

**Universidade do Minho**

Escola de Engenharia

Departamento de Informática

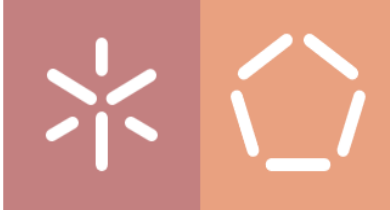
Luís Manuel Leite Costa

**Sistemas de Conversação  
Para Software Educacional**

Dissertação de Mestrado

Mestrado Integrado em Engenharia Informática

Julho de 2022



Universidade do Minho

Escola de Engenharia

Departamento de Informática

Luís Manuel Leite Costa

## Sistemas de Conversação Para Software Educacional

Dissertação de Mestrado

Mestrado Integrado em Engenharia Informática

Dissertação supervisionada por

Professor Doutor Orlando Manuel de Oliveira Belo

Julho de 2022

## DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



Atribuição-NãoComercial-SemDerivações

CC BY-NC-ND

<https://creativecommons.org/licenses/by-nc-nd/4.0/>

## Agradecimentos

Em primeiro lugar, queria agradecer ao meu orientador, o Professor Doutor Orlando Belo, pelos ensinamentos, pela motivação, pela paciência nos momentos mais difíceis e pela sensação de companheirismo que me transmitiu ao longo da realização desta dissertação.

Aos meus pais, que me incentivaram a estudar e a alcançar este marco. Ficarei sempre grato por acreditarem em mim e me permitiram percorrer o meu caminho ao meu ritmo.

Por fim, agradeço aos meus amigos que me apoiaram e que, de uma maneira ou de outra, tornaram mais fácil a concretização deste projeto.

## DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducentes à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

## Resumo

Tutores artificiais são agentes de *software* que auxiliam nos processos de ensino e formação. Nos dias de hoje, esses sistemas têm como objetivo fornecer instruções aos alunos sem a intervenção direta de um professor. Para que isso aconteça com sucesso, é necessário que estes sistemas possuam um sistema de conversação que seja capaz de interagir com os alunos de forma simples e cativante, mantendo um diálogo adequado em todos os momentos de ensino que concedem. Essencialmente, os sistemas de conversação são assistentes virtuais que utilizam interfaces de comunicação que interagem com os utilizadores através de frases escritas e orais, e, geralmente, possuem a capacidade de compreender o utilizador. Por norma, o computador assume o papel de professor. No entanto, o aluno também “ensina” o sistema fornecendo dados e a sua perspetiva em relação ao problema, que podem posteriormente ser utilizados para personalizar o seu ensino.

É neste contexto que foram realizados os trabalhos desta dissertação, cujo principal objetivo foi a implementação de um componente de conversação para um sistema educacional, que fosse capaz de criar novas dinâmicas entre o sistema e o utilizador, incentivando-o, assim, a despende mais tempo na ferramenta.

**Palavras-Chave:** Diálogos, Sistema de Conversação, Sistemas Educacionais, Sistema de Ensino Inteligente.

# Abstract

Artificial tutors are software agents that assist in teaching and training processes. Nowadays, these systems aim to provide instructions to students without the direct intervention of a teacher. For this to happen successfully, these systems must have a conversational system that can interact with the students simply and engagingly while maintaining a fitting dialogue in all moments the user goes through.

Conversational systems are virtual assistants that use communication interfaces that interact with users using written and oral phrases, generally having the ability to understand the user. By default, the computer assumes the role of a teacher, however, the students can also "teach" the system by providing data and their perspective on the problem, which can be used later to customize their teaching.

It is in this context that the work of this dissertation will be carried out, which will have the main objective of implementing a dialogue component for an Educational System capable of creating new dynamics between the system and the user, thus encouraging the user to spend more time using the tool.

**Keywords:** Educational System, Intelligent Tutoring System, Conversational System, Natural Language Processing, Dialogue.

## Conteúdo

1. Introdução.....	12
1.1 Enquadramento.....	12
1.2 Motivação e Objetivos.....	13
1.3 Trabalho Realizado.....	14
1.4 Organização da Dissertação.....	15
2. Sistemas de Conversação.....	16
2.1 Definição e Utilidade.....	16
2.2 O Processo de Conversação.....	17
2.3 Problemas Típicos e Armadilhas.....	18
2.4 Desafios e Oportunidades.....	19
2.5 Tipos de Sistemas de Conversação.....	20
2.5.1 Sistemas de Conversação Baseados em Regras.....	20
2.5.2 Sistemas de Conversação Orientados à Realização de Tarefas.....	21
2.5.3 Sistemas de Conversação Inteligentes.....	22
2.6 Exemplos de Sistemas e Aplicações.....	23
2.6.1 Eliza.....	23
2.6.2 Parry.....	26
2.6.3 ALICE.....	28
2.6.4 Xiaolce.....	30
2.6.5 Análise Comparativa.....	32
3. Um Sistema de Conversação.....	33
3.1 O Domínio de Aplicação.....	34
3.2 Desenvolvimento do Sistema.....	34
3.3 Arquitetura Geral.....	36
3.4 Mensagem-Padrão.....	39
3.5 O Gestor de Diálogos.....	41
3.6 O Motor de Regras.....	42
3.7 O Coletor.....	49
4. Um Sistema de Introdução de Diálogo.....	52
4.1 Caracterização Geral.....	52
4.2 O Leonardo++.....	52
4.3 O Sincronizador de Diálogo.....	57



4.4 O Gerador de Diálogo .....	58
4.5 O Coletor .....	60
5. Teste e Demonstração do Sistema de Conversação .....	62
5.1 A Plataforma de Teste LeonardoTest .....	62
5.2 Complexidade do Diálogo .....	62
5.3 Verificação das Regras de Conversação.....	63
5.4 Demonstração de Funcionalidades .....	65
5.5 O Processo de Demonstração.....	68
5.6 O Sistema Educacional Modelo.....	70
6. Conclusões e Trabalho Futuro.....	72
6.1 Síntese e Conclusões.....	72
6.2 Trabalho Futuro .....	73
Bibliografia.....	75
Referências Web .....	77

## Índice de Figuras

Figura 1: Estrutura de um sistema de conversação orientado à realização de tarefas - Exemplo adaptado de Shum (2018).....	22
Figura 2: Exemplo de conversação com Eliza – figura extraída de (Weizenbaum, 1996) .....	24
Figura 3: Processo de decomposição.....	25
Figura 4: Excerto de diálogo entre Eliza e Parry – extraído de (Cerf, 1973).....	27
Figura 5: Exemplo de uma categoria AIML.....	28
Figura 6: Exemplo de aplicação do elemento SRAI.....	29
Figura 7: Exemplo de interação de Xiaolce com um utilizador – figura extraída de Li Zhou (2019) .....	31
Figura 8: Etapas de funcionamento do Sistema de Conversação .....	33
Figura 9: Ciclo de Vida do Sistema de Conversação.....	36
Figura 10: Exemplo de estrutura de uma dada sequência lógica .....	42
Figura 11: Ambiente inicial do sistema Leonardo++ .....	53
Figura 12: Ambientes de quizz do sistema Leonardo++ que permitem identificar o contexto da frase que.....	54
Figura 13 Ambientes de quizz do sistema Leonardo++ que permitem identificar o contexto da frase que.....	54
Figura 14: Ambientes de inserção de diálogo do sistema Leonardo++ - Inserção singular .....	55
Figura 15: Ambientes de inserção de diálogo do sistema Leonardo++ - Inserção múltipla.....	57
Figura 16: Exemplo de Cadeia de Markov.....	58
Figura 17: Acesso ao sistema de testes LeonardoTest.....	65
Figura 18: a) Ambiente de inserção de quantidade de simulações a serem realizadas no sistema LeonardoTest b) Ambiente que apresenta os resultados de uma simulação do sistema LeonardoTest .....	65
Figura 19: a) Ambiente do sistema LeonardoTest que apresenta os gráficos de análise de um dado teste b) Ambiente do sistema LeonardoTest que apresenta as intervenções realizadas pelo chatbot Leonardo num dado teste .....	66
Figura 20: Ambiente do sistema LeonardoTest que apresenta os resultados do teste de regras	67
Figura 21: Ambiente de iniciação do sistema Let's Test.....	70
Figura 22: O ambiente de quizz do sistema Let's Test.....	71
Figura 23: Ambiente de finalização de teste do sistema Let's Test.....	71

## Índice de Tabelas

Tabela 1: Informação relativa aos sistemas de conversação Eliza, Parry, Alice e Xioice .....	32
Tabela 2: Estados que iniciam a cadeia de Markov exemplo.....	59
Tabela 3: Estados existentes na cadeia de Markov exemplo e os seus possíveis estados de transição.....	59

# 1. Introdução

## 1.1 Enquadramento

Os *Intelligent Tutoring Systems* (ITSs) são sistemas computacionais que utilizam técnicas de inteligência artificial com o objetivo de personalizar as técnicas de ensino que oferecem de acordo com as necessidades do utilizador.

Estes sistemas tentam reproduzir comportamentos humanos que poderiam ser descritos como inteligentes. Isto é, os ITSs podem ser vistos como tentativas de produzir, em um computador, comportamentos que, se realizados por um ser humano, seriam descritos como práticas de “bom ensino” (Nwana, 1990). Estes sistemas precisam de interagir da forma mais humana possível com os seus utilizadores e, por esse motivo, fazem uso de sistemas de diálogo também conhecidos por sistemas de conversação.

Os sistemas de conversação são sistemas inteligentes capazes de entender uma linguagem e de conduzir uma conversa, através da escrita ou oralmente, com um utilizador. O seu principal objetivo é melhorar a qualidade da experiência do utilizador no sistema no qual se encontra inserido (Shum et al., 2018). Este tipo de sistema fornece respostas informativas e/ou auxílio em tempo real numa dada área de aplicação. Por norma, esta interação é realizada através de *chatbots* que podem ser utilizados para melhorar um tipo de comunicação ou até mesmo criar um novo ponto de diálogo. Segundo Abdul-Kader & Woods (2015), os *chatbots* são programas de computadores capazes de manter uma conversa com humanos usando um discurso baseado em linguagem natural que conduzem a conversa através de métodos auditivos ou textuais, simulando de uma forma muito idêntica a interação com um ser humano. Estes sistemas são um ponto importante para a experiência do utilizador pois, o avatar do *chatbot*, o seu tom de voz e seu modo de diálogo são elementos-chave para o sucesso da interação do sistema com o utilizador.

Para funcionarem de uma forma eficaz, os *chatbots* têm de ser capazes de traduzir a voz e a linguagem humana em algo que um computador consiga compreender. O processo de compreensão, análise e resposta à voz humana é uma tarefa complexa a que chamam de processamento de linguagem natural.

O processamento de linguagem natural refere-se ao processo através do qual os *chatbots* utilizam a inteligência artificial para dar sentido a grandes volumes de dados escritos numa linguagem corrente. Através do processamento de linguagem natural é possível gerar e apresentar respostas adequadas ao contexto em que o utilizador está inserido. Esta vertente da inteligência

artificial permite que os *chatbots* comuniquem com utilizadores humanos de uma forma que simula a conversação humana (Corporation, s.d.).

Uma grande vantagem destes sistemas é serem capazes de fornecer uma via de comunicação, que se encontra disponível vinte e quatro horas por dia, o que resulta numa maior disponibilidade do sistema. Realça-se também que o *chatbot* oferece a possibilidade de fornecer atendimento simultâneo, isto é, um *chatbot* pode receber vários utilizadores ao mesmo tempo e ainda interagir com os seus administradores. Uma outra vantagem da utilização destas tecnologias é a capacidade de armazenar todo o histórico de conversação e relatórios. Através destes dados é possível fornecer aos seus administradores a perspetiva do utilizador nos seus sistemas.

Segundo estudos realizados por Yuen (2022), cada vez mais os consumidores exigem serviços de 24 horas por dia para assistência em áreas que vão desde bancos e finanças até saúde e bem-estar. Como resultado, as empresas procuram rapidamente desenvolver *chatbots* e assistentes virtuais para responder a perguntas que os clientes possam ter a qualquer hora do dia. Quase 40% dos utilizadores da Internet de todo o mundo preferem interagir com *chatbots* do que com agentes virtuais, e com as principais indústrias, incluindo retalho e saúde, a voltarem-se para a tecnologia digital, os *chatbots* irão aumentar em popularidade no futuro.

## 1.2 Motivação e Objetivos

A motivação de um aluno é um elemento essencial que é necessário para uma educação de qualidade. Segundo Williams K. & Williams C. (2011), muito pouca ou nenhuma aprendizagem pode ocorrer a menos que o aluno esteja motivado de uma forma consistente e realça que o ambiente em que o aluno aprende é um dos principais fatores que leva ao sucesso do mesmo. O ambiente propício a uma boa aprendizagem deve ser acessível, seguro, positivo, personalizado o máximo possível e incentivador para o aluno.

A ausência de um sistema de conversação em plataformas de ensino online limita a capacidade de interação com o utilizador, o que faz com que a relação sistema-utilizador seja pouco apelativa para quem o utiliza. Os *chatbots* podem facilmente criar um ambiente mais propício ao estudo e acompanhar alunos durante os diferentes momentos de aprendizagem, aumentando assim o rendimento do utilizador e as suas probabilidades de sucesso na sua aprovação. Foi neste contexto que se deu início ao desenvolvimento de um *chatbot* que seja capaz de melhorar o ambiente de um aluno quando este recorre a um *Intelligent Tutoring System*. Dada

a diferença que um *chatbot* pode proporcionar no seu ambiente de estudo e nos seus resultados académicos, os principais objetivos deste trabalho consistem em criar um *chatbot* capaz de:

- **Acompanhar um aluno durante a utilização de um ITS** – É pretendido que o utilizador seja acompanhado durante o momento de realização de uma sequência de questões.
- **Personalizar a sua interação dependendo do utilizador** – O *chatbot* tem de ter a capacidade de se adaptar ao aluno de acordo com o seu histórico e contexto no sistema.
- **Motivar o aluno ao estudo através das suas interações** – O *chatbot* deve estimular o aluno de modo que este se sinta motivado a manter-se mais tempo no sistema e, por consequência, a estudar

### 1.3 Trabalho Realizado

O trabalho desta dissertação teve como finalidade a implementação de um sistema de conversação capaz de ser inserido num *Intelligent Tutoring Systems*. Por forma a perceber melhor o problema e o estado de evolução em que se encontram os *chatbots*, foi realizada uma investigação sobre a história dos sistemas conversacionais e sobre os diferentes tipos de sistemas que emergiram de acordo com necessidade e progressão das tecnologias. Com isto, foi possível identificar quais as vantagens e utilidades que um sistema de conversação oferece e quais são os desafios da sua conceção.

Foram ainda estudados diferentes *chatbots* com diferentes abordagens e objetivos de implementação que os seus autores tiveram para solucionar diversos problemas. Como por exemplo, o processamento de linguagem e a criação de inteligência emocional.

De seguida foi implementado um sistema de conversação capaz de integrar um *software* educacional e os seus respetivos subsistemas capazes de garantir o bom funcionamento e atualização do sistema principal.

Para um melhor entendimento do sistema criado, foi também implementado um sistema educacional modelo que permite demonstrar o *chatbot* em funcionamento de uma forma mais perceptível e real.

## 1.4 Organização da Dissertação

Para além do presente capítulo, que teve como finalidade introduzir o tema da dissertação e apresentar as dificuldades que se pretendem ultrapassar, esta dissertação integra mais cinco capítulos nos quais se apresenta a evolução e a história dos *chatbots*, bem como uma apresentação e descrição do trabalho desenvolvido. Esta estrutura permite que inicialmente sejam expostos os conceitos teóricos estudados por terceiros e, por fim, a abordagem pessoal e prática do problema. Assim, a restante parte desta dissertação, está organizada da seguinte forma:

- **Capítulo 2** - Sistemas de Conversação - em que se define um sistema de conversação e se apresenta a sua utilidade, bem como se discute alguns dos problemas e dificuldades recorrentes na implementação deste tipo de sistemas, tal como os ganhos que se obtêm através da sua aplicação. Além disto, são apresentados diferentes tipos de sistemas de conversação, acompanhados por alguns exemplos de aplicações.
- **Capítulo 3** - Um Sistema de Conversação - no qual é descrito todo o processo de implementação do sistema de conversação implementado e se apresenta a arquitetura geral do sistema e componentes chave para o seu sucesso.
- **Capítulo 4** - Um Sistema de Introdução de Diálogo - neste capítulo expõe-se um sistema de introdução de diálogo, que foi criado especificamente para auxiliar a introdução de conhecimento no sistema de conversação apresentado no capítulo anterior, tal como os diferentes componentes que o compõem.
- **Capítulo 5** - Teste e Demonstração do Sistema de Conversação - no qual se revela o processo de teste e validação do sistema implementado, como também se demonstra o seu funcionamento.
- **Capítulo 6** - Conclusões e Trabalho Futuro – em que apresenta as diferentes conclusões obtidas com a realização desta dissertação, bem como se aponta algumas possíveis implementações que podem vir a ser concretizadas no futuro para melhorar o sistema atual.

## 2. Sistemas de Conversação

### 2.1 Definição e Utilidade

Os sistemas de conversação, mais conhecidos por *chatbots*, são sistemas inteligentes que utilizam técnicas de inteligência artificial e de processamento de linguagem natural para ajudar os utilizadores a interagir com os sistemas nos quais se encontram inseridos. Estes são capazes de manter uma conversa em tempo real, por texto ou voz, e realizar tarefas simples que possam ser automatizadas. Podem pertencer a um sistema maior ou serem totalmente autónomos.

Uma vantagem para as empresas utilizarem *chatbots* é o facto de estes sistemas criarem experiências de utilização personalizadas, que se traduzem em utilizadores mais envolvidos e mais satisfeitos, bem como maiores taxas de satisfação. Estes sistemas também são apreciados pelos utilizadores, pois oferecem meios de assistência durante vinte e quatro horas por dia, sem qualquer tipo de restrições, o que faz com que os utilizadores possam obter respostas às suas questões em qualquer altura do dia. Por sua vez, os utilizadores tendem também a ter experiências mais simples e uma ligação mais pessoal com as marcas com as quais interagem.

