



**UNIVERSIDADE DO MINHO**

Escola de Engenharia

Licenciatura em Engenharia Informática

## **Introdução à Inteligência Artificial**

Ano Letivo de 2025/2026

# **Sistema de Extração de Informação e Chatbot - ChefBot**

---

### **AUTORES**

|  |                |
|--|----------------|
| <b>Tomás Henrique Alves Melo</b>         | <b>PG60018</b> |
| <b>Rodrigo Miguel Granja Ferreira</b>    | <b>PG60392</b> |
| <b>Luís Pinto da Cunha</b>               | <b>PG60280</b> |
| <b>Nuno Filipe Leite Oliveira Araújo</b> | <b>PG61218</b> |

## Resumo

O presente relatório descreve o desenvolvimento do **ChefBot**, um chatbot conversacional inteligente para assistência culinária, desenvolvido no âmbito da unidade curricular de Introdução à Inteligência Artificial. O sistema permite aos utilizadores descobrir receitas com base em diversos critérios (categoria, tempo disponível, dificuldade, restrições alimentares), pesquisar por ingredientes disponíveis ou por nome específico de receita, e acompanhar a preparação passo a passo.

O ChefBot foi implementado utilizando o framework **RASA Open Source** para processamento de linguagem natural (NLP) em português, integrado com um dataset de aproximadamente **1597 receitas portuguesas** extraídas da plataforma **PetitChef**. O sistema demonstra capacidades avançadas de compreensão de intenções, extração de entidades e gestão de diálogo contextual.

# Índice

|   |           |
|---|-----------|
| <b>1. Introdução</b>                                  | <b>1</b>  |
| 1.1. Objetivos  | 1         |
| <b>2. Problema Escolhido e Motivação</b>              | <b>1</b>  |
| 2.1. Contextualização                                 | 1         |
| 2.2. Motivação  | 2         |
| 2.3. Desafios   | 2         |
| <b>3. Dataset e Preparação dos Dados</b>              | <b>2</b>  |
| 3.1. Fonte de Dados                                   | 2         |
| 3.2. Processo de Extração                             | 3         |
| 3.3. Processo de Limpeza & Identificação das Receitas | 3         |
| 3.4. Estatísticas do Conjunto de Dados                | 4         |
| 3.5. Pré-processamento                                | 5         |
| <b>4. Tecnologias Escolhidas e Justificação</b>       | <b>5</b>  |
| 4.1. Framework Principal: RASA Open Source            | 5         |
| 4.2. Alternativas Consideradas                        | 6         |
| 4.3. Pipeline de NLU                                  | 6         |
| 4.4. Políticas de Diálogo                             | 6         |
| <b>5. Arquitetura do Sistema</b>                      | <b>7</b>  |
| 5.1. Domínio (domain.yml)                             | 8         |
| 5.2. Regras (rules.yml)                               | 8         |
| 5.3. Stories (stories.yml)                            | 8         |
| 5.4. Ações Customizadas (actions.py)                  | 9         |
| 5.5. Sistema de Persistência                          | 9         |
| 5.5.1. favoritos.csv                                  | 9         |
| 5.5.2. recentes.csv                                   | 10        |
| 5.5.3. Sistema de Avaliação                           | 10        |
| 5.6. Interface Web                                    | 10        |
| 5.6.1. Tecnologias                                    | 10        |
| 5.6.2. Funcionalidades                                | 10        |
| 5.6.3. Comunicação com RASA                           | 11        |
| <b>6. Exemplos de Uso</b>                             | <b>11</b> |
| 6.1. Descoberta Guiada                                | 11        |
| 6.2. Pesquisa por Nome                                | 12        |
| 6.3. Pesquisa por Ingredientes                        | 12        |
| 6.4. Gestão de Favoritos                              | 13        |
| 6.5. Receitas Recentes                                | 13        |
| 6.6. Modo Passo-a-Passo                               | 13        |
| 6.7. Pesquisa Direta por Categoria                    | 14        |
| 6.8. Visualizar Receita Completa                      | 14        |
| <b>7. Reflexões Críticas</b>                          | <b>15</b> |
| 7.1. Implementação                                    | 15        |
| 7.2. Limitações identificadas                         | 15        |
| 7.3. Alternativa Considerada: RASA Pro com CALM       | 15        |
| 7.3.1. Comparação: RASA Clássico vs LLM               | 16        |
| 7.3.2. Desafios do NLP                                | 16        |
| <b>8. Conclusão</b>                                   | <b>16</b> |

|                             |           |
|-----------------------------|-----------|
| 8.1. Trabalho Futuro .....  | 16        |
| <b>9. Referências .....</b> | <b>17</b> |

# 1. Introdução

A culinária é uma atividade quotidiana que frequentemente envolve desafios de decisão: “O que posso cozinhar com os ingredientes que tenho?”, “Qual receita é adequada para o tempo disponível?”, “Que opções existem para restrições alimentares específicas?”. Estas questões representam um domínio ideal para a aplicação de técnicas de Processamento de Linguagem Natural (NLP) e sistemas de Question Answering (QA).

