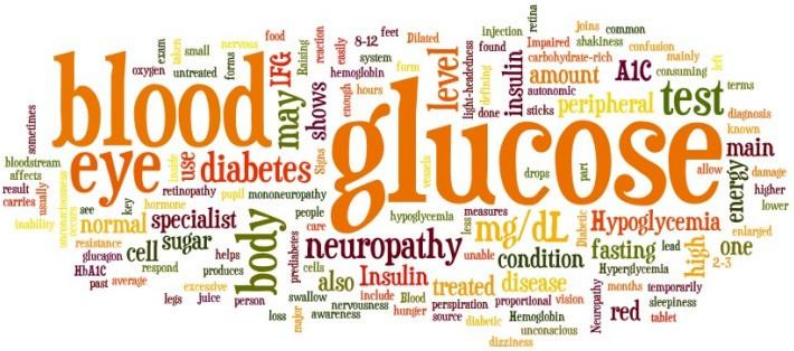

Apoio ao Controlo Diabético Baseado em Registos e Protocolos Médicos

Diogo Roberto de Melo e Diogo Machado
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2015

Orientador
Pedro Brandão, Professor Doutor, Faculdade de Ciências da Universidade do Porto

Coorientador
Vítor S.Costa, Professor Doutor, Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Resumo

As aplicações para dispositivos móveis invadiram o mercado e têm tido um grande sucesso. Destas aplicações, poucas são as que tentam ajudar o utilizador no seu dia-a-dia e ainda menos são as que tentam ajudar o utilizador com os seus problemas de saúde.

Uma doença que se tornou uma grande preocupação para os países do «mundo moderno» é a Diabetes. É uma doença crónica, cujos portadores se caracterizam por ter níveis elevados de açúcar no sangue, ou seja, de glicose. Estes níveis elevados devem-se à incapacidade do seu organismo os regular.

Tendo em conta que hoje em dia é comum o uso de *smartphones*, a implementação deste projeto foi pensada para estar disponível como uma aplicação em qualquer telemóvel com suporte *Android*.

Este projeto tenta, usando protocolos médicos e informação dada por parte do utilizador, a partir de **factos** *Prolog*, e aconselhá-lo de forma a superar diversas situações que ocorrem na vida de um Diabético. Este aconselhamento é feito a partir mensagens que são exibidas no telemóvel do utilizador, que indicam o que fazer a seguir.

Dado o apertado controlo que um Diabético necessita, foi pensado um sistema de lista de tarefas com diferentes níveis de urgência para que o utilizador esteja sempre consciente das diferentes atividades que tem de executar. O utilizador da aplicação é guiado de forma a conseguir viver um dia-a-dia mais saudável e longe de cenários de crise característicos de um doente Diabético.

Abstract

Mobile applications invaded the market and have been a huge success. From the available applications very few try to help the user in his daily life, and from these, even fewer try to help with user's health problems.

A disease that has become a major concern in the "modern world" countries is Diabetes. This is a chronic disease whose afflicted suffer from high levels of sugar in their blood, in other words they have high levels of glucose. These high levels result from not being able to regulate the level of glucose.

Since we now live in a world where smartphone usage is common, the implementation of this project was made into an application available to any android device.

This project tries to give advice in order to allow the user to get over multiple situations that happen in a Diabetic's life. This functionality uses medical protocols and information given by him. These advice are given in the form of messages displayed on the phone, that show what the user should do next.

Considering that most Diabetics have a tight control around their daily routine, we implemented a "to do list" that has different levels of urgency so that the user can be aware of the different activities that he has to perform. With this in mind, the user of this application is guided in a way that allows him to have a healthy daily life, far from the critical scenarios of Diabetic people.

Agradecimentos

A elaboração desta dissertação e projeto não seria possível sem a preciosa ajuda e apoio que obtive. Por isso quero aproveitar este ponto para expressar o meu profundo agradecimento.

Ao Professor Doutor Pedro Brandão queria agradecer a sua orientação e apoio ao longo do desenvolvimento do projeto e dissertação.

À Susana Ribeiro por toda a paciência e apoio para comigo, por todas as revisões para que esta dissertação fosse entregue com qualidade e rigor linguístico.

Ao Artur Ferreira agradeço a sua polivalência no apoio aos mais variados problemas em programação. Muitos engenheiros devem muito ao *stackoverflow.com*, mas tu consegues estar acima deste *website*.

Por fim quero agradecer à minha família, em especial aos meus pais, tudo o que fizeram por mim.

Quero dedicar esta tese aos meus pais, que fizeram de mim quem sou.

Conteúdo

Lista de Figuras

Lista de Blocos de Código

Glossário

Cetoacidose Condição grave que pode resultar em coma ou mesmo morte. Na ausência de insulina o nível de açúcar no sangue sobe e desenvolve-se um défice de energia a nível celular. Para manter as células a funcionar o corpo, como medida de emergência, começa a produzir energia a partir da gordura presente no corpo. Neste processo são geradas Cetonas, um composto orgânico, que se acumulam no sangue e que em grandes quantidades podem envenenar o corpo.

Coma Hiperosmolar Hiperglicémico Complicação da Diabetes *Mellitus* (predominantemente do tipo 2), na qual o elevado nível de açúcar no sangue causa desidratação acentuada, o aumento da osmolaridade e risco elevado de complicações, coma e morte.

Espaço Intersticial Espaço existente na estrutura de um órgão ou tecido orgânico.

Gastroparesia Na gastroparesia, o estômago apresenta dificuldade em expelir o que se encontra dentro dele, isto sem evidências de obstrução mecânica. Por não existir um fator obstrutivo mecânico, entende-se que há uma dificuldade de contratilidade da parede gástrica.

Hiperglicémia Condição quando os níveis de glicose no sangue estão acima do normal mais de 99 mg/dL pré-prandial (antes de comer) e mais de 140 mg/dL pós-prandial (depois de comer).

Hipoglicémia Condição quando os níveis de glicose no sangue estão abaixo do normal ou seja menos que 70 mg/dL.

Inanição estado em que a pessoa se encontra extremamente enfraquecida.

Sépsis infeção geral grave do organismo causado por germes patogénicos.

Capítulo 1

Introdução

O organismo humano necessita de açúcar no sangue (glicose) para que possa produzir energia, e assim funcionar. Esta glicose precisa de ser processada para que possa ser utilizada pelas células do nosso corpo. Este processo está dependente da hormona insulina, normalmente produzida pelo próprio corpo, pelo pâncreas.

Quando uma pessoa é diabética o seu nível de açúcar no sangue (glicemia) tem subidas anormais e descontroladas, devido a uma produção insuficiente de insulina ou devido à sua ação ser ineficaz.

Existem vários tipos de *Diabetes Mellitus*, sendo os principais tipos:

- *Diabetes Mellitus* Tipo 1 - neste tipo de Diabetes o doente é incapaz de produzir insulina. Para viverem precisam, obrigatoriamente, de administrar insulina. É um tipo de Diabetes que surge geralmente nos mais jovens. A incapacidade de produção de insulina deve-se à destruição das células responsáveis pela produção desta hormona no pâncreas. Esta destruição é feita pelo próprio organismo da pessoa, daí se considerar a Diabetes do Tipo 1 uma doença auto imune. Desconhece-se o porquê do organismo atuar contra ele mesmo, mas pensa-se que é devido à herança genética combinada com estímulos ambientais, como por exemplo infeções [9].

- *Diabetes Mellitus* Tipo 2 - este tipo de Diabetes é o mais comum. Estima-se que nove em cada dez diabéticos pertencem ao Tipo 2. Este tipo de Diabetes caracteriza-se pela menor quantidade e ineficiência da insulina produzida. A gestão desta Diabetes é feita a partir de comprimidos e com uma mudança de estilo de vida por parte do doente. A ineficiência da insulina produzida começa muito antes de a pessoa sofrer de Diabetes. Devido ao estilo de vida da pessoa e devido à herança genética, o organismo, com o passar do tempo, vai ficando resistente à insulina. Isto obriga o pâncreas a um esforço extra, pois tem de produzir mais insulina para compensar a resistência das células. Desta forma, o organismo vai sendo regulado a partir dos esforços do pâncreas, porém chega a uma altura em que o pâncreas começa a falhar e deixa de produzir a insulina necessária. É neste momento

que os valores glicémicos disparam, o organismo fica descontrolado, e a pessoa torna-se diabética. Não se sabe ao certo as razões para o pâncreas começar a falhar, porém sabe-se que o excesso de peso, o aumento da gordura no organismo e a inatividade física aumentam a resistência à insulina.

- Diabetes Gestacional - tal como o nome indica, esta Diabetes pode ocorrer durante as últimas fases da gravidez e geralmente desaparece com o nascimento do bebé. As mulheres com este tipo de Diabetes têm uma grande probabilidade de vir a desenvolver Diabetes do Tipo 2 numa fase já mais avançada da sua idade. Ao contrário dos outros tipos de Diabetes, a Diabetes gestacional, desde que acompanhada, não causa complicações nem à mãe nem ao filho. A mãe deve no entanto ter em atenção a sua saúde dado que mães que tiveram uma gravidez com Diabetes gestacional têm risco elevado de vir a contrair Diabetes do Tipo 2.

- Diabetes Auto-imune Latente do Adulto (LADA) - este tipo de Diabetes encontra-se entre a Diabetes Mellitus Tipo 1 e Tipo 2, sendo por esta razão conhecido como Diabetes de Tipo 1,5. Tal como a Diabetes do Tipo 1 existe, eventualmente, uma falha total na produção de insulina por parte do pâncreas. Porém, este tipo de Diabetes tem a particularidade de que o doente, normalmente, não necessita de insulina durante meses, ou até mesmo durante anos, mesmo depois de lhe ser diagnosticada a doença. Isto deve-se ao desenvolvimento gradual da doença. Esta doença surge em pessoas com idades acima dos trinta anos e apresenta sintomas característicos, tanto de Diabetes de Tipo 1, como de Diabetes de Tipo 2 o que leva à comunidade médica, muitas vezes, diagnosticar erradamente os doentes com Diabetes do Tipo 2 [17].

- Outros Tipos Específicos - para além destes tipos de Diabetes já referidos, existem ainda outros com diferentes causas. Estas causas têm origem em medicamentos, doenças, traumatismos abdominais graves e formas genéticas raras, porém são casos muito mais raros que os referidos.

A Diabetes por si só é suficientemente grave para fazer com que as pessoas tomem cuidado e controlem a sua glicose. Porém, para além dos problemas referidos, a Diabetes ainda leva a mais patologias se não for controlada.

A diminuição do fluxo sanguíneo, provocada pela Diabetes, os doentes têm mais dificuldades regenerativas. Isto faz com que uma simples ferida no pé possa levar a uma amputação. Deve haver um maior cuidado com lesões e no tratamento destas, isto porque os diabéticos também sofrem de neuropatia, ou seja, sofrem de uma redução da sensibilidade nas extremidades do corpo (mãos e pés). Estes fatores, quando combinados com baixos cuidados higiénicos, podem levar a uma Miíase, por outras palavras, podem levar ao aparecimento de larvas de mosca que infestam uma ferida e se alimentam do tecido em decomposição. Tudo isto aliado ao facto de os doentes em causa se encontram com dificuldades regenerativas, faz do problema algo ainda mais grave.

A presença de quantidades altas de glicose no sangue pode, a longo prazo, levar a problemas como insuficiência renal, pressão arterial alta, Cetoacidose, Gastroparesia, cegueira, Acidente Vascular Cerebral e Coma Hiperosmolar Hiperglicémico [10].

Um diabético, para levar uma vida normal, deve controlar vários aspetos da sua vida, desde o que come, as doses de insulina que toma e até o exercício físico que faz. Deve ter todos estes tipos de cuidados, porque todos estes fatores influenciam o valor glicémico.

A aplicação proposta no âmbito desta dissertação tenta, para além de ser uma simples aplicação de registos, aconselhar o utilizador no que fazer ao longo do dia, baseando-se em protocolos médicos, utilizando alarmes, conselhos e a partir de uma lista de tarefas onde o utilizador é relembrado das atividade que deve fazer. Esta aplicação usa como base a aplicação *Android MyDiabetes*, uma aplicação dirigida à população diabética, desenvolvida pelo aluno João Graça da Faculdade de Ciências da Universidade do Porto, que auxilia o utilizador a efetuar registos comuns ao doente diabético entre outras funcionalidades úteis. A esta aplicação foi acrescentado um sistema de aconselhamento baseado em regras lógicas. O objetivo deste sistema é evoluir a aplicação *MyDiabetes* de maneira a conseguir fazer um melhor acompanhamento do diabético, ajudando-o e aconselhando-o em diversas situações do seu dia-a-dia.

A aplicação original é composta por uma primeira vertente de base de dados de registos, onde temos presente todas as componentes gerais de uma aplicação, que tenta ajudar os diabéticos. Nela o doente pode inserir os seus registos da glicose, insulina, exercício físico, refeições, e outras informações relevantes à avaliação do estado de saúde do utilizador.

A segunda vertente do sistema, objeto desta dissertação, é de análise de dados a partir de um sistema de regras, baseado em protocolos médicos. Desta maneira é possível aconselhar o utilizador a mudar hábitos e a ter certos cuidados para prevenir crises ou outras situações graves.

O conjunto de regras para avaliação do estado do utilizador estará embutido na aplicação. Este conjunto é constituído por regras que, a partir dos dados do utilizador, avalia o seu estado e tenta aconselhá-lo de acordo com a situação. As regras lógicas do programa não são intemporais. A ciência evolui todos os dias e é esperado que no futuro sejam desenvolvidas mais, e possivelmente, melhores regras para ajudar os utilizadores desta aplicação.

Com o evoluir do sistema de regras, será preciso atualizar o programa para que trabalhe com a versão mais atual. Quando o utilizador liga a aplicação, esta deve detetar que existe uma nova versão do sistema de regras e atualizar logo que possível, não sendo preciso o utilizador preocupar-se em manter o sistema na sua última versão.

O sistema de regras que o programa vai usar foi desenvolvido na linguagem Prolog, uma linguagem de programação lógica que, não só traduz facilmente as regras médicas, como também permite desempenhos superiores às convencionais linguagens de programação na avaliação de regras.

Este sistema já demonstrou ter potencial ao conseguir ser eleito para ser exibido na Conftele

2015 [13], uma importante Conferência de Telecomunicações que decorreu em Aveiro, como póster [20].

Capítulo 2

Estado da Arte

Neste capítulo irá ser analisado o que existe na área de aconselhamento de portadores da doença de Diabetes e que aplicações já se encontram disponíveis para auxiliar a gestão da Diabetes.

2.1 Sistemas de Aconselhamento

Este tipo de sistemas está baseado numa comunicação direta paciente-médico. Esta comunicação pode ser feita diretamente num consultório, por telefone ou mesmo por vídeo-conferência. Embora seja um método distante do proposto nesta tese, é importante avaliar este método que, até à poucos anos com a criação de *smartphones*, era a única ajuda disponível ao diabético e ainda hoje tem um papel fundamental na gestão e prevenção da Diabetes.

2.1.1 Linha Saúde 24

A linha Saúde 24 é uma linha telefónica que tenta ampliar a acessibilidade das pessoas à saúde. É uma iniciativa por parte do Ministério da Saúde que, desta maneira, tenta responder às necessidades de saúde dos seus cidadãos. Este serviço é feito por profissionais de saúde qualificados que fazem a triagem, aconselhamento, e encaminhamento dos doentes de acordo com cada um dos tópicos de triagem.

As pessoas com Diabetes a partir deste serviço podem obter auxílio quando apresentam descompensações próprias da sua doença como hipoglicemia, hiperglicemias, ou até mesmo quadros de Cetoacidose.

Este serviço permite ainda o despiste de infeções frequentes em pessoas com Diabetes.

O Saúde 24, para além deste serviço, faz campanhas de sensibilização e consciencialização com ações de rastreio para que haja uma maior preocupação da parte da população para com este problema.

2.1.2 Acompanhamento nas Farmácias

Os farmacêuticos todos os dias entram em contacto com diabéticos, verificam as suas prescrições e ajudam os pacientes na leitura e análise de valores glicémicos. Este contacto faz com que os farmacêuticos tenham um maior conhecimento da situação real dos diabéticos. Esta informação pode ser útil, não só para dar um melhor aconselhamento, mas também se for dirigida ao médico que segue o doente. Pode ainda ser favorável para uma melhor avaliação do doente e uma prescrição mais especifica às necessidades do mesmo [12].

Muitas vezes são os farmacêuticos que encaminham os doentes não controlados para um médico.

O programa de acompanhamento por parte das farmácias prevê visitas periódicas à farmácia para que o farmacêutico possa recolher dados e identificar eventuais problemas com a medicação ou outros problemas de saúde, podendo haver assim um maior controlo da doença.

2.1.3 Tele-Medicina

A capacidade de tratar alguém, que se encontra a quilómetros de distância e que não podem dirigir-se a um estabelecimento médico, não só agrada tanto a doentes como a médicos e pode ajudar a salvar muitas vidas.

A *Airedale NHS Foundation Trust*, a partir de vídeo-conferência, ajuda vários prisioneiros de vinte hospitais ingleses. Os médicos inseridos neste projeto dão auxilio num variado número de doenças, incluindo a Diabetes.

A partir de uma simples câmara e acesso aos registos médicos dos pacientes, os médicos podem dar consultas remotamente e de maneira mais fiável [22].

Todos estes serviços e ajudas descritos aqui são muito úteis e devem continuar a ser desenvolvidos e apoiados, porem não é possível com eles fazer o acompanhamento diário do doente. Para uma avaliação mais próxima do doente deve ser adotada uma abordagem mais conveniente em que o utilizador possa ser vigiado, confortavelmente e diariamente, sem nunca descurar a opinião médica especializada.

