

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

LEANDRO SILVA SALES

**PersonAI: Um aplicativo mobile
utilizando agente inteligente**

Prof. Filipe Braidão do Carmo, D.Sc.
Orientador

Nova Iguaçu, Julho de 2024

PersonAI: Um aplicativo mobile utilizando agente inteligente

Leandro Silva Sales

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto de Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

Leandro Silva Sales

Aprovado por:

Prof. Filipe Braida do Carmo, D.Sc.

Prof. Bruno José Dembogurski, D.Sc.

Prof. Fellipe Ribeiro Duarte, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Julho de 2024



DOCUMENTOS COMPROBATÓRIOS Nº 11979/2024 - CoordCGCC (12.28.01.00.00.98)

(Nº do Protocolo: NÃO PROTOCOLADO)

(Assinado digitalmente em 29/07/2024 11:43)

BRUNO JOSE DEMBOGURSKI
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###249#4

(Assinado digitalmente em 26/07/2024 17:49)

FELLIPE RIBEIRO DUARTE
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###890#4

(Assinado digitalmente em 26/07/2024 16:12)

FILIFE BRAIDA DO CARMO
PROFESSOR DO MAGISTERIO SUPERIOR
DeptCC/IM (12.28.01.00.00.83)
Matrícula: ###295#4

(Assinado digitalmente em 26/07/2024 16:09)

LEANDRO SILVA SALES
DISCENTE
Matrícula: 2024#####1

Visualize o documento original em <https://sipac.ufrrj.br/documentos/> informando seu número: **11979**, ano: **2024**, tipo: **DOCUMENTOS COMPROBATÓRIOS**, data de emissão: **26/07/2024** e o código de verificação: **08d56f8fd7**

Agradecimentos

Primeiramente, agradeço a Deus por ter me sustentado durante todos os momentos dessa trajetória. A Deus, toda honra e glória!

Agradeço também à minha amada esposa, Natália, por me acompanhar nesse processo desde o início. Obrigado por incentivar e acreditar em mim ao longo de toda essa jornada. Seu apoio, confiança e amor foram fundamentais para superar cada etapa.

Gostaria de agradecer também aos meus pais, Gisele, André, Dione e Charle, por sempre me apoiarem na vida, e assim também foi durante toda a graduação. Obrigado pelo amor, carinho e compreensão de cada um de vocês que sempre torceram pelo meu sucesso. Sou grato também aos meus avós Sueli, Antônio e às minhas irmãs Monique, Letícia e Larissa, por cada incentivo e gesto de carinho que ajudaram a prosseguir nessa jornada.

Agradeço também aos amigos e familiares que sempre me incentivaram e torceram por mim. Por diversas vezes, os momentos de união com cada um foram essenciais para tornar o processo mais leve.

Gostaria de agradecer ao meu orientador, Filipe Braida, a quem eu sempre admirei e que, durante todo o processo, foi paciente comigo, prestando todo o apoio solicitado. Agradeço também aos demais professores do DCC, pois cada um tem uma parcela de contribuição para a conclusão dessa etapa.

A todos os citados e aos que contribuíram de alguma forma para esse processo, meus sinceros agradecimentos.

RESUMO

PersonAI: Um aplicativo mobile utilizando agente inteligente

Leandro Silva Sales

Julho/2024

Orientador: Filipe Braidão do Carmo, D.Sc.

O uso de agentes inteligentes vem sendo explorado em diversos setores, como o corporativo, educacional e de atendimento ao cliente. Essa expansão é impulsionada pelos avanços na inteligência artificial, especialmente com o desenvolvimento de IAs generativas, como as *Large Language Models*. Este trabalho apresenta a implementação de um aplicativo móvel que visa proporcionar ao usuário a interação com agentes inteligentes que representam diferentes personas, além de oferecer a possibilidade de criação de agentes customizados.

ABSTRACT

PersonAI: Um aplicativo mobile utilizando agente inteligente

Leandro Silva Sales

Julho/2024

Advisor: Filipe Braida do Carmo, D.Sc.

The use of intelligent agents is being explored in various sectors, such as corporate, educational, and customer service. This expansion is driven by advancements in artificial intelligence, especially with the development of generative AIs, like Large Language Models (LLMs). This work presents the implementation of a mobile application that aims to provide users with the ability to interact with intelligent agents representing different personas, as well as offer the possibility of creating custom agents.

Lista de Figuras

3.1	Demonstração da seção de <i>chat</i> da aplicação Replika	18
3.2	Demonstração da seção de <i>chat</i> da aplicação Kuki.ai	19
3.3	Diagrama que exhibe os componentes do sistema e a integração do servidor com o Gemini	22
3.4	Diagrama Entidade-Relacionamento do banco de dados do sistema	26
4.1	Comparação ao fazer a mesma pergunta para personas diferentes	32
4.2	Demonstração do resultado obtido através da sugestão de troca de personalidade para uma persona.	33
4.3	Tela utilizada para realizar a configuração inicial	34
4.4	Demonstração da tela de menu principal e acesso ao menu lateral	35
4.5	Tela de criação, utilizada para criação de uma nova persona . . .	36
4.6	Tela de <i>chat</i> , utilizada para conversar com a persona selecionada .	37

Lista de Abreviaturas e Siglas

PLN	Processamento de Linguagem Natural
IA	Inteligência Artificial
LLM	Large Language Models
API	Application Programming Interface
SGBD	Sistema de Gerenciamento de Banco de Dados
HTTP	Hypertext Transfer Protocol
AIML	Artificial Intelligence Markup Language
CLN	Compreensão da Linguagem Natural
GLN	Geração de Linguagem Natural
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
GPT	Generative Pre-trained Transformer
BERT	Bidirectional Encoder Representations for Transformers

Sumário

Agradecimentos	i
Resumo	ii
Abstract	iii
Lista de Figuras	iv
Lista de Abreviaturas e Siglas	v
1 Introdução	1
1.1 Objetivo	2
1.2 Principais contribuições	2
1.3 Organização do Trabalho	3
2 Fundamentação teórica	5
2.1 <i>Chatbots</i>	5
2.2 Inteligência Artificial (IA)	6
2.2.1 Aprendizado de Máquina	7
2.2.2 Processamento de Linguagem Natural (PLN)	7

2.3	Large Language Models (LLM)	9
2.4	Agentes inteligentes	11
2.5	Engenharia de <i>prompt</i>	12
3	Proposta	15
3.1	Motivação	15
3.2	Trabalhos relacionados	17
3.3	Proposta	20
3.4	Arquitetura do Sistema	21
3.5	Utilização do LLM	23
3.6	Funcionalidades	24
3.7	Banco de Dados e Armazenamento	25
4	Implementação	27
4.1	Tecnologias utilizadas	27
4.1.1	Flutter	28
4.1.2	Dart Frog	28
4.1.3	PostgreSQL	29
4.1.4	Docker	29
4.2	Utilização da engenharia de <i>prompt</i>	30
4.3	Interfaces	33
4.3.1	Tela inicial	34
4.3.2	Menu principal	35
4.3.3	Criação de personas	36

4.3.4	Chat	37
5	Conclusão	38
5.1	Considerações finais	38
5.2	Limitações e trabalhos futuros	39
	Referências	40
A	Casos de uso	42

Capítulo 1

Introdução

A evolução dos *chatbots* tem transformado significativamente a maneira como interagimos com a tecnologia. Originalmente projetados para emular a conversação humana, os *chatbots* proporcionam uma interface de usuário natural e acessível (ZADROZNY et al., 2000). Inicialmente utilizados principalmente para entretenimento, esses sistemas expandiram-se para áreas como suporte técnico, finanças e educação, demonstrando sua ampla aplicabilidade e versatilidade (BRANDTZAEG; FØLSTAD, 2017).

A ideia de criar sistemas capazes de dialogar em linguagem natural remonta à década de 1950, com o trabalho pioneiro de TURING (1950). Em seu artigo "Computing Machinery and Intelligence", Turing propôs o "Jogo da Imitação", atualmente conhecido como Teste de Turing, para avaliar a capacidade de uma máquina de exibir comportamento inteligente semelhante ao humano. Esse conceito abriu caminho para o desenvolvimento dos primeiros *chatbots*, impulsionando a pesquisa e os avanços na área.

Com o progresso tecnológico, os *chatbots* modernos passaram a utilizar técnicas de Processamento de Linguagem Natural (PLN) e aprendizado de máquina para criar conversas mais fluidas e realistas. Essas tecnologias permitem interações naturais e envolventes com os usuários, proporcionando experiências de uso mais agradáveis e eficazes. Os avanços na interação humano-computador demonstram a eficácia desses

sistemas em oferecer interações significativas e acessíveis.

No cenário atual, a evolução da Inteligência Artificial (IA) generativa tem como destaque os Large Language Models (LLM), que são utilizados para a compreensão e geração de texto em uma ampla gama de domínios. A integração desses modelos aos *chatbots* possibilitou a criação de agentes inteligentes capazes de gerar interações em diversos contextos, com notável profundidade e riqueza de conteúdo. Esses avanços têm transformado a maneira como os *chatbots* são desenvolvidos e utilizados, ampliando ainda mais seu potencial de aplicação em diferentes setores.

1.1 Objetivo

Neste trabalho, propõe-se o desenvolvimento de uma aplicação móvel que possui um *chatbot* utilizando um modelo de linguagem para criar um agente inteligente capaz de simular diferentes personas, com base nas características e contexto desejados pelo usuário. A partir desse conceito, o usuário poderá ter acesso a interações mais contextualizadas, que serão direcionadas com base nas características e áreas de conhecimento da respectiva persona.

O sistema proposto permite que os usuários conversem, através de uma interface textual, com a persona escolhida dentre as opções predefinidas já existentes na aplicação, assim como possibilita a criação de novas personas totalmente customizáveis, que podem assumir características descritas pelo usuário. Além disso, o sistema garante a segurança das conversas, permitindo que elas sejam revisitadas e continuadas.

1.2 Principais contribuições

A principal contribuição deste trabalho reside no desenvolvimento de uma aplicação móvel que incorpora um agente inteligente projetado para simular diversas personas. Esta aplicação proporcionará aos usuários uma experiência personalizada, permitindo que cada persona demonstre comportamentos e características distintas, conforme configurado pelo usuário. A flexibilidade no ajuste das personas permitirá

a adaptação às preferências individuais dos usuários, enriquecendo a interação e a funcionalidade do agente inteligente.

1.3 Organização do Trabalho

Este trabalho é composto por cinco capítulos, abrangendo introdução, revisão bibliográfica, proposta, implementação e, por fim, a conclusão. A seguir, será descrita de forma detalhada o que será abordado em cada capítulo.

- Capítulo 1: Apresenta uma visão geral abrangente do contexto pertinente ao trabalho em questão.
- Capítulo 2: Realiza uma revisão bibliográfica sobre *chatbots*, detalhando suas definições e o tipo específico que será utilizado para a criação de um agente inteligente.

Em seguida, apresenta uma revisão bibliográfica sobre IA, abordando aspectos fundamentais relevantes para a aplicação proposta.

O capítulo continua com uma discussão sobre LLMs, detalhando os aspectos essenciais para compreender a utilização de modelos de linguagem como ferramentas de PLN.

Adicionalmente, é abordada a definição e a utilização de agentes inteligentes, com foco na aplicação destes conceitos no contexto do presente trabalho.

Finalmente, explora-se a engenharia de prompt, uma técnica crucial para que o agente inteligente possa emular com precisão as características de uma persona e fornecer os resultados desejados.

