



Universidade do Minho

Conselho de Cursos de Engenharia

1º Ciclo

Licenciatura em Engenharia Informática

3ºAno

Disciplina de Laboratórios de Informática IV

Ano Lectivo de 2010/2011

Projecto 6:

Desenvolvimento de uma aplicação para apoiar
a selecção de Software de Apoio à decisão.

Grupo 13

Ana Isabel Anjos Sampaio | 54740

Miguel Pinto da Costa | 54746

Hugo Emanuel da Costa Frade | 54750

Tiago Alves Abreu | 54772

Maio, 2011

Data de Recepção	
Responsável	
Avaliação	
Observações	

Projecto 6:

Desenvolvimento de uma aplicação para apoiar a selecção de Software
de Apoio à decisão.

Grupo 13

Ana Isabel Anjos Sampaio | 54740

Miguel Pinto da Costa | 54746

Hugo Emanuel da Costa Frade | 54750

Tiago Alves Abreu | 54772

Maio, 2011

Resumo

Este projecto consistiu no desenvolvimento de uma aplicação que tem como objectivo auxiliar os seus utilizadores na escolha de um *software* de suporte à decisão.

O trabalho pode ser dividido em 3 fases distintas: Recolha e Análise de Requisitos, Especificação do sistema e, por fim, a sua Implementação.

A aplicação final está funcional, cumprindo com sucesso o processo de suporte à decisão.

Área de Aplicação: Sistemas de suporte à decisão.

Palavras-Chave: software, critérios de decisão, AHP, ValueFn, SMART, comparação, base de dados, suporte à decisão.

Índice

1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	1
1.3. Motivação e Objectivos	2
1.4. Estrutura do Relatório	2
2. Estudo do Problema	3
2.1. <i>Software de Suporte à Decisão</i>	3
3. Objectivos do Projecto	5
4. Análise de Requisitos	6
4.1. Requisitos da Interface	6
4.2. Requisitos da Base de Dados	7
4.3. Requisitos a nível de métodos de selecção	7
5. Planeamento de Actividades	9
6. Especificação UML	10
6.1. Diagramas de Casos de Uso	10
6.2. Diagramas de Casos de Uso Refinados	12
6.3. Descrição Textual de Casos de Uso	15
6.4. Diagramas de Sequência	16
6.5. Diagrama de Classes	17
6.6. Esquema Relacional de Base de Dados	18
7. Implementação	20
7.1. Ambiente de trabalho e tecnologias utilizadas	20
7.2. Organização e Arquitectura da Aplicação	20
7.3. Base de Dados	21
7.4. Classes Implementadas	22
8. Módulo de suporte à decisão	25

9. Funcionamento da Aplicação	29
9.1. Início	29
9.2. Registar Utilizador	30
9.3. Processo de Escolha	30
9.4. Visualizar Páginas Web	35
9.5. Consultar Tutoriais	35
9.6. Sobre nós	36
9.7. Editar a lista de Softwares	36
10. Conclusões e Trabalho Futuro	37

Anexos

I. Anexo 1: Descrições Textuais	42
II. Anexo 2: Diagramas de Sequência	55
2.1. Diagramas de Sequência relativos às operações de Registo	55
2.2. Diagramas de Sequência relativos às operações de Consulta	60
2.3. Diagramas de Sequência Relativos às operações sobre a Base de Dados	62
2.4. Diagramas de Sequência relativos às operações de Comparação	67
III. Anexo 3: Tabelas da Base de Dados	75
3.1. Tabela de Utilizadores (User)	75
3.2. Tabela de Softwares (softwares)	75
3.3. Tabela de Características (characteristics)	77
3.4. Tabela com a relação entre Softwares e Características (software_list)	77

Índice de Figuras

Figura 1 - Ilustração do diagrama com a planificação de actividades. Realizado utilizando a ferramenta Microsoft Project.	9
Figura 2 - Diagrama de casos de uso geral	10
Figura 3 – Diagrama de casos de uso refinado “Data Base Subsystem”	12
Figura 4 - Diagrama de casos de uso refinado “Software Management Subsystem”	13
Figura 5 - Diagrama de casos de uso refinado “Decision Support Subsystem”	14
Figura 6 – Diagrama de Sequência “Select SMART method”	16
Figura 7 - Diagrama de classes: camada de negócio	17
Figura 8 - Diagrama de classes: camada de dados	18
Figura 9 – Esquema relacional da base de dados	18
Figura 10 – Janela de Login	29
Figura 11 – Janela de Registo	30
Figura 12 – Lista de Softwares Existentes	30
Figura 13 – Lista de características disponíveis para comparação	31
Figura 14 – Método Smart (atribuição de pesos)	32
Figura 15 - Método AHP (atribuição de pesos)	32
Figura 16 - Método AHP (teste à consistência)	32
Figura 17 – ValueFn (atribuição de pesos)	33
Figura 18 - AHP (atribuição de pesos)	33
Figura 19 - ValueFn (teste à consistencia)	34
Figura 20 – Ranking final dos softwares	34
Figura 21 – Janela de visualização dos websites do software	35
Figura 22 – Janela do tutorial do método AHP	35
Figura 23 – Janela sobre nós	36
Figura 24 – Janela de edição de softwares	36

Índice de Tabelas

Tabela 1 – Descrição do caso de uso “Select SMART method”.	15
Tabela 2 – Exemplo da representação dos valores possíveis para uma característica.	19

1. Introdução

Este relatório diz respeito ao projecto que constitui um sistema de suporte à decisão, designado beSMART

1.1. Contextualização

No âmbito da Unidade Curricular de Laboratórios de Informática IV, presente no último semestre do curso de Engenharia Informática, de acordo com o plano de estudos do mesmo, foi proposto ao nosso grupo, pela orientadora Anabela Tereso e pelo professor Orlando Belo, elaborar uma pequena aplicação que permita ajudar utilizadores (experientes ou não) a escolher *software* de Apoio à Decisão. Requer-se, assim, que tentemos simular o desenvolvimento de *software* para um cliente num contexto não académico, sendo o nosso cliente representado pela orientadora que sugeriu o problema.

1.2. Apresentação do Caso de Estudo

Hoje em dia, tomar decisões no mundo empresarial e não só requer muita responsabilidade, em que todas as escolhas têm de ser bem pensadas. Deste modo, torna-se cada vez mais necessário ter em conta todas as alternativas existentes no mercado, de forma a escolher a melhor de todas as disponíveis.

O número de *Softwares* de Apoio à Decisão tem vindo a aumentar. Alguns possuem aspectos que podem ser úteis num determinado contexto, mas outros talvez não seja a melhor opção para um dado efeito. Por isso, é útil ter uma ferramenta que receba as características que são relevantes para o utilizador e, pesquisando numa base de dados, indique qual a melhor alternativa para auxiliar na decisão.

Em suma, o problema em questão é criar uma ferramenta que classifique os *softwares* conforme os atributos pretendidos.

1.3. Motivação e Objectivos

Este projecto promete ser um objecto de trabalho de elevada importância para todos os elementos do grupo. Vai permitir-nos ter uma percepção de como será elaborar um grande projecto num contexto empresarial. Será interessante aprender novas linguagens de programação, bem como utilizar novas ferramentas, com as quais não estamos familiarizados. Esperamos adquirir um maior leque de conhecimento sobre as tecnologias existentes, algumas nossas desconhecidas e que, por sua vez, nos poderão facilitar muito o trabalho. Estamos motivados, e assim ansiamos continuar ao longo de todo o desenvolvimento da aplicação.

1.4. Estrutura do Relatório

Neste relatório propomo-nos a explicar todo o processo de desenvolvimento da referida aplicação. Começaremos por tecer um estudo do problema à escala global, querendo isto dizer que iremos esclarecer alguns conceitos sobre este *software* e, também, algumas aplicações do mesmo. De seguida, passaremos para uma contextualização do problema. Explicaremos em concreto a sua abrangência e o porquê do seu desenvolvimento. A fase posterior comporta a recolha de requisitos, sendo que, para tal, tivemos de nos reunir algumas vezes com o nosso cliente para conhecermos a sua pretensão e suas expectativas quanto a este projecto. A fase que se segue passa pela modelação do problema, utilizando a linguagem UML. Transformaremos, então, os requisitos em esquemas importantes, que nos apoiarão na construção da aplicação numa fase mais avançada. Por fim, a última etapa, remete-nos à implementação do software em questão, utilizando a plataforma .NET.

2. Estudo do Problema

Para compreender o nosso problema é necessário ter conhecimento de alguns conceitos que consideramos fundamentais.

2.1. *Software de Suporte à Decisão*

De uma maneira simples, podemos definir um *software* de Apoio à Decisão como um sistema que auxilie o utilizador no processo de tomada de decisão. Estes pretendem auxiliar na gestão, nas operações e no planeamento de uma organização e ajudar a tomar decisões, cujos parâmetros podem ser especificados de acordo com os interesses do utilizador. Para além disso, incluem sistemas baseados no conhecimento, isto é, sistemas que, usando uma série de regras definidas por quem desenvolve o *software*, simulam o conhecimento humano em relação a uma determinada área. Este tipo de produtos compilam informação útil a partir de dados desorganizados, documentos, conhecimento pessoal ou modelos de negócios para identificar e resolver problemas, tal como tomar decisões. A título de exemplo, informação comum que este *software* pode recolher e apresentar é a comparação de vendas entre um período e outro ou, ainda, o lucro projectado baseado nos valores esperados de vendas.

Existem três componentes que são a base principal de um *software* de apoio à decisão:

- a base de conhecimento,
- o modelo de decisão (o contexto da decisão e o seu critério),
- a interface de utilizador.

Teoricamente, estes *softwares* podem ser integrados em qualquer domínio do conhecimento humano. Como um exemplo simples e conhecido, pode ser referido o sistema de apoio à decisão médica, que ajuda os médicos no diagnóstico dos seus pacientes. O médico preenche a informação referente aos sintomas do seu paciente e o *software* tentará fornecer o diagnóstico mais acertado. Outro exemplo ocorre nas instituições bancárias, em que um funcionário tenta verificar se um cliente está apto a receber um empréstimo bancário.

2.1.1. Utilizadores deste Software

Produtos de *software* deste tipo são utilizados intensivamente no mundo empresarial, essencialmente para efeitos estratégicos. Estes permitem-nos uma mais rápida tomada de decisão, identificação de tendências negativas e melhor alocação dos recursos da organização.

2.1.2. Importância deste Software

Cada vez mais, no mundo empresarial, é necessário tomar decisões rápidas e acertadas para poder obter vantagem sobre a concorrência, num meio em que a competição aumenta a cada dia.

Também, por vezes, as variáveis que afectam uma decisão são inúmeras e complexas, pelo que um ser humano demoraria demasiado tempo a assimilá-las e a tomar uma decisão. No entanto, com este tipo de *software*, a sugestão sobre a decisão a tomar é fornecida de forma praticamente imediata, não existindo o risco de esquecer algumas variáveis ou reear cálculos errados, e possuindo toda a precisão de um programa informático.

3. Objectivos do Projecto

Este capítulo visa descrever os objectivos do projecto e apresentar um conjunto de metas inerentes ao grupo e aos conhecimentos que este pretende adquirir em relação à concretização do *software*:

- Desenvolver um sistema de *software* passando por todas as fases da engenharia de *software*;
- Aprender a utilizar novas ferramentas que podem economizar tempo na realização do projecto;
- Alargar a experiência de utilização do ambiente *Windows* e no uso de ferramentas da Microsoft;
- Desenvolver o sentido de responsabilidade em relação a prazos e acordos estabelecidos;
- Desenvolver competências na gestão de projectos;
- Desenvolver a dinâmica de grupo.

Relativamente ao conteúdo do projecto os objectivos do grupo são:

- Aprender a criar e a gerir uma base de dados dinâmica;
- Utilizar algoritmos multicritério para escolher a melhor solução de um leque de opções.

4. Análise de Requisitos

4.1. Requisitos da Interface

- Linguagem do Interface: Inglês
- Menus:
 - Devem constar obrigatoriamente as seguintes opções em relação à base de dados:
 - **New**: esta opção permite a criação de uma nova base de dados.
 - **Open**: esta opção permite carregar uma base de dados já existente.
 - **Save**: esta opção permite guardar a base de dados que está a ser utilizada.
 - **Save As**: esta opção permite guardar a base de dados que está a ser utilizada, podendo alterar certas características, como o nome do ficheiro.
 - **Exit**: esta opção permite sair da aplicação.
 - Devem constar obrigatoriamente as seguintes opções em relação aos *softwares*:
 - **Edit software list**: esta permite a gestão dos conteúdos da base de dados.
 - **View Software Web Page**: esta opção permite a visualização da lista com os nomes de todos os *softwares* (do lado esquerdo) e página web do respectivo software (restante espaço do ecrã).
- Opção de escolha entre *Basic DB* e *Extended DB*.
- Após a abertura da base de dados deve existir um menu que permita a gestão dos *softwares*.
- Na comparação de *software* o utilizador pode seleccionar:
 - as características que pretende comparar.
 - os softwares que pretende comparar.
 - os métodos que pretende usar para essa comparação.

4.2. Requisitos da Base de Dados

- A base de dados deve ser dinâmica, isto é, deve ser possível acrescentar ou remover características dos softwares que um utilizador pretender.
- Possibilidade de um utilizador criar a sua própria base de dados, escolhendo os campos que pretende.
- Devem existir duas bases de dados: base de dados simples e base de dados expandida. É nesta última que deve ser possível adicionar campos, podendo ser gravada separadamente das restantes bases de dados.
- As bases de dados têm que conter os seguintes campos:
 - *Basic Data Base*
 - *Software name* (limite de 60 caracteres)
 - *WebPage Link* (limite de 200 caracteres)
 - *Extended Data Base*
 - Inclui os campos da *Basic DB*
 - *Compatibility between Operating Systems* (Conteúdo: Yes/No)
 - *Cost of a license* (Conteúdo: Valor real em Euros)
 - *Interaction with user* (Conteúdo: Bad/Fair/Good/Very Good/Excellent)
 - *User Manual* (Conteúdo: Yes/No)
 - *Tutorials* (Conteúdo: Yes/No)
 - *Application Examples* (Conteúdo: Yes/No)
 - *Online Help* (Conteúdo: Yes/No)
 - *Free Version* (Conteúdo: Yes/No)

4.3. Requisitos a nível de métodos de selecção

Como já vimos anteriormente, o nosso sistema tem que classificar outros *softwares* de acordo com as preferências do utilizador. Para efectuar estes cálculos o utilizador pode escolher um dos três métodos disponíveis, dependendo da situação:

- **SMART**

- **AHP**
- **ValueFn**

O método **SMART** é uma técnica simples e rápida para decidir a prioridade das diferentes alternativas. Foi usado a primeira vez em Novembro de 1981. Consiste em atribuir pontos a cada alternativa, em que as mais importantes têm mais pontos do que as menos importantes.

O método **AHP** é uma técnica estruturada para lidar com decisões complexas. Este é baseado na matemática e na psicologia e foi criada em 1970 por Thomas L. Scoty, sendo estruturada e refinada desde essa altura.

O **AHP** é frequentemente usado por grandes equipas para resolução de problemas muito complexos. Eis alguns exemplos de onde o AHP pode ser aplicado:

- Ranking: Colocar as alternativas do mais para o menos desejado.
- Definições de Prioridades.
- entre outros...

O **ValueFn** corresponde a uma função que mapeia directamente a avaliação das alternativas, podendo ser maximizada ou minimizada (consoante a pretensão do utilizador em maximizar ou minimizar o atributo em causa).

5. Planeamento de Actividades

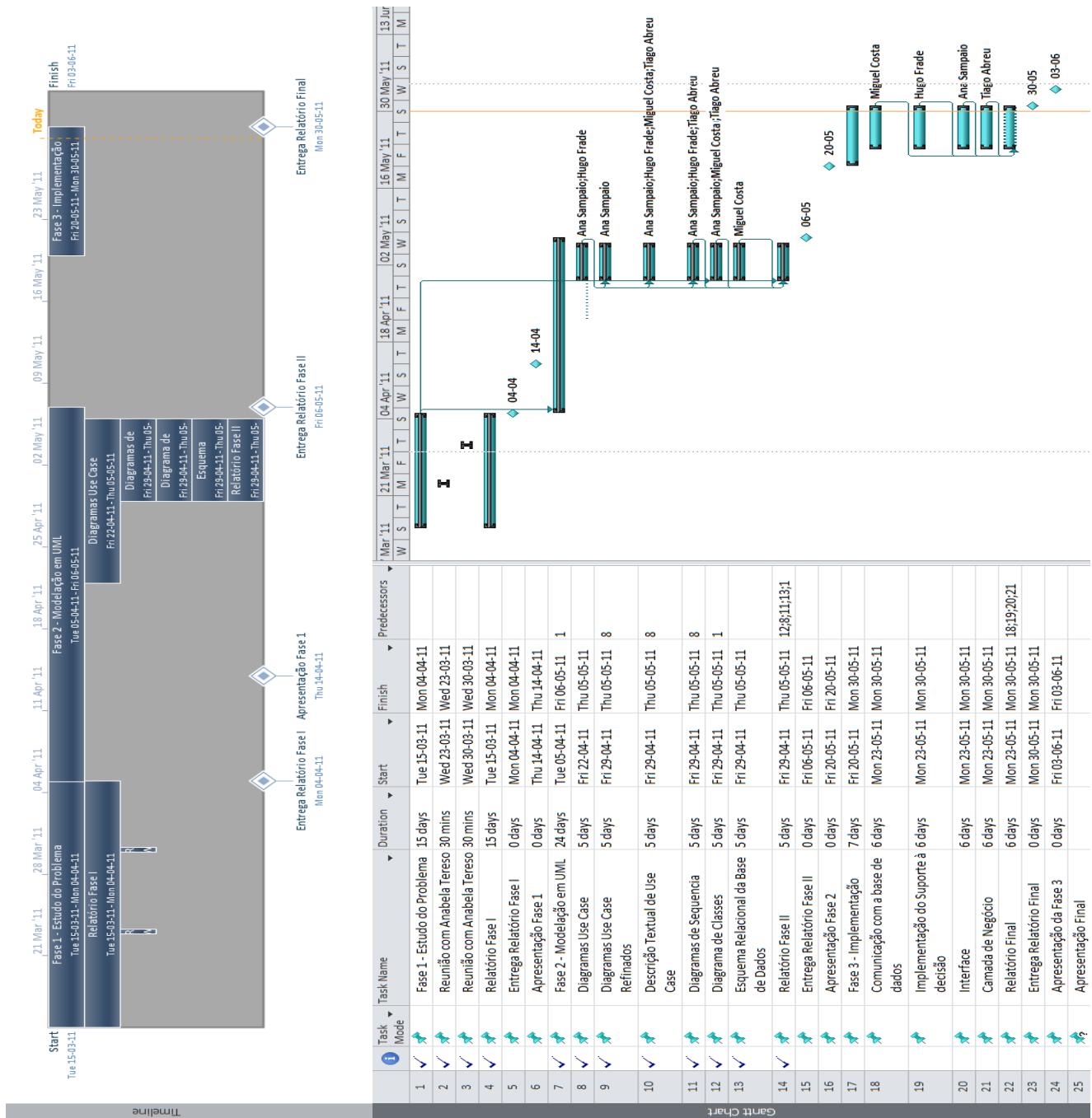


Figura 1 - Ilustração do diagrama com a planificação de actividades.

Realizado utilizando a ferramenta Microsoft Project.

6. Especificação UML

Após a recolha e análise dos requisitos necessários para a nossa aplicação, segue-se a fase de especificação de todo o Software.

Usamos diagramas de caso de uso, juntamente com os respectivos diagramas de sequência, para especificar todas as funcionalidades da aplicação. O diagrama de classes e o esquema conceptual da base de dados permitem estruturar a nossa camada de Negócio e a camada de Base de Dados, respectivamente.