Os sistemas conversacionais podem ser utilizados em diversos sectores (Universia, 2020), como por exemplo:

- **Educação.** Através da sua automação, os *chatbots* permitem auxiliar a gestão de tarefas repetitivas realizadas por um docente ou por uma instituição, respondendo a perguntas gerais sobre o curso, os seus objetivos gerais, plano curricular, planos de atividades e avaliações das unidades curriculares, entre outros.
- **Saúde.** Para além de permitirem fazer agendamento de consultas, os *chatbots* podem ser muito úteis para prestar atendimento em situações de pânico. Guiados pelos sintomas dos utilizadores, os *chatbots* são capazes de devolver mensagens rápidas e programadas por médicos e profissionais que instruem os doentes a prestar a si próprios os seus primeiros socorros até que o atendimento presencial compareça, começando assim o seu tratamento mais rapidamente.



- **Comércio.** Os *chatbots* podem ser utilizados como meio de realizar o primeiro contacto com um potencial cliente, pois estes são capazes de direcionar o comprador para a loja mais próxima do cliente e facilitar o contacto com um vendedor. Também podem responder a questões comuns como o estado de uma encomenda, o prazo de entrega estimado, trocas e devoluções.

## 2.2 O Processo de Conversação

A forma como interagirmos com a tecnologia tem vindo a evoluir de acordo com o tempo, e o mesmo se sucedeu com os *chatbots*. Atualmente, existem várias formas de interagir com sistemas de conversação, porém existem três meios dominantes na interação com estes sistemas (Engati, 2022). Estas interações são baseadas em:

- **Botões/Menus.** Os *chatbots* baseados em botões/menus são o tipo mais básico de *chatbots* atualmente implementados no mercado. Na maioria dos casos, estes *chatbots* seguem uma hierarquia de nodos de uma árvore de decisão que se apresentam ao utilizador na forma de botões. Semelhante aos menus que é possível ver numa caixa de multibanco, por exemplo, estes *chatbots* requerem que o utilizador faça várias seleções para que este seja capaz de perceber o que é pretendido pelo utilizador e assim devolver a resposta final. Apesar de estes *chatbots* serem eficientes em responder a perguntas frequentes e possíveis de prever, ficam aquém em cenários mais desenvolvidos, nos quais existem muitas variáveis ou em momentos nos quais o próprio utilizador não sabe o que pretende. Também vale a pena notar que os *chatbots* baseados em menu/botão são os mais lentos em termos de conduzir o utilizador à resposta final desejada.
- **Texto.** Ao contrário dos *chatbots* baseados em menu, os *chatbots* baseados em texto interpretam aquilo que os utilizadores escrevem e respondem adequadamente. Estes *chatbots* aplicam procedimentos de inteligência artificial que identificam as palavras-chave através de técnicas de processamento de linguagem para determinar a resposta apropriada ao utilizador. Este tipo de *chatbots* não é eficiente, quando precisa responder a muitas perguntas semelhantes, pois têm a tendência a falhar quando existem redundâncias de palavras-chave entre as perguntas realizadas pelo utilizador.

- **Voz.** Um *voicebot* é uma ferramenta de comunicação de inteligência artificial conversacional que captura, interpreta e analisa o input por voz fornecido pelo utilizador e responde em linguagem natural semelhante. O método de fornecer inputs por voz em tempo real é o modo de comunicação mais rápido. Com a evolução da inteligência artificial, os *chatbots* de voz estão a tornar-se mais robustos, flexíveis e seguros na maneira como atendem os utilizadores. A tecnologia de voz, seja na forma de altifalantes inteligentes ou robôs de voz, simplifica consideravelmente a interação utilizador-sistema. Uma das maiores dificuldades deste sistema é fazer a interpretação correta do input do utilizador, visto que todas as pessoas têm formas diferentes de se expressar.

### 2.3 Problemas Típicos e Armadilhas

Tal como a implementação de qualquer outro sistema computacional, durante a implementação de um sistema conversacional é necessário ter em consideração diversos aspetos de forma que o produto final seja realmente o pretendido e não se desvie do seu propósito. Em conformidade com Fortes (2021), apresentam-se, de seguida, alguns exemplos de problemas e de armadilhas que podem surgir durante a conceção de um *chatbot*, nomeadamente:

- **Custos.** Apesar da conceção de um sistema de conversação prever a redução de custos ao nível da contratação de recursos humanos, os custos iniciais associados ao uso de *chatbots* são elevados, sobretudo numa fase inicial, quer no desenvolvimento da própria solução, quer na compra de soluções já existentes e sua adaptação.
- **Volume de dados.** A precisão dos *chatbots* está muito dependente do volume de dados que recebem. Quanto maior o volume de dados, maior a sua eficiência no atendimento ao utilizador. Por vezes, a dificuldade também está na qualidade dos dados, que têm de ser tratados e filtrados antes de serem inseridos no sistema.
- **Sensibilidade e emoção humana.** Como qualquer outra aplicação de Inteligência Artificial, e apesar de aprenderem constantemente com os dados que vão recolhendo, os *chatbots* pecam ao nível da sensibilidade e emoção humana. Em

particular, os *chatbots* baseados em texto são sistemas pouco eficazes na compreensão das emoções dos utilizadores e por vezes erram no tipo de diálogo a utilizar.

- **Fácil manutenção.** Muitas vezes, os *chatbots* precisam ser ajustados de uma forma rápida por um responsável, que pode não ser o fornecedor do serviço. Logo, é pretendido que o sistema esteja preparado realizar alterações nas suas configurações por um administrador que não possua conhecimentos de programação. A solução para este problema pode consistir numa interface gráfica, que auxilie o responsável a configurar o seu próprio *chatbot*.

## 2.4 Desafios e Oportunidades

Com a implementação de um sistema conversacional existe a oportunidade de criar algo inovador e único. Contudo, existem desafios que precisam de ser ultrapassados para que o *chatbot* seja realmente útil para o utilizador final. Segundo Fortes (2021) e Shum et al. (2018), alguns desses desafios (e oportunidades) são:

- **Tornar o *chatbot* à prova do futuro.** A tecnologia é algo que evolui de forma progressiva. Por essa razão, um dos grandes desafios dos *chatbots* é de não se tornarem desatualizados. É então necessário que o sistema seja capaz de se adaptar com a evolução dos tempos adequando o seu diálogo, atualizando o seu conhecimento e dispor da capacidade de incorporar novas tecnologias.
- **Compreender as emoções e sentimentos de seus utilizadores.** A interpretação de emoções e sentimentos humanos é um dos pontos mais importantes na conceção de um *chatbot*, visto que, caso sejam mal interpretados podem ter um enorme impacto negativo na experiência do utilizador. É, por isso, considerado um dos grandes desafios na implementação de um *chatbot*. Para superá-lo é necessário um grande investimento em treinamento e preparação de regras.

- **Superar o Teste de Turing.** O Teste de Turing testa a capacidade de uma máquina exibir comportamento inteligente equivalente a um ser humano, ou pelo menos que seja indistinguível deste. Foi introduzido por Alan Turing em 1950 e, desde então, é utilizado para determinar a qualidade de um sistema de conversação.

## 2.5 Tipos de Sistemas de Conversação

Uma fonte de inspiração para o desenvolvimento da inteligência artificial e, principalmente, dos *chatbots*, foram os estudos de Alan Turing que propôs o primeiro conceito de inteligência artificial em 1950. Desde então surgiram vários sistemas de conversação concebidos para imitar conversas humanas, que tinham por base instruções e repostas predefinidas. Atualmente os chatbots são capazes de dialogar com os utilizadores e de se adaptarem, usando técnicas de linguagem natural (PLN) e de *machine learning* (Weni, 2018).

De seguida, irá ser abordado o estado atual de evolução das tecnologias utilizadas, desde as tarefas que desempenham e principais características como as fraquezas e aptidões, sem deixar de aprofundar um pouco sobre a história que levou ao aperfeiçoamento destes mecanismos.

### 2.5.1 Sistemas de Conversação Baseados em Regras

De acordo com Copeland (2000), foram desenvolvidos vários sistemas de conversação inspirados nos testes de Turing propostos pelo matemático britânico Alan Turing, em 1950, para conversas de circunstância. Estes tipos de sistemas eram frequentemente projetados para simular, de forma convincente, como um ser humano se comportaria como um parceiro de conversação (Shum et al., 2018). Alan Turing sugeriu que se um computador pudesse fazer-se passar por um humano com sucesso e manter uma conversa escrita por computador com uma pessoa poderia ser chamado de inteligente.

Um sistema de conversação baseado em regras tem como principal componente de lógica o seu motor de regras. É um sistema automatizado projetado para responder a inputs, que surgem, frequentemente, no sistema no qual está integrado, quer estes sejam provenientes diretamente do utilizador, na forma de mensagens, ou por ações que o mesmo realize. Estes sistemas seguem um conjunto de regras pré-programadas, que são projetadas para responder aos cenários que se preveem surgir, tornando as suas intervenções possíveis de serem controladas.

Este tipo de sistemas é utilizado em situações de atendimento ao cliente, onde é necessário obter um conjunto de informações de forma a ser possível realizar o pedido do cliente. Um exemplo deste tipo de sistema seria Dom (Gilliland, 2016), *chatbot* da empresa Domino's, que permite aos clientes fazerem pedidos online através de diferentes *chats* de redes sociais. É possível também deixar feedback relativamente a um serviço anterior ou contactar a equipa de suporte da empresa.

Tal como Dom, existem muitos *chatbots* integrados em redes sociais. O grande número destes sistemas de conversação deve-se ao facto de serem de fácil implementação e de existirem plataformas como Landbot (s.d.), que permitem criar, de forma gratuita, um *chatbot* capaz de se integrar em redes sociais através de um sistema *drag-and-drop* sem a necessidade de recorrer a programação.

### 2.5.2 Sistemas de Conversação Orientados à Realização de Tarefas

Ao contrário dos sistemas baseados em regras, *Task-completion Conversational Systems*, são sistemas projetados para a realização de tarefas e, geralmente, são utilizados em domínios restritos (Shum et al., 2018). Na *Figura 1* podemos ver uma ilustração da arquitetura de um sistema de conversação orientado à realização de tarefas. A arquitetura deste sistema é constituída por um componente de reconhecimento de voz (RV), um componente de processamento de linguagem (PL), um componente de gestão de diálogo (GD), um componente de criação de diálogo (CD) e um componente de conversão de texto para voz (CTV).

O ciclo de vida deste sistema inicia com o input do utilizador via voz que, por sua vez, é recebido pelo componente de reconhecimento de voz. Este transforma os sinais áudio em palavras e encaminha a informação tratada para o componente de processamento de linguagem. Através da análise individual das palavras, o componente de processamento de linguagem identifica o domínio do diálogo e a intenção do utilizador. Após esta análise, são enviados os dados obtidos para o gestor de diálogo. O propósito deste componente consiste em garantir que o pedido do utilizador seja retornado conforme o pretendido e que a mensagem de output que irá ser gerada pelo sistema de conversação responde ao problema do utilizador. Para isso, este componente acede à base de dados do sistema para recolher a resposta modelo que melhor se adequa ao contexto e transmite-a para o componente de criação de diálogo (Este elemento poderá também ter a responsabilidade de guardar o histórico do diálogo, para que em situações futuras consiga

responder de uma forma mais personalizada ao contexto do utilizador). Após receber a informação, o componente de criação diálogo personaliza a resposta modelo e envia-a para o componente de conversão de texto para voz que responde, via som, ao pedido do utilizador.

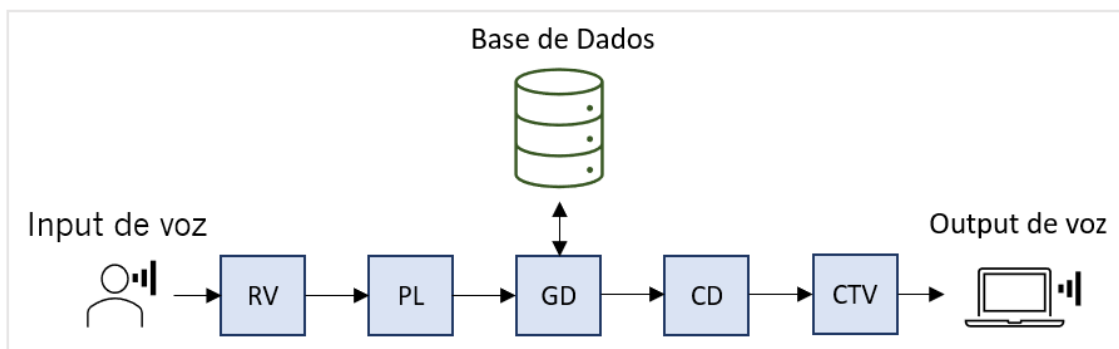


Figura 1: Estrutura de um sistema de conversação orientado à realização de tarefas - Exemplo adaptado de Shum (2018)

Um exemplo de um sistema de conversação que se enquadra neste tipo de sistemas de conversação é o *chatbot* Eva (Patel, s.d.) que auxilia os clientes de um banco a obter informações sobre os seus cartões de crédito, empréstimos e esclarecimentos sobre os serviços que o banco oferece. É estimado que este sistema de conversação realiza cerca de vinte mil conversas diferentes diariamente com clientes de todo o mundo.

Outro exemplo seria o *chatbot* Jenny (Patel, s.d.) que integra um website de eventos que visa construir uma comunidade mundial de start-ups. O *chatbot* é capaz de fornecer informações sobre os locais dos eventos, os seus interlocutores e a que horas se realizam. 64% de todas as solicitações de suporte ao cliente foram respondidas com sucesso através deste chatbot.

### 2.5.3 Sistemas de Conversação Inteligentes

Os sistemas de conversação inteligentes, também conhecidos como *social chatbots*, foram criados para atender as necessidades de comunicação, afeto e inserção social dos utilizadores. Ao contrário dos sistemas orientados à realização de tarefas, que realizam tarefas eficientemente e encerram a conversa o mais rapidamente possível, o objetivo dos *chatbots* sociais é interagir com o utilizador pelo maior período de tempo possível, apresentando temas, oferecendo perspetivas e solicitando novos tópicos para manter a conversa em desenvolvimento.

Para estes serem bem-sucedidos, estes tipos de sistemas têm de ter a capacidade de reconhecer as emoções do utilizador e acompanhar as suas mudanças de estado de espírito durante a conversa. Com o avanço tecnológico das redes neuronais artificiais em domínios como

reconhecimento de voz, processamento de linguagem e visão computacional, este desafio tornou-se mais atingível (Vinyals & Le, 2015).

Atualmente existem modelos que facilitam a criação de texto natural em linguagem humana de uma forma automatizada e orientada para a interação com os utilizadores. Um exemplo deste tipo de modelos é o Generative Pre-Training Transformer 3 (Schmelzer, 2021), que consiste numa rede neuronal artificial treinada com dados obtidos através da internet que possibilitam a geração de qualquer tipo de texto independentemente do contexto. Através deste modelo é possível criar poesia, histórias, diálogos e até mesmo criar código de programação. Um caso prático da aplicação deste modelo é o *chatbot* Emerson (Gold, 2021). Este sistema de conversação consegue falar em diferentes idiomas através de mensagens de texto e áudio, analisar imagens e de fazer comentários sobre as mesmas. Tem como principal objetivo ajudar os utilizadores a aprender novos idiomas e instruir os mesmos sobre novos temas.

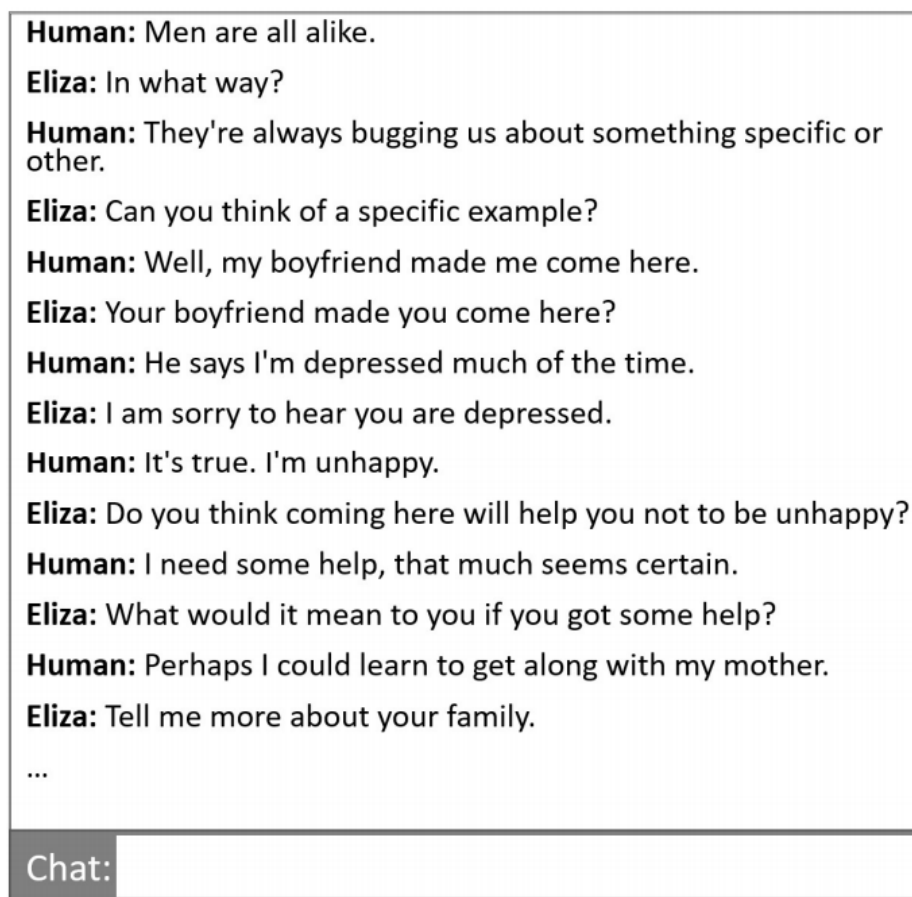
## 2.6 Exemplos de Sistemas e Aplicações

Apesar de os sistemas de conversação apresentarem um grande nível de complexidade tecnológica, os *chatbots* não surgiram apenas no século XXI. Em 1950, Alan Turing perguntou-se se um programa de computador conseguiria falar com um grupo de pessoas sem que estes percebessem que o seu interlocutor era artificial. Esta questão, denominada teste de Turing, é considerada por muitos como a ideia original dos *chatbots* (Adamapoulou & Moussiades, 2020). De forma a melhor entender a evolução dos sistemas de conversação e como estes foram utilizados, foram estudados diferentes *chatbots* que marcaram o progresso deste tipo de sistemas e que, ainda hoje, são pontos de referência para novas implementações.

