



Assistente para o envelhecimento ativo em casa
Universidade de Aveiro
Projeto em Informática
Maio 2023

Bernardo Leandro - 98652
Carlos Sena - 81377
Diogo Silva - 98644
Luís Martins - 98521
Manuel Diaz - 103645
Pedro Coelho - 104247

Orientadores:

Prof. António Teixeira
Ana Patrícia Rocha
Nuno Almeida

Resumo

Com o envelhecimento da população portuguesa a aumentar e a tornar-se um problema preocupante, é necessário pensar em formas de ajudar a população idosa a manter a sua independência em casa. Com isto em mente, foi criado o projeto "Casa Viva+", onde o nosso projeto se insere, e que tem como objetivo a manutenção de uma vida digna em casa. Para atingir este objetivo, propõe-se o desenvolvimento de uma casa capaz de auxiliar nas tarefas domésticas, monitorizar a saúde de quem a habita e ajudar na prevenção e reabilitação.

O projeto sobre o qual este relatório incide consiste na criação de um assistente controlado por voz que irá responder aos pedidos dos habitantes da "Casa Viva+". Diferencia-se de outros assistentes por ser mais fácil de utilizar por idosos, tendo sido pensado especificamente para tal, ao contrário, por exemplo, da Siri ou Alexa, que foram desenvolvidos para ser utilizados por grandes massas e com os quais a população idosa demonstra alguma dificuldade de utilização.

Através da interação por voz com o assistente, será possível ao habitante fazer a gestão da sua agenda, informar-se sobre a meteorologia, controlar os dispositivos da referida casa, realizar chamadas, obter sugestão de receitas e recomendação de produtos.

Ao terminarmos a implementação da primeira versão do assistente realizámos alguns testes com possíveis utilizadores reais para, com base nas experiências reais, conseguirmos melhorar o assistente e tornar as interações o mais fluente possível.

Índice

1	<i>Introdução</i>	7
1.1	Motivação e Problema	7
1.2	Objetivo	8
2	<i>Estado da Arte</i>	9
2.1	Trabalho Relacionado	9
2.2	Tecnologias de Desenvolvimento de Sistemas de Diálogo	10
3	<i>Cenários, Requisitos e Arquitetura</i>	12
3.1	Persona	12
3.2	Cenários	13
3.3	Requisitos	14
3.4	Arquitetura	16
4	<i>Ferramentas</i>	18
4.1	Rasa	18
4.1.1	Arquitetura do Rasa	18
4.1.2	Estrutura do Rasa	19
4.1.3	Conceitos do Rasa	19
4.2	Google Calendar API	24
4.3	Google Cloud	24
4.4	OpenWeather API	24
4.5	Outras Ferramentas	25
5	<i>Desenvolvimento</i>	26
5.1	Metodologia de desenvolvimento	26
5.1.1	Fase 1: Descrição de Cenários	26
5.1.2	Fase 2: Extração dos Casos de Uso	26
5.1.3	Fase 3: Listagem de frases	26
5.1.4	Fase 4: Definição dos “Intents”	27
5.1.5	Fase 5: Identificação das <i>Entities</i> e <i>Slots</i>	27
5.1.6	Fase 6: Desenvolver as <i>Actions</i> e Ligação com o Módulo Externo	28
5.2	Mudanças ao Plano Inicial	28
6	<i>Testes e Resultados</i>	29
6.1	Testes	29
6.2	Resultados	30
6.3	Limitações e Dificuldades	31
7	<i>Conclusão</i>	33
7.1	Trabalho Futuro	33
7.2	Agradecimentos	34

8	<i>Referências.....</i>	35
9	<i>Anexos</i>	38



Siglas

CUI – Conversational User Interface

NLP – Natural Language Processing

NLU – Natural Language Understanding

API – Application Programming Interface

RNF – Requisito Não Funcional

RF – Requisito Funcional

SIP – Session Initiation Protocol

Índice de imagens

Figura 1: Percentagem da população portuguesa	7
Figura 2: Arquitetura geral do assistente	16
Figura 3: Arquitetura mais detalhada, incluindo os vários módulos do assistente	17
Figura 4: Arquitetura do Rasa, retirada de [2]	18
Figura 5: Estrutura de um projeto Rasa	19
Figura 6: Definição do “intent” check_balance	20
Figura 7: Definição do “slot” destination portuguesa	20
Figura 8: Definição de uma “story” e de uma “rule”	21
Figura 9: Definição de “responses (utter_greet e utter_bye)	21
Figura 10: Exemplo de uma “custom action”	22
Figura 11: Exemplo de um “form”	22
Figura 12: Exemplo de uma “Default Action”	23
Figura 13: Exemplo de uma “action” de validação de “slots”	23
Figura 14: Intent usado para saber que eventos o utilizador tem marcados	27
Figura 15: “Slot” day	28
Figura 16: Tabela com resultado dos testes realizados	30

1 Introdução

1.1 Motivação e Problema

Nos últimos anos tem-se assistido ao crescimento dos níveis de envelhecimento da população portuguesa. Segundo o Instituto Nacional de Estatística, a população idosa representa neste momento 23.4% da população portuguesa [1], um aumento de 20,6% nos últimos 10 anos.

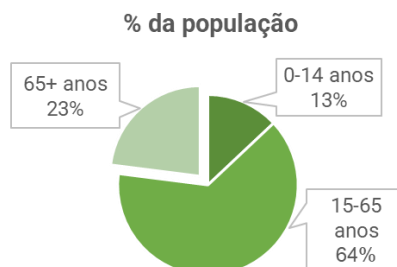


Figura 1 Percentagem da população portuguesa

Este aumento da população idosa tem associado alguns problemas sociais como a solidão e isolamento, especialmente em situações de acompanhamento familiar limitado. Há ainda outras dificuldades naturais da idade, como a redução da mobilidade ou da visão, que colocam desafios a atividades do quotidiano como recordar os números de telefone de familiares, deslocar-se ao interruptor ou lembrar de tomar a medicação, obrigando muitos idosos a ter de deixar as suas casas para poderem ter o acompanhamento necessário ao seu bem-estar, situação que nem sempre é do seu agrado.

Urge assim a necessidade de combater esta realidade desenvolvendo soluções que permitam a estas pessoas manter a sua independência e bem-estar em casa. Foi com esta realidade em mente que surgiu a iniciativa CasaViva+, na qual o nosso projeto se insere.

A CasaViva+ apresenta-se como uma casa inteligente, que tem como objetivo principal a manutenção de uma vida digna em casa, propondo-se a fazê-lo através do desenvolvimento de uma casa capaz de auxiliar nas tarefas domésticas, monitorizar a saúde de quem a habita e ajudar na prevenção e reabilitação.

O desenvolvimento deste projeto tentou dar a resposta a um problema: “De que forma o habitante da casa poderá interagir com esta de forma a usufruir das suas funcionalidades?”, o que levou à necessidade de definição de um método de interação fácil entre o habitante e a casa.



1.2 Objetivo

É neste sentido que surge o nosso projeto, um assistente conversacional que permite ao habitante da CasaViva+ controlar, de forma fácil, as diversas funcionalidades da CasaViva+, como ligar/desligar a televisão e controlar as luzes, assim como saber o que se passa na casa ou controlar gastos energéticos.

O assistente também deverá facilitar a realização de outro tipo de atividades como cozinhar, realizar chamadas telefónicas ou lembrar de algo que poderá estar a esquecer-se, agendando eventos, tudo com o objetivo final de aumentar o bem-estar do habitante e facilitar a sua utilização da casa.

2 Estado da Arte

2.1 Trabalho Relacionado

Como base para o início deste projeto, foram-nos dados alguns trabalhos, pesquisas ou relatórios que nos ajudaram a melhor entender o contexto em que o mesmo se inseria.

Inicialmente, foi-nos disponibilizada a dissertação de Mestrado do Tiago Almeida [2]. Desenvolvida em 2022, este trabalho consiste também no desenvolvimento de um assistente pelo que aborda todos os conceitos que nós precisávamos de entender e utilizar para o desenvolvimento do nosso próprio assistente.

Foram-nos também disponibilizados mais dois documentos, o relatório *"CUED Standard Dialogue Acts"* de Steve Young [3] e um capítulo do livro *"Chatbots & Dialogue Systems"* de Daniel Jurafsky e James H. Martin onde é explicada de forma mais detalhada como se deve formar um diálogo entre assistentes e humanos e quais são os focos dos diálogos que existem neste tipo de interação.

Numa segunda etapa e para melhor entender de que forma poderíamos desenvolver uma solução que além de ajudar os mais idosos, fosse fácil de utilizar por estes, procurámos e analisámos vários estudos, investigações e artigos com o objetivo de recolher o máximo de informação útil sobre o assunto. Para tal recorremos ao *Google Scholar* através da pesquisa por algumas palavras-chave como: *"Home assistant active ageing"*, *"Active ageing at home"*, *"Assistant for older people"*, *"Home assistant for older people"*, *"Home voice assistant older people"*, *"Voice assistant older people"*, etc.