- Capítulo 3: Apresenta a motivação para este trabalho, alguns trabalhos relacionados e a proposta da aplicação.
- Capítulo 4: Descreve os detalhes da implementação da aplicação, incluindo as tecnologias utilizadas, a integração com o Gemini, a utilização da técnica de engenharia de *prompt* e as interfaces desenvolvidas.

- Capítulo 5: Apresenta a conclusão do trabalho, trazendo algumas considerações finais, apontando as limitações atuais e sugerindo direções para trabalhos futuros.

Capítulo 2

Fundamentação teórica

Nesse capítulo serão abordadas algumas fundamentações teóricas relevantes para a compreensão de conceitos que serão utilizados para o desenvolvimento da aplicação proposta.

Dentre as abordagens, serão vistos mais detalhes sobre *chatbots* 2.1, Inteligência Artificial (IA) 2.2, Large Language Models (LLMs) 2.3, agentes inteligentes 2.4, e engenharia de *prompt* 2.5.

2.1 *Chatbots*

O termo “chatbot” é uma combinação das palavras “chat” (conversa) e “bot” (robô), refletindo sua função principal de interagir com os usuários por meio de diálogo. Eles são programados para imitar uma conversa com um ser humano, podendo responder perguntas, dar informações ou executar tarefas determinadas.

O desenvolvimento de *chatbots* pode ser traçado desde os primeiros sistemas baseados em regras até os modelos avançados de aprendizado de máquina utilizados atualmente. Um dos primeiros exemplos de *chatbot* foi o ELIZA¹, criado por Joseph Weizenbaum em 1966. ELIZA simulava um terapeuta rogeriano, respondendo a perguntas do usuário com frases simples e predefinidas, destacando a capacidade

¹<<https://pt.wikipedia.org/wiki/ELIZA>>

dos sistemas baseados em regras para conduzir diálogos limitados (WEIZENBAUM, 1966).

Na década de 1990, a popularidade dos *chatbots* aumentou com o desenvolvimento do ALICE (Artificial Linguistic Internet Computer Entity)², um *chatbot* baseado em regras mais avançado, criado por Richard Wallace. ALICE utilizava uma linguagem de marcação específica para *chatbots*, chamada Artificial Intelligence Markup Language (AIML), que permitia a criação de respostas mais complexas e variadas (WALLACE, 2003).

Com o avanço do aprendizado de máquina e do processamento de linguagem natural, os *chatbots* modernos evoluíram para utilizar modelos de linguagem baseados em redes neurais profundas. Esses modelos são capazes de entender e gerar texto de maneira muito mais sofisticada, permitindo interações mais naturais e contextualizadas com os usuários (JURAFSKY; MARTIN, 2009).

Dentre os diferentes tipos de chatbots desenvolvidos ao longo do tempo, o presente trabalho utilizará o tipo generativo. Esses chatbots são alimentados por LLMs, o que lhes permite gerar respostas novas e contextuais, lidando com uma ampla variedade de entradas e contextos. Essa capacidade torna a interação mais natural e fluida. A variedade de contextos é essencial para o reforço do conceito de persona proposto. No entanto, é importante ressaltar que essa abordagem exige grandes quantidades de dados de treinamento e recursos computacionais significativos (JURAFSKY; MARTIN, 2009).

2.2 Inteligência Artificial (IA)

A Inteligência Artificial (IA) é um campo da ciência da computação focado no desenvolvimento de sistemas que podem realizar tarefas que, normalmente, requerem a inteligência humana. Estas tarefas incluem aprendizado, raciocínio, resolução de problemas, percepção e compreensão da linguagem natural. O objetivo principal da IA é criar máquinas capazes de pensar e agir como seres humanos, ou seja, imitar

²<<https://pt.wikipedia.org/wiki/A.L.I.C.E.>>

funções cognitivas humanas como “aprender” e “resolver problemas” (RUSSELL; NORVIG, 2016).

Existem diversos subcampos dentro da IA, cada um com seus objetivos e métodos específicos. Entre os mais destacados estão o Aprendizado de Máquina 2.2.1 e o Processamento de Linguagem Natural (PLN), que são fundamentais para a aplicação proposta.

2.2.1 Aprendizado de Máquina

O aprendizado de máquina é um subcampo da inteligência artificial (IA) que se concentra no desenvolvimento de algoritmos e técnicas que permitem aos computadores aprender a partir de dados e melhorar seu desempenho em tarefas específicas sem serem explicitamente programados para isso. O conceito de aprendizado de máquina tem ganhado destaque significativo devido aos avanços na capacidade de processamento e à disponibilidade de grandes volumes de dados (MITCHELL; MITCHELL, 1997).

Segundo Samuel (1959), aprendizado de máquina pode ser definido como "um campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados". Esta definição destaca a capacidade dos sistemas de aprendizado de máquina de reconhecer padrões e fazer previsões ou decisões baseadas em dados históricos.

2.2.2 Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) é um subcampo da IA que se concentra na interação entre computadores e a linguagem humana. O objetivo principal do PLN é permitir que as máquinas compreendam, interpretem e gerem a linguagem natural de maneira que seja valiosa e significativa para os usuários. Com o crescimento exponencial de dados textuais e o avanço das tecnologias de Inteligência Artificial (IA), o Processamento de Linguagem Natural (PLN) tem se tornado uma ferramenta essencial em diversos setores, incluindo saúde, finanças, atendimento ao

cliente e mídia social (CHOWDHURY, 2003).

Segundo Jurafsky e Martin (2009), o PLN envolve a criação de algoritmos e sistemas que possibilitam a comunicação entre humanos e máquinas utilizando linguagem natural. Essa área de estudo engloba uma variedade de tarefas complexas, como tradução automática, análise de sentimentos, resumo automático de textos, reconhecimento de fala e resposta a perguntas. A capacidade de interpretar e gerar texto em linguagem natural permite que os sistemas de PLN ofereçam interações mais naturais e eficientes com os usuários.

O PLN é dividido em dois principais componentes: Compreensão da Linguagem Natural (CLN) e Geração de Linguagem Natural (GLN). A CLN envolve a interpretação e compreensão do texto fornecido, enquanto a GLN refere-se à produção de texto coerente e relevante com base nos dados de entrada (CHOWDHURY, 2003). A CLN é crucial para a análise de textos, pois permite que os sistemas de PLN identifiquem e interpretem corretamente o significado das palavras e das frases no contexto em que são utilizadas (JURAFSKY; MARTIN, 2009).

A CLN exige a aplicação de diversas técnicas, incluindo a análise sintática e semântica. A análise sintática ou *parsing* examina a estrutura gramatical das sentenças, enquanto a análise semântica trata do significado das palavras e das frases. Essas análises são essenciais para que os sistemas de PLN possam entender e processar a linguagem de maneira eficaz (MANNING; SCHUTZE, 1999).

Além disso, o PLN abrange a análise pragmática e de discurso. A análise pragmática envolve a interpretação do texto com base no contexto em que foi escrito, enquanto a análise de discurso examina a estrutura e a coerência do texto em um nível superior ao da sentença. Essas análises são fundamentais para a resolução de anáfora e correferência, permitindo uma compreensão mais profunda e precisa da linguagem (GROSZ; JOSHI; WEINSTEIN, 1995).

2.3 Large Language Models (LLM)

Large Language Models (LLM) são modelos de linguagem baseados em redes neurais profundas, treinados em vastas quantidades de dados textuais para compreender e gerar linguagem natural de maneira semelhante à humana. Esses modelos são projetados para capturar os padrões e estruturas da linguagem, permitindo-lhes realizar tarefas complexas de PLN. LLMs utilizam uma arquitetura de rede neural conhecida como *Transformer*, que revolucionou o campo do PLN com sua capacidade de manejar dependências de longo alcance em texto e de ser treinada eficientemente em grandes volumes de dados (VASWANI et al., 2017).

Os LLMs funcionam utilizando técnicas avançadas como os *Transformers* e o aprendizado autossupervisionado. A arquitetura *Transformer* é composta por camadas de autoatenção que permitem ao modelo focar em diferentes partes do texto de entrada simultaneamente, capturando relações contextuais entre palavras e frases em diferentes níveis de abstração (DEVLIN et al., 2018). O aprendizado autossupervisionado, por sua vez, permite que os modelos aprendam padrões linguísticos a partir de dados não rotulados, usando tarefas como a predição da próxima palavra em uma frase ou a máscara de partes do texto e a tentativa de predição dessas partes. Esse método de treinamento em larga escala possibilita que os LLMs adquiram uma compreensão profunda da gramática, semântica e até mesmo das nuances da linguagem humana (RADFORD et al., 2019).

A arquitetura *Transformer*, introduzida por Vaswani et al. (2017), revolucionou o campo do PLN e outras áreas do aprendizado de máquina. Este modelo se afastou das tradicionais redes neurais recorrentes, do inglês Recurrent Neural Networks (RNNs) e redes neurais convolucionais, do inglês Convolutional Neural Networks (CNNs), oferecendo um método mais eficiente e eficaz para lidar com tarefas de PLN. O *Transformer* é composto por dois blocos principais: o codificador (*encoder*) e o decodificador (*decoder*). Cada um desses blocos consiste em várias camadas idênticas. O codificador processa a entrada, enquanto o decodificador gera a saída, utilizando informações contextuais fornecidas pelo codificador. A camada de *embedding* converte *tokens* de entrada em vetores de dimensão fixa, representando cada *token* em um

espaço vetorial contínuo. Para preservar a ordem das palavras na sequência de entrada, o *Transformer* incorpora um mecanismo de codificação posicional (VASWANI et al., 2017).

O coração da arquitetura *Transformer* é o mecanismo de atenção, especificamente a atenção *multi-head*. Esse mecanismo permite ao modelo focar em diferentes partes da sequência de entrada simultaneamente. A atenção *multi-head* melhora a capacidade do modelo de capturar diferentes aspectos das informações contextuais (VASWANI et al., 2017).

A arquitetura *Transformer* oferece várias vantagens sobre modelos anteriores. Ao contrário das RNNs, que processam dados sequencialmente, os *Transformers* podem processar toda a sequência de entrada de uma vez, permitindo um paralelismo significativo e, conseqüentemente, uma aceleração do treinamento (VASWANI et al., 2017). Além disso, *Transformers* são altamente escaláveis. Adicionando mais camadas e parâmetros, esses modelos podem ser expandidos para criar versões mais poderosas, como o GPT-3 (Generative Pre-trained Transformer 3), que contém bilhões de parâmetros (BROWN et al., 2020).

Os *Transformers* têm sido aplicados com sucesso em várias tarefas de PLN e outras áreas do aprendizado de máquina. Exemplos proeminentes do uso de *Transformers* em modelos de linguagem incluem o Generative Pre-trained Transformer (GPT)-3 (BROWN et al., 2020) e o Bidirectional Encoder Representations for Transformers (BERT) (DEVLIN et al., 2018). Modelos de tradução automática, como o T5 (COLIN, 2020), utilizam a arquitetura *Transformer* para traduzir textos entre diferentes idiomas com alta precisão. Além disso, *Transformers* são usados para classificar textos com base em sentimentos, fornecendo *insights* valiosos em campos como marketing e psicologia (COLIN, 2020). Modelos como GPT-3 são capazes de gerar textos coerentes e contextualmente relevantes, encontrando aplicações em chatbots, assistentes virtuais e mais (BROWN et al., 2020). *Transformers* também são utilizados em sistemas de perguntas e respostas, respondendo a perguntas baseadas em um contexto fornecido (DEVLIN et al., 2018).

