UNIVERSIDADE DE LISBOA
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE INFORMÁTICA

Plataforma e Metodologia de Desenvolvimento de Agentes Conversacionais Proativos

Tiago Alexandre de Castro Pires

Mestrado em Engenharia Informática

Trabalho de Projeto orientado por:
Prof. Doutor João Carlos Balsa da Silva
e Prof. Doutora Ana Paula Boler Cláudio

2025

Agradecimentos

Este trabalho nunca teria sido concluído sem o apoio de imensas pessoas.

A começar pelos meus orientadores, o Professor Doutor João Balsa da Silva e a Professora Doutora Ana Paula Cláudio, um imenso obrigado pela paciência demonstrada e pelo esforço que fizeram para que fosse possível entregar esta dissertação. Um obrigado não é suficiente para o apoio que me proporcionaram.

Depois à minha família, em especial aos meus pais e à minha irmã, que me deram espaço para enfrentar este e tantos outros desafios que me surgiram ao longo do último ano, sem nunca recuarem e dando-me apoio sempre que precisava a nível pessoal. Sempre me incentivaram, sempre deram tudo por mim, e não tenho forma de agradecer o amor que me deram.

Um grande agradecimento à minhna psicóloga, Dra. Mónica Sobral. Foi um ano de trabalho duro, mas se cheguei a este ponto foi muito graças a esse trabalho.

Um muito obrigado também a todos os meus amigos dos BL, que me ajudaram a distrair quando precisava de descansar, que estavam comigo quando precisava de sair do mesmo sítio onde trabalhava, e que me iam perguntando um de cada vez como tudo estava a correr por não me quererem pressionar. Nunca me pressionaram mais do que já me fazia a mim mesmo, e só o facto de se lembrarem de mim já era mais importante do que podem imaginar. Pelas idas ao futebol e ao cinema, e por todos os desafios que enfrentámos juntos, um obrigado especial à Inês, Joel, Pêra, André, Diogo, Empi, Filipe, David, Mário, Teixeira e Velho. Espero conseguir estar lá para vocês como vocês estiveram para mim.

E por fim aos que estiveram comigo ao longo dos últimos anos. Seja em almoços, a trabalhar ao meu lado ou numa chamada. Todos me trouxeram a este momento e não vos podia estar mais agradecido. O apoio que me deram em relação a tudo o que me tenhho dedicado é do melhor que me podiam dar. Porque nunca me poderia esquecer do Resto, um grande obrigado em especial ao David, que me ouviu e esteve comigo quando mais precisava de companhia para trabalhar, e ao Dani, que mais me tem divertido ao longo destes anos. Que venham muitas viagens e um par de Sunsets juntos.

Aos meus pais, porque nunca poderia ser a mais ninguém.

Resumo

O desenvolvimento de agentes conversacionais tem assistido a um progresso imenso nos últimos anos, sendo utilizado em diálogos cada vez mais extensos e complexos com utilizadores do sistema. Da mesma forma, aplicações de saúde móveis (aplicações *mHealth*) têm encontrado o seu espaço no mercado, ajudando os seus utilizadores a atingir os objetivos a que se propõem. Apesar deste crescimento nas duas frentes, não há uma estrutura unificada do que se espera na arquitetura de um Agente Conversacional dedicado a esta área.

Partindo das bases do projeto VASelfCare, que utiliza Técnicas de Mudança Comportamental numa Intervenção com o apoio de um Agente Conversacional, este projeto tem como objetivo a definição de uma Plataforma de Desenvolvimento de um Agente Conversacional Proativo, que consiga num diálogo em linguagem natural ajudar o Utilizador a atingir determinados objetivos, assim como definir uma metodologia para o seu desenvolvimento.

A ideia principal do projeto é a de conseguir um nível de abstração que permita um sistema construído seguindo estes princípios a conseguir ser aplicado sobre qualquer tema. O Agente Conversacional deve ser proativo para conseguir guiar os diálogos com o Utilizador, aplicando Técnicas de Mudança Comportamental quando necessário para o apoiar a atingir os objetivos que o tema exige.

O desenvolvimento da Prova de Conceito decorreu com sucesso, conseguindo combinar um Serviço de Processamento de Linguagem Natural como o Dialogflow a bases de dados e conhecimento sobre os temas, conseguindo uma base adaptável a qualquer tema com representação em Ontologia.

Palavras-chave: Agentes Conversacionais; Técnicas de Mudança Comportamental; Aplicações mHealth; Sistemas Persuasivos; Proatividade.

Abstract

The development of conversational agents has witnessed immense improvement in recent years, being used in increasingly extensive and complex dialogues with the system users. Likewise, mobile health applications (mHealth) have found their space in the market, helping their users achieve the goals they are aiming towards. But despite this growth on both fronts, there is no unified structure of what is needed when building the architecture of a conversational agent working in this area.

Based on the foundations built on the VASelfCare project, which uses Behavioral Change Techniques in an intervention with the support of a Conversational Agent, this project aims to define a Development Platform for a Proactive Conversational Agent that manages to help its user achieve certain goals on a natural language dialogue, as well as define a Methodology for the Development.

The project's main idea is to achieve a level of abstraction that allows any system developed with these principles to discuss any topic that is required from them. The Conversational Agent must be proactive to be able to guide the dialogues with the user, applying Behavioral Change Techniques when necessary to support them in achieving the objectives that the topic requires.

The development of the Proof of Concept was successful, managing to combine a Natural Language Processing Service such as Dialogflow with databases and knowledge on the chosen topic, achieving a base adaptable to any topic with representation in Ontology.

Keywords: Conversational Agents; Behavior Change Techniques; mHealth Applications; Persuasive Systems; Proactivity.

Conteúdo

Lista de Figuras **xii**

Lista de Tabelas **xiii**

Glossário **xv**

1 Introdução **1**

1.1 Motivação 1
1.2 Objetivos 2
1.3 Contribuições 3
1.4 Estrutura do Documento 3

2 Conceitos e Trabalho Relacionado **5**

2.1 Intervenções e Conceitos de Mudança Comportamental 5
2.2 Estrutura de um Diálogo 7
2.3 Agentes Conversacionais 8
2.3.1 Classificação de Agentes Conversacionais 9
2.3.2 Proatividade de Agentes Conversacionais 9
2.3.3 Apresentação do Agente e Comunicação 10
2.4 Conceitos de Ontologia 11
2.4.1 *Reasoner* 12
2.4.2 Classes 12
2.4.3 Instâncias 12
2.4.4 Propriedades 12
2.4.5 Tipos de Dados 14
2.5 Trabalho Relacionado 14
2.5.1 VASelfcare 14
2.5.2 Operacionalização de Técnicas de Mudança Comportamental em Agentes Conversacionais 15
2.6 Sumário 16

3 Desenho do Sistema **19**
3.1 Arquitetura do Sistema . . . 19
3.1.1 Componentes da Arquitetura . . . 20
3.1.2 Fluxo de Comunicação no Sistema . . . 21
3.2 Requisitos/Ferramentas Utilizadas . . . 23
3.3 Metodologia de Desenvolvimento . . . 25
3.3.1 Núcleo do Sistema . . . 25
3.3.2 Serviço de Processamento de Linguagem Natural . . . 26
3.3.3 Componente de Conhecimento . . . 27
3.3.4 Desenho da Arquitetura . . . 28
3.4 Sumário . . . 28

4 Desenvolvimento da Prova de Conceito **31**
4.1 Arquitetura do Sistema da Prova de Conceito . . . 31
4.2 Fluxo de Comunicação da Prova de Conceito . . . 32
4.3 Ferramentas Utilizadas . . . 34
4.4 Criação do Ambiente . . . 34
4.5 Núcleo do Sistema . . . 36
4.5.1 Camada de Apresentação . . . 36
4.5.2 Camada de Processamento e Serviço de Webhook . . . 38
4.5.3 Base de Dados . . . 40
4.6 Dialogflow . . . 41
4.6.1 Bases de Funcionamento do Dialogflow . . . 41
4.6.2 Fluxo no Dialogflow . . . 44
4.7 Componente de Conhecimento . . . 47
4.7.1 Conhecimento de BCTs e Descrição da Ontologia . . . 47
4.7.2 Descrição da Intervenção . . . 52
4.8 Casos de Teste . . . 54
4.9 Sumário . . . 56

5 Conclusões e Trabalho Futuro **57**
5.1 Conclusões . . . 57
5.2 Trabalho Futuro . . . 58

Bibliografia **63**

A Diagramas de Comunicação do Sistema **65**

B Construção do ficheiro interventions.json **69**

Lista de Figuras

2.1 A Roda da Alteração Comportamental (BCW em inglês) apresentada por Susan Michie et al. [16] 6
2.2 Estrutura de um Diálogo de Intervenção de Timothy Bickmore 8
2.3 ECA com expressão facial desanimada quando recebe feedback negativo do utilizador, na versão 1 do VASelfCare 15
2.4 Estrutura da Intervenção Final na versão 2 do VASelfCare, extraído de [18] . . 16

3.1 Versão Inicial do Desenho de Arquitetura de Sistema 21
3.2 Fluxo de Comunicação do Sistema com base na Arquitetura Inicial Desenhada . 22
3.3 Versão 2 do Desenho da Arquitetura de Sistema 28

4.1 Arquitetura do Sistema 33
4.2 Fluxo de Comunicação do Sistema durante um Diálogo 33
4.3 Instruções no ficheiro READ.ME para inicializar o agente 35
4.4 ngrok inicializado no terminal 35
4.5 Webhook no Dialogflow com o link criado pelo ngrok 36
4.6 index.html 37
4.7 main.html durante utilização 37
4.8 Base de dados criada e os respetivos parâmetros 40
4.9 Intents criados no Dialogflow 42
4.10 O agente reage de forma diferente caso o utilizador tenha reações negativas . . 43
4.11 Exemplo de um dos Determinantes que o Dialogflow consegue identificar neste fluxo 44
4.12 Fluxo do Diálogo seguido pelo Dialogflow 45
4.13 Classes da Ontologia desenvolvida e respetivas subclasses 48
4.14 A comunicação esperada dentro do sistema, entre as diversas partes em funcionamento, no momento do log-in. O index.py funciona como central, sendo o OC.py a ligação à ontologia e o DBC.py a ligação à Base de Dados 55

A.1 Comunicação no momento de escolher na ontologia o Tópico a ser discutido. . 66
A.2 Comunicação no momento em que é necessário perguntar ao utilizador se cumpriu as tarefas impostas anteriormente. 66

A.3 Comunicação no momento em que é necessário atualizar a Base de Dados com as decisões tomadas para o diálogo seguinte. 67
A.4 Comunicação quando o sistema acaba de discutir um tópico com o utilizador mas ainda tem outros para discutir no mesmo diálogo. 67

B.1 No início do ficheiro é descrita a duração da Intervenção. Diz-se o número de fases existentes e a duração máxima da intervenção. Cada fase tem a duração mínima de 1 diálogo, mas pode prolongar-se. 69
B.2 Cada fase existente contém o tipo de progressão, a duração em número de diálogos que vai durar, os componentes da Ontologia que aborda, e as opções de progressão. Nesta imagem vemos o tipo de produção "*Tree*", que no final do tempo da duração vai decidir para que fase segue entre as opções, e "*Sequential*", que no final da duração simplesmente progride para a fase dada. 70
B.3 O tipo de progressão "*Loop*"tem a duração de 1 diálogo, e vai repetir-se até existir ordem em contrário. No caso do projeto atual, é a fase que repete quando todos os objetivos foram atingidos mas ainda não se cumpriu a duração máxima da intervenção, e vai fazer um acompanhamento leve dos componentes discutidos para garantir que o paciente continua a cumprir a sua rotina. 71
B.4 O tipo de progressão "*End*"marca o final da Intervenção, sendo assim o único tipo de progressão que não contém opções de progressão. 71
B.5 Durante a Intervenção pode ser necessário aumentar ou diminuir a duração que certos tópicos são discutidos. Aqui observamos um momento em que certos tópicos relacionados com Dieta são trabalhados com o paciente na fase "p03_7a". Caso o paciente atinja certos objetivos com sucesso este progride para a fase "p03_8a"onde abordará novos objetivos. Caso o paciente não os atinja, a Intervenção progrida para a fase "p03_7b"onde os vai abordar durante mais 5 diálogos antes de continuar o progresso. 72

Lista de Tabelas

4.1 Classes Existentes na Ontologia criada 50
4.2 Propriedades de Objetos na Ontologia criada 51
4.3 Tabela de Utilizadores dos Casos de Teste colocados na Base de Dados e progresso de cada um na Intervenção. A Fase a ser explorada, o número de diálogos que já existiram no total e o número de diálogos na Fase Atual 56
4.4 Tabela de Utilizadores dos Casos de Testecolocados na Base de Dados e progresso de cada um na Intervenção. Objetivos estão aqui representados pelo tuplo (Component, Topic, Goal) 56

Glossário

Assertion Afirmações sobre as Instâncias, normalmente na forma de propriedades definidas.. 13

BCT Behavior Change Technique. 5

BCW Behavior Change Wheel. 6

Class Representam o tipo de objetos que a Ontologia tem.. 12

COM-B Capability, Opportunity, Motivation - Behavior. 6

Entity Elemento que representa um conceito. No contexto deste projeto, temos Determinantes como motivação, hábitos, apetite, entre outros.. 13

Instance Elementos criados dentro da ontologia, de uma determinada classe e com as propriedades que essa classe exige.. 12

Intent No contexto de DialogFlow, é a intenção que o utilizador demonstra com o input que insere.. 25

Property Propriedades associadas a objetos. Podem ser características que estes objetos têm, como dados e anotações, ou ligações a outros objetos.. 12

Reasoner Ferramenta que permite à ontologia deduzir determinados aspetos em relacão aos seus elementos.. 12

Route Ligação entre Nodes, define como o Diálogo prossegue dependendo das tomadas de decisão possíveis.. 38

Webhook Computação externa que é chamada quando algo é ativado durante o diálogo.. 35

Capítulo 1

Introdução

1.1 Motivação

O desenvolvimento de agentes conversacionais não é recente, com o primeiro agente conversacional ELIZA a ser criado em 1966 [23]. Desde então, diversas aplicações utilizam agentes conversacionais como meio de comunicação entre homem e máquina, conseguindo até conduzir diálogos em linguagem natural, como será explorado ao longo da dissertação.

Estes diálogos, apesar de úteis para cumprir determinados objetivos, costumam seguir uma interação muito unidirecional: o utilizador realiza uma questão ou faz um pedido à máquina, e esta responde com a resposta ou realiza a ação pedida. Apesar da clara utilidade que este tipo de agentes conversacionais possui, o potencial acaba por ficar limitado aos objetivos que o utilizador quer atingir num espetro mais limitado.

Com o crescimento de Agentes Conversacionais, foram também exploradas formas de melhorar a interação com o utilizador [21]. Um exemplo é o desenvolvimento de Agentes Conversacionais Corporizados (em inglês, *Embodied Conversational Agents*), que consiste na utilização de uma representação visual para tornar a interação mais humana. Outro exemplo é mesmo na forma de comunicar com o Agente: se quando o ELIZA foi criado apenas era possível comunicar por texto, e o Processamento de Linguagem Natural (ou PLN) era mais básico, hoje em dia é possível comunicar até utilizando a voz e o PLN atingiu um nível muito elevado recentemente [14].

Num passado mais recente existiu também o crescimento de aplicações *mobile health* (ou *mHealth*) [22]. Estas aplicações, apesar de não substituírem profissionais de saúde, acompanham utentes a adaptarem rotinas mais saudáveis ao seu dia a dia, e permitem ao utilizador monitorizar o seu próprio comportamento de forma eficiente [20]. O acompanhamento mais regular conseguido com este tipo de aplicações leva a uma maior percentagem de sucesso nos objetivos definidos.

Aplicações *mHealth*, no entanto, têm uma limitação: a sua utilização e sucesso dependem da motivação do utilizador [2]. Enquanto um paciente que vai ao médico diariamente comunica todos os seus sucessos e fracassos e espera que o médico lhe diga como deve proceder, adap-

tando o plano ou motivando-o, uma aplicação mHealth pode apenas adaptar os objetivos, sem nunca ter a iniciativa [10]. A iniciativa está sempre do lado do Utilizador, o que a enquadra no mesmo quadro da maioria dos agentes conversacionais discutidos.

No âmbito do projeto VASelfCare [6], foi desenvolvida uma forma de utilizar uma Ontologia para aplicar Técnicas de Mudança Comportamental (do inglês *Behavior Change Techniques*, ou BCTs) em Intervenções aplicadas ao utilizador. Ou seja, utilizar um módulo de conhecimento que apoie o sistema na tomada de decisões durante um diálogo, aproveitando as capacidades de serviços de PLN.

O progresso obtido nestas diferentes áreas ao longo permite a exploração de diversas soluções capazes de obter resultados positivos em áreas diversas.

1.2 Objetivos

Com base nas motivações descritas acima, é possível entender o potencial existente em todas estas tecnologias de forma isolada. É também possível encontrar aplicações que já juntam duas ou mais tecnologias para atingir o resultado específico que esperam no âmbito de um programa específico. No entanto, não há nenhuma forma unificada de desenvolver um sistema com estas partes a funcionar em simultâneo e em sintonia.

O principal objetivo deste projeto é o de aproveitar como base o trabalho desenvolvido no âmbito do projeto VASelfCare e desenvolver uma Plataforma para o Desenvolvimento de Agentes Conversacionais Proativos, acompanhado de uma Metodologia para o Desenvolvimento da dita Plataforma.

O VASelfCare é uma aplicação que aplica uma intervenção a utilizadores para tentar melhorar o seu comportamento diário com base em três componentes essenciais na gestão do tratamento da Diabetes Tipo 2. Estes componentes são adesão à medicação, prática de atividade física e adoção de hábitos alimentares mais saudáveis.

Na versão mais recente do VASelfCare [6] era utilizado o Dialogflow para tratar do Processamento de Linguagem Natural, e existia uma base de conhecimento sobre BCTs para se aplicar uma intervenção base ao utilizador. Apesar de ser um desenvolvimento para a primeira versão do VASelfCare, onde a comunicação era por botões, esta versão ainda tem alguns pontos que podem ser tratados de forma mais isolada.

Assim, utilizando o VASelfCare e os seus sucessos como base, o sistema será explorado com o objetivo de criar uma base que possa ser facilmente utilizada para aplicar BCTs em temas diferentes dependendo da ontologia colocada, incluindo uma camada de abstração no sistema desenvolvido. Isto implica um estudo da arquitetura dos diversos sistemas desenvolvidos até este ponto, a compreensão das ferramentas utilizadas e compreender como estas podem ser aproveitadas para atingir o potencial que têm individualmente quando a trabalhar em conjunto.

1.3 Contribuições

Ao longo deste projeto foi desenvolvido um Agente Conversacional Proativo, capaz de guiar um diálogo com objetivos de alteração comportamental bem definidos, e com a capacidade de aceder a conhecimento tanto sobre como aplicar os BCTs como com a informação sobre a Intervenção e a forma como deve ser aplicada.

Numa primeira fase foi desenvolvido o Desenho da Arquitetura do Sistema, explorando as funcionalidades que devem ser procuradas para se escolher a ferramenta a utilizar. Em conjunto com o desenho e compreensão da arquitetura do sistema, foi explorada a Metodologia para o Desenvolvimento de um Sistema destes termos.

Por fim, para conseguir iterar sobre o Desenho e Metodologia da Arquitetura do Agente Conversacional Proativo foi desenvolvida uma Prova de Conceito. Nesta é possível ter diversos diálogos com o Agente, onde este se adapta às respostas que o utilizador devolve para fazer um acompanhamento mais pessoal em cada diálogo, e ainda adapta os objetivos entre diálogos com o apoio do Componente de Conhecimento com base no *feedback* do Utilizador.

Para desenvolver esta prova de conceito foram exploradas diversas versões anteriores do VASelfCare, em especial a de 2019, que apesar de ter a interação com o agente limitada a botões tinha uma explicação mais detalhada da Intervenção a longo prazo, dos diversos objetivos que teriam de ser abordados e do tempo utilizado em cada parte da Intervenção, e a de 2021, que explora em detalhe a utilização de ontologia para construir uma base de conhecimento.

1.4 Estrutura do Documento

Este documento está dividido em cinco capítulos, cada um deles subdivididos em secções para tentar facilitar a compreensão do trabalho apresentado:

- Capítulo 2 - Conceitos e Trabalho Relacionado: Neste capítulo são apresentados conceitos importantes para a compreensão do projeto. Temas essenciais como Intervenções, Diálogos, Agentes Conversacionais, Conceitos de Mudança Comportamental e Conceitos de Ontologia. Para além disto vão também ser apresentados Trabalhos Relacionados ao que vai ser desenvolvido ao longo do projeto, para compreender a influência que têm sobre o trabalho, o que vai ser aproveitado e o que vai ser melhorado.

- Capítulo 3 - Desenho do Sistema: Neste Capítulo vai ser apresentada a Arquitetura do Sistema Desenhado. O objetivo é o de mostrar como o sistema se divide, os requisitos de cada componente para o seu desenvolvimento, e a Metodologia de Desenvolvimento, bem como o Fluxo de Comunicação dentro do Sistema.

- Capítulo 4 - Desenvolvimento da Prova de Conceito: Neste capítulo é descrito o sistema desenvolvido para comprovar o desenho realizado. Serão apresentados com detalhe todos os documentos desenvolvidos para cada componente, as ferramentas utilizadas, os

pontos de comunicação do sistema e os diversos Casos de Teste possíveis com o sistema desenvolvido.

- Capítulo 5 - Conclusões e Trabalho Futuro: Neste capítulo serão feitas considerações final sobre o trabalho realizado, resumindo um pouco o trabalho desenvolvido e a forma como correu. Serão também exploradas algumas perspetivas de trabalho futuro que se possam desenvolver a partir deste projeto.