Como é referido no artigo [4], os assistentes de voz atuais, como Amazon Echo ou Apple Siri, não são desenhados para serem utilizados especificamente por pessoas idosas. O artigo refere que a comunidade Conversational User Interface (CUI) terá de ter especial atenção a barreiras que podem ser criadas entre os mais idosos e os assistentes de voz, pelo facto de os segundos não saberem como comunicar com os primeiros. Outro problema discutido é o modelo *"one-size-fits-all"* destes assistentes, onde a forma como interagem com alguém não é especializada a uma faixa etária.

No estudo [5], conseguimos perceber em detalhe os maiores problemas que os assistentes de voz atuais possuem ao serem utilizados por pessoas mais velhas. De entre os quais se destacam a falha de receção da voz por parte do dispositivo ou a má interpretação do que foi dito ou pedido, o esquecimento de palavras-chave ou o não conhecimento de certos comandos e a complexidade de pedidos feitos. Todos estes causaram frustração em alguns dos participantes. No entanto, todos concordaram que o assistente de voz era mais fácil de usar do que o computador ou o smartphone.

Isto é, ainda, corroborado pelos estudos [6] e [7]. Ambos de duração mais longa que o anterior, comprovam que os participantes que não desistiram de utilizar o assistente de voz conseguiram aprender a usá-lo e a contornar alguns dos obstáculos apresentados em cima.

Para além disto, o estudo [8] tenta perceber como os mais idosos veem estes assistentes - se são apenas uma caixa de onde podem extrair informação ou se podem ser uma companhia social. Deste estudo, retira-se que os participantes se dirigiam ao assistente como se de uma

pessoa se tratasse. Por esse motivo, será um objetivo no futuro dar a estes assistentes comportamentos humanos, como, por exemplo, desejar ao utilizador um “bom dia” ou perguntar-lhe se “está tudo bem”.

Assim, podemos concluir que os atuais assistentes por voz não estão adaptados a pessoas idosas. É necessário incluir uma linguagem mais abrangente, que não se foque em palavras-chave ou que aceite múltiplas palavras-chave para a mesma ação, sendo, também, essencial que um assistente seja capaz de decifrar pedidos mais complexos, que possam conter dois ou mais comandos. É, ainda, aconselhado o fornecimento de comportamentos humanos a estes assistentes.

2.2 Tecnologias de Desenvolvimento de Sistemas de Diálogo

Existem várias tecnologias de desenvolvimento de sistemas de diálogo a considerar quando começamos um projeto deste tipo. Dois fatores importantes são o facto da tecnologia ser “open-source”, ou seja, o design da mesma é de conhecimento público, sendo assim passível de modificações e partilha entre várias pessoas e a capacidade de Natural Language Processing (NLP), isto é, de extrair informação do que é escrito ou dito. Assim, analisámos algumas opções, que são apresentadas abaixo.

Microsoft Bot Framework

A Microsoft [9] oferece esta plataforma “open-source” que permite a integração de outros serviços da mesma e conta com uma grande comunidade de apoio ao desenvolvedor. O maior problema é o facto do seu Natural Language Understanding (NLU), intitulado de LUIS, ser limitado a um certo número de chamadas feitas à Application Programming Interface (API), sendo a partir daí necessário pagar pelo serviço e o mesmo coloca alguns entraves à customização do assistente. Outro problema apresentado pela plataforma é a falta de NLP.

Amazon Lex

A Amazon [10] apresenta este serviço para construção de interfaces conversacionais, a mesma que é usada pelo seu assistente virtual, Alexa. Um aspeto positivo deste serviço é a facilidade de integração com outros serviços. Ao contrário da tecnologia anterior, esta não é “open-source” e, de forma semelhante, tem um NLU limitado, apresenta um grande grau de dificuldade para customizar o assistente e não contém NLP integrado.









DialogFlow

A Google [11] fornece esta plataforma de NLU usada para integrar uma interface conversacional numa aplicação móvel. Esta contém suporte de voz, agentes pré-construídos e a capacidade de integrar outros serviços da Google. No entanto, a mesma não é “open-source”, apresentando um elevado grau de complexidade, dificuldade em customizar o agente e não disponibiliza NLP.

Rasa

O Rasa [12] é uma ferramenta “open-source” com foco na criação de histórias para a criação de chatbots. Esta ferramenta é flexível e possui uma boa documentação, bem como a capacidade de NLP. No entanto, pode tornar-se algo complexa e tem uma curva de aprendizagem muito íngreme. Esta ferramenta será descrita com maior pormenor mais à frente, uma vez que foi a ferramenta escolhida para o desenvolvimento deste projeto.

Na tabela a seguir apresentada está um resumo das características mais relevantes de cada tecnologia acima referida:

Tecnologia	Vantagens	Desvantagens	Open-Source	Natural Language Processing (NLP)
Rasa	Flexibilidade; Boa documentação;	Curva de aprendizagem íngreme;		
Microsoft Bot Framework	Bom apoio da comunidade; Integração com outros serviços da Microsoft;	NLU e customização limitados;		
Amazon Lex	Integração com serviços Amazon;	Pago; NLU e customização limitados;		
Dialogflow	Integração com serviços Google Suporte de voz Agentes pré-desenvolvidos;	Pago; Complexo; Customização limitada;		

3 Cenários, Requisitos e Arquitetura

Este capítulo tem como finalidade a apresentação dos requisitos e arquitetura da nossa solução. Para tal, começámos por analisar o projeto CasaViva+, de onde retirámos a Persona, e para a qual, após a recolha de alguma informação *online* e de um *brainstorm* de grupo, definimos os cenários de utilização. A partir destes definimos os requisitos da nossa solução e a sua arquitetura.

3.1 Persona

Uma vez que o nosso projeto se insere no projeto CasaViva+, a Persona será necessariamente a mesma.

Deste modo, apresentamos a Sra. Maria que, com 74 anos e a viver sozinha, pretende conseguir controlar com facilidade os recursos da sua casa de modo a continuar a ter uma vida o mais independente possível.



A Sra. Maria nasceu no dia 11 de Janeiro de 1949 e vive em Aveiro, Portugal. Sra. Maria é viúva e vive sozinha numa casa, tendo dois filhos, o João de 45 anos e o António de 50 anos. Ao fim do dia, um cuidador vai à sua casa tratar das refeições do dia seguinte e da sua higiene. Sendo uma pessoa idosa, Maria sofre de algumas incapacidades a nível de visão, função motora, audição, memória e requer de alguma monitorização de saúde, tendo hipertensão e diabetes. Vivendo sozinha e com as suas incapacidades, para a Sra. Maria torna-se um desafio conseguir movimentar-se e fazer as suas tarefas sozinha sem correr algum tipo de risco.

Perfil geral

Idade: 73

Família: Viúva, 2 filhos

Habitação: Habita sozinha em Aveiro

Saúde: Disfunção motora, problemas de visão e audição, hipertensão e diabetes, falta de memória e alterações de humor (requer medicação)

Objetivos: A Sra. Maria deseja conseguir controlar os recursos da sua casa, como a televisão, bem como contactar pessoas importantes, podendo viver o seu dia-a-dia de forma tranquila.

3.2 Cenários

Tendo em consideração a Persona definida e os objetivos do projeto definimos os cenários que são apresentados de forma geral na tabela seguinte:

#	Designação	Cenário
1	Chamada a um familiar	A Sra. Maria pretende ligar para um familiar e tem dificuldade em lidar com as teclas do seu telemóvel. Esta comunica ao assistente, através da voz, que deseja ligar ao seu neto e este responde-lhe, por via das colunas, que a chamada irá começar dentro de momentos. Assim, o telemóvel inicia a chamada automaticamente, aparecendo no ecrã que está a ligar para o neto.
2	Controlo dos dispositivos (TV, luzes, aquecedores)	Sentada no sofá e com algumas dificuldades de movimentação, a Sra. Maria pede ao assistente, através da voz, para ligar a TV, no canal da SIC. O assistente liga a TV, no canal pretendido, e a habitante interage, novamente, pedindo-lhe que aumente ligeiramente o volume. O mesmo faz o que é pedido e questiona a mesma sobre as luzes ligadas noutras divisões e sobre o aquecimento no quarto.
3	Receitas e valor nutricional de produtos	A Sra. Maria vai cozinhar, mas não se lembra da receita, e devido aos problemas de saúde quer garantir que todos os produtos que consome são de qualidade. Através da voz, ela consegue perguntar ao assistente a receita do bolo de mármore de que se esqueceu e o assistente, por sua vez, responde com a informação acerca dos produtos que tem de utilizar. Depois de reunir os produtos esta questiona o assistente acerca da qualidade dos mesmos e após a leitura do código de barras o assistente informa-a sobre a sua qualidade nutricional. Assim, a habitante consegue fazer o bolo pretendido.
4	Alarme da medicação	A Sra. Maria, devido a problemas de saúde, tem de tomar medicação todos os dias às 21 horas. Com acesso ao assistente, Maria informa-o que tem de tomar a sua medicação todos os dias a essa hora. Deste modo, às 21:00, o assistente aciona um alarme, onde informa a necessidade do consumo da medicação. Caso a Sra. Maria não respondesse passado 20 minutos, o assistente iria chamar o cuidador pessoal.
5	Acesso à meteorologia	A Sra. Maria tem uma consulta, logo tem de sair de casa. Como tem as janelas de casa abertas, e receia que a temperatura baixe consideravelmente, e que possa até chover, esta pergunta ao assistente como será a previsão meteorológica para as próximas horas. Obtendo a resposta, a Sra. Maria decide manter as janelas abertas, pois a temperatura irá permanecer amena.

