

**Universidade do Minho**

Escola de Engenharia

**Unidade Curricular:**

**Laboratórios de Informática III**

Ano Lectivo de 2014/2015

Janeiro, 2015

**Relatório de Desenvolvimento de Projecto**

**GestHiper - C**

**70719 Diogo Constâncio,**

**70513 Pedro Araújo**

# Índice

[Índice 2](#_Toc417652869)

[Índice de Figuras 3](#_Toc417652870)

[Fotos 4](#_Toc417652871)

[Resumo 5](#_Toc417652872)

[Introdução 6](#_Toc417652873)

[Estrutura da Aplicação 6](#_Toc417652874)

[GestHiper 7](#_Toc417652875)

[Catálogo de Produtos/Clientes 8](#_Toc417652876)

[Contabilidade 9](#_Toc417652877)

[Compras 10](#_Toc417652878)

[Modelação 11](#_Toc417652879)

[Modelo de Domínio 12](#_Toc417652880)

[Diagrama de *Use Cases* 12](#_Toc417652881)

[Diagrama de Classes 19](#_Toc417652882)

[Diagramas de Sequência 20](#_Toc417652883)

[Base de Dados 24](#_Toc417652884)

[Dificuldades encontradas 24](#_Toc417652885)

[Conclusões e Trabalho Futuro 25](#_Toc417652886)

# Índice de Figuras

Figura 1 – Diogo Constâncio 4

Figura 2 - Pedro Araújo 4

Figura 3 – Diagrama GestHiper 7

Figura 4 – Estrutura de Catálogos 8

Figura 4 – Estrutura de Contabilidade 9

Figura 4 – Estrutura de Compras 10

Figura 5 - Modelo de Domínio 12

Figura 6 - Diagrama de use Cases Principal 13

Figura 7 - Descrição Textual Do Use Case Autenticar 14

Figura 8 - Diagrama De Use Cases Gestão De Candidaturas 15

Figura 9 - Diagrama De Use Cases Gestão De Recursos 15

Figura 10 - Diagrama De Use Cases Gestão De Doações 16

Figura 11 - Diagrama De use Cases Gestão De Voluntários 17

Figura 12 - Diagrama De Use Cases Gestão De Stock 17

Figura 13 - Diagrama De Use Cases Gestão De Projectos 18

Figura 14 - Diagrama de Classes 19

Figura 15 - Diagrama De Classes Com DAOs 20

Figura 16 - Descrição Textual Do Use Case Dar Saída De Material 21

Figura 17 - Diagrama De Sequência De Sistema Dar Saída De Material 22

Figura 18 - Diagrama De Sequência Com Subsistemas Dar Saída De Material 23

Figura 19 - Modelo Da Base De Dados 24

# Fotos



Figura 1 – Diogo Constâncio



Figura 2 - Pedro Araújo

# Resumo

Este relatório complementa o Projecto Imperativo no âmbito da U.C. Laboratórios de Informática III, no qual se pôs em prática conhecimentos de modularidade, encapsulamento e desenvolvimento de estruturas de dados eficientes.

Conta ainda com conhecimentos e metodologias adquiridas nas U.C. Algoritmos e Complexidade, Arquitectura de Computadores, sob a forma de algoritmos optimizados e boas práticas de programação e escolha de estruturas de dados.

O projecto foi particionado em 4 fases sucessivas:

* Planeamentoe consequente desenvolvimento das estruturas de dados;
* Implementação dos módulos da biblioteca *GestHiper*;
* Escrita da interface com o utilizador em linha-de-comandos;
* Optimização das estruturas de dados;

Como apresentado, o projecto apresenta 3 módulos gerais:

**Main:** Interface em linha-de-comandos fornecida ao utilizador que executa operações sobre a biblioteca *GestHiper*, gere o input/output do utilizador fazendo também o tratamento destes.

