalabilidade versus tempo de resposta, auditabilidade versus confidencialidade, extensibilidade versus consistência, e manutenibilidade versus desempenho.

A quarta contribuição é a instância conceitual no domínio educacional, documentada com nível de parametrização suficiente para guiar uma implementação concreta. A instância introduz três especializações sobre a AR base (NLU com corpus educacio-

nal, Motor de Diálogo com modelagem do estudante e Base de Conhecimento organizada em grafo de conceitos por disciplina) e uma extensão própria (Módulo de Analytics Pedagógico), ilustrando o mecanismo de variabilidade que qualquer AR bem projetada deve suportar.

A rastreabilidade entre as etapas do ProSA-RA é um resultado transversal que atravessa todas as contribuições anteriores. A decisão de introduzir fila de mensagens entre Interface e Motor de Diálogo foi motivada pelo RNF de Escalabilidade, identificado em 41 dos 45 estudos da RSL como o atributo mais crítico do domínio. O padrão *Circuit Breaker* foi motivado pelos 19 estudos que o identificam explicitamente como mecanismo de resiliência para integrações externas. O Módulo de Monitoramento com padrão *Observer* foi motivado pelos 16 estudos que apontam observabilidade não intrusiva como requisito de manutenibilidade. Essa cadeia de evidência, requisito e decisão é o que distingue uma AR sistematicamente derivada de uma proposta de design ad-hoc, e é o que torna a AR evolúvel por terceiros sem dependência dos autores originais.

Para equipes de desenvolvimento de chatbots, a AR oferece um ponto de partida documentado que elimina a necessidade de tomar decisões arquiteturais fundamentais do zero a cada novo projeto. Uma equipe que adota a AR como referência concentra seu esforço na customização do domínio e na lógica de negócio, em vez de resolver continuamente os mesmos desafios infraestruturais. Os requisitos de qualidade, endereçados na fase de projeto, reduzem o risco de falhas arquiteturais que seriam muito mais custosas de corrigir em produção.

Para pesquisadores da área, a AR oferece um *baseline* documentado e sistematicamente derivado para comparação com abordagens alternativas e para validação empírica em estudos controlados. A rastreabilidade entre fontes, requisitos e componentes facilita a identificação de quais decisões arquiteturais têm maior impacto nos atributos de qualidade observados em sistemas reais.

7.1 Trabalhos Futuros

A extensão mais imediata e necessária é a **avaliação formal da AR com o framework FERA**. Os 93 itens do *checklist* de avaliação certamente identificarão pontos

de melhoria na documentação arquitetural, especialmente em relação à especificação de variabilidades, à cobertura de padrões internacionais e à adequação das visões para os diferentes grupos de *stakeholders*. A experiência de Nakagawa et al. (2014) com a SiMuS é ilustrativa: mesmo após um processo cuidadoso de derivação, a avaliação com FERA identificou problemas que não haviam sido percebidos pelos próprios autores. É provável que a AR aqui proposta apresente problemas análogos, e a aplicação do FERA é o caminho mais rigoroso para identificá-los e corrigi-los.

A segunda extensão é a **implementação concreta da instância educacional** e sua validação com usuários reais (estudantes e docentes de graduação em ciência da computação). Essa implementação testaria a viabilidade das decisões arquiteturais em condições reais de uso, e em particular verificaria se os limiares de confiança de NLU definidos conceitualmente (0,75 para a instância educacional) são adequados na prática, e se a estrutura de grafo de conceitos da Base de Conhecimento é suficiente para cobrir a variabilidade de formulações de dúvidas reais de estudantes.

A terceira extensão é a **elaboração de ARs de segundo nível** para domínios especializados. A distância arquitetural entre um chatbot de saúde (com foco em confiabilidade e explicabilidade) e um chatbot de entretenimento (com foco em criatividade e coerência de persona) é suficientemente grande para sugerir que uma hierarquia de ARs, com uma AR base genérica e ARs de segundo nível por domínio, pode ser mais adequada do que uma única AR generalista. Saúde e governo são os domínios prioritários para esse esforço, por imporem requisitos regulatórios que introduzem restrições arquiteturais não capturadas adequadamente pela AR base.