2.2 Aplicações de Apoio aos Doentes com Diabetes

No contexto em que é pretendido aplicar o projeto, irá ser analisado o que existe em termos de aplicações para *Android* na ajuda ao doente de Diabetes. Esta secção em particular é útil para ver quais as principais características das aplicações atuais destinadas à ajuda ao doente com

Diabetes. Com elas é possível concluir o que uma aplicação atual precisa de ter e avaliar como a aplicação proposta nesta tese pode ser mais vantajosa para o diabético.

2.2.1 *OnTrack Diabetes*

Uma aplicação [7] simples que permite inserir os valores de glicose e ver o histórico de valores em forma de gráfico. Em caso de resultados fora do normal, estes são exibidos em vermelho para dar *feedback* de perigo ao utilizador.

2.2.2 *HelpDiabetes*

Esta aplicação [5] ajuda na contagem de hidratos de carbono. Tem uma base de dados com vários tipos de alimentos e os seus valores de hidratos de carbono, gorduras, proteínas, e calorias. O utilizador, para saber as propriedades do que está a consumir, apenas tem de selecionar na aplicação o que vai comer e a quantidade. A aplicação também é capaz de calcular a quantidade de insulina necessária com base na percentagem de hidratos de carbono na refeição.

2.2.3 *Glucosebuddy*

Esta aplicação [4] funciona como base de dados pessoal do diabético. Aqui fica guardada a informação sobre:

- A glicose no sangue;
- A comida que a pessoa ingeriu (hidratos de carbono);
- A medicação (doses de insulina);
- A atividade física;
- A tensão arterial;
- A hemoglobina glicosilada;
- O peso.

Para além de guardar informação útil, esta aplicação lembra o utilizador de fazer os seus testes, o que pode ser muito útil especialmente para os «novos diabéticos» que não têm o hábito nem a rotina de fazer testes ao longo do dia.

2.2.4 *Minsulin*

O *Minsulin* [6] é uma aplicação de origem brasileira disponível, para já, apenas para *iPhone*. Esta aplicação faz a contagem de hidratos de carbono e ajuda no cálculo da dose de insulina a ser tomada a cada refeição.

É uma aplicação muito semelhante à proposta nesta tese, pois esta:

- Regista o número de hidratos de carbono e o nível de glicose no sangue das quatro principais refeições do dia;
- Permite o registo de atividade física;
- Calcula a dose, aconselhável, de insulina com base nos dados fornecidos pelo utilizador (hidratos de carbono, glicose no sangue, atividade física) e tendo em conta qual a altura do dia em que o utilizador se encontra a utilizar a aplicação;
- Disponibiliza a informação para acesso do médico que acompanha o doente.

Para alargar o seu mercado, a equipa que desenvolveu o *Minsulin* planeia criar uma versão para *Android* até ao final de 2014. Porém, passado o ano de 2014 ainda não existe data oficial de lançamento desta nova versão.

Não só está a pensar ser feito um alargamento a outras plataformas móveis, como está a ser desenvolvido um sensor de glicemia intersticial que, como o nome indica, é um sensor que é colocado no Espaço Intersticial por baixo da pele.

Este sensor fará medições a cada cinco minutos e a comunicação com a aplicação via *wireless*, transferindo as leituras para que se possam ser feitos cálculos mais rigorosos e detetar riscos como o de Hipoglicémia, acionando mecanismos de socorro e informando familiares [28].

2.2.5 *Glicontrol*

A *Glicontrol* [3] foi criada por um programador português, a quem foi diagnosticado Diabetes de Tipo 1, que tentou com esta aplicação facilitar a vida de quem também sofre com esta doença. A aplicação permite:

- Registar a glicemia antes/após refeições e da dose de insulina tomada no decorrer da mesma. A aplicação sugere, de seguida, uma dose de correção de insulina;
- Registar o local onde foi aplicada a insulina para que haja uma rotatividade de locais onde esta é aplicada;
- Aceder de forma rápida aos dados, a partir de gráficos;

- Dispor os dados das leituras a par. A informação dos registos disposta lado-a-lado torna a tarefa de comparação de valores mais fácil;
- Transferir as leituras de glicemia via *Bluetooth* (na edição *Android*) para a aplicação de dispositivos médicos;
- Enviar, por *e-mail*, os dados guardados ao médico que está responsável pelo doente.

2.2.6 *Dbees*

Dbee [1] é um projeto de apoio ao diabético que está dividido em duas partes: uma aplicação para *smartphone*, e uma página *online*. A componente do telemóvel funciona como outras aplicações já referidas, guardando todo o tipo de informação relevante do doente, calculando a dosagem de insulina a administrar. Contém lembretes para o utilizador não se esquecer de fazer os seus testes ao sangue e ainda tem a possibilidade de traçar um plano onde se descriminam horários de refeição, tipos de exames, e atividades. A segunda componente, *online*, tem o intuito de visualização de dados. Na página pessoal é possível ver o histórico do utilizador sob a forma de registos ou gráfico. Este histórico também pode ser acedido pelo médico a acompanhar o doente, se este o pretender, dando as credenciais próprias para acesso externo aos registos pessoais.

Como demonstrado anteriormente, existe um grande número de aplicações para *smartphone*, umas mais objetivas e simples, outras mais completas. Tendo em conta as mais completas é possível concluir que as características fundamentais, de uma aplicação de ajuda aos diabéticos, são:

- O registo da glicose, dos hidratos de carbono, da insulina, e das atividades físicas;
- O cálculo da dose correta de insulina a tomar;
- A possibilidade de planeamento de atividades e lembretes ao utilizador;
- O acesso, seguro, do médico que acompanha o doente.

2.2.7 Acompanhamento e Monitorização de Pessoas com Diabetes

Esta aplicação foi desenvolvida pela Fundação Vodafone em conjunto com a Associação Protetora dos Diabéticos de Portugal (APDP), com o intuito de melhorar os serviços prestados pela Associação. Este sistema permite o registo dos valores medidos nas bombas de insulina e medições glicémicas através de uma chamada de voz ou em inglês *Interactive Voice Response* (IVR), via *e-mail* ou *Short Message Service* (SMS).

Já o corpo clínico pode fazer a monitorização do paciente, a partir da *Web*, dos registos feitos e o histórico de terapêuticas aplicadas.

Este sistema tem ainda a possibilidade de definir notificações e alarmes para utentes, médicos e enfermeiros via *e-mail* ou SMS.

As aplicações aqui descritas mostram um caráter de banco de dados, com auxílio ao cálculo de insulina. Estas recolhem os dados, que depois podem ser acedidos pelo médico responsável pelo utilizador e a partir de fórmulas calculam a próxima dose de insulina a ser administrada.

A aplicação que proponho, para além deste papel, tenta aconselhar o paciente, função esta que não foi possível detetar em aplicações disponíveis, até então, ao público.

Capítulo 3

Arquitetura do Projeto

O projeto apresentado teve por base uma aplicação já existente, a *MyDiabetes*. A esta estrutura foi acrescentado uma nova componente, o Sistema de Aconselhamento Baseado em Regras (SABR). Com esta adição a aplicação é agora constituída por três componentes:

- A Aplicação Nativa *Android* (ANA);
- A Base de Dados *SQLite*;
- O SABR.

Os componentes apresentados têm funções distintas na aplicação. A ANA tem um papel ativo de interação com o utilizador e com os restantes componentes. Tanto a Base de Dados *SQLite* como o SABR têm um papel passivo, apenas interagindo com a aplicação e nunca com o utilizador. Tanto o SABR como a Base de Dados apenas operam quando requisitados.

Como componente principal a ANA é responsável por ativar o SABR para que este verifique se se justifica a exibição de um conselho e de que forma este deve ser apresentado ao utilizador, tendo em conta as condições atuais do mesmo.

A ativação e a comunicação entre a ANA e o SABR é feita a partir do YAP *Android*, uma versão *Android* do compilador de alto-desempenho de Prolog [14].

Tanto a ANA como o SABR podem aceder à base de dados. A ANA acede à base de dados para obter dados guardados e para registar novos. Já o SABR acede à base de dados apenas com o intuito de ler os dados, para que consiga saber o estado do utilizador.

Esta interação dos diferentes componentes está representada na Figura 3.1.

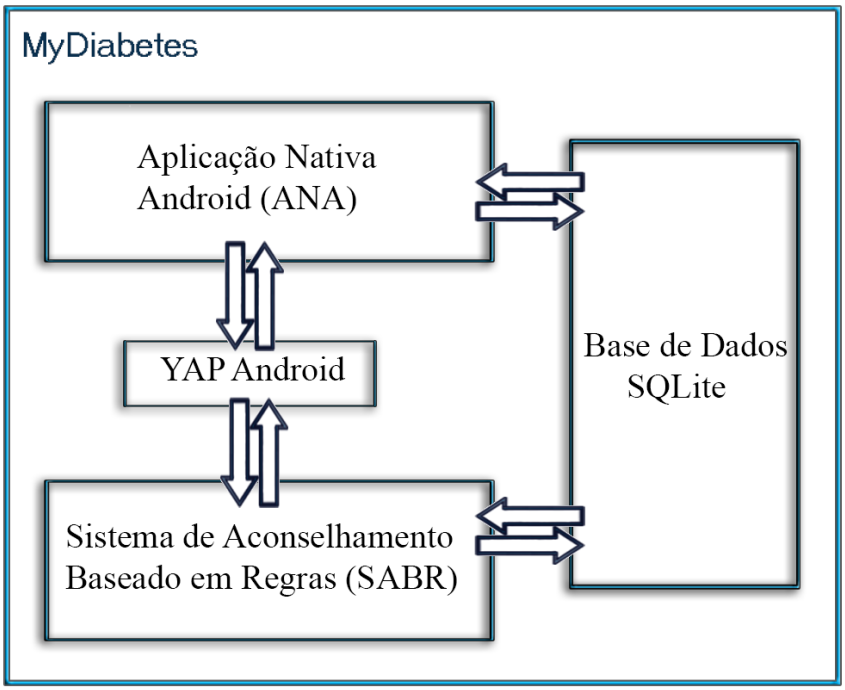

Figura 3.1: Arquitetura global do projeto

3.1 Componente *SQLite*

A aplicação necessita de guardar não só informação do utilizador, como também, os registos que este vai inserindo ao longo do tempo de utilização. Esta componente de Base de Dados *SQLite* foi criada com o intuito de facilitar o acesso à informação que é guardada. Esta informação, ao ser registada na Base de Dados, é organizada em diferentes entidades. A partir de *queries SQL* é possível obter informação concreta sobre estas entidades, por exemplo, toda a informação sobre o utilizador fica registada sob a entidade ***UserInfo*** e para se obter o atributo ***Gender*** do utilizador, executamos a *query* **"SELECT Gender FROM UserInfo"**. Esta *query* obtém o género do utilizador que foi registado na Base de Dados.

Como é possível denotar pelo exemplo dado anteriormente, esta informação é constituída por dados pessoais do utilizador (género, data de nascimento, etc.), registos de rotina do utilizador diabético (medições de glicose, doses de insulina tomada, etc.) e, por fim, os conselhos que lhe foram dados pelo próprio sistema.

3.2 Sistema de Aconselhamento Baseado em Regras

A componente *Prolog* do sistema pode ser vista como o cérebro que, quando requisitado, analisa o estado do utilizador e providencia conselhos e tarefas que este necessita. O SABR é constituído

por diferentes tipos de **factos** escritos em *Prolog*. A ANA, ao chamar este componente, indica-lhe a opção de registo que o utilizador selecionou, a língua que o utilizador escolheu para a aplicação correr, e se vai inserir um novo registo ou se este já foi registado. A partir desta informação o SABR tenta acionar todas as regras ligadas à informação fornecida, da de maior gravidade para a de menor gravidade. Se as condições do utilizador, registadas na base de dados, provocarem a ativação de uma das Regras, é então retornado um **ID** (Identificador) da mensagem/conselho a ser exibido.

3.3 Componente *Android*

A componente *Android* é essencial na medida em que faz a ponte entre o processamento de dados e o utilizador. Esta componente interage com o utilizador recebendo *input* e enviando *feedback* sob a forma de confirmações ou conselhos. Quando o utilizador seleciona uma opção de registo no menu principal da componente *Android*, é acionada uma função à qual chamamos *Master Rule*, pertencente ao SABR, que verifica se existem novos conselhos a ser dados. Após uma medição/inserção de dados é, novamente, feita uma chamada à mesma função. Isto é feito, pois a inserção de novos dados pode fazer despoletar um conselho. Caso não fosse feita esta verificação, uma possível situação de crise só iria ser detetada no próximo registo efetuado pelo utilizador.

Capítulo 4

Aplicação Nativa *Android*

O SABR foi pensado como um *upgrade* à aplicação *Mydiabetes*. Esta aplicação possui uma interface e tem, implementados, métodos para que se possa efetuar registos como da glicose ou da insulina tomada. Esta aplicação teve de ser alterada para que se consiga trabalhar com o SABR. As alterações vão desde a inserção de um novo menu para visualização posterior de conselhos dados, até à implementação de um Sistema de Alerta, entre outras.

O Sistema de Alerta foi introduzido para que seja possível enviar diferentes tipos de conselhos ao utilizador. Existem cinco tipos diferentes de *popups* que podem ser usados nas mais variadas situações:

- *Popup* Simples (**simpleAdvice**) - este é o tipo base de *popup* e consiste numa janela que faz a exibição de uma mensagem com um botão ***OK*** para confirmar a receção do conselho;
- *Popup* com Sugestão de Novo Registo (**suggestTest**) - um *popup* que mostra a mensagem e sugere ao utilizador que faça um registo. Este tipo de *popup* pode ser usado, por exemplo, para lembrar o utilizador, e sugerir-lhe que atualize um certo registo não atualizado há algum tempo;
- *Popup* com Temporização de Novo Registo (**adviceTimer**) - este tipo de *popup* mostra a mensagem do conselho e obriga, após um determinado período de tempo definido pelo mesmo, o utilizador a fazer um novo registo. Este *popup* foi pensado para os casos mais graves de hipo/hiperglicemia onde é importante re-testar a glicemia para saber se o caso continua grave. Contudo, pode ser usado num variado número de casos;
- *Popup* com Sequência de Questões (**sequencePopup**) - este tipo de *popup* recebe um conjunto de identificadores de questões que posteriormente usa para retornar do SABR as questões a ser feitas ao utilizador. Estas questões são exibidas sequencialmente e têm como opção de resposta «sim» e «não». Após o utilizador responder a uma questão, uma nova aparecerá até que não hajam novas perguntas a ser feitas. Este *popup* é uma extensão do *Popup* com Temporização de Novo Registo (**adviceTimer**) e, por isso, também recebe

um conselho base, com uma temporização de novo registo, que exibe antes de começar a fazer as perguntas requeridas ao utilizador;

- *Popup* com Questão Urgente (**urgentQuestionPopup**) - ao contrário do *popup* anterior, que exibe uma sequência de questões, este tipo de *popup* faz uma única questão que tem como hipóteses de resposta «sim» e «não». A principal particularidade deste *popup* é que, se o utilizador não responder após um certo tempo, o *popup* executa uma determinada ação. Este *popup* é essencialmente usado em casos em que o sistema necessita saber se o utilizador se encontra consciente. Ao não responder o utilizador indica que não o está, desencadeando assim uma resposta do sistema.

A ANA não é só responsável pela componente interativa. Existem ações que a componente *Prolog* não consegue executar, como por exemplo, temporizações que têm de ser feitas pela ANA. Outras opções como o bloqueio/desbloqueio de regras, que requerem a inserção/remoção de **factos**, é responsabilidade da componente *Android*.

O aconselhamento médico não se restringe a mensagens que auxiliam o utilizador informando-o de como proceder em certas situações, ou informando-o que deve ter hábitos mais saudáveis. O sistema de aconselhamento proposto pretende também que o utilizador seja responsável por cumprir certos objetivos. Para este fim foi criada uma **Lista de Tarefas**.

A **Lista de Tarefas**, que está diretamente ligada ao SABR, vai conter todas as tarefas que o utilizador deve cumprir. Estas têm diferentes níveis de urgência e estão dispostas da mais grave para a menos grave. Existem, entre as mais graves, as tarefas essenciais, que necessitam de ser cumpridas para que haja um bom funcionamento do SABR. Esta Lista de Tarefas é atualizada e exibida ao utilizador com o iniciar da aplicação e sempre que o utilizador pressiona o botão de acesso à Lista.

O funcionamento do SABR está condicionado pelo *input* que é feito. Se o utilizador, por exemplo, cometer um erro e inserir um valor crítico no sistema, este vai acionar os **factos** *Prolog*, que vão ser explicados mais à frente, preparados para esta situação, e o utilizador vai ser erradamente aconselhado. A situação não pode ser retificada dado que por agora não é possível modificar valores erradamente inserido na base de dados.

Além de estar sujeito a erros do utilizador, o SABR necessita de registos atuais para conseguir aconselhar corretamente. Todos os registos inseridos na base de dados têm um «prazo de validade» correspondente, que pode ser definido num **facto**. Passado este prazo o valor do registo é apenas acessório, se necessitarmos de verificar se o utilizador se encontra bem no momento presente não podemos ter apenas acesso a registos do dia anterior, pois estes já não são válidos. Portanto, o *Software* está dependente do bom uso que o utilizador pode ou não dar à aplicação. Em caso de mau uso, o único prejudicado é o próprio utilizador.

De forma a auxiliar uma correta utilização da aplicação, aquando da sua inicialização, são indicados, sob a forma de uma lista, todos os registos que estão prestes a expirar o prazo de validade, ou que já o passaram para que o utilizador proceda à atualização dos mesmos. A lista

de tarefas encontra-se sempre disponível ao utilizador sobre a forma de um ícone representando uma lista de verificação no menu principal da ANA, visível na Figura 4.1.