6.1. Diagramas de Casos de Uso

Os principais casos de uso que um utilizador pode realizar estão representados no seguinte diagrama:

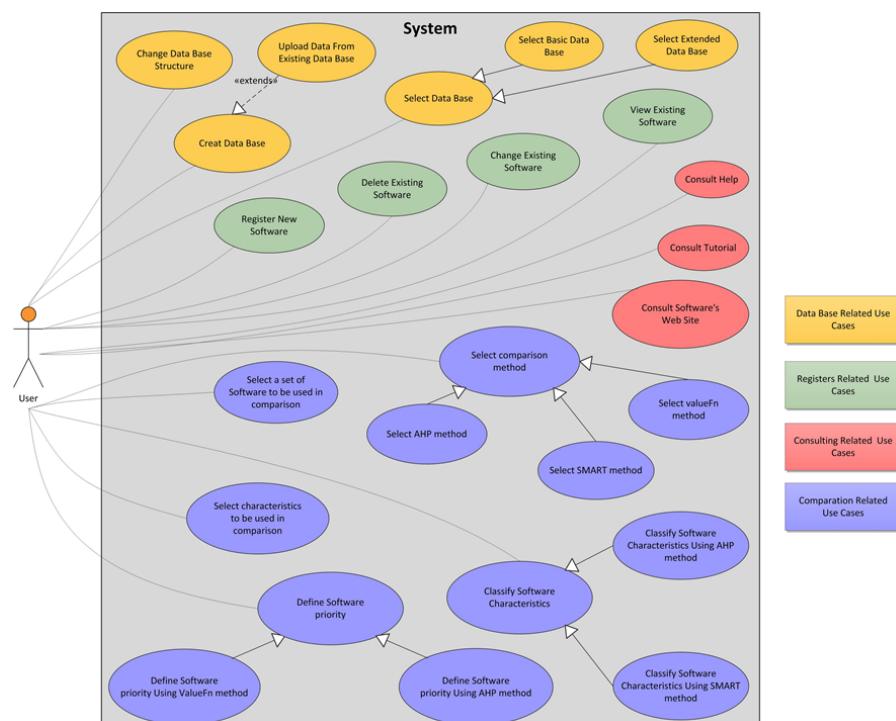


Figura 2 - Diagrama de casos de uso geral

Todas as funcionalidades da aplicação podem agrupar-se em 4 grupos distintos, mas relacionados entre si: *Data Base*, *Register Operations*, *Consulting* e *Comparation*.

O primeiro grupo, *Data Base Related Use Cases* corresponde a todas as acções que se podem efectuar sobre a base de dados. Essencialmente, um utilizador pode criar, seleccionar uma base de dados já existente e/ou alterar a sua estrutura. Podem distinguir-se os casos de uso mais relevantes, como a acção de alterar a estrutura da base de dados (*Change Database Structure*), que é dada a possibilidade ao utilizador de adicionar novas características aos seus softwares, ou de remover alguma característica já existente. Outra acção notória deste grupo é sobre as opções de visualização da base de dados de softwares, onde o utilizador poderá escolher se pretende uma vista simples (*Select Basic Database*) ou uma vista mais complexa destes (*Select Extended Database*). Também inserida neste grupo, está a possibilidade de um utilizador importar os dados de uma base de dados já existente ou criar uma nova base de dados.

Relativamente ao segundo grupo, *Registers Related Use Cases*, este consiste nas operações de inserção, remoção, consulta e edição dos softwares existentes na base de dados. Na edição dos softwares podem ser alterados os valores associados às suas características e na consulta dos softwares é possível visualizar todas as suas características e seus valores.

No grupo *Consulting Related Use Cases* é incluído toda a consulta um menu *Help* e também uma secção de tutoriais, que pode ser acedida através do caso de uso *Consult Tutorial*. Aí, são disponibilizadas páginas com os tutoriais disponíveis.

Por fim, os *Comparation Related Use Cases* são responsáveis por todo o processo de decisão do software mais adequado para uma determinada situação. Este grupo possui casos de uso que dão a possibilidade ao utilizador de escolher que softwares pretende usar na sua comparação, definir as características que pretende ter em conta no processo de tomada de decisão. Também neste grupo o utilizador toma a decisão de qual método pretende para o programa definir o melhor software, passando depois pelos processos de definição de prioridade destes e das suas características.

6.2. Diagramas de Casos de Uso Refinados

Analisado o Diagrama de Casos de Uso, verificamos que podemos de refinar algumas ações.

6.2.1. Data Base Management Subsystem

As tarefas relacionadas a base de dados e a sua gestão foram refinadas e estão ilustradas no diagrama seguinte:

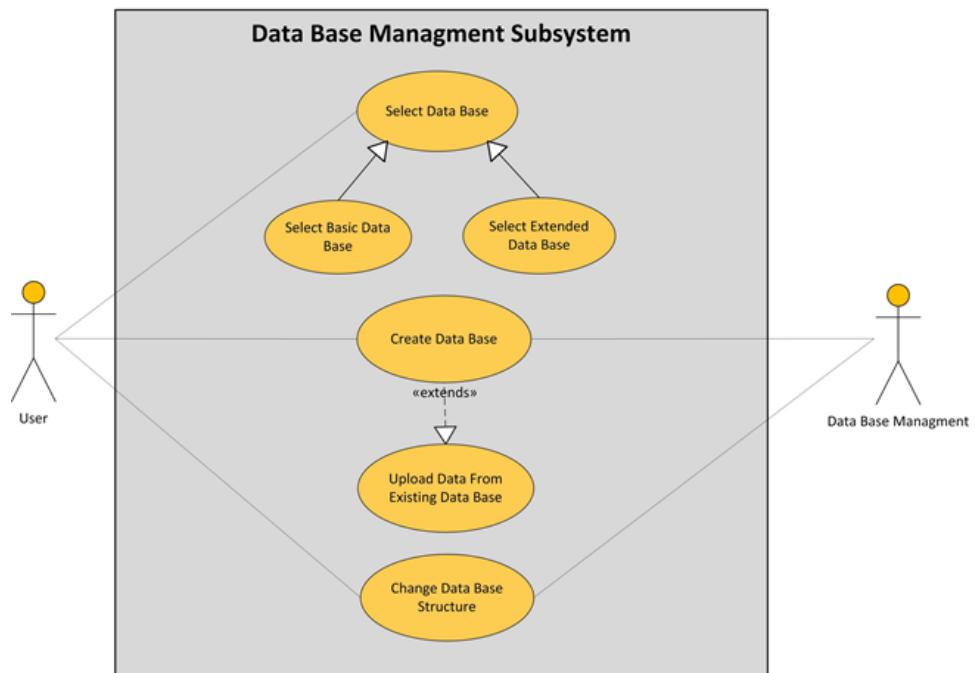


Figura 3– Diagrama de casos de uso refinado “Data Base Subsystem”

6.2.2. Software Management Subsystem

Tarefas relacionadas com os Softwares, podem ser agrupadas e refinadas. As tarefas como ver, registrar, alterar ou eliminar um Software envolvem as mesmas entidades:

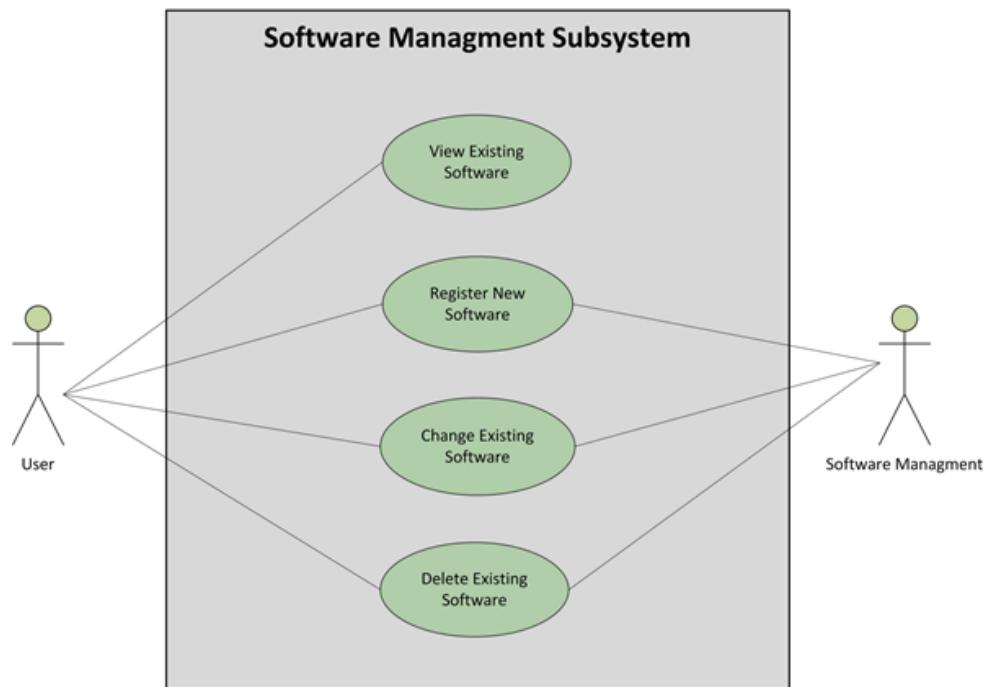


Figura 4 - Diagrama de casos de uso refinado “Software Management Subsystem”

6.2.3. Decision Support Subsystem

Todas as tarefas relativas à escolha de critérios de classificação e avaliação dos Softwares estão especificadas neste diagrama de casos de uso:

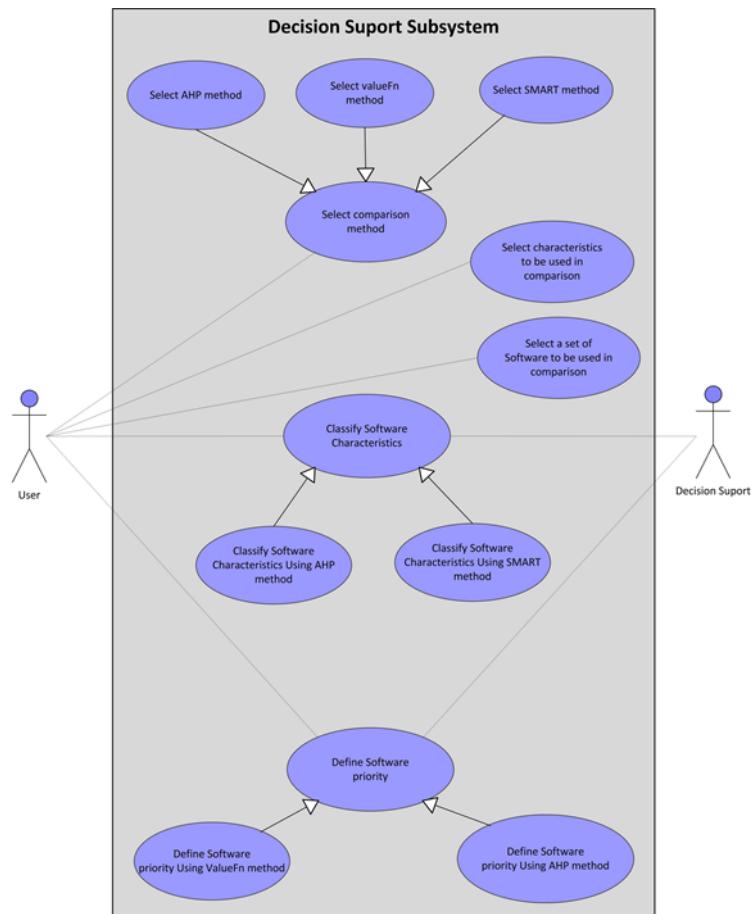


Figura 5 - Diagrama de casos de uso refinado “Decision Support Subsystem”

6.3. Descrição Textual de Casos de Uso

Após definidos todos os casos de uso, procedemos à descrição textual de cada um. Na ilustração abaixo apresenta-se a descrição textual do caso de uso “Select SMART method”. As restantes descrições podem ser encontradas no Anexo I.

USE CASE: Select SMART method		
Super Use Case	Select comparison method	
Author		
Date		
Brief Description	Choose the SMART method to compare the set of softwares	
Preconditions		
Post-conditions	The SMART method was selected	
Flow of Events	Actor Input	System Response
	1 Ask for the method list	
	2	Present the method list
	3 Choose the SMART method	
	4	Read the user choise
	5	Validate the user choise
	6	Report operation sucess

Tabela 1 – Descrição do caso de uso “Select SMART method”.

6.4. Diagramas de Sequência

Os diagramas de sequência permitem inferir sobre a sequência dos métodos necessários para cada acção. A título de exemplo, apresenta-se na figura abaixo o diagrama de sequência referente ao caso de uso “Select SMART method”. Os restantes diagramas encontram-se no Anexo II.

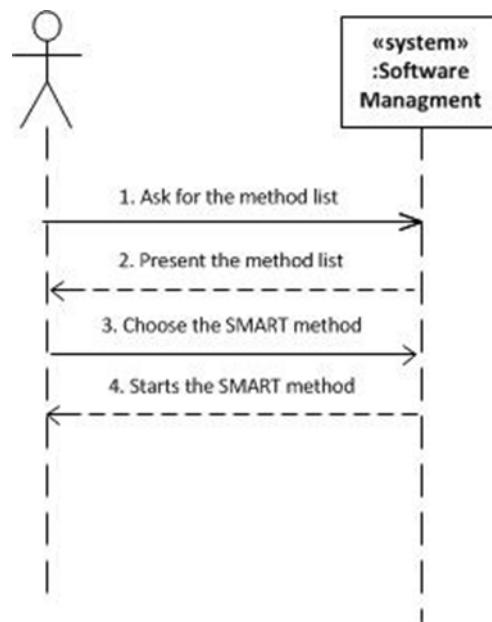


Figura 6 – Diagrama de Sequência “Select SMART method”

Após ser apresentada a lista de métodos existentes, o utilizador escolhe o método SMART. É comunicada a conclusão da operação após esta escolha ter sido validada.

6.5. Diagrama de Classes

O diagrama de classes permitiu-nos estruturar a nossa aplicação. Assim, neste diagrama foram designadas as classes, com as respectivas variáveis de instância e métodos. São também ilustradas as relações entre as classes.

Foi criada uma classe *Data Base*, que contém a informação de todos *softwares* que o *user* possui. Para gerir essa informação foi criada a classe *Data Base Management*.

A classe *Data Base* inclui o *User*, que corresponde há informação do utilizador, uma lista de Softwares (*software_list*) e uma lista de características (*charac*).

Characteristics é uma super classe que tem como subclasses *Yes/No Characteristics*, *Numeric Characteristics* e *Qualitative Characteristics*. Estas subclasses guardam a informação das classificações dos critérios que cada *software* possui.

A classe *Software* contém os campos e características em que o *software* está representado.

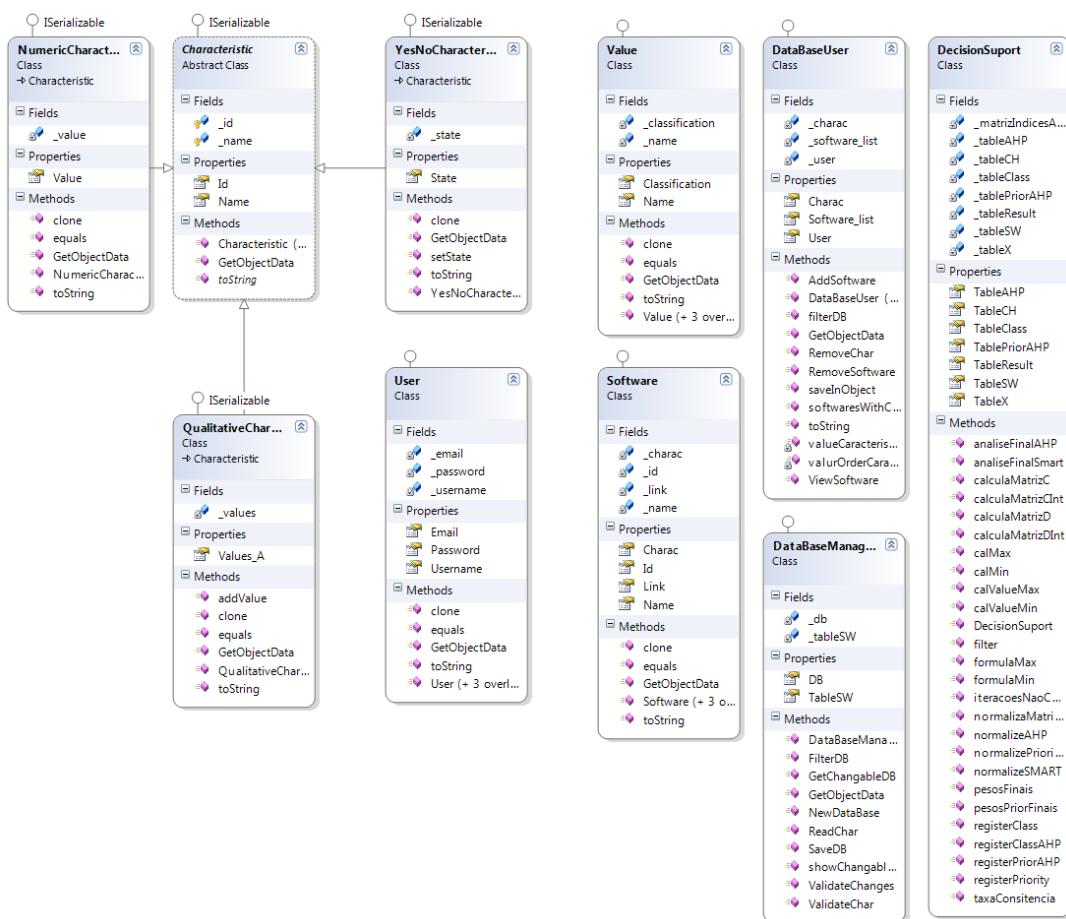


Figura 7 - Diagrama de classes: camada de negócio

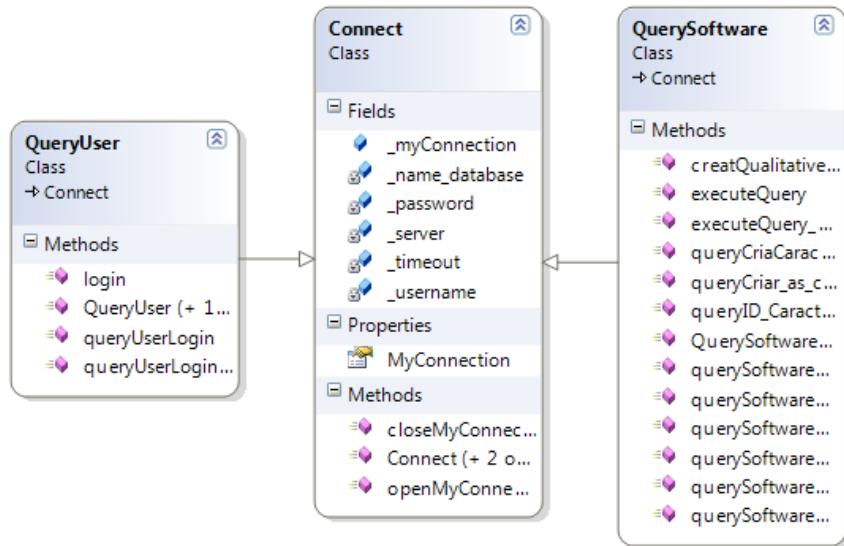


Figura 8 - Diagrama de classes: camada de dados

6.6. Esquema Relacional de Base de Dados

Para guardar a informação na base de dados de uma forma simples e funcional estruturamos da seguinte forma:

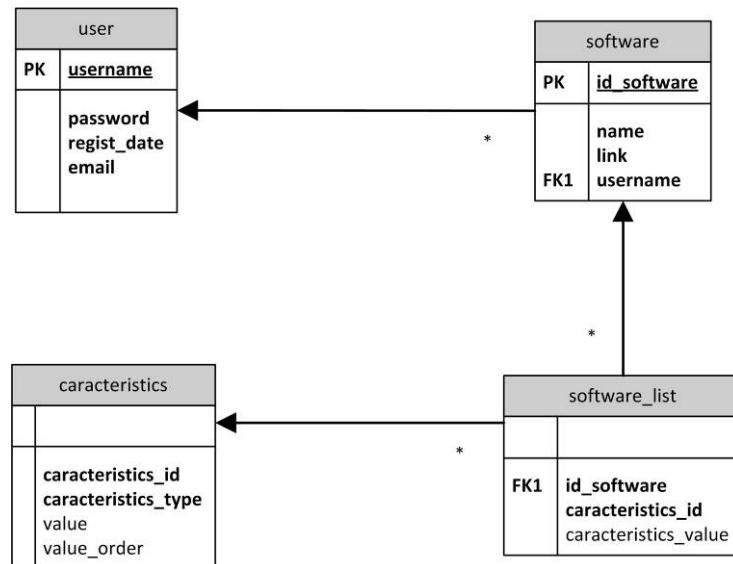


Figura 9 – Esquema relacional da base de dados

A tabela *user* é responsável por guardar toda a informação dos utilizadores. Os registos dos softwares são guardados na tabela *software*, em que cada registo contém o ID

(*id_software*), o nome (*name*), o respectivo *link* da página *web* e o *username* do utilizador que possui esse registo.