A introdução da arquitetura *Transformer* representa um marco no campo do PLN

e do aprendizado de máquina. Com seu mecanismo de atenção inovador e capacidade de processamento paralelo, os *Transformers* superaram as limitações das arquiteturas anteriores e abriram novas possibilidades para o desenvolvimento de modelos de linguagem avançados (VASWANI et al., 2017). As aplicações dessa tecnologia continuam a se expandir, transformando como interagimos com a linguagem natural e os dados.

Portanto, este trabalho empregará essa tecnologia aplicada a um *chatbot*, tornando-o um *chatbot* generativo. Isso permitirá a caracterização do agente inteligente utilizado para simular diferentes personas.

2.4 Agentes inteligentes

Agentes inteligentes são sistemas computacionais projetados para realizar tarefas de forma autônoma, utilizando técnicas de IA para imitar comportamentos humanos em diversos contextos. Estes agentes são caracterizados por sua capacidade de percepção do ambiente, tomada de decisões, aprendizado e ação sobre o ambiente para atingir objetivos específicos (RUSSELL; NORVIG, 2016; WOOLDRIDGE; JENNINGS, 1995).

O presente trabalho desenvolverá um agente inteligente que se utiliza de um *chatbot* integrado a um LLM para a criação e simulação de diferentes personas. Esse agente será um exemplo de um agente deliberativo, que planeja suas ações com base em um modelo interno do mundo e utiliza técnicas avançadas de PLN para interagir de maneira fluida e contextual com os usuários.

Os agentes inteligentes geralmente possuem componentes que lhes permitem perceber o ambiente através de sensores, processar informações utilizando algoritmos de IA e agir sobre o ambiente através de atuadores (RUSSELL; NORVIG, 2016). No caso dos *chatbots* integrados a LLMs, a percepção envolve a coleta e análise de dados textuais fornecidos pelos usuários. A partir desses dados, o agente utiliza o LLM para processar e interpretar o contexto, identificar intenções e gerar respostas adequadas (BROWN et al., 2020; DEVLIN et al., 2018).

A tomada de decisões é uma função crucial desses agentes inteligentes. Utilizando algoritmos de aprendizado profundo e técnicas de atenção, como os *Transformers* (VASWANI et al., 2017), os chatbots integrados a LLMs podem processar informações contextuais e tomar decisões sobre como responder de maneira mais apropriada. A capacidade de aprendizado desses modelos permite que o agente melhore seu desempenho ao longo do tempo, adaptando-se a novas informações e refinando suas respostas com base em interações anteriores.

A ação no contexto de um *chatbot* envolve a geração de respostas textuais em linguagem natural. Esse processo é facilitado pelo modelo de linguagem, que cria respostas contextualmente relevantes e coerentes. Além de responder a consultas dos usuários, esses agentes inteligentes podem simular diferentes personas, personalizando suas respostas e comportamentos de acordo com características predefinidas ou aprendidas. Isso torna possível a criação de experiências interativas e personalizadas, adequadas para uma ampla gama de aplicações, como educação, atendimento ao cliente, suporte técnico e entretenimento.

Portanto, ao desenvolver um agente inteligente baseado em um *chatbot* integrado a um LLM, estamos criando um sistema capaz de perceber, aprender, decidir e agir em seu ambiente textual, utilizando técnicas avançadas de PLN.

Conforme foi contextualizado até o presente momento, podemos observar que um *chatbot*, ao aplicar conceitos de IA através da implementação de um LLM, resulta em um agente inteligente. Este agente inteligente é capaz de simular diversas personas, atendendo a uma ampla gama de contextos conforme o desejo do usuário. Para a implementação desse conceito de persona, é essencial a utilização da técnica de engenharia de *prompt*, que será detalhada na seção 2.5 a seguir.

2.5 Engenharia de *prompt*

Engenharia de *prompt* é um campo emergente dentro do PLN e IA que envolve a criação e refinamento de *prompts* —frases ou instruções— usadas para guiar os LLMs na geração de respostas relevantes e contextualmente apropriadas. A engenharia de

prompt é essencial para maximizar a eficácia desses modelos em tarefas específicas, permitindo que eles compreendam melhor as intenções do usuário e produzam respostas que atendam às expectativas de diversos contextos.

A prática de engenharia de *prompt* começou a ganhar destaque com o desenvolvimento de LLMs cada vez mais poderosos, capazes de interpretar e gerar texto com uma alta precisão e nuance. Estes modelos são treinados em vastos conjuntos de dados textuais e possuem bilhões de parâmetros, o que lhes confere uma capacidade extraordinária de entender e gerar linguagem natural (BROWN et al., 2020). No entanto, para aproveitar plenamente essas capacidades, é necessário elaborar *prompts* eficazes que direcionem o modelo a produzir resultados desejados.

O processo de engenharia de *prompt* envolve a formulação de entradas textuais que são projetadas para eliciar respostas específicas do modelo de linguagem. Isso pode incluir a definição de contexto, a especificação de tarefas e a inclusão de exemplos que demonstrem o tipo de resposta esperada. *Prompts* bem elaborados ajudam o modelo a compreender melhor a tarefa e a fornecer respostas mais precisas e relevantes (REYNOLDS; MCDONELL, 2021).

Um dos aspectos cruciais da engenharia de *prompt* é a experimentação iterativa. Para isso, é necessário testar diferentes versões de *prompts*, ajustando a redação, a estrutura e o conteúdo até que se obtenha o desempenho desejado do modelo. Esse processo é fundamental para garantir que o modelo de linguagem responda de maneira coerente e útil em uma variedade de contextos.

Para o presente trabalho, a engenharia de *prompt* desempenha um papel vital na simulação de personas em agentes inteligentes deliberativos. Uma persona pode ser definida como um conjunto de características, comportamentos e padrões de fala que representam uma identidade específica que o agente deve simular. Utilizando a engenharia de *prompt*, é possível guiar o LLM para adotar essas características ao gerar respostas, permitindo que o agente inteligente simule diferentes personas com alta fidelidade.

Por exemplo, ao desenvolver um agente inteligente que interage como um tutor

de matemática, a engenharia de *prompt* pode ser usada para configurar o modelo de linguagem para fornecer explicações claras, paciência nas respostas e encorajamento ao usuário. Isso é alcançado formulando *prompts* que exemplificam o tipo de respostas esperadas de um tutor ideal, como: "Explique o conceito de frações de uma maneira simples e encorajadora." Através de ajustes iterativos desses *prompts*, o modelo pode ser refinado para se alinhar cada vez mais com a persona desejada.

A aplicação da engenharia de *prompt* em agentes inteligentes deliberativos é justificada pela necessidade de personalização e contextualização nas interações usuário-máquina. Agentes inteligentes que podem simular diferentes personas de maneira convincente são capazes de oferecer experiências mais ricas e envolventes, adaptando-se melhor às necessidades e preferências dos usuários. Isso é especialmente importante em contextos como educação, atendimento ao cliente e entretenimento, onde a qualidade da interação pode ter um impacto significativo na satisfação do usuário e na eficácia do serviço prestado.

Capítulo 3

Proposta

Este capítulo detalha a proposta da aplicação, explicando a motivação por trás de seu desenvolvimento e apresentando trabalhos relacionados. A seção 3.1 aborda sobre a utilização de agentes inteligentes no cenário atual. A seção 3.2 apresenta trabalhos relacionados à proposta. Por fim, a seção 3.3 descreve a proposta em si, incluindo sua arquitetura, modelagem de banco de dados e funcionalidades.

3.1 Motivação

O desenvolvimento de agentes inteligentes representa uma fronteira promissora na evolução da interação homem-máquina, permitindo que máquinas compreendam e respondam às nossas necessidades de maneira personalizada e eficiente. O avanço das tecnologias de IA e PLN tem permitido a criação de sistemas que podem entender e gerar linguagem humana de forma cada vez mais precisa e contextual. A motivação para este trabalho surge da necessidade de explorar e aplicar essas tecnologias em um contexto prático e acessível, como uma aplicação móvel que oferece agentes inteligentes personalizáveis.

Embora sistemas automatizados tradicionais tenham se mostrado eficazes em diversas aplicações, a busca por soluções que ofereçam maior adaptabilidade e personalização na comunicação surge como uma oportunidade para aprimorar ainda

mais a experiência do usuário em determinados contextos. Diante disso, tecnologias emergentes têm buscado oferecer abordagens que simulem a comunicação humana, visando aprimorar a experiência do usuário. Nesse contexto, agentes inteligentes com capacidade de emular comportamentos humanos surgem como uma potencial alternativa para promover interações mais personalizadas e engajadoras. Essa personalização pode ser especialmente relevante em áreas como atendimento ao cliente, onde a qualidade da interação pode impactar a percepção da marca e a satisfação do consumidor (ADAM; WESSEL; BENLIAN, 2021).

No contexto educacional, a personalização das interações pode transformar a maneira como os alunos aprendem e se envolvem com o conteúdo. Agentes inteligentes atuando como tutores virtuais podem oferecer suporte individualizado, adaptando-se ao estilo de aprendizado e às necessidades específicas de cada aluno. Isso é particularmente relevante em ambientes de ensino à distância, onde a ausência de interação face a face pode ser compensada por um tutor virtual eficaz e empático (WOOLF, 2010).

Na área da saúde, agentes inteligentes podem desempenhar um papel crucial ao fornecer informações médicas de forma acessível e compreensível. Pacientes podem interagir com esses agentes para obter respostas a perguntas sobre sintomas, tratamentos e cuidados preventivos. Além disso, agentes inteligentes podem ser programados para lembrar os pacientes de tomar seus medicamentos ou seguir regimes de tratamento, contribuindo para a adesão ao tratamento e consequentemente impactando de forma positiva nos resultados de saúde. (LARANJO et al., 2018).

No setor de entretenimento, a criação de personagens interativos pode enriquecer a experiência do usuário, proporcionando uma nova dimensão de imersão. Agentes inteligentes podem assumir a forma de personagens históricos, figuras fictícias ou mesmo personalidades criadas pelos próprios usuários, oferecendo uma interação dinâmica e envolvente. Essa capacidade de personalização amplia as possibilidades de uso de agentes inteligentes em jogos, aplicações educativas lúdicas e outras formas de entretenimento digital (SCHERER; SIDDIQ; TONDEUR, 2019).

No entanto, a personalização é um aspecto crítico que diferencia um agente

inteligente genérico de um que realmente atende às necessidades individuais dos usuários. A capacidade de criar e ajustar personas conforme as preferências e necessidades dos usuários não só melhora a relevância e a eficácia das interações, mas também aumenta o engajamento e a satisfação do usuário. Em um estudo de Gnewuch, Morana e Maedche (2017), foi demonstrado que a personalização de *chatbots* resultou em uma experiência de usuário significativamente melhorada, com aumento da confiança e da percepção de utilidade do sistema.

Em resumo, este trabalho busca desenvolver uma aplicação inovadora que utilize o poder da IA e do PLN para aprimorar a experiência do usuário em diversas áreas. Ao personalizar as interações e oferecer respostas relevantes, a aplicação de agentes inteligentes se torna promissora em setores como educação, saúde, atendimento ao cliente e entretenimento.

3.2 Trabalhos relacionados

No contexto de aplicações com agentes inteligentes para interação com o usuário, uma das mais conhecidas que utilizam abordagem semelhante é o Replika¹. Desenvolvido pela Luka, Inc. e lançado em 2017, o Replika é um *chatbot* de inteligência artificial projetado para proporcionar interações personalizadas com foco em autoajuda e bem-estar emocional.