**GestHiper:** Biblioteca que compõe os módulos de Catálogo de Produtos/Clientes, Contabilidade e Compras.

**Estruturas Básicas:** Bibliotecas de várias estruturas de dados que complementam os módulos da *GestHiper*, estruturas como *MinHeap*, *BST*, *HashTable* e *Stack*.

Este relatório trata principalmente a biblioteca *GestHiper*, e os módulos que a compõem.

# Introdução

## Estrutura da Aplicação

De acordo com a prática de modularidade utilizada no desenvolvimento do projecto, verifica-se que a biblioteca *GestHiper* foi construída de forma a que seja facilmente portável ao longo de sistemas operativos diferentes dada a atenção a métodos exclusivamente *standards*, e mesmo input/output diferentes, como por exemplo leitura de dados incremental, manual, ou apresentação de resultados/dados sob a forma de uma *GUI*.

Como tal o *Main* trata-se de um exemplo de utilização da biblioteca por parte de um utilizador em linha-de-comandos, sendo assim o único módulo a interagir directamente com o utilizador, pela *prompt*, ou em ficheiros.

[Makefile Diagram Here]

# GestHiper

Nesta secção são tratados cada um dos módulos pertencentes à biblioteca *GestHiper*¸ sendo eles **Catálogo** de Produtos/Clientes, **Contabilidade** e **Compras**.

É apresentada uma breve descrição dos encargos de cada um dos módulos, tratando de seguida a estrutura de dados utilizada no módulo, assim como as decisões que motivaram a sua escolha.

As estruturas de dados destes módulos foram desenvolvidas com atenção à resposta eficiente às perguntas que são colocadas no enunciado, pelo que a API será mencionada brevemente na descrição das estruturas de dados de cada módulo.

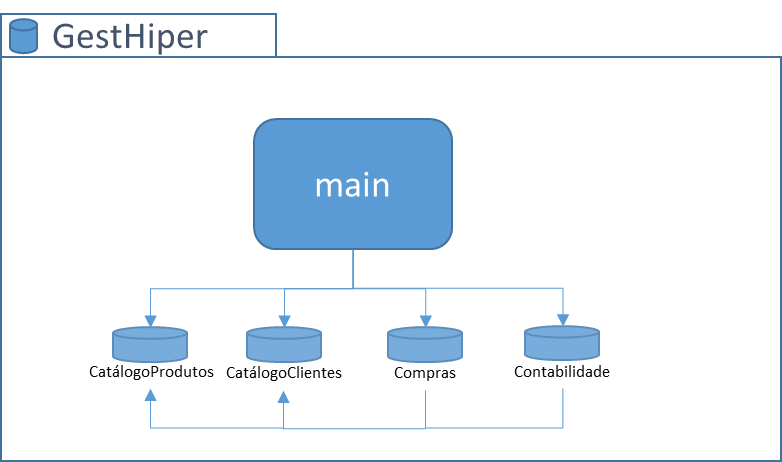


Figura 3 – Diagrama GestHiper

### Catálogo de Produtos/Clientes

São tratados aqui tanto o Catálogo de Produtos, como o Catálogo de Clientes, uma vez que são em tudo bastante semelhantes, diferenciando-se principalmente por terem comprimentos de código diferentes.

Estes módulos fazem a gestão das listas de Clientes/Produtos válidos, assim como as perguntas sob essas listas, que não incluem questões relativas a Compras ou Contabilidades, pelo que são totalmente independentes dos restantes módulos da biblioteca.