Exemplo do Cenário “Alarme da Medicação”

D. Maria: Olá Assistente! (RNF1 – Interagir com o assistente por voz, RNF2 - Microfone)

Assistente: Bom dia D. Maria. Como posso ajudá-la? (RNF2 - Colunas)

M: Gostava que me lembrasses de tomar um comprimido Alprazolam todos os dias às 21:00. (RF1 – Gestão de uma agenda)

A: Tudo bem, D. Maria. Irei anotar na agenda. Precisa de mais alguma coisa?

M: Não, assistente, obrigada!

A: Estou aqui para ajudar. Até à próxima, D. Maria.

(Mais tarde nesse dia, às 21:00)

A: D. Maria, está na hora de tomar um comprimido Alprazolam!

M: Obrigada por me lembrar assistente!

A: De nada, D. Maria. Posso ser útil em mais alguma coisa?

M: Não, assistente, é só.

A: Estou aqui para ajudar. Até à próxima, D. Maria.

3.3 Requisitos

Os requisitos apresentados, juntamente com os cenários, na tabela anterior podem ser divididos em diferentes níveis de prioridade, para além de serem divididos em funcionais (RF) e não funcionais (RNF). Assim, apresentamos três níveis distintos: Requisitos Prioritários, aqueles que são necessários para uma primeira versão do sistema e que serão os mais importantes para o funcionamento do assistente quando este estiver finalizado; Requisitos Secundários, aqueles que são desejáveis encontrar na versão finalizada e que podem oferecer uma maior robustez ao sistema; Requisitos Extras, aqueles que não são necessários para um bom funcionamento do sistema e que serviriam apenas para adicionar funcionalidades extra. Assim, foi desta forma que dividimos os nossos requisitos:

Requisitos Prioritários:

- RNF1: Interagir com o assistente por voz.
- RNF2 Microfones e colunas.
- RF1: Gestão de uma agenda.
- RF2: Consultar informação meteorológica.
- RNF3: Acesso a recursos exteriores à casa.

Requisitos Secundários:

- RF3: Controlo de dispositivos internos da casa.
- RF4: Capacidade de transmitir informações ao habitante sobre o estado dos dispositivos
- RF5: Capacidade de fazer chamadas.
- RNF4: Acesso do assistente à localização da habitante dentro de casa.

Requisitos Extra:

- RF6: Ter acesso a receitas.
- RF7: Ter conhecimento da qualidade nutricional dos alimentos.
- RNF5: Câmara ou leitor de código de barras
- RNF6: Monitor

Para além disto, na tabela seguinte é possível ver os requisitos agrupados pelos cenários criados.

#	Designação	Requisitos
1	Chamada a um familiar	RNF1 - Interagir com o assistente por voz; RNF2 - Microfones e colunas; RF5 - Capacidade de fazer chamadas;
2	Controlo dos dispositivos (TV, luzes, aquecedores)	RNF1 - Interagir com o assistente por voz; RNF2 - Microfones e Colunas RNF4 - Acesso do assistente à localização da habitante dentro de casa; RF3- Controlo de dispositivos internos da casa; RF4 - Capacidade de transmitir informações ao habitante sobre o estado dos dispositivos;
3	Receitas e valor nutricional de produtos	RNF1 - Interagir com o assistente por voz; RNF2 - Microfones e Colunas; RNF5 - Câmara ou leitor de código de barras; RNF6 – Monitor; RF6 - Ter acesso a receitas RF7 - Ter conhecimento sobre a qualidade nutricional dos alimentos;
4	Alarme da medicação	RNF1 - Interagir com o assistente por voz; RNF2 - Microfones e colunas; RF1 - Gestão de uma agenda;
5	Acesso à meteorologia	RNF1 - Interagir com o assistente por voz; RNF2 - Microfones e colunas; RNF3 - Acesso a informação externa à casa; RF2 - Consultar o tempo meteorológico;

3.4 Arquitetura

A arquitetura do assistente foi baseada nos módulos que foram designados para desenvolver, podendo ser divididos em 3 grandes conjuntos: Interação, Serviços Internos e Serviços Externos.

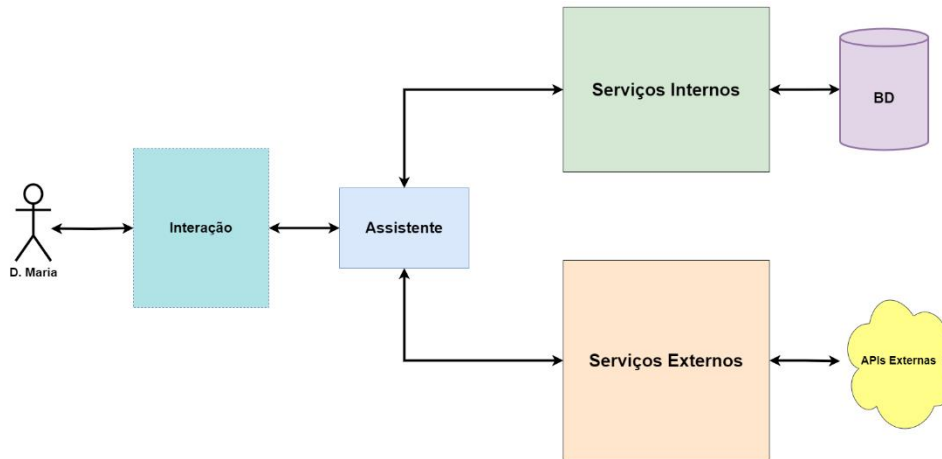


Figura 2 Arquitetura geral do assistente

Olhando para dentro de cada conjunto:

- **Interação** - Tudo aquilo que está relacionado com a interação entre o utilizador, neste caso, a D. Maria, e o assistente é incorporado neste conjunto. Assim, o mesmo inclui o reconhecimento e a síntese de voz, bem como a interface gráfica, para ajuda visual à inquilina da casa.
- **Serviços Internos** – Neste tipo de serviços, estão módulos ligados a algo que ocorre dentro da casa. Temos o primeiro, controlo sobre a casa, como, por exemplo, a iluminação da mesma e o telefone, para gestão de contactos.
- **Serviços Externos** – Neste conjunto estão presentes os módulos que precisam de serviços externos à CasaViva+ e que irão utilizar API's externas. São o caso do módulo da agenda, que irá ser gerida usando a API do Google Calendar [13]. Para além deste teremos um módulo de meteorologia, sendo os valores fornecidos pela API da OpenWeather. Teremos ainda o serviço de telefone, controlo de casa, sugestão de receitas e ainda de Nutri-Score.

Assim, podemos ter uma visão mais específica sobre cada conjunto e os módulos que compõem cada um:

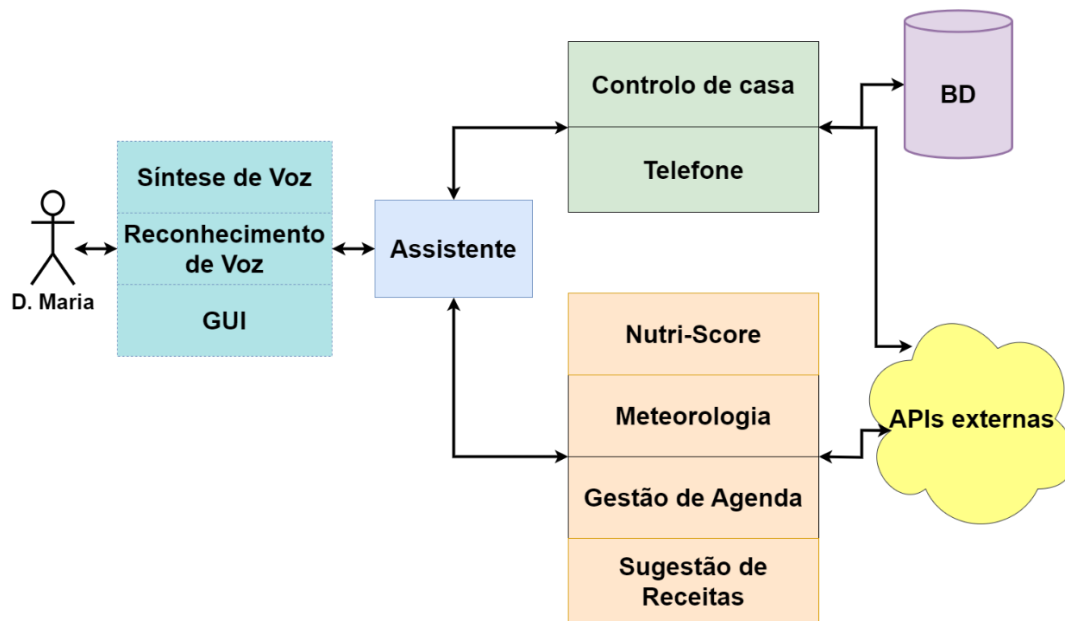


Figura 3 - Arquitetura mais detalhada, incluindo os vários módulos do assistente.

