Universidade do Minho

Grupo 13
Ana Isabel Anjos Sampaio |**54740**
Miguel Pinto da Costa|**54746**
Hugo Emanuel da Costa Frade | **54750**
Tiago Alves Abreu | **54772**

[RELATÓRIO DO PROJECTO DE LABORATÓRIOS DE INFORMÁTICA IV]

Projecto 6: Desenvolvimento de uma aplicação para apoiar a selecção de Software de Apoio à Decisão.

Conteúdo

Capítulo 1 | INTRODUÇÃO 1

Capítulo 2 | ESTUDO DO PROBLEMA 2

2.1. Conceitos fundamentais para a compreensão do problema 2

Conceito 1 | *Software* de Apoio à Decisão 2

Conceito 1.1 | Utilizadores deste *Software* 2

Conceito 1.2 | Importância deste *Software* 3

Conceito 2 | Para que serve um *Software* de apoio à selecção de *Software* de Apoio à Decisão 3

Conceito 2.1 | Utilizadores deste tipo de *Software* 3

Conceito 2.2 | Importância deste tipo de *Software* 3

2.2. Contextualização do problema 4

2.2.1 Em que se baseia o problema em questão? 4

2.2.2 Porquê a realização deste problema? 4

Capítulo 3 | OBJECTIVOS DO PROJECTO 5

Capítulo 4 | ANÁLISE DE REQUISITOS 6

4.1. Requisitos da Interface 6

4.2. Requisitos da Base de Dados 6

4.3. Requisitos a nível de métodos de selecção 7

Capítulo 5 | PLANEAMENTO DE ACTIVIDADES 8

5.1 Diagrama Previsto do planeamento de actividades 8

Capítulo 6 | ESPECIFICAÇÃO UML 9

Capítulo 7 | DIAGRAMAS DE CASOS DE USO 10

7.1. Diagrama de Casos de Uso Geral 10

7.2.2. Software Managment Subsytem 12

7.2.3. Decision Suport Subsystem 12

Capítulo 8 | DIAGRAMAS DE SEQUÊNCIA 26

8.1. Diagramas de Sequência relativos às operações de Registo 26

8.1.1. Register New Software 26

8.1.2. Delete Existing Software 27

8.1.3. Change Existing Software 28

8.1.4. View Existing Software 29

8.2. Diagramas de Sequência relativos às operações de Consulta 30

8.2.1. Consult Help 30

8.2.2. Consult Tutorial 31
8.2.3. Consult Software's Web Site 32
8.3. Diagramas de Sequência relativos às operações sobre a Base de Dados 32
8.3.1. Select Basic Database 32
8.3.2. Select Extended Database 33
8.3.3. Change Database Structure 34
8.3.4. Create Database 36
8.3.5. Upload Data from Existing Database 37
8.4. Diagramas de Sequência relativos às operações de Comparação 38
8.4.1. Select a set of Softwares to be used in comparison 38
8.4.2. Select characteristics to be used in comparison 39
8.4.3. Select AHP method 40
8.4.5. Select valueFN method 41
8.4.6. Select SMART method 41
8.4.7. Classify Software Characteristic using SMART method 42
8.4.8. Classify Software Characteristic using AHP method 43
8.4.9. Define Software priority using ValueFn method 44
8.4.10. Define Software priority using AHP method 45
Capítulo 9 | DIAGRAMAS DE CLASSES 47
Capítulo 10 | ESQUEMA RELACIONAL DA BASE DE DADOS 48
Capítulo 11 | CONCLUSÃO 49

Capítulo 1 | INTRODUÇÃO

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 a orientadora que sugeriu o problema.

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 (*Unified Modeling Language*). 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*.

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, quiçá 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.

Capítulo 2 | ESTUDO DO PROBLEMA

2.1. CONCEITOS FUNDAMENTAIS PARA A COMPREENSÃO DO PROBLEMA

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

Conceito 1 | *Software* de Apoio à Decisão

De uma maneira simples, podemos definir um *software* de Apoio à Decisão como um *software* que auxilie o seu 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.

Conceito 1.1 | Utilizadores deste *Software*

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

Conceito 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 recear cálculos errados, e possuindo toda a precisão de um programa informático.

Conceito 2 | Para que serve um *Software* de apoio à selecção de *Software* de Apoio à Decisão

À medida que o mundo informático vai evoluindo dentro das empresas, torna-se conveniente o uso de um Software de Apoio à Decisão. Actualmente, estão disponíveis diversos sistemas deste tipo, mas é importante saber escolher qual o melhor sistema a adoptar, tendo em conta as necessidades e as expectativas do utilizador. Essa escolha pode ser feita por intermédio de um *software* de apoio à escolha de Software de Apoio à Decisão.

Conceito 2.1 | Utilizadores deste tipo de *Software*

O *software* que vamos desenvolver pode ser utilizado no meio empresarial, essencialmente para efeitos estratégicos, isto é, que após a sua utilização seja eleito um software que se torne uma fonte de lucro para a empresa.

Esta "ferramenta" pode também ser usada por um utilizador isolado que pretenda, por quaisquer motivos, investir num *software* destes.

Conceito 2.2 | Importância deste tipo de *Software*

Decidir é uma actividade do nosso quotidiano, é feita diariamente no mundo empresarial e saber escolher pode ser um processo mais ou menos complexo.

Actualmente existem disponíveis no mercado diversos *Softwares* de Apoio à Decisão, mas como escolher o mais adequado?

Esta escolha passa pela utilização de uma ferramenta que auxilie os interessados, ou futuros compradores, destes *software* na escolha mais apropriada. Desta forma, é possível fazer o investimento no produto de *software* certo para atender às necessidades que o próprio utilizador expõe, evitando assim prejuízos e desperdícios originados por decisões erradas.

2.2. CONTEXTUALIZAÇÃO DO PROBLEMA

2.2.1 Em que se baseia o problema em questão?

O número de *Softwares* de Apoio à Decisão tem vindo a aumentar. Alguns possuem aspectos que podem ser úteis num determinado contexto, mas noutros 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.

2.2.2 Porquê a realização deste problema?

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.

Capítulo 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.

Capítulo 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 o carregamento de 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 *software* 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** - menemónica para Specific, Mesurable, Attainable, Relevant, Time-bound
- **AHP** - Analytic Hierarchy Process
- **ValueFn** – Value Functions

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.

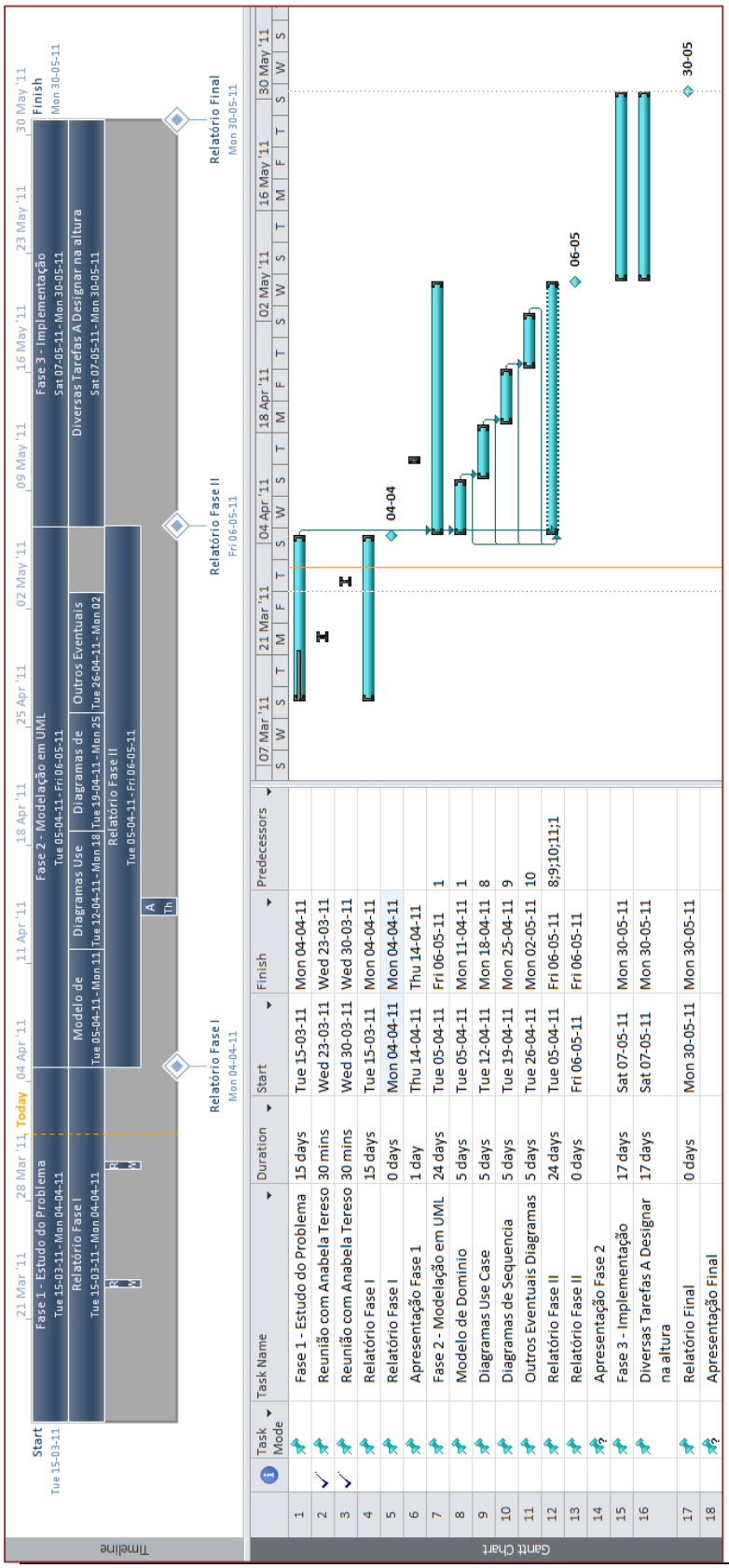
O **AHP** é frequentemente usado por grandes equipas para resolução de problemas muito complexos. Eis alguns exemplos de onde o AHP pode ser aplicado:

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

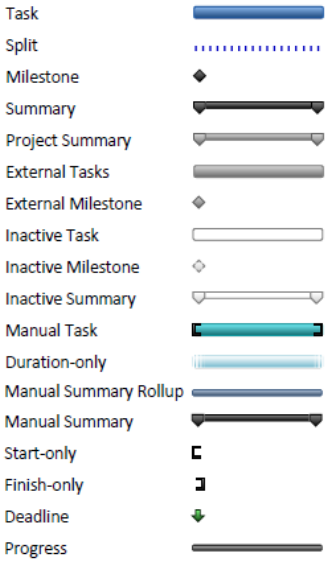
O **ValueFn** corresponde a uma função que mapeia directamente a avaliação das alternativas, podendo ser maximizada ou minimizada (consoante a pretensão do utilizador em maximizar ou minimizar o atributo em causa).

Capítulo 5 | PLANEAMENTO DE ACTIVIDADES

5.1 DIAGRAMA PREVISTO DO PLANEAMENTO DE ACTIVIDADES



Legenda



Este diagrama foi realizado utilizando a ferramenta Microsoft Project.