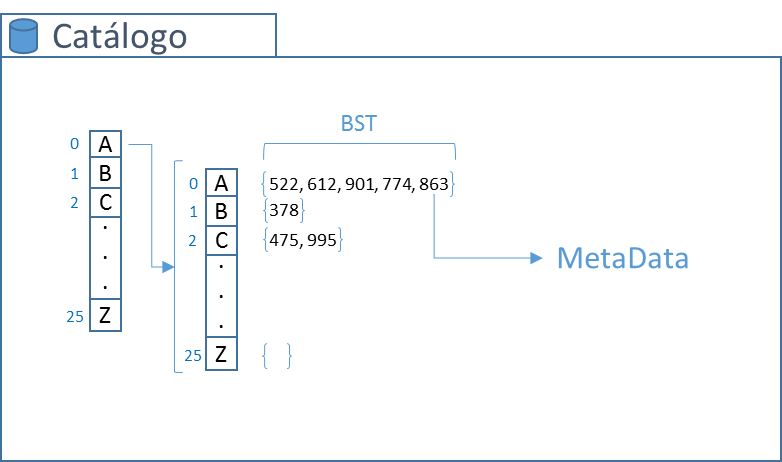


Figura 4 – Estrutura de Catálogos

#### Estrutura de Dados

O formato utilizado pelos catálogos de produtos/clientes trata-se de uma matriz de 26x26, os prefixos, que representam os dois caracteres que precedem os dígitos dos códigos no catálogo, a que cada elemento corresponde uma *BST* cujas chaves são os respectivos dígitos numéricos dos produtos/clientes, são no entanto tratados como valores inteiros neste ponto.

Como tal a inserção de um código nesta estrutura traduz-se em 2 simples operações, começando por extrair os caracteres nas primeiras duas posições do código, estes são utilizados para aceder aos índices da matriz, neste ponto temos já a *BST* cujos elementos têm como chave o número que sucede os caracteres, assim sendo basta inserir o novo código na árvore, sem nenhuma ordem particular em mente, uma vez que a única travessia ordenada pedida baseia-se em obter códigos com base no primeiro caracter, que é conseguido especificando apenas o primeiro índice-caracter da matriz.

Um elemento obtido da BST contém ainda um campo de *metadata* opcional, de modo a tornar o módulo mais versátil e aberto a outros tipos de utilizações. Neste caso a aplicação GestHiper opta por utilizar este campo como uma *HashTable*, que por sua vez contém índices que mais tarde se relacionam com as estruturas de Contabilidade e Compras.

A estruturação dos catálogos desta forma reflecte-se em ganhos de performance na aplicação, expandindo até os catálogos de forma a que possam conter adicionalmente outras informações que serão úteis para estabelecer relacionamentos com outras estruturas, ou qualquer outra informação que o utilizador da biblioteca pretenda associar aos elementos.

### Contabilidade

Responsável pela resposta a perguntas quantitativas sobre Produtos em relação a Clientes, e vice-versa. A implementação de *Contabilidade* conseguida aqui consiste em estruturas que seguram a estatística recolhida do ficheiro de compras, e a sua subsequente API.

Não têm portanto o relacionamento entre clientes e produtos que se observa no módulo *Compras*, ou seja, uma entrada nestas estruturas é 100% independente de todas as outras.

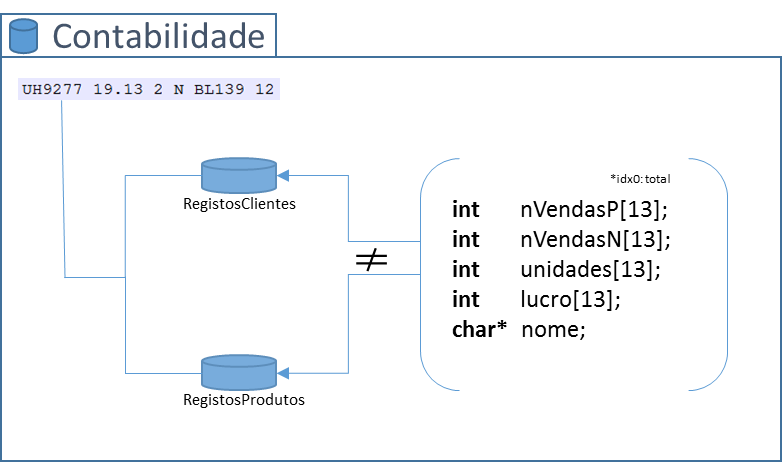


Figura 5 – Estrutura de Contabilidade

#### Estrutura de Dados

A estrutura de dados usada para este módulo é um simples *array* de registos, que contém a informação necessária à resposta eficiente as *queries* pedidas, contendo informação sobre o número de vendas em ambos os modos (P/N), unidades vendidas e lucro, separados por mês (estando no índice 0 o somatório dos seguintes campos).