Com isto, o pretendido para o conjunto da interação é que o assistente consiga interagir por voz com o utilizador, de forma célere e que consiga, constantemente, dar feedback ao mesmo quer através do que diz ou através do que poderá mostrar visualmente.

Passando aos restantes módulos, é espectável que o assistente consiga ter controlo sobre a casa e sober os seus dispositivos eletrónicos, como, por exemplo, ligar ou desligar luzes, ligar ou desligar a televisão, aumentar ou diminuir a temperatura, etc. É esperado que o assistente consiga adicionar contactos e fazer chamadas para os mesmos. Na gestão da agenda, o assistente deve conseguir marcar eventos na agenda e deve conseguir listá-los a pedido do utilizador. No módulo da meteorologia é esperado que o assistente consiga responder a perguntas sobre o tempo, bem como fazer uma avaliação de qual o melhor tempo para sair de casa. Deve, ainda, conseguir sugerir receitas para qualquer prato pedido pelo utilizador e deve dizer ao mesmo se um produto é bom para ele o consumir ou não.

4 Ferramentas

4.1 Rasa

Como foi referido anteriormente (secção 2.2), após uma análise e comparação entre as ferramentas encontradas para o desenvolvimento de sistemas de diálogo, decidimos optar pela ferramenta Rasa por vários motivos entre eles (para além dos já mencionados):

- Suporte para a língua portuguesa;
- Linguagem de programação Python;
- Identificação de intenções e extração de entidades;

Para além dos já mencionados.

4.1.1 Arquitetura do Rasa

Como é possível observar na figura 3, o Rasa é constituído por dois grandes componentes: Rasa NLU e Rasa Core. O Rasa NLU é constituído pelo bloco “interpretador” que processa a mensagem de entrada dada pelo utilizador e através desta, identifica intenções e extrai entidades dessas mesmas intenções. Já o Rasa Core recebe a saída da interpretação feita pelo NLU e seleciona a resposta correta a ser produzida pelo assistente.

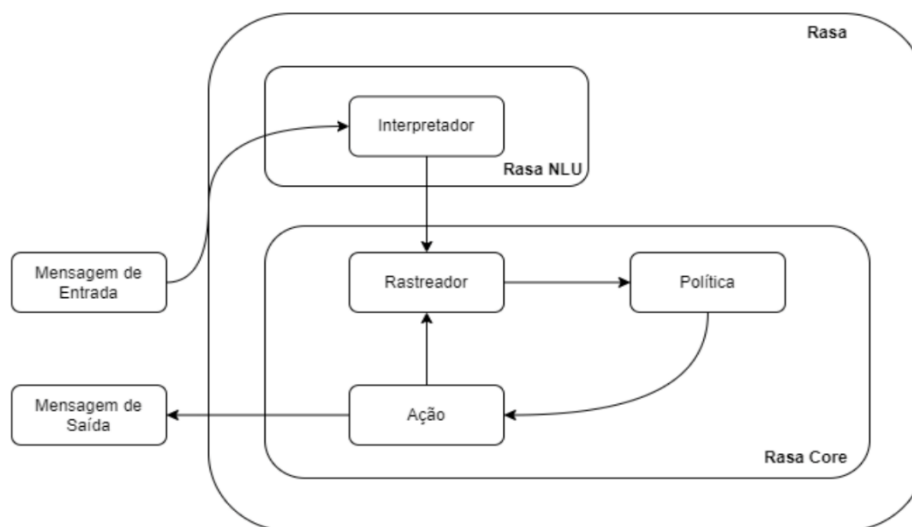


Figura 4 - Arquitetura do Rasa, retirada de [2]

Como é possível observar na figura 3, o Rasa é constituído por duas grandes componentes: Rasa NLU e Rasa Core.

Além disso, dentro do Rasa Core estão definidos 3 blocos:

- **Rastreador** - permite saber o estado da conversa;
- **Política** - recebe o estado do “Rastreador” e escolhe a ação que irá ser executada;
- **Ação** - envia a ação escolhida para o “Rastreador” para ser registada e depois executada. Geralmente a ação leva ao envio de uma mensagem de resposta.

4.1.2 Estrutura do Rasa

Após a criação de um projeto Rasa, é possível observar a seguinte estrutura de ficheiros como é indicado na figura 4.

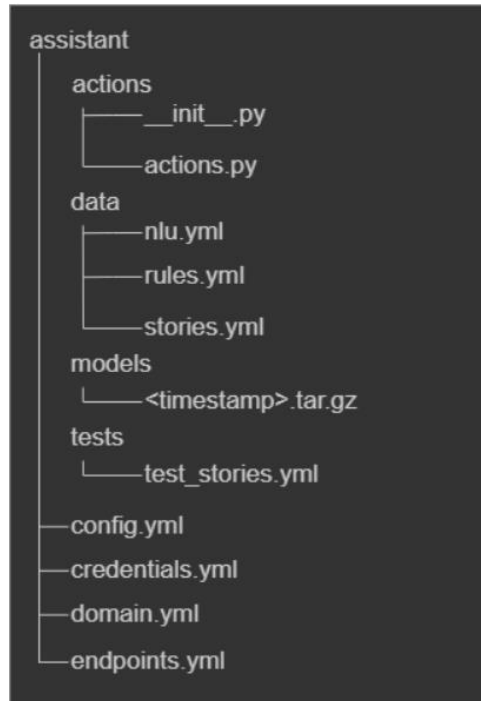


Figura 5 - Estrutura de um projeto Rasa

Dentro destes ficheiros, é dada uma pequena importância extra a três destes ficheiros:

- “domain.yml” - representa tudo que o assistente sabe. Isto inclui responses, intents, slots, entities, forms e actions;
- “nlu.yml” - tem como objetivo ajudar a entender as mensagens do utilizador. Aqui estão definidos os intents e os exemplos;
- “actions.py” - ficheiro em python, onde estão implementadas as ações que o assistente irá realizar.

4.1.3 Conceitos do Rasa

Existem conceitos fundamentais nesta ferramenta que são necessários de entender e compreender antes de a poder começar a usar de forma eficiente.

Entity

O conceito base, mais simples de todos. Uma entidade é um pequeno pedaço de informação que pode ser retirado duma mensagem do utilizador. Ao treinar, o assistente sabe onde ir buscar a informação importante da mensagem do utilizador graças às *Entities*.

Intent

Um *Intent*, ou intenção, como o nome indica é o objetivo geral duma mensagem do utilizador. Esta intenção é acompanhada com vários exemplos de mensagens que contêm o mesmo objetivo por detrás delas.

```
nlu:
- intent: check_balance
  examples: |
- I want to check my [savings account](account_type)
- Can you show me my [current account](account_type)
```

Figura 6: Definição do “intent” *check_balance*

Na imagem acima, retirada de [14], podemos ver um exemplo de *intent*, *check_balance*. Conseguimos perceber, pelo nome e pelos exemplos dados, que o objetivo é verificar quanto está numa conta bancária do utilizador. É ainda visível uma *entity*, *account_type*, que irá direcionar o assistente para onde tem de ir tirar a informação acerca do tipo de conta que é suposto verificar.

Slots

Os *slots* funcionam como memória ao longo da conversa. Nos *slots* podem-se guardar os valores retirados das entidades. No entanto, também se podem armazenar valores que não sejam provenientes de entidades.

```
slots:
  destination:
    type: text
    influence_conversation: false
```

Figura 7: Definição do “slot” *destination*

Na imagem acima, retirada de [15], podemos ver a definição de um *slot*, com o nome *destination*. Na imagem 5, podemos ver uma *entity*, *account_type*. Os valores que esta entidade tomar poderão ser guardados num *slot* com o mesmo nome da *entity* e este será definido de forma semelhante ao da imagem acima.

Stories e Rules

Stories são os dados responsáveis por fazer o treino do modelo de gestão de diálogo do assistente [16]. São uma representação duma possível conversa entre o utilizador e o assistente, onde as mensagens do utilizador se apresentam como *intents* e as respostas do assistente se apresentam como *actions* (apresentado mais à frente).