Este trabalho visa desenvolver um **agente de conhecimento conversacional** capaz de:

- Estabelecer diálogos naturais em português europeu
- Extrair informação relevante de uma base de conhecimento culinário
- Guiar utilizadores na descoberta e preparação de receitas
- Aprender preferências e manter histórico de utilização

## 1.1. Objetivos

Os objetivos específicos deste projeto incluem:

1. **Identificação do problema:** Definir um caso de uso relevante para extração de conhecimento
2. **Seleção de ferramentas:** Escolher e justificar tecnologias adequadas ao problema
3. **Implementação de pipeline:** Desenvolver um sistema de processamento e indexação de dados
4. **Desenvolvimento do chatbot:** Criar um sistema funcional de perguntas e respostas
5. **Avaliação:** Analisar o desempenho e limitações da solução
6. **Reflexão crítica:** Discutir desafios do NLP no contexto real

# 2. Problema Escolhido e Motivação

## 2.1. Contextualização

O domínio culinário apresenta características que o tornam particularmente interessante para sistemas de QA:

| Característica                  | Descrição   |
|---------------------------------|---|
| Vocabulário estruturado         | Ingredientes, técnicas e categorias bem definidas             |
| Informação semiestruturada      | Receitas com campos específicos (ingredientes, passos, tempo) |
| Interação natural               | Utilizadores formulam perguntas de forma conversacional       |
| Personalização                  | Necessidade de adaptar respostas a preferências individuais   |
| Múltiplos critérios de pesquisa | Categoria, tempo, dificuldade, restrições, ingredientes       |

Tabela 1: Caracterização do domínio culinário

## 2.2. Motivação

A escolha deste domínio foi motivada por:

1. **Relevância prática:** A culinária é uma atividade universal com necessidade real de assistência
2. **Complexidade adequada:** O problema requer processamento de múltiplas entidades e contextos
3. **Disponibilidade de dados:** Existem fontes ricas de receitas em português
4. **Variedade de interações:** Permite explorar diferentes padrões de diálogo (descoberta guiada, pesquisa direta, acompanhamento passo a passo)

## 2.3. Desafios

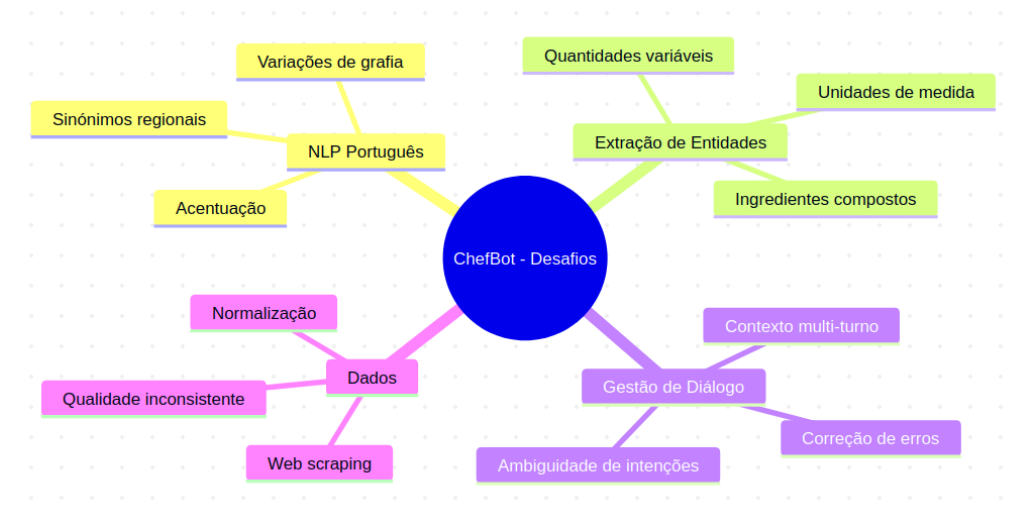


Figura 1: Desafios ChefBot

## 3. Dataset e Prepação dos Dados

### 3.1. Fonte de Dados

O dataset foi construído através de web scraping da plataforma PetitChef Portugal (<https://pt.petitchef.com/>), uma das maiores bases de dados de receitas em português. A escolha desta fonte deveu-se a:

- Receitas em português europeu
- Estrutura HTML consistente
- Grande variedade de categorias
- Informação nutricional incluída
- Ratings de utilizadores disponíveis

## 3.2. Processo de Extração

O script de extração (`extract_data.py`) utiliza as bibliotecas *requests* e *BeautifulSoup* para gerar um dataset (`petitchef_recipes.csv`) da seguinte forma:

```
# Categorias extraídas
CATEGORIAS = {
    "Entrada": "entrada",
    "Prato Principal": "prato-principal",
    "Sobremesa": "sobremesa",
}

# Critérios nutricionais identificados
CRITERIOS_POSSIVEIS = [
    "Sem glúten", "Vegan", "Vegetariano",
    "Sem lactose", "Sem açúcar", "Sem ovo",
]
```

**Campos extraídos por receita**

| Campo        | Descrição                | Exemplo   |
|--------------|--------------------------|---|
| titulo       | Nome da receita          | Creme de abóbora  |
| categoria    | Tipo de prato            | Entrada   |
| dificuldade  | Nível de complexidade    | Médio   |
| tempo_total  | Duração estimada         | 40 min  |
| calorias     | Valor nutricional        | 40 Kcal   |
| rating       | Avaliação média (0-5)    | 4.49  |
| porcoes      | Número de porções        | 8   |
| ingredientes | Lista de ingredientes    | 300g abóbora   2 batatas...   |
| passos       | Instruções de preparação | Cortar...   Cozinhar...   |
| criterios    | Restrições alimentares   | Sem glúten   Vegetariano  |
| imagem       | Imagem da receita        | <a href="https://pt.petitchef.com/imgupl/recipe/creme-de-abobora--160575p240773.webp">https://pt.petitchef.com/imgupl/recipe/creme-de-abobora--160575p240773.webp</a> |

Tabela 2: Campos extraídos por receita

## 3.3. Processo de Limpeza & Identificação das Receitas

Para além do script referido, foi criado também um programa para limpeza do dataset (`clean_csv.py`). Este lê o ficheiro de receitas gerado pelo script (`extract_data.py`) e gera um novo CSV limpo (`recipes_old.csv`).