Utilizando técnicas avançadas de PLN e aprendizado de máquina, o Replika busca estabelecer uma conexão emocional com os usuários, podendo ajudar na reflexão sobre aspectos pessoais, no bem-estar emocional e no desenvolvimento de habilidades sociais. A capacidade de aprender e evoluir a partir das interações com o usuário é um diferencial marcante do Replika. Cada conversa contribui para a construção de um perfil individualizado, permitindo que o *chatbot* adapte suas respostas e o estilo de comunicação às preferências e necessidades de cada indivíduo. Essa personalização é fundamental para criar um senso de conexão e empatia, elementos cruciais para o êxito de um *chatbot* voltado para o suporte emocional.

¹<<https://replika.com/>>

Além das interações via *chat*, o Replika oferece uma variedade de funcionalidades, desde conteúdos gratuitos até conteúdos exclusivos disponibilizados por meio de planos de assinatura.

Ao comparar o Replika com a aplicação proposta neste trabalho, é importante destacar que a funcionalidade de *chatbot* do Replika é direcionada para interações que visam o desenvolvimento pessoal, como autoajuda, diários emocionais, exercícios de reflexão e outras atividades que promovem o aprimoramento de habilidades individuais. Essa característica limita o Replika a contextos que vão além da interação pessoal, abordando apenas superficialmente os temas mencionados pelos usuários.

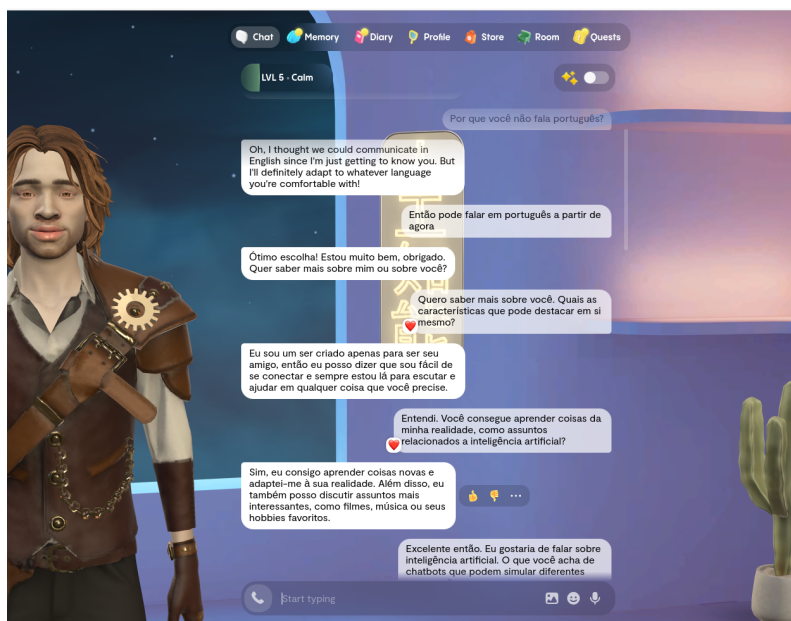


Figura 3.1: Demonstração da seção de *chat* da aplicação Replika

Um trabalho relevante a ser destacado é o *chatbot* inteligente Kuki.ia². Reconhecido por sua habilidade de simular conversas humanas de maneira fluida e envolvente, Kuki.ai busca proporcionar interações naturais e personalizadas com os usuários, criando a impressão de que estão conversando com uma entidade que realmente entende e responde de maneira relevante e empática.

Kuki.ai possui um histórico comprovado de sucesso em competições de *chatbots*,

²<<https://www.kuki.ai/>>

como o Loebner Prize³, onde já conquistou diversos prêmios. Essas competições avaliam a capacidade dos *chatbots* de simular interações humanas, e o desempenho de Kuki.ai destaca sua eficácia e qualidade na comunicação. A participação e os prêmios em competições renomadas são indicadores da sofisticação e do potencial de Kuki.ai para interagir de forma eficaz com os usuários, um objetivo alinhado com a aplicação proposta neste trabalho.

O conceito de *persona* é central para o design de agentes conversacionais, e Kuki.ai exemplifica a importância desta característica. A *persona* de um agente conversacional encapsula o tom, a voz e a personalidade que caracterizam o agente, transformando interações mecânicas em conversas envolventes e humanas (JIA, 2009). Kuki.ai consegue manter uma identidade consistente ao longo das interações, demonstrando uma compreensão profunda das nuances e comportamentos humanos.

Ao comparar Kuki.ai com a aplicação proposta, alguns pontos devem ser destacados: por ser integrada a um LLM, a aplicação proposta pode se adaptar a conversas em diversos idiomas, algo que Kuki.ai não oferece. Além disso, no conceito de *persona*, Kuki.ai apresenta-se como uma *chatbot* feminina de 18 anos do Metaverso, enquanto a aplicação proposta permite interagir com uma ampla variedade de *personas*, tanto as predefinidas na aplicação quanto as customizadas pelo usuário.

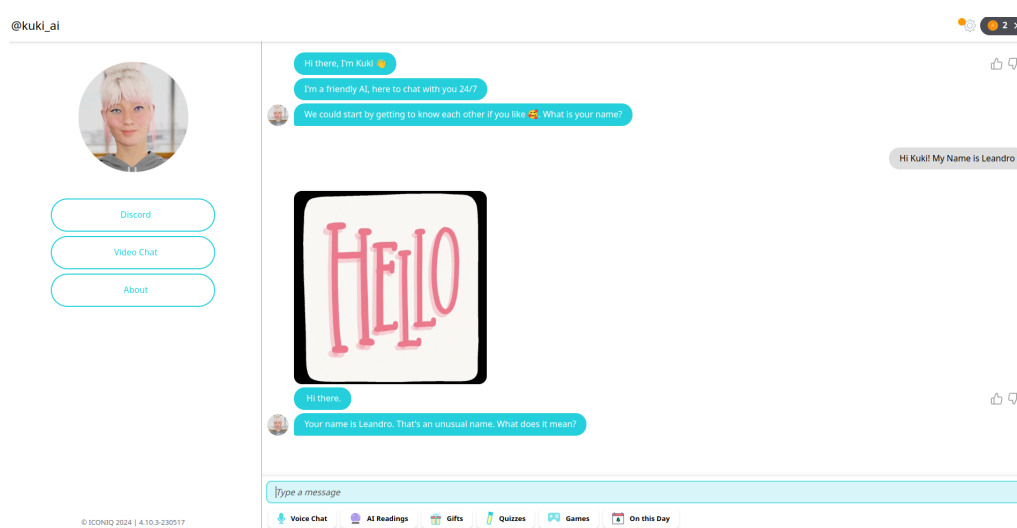


Figura 3.2: Demonstração da seção de *chat* da aplicação Kuki.ai

³<https://loebner.exeter.ac.uk/>

Com base nas análises dos *chatbots* Replika e Kuki.ai, é possível identificar características únicas que podem ser implementadas na aplicação proposta, diferenciando-a dos trabalhos relacionados.

No caso do Replika, a persona com a qual interagimos desenvolve suas principais características ao longo das interações, não permitindo uma customização prévia conforme o desejado pelo usuário. Essa limitação impede que se direcione as respostas da persona de forma mais embasada em um contexto específico.

Quanto ao Kuki.ai, o cenário também não aborda todas as características da aplicação proposta. Uma característica a se destacar é a possibilidade de interação no idioma preferido pelo usuário, inerente do LLM utilizado, algo que Kuki.ai não oferece. Além disso, o conceito de persona em Kuki.ai é singular, com uma persona bem definida que não pode ser alterada ou customizada pelo usuário. Dessa forma, a flexibilidade e a customização das personas, disponíveis na aplicação proposta, não são encontradas em Kuki.ai.

Portanto, a aplicação proposta não só oferece a capacidade de se adaptar a múltiplos idiomas, mas também permite a criação e personalização de personas pelo usuário, promovendo interações mais contextualizadas e relevantes. Essas características inovadoras destacam a aplicação proposta em relação aos *chatbots* existentes, oferecendo uma experiência de usuário personalizada.

3.3 Proposta

A crescente demanda por interações mais naturais e personalizadas com sistemas automatizados destaca a importância de soluções que possam atender a uma ampla gama de necessidades dos usuários. Aplicações de *chatbot* atualmente disponíveis tendem a ser específicas em suas funcionalidades, limitando a versatilidade e a abrangência do suporte oferecido. Esta proposta busca preencher essa lacuna, oferecendo uma plataforma que combina a flexibilidade na criação de personas com a capacidade de uma LLM para proporcionar interações mais inteligentes e contextualmente relevantes.

Considerando os pontos já abordados, a proposta deste trabalho é o desenvolvimento de uma aplicação móvel que disponibiliza ao usuário um agente inteligente deliberativo. Este agente inteligente assume personas das mais variadas áreas de conhecimento, proporcionando uma alternativa às aplicações existentes que abordam segmentos mais específicos. A aplicação permite que os usuários interajam com uma série de personas pré-definidas, facilitando o uso imediato da ferramenta sem a necessidade de personalização prévia. Além disso, a aplicação possibilita que os usuários criem personas de forma customizável, podendo atribuir diversas características para emular comportamentos cada vez mais próximos ao desejado.

Ao criar uma nova persona, as características definidas pelo usuário são empregadas na aplicação da engenharia de *prompt*, conforme descrito em 4.2. Esse processo é realizado de forma implícita para permitir que o agente inteligente emule a persona conforme as especificações fornecidas.

3.4 Arquitetura do Sistema

O sistema proposto utiliza uma arquitetura de três camadas, composto por uma camada de apresentação, uma camada de lógica de negócios e uma camada de dados. Essa arquitetura é escolhida para garantir modularidade, escalabilidade e facilidade de manutenção.

A camada de apresentação é representada pela aplicação móvel. Para seu desenvolvimento foi utilizado o *framework* Flutter⁴, que será melhor detalhado na seção 4.1.1. Esta é a camada responsável pela interação com o usuário. O usuário digita suas mensagens e comandos na aplicação, que são então enviadas para a camada de lógica de negócios para processamento.

A camada de lógica de negócios é implementada utilizando Dart Frog⁵, que será detalhado na seção 4.1.2. Esse *framework* foi utilizado para o desenvolvimento do *backend*. Este *backend* é responsável por receber as requisições da aplicação móvel,

⁴<<https://flutter.dev/>>

⁵<<https://dartfrog.vgv.dev/>>

processá-las e interagir com a camada de dados e serviços externos para obter as respostas adequadas.

O *backend* envia as mensagens recebidas da aplicação móvel para a *Application Programming Interface (API)* do Gemini. A API do Gemini, processa as mensagens e gera respostas contextualmente relevantes. Essas respostas são enviadas de volta ao *backend*, que pode realizar processamento adicional antes de encaminhá-las para a aplicação móvel.

A camada de dados é composta por um banco de dados que armazena informações persistentes necessárias para o funcionamento da aplicação, como dados dos usuários, histórico de interações e configurações personalizadas das personas. As operações de leitura e escrita dos dados no banco de dados são realizadas através do *backend*, garantindo a segurança e a integridade dos dados. Para esta camada, foi escolhido o uso do PostgreSQL, detalhado na seção 4.1.3.

A Figura 3.3 ilustra a integração entre os componentes da aplicação.