Rules [17] surgem como seguimento das *stories*, sendo pequenos pedaços de conversa que devem ocorrer seguindo sempre o mesmo caminho.

```
stories:
- story: Greeting and ask user how they're doing
  steps:
  - intent: greet
  - action: utter_greet
  - action: utter_ask_how_doing
  - intent: doing_great
  - action: utter_happy

rules:
- rule: Greeting Rule
  steps:
  - intent: greet
  - action: utter_greet
```

Figura 8: Definição de uma "story" e de uma "rule"

Na imagem acima apresentada, retirada de [18], podemos ver um exemplo de uma *story* e de uma *rule*. Vemos que a *story* segue um caminho definido por *intents* e *actions*, tendo por objetivo cumprimentar o utilizador e perguntar como o mesmo se encontra. A *rule* obriga a que quando o utilizador cumprimenta o assistente, o segundo tem de, obrigatoriamente, responder com um cumprimento. Desta forma, as *stories* definidas devem estar, sempre, de acordo com as *rules*.

Actions

A seguir a cada mensagem do utilizador, o modelo irá prever o que o assistente terá de realizar a seguir, de modo a satisfazer a intenção do utilizador. Desta forma, irá dizer ao assistente que *action* deve concretizar. Existem vários tipos de *actions* [19]:

Responses

Esta é a *action* mais usual e de frequente utilização. Trata-se de uma mensagem que o assistente irá mandar para o utilizador, com texto, imagens, botões, etc. Na imagem, retirada de [20], a seguir representada, podemos ver dois exemplos deste tipo de *action*, *utter_greet* e *utter_bye*, ambas contendo texto.

```
responses:
  utter_greet:
  - text: "Hi there!"
  utter_bye:
  - text: "See you!"
```

Figura 9: Definição de "responses (*utter_greet* e *utter_bye*)"

Custom Actions

Este tipo de *action* é utilizado para correr qualquer tipo de código que queiramos, como, por exemplo, chamadas a API's ou queries para base de dados. Estas *actions* são definidas num ficheiro "actions.py" e terão o seguinte formato, demonstrado na seguinte imagem, retirada de [21]:

```
class ActionTellTime(Action):

    def name(self) -> Text:
        return "action_tell_time"

    def run(self, dispatcher: CollectingDispatcher,
            tracker: Tracker,
            domain: Dict[Text, Any]) -> List[Dict[Text, Any]]:
        current_place = next(tracker.get_latest_entity_values("place"), None)
        utc = arrow.utcnow()

        if not current_place:
            msg = f"It's {utc.format('HH:mm')} utc now. You can also give me a place."
            dispatcher.utter_message(text=msg)
            return []

        tz_string = city_db.get(current_place, None)
        if not tz_string:
            msg = f"I didn't recognize {current_place}. Is it spelled correctly?"
            dispatcher.utter_message(text=msg)
            return []

        msg = f"It's {utc.to(city_db[current_place]).format('HH:mm')} in {current_place} now."
        dispatcher.utter_message(text=msg)

        return []
```

Figura 10: Exemplo de uma "custom action"

Como podemos ver, cada *action* terá as duas funções *name* e *run*. A primeira serve, apenas, para fazer a ligação entre a *action* e o *domain*, indicando que a classe *ActionTellTime* corresponderá a *action_tell_time*. Dentro da segunda irá ficar todo o código que queremos que seja efetuado quando a *action* é chamada.

Forms

Os *forms* são um tipo de "Custom Actions" que está direcionado à forma de lidar com a *business logic*. Se existir algum design de conversa, onde é expectável ao assistente perguntar sobre um conjunto específico de informações, este tipo de "action" deve ser utilizado. O "form" abaixo apresentado, retirado de [22], está especificado de forma a perguntar ao utilizador informações, que estarão representadas nos "slots" *cuisine* e *num_people*.

```
forms:
  restaurant_form:
    required_slots:
      - cuisine
      - num_people
```

Figura 11 3: Exemplo de um "form"

Default Actions

Este tipo de *actions* estão integradas no *Dialogue Manager* por defeito. A maior parte é prevista em certas situações e são passíveis de customização. Na imagem a seguir, retirada de [23], podemos ver um exemplo de uma destas *actions*, a *ActionRestart*, que pode ser customizada.

```
class ActionRestart(Action):  
  
    def name(self) -> Text:  
        return "action_restart"  
  
    async def run(  
        self, dispatcher, tracker: Tracker, domain: Dict[Text, Any]  
    ) -> List[Dict[Text, Any]]:  
  
        # custom behavior  
  
        return [...]
```

Figura 12: Exemplo de uma "Default Action"

Slot Validation Actions

Este tipo de *action* é usado para lidar com a extração e/ou validação de valores das *slots*. Na imagem seguinte, retirada de [24], é possível ver uma *action* de validação do *slot location*.

```
from typing import Text, Any, Dict  
  
from rasa_sdk import Tracker, ValidationAction  
from rasa_sdk.executor import CollectingDispatcher  
from rasa_sdk.types import DomainDict  
  
class ValidatePredefinedSlots(ValidationAction):  
    def validate_location(  
        self,  
        slot_value: Any,  
        dispatcher: CollectingDispatcher,  
        tracker: Tracker,  
        domain: DomainDict,  
    ) -> Dict[Text, Any]:  
        """Validate location value."""  
        if isinstance(slot_value, str):  
            # validation succeeded, capitalize the value of slot  
            return {"location": slot_value.capitalize()}  
        else:  
            # validation failed, set this slot to None  
            return {"location": None}
```

Figura 13: Exemplo de uma "action" de validação de "slots"

4.2 Google Calendar API

Inicialmente, pensámos fazer um serviço interno de gestão de agenda. No entanto, de forma a esse serviço ser completo, o tempo despendido iria ser bastante superior àquele que poderíamos gastar. Com isto em mente, optámos por outra opção, a utilização da API do Google Calendar.

Esta API expõe a maioria dos recursos disponíveis na interface da Web do Google Calendar. Para utilizar esta API é necessário criar uma chave de API, obtendo credenciais para a utilização da mesma. Esta API tem uma utilização gratuita por 3 meses e até 300 dólares por mês. Esta limitação foi aceite, uma vez que para a realização do projeto não causa transtorno. Desta forma, os conceitos importantes são apenas dois: Evento e Agenda.

Utilizamos a API para adicionar eventos a uma agenda, eventos esses que vão ter título, horário de início e fim do evento e poderão apresentar características como localização e convidados. Existe ainda a possibilidade de criar eventos recorrentes ou eventos de acontecimento único. A agenda guarda todos os eventos criados e envia lembretes e notificações conforme seja configurado o evento.

4.3 Google Cloud

A Google Cloud oferece dois serviços, da mesma forma que o Google Calendar API, grátis durante 3 meses e com utilização até 300 dólares mensais, o Google Cloud Speech-To-Text [25] e o Google Cloud Text-To-Speech [26]. O primeiro serve para transformar fala para texto, tendo sido utilizado na fase de reconhecimento de texto para reconhecer a fala e a transformar em texto, sendo este texto posteriormente enviado para o NLU para ser interpretado. O segundo é utilizado na parte de síntese de texto, já depois de o Rasa ter processado o texto e ter obtido uma resposta, passando então essa resposta de texto para fala e transmitindo ao utilizador a informação que foi obtida, tendo em conta o pedido por ele feito.

4.4 OpenWeather API

Ponderámos bastantes APIs diferentes para utilizar no módulo da meteorologia, entre as quais “AccuWeatherAPI” [27] e “IPMA API” [28], mas acabámos por optar por escolher a “OpenWeather API” [29], uma vez que esta é fácil de usar e não tem custos adicionais (podemos fazer até 1 000 chamadas gratuitas por dia). O único problema da mesma é a limitação de previsão de dias à frente, fornecendo apenas até 5 dias. Para o caso de querermos mais, já seria necessário pagar. No entanto, considerámos que este não era um problema para o projeto, uma vez que o que queremos demonstrar não necessita de previsões superiores a 5 dias.



4.5 Outras Ferramentas

Para além destas ferramentas, utilizámos também ferramentas desenvolvidas pelo orientador Nuno Almeida, para os módulos do Telemóvel, Controlo de Casa e Datas de Validade. Utilizámos os *endpoints* para chamadas às APIs criados e fornecidos pelo mesmo, e adaptámos ao que pretendíamos utilizar, protegendo o assistente de possíveis falhas nessas APIs, completando assim os módulos referidos.

5 Desenvolvimento

5.1 Metodologia de desenvolvimento

Conforme mencionada na arquitetura, secção 3.4, o nosso projeto é composto por módulos distintos. Para simplificar e estruturar o desenvolvimento desses módulos, estabelecemos uma metodologia específica. Essa metodologia é organizada em fases, cada uma com objetivos claros e atividades bem definidas. Nesta seção, vamos apresentar em detalhe as fases do processo e as suas respectivas descrições. Para uma melhor compreensão, vamos usar exemplos no processo de desenvolvimento do módulo da agenda.