Quando um utilizador pretende adicionar mais um atributo para classificar os seus registos dos *softwares*, essa alteração será registada na tabela *software_list*, que contém o ID do software (*id_software*), a identificação do tipo de característica (*characteristics_id*), por exemplo: verdadeiro ou falso, e o valor da característica (*characteristics_value*).

A informação de cada característica está guardada em *characteristics*. Esta tabela contém o ID (*characteristics_id*) e o tipo (*characteristics_type*) da característica. Para as características qualitativas, há também o atributo *value* que contém um dos valores possíveis e a ordem de importância que têm, ou seja, algo como o exemplo seguinte:

<i>characteristics_id</i>	<i>characteristics_type</i>	<i>value</i>	<i>value_order</i>
14	qualitative	bad	1
14	qualitative	normal	2
14	qualitative	good	3

Tabela 2 – Exemplo da representação dos valores possíveis para uma característica.

7. Implementação

7.1. Ambiente de trabalho e tecnologias utilizadas

A nossa aplicação foi desenvolvida em ambiente Windows, utilizando o IDE Visual Studio 2010, da Microsoft. A linguagem de programação utilizada foi C#, com plataforma .NET.

Foi utilizado .NET Framework, que é um componente da Microsoft que permite integrar várias ferramentas e tecnologias, para que tenham um correcto funcionamento no seu conjunto. Deste modo, foi possível criar uma aplicação consistente e que envolve-se diversas ferramentas.

O motor de base de dados utilizado foi SQL Sever 2008, criado pela Microsoft em parceria com a empresa Sybase. Este sistema de base de dados fornece uma plataforma de dados confiável, produtiva e inteligente que permite executar correctamente a aplicação.

7.2. Organização e Arquitectura da Aplicação

A aplicação encontra-se devidamente organizada, de forma modular. Possui uma estrutura dividida em três camadas diferentes, cada uma com funcionalidades bastante distintas.

De seguida, será abordada cada uma das camadas que compõem a estrutura organizacional da aplicação.

7.2.1. Data Base

Esta é a camada que se encontra na base, responsável pela manipulação da base de dados. Deste modo, podemos considerá-la como ponte de ligação entre a base de dados e a aplicação.

7.2.2. Business

A camada de negócio diz respeito à camada intermédia da organização da aplicação, correspondendo à parte “lógica” do sistema.

7.2.3. Interface

A interface é a camada localizada no topo, responsável pela interacção do utilizador com a aplicação. Esta camada é a única que está perceptível ao utilizador, permitindo-o comunicar com o sistema. A interface foi construída a partir de *Windows Forms*.

7.3. Base de Dados

A nossa aplicação é constituída por 4 tabelas: *user*, *characteristics*, *software* e *software_list*.

7.3.1. Tabela User

Esta tabela contém os utilizadores registados no sistema. Sempre que uma nova entidade efectua o seu registo, os seus dados são armazenados nesta tabela.

7.3.2. Tabela Characteristics

Nesta tabela encontram-se as possíveis características para avaliar um *software*, bem como o tipo e, caso exista, a gama de valores que esta característica pode tomar.

As características existentes são: “*Compatible operating systems*”, “*Cost*”, “*Interaction with the user*”, “*Manuals and Tutorials*”, “*Examples through applications*”, “*on-line Help*” e, por fim, “*Free version*”.

7.3.3. Tabela Software

Esta tabela armazena todos os *softwares* existentes na base de dados. Para além do nome, inclui também o *link* para a página principal do *website* e a referência do proprietário do *software* (entidade que o inseriu na base de dados), representado por *username*.

Os *softwares* integrados no sistema possuem *user name* “simple_database” ou “extra_database”. Os *softwares* de “simple_database” são os *softwares* que correspondem à *Basic DB*. Analogamente, os *softwares* de “extra_database” são os *softwares* que pertencem à *Extended DB*.

7.3.4. Tabela Software_list

Nesta tabela estão representadas as relações existentes entre cada *software* e as suas características, em que cada uma destas entidades está representada pelo seu ID. A partir desta relação obtém-se o valor associado a uma característica, de um determinado *software*.

É de notar que apenas os *softwares* de “extra_database” estão incluídos nesta tabela, uma vez que são estes que possuem as suas características definidas.

7.4. Classes Implementadas

Nesta secção encontra-se uma breve descrição das classes que pertencem à camada de negócio e que são bastante relevantes para todo o processo lógico da aplicação.

7.4.1. Classe User

Esta é a classe que define as características de um utilizador. Estas características são: o nome de utilizador, o endereço de correio electrónico (*e-mail*) e a *password* respectiva. O utilizador necessita da sua *password* cada vez que pretender utilizar o software.

7.4.2. Classe Software

Esta a classe usada para definir um *software* que esteja incluído na base de dados. Esta possui três variáveis: ID (que é um valor único e é atribuído a cada *software* novo que é adicionado), nome e o *link* do seu *website*. Depois também possui uma estrutura,

nomeadamente um *Dictionary*, que contém as suas características. A chave do *Dictionary* é o ID da característica (ver Classe Característica) e o seu valor respectivo.

7.4.3. Classe Characteristic

A classe Characteristic é uma classe abstracta (pois nunca existirão instâncias dela) e também uma super-classe dos 3 tipos de características: características qualitativas, quantitativas e booleanas (ver QualitativeCharacteristic, NumericCharacteristic e YesNoCharacteristic, respectivamente). Esta define, portanto, o que é que os 3 tipos de características possuem em comum. Estas possuem um valor numérico denominado ID, valor único a cada característica e o seu nome.

7.4.4. Classe QualitativeCharacteristic

Esta classe define o tipo de característica que é constituído por valores qualitativos (como por exemplo: Mau, Razoável, Muito Bom). A cada palavra é atribuído o valor respectivo (definido na classe Value) e esses dados são armazenados num *Dictionary* denominado values.

7.4.5. Classe NumericCharacteristic

As características do tipo quantitativo são definidas por esta classe. Uma dessas características pode ser, por exemplo, o preço de cada software. A variável única desta característica é, portanto, o valor dela, definido por um inteiro na variável value.

7.4.6. Classe YesNoCharacteristic

As características booleanas (de apenas Sim ou Não), são definidas na classe YesNoCharacteristic. Possuem como única variável, designada state, um valor booleano que indica o estado da característica (*True* se Sim, *False* se Não). Este tipo de características é útil quando para saber se um software possui ou não alguma propriedade (como por exemplo se possui um módulo de Ajuda).

7.4.7. Classe Value

A classe Value é utilizada pela classe QualitativeCharacteristic para definir a relação entre o valor qualitativo (definido por uma palavra) e um valor numérico (para ser usado no processo de decisão). É através do valor numérico presente nesta classe que é definida a ordem de valor das características qualitativas na classe Qualitative Characteristic.

7.4.8. Classe DataBaseUser

Esta classe associa um utilizador (ver classe User) a um conjunto de *softwares* (ver classe Software) e também a um conjunto de características (ver classe Characteristic e respectivas sub-classes). Um utilizador possui, na sua base de dados, um conjunto de *softwares* e de características personalizado, permitindo que estes possuam características exclusivas, que não são acedidas por outros utilizadores que possuam os mesmos softwares na sua base de dados. Tanto os *softwares* como as características são armazenados numa estrutura *Dictionary*, em que a chave é o seu ID e o valor é um apontador para o respectivo objecto.

Os dados do utilizador a quem está associado essa base de dados são acedidos através de uma variável do tipo User, que apontará para o objecto respectivo.

7.4.9. Classe DataBaseManagement

A classe DataBaseManagement é utilizada para gerir a classe DataBaseUser. Esta, na variável `tableSW` possui um *Dictionary*, em que, a cada chave (que corresponde ao nome de um *software*), está associado um segundo *Dictionary*, sendo a chave o nome da característica e o valor respectivo. Está também associada um objecto do tipo DataBaseUser, na variável `db`.

8. Módulo de suporte à decisão

Para iniciar o processo de decisão, o utilizador deve escolher os *softwares* que pretende comparar. No mínimo, terá de escolher dois *softwares*. De seguida, é pedido ao utilizador que indique quais as características que pretende que o programa tenha em consideração nesse processo de decisão.

Aí o utilizador depara-se com uma escolha: atribuir pesos às características escolhidas utilizando o método SMART ou o método AHP.

Se escolher o método SMART, serão apresentadas as características e o utilizador deve atribuir 10 pontos à característica que achar menos importante. Depois terá de atribuir um número mais elevado de pontos às restantes características, consoante a sua importância relativamente ao atributo que possui 10 pontos (o menos importante). De seguida é necessária a normalização dos valores inseridos de seguida, isto é, divide-se cada valor pela soma de todos os valores inseridos e apresenta-se os resultados numa tabela, como no exemplo seguinte:

	Pontuação	Prioridades
Interacção com o Utilizador	10	$10/25 = 0.4$
Custo	15	$15/25 = 0.6$
Soma	25	1

O programa, quando o utilizador tiver terminado, lê a tabela vinda do interface, identificando o ID das características e a pontuação respectiva, através do método *registerClass()* da classe DecisionSupport chamando então, para a estrutura devolvida, um método da mesma classe, denominado *normalizeSMART()*, que normaliza essa tabela a devolve ao interface.

No caso de ter sido escolhido o método AHP, é apresentada ao utilizador uma tabela relacionando todas as características. Pretende-se, portanto, que o utilizador atribua um grau de importância de cada característica relativamente a outra. A diagonal principal da tabela

(que relaciona a característica com ela própria) encontra-se preenchida automaticamente. Os valores devem ser preenchidos abaixo da diagonal principal e devem respeitar a seguinte escala, que foi proposta por Saaty, embora possam estar presentes valores intermédios:

Se x é ... (do) que y	Então o nº de preferência a atribuir é:
Igualmente importante	1
Um pouco mais importante	3
Muito mais importante	5
Muitíssimo mais importante	7
Absolutamente mais importante	9

Para cada coluna é calculada a soma de todos os valores presentes nela. A matriz é, de seguida, normalizada e é calculada a soma dos valores contidos em cada linha. No final, é calculada a média desses mesmos valores ou seja, divide-se o valor da soma de cada linha pelo número total de características em comparação e esta é armazenada numa matriz denominada matriz de pesos finais.

De seguida, é calculada taxa de consistência para essa matriz. Isto é, inicialmente, a matriz que o utilizador preencheu é multiplicada pela matriz de pesos finais calculada anteriormente, obtendo-se assim uma matriz com apenas uma coluna e com um nº de linhas correspondente ao nº de características que serão comparadas. Então, é dividido cada valor dessa matriz pelo valor correspondente da matriz de pesos, que também foi usada na multiplicação anterior, obtendo-se um vector. Após essa operação, é calculada a média dos valores presentes nesse vector, sendo essa média uma aproximação ao maior valor próprio. É então necessário o cálculo do índice de consistência, que para uma matriz de tamanho N e utilizando a média calculada anteriormente, é dado pela seguinte fórmula:

$$IC = \frac{\lambda_{\max} - N}{N - 1}$$

Depois, é utilizada a tabela de Saaty com uma série de Índices Aleatórios, tabela essa que é apresentada de seguida:

N	1	2	3	4	5	6	7	8	9	10	11
IA	0.00	0.00	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49	1.51

Conforme o nº de características a ser comparadas, escolhe-se o valor de IA correspondente. É então feita a divisão entre o índice de consistência e o valor obtido da

tabela de IA. Se o valor dessa divisão, denominado Taxa de Consistência, for inferior a 0.1, a matriz é considerada consistente. Se porventura esse valor for superior, deve ser aplicado um método iterativo para melhorar a Taxa de Consistência.

Caso o valor seja superior a 0.1, será aplicado um método iterativo com vista a melhorar essa Taxa de Consistência.

Utilizando a matriz de pesos finais, multiplica-se esta pela matriz que contém os valores inseridos, obtendo-se o que se denomina uma matriz de pesos. De seguida, deve-se normalizar esta mesma matriz, sendo esta nova matriz de pesos finais. Deve-se então comparar esta nova matriz de pesos finais com a matriz que foi obtida anteriormente. Caso a diferença entre cada valor seja no máximo 0.0001, volta-se a calcular a Taxa de Consistência, da mesma forma que anteriormente. Se for superior, os passos serão repetidos, isto é, deve-se multiplicar esta última matriz de pesos finais pela matriz que contém os dados, normalizar a matriz resultante e voltar a verificar a diferença em relação à matriz de pesos finais anterior.

Após esta fase ter sido concluída com sucesso, começa uma nova a que se pode chamar Definição de Prioridades. Esta fase é principalmente orientada à característica. Para cada característica, o utilizador pode escolher métodos de selecção: AHP ou valueFn.

Caso o utilizador escolha o método valueFn, para cada característica, é pedido ao utilizador que escolha se pretende uma decisão em função do mínimo ou do máximo. Por exemplo, no caso do custo de um *software*, é normal um utilizador pretender que este seja o mínimo possível. Já no caso da qualidade de interacção com o utilizador, este estará interessando em que seja o máximo possível.

Então, para cada característica e para cada *software*, é aplicada uma de duas fórmulas, consoante se for pretendido maximizar ou minimizar uma característica. No caso da minimização é aplicada a seguinte fórmula:

$$y = \frac{x - Min}{Max - Min}$$

Já no caso da maximização, a fórmula é a seguinte:

$$y = \frac{Max - x}{Max - Min}$$

Os valores Min e Max são os valores mínimo e máximo presentes nos softwares escolhidos. No exemplo seguinte demonstra-se como são calculadas as prioridades utilizando o valueFn (num caso em que se pretende maximizar a característica):

Software	Interacção com o Utilizador		Prioridades	Prioridades Normalizadas
A	1		0	0.000
B	3		0.5	0.333
C	5		1	0.667
Min	1(A)	Soma	1.5	1
Max	5(C)			

Como se pode ver pela consulta da tabela, após serem calculadas as prioridades, a coluna onde estão armazenadas é normalizada.

Para um caso em que se pretenda minimizar a característica, segue um exemplo do processo que é aplicado:

Software	Custo		Prioridades	Prioridades Normalizadas
A	100		1	0.818
B	800		0.22	0.182
C	1000		0	0.000
Min	100 (A)	Soma	1.22	1
Max	1000 (C)			

A outra opção para o passo de definir as prioridades é utilizando o método AHP. Este funciona de maneira análoga ao explicado acima para o mesmo método, mas com a diferença de que a tabela, em vez de ser constituída pelas características, é constituída pelos *softwares* escolhidos.

São depois feitos os cálculos finais antes de se apresentar uma tabela com o *ranking* dos *softwares*. Inicialmente, no caso de ter sido utilizado o AHP, multiplica-se o valor do Software correspondente na matriz de pesos finais pela matriz obtida na definição das prioridades. Esta pode ter sido gerada ou pelo AHP ou pelo valueFn. Se na fase inicial tiver sido utilizado o método SMART, utiliza-se o valor final correspondente obtido neste método.

No final, somam-se todas as matrizes resultantes dessa multiplicação, sendo que o Software que obtiver o valor maior, será o que aparecerá no topo do *ranking*, e os restantes por ordem decrescente, sendo esse resultado apresentado ao utilizador.

9. Funcionamento da Aplicação

9.1. Início

Quando a aplicação é iniciada, surge a janela que está ilustrada na figura seguinte.

Para além de descrever sucintamente os passos que o processo envolve, possui as seguintes funcionalidades:

- Efectuar Login;
- Efectuar o Registo de um novo utilizador.

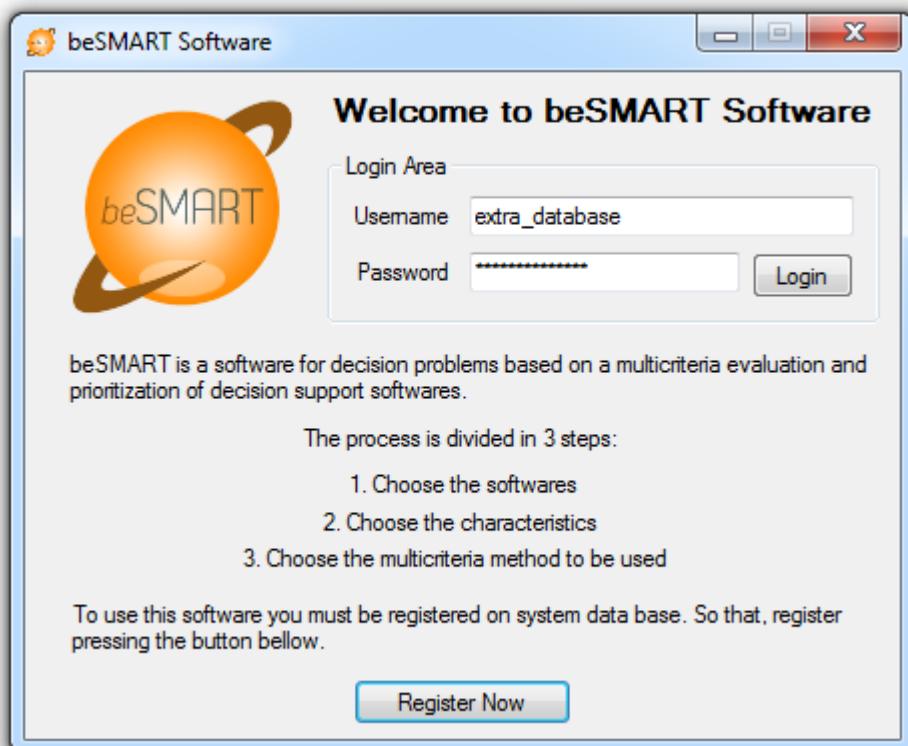


Figura 10 – Janela de Login

9.2. Registar Utilizador

Para o registo de um novo utilizador, é necessário o preenchimento dos campos que se seguem:

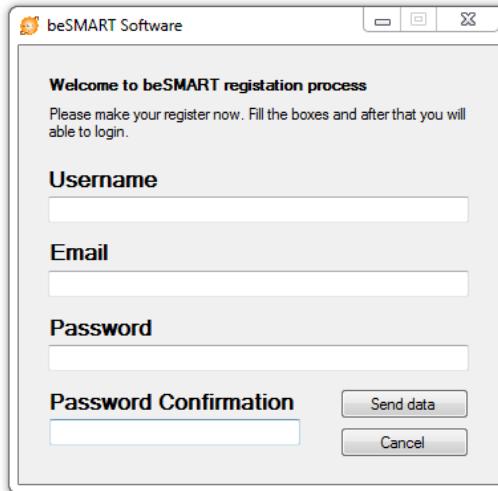


Figura 11 – Janela de Registo

9.3. Processo de Escolha

Após fazer *Login*, o utilizador inicia o seu processo de auxílio à escolha de um *software* de apoio à decisão.

9.3.1. Escolha dos Softwares

Na primeira etapa devem ser seleccionados os *softwares* pretendidos:

A screenshot of a Windows application window titled "beSMART Software". The menu bar includes "Database", "Software", "Help", "Choose Softwares", "Choose Criteria", "Definition of Weights", "Definition of Priorities", and "Final Results". A toolbar below the menu has icons for "New", "Open", "Save", "Print", and "Exit". The main area is titled "Softwares List" and contains a table with 30 rows of software information. The columns are: ID, Name, Link, Compatible operating systems, Cost, Interaction with the user, and Manuals and Tutorials. The table shows various software names like "Decision Explorer", "Criterion Decision Plus", "Decision Lab", etc., with their respective links and evaluation metrics. At the bottom of the table are buttons for "View Software WebPage" and "Next >".

ID	Name	Link	Compatible operating systems	Cost	Interaction with the user	Manuals and Tutorials
16	Decision Explorer	http://www.banxia.com/dexplore/	false	456	good	true
17	Criterion Decision Plus	http://www.infohavenwest.com/nroot/index.asp	false	636	very good	true
18	Decision Lab	http://idecisionlab.org.uk/	false	990	good	true
19	Electre II-V	http://www.sciencedirect.com/science/article/pii/S0377221796002524	false	519	acceptable	true
20	Electre IS	http://electre-is.software.informer.com/	true	519	good	true
21	Equity	http://www.catalyze.co.uk/?id=229	true	2143	good	true
22	Expert Choice	http://www.expertchoice.com/	false	1955	good	true
23	Hiview	http://www.catalyze.co.uk/?id=230	true	1100	good	true
24	Logical Decisions	http://www.logicaldecisions.com/	false	566	good	true
25	MACBETH	http://www.m-macheth.com/en/m-home.html	true	1750	good	true
26	OnBalance	http://www.quantistar.com/	false	640	good	true
27	PrefCalc	http://www.precalc.net/	false	0	acceptable	false
28	Prime Decisions	http://www.saltbk.tn/en/resources/downloadables/prime	false	0	good	true
29	SANNIA	http://sannia.com.br/	false	0	good	true
30	TOPSIS	http://www.topsis.com.br/	false	696	good	true

Figura 12 – Lista de Softwares Existentes

O botão “Next” permite a transição entre a primeira e a segunda etapa.