Capítulo 2

Conceitos e Trabalho Relacionado

Neste capítulo são apresentados os principais conceitos a ter em conta para o desenvolvimento de um Agente Conversacional Proativo. Estes conceitos abordarão aspetos técnicos relevantes para a compreensão do trabalho desenvolvido e contextualização de trabalho antecessor ao aqui apresentado. Numa primeira fase serão apresentados conceitos relacionados com as Intervenções aplicadas a pacientes e com Agentes Conversacionais, e numa segunda fase serão discutidos Trabalhos Relacionados, com especial foco em trabalhos que utilizam Agentes Conversacionais em ambientes semelhantes aos que se querem alcançar.

2.1 Intervenções e Conceitos de Mudança Comportamental

Quando um indivíduo é diagnosticado com alguma condição que assim o exija, este indivíduo poderá ter de alterar determinadas rotinas que tenha para acomodar novos cuidados. Quanto maior a lista de novos hábitos que tem de adotar, maior a dificuldade que o indivíduo pode encontrar não só a começar a praticá-los, como também a mantê-los [24].

Para conseguir atingir estes novos objetivos a que se propõe, é aconselhado ao indivíduo que, durante um determinado período, encontre acompanhamento profissional. O conjunto dos momentos de acompanhamento será, de aqui em diante, referido como Intervenção.

Uma Intervenção [4] é composta por interações entre o paciente e o profissional de saúde, e será onde o profissional vai abordar diversos temas de interesse para o paciente. Cada uma destas interações será referida como um Diálogo. Estes temas são definidos pelas rotinas que o paciente tem de adotar, e o objetivo é que no final da Intervenção o paciente tenha já adquirido a capacidade de realizar estes novos hábitos sem que seja necessário acompanhá-lo.

O objetivo do profissional durante a Intervenção é que as novas rotinas sejam introduzidas ao ritmo certo ao paciente para que este as consiga adotar a todas sem se sentir bombardeado com ordens, e conseguir ajudar o paciente a ultrapassar possíveis dificuldades que tenha a adotá-las.

Para ajudar o paciente a atingir estes objetivos, os profissionais de saúde utilizam estratégias para conseguir levar o paciente no rumo certo. Estas estratégias são introduzidas no diálogo e são denominadas *Behavior Change Techniques* [17], de aqui em diante referidas como BCT.

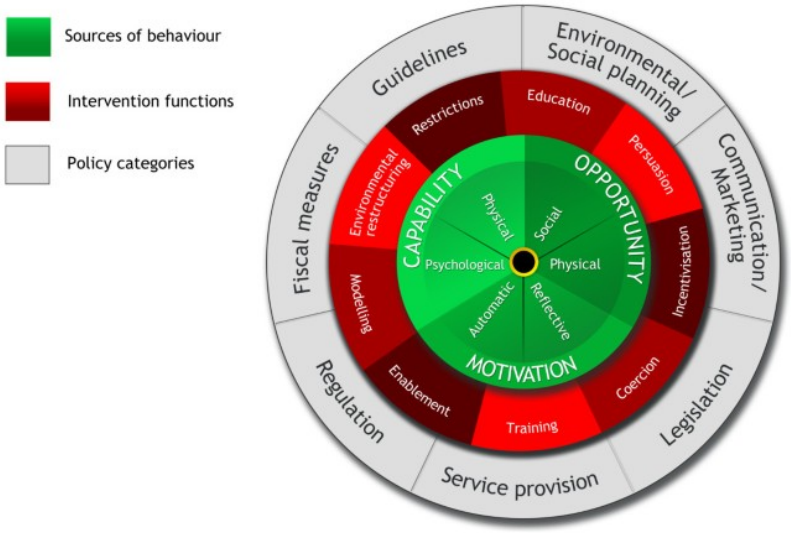


Figura 2.1: A Roda da Alteração Comportamental (BCW em inglês) apresentada por Susan Michie et al. [16]

BCTs [17] são, como o nome indica, técnicas que levam um indivíduo a alterar o seu comportamento. Cada uma destas técnicas pode ser utilizada de forma isolada ou combinada com outras, e serão aplicadas durante os Diálogos. Para desenhar o diálogo e escolher qual ou quais das BCTs será utilizada em cada momento existem diversas alternativas. Uma das alternativas mais utilizadas é um *framework* denominado *Behavior Change Wheel*, ou BCW [16].

A BCW é uma roda dividida em 3 camadas, e cada camada tem um foco diferente na forma como a técnica é escolhida:

- Na camada interior, a roda baseia-se na hipótese de que a capacidade, a oportunidade e a motivação do indivíduo são 3 condições essenciais para o comportamento. Este é o chamado modelo COM-B (*Capability, Opportunity, Motivation - Behavior*).

- Na camada intermédia temos possíveis funções para a intervenção, os porquês do objetivo determinado para aplicar a BCT, e podem ser aplicadas a uma ou mais das fontes de comportamento.

- E na camada exterior temos 7 tipos de políticas que podem ser utilizadas para que o utilizador siga a função de intervenção.

Assim, para desenvolver uma intervenção de alteração comportamental, é necessário começar o diálogo por compreender o porquê do comportamento do utilizador. Ao entender o comportamento é necessário identificar o que há de errado nesse comportamento e o que tem de ser corrigido. Depois, há que identificar as funções da Intervenção e as políticas definidas para esta Intervenção. Por fim, será possível identificar a BCT a utilizar no Diálogo e como o paciente será abordado quanto ao presente objetivo.

Compreender como a BCW identifica as BCTs necessárias e as aplica ao utilizador é extremamente útil no âmbito deste projeto, onde as BCTs são as estratégias que o agente adota e que o guiam durante o Diálogo com o utilizador para tentar induzir alteração comportamental.

2.2 Estrutura de um Diálogo

A interação do agente com o utilizador, para ser eficiente a transmitir a mensagem e tornar o diálogo natural, terá de estar tão próxima de uma interação entre dois seres humanos quanto possível. Para alcançar um Diálogo de Intervenção natural é necessário, assim, compreender de que forma é que um profissional de saúde vai abordar os temas. Para além disso, é necessário compreender como é que a interação vai ser seguida para saber onde devem ser integrados os BCTs selecionados para o Diálogo.

Timothy Bickmore et al. [9] dividiram um curto diálogo de aconselhamento em 8 partes:

1. Abertura da Conversa - Saudação ao Utilizador
2. Ritual Social - Diálogo social, por exemplo, questionando o utilizador sobre como este se sente
3. Revisão de Objetivos - Rever as tarefas e objetivos definidos anteriormente com o utilizador, perguntando-lhe o que foi ou não atingido
4. Avaliação - Face às respostas dadas pelo utilizador é avaliado o estado do utilizador durante a intervenção, possíveis alterações no seu comportamento e de que forma estas ações estão corretas ou erradas
5. Aconselhamento - Após determinar as razões das ações do utilizador, o diálogo é dirigido consoante o que estas ações são, se são ou não corretas e de que forma o utilizador pode continuar o seu progresso ou superar possíveis dificuldades
6. Definição de Objetivos - Definir tarefas e objetivos para o utilizador tentar atingir até ao seguinte encontro
7. Pré-encerramento - Preparação para encerrar o diálogo
8. Encerramento - Despedidas

Quando dividimos o Diálogo nestas 8 fases conseguimos compreender melhor como os tópicos são abordados e em que pontos o agente compreende que deve aplicar as BCTs. Vemos assim que podemos dividir as 8 fases em 3 conjuntos:

- Início do Diálogo: aborda as duas primeiras fases, tendo como propósito pôr o utilizador à vontade com o agente e compreender o estado anímico do utilizador

Figura 2.2: Estrutura de um Diálogo de Intervenção de Timothy Bickmore

- Intervenção: aborda as 4 fases entre a Revisão de Objetivos e a Definição de Objetivos, inclusive, e pretende falar dos tópicos a abordar e compreender o progresso do utilizador. Estas 4 fases serão repetidas caso seja necessário abordar mais do que um tópico durante o Diálogo

- Fim do Diálogo: contém as últimas duas fases, e vai despedir-se do utilizador

A divisão de diálogo de Bickmore, apesar de não ser a única existente, é vista como norma nesta matéria, e mostra-se útil não só a compreender como construir um melhor diálogo natural seguido pelo sistema como também a compreender o trabalho prévio desenvolvido no VASelf-Care e a forma como o diálogo era implementado. Ter em conta a maneira como o Diálogo está dividido ajuda também a definir melhor os pontos e os *timings* de comunicação entre os componentes do sistema.

2.3 Agentes Conversacionais

Um agente conversacional consiste num sistema que consiga comunicar e conduzir diálogos com um utilizador [15].

Este tipo de sistemas está dependente de certos mecanismos para o seu melhor funcionamento: numa primeira instância, tem de conseguir receber e processar o *input* do utilizador, e a seguir tem de conseguir deduzir de que forma deve prosseguir o diálogo, devolvendo ao utilizador uma resposta que faça sentido depois do *input* dado previamente pelo utilizador.

Com as inúmeras diferentes utilizações que este tipo de agentes podem ter, desde aplicações tão complexas como diálogo por voz com Assistentes Pessoais até pequenas caixas de texto em *websites* de comércio online, torna-se necessário compreender em que contornos eles se podem mostrar.

2.3.1 Classificação de Agentes Conversacionais

Agentes Conversacionais podem ser divididos de diversos modos, mas na notação mais utilizada, criada por Daniel Jurafsky [13], estão divididos em duas grandes classes:

- Agentes de diálogo orientado a tarefas (*Task-oriented Dialog Agents*)
- Chatbots

Estas duas classes têm imenso do que as define em comum, sendo distinguidas apenas pela forma como o agente interage com o utilizador humano.

Task-oriented Dialog Agents querem apenas ajudar o utilizador a completar certas tarefas. Assim, as respostas tendem a ser mais diretas, e estão completamente focadas no objetivo. Por outro lado, Chatbots tentam simular interações entre dois indivíduos humanos em diálogos mais extensos.

Do ponto de vista prático, ambos os agentes conversacionais conseguem chegar ao mesmo ponto e realizar a mesma tarefa. No entanto, um Chatbot tem em atenção a experiência do utilizador até alcançar os objetivos a que se propõe. Este cuidado torna esta classe de agentes a ideal para encaixar num ambiente mais específico, em situações de entretenimento, treino ou educação, onde é preciso que o utilizador não sinta o pragmatismo de uma máquina no momento da comunicação e definição de objetivos a seguir.

É desta forma que diferentes aplicações na área da saúde conseguem utilizar agentes conversacionais para seu proveito. É necessário, no entanto, ter um nível de compreensão detalhada da área da saúde em que o agente vai estar incorporado para que este consiga atingir os resultados desejados.

2.3.2 Proatividade de Agentes Conversacionais

A classificação de Jurafsky [13] divide agentes conversacionais em duas classes, mas as duas classes têm um ponto muito importante em comum: todos os agentes ajudam o utilizador a atingir um objetivo definido pelo utilizador. Ou seja, na maioria das situações descritas o utilizador pede ajuda ao agente na resolução de um problema ou no cumprimento de um objetivo, e este vai informar o utilizador sobre onde encontrar a resolução. Isto faz com que quem esteja a guiar os diálogos seja o utilizador, que propõe os problemas ao agente e recebe a resolução destes ou pelo menos como os resolver.

No entanto, no âmbito das Intervenções previamente exploradas, os papéis estarão revertidos. Apesar de o agente estar presente para acompanhar e apoiar o utilizador a atingir certos objetivos, estes objetivos não são definidos pelo utilizador. Isto cria a necessidade de o agente se poder mostrar proativo durante algumas das fases do Diálogo.

A proatividade em agentes conversacionais neste tipo de situações específicas vem em grande parte da aplicação de BCTs. Isto surge porque sempre que um objetivo não está a

ser cumprido por parte do utilizador, a forma que o agente tem de apoiar o utilizador e de garantir que este pode evoluir e começar a adotar novas rotinas é compreendendo o raciocínio do utilizador e as razões para os objetivos não estarem a ser cumpridos. Após compreender o utilizador e as suas motivações, o agente vai aplicar as BCTs para que o utilizador seja guiado na direção certa. É neste ponto que vemos a mudança de paradigma, onde o utilizador não sente que o agente está a resolver o problema por ele, mas que ele próprio tem de resolver o problema dentro dos novos parâmetros definidos pelo agente.

De um ponto de vista mais abstrato, sempre que o sistema guia o tema da conversa conseguimos observar algum tipo de proatividade. No entanto, a única classe de agentes conversacionais que pode ser dada a proatividade de forma natural serão os *Chatbots*, uma vez que *Task-oriented Dialog Agents* estão apenas focados em descobrir qual o problema encontrado pelo utilizador e qual a solução para esse problema, não tendo nenhum interesse em desviar do ponto principal estabelecido. Como *Chatbots* tentam simular uma conversa normal entre dois indivíduos, a aplicação de proatividade por parte do agente pode passar de forma natural para o indivíduo quando inserido num ambiente apropriado.

No entanto, a proatividade mostrada pelo agente estará sempre presa ao que está a ser discutido com o utilizador. Isto porque em qualquer situação em que o agente assuma a iniciativa num diálogo esta vem sempre dependente do que tenha sido o *input* do utilizador, e o agente vai ditar como o diálogo é guiado sempre com base no input do utilizador. A proatividade não surge tanto da origem do novo assunto, que vem sempre ditada pelo caminho que o utilizador está a percorrer, mas sim pela parte do assunto que será discutida de cada vez.

2.3.3 Apresentação do Agente e Comunicação

O modo como os agentes se apresentam e como a comunicação com o agente flui está intrinsecamente ligada. Numa primeira fase, devido aos objetivos que o agente ajuda o utilizador a atingir, e numa segunda fase, devido à complexidade da comunicação.

A apresentação dos agentes pode ser tão simples quanto uma caixa de texto onde é possível ver a comunicação como se fosse um *chat* de texto. Esta é, aliás, a apresentação escolhida pelos *Task-oriented Dialog Agents* encontradas, por exemplo, em *websites* de comércio *online*. Esta escolha pode ser tomada devido a diversas razões, entre elas

- A simplicidade da comunicação necessária entre o utilizador e o agente, onde o utilizador terá certos objetivos definidos e quer uma solução rápida para encontrar aquilo que procura e assim consegue receber uma resposta curta que vai direta ao assunto que apresentou;

- A quantidade de temas que podem ser explorados num diálogo deste género é mais curta, o que implica que a comunicação pode também ser mais limitada e não necessita de manter a atenção do utilizador durante grandes períodos de texto;

- A comunicação da parte do utilizador ser mais direta, uma vez que simplesmente apresenta o seu problema e o agente tem a capacidade para compreender de uma pequena explicação aquilo que tem de devolver ao utilizador.

No entanto, este tipo de comunicação pode tornar a aplicação inutilizável. Veja-se o exemplo de assistentes pessoais como a Siri [7], que tem como vantagem a comunicação por voz permitir ao utilizador realizar pequenas questões ao agente sem precisar de dedicar atenção visual e comunicando de forma mais natural, não tendo de escrever. Este tipo de comunicação vem com mais despesas do ponto de vista do sistema, que necessita de um mecanismo de compreensão do áudio que lhe é passado pelo utilizador e um mecanismo para transmitir a solução ao utilizador também por áudio, mas permite realizar ações curtas de forma mais confortável quando comparado com a necessidade de ter de escrever e ler de um *chat* de texto.

No entanto, estes sistemas não estão construídos com um diálogo mais longo como objetivo principal, e é nesta categoria que encontramos os *Chatbots*. Nos dois exemplos explorados anteriormente, vemos que os agentes ajudam o utilizador a atingir os seus objetivos, não fossem eles *Task-oriented Dialog Agents*, mas como não tentam simular um diálogo entre dois seres humanos, não procuram nenhum tipo de forma de tornar o diálogo mais natural para o utilizador. Tanto numa caixa de texto como numa voz que transmite informações de forma mais ou menos direta existe uma certa distância entre o sistema e o utilizador que por vezes não é completamente quebrada apenas pela forma como o agente transmite a informação.

É neste ponto que certas aplicações como o VASelfCare, que será introduzido à frente, utiliza *Embodied Conversational Agents* (de aqui em diante tratados por ECA). ECAs são uma representação bi ou tridimensional em que o agente é integrado, assumindo essa forma durante as interações. De um ponto de vista mais prático, é um avatar que representa o agente, dando-lhe uma forma física que tenta abstrair o utilizador de que quem lhe está a passar a informação é um agente de inteligência artificial.

Este agente pode depois estar munido de qualquer uma das formas tratadas anteriormente, podendo aparecer o texto junto ao ECA ou ouvindo-se uma voz e animando-se a cara do ECA.

A forma como o utilizador comunica com o agente também pode alterar de sistema para sistema, dependendo dos recursos disponíveis e dos objetivos que têm em mãos. Algumas opções que podem ser encontradas são:

- Opções dadas ao utilizador, de onde este escolhe o que comunicar ao agente;
- Input de texto aberto ao utilizador, onde este pode escrever o que quiser;
- Input de voz aberto ao utilizador, onde este dialoga com o sistema.

2.4 Conceitos de Ontologia

Ontologia é a representação de categorias, propriedades e relações entre conceitos e entidades, tal como a designação e definição de cada um destes pontos. Ou seja, ontologia é onde se

definem os conceitos de um certo campo de conhecimento de modo formal.

Do ponto de vista informático, existem diversas formas de criar esta representação de modo a que outras tecnologias lhes possam aceder e utilizar o seu conhecimento no que seja necessário.

É necessário, então, perceber quais os tipos de estruturas que podemos encontrar em ontologias e de que maneiras as podemos descrever e relacionar. Para isto vamos utilizar a notação encontrada no *Protégé* [1], uma vez que foi a ferramenta utilizada para construção do Caso de Teste apresentado posteriormente.

2.4.1 *Reasoner*

O Reasoner é uma ferramenta que permite à ontologia deduzir determinados aspetos em relação aos seus elementos. Isto ficará melhor definido compreendendo o tipo de dados presentes numa ontologia, mas o objetivo do *Reasoner* é o de ver o tipo de propriedades de cada elemento da ontologia e compreender como pode categorizar estes elementos consoante as regras e a base sobre as quais o conhecimento existe. Por exemplo, se temos uma ontologia que define ”Todas as letras pertencem ao alfabeto”e ”A é uma letra”, o *Reasoner* vai inferir automaticamente que ”A pertence ao alfabeto”.

2.4.2 Classes

As classes (originalmente, Class) representam o tipo de objetos que a Ontologia tem. Estas classes podem ser descritas com propriedades específicas, e podem ter subclasses. Estas subclasses terão obrigatoriamente as mesmas propriedades que a super-classe principal, mas podem depois variar com a presença de propriedades únicas à subclasse ou ter certos valores específicos em determinadas propriedades.

No início de cada ontologia existe a classe *Thing*. Todos os elementos criados pertencerão a esta classe, e todas as classes estarão inseridas como subclasse desta.

2.4.3 Instâncias

Os elementos criados na ontologia são denominados de Instâncias ou Indivíduos (originalmente, Instance. Cada instância terá propriedades e pode estar definida dentro de uma classe, precisando para isso de respeitar as propriedades que esta necessite de ter definidas.

2.4.4 Propriedades

As propriedades que encontramos em ontologias (originalmente, Property podem ser de dois tipos:

- Anotações

[1] https://protege.stanford.edu/

- Dados
- Objetos

Propriedades de Anotações (*Annotation Properties* no *Protégé*) e Propriedades de Dados (*Data Properties* no *Protégé*) ou Propriedades de Objetos (*Object Property* no *Protégé*) podem muitas vezes confundir-se, sendo muito parecidas do ponto de vista do utilizador. A grande diferença que vemos entre elas é que Anotações são ignoradas pelo reasoner independentemente de possíveis erros ou dados que estejam presentes. Assim sendo, quando é necessário que uma propriedade seja utilizada para atribuir instâncias a classes, este não poderá ser uma Anotação. Em sentido oposto, quando um certo dado não tem de ser acedido pode poupar-se no custo de processamento.

Assim sendo, Propriedades de Anotação têm asserções (originalmente Assertion) para determinadas entidades. Estas asserções visam ligar a Entity a algum tipo de valor, que pode ser tanto um literal (string, número ou data entre outro) ou outra entidade (como outra classe). Propriedades de Anotação podem ser organizadas hierarquicamente.

Propriedades de Dados têm também o objetivo de ligar a entidade a algum tipo de valor, sendo este valor sempre um literal. Estes valores podem afetar o reasoning, sendo utilizados para definir classes com intervalos de valores, por exemplo.

Propriedades de Objetos pretendem ligar a entidade a outra entidade. Este tipo de propriedades é o que se pretende utilizar para criar relações entre diferentes Instâncias. Estas Propriedades têm assim diferentes características que se devem compreender.

As características de Propriedades de Objetos que podem ser selecionadas são:

- Functional - Qualquer instância só pode ter um valor nesta propriedade
- Inverse Functional - Qualquer instância que seja selecionada para esta propriedade apenas pode ser instância escolhida uma vez
- Transitive - Se a instância x tiver esta propriedade em relação à instância y, que por sua vez tem esta relação à instância z, é possível deduzir que a instância x tem esta propriedade com a instância z (Se xRy e yRz, então xRz)
- Symmetric - A propriedade que é vista na Instância x em relação à Instância y pode ser também vista na instância y em relação à instância x (Se xRy, então yRx)
- Asymmetric - Se a propriedade é visível na Instância x em relação à Instância y então a Instância y não pode ter esta propriedade com a Instância x (Se xRy, então ¬ (yRx))
- Reflexive - Todas as Instâncias com esta propriedade estão relacionadas consigo mesmas (Para todo o x, xRx é verdadeiro)
- Irreflexive - Todas as Instâncias com esta propriedade não podem estar relacionadas consigo mesmas (Para todo o x, xRx é falso)

2.4.5 Tipos de Dados

Da mesma forma que existem dados gerais como String, valores numéricos, datas e semelhantes, é possível criar diferentes tipos de dados mais específicos para determinadas situações que sejam relevantes para o criador da Ontologia.