Durante o processo, mantém a mesma estrutura de colunas, altera o separador de campos para ponto e vírgula e remove aspas simples e duplas dos campos de texto título e passos. Ainda, modifica o valor do campo 'dificuldade' de 'Muito Fácil' para 'Fácil'. O resultado é um ficheiro CSV mais simples e consistente.

Por fim, foi criado o script (add\_id.py) que lê o ficheiro (recipes\_old.csv) e cria um novo ficheiro (recipes.csv), adicionando uma coluna id no início.

O cabeçalho original é mantido, mas passa a ter o campo id como primeira coluna. Em seguida, cada linha de dados recebe um identificador numérico incremental, começando em 1, garantindo que cada registo tem um identificador único.

O resultado é o ficheiro final (recipes.csv) com a mesma informação do ficheiro anterior, mas com uma chave identificadora explícita para cada linha.

### 3.4. Estatísticas do Conjunto de Dados

O dataset final contém **1597 receitas** distribuídas da seguinte forma:

Pie Chart

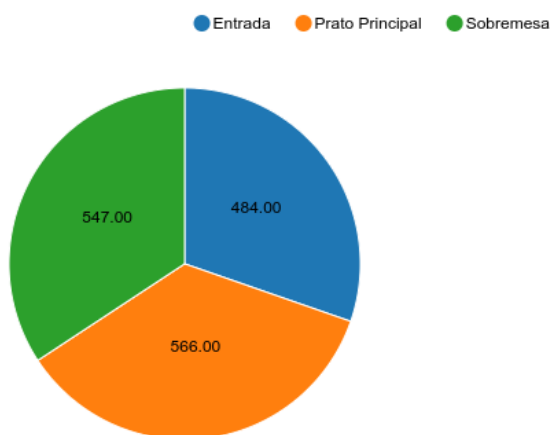


Figura 2: Distribuição por Categoria

Pie Chart

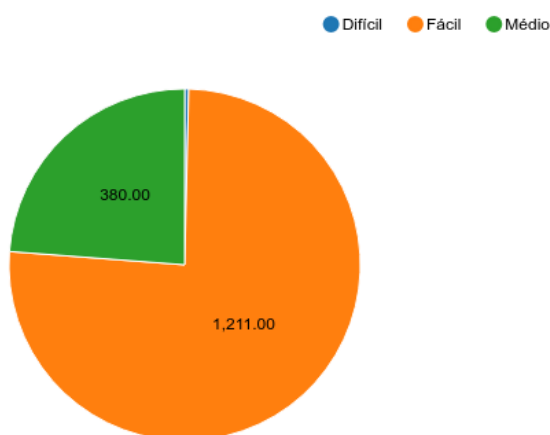


Figura 3: Distribuição por Dificuldade

#### Métricas de qualidade:

- Receitas com informação nutricional: 100%
- Receitas com rating: 100%
- Receitas com tempo total estimado: 100%
- Receitas com categoria: 100%



- Receitas com dificuldade: 100%
- Receitas com critérios alimentares: 89%
- Média de ingredientes por receita: 8-12
- Média de passos por receita: 6-10

A **qualidade dos dados** é garantida pelos critérios de extração de receitas que foram impostos no script de web scraping (`extract_data.py`).

### 3.5. Pré-processamento

O processamento dos dados incluiu:

1. **Tratamento de listas:** Separação por | de ingredientes e passos
2. **Filtragem de qualidade:** Exclusão de receitas incompletas (sem informação das calorias, rating, imagem e tempo estimado), com formatos não desejados (formato da lista de ingredientes divididos por categoria) ou com caracteres não desejados (títulos com '?' e ingredientes e passos com ';')
3. **Normalização:** Transformação da dificuldade do tipo 'Muito Fácil' para 'Fácil'

## 4. Tecnologias Escolhidas e Justificação

### 4.1. Framework Principal: RASA Open Source

A escolha para o desenvolvimento do chatbot recaiu sobre o RASA Open Source (versão 3.1). Esta decisão fundamenta-se, principalmente, na natureza de código aberto da plataforma, o que garante total soberania sobre os dados. Ao permitir o uso local e o funcionamento offline, o RASA assegura a privacidade das interações e elimina a dependência de serviços de terceiros na cloud, oferecendo um controlo que soluções proprietárias não permitem.

Do ponto de vista técnico, o RASA destaca-se pela sua arquitetura modular e extensível, permitindo uma customização profunda dos componentes de processamento de linguagem natural (NLP), o que é crucial para afinar o reconhecimento da língua portuguesa. A framework oferece suporte nativo robusto para a gestão de estado e preenchimento de slots, essencial para manter o contexto durante conversas longas. Adicionalmente, a possibilidade de escrever "Ações Customizadas" em Python confere flexibilidade para integrar lógica de negócio complexa. Por fim, o facto de ser uma solução gratuita, apoiada por uma comunidade ativa e documentação abrangente, valida a sua viabilidade económica e sustentabilidade a longo prazo.