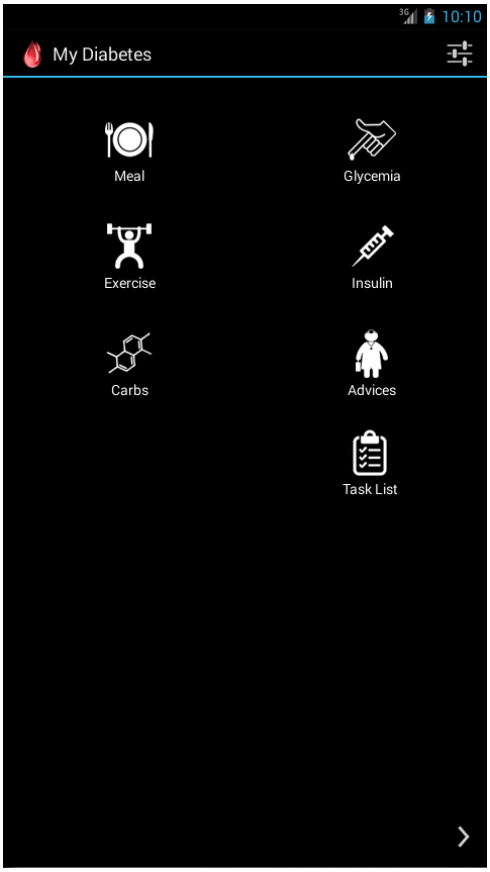

Figura 4.1: Imagem menu principal da ANA

Se existir(em) registo(s) cujo prazo expirou aquando da seleção de um registo por parte do utilizador, este será alertado que necessita de atualizar o(s) registo(s) indicado(s).

Capítulo 5

Linguagem *Prolog*

A componente *Prolog* desempenha um papel fundamental na aplicação apresentada, portanto, é importante que se entenda o seu funcionamento. Um programa *Prolog* consiste num conjunto de cláusulas. Cada uma destas cláusulas pode ser um **facto** ou uma regra. Após ser carregado/consultado por um interpretador *Prolog*, é possível submeter "questões" às quais o interpretador vai responder consoante as regras e **factos** que possui. O SABR usa um variado conjunto tanto de regras como de **factos** que será explicado no próximo capítulo.

5.1 Factos

Um **facto** pode ser visto como uma verdade. Se o **facto** não existir então considera-se uma falácia. Os **factos** começam com um predicado (denominado também por átomo, na linguagem *Prolog*), que pode ser visto como o nome do **facto**, e terminam com um ponto final. O predicado pode ser seguido por um ou mais argumentos que ficam entre parênteses. Estes argumentos podem ser átomos (constantes), números, variáveis (representados por átomos com letra inicial capitular) ou listas. Todos os argumentos devem estar separados por vírgulas. A partir de **factos** podemos, não só obter valores de verdade, como também informações. Por exemplo, ao criar um **facto** como o representado na Lista de Código 5.1, podemos verificar que a regra existe se fornecermos ao interpretador o *input* **maxValue(glucose,100).**, ao que ele irá retornar ***True***, pois o **facto** existe.

```
maxValue(glucose,100).
```

Bloco de Código 5.1: Facto exemplo

Esta informação pode não ser muito útil, porém se ao invés de usar o número 100 usarmos uma variável **Valor**, ou seja, se inserirmos **maxValue(glucose,Valor).**, irá ser retornado 100. Isto acontece porque existe um **facto *maxValue*** com argumento **glucose** e com um valor 100.

O interpretador de *Prolog* vê a variável (átomo não inicializado) e tenta emparelha-la com os valores que tem para que a regra seja uma verdade. Visto o sistema só ter um único **facto** em que o valor é 100, então o *Prolog* considera que a variável deve ser 100. O *Prolog* retorna, então, como resposta possível 100 e ***True***, pois o **facto** existe. Se existissem mais **factos** que respeitassem o *input*, o *Prolog* retornaria todas as hipóteses diferentes. A esta capacidade do *Prolog* é dado o nome de unificação. Se existirem diferentes possibilidades, a partir de *Backtracking*, o *Prolog* é capaz de retornar todas as possibilidades diferentes de resposta. Esta técnica é muito útil, porém para o sistema de regras que pretendemos apresentar não é usada.

O sistema considera apenas a primeira ocorrência, que corresponde à mais grave. Desta forma é possível, por exemplo, definir **factos** com valores máximos para cada tipo de teste. Quando existir a necessidade de testar se um valor está acima do máximo estipulado, basta chamar o **facto** correspondente e fazer a comparação dos valores.

5.2 Regras

As Regras *Prolog* podem ser vistas como uma extensão de um **facto**. São constituídas por duas partes: uma cabeça e um corpo. A cabeça da regra é semelhante a um **facto**, constituída por um predicado com argumentos; o corpo da regra consiste em diferentes cláusulas que necessitam de ser verdade para que a regra também o seja. A cabeça e o corpo da Regra são separados por ":-", que pode ser interpretado como um **se e só se** para uma melhor perceção do significado da regra.

As regras *Prolog* abrem a possibilidade de testar vários **factos** para conseguir avaliar de forma mais concreta.

5.3 Consulta de Ficheiros

A linguagem *Prolog* possibilita a importação de **factos** de outros ficheiros. Ou seja, se existirem Regras definidas num outro ficheiro, estas podem ser importadas para o ficheiro base que está a correr. Esta utilidade permite que, caso exista uma modificação isolada de um ficheiro, apenas o ficheiro modificado é, de facto, atualizado. Os restantes ficheiros que constituem a aplicação não precisam de ser alterados, algo que não é comum nas aplicações para telemóvel.

5.4 Consulta de Bases de Dados

Como já foi referido no Capítulo 3 (Arquitetura do Projeto) as pesquisas à base de dados são fundamentais para o conhecimento do estado do utilizador. A linguagem *Prolog*, mais especificamente o *SWI-prolog*, permite, usando o módulo ***prosqlite***, a execução de *queries* SQLite a partir de **factos** específicos. A inclusão destes **factos** específicos em todos os **factos** de

pesquisa à base de dados, pode tornar estes extensos e repetitivos. A solução encontrada para este problema foi a criação de **factos** que recebem uma *query* e executam todos os **factos** exclusivos juntamente com a *query* fornecida, retornando o resultado obtido. Desta forma, conseguimos tornar os **factos** que pesquisam a base de dados mais percetíveis e simples. Existem dois tipos possíveis de pesquisa à base de dados:

- Pesquisas simples - onde apenas é usada uma *query* simples (sem argumentos) como **select name from user**.
- Pesquisas complexas - nesta *query* são necessários argumentos extra para especificar a pesquisa como a *query* **select name from user where age>=Age**. Esta *query* necessita de um argumento **Age** que vai especificar a idade do utilizador.

Tendo em conta estas duas variantes de pesquisas à base de dados, foram criados dois factos para fazer cada uma destas pesquisas. Para conseguir fazer uma *query* simples deve ser utilizado o **facto** representado na Lista de Código 5.2, que aglomera todos os **factos** necessários para executar uma *query* simples (sem argumentos). Para executar uma *query* complexa (com Argumentos) deve ser usado o **facto** representado na Lista de Código 5.3.

```
runSql(Query, Result) :- use_module(library(prosqlite)),
    sqlite_disconnect(Conn), sqlite_connect('DB.sqlite',Conn ),
    sqlite_query(Conn, Query, Result); sqlite_disconnect(Conn).
```

Bloco de Código 5.2: Facto base que permite a execução de uma *query SQLite* na base de dados da aplicação

```
runSql_with_args(Query, Variables, Result) :- use_module(library(prosqlite)),
    sqlite_disconnect(Conn), sqlite_connect('DB.sqlite',Conn ),
    sqlite_format_query(Conn, Query-Variables, Result); sqlite_disconnect(Conn).
```

Bloco de Código 5.3: Facto base que permite a execução de uma *query SQLite* na base de dados da aplicação

Todas as capacidades da linguagem *Prolog* demonstradas são utilizadas pelo SABR, portanto fundamentais para o seu funcionamento, sistema que irá ser descrito no ponto seguinte.

Capítulo 6

Sistema de Aconselhamento Baseado em Regras

Num mundo ideal, cada doente teria ao seu dispor um médico vinte e quatro horas por dia para o ajudar. Em caso de dúvida ou crise, o médico que acompanha o doente podia auxiliá-lo e tudo era resolvido rapidamente. Esta situação ideal não é possível, porém já seria uma grande ajuda se conseguíssemos transportar estes conselhos médicos até ao paciente sem a presença física de um médico. Um sistema de aconselhamento pode solucionar este desafio, comprometendo-se a ajudar o utilizador fornecendo conselhos que um médico daria ao encontrar o utilizador numa determinada situação. Ao transformar regras médicas em **factos** *Prolog* é possível verificar certos estados do utilizador. Se anexarmos o conselho que o médico daria nessa situação ao **facto** que foi criado, conseguimos aconselhar o utilizador usando conselhos médicos sem a presença de um profissional de saúde. A aplicação *MyDiabetes*, sem um sistema de aconselhamento, é apenas mais uma aplicação como as já referidas no estado da arte. Porém, ao ser adicionado um sistema capaz de aconselhar os utilizadores em situações de risco, criamos um novo tipo de aplicação que pode abrir a porta a uma nova maneira de pensar e desenvolver aplicações para diabéticos. Dado ser uma área sensível, é necessário que haja um forte apoio médico durante o desenvolvimento deste sistema. É importante que o colaborador médico entenda exatamente o que cada regra, que está a ser criada, faz e a sua influência no sistema. Neste capítulo irá ser apresentada, com maior detalhe, a arquitetura deste sub-sistema/componente da aplicação, o seu funcionamento, e o papel do colaborador médico na criação do sistema de regras.

O SABR possui quatro tipos distintos de regras:

- Regras do Sistema;
- Regras Médicas;
- Pesquisas à Base de Dados;
- Mensagens/Conselhos.

Cada um dos tipos apresentados está definido no seu próprio ficheiro *Prolog*. O apoio médico na elaboração de regras é necessário nos ficheiros de Regras Médicas e de Conselhos/Mensagens. O ficheiro que contém as Regras do Sistema não deve ser alterado.

6.1 Regras do Sistema

O ficheiro com as Regras do Sistema contém todas as regras necessárias para o SABR conseguir funcionar. Neste ficheiro podemos encontrar a *Master Rule*, que irá ser chamada pela ANA, assim como um conjunto de regras que faz o sistema funcionar como é pretendido. Ao iniciar a execução a partir deste ficheiro, os **factos** definidos nos restantes ficheiros do SABR, são importados para este ficheiro. A opção de ter este método de operação traz como vantagens:

- Organização - ao separar diferentes tipos de **factos** por diferentes ficheiros, a informação torna-se mais legível para quem tem de trabalhar no SABR;
- Processamento Local - ao importar as regras para um único ficheiro, não existe a necessidade de trocar entre ficheiros durante a execução.
- Atualização de *Software* Otimizada - se existir uma modificação de uma regra num ficheiro, a atualização é feita apenas no ficheiro alterado.

As alterações a este ficheiro devem ser feitas por programadores experientes e devem ser amplamente testadas, devido a qualquer alteração influenciar todo o SABR.

6.2 Pesquisas à Base de Dados

As pesquisas à base de dados são essenciais para o conhecimento do estado do utilizador. Devido ao seu tipo específico e ao seu número, todos os **factos** que pesquisam a base de dados estão definidos no seu próprio ficheiro. Se existir a necessidade de expandir o leque de regras para incluir uma *query* ainda não pensada, basta acrescentar os **factos** extra ao ficheiro, sendo o único requisito os novos **factos** terem um identificador único. Por outras palavras, os **factos** inseridos devem ter um nome diferente do das regras já existentes.

6.2.1 Factos de Pesquisa à Base de Dados

As regras de pesquisa à base de dados têm um grande potencial na medida em que podem fazer um processamento prévio da informação presente na base de dados. Na sua maioria os Facto de Pesquisa à Base de Dados (FPBD) são **factos** *Prolog* que executam uma *query SQLite* que retorna informação específica do utilizador, como por exemplo, o FPBD representado na Lista de Código 6.1.

```
timeOfLastRegistry(insulin, Time) :- Time = runSql('select (strftime("%s" ,
    "now") - strftime("%s" , DateTime)) from Reg_Insulin order by DateTime Desc
    limit 1', Time), float(Time).
```

Bloco de Código 6.1: Exemplo de um FPBD que retorna o tempo em segundos desde o último registo de insulina

Cálculos como o nível médio de glicose ou número de hipoglicemias num certo espaço de tempo, são questões que podem ser respondidas a partir de *queries SQLite*.

6.2.2 Regras de Avaliação de Estado

As *queries SQLite*, já referidas, são um elemento fundamental para se obter informação do utilizador, porém são inconsequentes, pois para se tomar decisões sobre o estado do utilizador não basta ter um valor. É preciso fazer uma verificação do valor obtido para saber se este passa certas balizas, delimitadas pelo colaborador médico. Para conseguir contextualizar os valores é necessário inserir **factos** que verificam se o utilizador se encontra numa situação concreta. Um exemplo de um **facto** que avalia se o valor Glicêmico se encontra acima de um valor máximo previsto, ou seja, se tem a glicemia alta, seria:

```
hasHighGlycemia :- lastGlycemiaValue(Value),
                   maxValue(glycemia, ValorMax),
                   Value >= ValorMax.
```

Bloco de Código 6.2: Código exemplo para verificação de existência de hiperglicemia

Esta regra, Lista de Código 6.2, definida a título de exemplo, executa em primeiro lugar um **facto** *SQLite* para obter o valor Glicêmico mais recente; em seguida vai buscar o valor máximo definido pelo médico cooperante; por fim compara os dois valores. Se o último valor Glicêmico registado é mais alto que o definido pelo colaborante médico, então é retornado um valor booleano ***True***; em caso contrário é retornado um ***False***. Desta forma conseguimos saber se o utilizador está ou não com glicemia alta.

6.3 Regras Médicas

As Regras Médicas são o conjunto de **factos** desenvolvidos a partir de procedimentos médicos para avaliar o estado do utilizador.

Neste ficheiro são definidos tempos máximos que o utilizador pode estar sem fazer registos de diferentes tipos, valores máximos e mínimos para os diferentes parâmetros, as diferentes

subdivisões que cada tipo de registo pode ter, as tarefas que o utilizador deve cumprir e os diferentes conselhos com a definição da mensagem a ser exibida.

6.3.1 Variáveis Globais

O SABR não é apenas constituído por Regras complexas de avaliação de estado. Para conseguir criar Regras de maior nível são necessárias Regras mais simples, entre elas, Regras que definem valores globais do sistema. Estes valores, como valores máximos para certos parâmetros que posteriormente vão ser utilizados em diversas Regras complexas, precisam de ser definidos. O facto de estas Regras serem em geral mais simples, não implica que todas elas pertençam a este grupo de complexidade. Versões complexas deste tipo de Regra são importantes para que se consiga personalizar os valores globais para o utilizador.

6.3.1.1 Tempos Máximos de Teste

O SABR necessita de diferentes tipos de informação. Esta informação que o sistema vai pesquisar à base de dados não é intemporal. Diferentes registos têm diferentes prazos de validade e estes devem ser definidos pelo médico que acompanha o desenvolvimento da aplicação. Para definir estes diferentes tempos são definidos **factos** do tipo:

maxTimeUntested(tipo de registo, prazo de validade em segundos).

6.3.1.2 Valores Máximos e Mínimos Gerais

Durante o desenvolvimento do sistema de aconselhamento é, muitas vezes, necessário indicar valores máximos e mínimos para os quais os **factos** são verdade. Por uma questão de organização e de conveniência para quem cria os **factos** que vão avaliar o estado do utilizador, devem ser criados **factos** que definem valores globais, por exemplo:

maxValue(tipo de registo, valor máximo permitido).

minValue(tipo de registo, valor mínimo permitido).

Ao serem definidos e posteriormente usados, estes **factos** permitem que, quando um valor necessitar alterações, basta modificar o valor no **facto** que o define.

Ao redefinir o valor, a sua representação utilizada nos **factos** é automaticamente alterada. Esta técnica impede que a cada alteração seja necessário mudar o valor localmente em todos os **factos** que utilizam esta variável.

6.3.1.3 Valores Globais Complexos

A definição de valores simples pode muitas vezes não ser suficiente para conseguir aconselhar o utilizador. Como já referido anteriormente, pode existir a necessidade de criar Valores Globais Complexos. Estas regras têm uma constituição diferente e não retornam diretamente o valor pedido. Antes de retornar o valor pretendido é feita uma pesquisa à base de dados da informação do utilizador, de seguida é calculado o valor desejado e só depois este é retornado. Estes valores são extremamente úteis no cálculo de diferentes administrações de insulina. O diabético durante a sua rotina diária depara-se com várias ocasiões em que tem de calcular a sua dosagem de insulina tendo em conta diferentes contextos. Em seguida vão ser demonstrados diferentes exemplos de casos que podem ocorrer seguidos das regras que podem ajudar o utilizador a partir de Valores Complexos.

Os casos aqui referidos foram retirados do site da *Diabetes Education Online Diabetes Teaching Center at the University of California, San Francisco* [18].

Insulina para compensação pelos hidratos de carbono ingeridos à refeição Neste caso o utilizador deve em primeiro lugar calcular o quanto a dose de insulina compensa os hidratos de carbono ingeridos. Isto pode ser feito a partir da fórmula **"Dose de insulina necessária tendo em conta os hidratos de carbono ingeridos"= "Totalidade de Hidratos de Carbono presentes na refeição" (em gramas) ÷ "Número de gramas que uma dose de insulina consegue compensar"**. Um exemplo de uso desta fórmula é o que se segue: O utilizador vai ingerir sessenta gramas (60g) de hidratos de carbono ao almoço. O rácio de eficácia da insulina em relação à capacidade para compensar a ingestão de hidratos de carbono é de 1:10. Para conhecer a dosagem de insulina a ser administrada aplicamos a fórmula já apresentada da seguinte forma 60 g ÷ 10 = 6 unidades. A partir da fórmula é possível verificar que o utilizador necessita de administrar seis unidades de insulina de ação rápida para compensar os hidratos de carbono ingeridos. A tradução desta função para um **facto** resultaria no **facto** representado na Lista de Código 6.3.

```
correctValue(insulin, Racio, NumberCarbs, Value) :- Value =
    Racio/NumberCarbs.
```

Bloco de Código 6.3: Exemplo de tradução de uma função do cálculo da dose correta de insulina

Este **facto** recebe como argumentos o tipo **insulin**, o **Racio** número de hidratos de carbono que uma dose de insulina consegue compensar e o NumerCarbs número de hidratos de carbono o utilizador vai ingerir. A partir destes argumentos o **facto** calcula a dosagem a ser administrada e retorna a partir do argumento **Value**.