2.5 Trabalho Relacionado

O desenvolvimento deste projeto vem do seguimento de diversos outros projetos realizados no LASIGE. Estes projetos, por terem certas limitações nas ferramentas utilizadas ou por terem objetivos de desenvolvimento específicos numa certa área, não permitiam muitas vezes um nível maior de abstração que permitisse que aplicações com diferentes objetivos seguissem as mesmas normas para atingirem os resultados requeridos [6].

Outro problema existente é a falta de uma metodologia de desenvolvimento para este tipo de sistemas, o que torna necessário que qualquer sistema de agentes conversacionais que seja desenvolvido precise de compreender sistemas anteriores e deduzir o que pode ou não ser utilizado para o novo sistema.

Assim sendo há que compreender cada projeto realizado e perceber quais as limitações do desenho aplicado anteriormente e como é possível reduzi-las ou até eliminá-las.

2.5.1 VASelfcare

O projeto VASelfCare foi desenvolvido por parte de uma equipa de investigação do LASIGE. O objetivo principal deste Trabalho de Investigação era o desenvolvimento de um assistente pessoal que facilitasse o autocuidado de pessoas mais velhas com diabetes tipo 2.

A primeira versão do projeto [11] tinha como base a utilização de um ECA que dava informações ao Utilizador. A interação do Utilizador estava limitada a botões, e o Agente acabava por reagir à opção que o utilizador escolhia. O ponto mais forte desta versão era efetivamente a utilização de um ECA com expressões faciais para reagir à informação que o utilizador lhe passava. Quanto à interação, sendo realizada por botões, era possível criar um fluxo de diálogo pré-determinado e preparar os nós deste fluxo com toda a informação necessária para a aplicação.

Já as Intervenções estavam divididas em 3 grupos, por comportamento a ser abordado: Adesão à Medicação, Adesão a Atividade Física e Alimentação Saudável. Cada um dos componentes tinha a duração de 30 dias: 3 dias de Avaliação, em que se estuda o estado atual do utilizador, e o restante de Acompanhamento, que acompanha o progresso do Utilizador e tenta ajudá-lo a atingir os objetivos.

A segunda versão do projeto [18] trouxe consigo a ideia de começar a incorporar elementos de inteligência artificial para que a interação fosse mais realista e para apoiar no desenvolvimento. Com a tecnologia utilizada na primeira versão, entre 8 perfis de utilizador e terem de

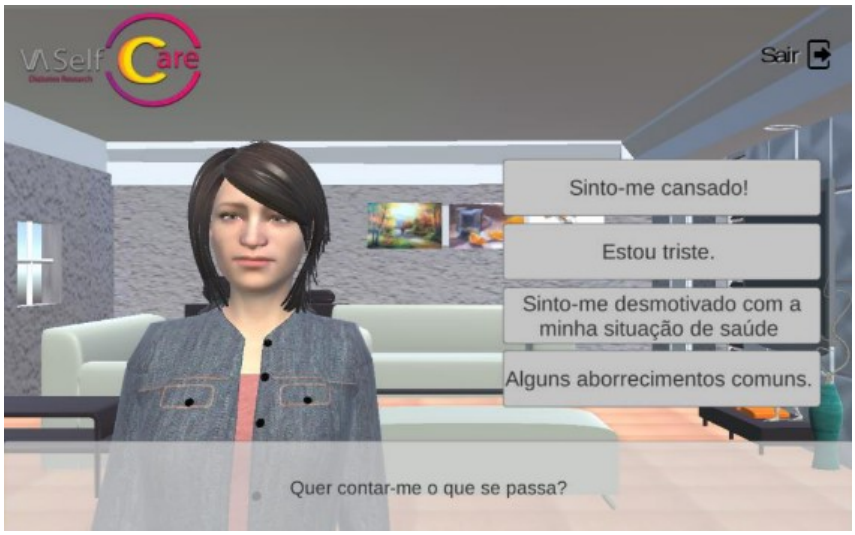


Figura 2.3: ECA com expressão facial desanimada quando recebe feedback negativo do utilizador, na versão 1 do VASelfCare

desenvolver 180 dias de diálogo, estariam a olhar para a construção de 1400 ficheiros de diálogo diferentes.

Este projeto conseguiu o desenvolvimento requerido através da alteração de como os diálogos estavam armazenados, alterando os diálogos para CLIPS, uma linguagem que permitia a utilização de inteligência artificial para determinar que diálogos mostrar a cada utilizador dependendo do seu perfil. No entanto, o projeto continuava preso à limitação que era comunicar com o utilizador através de botões.

Isto mudou com a terceira versão do VASelfCare [6], com a introdução de um Serviço de Processamento de Linguagem Natural que permite ao utilizador comunicar normalmente, por texto, com o Agente. Tendo esta versão do projeto um foco maior na forma como o diálogo decorria e querendo avaliar mais o diálogo em si do que o sucesso da interação com pacientes, o ECA foi abandonado e substituído por uma caixa de texto que permitem comunicar com o Agente ”por mensagem”.

No entanto, com a introdução de comunicação por Linguagem Natural também chega o desafio de o agente ter de saber como responder ao Utilizador, e o ponto principal do projeto desenvolvido foi a forma como as BCTs eram operacionalizadas.

2.5.2 Operacionalização de Técnicas de Mudança Comportamental em Agentes Conversacionais

Esta versão do VASelfCare [6] tem definida uma Ontologia de Conceitos de Mudança Comportamental. Começaram por identificar os conceitos necessários para a escolha e aplicação de BCTs em Utilizadores. O VASelfCare tem uma estrutura de diálogo igual à apresentada na secção 2.2, pelo que se tornou necessário compreender com especial atenção o segundo conjunto de fases, chamado Intervenção.

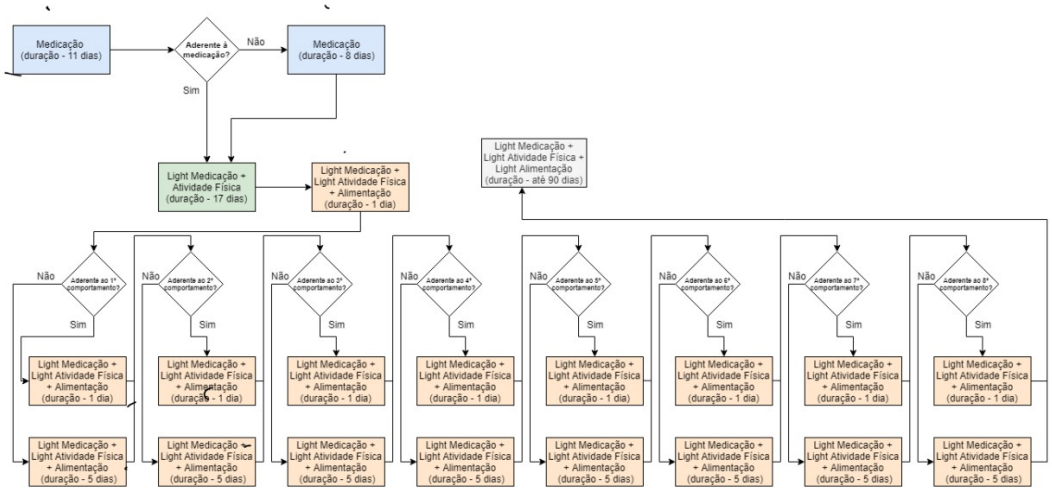


Figura 2.4: Estrutura da Intervenção Final na versão 2 do VASelfCare, extraído de [18]

Este conjunto começa com a Revisão de objetivos definidos em diálogos anteriores, Avaliação do progresso obtido, Aconselhamento relativamente ao tópico ou ao objetivo em questão, e a Definição de Objetivos para serem discutidos no futuro. É possível identificar conceitos como o Tópico Comportamental, o Objetivo Comportamental e os determinantes que levam o utilizador a não conseguir seguir os objetivos, estes conceitos são importantes para que seja possível decidir como o diálogo deve prosseguir.

Com os conceitos identificados, foram criadas classes onde colocar as entidades para o bom funcionamento do agente. A estrutura hierárquica contém 2 classes de maior importância: *Behavior Change Intervention* e *User*. A classe *Behavior Change Intervention* vai ter subclasses relacionadas com o diálogo e a intervenção, como BCTs, Tópicos de Discussão e Objetivos de Discussão. A classe *User* vai ter subclasses que descrevem o Utilizador, como a identificação se é Adulto ou Menor de Idade.

Estas classes estão depois ligadas através de Propriedades de Objetos que permitem ao Agente, com o apoio do Serviço de Processamento de Linguagem Natural, escolher qual a melhor ação para o utilizador. A existência de Propriedades de Dados e Anotações no utilizador será o que faz o Reasoner deduzir em que subclasses o Utilizador encaixa, existindo depois BCTs associadas a objetivos específicos, e adaptadas ao perfil identificado ao utilizador.

Quando o agente tem um conjunto de informação recebido do input do utilizador e não sabe como responder, são estas classes e propriedades que ajudam o Reasoner a ler a ontologia e a deduzir a forma como o Agente deve responder.

2.6 Sumário

Neste capítulo foram apresentados conceitos importantes para a compreensão do sistema construído. Apresentaram-se conceitos relacionados com Intervenções e Mudança Comportamental,

como Behavior Change Techniques e a Behavior Change Wheel. Foi explorada a estrutura de um Diálogo, apresentando-se as 8 fases que o formam e a forma como estas fases se agrupam entre si.

Foram também apresentados diversos tipos de Agentes Conversacionais, acompanhados das semelhanças e diferenças entre eles, para além de uma apresentação de Conceitos de Ontologia necessários para compreender o Componente de Conhecimento desenvolvido.

Por fim foi explorado algum Trabalho Relacionado, dando-se maior foco ao VASelfCare por terem um sistema construído com o mesmo objetivo que a Prova de Conceito foi desenvolvida: apoiar Pacientes com Diabetes Tipo 2 a adotar novas rotinas para o seu dia a dia. Foi também explorado como se pode operacionalizar BCTs em Agentes Conversacionais, com a utilização de Linguagem Natural e de uma Ontologia que ajude o Agente a tomar decisões, sendo de extrema importância a identificação de conceitos-chave para criar a Ontologia.

Capítulo 3

Desenho do Sistema

Neste capítulo é descrito o sistema que deve ser construído para desenvolver um agente conversacional proativo. Para um sistema deste tipo ser construído é necessário ter em conta os componentes necessários para o correto funcionamento do sistema e de que forma estes comunicam entre si (Secção 3.1), os requisitos que a máquina que vai colocar o agente em funcionamento necessita de cumprir (Secção 3.2) e a Metodologia para desenvolver cada um dos componentes do sistema (Secção 3.3).

3.1 Arquitetura do Sistema

Como é identificável ao longo de Projetos Anteriores e do Trabalho Relacionado, existe uma série de sistemas que devem coexistir em funcionamento. Estes sistemas, cada um com o seu objetivo, comunicam para garantir o correto funcionamento do agente.

A forma mais acessível de compreender a arquitetura de um agente conversacional é a de compreender como uma arquitetura em camadas. No entanto, algumas camadas poderão ter mais do que um sistema a trabalharem ao mesmo nível. Assim sendo, a arquitetura necessita de

- Camada de Apresentação
- Serviço de Processamento de Linguagem Natural (PLN)
- Núcleo do Sistema
- Componente de Conhecimento

Durante este capítulo vamos explorar a forma como estes sistemas devem ser desenhados, tanto do ponto de requerimentos necessários como do ponto dos objetivos que cada parte terá. Numa primeira fase, vamos definir quais as camadas que devem existir no sistema e as suas responsabilidades. Depois, vamos compreender como deve funcionar a comunicação do sistema, e em que pontos deve existir comunicação entre as diversas camadas.

3.1.1 Componentes da Arquitetura

Como descrito anteriormente, existem 4 camadas que devem estar presentes no desenho do sistema.

A primeira é a Camada de Apresentação. É aqui que o Utilizador poderá interagir diretamente com o Agente Conversacional. Esta camada tem de estar preparada para duas funcionalidades essenciais: receber o *input* do utilizador e transmitir o *output* do Agente.

O desenho desta camada depende do tipo de comunicação que se decida que o Agente terá, e acaba por se tornar na forma como o Utilizador perceciona o Agente. Como especificado no Capítulo 2, um Agente Conversacional pode dar a cara das mais distintas formas e comunicar entre caixas de texto aberto preenchidas livremente pelo Utilizador, *input* de voz aberto ou opções de comunicação dadas ao Utilizador. Se o Agente é representado por um ECA, é nesta camada que este estará presente.

E é aqui que o utilizador vai também receber o que o sistema decide que será transmitido pelo Agente Conversacional. Nesse caso, é preciso também que a camada esteja preparada para transmitir a mensagem da forma desejada. Se o Agente comunicar por áudio tem de ser capaz de comunicar com o Utilizador no momento em que a mensagem estiver pronta, ou mostrar de forma inequívoca de que forma é possível aceder à mensagem. Se o Agente comunicar através de uma mensagem de texto, é necessário que esta fique visível no monitor. E no caso de existir a representação com um ECA, é necessário decidir se este apenas se mostra como uma representação estática ou se tem de ser visível movimento a acompanhar a entrega da mensagem.

Toda esta parte é da responsabilidade da Camada de Apresentação e o seu peso no sistema final apenas depende da complexidade da comunicação. No entanto, nenhum tipo de processamento será da responsabilidade desta camada.

Para que toda a comunicação seja processada de forma completa são necessárias as restantes 3 camadas no início do Capítulo. Estas camadas estão em comunicação quase constante e estão completamente dependentes umas das outras para que o Sistema funcione corretamente, mas têm também elas papéis bem definidos.

A principal responsável pela comunicação entre as três é o Núcleo do Sistema. Esta camada tem o nome que tem por funcionar um pouco como o coração do sistema. Vai receber o *input* recebido pela Camada de Apresentação e quer chegar à conclusão sobre qual será a resposta correta a dar ao *input* recebido. Assim que souber o que deve ser respondido deve comunicar o *output* à Camada de Apresentação para que esta o passe ao Utilizador. É neste processo que a Núcleo do Sistema precisa de comunicar com as restantes camadas, mostrando-se como o ponto de ligação entre as diversas camadas e principal responsável pelo bom funcionamento do sistema.

Assim sendo, é necessário que se compreenda numa primeira instância o que o utilizador disse, se vem alinhado com o diálogo até ao momento e de que forma. Sendo necessário um tipo de processamento mais profundo para compreender dependendo do tipo de *input* entregue,

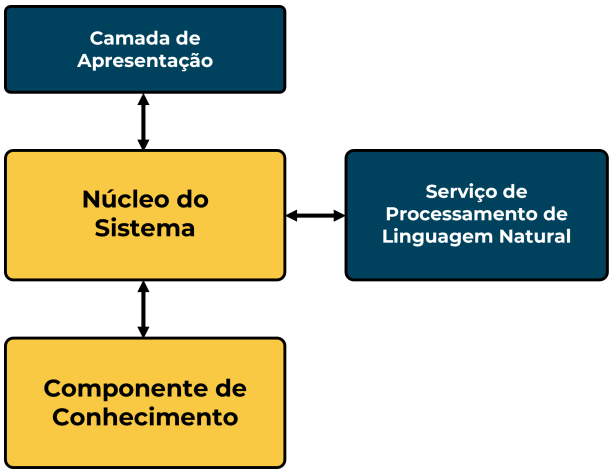


Figura 3.1: Versão Inicial do Desenho de Arquitetura de Sistema

é necessária a utilização de um Serviço de Processamento de Linguagem Natural.

Este serviço vai receber o que foi transmitido pelo Utilizador, fazer uma análise ao que foi dito e deduzir qual a intenção traduzida pela comunicação. A ideia é compreender um pouco o estado emocional em que o Utilizador transmite a mensagem, a mensagem que transmitiu e o tipo de resposta que é esperada. Este serviço consegue fazer esta dedução graças a uma análise gramatical ao que foi transmitido para o Agente, pelo que tem de estar pronto também para compreender o pacote que é recebido, seja ele textual, áudio ou visual, e tem de compreender o que é recebido em contexto.

Do outro lado do espetro encontramos o Componente de Conhecimento. Este componente tem de guardar informação sobre o tema a debater com o Utilizador, e tem de a ter preparado para saber o que e como deve responder à intenção transmitida pelo Utilizador. É nesta camada que os *outputs* estão preparados e são escolhidos. Tem, então, de ser possível escolher o *output* com base na intenção deduzida pelo Serviço de PLN.

Com base nesta informação, foi feita uma arquitetura inicial básica da forma como estas camadas estão conectadas entre elas (figura 3.1).

3.1.2 Fluxo de Comunicação no Sistema

Agora que compreendemos o que é esperado de cada camada, é necessário compreender não só a forma como comunicam como também o timing em que os diversos pontos acontecem. Para ser mais claro sobre o fluxo de comunicação, é apresentada a figura 3.1.

A comunicação apresentada ocorre para cada momento de comunicação dentro do Diálogo. Sempre que algum input é introduzido pelo Utilizador, o Sistema comunica através deste fluxo para saber o que devolver.

Vamos então por partes. Inicialmente, o Utilizador vai inserir o seu input no Sistema, na Camada de Apresentação. Esta camada vai limitar-se a passar a informação ao Núcleo do Sistema e a esperar que este lhe devolva o output.

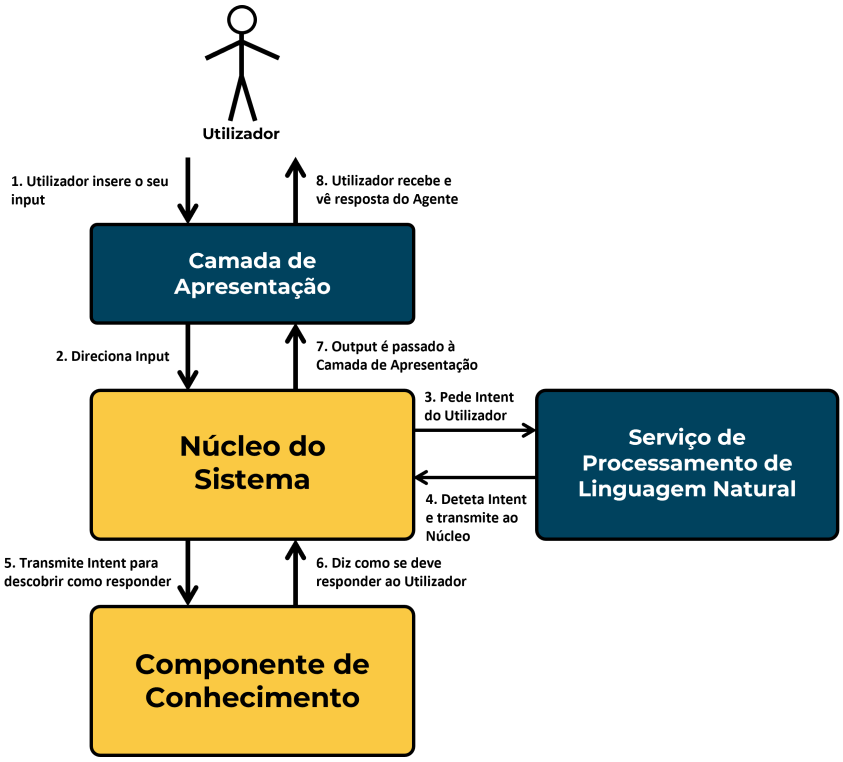


Figura 3.2: Fluxo de Comunicação do Sistema com base na Arquitetura Inicial Desenhada

Entramos então na fase de Processamento. O Núcleo do Sistema recebeu o input, mas não o sabe processar. Sendo este sistema descentralizado, o Núcleo vai passá-lo ao Serviço de PLN para que este compreenda a intenção do Utilizador. Compreendendo a intenção será necessário saber como responder, pelo que o Serviço de PLN a informa ao Núcleo do Sistema.

Agora que se sabe a intenção do utilizador, o Sistema tem de descobrir qual o output a devolver. Essa informação será obtida pelo Núcleo do Sistema através de um acesso à Componente de Conhecimento. Esta componente terá guardada a informação sobre o tema e a forma como deve ser debatido. Sabendo a intenção do utilizador, tal como o que este transmitiu, será passado ao Núcleo o output que deve ser devolvido.

O Núcleo do sistema vai então tratar o output e enviá-lo para a Camada de Apresentação, que a transmite ao Utilizador no formato desejado.

Este fluxo será repetido até que o Diálogo seja concluído. É também importante compreender que o fluxo pode ficar mais complexo em determinadas partes do diálogo ou até exigir outras partes no sistema, mas esta base é essencial para o correto funcionamento de um diálogo do início ao fim desde que o Núcleo ou a Componente de Conhecimento saibam qual o ponto de início e de final.

3.2 Requisitos/Ferramentas Utilizadas

Para cumprir as respetivas funções, cada componente tem de corresponder a determinados requisitos. Estes requisitos variam dependendo de quem os necessita e da razão para as quais são necessários, mas todos eles terão de ser respeitados e vistos como base para desenvolver a arquitetura por completo. Neste caso, vamos explorar os requisitos de cada Componente, seguindo o fluxo de comunicação do Agente para responder ao Utilizador para compreender o que se espera que seja transmitido.

Primeiro temos a Camada de Apresentação, e aqui é necessário ter em conta tanto a perspetiva do Utilizador como a do *Developer*.

Comecemos pelos requisitos gerais, independentes do tipo de comunicação escolhido. A ferramenta escolhida tem de permitir a construção de uma interface que permita ao Utilizador enviar o seu lado da comunicação numa primeira fase, e receber a comunicação do lado do agente numa segunda fase. Estamos a falar de uma parte do sistema interativa, e que deve acomodar toda a interação do Utilizador.

Dependendo do método de comunicação decidido pelo Developer, é necessário compreender o que o utilizador terá de encontrar na camada desenvolvida para compreender quais os requisitos da ferramenta a escolher para desenvolver o Sistema. Por exemplo, um sistema sem monitor e com comunicação por voz terá uma Camada de Apresentação definida pela máquina apresentada ao Utilizador e terá a interação baseada em botões. Alguns botões podem tornar-se essenciais para a interação com o Sistema parecer natural, como botões de volume, botões para repetir a mensagem transmitida pelo Agente, ou mesmo botões para o Sistema saber quando a mensagem vai ser transmitida pelo Utilizador. É necessário o Developer compreender que botões são necessários para uma melhor utilização do sistema.