9.3.2. Escolha dos Critérios

Na segunda etapa o utilizador deve indicar os critérios que pretende utilizar no processo.

ID	Name
1	Compatible operating systems
2	Cost
3	Interaction with the user
4	Manuals and Tutorials
5	Examples through applications
6	on-line Help
7	Free version

Figura 13 – Lista de características disponíveis para comparação

9.3.3. Atribuição dos Pesos

Na terceira etapa o utilizador tem que descriminar o método que pretende aplicar para atribuir pesos aos critérios anteriormente escolhidos. A sua escolha recai entre os métodos AHP ou SMART.

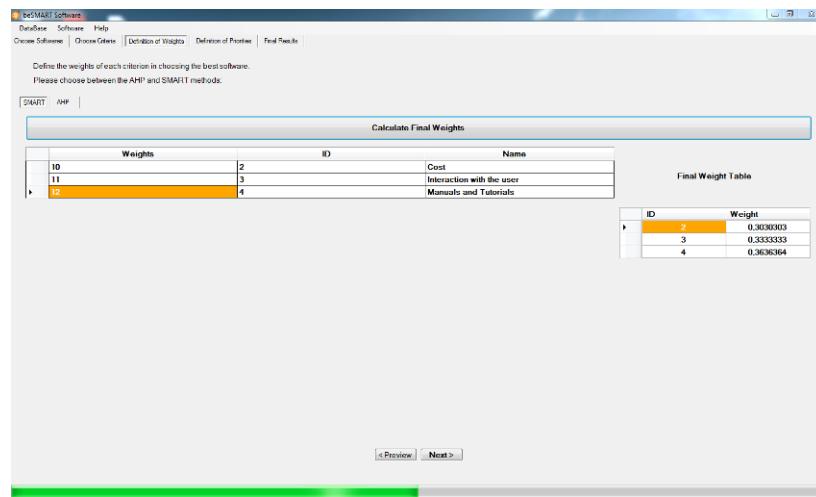


Figura 14 – Método Smart (atribuição de pesos)

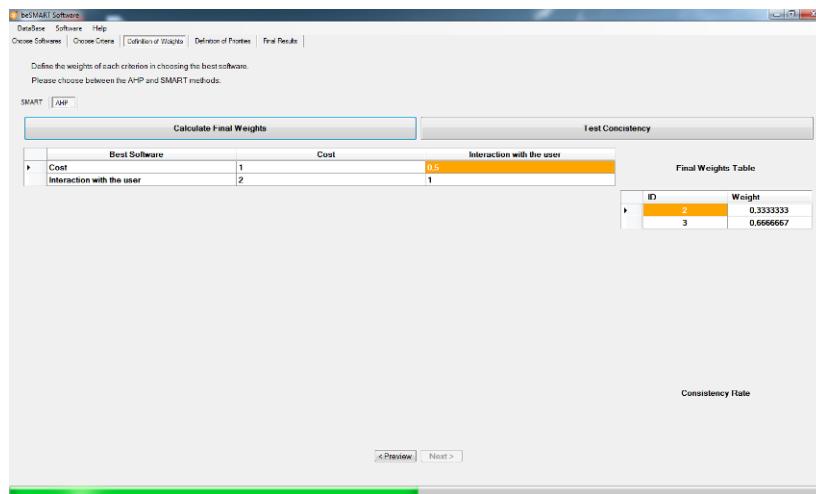


Figura 15 - Método AHP (atribuição de pesos)

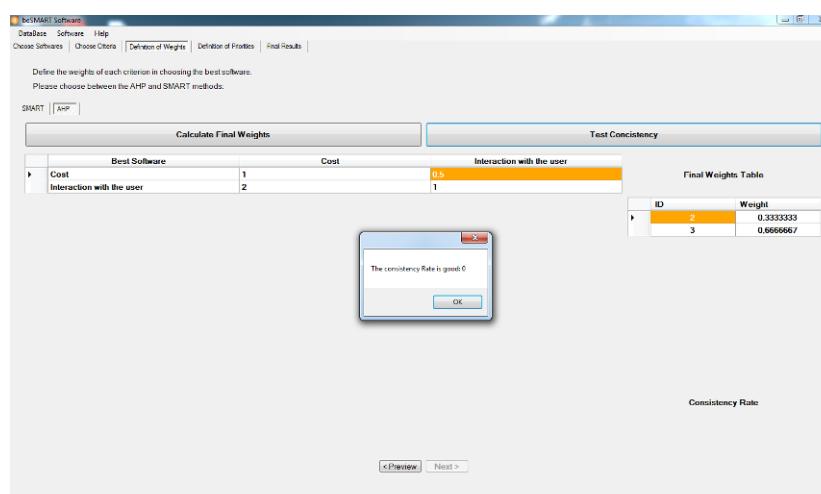


Figura 16 - Método AHP (teste à consistência)

9.3.4. Definição das Prioridades

Seguidamente, é necessário identificar o método que define a prioridade dos pesos de cada critério. Para cada um deles, o utilizador determina se quer utilizar o método AHP ou o método ValueFn, e atribuir os respectivos valores.

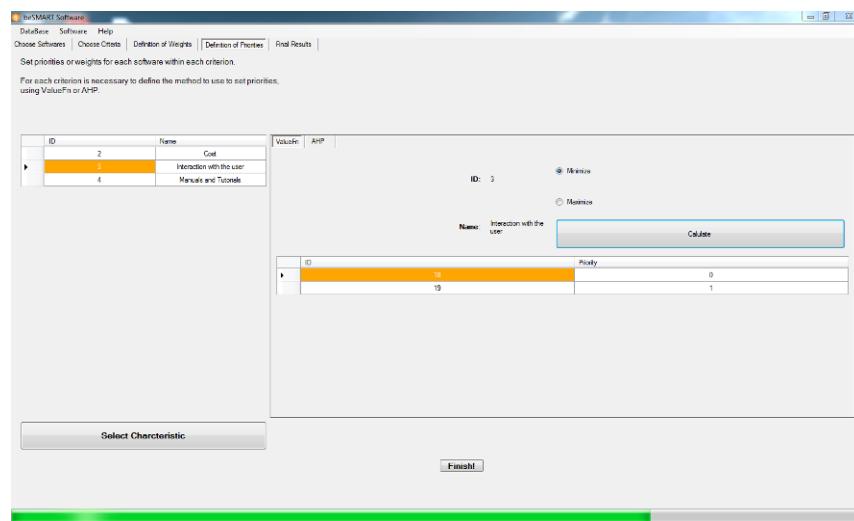


Figura 17 – ValueFn (atribuição de pesos)

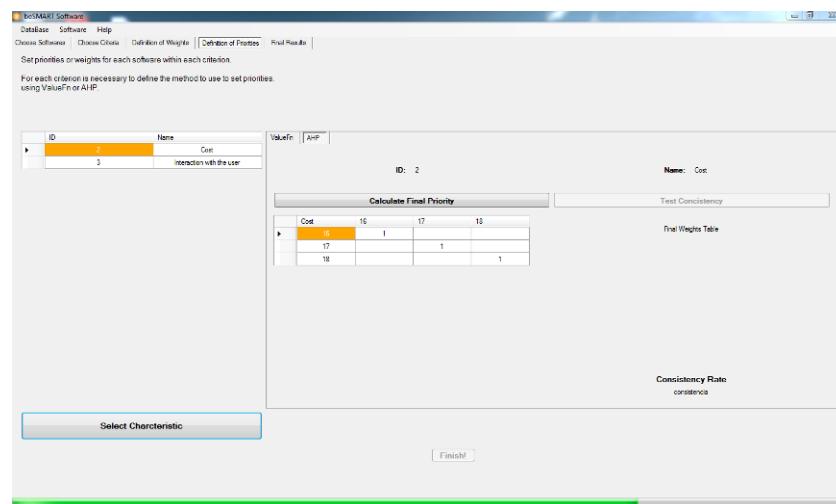


Figura 18 - AHP (atribuição de pesos)

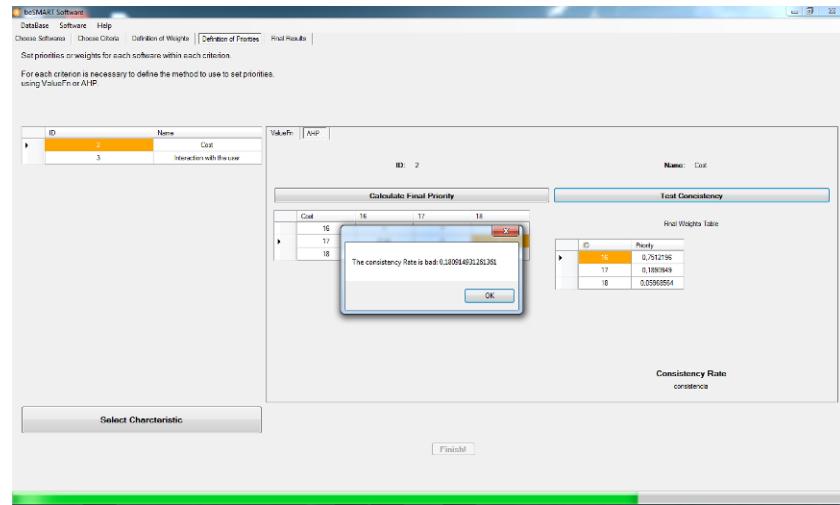


Figura 19 - ValueFn (teste à consistencia)

9.3.5. Resultados Finais

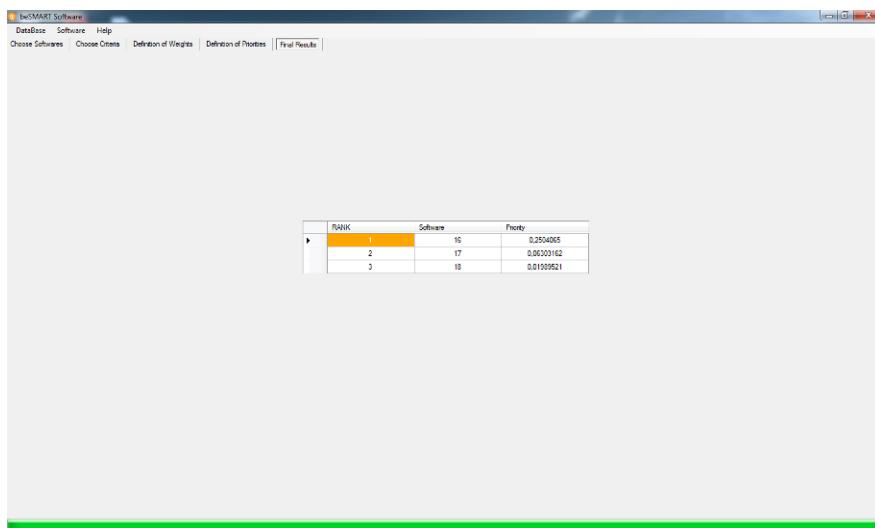


Figura 20 – Ranking final dos softwares

9.4. Visualizar Páginas Web

É possível visualizar a página web de cada software, no menu “Software”.



Figura 21 – Janela de visualização dos websites do software

9.5. Consultar Tutoriais

O utilizador pode consultar os tutoriais disponíveis, sobre os métodos de multicritério, no menu “Help”.

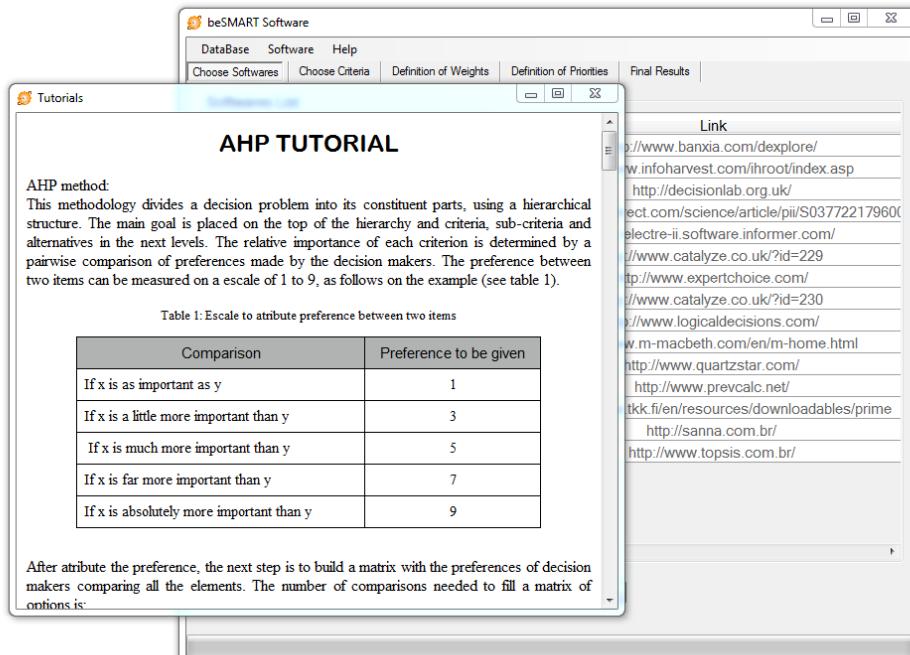


Figura 22 – Janela do tutorial do método AHP

9.6. Sobre nós

O utilizador pode consultar o Sobre Nós, no menu “Help”, seleccionando a opção “About”.



Figura 23 – Janela sobre nós

9.7. Editar a lista de Softwares

Nesta janela o utilizador consegue adicionar e remover um *software* da sua lista de softwares.

A screenshot of a Windows-style application window titled "Edit Software List". The window displays a table of software entries with the following columns: ID, Name, Link, Compatible operating systems, Cost, Interaction with the user, Manuals and Tutorials, Examples through applications, on-line Help, and Free version. The table contains 30 rows of data. At the bottom of the window are two buttons: "Add new Software..." and "Add new Characteristics".

ID	Name	Link	Compatible operating systems	Cost	Interaction with the user	Manuals and Tutorials	Examples through applications	on-line Help	Free version
16	Decision Ex...	http://www...	false	456	good	true	true	true	true
17	Criterion D...	http://www.i...	false	836	very good	true	true	true	true
18	Decision Lab	http://deosci...	false	990	good	true	true	true	true
19	Electre III-IV	http://www.ele...	false	519	acceptable	true	true	false	true
20	Electre IS	http://elecfcr...	true	519	good	true	true	false	false
21	Equity	http://www.e...	true	2143	good	true	true	true	true
22	Expert Choi...	http://www.e...	false	1955	good	true	true	true	true
23	Hview	http://www.h...	false	1040	good	true	true	true	true
24	Loupe Deci...	http://www.l...	false	556	good	true	true	true	true
25	MACBETH	http://www...	true	1750	good	true	true	true	true
26	OnBalance	http://www...	false	840	good	true	true	true	true
27	PrefCalc	http://www...	false	0	acceptable	false	false	false	true
28	Prime Deci...	http://www.p...	false	0	good	true	false	false	true
29	SANNA	http://sanna...	false	0	good	true	true	false	true
30	TOPSIS	http://www.t...	false	896	good	true	true	true	true

Figura 24 – Janela de edição de softwares

10. Conclusões e Trabalho Futuro

Após a concretização deste projecto, consideramos que adquirimos noções e rotinas que um projecto de desenvolvimento de um *software* envolve, principalmente nas primeiras fases da sua realização. Pela primeira vez, passamos por uma fase de levantamento de requisitos, em que não possuímos um enunciado concreto, que foram definidos em conjunto com o nosso cliente.

Na segunda fase de realização do projecto, definimos concretamente os comportamentos das funções que iriam responder aos requisitos pedidos. A abordagem desta fase iria ditar a estrutura do sistema, e ficaram definidos os métodos que seriam implementados na terceira fase: a fase em que o produto é finalmente criado.

Fazendo uma avaliação do nosso trabalho, verificamos que tivemos dificuldade na implementação de alguns requisitos, nomeadamente o módulo de Ajuda que, embora fosse um grande desafio, devido à sua complexidade não foi também possível implementar. Também não conseguimos concluir alguns métodos de manipulação da base de dados.

Apesar disso, considerámos o sucesso do funcionamento do módulo de suporte à decisão como um ponto positivo, pois a sua implementação foi bastante complexa e demorada. Também considerámos que a interface implementada é intuitiva para um utilizador que não conheça os métodos que são utilizados no processo de decisão.

Como trabalho futuro, e tendo em conta que se trata de um programa simples e funcional, a adição de novas funcionalidades, como por exemplo novas operações sobre os *softwares* presentes na base de dados ou até uma oferta mais vasta de softwares e características iria enriquecer a nossa aplicação. Poderíamos também ter implementado um módulo de Ajuda com um sistema de procura capaz apresentar ao utilizador os artigos mais adequados para que possa esclarecer sua dúvida, durante o processo de decisão.

Para concluir, julgamos que este projecto se revelou um grande desafio para todo o grupo, não só no que diz respeito à aquisição de conhecimentos de gestão de projectos, mas também em relação à dinâmica de grupo, que funcionou muito bem.

Bibliografia

Tereso A, Seixedo C, "Multicriteria Decision Aid: Evaluation and Comparison of the Main Tools", 2010.

Tereso A, Seixedo C, "A Multicriteria Decision Aid Software Application for selecting MCDA Software using AHP", 2010.

Seixedo C, "Dissertação apresentada para cumprimentos dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Sistemas, realizada sob a orientação científica da Doutora Anabela Pereira Tereso", 2009.

Referências WWW

[01] <http://msdn.microsoft.com>

Página da Microsoft onde podemos encontrar muitas informações sobre as tecnologias criadas por esta empresa. Inclui também tutorial em diversas linguagens de programação.

[02] <http://www.wikipedia.org/>

A Wikipédia é uma enciclopédia onde se encontra definição de conceitos dos mais variados temas.

Lista de Siglas e Acrónimos

Nesta secção é apresentada uma lista de siglas e acrónimos utilizados durante todo o relatório.

UML	Unified Modeling Language
DB	<i>Data Base</i>
SQL	<i>Structured Query Language</i>
IDE	<i>Integrated Development Environment</i>
ID	Identificação
AHP	<i>Analytic Hierarchy Process</i>
ValueFN	<i>Value Functions</i>
SMART	<i>Specific, Mesurable, Attainable, Relevant, Time-bound</i>
IA	Índices Aleatórios
IC	Índices de Consistência
TC	Taxa de Consistência

Anexos

Na secção Anexos encontram-se as descrições textuais e os diagramas de sequência que foram realizados na fase da especificação do sistema, e também as tabelas da base de dados, que realizamos na fase da implementação.