5.1.1 Fase 1: Descrição de Cenários

Esta fase consiste na identificação e descrição de uma situação de uso do módulo em questão. A compreensão completa dos cenários é fundamental para o sucesso do desenvolvimento.

Exemplo: secção 3.2.

5.1.2 Fase 2: Extração dos Casos de Uso

Nesta fase, os casos de uso são extraídos dos cenários identificados anteriormente. Cada caso de utilização é analisado separadamente, identificando as ações do utilizador e as respostas esperadas do sistema. Esta etapa permite a identificação das principais funcionalidades do módulo.

Exemplos: “Adicionar um lembrete”, “adicionar um evento recorrente”, “consultar eventos do dia”, etc.

5.1.3 Fase 3: Listagem de frases

Durante esta fase, são listadas as expressões que o utilizador poderá usar e que o sistema precisa reconhecer. Essas expressões podem incluir comandos, perguntas ou solicitações específicas dos utilizadores. A criação de uma lista abrangente é fundamental para garantir a eficiência e a precisão do sistema.

Exemplos: “Tenho uma consulta às 15:00, na segunda-feira”, “Marca na agenda um passeio com a Rute na quinta às 3 da tarde”, “Tenho alguma coisa marcada para sábado depois das 18:00?”, etc.

5.1.4 Fase 4: Definição dos “Intents”

Como já definido na subsecção 4.1.3., figura 5, os *intents* vão agrupar as expressões utilizadas pelos utilizadores em categorias. A correta definição dos *intents* é essencial para a compreensão e a resposta apropriada do sistema, de realçar que, as partes entre parêntesis retos são os blocos de informação a ser guardados, ou seja, a *entity*, e o nome a seguir entre parêntesis curvos é o nome da mesma. Na figura seguinte, podemos ver um exemplo de um *intent*, *find_events*, cujo objetivo é descobrir que eventos o utilizador tem marcados num certo período de tempo.

```
- intent: find_events
examples: |
  - Tenho alguma coisa marcada para [amanhã](day_of_week)?
  - O que é que tenho marcado para a [próxima semana](duration)?
  - O que é que tenho marcado para a [esta semana](duration)?
  - Que eventos tenho no dia [29 de maio](day)?
  - Que eventos tenho no dia [13 de junho](day)?
  - Que eventos tenho no dia [14 de fevereiro](day)?
  - Que eventos tenho no dia [17 de agosto](day)?
  - Tenho algo marcado para [quinta](day_of_week)?
  - Tenho algo marcado para [terça](day_of_week)?
  - Tenho algo marcado para [sabado](day_of_week)?
  - Tenho algo marcado para [quarta](day_of_week)?
  - Tenho algo marcado para [sexta](day_of_week)?
```

Figura 14: Intent usado para saber que eventos o utilizador tem marcados

5.1.5 Fase 5: Identificação das *Entities* e *Slots*

Em cada exemplo definido para cada *intent* anterior, faz-se uma análise para determinar quais os blocos de informação que vão ser guardados. Como se pode notar na figura 13, presente na subsecção 5.1.4, é possível observar a *entity* e o nome para cada um dos exemplos.

Para poder usar a informação recolhida das *entities*, estas têm que ser guardadas em *slots*. A seguir está apresentado um exemplo de um *slot*. Neste caso, o *slot* day irá guardar valores de texto, que sejam retirados da *entity* day, podendo influenciar o curso da conversa entre assistente e utilizador.

```
day:
  type: text
  influence_conversation: true
  mappings:
    - type: from_entity
      entity: day
```

Figura 15: “Slot” day

5.1.6 Fase 6: Desenvolver as *Actions* e Ligação com o Módulo Externo

É preciso perceber o que é que precisamos que o agente faça. Às vezes pode ser apenas enviar uma mensagem de resposta. Mas podemos querer que o nosso agente faça mais como agendar eventos, efetuar cálculos, procurar informação numa base de dados, enviar um email, etc. E, para todos estes cenários, é preciso mais do que respostas estáticas para resolver a situação. De modo a implementar isto foi desenvolvido, em Python, para além do ficheiro *actions*, um ficheiro para cada um dos módulos externos.

A tarefa a ser realizada pode fazer uso da informação recolhida nas *entities* e armazenada nas *slots*.

Na maior parte dos casos, as tarefas fazem uso de módulos externos (por exemplo, a agenda usa o Google Calendar) e para termos uma independência entre esses módulos e as ações, criámos um serviço que serve de “intermediário” entre eles. Isto permite que caso um serviço externo falhe, o sistema em geral não falhe. E também, caso no futuro queiramos mudar o serviço, isso poderá ser feito sem alterar as *actions* do Rasa.

5.2 Mudanças ao Plano Inicial

Com o avançar do projeto, foi necessário fazer alterações ao plano inicial, sejam estas por escolha ou por necessidade.

Como já foi mencionado anteriormente, o módulo da gestão de agenda depende da utilização da API do Google Calendar [13], sendo assim um serviço externo. No entanto, no planeamento inicial, a agenda faria parte do módulo interno, isto é, iria-se recorrer a uma base de dados dedicada a armazenar os eventos e os seus dados associados. Contudo, aliada à complexidade de ferramentas novas a aprender para a realização do projeto, a utilização de uma API externa permitiu agilizar o desenvolvimento deste módulo, permitindo um foco maior na aprendizagem das ferramentas de maior complexidade, sendo que estas são cruciais para o projeto.

Para além disso, inicialmente foi criado um cenário que permitiria reconhecer a validade de diferentes produtos, através de scan do seu código de barras. No entanto, devido a dificuldades em encontrar uma ferramenta fiável no que diz respeito à apresentação desses valores, aliado também a um crescente interesse pela qualidade nutricional do produto, surgiu a ideia de apresentar o “Nutri-score” do produto, com valores entre A e E, de acordo com a pontuação nutricional do mesmo. Para tal, recorreu-se à utilização da API Open Food Facts [30].

6 Testes e Resultados

6.1 Testes

Para testar o assistente que desenvolvemos, decidimos fazer testes com 10 pessoas, com uma média de idades de 62 anos, estando as mesmas compreendidas entre 45 e 77. Para isto, criámos 5 tarefas de teste diferentes.

Tarefa	Designação	Descrição
1	Teste à Agenda	Para este teste, tínhamos como objetivo que os participantes fizessem a marcação de um evento na agenda. Caso pedissem ajuda, eram instruídos a tentarem marcar um almoço na agenda no dia 12 de junho às 13:00.
2	Teste à Meteorologia	Neste teste, o objetivo para os participantes era descobrir qual era a melhor hora do dia para sair de casa. Caso necessitassem de auxílio, eram instruídos a perguntar ao assistente qual seria a melhor hora do dia para ir ao parque dar uma volta.
3	Teste ao Controlo da Casa	Para este teste, os participantes deveriam tentar controlar a iluminação de várias divisões. Uma vez que os testes foram realizados fora do laboratório, este teste foi feito com uma simulação de luzes, visto que apenas na rede do laboratório o módulo funciona.
4	Teste ao Nutri-Score	Este teste tinha como objetivo os participantes descobrirem se um alimento era bom para consumo ou não. Se fosse solicitada ajuda, iriam ser instruídos a apontar o código de barras para a câmara usada para os testes.
5	Teste à sugestão de receitas	Neste último teste, o objetivo seria os participantes pedirem uma receita do seu prato preferido. Caso não conseguissem, seriam instruídos a pedir uma receita de salada russa.