Cálculo Geral do Requerimento Diário de Insulina do Corpo Este exemplo contêm um método para o cálculo geral da dose de insulina que o corpo necessita. Este cálculo é feito a

partir do peso do utilizador, que é um parâmetro que pode ser obtido da base de dados, o que não era possível no exemplo anterior.

A função para fazer este calculo é **Total de Insulina Necessário Diariamente = 0.55 X Peso total em Quilogramas**

A tradução desta função resultaria no **facto** representado na Lista de Código 6.4.

```
getValue(insulin, Value) :- getValue(weight, Weight), Value = 0.55*Weight.
```

Bloco de Código 6.4: Exemplo de tradução de uma função do cálculo geral do requerimento diário de insulina do corpo

Como é possível ver na Lista de Código 6.4 o **facto** usa um **facto** de Pesquisa à Base de Dados para conhecer o peso do utilizador antes de calcular o valor **Value** pretendido.

6.3.2 Definição de Afinidade entre Parâmetros

O utilizador pode fazer, na aplicação, diferentes tipos de registos que vão por sua vez originar diferentes tipos de conselhos, dependendo do estado em que se encontra. Contudo pode ser pertinente, ao fazer um registo de um tipo, por exemplo glicemia, também testar **factos** relacionados com outros tipos de registo. Tendo em conta que esta expansão de um tipo de registo para outros tipos pode ser proveitosa para o sistema, foi introduzida a opção de criação de **Listas de Parâmetros**.

O colaborador médico pode inserir um **facto** que define que, ao testar um tipo de registo, o SABR também deve avaliar outros tipos de registo, referidos de seguida na Regra. Para implementar esta opção deve ser inserido um **facto** do tipo:

adviceRelatedParameters("Tipo a ser dividido",["Subtipo 1","Subtipo 2",..."Subtipo n"]).

Desta forma se, por exemplo, o colaborador médico pretender, cada vez que se faz um teste à glicose, testar também regras pensadas para responder a novos registos do tipo insulina ou exercício deve inserir o **facto** exibido na Lista de Código 6.5.

```
adviceRelatedParameters(exercise,[insulin, glucose]).
```

Bloco de Código 6.5: Facto exemplo de definição de relações entre parâmetros

A definição pura de afinidade não é suficiente. O sistema necessita que na criação de novos **factos** seja definido o subtipo apropriado. Ou seja, por exemplo considerando o **facto** exibido na Lista de Código 6.5, consideramos que ao fazer um novo registo de glicemia devemos testar os **factos** relacionados com insulina e exercício. Se este sistema funcionasse assim sem restrições, todas as regras relacionadas com estes tipos de registo seriam testadas. Algumas destas regras

podem ser relevantes, porém muitas podiam não fazer sentido no contexto base de um novo registo de exercício que estamos a fazer. Portanto para que apenas sejam considerados os **factos** relevantes são testados, em primeiro lugar, todos os **factos** de tipo **exercise**, depois são considerados todos os **factos** de tipo **insulin** que têm como subtipo **exercise** e por fim são considerados todos os **factos** de tipo **glucose** que têm também como subtipo **exercise**. Desta forma conseguimos por um lado encontrar todos os **factos** relacionados com o registo que está a ser feito e ao mesmo tempo conseguimos filtrar todos os **factos** que não fazem sentido no contexto atual. Esta solução requer um pouco mais de atenção por parte tanto do programador como do médico colaborador dado que é necessário perceber o contexto real do **facto** e como este pode ser útil na situação particular. Numa situação direta este processo é simples, se é feito um registo de novo exercício os **factos** funcionam apenas para este fim de registo. Ao expandir o tipo fazemos com que uma regra funcione para mais de um tipo de registo, ou seja se uma regra de **exercise** tem uma afinidade para com **glucose** e é criada um **facto** com tipo **exercise** e subtipo **glucose** então esta regra será testada tanto em registos de **exercise** como em registos de **glucose**. Portanto, é preciso pensar bem quando e como usar esta opção do SABR.

Um caso onde esta sub-parametrização pode ser útil é ao relacionarmos o registo de refeições com o registo de glicemia. Se existir uma Regra que testa o valor de Hidratos de Carbono da última refeição e que verifique se o último valor de glicemia obtido está em valores preocupantes, fazendo a distinção entre valores altos e baixos sendo para isso necessário criar duas regras uma para cada tipo de valor. Com esta informação podemos aconselhar o utilizador se este comeu muito, pouco ou o suficiente para o seu estado glicémico. Esta sub-parametrização não foi discutida com médicos cooperantes, tratando-se, portanto, de uma sugestão que deve ser discutida posteriormente a fim de ser usada numa versão funcional do SABR.

6.3.3 Factos para Definição de Tarefas

O aconselhamento médico não está restrito a conselhos. Existem boas práticas que um Diabético deve ter e que não podem ser negligenciadas. Estas tarefas diárias podem ir das mais gerais, recomendáveis a toda à população em geral, incluindo os não diabéticos, como, por exemplo, a prática de exercício físico diário, até às tarefas mais personalizadas, normalmente definidas pelo médico que acompanha o utilizador.

A aplicação atualiza as Tarefas a ser executadas pelo utilizador, ao iniciar o sistema, cada vez que é feito um registo e/ou quando é selecionado o ícone da Lista de Tarefas.

Existem quatro tipos diferentes de Tarefas:

- Tarefas Essenciais ao Sistema - o SABR, para conseguir aconselhar o utilizador, necessita de conseguir aceder a registos atuais. Cada tipo de registo tem um prazo de validade. Passado este prazo, o registo deixa de ter valor funcional no aconselhamento médico do utilizador. Como o sistema necessita de registos válidos, sempre que um registo está a chegar ao fim do seu prazo de validade, este **facto** é exibido como sendo uma Tarefa necessária para que

o sistema consiga funcionar em pleno.

- Tarefas Essenciais ao Utilizador - este tipo de Tarefas não é essencial ao bom funcionamento da aplicação, porém é esperado que o utilizador as cumpra.
- Tarefas Recomendadas - este tipo de Tarefas são recomendadas como boa prática para uma vida saudável, não sendo estas essenciais;
- Tarefas Personalizadas - estas tarefas são criadas pelo utilizador a partir do menu de criação de nova Tarefa da ANA.

A sintaxe das Tarefas é semelhante à dos conselhos. A título de exemplo, vamos analisar um Facto genérico de uma Tarefa:

hasTask(ID, Description):- Condition, Description = 'description'.
Existe um átomo **hasTask** que define que esta é uma Tarefa. Os seus argumentos são: um **ID** que vai definir o tipo de tarefa que está a ser representada seguida de um parâmetro **Description** onde será retornada uma descrição da Tarefa, de forma equivalente à mensagem dos conselhos.

Existem quatro tipos possíveis de **IDs** para uma tarefa:

- **system** - Representa as Tarefas que implementam Tarefas Essenciais ao Sistema;
- **essential** - Representa as Tarefas que implementam Tarefas Essenciais ao Utilizador;
- **basic** - Representa as Tarefas que implementam Tarefas úteis para o utilizador;
- **custom** - Representa as Tarefas que implementam Tarefas que foram produzidas pelo utilizador.

A descrição da Tarefa deve conter texto com a ação pretendida, seguida de um separador / (barra), seguido de um valor que indica a urgência da Tarefa. Este valor está definido numa escala de zero a dez sendo 0 a situação de menor gravidade e dez a representação da situação de maior urgência. Além de definir descrições, é necessário indicar as condições a serem cumpridas na Tarefa. Cada Tarefa pode ser constituída por uma ou mais condições, dependendo da sua complexidade, como por exemplo, se quisermos que o utilizador faça um registo de Glicemia a cada duas horas, a Tarefa a inserir no SABR seria a representada na Lista de Código 6.6.

```
hasTask(basic, Description):- lastGlycemiaRegistry < 7200, Description = 'Deve
    fazer uma medição à glicemia a cada 2 horas/7'.
```

Bloco de Código 6.6: Exemplo de um **facto** que implementa uma tarefa

Sempre que a data de registo da última medição de glicemia for maior que 7200 segundos (7200 s), duas horas (2h), então o **facto** é ativado e é acrescentada uma nova Tarefa à Lista de Tarefas do utilizador.

6.3.4 Tarefas Personalizadas

As Tarefas base do sistema tentam ser o mais abrangentes possível, porém existem tarefas úteis em situações particulares. Ao contrário das Tarefas mais gerais, as Tarefas Particulares/Personalizadas são incluídas no SABR posteriormente à obtenção da Aplicação pelo médico que acompanha o utilizador ou pelo próprio utilizador. A não inclusão de Regras Personalizadas deve-se ao **facto** de uma Tarefa ser útil para um grupo particular de utilizadores, mas não ser apropriada para a restante população diabética utilizadora da Aplicação. Um exemplo que comprova esta afirmação é a verificação do colesterol. Para os utilizadores com mais idade ou com problemas de colesterol, é válida a criação de uma Tarefa para que seja feito um novo registo de **Colesterol** regularmente. Contudo, para utilizadores que não padecem deste problema, não é razoável ter esta tarefa ativa.

A opção de inserir as Tarefas Personalizadas posteriormente origina um problema. As Tarefas personalizadas inseridas pelo utilizador após a obtenção do *software* não serão avaliadas pelo médico colaborador, nem pelo programador responsável pelo desenvolvimento do sistema. Existem duas formas de disponibilizar ao utilizador a possibilidade de desenvolver tarefas personalizadas:

- Fornecer toda a informação ao utilizador - Nesta abordagem seriam disponibilizados ao utilizador todos os **factos** possíveis para que este consiga produzir novas Tarefas.
- Fornecer apenas um determinado número de Tarefas pré-concebidas - O utilizador pode escolher, a partir de um conjunto de Tarefas já desenvolvidas e aprovadas pelo colaborador médico, uma Tarefa para personalizar. Dado a Tarefa já estar definida, apenas certos parâmetros, como o tempo a executar a Tarefa, podem ser personalizados.

As duas possibilidades têm pontos fortes e fracos. Por um lado, dar uma grande variedade de escolha ao utilizador permite a criação de Tarefas mais personalizadas e mais adaptadas ao que o utilizador pretende. Por outro lado, o utilizador, devido a desconhecimento, inexperiência ou mesmo por erro comum, pode inserir erros ou criar Tarefas que, embora pareçam fazer o que este pretende, podem não o estar a fazer. A introdução de uma Tarefa com erros pode levar a consequências imprevisíveis, que na sua maioria irão prejudicar o próprio utilizador.

O fornecimento de Tarefas permite controlar os erros. Dado este **facto**, foi decidido permitir ao utilizador apenas a parametrização de Tarefas já desenvolvidas.

As Tarefas são desenvolvidas e testadas, e só depois fornecidas ao utilizador para que este possa personalizar parâmetros simples específicos que não introduzem erros no sistema. O único parâmetro, cuja edição é permitida ao utilizador, é a temporização da Tarefa.

Como é possível ver na Figura 6.1, o utilizador deve selecionar uma das possíveis Tarefas de uma Lista de Tarefas já definida no sistema.

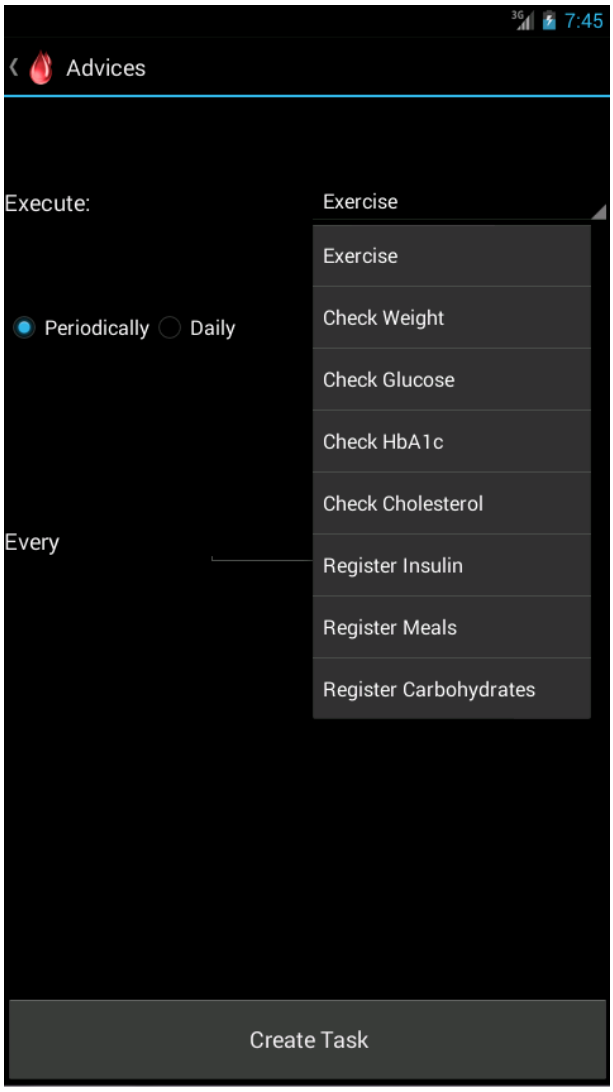

Figura 6.1: *Display* de adição de nova tarefa com lista de tarefas Aberta

Após a sua seleção, o utilizador deve escolher uma frequência em que a Tarefa deve ser executada. O utilizador pode definir uma Tarefa a deve ser executada **n** número de vezes ao dia, ou definir executar a Tarefa dado um certo intervalo, ou seja, este define que a Tarefa deve ser executada a cada **n** número de horas. Depois de especificada a Tarefa pode ser inserida na Lista de Tarefas, pressionando o botão ***Create Task***.

Caso as Tarefas deixem de ser necessárias, tenham sido inseridas por engano ou com algum erro no tempo, estas podem ser facilmente eliminadas. Para executar esta ação basta escolher a opção de eliminar Tarefas. Uma Lista de Tarefas Personalizadas é apresentada ao utilizador, onde pode ser escolhida uma Tarefa para ser eliminada. Este menu pode ser visto na Figura 6.2.

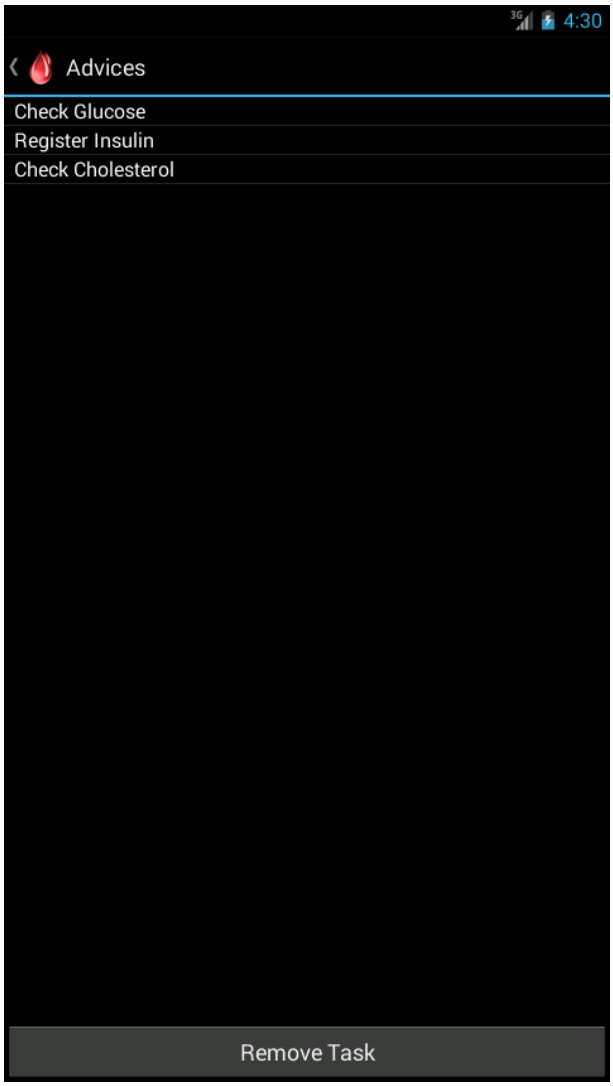

Figura 6.2: *Display* de remoção de uma tarefa personalizada

Após a seleção da Tarefa pretendida, o utilizador deve pressionar o botão ***Remove Task*** para remover em definitivo a regra que escolheu. A Lista de Tarefas Personalizadas contém apenas Tarefas personalizadas, dado que as restantes Tarefas do SABR não podem ser retiradas pelo utilizador.

6.3.4.1 Constituição das Tarefas Personalizadas

A criação de Tarefas Personalizadas, do ponto de vista do utilizador, é composta por duas ações: escolha de uma Tarefa da lista de possíveis Tarefas e definição da frequência temporal da Tarefa. Após a execução das duas ações e pressionar o botão para inserir nova Tarefa por parte do utilizador, a nova Tarefa é criada e inserida no SABR.

Do ponto de vista do sistema o processo é um pouco mais complexo. O utilizador, ao abrir o menu de criação de uma nova Tarefa, aciona a ANA que, por sua vez, vai usando a *Master Rule*

do SABR para obter todas as tarefas personalizadas existentes. Estas são depois disponibilizadas ao utilizador para que este possa escolher qual pretende.