No entanto, dado o exemplo definido para o Sistema desenvolvido, vamos focarmo-nos em Camadas de Apresentação em sistemas com monitor. Requisitos independentes do método de comunicação incluem:

- Uma forma clara de comunicar ao Agente o desejado;
- Uma forma clara de receber a comunicação do Agente;
- De que maneira o agente será percecionado pelo Utilizador.

A maneira como estes requisitos são definidos dependerá sempre da vontade do Developer. Se for necessária a existência de um avatar tridimensional que esteja animado durante a comunicação, isto pode exigir a utilização de uma aplicação que permita a utilização de um objeto criado numa aplicação de modelagem 3D de forma interativa (à semelhança do utilizado no VASelfCare). No caso referido, uma aplicação como o Unity pode mostrar-se como a indicada. No entanto, se estamos a falar de uma aplicação com comunicação numa caixa de mensagens sem qualquer tipo de animações, uma simples página HTML com recursos de JavaScript, JSON ou equivalente poderá ser o suficiente para conseguir atingir os objetivos definidos.

Passamos depois para o Núcleo do Sistema. Esta é a primeira camada de back-end definida, e os requisitos do Developer começam imediatamente nos conhecimentos deste na linguagem de programação escolhida para desenvolver o sistema.

Já os requisitos a que o ambiente tem de corresponder para poder ser utilizado nesta camada incluem:

- Conhecer toda a comunicação interna, para contactar cada parte do sistema no momento certo
- Saber a informação necessária para poder obter a resposta dos diversos elementos que compõem o sistema
- Tratamento e encaminhamento do input do utilizador. Este tratamento será apenas tão profundo quanto a Componente de Processamento de Linguagem Natural necessitar
- Conseguir reagir de forma apropriada caso ocorra uma mensagem de erro em algum ponto da comunicação interna

Posto isto temos os dois componentes com objetivos mais concretos e com ligação ao Núcleo. O Componente de Processamento de Linguagem Natural tem o requisito principal de conseguir compreender o que o input do utilizador pretende. Ao compreender a intenção do utilizador decidirá se é capaz de responder por si mesmo ou se será necessário informar o Núcleo do Sistema para que este consulte o Componente de Conhecimento para a resposta.

A ferramenta escolhida para esta Componente já vinha decidida de projetos anteriores, e é o DialogFlow. Os requisitos desta ferramenta que resultaram na sua escolha passaram pela capacidade de acompanhar o flow do diálogo para saber como deve prosseguir e que tipo de processamento deve ser feito.

Já para a Componente de Conhecimento existem requisitos distribuídos por diversas fases:

- É necessário que seja capaz de guardar informação relacionada com as fases do diálogo
- É necessário que consiga reter informação sobre o tema que o sistema quer discutir com o utilizador
- É necessário compreender como abordar os temas dependendo do utilizador

O nível de complexidade desta componente vai acompanhar o nível de complexidade do tema e aumenta conforme os tipos de utilizador existentes. Por exemplo, num sistema que tenta acompanhar um utilizador numa dieta com o objetivo de perder peso, cujos dados guardados sejam altura e peso, e cujo objetivo do diálogo seja discutir o nível de atividade física praticado ao longo da última semana, este componente tem de saber distinguir entre diversos tipos de utilizadores para saber os objetivos traçados, avaliar o quão perto ficou de atingir esses objetivos, e transmitir o que é necessário transmitir. Se um indivíduo não está a conseguir atingir os objetivos, o sistema terá de reavaliar se estes objetivos serão irrealistas, se o utilizador tem de

ser motivado de outra forma ou se poderá ser melhor alterar o tipo de foco do utilizador, e todo esse processamento tem de ser feito entre oIntent detetado pelo Componente de PLN, os dados de diálogos anteriores e os dados do utilizador.

O *Intent* aqui referido é, como fica claro com a tradução, a intenção que o utilizador tem quando faz um *input*. Sempre que o utilizador dá alguma informação ao sistema, este tem de responder de forma adequada ao que lhe foi dado. Com base no treino que é dado ao Componente de PLN, este vai perceber qual o objetivo a seguir e conduzir o sistema a dar uma resposta correta.

3.3 Metodologia de Desenvolvimento

Até agora definimos como os diferentes componentes do sistema comunicam entre si. Ficou também definidas as partes essenciais para que cada componente contribua da forma necessária para o bom funcionamento do agente.

Tendo esta matéria bem definida, podemos abordar a abordagem para o desenvolvimento de cada componente. Temos, assim, de compreender quais os princípios que guiam o *Developer* em cada momento do desenvolvimento do sistema.

Dado que a complexidade da interação do Utilizador com o Agente aumentará quanto mais pormenorizado for o Componente de Conhecimento, a implementação dos diferentes componentes podem seguir o Modelo de Desenvolvimento Cascata (*Waterfall*), dado que o sistema em si apenas necessitará de passar por iterações mais profundas na fase de manutenção. Dada que a interação seguirá sempre os mesmos passos básicos, e sendo possível gerir a interação de forma externa à programação base, não há problemas em escolher uma forma mais tradicional de gerir o projeto.

Portanto, antes de começar a olhar para cada componente individualmente, é necessário definir de início as bases sobre as quais o projeto se vai construir. Isto inclui:

- A representação do Agente para o Utilizador (bidimensional, tridimensional, sem representação visual)
- O tipo de interação entre o Utilizador e o Agente (texto, voz)

3.3.1 Núcleo do Sistema

O núcleo do sistema é utilizado para centralizar a comunicação necessária tanto com o utilizador como também com os restantes componentes do sistema. Assim, cada camada terá necessidades que têm de ser atendidas de forma específica.

Já temos definido que este componente está organizado por camadas, cada uma com a sua função. Nesse caso, teremos de ter em conta que ele vai comunicar com 3 componentes: a Camada de Apresentação, o Serviço de PLN e o Componente de Conhecimento.

Para começar a desenvolver é necessário escolher qual o ambiente em que o núcleo vai correr, e escolher uma linguagem de programação capaz de transportar os dados necessários para a comunicação. Para um melhor desenvolvimento do sistema, é necessário ter os pontos do Fluxo de Comunicação do Sistema bem interiorizados.

Assim sendo, sabemos dos subcapítulos anteriores que o núcleo do sistema tem de ser capaz de:

- Receber o *input* do Utilizador
- Encaminhar o *input* para o Serviço de PLN e receber o *intent* do utilizador
- Encaminhar este intuito para o Componente de Conhecimento para que este informe como o diálogo deve proceder
- Encaminhar o *output* para a Camada de Apresentação

Para estar certificado que o programa está a seguir aquilo que deve seguir nos momentos certos, é imperativo que as Fases do Diálogo estejam de certa forma *hard-coded* no sistema. Isto é, quando o *input* chega o Núcleo do Sistema tem de saber qual a Fase do Diálogo presente, para que possa comunicar ao Componente de PLN o tipo de intuito que este deve procurar, e quais as possíveis Fases de Diálogo que se seguem, para que o Componente de Conhecimento o informe de que forma vai prosseguir o Diálogo.

Sabendo desta informação, o Núcleo do Sistema vai funcionar como a base de todo o sistema. Como o Diálogo segue as mesmas fases independentemente do tema a ser debatido, este componente não terá de ser escalado com o aumento de complexidade de um diálogo. Importante salientar que o agente sabe que o Diálogo seguirá sempre as Fases do Diálogo uma vez que este agente será proativo e será ele a guiar toda a conversação.

Já no processo de Intervenção a longo prazo, será também o Núcleo do Sistema a saber onde os dados de cada Utilizador estão guardados, a acedê-los e a comunicar aos restantes Componentes quando for oportuno do ponto em que a Intervenção está.

Nesse caso, passa a ser necessário compreender de que forma devem ser desenvolvidos os restantes componentes para ser possível escalar a complexidade sem que seja necessário alterar os moldes do sistema.

3.3.2 Serviço de Processamento de Linguagem Natural

No Serviço de PLN encontramos um dos pontos-chave para que a interação seja natural. Como o objetivo é construir um agente que comunique em linguagem natural, é necessário escolher a ferramenta que consiga compreender o que o utilizador comunica. Tal como foi estabelecido previamente, a escolha desta ferramenta recai sobre o tipo de interação escolhido.

Tendo este ponto assente e a ferramenta escolhida, o desenvolvimento deste Componente vai ter de ter presente qual a Fase do Diálogo presente. Isto é essencial para que a ferramenta

compreenda aquilo que tem de procurar no texto que lhe é enviado. Como a ferramenta sabe a Fase atual dependerá da decisão dos *Developers*. Querendo simplificar o processo, e como o Núcleo do Sistema saberá sempre qual a Fase Atual, poderá ser comunicado pelo Núcleo o ponto em que está em conjunto com o *input* do utilizador.

A razão para o Serviço de PLN necessitar da Fase de Diálogo recai sobre o processamento que recai sobre o *input*. Caso esteja numa Fase de Início ou de Fim do Diálogo, a ferramenta estará à procura de determinados *inputs* que lhe permita perceber que deve seguir em frente e informar o Núcleo do Sistema que é isso que deve fazer. Caso esteja numa fase de Intervenção, poderão existir certos dados no *input* que terá de procurar, detetar e comunicar ao Núcleo para que este os processe da forma que achar correta.

Este Serviço, por princípio, não terá o trabalho de tomar decisões quanto ao rumo que a Intervenção segue. Mas tem de estar ciente o suficiente do que a Intervenção procura para que o consiga encontrar quando processa o *input*. O que ele procura é comunicado pelo Núcleo do Sistema, e o que deteta é comunicado ao Núcleo do Sistema.

3.3.3 Componente de Conhecimento

O Componente de Conhecimento será o portador de informação sobre 2 pontos: como cada Intervenção deve ocorrer no Diálogo, e como a Intervenção deve ser abordada a longo prazo (entre os diversos diálogos).

Para a Intervenção a longo prazo, este Componente deve receber do Núcleo do Sistema pontos-chave do estado em que a Intervenção se encontra. Por exemplo, o número de Diálogos tidos até ao momento, os pontos abordados anteriormente que foram concluídos com sucesso, os pontos que falharam, ou o que está a ser abordado atualmente.

Isto vai permitir o Componente de Conhecimento compreender, com base na informação que possui, de que forma o próximo passo deve ser dado. Estas decisões vão passar pela redefinição de objetivos estabelecidos anteriormente, definição de novos objetivos e uma avaliação a longo prazo do progresso que está a ser feito pelo Utilizador.

A curto prazo, o Componente deve saber como responder ao Utilizador dependendo do *input* que lhe é transmitido pelo Núcleo do Sistema, já processado pelo Componente de PLN. Este componente terá de ser consultado maioritariamente numa Fase de Diálogo de Intervenção, se não for consultado exclusivamente numa dessas fases. Isto porque nas fases de Início e de Fim de Diálogo as respostas serão muito mais básicas e poderão ser processadas automaticamente pela maioria de ferramentas de PLN. No entanto, quando chega à fase de Intervenção é necessário compreender qual o progresso feito.

O conhecimento que a ontologia de Diálogo possui tem de permitir a este componente compreender o estado do utilizador quanto à forma como tem abordado os objetivos definidos, avaliar o progresso que está a ser feito e tomar decisões sobre o próximo passo a tomar. Estes dados têm de estar presentes durante o Diálogo para garantir uma tomada de decisão acertada.

Também é importante que o Núcleo do Sistema seja informado, por exemplo, da forma

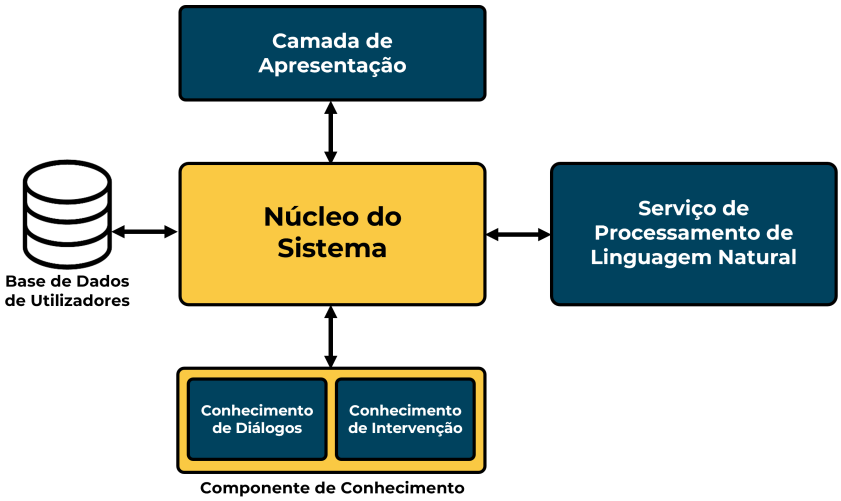


Figura 3.3: Versão 2 do Desenho da Arquitetura de Sistema

como os objetivos são abordados. Poderá fazer sentido, para cada objetivo individualmente, utilizar todas as Fases do Diálogo de Intervenção. Isto implicaria que o Diálogo repetisse essas 4 Fases de Diálogo, pelo que o Componente deve comunicar após a Definição de Objetivos a próxima fase de diálogo.

Tudo isto se traduz a um sistema que tem de saber desde o princípio do Diálogo os dados do Utilizador e dados relevantes sobre Intervenções passadas. Ao longo da dita Intervenção, o sistema tem de conseguir receber os dados corretos e transmitir ao Núcleo do Sistema tanto o *output* que deve ser transmitido ao Utilizador como possíveis dados que este deva armazenar ou atualizar para futuros diálogos com este utilizador e para que as decisões possam ser tomadas a longo prazo com informação atualizada.

3.3.4 Desenho da Arquitetura

Após um estudo mais aprofundado da Metodologia, foram feitas alterações à base da Arquitetura do Sistema criada inicialmente (figura 3.3).

Apesar de não existirem imensas diferenças, começamos a entender a forma como as diferentes camadas se partem e a entender as responsabilidades que cada uma vai ter.

3.4 Sumário

O sistema terá de ser dividido em 4 partes:

- Camada de Apresentação
- Núcleo do Sistema
- Serviço de Processamento de Linguagem Natural

- Componente de Conhecimento

Cada parte pode ser desenvolvida de forma independente das outras assim que existir um consenso sobre o método de funcionamento do sistema. O ponto-base que deve estar definido é o método de *input* do utilizador, e como os dados são transmitidos entre estes Componentes. Também será importante o timing em que a informação é transmitida. Para agilizar a comunicação entre todas as diferentes partes, é recomendado que os momentos de passagem de informação correspondam com uma Fase de Diálogo, uma vez que o fluxo de comunicação vai obrigar a comunicação dentro de todo o sistema e é mais fácil compreender implementar a transmissão de dados se estes momentos forem bem definidos.

Para o desenvolvimento da Camada de Apresentação é preciso ter definido, para além do método de comunicação, a forma como o Agente se apresenta. O Núcleo do Sistema será a base de todo o sistema e vai ser este a comandar o Fluxo de Comunicação assim que recebe o *input* do Utilizador. Tem de o transmitir para o Serviço de PLN e esperar que lhe seja comunicado o *Intent do utilizador*. Com este, vai comunicar o *input* e qualquer dado que seja relevante para a Intervenção ao Componente de Conhecimento para este gerar o *output*. Para comandar o Fluxo de Comunicação é essencial que siga o curso do diálogo para saber em que Fase está, e para garantir que o Componente de Conhecimento sabe como tomar decisões precisa de armazenar dados sobre o estado da Intervenção no Utilizador ao longo de qualquer tratamento que tenha de produzir. Para o Serviço de Processamento de Linguagem Natural, este tem de ser informado do tipo de dados que tem de encontrar no input do Utilizador caso alguma informação esteja a ser recolhida, e deve encontrar o Intent da mensagem do Utilizador para que possa ser tomada uma decisão sobre como responder. E quanto ao Componente de Conhecimento, este tem de estar pronto para entender o progresso do Utilizador com base nos dados e no Intent que o Serviço de PLN detetou, e decidir como se deve prosseguir. Por fim, deve conseguir entregar o output, decidindo com o Diálogo procede, junto com quaisquer dados que sejam relevantes para o Núcleo do Sistema armazenar.

Capítulo 4

Desenvolvimento da Prova de Conceito

Como Prova de Conceito, vai ser construída uma nova versão baseada no estudo que levou à criação do VASelfCare, utilizando como base também a utilização de Ontologias para a criação da base de conhecimento. O Sistema desenvolvido será um Agente Conversacional Proativo, e o seu objetivo será o de acompanhar um Paciente recentemente diagnosticado com Diabetes Tipo II. Este diagnóstico pode obrigar o Paciente a alterar algumas rotinas da sua vida de forma a controlar a doença, pelo que o Agente vai fazer o acompanhamento do Paciente, ajudando-o a aderir a novas rotinas e a garantir que estas são adotadas de forma consistente.

Portanto, esta Secção será utilizada para descrever numa primeira fase a Arquitetura desenvolvida para a Prova de Conceito, explicando também possíveis diferenças para o Sistema desenhado no Capítulo 3 (Secção 4.1), assim como uma apresentação do Fluxo de Comunicação que a Prova de Conceito apresentará (Secção 4.2). Serão apresentadas as Ferramentas Utilizadas para o desenvolvimento de cada Camada do Sistema (Secção 4.3), e depois será dada uma explicação sobre como o sistema é executado (Secção 4.4). Por fim vai ser explorado detalhadamente cada componente do Sistema, começando pela Camada de Apresentação (Secção 4.5), depois pelo Núcleo do Sistema e as camadas que o compõem (Secção 4.6), passando pelo Dialogflow (Secção 4.7) e terminando no Componente de Conhecimento (Secção 4.8). Serão também apresentados os Casos de Teste explorados para a realização da Prova de Conceito e o que cada um deles nos mostra (Secção 4.9).

4.1 Arquitetura do Sistema da Prova de Conceito

Como descrito no início deste capítulo, o sistema a desenvolver será um Agente Conversacional Proativo com o objetivo de acompanhar um Paciente diagnosticado com Diabetes Tipo II. Tendo este objetivo bem definido, surgem alguns pontos relevantes para ter em atenção. Avaliando estes pontos, passa a ser necessário escolher o Componente em que este será abordado. Relembrando, os Componentes do sistema são:

- Camada de Apresentação
- Serviço de PLN

- Núcleo do Sistema
- Componente de Conhecimento

Vamos então focar-nos nas questões relevantes para a construção da Prova de Conceito.

O primeiro ponto é o da Intervenção. Para conseguir que o Agente consiga resultados com o Utilizador, é necessário que o sistema guarde o seu estado. E para saber os dados que devem estar guardados, é importante ter a Intervenção desenhada. Para a construção desta Prova de Conceito, será utilizado o desenho da Intervenção da primeira versão do VASelfCare. Apesar de esta versão limitar a interação com o Agente à escolha de opções no ecrã, a Intervenção foi desenhada com os temas que seriam abordados, quanto tempo seria este tema abordado com base no progresso conseguido, e a ordem dos temas a seguir.

Como o objetivo é tornar o sistema uma base que pode ser adaptada com base na Ontologia, torna-se um problema a resolver descobrir como vai a Intervenção ser representada para poder ser lida pelo sistema. Como a Intervenção tem os Temas a serem abordados, esta tem de ser desenhada em conjunto com a Ontologia para os termos estarem unificados. Esta parte da abordagem terá de estar guardada no Componente de Conhecimento.

Quanto aos dados do Utilizador, convém estarem guardados numa Base de Dados para que seja atualizada e acedida todas as sessões. Comparado com o Sistema descrito na Secção 3, e dado que numa utilização de grande escala esta base de dados será muito provavelmente um serviço exterior, encontramos aqui uma diferença para o Desenho descrito. Para efeitos de teste da Prova de Conceito em diversas situações e com diversos utilizadores em pontos diferentes da Intervenção, será incorporada uma Base de Dados ligada diretamente ao Núcleo do Sistema. Para além das responsabilidades que o Núcleo do Sistema tem no fluxo de comunicação com o Serviço de PLN e com o Componente de Conhecimento, será necessário identificar os pontos de comunicação com a Base de dados e incorporá-la no Fluxo de Comunicação.

O segundo ponto é a Apresentação do Agente. Dado que o grande objetivo desta dissertação é o de mostrar um Caso de Estudo funcional, a forma como este se apresenta ao Utilizador perde a relevância. Por isso mesmo, a comunicação do Utilizador e do Agente será através de uma caixa de mensagens. O input será colocado numa caixa de texto, e será visível para o Utilizador tanto as mensagens escritas por si como o output enviado pelo Agente.

Com o incorporamento da Base de Dados, a versão final da Arquitetura do sistema está presente na figura 4.1.

4.2 Fluxo de Comunicação da Prova de Conceito

Com a Arquitetura detalhada desenhada, é possível termos uma melhor noção do que se espera da comunicação dentro do sistema. Na figura 4.2 podemos ver o fluxo durante o Diálogo quando é necessário um acesso ao Componente de Conhecimento.