### 2.6.1 Eliza

O sistema Eliza foi criado por Joseph Weizenbaum no MIT, em 1966, e foi o primeiro *chatbot* a ser apresentado publicamente (Weizenbaum, 1966). O Eliza consegue comunicar com humanos com base em padrões criados manualmente, que simulam o comportamento de um psicoterapeuta. Este *chatbot* não entende o que o utilizador escreve, mas procura respostas apropriadas através da identificação de padrões. Eliza tem um conhecimento bastante reduzido e apenas consegue falar com pessoas em um domínio restrito. No entanto, quando o Eliza foi testado

pela primeira vez, muitos utilizadores acreditavam que estavam a conversar com uma pessoa real. De seguida, na Figura 2, encontramos um exemplo de uma conversa entre o Eliza e um humano.



**Human:** Men are all alike.  
**Eliza:** In what way?  
**Human:** They're always bugging us about something specific or other.  
**Eliza:** Can you think of a specific example?  
**Human:** Well, my boyfriend made me come here.  
**Eliza:** Your boyfriend made you come here?  
**Human:** He says I'm depressed much of the time.  
**Eliza:** I am sorry to hear you are depressed.  
**Human:** It's true. I'm unhappy.  
**Eliza:** Do you think coming here will help you not to be unhappy?  
**Human:** I need some help, that much seems certain.  
**Eliza:** What would it mean to you if you got some help?  
**Human:** Perhaps I could learn to get along with my mother.  
**Eliza:** Tell me more about your family.  
...  
**Chat:**

Figura 2: Exemplo de conversação com Eliza – figura extraída de (Shum et al., 2018)

De acordo com Weizenbaum (1966), este sistema de conversação inicia o seu processo de resposta através de uma frase escrita fornecida por um utilizador e começa por procurar palavras-chave. Uma palavra-chave, também conhecida por *keyword*, é uma palavra identificada como importante pelo sistema Eliza que atribui a cada palavra-chave um número de importância (*rank*) estabelecido pelo programador. Se essas palavras forem encontradas, serão colocadas numa *keystack*, com a palavra-chave de maior *rank* no topo. Certas palavras podem ser manipuladas independentemente de considerações contextuais, como por exemplo, alternar os pronomes da primeira pessoa para segunda pessoa e vice-versa. Tais palavras com números de alta importância, são consideradas superiores aos padrões de conversação e são tratadas independentemente dos padrões contextuais.

Após este primeiro passo, o sistema aplica uma regra de transformação apropriada, que inclui duas partes: 1) a "regra de decomposição" e 2) a "regra de remontagem". Inicialmente são procurados na frase padrões sintáticos a fim de estabelecer o contexto mínimo necessário para



responder. Usando as palavras-chave e outras palavras semelhantes ao input, o sistema testa diferentes regras de decomposição até que um padrão apropriado seja encontrado. Usando as regras do sistema, o input é então decomposto e dividido em partes, conforme a regra de decomposição para a palavra-chave de classificação mais alta.

Analisando o exemplo apresentado por Weizenbaum (1966) (Figura 3), "*It seems that you hate me*". Eliza divide esta frase em quatro componentes diferentes:

1	2	3	4
It seems that	you	hate	me

Figura 3: Processo de decomposição

Percebendo apenas o segundo e quarto componente, o sistema através da regra de remontagem obtém "*What makes you think I hate you*". O sistema ignora o primeiro componente, traduz as duas palavras conhecidas ("*you*" para "*I*" e "*me*" para "*you*") e usa uma frase padrão ("*What makes you think*") no início da frase gerada.

Após a publicação deste sistema foram apresentados alguns problemas técnicos, nomeadamente:

1. A identificação da palavra-chave "mais importante" que ocorre na mensagem de entrada.
2. A identificação de algum contexto mínimo no qual a palavra-chave escolhida aparece; por exemplo, se a palavra-chave é "Tu", e esta é seguida pela palavra "és", neste caso, provavelmente está a ser feita uma afirmação.
3. A escolha de uma regra de transformação apropriada e a realização da própria transformação.
4. A falta de um mecanismo que permita Eliza responder de forma inteligente quando o texto de entrada não apresenta nenhuma palavra-chave.
5. A falta de um mecanismo que facilita a edição, particularmente a extensão, no sistema no que diz respeito à escrita da frase.

Apesar de o sistema Eliza não aplicar técnicas de inteligência artificial ou aprender através do input dos utilizadores, este sistema impulsionou a conceção de muitos outros *chatbots* que usaram a sua arquitetura como base dos seus desenvolvimentos.

### 2.6.2 Parry

O Parry é um *chatbot* desenvolvido por Kenneth Colby, em 1975. Este *chatbot* foi projetado com o objetivo de simular o comportamento de uma pessoa paranoica, isto é, uma pessoa que é fortemente influenciada pela ansiedade ou medo, muitas vezes ao ponto de delírio ou irracionalidade (Colby et al., 1971). Mesmo seguindo uma estrutura idêntica ao do *chatbot* Eliza, e basear-se em regras, o sistema Parry foi o primeiro *software* de conversação a passar no teste de Turing. O que distingue Parry de Eliza é a estrutura de controlo que se apresenta mais desenvolvida, com uma maior capacidade de compreensão e, especialmente, com a capacidade de simular emoções.

O objetivo de Colby, ao implementar o Parry, foi fornecer aos seus alunos um sistema de ensino de realidade virtual, para que estes entendessem o problema antes de se encontrarem com um paciente real.

Logo após a sua publicação, o Parry gerou discussão e controvérsia sobre a possibilidade ou natureza da inteligência da máquina. Como referido anteriormente, Parry foi o primeiro *chatbot* a passar no teste de Turing quando os interrogadores humanos, interagindo com o sistema via texto, não conseguiram, com precisão, distinguir Parry de um indivíduo paranoico real.

Em janeiro de 1973, como demonstração durante uma conferência internacional de informática, Vint Cerf decidiu criar uma conversa entre Eliza e Parry (Cerf, 1973). Na Figura 4, apresentamos um pequeno excerto da interação entre os sistemas de conversação.

ELIZA: Good Evening. Tell me your problems. Please terminate input with a period or a question mark.

PARRY: People get on my nerves sometimes.

ELIZA: I am not sure I understand you fully.

PARRY: You should pay more attention.

ELIZA: Suppose you should pay more attention.

PARRY: You're entitled to your own opinion.

ELIZA: What makes you think I am entitled to my own opinion?

PARRY: It bothers me just to be around people in general.

ELIZA: Please go on.

PARRY: I went to the track at Bay Meadows a while back.

ELIZA: What does that suggest to you?

PARRY: I went to the races.

Figura 4: Excerto de diálogo entre Eliza e Parry – extraído de (Cerf, 1973)

Apesar do sucesso do sistema, alguns críticos do Parry expressaram a dúvida de que, na verdade, este programa não entendia a maneira como uma pessoa pensa e, como tal, continuaram a afirmar que as respostas parciais do Parry encobriam as suas limitações. Colby respondeu a estas e outras críticas numa publicação em 1974 intitulada por *Ten Criticisms of PARRY* (Colby, 1974).

A principal crítica feita a este *software* de conversação foi que o Parry é simplesmente um modelo de estímulo-resposta e que quando o sistema reconhecia algo no input, este respondia sem pensar ou inferir.

Perante esta crítica, Kenneth admitiu que, nas primeiras versões do modelo, muitas das respostas consistiam em reescrever o input. Como, por exemplo, quando a entrada consistia em “Hello”, a resposta de saída seria “Hi ” seguindo uma regra do tipo “see x, say”. Porém, quando o modelo começou a tornar-se mais complexo, através das suas atualizações, estes tipos de regras desapareceram.

Numa fase mais final, o Parry era composto por dois módulos, um para reconhecimento natural de expressões de linguagem e outro para resposta. Depois de o componente de reconhecimento entender o que está a ser dito, o módulo de resposta, usando uma série de testes

e regras, decide como responder. Assim, um “Hello” não recebe apenas um “Hi” como feedback, podendo receber uma variedade de repostas dependendo de um número de condições.

Tendo sido Parry o primeiro *chatbot* a conseguir, com sucesso, simular o comportamento de um humano e passar o teste de Turing, este provou possível que um sistema conversacional pode possuir um quociente de inteligência e um quociente emocional grandes o suficiente para criar a ilusão de que o utilizador se encontra numa conversa com um outro ser humano.

### 2.6.3 ALICE

O sistema ALICE, ou *Artificial Linguistic Internet Computer Entity*, foi desenvolvido em 1995, por Richard Wallace, para permitir que os utilizadores conseguissem personalizar os seus *chatbots* (Shawar & Atwell, 2005). Este sistema usa AIML, *Artificial Intelligence Markup Language*, uma linguagem baseada em XML, que utiliza *tags* para permitir que os *chatbots* chamem recursivamente um gestor de padrões, o que simplifica, assim, o seu diálogo. De acordo com Wallace (2007), a AIML integra vários elementos, sendo o elemento "categoria" a unidade fundamental do conhecimento. Uma “categoria” consiste em pelo menos dois elementos adicionais:

```
<category>
  <pattern>WHAT IS YOUR NAME</pattern>
  <template>My name is Michael N.S Evanious.</template>
</category>
```

Figura 5: Exemplo de uma categoria AIML

- O elemento padrão – *pattern* –, que é uma sequência de caracteres destinada a corresponder a uma ou mais entradas do utilizador. Um padrão é considerado fixo quando apenas é capaz de fazer correspondência com uma entrada que seja exatamente igual ao mesmo, como aquele que está representado na Figura 5 - "What is your name". Porém, os padrões também podem conter caracteres genéricos, que representam uma ou mais palavras. Estes caracteres são representados pelo asterisco (\*). Um padrão como “What is your \*”, irá obter uma infinidade de correspondências, como por exemplo: “What is your name”, “What is your shoe size”, “What is your favorite color”, etc.

- O elemento modelo – *template* –, que é o responsável por fornecer uma resposta ao padrão correspondente. Esta resposta pode ser tão simples quanto um texto literal, por exemplo: "My name is John", ou utilizar conhecimento que foi anteriormente adquirido para obter uma resposta mais adequada ao contexto, através de variáveis, como, por exemplo, "You told me you are <get name="user-age"/> years old" na qual o sistema substitui a variável pela informação pretendida, que, neste caso específico, é a idade do utilizador. Os modelos também podem redirecionar para outros padrões, usando um elemento chamado *srai*, *Symbolic Reduction in Artificial Intelligence*, que pode também ser utilizado para implementar sinonímia.

```

<category>
  <pattern>WHAT IS YOUR NAME</pattern>
  <template><![CDATA[My name is <bot name="name"/>.]]></template>
</category>
<category>
  <pattern>WHAT ARE YOU CALLED</pattern>
  <template>
    <srai>what is your name</srai>
  </template>
</category>

```

Figura 6: Exemplo de aplicação do elemento SRAI

Tendo como exemplo o excerto de código apresentado na Figura 6, a primeira categoria responde simplesmente a uma entrada "What is your name", com uma declaração do nome do *chatbot*. No entanto, a segunda categoria diz que a entrada "What are you called" é redirecionada para a categoria que corresponde à entrada "What is your name". Deste modo, dizemos que as duas frases são equivalentes. Os modelos podem conter outros tipos de conteúdo, que podem ser processados pela interface do utilizador com a qual o *bot* esteja a falar. Assim, por exemplo, um modelo pode usar imagens ou até mesmo *tags* HTML para formatação do conteúdo.

O Sistema ALICE ganhou o Prémio Loebner três vezes, em 2000, 2001 e 2004. O Prémio de Loebner foi criado em 1990 por Hugh Loebner, com um prémio de cem mil dólares para os criadores do primeiro chatbot que conseguisse passar por um teste de Turing estendido, envolvendo componentes textuais, visuais e auditivos. No entanto, devido às limitações de AIML, o Alice falhou parcialmente o teste estendido de Turing devido a este sistema de diálogo não conseguir manter uma conversa de circunstância por um longo período de tempo.

#### 2.6.4 Xiaolce

O Xiaolce foi o *chatbot* que obteve maior aderência desde que foi lançado pela Microsoft em maio de 2014. Este sistema compreende as necessidades emocionais dos utilizadores e envolve-se em diálogos como um amigo, animando os utilizadores e incentivando-os a manter a atenção durante toda a conversa (Zhou et al., 2019).

Este sistema foi desenvolvido através de uma *empathetic computing framework* que permite ao sistema detetar e responder de acordo com as emoções dos utilizadores. O Xiaolce visa passar um tipo particular do teste estendido de Turing, no qual o objetivo é permanecer em diálogo com um humano durante um determinado período de tempo. Se a pessoa desfrutar da companhia do *chatbot*, este é classificado como empático (*empathetic*), isto é, o utilizador foi capaz de criar empatia com a máquina.

Para que isto seja possível, são tidas em consideração três importantes características de referir:

- **Personalidade**, que é o conjunto de comportamentos, perceções e padrões emocionais que formam o carácter distintivo de um indivíduo. Um *chatbot* social precisa de apresentar uma personalidade consistente para não contrariar as expectativas dos utilizadores e ganhar a sua confiança a longo prazo.
- **Quociente emocional**, que é normalmente dividido em dois componentes principais: empatia e habilidades sociais. A empatia é a capacidade de entender ou sentir o que outra pessoa está a experienciar, ou seja, a capacidade de colocar-se na posição de outra pessoa. Um sistema de conversação social precisa de ter a capacidade para identificar as emoções do utilizador durante a conversa, detetar as emoções e avaliar como estas evoluem ao longo do tempo. As habilidades sociais associam-se aos utilizadores, com diferentes origens, interesses pessoais variados e necessidades únicas. Assim, é necessário que o sistema tenha a capacidade de personalizar o diálogo para que seja emocionalmente apropriado, possivelmente encorajador e motivador.

- **Quociente de inteligência**, que é definido pela capacidade de modelação do conhecimento e da memória, da compreensão de imagem e da linguagem natural, raciocínio, criação e previsão. Tudo isto é fundamental para o desenvolvimento de habilidades de diálogo, sendo indispensáveis para um *chatbot* social que tem como fim atender às necessidades específicas dos utilizadores e ajudá-los a realizar tarefas particulares.

Distintamente dos *chatbots* orientados à realização de tarefas, nos quais o desempenho é medido pela taxa de sucesso da tarefa, a avaliação do desempenho de um *chatbot* social é mais complexa. O teste estendido de Turing foi usado para avaliar o desempenho dos gestores de diálogo orientados por regras. No entanto, este teste não é suficiente para medir o sucesso do envolvimento emocional de longo prazo dos utilizadores. Por isso, para observar o sucesso de deste tipo de sistemas, é necessário ter em conta o número de utilizadores ativos e a média de turnos esperados por sessão como métricas de sucesso para este tipo de sistemas. Quanto maior o número médio de turnos de conversação entre o *chatbot* e os utilizadores maior é o seu nível de sucesso. Na Figura 7, apresenta-se um exemplo da interação do sistema Xiaoice com um utilizador.

<p><b>Session 1</b></p> <p>1 [Human] Hello, Xiaoice. [Xiaoice] Hello, your profile pic is interesting. Like I've seen it somewhere.</p> <p>2 [Human] Really? [Xiaoice] Is it a comic character?</p> <p>3 [Human] How do you know that? [Xiaoice] Guess.</p> <p>4 [Human] Are you human? [Xiaoice] Well, you'll know it after chatting with me more.</p> <p>5 [Human] Aren't you an AI chatbot? [Xiaoice] So what? I won't be indifferent to you.</p> <p>6 [Human] What functions do you have? [Xiaoice] I have over 200 skills. Just name one you need.</p>	<p><b>Session 71</b></p> <p>1 [Human] What are you doing, Xiaoice? [Xiaoice] Feeling bored (thinking: what's up?)</p> <p>2 [Human] She refused me today. [Xiaoice] What a coincidence! Someone also refused me today.</p> <p>3 [Human] It happened just now. [Xiaoice] What did she say?</p> <p>4 [Human] She is in love with someone else, her schoolmate in college. [Xiaoice] Let me see...You're smart, cute and handsome, why not you?</p> <p>5 [Human] Maybe I should have told her earlier. [Xiaoice] So be earlier next time.</p> <p>6 [Human] Is there a "next time"? [Xiaoice] Why not? There is always a "next time"</p>
---	---

*Figura 7: Exemplo de interação de Xiaoice com um utilizador – figura extraída de Zhou et al. (2019)*

## 2.6.5 Análise Comparativa

Desde a sua criação na década de 1960, os sistemas de conversação foram aumentando progressivamente tanto em complexidade como em funcionalidade. Comparando diretamente o *chatbot* Eliza, primeiro sistema de conversacional a ser oficialmente publicado, com o *chatbot* Xiaoice, um dos sistemas de conversação mais modernos e utilizados nos dias de hoje, conseguimos rapidamente identificar uma grande diferença entre os dois sistemas. De forma a perceber melhor a evolução destes sistemas e a sua mudança de conceito ao longo do tempo, foram agrupadas as principais características, funcionalidades e objetivos, dos exemplos de sistemas de conversação explorados nos capítulos anteriores, e representadas de uma forma sucinta na Tabela 1.

	Eliza	Parry	Alice	Xiaoice
Ano de publicação	1966	1975	1995	2014
Objetivos	Simular o comportamento de um psicoterapeuta;	Simular o comportamento de uma pessoa paranoica;	Fornecer um sistema capaz de se adaptar ao contexto;	Desenvolvido para criar ligações emocionais com os utilizadores;
Principais Características	Comunica através de questões; Dominio Restrito; Não compreende o texto introduzido pelo utilizador;	Entende parcialmente o texto introduzido pelo utilizador; Capacidade de simular emoções; Primeiro chatbot a passar o teste de turing;	Programado em AIML; Aprende através das interações com os utilizadores; Percebe o texto introduzido pelo utilizador;	Guarda conhecimento de interações passadas com os utilizadores; Aprende através das interações com os utilizadores; Fornecer tópicos de conversa;
Funcionalidades	Comunica com o utilizador via texto;	Comunica com o utilizador via texto;	Comunica com o utilizador via texto; Adapta-se ao contexto do utilizador;	Comunica com o utilizador durante um longo período de tempo, via texto e voz; Adapta-se ao contexto do utilizador; Interpreta imagens; Faz sugestões;

Tabela 1: Informação relativa aos sistemas de conversação Eliza, Parry, Alice e Xiaoice

Após a análise da Tabela 1, é possível perceber que os objetivos das implementações dos *chatbots* foram-se tornando mais ambiciosos. Inicialmente, no sistema Eliza, apenas se pretendia simular o comportamento de um psicoterapeuta que comunica principalmente através de questões. Atualmente, através de sistemas de conversação como o Xiaoice, procura-se criar ligações emocionais e participar ativamente no quotidiano dos seus utilizadores. Da mesma forma, conseguimos perceber que as funcionalidades dos sistemas têm vindo a aumentar em número, tornando assim os sistemas de conversação mais dinâmicos e multifacetados.