Capítulo 6 | ESPECIFICAÇÃO UML

Após a recolha e análise dos requisitos necessários para a nossa aplicação, segue-se a fase de especificar 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.

Capítulo 7 | DIAGRAMAS DE CASOS DE USO

7.1. DIAGRAMA DE CASOS DE USO GERAL

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

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 base de dados nova.

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 são 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 de páginas ou artigos exteriores ao programa, tal como os ficheiros de ajuda, em que o utilizador poderá, através de uma procura com inserção de palavras chave, encontrar artigos que possam responder à sua dúvida. Terá também uma secção de Tutoriais, que pode ser acedida através do caso de uso *Consult Tutorial*. Aí, é disponibilizada uma página com links para os tutoriais disponíveis.

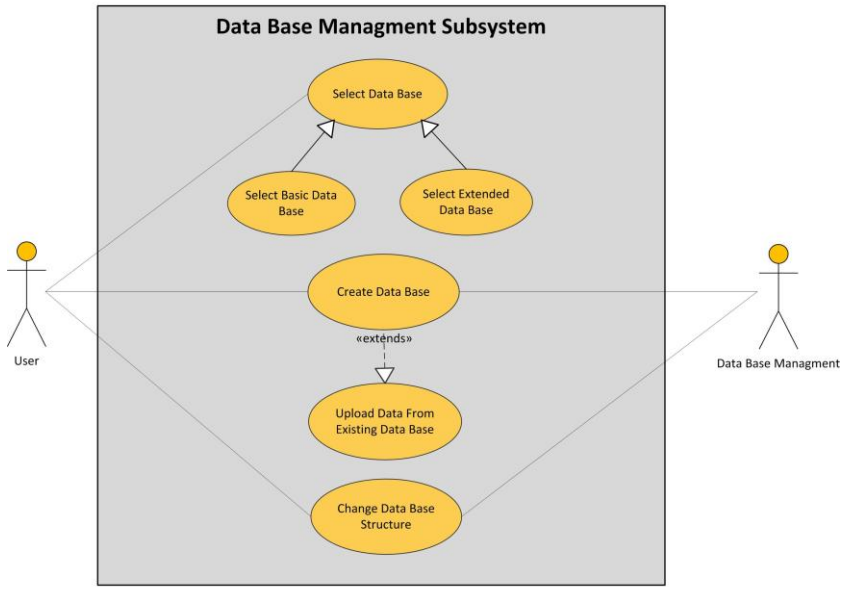
Por fim, os *Comparation Related Use Cases* são responsáveis por todo o processo de decisão do software mais adequado para uma determinada situação. Este grupo possui casos de uso que dão a possibilidade ao utilizador de escolher que softwares pretende usar na sua comparação, definir as características 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.

7.2. DIAGRAMAS DE CASOS DE USO REFINADOS

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

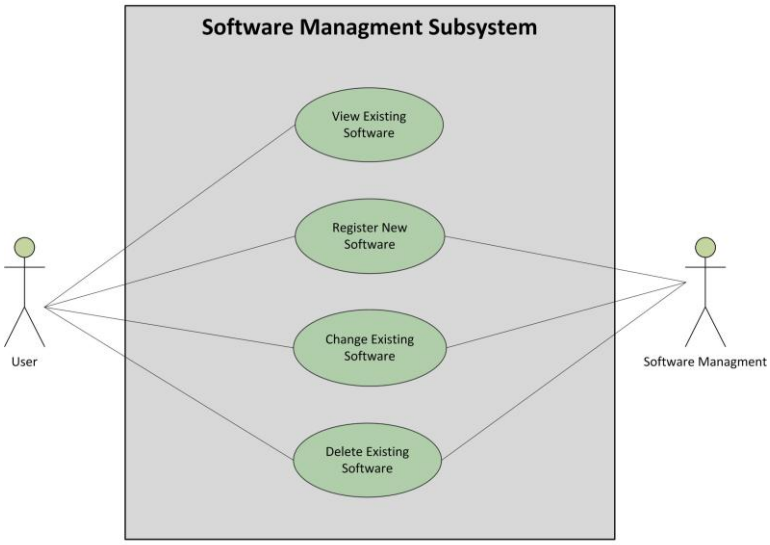
7.2.1. Data Base Managment Subsystem

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



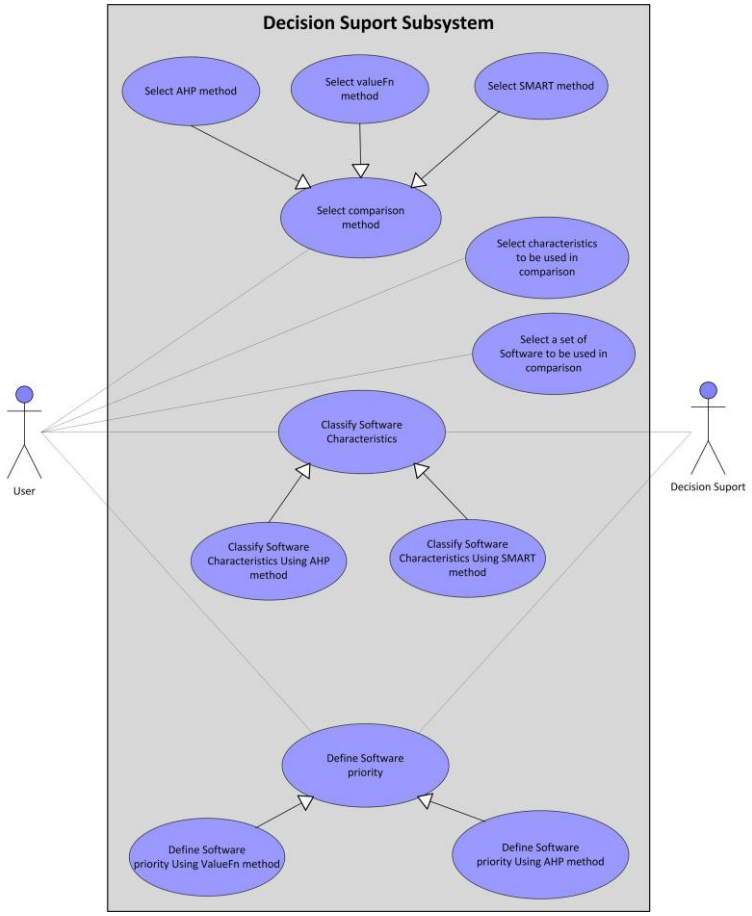
7.2.2. Software Managment Subsytem

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



7.2.3. Decision Suport Subsystem

Todas as tarefas relativas à escolha de critérios de classificação e avaliação dos Softwares estão especificadas neste Use Case:



7.3. DESCRIÇÃO TEXTUAL DE CASOS DE USO

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 Registed		
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			
Author			
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 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

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

Exception 4a: Invalid Software Data		**Actor Input**	**System Response**
	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**	**System Response**
	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**	**System Response**
	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**	**System Response**
	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**	**System Response**
	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 numbesr 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 prioritys 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	System Response
	1	Select minimize	
	2		Uses the minimize formula to calculate the priority
	3		Returns to 5 of the Main Flow

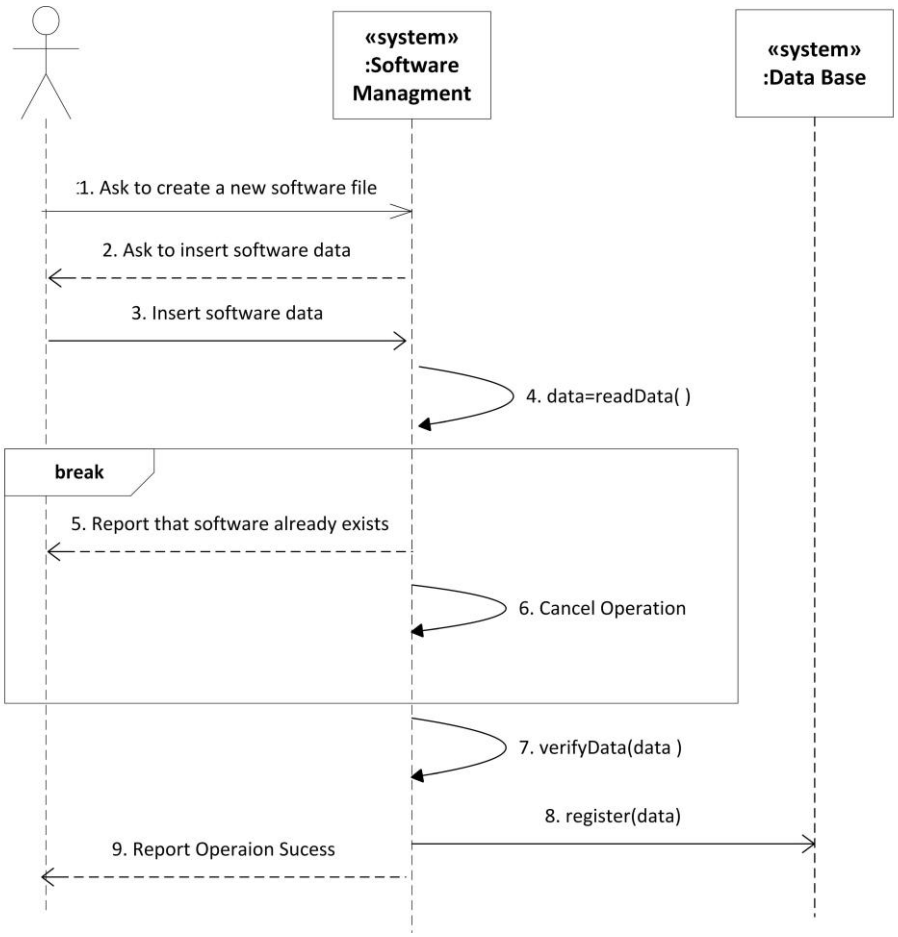
USE CASE: Define Software Characteristics priority Using AHP method			
Super Use Case	Define Software Characteristics priority		
Author			
Date			
Brief Description			
Preconditions	AHP method is selected and the characteristics to be used in comparasion are classified		
Post-conditions	Characteristics 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: The numbers aren't equals		Actor Input	System Response
	1		Return to 3 of the Main Flow
Alternative 11a: The numbers aren't equals		Actor Input	System Response
	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	