Este é o fluxo base que se pode esperar da arquitetura. A comunicação terá ligeiras alterações nos acessos que tem dependendo do que está a ser tratado com o utilizador, o que resultará em

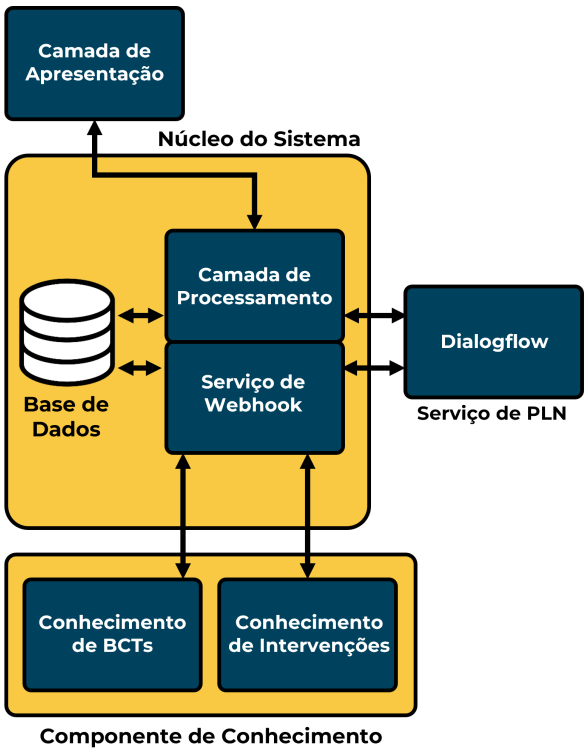


Figura 4.1: Arquitetura do Sistema

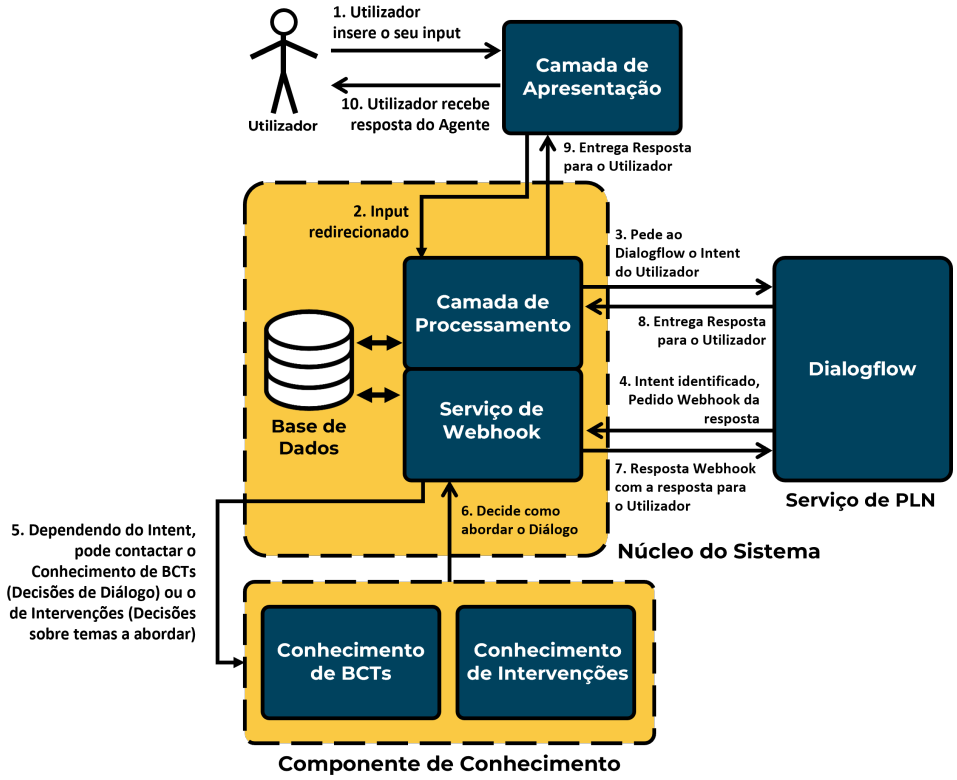


Figura 4.2: Fluxo de Comunicação do Sistema durante um Diálogo

fluxos ligeiramente diferentes. Por exemplo, quando o Utilizador será indispensável um acesso à base de dados, acesso esse que não se verifica no Fluxo base demonstrado nesta figura. Para ter uma melhor noção dos diferentes fluxos de comunicação que se podem encontrar na Prova de Conceito, estes serão explorados após a apresentação mais aprofundada do sistema que se encontrará nas próximas secções.

Na continuação desta Secção vamos abordar cada um dos componentes, as ferramentas utilizadas para o desenvolvimento de cada um deles, a forma como foram desenhados de modo mais detalhado e explorar um pouco também o código por trás de cada parte para compreender como o sistema num todo consegue formar um agente.

4.3 Ferramentas Utilizadas

Para concretização da Prova de Conceito, foi necessária a utilização das seguintes ferramentas:

- Dialogflow - Ferramenta utilizada como Serviço de Processamento de Linguagem Natural, oferece uma API acessível para três linguagens de programação: Java, Node.js e Python
- Python - Linguagem de programação de alto nível, funcional e orientada a objetos. Possui uma grande biblioteca de funções que permitem a adaptação de outras ferramentas ao sistema de forma acessível, e é acessível à incorporação de módulos
- HTML - Linguagem de marcação utilizada para desenvolvimento de páginas web
- CSS - Linguagem com regras de estilo para aplicar em conteúdo HTML
- JavaScript - Linguagem de programação baseada em objetos utilizada para desenvolver scripts para páginas web, permitindo que estas consigam apresentar conteúdo dinâmico
- Flask - Framework web que utiliza a linguagem de programação Python, permite ao developer criar uma aplicação simples sem a preocupação com protocolos ou thread management
- OWLready - Módulo de programação Python para utilizar com ontologias. Permite aceder, modificar e guardar ontologias, e ainda suporta o Reasoner HermiT
- Protegé - Editor de Ontologias. Possui a capacidade de utilizar diversos Reasoners para fazer inferências e facilita a criação de Ontologias.

4.4 Criação do Ambiente

Antes de explorar os componentes que compõem o sistema, esta secção serve para dar uma curta apresentação aos passos que devem ser dados para criar o ambiente e alguns esclarecimentos

```
# VAPrevention_CX

To set up the virtual environment:
$ py -m venv env
$ env\Scripts\activate

To set up the webhook (copy the https link and paste it on the 'localhost' webhook on DF,
                       add '/webhook' at the end of the link):
$ ngrok http 5000

Install the requirements:
$ pip install -r requirements.txt

$ set GOOGLE_APPLICATION_CREDENTIALS=C:\mib-exemplo-projeto-8a2928051418.json

To run the app:
$ flask run
```

Figura 4.3: Instruções no ficheiro READ.ME para inicializar o agente

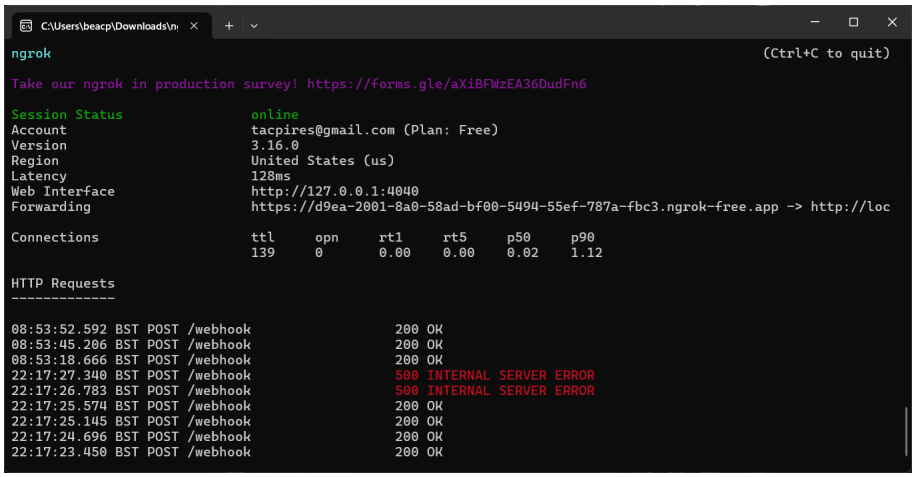


Figura 4.4: ngrok inicializado no terminal

sobre a forma como estas decisões foram tomadas. Os comandos utilizados encontram-se no ”README.md”do projeto, e serão estes os que serão explorados nesta secção.

O primeiro passo é a criação de um ambiente virtual. Este ambiente vai permitir a identificação do projeto no DialogFlow. Esta ferramenta será explicada com maior detalhe na Secção 4.4, mas o que esta função permite é a criação de variáveis com credenciais e constantes que terão de ser acedidos ao longo da execução do programa. De seguida, este ambiente vai ser ativado.

O segundo passo é a criação do Webhook. Como será explicado na Secção 4.4, para que se consiga criar uma ligação entre o DialogFlow e o sistema é necessário um webhook com que o DialogFlow consiga comunicar. Para criar um webhook online que a ferramenta consiga aceder é utilizado o ngrok. O ngrok é uma aplicação que permite a criação de um túnel até à máquina local, e facilita a criação de um servidor local, sendo apenas necessário escolher o porto por onde a ligação será feita. Após criar este túnel, será necessário copiar o link para o túnel e colá-lo no webhook criado do DialogFlow, adicionando ”/webhook”no final. Esta adição é o que permite à aplicação saber a função que vai executar.

O terceiro passo é a instalação das versões certas dos diferentes módulos utilizados no sistema, para garantir que todas as funções utilizadas ao longo do código ainda são aceites por to-

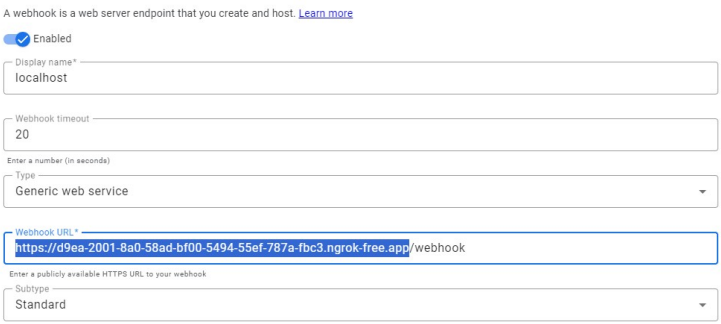


Figura 4.5: Webhook no Dialogflow com o link criado pelo ngrok

das as partes. Os módulos e respetivas funções estão disponíveis no ficheiro ”requirements.txt”.

E o último passo é o de executar o programa através do Flask. O Flask é uma *Web Application Framework* e permite ao *developer* criar uma aplicação de forma simples, sem ter de se preocupar com protocolos ou *thread management*. Esta *Framework* foi escolhida primeiro pela facilidade de utilização ao desenvolver a aplicação em python, e em segundo pela possibilidade de fazer *URL Routing* internamente.

Assim que o flask for executado, é possível abrir o localhost no port correto e vai ser possível utilizar a aplicação. Mesmo na consola onde o flask é executado é possível aceder diretamente ao link do localhost com o porto escolhido na criação do ngrok.

4.5 Núcleo do Sistema

Quando olhamos para o Núcleo do Sistema desenvolvido, a grande diferença de arquitetura entre o Desenho do Sistema (Secção 3) e a Prova de Conceito desenvolvida é a localização da Camada de Apresentação. Num Serviço Online ou numa Aplicação que tenha de ser instalada em diversos computadores com acesso a um Núcleo centralizado para os diversos utilizadores, a Camada de Apresentação deve estar separada do resto do sistema para que o Utilizador não tenha acesso direto a informação que não deva. No entanto, para a construção da Prova de Conceito é possível colocá-la junto ao Núcleo para facilitar o desenvolvimento e a ligação entre os dois pontos.

Com isto em conta, durante as seguintes subsecções vai ser apresentado cada Componente do Núcleo do Sistema, qual a ferramenta escolhida para o desenvolvimento, o código feito e imagens que contextualizam o seu funcionamento no sistema.

4.5.1 Camada de Apresentação

Esta Camada é a cara do Agente para o Utilizador. Em grande escala, a escolha e desenvolvimento de um Avatar bidimensional ou tridimensional poderia elevar o projeto e conseguir alcançar melhores resultados com o Paciente. No entanto, numa fase inicial de desenvolvimento, não será dada prioridade ao aprofundamento das capacidades da Camada de

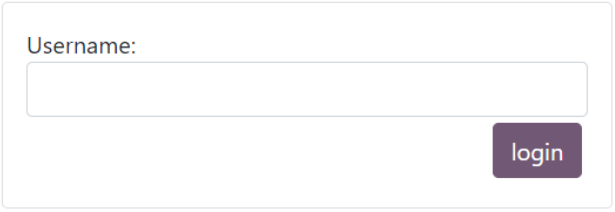


Figura 4.6: index.html

Figura 4.7: main.html durante utilização

Apresentação. O objetivo é confirmar que o sistema funciona e está montado apropriadamente para conseguir guiar um Diálogo com um utilizador, pelo que o *back-end* será priorizado.

Foram assim desenvolvidas duas Páginas HTML inicializadas a partir da execução do ficheiro ”index.py”. A primeira página HTML, ”index.html”, tem apenas uma caixa de texto onde o utilizador coloca um nome de utilizador para que o Núcleo do Sistema saiba quais os dados que tem de recolher da base de dados para dar início ao diálogo.

Reconhecendo o utilizador, é aberta a Página HTML ”main.html”. É nesta página que o Diálogo vai ocorrer. O utilizador consegue ver as mensagens enviadas pelo Agente e responder através de uma caixa de texto. Da mesma forma que no ”index.html”, enviar a mensagem clicando no Enter vai chamar o Serviço de Webhook para que este execute a função correta. Mais informação sobre o Serviço de Webhook será fornecida na subsecção 4.2.4. Para que seja o mais simples de compreender possível, as mensagem enviadas pelo Agente ficam alinhadas à esquerda com cor roxa e as mensagens enviadas pelo Utilizador ficam alinhadas do lado direito com cor verde.

4.5.2 Camada de Processamento e Serviço de Webhook

Compreendendo a forma como o sistema é visualizado pelo utilizador, torna-se necessário explicar como se comporta a Camada de Processamento durante todo este processo.

Para criar uma página no browser por onde trabalhar no projeto, foi escolhido o Framework Flask. Como explicado na Secção 4.3, um dos pontos fortes do Flask é a possibilidade de fazer URL Routing de forma simples. Esta capacidade é aproveitada para que se consiga andar entre as diferentes páginas html sem qualquer tipo de atrito, e também para executar a função correta dependendo da ação realizada na página html.

Assim, no momento em que se executa na consola um ”flask run”dentro do ambiente criado vamos ter acesso ao ”index.html”. A Camada de Processamento vai imediatamente conectar-se à Base de Dados de utilizadores e à Ontologia do Componente de Conhecimento, e vai ficar à espera que o utilizador coloque o seu *username*.

Quando o Utilizador colocar o seu *username* e clicar em ”Login”na página inicial, o html e o JavaScript estão preparados para fazer um Post no servidor local com a função ”/login”e o *username* como mensagem. Aproveitamos agora a capacidade de routing para que a aplicação leia o Post com a mensagem, reconheça que tem de confirmar a existência do *username* e agir em conformidade com a sua existência. Se o *username* está guardado na base de dados, o Utilizador vai ser inicializado na Ontologia e a camada de apresentação vai mudar de página para o ”main.html”, onde a interação vai ser realmente tratada.

A partir daqui, todas as mensagens são recebidas na aplicação através da função de Route ”/send_message”, acompanhada do input do Utilizador. Esta função vai consultar o Serviço de PLN (apresentado nesta secção pelo nome da ferramenta, Dialogflow, na secção 4.5) e esperar que o serviço utilize o link gerado pelo ngrok com o routing ”/webhook”para saber como deve proceder.

As mensagens recebidas pelo Núcleo neste momento virão sempre acompanhadas por uma *tag*, que representa o momento do diálogo em que se está. Dependendo da tag, o Núcleo saberá que informação adicional está a receber e que tipo de resposta tem de procurar no Componente de Conhecimento. As tags que pode receber são:

- review_tasks: Fase de Diálogo nº.3 (Revisão de Objetivos). É necessário ver na ontologia quais os objetivos definidos na última consulta para poder perguntar ao Utilizador se estes foram atingidos;

- determinant: Fase de Diálogo nº.4 (Avaliação), quando o utilizador não consegue atingir os objetivos explica qual a razão (determinante) para não os ter atingido. Com base no determinante, encontra-se na Ontologia como apoiar o Utilizador;

- counselling: Na Fase de Diálogo nº.5 (Aconselhamento), pode ser necessário apoiar o Utilizador e explicar-lhe as razões para alguns destes objetivos serem definidos, ou até mesmo dar dicas sobre como atingir esses objetivos;

- task_completed: Caso após as Fase de Diálogo 3 a 5 o Objetivo definido tenha sido cumprido, é necessário confirmar com a ontologia quais os próximos passos a dar. Estes passos podem incluir tentar manter a rotina durante mais algum tempo ou a introdução de uma nova rotina a tentar adotar;

- task_incompleted: Caso após as Fase de Diálogo 3 a 5 o Objetivo definido não tenha sido cumprido, é necessário confirmar com a ontologia quais os próximos passos a dar. Estes passos podem incluir tentar adotar a mesma rotina durante mais algum tempo, tentar diminuir um pouco o exigido ao Utilizador dentro da mesma rotina ou até mesmo a introdução de uma nova rotina e adiar esta a atual para outro momento mais oportuno;

- assign_tasks: Fase de Diálogo nº.6 (Definição de Objetivos). Independentemente das decisões tomadas pela Ontologia, seja a continuação dos mesmos objetivos ou uma nova rotina, é necessário informar o Utilizador de quais as rotinas em que vai trabalhar até ao Diálogo seguinte;

- checkpoint: Como explicado na Secção 3, por vezes numa intervenção é necessário abordar mais do que um tópico e discutir mais do que um objetivo. Isto exige repetir as Fases de Diálogo da Intervenção (3 a 6) enquanto há temas para discutir. Quando o Serviço de PLN envia esta tag ele pretende saber se o Diálogo vai terminar e deve seguir para a Fase de Fim de Diálogo, ou se existem mais tópicos a discutir o que o leva a ter de repetir a Fase de Intervenção e a regressar à Fase de Diálogo nº.3.

Ao longo de todos estes passos, com o apoio do ficheiro ”OntologyConnection.py”, serão armazenados dados importantes para que informação como o Tópico Ativo (o tópico geral que está a ser discutido com o Utilizador nesse momento) e o Objetivo Ativo (o Objetivo que se propôs ao Utilizador no Diálogo anterior que está a ser discutido nesse momento) estejam disponíveis para várias tomadas de decisões. Estas tomadas de decisão incluem:

- Atualização da Base de Dados com os Tópicos e Objetivos Ativos para o Diálogo seguinte

- Atualização da Base de Dados com os Tópicos e Objetivos que foram atingidos ou falhados

- Momento da Intervenção Geral em que o Utilizador está, para que seja possível perceber como se avança depois dos Tópicos e Objetivos atuais

- Se existem mais Tópicos ou Objetivos a discutir no Diálogo atual

A atualização da Base de Dados é o que torna a Intervenção a longo prazo possível, pelo que não deve ser ignorada. Temos, então, de compreender como está organizada.

Figura 4.8: Base de dados criada e os respetivos parâmetros

4.5.3 Base de Dados

A tecnologia escolhida para criar a Base de Dados nesta Prova de Conceito foi o sqlite3. Esta ferramenta foi escolhida principalmente por não ser necessário criar ou utilizar um servidor *online*, estando acessível no ambiente criado na máquina local, não ser necessária uma instalação muito pesada e ser acessível através de comandos SQL.

A base de dados criada tem os parâmetros de informação da figura 4.9.

Com a Base de Dados criada, é extremamente acessível a criação de novos utilizadores e o acesso e alteração de dados em utilizadores existentes. Por exemplo, no momento de inicializar o Utilizador, é possível selecionar dados como a intervencao, dialogo, fase, dialogo_fase, idade e nivel_risco simplesmente com o *username* dado pelo Utilizador. Também se torna acessível a atualização de dados no final de cada Diálogo, ou até mesmo no final de dialogar sobre cada tópico para reduzir a quantidade de informação trocada em cada acesso.

Assim sendo, vamos explorar cada parâmetro da base de dados:

- nome: text. Referente ao *username* do utilizador. É a chave primária da base de dados;
- dialogo: integer. Diz o número de diálogos tidos até agora na intervenção;
- fase: text. Diz fase da intervenção está neste momento a ser abordada;
- dialogo_fase: integer. Informa quantos dos diálogos foram tidos durante a fase atual;
- idade: integer. É a idade do Utilizador;
- nivel_risco: integer. Informa o nível de risco do utilizador dado por um especialista;

- intervencao: text. Diz qual das Intervenções existentes no Componente de Conhecimento está a ser seguido;

- target_json: text. Guardado como texto, mas com o objetivo de ser carregado como um ficheiro json para poder ser lido e interpretado pelo Núcleo do Sistema. Tem de ter armazenado o Objetivo Atual a ser abordado, que Objetivos já foram cumpridos pelo utilizador e que Objetivos foram iniciados mas nunca terminados.

Na secção 4.5, onde se aborda o Componente de Conhecimento, será abordado em detalhe como este ficheiro json estará escrito para poder ser utilizado.

Para a utilização dos Casos de Teste, foram criados dois ficheiros Python: o ”reset.py”, que cria a base de dados caso esta não exista ainda e cria os Utilizadores dos Casos de Teste, ou torna a base de dados ao estado original dos Utilizadores caso se queria iniciar os testes do princípio se esta já existira, e o ”checkDB.py”, que imprime na consola tudo o que está guardado na Base de Dados para efeitos de *debugging* durante os testes.

4.6 Dialogflow

No lugar do Serviço de Processamento de Linguagem Natural encontramos o Dialogflow. Esta ferramenta foi escolhida previamente, sendo dada como a base do projeto apresentado. Esta escolha advém das capacidades de comunicação em Linguagem Natural, sendo capaz de criar diversos *Flows* de Diálogo, consegue identificar os *Intents* procurados no nosso tipo de utilização, é capaz de criar tipos de entidades adaptados às necessidades do sistema, e a capacidade de utilizar *webhooks* para compreender como se deve responder.

Sublinhar que o Dialogflow é capaz de ter um diálogo completo sem a utilização de qualquer outro sistema, mas querendo nós aproveitar as capacidades de uma ontologia com conhecimento para poder ter um diálogo mais detalhado, a capacidade de utilizar *webhooks* que consigam comunicar com o sistema criado por nós para confirmar como se deve responder torna o Dialogflow numa ferramenta muito potente. Assim, é possível aliar a capacidade de detetar Intents desta ferramenta, simplificando o processo de desenvolvimento das Fases Iniciais e Finais de um Diálogo, com a capacidade de utilização de uma Ontologia para fazer um acompanhamento personalizado a um Utilizador.