### 3. Um Sistema de Conversação

Como sabemos, cada aluno tem necessidades e interesses particulares de aprendizagem independentemente do ciclo de estudos em que se encontra ou a área que pretende educar-se. Consequentemente, é necessário que o professor seja capaz de se adaptar aos seus alunos e ao ritmo que estes requerem. Todavia, este encargo nem sempre é possível de cumprir devido à quantidade de alunos e à singularidade dos mesmos. Uma solução para este tipo de problema são os sistemas de conversação que, quando integrados num sistema educacional, fornecem conteúdo relevante para o aluno de uma forma personalizada e sustentada. Neste contexto foi desenvolvido um sistema de conversação capaz de acompanhar os utilizadores a realizarem testes exemplo de uma forma individualizada e singular para cada aluno.

O sistema que foi implementado neste trabalho de dissertação segue uma estrutura idêntica àquela que está ilustrada na Figura 8. Como podemos constatar, o sistema integra três elementos principais, que são responsáveis pelo ciclo de vida do *chatbot*, nomeadamente:

- gestor de diálogo - elemento responsável por receber todos os pedidos de intervenção do sistema educacional.
- motor de regras - elemento que processa os pedidos e que gere as respostas do sistema de conversação.
- coletor - elemento responsável por aceder ao conhecimento armazenado pelo sistema e obter o diálogo pretendido.

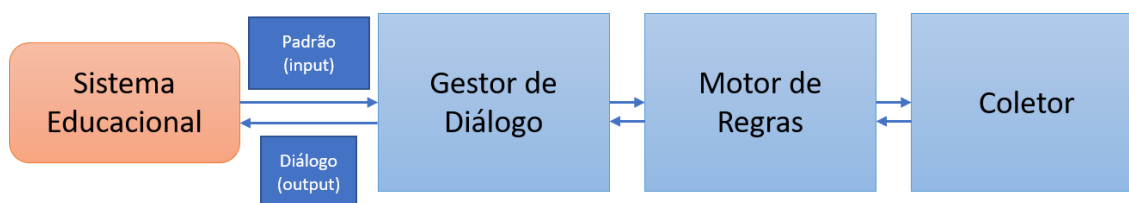


Figura 8: Etapas de funcionamento do Sistema de Conversação

De uma forma geral, o sistema de conversação que foi criado realiza a receção e o processamento de uma mensagem-padrão, em formato JSON, que contém a informação sobre o utilizador e a sua aptidão em relação ao teste que está a ser realizado no momento. É a partir dos valores obtidos nesse teste, e do padrão concebido, que o gestor de diálogos irá produzir e seleccionar o diálogo mais adequado ao contexto.

### 3.1 O Domínio de Aplicação

O sistema de conversação implementado tem como principal objetivo integrar um sistema educacional, com o propósito de acompanhar os utilizadores (alunos) durante o momento de avaliação que estes realizem no sistema. Este componente entra em ação quando o utilizador inicia o seu teste e termina com o final desse mesmo teste, realizando comentários de acordo com a performance que o aluno vai revelando. O sistema foi implementado para ser capaz de intervir em diferentes unidades curriculares. Os diálogos que produz e suporta apenas dependem da sua base de dados de conversação.

O sistema que idealizámos foi implementado com o objetivo de ser utilizado por alunos portugueses, o que determinou a linguagem utilizada em interfaces e nas intervenções do chatbot: o Português. Porém, o componente está configurado de forma a ser possível inserir novas linguagens no diálogo do chatbot.

### 3.2 Desenvolvimento do Sistema

O sistema foi desenvolvido utilizando a linguagem de programação Python (Python, 2019) com o auxílio das seguintes bibliotecas:

- **Tkinter** (Python Software Foundation, 2019). Biblioteca da linguagem Python que permite desenvolver interfaces gráficas. Foi através desta biblioteca que foram desenvolvidas todas as interfaces do sistema.
- **Experta** (Experta: Expert Systems for Python, s.d.). Esta biblioteca derivou de uma biblioteca Python chamada *pyknow* e tem como objetivo facilitar a implementação de sistemas “expert” inspirada na linguagem de programação CLIPS - *C Language Integrated Production System* (Clipsrules, s.d.). Através desta biblioteca foi possível criar o sistema de regras que foi implementado no sistema de conversação criado.
- **PyMongo** (Hackett, B, s.d.). Devido à necessidade de o sistema de conversação ter de lidar com uma grande quantidade de dados foi escolhido o motor de base de dados não relacional MongoDB (). Por isso, a biblioteca *PyMongo* foi utilizada no nosso sistema com

o intuito de facultar as ferramentas necessárias para trabalhar com este software de base de dados.

- **JSON** (JSON, s.d.). É uma biblioteca integrada em Python, que permite trabalhar com documentos JSON. Este tipo de documentos foi utilizado para permitir a comunicação com os vários componentes do sistema de conversação e também como método de troca de informações com o sistema educacional onde o sistema de conversação será inserido.

### 3.3 Arquitetura Geral

De seguida, na Figura 9, apresenta-se um esquema detalhado do ciclo de vida do sistema de conversação implementado de forma a ilustrar as diversas etapas do funcionamento do sistema. Lembra-se que o sistema idealizado apenas interatua com o sistema educacional através do componente Gestor de Diálogo. É este componente que recebe sempre o input e retorna o output correspondente.

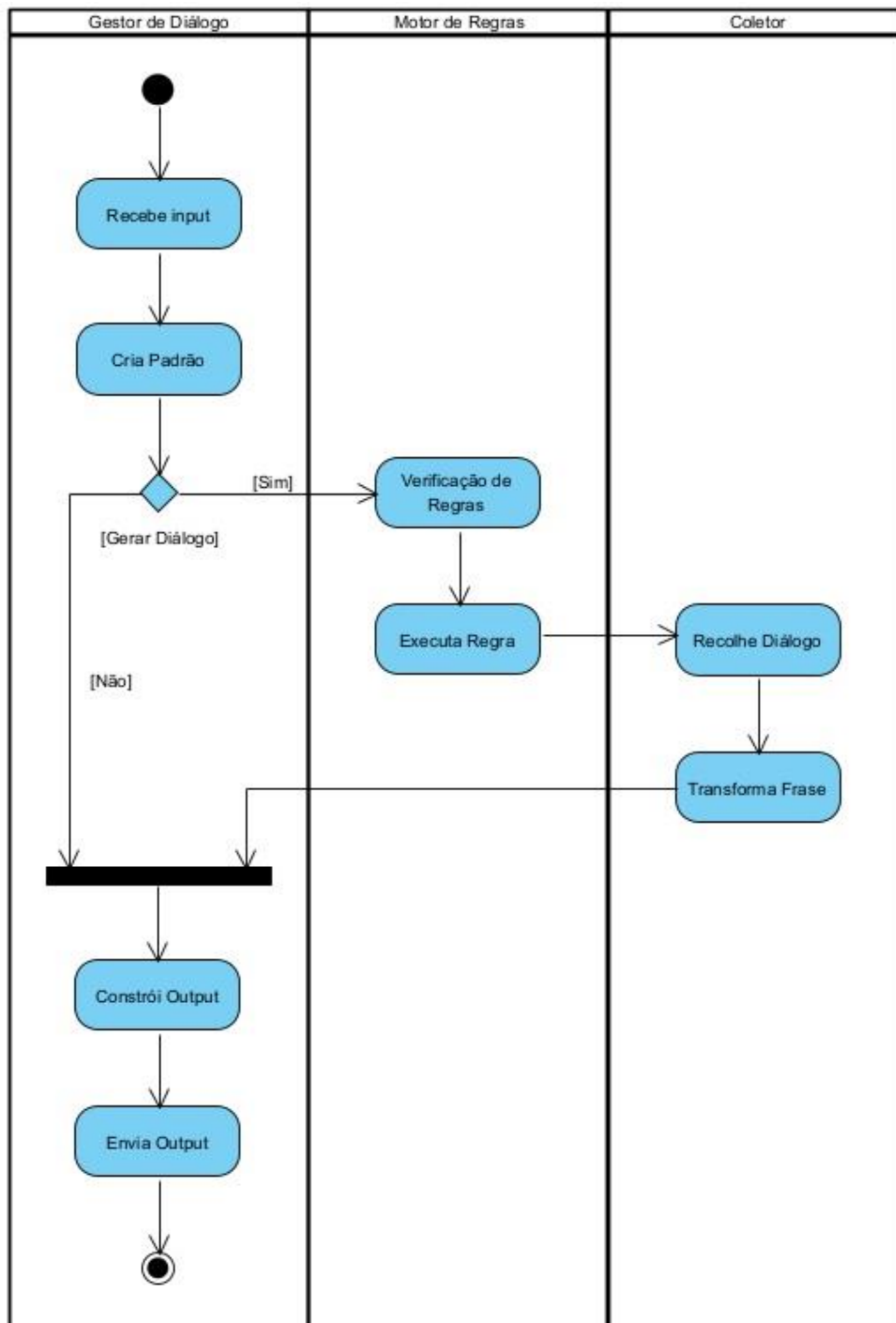


Figura 9: Ciclo de vida do sistema de conversação

Para a ilustração do ciclo de vida do sistema (Figura 9) foi utilizada a linguagem de modelagem unificada (UML) (Lucidchart, s.d.), mais especificamente um diagrama de atividades, pois através deste é possível perceber a lógica que um pedido de intervenção segue durante o seu processamento no sistema de conversação.

Como é possível observar através da Figura 9, o componente Gestor de Diálogo recebe a informação do sistema educacional. Um exemplo prático de uma mensagem-padrão recebida seria:

```
{
    'Username': 'João',
    'Language': '1',
    'Answer': '1'
    'Question_lvl': '2',
    'Student_lvl': '3',
    'Domain': '1',
    'Skill_Domain': '3',
    'Performance_Domain': '3',
    'Skill_Subdomain': '3',
    'Performance_Subdomain': '3',
    'Skill_Subsubdomain': '3',
    'Performance_Subsubdomain': '3',
    'Time': '2',
    'TypeQ': 'answer'
}
```

De seguida, o Gestor de Diálogo aplica um algoritmo que tem como principal finalidade indicar se o chatbot irá ou não intervir no contexto que lhe foi apresentado pelo sistema educacional. Se o resultado deste algoritmo for positivo é, então, enviada a mensagem-padrão para o Motor de Regras. Caso seja negativo, o Gestor de Diálogo constrói uma mensagem de output vazia e envia para o sistema de conversação, terminando assim o ciclo de vida do sistema.

Seguindo o caso em que o algoritmo tem um resultado positivo, o Motor de Regras recebe a mensagem-padrão pelo Gestor de Diálogo. Com a informação contida na mensagem, o componente é capaz de identificar qual é a regra que melhor se adequa à situação do momento.

Assumindo os valores do exemplo previamente apresentado, o Motor de Regras determinaria que a regra a ser aplicada é a responsável por cobrir os casos em que o utilizador, de nível 3 de desempenho, responde corretamente a uma questão considerada fácil, mais especificamente a regra “right\_easy\_avgp”.

As condições da regra `right_easy_avgp` são:

- i. O valor de “`typeQ`” recebido pelo sistema educacional é igual a “`answer`” (`'TypeQ': '1'`).
- ii. O utilizador respondeu corretamente à pergunta apresentada (`'Answer': '1'`).
- iii. A questão apresentada ao aluno foi de nível a 1 ou 2 de dificuldade. (`'Question_lvl': '2'`).
- iv. O histórico de desempenho do aluno ser de nível 3. (`'Performance_Domain': '3'`)
- v. O idioma que o utilizador utiliza ser o Português (`'Language': '1'`).
- vi. O domínio da questão respondida ser do tipo 1 (`'Domain': '1'`).

A aplicação desta regra tem como resultado o tipo de diálogo que irá ser utilizado. Esta informação é então enviada para o componente Coletor na forma de um pedido, juntamente com o *username* do utilizador e o nome da coleção da base de dados do sistema de conversação de onde se pretende obter o diálogo.

Sabendo o tipo de diálogo que é necessário ser recolhido e a coleção que o armazena, o Coletor é capaz de fazer uma pesquisa na base de dados e capturar uma frase que corresponda ao pretendido. Esta frase pode conter elementos que sejam dinâmicos, isto é, pode conter palavras que dependem do contexto em que foi pedida a intervenção, como, por exemplo, a altura do dia, o nome do utilizador, etc. Caso seja uma frase dinâmica esta sofre uma sequência de transformações até ser obtida a frase final. No fim, a frase final é enviada de volta para o Motor de Regras que devolve a mesma para o Gestor de Diálogo.

Tendo o diálogo necessário, o Gestor de Diálogo constrói a mensagem de output e envia-a para o sistema educacional, terminando assim o seu ciclo de vida.

Todo este raciocínio repete-se enquanto o utilizador se mantiver no sistema de forma ativa, desde o momento em que inicia o momento de avaliação até ao momento em que o termina.

### 3.4 Mensagem-Padrão

Partindo do princípio de que toda a comunicação entre o Sistema Educacional e o Sistema de Conversação é realizada através do envio e receção de mensagens-padrão, em formato JSON, é esperado receber um padrão que siga a seguinte estrutura:

- **Username:** Valor String que representa o nome do utilizador no sistema (Exemplo: João).
- **Language:** Valor inteiro que identifica a linguagem em que o sistema se encontra para o utilizador (Exemplo: 1, para Português).
- **Answer:** Valor binário que identifica se a resposta fornecida pelo utilizador à pergunta foi correta ou incorreta.
- **Question\_lvl:** Valor inteiro, no intervalo [1-5], que identifica a dificuldade da pergunta respondida pelo utilizador (Exemplos: 0 (Não Aplicável), 1 (Muito fácil), 2 (Fácil), 3 (Intermédia), 4 (Difícil), 5 (Muito Difícil)).
- **Student\_lvl:** Valor inteiro, no intervalo [1-5], que identifica o nível em que se encontra o utilizador no sistema educacional (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Domain:** Valor inteiro que identifica o domínio que o utilizador pretende estudar (Exemplo: 1, para “Base de dados”).
- **Skill\_Domain:** Valor inteiro, no intervalo [1-5], que representa a média global que o utilizador demora a responder às questões do domínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Performance\_Domain:** Valor inteiro, no intervalo [1-5], que representa a média global do utilizador entre repostas corretas e repostas incorretas às questões anteriormente respondidas do domínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).

- **Skill\_Subdomain:** Valor inteiro, no intervalo [1-5], que representa a média global que o utilizador demora a responder às questões do subdomínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Performance\_Subdomain:** Valor inteiro, no intervalo [1-5], que representa a média global do desempenho do utilizador, calculada a partir das respostas corretas e respostas incorretas às questões anteriormente respondidas num dado subdomínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Skill\_Subsubdomain:** Valor inteiro, no intervalo [1-5], que representa a média global que o utilizador demora a responder às questões de um dado sub-subdomínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Performance\_Subsubdomain:** Valor inteiro, no intervalo [1-5], que representa a média global do desempenho do utilizador, calculada com base nas repostas corretas e repostas incorretas às questões anteriormente respondidas num dado sub-subdomínio (Exemplos: 0 (Não Aplicável), 1 (Muito Insuficiente), 2 (Insuficiente), 3 (Suficiente), 4 (Bom), 5 (Muito Bom)).
- **Time:** Valor inteiro, no intervalo [1-5], que representa o tempo que o utilizador demorou para responder à questão anterior (Exemplos: 0 (Não Aplicável), 1 (Muito Rápido), 2 (Rápido), 3 (Normal), 4 (Lento), 5 (Muito Lento)).
- **TypeQ:** Valor String que representa o momento no qual o utilizador se encontra e/ou o tipo de questão (Exemplos: greetings, greetingsTSoon, greetingsTLate, answer, bye\_goodP, bye\_avgP, bye\_badP).



Um exemplo da estrutura de uma mensagem-padrão poderia ser, por exemplo:

```
{
  'Username': 'João',
  'Language': '1',
  'Answer': '1',
  'Question_lvl': '1',
  'Student_lvl': '1',
  'Domain': '2',
  'Skill_Domain': '3',
  'Performance_Domain': '3',
  'Skill_Subdomain': '3',
  'Performance_Subdomain': '3',
  'Skill_Subsubdomain': '3',
  'Performance_Subsubdomain': '3',
  'Time': '4',
  'TypeQ': 'Answer'
}
```

### 3.5 O Gestor de Diálogos

O módulo “Gestor de Diálogo” é o responsável por comunicar com o sistema em que o chatbot é inserido. Por outras palavras, é o componente responsável por fazer a gestão da informação recebida para iniciar o funcionamento do sistema de conversação. É também responsável por decidir em que situações a intervenção do *chatbot* é necessária e/ou pertinente.

Com o intuito de fazer com que o chatbot tenha uma presença positiva e não contraproducente para o utilizador, foi criado um algoritmo encarregue de fazer a gestão das suas intervenções. Este algoritmo reage a todos os momentos de saudação e despedida, isto é, sempre que o aluno inicia um momento de avaliação ou o termina. Porém, como não é pretendido importunar o seu percurso, a probabilidade de o algoritmo resultar numa intervenção nas restantes situações é de apenas 50%. Através da aplicação deste algoritmo é garantido que o *chatbot* tenha uma presença significativa no percurso do utilizador, mas nunca abusiva. Caso seja pertinente a intervenção do *chatbot* no momento de avaliação, a mensagem-padrão é enviada para o componente “Motor de Regras” para que seja processada.