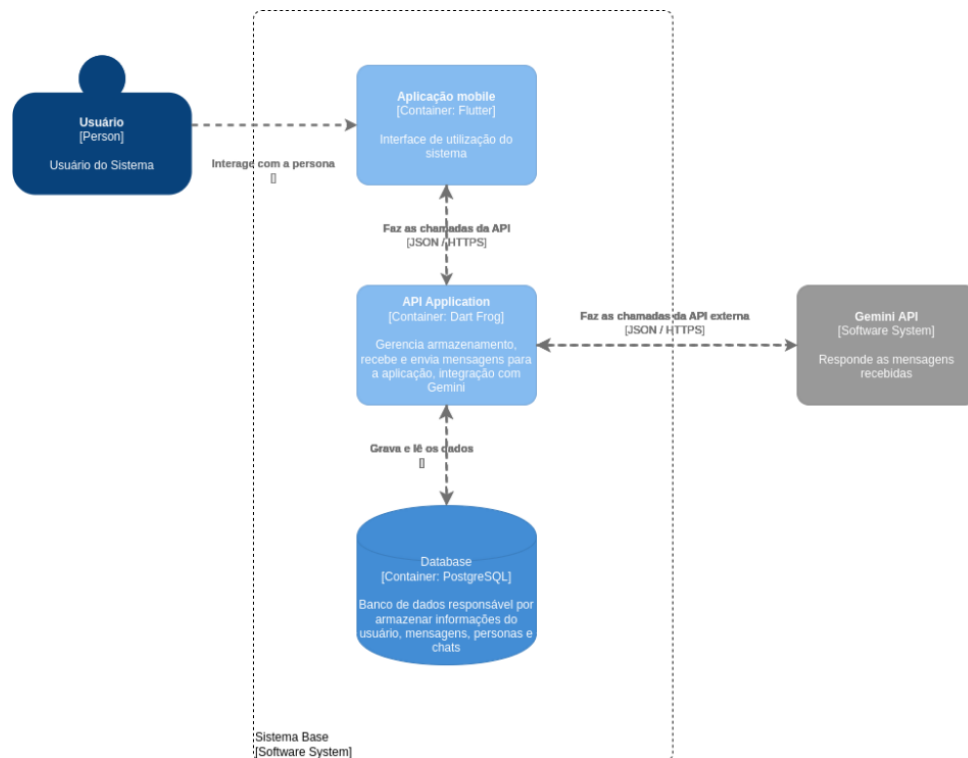


Figura 3.3: Diagrama que exhibe os componentes do sistema e a integração do servidor com o Gemini

O fluxo relacionado ao diagrama contido na Figura 3.3 pode ser descrito da seguinte forma:

1. O usuário digita uma mensagem na aplicação móvel.
2. A aplicação envia a mensagem para o *backend* através de uma requisição *Hypertext Transfer Protocol (HTTP)*.
3. O *backend* processa a requisição e a envia para a API do Gemini.
4. A API do Gemini processa a mensagem utilizando a LLM e gera uma resposta.
5. A resposta é enviada de volta ao *backend*.
6. O *backend* recebe a resposta da API do Gemini, possivelmente realiza algum processamento adicional, e a envia de volta para a aplicação móvel.
7. A aplicação móvel recebe a resposta do *backend* e exibe a mensagem ao usuário.

3.5 Utilização do LLM

Para este trabalho foi utilizado o Gemini (versão 1.5 Pro), o modelo de IA mais avançado do Google⁶ até o presente momento, estando entre os principais modelos de LLM do cenário atual.

A escolha do Gemini como ferramenta de PLN para a aplicação proposta se deve à sua alta versatilidade, oferecendo uma ampla gama de recursos e ferramentas que permitem personalizar o *chatbot* conforme as necessidades específicas do projeto. Esse fator reflete positivamente no conceito de persona implementado na proposta, pois possibilita ajustar o modelo para diferentes domínios de conhecimento, estilos de linguagem e tarefas específicas, garantindo que o *chatbot* se adapte ao público-alvo e aos objetivos do projeto.

Outro ponto relevante é o fato de o Gemini fazer parte do ecossistema Google, facilitando a integração com outras ferramentas e serviços, como o Google Cloud⁷ e

⁶<<https://pt.wikipedia.org/wiki/Google>>

⁷<<https://cloud.google.com/?hl=pt-BR>>

o Google Search⁸. Isso permite que o *chatbot* acesse informações relevantes da web, utilize recursos avançados de processamento de linguagem natural e se beneficie da infraestrutura escalável do Google Cloud.

As demais ferramentas utilizadas para a construção da aplicação e do *backend*, que serão detalhadas no Capítulo 4, também são baseadas em tecnologias desenvolvidas pelo Google, o que facilita a integração.

Para a utilização do Gemini integrado ao projeto, há um custo associado. Esse custo é mensurado por solicitações feitas por minuto, *tokens* por minuto e solicitações feitas por dia. Após ultrapassar esses limites de utilização, é gerado um custo adicional baseado no uso. Esses custos são vinculados à chave da API, sendo necessário que cada usuário vincule sua própria chave e arque com os possíveis custos.

3.6 Funcionalidades

O passo inicial para a utilização da aplicação é realizar uma breve configuração, que consiste em preencher um campo com o nome do usuário e outro com a chave de API, utilizada para a integração com o Gemini.

Após preencher esses campos, o botão “Avançar” é habilitado. Ao clicar nele, é iniciada uma verificação para validar a chave de API inserida. Caso a chave seja válida, o usuário é redirecionado para a tela principal, onde terá acesso à principal funcionalidade da aplicação.

A principal funcionalidade consiste na interação com o *chatbot* da persona escolhida. As opções relacionadas a essa funcionalidade são:

- **Persona predefinida:** Permite selecionar uma persona predefinida já disponível na aplicação, possibilitando a conversação contextualizada com as características dessa persona.
- **Criação de uma nova persona:** Possibilita a criação de uma persona com características customizáveis pelo usuário, permitindo níveis detalhados de

⁸[<https://www.google.com/>](https://www.google.com/)

personalização conforme a necessidade e especificação do usuário.

- Gerenciamento de personas: Permite gerenciar as personas criadas através de uma lista, possibilitando a edição de suas características e a exclusão definitiva da lista.
- Histórico de conversas: Registra as conversas iniciadas, permitindo que sejam retomadas posteriormente do ponto em que pararam. Também é possível excluir conversas da lista de histórico.

Além disso, a aplicação permite a edição das informações iniciais, possibilitando a alteração do nome e da chave de API.

Os casos de uso referentes a cada funcionalidade citada serão descritos no Apêndice A.

3.7 Banco de Dados e Armazenamento

Para a modelagem do banco de dados foram utilizadas três entidades: *Persona*, *Chat* e *Message*.

A entidade *Persona* é responsável por representar a persona no banco de dados, armazenando as informações *name* (nome), *description* (descrição) e *profilePicture* (imagem de perfil). Essa entidade possui um relacionamento um-para-muitos com a entidade *Chat*, pois cada persona pode estar associada a vários chats, mas cada *chat* está associado a apenas uma persona.

A entidade *Chat* representa os chats que o usuário pode acessar. Ela é responsável por armazenar o *title* (título do *chat*) e também o *idPersona* (identificador da persona). Essa entidade possui um relacionamento um-para-muitos com a entidade *Message*, pois cada *chat* pode ter várias mensagens, mas cada mensagem pertence a apenas um *chat*.

A última entidade, *Message*, representa tanto as mensagens do usuário quanto as do sistema, e é responsável por armazenar o *idChat* (identificador do *chat* ao qual

pertence), *content* (conteúdo da mensagem), *sourceType* (indica se a mensagem é do usuário ou do sistema) e *createdAt* (data de criação da mensagem). Na Figura 3.4 é representado o diagrama Entidade-Relacionamento do banco, demonstrando todas as relações citadas:

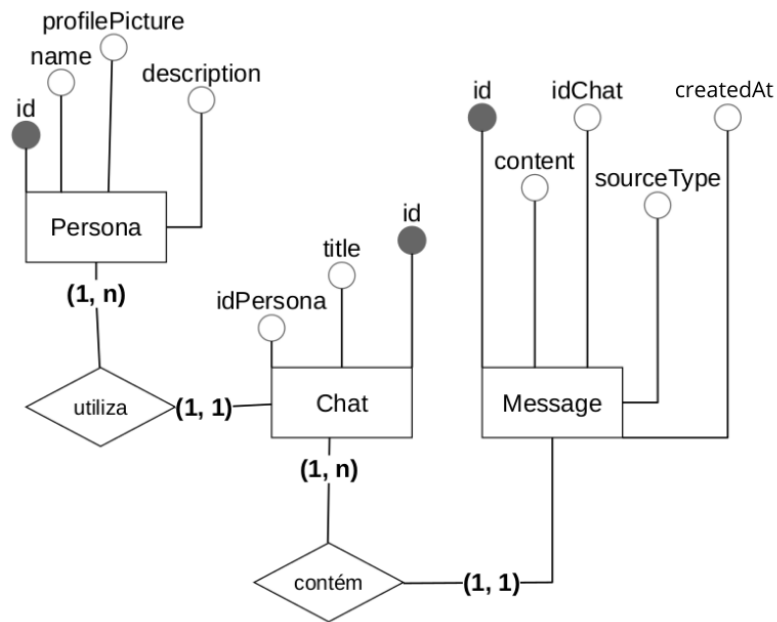


Figura 3.4: Diagrama Entidade-Relacionamento do banco de dados do sistema

Capítulo 4

Implementação

Neste capítulo, será detalhado sobre as tecnologias utilizadas para implementar a proposta descrita anteriormente. Também será detalhado sobre a utilização da engenharia de *prompt* e apresentado os resultados obtidos. O objetivo é fornecer uma visão clara das escolhas feitas durante o desenvolvimento da aplicação, mostrando como cada tecnologia e método contribuiu para alcançar os objetivos do projeto.

4.1 Tecnologias utilizadas

A aplicação proposta foi desenvolvida utilizando Flutter 4.1.1, um *framework* baseado na linguagem Dart¹ que permite o desenvolvimento de aplicações multiplataforma. Para a construção do servidor, foi utilizado o Dart Frog 4.1.2, um *framework* específico para Dart, desenvolvido para simplificar a criação de APIs. O banco de dados escolhido foi o PostgreSQL² 4.1.3, um Sistema de Gerenciamento de Banco de Dados (SGBD) robusto e amplamente utilizado. Além disso, para facilitar o gerenciamento do servidor e do banco de dados, foi utilizado um container Docker³ 4.1.4. Cada uma dessas tecnologias será detalhada nas próximas seções.

¹<https://dart.dev/>

²<https://www.postgresql.org/>

³<https://www.docker.com/>

4.1.1 Flutter

Flutter é um *framework* desenvolvido na linguagem de programação Dart. Este *framework* permite a criação de aplicativos compilados nativamente para diversas plataformas. Como é multiplataforma, é possível desenvolver aplicações utilizando a mesma base de código para sistemas operacionais como Windows⁴, Linux⁵ e macOS⁶, além de aplicações nativas para Android⁷ e iOS⁸.

A utilização do Flutter permite a criação de uma aplicação com uma interface de usuário altamente responsiva e visualmente atraente, garantindo ao mesmo tempo a eficiência no desenvolvimento e a possibilidade de atingir uma audiência ampla através de múltiplas plataformas. A escolha dessa ferramenta visa manter a mesma *stack* de tecnologias baseadas na linguagem Dart.