Por forma a manter *performance* à medida que o número de registos aumenta, este módulo depende da *metadata* nos catálogos para garantir acessos directos aos registos, desta forma à medida que o número de registos aumenta o custo de aceder a um certo registo mantém-se constante.

A inserção de um elemento nesta estrutura inicia-se lendo uma linha do ficheiro de compras, estando definida como *Compra*, identifica-se o Produto na *Compra* e caso ainda não exista no *array* de entradas, é criado. De seguida os valores desse Produto, para o mês especificado na *Compra*, são incrementados com o valor desta. O processo é simétrico para com os Clientes.

De modo a responder a questões relativas à ordenação entre produtos (mais comprados, etc.) os elementos do *array* são ordenados de forma crescente, recorrendo a uma *minHeap*, permitindo responder a certas *queries* com custo linear incorrendo apenas num custo inicial um pouco mais elevado.

Este processo de ordenação faz uso da funcionalidade de *metadata* opcional disponível nos catálogos. Consiste em ordenar as entradas com critério baseado no somatório das unidades desse Produto/Cliente. O novo índice ordenado desta entrada é associado ao respectivo Produto/Cliente nos catálogos. Isto trata-se de uma operação relativamente intensiva, pelo que a ordenação só é executada terminada a ronda de inserções.

Esta ordenação não é no entanto obrigatória, servindo apenas para dispor as entradas de forma arbitrária. Se um utilizador da biblioteca estiver numa situação em que já tem o código de Produto/Cliente, ou seja, não está no seu interesse descobrir ‘os mais/menos comprados’, tem a mesma facilidade de acesso independentemente da estrutura estar, ou não, ordenada.

### Compras

Este módulo trata de estabelecer a tabela relacional entre Produtos e Clientes, e vice-versa, relativamente a compras.

Estes módulos fazem a gestão das listas de Clientes/Produtos válidos, assim como as perguntas sob essas listas, que não incluem questões relativas a Compras ou Contabilidades, pelo que são totalmente independentes dos restantes módulos da biblioteca.

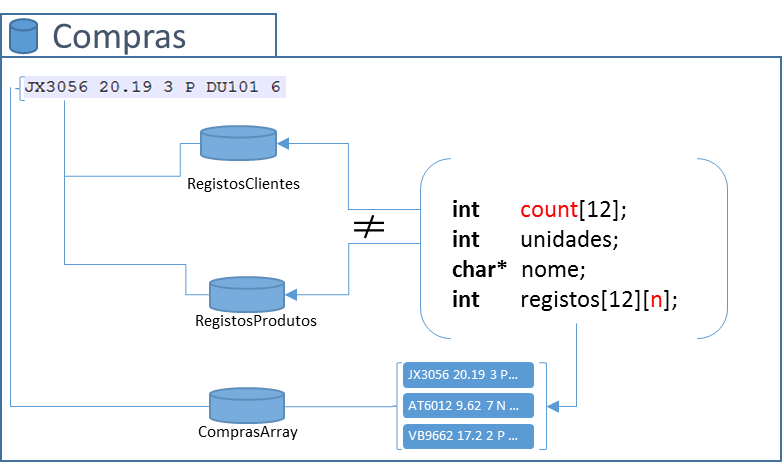


Figura 6 – Estrutura de Compras

#### Estrutura de Dados

A estrutura de dados usada é muito semelhante à utilizada no módulo de Contabilidade, diferindo no conteúdo de cada entrada, e no separar dos registos de clientes e produtos em diferentes *arrays*, sendo também ordenado da mesma forma que os registos na Contabilidade, de forma a responder a *queries* com as mesmas características.