Capítulo 8 | DIAGRAMAS DE SEQUÊNCIA

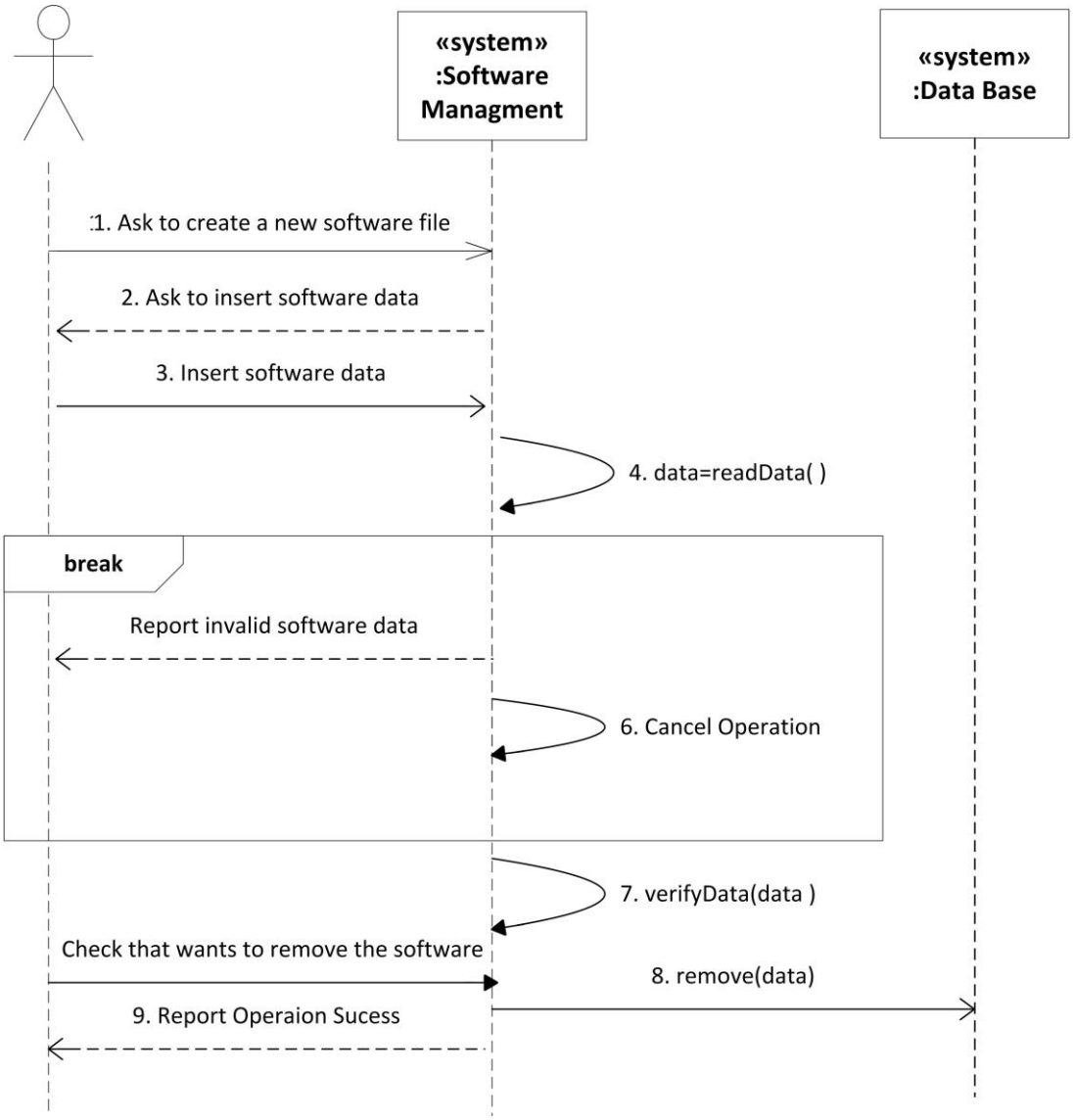
8.1. DIAGRAMAS DE SEQUÊNCIA RELATIVOS ÀS OPERAÇÕES DE REGISTO

8.1.1. 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 dados correspondentes na base de dados. Quando concluído o registo, é comunicado o sucesso da operação ao utilizador.

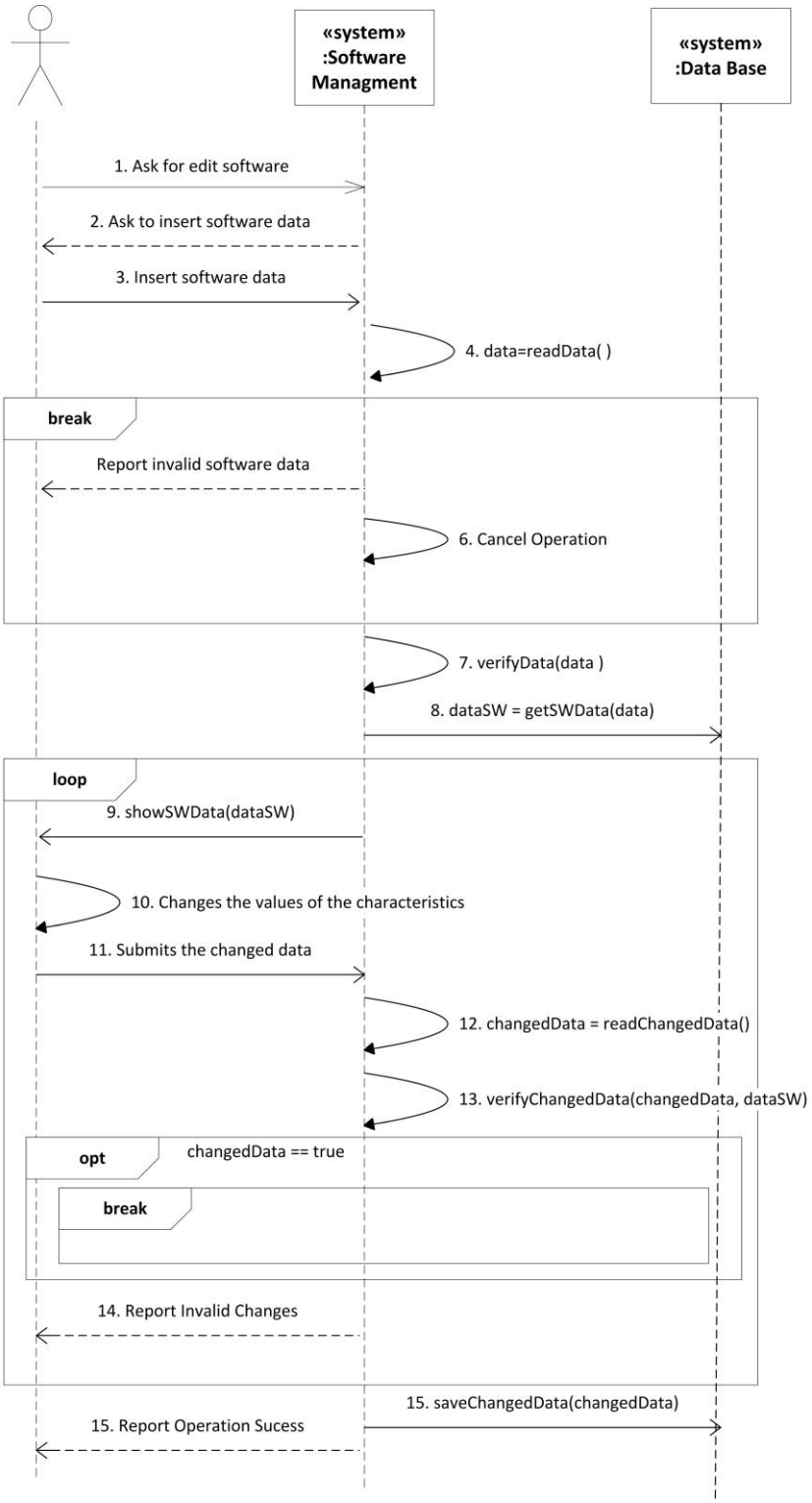
8.1.2. Delete Existing Software



Aqui, 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 excepção e será comunicado esse erro ao utilizador, sendo a operação cancelada de seguida.

Caso essa excepção 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 concluída esta operação, o utilizador é informado sobre o sucesso da operação.

8.1.3. Change Existing Software



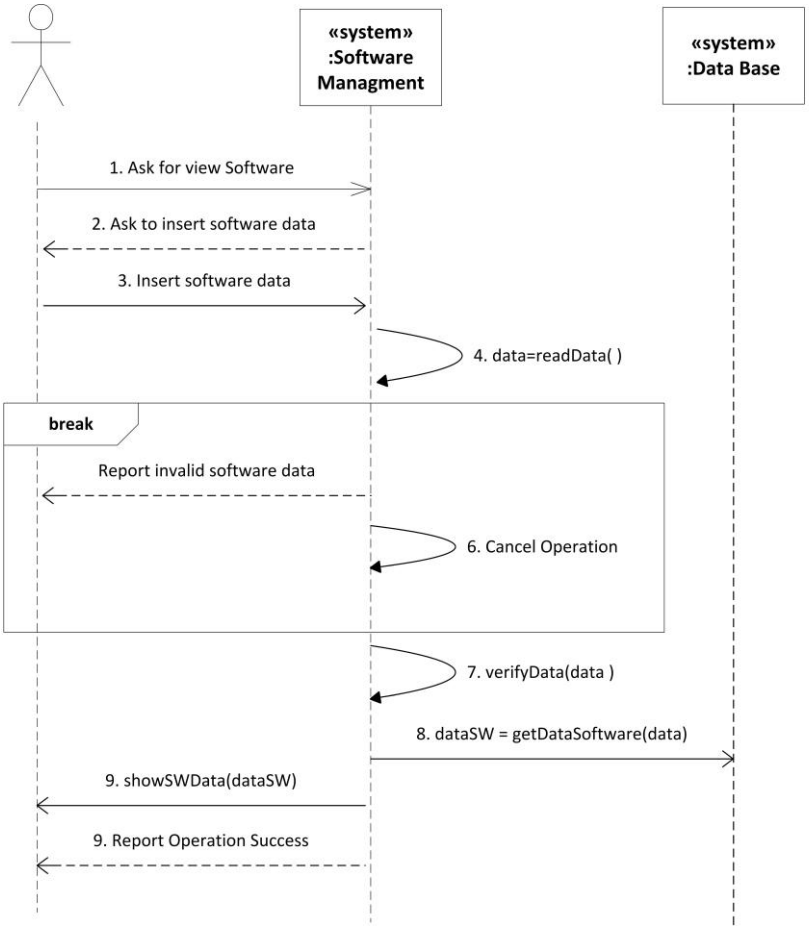
Após o utilizador pedir para editar um software, é-lhe pedido que insira os dados desse software, mais precisamente o nome, que será lido através do método **`readData()`** e será

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)`**. Depois, 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 passados 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.