A quarta extensão é a **incorporação de orientações específicas para chatbots baseados em LLM**, endereçando os desafios arquiteturais próprios desses sistemas: controle de alucinação mediante arquiteturas RAG (Retrieval-Augmented Generation) (LEWIS et al., 2020), gerenciamento de custo de inferência, escalabilidade de modelos com bilhões de parâmetros e mecanismos de moderação de conteúdo. A AR proposta é suficientemente abstrata para acomodar LLMs como implementação do Módulo de NLG, mas o padrão RAG exige um conector específico entre o Motor de Diálogo e o banco vetorial que a Visão de Implementação atual trata como subcomponente da Base de Co-

nhecimento; uma AR de segundo nível para chatbots baseados em LLM deveria elevar esse conector ao status de componente de primeiro nível, com seus próprios requisitos de qualidade (custo de inferência, latência de recuperação e atualidade do índice vetorial).

A quinta extensão é a **automação do processo de derivação e instanciação** da AR, mediante ferramentas de suporte ao ProSA-RA que facilitem a aplicação do processo por equipes sem formação especializada em arquitetura de software, reduzindo a barreira de adoção da AR proposta.

7.2 Palavra Final

O desenvolvimento de sistemas conversacionais de qualidade é um desafio de engenharia que não se resolve apenas com o avanço dos modelos de inteligência artificial subjacentes. Modelos mais poderosos instalados em arquiteturas mal projetadas continuarão a produzir sistemas frágeis, inseguros e de difícil manutenção. A Arquitetura de Referência proposta neste trabalho é uma resposta a esse desafio: um modelo estruturado, derivado sistematicamente de evidências da literatura e de sistemas reais, documentado em forma acessível a diferentes perfis de *stakeholders* e parametrizável para domínios específicos sem perda de coerência estrutural.

Este trabalho é um convite à comunidade de engenharia de software a tratar chatbots com o mesmo rigor arquitetural que já se aplica a sistemas críticos de outros domínios, reconhecendo que a escala de adoção desses sistemas os tornou componentes centrais da infraestrutura digital contemporânea, com impacto real sobre decisões de saúde, aprendizagem, acesso a serviços públicos e experiências de milhões de usuários.

Bibliografia

BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice*. 3. ed. [S.l.]: Addison-Wesley, 2012.

CAHN, J. *CHATBOT: Architecture, Design, & Development*. [S.l.], 2017.

DALE, R. The return of the chatbots. *Natural Language Engineering*, v. 22, n. 5, p. 811–817, 2016.

DEVLIN, J. et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. [S.l.: s.n.], 2019. p. 4171–4186.

HUANG, X.; CIS, A. *Chatbot: Design, Architecture, and Applications*. [S.l.], 2021.

KITCHENHAM, B. *Procedures for Performing Systematic Reviews*. [S.l.], 2004.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. [S.l.], 2007.

KRUCHTEN, P. B. The 4+1 view model of architecture. *IEEE Software*, v. 12, n. 6, p. 42–50, 1995.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2020. p. 9459–9474.

NAKAGAWA, E. Y. et al. Consolidating a process for the design, representation, and evaluation of reference architectures. In: *2014 IEEE/IFIP Conference on Software Architecture (WICSA)*. [S.l.]: IEEE, 2014. p. 143–152.

NAKAGAWA, E. Y.; OQUENDO, F.; BECKER, M. RAModel: A reference model for reference architectures. In: *2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA-ECSA)*. [S.l.]: IEEE, 2012. p. 297–301.

PAGE, M. J. et al. PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews. *BMJ*, v. 372, p. n160, 2021.

RADFORD, A. et al. *Improving Language Understanding by Generative Pre-Training*. [S.l.], 2018. Disponível em: ⟨https://openai.com/research/language-unsupervised⟩.

SHAWAR, B. A.; ATWELL, E. Chatbots: Are they really useful? *Journal for Language Technology and Computational Linguistics*, v. 22, n. 1, p. 29–49, 2007.

VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2017. p. 5998–6008.

WEIZENBAUM, J. ELIZA: A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, v. 9, n. 1, p. 36–45, 1966.