I. Anexo 1: Descrições Textuais

USE CASE: Register New Software		
Super Use Case	Do Not Exist	
Author		
Date		
Brief Description	Insert into Data Base a new software	
Preconditions		
Post-conditions	New Software Registered	
Flow of Events	Actor Input	System Response
	1 Ask to create a new software file	
	2	Ask to insert software data
	3 Insert software data	
	4	Read software data
	5	Verify software data
	6	Register software on system
	7	Report operation sucess
Exception 4a: Software Already Exists	Actor Input	System Response
	1	Report that software already exists
	2	Cancel Operation

USE CASE: Delete Existing Software		
Super Use Case	Do Not Exist	
Author		
Date		
Brief Description	Remove from Data Base a software	
Preconditions		
Post-conditions	Software was removed	
Flow of Events	Actor Input	System Response
	1 Ask for remove software	
	2	Ask to insert software data
	3 Insert software data	
	4	Read software data
	5	Verify software data
	6 Check that wants to remove software	
	7	Remove software from system
	8	Report operation sucess
Exception 4a: Invalid Software Data	Actor Input	System Response
	1	Report invalid software data
	2	Cancel Operation

USE CASE: Select a set of Softwares to be used in comparison		
Super Use Case		
Date		
Brief Description	Choose which softwares a client wants to use in a comparison	
Preconditions		
Post-conditions	The softwares were chosen	
Flow of Events	Actor Input	System Response
	1 Ask for the software list	
	2	Lists the softwares available for comparison
	3 Checks the softwares that user wants to use	
	4 Clicks on the Proceed Button	
	5	Read the choices
	6	Filters the database for the softwares chosen
	7	Report operation sucess
Exception 2a: Invalid Software Data	Actor Input	System Response
	1	Report invalid software data
	2	Cancel Operation
Alternative 5a: No Softwares Chosen	Actor Input	System Response
	1	Report no softwares chosen
	2	Returns to 2 of the Main Flow

USE CASE: Select AHP method		
Super Use Case	Select comparison method	
Author		
Date		
Brief Description	Choose the AHP method to compare the set of softwares	
Preconditions		
Post-conditions	The AHP method was selected	
Flow of Events	Actor Input	System Response
	1 Ask for the method list	
	2	Present the method list
	3 Choose the AHP method	
	4	Read the user choise
	5	Validate the user choise
	6	Report operation sucess

USE CASE: Select ValueFn method		
Super Use Case	Select comparison method	
Author		
Date		
Brief Description	Choose the ValueFn method to compare the set of softwares	
Preconditions		
Post-conditions	The ValueFn method was selected	
Flow of Events	Actor Input	System Response
	1 Ask for the method list	
	2	Present the method list
	3 Choose the ValueFn method	
	4	Read the user choise
	5	Validate the user choise
	6	Report operation sucess

USE CASE: Consult Help		
Super Use Case		
Author		
Date		
Brief Description	Consults the help and gets the article the user needs	
Preconditions		
Post-conditions	User has found help about his query	
Flow of Events	Actor Input	System Response
	1 Writes keywords about his query	
	2	Filters the Help database with his keywords
	3	Presents the search results, with links to the articles
	4 Clicks on a link to an article	
	5	Presents the article
	6 Finds the answer to his query	
	7	Report operation sucess
Alternative 6a: The user did not find an answer on the presented article	Actor Input	System Response
	1 Presses the Back button	
	2	Returns to 3 of the Main Flow
Alternative 4a: The user reached the end of the search results	Actor Input	System Response
	1	Returns to 1 of the Main Flow

USE CASE: Select the characteristics to be used in comparison		
Super Use Case		
Author		
Date		
Brief Description	Choose which characteristics a client wants to use in a comparison	
Preconditions		
Post-conditions	The characteristics were chosen	
Flow of Events	Actor Input	System Response
	1 Ask for the characteristics list	
	2	Lists the characteristics available for comparison
	3 Checks the characteristics that user wants to use	
	4 Clicks on the Proceed Button	
	5	Read the choices
	6	Filters the database for the characteristics chosen

	7		Report operation sucess
Exception 2a: Invalid Software Data		Actor Input	System Response
	1		Report invalid characteristics data
	2		Cancel Operation
Alternative 5a: No Softwares Chosen		Actor Input	System Response
	1		Report no characteristic chosen
	2		Returns to 2 of the Main Flow

USE CASE: Consult Tutorial			
Super Use Case			
Author			
Date			
Brief Description	The user consults the Tutorial List and chooses one		
Preconditions			
Post-conditions	The user found a tutorial		
Flow of Events		Actor Input	System Response
	1	Consults the list of Tutorials	
	2		Presents a list of links to the available Tutorials
	3	Clicks on a link to a tutorial	
	4		Presents the tutorial
	5		Report operation success

USE CASE: Consult Software's Web Site			
Super Use Case			
Author			
Date			
Brief Description	The user consults the web site for the chosen Software		
Preconditions			
Post-conditions	The web site is presented with success		
Flow of Events		Actor Input	System Response
	1	Presses the Consult WebSite button	
	2		Loads the WebSite in a side window
	3		Report operation success

USE CASE: View Existing Software		
Super Use Case		
Author		
Date		
Brief Description	View a software and all its characteristics	
Preconditions		
Post-conditions	The characteristics were presented with sucess	
Flow of Events	Actor Input	System Response
	1 Ask for view software	
	2	Ask to insert software data
	3 Insert software data	
	4	Read software data
	5	Verify software data
	6	Presents the Software data
	7	Report operation sucess
Exception 4a: Invalid Software Data	Actor Input	System Response
	1	Report invalid software data
	2	Cancel Operation

USE CASE: Change Existing Software		
Super Use Case		
Author		
Date		
Brief Description	Change the characteristics of an existing Software	
Preconditions		
Post-conditions	The characteristics were changed with success	
Flow of Events	Actor Input	System Response
	1 Ask for edit software	
	2	Ask to insert software data
	3 Insert software data	
	4	Read software data
	5	Verify software data
	6	Presents the Software data
	7 Changes the values of the characteristics	
	8 Submits the changed data	
	9	Read changed data
	10	Verify changed data
	11	Save changed data
	12	Report Operacion Sucess

		Actor Input	System Response
Exception 4a: Invalid Software Data	1		Report invalid software data
	2		Cancel Operation
Alternative 10a: Invalid changes		Actor Input	System Response
	1		Report invalid changes
	2		Return to 6 of the Main Flow

USE CASE: Select Basic Database		
Super Use Case		
Author		
Date		
Brief Description	Select and view the Basic Database	
Preconditions		
Post-conditions	The Basic Database was presented with success	
Flow of Events		Actor Input
	1	Selects the Basic Database button
	2	Filters the database for the Basic Database fields
	3	Presents the Basic Database
	4	Report Operation Success
Exception 2a: Invalid Software Data		Actor Input
	1	Report invalid software data
	2	Cancel Operation

USE CASE: Select Extended Database		
Super Use Case		
Author		
Date		
Brief Description	Select and view the Extended Database	
Preconditions		
Post-conditions	The Extended Database was presented with success	
Flow of Events		Actor Input
	1	Selects the Extended Database button
	2	Organizes the database to be presented
	3	Presents the Extended Database
	4	Report Operation Success
Exception 2a: Invalid Software Data		Actor Input
	1	Report invalid software data
	2	Cancel Operation

USE CASE: Upload Data From Existing Data Base		
Super Use Case		
Author		
Date		
Brief Description	Upload data from an existing data base file	
Preconditions		
Post-conditions	The file choose was uploaded	
Flow of Events	Actor Input	System Response
	1 Ask for the file list to select the file to upload	
	2	Show the file list
	3 Select the file that wants to upload	
	4	Read the file
	5	Verify Data
	6	Report Operation Success
Exception 4a: Impossible to read the file	Actor Input	System Response
	1	Report that was impossible to read the file
	2	Cancel Operation

USE CASE: Change Database Structure		
Super Use Case		
Author		
Date		
Brief Description	Edit the attributes of the softwares	
Preconditions		
Post-conditions	The Database was edited with success	
Flow of Events	Actor Input	System Response
	1 Selects the Change Database Structure button	
	2	Organizes the database to be presented
	3	Presents the Database
	4 Chooses to add a characteristic	
	5	Asks the type of the value
	6 Chooses the type of value	
	7 Choose default value	
	8	Reads the value type field and the default value field
	9	Adds a Column and fills it with the default value

	10	Changes the value associated with the desired softwares	
	11	Submits the changes	
	12		Reads the new column values
	13		Validates the changes
	14		Asks for continue editing
	15	Answers No	
	16		Saves the Database
	17		Report Operation Success
Alternative 4a: Remove characteristic		Actor Input	System Response
	1	Chooses to remove a characteristic	
	2		Ask for the characteristic to be removed
	3	Chooses the characteristic to be removed	
	4		Reads the field
	5		Validates the characteristic
	6		Ask for confirmation
	7	Answers Yes	
	8		Removes the Column of the characteristic chosen
	9		Returns to 14 of the Main Flow
Alternative 7a: No default value inserted		Actor Input	System Response
	1		Reads the value type field and the default value field
	2		Adds a Column and fills it with the a "NULL" value
	3		Returns to 10 of the Main Flow
Alternative 15a: Continues Editing		Actor Input	System Response
	1	Answers Yes	
	2		Returns to 3 of the Main Flow
Alternative 7aa: The answer is No		Actor Input	System Response
	1	Answers No	
	2		Returns to 3 of the Main Flow
Exception 2a: Invalid Software Data		Actor Input	System Response
	1		Report invalid software data
	2		Cancel Operation

USE CASE: Classify Software Characteristics Using SMART method		
Super Use Case	Classify Software Characteristics	
Author		
Date		
Brief Description		
Preconditions	SMART method was selected	
Post-conditions	Characteristics was classified according the SMART method	
Flow of Events	Actor Input	System Response
	1 Select the software characteristic	
	2	Read Characteristic ID
	3 Check characteristics importance	
	4 Give 10 points	
	5	Read points given
	6	Register classification
	7	Increase the number of characteristics classified
	8	Check if the number of characteristics classified is equals to the number of characteristics selected
	9 Validate classification	
	10	Report Operation Success
Alternative 4a: Characteristic is not the least important	Actor Input	System Response
	1 Give more than 10 points according to the importance	
	2	Returns to 5 of the Main Flow
Alternative 8a: The numbers aren't equals	Actor Input	System Response
	1	Returns to 1 of the Main Flow

USE CASE: Classify Software Characteristics Using AHP method			
Super Use Case	Classify Software Characteristics		
Author			
Date			
Brief Description			
Preconditions	AHP method is selected and the characteristics to be used in comparasion are selected		
Post-conditions	Characteristics are classified according the AHP method		
Flow of Events		Actor Input	System Response
	1	Select a characteristic	
	2		Read characteristic ID
	3	Give classification number according to the scale	
	4		Read points
	5		Register classification
	6		Increase the number of characteristics classified
	7		Check if the number of characteristics classified is equals to the number of characteristics selected
	8	Validate classification	
	9		Report Operation Success
Alternative 7a: The numbers aren't equals		Actor Input	System Response
	1		Returns to 1 of the Main Flow

USE CASE: Define Software Characteristics priority Using ValueFn method			
Super Use Case	Define Software Characteristics priority		
Author			
Date			
Brief Description			
Preconditions	ValueFn method is selected and the characteristics to be used in comparasion are classified		
Post-conditions	Characteristics priorities are defined according the ValueFn method		
Flow of Events		Actor Input	System Response
	1	Select a charcteristic	
	2		Read Characteristic ID

	3	Asks if the user wants to maximize or minimize the characteristic
	4	Select maximize
	5	Uses the maximize formula to calculate the priority
	6	Register priorities
	7	Validate priority definition
	8	Report Operation Success
Alternative 3a: User wants to minimize		Actor Input
	1	Select minimize
	2	Uses the minimize formula to calculate the priority
	3	Returns to 5 of the Main Flow

USE CASE: Define Software Characteristics priority Using AHP method		
Super Use Case	Define Software Characteristics priority	
Author		
Date		
Brief Description		
Preconditions	AHP method is selected and the characteristics to be used in comparasion are classified	
Post-conditions	Characteristics priorities are defined according the AHP method	
Flow of Events	Actor Input	System Response
	1 Select a characteristic	
	2	Read characteristic ID
	3 Select Software	
	4	Read Software ID
	5 Give a priority number according to the scale	
	6	Read Priority
	7	Register the software id and priority in a table
	8	Increase the number of softwares with priority defined
	9	Check if the number of softwares defined is equals to the number of softwares selected
	Validate priority definition	
		Report Operation Success
Alternative 9a:	Actor Input	System Response

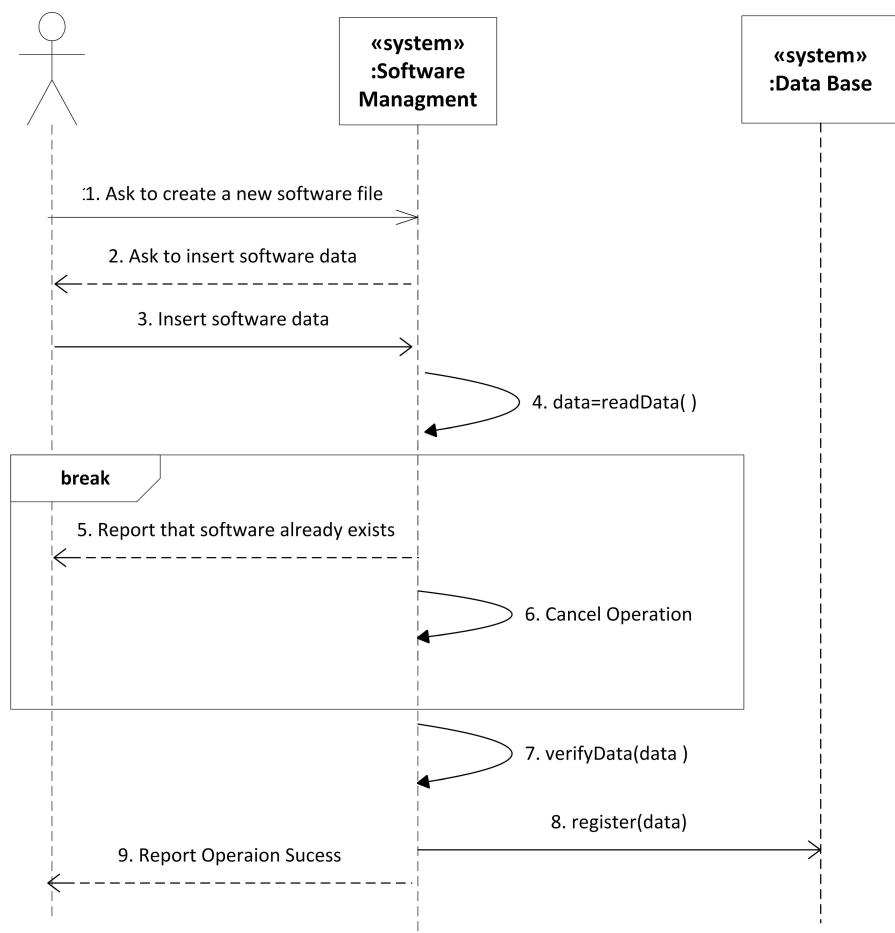
The numbers aren't equals	1		Return to 3 of the Main Flow
Alternative 11a:		Actor Input	System Response
The numbers aren't equals	1		Return to 1 of the Main Flow

USE CASE: Create Data Base			
Super Use Case	Do Not Exist		
Author			
Date			
Brief Description	Create a personal data base of user		
Preconditions			
Post-conditions	A new Data Base was define		
Flow of Events		Actor Input	System Response
	1	Ask for create new data base	
	2		Ask if the user wants to upload data from existing data base
	3		
	4	Want to upload the data	
	5		Extend: Upload Data From Existing Data Base
	6		Display table with data from existing data base
	7	Make the desired changes	
	8	Save data base	
	9		Save data base in the system
	10		Communicate successful operation
Alternative 4a: Don't want to upload data from existing data base		Actor Input	System Response
	1		Show an empty table
	2	Fill the table	
	3	Return to 8	

II. Anexo 2: Diagramas de Sequência

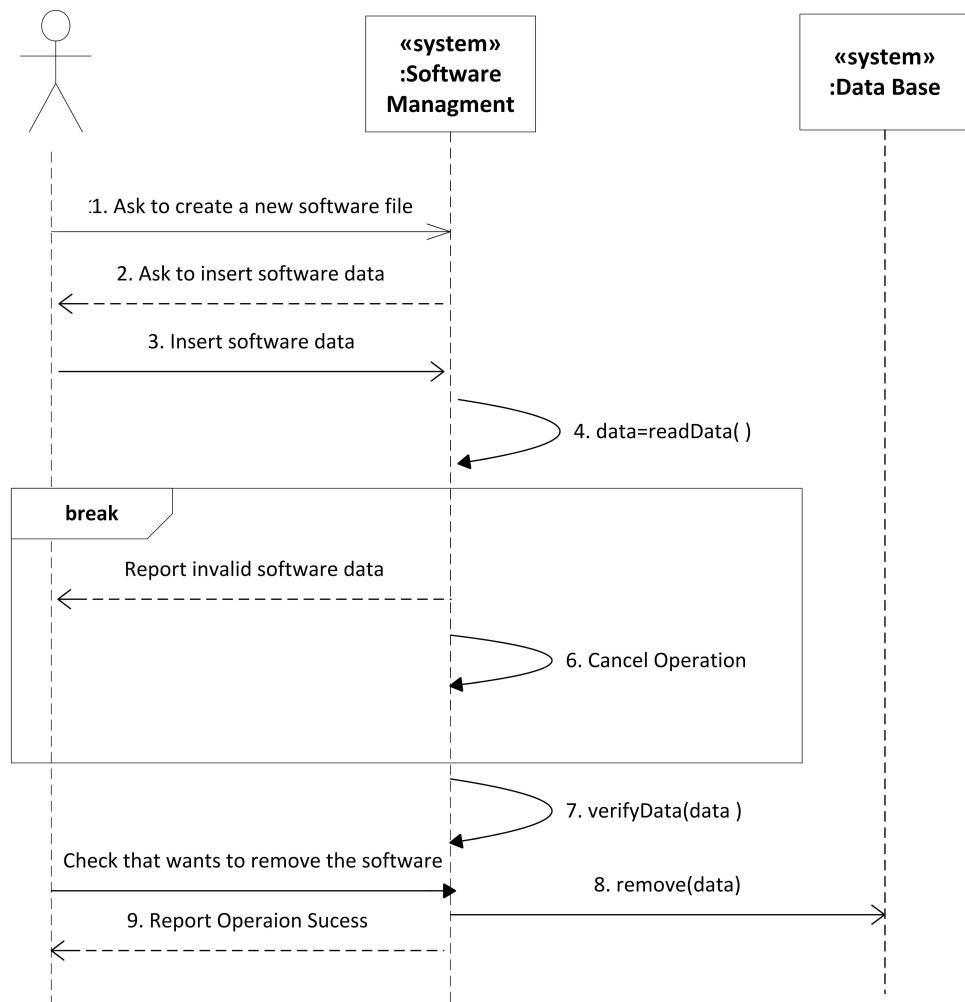
2.1. Diagramas de Sequência relativos às operações de Registo

2.1.1. Register New Software



Primeiramente, o utilizador pede para adicionar o novo software, ao qual o sistema lhe pergunta os dados do software que este pretende, tal como o nome, o website, etc. De seguida, o sistema lê os dados que foram inseridos através do método **readData()**, em que estes dados são guardados na variável **data**. Se estes já existirem na base de dados, é comunicada já a sua existência ao utilizador e a operação é cancelada. Caso contrário, o sistema procederá à verificação da validade dos dados usando o método **verifyData()**, em que a variável **data** é passada como referência. Após essa verificação, esta variável é passada para o método **register()**, que tratará de inserir o software e os dados correspondentes na base de dados. Quando concluído o registo, é comunicado o sucesso da operação ao utilizador.

2.1.2. Delete Existing Software

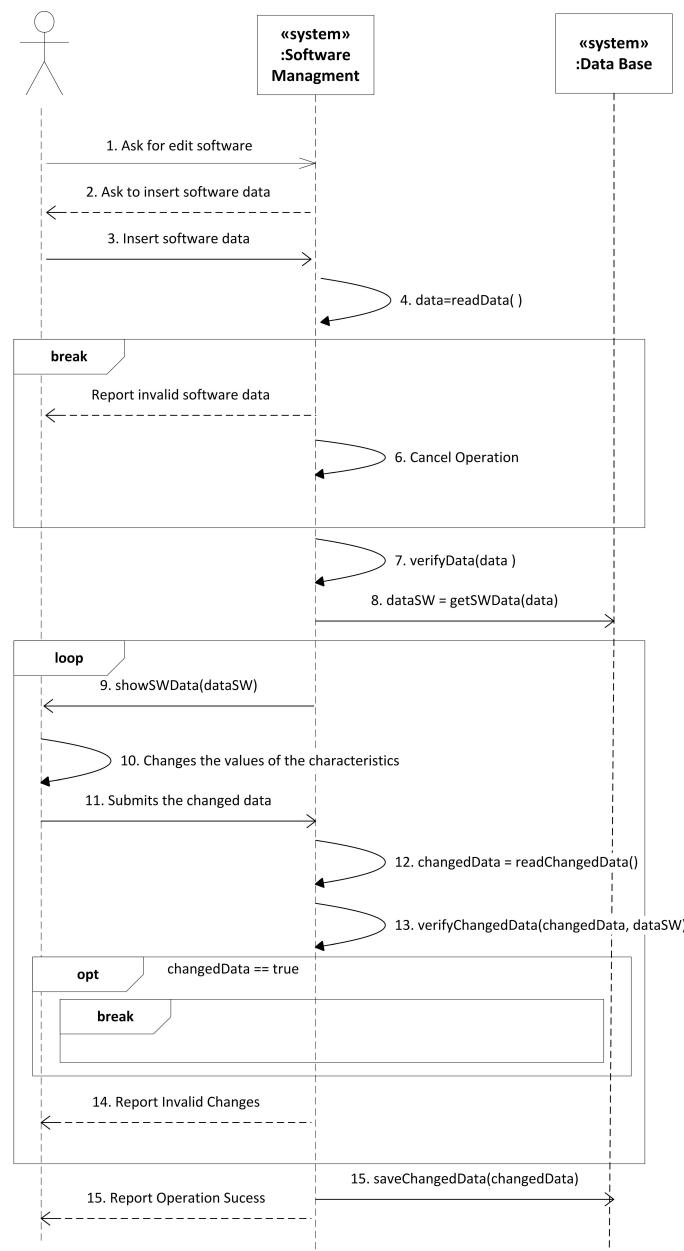


Para esta acção o utilizador indica qual o Software que pretende remover. Essa informação será lida pelo método **readData()** e é armazenada na variável **data**. Se este Software não existir na base de dados ou se não foi possível carregar a base de dados, será

activada a exceção e será comunicado esse erro ao utilizador, sendo a operação cancelada de seguida.