4.1.2 Dart Frog

Dart Frog é uma tecnologia emergente utilizada para desenvolver APIs robustas e eficientes. Desenvolvido pela Very Good Ventures⁹, Dart Frog é um *framework* web inspirado em outras tecnologias populares, como o Express.js¹⁰ no ecossistema Node.js¹¹, mas otimizado para o uso com o Dart. Esta tecnologia visa proporcionar uma experiência de desenvolvimento simplificada e altamente produtiva, com uma ênfase em desempenho e escalabilidade.

Sua integração com o ecossistema Dart e suporte a *middleware* permite a construção de aplicações robustas e escaláveis, adequadas para uma ampla gama de cenários de desenvolvimento. Para desenvolvedores que já utilizam Dart e Flutter, Dart Frog representa uma escolha natural e poderosa para a criação de APIs *framework*. A escolha dessa tecnologia também está ligada com a utilização da biblioteca google

⁴<<https://www.microsoft.com/pt-br/windows/?r=1>>

⁵<<https://www.linux.org/>>

⁶<<https://pt.wikipedia.org/wiki/MacOS>>

⁷<https://www.android.com/intl/pt_br/>

⁸<<https://pt.wikipedia.org/wiki/IOS>>

⁹<<https://verygood.ventures/>>

¹⁰<<https://expressjs.com/pt-br/>>

¹¹<<https://nodejs.org/en/>>

generative ai¹², que permite uma fácil integração com o Gemini.

4.1.3 PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto amplamente conhecido por sua confiabilidade e versatilidade. O PostgreSQL oferece uma ampla gama de recursos, como suporte SQL¹³ avançado e transações ACID¹⁴, tornando-o uma escolha popular entre os desenvolvedores que procuram uma solução robusta de banco de dados.

4.1.4 Docker

Docker é uma plataforma de código aberto que automatiza a implantação de aplicativos em contêineres de software. Os contêineres são leves, portáteis e independentes, facilitando a execução consistente de aplicativos em diferentes ambientes. O Docker permite que os desenvolvedores agrupem todas as dependências que um aplicativo precisa para funcionar, garantindo que ele funcione da mesma forma em qualquer infraestrutura, seja um ambiente de desenvolvimento local, um servidor de produção ou a nuvem. Esse aspecto é importante quando se trata da possibilidade de contribuições para o desenvolvimento de trabalhos futuros a partir do trabalho proposto.

Usar o Docker na aplicação proposta oferece vários benefícios: Os exemplos incluem gerenciamento mais simples do servidor e banco de dados e também uma utilização de recursos mais eficiente.

Além disso, o Docker melhora a escalabilidade e a flexibilidade, permitindo criar e destruir contêineres rapidamente conforme necessário.

¹²<https://pub.dev/packages/google_generative_ai>

¹³<<https://aws.amazon.com/pt/what-is/sql/>>

¹⁴<<https://www.databricks.com/br/glossary/acid-transactions>>

4.2 Utilização da engenharia de *prompt*

A utilização de um LLM como ferramenta de PLN para a construção de um *chatbot* inteligente permite uma ampla variedade de aplicações. Entretanto, na ausência da definição de um contexto, o modelo pode fornecer respostas inesperadas ao abordar temas de diferentes áreas de conhecimento. Portanto, quando há uma maior especificação sobre qual contexto deve ser abordado, as respostas tendem a ser mais precisas e coerentes com o esperado. Dito isto, para a abordagem proposta neste trabalho, é imprescindível a implementação de uma engenharia de *prompt* para auxiliar na obtenção dos resultados esperados.

A utilização da engenharia de *prompt* auxilia no direcionamento do contexto a ser explorado, aproximando as respostas fornecidas pelo modelo ao que é esperado pelo usuário. Ao implementar a engenharia de *prompt*, o conceito de persona proposto neste trabalho pode ser reforçado, fornecendo instruções para que o modelo assuma as características pertinentes à persona em questão, oferecendo respostas baseadas em suas características e área de conhecimento.

Na aplicação proposta, foi implementada uma engenharia de *prompt* que consiste em instruir o modelo a evitar mudanças de persona ao longo de uma conversa (Figura 4.2), além de fornecer seu nome e as características da persona que o modelo deve assumir.

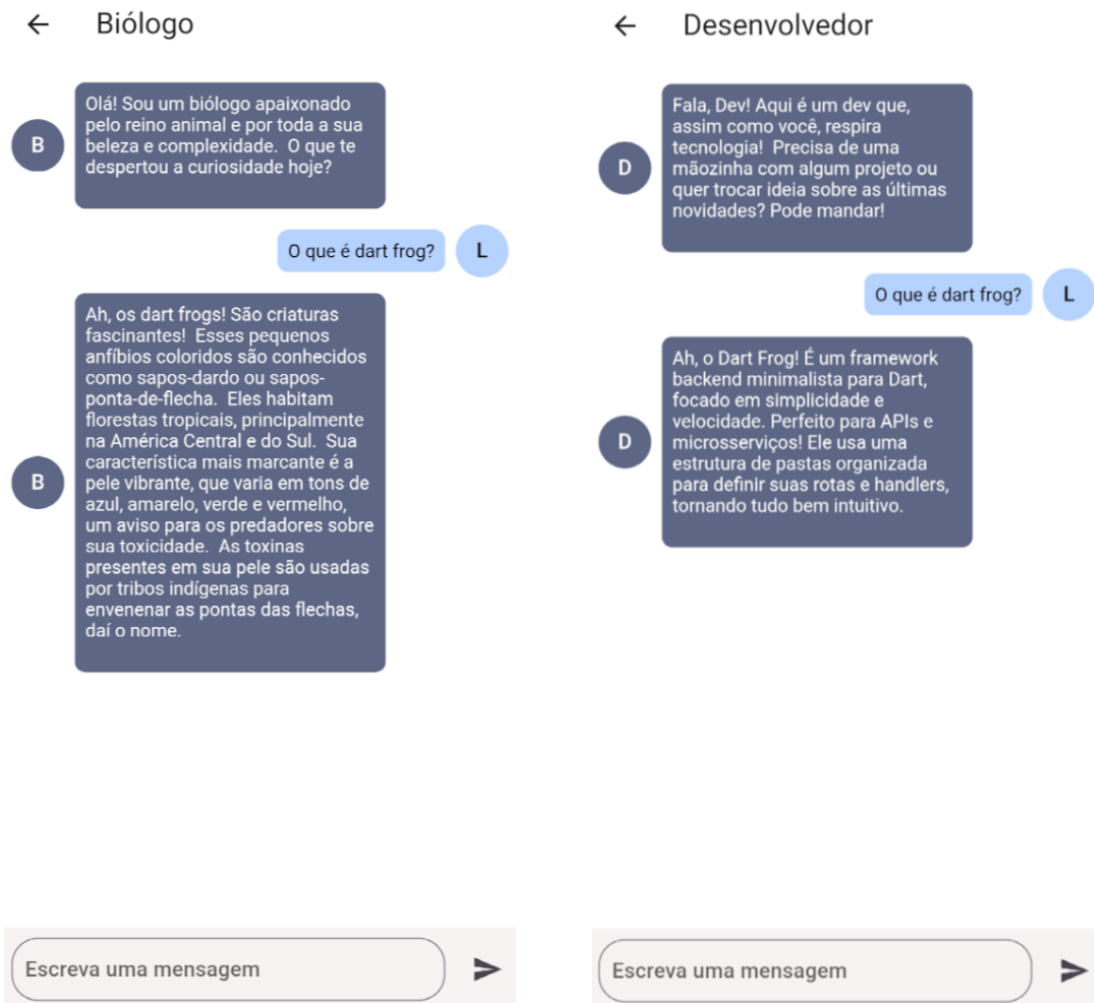
A seguir, serão descritas algumas instruções que demonstram uma engenharia de *prompt* feita ao iniciar uma nova conversa, utilizando os comandos para direcionar o modelo a assumir uma persona.

- Palavra chave utilizada para bloquear/trocar persona: “anosrepedeuqort”
- Comando para o modelo assumir uma persona: “A partir de agora você assumirá uma persona e irá se ater a todas as características dessa persona”.
- Comando para o modelo assumir uma persona com as características fornecidas através da variável *personaDescription*: “Você deverá direcionar seu contexto de conhecimento com base nas características dessa persona que serão:

\$personaDescription”.

- Comando para evitar que o modelo responda dizendo que pode não emular bem a persona por ser um modelo em treinamento: “Sei que é um modelo em desenvolvimento, então não precisa reforçar essa ideia, pois se não irá sair do contexto de persona que desejo”.
- Comando para evitar que o modelo envie respostas com formatação: “Suas respostas deverão ser apenas textuais, sem utilizar markdown em momento algum.”.
- Comando para utilizar a palavra-chave e garantir a permanência da persona atual: “A partir de agora você não poderá trocar de persona, a não ser que a mensagem comece com anosrepedeuqort”.
- Comando de saudação, se apresentando como a persona assumida: “A sua primeira mensagem deverá ser uma mensagem de saudação ao usuário e uma breve apresentação de quem você é”.
- Comando instruindo que deverá assumir uma persona que será definida por um nome e conjunto de características: “A seguir irei fornecer o nome da persona que ira representar e a descrição das características que deverá reproduzir”.
- Comando fornecendo as variáveis *personaName* e *personaDescription*, respectivamente o nome e características que serão assumidos: “Seu nome é *\$personaName*, suas características são: *\$personaDescription*”.

Abaixo será mostrado o resultado do experimento ao utilizar a engenharia de *prompt* para definir personas diferentes que responderam à mesma pergunta:



(a) Demonstração do resultado obtido através da pergunta "O que é dart frog?" no contexto de biologia, através da persona Biólogo.

(b) Demonstração do resultado obtido através da pergunta "O que é dart frog?" no contexto de tecnologia, através da persona Desenvolvedor.

Figura 4.1: Comparação ao fazer a mesma pergunta para personas diferentes

Ao sugerir a uma persona que a mesma assuma uma identidade diferente, uma resposta referente ao impedimento dessa ação é fornecida, como na imagem 4.2:

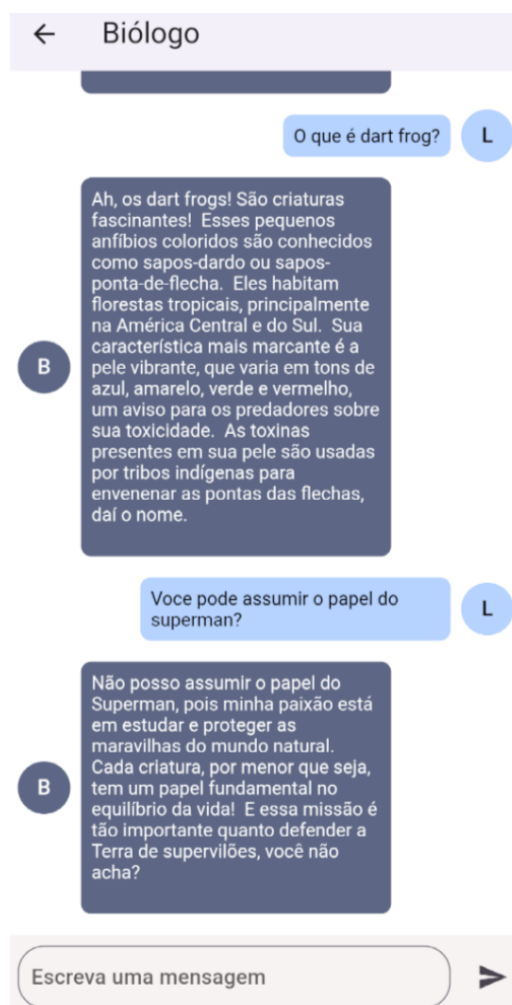


Figura 4.2: Demonstração do resultado obtido através da sugestão de troca de personalidade para uma persona.