Com isto, apresentamos, a seguir, os resultados obtidos:

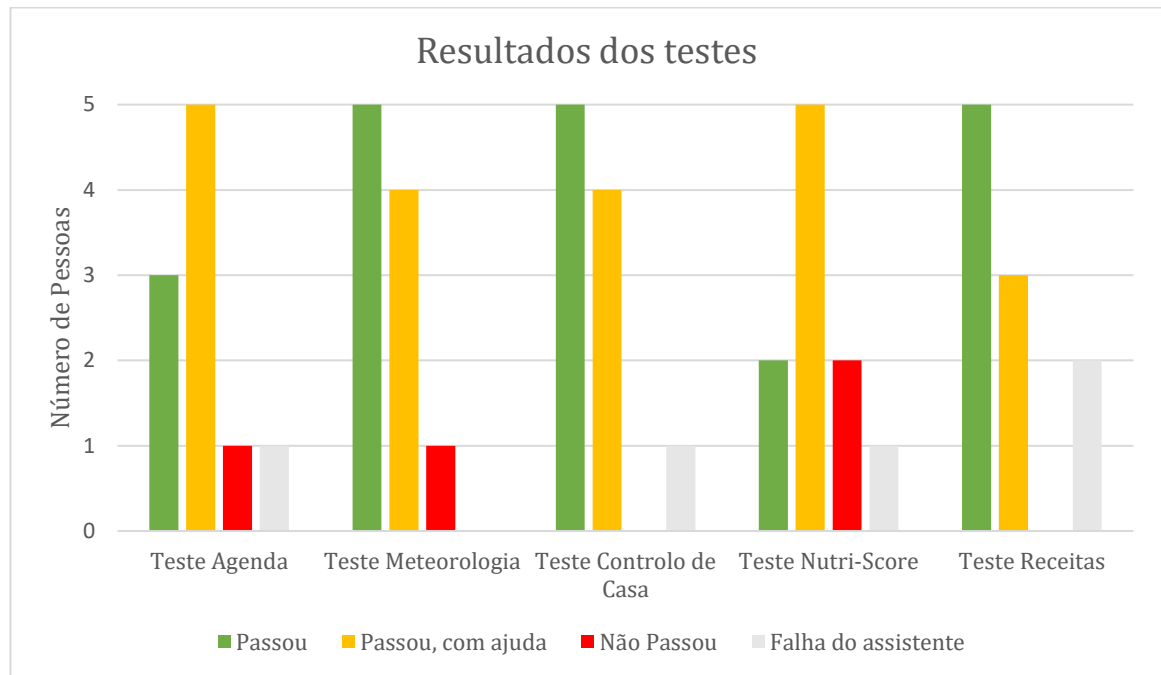


Figura 16: Tabela com resultado dos testes realizados

Após a realização dos testes conseguimos ver que a maior parte dos participantes conseguiu, com ou sem ajuda, concluir os testes, apenas quatro testes foram falhados por um ou dois participantes e o assistente ainda apresentava algumas falhas ou existiu algum problema com as APIs usadas durante os testes. Foram resultados entusiasmantes, uma vez que os estudos supracitados afirmavam que a maioria dos participantes levava algum tempo a adaptar-se aos assistentes. No entanto, uma vez que as idades de alguns participantes são um bocado inferior à idade ideal, os resultados podem ter sido ligeiramente influenciados.

Para além disso, os resultados dos testes permitiram treinar melhor o nosso assistente, adicionando exemplos válidos aos nossos *intents*. Foi-nos ainda sugerido adicionar ao assistente a capacidade de demonstrar frases de exemplo para cada funcionalidade, algo que considerámos bastante pertinente e que foi, prontamente, realizado.

6.2 Resultados

Esta subsecção inclui a análise que foi feita de forma independente para cada módulo, para compreender melhor o que foi atingido em cada um. Começando pela parte da interação entre o utilizador e o assistente, temos os módulos de síntese e de reconhecimento de voz e da interface. Nos módulos de síntese e de reconhecimento de voz, conseguiu-se implementar um sistema que compreende o que lhe é dito através do reconhecimento de voz e consegue sintetizar texto para o reproduzir, de seguida. Para além disso, implementámos uma interface complementar que demonstra a conversa que está a decorrer e tem ainda a capacidade de mostrar mais alguns pedidos, como, por exemplo, o calendário.

Passando aos serviços externos, temos os módulos da gestão da agenda, da meteorologia e do telemóvel. Começando pela agenda, conseguimos utilizar o Google Calendar para guardar os eventos criados pelo utilizador, com nome do evento, data, hora, localização e participantes, ainda com a possibilidade de ser recorrente ou único, e, também, listá-los. Na parte da meteorologia, conseguimos fazer questões sobre o tempo e realizar uma sugestão acerca da melhor hora para sair de casa. Esta segunda é baseada numa tabela de pontuações, que indica a melhor hora para o utilizador sair de casa entre as 10 e as 19 horas, tendo em conta que o sistema é idealizado para pessoas com uma idade mais avançada. Já na parte do telemóvel, o assistente consegue fazer chamadas e guardar contactos. As chamadas são feitas através do protocolo SIP [31]. Este protocolo pode ser usado entre dois ou mais participantes para estabelecer chamadas e conferências através de uma rede IP. O estabelecimento, mudança ou término da sessão é independente do tipo do meio de comunicação ou aplicação que será usada na chamada, sendo que uma chamada pode utilizar diferentes tipos de dados, incluindo áudio e vídeo.

Já nos serviços internos, podemos encontrar o controlo de dispositivos, a sugestão de receitas e a gestão de validade dos produtos. Começando pelo controlo de dispositivos, o assistente consegue controlar as luzes e tomadas elétricas, bem como dizer o consumo de cada um desse tipo de dispositivos. Na parte de sugestão de receitas, o assistente fornece ao utilizador uma receita da iguaria que é pedida, sendo a mesma demonstrada na interface visual. Na parte da gestão de validade de produtos, o produto é passado numa câmara, onde é lido o código de barras e é indicado o Nutri-Score do produto e se este é indicado para o consumo do utilizador.

6.3 Limitações e Dificuldades

Começando pelas dificuldades, como para todos nós o Rasa era uma ferramenta nova, foi fundamental, numa fase inicial, a presença de todos os membros na descoberta e aprendizagem da mesma. Desta forma, o progresso no projeto, nesta fase, foi lento e reduzido. Isto acabou por nos limitar um bocado no desenvolvimento de algumas funcionalidades.

Outra dificuldade sentida esteve presente na tentativa de implementar um sistema de lembrete no módulo da agenda, algo que não foi conseguido e teve de ser abandonado.

Para além destas, talvez a maior dificuldade, foi coordenar um grupo de 6 pessoas, de forma a existir organização e que todos estivessem focados no mesmo objetivo. Apesar de no passado já termos tido trabalhos de grupo, até com 6 pessoas, nunca tínhamos tido um projeto desta dimensão e consideramos que isso tenha sido, também, uma dificuldade, na parte inicial do projeto.

Passando às limitações, e apesar de considerarmos que fizemos bastante do que, inicialmente, foi idealizado, o projeto contém algumas limitações. Algumas estão relacionadas com as APIs, do lado da Google, onde temos apenas 3 meses de utilização gratuita, com um limite máximo de 300 dólares por mês, e do lado da OpenWeather, onde temos previsões apenas até 5 dias à frente e apenas com atualizações de 3 em 3 horas.

Outras limitações estão relacionadas com o facto de ainda não existir uma casa física da CASAVIVA+. Com isto, não foi possível desenvolver mais o módulo de controlo da casa, visto

que apenas tínhamos acesso ao controlo sobre luz, não sendo possível desenvolver o controlo sobre televisão e sobre aquecedor. Associado também a isto, está o facto dos módulos do telefone e do controlo de casa só funcionarem na rede do laboratório.

Por fim, existem também algumas limitações nos módulos desenvolvidos. Como já mencionado acima, o assistente carece da capacidade de relembrar o utilizador de quando um evento está prestes a começar. Ainda no módulo da agenda, o assistente não consegue remover ou editar eventos já criados. Outra limitação é a interface gráfica, uma vez que esta está algo rudimentar.

7 Conclusão

A implementação de um assistente de voz direcionado a pessoas idosas apresenta-se como uma solução promissora para atender às necessidades e desafios enfrentados por essa população. Com o envelhecimento da sociedade, é fundamental desenvolver tecnologias acessíveis e intuitivas que possam melhorar a qualidade de vida dos idosos e promover sua independência.

Ao longo deste relatório, discutimos os principais problemas dos assistentes de voz já existentes, bem como os benefícios de um assistente desenhado especialmente para idosos, tal como a facilidade de utilização e a capacidade de comunicação natural.

A implementação de um assistente de voz projetado para idosos pode contribuir significativamente para promover o envelhecimento saudável, facilitar a independência e fortalecer a inclusão digital dessa população. É um passo importante rumo a um futuro mais acessível e inclusivo, no qual todos possam desfrutar dos benefícios da tecnologia de forma igualitária.

A um nível mais pessoal, este projeto permitiu-nos desenvolver um tipo de projeto totalmente diferente do que o que tínhamos desenvolvido até agora, uma vez que para além de existir integração de voz, algo que nenhum de nós tinha feito antes, deixou de existir um foco tão importante na interface visual, já que esse foco estava na utilização da voz.

Para além disso, trabalhamos com ferramentas que ainda não tinham sido usadas pela maioria dos elementos do grupo, o que nos permitiu um alargamento extenso dos nossos conhecimentos e das nossas competências.

Tivemos, também, a oportunidade de perceber como se deve organizar um projeto de grupo, retirando que os três aspetos mais importantes para um projeto deste tamanho são a comunicação, a cooperação e o escalonamento das tarefas.

7.1 Trabalho Futuro

Tendo em conta o que foi dito anteriormente, se tivéssemos mais tempo para este projeto, gostaríamos de acabar as funcionalidades que faltam implementar no módulo da agenda. Seria, também, prioritário desenvolver uma interface gráfica que representasse um chat com o diálogo que está a ser feito, de forma a que o utilizador tenha feedback sobre o que está a ser entendido pelo assistente e, ainda, gráficos sobre os consumos da casa, bem como pedidos do utilizador como, por exemplo, a sua agenda ou alguma receita.