## 4.2. Alternativas Consideradas

Antes de optar pelo RASA Open Source, foram avaliadas outras abordagens:

| Abordagem                 | Vantagens                                     | Razão da Exclusão                                |
|---------------------------|---|--|
| LLM Local (LLaMA/Mistral) | Respostas naturais, flexibilidade linguística | Latência elevada (2-4s), hardware insuficiente   |
| TF-IDF + Python           | Simples de implementar, rápido                | Sem gestão de diálogo, sem contexto entre turnos |
| Híbrido (RASA + LLM)      | Combina estrutura com flexibilidade           | Complexidade excessiva, dependência de APIs      |

Tabela 3: Alternativas consideradas

A escolha do **RASA Open Source** ofereceu o melhor equilíbrio entre funcionalidade, velocidade e independência de serviços externos.

## 4.3. Pipeline de NLU

O pipeline de processamento de linguagem natural foi configurado com os seguintes componentes:

```
pipeline:
- name: WhitespaceTokenizer           # Tokenização por espaços
- name: RegexFeaturizer              # Features baseadas em regex
- name: LexicalSyntacticFeaturizer   # Features lexicais e sintáticas
- name: CountVectorsFeaturizer       # Bag-of-words
- name: CountVectorsFeaturizer       # Character n-grams (1-4)
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier                # Classificador dual (intent + entity)
  epochs: 100
  constrain_similarities: true
- name: EntitySynonymMapper          # Mapeamento de sinónimos
- name: FallbackClassifier             # Gestão de baixa confiança
  threshold: 0.7
```

**Justificação dos componentes:**

1. **DIETClassifier**: Modelo *state-of-the-art* que combina classificação de intenções e extração de entidades numa única arquitetura transformer
2. **Character n-grams**: Essencial para português, capturando variações de escrita e tolerância a erros ortográficos
3. **FallbackClassifier**: Garante respostas *gracious* quando a confiança é baixa (threshold 0.7)

## 4.4. Políticas de Diálogo

```
policies:
- name: MemoizationPolicy            # Memorização de histórias
  max_history: 5
- name: RulePolicy                   # Regras determinísticas
  core_fallback_threshold: 0.4
- name: TEDPolicy                    # Transformer para diálogo
  epochs: 100
- name: UnexpectTEDIntentPolicy      # Detecção de intenções inesperadas
  max_history: 5
```

## 5. Arquitetura do Sistema

O ChefBot segue uma arquitetura modular e escalável, organizada em camadas distintas que separam responsabilidades e facilitam a manutenção e evolução do sistema. O sistema suporta uma **interface web moderna** desenvolvida em HTML + TailwindCSS.