8.1.4. View Existing Software



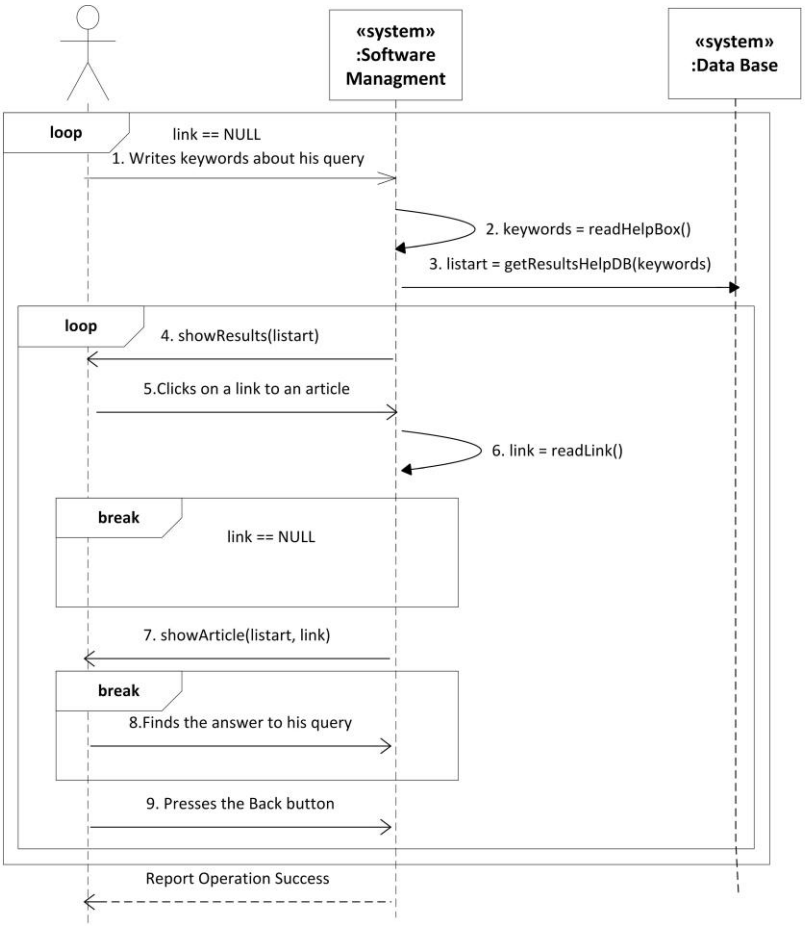
O utilizador selecciona a opção de ver um Software, pelo que lhe é pedido de seguida que insira dados do Software que este pretende consultar. Após essa inserção, o método **`readData()`** é chamado para ler esses dados e guarda a leitura na variável data, sendo testados nesse método se os dados desse 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.

8.2. Diagramas de Sequência relativos às operações de Consulta

8.2.1. Consult Help

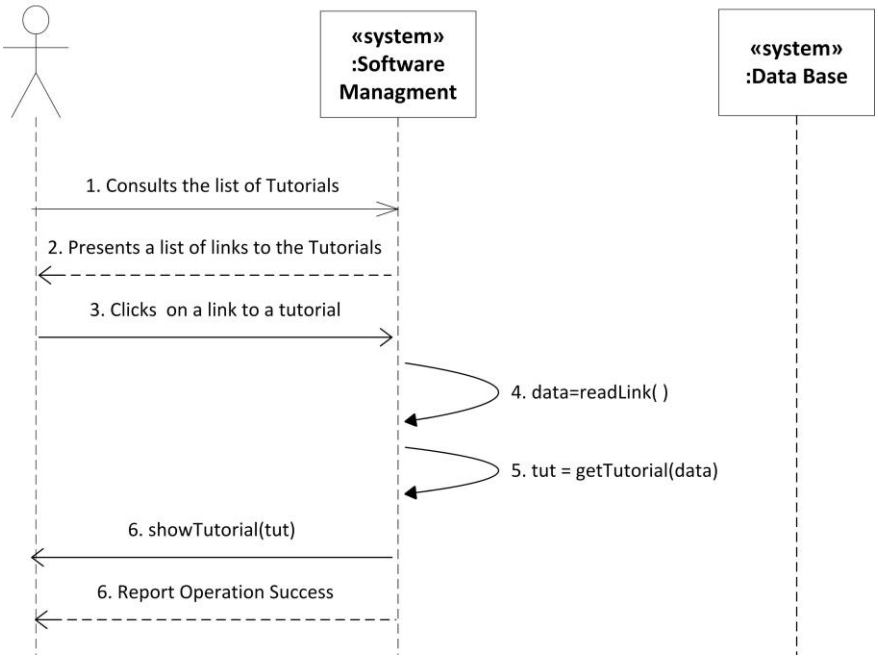


Primeiramente, o utilizador deve inserir palavras-chave sobre o problema que teve ou a dúvida que pretende tirar. Estas palavras serão armazenadas na variável **`keywords`**, sendo escritas lá através do método **`readHelpBox()`**. A variável **`keywords`** é então passada como parâmetro para o método **`getResultsHelpDB()`**, método este que tem como função seleccionar os artigos da base de dados de ajuda que se relacionam com as palavras inseridas pelo utilizador. Os resultados são guardados na variável **`listart`**.

Os links para os artigos são então apresentados através do método **`showResults()`**.

Os artigos que mais palavras-chave contiverem nos seus textos, serão apresentados em primeiro lugar nos resultados. Aí o utilizador escolhe um artigo e clica no seu *link*. Esse *link* é lido pelo método **`readLink()`** e é guardado na variável **`link`**. Então é chamado o método **`showArticle`**, que tem como função abrir o artigo, sendo para este método passados a lista de artigos (**`listart`**) e o *link* para o artigo respectivo. Se o utilizador encontrar a resposta à sua pergunta no artigo, é comunicado o sucesso da operação, caso contrário o utilizador poderá retroceder, sendo de novo chamado o método **`showResults`** e o processo de escolha um artigo recomeçará. Se porventura o utilizador não encontrar nenhum *link* que corresponda ao que procura, ele poderá fazer uma nova procura, inserindo novas palavras-chave, sendo todo o processo de procura de artigos recomeçado.

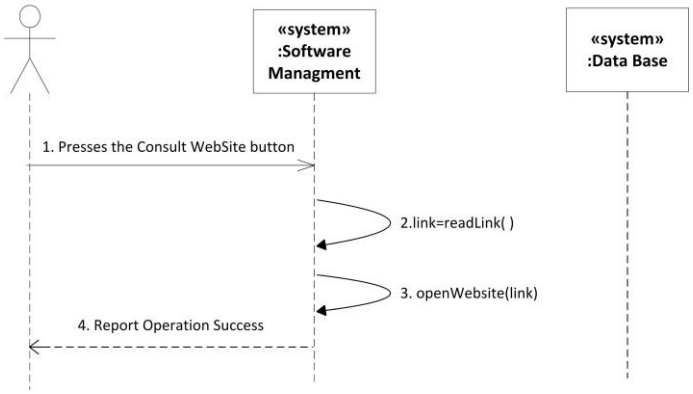
8.2.2. Consult Tutorial



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.

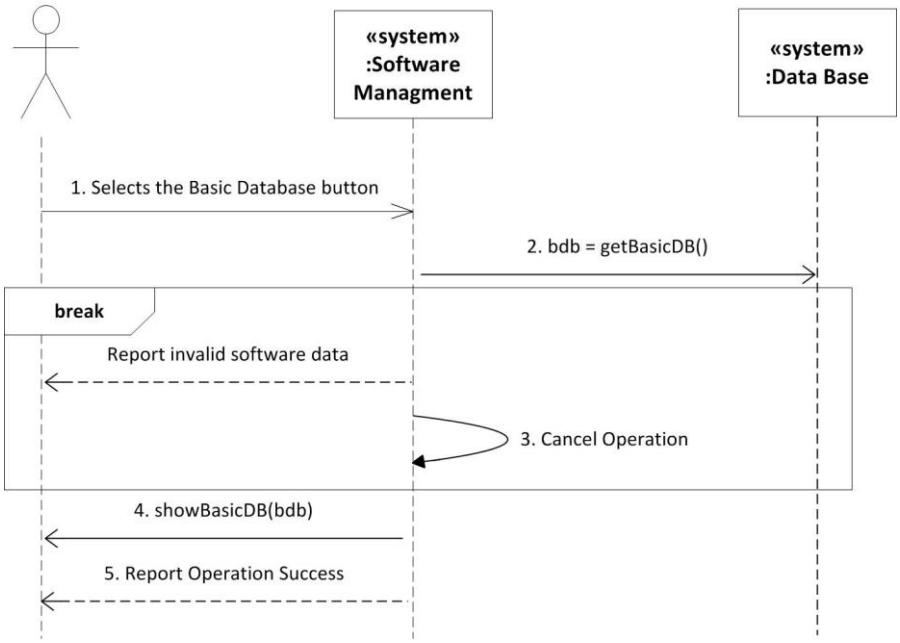
8.2.3. Consult Software's Web Site



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 pretende, sendo o sucesso da operação comunicado de seguida.

8.3. Diagramas de Sequência relativos às operações sobre a Base de Dados

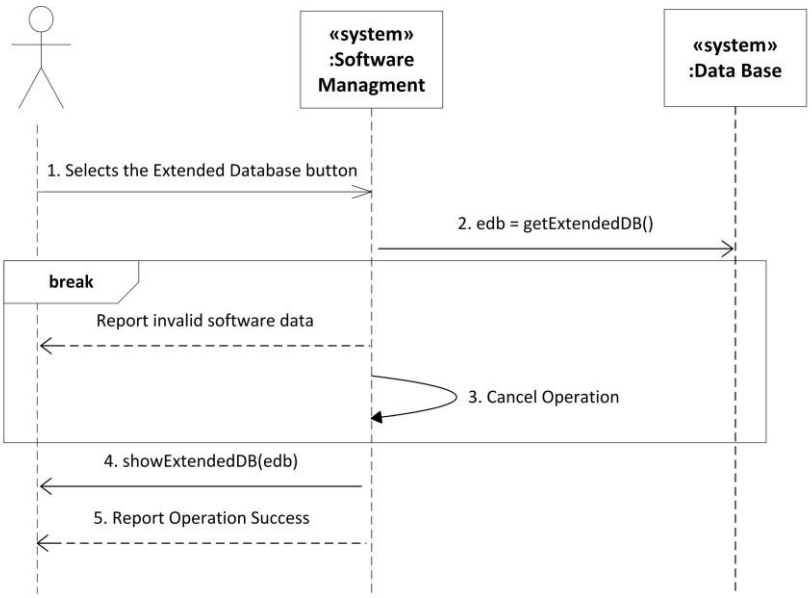
8.3.1. Select Basic Database



O utilizador carrega no botão para poder ver a base de dados básica. Aí, é chamado o método **`getBasicDB()`** que tratará de gerar essa base de dados e armazenar os resultados na variável