O utilizador, ao carregar no botão para inserir a nova tarefa, aciona um conjunto de ações por parte do sistema. É feita uma verificação dos dados inseridos para não permitir a inserção de dados com tipos errados, como *Strings* ou valores acima do que é possível. Se o valor inserido passar a verificação então:

- Este valor inserido é guardado numa variável designada ***timeInput***;
- O sistema verifica se a Tarefa escolhida tem condições extra para ser ativada;
- Dependendo do tipo de frequência temporal escolhida, é criado um **facto** que é adicionado ao conjunto de **factos** do SABR.

O sistema, para verificar se a tarefa tem condições extra de forma a poder adiciona-las à nova Tarefa que vai ser criada, executa o código representado na Lista de Código 6.7. Este código executa uma *query* cujo parâmetro **text** é recebido como *input* da função *Java*

```
public static String getPersonalizedExtraConditions(String text) {
    Object result;
    result = executeProlog("msg(_, _, "+text+", ExtraConditions).");
    if(result){
        return result;
    }
    return "";
}
```

Bloco de Código 6.7: Código responsável por retornar condições extra

getPersonalizedExtraConditions. Como já foi explicado no capítulo 5, ao executar uma *query* como a apresentada na Lista de Código 6.7, irão ser retornadas, se existirem, as diferentes possibilidades para **ExtraConditions**. Se estas condições existirem, então a função *java* retorna estas condições para que sejam anexadas à nova Tarefa; caso contrário é retornada uma **String** vazia.

Após a obtenção da **String** resultante da função **getPersonalizedExtraConditions**, o restante processo consiste em juntar as diferentes peças numa nova Tarefa. Um exemplo de uma junção dos diferentes *inputs* numa tarefa nova é o Facto **"hasTask(custom, "+Mensagem a apresentar+"/5) :- NOT(hasPeriodic("+insulin+", "+10800+")), "+hasLastValue-Above(glucose, 100)+"."**. Todos os **factos** das Tarefas Personalizadas são construidos da seguinte maneira:
O átomo que simboliza o nome do **facto** é definido como **hasTask**, pois é assim que as Tarefas são definidas no SABR. O **facto** que estamos a construir tem como argumento "**Mensagem a apresentar**", que vai ser a mensagem a exibir na lista para que o utilizador perceba o que deve

fazer. Por fim, temos as condições para que este **facto** seja verdade: **NOT(hasPeriodic(insulin, 10800))** é uma negação da condição **hasPeriodic(insulin, 10800)**, que pode pode ser vista na Lista de Código 6.9. Esta é usada para confirmar se o utilizador está a cumprir a periodicidade da Tarefa. Este **facto** começa por ver o tempo atual e compara-o com o tempo do último registo somado com o intervalo definido de periodicidade, ou seja, se atualmente o utilizador está dentro do intervalo definido, então o **facto** retorna **True**. Dado que a Tarefa deve aparecer sempre que o utilizador está em incumprimento, então este **facto** está negado, porque se o utilizador estiver fora do intervalo definido o **facto hasPeriodic(insulin, 10800)** retorna **False**. Como este está negado, o Booleano passa a **True** e o **facto "hasTask("+Mensagem a apresentar+") :- NOT(hasPeriodic("+insulin+", "+10800+")), "+hasLastValue-Above(glucose, 100)+"."** pode ser ativado caso as restantes condicionantes se confirmem. A condicionante **hasLastValueAbove(glucose, 100)**, é uma Regra de Avaliação de Estado (RAE) que verifica, neste caso, se o parâmetro **glucose** está acima dos cem. O **facto** geral pode ser visto na Lista de Código 6.8. A junção dos diferentes **factos** cria um novo **facto** que traduz uma Tarefa Personalizada pronta a ser inserida no SABR.

```
hasLastValueAbove(Parameter, Value) :- lastValue(Parameter, LastValue),
    LastValue >= Value.
```

Bloco de Código 6.8: Código do **facto** hasLastValueAbove

Desenvolvidas principalmente para ser usadas na criação de Tarefas Personalizadas, os **factos** ***hasPeriodic*** que pode ser visto na Lista de Código 6.9 e ***Daily*** representado na Lista de Código 6.10 também podem ser usados pelo médico colaborador como ferramenta na criação de Conselhos ou Tarefas do SABR.

```
hasPeriodic(Parameter, TimeGap) :- timeOfLastRegistry(Parameter,
    LastRegistryTime), getCurrentTime(CurrentTime), CurrentTime <=
    (LastRegistryTime+TimeGap).
```

Bloco de Código 6.9: Código do **facto** hasPeriodic

```
hasDaily(Parameter, Count) :- numberOfRegistriesToday(Parameter, TodayCount),
    TodayCount<=Count.
```

Bloco de Código 6.10: Código do **facto** hasDaily

6.3.5 Regras de Avaliação de Risco

Com os elementos apresentados conseguimos obter valores concretos do utilizador. É possível balizar estes valores e concluir se o utilizador se encontra numa situação específica. O sistema é ainda capaz de fazer esta avaliação para vários subtipos. Com o apoio destas ferramentas podemos criar regras de mais alto nível, que avaliam diferentes estados e condições concretas do utilizador. Ao conhecer uma situação concreta, o SABR consegue aconselhar o utilizador de forma personalizada para uma situação específica em que este se encontra. Para obter estas regras de maior nível, a que chamamos Regras de Avaliação de Risco, é necessária a combinação de uma ou mais Regras de Avaliação de Estado. A utilização de outras regras é viável, porém, por questões de organização, se for necessária alguma regra de verificação deve ser criada uma RAE para este fim, no local indicado.

Uma possível regra, a título de exemplo, que verifica se o utilizador se encontra com uma hiperglicemia recorrente seria a representada na Lista de Código 6.11.

```
inRisk(end, glucose, _, 10, ID) :- ID = hasHighGlycemia,
    hasRecurrentHyperGlycemias.
```

Bloco de Código 6.11: Regra Avaliadora de Risco para deteção de hiperglicemias recorrentes

Como podemos ver temos um **facto** com a especificação **end** que significa que o conselho só pode ser ativado após haver um registo novo no sistema. O **facto** possui o tipo **glucose**, que aceita qualquer subtipo devido ao caracter '_'(*underscore*), cuja gravidade é de valor '10' e cujo ID (identificador) do conselho a ser entregue é '**hasHighGlycemia**'. O nome escolhido como ID teve em conta o fim da regra, porém qualquer nome seria válido. O único requisito na criação de um ID é que este seja único. O **facto** de estar relacionado com a função da regra é apenas aconselhável para que exista uma melhor organização e legibilidade.

Como condições de ativação temos duas Regras de Avaliação de Estado: **hasHighGlycemia**, já apresentada anteriormente, que verifica se o utilizador tem o seu valor glicémico baixo; **hasRecurrentHyperGlycemias** é uma RAE que verifica se esta é a primeira vez que está a ocorrer uma hiperglicemia. Caso seja verificado que o utilizador está com um nível elevado de glicose no sangue e já não é a primeira hiperglicemia que está a sofrer então a regra é acionada. A regra que chamou esta Regra de Avaliação de Risco (RAR) recebe então o **ID hasRecurrentHyperGlycemias** para que se consiga chegar à mensagem que deve ser entregue ao utilizador.

Da mesma forma é possível verificar todo o tipo de ocorrências sendo, para isto, preciso inserir uma regra que, corretamente, verifique se a condição pretendida ocorreu. O desenvolvimento da Regras Médicas, embora pareça intuitivo carece da experiência de um programador, preferencialmente do ramo da programação em lógica. Não menos essencial é o conhecimento que o colaborador tem sobre o que cada **facto** analisa e retorna.

6.4 Mensagens/Conselhos

Este ficheiro contém os diferentes conselhos e as suas respetivas opções. No intuito de chegar a utilizadores de diferentes nacionalidades, este ficheiro vai ter diferentes versões, ou seja, ao contrário do ficheiro com as Regras do Sistema ou o Ficheiro com as Regras Médicas, vai ser necessário ter vários ficheiros Mensagens/Conselhos, um por cada língua que se deseje inserir na aplicação. A estrutura do **facto** é intuitiva. Precisamos de um ID único e de uma mensagem que queremos passar ao utilizador. A ideia que pode ser mais complexa é a das opções extra. Estas opções são obrigatórias, pois a ANA necessita desta informação para saber de que maneira vai passar a mensagem ao utilizador. As opções têm de ser inseridas com a mensagem que vai ser entregue, embora as opções em si não apareçam no corpo do conselho apresentado. As opções que já foram implementadas são:

- Tipo de *popup* (Campo Obrigatório) - esta opção especifica o tipo de *popup* que deve ser usado para passar a mensagem ao utilizador. Os tipos de *popup* são os já apresentados no capítulo anterior "Aplicação Nativa *Android*". Cada um dos diferentes tipos requer diferentes argumentos, necessários para a sua criação:
  - *Popup* Simples - não requer argumentos extra;
  - *Popup* com Sugestão de Registo - requer a inserção do tipo de Registo que se quer propor ao utilizador;
  - *Popup* com Temporização de novo Registo - requer a inserção do tipo de Registo a ser feito e o tempo de espera antes de iniciar este mesmo registo.
- Bloqueio de Regra - por vezes pode ser útil o bloqueio de uma regra, por exemplo, um conselho que indique ao utilizador para que faça mais exercício, se não existirem conselhos mais urgentes a ser dados, este pode ser apresentado ao utilizador repetidas vezes, isto pode causar frustração ao utilizador. Para evitar estas situações, e dado este conselho não ser urgente, deve ser feito um bloqueio do mesmo após a sua apresentação durante um período de tempo aceitável. No término deste, a ANA retira o **facto** que provoca o bloqueio da Regra.

  Para utilizar esta opção é obrigatória a especificação do **ID** do Conselho que se quer bloquear, do valor temporal a ser esperado e a sua representação temporal (s-segundos, m-minutos e h-horas), desta maneira definindo o tempo que dura o bloqueio.
- Som de Alerta - é possível fazer tocar um som de alerta para conseguir a atenção do utilizador.

A título de exemplo, um **facto** que use todas as opções implementadas seria o representado pelo *Listing* 6.12.

Como é possível ver a partir deste exemplo o conselho é descrito por vários atributos separados pelo caracter '/' (barra). A descrição destes atributos é separada a partir do caracter '-' (hífen).

```
msg(highGlycemiaValue,'Your Glycemia level is high, a new Glycemia test will be
    scheduled to see if your glycemia values stabilize./ adviceTimer - Glycemia
    - 600-s / alarm / block - 600 - s').
```

Bloco de Código 6.12: Exemplo de um **facto** que implementa uma mensagem/conselho

A ordem de colocação das diferentes opções necessita ser a indicada, pois o *parser* que recebe o texto com o conselho analisa cada um dos «blocos», que definem uma opção, e dependendo do seu conteúdo executa uma ação diferente. Contudo, a ordem pela qual se define cada um dos parâmetros da opção deve ser respeitado. Se uma opção não contiver os parâmetros esperados ou se estes não estiverem na ordem indicada irá ocorrer um erro e o conselho não será apresentado ao utilizador.

No caso do *popup* com teste deve ser inserido primeiro o tipo de *popup* seguido do tipo de teste a ser feito. Se o conselho necessitar de um temporizador para execução posterior de um registo, então o intervalo de espera até à sua execução deve estar definido a partir do valor temporal e da sua unidade de tempo.

6.5 Arquitetura do Sistema de Aconselhamento Baseado em Regras

Os diferentes ficheiros que compõem o Sistema de Ficheiros e as suas Regras, foram apresentados, porém, ficou por conhecer de que maneira todos estes elementos interagem.

O SABR, como já foi referido anteriormente, mantêm-se inativo até ser chamado pela ANA. Ao ser chamada, a *Master Rule* é responsável por:

- importar os **factos** dos restantes ficheiros;
- importar as mensagens do ficheiro da língua do sistema;
- iniciar o processo de verificação de necessidade de conselhos;
- retornar o conselho obtido à ANA.

A sequência de ações é demonstrada na Figura 6.3 onde conseguimos ver os diferentes passos para a obtenção de um conselho.

É visível a passagem pelos vários ficheiros, porém é de ter em conta que o sistema não percorre os diferentes ficheiros, pois importa inicialmente as regras desses mesmos ficheiros. A Figura 6.3 exibe os diferentes ficheiros por uma questão de organização e para que seja percetível a comunicação entre as várias funções que pertencem a diferentes ficheiros.

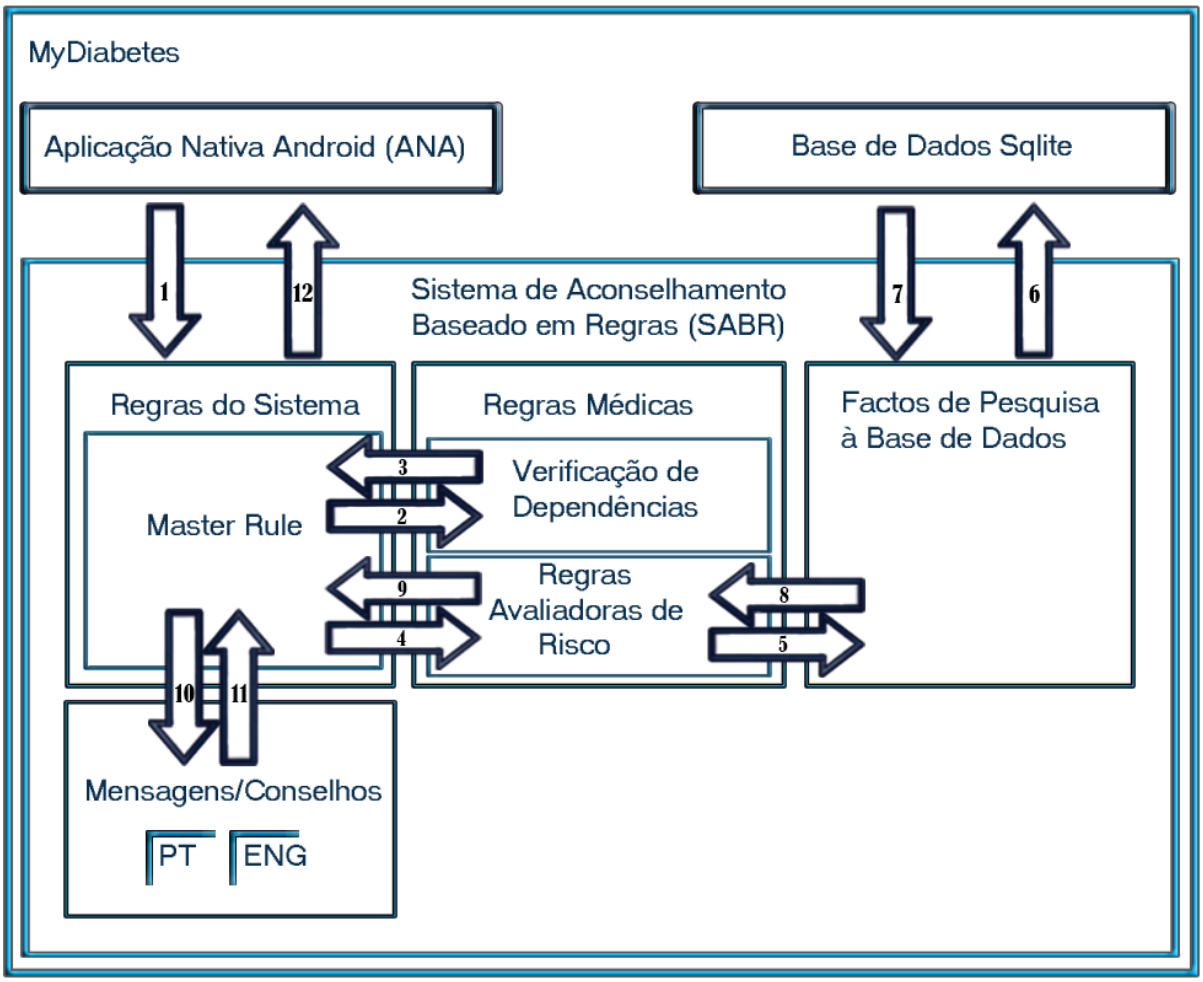

Figura 6.3: Arquitetura do SABR

Para melhor compreensão da Figura 6.3, vai ser feita uma descrição passo a passo do processo de obtenção de um conselho.

A ANA chama a *Master Rule* pertencente ao ficheiro de Regras do Sistema que faz parte do SABR sempre que é acionada uma condição em que exista a necessidade de obter um conselho, como por exemplo, quando o utilizador clica para fazer um registo.

Feita a chamada, a ANA indica à *Master Rule* a linguagem do sistema, o registo selecionado pelo utilizador e se o utilizador está a inserir dados ou se estes já foram inseridos. Esta ação é representada no passo (1).

Ao receber um pedido, a *Master Rule* importa os diferentes ficheiros do SABR, entre eles o ficheiro com as mensagens na linguagem que a ANA indicou.

Já atualizado e com todas as regras necessárias ao seu funcionamento à disposição da *Master Rule*, é iniciada a **verificação de dependências** representado no passo (2).

Após feita a verificação, é devolvida à *Master Rule* uma lista de parâmetros que devem ser avaliados, caso exista uma subdivisão de parâmetros definida. No caso da não existência de sub parametrização, é retornada uma lista com um único elemento, o próprio parâmetro. Este processo de retorno da lista de parâmetros a avaliar é representado no passo (3).

Com todos os parâmetros necessários, a *Master Rule* passa à avaliação do estado do utilizador. Para proceder a esta avaliação são chamados os **factos** do ficheiro de **Regras Médicas**, procedimento representado no passo (4).

No ficheiro de **Regras Médicas** existem várias **Regras Avaliadoras de Risco**. O conjunto de regras é percorrido de cima para baixo e, quando encontra uma regra que seja concretizável para a sua busca, retorna o **ID** dessa regra e a sua urgência. É deste facto que advém a importância de escrever as Regras por ordem decrescente de urgência e, dentro da mesma urgência, da mais para a menos específica.