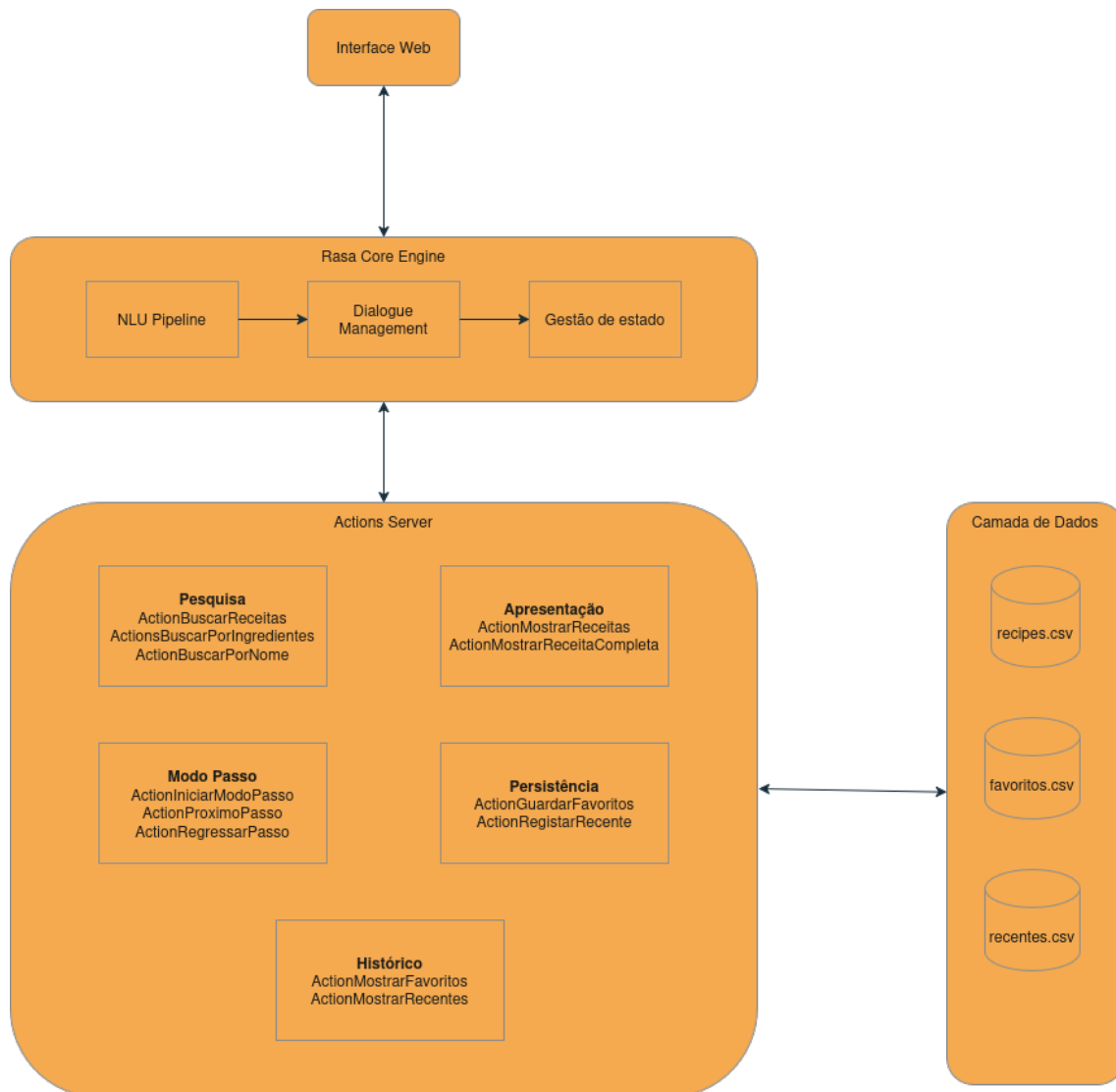


Figura 4: Arquitetura do Projeto

## 5.1. Domínio (domain.yml)

O domínio define todos os elementos que o chatbot “conhece”. A tabela seguinte resume os componentes:

| Componente | Quantidade | Exemplos  |
|------------|------------|---|
| Intenções  | 36         | greet, quero_cozinhar, escolher_categoria, proximo_passo... |
| Entidades  | 10         | categoria, tempo, dificuldade, ingrediente_possuido...      |
| Slots      | 14         | categoria, tempo, receitas_encontradas, passo_atual...      |
| Ações      | 28         | action_buscar_receitas, action_proximo_passo...             |
| Respostas  | 15         | utter_greet, utter_perguntar_categoria...                   |

Tabela 4: Componentes do Domínio

Os **slots** são variáveis de memória que mantêm o contexto da conversa. Por exemplo, quando o utilizador escolhe “prato principal”, o slot categoria é preenchido e mantido até ao fim da pesquisa.

## 5.2. Regras (rules.yml)

Definem comportamentos que devem acontecer **sempre** da mesma forma:

| Trigger           | Ação                         |
|-------------------|------------------------------|
| greet             | utter_greet                  |
| goodbye           | utter_goodbye                |
| proximo_passo     | action_proximo_passo         |
| finalizar_receita | action_perguntar_avaliacao   |
| favoritar_sim     | action_guardar_favoritos_csv |
| nlu_fallback      | utter_default                |

Tabela 5: Algumas regras implementadas

## 5.3. Stories (stories.yml)

As *stories* definem sequências de interação para treinar o modelo de diálogo. O sistema suporta três fluxos principais:

1. **Descoberta guiada:** O utilizador é guiado por perguntas sequenciais (categoria → tempo → dificuldade → restrições → calorias) até obter sugestões de receitas.
2. **Pesquisa por ingredientes:** O utilizador indica os ingredientes disponíveis e o sistema calcula um score de correspondência para sugerir receitas compatíveis.
3. **Modo passo-a-passo:** Após selecionar uma receita, o utilizador pode verificar novamente o conjunto de ingredientes que a receita possui, navegar pelos passos de preparação, com opções de avançar, recuar ou abandonar a receita.

## 5.4. Ações Customizadas (actions.py)

As ações são funções Python que executam lógica complexa impossível de definir apenas com templates YAML.

| Categoria    | Ações                   | Função  |
|--------------|-------------------------|---|
| Pesquisa     | buscar_receitas         | Filtra por categoria, tempo, dificuldade e calorias   |
| Pesquisa     | buscar_por_ingredientes | Calcula score de match com ingredientes do utilizador |
| Pesquisa     | buscar_por_nome         | Pesquisa fuzzy por nome de receita                    |
| Modo Passo   | proximo_passo           | Avança no contador e mostra próxima instrução         |
| Persistência | guardar_favoritos_csv   | Escreve receita no ficheiro CSV de favoritos          |

Tabela 6: Principais ações customizadas

## 5.5. Sistema de Persistência

O ChefBot mantém dados do utilizador através de dois ficheiros CSV:

### 5.5.1. favoritos.csv

Guarda as receitas marcadas como favoritas pelo utilizador.

| Campo          | Descrição                        |
|----------------|----------------------------------|
| data_favorito  | Timestamp de quando foi guardado |
| id             | Número identificador da receita  |
| titulo         | Nome da receita                  |
| categoria      | Tipo de prato                    |
| dificuldade    | Nível de complexidade            |
| tempo_total    | Duração estimada                 |
| tempo_minutos  | Duração estimada em minutos      |
| calorias       | Valor nutricional                |
| rating_dataset | Avaliação média (0-5)            |
| critérios      | Restrições alimentares           |
| ingredientes   | Lista de ingredientes            |
| passos         | Instruções de preparação         |

Tabela 7: Estrutura do favoritos.csv

### 5.5.2. recentes.csv

Regista as receitas finalizadas pelo utilizador, incluindo a **avaliação pessoal**.

| Campo                | Descrição                                |
|----------------------|--|
| data_finalizacao     | Timestamp de quando terminou a receita   |
| id                   | Número identificador da receita          |
| titulo               | Nome da receita                          |
| categoria            | Tipo de prato                            |
| dificuldade          | Nível de complexidade                    |
| tempo_total          | Duração estimada                         |
| tempo_minutos        | Duração estimada em minutos              |
| calorias             | Valor nutricional                        |
| rating_dataset       | Avaliação média (0-5)                    |
| criterios            | Restrições alimentares                   |
| ingredientes         | Lista de ingredientes                    |
| passos               | Instruções de preparação                 |
| avaliacao_utilizador | Nota dada pelo utilizador (1-5 estrelas) |

Tabela 8: Estrutura do recentes.csv

### 5.5.3. Sistema de Avaliação

Após finalizar uma receita no modo passo-a-passo, o utilizador é convidado a avaliar a experiência de 1 a 5 estrelas. Esta avaliação é guardada no `recentes.csv`.