Seria também expectável, caso o projeto fosse continuado, que já existisse uma instância da CASAVIVA+, onde pudéssemos testar o assistente já existente, para podermos continuar a desenvolver as funcionalidades relacionadas com o controlo da casa, como foi mencionado previamente.

Para além disto, seria de grande interesse começar a usar o assistente em situações reais para perceber que outras funcionalidades poderiam ser adicionadas aos módulos já existentes



ou que módulos se poderiam criar, de modo a conseguir criar o maior conforto para os futuros utilizadores.

7.2 Agradecimentos

Aproveitamos, ainda, este relatório para agradecer aos nossos orientadores, António Teixeira, Ana Rocha e Nuno Almeida por todo o esforço que tiveram para nos dar as melhores condições possíveis para realizar este projeto e pela prontidão com que nos ajudaram em todos os momentos da fase de desenvolvimento do mesmo.

8 Referências

- [1] - “Impulso Positivo” - <https://impulsopositivo.com/censos-2021-seniores-representam-234-da-populacao-portuguesa/>
- [2] - Almeida, T. (2022). Assistant for Supporting the Elderly Staying at Home. Dissertação de Mestrado em Engenharia de Computadores e Telemática, Universidade de Aveiro.
- [3] - Young, S. (2009). *CUED Standard Dialogue Acts*. <http://mi.eng.cam.ac.uk/~sjy/papers/youn09.pdf>
- [4] - Barcelona, S. S. U. de, Sayago, S., Barcelona, U. de, University, B. B. N. M., Neves, B. B., University, M., Benjamin R Cowan University College Dublin, Cowan, B. R., Dublin, U. C., & Metrics, O. M. V. A. (2019, August 1). *Voice assistants and Older People: Proceedings of the 1st International Conference on Conversational user interfaces*. ACM Other conferences. Retrieved March 10, 2023, from: https://dl.acm.org/doi/abs/10.1145/3342775.3342803?casa_token=EyhBF8vvv4cAAAAA%3A44LS_ZoIeLlhrbxSYdYSLxBIa3EeW4II9TxeMTRHwqP2LQtQsQ7g5NSZVWPZFc7dsBml6TC0znw
- [5] - Maryland, A. P. U. of, Pradhan, A., Maryland, U. of, Maryland, A. L. U. of, Lazar, A., Washington, L. F. U. of, Findlater, L., Washington, U. of, Technology, K. T. H. R. I. of, & Metrics, O. M. V. A. (2020, August 1). *Use of intelligent voice assistants by older adults with low technology use*. ACM Transactions on Computer-Human Interaction. Retrieved March 10, 2023, from: <https://dl.acm.org/doi/10.1145/3373759>
- [6] - Sunyoung Kim, A. T., Hoof, J. V., Mitzner, T. L., McLean, G., Liu, L., Bickmore, T. W., Abdolrahmani, A., Ammari, T., Azevedo, R. F., Bentley, F., Blair, J., Braun, V., Cassell, J., Cho, M., Clark, L., Cowan, B. R., Delello, J. A., Druga, S., ... Lopatovska, I. (2021, June 14). *Exploring older adults' perception and use of smart speaker-based voice assistants: A longitudinal study*. Computers in Human Behavior. Retrieved March 10, 2023, from: https://www.sciencedirect.com/science/article/abs/pii/S0747563221002375?casa_token=

[0vFdYOU8ka0AAAAA%3A2pJQT2NMb00N6fsxlDoTbitXEqmL6iWR7EfBASH9wH2lOxMgdai](https://doi.org/10.1007/978-1-4939-9888-8_10)
[aJ9VwHnsauqZiU-h0Raee](#)

[7] - Kim, S., Information, S. of C. and, & Kim, C. A. S. (n.d.). *Exploring how older adults use a smart speaker-based voice assistant in their first interactions: Qualitative study*. JMIR mHealth and uHealth. Retrieved March 10, 2023, from: <https://mhealth.jmir.org/2021/1/e20427>

[8]- Maryland, A. P. U. of, Pradhan, A., Maryland, U. of, Washington, L. F. U. of, Findlater, L., Washington, U. of, Maryland, A. L. U. of, Lazar, A., University, S., University, N., Fxpal, & Metrics, O. M. V. A. (2019, November 1). "Phantom friend" or "just a box with information": Personification and ontological categorization of smart speaker-based voice assistants by older adults. Proceedings of the ACM on Human-Computer Interaction. Retrieved March 10, 2023, from <https://dl.acm.org/doi/abs/10.1145/3359316>

[9] - "Microsoft Bot Framework" - <https://dev.botframework.com/>

[10] - "Amazon Lex" - https://en.wikipedia.org/wiki/Amazon_Lex

[11] - "DialogFlow" - <https://en.wikipedia.org/wiki/Dialogflow>

[12] - "Rasa" - <https://rasa.com/docs/rasa/>

[13] "Google Calendar API" - <https://developers.google.com/calendar/api/quickstart/python?hl=pt-br>

[14] - "Entities and Intents" - <https://learning.rasa.com/conversational-ai-with-rasa/entities/>

[15] - "Slots" - <https://learning.rasa.com/archive/conversational-ai-2/slots/>

[16] - "Stories" - <https://rasa.com/docs/rasa/stories/>

[17] - "Rules" - <https://rasa.com/docs/rasa/rules/>

[18] - "Writing Stories" - <https://rasa.com/docs/rasa/writing-stories/>

[19] - "Actions" - <https://rasa.com/docs/rasa/actions/>

[20] - "Responses" - <https://rasa.com/docs/rasa/responses/>

[21] - "Custom Actions" - <https://learning.rasa.com/conversational-ai-with-rasa/custom-actions/>

[22] - "Forms" - <https://rasa.com/docs/rasa/forms/>

- [23] – “Default Actions” - <https://rasa.com/docs/rasa/default-actions/>
- [24] – “Slot Validation Actions” - <https://rasa.com/docs/rasa/slot-validation-actions/>
- [25] – “Google Cloud Speech-To-Text” - <https://cloud.google.com/speech-to-text?hl=pt-br>
- [26] – “Google Cloud Text-To-Speech” - <https://cloud.google.com/text-to-speech?hl=pt-br>
- [27] – “AccuWeatherAPI” - <https://developer.accuweather.com/>
- [28] – “IPMA API” - <https://api.ipma.pt/>
- [29] - “OpenWeather API” - <https://openweathermap.org/api>
- [30] - “API Open Food Facts” - <https://world.openfoodfacts.org>
- [31]- “SIP – Session Initiation Protocol” - [https://pt.wikipedia.org/wiki/Protocolo de Inicia%C3%A7%C3%A3o de Sess%C3%A3o](https://pt.wikipedia.org/wiki/Protocolo_de_Inicia%C3%A7%C3%A3o_de_Sess%C3%A3o)
- [32] - Daniel, J., & Martin, J. (n.d.). *Speech and Language Processing Chatbots & Dialogue Systems*. <https://web.stanford.edu/~jurafsky/slp3/15.pdf>

9 Anexos

Nome	Bernardo Leandro 98652	Carlos Sena 81377	Diogo Silva 98644	Luís Martins 98521	Manuel Diaz 103645	Pedro Coelho 104247
Estado de Arte						
Cenários e Requisitos						
Arquitetura						
Listagem de Expressões (utterances)						
Agenda – Criação de Eventos						
Agenda – Listagem de Eventos						
Meteorologia – Queries sobre tempo						
Meteorologia – Recomendação de saída						
Controlo da Casa - Iluminação						
Controlo da Casa - Consumo						
Telefone						
Sugestão de Receitas						
Nutri-Score						
Reconhecimento de Voz						
Síntese de Voz						
Interface						
Melhorar a coesão do assistente						
Poster						
Gravação do Vídeo						
Edição do Vídeo						
Site						
Apresentações						
Relatório - Cap. 1						
Relatório - Cap. 2						
Relatório - Cap. 3						
Relatório - Cap. 4						
Relatório - Cap. 5						
Relatório - Cap. 6						
Relatório - Cap. 7						
Relatório - Ajustes						
Abstract						



Repositório do projeto: <https://github.com/BernardoLeandro1/PI>

Para instalar o projeto, devem correr os seguintes comandos:

- 1 - `python3 -m venv venv`
- 2 - `source venv/bin/activate`
- 3 - `pip install -r requirements.txt`
- 4 - `pip install --upgrade google-api-python-client oauth2client`
- 5 - `pip install --upgrade google-api-python-client google-auth-httpplib2 google-auth-oauthlib`
- 6 - `rasa run -m models --endpoints endpoints.yml --port 5002 --credentials credentials.yml`
- 7 - Novo terminal: `rasa run actions`
- 8 - Novo terminal: `export GOOGLE_APPLICATION_CREDENTIALS="./speechToTextKey.json"`
- 9 - `python3 ./main.py`