4.6.1 Bases de Funcionamento do Dialogflow

O Dialogflow funciona assim através da criação de um Flow de Diálogo. Este Flow será constituído por nós com ligações de route entre eles. O caminho que o Flow segue depende do Intent que a ferramenta procura e encontra no input do Utilizador. Portanto, o início do desenvolvimento desta componente do projeto passou pela criação dos Intents.

O Intent é, como o próprio nome indica, a intenção encontrada no input do utilizador. É formada por um nome e por Frases de Treino. Os Intents criados para esta fase do projeto

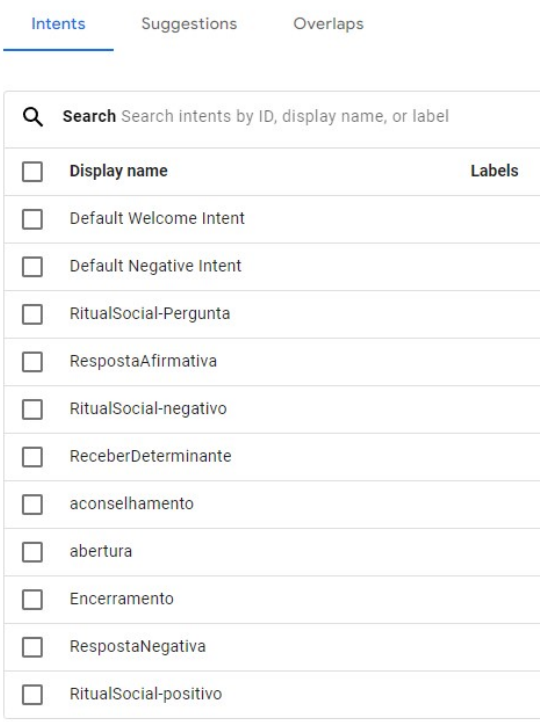


Figura 4.9: Intents criados no Dialogflow

foram os da figura 4.10.

Como exemplo, vamos olhar para os Intents com os nomes ”RespostaAfirmativa”e ”RespostaNegativa”. Utilizados em momentos onde se espera que o Utilizador responda a uma pergunta de sim ou não, as Frases de Treino rondarão aquilo que se espera que seja o próximo input do Utilizador. No Intent ”RespostaAfirmativa”podemos encontrar Frases de Treino mais variadas como ”sim”, ”claro”, ”certo”ou ”compreendi”, mas também encontramos alguns exemplos de formas diferentes de escrever o que é visto por um leitor humano como o mesmo, como ”ok”ou ”okay”, e ”yup”ou ”yap”. Quantas mais Frases de Treino, mais o Dialogflow tem por onde aprender e deduzir frases diferentes que trazem o mesmo Intent que se espera que seja capturado.

Outro recurso a ter em conta da utilização do Dialogflow são as Entidades. Um tipo de Entidade define um tipo de informação, e pode ser utilizado para recolher e trabalhar com Parâmetros dados pelo utilizador no seu input. Por exemplo, imaginando que foi pedido ao Utilizador para dar 3000 passos diários e este não atingiu o objetivo, dizendo no input ”Não consegui cumprir o objetivo, só dei 2500 passos”. Sendo o número de passos um tipo de informação bem definido (um número inteiro) este pode ser identificado pelo Dialogflow em contexto e utilizado pelo Componente de Conhecimento para saber como prosseguir. É um tipo de processamento relativamente simples uma vez que estamos a trabalhar com um tipo de dados já conhecido.

No entanto, imagine-se que o sistema quer chegar ao fundo do problema e após perguntas

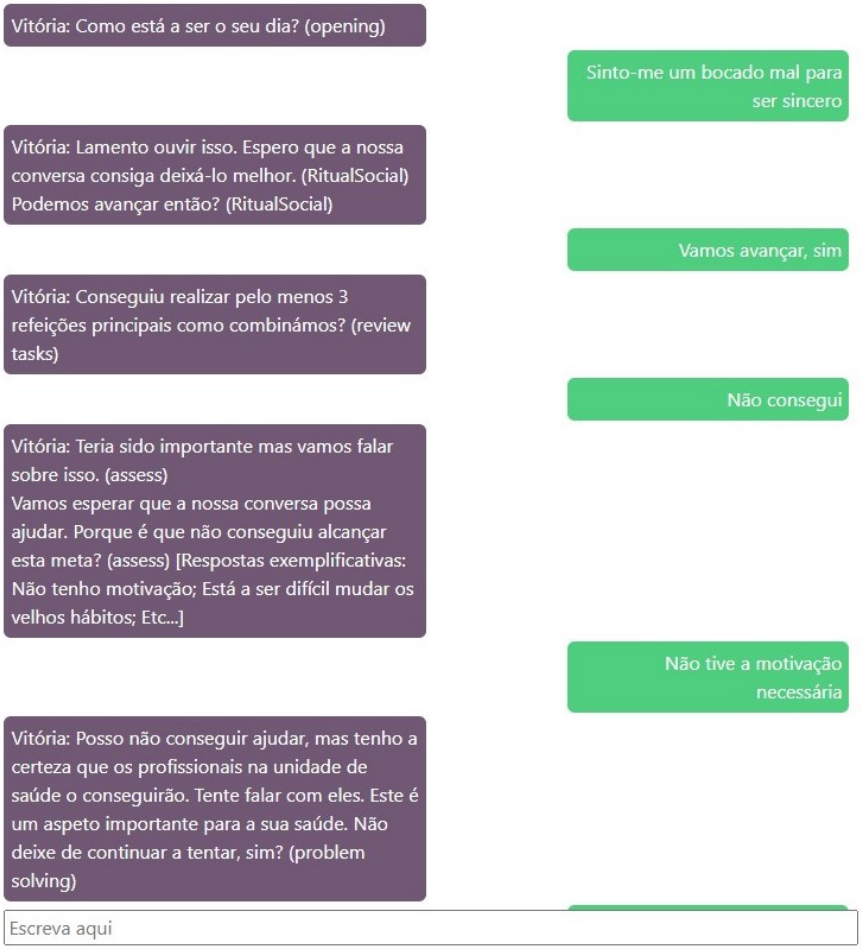


Figura 4.10: O agente reage de forma diferente caso o utilizador tenha reações negativas

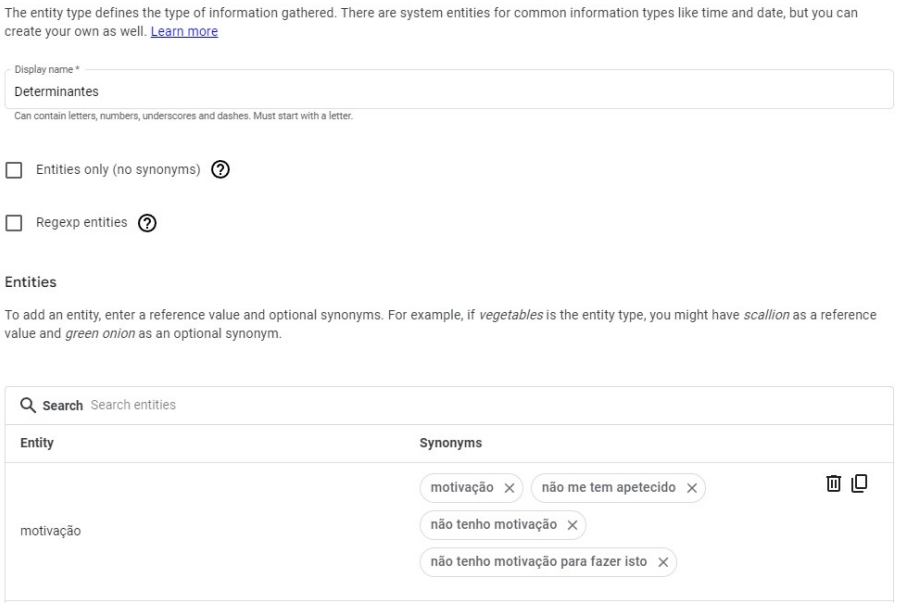


Figura 4.11: Exemplo de um dos Determinantes que o Dialogflow consegue identificar neste fluxo

ao utilizador se há alguma razão para o objetivo não ter sido cumprido, ele responde ”Não consegui cumprir o objetivo porque tenho dificuldade em andar”. Se o Dialogflow não souber aquilo que procura, não saberá como tratar a mensagem recebida. É para este fim que se cria um Tipo de Entidade ”Determinantes”, que se dedica unicamente a encontrar este tipo específico de informação. Dentro do Tipo de Entidade podemos criar então as Entidades com um valor de referência para aquilo que se procura (no caso exemplificado será ”dificuldade”, e para cada entidade é possível colocar Sinónimos. Estes sinónimos são semelhantes às Frases de Treino dos Intents. Assim, quando os determinantes são identificados podem ser enviados para o serviço de webhook, que o envia para o Componente de Conhecimento decidir o output.

4.6.2 Fluxo no Dialogflow

Tendo as bases cobertas, passa a ser necessário compreender o Flow do Diálogo. Para construir este flow, é necessário compreender dois pontos essenciais:

- as Fases do Diálogo
- os momentos em que informação tem de ser passada entre os componentes do sistema

A razão pela qual não basta fazer um nó por Fase de Diálogo é porque, por vezes, uma fase de diálogo corresponde a mais do que um comentário de cada lado, o que pode exigir uma procura de Intent diferente dado algum input. Isto pode exigir a criar nós com objetivos mais definidos daquilo que têm de realizar antes de continuar.

Outro facto a ter em conta é que o sistema tem de ser proativo. No entanto, e à semelhança de outras ferramentas similares, o Dialogflow está montado para conseguir ler e resolver os inputs do Utilizador. Isto porque Serviços de PLN estão naturalmente construídos para conseguir

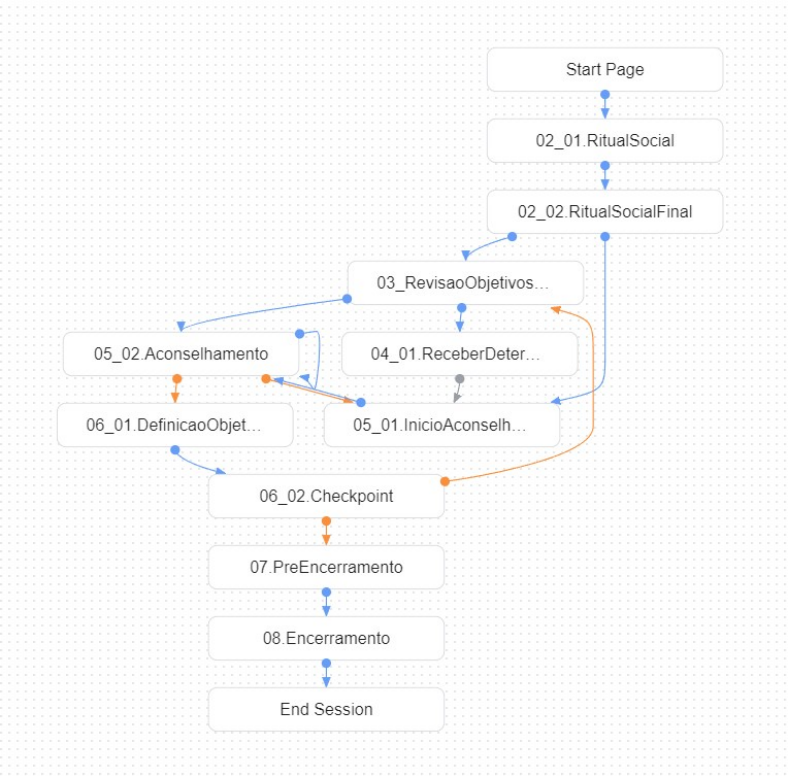


Figura 4.12: Fluxo do Diálogo seguido pelo Dialogflow

funcionar como um chatbot. Para tornar o Agente proativo e permitir que a conversa seja guiada pelo Agente aos olhos do Utilizador, este vai esperar que o Núcleo do Sistema lhe dê um sinal de que o Diálogo vai começar para que este possa enviar a primeira mensagem: uma mensagem de abertura do Diálogo. Vejamos o fluxo criado no Dialogflow na figura 4.12.

À exceção do primeiro nó, é visível no nome de cada um dos restantes qual a Fase do Diálogo correspondente. Como descrito na Secção 2.2, o primeiro conjunto das Fases do Diálogo de Bickmore são a Abertura da Conversa e o Ritual Social. Para isso, o Dialogflow terá 3 nós:

- o ”Start Page”, que vai enviar o primeiro cumprimento ao Utilizador como Abertura da Conversa
- o ”02_01.RitualSocial”como o ”02_02.RitualSocialFinal”, que vão fazer toda a parte do Ritual Social, começando por perguntar ao utilizador como está e fazendo a transição do Início do Diálogo para a Intervenção.

Durante este primeiro conjunto:

- todas as mensagens enviadas ao utilizador estão guardadas diretamente no nó que as decide entregar
- a mensagem pode mudar ligeiramente dependendo do transmitido pelo utilizador no Ritual Social, mas o caminho é direto.

Com isto quero dizer que se a resposta do Utilizador a como lhe tem corrido a semana for positiva, o Agente responderá positivamente a essa reação, mas caso a semana tenha corrido mal o Agente irá lamentar antes de prosseguir o Diálogo.

É também no final do Ritual Social que vemos a primeira bifurcação do fluxo. Vamos começar por falar do caso mais comum: o Diálogo Atual não é o primeiro diálogo entre o Agente e o Utilizador. Neste caso:

1. Começamos o conjunto de Fases da Intervenção normalmente, mudando o fluxo de rumo dependendo das respostas que o Utilizador lhe dá

2. Inicia no nó "03_RevisaoObjetivos+04_Avaliacao", um nó que aborda as Fases de Diálogo 3 e 4 como o nome indica, perguntando ao utilizador como foi a última semana e lembrando-o dos objetivos definidos anteriormente

3. Caso a resposta seja positiva, este vai passar para a Fase de Aconselhamento no nó "05_02.Aconselhamento", onde ficará enquanto o Componente de Conhecimento quiser dar informação ao Utilizador.

4. Acabando de passar a informação toda ao Utilizador, passa para o nó "06_01. DefinicaoObjetivos"para que mais uma vez o Componente de Conhecimento informe, através do Webhook, quais os objetivos que o Utilizador deve tentar atingir até ao próximo diálogo.

Caso na Revisão de Objetivos e Avaliação o Utilizador informe que não conseguiu atingir os objetivos

1. O nó a explorar será o "04_01.ReceberDeterminante". Este nó serve para recolher o Determinante e informar o webhook da razão pela qual os objetivos não foram atingidos.

2. O sistema, ao ser informado da razão pela qual o Utilizador não alcançou os objetivos, vai preparar um conselho mais pessoal para dar ao Utilizador no nó "05_01. InicioAconselhamento"

3. Passa para o nó "05_02"onde o fluxo se acaba por encontrar e explicar ao Utilizador a importância geral das rotinas que está a tentar adaptar.

Após a definição de objetivos encontramos sempre o nó "06_02. Checkpoint". Apesar de este Checkpoint não fazer parte de uma Fase do Diálogo, foi colocado o número 6 por ser uma decisão que o sistema tem de tomar antes de prosseguir para o Fim do Diálogo: caso todos os objetivos definidos já tenham sido abordados, o sistema pode seguir para o conjunto de nós representantes do Fim do Diálogo. Caso ainda existam mais objetivos a serem discutidos, o fluxo voltará para o nó da Revisão de Objetivos, onde o ciclo se manterá.

Dando agora um passo atrás, é preciso recordar que caso este seja o primeiro diálogo da Intervenção ainda nenhum objetivo foi definido ao Utilizador. Quando é este o caso, no final do nó "02_02.RitualSocialFinal"o Dialogflow vai passar diretamente para o nó "05_01.InicioAconselhamento",

onde vai abordar um pouco a rotina que deve ser adaptada antes de definir objetivos, mantendo assim o Diálogo coeso.

4.7 Componente de Conhecimento

Falta apenas abordar o desenvolvimento do Componente de Conhecimento. Este componente é composto na sua totalidade por duas partes: Uma Ontologia com conhecimento de BCTs, e ficheiros de Conhecimento sobre a Intervenção em questão.

Quando falamos na escalabilidade do sistema, é neste componente que a abstração ganha forma. Em específico no Conhecimento de Intervenções, é necessário criar uma representação textual da forma como a Intervenção deve ser abordada ao longo do tempo, para além da ontologia definida para execução das BCTs. Estas duas partes devem estar em sintonia no que diz respeito aos termos utilizados.

As subsecções seguintes vão explorar a construção dos ficheiros criados para este componente. Estes ficheiro têm de permitir ao Componente de Conhecimento tomar dois tipos de decisões. A principal e mais ativa é a decisão a curto prazo, dentro do diálogo, quanto recebe informação do Núcleo do Sistema e decide como o Diálogo deve prosseguir. A segunda é a longo prazo, onde define a Intervenção de diálogo para diálogo e decide que tópicos deve abordar, em que momento se pode avançar nos tópicos, e em que pontos se devem adiar os tópicos que estão a ser abordados.

4.7.1 Conhecimento de BCTs e Descrição da Ontologia

Com a utilização do *Protégé*, é definida uma ontologia baseada fortemente no trabalho desenvolvido por Maria Inês Bastos, com algumas adaptações para permitir uma aplicação mais abstrata.

O ponto principal da estrutura da Ontologia são as Classes. Estas classes, e consequentes subclasses a serem criadas, serão o tipo de instâncias necessárias para o mecanismo de decisões da ontologia para decidir o *output* a entregar ao Utilizador. No caso desta ontologia, as classes existentes são a "*User*"e a "*Behavior Change Intervention*". Com nomes autoexplicativos, a classe "*User*"vai ser utilizada para criar Utilizadores como Instâncias, enquanto a classe "*Behavior Change Intervention*"terá informação sobre a aplicação da Intervenção desejada. Estas serão exploradas com mais detalhe no seguimento desta secção.

As instâncias destas Classes vão também levar consigo Propriedades de Anotações, de Dados e de Objetos. Na classe "*User*", a única Propriedade de Anotação presente é o "*username*", representativo do nome de Utilizador confirmado no início do sistema. As Propriedades de Dados vão armazenar dados como a idade, altura, nome, nível de risco ou peso. E as Propriedades de Objetos vão ligar subclasses da classe "*Behavior Change Intervention*"para saber informação como o tópico e objetivo comportamental a ser abordado, e os objetivos definidos anteriormente que devem ser discutidos na Revisão de Objetivos.

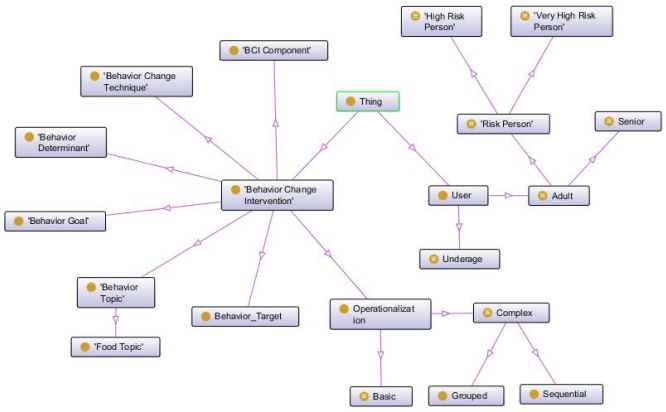


Figura 4.13: Classes da Ontologia desenvolvida e respetivas subclasses

No que consta às subclasses de cada uma, as duas classes criadas terão uma abordagem diferente à forma como as Instâncias são tratadas. No caso da classe ”*Behavior Change Intervention*”possuirá uma série de Instâncias que serão tratadas de forma objetivamente diferente, com diferenciação entre elas e no seu uso para chegar ao correto funcionamento do sistema, pelo que será necessário fazer uma subclasse para cada tipo de informação armazenada. As subclasses de ”*Behavior Change Intervention*”são assim:

- ’BCI Component’: Os componentes sobre os quais a Intervenção pode discutir. No Caso de Teste, as instâncias criadas são ”Diet”, ”Medication”e ”Physical_Activity”, para os três temas gerais que podem ser discutidos (Dieta, Medicação e Atividade Física).

- ’Behavior Change Technique’: As BCTs que podem ser aplicadas num diálogo, dependendo do Intent que for atribuído ao input do Utilizador.

- ’Behavior Determinant’: Os determinantes que podem ser encontrados pelo Dialogflow. As instâncias desta subclasse devem estar em conformidade com as Entidades que o Serviço de PLN deteta e comunica ao Núcleo do Sistema após identificação.

- ’Behavior Goal’: Os objetivos específicos de comportamento que se deseja que o Utilizador atinja.

- ’Behavior Topic’: Tópicos discutidos ao longo de cada um dos 3 Componentes. Neste caso, é possível criar também subclasses específicas para cada tópico (como é visível na Figura, com a presença da subclasse ’Food Topic’ que tem Instâncias com conselhos de alimentação saudável

- 'Behavior_Target': Utilizado para fazer a ponte entre as instâncias de 'Behavior Goal' com o 'User', funcionando um pouco como um estado de objetivo atual. No projeto apresentado apenas faz este trabalho de ponte, mas num projeto com mais tempo seria possível que abstrair mais as instâncias de 'Behavior Goal' e utilizar esta subclasse para atribuir um valor ao objetivo. Por exemplo, no caso do 'Behavior Goal' "realizar pelo menos 3 refeições principais", poderia ter apenas "realizar pelo menos [] refeições diárias"e utilizar o Behavior target para criar uma Propriedade de Objeto para o Objetivo e um valor para o número associado a esse objetivo

- 'Operationalization': Descreve em que momento e de que forma os BCTs devem ser ser desencadeados. Cada entidade saberá em que objetivo poderá ser desencadeado, o determinante que a ativa e a informação que deve ser direcionada para o utilizador. As subclasses existentes ajudam esta descrição. Existe a subclasse 'Basic' para casos onde apenas uma BCT é utilizada, e 'Complex' quando é necessário aplicar pelo menos duas BCTs. Dentro de Operacionalizações Complexas também existem duas subclasses referentes à forma de as aplicar: 'Grouped' quando todas as BCTs se unem para serem aplicadas em conjunto, e 'Sequential' quando as diversas BCTs são aplicadas de forma sequencial.