As entradas neste módulo contém agora 12 *arrays* de registos, em que cada entrada aponta para uma compra realizada no respectivo mês. Foi feita a decisão de manter intactos os dados pertinentes a cada venda por forma a permitir extensibilidade apesar de um maior uso de memoria, permitindo assim, se necessário, adaptar o modulo a novas necessidades.

# Modelação

Para a etapa final do projeto era pedido que o grupo elaborasse o levantamento de requisitos, assim como a modelação e descrição de Use Cases, Diagrama de Classes e modelo de Base de dados do sistema apresentado no capítulo anterior. Junto a isto, deverá ser entregue também uma representação da Interface com o utilizador. De entre os vários modelos existentes, o grupo selecionou:

* O Modelo de Domínio para a representação do universo em que o problema está inserido;
* Os Diagramas de Use Cases para exposição das funcionalidades presentes na aplicação e as respetivas descrições textuais sobre as interações no sistema;
* O Diagrama de Classes, parte integral da representação estrutural do programa, que mostra as classes a utilizar na implementação da aplicação;
* Diagramas de Sequência de Sistema, que mostram a interação entre o utilizador e o sistema de uma forma simples e intuitiva (em relação à descrição textual);

Diagrama de Sequência com Subsistemas, desenvolvido em paralelo com o Diagrama de Classes, que visa mostrar a interação entre as várias estruturas (objetos) do programa, sendo então uma melhor especificação técnica dos Diagramas de Sequência.

Em seguida serão apresentados cada um destes modelos e diagramas.

### Modelo de Domínio

O modelo de Domínio é um modelo conceptual de todos os tópicos relacionados com um determinado problema. O seu objetivo é descrever as várias entidades (e respetivos atributos), relações e restrições que governam o domínio do problema. Este diagrama não descreve soluções para o problema em questão, no entanto, é usado para representar o vocabulário e conceitos chave do problema. Este modelo fornece uma primeira visão estrutural do processo de modelação, que pode então ser complementado com outras visões, como a do Diagrama de Use Cases. Após a elaboração deste modelo, o grupo chegou ao seguinte resultado:

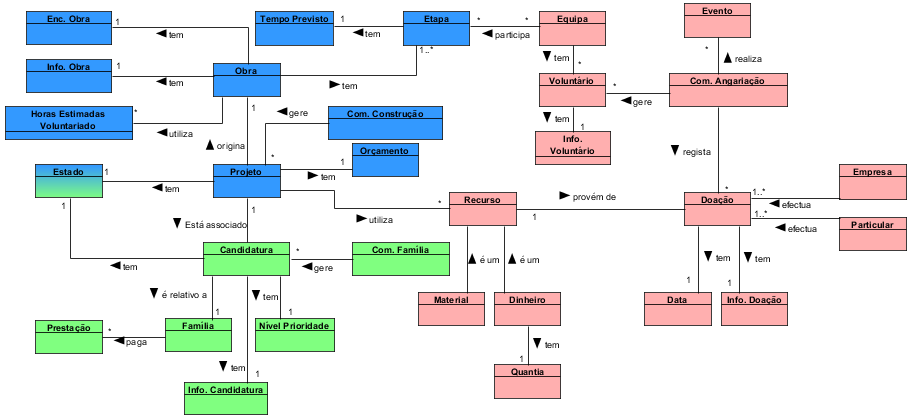


Figura 7 - Modelo de Domínio

No modelo acima é percetível a grande quantidade de informação envolvida no problema a tratar. Com base neste modelo o grupo pretende definir quais as entidades que devem ser promovidas a classes e quais devem ser apenas atributos. Tal definição irá resultar futuramente no diagrama de classes, no qual serão tomadas algumas decisões estruturais importantes.