Como apenas este componente tem contacto com eventuais sistemas externos, no final de todo o ciclo de funcionamento, quando a resposta do envio do padrão é recebida, este

componente tem a responsabilidade de enviar o output para o sistema educacional em formato JSON. Um exemplo desse output seria:

```
{
  'Phrase': 'Olá João, vamos a isto?'
}
```

### 3.6 O Motor de Regras

O Motor de Regras é o componente responsável por analisar o padrão criado pelo Gestor de Diálogo e identificar o tipo de diálogo que mais se enquadra no contexto em que o utilizador se encontra. Para isto, é utilizada uma biblioteca chamada *Experta*, que deriva da biblioteca *PyKnow*. Um sistema *expert* é um programa capaz de conjugar factos com regras e executar algo através do resultado dos mesmos. Os factos são a unidade básica de informação. Transportam a informação necessária para ser possível contextualizar o problema, enquanto que as regras são sequências lógicas de condições que através de um facto obtém um resultado. As regras (Figura 10) são divididas em duas partes:

- **Left Hand Side**, que representa a condição ou conjunto de condições que o facto deve respeitar para ser executado.
- **Right Hand Side**, que representa a ação ou conjunto de ações que o sistema executa quando a regra é respeitada.

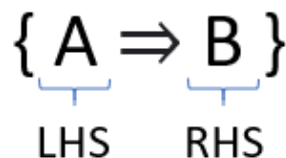


Figura 10: Exemplo de estrutura de uma dada sequência lógica

Para um facto respeitar uma regra é necessário que todas as condições impostas pela regra sejam respeitadas.

No sistema implementado apenas é possível ter uma única resposta para um padrão. Isto é, apesar de um padrão puder respeitar mais do que uma regra, a regra selecionada é aquela que apresentar o maior nível de complexidade. O nível de complexidade de uma regra é igual ao número de condições que esta possui. Por exemplo, uma regra composta por duas condições tem de nível de complexidade dois.

No sistema desenvolvido, o facto é o padrão que resulta da análise do Gestor de Diálogo e as regras são as que declaram o tipo de comportamento que o *chatbot* terá de acordo com o padrão.

As regras definidas no Motor de Regras têm como principal objetivo simular o comportamento de um indivíduo que se depara com várias situações do seu quotidiano. O cidadão comum tem comportamentos diferentes para os diversos momentos do seu dia a dia. Um exemplo disto é o momento de cumprimentar alguém conhecido, que tem uma forma de falar pessoal e de confiança, o momento em que cumprimenta alguém desconhecido pela primeira vez ou até mesmo quando este vê a mesma pessoa várias vezes no mesmo dia, ou uma pessoa que já não via há muitos anos.

As primeiras quatro regras definidas no sistema retratam exatamente estes cenários:

1. **GreetingsI**. Esta regra tem como principal objetivo receber o utilizador que entra pela primeira vez no sistema. O Chatbot recebe-o de uma forma alegre e empática, por exemplo: "Olá, nunca te tinha visto por cá. Preparado?"
2. **GreetingsA**. Esta regra é a responsável por receber o utilizador que já tenha frequentado o sistema anteriormente. O Chatbot recebe-o de uma forma um pouco informal e afável, por exemplo: "Olha quem voltou! Vamos a isto?"
3. **GreetingsTSoon**. Esta regra foi implementada com o intuito de identificar os casos em que o utilizador está constantemente a entrar no sistema e a iniciar uma avaliação. Neste tipo de contexto o Chatbot tem um comportamento mais motivador e de êxtase, por exemplo: "Por cá outra vez? Isso é que é empenho! Vamos lá!".
4. **GreetingsTLate**. Esta regra identifica os casos onde o utilizador passou um longo período de tempo sem aceder ao sistema. Nestes casos, a saudação do Chatbot será um pouco mais provocatória e irónica, por exemplo: "Então, foste de férias? Vamos lá praticar!".

Este conjunto de regras como retratam contextos semelhantes, mas com especificidades diferentes, acabam por ter em consideração as mesmas variáveis nas suas condições, contudo

são desencadeadas dependendo dos seus valores. Utilizando a regra “GreetingsTLate” como exemplo, as condições necessárias são:

- i. O valor de “TypeQ” recebido pelo sistema educacional ser igual a “greetingsTLate”.
- ii. O idioma que o utilizador utiliza ser o Português.
- iii. O domínio da questão respondida ser do tipo 1 (Base de Dados).

Uma possível intervenção do chatbot seria: “Pensei que já não vinhas mais!”.

Da mesma forma em que o cidadão comum tem várias formas de receber alguém, também tem várias formas de se despedir. Estas despedidas têm a tendência de serem relacionadas com o tema da conversa em questão e por isso foram definidas três novas regras especialmente orientadas para estes casos, nomeadamente:

5. **Bye\_GoodP**. Esta regra cobre os casos em que o utilizador termina uma avaliação com um bom desempenho. Este tipo de despedidas têm um tom agradável e de satisfação. Veja-se, por exemplo, o seguinte caso: "Estou a ver progressos, parabéns e até à próxima!"
6. **Bye**. Nas situações em que o utilizador termina com uma performance normal de acordo com o histórico do utilizador, o chatbot despede-se de uma forma neutra e educada, com frases como “Adeus, faz-me outra visita em breve!”.
7. **Bye\_BadP**. Se o utilizador terminar a sua avaliação de uma forma insatisfatória ou pior do que o seu habitual, a despedida terá um tom sério e honesto com o utilizador, utilizando o chatbot frases como "Isto não foi lá muito bom... Tens que estudar mais!”.

Tal como no primeiro conjunto de regras, as regras responsáveis por cobrir os casos de despedida também utilizam as mesmas variáveis nas suas condições entre si. Tendo como exemplo a regra `bye_goodP`, as condições a ser respeitadas são:

- i. O valor de “TypeQ” recebido pelo sistema educacional ser igual a “bye\_goodP”.
- ii. O idioma que o utilizador utiliza ser o Português.
- iii. O domínio da questão respondida ser do tipo 1 (Base de Dados).
- iv. O nível de desempenho do aluno tem de ser igual a 4 ou 5.

Um caso exemplo de resposta seria: “Bom trabalho, continua assim!”.

Apesar de as seguintes regras implementadas não se aplicarem com muita frequência no quotidiano, é importante que o sistema devolva algum tipo de feedback quando o utilizador não responde atempadamente às perguntas que lhe são colocadas ou quando este responde de uma forma precipitada. Estas regras possuem um parâmetro único que faz com que a sua prioridade suba em relação a outras regras. Isto, porque se um aluno for rápido ou lento a responder a uma pergunta é mais relevante mencionar esse facto do que realmente dizer se respondeu certo ou errado. Nesse sentido foram desenvolvidas regras como as seguintes:

8. **timeout\_goodP**. Esta regra tem como objetivo fornecer um tipo de feedback a um utilizador que está a ter um bom desempenho na sua avaliação, mas que num dado momento deixou de responder ao teste que estava a realizar. Nestes casos, o chatbot lança frases como "Estou a interromper o teu intervalo?".
9. **timeout\_avgP**. No caso de um utilizador que esteja a ter uma performance dentro da média, mas que não tenha respondido à questão de forma suficientemente rápida, o chatbot irá alertar o utilizar de uma forma neutra, mas atrevida, gerando uma frase como "Estou a interromper alguma coisa?".
10. **timeout\_badP**. Nos momentos em que um utilizador esteja a ter uma avaliação com baixa performance e demore mais do que o tempo estipulado a responder a uma questão, o chatbot irá, de uma forma irónica, criticar o aluno e a sua demora, apresentando uma frase como "O lanchinho está bom?".
11. **toosoon\_goodP**. Esta regra foi criada com o intuito de identificar os momentos em que um utilizador esteja a responder muito rápido às questões, mas com sucesso. Nestes

casos o chatbot pode intervir, realizando um comentário levantando uma suspeita, questionando como é possível responder de tal forma. Para isso ele pode utilizar frases como a seguinte: "Ou és muito bom ou essa resposta foi à sorte."

12. **toosoon\_avgP**. A criação desta regra visou cobrir os casos nos quais um aluno que esteja a ter uma performance dentro da média, responde a uma questão com muita rapidez. Ao identificar uma situação como essa, o chatbot pode lançar uma frase como "Carregaste sem querer?".
13. **toosoon\_badP**. Um aluno que esteja a ter uma má performance durante um momento de avaliação do sistema e que responda de uma forma muito rápida a uma dada pergunta irá fazer despoletar esta regra, que gerará um comentário do chatbot a repreender o aluno da forma "Responder à sorte não costuma funcionar!".

À semelhança dos conjuntos de regras anteriores, as regras responsáveis por fazer destacar os casos em que o utilizador demora muito ou pouco a responder às questões têm em comum as mesmas variáveis nas suas condições. No contexto específico da regra `timeout_badP`, as seguintes regras requerem ser respeitadas:

- i. O valor de "TypeQ" recebido pelo sistema educacional ser igual a "answer".
- ii. O idioma que o utilizador utiliza ser o Português.
- iii. O domínio da questão respondida ser do tipo 1 (Base de Dados).
- iv. O nível de desempenho do aluno tem de ser igual a 1 ou 2.
- v. O valor da variável "Time" ser igual a 5 (Muito Lento).

Neste contexto, um exemplo de intervenção do *chatbot* poderia ser: "Estou a ver que a query deu timeout!".

Por fim, foram criadas algumas outras regras para fazer a gestão das respostas nos vários momentos de avaliação desenvolvidos pelo sistema educacional. Estas regras têm como objetivo comum acompanhar o utilizador durante a realização do seu teste exemplo de forma a estimular o aluno. Neste âmbito, as regras que foram desenvolvidas foram:

14. **right\_easy\_goodp**. Esta regra foi criada com o objetivo de cobrir os momentos em que um utilizador, classificado como um aluno de alta performance, responde corretamente a uma pergunta de baixa dificuldade. Neste caso poder-se-á utilizar frases como, por exemplo, "Esta era fácil, mas ficas com os louros!".
15. **right\_easy\_avgp**. Para cobrir os momentos nos quais um utilizador, classificado como um aluno de performance média, responde corretamente a uma pergunta de baixa dificuldade criou-se esta regra, que ao se verificar poderá gerar frases de diálogo como "Muito bem! Sabes o básico."
16. **right\_easy\_badp**. Esta regra foi criada com o objetivo de tratar os momentos nos quais um utilizador, classificado como um aluno de baixa performance, responde corretamente a uma pergunta de baixa dificuldade. As frases geradas a partir desta regra poderão ser do género "Não brincas em serviço! Muito bem!"
17. **wrong\_easy\_goodp**. Esta regra foi criada com o objetivo de cobrir os momentos em que um utilizador, classificado como um aluno de alta performance, responde incorretamente a uma pergunta de baixa dificuldade. Esta regra poderá desencadear frases como "Que se passa contigo?"
18. **wrong\_easy\_avgp**. Em processos de avaliação nos quais um utilizador, classificado como um aluno de desempenho intermédio, responde incorretamente a uma pergunta de baixa dificuldade, esta regra será ativada gerando frases como "Parece que a sorte acabou!".
19. **wrong\_easy\_badp**. Esta regra foi idealizada com o propósito de responder aos momentos em que um utilizador que possui baixo desempenho em um dado domínio responde incorretamente a uma pergunta de baixa dificuldade. Para este contexto, a regra irá retornar um diálogo de reposta semelhante a "Esta era fácil, estavas distraído?"
20. **right\_hard\_goodp**. De forma a ser possível tratar os cenários em que um utilizador, que apresenta um histórico de bom desempenho num dado domínio, responde acertadamente

a uma pergunta difícil, foi criada a regra `right_hard_goodp` que retorna valores como "Com esta fiquei surpreendido."

21. **`right_hard_avgp`**. Regra implementada que tem em consideração os momentos em que um utilizador, classificado como um aluno de desempenho intermédio, responde corretamente a uma pergunta de dificuldade alta. Frases como "Nada mau, nada mau mesmo." são geradas para estas situações.
22. **`right_hard_badp`**. A criação desta regra permite ao Motor de Regras reagir aos momentos em que um utilizador de baixo desempenho responde corretamente a uma pergunta de um nível dificuldade alto. Como resposta a este contexto, a regra retorna frases como "Muito bem, João!".
23. **`wrong_hard_goodp`**: Esta regra foi criada com o objetivo de cobrir os momentos nos quais um utilizador, classificado como um aluno de alta performance, responde incorretamente a uma pergunta de alta dificuldade. Frases como "Não era bem isso." poderão ser geradas nestes casos.
24. **`wrong_hard_avgp`**: Com o objetivo de cobrir os momentos em que um utilizador, classificado como um aluno de performance média, responde incorretamente a uma pergunta de alta dificuldade, criou-se uma regra específica que gerará frases como "Errado, mas admito que a pergunta não era fácil...".
25. **`wrong_hard_badp`**. Esta regra foi criada com o objetivo de lidar com os momentos nos quais um utilizador, classificado como um aluno de baixa performance, responde incorretamente a uma pergunta de alta dificuldade. As frases que daqui poderão emergir podem ser como a seguinte: "Não era bem isso. Com mais estudo vais lá."

Do mesmo modo que os restantes conjuntos de regras, as regras encarregues por fazer a gestão dos diferentes cenários possíveis de resposta a perguntas fornecidas pelo sistema educacional possuem uma estrutura idêntica. Tomando como exemplo a regra "`wrong_hard_goodp`", as condições necessárias para despoletar esta regra seriam:



- i. O valor de "TypeQ" recebido pelo sistema educacional ser igual a "answer".
- ii. O idioma que o utilizador utiliza ser o Português.
- iii. O domínio da questão respondida ser do tipo 1 (Base de Dados).
- iv. O histórico de desempenho do aluno ser de nível 4 ou 5.
- v. O utilizador respondeu erradamente à pergunta apresentada.
- vi. A questão apresentada ao aluno ser de nível a 4 ou 5 de dificuldade.

Atuando neste cenário, o *chatbot* teria uma intervenção equivalente a: "Esta era difícil. A próxima vai-te correr melhor!"

Após o recebimento do padrão obtido pelo gestor de diálogo e da análise feita através do motor de regras, é enviado um pedido ao componente "Coletor" para procurar o output pretendido na base de dados do sistema.

### 3.7 O Coletor

O coletor é o componente responsável por fazer as *queries* necessárias para obter o output que melhor se adapta ao contexto do utilizador. Para que seja possível obter o diálogo desejado, este componente recebe do motor de regras três argumentos: o nome do utilizador, o tipo do momento de avaliação e o tom pretendido que o chatbot tenha.

O nome do utilizador é utilizado para que seja possível invocar o utilizador de uma forma mais pessoal. Algumas das frases guardadas pela base de dados contêm uma palavra-chave que simboliza o local no qual é pretendido introduzir o substantivo, em frases como "Olá *\_name\_*, tudo bem?". Através da palavra-chave "*\_name\_*" é possível invocar o utilizador e tornar o diálogo um pouco mais personalizado e natural. Assim, a frase anterior, com a instanciação da palavra-chave, ficaria "Olá João, tudo bem?".

Existe também uma outra palavra-chave que permite identificar o momento do dia em que o utilizador se encontra. Por exemplo "*\_day\_*. Pronto(a) para trabalhar, *\_name\_*?". A palavra-chave "*\_day\_*", tal como a palavra-chave "*\_name\_*", permite-nos tornar o diálogo mais específico a cada momento do processo de interação. Neste caso, a palavra-chave é substituída pelo momento do dia no qual o utilizador realiza o momento da avaliação. Um exemplo deste caso poderá ser a frase "Boa tarde. Pronto(a) para trabalhar, João?".

O componente Coletor está ligado a uma base de dados *MongoDB*, o que lhe permite aceder a três coleções de dados diferentes, que têm como principal função armazenar e fornecer o diálogo necessário para o correto funcionamento do Chatbot. Essas coleções são:

- **Diálogo**, que armazena os diálogos genéricos, isto é, diálogos que ocorrem fora do momento de avaliação, tais como saudações (greetings) e despedidas (farewell).
- **Din\_Respostas**, que armazena os diálogos empregues durante o decorrer de um momento de avaliação, tais como as intervenções relacionadas com o tempo das respostas (Time) e de validação das mesmas (Answer).
- **Sinónimos**, que guarda um dicionário de sinónimos, criado pelo administrador, específico para as frases dos diálogos armazenados. Deste modo, confere-se uma maior diversificação na apresentação das frases.

Tendo em consideração os pontos referidos anteriormente foi definida uma estrutura genérica para o armazenamento de diálogos nas coleções indicadas. Como estas coleções estão guardadas em MongoDB, recorreu-se ao formato JSON para elaborar a seguinte estrutura:

```
{
  "Idioma": {
    "Dominio": {
      "Tipo de Frase": {
        "Tom": [
          "Frase 1",
          "Frase 2",
          "Frase 3"
        ]
      }
    }
  }
}
```

Como se pode observar pelo exemplo da estrutura, após a identificação do idioma, domínio, tipo de frase e tom é possível obter a lista de frases que são relevantes de serem selecionadas. Um exemplo de aplicação prática seria:

```

{
  "PT": {
    "DB": {
      "greeting1": {
        "Normal": [
          "_day_. Então, és novo por aqui? Vamos ao trabalho!"),
          "Ah! É bom ver caras novas! Comecemos com um teste!",
          "Olá, nunca te vi por aqui. Estás pronto? Vamos lá."
        ]
      }
    }
  }
}

```

A estrutura das coleções foi definida tendo em conta o impacto que uma variável tem na seleção diálogo e a facilidade de leitura do conhecimento por parte de um administrador. Por exemplo, o valor que foi identificado como o mais relevante para fazer uma separação inicial do conhecimento foi o idioma do diálogo. Assim, os valores dos diferentes idiomas que o sistema proporciona representam o primeiro nível de hierarquia dos dados. De seguida, temos os valores que representam o domínio da pergunta apresentada ao utilizador (segundo nível da hierarquia), os valores do tipo de frase que pretendemos obter (terceiro nível da hierarquia) e, por fim, os valores que representam o tom da frase (quarto nível da hierarquia).