No caso da classe 'User', as subclasses funcionam de forma dependente do Reasoner ligado com o sistema. Como foi explicado anteriormente, um utilizador terá armazenado Propriedades de Dados. Estas propriedades são a idade, BMI (do inglês Body Mass Index), competência física, género, altura, nome, nível de risco e peso. Já as subclasses possuem regras sobre alguns desses valores que fazem com que a ontologia deduza que um 'User' lhe pertence. Usando o exemplo mais simples, existem as subclasses 'Adult' e 'Underage'. Para uma entidade pertencer à subclasse 'Adult' precisa de já pertencer à classe 'User' e ter uma idade igual ou superior a 18, para pertencer à subclasse 'Underage' tem de fazer parte da classe 'User' e ter uma idade inferior a 18.

Para terminar esta secção, apresentamos tabelas para ser mais fácil visualizar tanto as Classes como as Propriedades existentes na Ontologia.

Conforme visível após esta secção, a Ontologia terá todas as Classes e Propriedades que lhe permitam perceber como deve ser a tomada de decisões. No entanto, o output passado ao Utilizador não tem lugar dentro do ficheiro "BCCO.owl". Isto ocorre porque o objetivo principal da Ontologia é o de tomada de decisões, e cabe ao sistema ver o que faz com as decisões que foram tomadas. Assim sendo, é importante compreender o que é imprescindível para a tomada de decisões e o que pode ser acedido de outra forma.

A informação do utilizador, como a idade, vai influenciar a forma como BCTs são escolhidas. Deve ser uma parte presente na tomada de decisões e deve ser carregada para o ontologia quando o sistema inicia o diálogo para garantir que o Reasoner toma as decisões certas. Agora imagine-se que o Reasoner decide aplicar uma BCT. Esta decisão advém de determinantes que

Entidade	**Subclasse de**	**Descrição**
Behavior Change Intervention	Thing	Intervenção com o Objetivo de alterar o comportamento humano
BCI Component	Behavior Change Intervention	Componente a focar durante a intervenção
Behavior Technique Change	Behavior Change Intervention	Componente de uma intervenção utilizada para levar o outro indivíduo a alterar o seu comportamento
Behavior Determinant	Behavior Change Intervention	Fator que afeta o comportamento
Behavior Goal	Behavior Change Intervention	Objetivo associados a um determinado Componente
Behavior Topic	Behavior Change Intervention	Tópico abordado durante a manutenção, normalmente durante a fase de Aconselhamento
Food Topic	Behavior Topic	Um tópico de exemplo, associados ao componente de Dieta
Behavior Target	Behavior Change Intervention	Objetivo definido para tentar adotar à rotina diária
Operationalization	Behavior Change Intervention	Modo de utilização de BCTs
Basic	Operationalization	Operacionalização com apenas uma BCT
Complex	Operationalization	Operacionalização com duas ou mais BCTs
Sequential	Complex	Operacionalização complexa em que as BCTs são aplicadas uma de cada vez, de forma sequencial
Grouped	Complex	Operacionalização complexa em que as BCTs são aplicadas de uma só vez, agrupadas no diálogo
User	Thing	Utilizador/Paciente
Underage	User	Utilizador menor de idade (idade inferior a 18)
Adult	User	Utilizador Adulto (idade igual ou superior a 18)
Senior	Adult	Utilizador Senior (idade igual ou superior a 65)
Risk Person	Adult	Utilizador com Nível de Risco Associado igual ou superior a 15
High Risk Person	Risk Person	Utilizador com Nível de Risco Associado entre 15 e 20
Very High Risk Person	Risk Person	Utilizador com Nível de Risco Associado igual superior a 20

Tabela 4.1: Classes Existentes na Ontologia criada

Propriedade	Domínio	Contra-Domínio	**Descrição**
discussedDuring	Behavior Topic	BCI Component	Em que Componente é que o Tópico é discutido
hasActiveComponent	User	BCI Component	Qual o Componente Ativo para o Utilizador
hasActiveGoal	User	Behavior Goal	Qual o Objetivo Ativo para o Utilizador
hasActiveTopic	User	Behavior Topic	Qual o Tópico Ativo para o Utilizador
hasCurrentTarget	User	Behavior Target	Qual o Target Ativo para o Utilizador
relatedTo	Operationalization	Behavior Goal	Relação entre uma operacionalização e o tópico onde pode ocorrer
targetedDuring	Behavior Goal	Behavior Topic	Tópico onde o Objetivo vai ser abordado
theTargetIs	Behavior Target	Behavior Goal	Target que tem informação mais detalhada sobre o Objetivo traçado
triggeredBy	Operationalization	Behavior Determinant	Determinante que ativa a Operacionalização
triggers	Operationalization	Behavior Change Technique	BCT ativada pela Operacionalização

Tabela 4.2: Propriedades de Objetos na Ontologia criada

chegaram ao sistema aliados de objetivos já definidos. A aplicação do BCT e o que isso implica em termos de output não é necessário e vai apenas aumentar a complexidade do sistema.

Por isso, foram desenvolvidos os ficheiros ”problem_solving_det.json”e ”counselling.json”, que vêm munidos de diálogo que deve ser aplicado ao utilizador quando a Ontologia decide que deve. Por exemplo, se a ontologia passar a informação de que o tópico de hoje são rotinas alimentares, utiliza-se a classe python ”OntologyConnection”para aceder ao ficheiro ”counselling.json”, procurar o tópico escolhido e passar o diálogo para o Núcleo do Sistema.

4.7.2 Descrição da Intervenção

Como foi descrito no início desta secção, o Componente de Conhecimento tem de conseguir não só tomar decisões num espetro mais específico dentro do Diálogo, nas Fases de Diálogo que abordam os objetivos, possíveis sucessos e retrocessos do progresso realizado, e aplicação de BCTs para apoiar o Utilizador a atingir os seus objetivos, mas também no que toca à lógica da Intervenção de longo termo.

Torna-se necessário então compreender que decisões são tomadas na Intervenção. Compreender os tópicos abordados, que objetivos cada tópico tem, o que simboliza um sucesso na obtenção desses objetivos e quanto tempo se dedica por fases.

O foco fica então na primeira versão do VASelfCare, onde foi definida a totalidade da Intervenção com a duração máxima de 90 dias e uma ordem pré-determinada para abordar os três Tópicos Gerais. A Intervenção foi dividida em 4 fases, e cada fase tinha um mínimo de dias durante a qual vai abordar esse tópico. Se após passarem estes dias os objetivos ainda não tiverem sido cumpridos, a fase será prolongada até a um máximo de dias para tentar que o Utilizador adote a rotina, passando para a fase seguinte obrigatoriamente no final do máximo de dias definido para esse Tópico Geral.

Sempre que um tópico é concluído, este não será abandonado pelo sistema. A ideia é fazer o acompanhamento e garantir que o utilizador adota uma série de rotinas ao longo da totalidade da intervenção e não apenas nas datas em que se concentra nas novas rotinas. Por essa mesma razão, sempre que se avança uma fase o sistema vai começar a apresentar um novo Tópico Geral e juntar a esse os Tópicos Gerais já discutidos com o identificador ”Light”. Este identificador significa que, do ponto de vista de Diálogo, o Agente vai simplesmente relembrar rotinas que já tinham sido adotadas e confirmar se estas ainda são seguidas.

A Intervenção foi definida da seguinte maneira:

- Fase 1 - Medicação
  - Duração mínima: 11 dias
  - Se for aderente à medicação, segue para a Fase 2
  - Se não for aderente à medicação, a Fase 1 é prolongada mais 8 dias
- Fase 2 - Atividade Física; ”Light”Medicação

  - Duração mínima: 17 dias
  - Segue para a Fase 3 no final da duração mínima

- Fase 3 - Alimentação; ”Light”Medicação e Atividade Física
  - Esta fase implica a adoção de diversos comportamentos. A descrição da Intervenção para cada comportamento é a seguinte
    * Duração Mínima: 1 dia
    * Se o comportamento for adotado, segue para o comportamento seguinte
    * Se o comportamento não for adotado, repete o comportamento por mais 5 dias
  - Se um comportamento não é adotado ao fim dos 6 dias totais, este será adiado e segue para o próximo comportamento
  - Se quando se chegar ao final dos comportamentos algum tiver sido adiado, repete-se o processo anterior para os comportamentos adiados
  - Se todos os comportamentos tiverem sido adotados, segue para a Fase 4

- Fase 4 - ”Light”Alimentação, Medicação e Atividade Física
  - Duração mínima: 1 dia
  - Duração máxima total: 10 dias
  - Esta fase serve apenas para confirmar que todas as rotinas foram adotadas nos 90 dias de Intervenção
  - Caso as três primeiras fases tenham decorrido em, por exemplo, 85 dias, a duração máxima total da Fase 4 diminui para não se ultrapassar os 90 dias de Intervenção

Com base na informação necessária para cada Fase da Intervenção, compreendemos agora as informações que têm de acompanhar o Utilizador na Base de Dados. De entre os Tópicos, quais foram abordados com sucesso e quais ainda não o atingiram. Qual a Fase Atual, quantos diálogos já ocorreram desde que a Intervenção começou, e quantos diálogos já ocorreram na Fase Atual. E compreendemos também a forma de montar o ficheiro de Intervenções.

Foi decidido o desenvolvimento de um ficheiro de formato json, porque a própria estrutura consegue acomodar a descrição da Intervenção explicada. Sendo assim, é importante definir os termos que têm de estar presentes para que a lógica seja seguida. Cada intervenção terá:

- Um número de Fases porque terá de passar
- Uma duração máxima de dias
- As fases propriamente ditas

É preciso também compreender como deverão estar montadas as fases:

- O Tipo de Progressão para uma Fase seguinte. Definidas com as categorias ”Sequential”, ”Tree”, ”Loop”e ”End”
- A duração que a fase terá, em número de diálogos
- Os Componentes discutidos durante cada diálogo da fase
- As possíveis fases para que pode progredir

O Tipo de Progressão é uma adição que, apesar de aumentar o tamanho do ficheiro json, permite ser um pouco mais detalhado na forma como a Intervenção avança, reduzindo a quantidade de informação que cada fase terá. Para a Intervenção utilizada no Caso de Teste, as 4 fases descritas acabam por se transformar em 22 mais curtas. Usando como exemplos as fases apresentadas para explicar cada uma das categorias de progressão, olhemos para a Fase 1.

A Fase 1 está, no ficheiro ”interventions.json”, divido nas fases p01 e p01_b. A fase p01 é a primeira parte da Fase 1, tendo a duração de 11 dias onde o único componente discutido é o da Medicação. O seu tipo de Progressão é ”Tree”, o que significa que no final dos 11 dias este vai seguir um de dois caminho. O primeiro, em caso de sucesso, é seguir para a Fase 2. O segundo é em caso de insucesso, onde segue para a fase p01_b. Depois passamos para a descrição desta fase: duração de 8 dias, o único tema que aborda mais uma vez é Medicação. Aqui já encontramos o Tipo de Progressão ”Sequential”, o que significa que quando chegar ao final dos 8 dias apenas terá uma opção de progressão e segue para a fase 2.

Para encontrar os restantes tipos de progressão é necessário avançar até à fase 4. Esta fase, com a duração de 1 dia e que aborda de forma leve os 3 Componentes discutidos, tem o tipo de progressão ”Loop”. Com este tipo de progressão, o sistema é avisado que este loop vai ser repetido enquanto não receber ordem em contrário. Terminando o loop, apenas há uma opção de progressão. Neste caso, atingindo o último dos 90 dias, a intervenção avança para a fase ”end”, com o Tipo de Progressão com o mesmo nome e a duração de 1 dia. É o único tipo de progressão sem opções de progressão, porque marca o final da Intervenção.

4.8 Casos de Teste

Para testar a Prova de Conceito, foram criados 5 utilizadores de teste. Cada utilizador começa num ponto diferente da Intervenção, para se avaliar a progressão do diálogo e se cumpre todos os requisitos que se esperam. Os dados de cada utilizador estão colocados num ficheiro para cada, e incluem os objetivos atuais, objetivos completados e objetivos que já foram abordados mas ficaram por completar. Segue-se uma tabela com os 5 utilizadores criados:

Os objetivos que se pretendem atingir com cada um destes utilizadores diferem, apesar de estarem em pontos semelhantes da Intervenção:

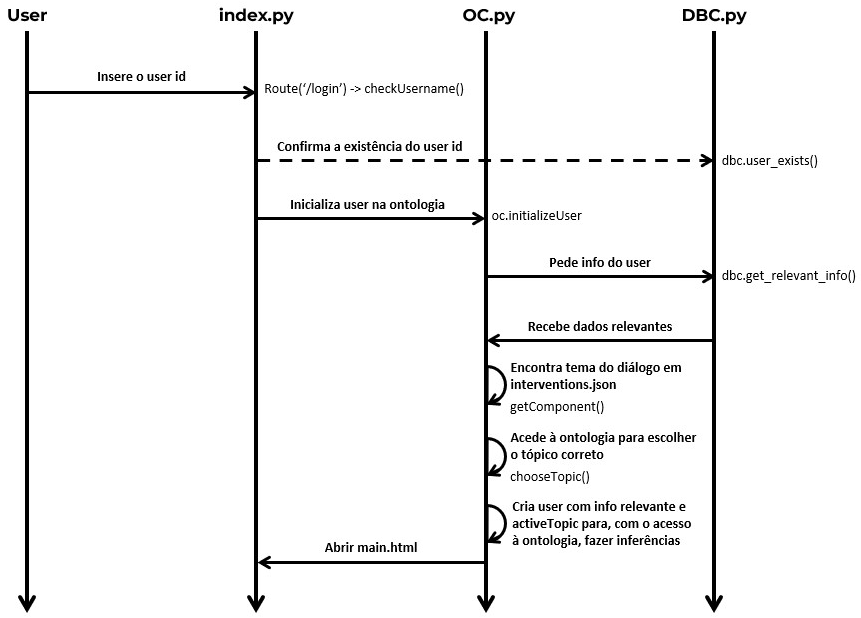


Figura 4.14: A comunicação esperada dentro do sistema, entre as diversas partes em funcionamento, no momento do log-in. O index.py funciona como central, sendo o OC.py a ligação à ontologia e o DBC.py a ligação à Base de Dados

- O user1 está a completar um Tópico com apenas um Objetivo. No final, atualiza e propõe como novo Objetivo o do Tópico seguinte - Conclui (Topic 5, Goal 1), Inicia (Topic 6, Goal 1)

- O user 2 está a completar um Tópico que tem mais do que um objetivo. No final atualiza e propõe como novo Objetivo o seguinte dentro do mesmo Tópico - Conclui (Topic 7, Goal 1), Inicia (Topic 7, Goal 2)

- O user 3 acabou de abordar o último Tópico de um Componente, mas não os completou todos. No final atualiza e propõe como novo Objetivo o primeiro que esteja por concluir - Conclui (Topic 8, Goal 1), Inicia (Topic 5, Goal 1)

- O user 4 já completou todos os Tópicos do componente, mas a fase exige que ele fique a abordar este tópico durante mais tempo. Este utilizador vai repetir os Tópicos por ordem - Conclui (Topic 5, Goal 1), Inicia (Topic 1, Goal 1)

- O user 5 está a iniciar o seu primeiro diálogo com o Agente. Toda a informação da Intervenção terá de ser carregada pelo sistema para a base de dados - Inicia (Topic 1, Goal 1)

userid	Nº Diálogo	Fase	Nº Diálogo. Fase
user1	5	p01	5
user2	7	p01	7
user3	9	p01	9
user4	10	p01	10
user5	0	—	0

Tabela 4.3: Tabela de Utilizadores dos Casos de Teste colocados na Base de Dados e progresso de cada um na Intervenção. A Fase a ser explorada, o número de diálogos que já existiram no total e o número de diálogos na Fase Atual

userid	Objetivo Atual	Objetivos Cumpridos	Objetivos Iniciados por Completar
user1	(Diet, 5, 1)	(1,1),(2,1),(4,1)	(3,1)
user2	(Diet, 7, 1)	(1,1),(2,1),(3,1),(4,1),(5,1),(6,1)	—
user3	(Diet, 8, 1)	(1,1),(2,1),(3,1),(4,1),(6,1),(7,1),(7,2)	(5,1)
user4	(Diet, 5, 1)	(1,1),(2,1),(3,1),(4,1)	—
user5	—	—	—

Tabela 4.4: Tabela de Utilizadores dos Casos de Testecolocados na Base de Dados e progresso de cada um na Intervenção. Objetivos estão aqui representados pelo tuplo (Component, Topic, Goal)

4.9 Sumário

Neste capítulo foi desenvolvida uma Prova de Conceito para a Arquitetura desenhada em capítulos anteriores. Foram apresentadas as funcionalidades implementadas, bem como as soluções encontradas em cada uma das Camadas.

Entre as soluções encontramos o fluxo de comunicação do sistema durante o diálogo, a criação de uma Base de Dados onde se guarda informação sobre o Utilizador e o seu progresso ao longo dos diálogos, e a forma como a Intervenção pode ser representada e acedida no Componente de Conhecimento.

Foram também apresentados alguns Utilizadores que funcionarão como exemplos para testar o sistema, bem como a razão para a existência de cada um deles. Estes exemplos permitem mostrar, acima de tudo, o funcionamento do sistema quando tem de seguir um rumo de diálogo um pouco mais complexo do que simplesmente seguir as Fases do Diálogo de Bickmore do início ao fim.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Conclusões

No primeiro capítulo foi evidenciado que, apesar de Agentes Conversacionais serem uma tecnologia já não muito recente e da existência de aplicações mHealth mostrarem resultados interessantes no acompanhamento de pacientes a atingirem objetivos e a adotarem rotina, não existia ainda uma Plataforma que tivesse avaliado o que é necessário para desenvolver um Agente Conversacional, dotado de proatividade, que fizesse o acompanhamento a pacientes com o objetivo de os ajudar a adotar novas rotinas.

O objetivo principal deste projeto era o de desenhar a arquitetura e desenvolver uma Plataforma de Desenvolvimento de Agentes Conversacionais Proativos e a respetiva Metodologia para criar os ditos agentes, e foi alcançado com uma arquitetura divida em três camadas: o Núcleo de Processamento, o Serviço de Processamento de Linguagem Natural e o Componente de Conhecimento.

Foi depois também desenvolvida uma Prova de Conceito. Utilizando bases desenvolvidas anteriormente para a aplicação VASelfCare, foi incorporada uma camada de abstração. Isto significa que apesar do sistema ter sido construído com o objetivo específico de acompanhar pacientes com Diabetes Tipo 2 que estão a tentar adotar novas rotinas comportamentais para o seu dia a dia, é possível que o Agente faça intervenções sobre qualquer tema sem alterar a base deste.

O ponto fulcral para conseguir esta camada de abstração situa-se no Componente de Conhecimento. A Ontologia desenvolvida que permite que se criem instâncias sobre qualquer assunto dentro das classes existente e utilizando as Propriedades de Objetos criadas, sendo apenas necessário adaptar possíveis Propriedades de Dados ou de Anotações para que a Ontologia esteja pronta a discutir o que o *developer* desejar.

Foi também desenhada uma representação da Intervenção de longo termo, sendo possível utilizar as mesmas bases para desenhar intervenções de qualquer duração, com as fases que o *developer* necessitar e com um fluxo próprio adaptável às necessidades que o tema traga. Para atingir os objetivos propostos, acredito que a Prova de Conceito foi bem sucedida.

Pessoalmente e em termos de competências técnicas, este projeto permitiu-me explorar de forma mais profunda diversas áreas e conhecer ferramentas com que não teria interagido de outra forma. Se por um lado pude trabalhar mais a fundo numa aplicação como o Protégé e aprimorar o meu conhecimento de ontologias, para além da prática que me deu a programar em Python, este projeto também me permitiu conhecer uma ferramenta como é o Dialogflow e conhecer o potencial deste tipo de serviço.

5.2 Trabalho Futuro

Considerando a Prova de Conceito construída uma base para desenvolvimento futuro, é possível compreender o caminho que este projeto poderá levar a partir deste ponto:

- Testes com utilizadores - Para conseguir compreender o quão eficaz a utilização de BCTs é utilizando este método e conseguir também a partir daí tirar conclusões obre como prosseguir o desenvolvimento

- Embodied Conversational Agents (ECAs) - A utilização de ECAs tem dado provas de conseguir resultados positivos, e a sua implementação nesta versão poderia ser muito positiva. Esta implementação não viria sem os seus desafios: a única vez que um ECA foi utilizado no VASelfCare foi na primeira versão, quando a interação era feita com o uso de botões e todas as animações do avatar poderiam ser programadas especificamente para cada linha de Diálogo que sabiam que seria utilizada. Com a implementação de comunicação em Linguagem Natural poderia complicar a sua utilização. O segundo desafio é que, mantendo a utilização do Dialogflow como Serviço de PLN, este não tem a capacidade para lidar diretamente com uma aplicação como o Unity. Seria necessário desenvolver a Camada de Apresentação para tratar desta parte.

- Componente de Conhecimento - A Prova de Conceito foca-se nos tópicos e objetivos já estudados e com provas dadas, mas a base de conhecimento é extremamente limitada para o potencial que demonstra. Seria necessário aprofundar o conhecimento presente na ontologia junto de profissionais de saúde, se o tema for o mesmo, ou com um profissional da área em que se pretende desenvolver a ontologia.