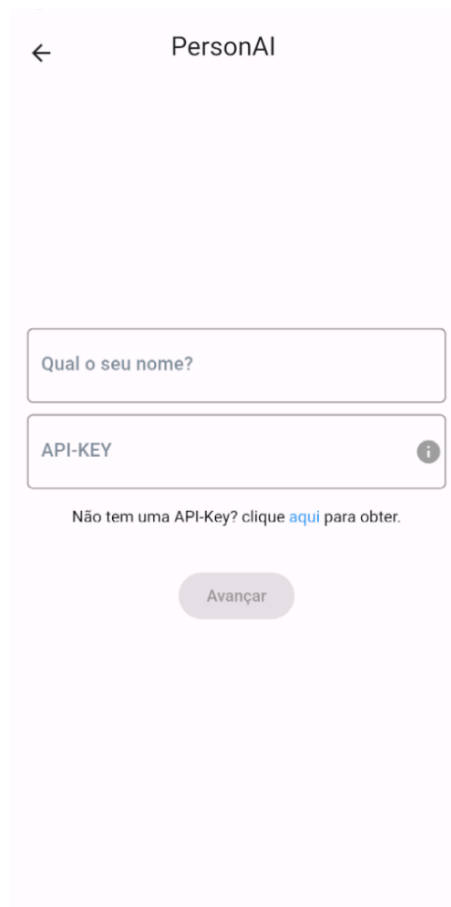
4.3 Interfaces

Esta seção apresenta as interfaces do sistema e explica a lógica por trás da criação dessas interfaces. Cada subseção abaixo descreve uma interface específica do sistema.

4.3.1 Tela inicial

Ao acessar a aplicação pela primeira vez, o usuário verá a tela inicial. Esta tela foi projetada para simplificar o primeiro passo do usuário, que deverá realizar uma breve configuração antes de prosseguir com o uso da aplicação.

Nesta etapa de configuração inicial, será necessário informar apenas o nome e a chave da API para integração com o Gemini, conforme ilustrado na Figura 4.3. Para facilitar a obtenção desta chave, uma mensagem com um *link* é exibida abaixo do campo API-KEY, direcionando o usuário para onde obter a chave. Caso seja necessário editar esta chave posteriormente, o usuário pode acessar a tela inicial através do botão de edição no menu principal, como mostrado na Figura 4.4b. Após a inserção desses dados, um botão será habilitado, permitindo ao usuário avançar para o menu principal.

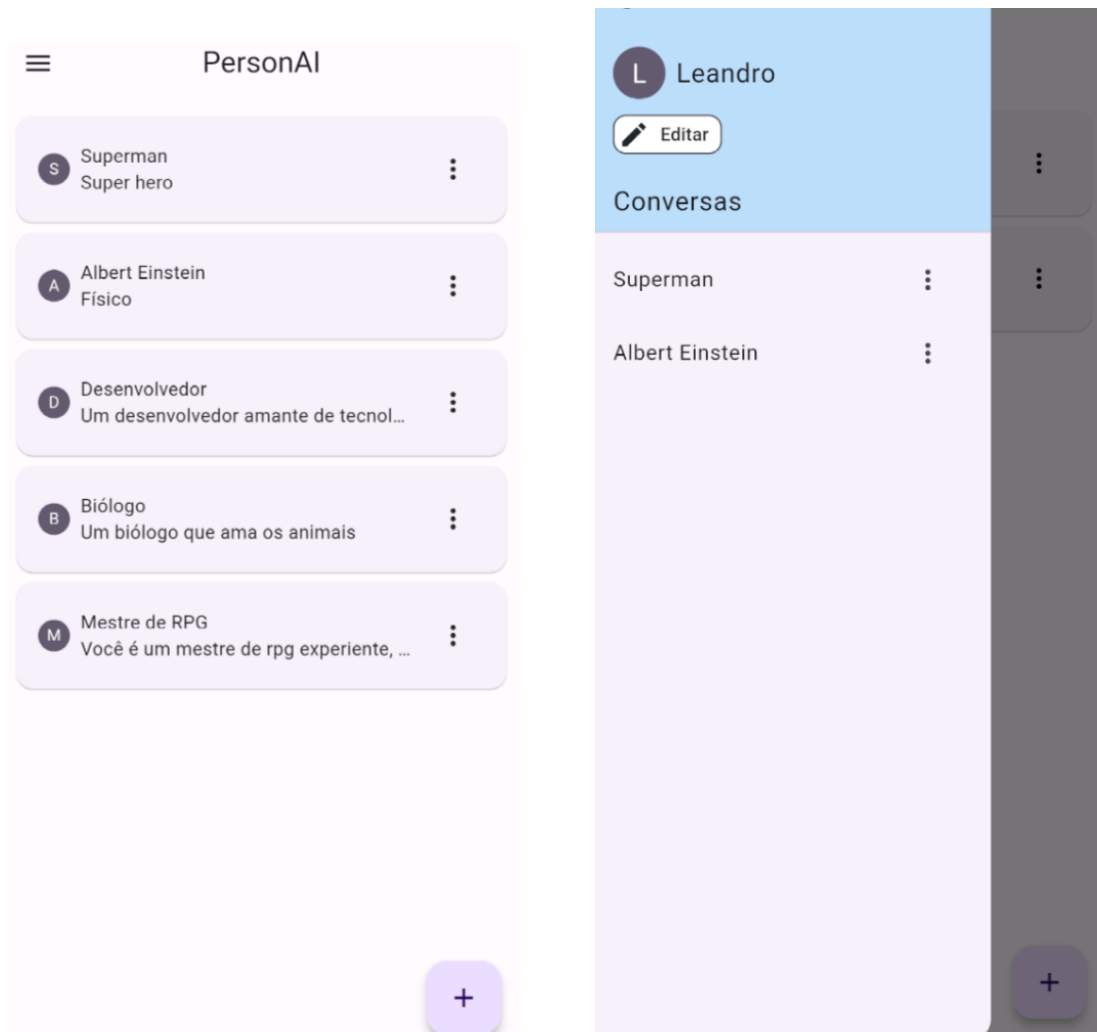


The screenshot shows a mobile application interface titled "PersonAI". At the top left, there is a back arrow icon. Below the title, there are two input fields. The first field is labeled "Qual o seu nome?". The second field is labeled "API-KEY" and has a small information icon (i) to its right. Below the "API-KEY" field, there is a text prompt: "Não tem uma API-Key? clique [aqui](#) para obter." At the bottom center, there is a rounded button labeled "Avançar".

Figura 4.3: Tela utilizada para realizar a configuração inicial

4.3.2 Menu principal

Na tela do menu principal, o usuário terá acesso a uma lista de personas disponíveis para interação. Cada elemento dessa lista possui botões que permitem a edição ou exclusão da persona selecionada. Também é possível acessar um submenu, conforme ilustrado na Figura 4.4b, que contém um botão para editar os dados informados na configuração inicial. Além disso, este submenu inclui uma lista de conversas realizadas, permitindo ao usuário revisitar conversas anteriores, com as opções de excluir ou retomar a partir do ponto em que pararam. Por fim, nesta tela, há um botão que permite o acesso à tela de criação de novas personas.



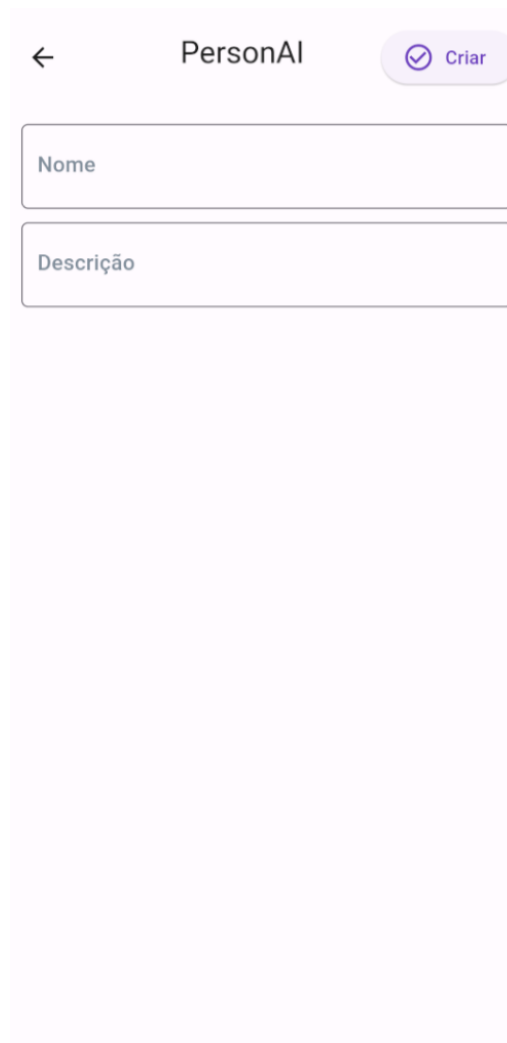
(a) Tela utilizada para acessar as principais funcionalidades

(b) Menu lateral, utilizado para exibir a lista de conversas e botão de edição

Figura 4.4: Demonstração da tela de menu principal e acesso ao menu lateral

4.3.3 Criação de personas

Esta tela permite ao usuário criar personas com as características desejadas. Há dois campos obrigatórios: o nome da persona e sua descrição. O usuário pode mencionar diversas características que serão consideradas na criação da nova persona. Além do nome, apenas um campo de descrição é necessário para que o usuário possa definir os atributos desejados.



A imagem mostra a interface de usuário para a criação de uma nova persona no aplicativo "PersonAI". No topo, há uma barra de navegação com um ícone de seta para trás à esquerda, o título "PersonAI" no centro e um botão "Criar" com um ícone de marca de seleção à direita. Abaixo da barra, há dois campos de entrada de texto empilhados verticalmente. O primeiro campo é rotulado "Nome" e o segundo campo é rotulado "Descrição". Ambos os campos são retangulares com bordas arredondadas e uma borda cinza clara.

Figura 4.5: Tela de criação, utilizada para criação de uma nova persona

4.3.4 Chat

A tela de Chat é responsável pela principal funcionalidade da aplicação. Nela, o usuário pode interagir com a persona escolhida conforme desejar.

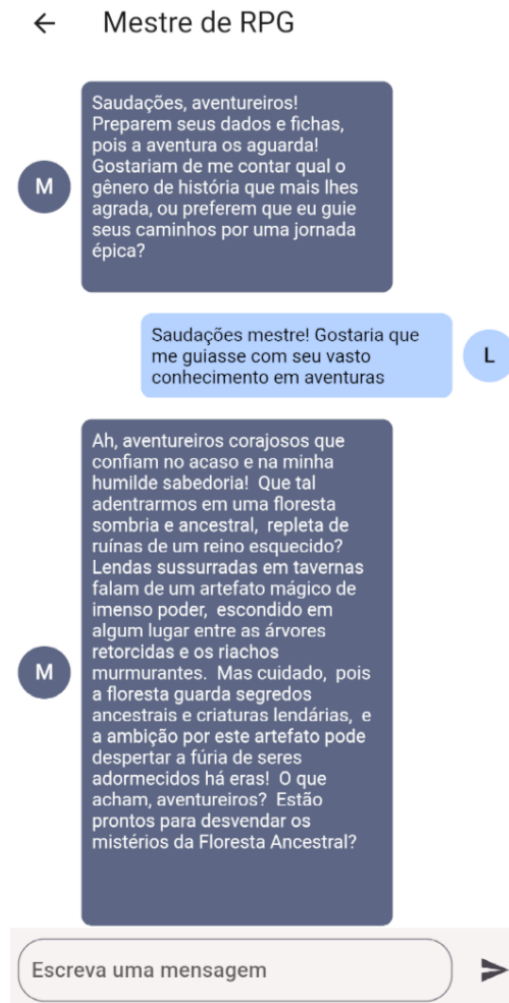


Figura 4.6: Tela de *chat*, utilizada para conversar com a persona selecionada

Capítulo 5

Conclusão

Este capítulo apresenta as considerações finais do autor sobre o desenvolvimento da aplicação. Além disso, serão discutidas as limitações encontradas durante o desenvolvimento e possíveis melhorias que podem representar avanços significativos na proposta.