O sistema também identifica:

- **Receita mais realizada:** a que aparece mais vezes no histórico
- **Última receita:** a mais recentemente finalizada

## 5.6. Interface Web

### 5.6.1. Tecnologias

- **HTML5** — estrutura da página
- **TailwindCSS** — estilização moderna via CDN
- **JavaScript** — comunicação com RASA (fetch API)
- **localStorage** — persistência do histórico

### 5.6.2. Funcionalidades

- **Chat em tempo real** com o servidor RASA
- **Histórico de conversas** guardado localmente (até 50 conversas)
- **Botões interativos** renderizados automaticamente
- **Quick Actions:** atalhos para “Quero cozinhar algo”, “Favoritos”, “Recentes”

- **Sidebar retrátil** para gestão de conversas anteriores
- **Design responsivo** adaptável a desktop e mobile

### 5.6.3. Comunicação com RASA

A interface comunica com o endpoint `http://localhost:5005/webhooks/rest/webhook` enviando mensagens em JSON e recebendo respostas com texto e botões.

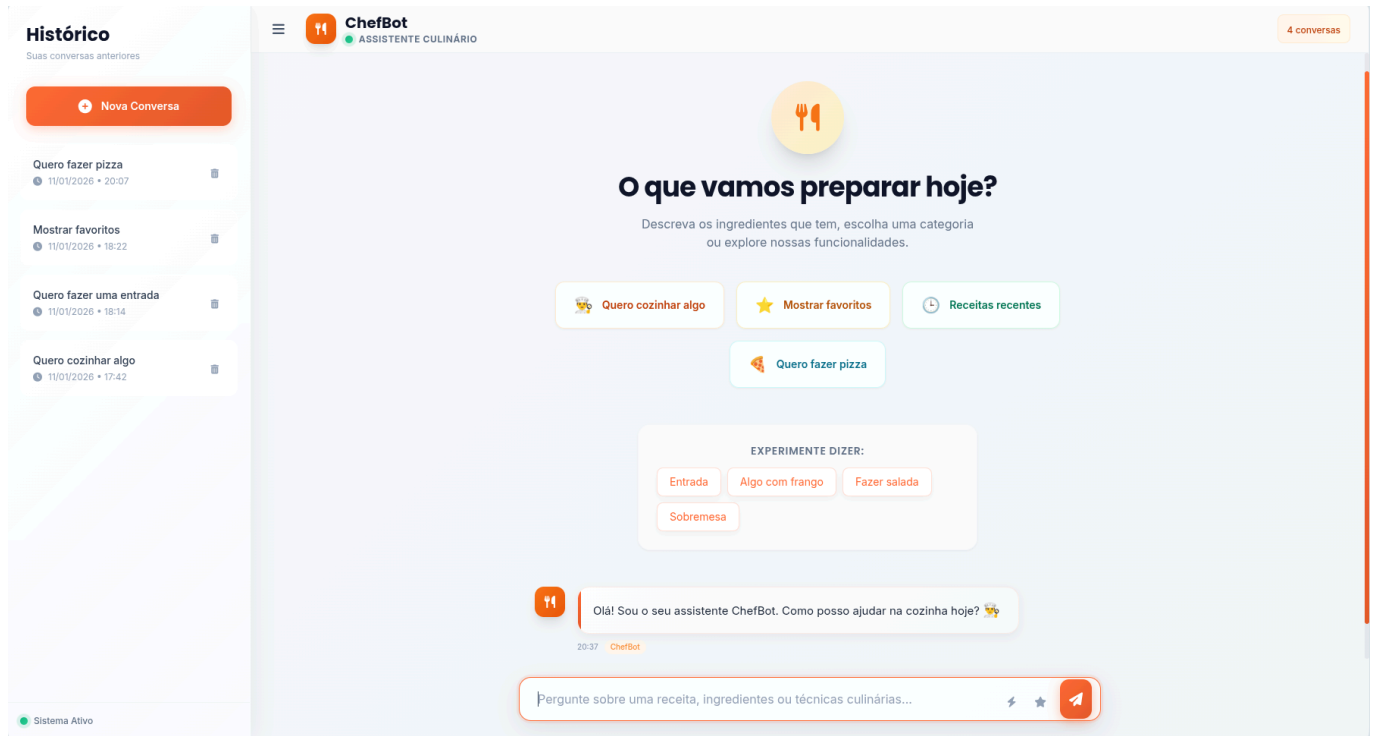



Figura 5: Interface Web do ChefBot

## 6. Exemplos de Uso


### 6.1. Descoberta Guiada

- Utilizador:** Quero cozinhar algo
- ChefBot:** Ótimo! Que tipo de prato queres fazer?  
[🥗 Entrada] [🍲 Prato principal] [🍰 Sobremesa]
- Utilizador:** Prato principal
- ChefBot:** Quanto tempo tens disponível?  
[⏰ Até 30min] [🕒 30-60min] [🕒 Mais de 1h] [👤 Tanto faz]
- Utilizador:** Tanto faz
- ChefBot:** Qual o nível de dificuldade que procuras?  
[😊 Fácil] [😐 Médio] [😞 Difícil] [🎯 Qualquer]

