|  |  |
| --- | --- |
|  | G:\um.png |
|  | 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](#_Toc289716987)

[Capítulo 2 | Estudo do problema 2](#_Toc289716988)

[2.1. Conceitos fundamentais para a compreensão do problema 2](#_Toc289716989)

[Conceito 1 |](#_Toc289716990) *[Software](#_Toc289716990)* [de Apoio à Decisão 2](#_Toc289716990)

[Conceito 1.1 | Utilizadores deste](#_Toc289716991) *[Software](#_Toc289716991)* [2](#_Toc289716991)

[Conceito 1.2 | Importância deste](#_Toc289716992) *[Software](#_Toc289716992)* [3](#_Toc289716992)

[Conceito 2 | Para que serve um](#_Toc289716993) *[Software](#_Toc289716993)* [de apoio à selecção de](#_Toc289716993) *[Software](#_Toc289716993)* [de Apoio à Decisão 3](#_Toc289716993)

[Conceito 2.1 | Utilizadores deste tipo de](#_Toc289716994) *[Software](#_Toc289716994)* [3](#_Toc289716994)

[Conceito 2.2 | Importância deste tipo de](#_Toc289716995) *[Software](#_Toc289716995)* [3](#_Toc289716995)

[2.2. Contextualização do problema 4](#_Toc289716996)

[2.2.1 Em que se baseia o problema em questão? 4](#_Toc289716997)

[2.2.2 Porquê a realização deste problema? 4](#_Toc289716998)

[Capítulo 3 | Objectivos do projecto 5](#_Toc289716999)

[Capítulo 4 | Análise de requisitos 6](#_Toc289717000)

[4.1. Requisitos da Interface 6](#_Toc289717001)

[4.2. Requisitos da Base de Dados 6](#_Toc289717002)

[4.3. Requisitos a nível de métodos de selecção 7](#_Toc289717003)

[Capítulo 5 | Planeamento de Actividades 8](#_Toc289717004)

[5.1 Organização das fases seguintes 8](#_Toc289717005)

[5.2 Diagrama Previsto do planeamento de actividades 9](#_Toc289717006)

[Capítulo 6 | Conclusão 10](#_Toc289717007)

# 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 *Software*s 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.

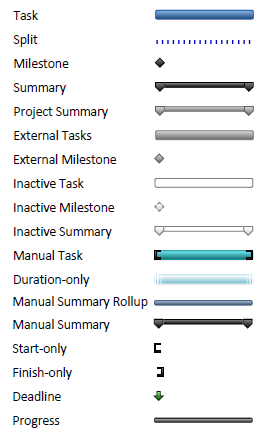
O **AHP** é frequentemente usado por grandes equipas para resolução de problemas muito complexos. Eis alguns exemplos de onde o AHP pode ser aplicado:

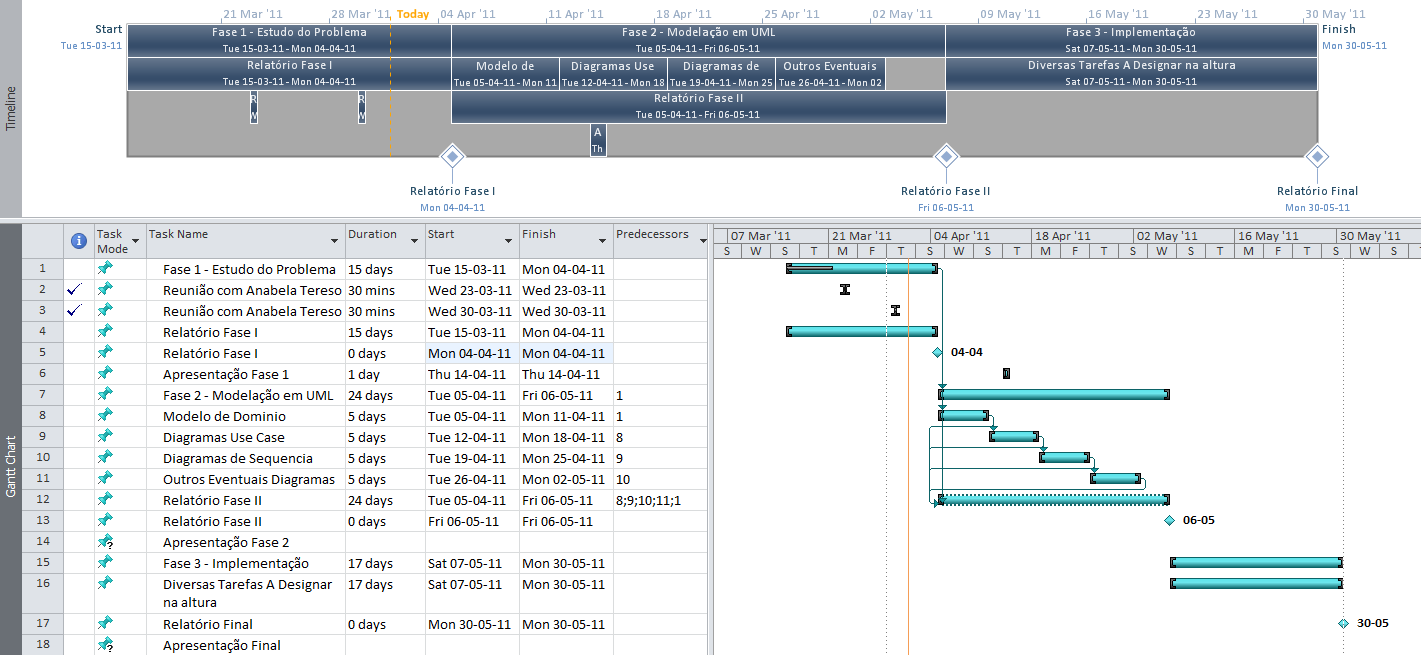
* Ranking: Colocar as alternativas do mais para o menos desejado.
* Definições de Prioridades:
* entre outros...

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:

(Imagem)

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:

IMAGEM

### 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:

IMAGEM

### 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:

IMAGEM

# 8

## 8.1. Diagramas de Sequência relativos às operações de Registo

### 8.1.1. Register New Software

IM

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

IM

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

IM

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.

### 8.2.6. 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.

### 8.2.7. 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

## 8.4. Diagramas de Sequência relativos às operações de Comparação

# Capítulo 9 | DIAGRAMAS DE CLASSES

# Capítulo 10 | ESQUEMA LÓGICO DA BASE DE DADOS

# Capítulo x | Conclusão

Numa primeira fase, tivemos como desafio perceber em que consistia o projecto proposto. Para tal, foi necessário de reflectir e apreender os aspectos fulcrais do nosso projeto, e posteriormente efectuar um estudo sobre o mesmo.

Nesta presente fase, ...

Por fim, foi importante desenvolver um plano de trabalho, ainda que não seja totalmente completo, mas contendo já as linhas gerais, para nos consciencializarmos das diferentes fases de concretização e entrega do projecto, dentro dos prazos estabelecidos. Assim, conseguimos uma pequena semelhança com o mundo empresarial.

Estamos satisfeitos com o trabalho desenvolvido até ao momento, pensamos ter atingido os objectivos a que nos propusemos para a realização desta etapa.