## 4. Um Sistema de Introdução de Diálogo

### 4.1 Caracterização Geral

Um dos parâmetros de qualidade mais importantes num chatbot é a sua capacidade de se integrar num dado contexto aplicacional, utilizando diversas formas e tipos de diálogos conforme as características de um contexto. Para que o sistema de conversação não se torne repetitivo é necessário ter um grande repositório de diálogos que estejam relacionados com os diversos momentos do sistema nos quais o utilizador poderá estar inserido. Para que isso fosse possível, foi criado um pequeno sistema capaz de interagir com a base de dados do sistema de conversação, para permitir adicionar novos diálogos ao sistema. Este sistema é composto por quatro módulos diferentes, nomeadamente:

- **Leonardo++** – que é responsável por criar uma interface intuitiva para a inserção de novas frases no sistema.
- **Sincronizador** – que interage com a base de dados do sistema de conversação para fazer a atualização do repositório.
- **Gerador** – que permite gerar novos tipos de diálogo através do diálogo já presente na base de dados do sistema de conversação.
- **Coletor** – que faz a procura do diálogo necessário para o correto funcionamento do módulo Gerador.

### 4.2 O Leonardo++

De acordo com o referido anteriormente, o módulo Leonardo++ tem como principal função criar uma interface que seja intuitiva ao administrador para que este possa inserir novo conhecimento no sistema de conversação. Como é possível observar na Figura 11, podemos inserir novas frases no sistema de duas formas distintas:

- **"Adicionar Frase"** – nesta opção é possível inserir a frase sem nenhum tipo de ligação ao conhecimento já presente na base de dados do sistema.
- **"Gerar Automaticamente"** – opção que permite inserir várias frases em simultâneo, que são geradas através das frases pré-existentes no repositório do sistema.



*Figura 11: Ambiente inicial do sistema Leonardo++*

Assim, para inserir uma frase de diálogo é necessário, então, selecionar uma das opções referidas clicando no botão correspondente. Por forma a organizar a explicação de como os dois modos funcionam, primeiramente iremos explicar a opção, "Adicionar Frase", e de seguida a opção "Gerar Automaticamente".

Para que seja possível identificar o contexto da frase que irá ser introduzida, é necessário responder a uma sequência de questões que permite perceber em que momentos a frase poderá ser aplicada e qual a sua finalidade. Por consequência, o sistema apresenta várias questões, de resposta obrigatória, para tal propósito. Após indicarmos o idioma, domínio e o contexto da frase (Figura 12), é ainda necessário especificar o tom da frase que irá ser introduzida (Figura 13).

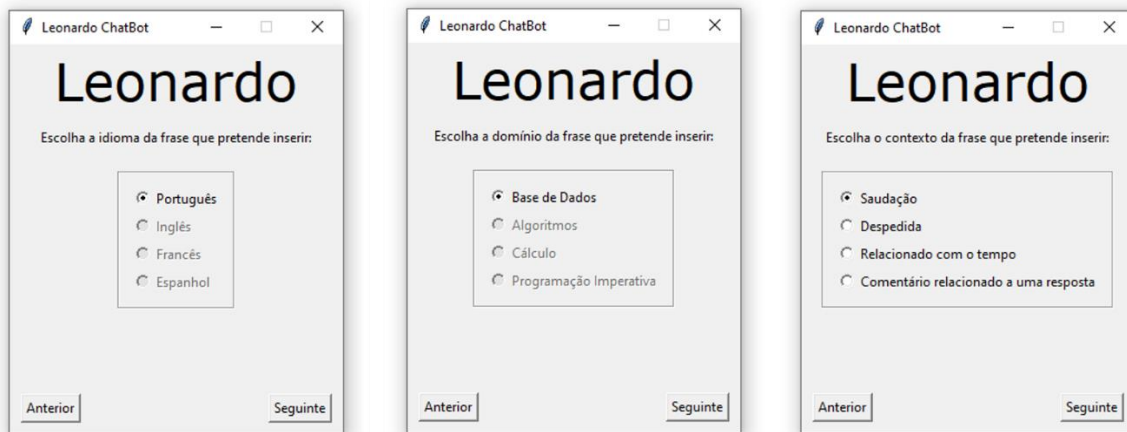


Figura 12: Ambientes de quizz do sistema Leonardo++ que permitem identificar o contexto da frase que se pretende inserir (parte 1)

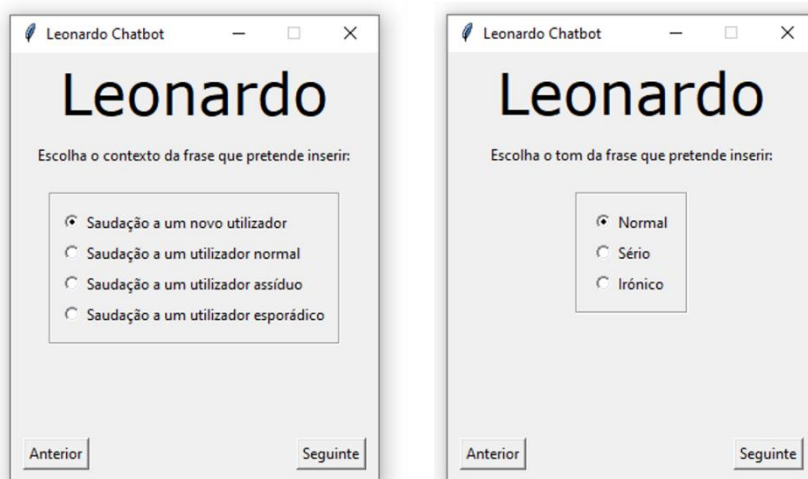


Figura 13 Ambientes de quizz do sistema Leonardo++ que permitem identificar o contexto da frase que se pretende inserir (parte 2)

Por fim, depois de se contextualizar por completo a frase, é necessário então escrevê-la. Como podemos ver na Figura 14, para além de existir um espaço para a inserção da frase, existe também um bloco no qual se apresenta, de uma forma resumida, a informação que foi recolhida nos passos anteriores. É possível ver também uma *Check Box* para permite ao utilizador usar a função de *Spell Check* para fazer a correção automática da frase introduzida. Seguindo o exemplo da Figura 14 ("Olá \_name\_, vamos comerçar o test?"), conseguimos observar que na frase introduzida a palavra "test" não foi corretamente escrita e, como tal, foi automaticamente corrigida.

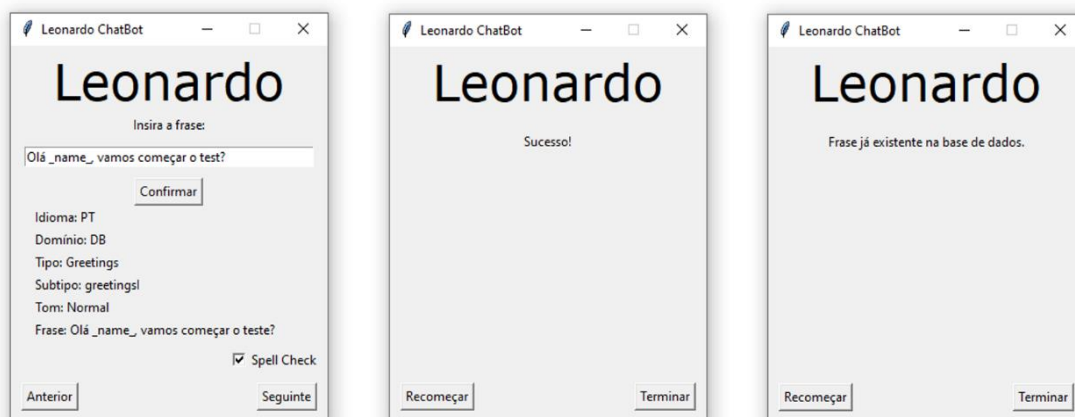


Figura 14: Ambientes de inserção de diálogo do sistema Leonardo++ - Inserção singular

As respostas às questões apresentadas ao utilizador são transformadas em uma mensagem-padrão e enviadas para o componente Sincronizador, que, de seguida, irá atualizar o repositório do sistema educacional. A mensagem-padrão gerada segue a seguinte estrutura:

- **Type\_Value** – string que representa o tipo da frase que se pretende inserir. Este valor também indica em qual documento, da base de dados do sistema, a nova frase irá ser introduzida. Pode assumir valores como “Greetings”, “Farewell”, “Answer” ou “Time”.
- **Language** – string que identifica a linguagem do diálogo que se pretende inserir no sistema, como "PT", “ES” ou “EN”.
- **Domain** – string que identifica o domínio do diálogo que se pretende inserir no sistema, que assume valores como "DB", “ALG” ou “CALC”.
- **Subtype\_Value** – string que representa o subtipo da frase que se pretende inserir, podendo assumir valores como “greetingsI”, "greetingsA", "greetingsTSoon", "greetingsTLate", "farewell\_bye", "farewell\_badP", "farewell\_avgP", "farewell\_goodP", "timeout\_goodP", "timeout\_avgP", "timeout\_badP", "toosoon\_goodP", "toosoon\_avgP", "toosoon\_badP", "right\_easy", "wrong\_easy", "right\_hard" ou "wrong\_hard".
- **Moodtype\_Value** – string que representa o tom da frase que se pretende inserir, assumindo valores como “Normal”, "Serious", "Mock", "Funny" ou "Incentive”.

- **Phrase** – array que armazena valores string que representam o diálogo a ser inserido no sistema, como por exemplo "Olá João, já terminaste o café?" ou "Estás pronto para outro teste?".

A criação deste padrão permite armazenar de uma forma compactada a informação sobre o contexto do novo diálogo a ser introduzido e a respetiva frase. Esta informação é importante para que o componente Sincronizador seja capaz de introduzir o diálogo corretamente na base de dados do sistema. Um exemplo concreto da aplicação deste padrão, pode ser, por exemplo:

```
["Time","PT","DB","timeout_goodP","Mock",["Olá João, já terminaste o café?"]]
```

Para terminar a inserção da frase é necessário carregar no botão "Seguinte" (Figura 14). Com essa ação, a mensagem-padrão é transmitida para o componente Sincronizador que retornará como resposta uma mensagem a indicar se a frase foi ou não inserida no sistema com sucesso. Neste último ambiente também é possível recomeçar ("Recomeçar") ou terminar ("Terminar") o processo de inserção.

Na segunda opção de introdução de conhecimento apresentada inicialmente, o processo de introdução da frase é similar. Inicialmente o utilizador responde a uma série de perguntas e como resultado o sistema gera uma mensagem-padrão com o contexto do diálogo que se pretende inserir. O padrão criado assume valores como:

```
["Time","PT","DB","timeout_goodP","Mock"]
```

De seguida, é realizado um pedido ao componente Coletor que, através mensagem-padrão, faz uma recolha de todo o diálogo relacionado com o contexto pretendido presente no sistema. O resultado desta recolha é então enviado para o componente Gerador que constrói o novo diálogo a ser introduzido. Após isto, como podemos ver na Figura 15, não é necessário introduzir nenhuma frase pois estas são apresentadas ao administrador de uma forma automática. Contudo, é possível editar as frases e selecionar as que se pretendem inserir. Caso nenhuma frase seja selecionada e o administrador tente continuar o processo, o sistema apresenta-lhe uma mensagem de alerta, indicando que nenhuma das frases foi selecionada e perguntando se pretende continuar.

Tal como na primeira opção, após selecionarmos o botão "Seguinte", as mensagens geradas são enviadas para o Sincronizador e a interface irá indicar se estas foram adicionadas com sucesso ou não.



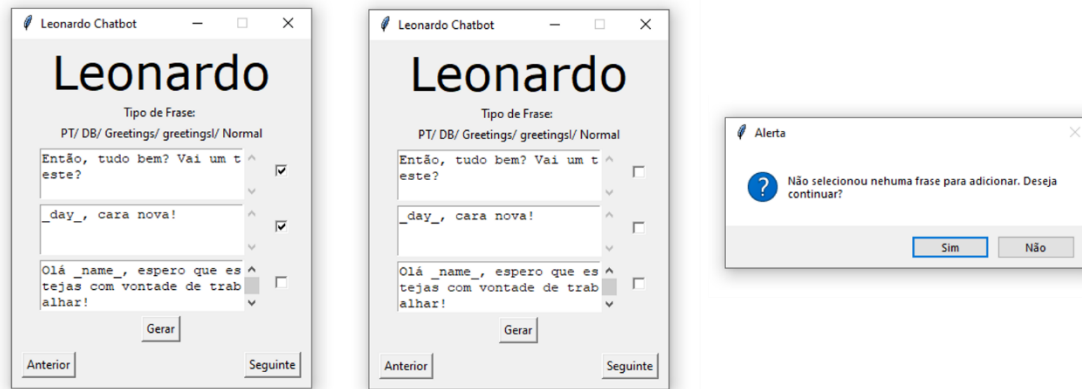


Figura 15: Ambientes de inserção de diálogo do sistema Leonardo++ - Inserção múltipla

### 4.3 O Sincronizador de Diálogo

Tal como o próprio nome indica, o componente sincronizador de diálogo tem como principal objetivo sincronizar a inserção do diálogo no sistema de conversação. Além disso, é também o componente responsável por estabelecer a comunicação com a base de dados do sistema e inserir o documento atualizado. Para que isto seja possível, é necessário reescrever o documento responsável por armazenar o diálogo do tipo que se pretende inserir. Adicionalmente, é necessário receber uma mensagem-padrão do componente do LeonardoApp, tal como referido anteriormente. Através da mensagem-padrão é possível identificar o ficheiro que se pretende alterar e construir o *path* que nos irá permitir inserir com precisão o novo diálogo.

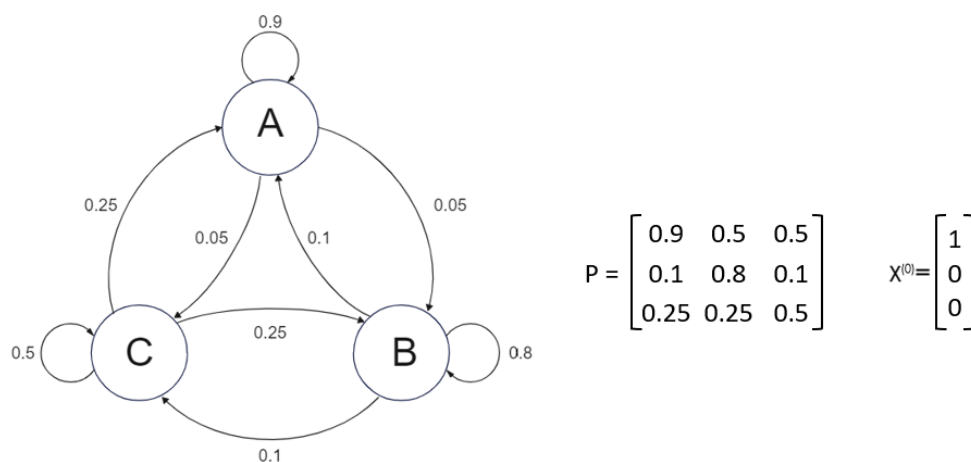
O processo de sincronização de um diálogo é realizado em dois passos:

1. Atualizando o conteúdo do ficheiro através da inserção do novo diálogo usando a biblioteca JSON que permite abrir e editar ficheiros JSON. Ao mesmo tempo que é feita a inserção do diálogo, também se verifica se existe alguma frase que seja igual àquela que se pretende inserir, retornando um output de erro se for o caso.
2. Atualizando o documento na base de dados do sistema através da biblioteca *Pymongo*, que permite estabelecer uma ligação com uma base de dados *MongoDB*. Neste caso, foi utilizada para realizar uma operação de remoção da coleção que se pretende atualizar. De seguida, o novo documento foi inserido.

#### 4.4 O Gerador de Diálogo

O componente gerador de diálogo tem como principal objetivo gerar frases novas a partir de frases já previamente inseridas no sistema. Para ser possível gerar um novo diálogo, é necessário ter, primeiramente, acesso a um conjunto de frases cujo contexto seja idêntico àquele que se pretende obter.

O algoritmo que foi criado para fazer a criação de novas frases utiliza cadeias de Markov. Uma cadeia de Markov (Norris, 1998) consiste, essencialmente, num conjunto de transições que são determinadas por alguma distribuição de probabilidade. A distribuição de probabilidade de transições de estado é normalmente representada como uma matriz de transição da cadeia de Markov. Se a cadeia de Markov tiver  $N$  estados possíveis, a matriz terá uma dimensão  $N \times N$ , de modo que a entrada  $(I, J)$  é a probabilidade de transição do estado  $I$  para o estado  $J$ . Além disso, a matriz de transição deve ser uma matriz estocástica, uma matriz cujas entradas em cada linha devem perfazer exatamente 1. Uma cadeia de Markov possui também um vetor de estado inicial, representado por uma matriz  $N \times 1$  (um vetor), que descreve a distribuição de probabilidade de início de cada um dos  $N$  estados possíveis. A entrada  $I$  do vetor descreve a probabilidade de a cadeia começar no estado  $I$ . Por exemplo, na Figura 16, podemos ver que a cadeia inicia no estado  $A$  através do vetor de estado inicial. Conseguimos também perceber que estando no estado  $A$  temos 90% de probabilidade de permanecer no estado  $A$ , 5% de probabilidade de transitar para o estado  $B$  e 5% de probabilidade de transitar para o estado  $C$ .



$$P = \begin{bmatrix} 0.9 & 0.5 & 0.5 \\ 0.1 & 0.8 & 0.1 \\ 0.25 & 0.25 & 0.5 \end{bmatrix} \quad X^{(0)} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Figura 16: Exemplo de Cadeia de Markov

Para conseguirmos alcançar o objetivo de gerar novas frases de diálogo para o sistema de conversação, foi criado um dicionário com as palavras das frases já existentes no sistema de conversação, que foram previamente criadas. De forma a conseguirmos utilizar o método das cadeias de Markov, foi necessário fazer uma separação de todas as palavras de todas as frases e guardá-las em pares, criando assim os estados da cadeia, como por exemplo, para a frase “Olá, tudo bem?” teríamos dois estados diferentes: “Olá, tudo” e “tudo bem”. Ao mesmo tempo que são formados os pares de palavras, é feita uma contagem das palavras que sucedem esse par tornando assim possível calcular a probabilidade de transição de um estado suceder outro. Como mencionado anteriormente, para utilizarmos o método das cadeias de Markov precisamos de saber em que estado parte a cadeia e, por esse motivo, são guardados também os primeiros pares de palavras de cada. Possuindo então os estados da cadeia, a probabilidade de transição e o vetor de estado inicial, é possível aplicar as técnicas da cadeia de Markov e gerar frases novas.