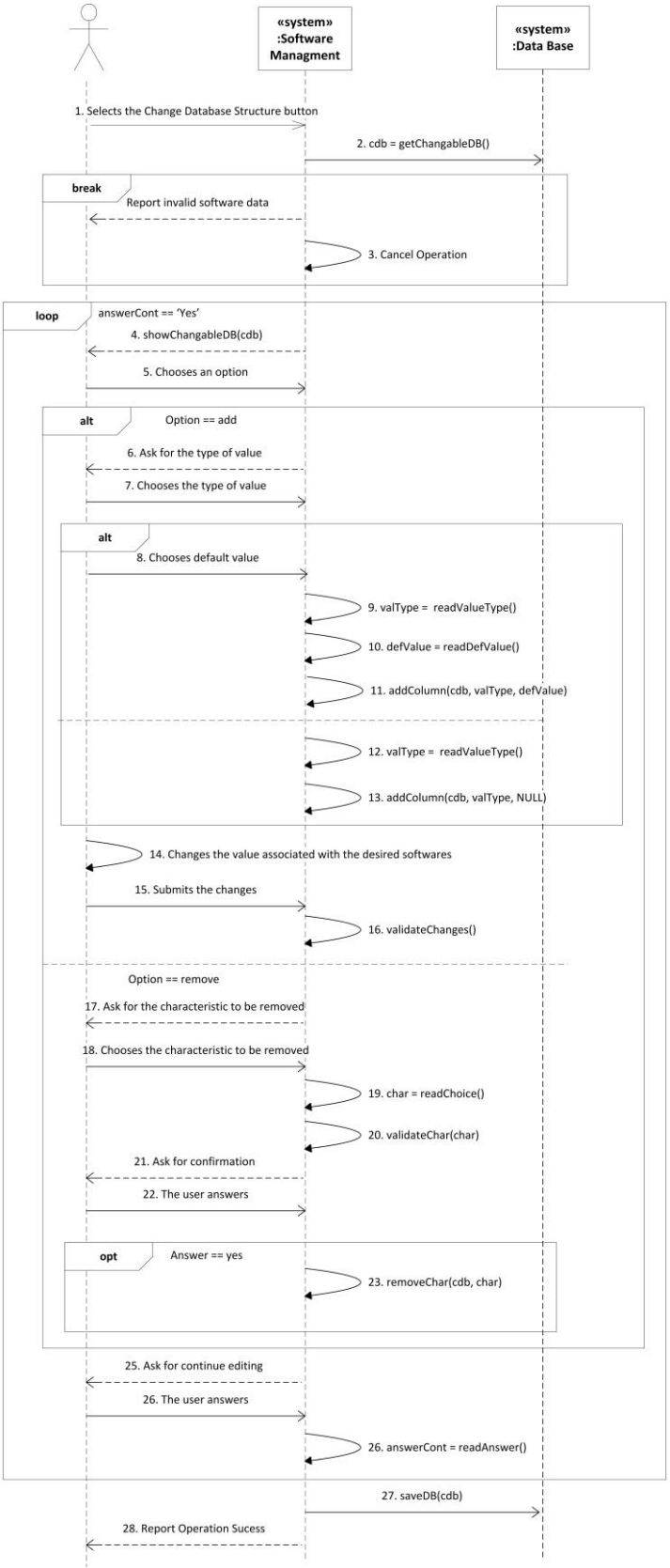
bdb. Caso isso não seja possível, é reportada uma excepção e a operação é cancelada. Não ocorrendo essa excepção, a base de dados é mostrada de forma organizada através do método **`showBasicDB()`** e é reportado o sucesso da operação.

8.3.2. Select Extended Database



Este caso de uso funciona de maneira análoga ao *Select Basic Database*, só que ao invés de mostrar a base de dados básica, mostrará a base de dados completa, isto é, com todas as características dos Softwares e os seus valores.

8.3.3. 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 será confrontado com 2 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. Então, 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 NULL em vez do **defValue**.

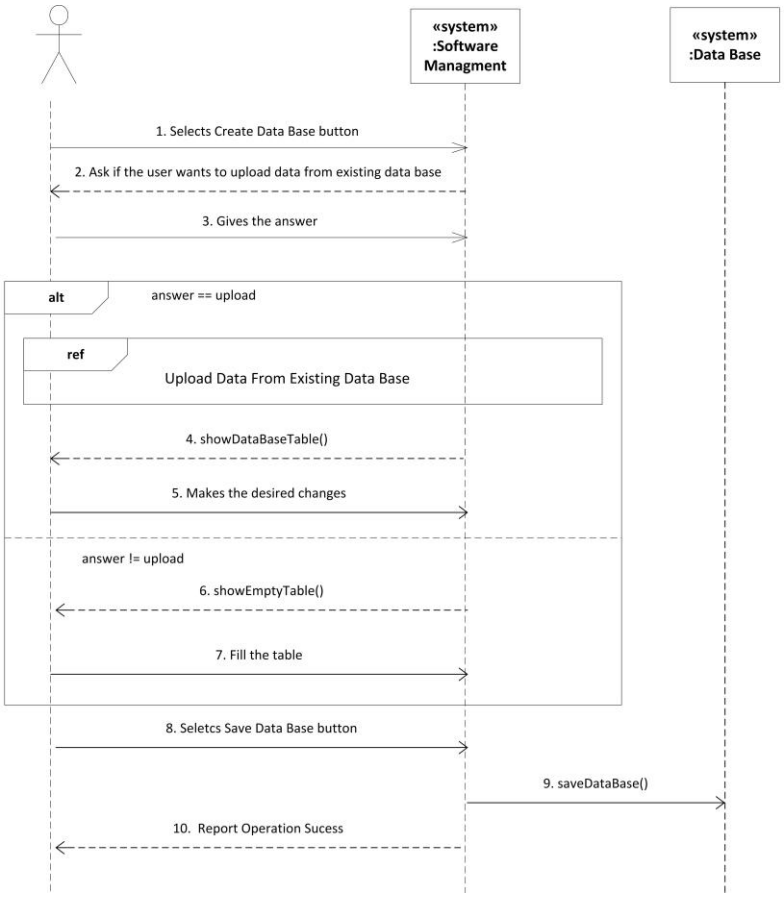
Aí, o utilizador poderá alterar o valor de todos os Softwares que pretender, especificando-os. Isto é depois validado através do método **validateChanges()**.

Mas o utilizador pode também querer remover uma característica. É-lhe pedido então que indique qual é a característica que este quer remover. Essa característica é identificada através do método **readChoice()** e é armazenada na variável **char**, sendo depois validada através do método **validateChar()**.

O utilizador é então 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 tanto a remoção como a adição de características, o utilizador terá de responder à pergunta sobre se este pretende continuar a editar a tabela. Se disser que sim, 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 parâmetros. Este método gravará a tabela nova na base de dados, sendo depois comunicado a conclusão da operação.

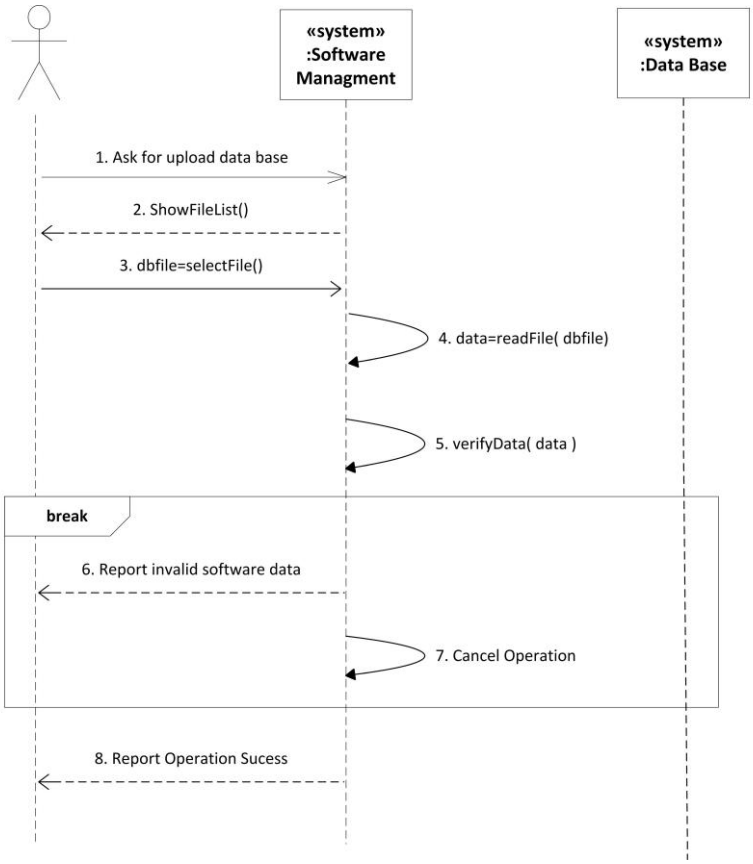
8.3.4. Create Database



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 **`showEmpyTable()`**, 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.

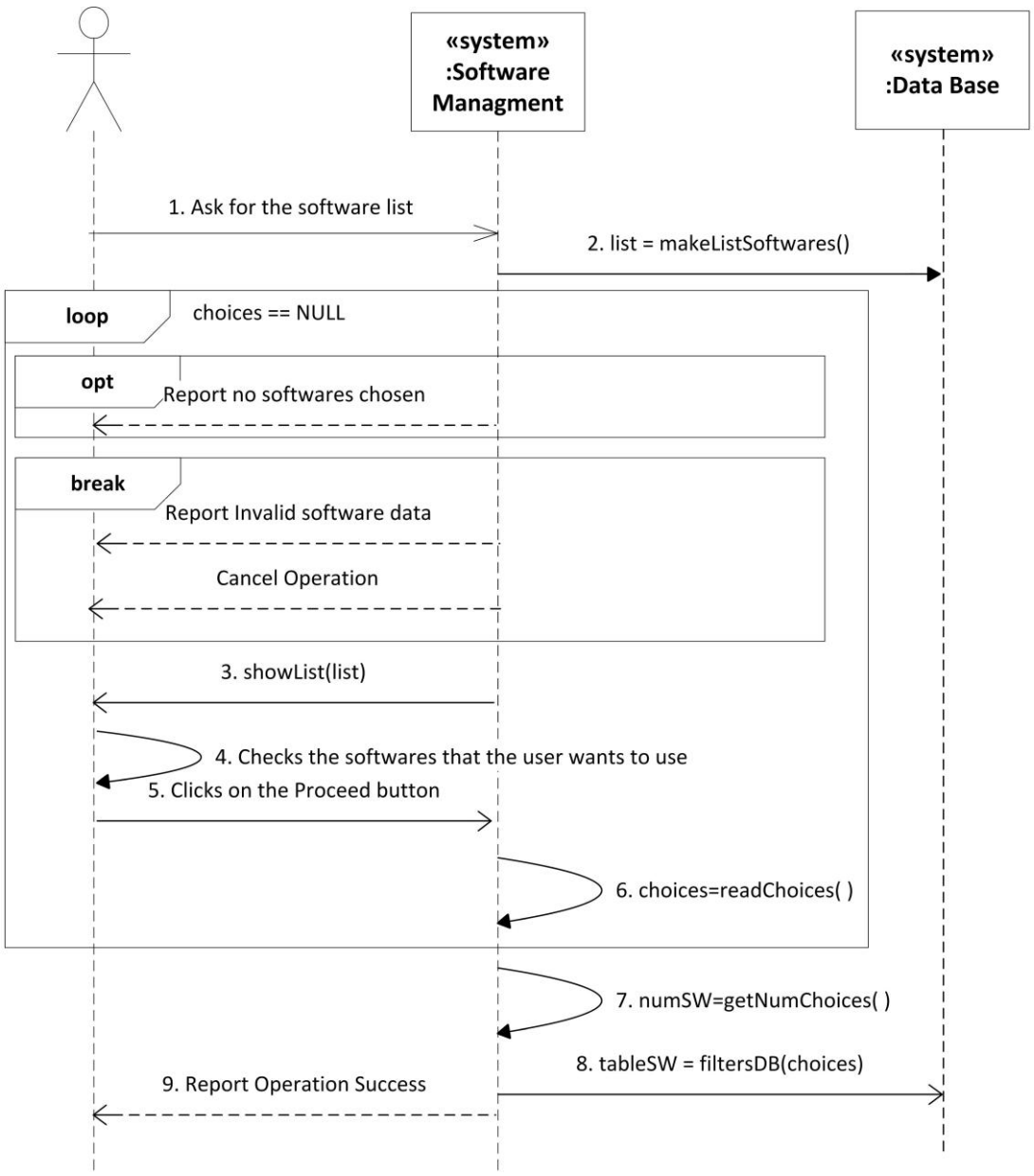
8.3.5. Upload Data from Existing Database



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.