Cada uma das regras verifica certas condições. Estas condições são factos particulares da situação do utilizador. Para poder verificar se o utilizador se encontra na situação a ser analisada, são feitas pesquisas à base de dados. Pesquisas à base de dados inseridas diretamente na Regra podem torná-la muito complexa, logo são usadas pesquisas já pré-definidas no ficheiro de FPBD, demonstrado no passo (5). Este conjunto de **factos** acede à base de dados de forma a obter a informação pretendida. Este processo é representado pelo passo (6).

Os passos (7) e (8) demonstram o retorno da informação, primeiro para os FPBD, e depois para as **Regras Avaliadoras de Risco**. Se as condições do **facto** não se verificarem, então o sistema segue para o próximo **facto** até um destes se confirmar. Se nenhum **facto**, dos definidos com a colaboração do médico, se ativar, o **facto** *default*, que simboliza a não necessidade de dar conselhos, é acionada. Quando um destes **factos** é acionado, o seu **ID** e **Urgência** é passado à *Master Rule*, representado no passo (9). A *Master Rule* recolhe os pares **ID** e **Urgência** dos diferentes **factos** que foram ativos e escolhe o de maior urgência.

Após fazer esta avaliação a partir do **ID**, a *Master Rule* procura a mensagem correspondente ao **ID** pretendido no ficheiro da linguagem que foi importado no início do processo. Esta procura está representada no passo (10), sendo o passo (11) a resposta com a mensagem pretendida.

Por fim, no passo (12), já em posse da mensagem pretendida, a *Master Rule* retorna à ANA o conselho a exibir ao utilizador.

Desta forma ficamos a conhecer não só os vários componentes do SABR como a forma em que estes interagem. A arquitetura do SABR e as diferentes interações entre os ficheiros que o compõem são elementos importantes para a compreensão do sistema e o papel de cada componente que o constitui. A organização é um elemento fundamental para o bom funcionamento de um sistema. Ao inserir elementos novos no sistema, desde uma Regra nova até um novo ficheiro com conselhos numa outra Língua, deve haver um conhecimento do sistema para que esta organização se mantenha.

Para o médico cooperante o conhecimento da arquitetura e do funcionamento especifico não é fundamental. Tendo em conta o seu papel no desenvolvimento de novas Regras para o SABR é importante que o médico cooperante conheça todas as «ferramentas» que tem disponíveis. O médico deve conhecer a existência e utilidade de componentes como os FPBD ou as Variáveis Globais. Todas as preocupações técnicas como a inserção de novas Regras nos seus diferentes ficheiros ou mesmo desenvolvimento de novas Regras do Sistema devem ser deixadas à responsabilidade do programador responsável pelo desenvolvimento do projeto. Desta forma o médico cooperante pode concentrar-se na melhor forma de detetar uma certa situação e que aconselhamento dar ao utilizador.

Capítulo 7

Produção de Regras

Apresentadas as componentes e a arquitetura da Aplicação, é importante analisar como se processa a criação das Regras que compõem o SABR. A definição de valores máximos/mínimos, tempos, e outros valores globais, é um processo simples, bastando seguir as instruções já referidas no Capítulo 6, secção de Variáveis Globais. As Regras de mais alto nível, como as **Regras de Avaliação de Risco** ou os **Factos para Definição de Tarefas**, são as que podem trazer mais dúvidas durante a sua produção. A transformação de uma regra médica para um **facto** deve ser feita por meio das seguintes fases:

- Definição de condições para a regra ser ativada - todos os conselhos do SABR são criados para responder a uma determinada situação do utilizador, portanto o primeiro passo deve ser definir em que condições o utilizador se deve encontrar para receber o conselho pensado, e que ainda precisa de ser definido.
- Definir altura de exibição - existe a possibilidade de exibir um conselho antes da efetuação de um registo e/ou depois do registo de novos dados. Deve ser pensada qual a altura apropriada para mostrar o conselho ao utilizador.
- Definir um **ID** - os Identificadores têm de ser únicos para que não exista redundância e, posteriormente, erros. Além de únicos, para que seja mais fácil o entendimento da Regra, o **ID** deve ser criado de forma a que, a partir dele, seja percetível o que é pretendido no **facto** criado.
- Criar uma Mensagem/Conselho - a mensagem a ser exibida ao utilizador deve ser direta e percetível. O utilizador, ao ler a mensagem, não pode ficar com dúvidas sobre o que deve fazer.
- Definir o tipo de *popup* a exibir - existem cinco tipos de *popups*. Destes, deve ser pensado qual o melhor para passar o Conselho ao utilizador.

Estes são os passos fundamentais para a criação de um **facto** que traduz uma Regra. Cada um destes passos deve ser pensado com cuidado para cumprir os seus objetivos.

Existe um número ilimitado de situações que podem ocorrer e não é razoável pensar que o SABR consegue abranger todos os casos possíveis de crise após a sua criação, portanto regras novas podem, e devem, ser implementadas conforme a evolução do SABR.

De seguida será abordado, com maior pormenor, cada um destes passos e os cuidados a ter.

7.1 Condições de Ativação da Regra

O desenvolvimento das **Regras de Avaliação de Risco** (base do SABR) é imprescindível à criação de um conjunto de **Regras de Avaliação de Estado** para que funcionem como ferramentas base na criação de condições de ativação. Porém, tal como as **Regras de Avaliação de Risco**, estas podem sofrer alterações ou pode ser necessária a implementação de novas Regras.

O processo de criação de novas **Regras de Avaliação de Estado** está intrínseco à necessidade do colaborador médico de conhecer certas informações do utilizador. Se for necessário conhecer um tipo de informação não disponível até à data, então uma nova RAE deve ser pensada e criada.

7.2 Criação de Novas Regras a Partir de Protocolos Médicos

O algoritmo a ser seguido durante a criação de novas Regras já foi apresentado. Contudo o seu uso pode facilitar a sua compreensão. Com este fim irá ser demonstrada a tradução de alguns protocolos médicos em Regras, que estão incluídos no SABR.

7.2.1 Identificação e aconselhamento em caso de hipoglicemia

Existem diferentes causas para uma hipoglicemia sendo algumas delas:

- Dose excessiva de insulina;
- Ingestão insuficiente de Hidratos de Carbono ou incumprimento dos horários das refeições/omissão de uma refeição;
- Aumento da utilização da glicose já no sistema (exercício físico);
- Aumento da sensibilidade à insulina devido a exercício físico, perda de peso ou melhoria do controlo glicémico;
- Diminuição da produção endógena de glicose (ex: ingestão de álcool);
- Doença renal;
- Doença hepática;
- Ingestão Medicamentosa especialmente insulina, sulfonilureias;

- Álcool;
- Devido a Doença grave:
  - Insuficiência hepática;
  - Insuficiência cardíaca;
  - Insuficiência renal;
  - Sépsis;
  - Inanição.

Cada uma destas causas pode ser identificada por uma diferente RAE. Cada uma das diferentes causas deve gerar um conselho diferente. Muitas das causas indicadas podem ser verificadas a partir de registos na base de dados, porém existem outras causas que ainda não possuem forma de registo. Se não for possível identificar uma causa particular, deve ser criada uma Regra que retorna um conselho geral que vai ajudar o utilizador.

7.2.1.1 Criação de uma Regra de identificação de causa de hipoglicemia

A deteção de causas pode ser importante para o aconselhamento mais personalizado ao utilizador. Se conhecermos a causa da sua hipoglicemia, podemos alertá-lo para este facto, de forma a que não volte a cometer o mesmo erro. Uma Regra de identificação é uma RAR que determina se o utilizador se encontra em hipoglicemia e apresenta algum sintoma característico. Estes sintomas são procurados na Base de Dados por FPBD que verificam se a situação dada como argumento ocorre. Podemos ver um exemplo de uma RAR usada como Regra de identificação de uma causa de hipoglicemia na Lista de Código 7.1. A RAR apresentada tem duas RAE como condicionantes de ativação. A primeira o facto de o utilizador estar em hipoglicemia, condição necessária pois se o utilizador não se encontrar em hipoglicemia a RAR apenas estaria a verificar se o utilizador padecia de uma certa doença. A segunda verifica se o utilizador se encontra com uma certa doença. As duas RAE em conjunto produzem uma relação de causalidade que pode ser útil para concluir a causa da hipoglicemia. Neste exemplo apenas foi verificada uma possível causa, no entanto é possível a verificação de outras causas ou mesmo a junção de condições adicionais para que se consigam obter mais conclusões sobre as diferentes causas a afetar o utilizador.

```
inRisk(end, glucose, glucose, 10, ID) :- ID = 'hipoCause',
    hasHipoglicemia(_), hasDisease(sepsis).
```

Bloco de Código 7.1: Regra de Avaliação de Risco para identificação de sepsis como causa de hipoglicemia

7.2.1.2 Deteção e aconselhamento em caso de hipoglicemia sintomática ligeira ou moderada

Nestes casos aplica-se a «Regra dos 15», um protocolo médico de ação em caso de hipoglicemias ligeiras/moderadas. Esta regra dita que alguém a sofrer de hipoglicemia (ligeira/moderada) deve:

- Administrar quinze gramas de Hidratos de Carbono de absorção rápida (ex: glicose ou sacarose);
- Aguardar quinze minutos e fazer uma nova medição do valor de glicemia;
- Se glicemia continuar abaixo de setenta miligramas por decilitro (70 mg/dL), devem ser administrados mais quinze gramas de Hidratos de Carbono da mesma tipologia;
- Repetir medição e tratamento até atingir níveis normais;
- Se faltar mais de uma hora para a refeição seguinte ingerir quinze gramas de Hidratos de Carbono complexos adicionais.

Os quinze gramas de Hidratos de Carbono de absorção rápida podem ser administrados sob a forma de:

- Dois pacotes de açúcar: cerca de quinze gramas (15g) de Hidratos de Carbono;
- Coca-cola (meia lata): cerca de dezasseis gramas e meia (16,5g) de Hidratos de Carbono;
- Ice Tea (uma lata): cerca de vinte gramas (20g) de Hidratos de Carbono;
- Compal *light* (uma embalagem 330 mililitros): quinze gramas (15g) de Hidratos de Carbono;
- Marmelada vinte gramas (20g): quinze gramas (15g) de Hidratos de Carbono.

Devem ser evitados produtos como o chocolate, gelados e outros produtos com lípidos durante o tratamento da hipoglicemia aguda.

No caso de hipoglicemia moderada pode haver a necessidade de administrar uma quantidade maior de Hidratos de Carbono. Se a colaboração do utilizador para administrar os Hidratos de Carbono não for possível, deve ser usada uma injeção intra muscular ou subcutânea de ***glucagon*** [1].

O estado de consciência é ainda um assunto em desenvolvimento. Este será debatido no ponto seguinte referente à deteção/aconselhamento de hipoglicemias graves, onde o facto do utilizador poder estar inconsciente é mais provável e requer uma ação mais rápida.

Outros conselhos úteis para o utilizador são:

[1] *Glucagon* é uma hormona produzida pelas células *alpha* numa zona do pâncreas, conhecida como *the islets of Langerhan.*

- Considerar diminuir débito basal (baixar 0,5 a uma Unidades por cada vinte miligramas por decilitro [20 mg/dL] abaixo de sessenta miligramas por decilitro [60 mg/dL]) ou suspender temporariamente a Bomba Infusora de Insulina;
- Atrasar a administração do bólus prandial;
- No caso de ocorrerem três ou mais hipoglicemias não justificadas numa semana:
  - reajustar perfusão basal, rácio insulina/hidratos de carbono, fator de sensibilidade à insulina ou uma combinação destes fatores.

Apresentado o protocolo médico, irá ser agora feita a sua tradução para Regras do SABR. Em primeiro lugar vamos considerar o problema da «Regra dos 15». O problema desta regra é a temporização. O SABR não tem a capacidade de temporizar ações. Para estas serem possíveis, o sistema depende da ANA. Esta componente é capaz de criar um Conselho que agenda um novo registo, o que é extremamente útil para este protocolo que estamos a considerar. Portanto, podemos considerar o protocolo numa linguagem simplificada, como sendo:

- Aconselhar a ingestão de quinze gramas de Hidratos de Carbono de absorção rápida, podendo usar um elemento da lista de produtos, como exemplo, que contêm o número indicado de gramas de Hidratos de Carbono. Se faltar mais de uma hora para a refeição seguinte aconselhar a ingestão de quinze gramas de Hidratos de Carbono complexos adicionais.
- Agendar um novo teste de glicemia para daqui a quinze minutos;
- Caso os valores voltem ao normal dar conselhos adicionais.

Ao analisar a simplificação do protocolo, é possível verificar a ausência do passo que induz um ciclo no protocolo. Isto é feito porque a própria natureza do sistema provoca o ciclo pretendido. Por outras palavras, o utilizador, ao fazer o novo teste, se continuar com os seus valores glicémicos altos, vai provocar a ativação da regra novamente. O ciclo só termina quando esta Regra principal que vamos criar, que verifica se o utilizador se encontra em estado de hipoglicemia, não conseguir ser ativada.

Tal como todas as RAR, a Regra que vamos criar deve designar-se **inRisk**. Dado o seu contexto, queremos que esta regra seja considerada sempre que o utilizador faça um novo registo do tipo **glucose**. Para conseguir este efeito é necessário definir como condições de ativação da nova Regra o argumento **end**. Para a regra ser considerada apenas em testes glicémicos, inserimos como segundo argumento **glucose**, ou seja, a Regra terá como parâmetro e sub-parâmetro **glucose**. Como gravidade podemos usar o valor **6**, dado ser uma situação a ter em consideração, mas não demasiado grave. Concluída a definição dos parâmetros, vamos agora analisar as Regras de Avaliação de Estado que necessitamos neste protocolo.

Em primeiro lugar devemos dividir a regra em diferentes tipos de gravidade de hipoglicemia. Neste exemplo, estamos a verificar se o utilizador se encontra num estado de hipoglicemia ligeira

ou moderada. Para este efeito, podemos usar, ou caso não exista, criar a RAE **hasHipoglicemia(light)**.

Para conseguir saber se o utilizador se encontra em estado de hipoglicemia ligeira ou moderada temos de obter o valor do último registo de glicemia do utilizador, e ver se este é igual ou está abaixo do valor de setenta miligramas por decilitro (70 mg/dL). Esta verificação pode ser feita a partir do **facto** representado na Lista de Código 7.2 que usa um FPBD para encontrar o valor do último registo de glicose e posteriormente confirma se este valor é igual ou menor a setenta miligramas por decilitro (70 mg/dL) e maior a cinquenta miligramas por decilitro (50 mg/dL). Os valores apresentados podem e devem ser substituídos por Variáveis Globais que definem os valores máximos e mínimos para cada caso. Neste exemplo não foram usados para que seja mais percetível a tradução que está a ser feita. O FPBD usado, representado na Lista de Código 7.3, não é da responsabilidade do médico cooperante.

```
hasHipoglicemia(lightModerate) :- lastValue(glucose, Value), Value
    <=70, Value >50.
```

Bloco de Código 7.2: Regra de Avaliação de Estado para determinação de hipoglicemia ligeira ou moderada

```
lastValue(glucose, Value) :- runSql('select Value from Reg_BloodGlucose
    order by Timestamp Desc limit 1', Value).
```

Bloco de Código 7.3: Facto de Pesquisa à Base de Dados

A partir da RAE conseguimos saber o estado do utilizador e aconselha-lo de acordo com a sua situação. Antes de criar uma nova Mensagem/Conselho é necessário criar um **ID**. Dado que esta Regra identifica uma hipoglicemia ligeira ou moderada, podemos usar como **ID** «lightModHipoglicemia».

Depois de definido o **ID**, a RAE fica concluída e com a composição que podemos ver na Lista de Código 7.4.

```
inRisk(end, glucose, glucose, 8, ID) :- ID = 'lightModHipoglicemia',
    NOT(blocked(lightModHipoglicemia)), hasHipoglicemia(lightModerate).
```

Bloco de Código 7.4: Regra de Avaliação de Risco para verificação de hipoglicemia ligeira ou moderada

Podemos ver um elemento novo na Regra que criamos: **NOT(blocked(lightModHipoglicemia))**. Este elemento implementa na Regra uma operação lógica que a impede de ser ativada se o seu **ID** estiver bloqueado. A razão deste bloqueio irá ser referida posteriormente.

Necessitamos agora de criar, nos ficheiros de Conselhos/Mensagens, o conselho a ser atribuído, caso este **facto** seja ativado, nas respetivas línguas. Como mensagem a ser exibida ao utilizador, podemos mostrar um conselho como: «O seu valor glicémico encontra-se baixo. Por favor tome 15 gramas de Hidratos de Carbono (Dois pacotes de açúcar). Um novo teste irá ser agendado dentro de 15 minutos para verificar se este valor volta ao normal.». A esta mensagem é necessário acrescentar um tipo de *popup*. Para este caso, em que é necessário que o utilizador repita o registo, devemos usar o *popup* **adviceTimer**, a que devemos juntar os parâmetros **Glycemia**, que representa o tipo de registo a ser feito. Devemos ainda juntar o parâmetro **15**, que representa o valor temporal, e, por fim, **m** que representa a unidade temporal em minutos. Ao concatenar estes dois componentes, obtemos a mensagem final pronta a ser inserida numa nova mensagem. Por fim, obtemos o **facto** representado na Lista de Código 7.5.

```
msg('lightModHipoglicemia', 'O seu valor glicémico encontra-se baixo.
    Por favor, tome 15 gramas de Hidratos de Carbono (Dois pacotes de
    açúcar). Um novo teste irá ser agendado dentro de 15 minutos para
    verificar se este valor volta ao normal. Entretanto, por favor,
    responda
às Questões que irão surgir.
/sequencePopup-dizziness-feebleness-sweating
/adviceTimer-Glycemia-15-m
/block-lightHipoglicemia-0-s
/block-hipo-15-m').
```

Bloco de Código 7.5: Mensagem/conselho de alerta em caso de hipoglicemia ligeira ou moderada

Até este ponto conseguimos detetar uma hipoglicemia leve ou moderada, porém ainda não é possível fazer a distinção entre as duas. Esta separação e reconhecimento é feito a partir de sintomas que o utilizador pode sentir. Para conhecer estes sintomas, usamos um tipo de *popup* específico da ANA o **sequencePopup**, que recebe como argumentos os **IDs** das perguntas a ser feitas ao utilizador. A ANA, depois de receber estes **IDs** vai procurar a mensagem que possui cada um destes. Tal como é explicado no Capítulo 4, o tipo de *popup* **sequencePopup** corre na ANA como um conselho com temporizador de novo registo, com a vantagem de após o agendamento fazer uma série de perguntas. Estas perguntas são feitas uma após a outra até todas terem sido respondidas.