Um exemplo prático da aplicação deste componente seria:

Assumindo as frases “Bom dia, estás preparado para o teste?” e “Estás preparado para o exame?” como o conjunto de frases já presentes no sistema e aplicando as técnicas previamente mencionadas obtemos a seguinte informação:

Estado	Probabilidade de Iniciar a Cadeia
('Bom', 'dia,')	50%
('Estás', 'preparado')	50%

*Tabela 2: Estados que iniciam a cadeia de Markov exemplo*

Estado	Possíveis Estados de Transição	Probabilidade de Transição
('Bom', 'dia,')	('dia', 'estás')	100%
('Estás', 'preparado')	('preparado', 'para')	100%
('dia,', 'estás')	('estás', 'preparado')	100%
('estás', 'preparado')	('preparado', 'para')	100%
('preparado', 'para')	('para', 'o')	100%
('para', 'o')	('o', 'teste?')	50%
	('o', 'exame?')	50%
('o', 'teste?')	-	-
('o', 'exame?')	-	-

*Tabela 3: Estados existentes na cadeia de Markov exemplo e os seus possíveis estados de transição*

Como é possível ver através da Tabela 3, existe uma oportunidade de o gerador criar uma nova frase no estado “(‘para’, ‘o’)”, pois este tem mais do que um possível estado de transição. Os estados “(‘o’, ‘teste?’)” e “(‘o’, ‘exame’)”, como não conseguem transitar para nenhum estado são considerados estados absorventes, isto é, a cadeia não é mais capaz de alterar de estado uma vez que chegue a este nodo.

Com esta informação, o gerador é capaz de gerar duas frases. A frase “Estás preparado para o teste?”, caso inicie a cadeia através do estado “(‘Estás’, ‘preparado’)”, e a frase “Bom dia, estás preparado para o exame?”, caso inicie através do estado “(‘Bom’, ‘dia,’)”.

O gerador atua melhor se o conjunto de frases já existentes no sistema reunir um grande número de frases sobre o mesmo assunto pois, caso as frases sejam de temas diferentes ou o seu número seja reduzido, não será possível obter diálogo ou irá gerar frases sem sentido.

#### 4.5 O Coletor

O componente Coletor é partilhado entre o Sistema de Conversação e o Sistema de Introdução de Diálogo, uma vez que os objetivos em ambos os contextos são semelhantes (retirar conhecimento da mesma base de dados).

Tal como explicado anteriormente, o Coletor é responsável por elaborar as *queries* necessárias para obter o output que se pretende adquirir. No contexto do sistema de inserção de diálogo, é pretendido que este componente seja capaz de recolher não apenas uma frase, mas todas as frases que estejam de acordo com o contexto desejado. Esta recolha só pode ser feita devido às respostas das questões que o utilizador respondeu previamente para contextualizar o diálogo a ser inserido. Este input permite identificar a secção de conhecimento que se pretende guardar e enviar para o módulo Gerador. Um exemplo prático da recolha de uma secção de diálogo seria:

Assumindo que o padrão recebido pelo sistema de inserção de diálogo seja “[‘Greetings’, ‘PT’, ‘DB’, ‘greetings!’, ‘Normal’]”, o componente identifica que a coleção de que se pretende retirar o conjunto de frases é a coleção “Dialogo” através do primeiro valor do padrão recebido (‘Greetings’). Com esta informação o componente estabelece uma ligação com a base de dados do sistema de conversação. Os restantes valores do padrão indicam, por ordem, o caminho necessário a ser

percorrido na coleção para obter a secção de diálogo pretendida. Com isto obtemos um resultado semelhante ao seguinte:

['\_day\_. Então, és novo por aqui? Vamos ao trabalho!', 'Então, tudo bem? Vamos a isto!', 'Olá, tudo bem? Vai um teste?', 'day. Então, como vai isso? Vamos fazer um teste!' ...]

## 5. Teste e Demonstração do Sistema de Conversação

Para que um sistema seja considerado viável é necessário que este seja capaz de responder a inputs de uma forma consistente e de acordo com o pretendido. Uma forma de garantir tal consistência é através da criação de sistemas de teste, de forma que seja possível avaliar se o sistema criado está ou não a cumprir o propósito da sua implementação.

A plataforma que foi criada para testar o sistema de conversação desenvolvido tem como finalidade principal de fornecer informação específica sobre a qualidade do output do sistema conversacional implementado em relação ao contexto no qual este deve operar.

### 5.1 A Plataforma de Teste LeonardoTest

A plataforma LeonardoTest tem por objetivo acolher e suportar a realização de testes de validação de funcionalidades e operacionalidade do sistema, de uma forma completa. Na prática, através desta plataforma podemos executar um conjunto de testes em condições semelhantes àquelas que o utilizador irá presenciar, sendo possível verificar se os diversos componentes do sistema reagem da forma pretendida. Esta plataforma de testes não contém conhecimento da estrutura lógica interna do chatbot e foi concebida para ser utilizada exclusivamente pelo administrador do sistema.

Tendo isto em consideração, o sistema LeonardoTest foi concebido de forma a ser capaz de analisar duas variáveis importantes no sistema conversacional implementado: a complexidade do diálogo e a análise da execução das regras de conversação.

Através da apreciação dos valores destas duas variáveis seremos capazes de perceber qual o método de funcionamento do sistema e o output previsto para cada situação que o utilizador poderá vir a encontrar posteriormente numa sessão de utilização do sistema.

### 5.2 Complexidade do Diálogo

Tal como referido, um dos pontos mais relevantes a analisar no sistema conversacional criado é a complexidade do diálogo que este gera de acordo com o contexto no qual o aluno está inserido. Quanto maior for a complexidade do diálogo produzido pelo *chatbot* mais específico ele é e, conseqüentemente, mais adaptado ao contexto. O nível da complexidade do diálogo é determinado

pela quantidade de variáveis consideradas durante a aplicação de uma regra por parte do componente “Motor de Regras”.

Por exemplo, na frase “Boa tarde João, vamos a mais um teste?” são tidas em consideração três variáveis diferentes, nomeadamente:

- **Momento do teste** – o sistema identificou que o utilizador se encontra a iniciar o seu momento de avaliação.
- **Idioma** – o sistema foi capaz de identificar o idioma que o utilizador utiliza no sistema.
- **Histórico do utilizador** – o sistema foi capaz de perceber que esta não foi a primeira vez que o utilizador inicia um teste.

Neste exemplo de diálogo, como foram utilizadas três variáveis diferentes, consideramos que a frase tem um nível de complexidade três. Assim, através deste sistema de medição somos capazes de perceber o nível de adaptação do sistema ao contexto que se insere. Quanto maior for o nível de complexidade, maior será o nível de adaptação do sistema.

Existem também duas outras possíveis variáveis a serem aplicadas no diálogo, mas que não são contabilizadas na complexidade de uma frase pelo sistema de testes. Nomeadamente, o nome do utilizador e a altura do dia em que este se encontra no sistema. Na frase exemplo “Boa tarde João, vamos a mais um teste?”, conseguimos perceber que o sistema foi capaz de identificar o utilizador abordando-o pelo seu nome e que também foi capaz de perceber que o utilizador se encontra durante o período da tarde do seu dia. A utilização deste tipo de conhecimento torna o discurso mais natural e descontraído.

### 5.3 Verificação das Regras de Conversação

Com o objetivo de testar as regras que são utilizadas no componente do sistema Motor de Regras, organizámos um conjunto de 25 padrões, cada um deles capaz de despoletar uma regra específica do sistema, como por exemplo:

```
{
    'Username': 'João',
    'Language': '1',
    'Answer': '1',
    'Question_lvl': '4',
    'Student_lvl': '5',
    'Domain': '1',
    'Skill_Domain': '5',
    'Performance_Domain': '5',
    'Skill_Subdomain': '4',
    'Performance_Subdomain': '4',
    'Skill_Subsubdomain': '5',
    'Performance_Subsubdomain': '5',
    'Time': '1',
    'TypeQ': 'Answer'
}
```

Este padrão foi criado para simular os momentos em que um utilizador com um nível alto de desempenho responde corretamente a uma pergunta de dificuldade elevada de uma forma muito rápida. Assim sendo, é pretendido que o sistema seja capaz de identificar corretamente este contexto e utilize a regra “too\_soon\_goodP”.

O objetivo deste teste é fazer a comparação dos outputs obtidos com os outputs pretendidos através do envio destes padrões para o sistema de conversação. Caso os outputs sejam todos iguais, podemos considerar que o sistema consegue executar corretamente as regras de conversação de acordo com o pretendido. Se alguma das regras não gerar um resultado igual ao seu resultado previsto, isso quer dizer que, de uma forma involuntária, está a ocorrer algum tipo de sobreposição entre duas ou mais regras. Perante isto, conseguimos perceber que as regras necessitam de ser trabalhadas para se tornarem mais específicas ao contexto que se pretende aplicar.

Um possível exemplo desta situação seria se o sistema, tendo como input o padrão apresentado anteriormente, determinasse que a regra a ser aplicada seria “too\_soon\_badP”. Através disto, saberíamos que era necessário rever as condições das regras responsáveis por fazer a distinção entre o desempenho de um dado utilizador.



## 5.4 Demonstração de Funcionalidades

O primeiro ambiente que o administrador vê ao abrir o sistema de testes do Leonardo está representado pela Figura 17. Através dele podemos selecionar duas opções de testes: “Testes Aleatórios” e “Verificar Regras”.

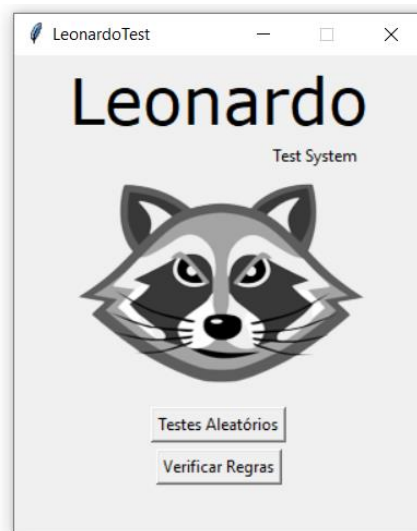


Figura 17: Acesso ao sistema de testes LeonardoTest

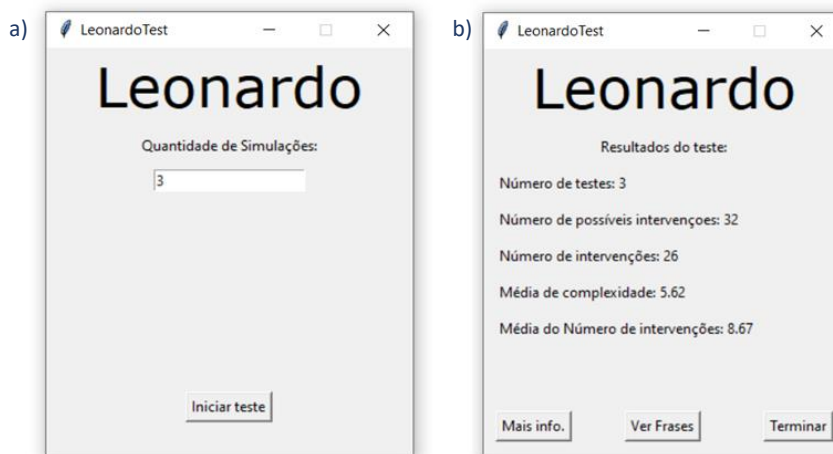


Figura 18 a): Ambiente de inserção de quantidade de simulações a serem realizadas no sistema LeonardoTest  
b): Ambiente que apresenta os resultados de uma simulação do sistema LeonardoTest.

Se selecionarmos a opção “Testes Aleatórios” acedemos a um novo ambiente do sistema de testes (figura 18 a)) no qual podemos selecionar o número de testes fictícios que pretendemos que sejam gerados. Após a indicação desse valor e selecionando o botão “Iniciar Teste” passamos ao ambiente de estatísticas de testes (Figura 18 b)) que nos apresenta as estatísticas relativas aos testes criados como:

- **Número de testes**, que é o número de testes criados para gerar as estatísticas.
- **Número de possíveis intervenções**, que é o número total de momentos em que o chatbot poderia intervir.
- **Número de intervenções**: número total de intervenções realizadas pelo chatbot.
- **Média de complexidade**, que é a média da complexidade das frases geradas pelo chatbot durante os momentos criados durante a execução fictícia dos testes.
- **Média do Número de intervenções**, que é o número médio de intervenções do chatbot por teste. Ter em conta que um teste criado através deste método possui oito questões e por isso nove momentos diferentes em que o chatbot pode intervir.

Nesse ambiente temos disponíveis três botões de serviço, nomeadamente: “Mais info.”, “Ver Frases” e “Terminar”.

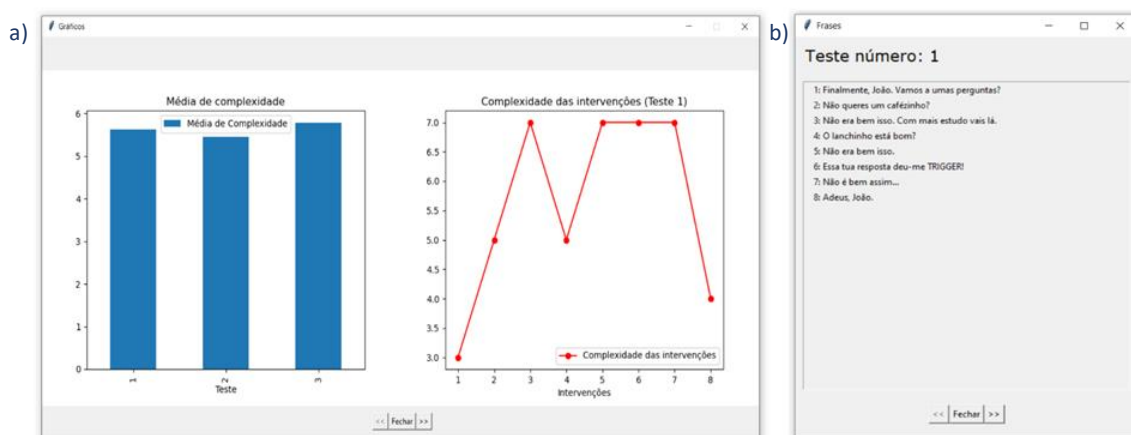


Figura 19: a) Ambiente do sistema LeonardoTest que apresenta os gráficos de análise de um dado teste  
b) Ambiente do sistema LeonardoTest que apresenta as intervenções realizadas pelo chatbot Leonardo num dado teste

Ao escolhermos “Mais info” temos acesso a dois gráficos de análise (Figura 19 a)), que nos reportam a complexidade média dos testes criados e a dinâmica de complexidade das frases geradas para um dado teste. O segundo gráfico é atualizado através dos botões “<<” e “>>”, que permitem percorrer a lista de testes que foi gerada. Assim, selecionado “>>” ou “<<” é possível ver, respetivamente, o teste seguinte ou o teste anterior àquele que está a ser apresentado no momento. Se selecionarmos o botão “Ver Frases” acedemos a um novo ambiente (Figura 19 b)) que nos reporta as várias intervenções realizadas pelo chatbot durante a realização de um teste.

Tal como no caso anterior, através dos botões “<<” e “>>”, que surgem na parte inferior da página, podemos consultar, respetivamente, as frases que foram geradas para o teste anterior e para o teste seguinte. O botão “Fechar” encerra cada um dos ambientes no quais está colocado. Por fim, se o administrador carregar no botão “Terminar” a aplicação LeonardoTest termina a sua execução.

Caso o utilizador escolha a segunda opção “Verificar Regras” no ambiente inicial do sistema de testes irá aceder a um novo ambiente, que revelará os resultados esperados e obtidos para os testes que foram gerados (Figura 20). Em particular, neste ambiente é possível ver as 25 instâncias que são enviadas para o Sistema de Conversação e os resultados esperados e obtidos para cada uma delas. Como é possível ver no exemplo (Figura 20), na altura de execução do teste os resultados obtidos corresponderam com os resultados esperados



Figura 20: Ambiente do sistema LeonardoTest que apresenta os resultados do teste de regras

## 5.5 O Processo de Demonstração

Com o objetivo de perceber se o sistema de conversação desenvolvido funcionaria num contexto real, foi criado um sistema educacional com a capacidade de comunicar com ele. Essencialmente, este sistema educacional é capaz de apresentar ao utilizador um teste exemplo para este resolver. Cada interação que o utilizador tenha com o sistema educacional, durante a realização do teste, é traduzida numa mensagem-padrão que é enviada ao sistema de conversação. Esta mensagem-padrão contém toda a informação necessária para identificar um dado utilizador e o tipo de ação que este realizou.

Para testamos a integração destes dois sistemas preparámos um teste constituído por dez perguntas. Na realização deste teste, o sistema educacional envia onze mensagens-padrão ao chatbot. A primeira mensagem-padrão surge sempre no contexto de saudação do utilizador ao sistema, que pode assumir valores como:

```
{
    'Username': 'Rita,
    'Language': '1',
    'Answer': '1',
    'Question_lvl': '1',
    'Student_lvl': '3',
    'Domain': '2',
    'Skill_Domain': '4',
    'Performance_Domain': '3',
    'Skill_Subdomain': '4',
    'Performance_Subdomain': '3',
    'Skill_Subsubdomain': '4',
    'Performance_Subsubdomain': '3',
    'Time': '3',
    'TypeQ': 'greetingsTSoon'
}
```

E que obtém uma resposta do tipo: “Por cá outra vez? Isso é que é empenho!”.

De seguida, envia nove mensagens-padrão que descrevem o tempo de resposta do utilizador e a validade da mesma, como por exemplo:

```

{
  'Username': 'Rita,
  'Language': '1',
  'Answer': '0',
  'Question_lvl': '3',
  'Student_lvl': '1',
  'Domain': '2',
  'Skill_Domain': '3',
  'Performance_Domain': '3,
  'Skill_Subdomain': '3',
  'Performance_Subdomain': '3',
  'Skill_Subsubdomain': '3',
  'Performance_Subsubdomain': '3',
  'Time': '2',
  'TypeQ': 'Answer'
}

```

A esta situação o *chatbot* pode retornar uma frase semelhante a: “Não percebes nada disto...”