Caso essa exceção não se verifique, é analisada a validade dos dados e é pedido ao utilizador que confirme a sua intenção de remover este Software da base de dados. Após ele dar a sua confirmação, é chamado o método **remove**, sendo a variável **data** passada como parâmetro. Este método tratará de todo o processo de remoção do Software da base de dados. Após concluir esta operação, o utilizador é informado sobre o sucesso da operação.

2.1.3. Change Existing Software

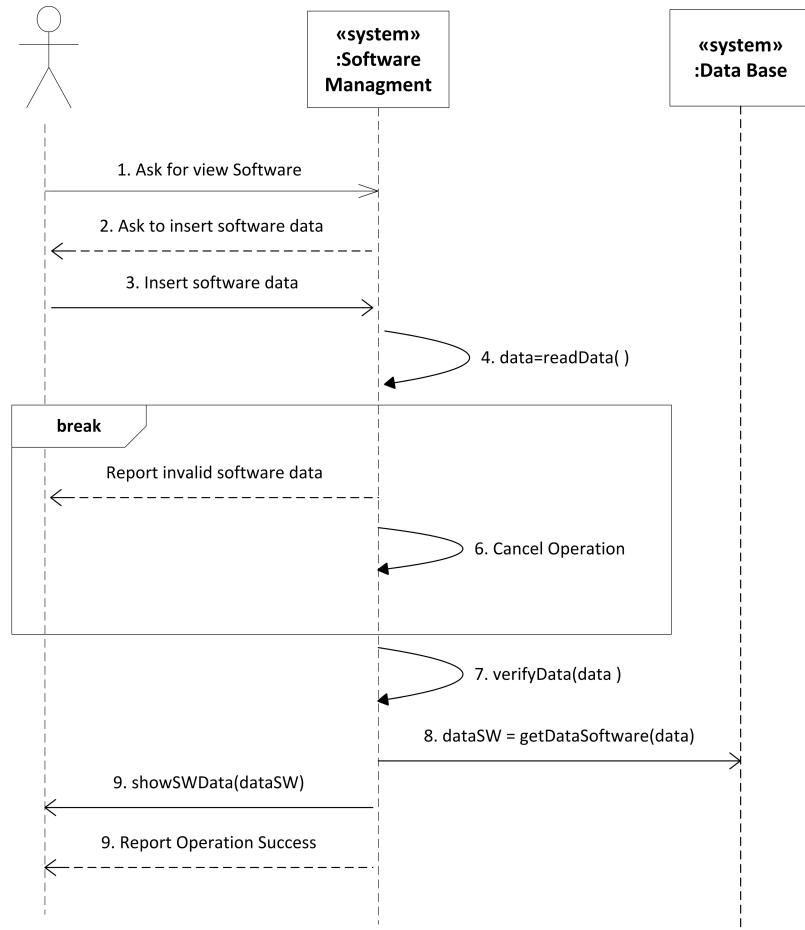


Após o utilizador pedir para editar um software, é-lhe pedido que insira os dados desse software, mais precisamente o nome, que será lido através do método **readData()** e será armazenado na variável **data**. Se o Software não existir na base de dados, ou se houver um problema no acesso à base de dados, a operação será cancelada. Caso contrário, o caso de uso continuará com a verificação dos dados através do método **verifyData(data)**. Seguidamente, na variável **dataSW** será armazenada toda a informação referente a esse Software, que será obtida através do método **getSWData**, passando **data** como parâmetro.

Em seguida, toda a informação do Software é mostrada ao utilizador, de forma organizada, através do método **showSWData**, que recebe **dataSW** como parâmetro.

Aí o utilizador alterará os dados que pretender, e depois fará a submissão desses dados. Os dados que foram alterados serão lidos pelo método **readChangedData()** e são guardados na variável **changedData**. De seguida, é feita a verificação da validade dos dados, por comparação com os dados anteriores. Para tal, é chamado o método **verifyChangedData**, sendo passado como parâmetro a variável que contém os dados alterados (**changedData**) e os dados anteriores (**dataSW**). Se, porventura, existe alguma incompatibilidade nos dados, o utilizador é informado do erro e é chamado novamente o método **showSWData**. Depois, o processo de alteração dos dados recomeçará. Quando as alterações forem válidas, a informação é gravada através do método **saveChangedData** (passando **changedData** como parâmetro) e o sucesso da operação é comunicado ao utilizador.

2.1.4. View Existing Software



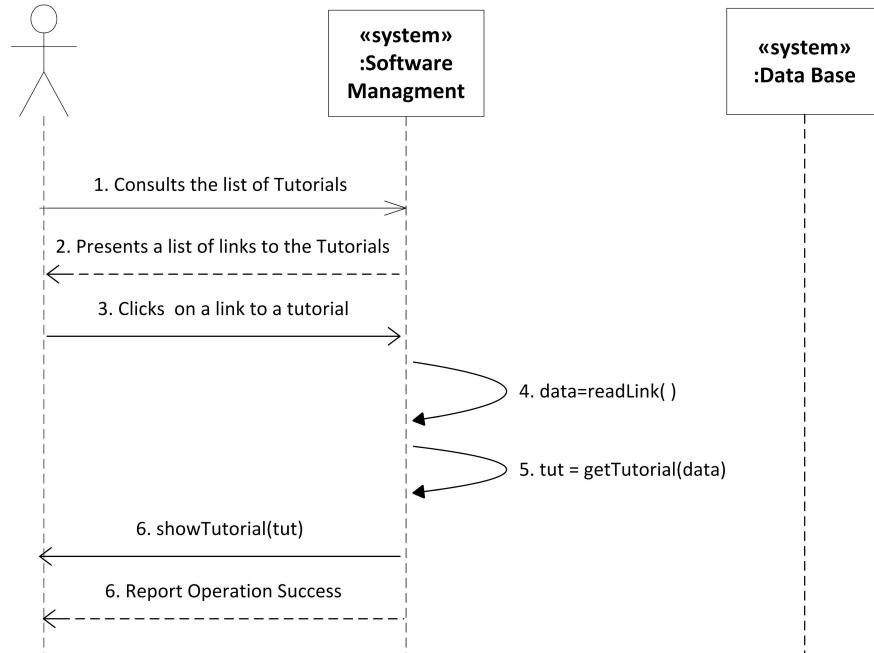
O utilizador selecciona a opção de ver um Software, pelo que lhe é pedido de seguida que insira dados do Software que este pretende consultar. Após essa inserção, o método **readData()** é chamado para ler esses dados e guarda a leitura na variável **data**, sendo testados nesse método se os dados desse software existem na base de dados e se podem ser acedidos. Caso contrário, é reportada a excepção e a operação é cancelada.

A validade dos dados inseridos será verificada com o método **verifyData()**. De seguida, o método **getDataSoftware(data)**, faz a recolha de toda a informação relativa a esse Software, e é armazenada na variável **dataSW**.

Finalmente, é chamado o método **showSWData**, que mostrará a informação desse Software, de forma organizada, para o utilizador a poder consultar. Logo de seguida, é reportado o sucesso da operação.

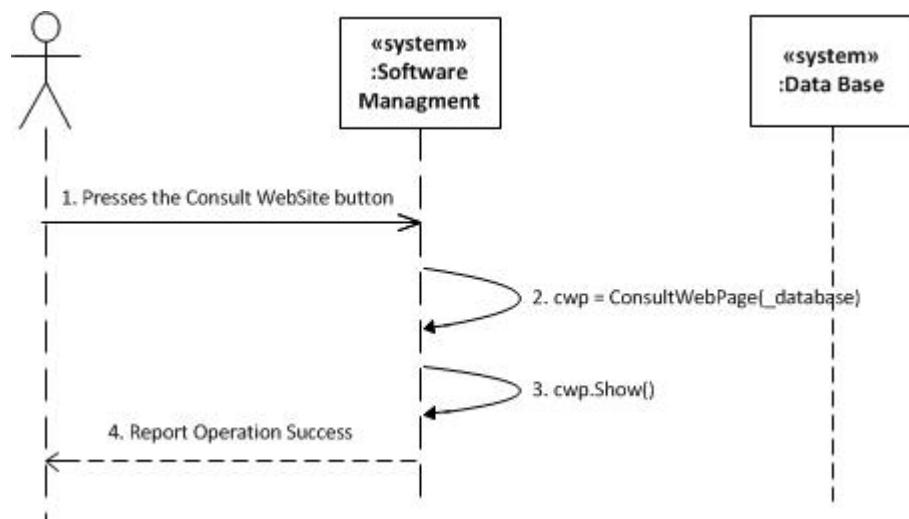
2.2. Diagramas de Sequência relativos às operações de Consulta

2.2.1. Consult Tutorial



Quando o utilizador pretender consultar um Tutorial, é-lhe apresentada uma lista de *links* para todos os Tutoriais disponíveis. Aí, este escolherá o tutorial que pretende. A escolha é lida através do método `readLink()` e é armazenada na variável `data`. O método `getTutorial()` irá obter então o tutorial correspondente, sendo este apresentado ao utilizador através do método `showTutorial()`. Logo de seguida, é comunicado que a operação foi bem sucedida.

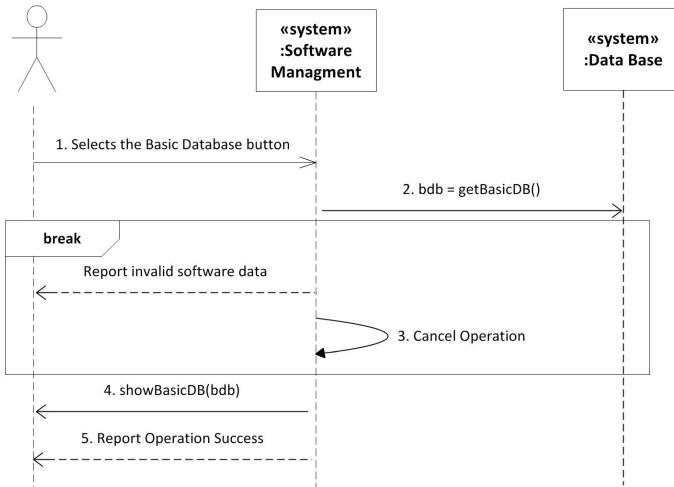
2.2.2. Consult Software's Web Site



Após o utilizador clicar no botão para consultar o *WebSite* de um software, o *link* é lido através do método **readLink**. Depois, método **openWebsite** (em que *link* é passada como parâmetro) tratará do processo de abrir, no espaço devido, o *WebSite* oficial do Software que o utilizador escolheu, sendo o sucesso da operação comunicado de seguida.

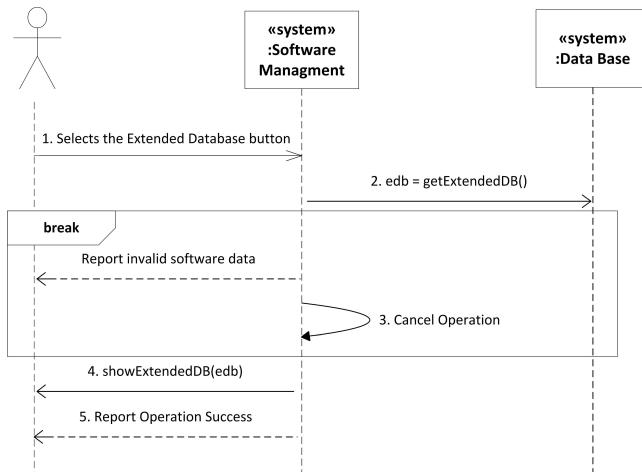
2.3. Diagramas de Sequência Relativos às operações sobre a Base de Dados

2.3.1. Select Basic Database



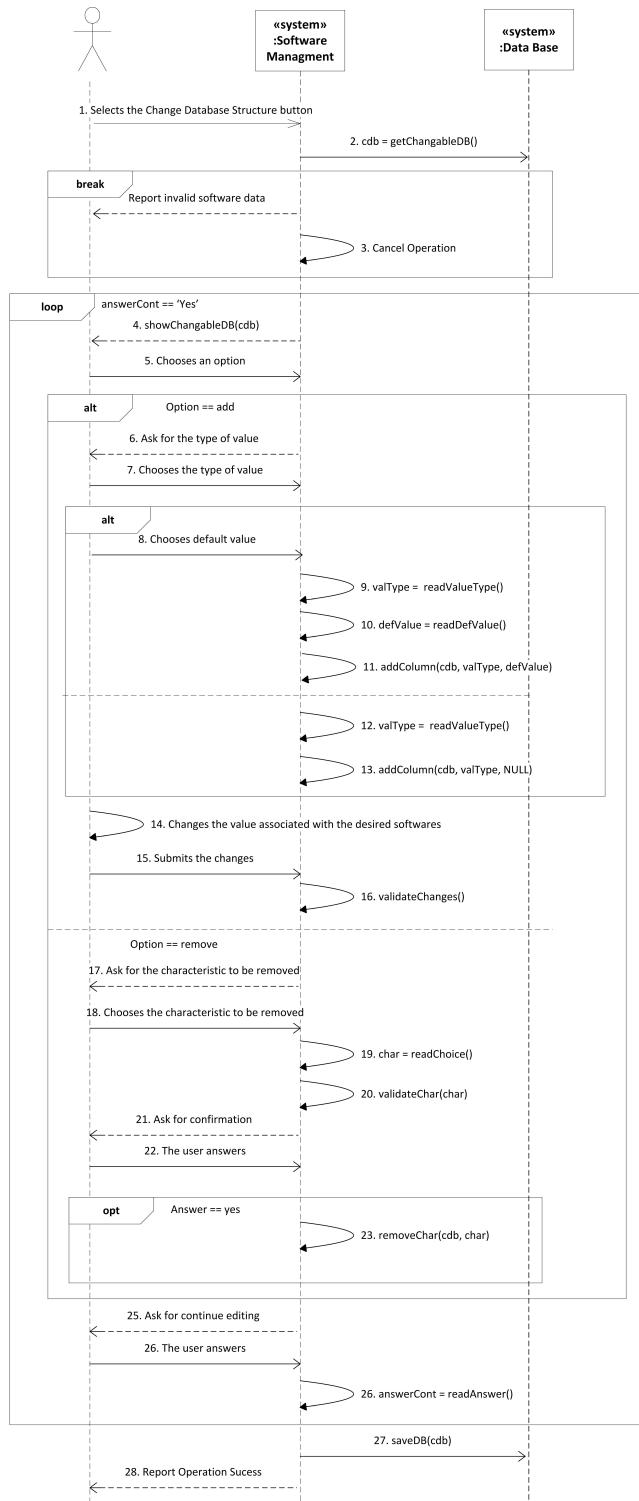
O utilizador seleccionada que pretende visualizar a base de dados na sua forma básica. Aí, é chamado o método **getBasicDB()**, que tratará de gerar essa base de dados e armazenar os resultados na variável **bdb**. Caso isso não seja possível, é reportada uma excepção e a operação é cancelada. Não ocorrendo essa excepção, a base de dados surge de forma organizada, através do método **showBasicDB()**. Por fim é reportado o sucesso da operação.

2.3.2. Select Extended Database



Este caso de uso funciona de maneira análoga ao *Select Basic Database*. No entanto, destina-se a mostrar a base de dados na sua forma estendida, isto é, com todas as características dos Softwares e os seus valores.

2.3.3. Change Database Structure



Se o utilizador pretender adicionar ou remover características dos Softwares, será este processo que ocorrerá. Inicialmente, será corrido o método **getChangableDB()**, que irá armazenar a base de dados de softwares numa variável denominada **cdb**. Caso se verifiquem anomalias com esse método, será reportada uma excepção e a operação será cancelada. Caso contrário, a variável **cdb** irá ser impressa numa interface própria para edição, através do método **showChangableDB**. Aí, o utilizador pode escolher uma das duas opções: adicionar uma nova característica ou remover uma característica já existente.

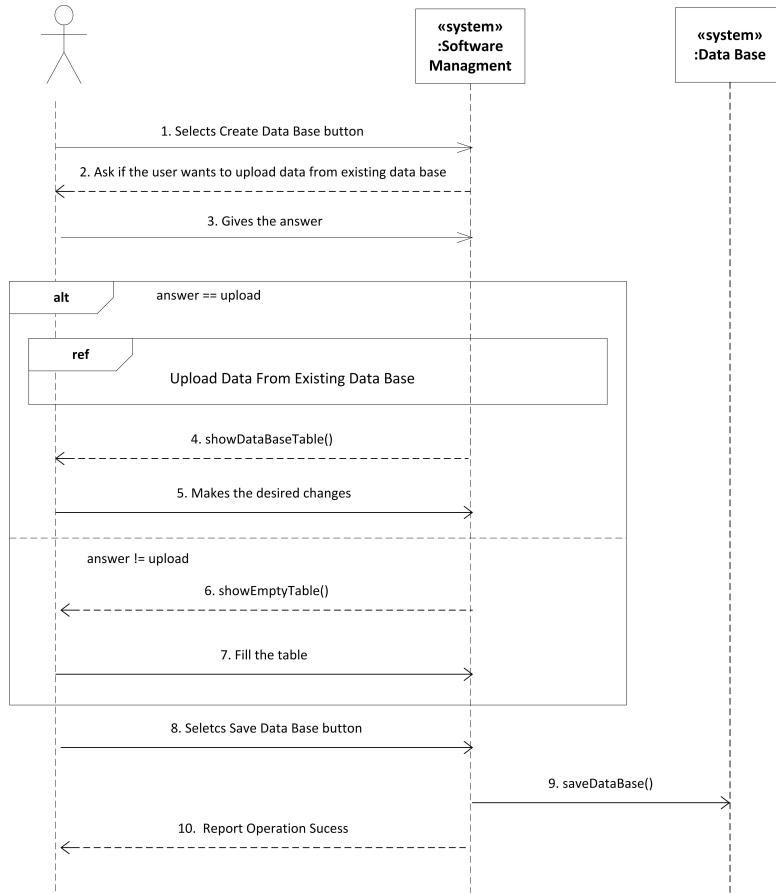
Se ele pretender adicionar uma nova característica, ser-lhe-á pedido que especifique qual é o tipo de valor para essa característica (numérico, booleano ou qualitativo), sendo confrontado com uma opção: este pode querer definir um valor que será utilizado para todos os Softwares em que não especifique o valor. Deste modo, em **valType** é armazenado o tipo da característica e em **defValue** é armazenado o valor pré-definido para esta. É então chamado o método **AddColumn**, em que são passados como argumentos a tabela de Softwares, o tipo de valor e o valor pré-definido. No caso de o utilizador não ter especificado o valor pré-definido, apenas o tipo da característica é lido e no caso do método **AddColumn**, é passado um valor nulo em vez do **defValue**.

Nesse caso, o utilizador poderá alterar o valor de todos os Softwares que pretender, especificando-os. Essa alteração é validada através do método **validateChanges()**.

No entanto, o utilizador pode optar por remover uma característica. É-lhe pedido então que indique qual é a característica que pretende eliminar. Essa característica é identificada através do método **readChoice()** e é armazenada na variável **char**, sendo depois validada através do método **validateChar()**. O utilizador é inquirido a confirmar essa remoção e, no caso de resposta positiva, o método **removeChar** (que receberá as variáveis **cdb** e **char** como parâmetros) é chamado.