8.4. Diagramas de Sequência relativos às operações de Comparação

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



Depois do utilizador pedir a lista de softwares, esta é gerada através do método **`makeListSoftwares()`** e é armazenada na variável **`list`**.

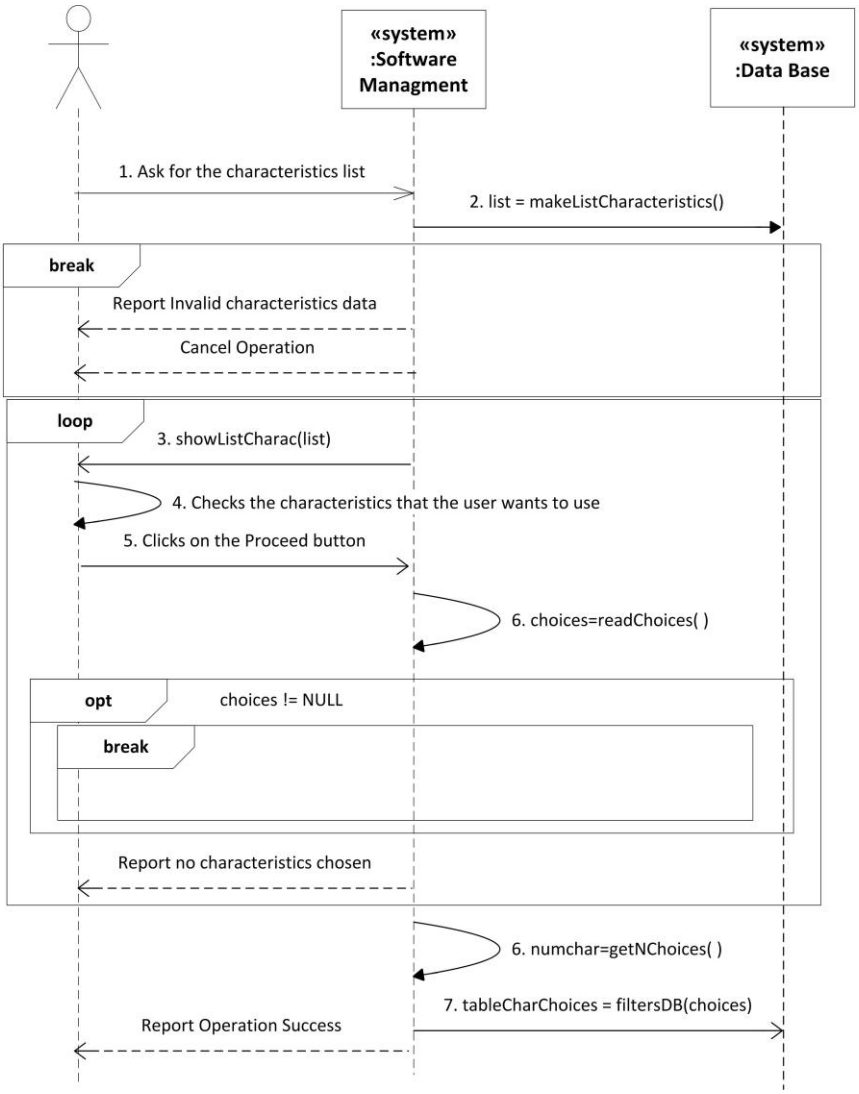
A lista é apresentada através do método **`showList()`**. Aí, o utilizador irá seleccionar quais os Softwares que quererá usar mais tarde na comparação. Quando tiver terminado a sua selecção, este deverá clicar no botão *Proceed*. As escolhas são lidas através do método

`readChoices()` e guardadas na variável **`choices`**. No caso de não terem sido feitas escolhas, será reportado ao utilizador que nenhuma escolha foi feita e o processo de escolha será recomeçado. Também no caso de haver algum problema com a leitura dos Softwares, será reportado um erro e a operação será cancelada.

Se o processo de escolha foi bem sucedido, o número de escolhas é armazenado na variável **`numSW`**. Na variável **`listSW`**, é armazenada a tabela dos Softwares que o utilizador escolheu. Esta tabela é gerada pelo método **`filtersDB`**, em que as escolhas (a variável **`choices`**) são passadas como argumento.

No final, é anunciado o sucesso da operação.

8.4.2. Select characteristics to be used in comparison

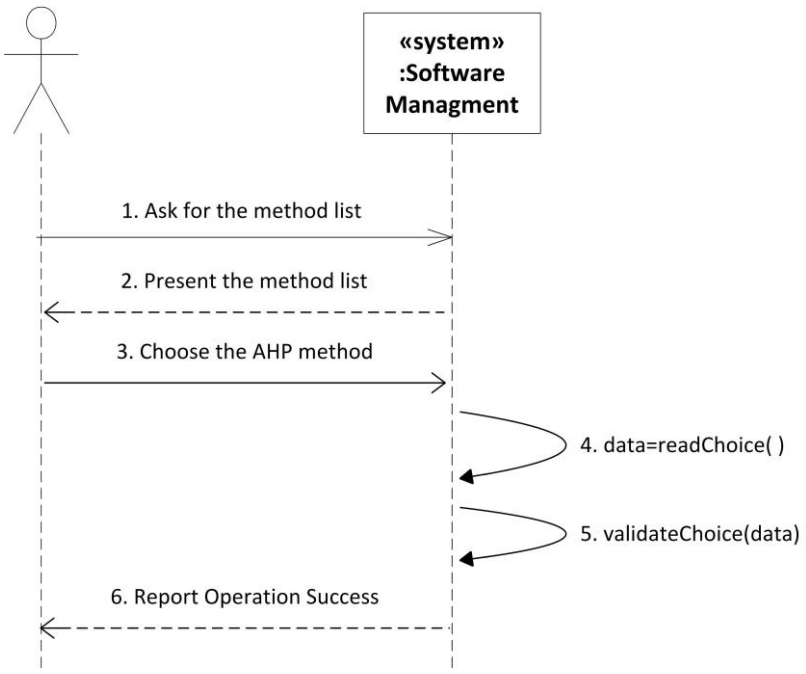


O utilizador deve escolher aqui as características que pretende usar na comparação. A lista de características é gerada pelo método **`makeListCharacteristics()`** e é guardada

em **`list`**. No caso de haver um problema na geração dessa lista, é reportada uma excepção e a operação é cancelada. Se não houver nenhum problema, esta lista é mostrada através do método **`showListCharac`**, sendo **`list`** passada como argumento.

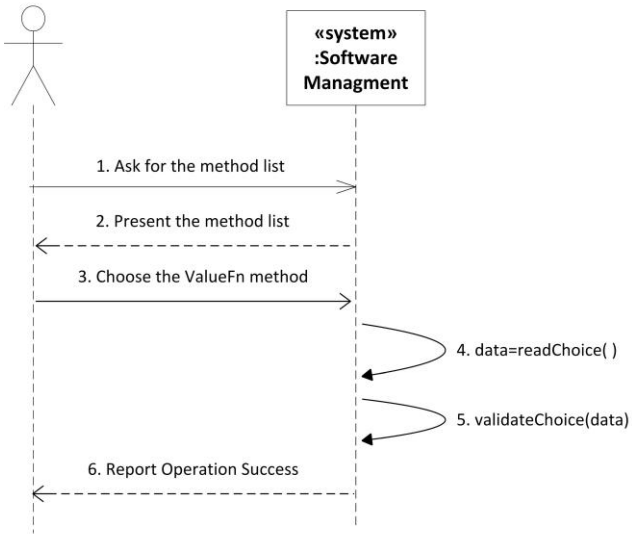
Aí o utilizador escolherá quais as características que quer então usar e, quanto tiver terminado, deve clicar no botão *Proceed*. As escolhas são lidas posteriormente pelo método **`readChoices()`** e armazenadas em **`choices`**. No caso de não terem sido feitas escolhas, tal situação será reportada ao utilizador e o processo de selecção voltará ao início. Se elas tiverem sido feitas com sucesso, em numchar é armazenado o número de escolhas que foram feitas (com o método **`getNChoices()`**). Já a base de dados é filtrada através do método **`filtersDB`**, sendo depois comunicado o sucesso da operação.

8.4.3. Select AHP method



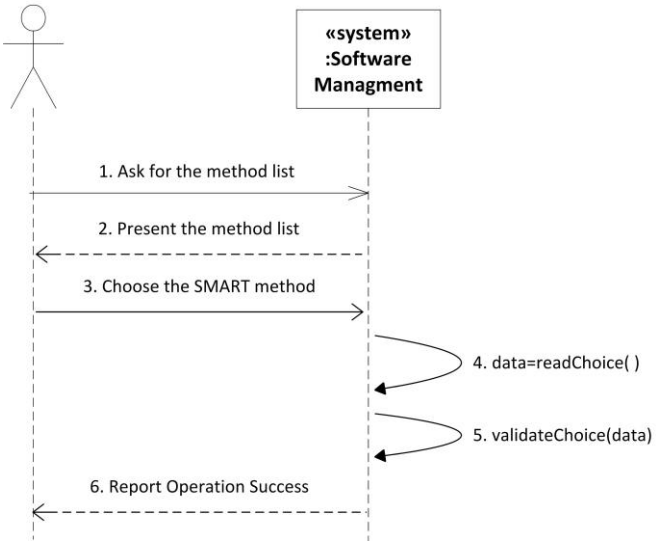
O utilizador, inicialmente, pede a lista de métodos existentes, sendo ela apresentada de seguida. Aí, este escolherá o método AHP (***Analytic Hierarchy Process)***, sendo essa escolha lida por **`readChoice`** e armazenada em data. Logo de seguida, a escolha é validada e a boa conclusão da operação é anunciada ao utilizador.