É possível verificar na Regra que criamos que existem dois bloqueios: um com um valor temporal de **0** (permanente) e o segundo com um valor temporal de **15** com a especificação de **m** (minutos). O primeiro bloqueio é feito, pois esta Regra deve pertencer ao topo da hierarquia de Regras e, se esta não for bloqueada, o utilizador, caso continue a padecer de uma hipoglicemia ligeira ou moderada, causa uma repetição das perguntas que já foram feitas.

O segundo bloqueio é feito por questões de segurança relativas à má utilização. Quando lançamos uma Regra desta importância, que marca um estado de hipoglicemia e pede que se espere quinze minutos, para alguns utilizadores pode haver a tentação de fazer um registo antes do

tempo. Esta ação podia provocar um novo ciclo de descoberta/tratamento de uma hipoglicemia que já está a ser tratada. Para evitar esta situação desagradável, é feito um segundo bloqueio que impede que regras relacionadas com hipoglicemia sejam ativadas. Desta forma, quando o utilizador vai a fazer o registo programado, o bloqueio já foi retirado e as Regras podem funcionar como planeado. Podíamos apenas bloquear as Regras relacionadas com hipoglicemia e deixar que outros conselhos, de menor gravidade, sejam ativados, todavia, num estado de hipoglicemia, o utilizador deve estar focado em cumprir os conselhos que lhe foram dados. Por conseguinte enquanto este segundo bloqueio se encontrar ativo, nenhuma Regra relacionada com glicemia será ativada. Se o utilizador tentar fazer um registo de glicemia antes do tempo, irá ativar uma Regra especial que deteta o bloqueio e receberá um conselho para permanecer calmo, seguindo os conselhos que lhe foram fornecidos, e esperar que seja exibido o próximo menu de registo. Esta capacidade pode ser moldada de acordo com a vontade do médico cooperante, este caso foi apresentado de maneira a mostrar as potencialidades da opção de bloqueio, porém a sua utilização deve ser utilizada em conformidade com a opinião especializada do médico.

Chegamos a um ponto do protocolo em que fizemos as perguntas necessárias ao utilizador, e agora devemos aconselhá-lo de acordo com o que respondeu. A «Regra dos 15» dita as mesmas Regras para ambas as hipoglicemias (ligeira ou moderada), distinguindo-se apenas na quantidade de Hidratos de Carbono que o utilizador deve ingerir. Portanto, devemos agora, com novos elementos, diferenciar o aconselhamento a partir de duas Regras distintas:

- Uma RAR e uma Mensagem/Conselho que apoiam o utilizador em caso de Hipoglicemia Ligeira - *Lista de Código* 7.6 e *Lista de Código* 7.7, respetivamente;
- Uma RAR e uma Mensagem/Conselho que apoiam o utilizador em caso de Hipoglicemia Moderada - *Lista de Código* 7.8 e *Lista de Código* 7.9, respetivamente.

Deve ser tido em conta que a RAR representada pela *Lista de Código* 7.8 é apenas um exemplo em que o utilizador respondeu afirmativamente a todas as questões apresentadas. Contudo, esta Regra não invalida a criação de outras RAR que detetem combinações destes diferentes tipos de sintomas ou de mais sintomas que podem ser inseridos no sistema.

```
inRisk(end, glucose, glucose, 8, ID) :- ID = 'hipoglicemiaLig',
    hasHipoglicemia(lightModerate).
```

Bloco de Código 7.6: Regra de Avaliação de Risco para verificação de hipoglicemia ligeira

Como podemos ver, enquanto uma Regra verifica se existem os diferentes tipos de sintomas que caracterizam uma hipoglicemia moderada, a outra limita-se a verificar se o utilizador se encontra em estado de hipoglicemia. Dado que a Regra que deteta uma hipoglicemia moderada é mais específica e tem uma maior urgência, irá ficar acima da sua «rival», ou seja, se a primeira não ativar, a segunda tem a oportunidade de ser testada. Se nenhuma das duas for ativada, então o utilizador já não se encontra em estado de hipoglicemia e deve agora receber um

```
msg('hipoglicemiaLig', 'O seu valor glicémico encontra-se baixo.Por
    favor tome 15 gramas de Hidratos de Carbono(Dois pacotes de
    açúcar). Um novo teste irá ser agendado dentro de 15 minutos para
    verificar se este valor volta ao
    normal./adviceTimer-Glycemia-15-m/block-hipo-15-m').
```

Bloco de Código 7.7: Mensagem/conselho de alerta em caso de hipoglicemia ligeira

```
inRisk(end, glucose, glucose, 10, ID) :- ID = 'hipoglicemiaMod',
    hasHipoglicemia(lightModerate), has(dizziness), has(sweating),
    has(feebleness).
```

Bloco de Código 7.8: Regra de Avaliação de Risco para verificação de hipoglicemia Moderada

outro tipo de aconselhamento. Não podemos inserir uma Regra que ativa indiscriminadamente, pois isso iria causar um bloqueio para outras Regras. Isto acontece porque uma Regra, que lida com uma pos-hipoglicemia, tem um nível de gravidade acima de outras Regras, como as que lembram que o utilizador já não faz um certo registo hà algum tempo. Devemos definir condições que causem a Regra a ser ativada apenas nestes casos particulares, ou seja, temos duas condicionantes: o utilizador está bem (fora de hipoglicemia); ou o utilizador sofreu recentemente de uma hipoglicemia. Dada a conclusão feita anteriormente que se a Regra que estamos a criar chega a ser testada, deve-se ao facto de as Regras que determinam que o utilizador está a sofrer uma hipoglicemia falharam, logo o utilizador está bem. Resta-nos testar se recentemente o utilizador teve uma hipoglicemia e aconselhá-lo de acordo com a situação. A RAR que resulta desta linha de pensamento é a representada pelo *Lista de Código* 7.10. Esta Regra usa um RAE representado pelo Lista de Código 7.11 que é verdade se o parâmetro fornecido foi registado na aplicação recentemente. Por fim a mensagem criada, representada na Lista de Código 7.12, para aconselhar o utilizador, seleciona como meio de exibição ao utilizador um *popup* simples. Esta mensagem tem um parâmetro extra especial que diz à ANA para retirar o bloqueio que existe sobre o **ID lightHipoglicemia**, desta forma se o utilizador insurgir novamente numa hipoglicemia ligeira ou moderada, a Regra responsável por distinguir estas duas formas de

```
msg('hipoglicemiaMod', 'O seu valor glicémico encontra-se baixo, e
    segundo a informação que forneceu à aplicação, encontra-se em
    estado de Hipoglicemia Moderada. Por favor, tome 20 a 30 gramas de
    Hidratos de Carbono. Um novo teste irá ser agendado dentro de 15
    minutos para verificar se este valor volta ao
    normal./adviceTimer-Glycemia-15-m/block-hipo-15-m').
```

Bloco de Código 7.9: Mensagem/conselho de alerta em caso de hipoglicemia moderada

hipoglicemia pode ser novamente ativada. Da mesma forma são apagados os **factos** que definem os sintomas do utilizador, pois estes já não são necessários dado o utilizador agora estar bem. Posteriormente, num SABR mais avançado, pode ser possível, invés de simplesmente apagar os sintomas, o sistema guardar/usar as várias ocorrências destes sintomas em Regras que consigam um tipo de aconselhamento mais personificado.

```
inRisk(end, glucose, glucose, 9, ID) :- ID = 'posHipo',
    hadRecently(hipoglycemia).
```

Bloco de Código 7.10: Regra de Avaliação de Risco para verificação saída de estado de hipoglicemia

```
hadRecently(RegistryType) :- timeLastRegistry(RegistryType, Tempo),
    recentTimeDefenition(RegistryType, RecentTime), Tempo < RecentTime.
```

Bloco de Código 7.11: Regra de Avaliação de Estado para verificar se o utilizador registou recentemente uma refeição

```
msg('posHipo', 'Acabou de recuperar de uma Hipoglicemia. Por favor,
    vigie a sua glicemia, evite exercícios físicos, e faça as suas
    refeições a tempo.
    /simpleAdvice/unlock-lightHipoglicemia/unlock-sintomas').
```

Bloco de Código 7.12: Mensagem/conselho de alerta em caso de hipoglicemia moderada

Terminamos assim a fase de aconselhamento de uma hipoglicemia ligeira ou moderada. O utilizador encontra-se agora fora de perigo, e o SABR tem todas as suas regras preparadas para responder a uma nova crise que possa surgir.

7.2.1.3 Deteção e aconselhamento em caso de hipoglicemia sintomática grave

A Hipoglicemia sintomática grave tem um protocolo ligeiramente diferente. Segundo o protocolo médico, deve ser feito o seguinte:

- Injecção intra muscular ou sub-cutânea de *glucagon* na zona do deltóide ou na face anterior da coxa. Usar *glucagon* (IM ou SC) ou glicose. Se disponível, é preferível a administração de glicose;
- Melhoria clínica (estado de consciência) em dez a quinze minutos;
- Se o nível de consciência não melhorar, administrar glicose endovenosa (ev);
- Após recuperação, ingerir Hidratos de Carbono complexos e vigiar glicemias capilares.

Neste caso existem dois pontos no tratamento do utilizador que se encontra com uma hipoglicemia sintomática grave:

O primeiro ponto é semelhante à deteção feita nas RAR de hipoglicemia ligeira ou moderada. Porém, a hipoglicemia grave tem uma característica que pode influenciar todo o protocolo: o estado de consciência do utilizador. Se o utilizador não estiver consciente, não pode prosseguir com o protocolo, que desta forma não consegue ajudar o utilizador. Para determinar o estado de consciência do utilizador é usado um *popup* específico para tratar este tipo de casos, o *popup* ***urgentQuestionPopup***. Este *popup*, como explicado no Capítulo 4, questiona o utilizador se deseja enviar um alerta a um familiar. Se o utilizador responder, encontra-se consciente, se não responder em tempo útil, então encontra-se inconsciente e uma mensagem é enviada a um familiar para que o utilizador possa ser socorrido. Este sistema de envio de mensagens de alerta está pensado, contudo ainda não foi desenvolvido pelos responsáveis da componente ANA. Assim que este extra for desenvolvido, pode ser usado em situações reais pelo SABR. Para detetar e reagir a uma primeira situação de hipoglicemia grave, devemos inserir no SABR as seguintes regras:

- Uma RAR que deteta o estado de hipoglicemia grave - representado pelo Lista de Código 7.13;
- Uma Mensagem/Conselho para alertar o utilizador do seu estado grave e perguntar se requer que seja enviada uma mensagem a um familiar - representado pelo Lista de Código 7.14.

```
inRisk(end, glucose, glucose, 10, ID) :- ID = 'hipoHigh',
    hasHipoglicemia(high).
```

Bloco de Código 7.13: Regra de Avaliação de Risco que verifica se o utilizador se encontra em hipoglicemia grave

```
msg('posHipo', 'O seu nível de Glicemia encontra-se gravemente baixo.
    Por favor, administre \textit{glucagon} na zona do deltóide ou na
    face anterior da coxa. Deseja que um alerta seja enviado a um
    familiar?/urgentQuestionPopup-Glycemia-10-m/block-hipo-10-m').
```

Bloco de Código 7.14: Mensagem/conselho de alerta em caso de hipoglicemia grave

Após ser feita a questão ao utilizador, temos, como no caso da hipoglicemia ligeira ou moderada, dois possíveis resultados: o utilizador está consciente e deseja, ou não, que um familiar seja contactado; ou utilizador está inconsciente e não respondeu à questão. Se o utilizador continuar consciente, então devemos prosseguir com as regras que já temos. Não é feito o bloqueio da Regra que lança a questão, pois um caso desta gravidade requer que se continue a fazer a questão indicada, devido a, a qualquer altura, o utilizador poder desmaiar. Se o utilizador estiver

inconsciente, então a ANA é responsável por inserir um novo **facto has(unconsciousness)** que serve como sintoma para que outra Regra de deteção de hipoglicemia grave ser ativada. Uma Regra que detete, por um lado o estado de hipoglicemia grave, e por outro a presença de um **facto** que prova o estado de inconsciência do utilizador, pode não só enviar um *popup* para quem possa encontrar o telemóvel e assim ajudar o utilizador, como também pode tomar outras medidas, como fazer uma chamada de urgência, sendo esta última medida um caso ainda a discutir e a desenvolver.

O segundo ponto da verificação e aconselhamento de uma hipoglicemia grave é a deteção do estado de melhoria do utilizador que, não só se encontra melhor, como também saiu de um estado de hipoglicemia sintomática grave. Para conseguir este efeito podemos criar uma nova RAE que verifica esta situação. Neste caso a construção da nossa nova RAR será **inRisk(end, glycemia, glycemia, 10, ID) :- ID = 'outOfHipoGlicemiaH', hadHipoglicemiaRecently(high).**, onde **hadHipoglicemiaRecently(high)** será uma RAE como a representada na Lista de Código 7.15. A Regra agora criada usa um FPBD que retorna o número de hipoglicemias com um certo valor que o utilizador teve recentemente. Tanto o valor considerado para avaliar a hipoglicemia, como o valor temporal, são obtidos a partir de **Variáveis Globais**, que devem ser definidas se ainda não o estiverem. O FPBD usado está representado na Lista de Código 7.16.

```
hadRecently(hipoglicemiaHigh) :- minValue(glucose, MinGValue),
    maxTimeToBeRecent(glucose, MaxTime),
    numberOfRecentHypoglycemias(NumberHipo, MinGValue, MaxTime),
    NumberHipo>=1.
```

Bloco de Código 7.15: Regra de Avaliação de Estado para verificação de ocorrência recente de uma hipoglicemia grave

```
numberOfRecentHypoglycemias(HypoglycemiaCount, HypoglycemiaValue, TimeSpan
    ) :- runSql_with_args('select count (*) fromReg_BloodGlucose as g where
    g.DateTime in (select DateTime from Reg_BloodGlucose where (
    strftime("%s", "now") - strftime("%s" , DateTime)) < ~d ) and g.Value
    <= ~d order by count(*) desc limit 1 ', [HypoglycemiaValue, TimeSpan],
    HypoglycemiaCount).
```

Bloco de Código 7.16: Facto de Pesquisa à Base de Dados que retorna o número de hipoglicemias recentes de um valor dado

Ao analisar a RAE criada (Lista de Código 7.15) é possível ver que não é feita a verificação se o utilizador saiu de um estado de hipoglicemia. Isto acontece dada a forma como as Regras estão dispostas e são percorridas. Dado que esta Regra é responsável por uma situação de pós-hipoglicemia, não tem a mesma urgência que uma Regra que trate de uma hipoglicemia, logo deve ser colocada abaixo destas Regras. Ao fazer isto se o **YAP Android** chegar a esta Regra, significa que nenhuma Regra que deteta hipoglicemias foi ativada, o que nos leva a concluir que

o utilizador já não se encontra nesta situação. Resta agora criar uma nova Mensagem/Conselho com o **ID** que definimos (**outOfHipoGlicemiaH**) que alerte o utilizador para ingerir Hidratos de Carbono complexos e que vigie a sua glicemia capilar.

Tendo em conta a melhoria no estado do utilizador, o *popup* a exibir ao utilizador não precisa ser **adviceTimer**, pois um **simpleAdvice** pode ser suficiente. Todavia, esta decisão deve ficar ao critério do médico cooperante que irá decidir a relevância de ser feito um novo registo da glicemia do utilizador após haver uma melhoria do seu estado de hipoglicemia grave.

7.2.2 Aconselhamento em situações de Não-Crise

No ponto anterior foi descrito um quadro clínico de hipoglicemia, os vários estados que o utilizador pode ter, e como criar Regras que, seguindo protocolos médicos, podem reagir a estas mesmas situações. Neste ponto vamos analisar conselhos fora do esquema de urgência médica, conselhos que podem, e devem, ser dados em situações comuns do dia-a-dia do utilizador. Segundo os médicos especialistas, os doentes diabéticos de tipo 1 devem agir de forma diferente tendo em conta a sua atividade física. A categoria de atividade física pode ser dividida em:

- Atividade física planeada - O utilizador planeia antecipadamente o seu exercício físico. Este planeamento varia entre utilizadores. Existem utilizadores que cumprem um calendário exigente em que todas as horas do dia têm um objetivo a ser cumprido e não varia muito. Depois existem utilizadores que não têm um plano exigente e tomam medidas para um exercício que planeiam fazer mais tarde;
- Atividade física não planeada - O utilizador exercita sem nenhum tipo de planeamento. Este tipo de atividade física ocorre muitas vezes pois o utilizador se vê forçado a fazer um esforço, por exemplo uma caminhada, que não estava a planear fazer e não por pura negligencia do utilizador.

No caso do utilizador exercitar de forma não programada:

- Caso decida fazer exercício após a refeição, o utilizador deve administrar metade da dose de bólus prandial [2];
- Caso decidam fazer exercício antes da refeição, o utilizador deve ingerir uma dose suplementar de Hidratos de Carbono, isto se a glicemia capilar for menor ou igual a 100 miligramas por decilitro (mg/dL);
- Quando a dose de insulina não é reduzida, o utilizador deve ingerir entre dez e quinze gramas (10 g - 15 g) de Hidratos de Carbono por cada trinta minutos de exercício físico que fez.