 **Utilizador:** Qualquer


 **ChefBot:** Tens alguma restrição alimentar?


[🥬 Vegetariano] [🌱 Vegano] [🚫 Sem glúten] [🚫 Sem lactose]  
[🚫 Sem açúcar] [🥚 Sem ovo] [✅ Nenhuma]


 **Utilizador:** Sem Açúcar

 **ChefBot:** Queres uma opção mais leve ou não te importas com as calorias?

[🥗 Leve (até 300 cal)] [🍽️ Moderado (300-600)] [🍔 Reforçado (600-900)]  
[🔥 Hipercalórico (+900)] [🍷 Sem restrição]


 **Utilizador:** Leve


 **ChefBot:** A procurar as melhores receitas para ti... 🔍

 **ChefBot:** Encontrei 5 receitas! 🔍

1. 😊 **Rolos de berinjela à parmegiana** — ⌚ 1h | 🔥 223 Kcal
2. 😊 **Quiche de tomate e queijo curado** — ⌚ 40 min | 🔥 235 Kcal
3. 😊 **Tacos chamuças de carne picada** — ⌚ 35 min | 🔥 94 Kcal
4. 😊 **Hambúrguer vegetariano de abobrinha** — ⌚ 45 min | 🔥 278 Kcal
5. 😊 **Flan de legumes** — ⌚ 40 min | 🔥 218 Kcal


## 6.2. Pesquisa por Nome

 **Utilizador:** Quero fazer pizza

 **ChefBot:** Encontrei 8 receitas! 🔍

1. 😊 **Pizza kitkat** — ⌚ 42 min | 🔥 167 Kcal
2. 😊 **Pizza em trouxas** — ⌚ 30 min | 🔥 469 Kcal
3. 😊 **Pizza de melancia** — ⌚ 20 min | 🔥 41 Kcal
4. 😊 **Pizza doce nutella e banana** — ⌚ 25 min | 🔥 132 Kcal
5. 😊 **Waffle pizza** — ⌚ 34 min | 🔥 2608 Kcal

## 6.3. Pesquisa por Ingredientes


 **Utilizador:** Tenho batata e ovos, o que posso fazer?


 **ChefBot:** Encontrei 5 receitas com esses ingredientes! 🔍

1. 😊 **Tortilha espanhola** — ⌚ 35 min | 🔥 245 Kcal
2. 😊 **Ninho de páscoa** — ⌚ 42 min | 🔥 100 Kcal




## 6.4. Gestão de Favoritos


 **Utilizador:** Mostra os meus favoritos

 **ChefBot:** Tens 5 receitas guardadas nos favoritos:


1. 😊 **Sopa fria de ervilhas** — ⌚ 20 min | 🔥 965 Kcal
2. 😊 **Waffles super proteicos com whey** — ⌚ 8 min | 🔥 125 Kcal
3. 😊 **Salada de pepino com molho de iogurte** — ⌚ 5 min | 🔥 52 Kcal
4. 😊 **Salada de damasco assado e burrata** — ⌚ 20 min | 🔥 976 Kcal
5. 😊 **Croquete de peixe com limão** — ⌚ 20 min | 🔥 92 Kcal

## 6.5. Receitas Recentes


 **Utilizador:** Mostra as receitas recentes


 **ChefBot:** Tens 6 receitas feitas recentemente:

★ **Mais feita:** Tigela de skyr com frutas vermelhas


 **Última:** Sopa fria de ervilhas

## 6.6. Modo Passo-a-Passo

 **Utilizador:** Começar modo passo-a-passo

 **ChefBot:** Tens todos os ingredientes?


[✅ Sim, vamos lá] [🛒 Mostrar ingredientes primeiro]

 **Utilizador:** Sim, vamos lá

 **ChefBot:** **PASSO 1 de 5**

Pré-aqueça o forno a 180°C.

[← Próximo passo] [🔴 Abandonar receita]

 **Utilizador:** Próximo


 **ChefBot:** **PASSO 2 de 5**

Abra a massa folhada e espalhe o Nutella.

[← Próximo passo] [↩ Regressar passo] [🔴 Abandonar receita]

... continua até ao último passo ...


 **Utilizador:** Finalizar

 **ChefBot:** 🎉 Parabéns! Terminaste a receita! Como correu?

[1 estrela ★] [2 estrelas ★★] [3 estrelas ★★★]


[4 estrelas ★★★★] [5 estrelas ★★★★★]


 **Utilizador:** 5 estrelas

 **ChefBot:** Guardar nos favoritos?

[✅ Sim, guardar] [❌ Não guardar]

## 6.7. Pesquisa Direta por Categoria

 **Utilizador:** Quero fazer um prato principal

 **ChefBot:** Encontrei 3 receitas de prato principal! 🔍

1. 😊 **Arroz de pato** — ⌚ 1h 30 min | 🔥 450 Kcal
2. 😊 **Bacalhau à Brás** — ⌚ 45 min | 🔥 380 Kcal
3. 😊 **Frango assado** — ⌚ 1h | 🔥 320 Kcal

## 6.8. Visualizar Receita Completa

 **Utilizador:** Ver receita 2


 **ChefBot:**



### **WAFFLES SUPER PROTEICOS COM WHEY**

⌚ 8 min | 🇧🇷 Fácil | 🔥 125 Kcal

★ Rating: 5.0/5

 Porções: 6

✅ Info: Vegetariano, Sem açúcar

#### **Ingredientes:**

- 100g farinha de trigo
- 2 ovos
- 1 col. chá de fermento
- 180ml de leite
- 35g de whey baunilha

#### **Passos:**

1. Coloque todos os ingredientes no mixer.
2. Misture até ficar homogêneo.
3. Unte a forma de waffle.
4. Despeje a massa na forma de waffle. Cozinhe os waffles por cerca de 3 minutos (dependendo da potência da sua máquina de waffles).
5. Retire os waffles quando estiverem dourados.
6. Está pronto. Bom apetite!