8.4.5. Select valueFN method



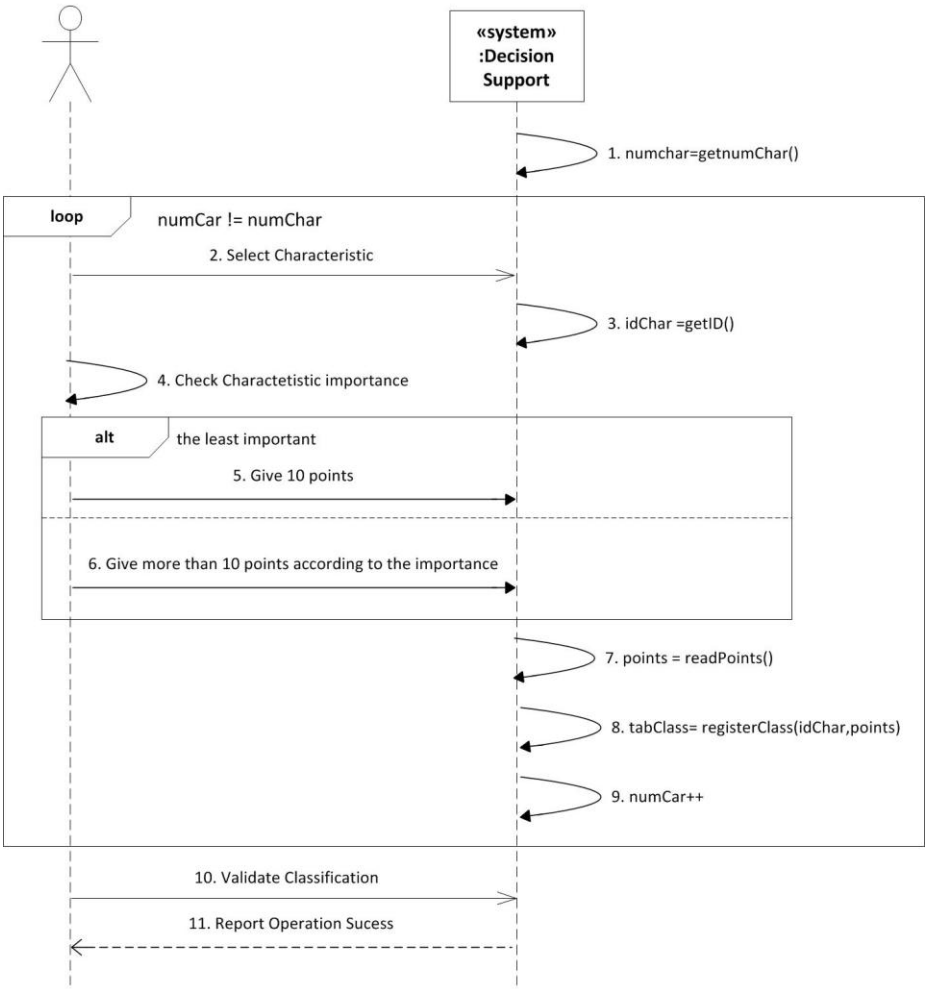
O utilizador começa por pedir a lista de métodos disponíveis, sendo esta apresentada de seguida. É seleccionado o método **valueFN** (valor da função), sendo essa escolha lida por **`readChoice`** e armazenada em data. Por fim, a escolha é validada e a é anunciado a conclusão da operação.

8.4.6. Select SMART method



Após ser apresentada a lista de métodos existentes, o utilizador escolhe o método **SMART** (*Specific Measurable Attainable Relevant Time-bound Evaluate Reevaluate*), sendo essa escolha lida por **`readChoice`** e armazenada em data. É comunicada a conclusão da operação após esta escolha ter sido validada.

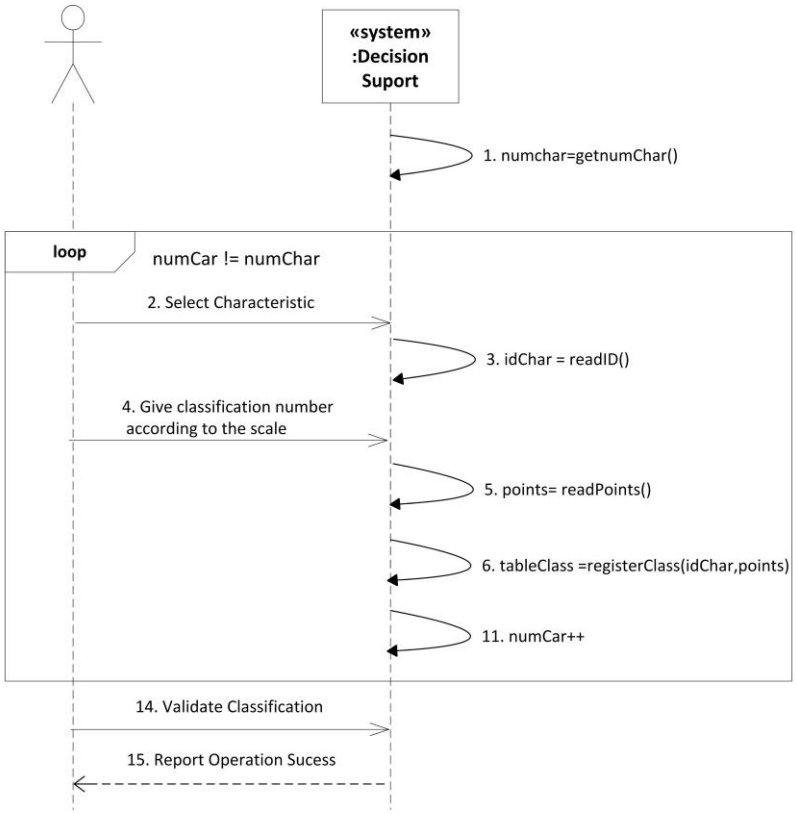
8.4.7. Classify Software Characteristic using SMART method



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 Use Case **Select characteristics to be used in comparison**. Como tal a primeira coisa que este faz é seleccionar da lista de selecções que efectuou anteriormente uma característica. De seguida quando esta é seleccionada o sistema lê o ID da característica para posteriormente a poder registar. Esta leitura é feita com o método **`getID().`** O passo seguinte será o utilizador que o terá que fazer. Este tem de ver qual a importância que aquela característica tem no resultado que pretende obter. Sendo assim este depara-se com duas situações, ou a característica é a que na sua opinião detém a menor importância, ou a é mais importante do que a com menor classificação. De acordo com a sua opinião este atribui 10 pontos se a característica for a menos importante de todas, caso nao seja a menos importante o utilizador atribui mais de 10 pontos de acordo com a classificação que atribuiu às outras características. As pontuações são lidas com o método **`readPoints().`** Como a característica já está classificada o sistema regista numa tabela que dada característica tem a classificação atribuída. O método utilizado para o registo é

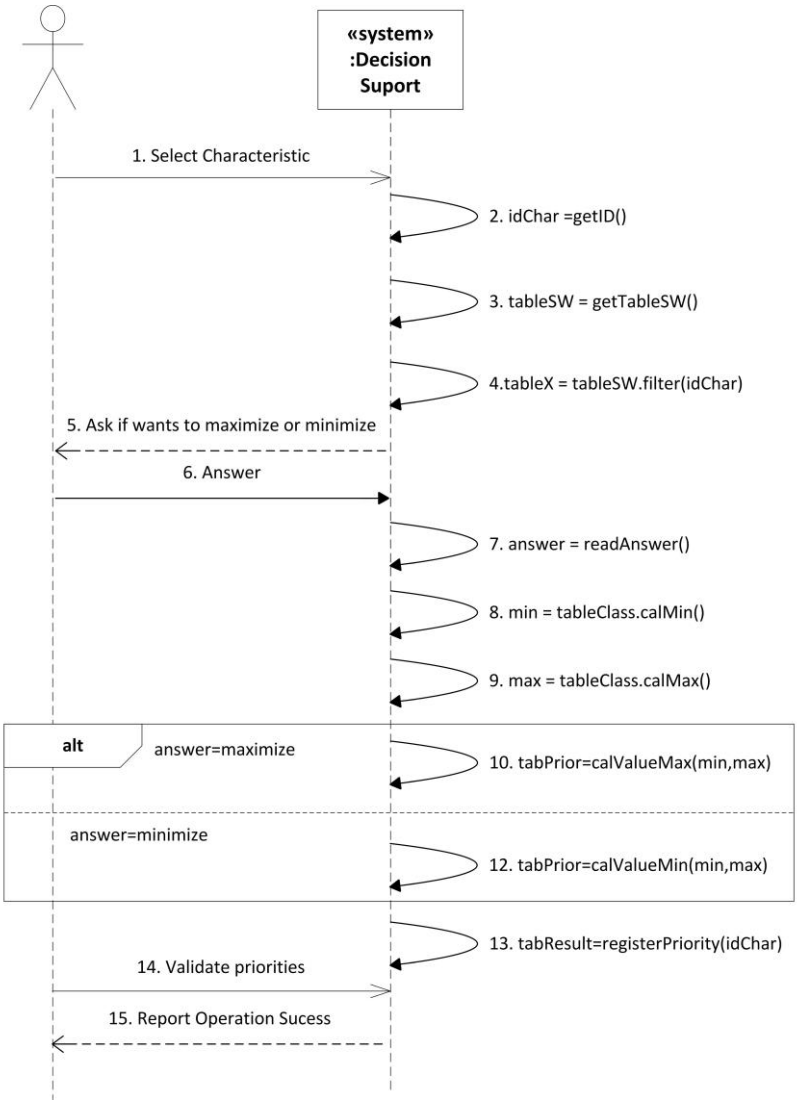
`registerClass(idChar,points),` que recebe como parâmetros o ID da característica e respectiva classificação. O número de classificações feitas é incrementado. Este processo é repetido enquanto o número de classificações não for igual ao número de características. Por fim quando todas as classificações forem atribuídas o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

8.4.8. Classify Software Characteristic using AHP method



Quando entra na fase de classificação de características, o utilizador pode escolher um de dois métodos. Neste caso o utilizador escolheu o método AHP. Nesta primeira fase o utilizador apenas tem dar uma classificação às características que seleccionou anteriormente como referenciado no Use Case **Select characteristics to be used in comparison**. Como tal a primeira coisa que este faz é seleccionar da lista de selecções que efectuou anteriormente uma característica. De seguida quando esta é seleccionada o sistema lê o ID da característica para posteriormente a poder registar. O utilizador terá, a posteriori, de acordo com a escala do método classificar a característica, dando um número fixo de pontos. Estes pontos serão lidos posteriormente com o método **`readPoints()`**. O passo seguinte é o registo da classificação da característica. Esta é feita com o método **`registerClass(idChar,points),`** ou seja, 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. Por fim quando todas as classificações forem atribuídas o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