[2] Após a refeição.

No caso do utilizador exercitar de forma pós-prandial programada:

- Deve reduzir o bólus prandial entre vinte e cinco a cinquenta por cento (20-50%) para níveis moderados de atividade física;
- Se a atividade é fatigante então o utilizador deve ingerir Hidratos de Carbono adicionais;

  No caso do utilizador usar bomba de insulina:

  – Em casos de exercício sustentado com duração superior a sessenta minutos, deve ser programada uma redução temporária da perfusão basal em vinte a quarenta por cento (20-40%);
  – Para evitar hipoglicemias pós-exercício, deve haver uma redução temporária da perfusão basal em vinte e cinco por cento (25%) durante algumas horas após o exercício físico.

Após analisar os dois protocolos, a maior dificuldade visível é a transcrição para Regra da componente **programado** de exercício. Se este atributo não for expressamente dado ao SABR, é muito difícil o sistema detetar que o exercício feito foi programado. Uma forma de conseguir descobrir que o exercício foi programado é analisar o historial do utilizador e procurar padrões de registo de exercício. O utilizador, tendo em conta que tem toda a atividade física programada, vai exercitar todos os dias à mesma hora, com maior ou menor atraso. Isto considerando que o utilizador não se esquece de fazer o registo do novo exercício.

Uma forma de conseguir esta análise é a partir de Regras que avaliam quando um utilizador faz um registo novo. Caso no dia anterior, por volta das mesmas horas, houve algum registo de exercício, então este deve ter sido planeado. O problema com esta abordagem é que periodicidade não implica que a atividade foi programada. Durante dois dias isolados pode ter acontecido que, as horas dos registos foram à mesma hora, porém o exercício feito não foi programado. Outra solução seria fazer uma avaliação mais alargada, onde invés de considerar apenas o dia anterior, consideramos os três ou quatro dias anteriores. Esta abordagem tem dois defeitos: um é o caso de o utilizador fazer exercícios programados e se isto acontecer, só ao fim de três ou quatro dias, se o utilizador realmente cumprir o seu calendário, é que vai receber o aconselhado correto; e o segundo é se o utilizador por alguma razão falha um dia de exercício. Nesse caso a condição iria falhar e todo o processo iria reiniciar. O utilizador teria então de treinar mais três ou quatro dias até começar a ser considerado como treino programado. Até este ponto o utilizador vai receber conselhos como se fizesse exercícios não programados.

Existe ainda um segundo tipo de exercício programado, a programação sem agenda, onde o utilizador programa que vai fazer exercício mais tarde e toma certas precauções. Esta ação não é constante no tempo, embora seja programada, o que torna impossível à aplicação a deteção automática deste exercício como programado.

A melhor opção seria, para os utilizadores que programam com agendamento os seus exercícios, fazer uma média entre os dias em que o utilizador fez exercício e o total de dias passados, seguindo

esta abordagem. Assim, mesmo havendo um ou dois dias de descanso de treinos, o sistema iria continuar a considerar os treinos como programados.

Na situação atual do projeto, entende-se que o mais razoável será uma de duas opções: criar um novo parâmetro no registo de novo exercício, onde o utilizador pode definir se o exercício foi ou não programado; ou "ignorar" o **facto** de o exercício ser programado, e aconselhar o utilizador na medida da informação disponível.

O restante processo resume-se à análise de se o exercício está a ser feito antes de uma refeição, depois de uma refeição ou em jejum (não aconselhável). Depois de determinada esta condição podemos aconselhar o utilizador consoante a intensidade do exercício que praticou e registou.

Tal como as RAR mais graves, analisadas no ponto anterior, as Regras necessárias para responder a situações comuns necessitam de argumentos e condicionantes. Dado o aconselhamento ser feito após o registo de um exercício, devemos atribuir a esta Regra o parâmetro **end**. Este tipo de aconselhamento é muito específico, portanto devemos atribuir como parâmetro e sub-parâmetro **exercise**. Quanto às condições de ativação, a Regra que estamos a desenvolver pode responder a dois casos: exercício antes da refeição; ou exercício após a refeição. Tendo em conta a simetria dos dois casos irá ser focada a construção da Regra que trata do caso do utilizador fazer exercício após a refeição. Para responder a esta situação foi criada a RAR representada na Lista de Código 7.17. Esta Regra usa a RAE representada na Lista de Código 7.11 que, ao receber como argumento **meal**, vai retornar se foi feito algum registo recentemente tendo em consideração o que significa «recentemente» para este tipo de registo. Isto é feito a partir do **facto** que implementa uma Variável Global **recentTimeDefenition(RegistryType, RecentTime)**, onde são definidos os tempos a partir dos quais se considera um registo «recente».

```
inRisk(end, exercise, exercise, 5, ID) :- ID = 'exerciseAfterMeal',
    hadRecently(meal).
```

Bloco de Código 7.17: Regra de Avaliação de Risco para aconselhar o utilizador caso este registe um exercício após uma refeição recente

Depois de definidas as Regras que vão responder à situação em causa, devemos definir a Mensagem/Conselho a ser exibida ao utilizador. A mensagem escolhida para aconselhar o utilizador foi a representada na Lista de Código 7.18. O tipo de Conselho a ser apresentado foi definido como um **simpleAdvice**, dado não existir a necessidade de obrigar o utilizador a fazer um novo registo.

```
msg('exerciseAfterMeal', 'Fez uma Refeição recentemente. Deve portanto
    administrar metade da dose de bólus prandial ./simpleAdvice/').
```

Bloco de Código 7.18: Mensagem/conselho de alerta em caso de exercício após uma refeição

A partir das Regras que criamos, é possível ver como é feito o desenvolvimento das várias Regras que compõem o SABR. Os protocolos médicos, seguindo o processo apresentado, são facilmente adaptados a um conjunto de Regras que, trabalhando entre si, conseguem alertar e aconselhar o utilizador nas situações que detetam. Os maiores problemas da criação de Regras para o SABR prendem-se com o facto de, por vezes, serem necessários dados que não é possível registar facilmente na aplicação. Padrões de vida e certas atividades frequentes podem ser detetadas usando Regras e imprimindo um maior ou menor rigor na determinação destes mesmos padrões. Contudo, o medo de deteção de falsos positivos faz com que Regras não sejam, para já, incluídas no SABR. Talvez no futuro, com a inclusão de um sistema de *Machine Learning*, se supere estas Regras através processos mais sofisticados que consigam ajudar e aconselhar melhor os utilizadores. Outro desafio que pode surgir é o encadeamento de Regras e os seus bloqueios. Como foi possível ver nos pontos em que foram criadas as Regras para deteção e aconselhamento de hipoglicemias, por vezes, para seguir certos protocolos, com pequenas particularidades, é necessário a criação de mais do que uma Regra e o bloqueio de outras Regras, para que os conselhos não se sobreponham. Este processo pode ser muito exigente para um médico não acostumado a este tipo de interações. Mesmo para um programador, este processo pode ser desafiante, e por isso, a criação desta interatividade entre regras, deve ser da exclusiva responsabilidade do programador que, após testar a interação das diferentes Regras, pode explicar o seu funcionamento ao colaborador médico para que este valide as decisões feitas.

7.3 Criação de Tarefas

A criação de tarefas é um processo mais direto que o até agora visto na criação de conselhos. As tarefas não têm precedências ou dependências, são «leis» que o utilizador deve cumprir para estar bem. Portanto, a sua criação é uma tradução de um conselho médico, de uma ação que o utilizador deve cumprir para um **facto** *Prolog* que vai implementar uma nova Regra.

Uma Tarefa que todos os utilizadores da aplicação devem fazer é um teste à glicemia de pelo menos três em três horas. Esta Regra é facilmente traduzida para o **facto** representado na Lista de Código 7.19. Esta tarefa verifica quando ocorreu o último registo de glicose, recolhe o valor do máximo que o parâmetro **glucose** pode estar sem registos, recolhe a hora atual e compara os valores. Se o tempo atual for maior que o tempo do último registo com a adição do tempo máximo possível sem serem feitos registos, a Tarefa é ativada e irá aparecer na lista de tarefas do utilizador.

```
hasTask(baseTask, Description) :- timeOfLastRegistry(glucose,
    LastRegTime), maxTimeUntested(glucose, MaxTime),
    getCurrentTime(CurrentTime), CurrentTime > LastRegTime + MaxTime,
    msg('Tsk_cholesterolReg', Description).
```

Bloco de Código 7.19: Exemplo de tarefa para registo de glicemia periodicamente

Desta forma podemos desenvolver tanto Tarefas úteis no dia-a-dia do utilizador, como Tarefas essenciais. O processo é direto, pois basta passar da Regra que o médico estipula para um **facto** que verifica se o utilizador está a cumprir o que é pedido.

Capítulo 8

Conclusão

Chegados a este ponto, podemos ver que o SABR tem potencial e pode, já com as regras que possui, ajudar muitos utilizadores. Entre estes utilizadores devem ser destacados os diabéticos mais jovens que ainda estão a aprender a conviver com a sua nova realidade. Ter um sistema móvel que os guie e aconselhe não só ajuda estes utilizadores, como também auxilia os adultos responsáveis por eles.

Em questão à criação de Regras, o entendimento do sistema pode não ser direto, porém dada a sua arquitetura e natureza, tanto colaboradores médicos como programadores, podem rapidamente adaptar-se a ele e ficar prontos a desenvolve-lo.

Durante o desenvolvimento do SABR foi possível obter um pouco de *feedback* por parte do Médico que colabora com este projeto, o Dr. Celestino Neves, Médico Endocrinologista, que compreendeu a tradução feita da «Regra dos 15», tanto que, após ver o conjunto de regras, fez a sugestão da adição de **sintomas** à RAR. A partir desta sugestão foi criada a versão final aqui presente e representada pela Lista de Código 7.8.

A compreensão e sugestão de parâmetros extra não é o mesmo que criar um largo conjunto de Regras. Contudo, a partir deste pequeno exemplo, já fica provada a exequibilidade do SABR.

Como foi possível verificar durante a produção de regras, por vezes certos protocolos médicos podem ter particularidades que os tornam de difícil tradução. Muitos problemas podem ser solucionados com bloqueio/desbloqueio de regras ou posicionamento na lista, mas outros problemas como obtenção de informação que não está na Base de Dados pode ser difícil ou até mesmo impossível de obter. Isto provoca que certos casos sejam ignorados ou que recebam um aconselhamento mais generalizado para uma determinada situação. Creio que receber um conselho geral é melhor do que deixar o utilizador perdido sem saber o que fazer, portanto a falha não é incapacitante.

8.1 Trabalho Futuro

O projeto aqui apresentado é, como já referido, um projeto em constante desenvolvimento. Novas Regras, protocolos, Tarefas e conselhos podem surgir e devem então, ser implementados no Sistema de SABR.

Opções como a possibilidade de alteração de valores erradamente inseridos na base de dados seriam benéficos para a aplicação que, para já, não tem como combater o mau uso por parte do utilizador. Uma possibilidade que poderia ajudar não só a combater o mau uso da aplicação, como a ajudar o utilizador a inserir mais facilmente os dados, seria a inserção a partir da comunicação entre dispositivos médicos e a aplicação. Desta forma os dados seriam preenchidos automaticamente sem o risco de erro humano.

Outra possibilidade já referida, é a de contactar um determinado número, para casos em que se deteta o estado de inconsciência do utilizador numa situação muito urgente.

Uma ideia que poderia ajudar em muito o utilizador seria a integração de *Machine Learning*. Desta forma o SABR poderia evoluir a partir dos registos do utilizador, ao contrário do que existe agora em que as Regras são estáticas. Desta forma seria possível fazer um aconselhamento personalizado e de maior qualidade.

De momento, o rumo a seguir e que deve ser executado o mais cedo possível, para que se consiga testar o sistema, é a integração do YAP na aplicação móvel. Desta forma torna-se realizável a elaboração de um teste com vários utilizadores. Para que desta forma se consiga obter um *feedback* dos futuros utilizadores para os quais desenvolvemos este sistema.

Bibliografia

[1] Dbee [sítio oficial]. http://dbees.com/. [Online; acedido a 2014-11-24].

[2] O que é a Diabetes? . http://www.apdp.pt/diabetes/a-pessoa-com-diabetes/o-que-e-a-diabetes. [Online; acedido a 2014-11-24].

[3] Glicontrol [sítio oficial]. http://www.glicontrol.pt/. [Online; acedido a 2014-11-24].

[4] Glucosebuddy [sítio oficial]. http://www.glucosebuddy.com/. [Online; acedido a 2014-11-24].

[5] HelpDiabetes [google app store]. https://play.google.com/store/apps/details?id=com.hippoandfriends.helpdiabetes. [Online; acedido a 2014-11-24].

[6] Minsulin [sítio oficial]. http://www.minsulin.com.br/. [Online; acedido a 2014-11-24].

[7] OnTrack Diabetes [google app store]. https://play.google.com/store/apps/details?id=com.gexperts.ontrack&hl=pt_PT. [Online; acedido a 2014-11-24].

[8] Saude24. http://www.saude24.pt/PresentationLayer/ctexto_00.aspx?local=15. [Online; acedido a 2014-11-24].

[9] Controlar a Diabetes. Entender a Diabetes. http://controlaradiabetes.pt/entender-a-diabetes/acerca-da-diabetes-mellitus. [Online; acedido a 2014-11-24].

[10] American Diabetes Association. Complications. http://www.diabetes.org/living-with-diabetes/complications/. [Online; acedido a 2014-11-24].

[11] Leite, C. S. F. (2014). *Gestão Pessoal de Diabetes –Sistema de Aconselhamento*. Master thesis, Faculdade de Ciências da Universidade do Porto, Porto, Portugal.

[12] Isabel Carriço. Farmácias ajudam doentes a controlar Diabetes. http://medicosdeportugal.saude.sapo.pt/utentes/diabetes/farmacias_ajudam_doentes_a_controlar_diabetes. [Online; acedido a 2014-11-24].

[13] Conftele. Conftele2015, 2015. [Online; acedido a 2015-09-12].

[14] Vítor Santos Costa, Ricardo Rocha, and Luís Damas. The YAP prolog system. *TPLP*, 12 (1-2):5–34, 2012. doi:10.1017/S1471068411000512.

[15] Anderson Faustino da Silva and Vítor Santos Costa. The design of the YAP Compiler: An Optimizing Compiler for Logic Programming Languages. *J. UCS*, 12(7):764–787, 2006.

[16] Gema García-Sáez, M Elena Hernando, Iñaki Martínez-Sarriegui, Mercedes Rigla, Verónica Torralba, Eulalia Brugués, Alberto de Leiva, and Enrique J Gómez. Architecture of a wireless personal assistant for telemedical diabetes care. *International Journal of Medical Informatics*, 78(6):391–403, 2009.

[17] Dr. David Kayne. Diabetes 1.5 – A Diabetes Hybrid. http://dev.healthfuldiabetes.com/diabetes-1-5-a-diabetes-hybrid/. [Online; acedido a 2014-11-24].

[18] Martha Nolte Kennedy. Calculating Insulin Dose. http://dtc.ucsf.edu/types-of-diabetes/type1/treatment-of-type-1-diabetes/medications-and-therapies/type-1-insulin-therapy/calculating-insulin-dose/. [Online; acedido a 2015-06-05].

[19] V. Santos Costa L. Damas. YAP User's Manual. http://www.dcc.fc.up.pt/~vsc/Yap/documentation.html. [Online; acedido a 2015-09-10].

[20] D.M. Machado, V.S.C. Costa, and P. Brandão. Mydiabetes - rule based advice system. In *Conf. on Telecommunications - ConfTele*, volume NA, pages NA – NA, September 2015.

[21] Colin D Mathers and Dejan Loncar. Projections of global mortality and burden of disease from 2002 to 2030. *Plos med*, 3(11):e442, 2006.

[22] RICHARD. A Better life for chronic patients. http://www.richardproject.eu/richard/diabete.html. [Online; acedido a 2014-11-24].

[23] Gary Scheiner. Insulin-to-carb ratios made easy, 2007.

[24] Wen-Tsai Sung and Kuo-Yi Chang. Evidence-based multi-sensor information fusion for remote health care systems. *Sensors and Actuators A: Physical*, 204(0):1 – 19, 2013. ISSN 0924-4247. doi:10.1016/j.sna.2013.09.034.

[25] SWI-Prolog. SWI-Prolog Reference manual. http://www.swi-prolog.org/pldoc/doc_for?object=manual. [Online; acedido a 2015-09-10].

[26] S. Vivekanandan and M. Devanand. Remote monitoring for diabetes disorder: Pilot study using indiatel prototype. *European Research in Telemedicine / La Recherche Européenne en Télémédecine*, (0):–, 2015. ISSN 2212-764X. doi:10.1016/j.eurtel.2015.04.002.

[27] Vodafone. Acompanhamento e Monitorização de Pessoas com Diabetes. http://www.vodafone.pt/main/A+Vodafone/PT/Fundacao/ProjectosIniciativas/Saude/diabetes. [Online; acedido a 2014-11-24].

[28] Karine Wenzel. Tecnologia catarinense ajuda pacientes diabéticos a controlar a glicemia. http://diariocatarinense.clicrbs.com.br/sc/geral/noticia/2014/11/tecnologia-catarinense-ajuda-pacientes-diabeticos-a-controlar-a-glicemia-4642573.html. [Online; acedido a 2014-11-24].

[29] WHO. Global status report on noncommunicable diseases 2014, 2014.

Apêndice A

Acrónimos

ANA Aplicação Nativa *Android*

APDP Associação Protetora dos Diabéticos de Portugal

EV Endovenosa

FPBD Facto de Pesquisa à Base de Dados

IM Intramuscular

IVR *Interactive Voice Response*

RAE Regra de Avaliação de Estado

RAR Regra de Avaliação de Risco

SC Subcutânea

SABR Sistema de Aconselhamento Baseado em Regras

SMS *Short Message Service*