## 7. Reflexões Críticas

### 7.1. Implementação

- Fluxo de descoberta guiada intuitivo com botões
- Modo passo-a-passo prático para cozinhar
- Pesquisa por ingredientes com sistema de scoring
- Pesquisa por nome de receitas

### 7.2. Limitações identificadas

- **Vocabulário limitado:** Depende dos exemplos de treino
- **Pesquisa aproximada:** “tomate” pode não encontrar “tomates cherry”
- **Sem compreensão semântica:** Não entende equivalências (ex: “presunto” ≠ “fiambre”)

### 7.3. Alternativa Considerada: RASA Pro com CALM

Durante a fase inicial de arquitetura do sistema, foi explorada a utilização do **RASA Pro** com a tecnologia **CALM (Conversational AI with Language Models)**. Esta abordagem representa uma mudança de paradigma, delegando a gestão do fluxo de conversação e a compreensão do contexto para Grandes Modelos de Linguagem (LLMs), em vez de depender estritamente de classificadores de intenções e histórias pré-definidas.

Para o motor de raciocínio (**reasoning engine**), foi realizada uma integração com o modelo **Google Gemini**. No entanto, esta implementação enfrentou obstáculos significativos:

1. **Limitações de API:** A dependência da API da Google resultou em bloqueios frequentes devido a **rate limits** (quotas de utilização) e instabilidade decorrente de atualizações na plataforma, o que comprometeu o fluxo de desenvolvimento contínuo.
2. **Latência em Modelos Locais:** Como tentativa de mitigação para reduzir a dependência de APIs externas, procurou-se substituir o Gemini por um **LLM local**. Contudo, esta solução revelou-se inviável em termos de **performance** no hardware disponível. O tempo de inferência elevado resultava numa latência inaceitável para uma interação em tempo real, degradando a experiência do utilizador.

Face a estes desafios técnicos, a imprevisibilidade de serviços externos e o custo computacional proibitivo de soluções locais, optou-se por regressar à arquitetura clássica do **RASA Open Source**. Esta decisão garantiu um sistema mais robusto, rápido e previsível, onde o controlo do diálogo é assegurado por políticas explícitas e um pipeline de NLU otimizado, sem a latência associada à geração de texto por LLMs.

### 7.3.1. Comparação: RASA Clássico vs LLM

| Aspeto          | RASA Open Source          | RASA Pro + LLM              |
|-----------------|---------------------------|-----------------------------|
| Latência        | 50-100ms                  | 2s-4s                       |
| Offline         | Sim                       | Não (API)                   |
| Custo           | Gratuito                  | Pago (tokens)               |
| Controlo        | Total (regras explícitas) | Limitado (LLM decide)       |
| Previsibilidade | Alta                      | Baixa (respostas variáveis) |
| Flexibilidade   | Média (treino manual)     | Alta (compreende variações) |

Tabela 9: Comparação entre abordagens

A solução atual privilegia **robustez, velocidade e previsibilidade** sobre a flexibilidade linguística que um LLM ofereceria. Para um domínio estruturado como receitas, onde as interações são relativamente previsíveis, esta troca revelou-se vantajosa.

### 7.3.2. Desafios do NLP

- Variabilidade linguística do português
- Ambiguidade contextual (“leve” pode ser calorias ou dificuldade)
- Dados do web scraping com inconsistências que necessitaram de maior atenção

## 8. Conclusão

Este trabalho demonstrou a viabilidade de desenvolver um assistente culinário conversacional utilizando o RASA Open Source. O ChefBot consegue:

- Estabelecer diálogos naturais em português
- Guiar utilizadores na descoberta de receitas através de múltiplos critérios
- Pesquisar receitas por ingredientes ou nome
- Acompanhar a preparação passo-a-passo
- Manter histórico de favoritos, receitas feitas recentemente e conversas

### 8.1. Trabalho Futuro

1. **Integração com LLM:** Usar modelos locais (LLaMA/Mistral) para respostas mais naturais
2. **Sistema de recomendação:** Sugerir receitas com base no histórico do utilizador
3. **Substituições inteligentes:** Sugerir alternativas para ingredientes em falta
4. **Planeamento semanal:** Geração automática de menus

## 9. Referências

1. RASA Open Source Documentation. (2024). *Building Contextual AI Assistants*. <https://rasa.com/docs/>
2. PetitChef Portugal. (2024). *Receitas de Culinária*. <https://pt.petitchef.com/>
3. Bunk, T., Varber, D., & Morise, H. (2020). *DIET: Lightweight Language Understanding for Dialogue Systems*. arXiv preprint.
4. Como Utilizar Beautiful Soup para Web Scraping. (2024). Asimov Academy. <https://hub.asimov.academy/tutorial/como-utilizar-beautiful-soup-para-web-scraping/>