### Diagrama de *Use Cases*

Um Diagrama de Use Cases, na sua forma mais simples, é a representação da interação de um utilizador com o sistema. Estes diagramas podem mostrar a interação de vários utilizadores distintos. Os diagramas de Use Case são geralmente acompanhados por outros tipos de diagrama, como os Diagramas de Sequência. É comum (ou mesmo complementar) escrever uma descrição textual (ou tabular), que visa descrever a interação entre utilizador e sistema de forma mais clara. A aplicação destes diagramas resultou na seguinte figura:

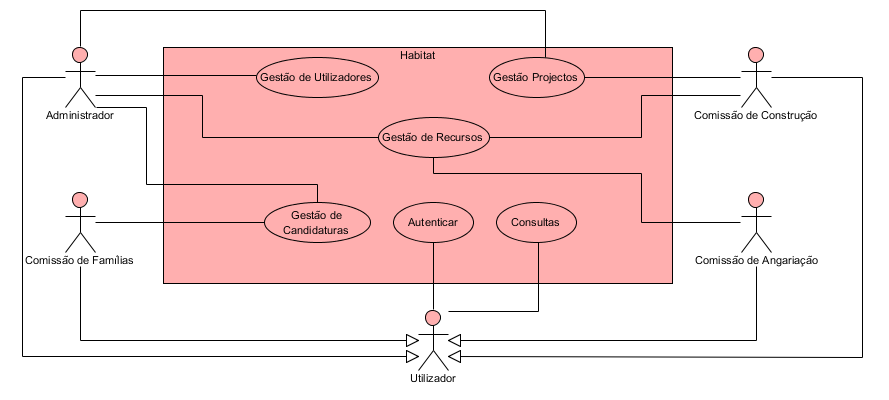


Figura 8 - Diagrama de use Cases Principal

Na figura estão representados os tipos de atores que o sistema reconhece e a divisão das funcionalidades do sistema em grupos. Nas secções que seguem vão ser descritos perfis de utilização de cada um dos atores, bem como algumas das funcionalidades a que têm acesso.

#### Utilizador

De forma a simplificar a leitura do diagrama, decidimos agrupar as funcionalidades acessíveis por qualquer ator (Administrador, Com. Famílias, Com. Angariação e Com. Fundos), dando-lhe o nome de Utilizador.

##### Autenticar

Para a principal funcionalidade prevista no sistema para o Utilizador temos “Autenticar”. Esta funcionalidade permite ao utilizador autenticar-se no sistema e, consequentemente passar a ser reconhecido como um dos tipos de ator. A interação para esta funcionalidade segue os seguintes passos:

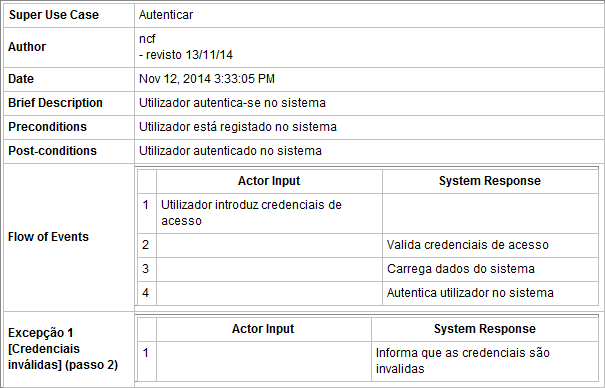


Figura 9 - Descrição Textual Do Use Case Autenticar

Após a realização bem-sucedida dos passos representados, o utilizador passará a ser reconhecido no sistema como representante da comissão a que pertence, ou como Administrador do sistema).

#### Administrador

Para garantir o correto funcionamento do sistema, é necessário criar um tipo de utilizador capaz de realizar todas as tarefas. Para esse efeito, o grupo optou por ter um perfil denominado Administrador (ou Admin), a função principal deste perfil é a criação de novos utilizadores, bem como a manutenção do sistema.

#### Comissão de Famílias

Para que seja possível a criação de um projeto, a sua candidatura tem que ser analisada e inserida no sistema. Esta inserção e análise é feita pela Comissão de Famílias, que tem, por sua vez, não só o objetivo de inserir candidaturas e atribuir-lhes um estado (que determina o progresso aliado a essa candidatura) mas também o de alterar candidaturas existentes.