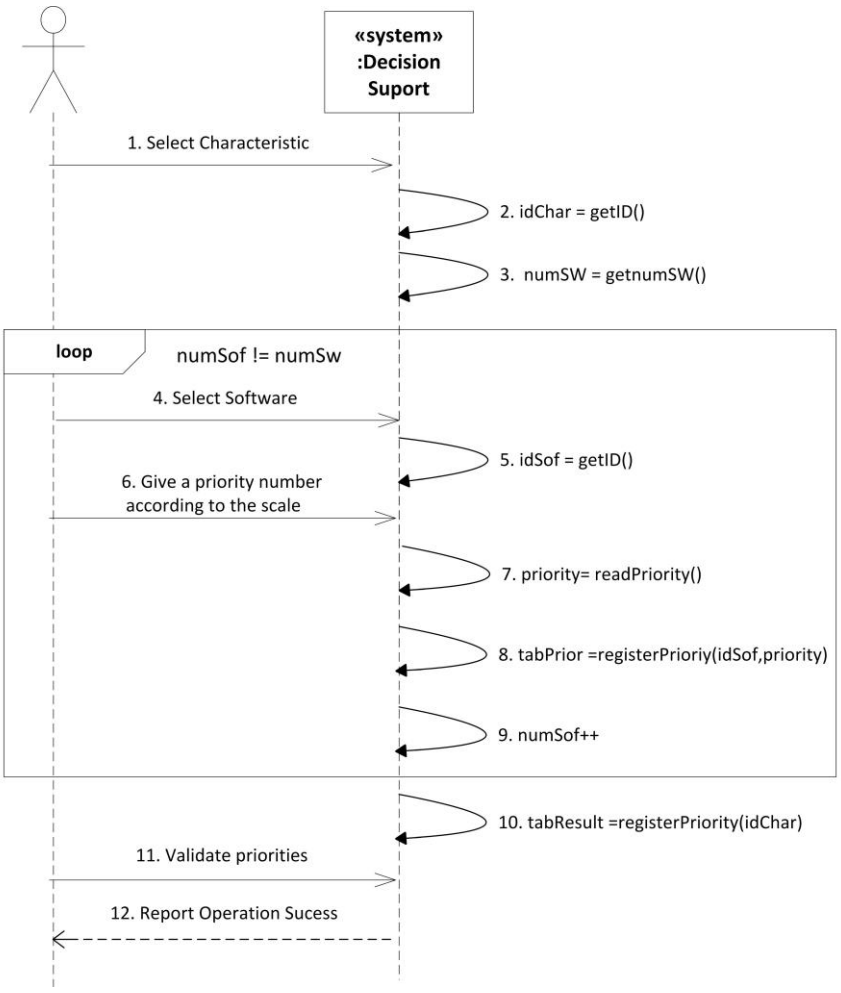
8.4.9. Define Software priority using ValueFn method



A definição de prioridades é a etapa seguinte da comparação de softwares. Como tal depois de definidas as classificações o utilizador pode agora escolher entre dois modos de cálculo de prioridades. Nesta situação o utilizador escolheu o método **ValueFn** para definir as prioridades. Como o utilizador pode escolher um método para cada característica, esta situação apenas retrata para uma característica. A primeira coisa que o utilizador faz é seleccionar uma das características. De seguida o sistema lê o ID da característica, depois vai buscar uma tabela de software's seleccionados com todas as suas características para comparação (esta tabela deriva do Use Case **Select a set of Softwares to be used in comparison**) e depois filtra esta mesma para ficar apenas o software e a classificação da característica escolhida ao início. Os métodos utilizados para efectuar estas operação são respectivamente **`getID(), getListSW(), filter(idChar)`**. Esta última recebe como parâmetros a tabela de softwares e o ID da característica. De seguida o sistema pergunta se o utilizador pretende maximizar ou minimizar. O utilizador responde e o sistema lê a resposta. A seguir o sistema calcula 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 é o 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, calculadas através dos dois métodos anteriores 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. Por fim quando todas as classificações forem atribuídas o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

8.4.10. Define Software priority using AHP method



A definição de prioridades é a etapa seguinte da comparação de softwares. Como tal depois de definidas as classificações o utilizador pode agora escolher entre dois modos de cálculo de prioridades. Nesta situação o utilizador escolheu o 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 coisa que o utilizador faz é seleccionar uma das

características. De seguida o sistema lê o ID da característica e depois vai buscar o número total de softwares seleccionados (este deriva do Use Case **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 deu para aquela característica daquele software como o método **`readPriority().`** De seguida o sistema regista 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 passo a seguir é incrementar o número de softwares classificados.

Este processo é repetido enquanto o número de classificações não for igual ao número de softwares. Antes de terminar o sistema associa para aquela 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. Por fim quando todas as classificações forem atribuídas o utilizador valida as suas classificações e o sistema retorna uma mensagem de sucesso da operação.

Capítulo 9 | DIAGRAMAS DE CLASSES

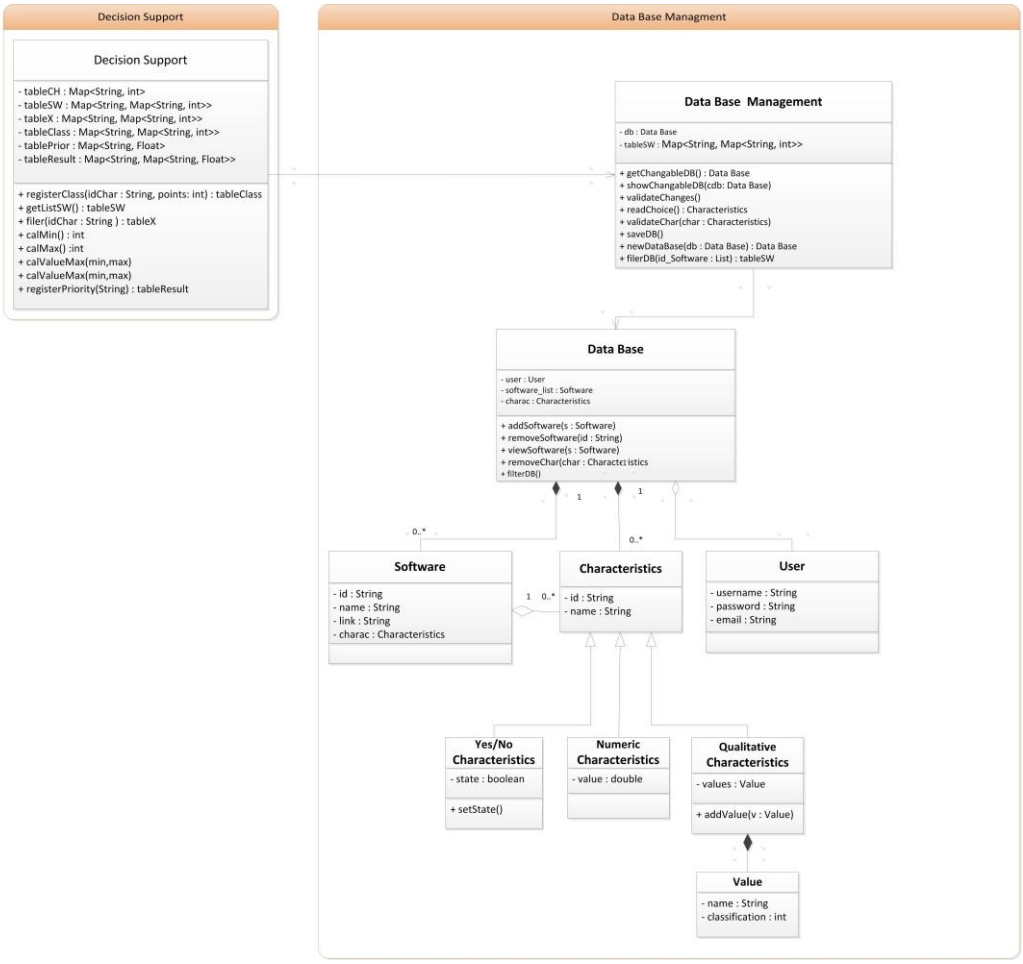
O Diagrama de Classes permitiu-nos estruturar a nossa aplicação. Assim, neste diagrama foram designadas as classes, com as respectivas variáveis de instância e métodos. São também ilustradas as relações entre as classes.

Essencialmente temos uma classe *Data Base* que contem a informação de todos softwares que o *user* possui e para gerir essa informação foi criada a classe *Data Base Management*.

A classe *Data Base* contem o *User* que corresponde há informação do utilizador, contem uma lista de Softwares (*software_list*) e uma lista de caracteristicas (*charc*).

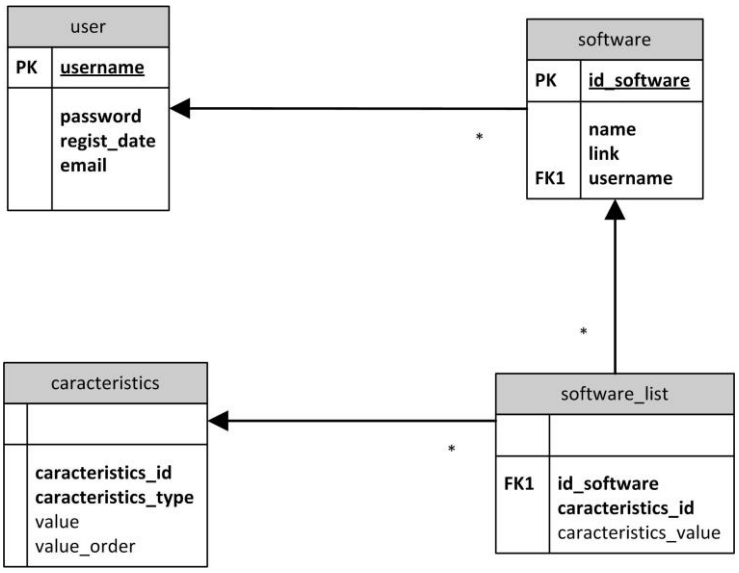
Characteritics é uma super classe que tem como subclasses *Yes/No Characteristics*, *Numeric Characteristics*, *Qualitative Characteristics*. Estas são as que guardam a informação que cada classifica cada software.

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



Capítulo 10 | ESQUEMA RELACIONAL DA BASE DE DADOS

Para guardar a informação na Base de Dados de uma forma simples e funcional estruturamos da seguinte forma:



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 contêm o id (*id_software*), o nome (*name*), o respectivo link da página web (*link*) 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 contêm o id do software (*id_software*), a identificação do tipo de característica (*caracteristics_id*), por exemplo: verdadeiro ou falso, e o valor da característica (*caracteristics_value*).

A informação de cada caracteristica está guardada em *caracteristics*, esta tabela contem o id (*caracteristics_id*) o tipo (*caracteristics_type*). Para as caracteristicas qualitativas, há também o atributo *value* que contem um dos valores possíveis e a ordem de importância que têm, ou seja, vamos ter algo como:

caracteristics_id	*caracteristics_type*	*value*	*value_order*
14	qualitative	bad	1
14	qualitative	normal	2
14	qualitative	good	3

Capítulo 11 | CONCLUSÃO

Nesta fase intermédia do projecto, tivemos de lidar com a especificação em UML do projecto, tendo como objectivo especificar o projecto de forma suficientemente precisa para nos auxiliar na 3ª fase, em que procederemos à sua implementação.

Esta especificação foi estruturada em várias fases. Inicialmente definimos os casos de uso, para que estes incluíssem todas as funcionalidades definidas na etapa 1. Depois, passamos para o refinamento dos casos de uso, onde foram definidos os subsistemas em que a aplicação se divide. Seguidamente foram feitas as descrições textuais de todos os casos de uso.

Na fase seguinte, foram concebidos os diagramas de sequência, de acordo com os casos de uso e as respectivas descrições textuais.

De seguida, foi desenvolvido o diagrama de classes, que representa as classes que serão implementadas no projecto.

Por fim, elaboramos o esquema da base de dados, que ilustra a sua estrutura, a relação entre as diferentes tabelas e o seu funcionamento.

Após esta etapa de especificação do projecto, todos os elementos estão reunidos para iniciarmos a fase final do projecto, que retracta a implementação da aplicação.