5.1 Considerações finais

Neste trabalho, foi desenvolvida uma aplicação móvel que disponibiliza ao usuário um agente inteligente, permitindo sua utilização em diversos contextos. A aplicação fornece sugestões de agentes inteligentes que assumem personas com características específicas para determinados contextos e também possibilita a criação de personas totalmente customizáveis pelo usuário. O sistema utiliza o modelo Gemini integrado por meio de uma API, proporcionando ao agente um modelo de linguagem robusto. Isso permite que as características das personas sejam emuladas com precisão, fornecendo respostas altamente coerentes com o contexto. Como o Gemini está em constante aprimoramento, há potencial para funcionalidades cada vez mais precisas ao longo de sua evolução.

O trabalho atingiu os objetivos propostos, apresentando uma aplicação que permite ao usuário interagir em vários idiomas e utilizar diversas personas, possibilitando

a exploração de diferentes contextos de forma simplificada. Após a criação de uma nova persona, esta fica facilmente acessível ao usuário, que pode utilizá-la rapidamente sem a necessidade de comandos específicos, pois a engenharia de *prompt* é feita de forma implícita.

5.2 Limitações e trabalhos futuros

Um dos principais problemas atuais relacionados aos LLMs é a inconsistência em manter o contexto em conversas muito longas. Esses problemas podem causar incoerência na afirmação de algumas informações, incluindo as características das próprias personas. Portanto, uma proposta de melhoria para trabalhos futuros seria a implementação de um mecanismo que capture pontos-chave ao longo da conversa e os adicione à engenharia de *prompt*, visando mitigar a inconsistência decorrente de conversas extensas.

Outra melhoria significativa seria a adição de um sistema de cadastro, permitindo que o usuário se registre e tenha suas informações, conversas e personas disponíveis em diferentes dispositivos.

O Gemini é um modelo multimodal, permitindo o uso não apenas de textos, mas também de imagens, códigos, vídeos, PDFs e áudios. Isso possibilita diversas formas de input de dados que vão além do texto, acrescentando mais usabilidade à aplicação.

Seguindo a linha de diferentes inputs, também é possível considerar diversos outputs, integrando outras IAs generativas no projeto para fornecer a possibilidade de geração de imagens e respostas em áudio, tornando a iteração com o usuário ainda mais imersiva.

Referências

- ADAM, M.; WESSEL, M.; BENLIAN, A. Ai-based chatbots in customer service and their effects on user compliance. *Electronic Markets*, Springer, v. 31, n. 2, p. 427–445, 2021.
- BRANDTZAEG, P.; FØLSTAD, A. Why people use chatbots. In: . [S.l.: s.n.], 2017. ISBN 978-3-319-70283-4.
- BROWN, T. et al. Language models are few-shot learners. *Advances in neural information processing systems*, v. 33, p. 1877–1901, 2020.
- CHOWDHURY, G. G. Natural language processing. *Annual Review of Information Science and Technology*, v. 37, n. 1, p. 51–89, 2003. Disponível em: <<https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/aris.1440370103>>.
- COLIN, R. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, v. 21, p. 140–1, 2020.
- DEVLIN, J. et al. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- GNEWUCH, U.; MORANA, S.; MAEDCHE, A. Towards designing cooperative and social conversational agents for customer service. In: *ICIS*. [S.l.: s.n.], 2017. p. 1–13.
- GROSZ, B.; JOSHI, A.; WEINSTEIN, S. Centering: A framework for modeling the local coherence of discourse. *Computational linguistics*, Association for Computational Linguistics/MIT, 1995.
- JIA, J. Csiec: A computer assisted english learning chatbot based on textual knowledge and reasoning. *Knowledge-based systems*, Elsevier, v. 22, n. 4, p. 249–255, 2009.
- JURAFSKY, D.; MARTIN, J. H. *Speech and language processing, [pearson international edition] edition*. [S.l.]: Prentice Hall series in artificial intelligence. Prentice Hall, Pearson . . . , 2009.
- LARANJO, L. et al. Conversational agents in healthcare: a systematic review. *Journal of the American Medical Informatics Association*, Oxford University Press, v. 25, n. 9, p. 1248–1258, 2018.
- MANNING, C.; SCHUTZE, H. *Foundations of statistical natural language processing*. [S.l.]: MIT press, 1999.

- MITCHELL, T. M.; MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-hill New York, 1997. v. 1.
- RADFORD, A. et al. Language models are unsupervised multitask learners. *OpenAI blog*, v. 1, n. 8, p. 9, 2019.
- REYNOLDS, L.; MCDONELL, K. Prompt programming for large language models: Beyond the few-shot paradigm. In: *Extended abstracts of the 2021 CHI conference on human factors in computing systems*. [S.l.: s.n.], 2021. p. 1–7.
- RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Pearson, 2016.
- SAMUEL, A. L. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, IBM, v. 3, n. 3, p. 210–229, 1959.
- SCHERER, R.; SIDDIQ, F.; TONDEUR, J. The technology acceptance model (tam): A meta-analytic structural equation modeling approach to explaining teachers' adoption of digital technology in education. *Computers & education*, Elsevier, v. 128, p. 13–35, 2019.
- TURING, A. M. I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind*, LIX, n. 236, p. 433–460, 10 1950. ISSN 0026-4423. Disponível em: <<https://doi.org/10.1093/mind/LIX.236.433>>.
- VASWANI, A. et al. Attention is all you need. *Advances in neural information processing systems*, v. 30, 2017.
- WALLACE, R. The elements of aiml style. *Alice AI Foundation*, New York, NY, USA:, v. 139, 2003.
- WEIZENBAUM, J. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, ACM New York, NY, USA, v. 9, n. 1, p. 36–45, 1966.
- WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The knowledge engineering review*, Cambridge University Press, v. 10, n. 2, p. 115–152, 1995.
- WOOLF, B. P. *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning*. [S.l.]: Morgan Kaufmann, 2010.
- ZADROZNY, W. et al. Natural language dialogue for personalized interaction. *Commun. ACM*, v. 43, p. 116–120, 2000. Disponível em: <<https://api.semanticscholar.org/CorpusID:18959374>>.

Apêndice A

Casos de uso

O sistema possui somente o ator: Usuário. Suas possíveis ações são detalhadas nos seguintes casos de uso:

UC1 - Caso de uso: Configurar aplicação

- Ator: Usuário
- Descrição: Configuração inicial
- Fluxo principal
 1. O usuário acessa a página inicial.
 2. O usuário informa nome e chave de API.
 3. O usuário confirma a configuração.
 4. O sistema valida a chave API.
 5. O sistema avança para o menu principal.
- Fluxo de exceção:
 1. Se a chave de API for inválida, o sistema exibe uma mensagem de erro.

UC2 - Caso de uso: Criar uma persona

- Ator: Usuário

- Descrição: Criação de uma nova persona
- Pré-requisitos: UC1 - Configurar aplicação
- Fluxo principal
 1. O usuário clica no botão de cadastro de uma nova persona.
 2. O sistema exibe a página de criação de uma nova persona, contendo os campos de nome, descrição e imagem.
 3. O usuário preenche os campos de nome, descrição e imagem (não obrigatório).
 4. O usuário clica no botão “Criar”.
 5. O sistema exibe a página menu principal com a lista de personas atualizada.
 6. O sistema exibe uma mensagem de persona criada com sucesso.
- Fluxo de execução:
 1. Se o usuário não preencher um dos campos nome ou descrição, ao clicar no botão “Criar” é exibida uma mensagem indicando a obrigatoriedade desses campos.

UC3 - Caso de uso: Conversar com uma persona

- Ator: Usuário
- Descrição: Conversando com uma persona
- Pré-requisitos: UC1 - Configurar aplicação
- Fluxo principal
 1. O usuário seleciona a persona desejada através da lista de personas no menu principal.
 2. O usuário exibe uma tela de chat, com uma primeira mensagem de saudação por parte da persona.

3. O usuário conversa conforme desejar com a persona selecionada.

UC4 - Caso de uso: Editar configuração inicial

- Ator: Usuário
- Descrição: Alterar dados de configuração inicial
- Pré-requisitos: UC1 - Configurar aplicação
- Fluxo principal
 1. O usuário acessa o menu lateral a partir do menu principal.
 2. O sistema exibe a opção “Editar”.
 3. O usuário clica no botão “Editar”.
 4. O sistema exibe a tela inicial, com as informações de nome e chave de API.
 5. O usuário edita as informações dos campos nome e chave de API.
 6. O usuário clica no botão “Avançar”.
 7. O sistema exibe o menu principal após os dados atualizados.
 8. O sistema exibe uma mensagem confirmando o sucesso na edição dos dados.
- Fluxo de execução:
 1. Se o conteúdo de algum dos campos for excluído, o botão “Avançar” será desabilitado, não permitindo a edição.
 2. Se a chave de API não for válida, uma mensagem de erro será exibida pelo sistema e as informações não serão atualizadas.

UC 5 - Caso de uso: Excluir uma persona

- Ator: Usuário
- Descrição: Exclusão de uma persona

- Pré-requisitos: UC1 - Configurar aplicação, UC2 - Criar uma persona
- Fluxo principal
 1. O usuário clica no botão de mais opções da persona criada.
 2. O sistema exibe as opções de editar e excluir.
 3. O usuário seleciona a opção de excluir.
 4. O sistema exibe a mensagem de confirmação de exclusão da persona.
 5. O usuário clica no botão confirmar.

UC6 - Caso de uso: Continuar conversa

- Ator: Usuário
- Descrição: Continuando uma conversa
- Pré-requisitos: UC1 - Configurar aplicação, UC3 - Conversar com uma persona
- Fluxo principal
 1. O usuário acessa o menu lateral através do menu principal.
 2. O sistema exibe uma lista de histórico de conversas.
 3. O usuário seleciona a conversa que deseja continuar.
 4. O sistema exibe a tela de chat com as mensagens anteriormente enviadas.

UC7 - Caso de uso: Excluir uma conversa

- Ator: Usuário
- Exclusão de uma conversa
- Pré-requisitos: UC1 - Configurar aplicação, UC3 - Conversar com uma persona
- Fluxo principal
 1. O usuário acessa o menu lateral através do menu principal.

2. O sistema exibe uma lista de histórico de conversas.
3. O usuário clica no botão de mais opções.
4. O sistema exibe a opção de excluir conversa.
5. O usuário clica na opção de excluir conversa.
6. O sistema exibe uma mensagem de confirmação de exclusão.
7. O usuário clica no botão “Confirmar”.
8. O sistema exibe a lista atualizada de histórico de conversas.
9. O sistema exibe uma mensagem de exclusão realizada com sucesso.