Após ter aparentemente terminado processo, o utilizador terá de informar o sistema se pretende continuar a editar a tabela. Em caso afirmativo, o método **showChangableDB** volta a ser chamado e todo o processo recomeçará. Se tiver terminado a sua edição, o método **saveDB** é chamado (em que é passado **cdb** como parâmetro). Este método grava a nova tabela na base de dados, sendo depois comunicado a conclusão com sucesso da operação.

2.3.4. Create Database



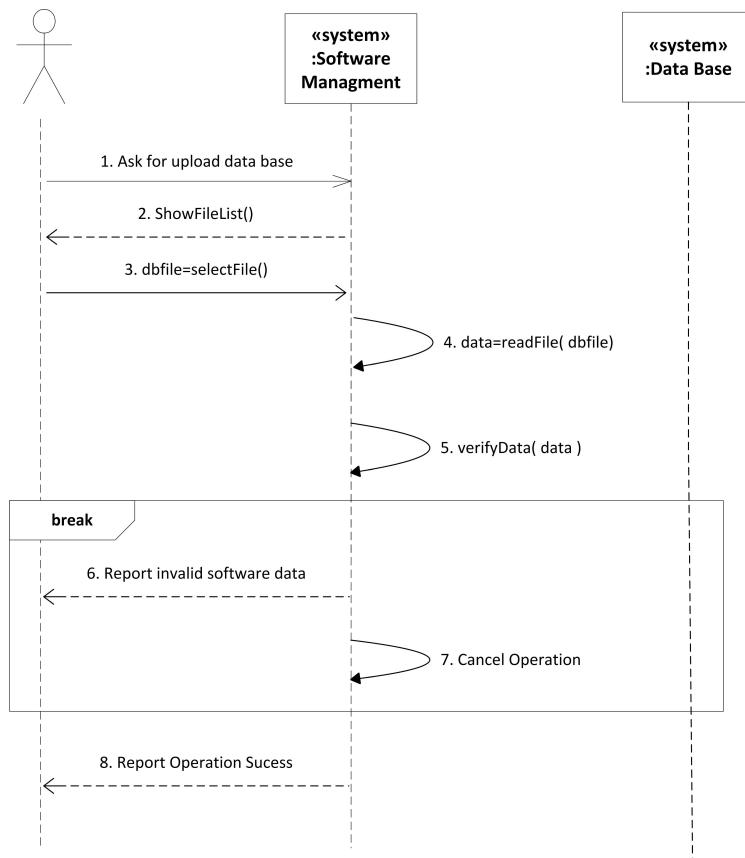
Cada utilizador pode optar por criar a sua base de dados. Para isso, deve seleccionar a opção “*Create Data Base*” no menu “*Data Base*”. É possível fazer *upload* da base de dados existente no sistema, ou criar uma base de dados de raiz.

Caso o utilizador decida fazer *upload* da base de dados existente, o seu conteúdo irá ser copiado de forma automática para a nova base de dados, através do método `showDataBaseTable()`, onde poderão ser feitas todas as alterações que o utilizador pretende.

Caso contrário, será criada uma nova base sem qualquer conteúdo, pelo método `showEmptyTable()`, em que o seu preenchimento é da responsabilidade do utilizador.

Após a criação da nova base de dados, esta deve ser gravada pelo método `saveDataBase()`, sendo comunicado o sucesso da operação.

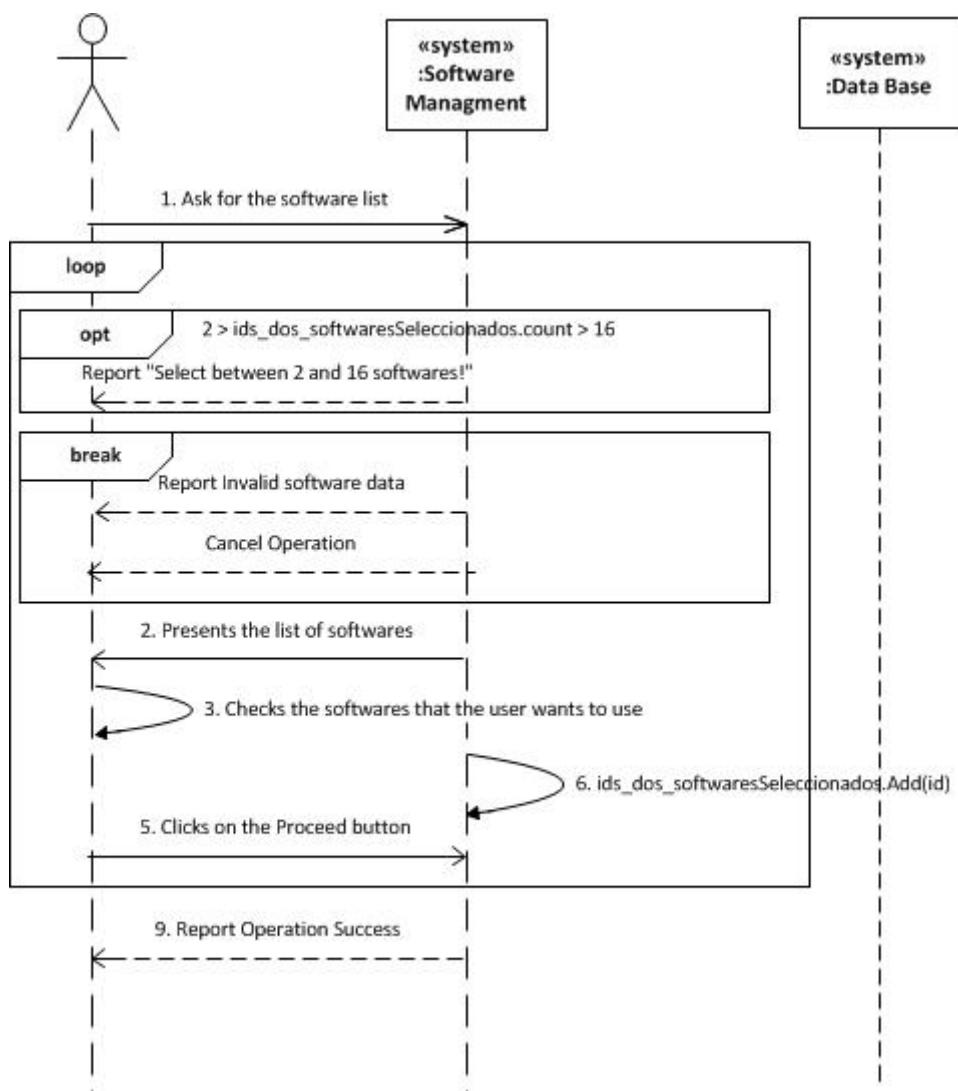
2.3.5. Upload Data from Existing Database



Na criação de uma base de dados do utilizador, este pode optar por incluir, de forma automática, os dados da base de dados existente no sistema. Para tal, deve seleccionar que o pretende fazer e escolher o ficheiro que corresponde à base de dados do sistema (que previamente foi descarregado a partir do site), que fica armazenado na variável **dbfile**. Esse ficheiro é lido, através do método **readData(dbfile)** e é verificada a validade dos dados. Caso os dados sejam válidos, é reportado o sucesso da operação. Caso contrário, o utilizador é informado que não foi possível fazer *upload* do ficheiro escolhido.

2.4. Diagramas de Sequência relativos às operações de Comparação

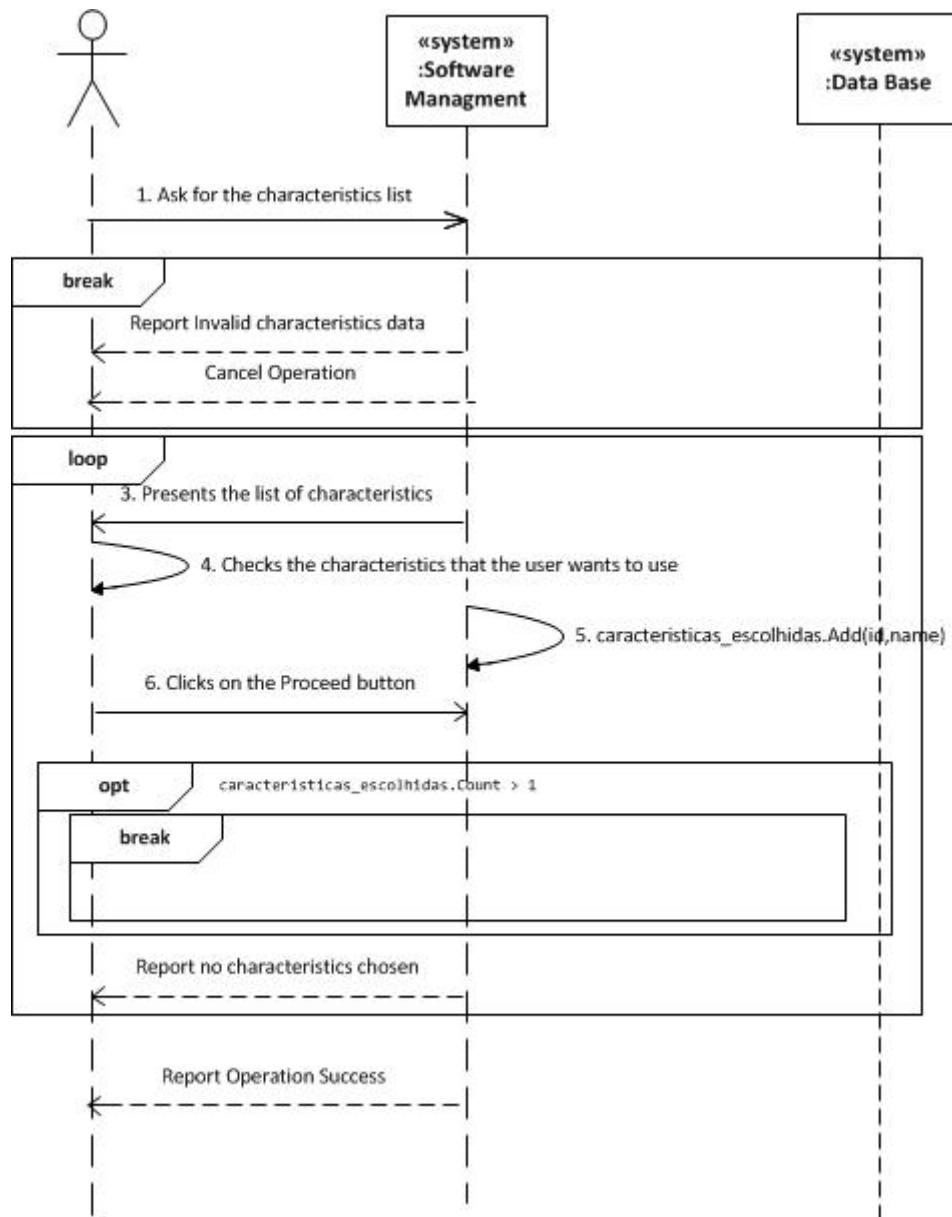
2.4.1. Select a set of Softwares to be used in comparison



Após o utilizador efectuar o *login* no sistema, é-lhe apresentado a lista dos *softwares* que pode escolher. Aí, o utilizador irá seleccionar quais os que pretende e clicar no botão *Next*. As escolhas são lidas e serão guardados os ID dos *softwares* seleccionados. No caso de não terem sido feitas escolhas, ou se as escolhas excederem os 16 *softwares*, será reportado ao

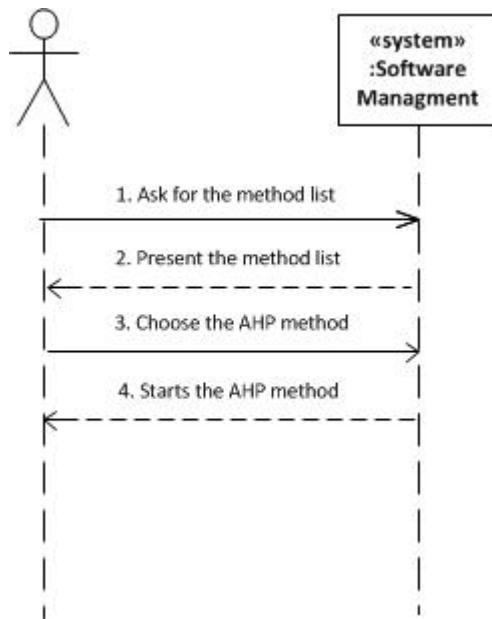
utilizador que deverá escolher entre 2 e 16 softwares. No final, é anunciado o sucesso da operação.

2.4.2. Select characteristics to be used in comparison



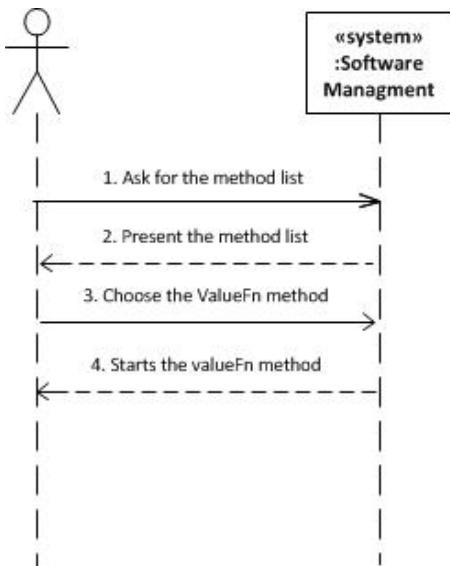
Nesta acção o utilizador deve escolher as características que pretende usar na comparação. A lista de características é apresentada, e o utilizador deverá escolher quais as que pretende utilizar na comparação. No final deverá dar continuidade ao processo, clicando no botão *Next*. As escolhas são lidas posteriormente e guardado o ID e o nome de cada uma. No caso de não terem sido feitas escolhas, tal situação será reportada ao utilizador e o processo de selecção voltará ao início. Caso contrário é comunicado o sucesso da operação.

2.4.3. Select AHP method



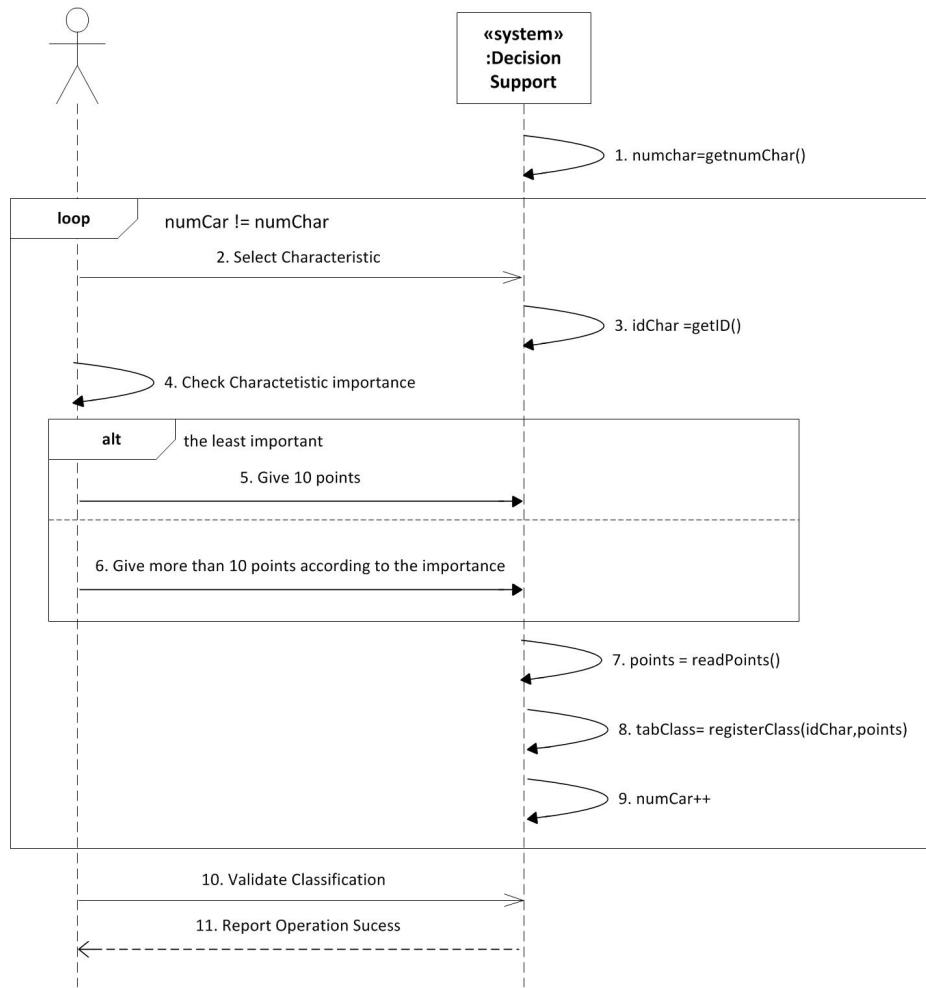
O utilizador, inicialmente, pede a lista de métodos existentes, sendo esta apresentada de seguida. Aí, este escolherá o método AHP, sendo essa escolha lida. Logo de seguida, a escolha é validada e a conclusão da operação é anunciada ao utilizador.

2.4.4. Select valueFN method



O utilizador começa por pedir a lista de métodos disponíveis, sendo esta apresentada de seguida. É seleccionado o método **valueFn**, sendo essa escolha lida. Por fim, a escolha é validada e a é anunciado a conclusão da operação.

2.4.5. Classify Software Characteristic using SMART method



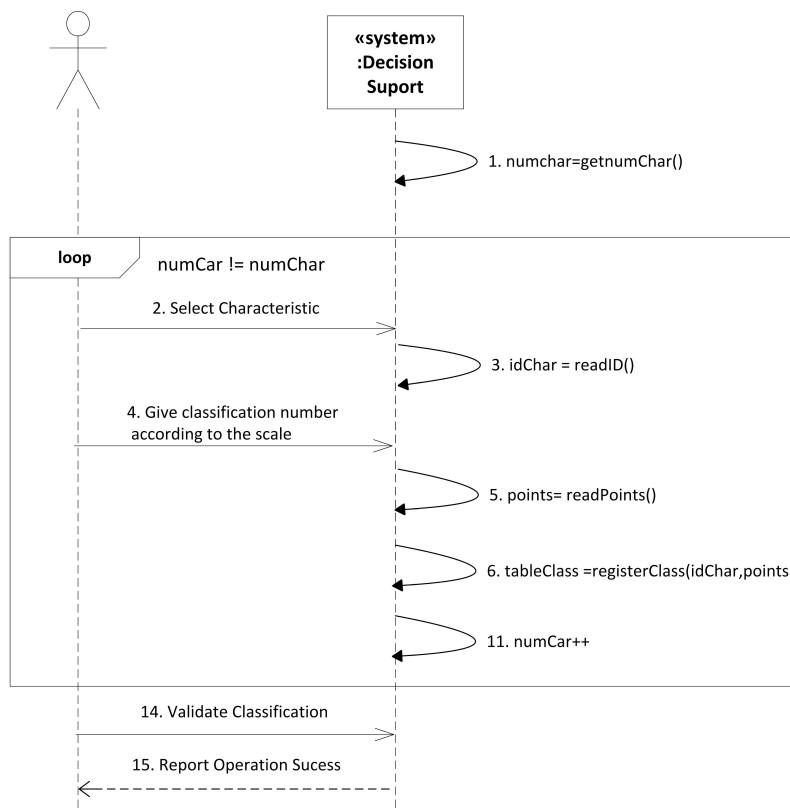
O utilizador, quando se encontra na fase de classificação de características, pode escolher um de dois métodos. Neste caso, o utilizador escolheu o método **SMART**. Nesta primeira fase o utilizador apenas tem dar uma classificação às características que seleccionou anteriormente como referenciado no caso de uso designado “Select characteristics to be used in comparison”. Como tal, é seleccionar da lista dos critérios definidos na fase anterior. De seguida, é seleccionada uma característica e o sistema lê o seu ID e regista-o. Esta leitura é feita com o método **getID()**. No passo seguinte, o utilizador tem de reflectir qual a importância que da característica no resultado que pretende obter. De acordo com a sua opinião, o utilizador atribui 10 pontos se a característica for a menos importante de todas. Às restantes características, o utilizador atribui mais de 10 pontos de acordo com a relevância de cada uma.

As pontuações são lidas com o método **readPoints()**. Quando a característica já está classificada, o sistema regista numa tabela que dada característica tem a classificação

atribuída. O método utilizado para o registo é **registerClass(idChar,points)**, que recebe como parâmetros o ID da característica e respectiva classificação. O número de classificações feitas é incrementado. Este processo é repetido enquanto o número de classificações não for igual ao número de características.