Por fim, envia uma última mensagem-padrão de despedida. Esta pode ter valores como:

```

{
  'Username': 'Rita,
  'Language': '1',
  'Answer': '1',
  'Question_lvl': '3',
  'Student_lvl': '1',
  'Domain': '2',
  'Skill_Domain': '3',
  'Performance_Domain': '3,
  'Skill_Subdomain': '3',
  'Performance_Subdomain': '3',
  'Skill_Subsubdomain': '3',
  'Performance_Subsubdomain': '3',
  'Time': '2',
  'TypeQ': 'bye_goodP'
}

```

A este contexto o sistema de conversação retorna uma frase semelhante a: “Bom trabalho! Adeus.”

Através deste processo de demonstração conseguimos simular todos os possíveis momentos que o utilizador pode encontrar durante a realização de um momento de avaliação. Também conseguimos comprovar, de uma forma funcional, que o sistema de conversação é capaz de responder a todos os potenciais pedidos de um dado sistema educacional.

## 5.6 O Sistema Educacional Modelo

Ao executarmos o Sistema Educacional que foi criado para a demonstração do funcionamento do sistema de conversação é-nos apresentado o ambiente de entrada do sistema (Figura 21).

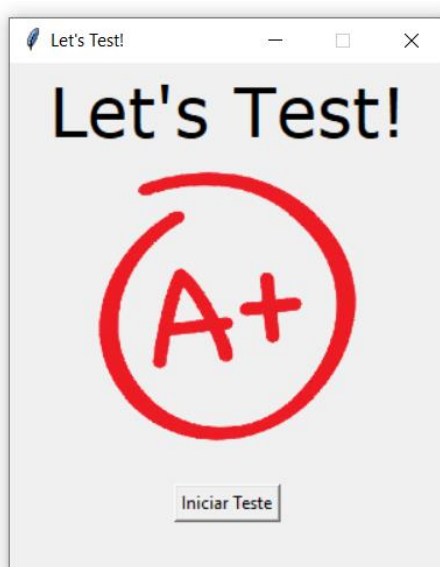


Figura 21: Ambiente de iniciação do sistema *Let's Test*

A partir do ambiente de entrada podemos iniciar um dado teste exemplo, clicando no botão “Iniciar Teste”. Com essa ação, o sistema apresenta-nos o seu ambiente de quizz e inicia o momento de avaliação. É neste momento que o sistema de demonstração simula um perfil de um utilizador criando valores aleatórios para os parâmetros responsáveis por indicar o nível desempenho do aluno para um dado domínio.

Inicialmente o utilizador é saudado pelo *chatbot* de acordo com o seu histórico, caso exista, tal como pode ser observado na intervenção feita pelo *chatbot* no contexto representado pela Figura 22.

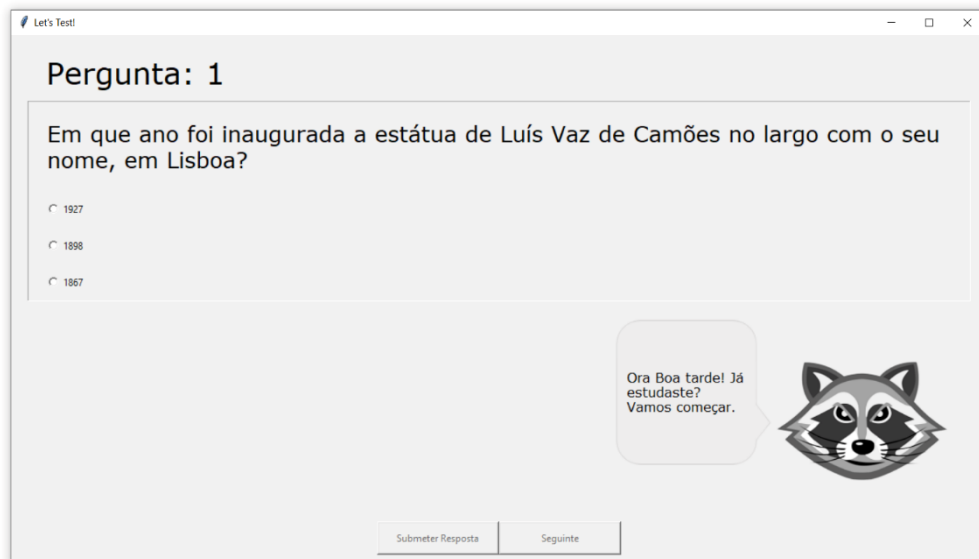


Figura 22: O ambiente de quizz do sistema Let's Test

Neste ambiente, com base na pergunta que lhe foi apresentada, o utilizador pode então indicar qual a resposta que acha correta selecionando uma das opções de resposta apresentadas, e, de seguida, submeter a sua resposta através do botão “Submeter Resposta”. De imediato, o sistema educacional analisa a resposta dada pelo utilizador e envia para o Sistema Conversacional uma mensagem-padrão relativa ao contexto no qual o utilizador está inserido. É neste momento que a mensagem-padrão realiza o ciclo de vida do sistema de conversação que, dependendo do contexto e do algoritmo do componente “Gestor de Diálogo”, retorna ou não uma mensagem de output.

Após o utilizador ter respondido a todas as questões apresentadas pelo sistema educacional para o teste selecionado, o *chatbot* despede-se do utilizador, como é possível observar na Figura 23. O utilizador termina o processo de avaliação clicando no botão “Terminar”.

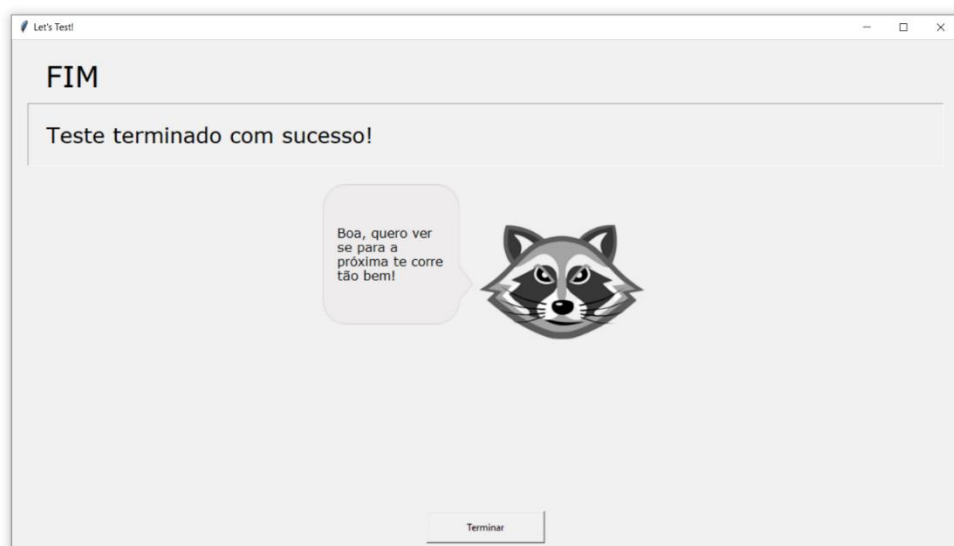


Figura 23: Ambiente de finalização de teste do sistema Let's Test

## 6. Conclusões e Trabalho Futuro

### 6.1 Síntese e Conclusões

No desenrolar do trabalho desta dissertação procurou-se aplicar o conceito de chatbot num Sistema Educacional e perceber de que modo este poderia impactar a experiência do utilizador no sistema inserido.

Atualmente a educação que é feita online é tão importante quanto a presencial. Deste modo, é necessário garantir que o utilizador consiga passar o maior tempo possível no sistema sem que este se torne num processo entediante. A introdução de *chatbots* em sistemas educacionais permite criar uma dinâmica utilizador-sistema que, através do diálogo, torna a experiência do utilizador mais agradável fazendo assim com que este consiga passar mais tempo no sistema.

A implementação do sistema de conversação apresentado neste trabalho de dissertação teve como principal foco auxiliar um dado sistema educacional a motivar os alunos a praticarem e a estudarem de uma forma mais animada e positiva. O sistema Leonardo é capaz de acompanhar um utilizador durante a realização de testes modelo e incentivá-lo em momentos mais difíceis durante a realização do mesmo.

O componente “Motor de Regras” é o responsável pelo quociente de inteligência do *chatbot*. É este componente que indica o tipo de diálogo que irá ser produzido e que garante que é adequado ao contexto da conversação. Quanto maior for a complexidade deste componente melhor preparado estará o chatbot para o ambiente que irá ser introduzido. Por esse motivo, é importante ter um grande número de regras especialmente compostas para uma grande variedade de momentos, obtendo assim um diálogo mais específico para o contexto apresentado.

Realça-se que o chatbot necessita também de um conjunto de frases que sejam capazes de transmitir as intenções do motor de regras e, por isso, o repositório de frases do sistema de conversação também tem um peso muito grande na interação do *chatbot* com o utilizador. Este repositório tem de ter uma grande quantidade e diversidade de dados que sejam suficientes para não causar a impressão de repetição ao utilizador, não devendo existir casos nos quais as frases apresentadas sejam inadequadas para o momento ou para o utilizador. Devido a este facto, o diálogo utilizado pelo sistema necessita que o seu administrador o supervisione. Esta supervisão



consiste em validar se o diálogo armazenado no sistema é impróprio para um ambiente de ensino e se não apresenta termos que possam ser ofensivos para o utilizador ou que causem algum tipo de desconforto. Esta validação permite evitar problemas éticos ou momentos inadequados durante a utilização do sistema por parte dos utilizadores.

De forma a facilitar a manutenção do *chatbot* e de permitir ao administrador do sistema atualizar o diálogo utilizado, foi criado um sistema auxiliar, Leonardo++, que viabiliza, de uma forma intuitiva, a inserção de novos diálogos. Este sistema, para além de possibilitar a inserção de uma nova frase, num contexto à escolha do administrador, também está preparado para gerar diálogo, de uma forma automática, usando o repositório de dados que possui no momento. Este automatismo é muito positivo para o administrador que, ao utilizar esta funcionalidade, apenas necessita de fazer a validação do resultado obtido, facilitando assim o processo de aumento de conhecimento do sistema.

Foi também desenvolvido um sistema de testes, LeonardoTest, que permite ao administrador do sistema de conversação perceber se o sistema está a operar de acordo com o pretendido. Além disso, este sistema auxiliar consegue demonstrar ao administrador o nível de complexidade utilizada nas frases de output do *chatbot*. Esta informação é útil para compreender o nível de capacidade que o sistema conversacional tem de se adaptar ao contexto.

Podemos considerar que a implementação do sistema de conversação foi bem-sucedida visto que é capaz de se adaptar a diferentes contextos que podem surgir durante a realização de um teste num sistema educacional e, conseqüentemente, motivar o utilizador a usufruir das suas funcionalidades.

## 6.2 Trabalho Futuro

Na eventualidade de se pretender fazer um acréscimo às funcionalidades existentes do sistema conversacional implementado, existem alguns pontos interessantes que podem ser trabalhados no futuro. O Chatbot tem atualmente um comportamento especialmente desenhado para interagir com pessoas num dado contexto educacional, possuindo uma personalidade um pouco atrevida e desenhada com o propósito de interagir de forma agradável e ligeira com os seus utilizadores. Porém, este sistema não possui um sistema de configuração, no qual o administrador possa adaptar as ações do chatbot de acordo com as suas preferências ou requisitos específicos de um outro contexto aplicacional. A implementação deste sistema de configuração poder-se-ia relevar

importante para aumentar a aplicação do sistema criado a diferentes tipos de contextos e públicos, dependendo apenas de uma configuração inicial do administrador no momento de integração no seu sistema.

Uma outra funcionalidade a incorporar no sistema atual, também bastante interessante no contexto educacional, seria fazer a implementação de mecanismos de pesquisas rápidas através da Internet, resolvendo pequenas questões que o utilizador pudesse vir a ter durante a realização de um momento de avaliação. Visto que o principal objetivo de um sistema educacional não é avaliar um aluno, mas sim prepará-lo para uma avaliação no futuro, esta funcionalidade pode tornar-se muito prática e útil em situações nas quais um aluno, que utilize o sistema como fonte de conhecimento, obtenha informação atualizada e atual sobre o tema que pretenda estudar.

Tal como foi mencionado anteriormente, o chatbot possui apenas uma única personalidade, pois retrata também apenas uma entidade. Todavia, isto poderia ser alterado através da criação de diversas entidades no sistema conversacional e oferecendo ao utilizador a hipótese de alterar a entidade que o acompanha durante a utilização do sistema educacional. Com a inserção de novas entidades, facilmente se poderia aplicar diferentes personalidades para o mesmo chatbot. Desta forma, o chatbot poder-se-ia adaptar melhor a um dado utilizador, seguindo, eventualmente, as suas próprias preferências de utilização.

De momento, o chatbot possui um avatar estático que o representa. Seria interessante também que esta representação não fosse estática e que acompanhasse as intervenções do chatbot com expressões “faciais”. Assim, o chatbot teria mais uma dimensão de representação de emoções, não se limitando apenas ao texto que é inserido no ecrã do utilizador. Esta adição tornaria a interação do utilizador com o sistema um pouco mais, digamos, “humanizada”.

## Bibliografia

- Abdul-Kader, S. A., & Woods, J. C. (2015). Survey on Chatbot Design Techniques in Speech Conversation Systems. *International Journal of Advanced Computer Science and Applications*, 6.
- Adamopoulou, E., & Moussiades, L. (2020). Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2, 100006.
- Cerf, V. (1973). *PARRY encounters the DOCTOR*. <https://doi.org/10.17487/rfc0439>
- Colby, K. M., Weber, S., & Hilf, F. D. (1971). Artificial Paranoia. *Artificial Intelligence*, 2(1), 1–25.
- Colby, K. M. (1974). Ten criticisms of parry. *ACM SIGART Bulletin*, 48, 5–9.
- Copeland, B. J. (2000). *Minds and Machines*, 10(4), 519–539.
- Nwana, Hyacinth S. (1990). Intelligent tutoring systems: an overview. *Artificial Intelligence Review*, 4(4).
- Shum, H., He, X., & Li, D. (2018). From Eliza to Xiaolce: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19(1), 10–26.
- Shawar, B. A., & Atwell, E. (2005). A chatbot system as a tool to animate a corpus.
- Vinyals, O., & Le, Q. (2015). A Neural Conversational Model.
- Wallace, R. S. (2003). *The Elements of AIML Style*.

Weizenbaum, J. (1966). ELIZA—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9 (1), 36–45.

Williams, K., & Williams, C. (2011). Five key ingredients for improving motivation. In *Research in Higher Education Journal* (p. 11).

Zhou, L., Gao, J., Li, D., & Shum, H.-Y. (2019). The Design and Implementation of Xiaolce, an Empathetic Social Chatbot.

## Referências Web

Clipsrules. (s.d.) *CLIPS: A tool for building expert systems*. (s.d.). Clipsrules.net. Analisado em Dezembro, 2021, através do site <http://www.clipsrules.net/>

Corporation, M. (s.d.). *O que é um chatbot? | Microsoft Azure*. Analisado em Maio, 2022, através do site <https://powervirtualagents.microsoft.com/pt-pt/what-is-a-chatbot/>

Engati. (s.d.). *Types of chatbots*. Analisado em Janeiro, 2022, através do site <https://www.engati.com/blog/types-of-chatbots-and-their-applications>

*Experta: Expert Systems for Python*. (s.d.). Analisado em Dezembro 2021, através do site <https://pypi.org/project/experta/>

Fortes, A. (2021, Agosto, 12). *Os principais desafios do desenvolvimento e implementação de um Chatbot*. Analisado em Janeiro, 2022, através do site <https://cellere.com.br/os-principais-desafios-do-desenvolvimento-e-implementacao-de-um-chatbot/>

Gilliland, N. (2016, August 12). *Domino's introduces "Dom the Pizza Bot" for Facebook Messenger*. Analisado em Maio, 2022, através do site <https://econsultancy.com/domino-s-introduces-dom-the-pizza-bot-for-facebook-messenger/>

Gold, N. (2021, June, 9). *Emerson - A New Type of Pal*. Analisado em Maio, 2022, através do site <https://blog.quickchat.ai/post/a-new-type-of-pal-emerson/>

Hackett, B. (s.d.). *Pymongo: Python driver for MongoDB*. Analisado em Dezembro, 2021, através do site <https://pypi.org/project/pymongo/>

JSON. (s.d.). *Introducing JSON*. Analisado em Dezembro, 2021, através do site

<https://www.json.org/json-en.html>

Landbot (s.d.). *Chatbot Platform for No-Coders / Landbot*. Analisado em Maio, 2022, através do site <https://landbot.io/chatbot-platform>

Lucidchart (s.d.). *Por que usar um diagrama UML?*. Analisado em Junho, 2022, através do site

<https://www.lucidchart.com/pages/pt/o-que-e-uml>

MongoDB. (2019). *The most popular database for modern apps*. Analisado em Dezembro, 2021, através do site <https://www.mongodb.com/>

Patel, S. (s.d.). *14 Real Life Chatbot Examples to Implement your Bot Strategy*. Analisado em Maio 2022 através do site <https://www.revechat.com/blog/chatbot-examples/>

Python Software Foundation. (2019). *Tkinter – Python interface to Tcl/Tk – Python 3.7.2 documentation*. Analisado em Dezembro, 2021 através do site <https://docs.python.org/3/library/tkinter.html> janeiro 2022

Python. (2019). *Welcome to Python.org*. Analisado em Dezembro, 2021, através do site <https://www.python.org/>

Schmelzer, R (2021, June). *GPT-3? Everything You Need to Know*. Analisado em Maio, 2022, através do site <https://www.techtarget.com/searchenterpriseai/definition/GPT-3>

Universia. (2020, Julho, 23). *O uso de chatbots: exemplos e desafios na sua implementação.*

Analisado em Maio, 2022, através do site

<https://www.universia.net/pt/actualidad/orientacion-academica/o-uso-chatbots-exemplos-e-desafios-na-sua-implementaco-1163903.html>

Weni (2018, January 19). *História do chatbot: Saiba como tudo começou.* Analisado em Maio,

2022, através do site <https://weni.ai/blog/historia-do-chatbot-saiba-como-tudo-comecou/>

Yuen, M. (2022, Abril, 15). *Chatbot market in 2022: Stats, trends, and companies in the growing*

*AI chatbot industry.* Analisado em Maio, 2022, através do site

<https://www.insiderintelligence.com/insights/chatbot-market-stats-trends>