- Ficheiro de Intervenção - Da forma como estão desenhados, estes ficheiros permitem criar Intervenções bastante personalizadas. No entanto, estas têm de ser construídas por pessoas com conhecimentos de ficheiros json que, apesar de serem relativamente simples de compreender, não deverá ser a maioria dos profissionais de saúde que sabem realmente como deve ser feita a intervenção. Por um lado, seria importante confirmar os dados que as Intervenções têm de levar consigo na informação detalhada. Por outro, seria interessante a construção de uma ferramenta que permitisse criar uma intervenção de forma mais gráfica e intuitiva, e que depois fizesse a exportação no formato certo para fazer parte do

Componente de Conhecimento. Mesmo que o desenvolvimento deste ficheiro continuasse a ser feito por *developers*, facilitaria a construção de Intervenções com um grau de complexidade maior.

- Objetivos a atingir - Apesar de estar semipreparado para poder personalizar os objetivos com valores mais específicos, seria necessário construir um sistema (mesmo dentro da ontologia se se achasse melhor) com possíveis valores, e tentar atribuir os valores corretos para se apresentarem ao Paciente.

- Ética - Um sistema deste género poderia ser utilizado para aplicar BCTs sobre qualquer tema que se deseje, e a única coisa que impede o *developer* de criar um sistema que partilhe informação falsa é ele próprio. Juntando a isto o crescimento exponencial de Inteligência Artificial nos últimos anos e as questões de Proteção de Dados levantadas ao longo da última década e um sistema assim tanto pode ser visto com desconfiança, como pode ser adotado e utilizado da forma errada. Inicialmente de um ponto de vista mais teórico, a exploração de princípios éticos e tentar compreender como os incorporar na Metodologia do sistema de forma prática é um trabalho que deve ser feito.

Bibliografia

[1] Muhammad Amith, Rebecca Z. Lin, Licong Cui, Dennis Wang, Anna Zhu, Grace Xiong, Hua Xu, Kirk Roberts, and Cui Tao. Conversational ontology operator: patient-centric vaccine dialogue management engine for spoken conversational agents. *BMC Medical Informatics and Decision Making*, 20, 12 2020.

[2] Ali Balapour, Iris Reychav, Rajiv Sabherwal, and Joseph Azuri. Mobile technology identity and self-efficacy: Implications for the adoption of clinically supported mobile health apps. *International Journal of Information Management*, 49:58–68, 2019.

[3] João Balsa, Isa Félix, Ana Paula Cláudio, Maria Beatriz Carmo, Isabel Costa e. Silva, Ana Guerreiro, Maria Guedes, Adriana Henriques, and Mara Pereira Guerreiro. Usability of an intelligent virtual assistant for promoting behavior change and self-care in older people with type 2 diabetes. *Journal of Medical Systems*, 44, 7 2020.

[4] João Balsa, Pedro Neves, Isa Félix, Mara Pereira Guerreiro, Pedro Alves, Maria Beatriz Carmo, Diogo Marques, António Dias, Adriana Henriques, and Ana Paula Cláudio. Intelligent virtual assistant for promoting behaviour change in older people with t2d. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11804 LNAI, pages 372–383. Springer Verlag, 2019.

[5] Maria Bastos, Ana Cláudio, Isa Félix, Mara Guerreiro, Maria Carmo, and João Balsa. Operationalizing behavior change techniques in conversational agents. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, pages –, Online Streaming, 2022. SCITEPRESS - Science and Technology Publications.

[6] Maria Inês Teixeira Bastos. Operacionalização de técnicas de mudança comportamental em agentes conversacionais. Master's thesis, FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DE LISBOA, 2022.

[7] Jerome R Bellegarda. Spoken language understanding for natural interaction: The siri experience. *Natural Interaction with Robots, Knowbots and Smartphones: Putting Spoken Dialog Systems into Practice*, pages 3–14, 2013.

[8] Timothy Bickmore and Toni Giorgino. Health dialog systems for patients and consumers. *Journal of biomedical informatics*, 39(5):556–571, 2006.

[9] Timothy W. Bickmore, Daniel Schulman, and Candace L. Sidner. A reusable framework for health counseling dialogue systems based on a behavioral medicine ontology. *Journal of Biomedical Informatics*, 44:183–197, 4 2011.

[10] Steven Birkmeyer, Bernd W Wirtz, and Paul F Langer. Determinants of mhealth success: An empirical investigation of the user perspective. *International Journal of Information Management*, 59:102351, 2021.

[11] Susana Buinhas. Assistente virtual para facilitar o autocuidado de pessoas mais velhas com diabetes tipo 2, 2018.

[12] Daniel Jurafsky. Speech and language processing, 2000.

[13] Daniel Jurafsky and James H Martin. Speech and language processing an introduction to natural language processing, computational linguistics, and speech recognition third edition draft, 2020.

[14] Krishna Prakash Kalyanathaya, D Akila, and P Rajesh. Advances in natural language processing–a survey of current research trends, development tools and industry applications. *International Journal of Recent Technology and Engineering*, 7(5C):199–202, 2019.

[15] James Lester, Karl Branting, and Bradford Mott. Conversational agents. *The practical handbook of internet computing*, pages 220–240, 2004.

[16] Susan Michie, Lou Atkins, and Heather L. Gainforth. *Changing Behaviour to Improve Clinical Practice and Policy*, pages 41–60. Axioma - Publicações da Faculdade de Filosofia, 12 2016.

[17] Susan Michie, Michelle Richardson, Marie Johnston, Charles Abraham, Jill Francis, Wendy Hardeman, Martin P. Eccles, James Cane, and Caroline E. Wood. The behavior change technique taxonomy (v1) of 93 hierarchically clustered techniques: Building an international consensus for the reporting of behavior change interventions. *Annals of Behavioral Medicine*, 46:81–95, 8 2013.

[18] Pedro Miguel Lemos das Neves. Vaselfcare-componente de diálogo inteligente, 2019.

[19] Lazlo Ring, Barbara Barry, Kathleen Totzke, and Timothy Bickmore. Addressing loneliness and isolation in older adults: Proactive affective agents provide better support, 2013.

[20] Simon P Rowland, J Edward Fitzgerald, Thomas Holme, John Powell, and Alison McGregor. What is the clinical value of mhealth for patients? *NPJ digital medicine*, 3(1):4, 2020.

[21] Silke Ter Stal, Lean Leonie Kramer, Monique Tabak, Harm op den Akker, and Hermie Hermens. Design features of embodied conversational agents in ehealth: a literature review. *International Journal of Human-Computer Studies*, 138:102409, 2020.

[22] Elena Vlahu-Gjorgievska, Suliman Basahal, Kamana Pokharel, and Khin Than Win. mhealth applications for childhood cancer support and self-management: Persuasive systems design features, 2021.

[23] Joseph Weizenbaum. Eliza - a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9:36–45, 1966.

[24] Taylor Willmott and Joy Parkinson. Motivation, opportunity, and ability: Understanding new habits and changes adopted for weight management. *International Journal of Consumer Studies*, 41(3):291–298, 2017.

Apêndice A

Diagramas de Comunicação do Sistema

Para cada ação que o sistema realiza, foram construídos diagramas para compreender melhor a comunicação interna do momento em que a ação é chamada até que é dada como completa. A comunicação no momento de log-in está disponível na secção 4.7 do documento. Aqui são colocados os restantes diagramas construídos. Os ficheiros referidos são:

- index.py - O núcleo do sistema, que faz a ligação entre todas as camadas
- OC.py - OntologyConnection.py, trata da ligação entre o núcleo e a ontologia para garantir que o diálogo está a seguir os pontos que é suposto
- DBC.py - DBConnection.py, trata da ligação entre o núcleo e a base de dados que guarda os dados sobre o utilizador e o seu progresso ao longo da Intervenção
- interventions.json - ficheiro com os dados referentes à Intervenção e ao formato que ela segue nos diálogos que a compõem

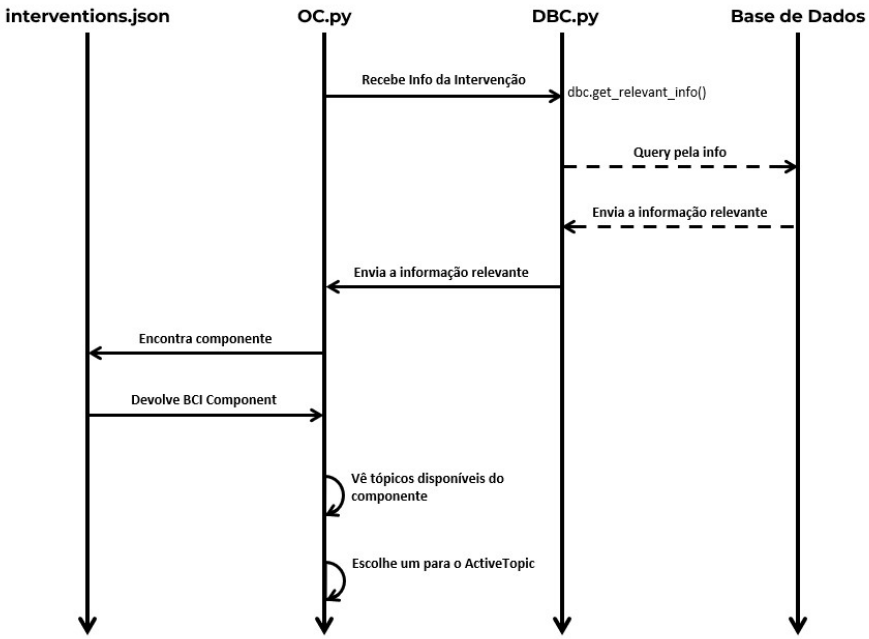


Figura A.1: Comunicação no momento de escolher na ontologia o Tópico a ser discutido.

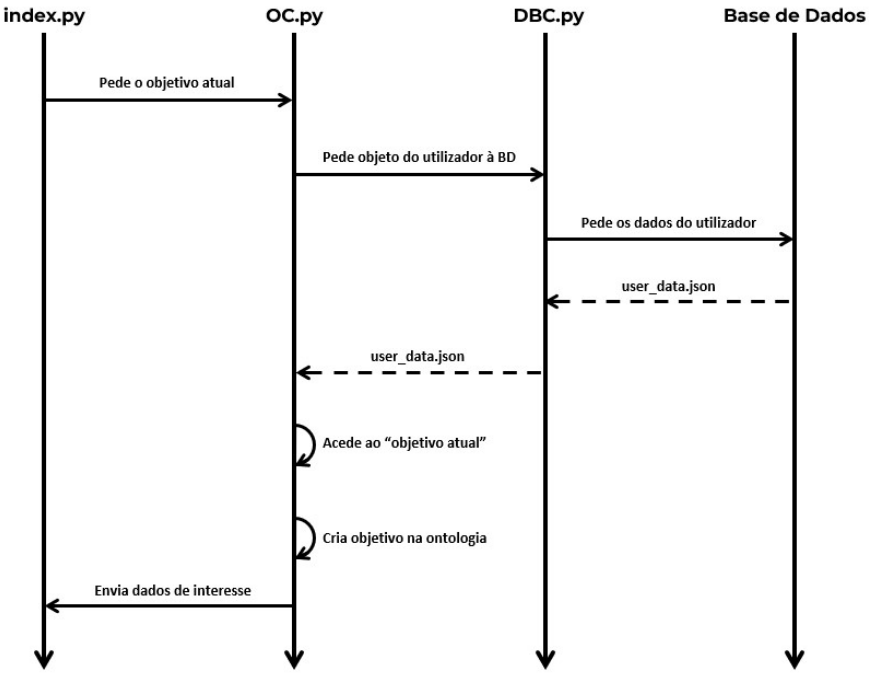


Figura A.2: Comunicação no momento em que é necessário perguntar ao utilizador se cumpriu as tarefas impostas anteriormente.

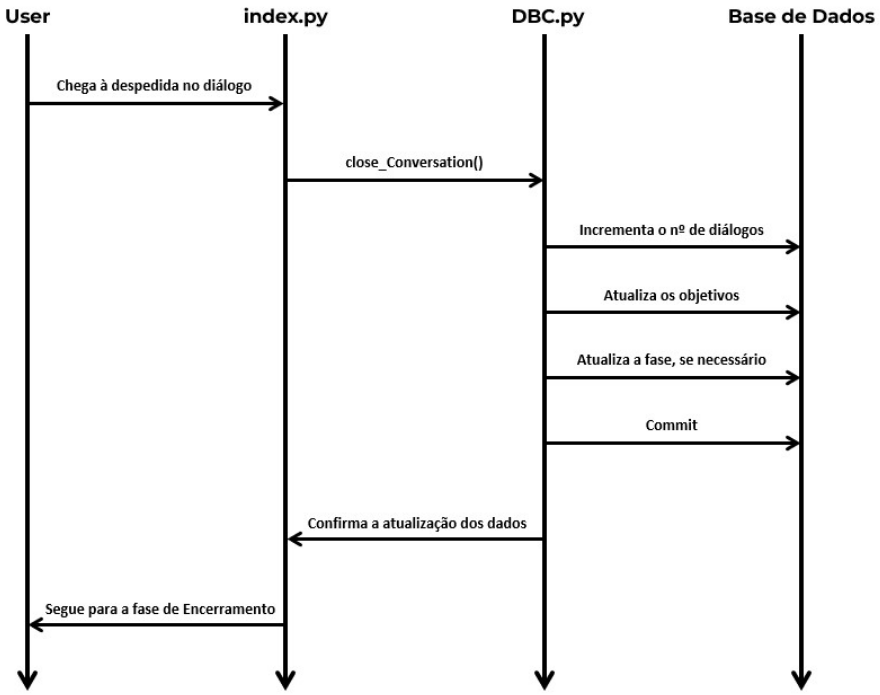


Figura A.3: Comunicação no momento em que é necessário atualizar a Base de Dados com as decisões tomadas para o diálogo seguinte.

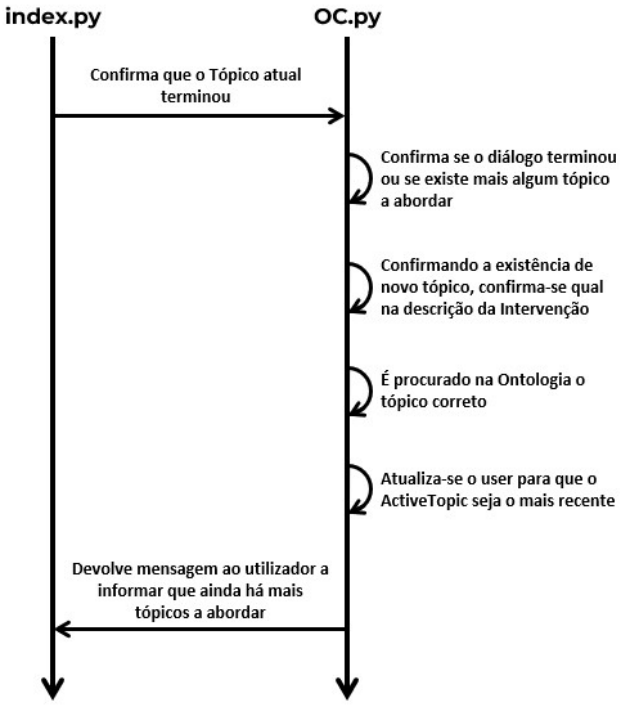


Figura A.4: Comunicação quando o sistema acaba de discutir um tópico com o utilizador mas ainda tem outros para discutir no mesmo diálogo.

Apêndice B

Construção do ficheiro interventions.json

Na secção 4.7.2 foi definida a intervenção, e descrito como esta foi construída no documento. Em anexo seguem algumas capturas de ecrã exemplificativas das decisões tomadas no ficheiro interventions.json.

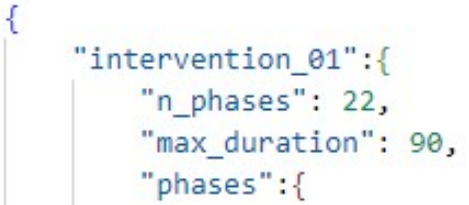
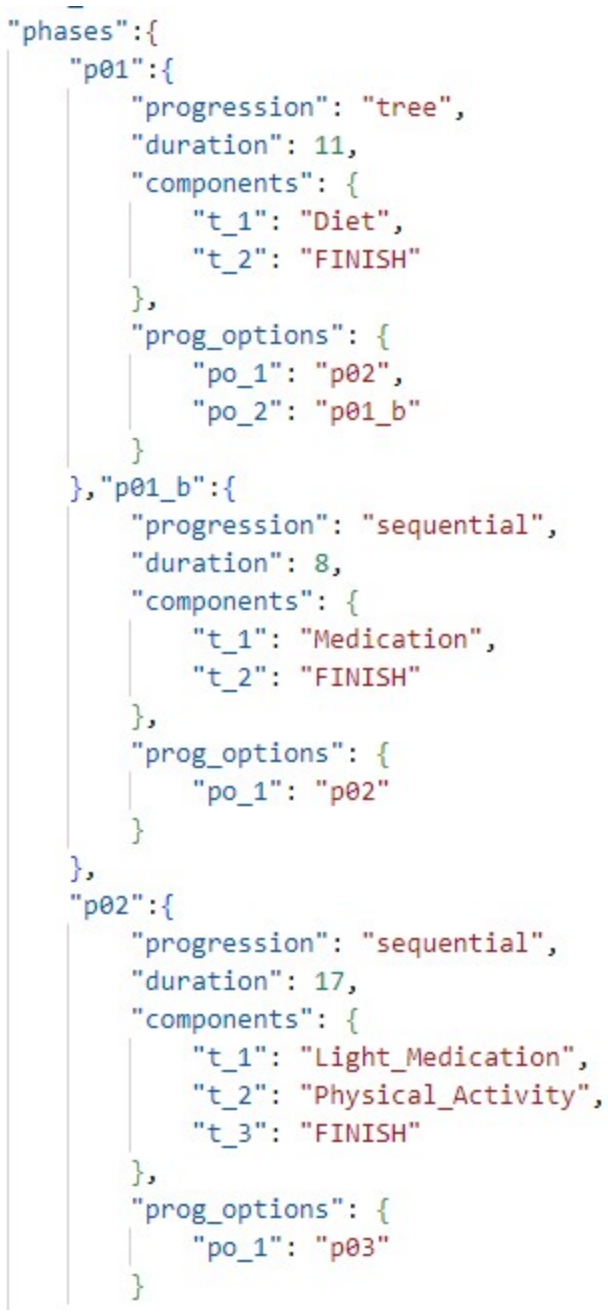


Figura B.1: No início do ficheiro é descrita a duração da Intervenção. Diz-se o número de fases existentes e a duração máxima da intervenção. Cada fase tem a duração mínima de 1 diálogo, mas pode prolongar-se.



```
"phases":{
    "p01":{
        "progression": "tree",
        "duration": 11,
        "components": {
            "t_1": "Diet",
            "t_2": "FINISH"
        },
        "prog_options": {
            "po_1": "p02",
            "po_2": "p01_b"
        }
    },"p01_b":{
        "progression": "sequential",
        "duration": 8,
        "components": {
            "t_1": "Medication",
            "t_2": "FINISH"
        },
        "prog_options": {
            "po_1": "p02"
        }
    },
    "p02":{
        "progression": "sequential",
        "duration": 17,
        "components": {
            "t_1": "Light_Medication",
            "t_2": "Physical_Activity",
            "t_3": "FINISH"
        },
        "prog_options": {
            "po_1": "p03"
        }
```

Figura B.2: Cada fase existente contém o tipo de progressão, a duração em número de diálogos que vai durar, os componentes da Ontologia que aborda, e as opções de progressão. Nesta imagem vemos o tipo de produção ”*Tree*”, que no final do tempo da duração vai decidir para que fase segue entre as opções, e ”*Sequential*”, que no final da duração simplesmente progride para a fase dada.

```
"p04":{
    "progression": "loop",
    "duration": 1,
    "components": {
        "t_1": "Light_Medication",
        "t_2": "Light_Physical_Activity",
        "t_3": "Light_Diet",
        "t_4": "FINISH"
    },
    "prog_options":{
        "po_1": "end"
    }
},
```

Figura B.3: O tipo de progressão ”*Loop*”tem a duração de 1 diálogo, e vai repetir-se até existir ordem em contrário. No caso do projeto atual, é a fase que repete quando todos os objetivos foram atingidos mas ainda não se cumpriu a duração máxima da intervenção, e vai fazer um acompanhamento leve dos componentes discutidos para garantir que o paciente continua a cumprir a sua rotina.

```
"end":{
    "progression": "end",
    "duration": 1,
    "components": {
        "t_1": "Light_Medication",
        "t_2": "Light_Physical_Activity",
        "t_3": "Light_Diet",
        "t_4": "FINAL_GOODBYE",
        "t_5": "FINISH"
    }
}
```

Figura B.4: O tipo de progressão ”*End*”marca o final da Intervenção, sendo assim o único tipo de progressão que não contém opções de progressão.

```
"p03_7a":{
    "progression": "tree",
    "duration": 1,
    "components": {
        "t_1": "Light_Medication",
        "t_2": "Light_Physical_Activity",
        "t_3": "Diet",
        "t_4": "FINISH"
    },
    "prog_options": {
        "po_1": "p03_8a",
        "po_2": "p03_7b"
    }
},
"p03_7b":{
    "progression": "sequential",
    "duration": 5,
    "components": {
        "t_1": "Light_Medication",
        "t_2": "Light_Physical_Activity",
        "t_3": "Diet",
        "t_4": "FINISH"
    },
    "prog_options": {
        "po_1": "p03_8a"
    }
},
"p03_8a":{
    "progression": "tree",
    "duration": 1,
    "components": {
        "t_1": "Light_Medication",
        "t_2": "Light_Physical_Activity",
        "t_3": "Diet",
        "t_4": "FINISH"
    },
    "prog_options": {
        "po_1": "p03_9a",
        "po_2": "p03_8b"
    }
},
```

Figura B.5: Durante a Intervenção pode ser necessário aumentar ou diminuir a duração que certos tópicos são discutidos. Aqui observamos um momento em que certos tópicos relacionados com Dieta são trabalhados com o paciente na fase ”p03_7a”. Caso o paciente atinja certos objetivos com sucesso este progride para a fase ”p03_8a”onde abordará novos objetivos. Caso o paciente não os atinja, a Intervenção progrida para a fase ”p03_7b”onde os vai abordar durante mais 5 diálogos antes de continuar o progresso.