Por fim, quando todas as classificações forem atribuídas, o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

2.4.6. Classify Software Characteristic using AHP method



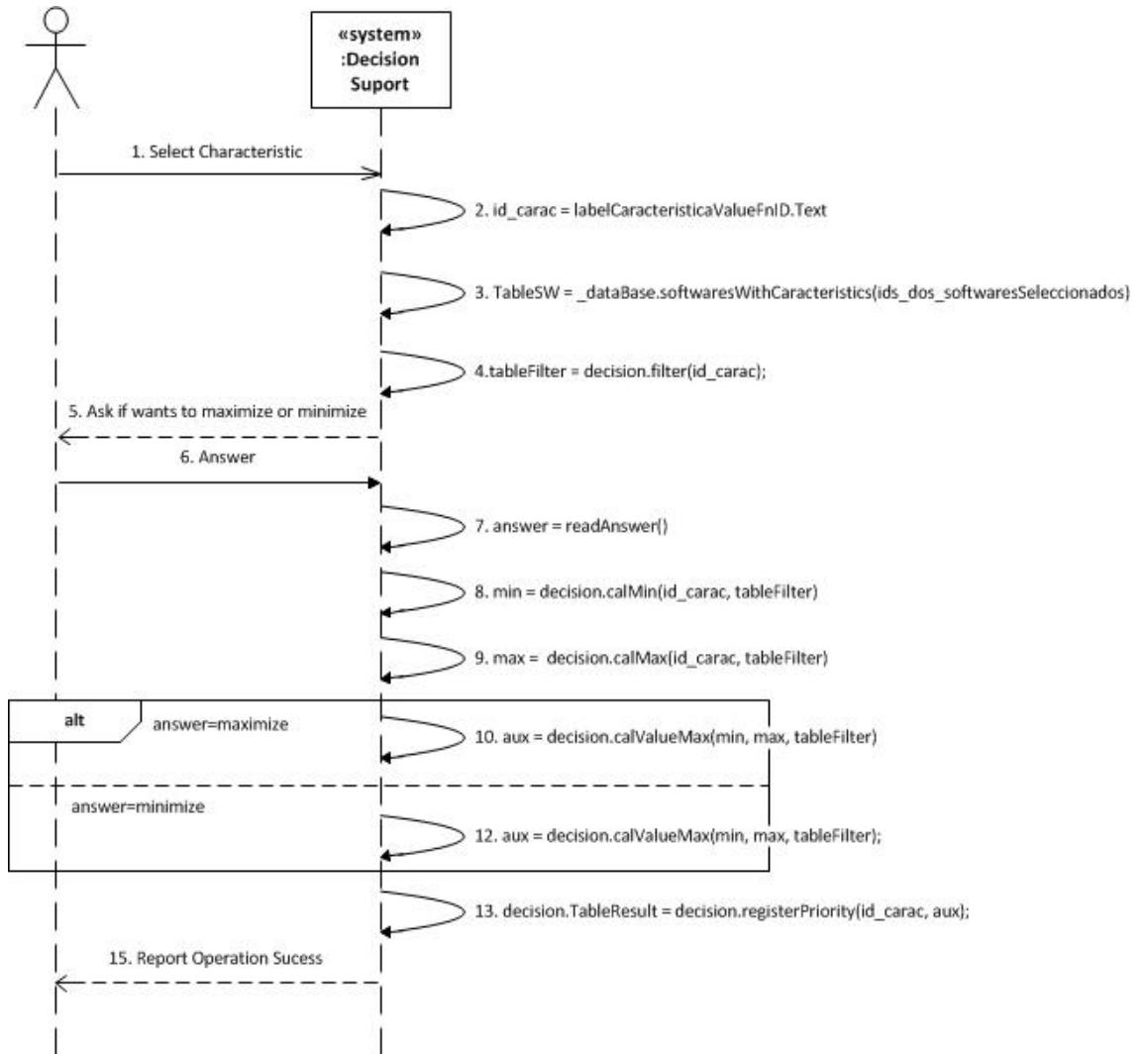
Neste caso reporta o processo que sucede após do utilizador escolher o método **AHP**. Numa primeira fase, o utilizador apenas atribui uma classificação às características que seleccionou anteriormente, como referenciado no caso de uso “Select characteristics to be used in comparison”. Como tal, é seleccionada uma característica da lista de critérios que foi definida anteriormente. De seguida, o sistema lê o ID da característica, para posteriormente proceder ao seu registo.

Posteriormente, o utilizador terá de acordo com a escala do método, classificar a característica atribuindo um número fixo de pontos. Estes pontos serão lidos posteriormente com o método **readPoints()**. O passo seguinte é o registo da classificação da característica. Este é feito através do método **registerClass(idChar,points)**, que recebe como

parâmetros o ID da característica e os pontos de classificação. Quando isto se sucede, o número de características é incrementado. Este processo é repetido enquanto o número de classificações não for igual ao número de características escolhidas.

Por fim, quando todas as classificações forem atribuídas, o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

2.4.7. Define Software priority using ValueFn method



A definição de prioridades é a etapa seguinte da comparação de softwares. Como tal, depois de definidas as classificações, o utilizador deve escolher entre dois modos de cálculo de prioridades. Nesta situação, o utilizador escolheu o método. Como o utilizador pode escolher um método para cada característica, esta situação apenas retrata uma característica.

Deste modo, o utilizador selecciona uma característica. O sistema procede à leitura do seu ID, e de seguida, utiliza uma tabela onde estão registados os softwares seleccionados e as

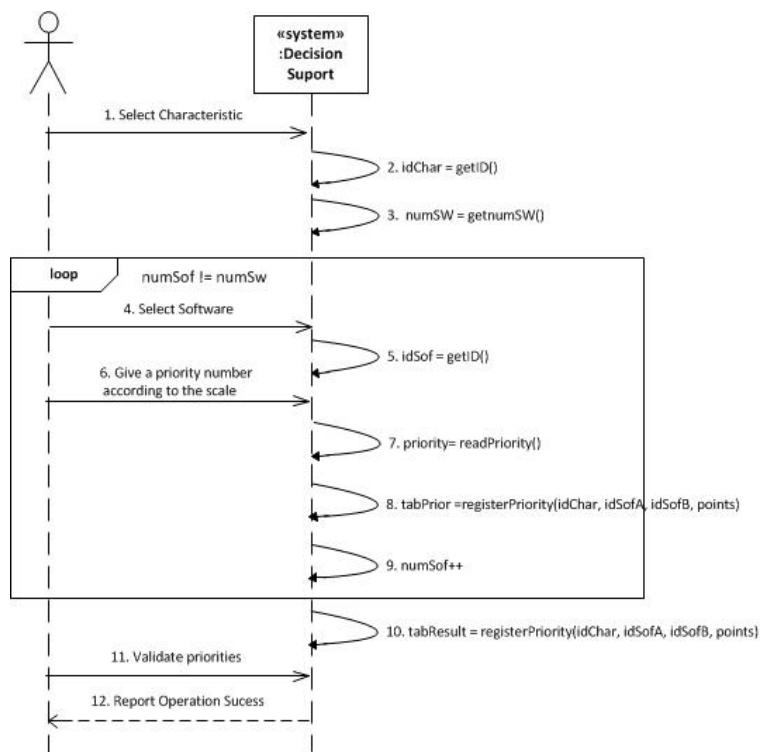
respectivas características para comparação. Esta tabela deriva do caso de uso “Select a set of softwares to be used in comparison”. O conteúdo desta tabela é filtrado para que contenha apenas o *software* e a classificação da característica escolhida ao início. Os métodos utilizados para efectuar esta operação são **getID()**, **getListSW()** e **filter(idChar)**.

De seguida, o sistema questiona o utilizador se este pretende maximizar ou minimizar o critério em questão. O sistema determina o mínimo e o máximo da lista filtrada. Estes são calculados respectivamente através do métodos **calMin()** e **calMax()**. Estes dois métodos recebem como parâmetro a tabela filtrada.

Caso o utilizador pretenda maximizar o critério, é utilizado o método **calValueMax(min,max)**, mas caso pretenda minimizar é utilizado o método **calValueMin(min,Max)**. Estes últimos dois métodos gravam o resultado numa tabela e recebem como parâmetro o mínimo e o máximo dos valores da tabela filtrada, ou seja, o mínimo e o máximo das classificações dadas de todos os *softwares*.

Por fim, as prioridades calculadas para cada um dos softwares, da característica seleccionada, são gravadas numa nova tabela final através do método **register(idChar)**, o qual recebe como parâmetro o ID da característica e as suas tabelas de prioridades. Finalmente, quando todas as classificações forem atribuídas, o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

2.4.8. Define Software priority using AHP method



Esta situação reporta a escolha do método AHP para definir as prioridades. Como o utilizador pode escolher um método para cada característica, esta situação apenas retrata para uma característica.

A primeira tarefa do utilizador é seleccionar uma das características. O sistema lê o ID da característica e depois vai procura qual o número total de softwares seleccionados, que deriva do caso de uso designado “Select a set of Softwares to be used in comparison”. Os métodos utilizados para efectuar estas operação são respectivamente **getID()**, **getNumSW()**.

Contrariamente ao outro método de definição de prioridades, o utilizador tem que definir, para cada software, a prioridade que esta característica tem para o mesmo. Como tal, o passo seguinte do utilizador será a selecção do *software* pelo qual pretende começar a definir prioridades. O sistema lê este ID com o método **getID()**.

De seguida o utilizador dá a prioridade que pretende de acordo com a escala de classificações. O sistema lê a prioridade que o utilizador atribuiu à característica através do método **readPriority()**. O sistema regista também as prioridades dadas ao software com o método **registerPriority(idSof, priority)**. Este recebe o ID do *software* e a prioridade dada para o mesmo como parâmetros. O número de *softwares* classificados é incrementado.

Este processo é repetido enquanto o número de classificações não for igual ao número de *softwares*. Antes de terminar, o sistema associa para a característica uma tabela com as prioridades de cada *software*. Isto é conseguido através do método **registerPriority(idChar)**, que recebe como parâmetro o ID da característica e a tabela de prioridades.

Finalmente, quando todas as classificações forem atribuídas o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

III. Anexo 3: Tabelas da Base de Dados

3.1. Tabela de Utilizadores (User)

username	password	email	regist_date
anasampaio	97	anaisamp@gmail.com	30-05-2011
extra_database	extra_database	miguelpintodacosta@gmail.com	30-05-2011
hugofrade	102	emanspace@gmail.com	30-05-2011
miguelcosta	83	miguelpintodacosta@gmail.com	30-05-2011
simple_database	simple_database	miguelpintodacosta@gmail.com	30-05-2011
tiagoabreu	fcp	tiagoalvesabreu@gmail.com	30-05-2011

3.2. Tabela de Softwares (softwares)

id_software	name	link	username
1	Decision Explorer	http://www.banxia.com/dexplore/	simple_database
2	Criterium Decision Plus	http://www.infoharvest.com/ihroot/index.asp	simple_database
3	Decision Lab	http://decisionlab.org.uk/	simple_database
4	Electre III-IV	http://www.sciencedirect.com/science/article/pii/S0377221796002524	simple_database
5	Electre IS	http://electre-ii.software.informer.com/	simple_database
6	Equity	http://www.catalyze.co.uk/?id=229	simple_database
7	Expert Choice	http://www.expertchoice.com/	simple_database
8	Hiview	http://www.catalyze.co.uk/?id=230	simple_database
9	Logical Decisions	http://www.logicaldecisions.com/	simple_database
10	MACBETH	http://www.m-macbeth.com/en/m-home.html	simple_database
11	OnBalance	http://www.quartzstar.com/	simple_database
12	PrefCalc	http://www.prevcalc.net/	simple_database
13	Prime Decions	http://www.sal.tkk.fi/en/resources/downloadables/prime	simple_database
14	SANNA	http://sanna.com.br/	simple_database
15	TOPSIS	http://www.topsis.com.br/	simple_database
16	Uta Plus	http://www.lamsade.dauphine.fr/english/software.html#uta+	simple_database
17	Vip Analysis	http://www4.fe.uc.pt/lmcdias/english/vipa.htm	simple_database
18	V.I.S.A	http://www.simul8.com/products/visa.htm	simple_database
19	Winpre	http://www.sal.hut.fi/Downloadables/winpre.html	simple_database
20	Ergo	http://www.ergogroup.ie/	simple_database

21	Solvex	http://www.ccas.ru/pma/product.htm	simple_database
22	Fgm	http://www.ccas.ru/mmes/mmeda/fgm.htm	simple_database
23	WW-W-Nimbus	http://nimbus.mit.jyu.fi/	simple_database
24	Electre Tri	http://www.lamsade.dauphine.fr/english/software.html#TRI	simple_database
25	Iris	http://www4.fe.uc.pt/lmcdias/iris.htm	simple_database
26	Automan	http://tgtoil.com/en/automan_software/	simple_database
27	Decision Deck	http://www.decision-deck.org/	simple_database
28	Lpa Visirule	http://www.lpa.co.uk/	simple_database
29	Moira	http://user.tninet.se/~fde729o/MOIRA/Software.htm	simple_database
30	Sanex	http://www.ies.ch/EcoEng001/EcoEng001_R3.html	simple_database
31	Water Quality Planning Dss	http://www.ccas.ru/mmes/mmeda/papers/vodhoz.htm	simple_database
32	Hipriority	http://www.quartzstar.com/	simple_database
33	Mediator	http://www.matchware.com/en/products/mediator/	simple_database
34	Decision Explorer	http://www.banxia.com/dexplore/	extra_database
35	Criterium Decision Plus	http://www.infoharvest.com/ihroot/index.asp	extra_database
36	Decision Lab	http://decisionlab.org.uk/	extra_database
37	Electre III-IV	http://www.sciencedirect.com/science/article/pii/S0377221796002524	extra_database
38	Electre IS	http://electre-ii.software.informer.com/	extra_database
39	Equity	http://www.catalyze.co.uk/?id=229	extra_database
40	Expert Choice	http://www.expertchoice.com/	extra_database
41	Hiview	http://www.catalyze.co.uk/?id=230	extra_database
42	Logical Decisions	http://www.logicaldecisions.com/	extra_database
43	MACBETH	http://www.m-macbeth.com/en/m-home.html	extra_database
44	OnBalance	http://www.quartzstar.com/	extra_database
45	PrefCalc	http://www.prevcalc.net/	extra_database
46	Prime Decions	http://www.sal.tkk.fi/en/resources/downloadables/prime	extra_database
47	SANNA	http://sanna.com.br/	extra_database
48	TOPSIS	http://www.topsis.com.br/	extra_database
49	Uta Plus	http://www.lamsade.dauphine.fr/english/software.html#uta+	extra_database
50	Vip Analysis	http://www4.fe.uc.pt/lmcdias/english/vipa.htm	extra_database
51	V.I.S.A	http://www.simul8.com/products/visa.htm	extra_database
52	Winpre	http://www.sal.hut.fi/Downloadables/winpre.html	extra_database
53	Ergo	http://www.ergogroup.ie/	extra_database
54	Fgm	http://www.ccas.ru/mmes/mmeda/fgm.htm	extra_database
55	Solvex	http://www.ccas.ru/pma/product.htm	extra_database
56	WW-W-Nimbus	http://nimbus.mit.jyu.fi/	simple_database
57	Electre Tri	http://www.lamsade.dauphine.fr/english/software.html#TRI	extra_database
58	Iris	http://www4.fe.uc.pt/lmcdias/iris.htm	extra_database
59	Automan	http://tgtoil.com/en/automan_software/	extra_database
60	Decision Deck	http://www.decision-deck.org/	extra_database
61	Lpa Visirule	http://www.lpa.co.uk/	extra_database
62	Moira	http://user.tninet.se/~fde729o/MOIRA/Software.htm	extra_database
63	Sanex	http://www.ies.ch/EcoEng001/EcoEng001_R3.html	extra_database
64	Water Quality Planning Dss	http://www.ccas.ru/mmes/mmeda/papers/vodhoz.htm	extra_database

65	Hipriority	http://www.quartzstar.com/	extra_database
66	Mediator	http://www.matchware.com/en/products/mediator/	extra_database

3.3. Tabela de Características (characteristics)

characteristics_id	characteristics_type	characteristics_name	value	value_order
1	bool	Compatible operating systems	NULL	NULL
2	numeric	Cost	NULL	NULL
3	qualitative	Interaction with the user	very bad	1
3	qualitative	Interaction with the user	bad	2
3	qualitative	Interaction with the user	acceptable	3
3	qualitative	Interaction with the user	good	4
3	qualitative	Interaction with the user	very good	5
3	qualitative	Interaction with the user	excellent	6
4	bool	Manuals and Tutorials	NULL	NULL
5	bool	Examples through applications	NULL	NULL
6	bool	on-line Help	NULL	NULL
7	bool	Free version	NULL	NULL

3.4. Tabela com a relação entre Softwares e Características (software_list)

id_software	characteristics_id	characteristics_value
34	1	false
34	2	456
34	3	good
34	4	true
34	5	true
34	6	true
34	7	true
35	1	false
35	2	636
35	3	very good
35	4	true
35	5	true
35	6	true
35	7	true
36	1	false
36	2	990
36	3	good
36	4	true

id_software	characteristics_id	characteristics_value
51	1	true
51	2	350
51	3	good
51	4	true
51	5	true
51	6	false
51	7	true
52	1	false
52	2	0
52	3	good
52	4	false
52	5	true
52	6	false
52	7	true
53	1	false
53	2	2134
53	3	very good
53	4	true

36	5	true
36	6	true
36	7	true
37	1	false
37	2	519
37	3	acceptable
37	4	true
37	5	true
37	6	false
37	7	true
38	1	true
38	2	519
38	3	good
38	4	true
38	5	true
38	6	false
38	7	false
39	1	true
39	2	2143
39	3	good
39	4	true
39	5	true
39	6	true
39	7	true
40	1	false
40	2	1955
40	3	good
40	4	true
40	5	true
40	6	true
40	7	true
41	1	true
41	2	1100
41	3	good
41	4	true
41	5	true
41	6	true
41	7	true
42	1	false
42	2	566
42	3	good
42	4	true
42	5	true
42	6	true

53	5	true
53	6	true
53	7	true
54	1	false
54	2	0
54	3	good
54	4	true
54	5	true
54	6	true
54	7	true
55	1	true
55	2	0
55	3	acceptable
55	4	true
55	5	false
55	6	false
55	7	true
56	1	true
56	2	519
56	3	very good
56	4	true
56	5	true
56	6	false
56	7	true
57	1	false
57	2	519
57	3	acceptable
57	4	true
57	5	true
57	6	false
57	7	true
58	1	false
58	2	1000
58	3	good
58	4	true
58	5	true
58	6	true
58	7	true
59	1	false
59	2	56
59	3	acceptable
59	4	true
59	5	true
59	6	true

42	7	true
43	1	true
43	2	1750
43	3	good
43	4	true
43	5	true
43	6	true
43	7	true
44	1	false
44	2	640
44	3	good
44	4	true
44	5	true
44	6	true
44	7	true
45	1	false
45	2	0
45	3	acceptable
45	4	false
45	5	false
45	6	false
45	7	true
46	1	false
46	2	0
46	3	good
46	4	true
46	5	false
46	6	false
46	7	true
47	1	false
47	2	0
47	3	good
47	4	true
47	5	true
47	6	false
47	7	true
48	1	false
48	2	696
48	3	good
48	4	true
48	5	true
48	6	true
48	7	true
49	1	false

59	7	true
60	1	true
60	2	0
60	3	good
60	4	true
60	5	false
60	6	false
60	7	true
61	1	true
61	2	0
61	3	good
61	4	true
61	5	true
61	6	true
61	7	true
62	1	false
62	2	0
62	3	acceptable
62	4	false
62	5	false
62	6	false
62	7	true
63	1	false
63	2	0
63	3	very good
63	4	true
63	5	true
63	6	false
63	7	true
64	1	false
64	2	0
64	3	acceptable
64	4	false
64	5	false
64	6	false
64	7	true
65	1	false
65	2	1969
65	3	very good
65	4	true
65	5	true
65	6	true
65	7	true
66	1	false

49	2	519
49	3	good
49	4	true
49	5	false
49	6	false
49	7	true
50	1	false
50	2	0
50	3	good
50	4	true
50	5	true
50	6	false
50	7	true

66	2	694
66	3	good
66	4	true
66	5	true
66	6	true
66	7	true