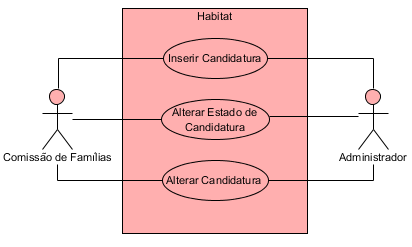


Figura 10 - Diagrama De Use Cases Gestão De Candidaturas

#### Comissão de Angariação

Por forma a conseguir o avanço de um projeto, vários fatores requerem resposta, tais como a aquisição de mão-de-obra, fundos e recursos. A entidade que dá resposta a estes requerimentos é a Comissão de Angariação que, através da organização de eventos, consegue angariar os já referidos fatores. Além de conseguir gerir eventos, a Comissão de Angariação é responsável por gerir também os Voluntários, i.e., a mão-de-obra e equipas a integrar no projeto, bem como gerir doações feitas à Habitat, dando então entrada de stock obtido com as angariações.

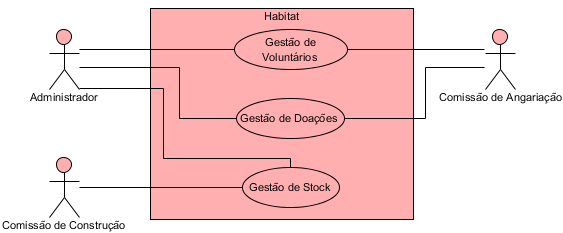


Figura 11 - Diagrama De Use Cases Gestão De Recursos

De forma a melhor agrupar os Use Cases, o grupo decidiu criar um agrupado de use cases, chamado Gestão de Recursos, onde se encaixam os 3 grupos de Use Cases vistos na imagem.

Abaixo está visível, através de imagens, o conteúdo de cada um destes grupos.

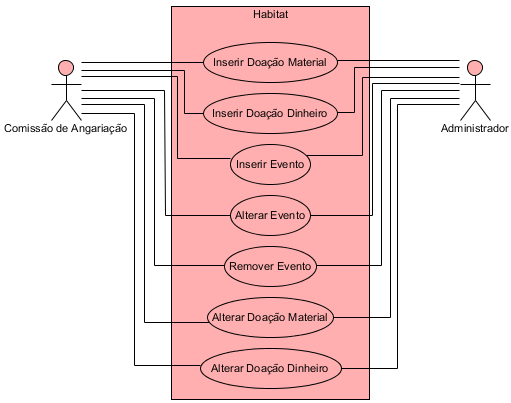


Figura 12 - Diagrama De Use Cases Gestão De Doações

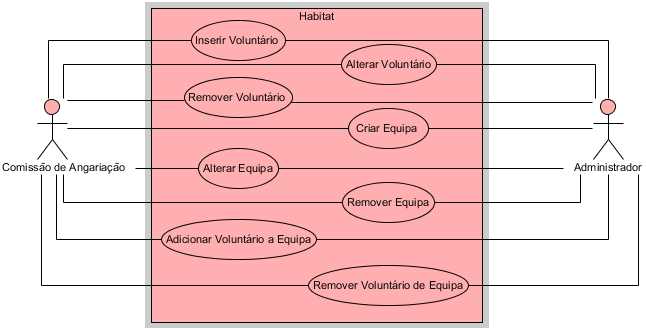


Figura 13 - Diagrama De use Cases Gestão De Voluntários

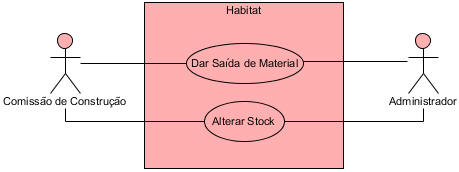


Figura 14 - Diagrama De Use Cases Gestão De Stock

#### Comissão de Construção

A Comissão de Construção é então responsável por iniciar os projetos que darão origem ao trabalho final, isto é, à habitação. Para isso, é necessário dividir um projeto em várias etapas. Etapas estas usarão materiais em stock, mão-de-obra e terão também um tempo previsto. Como ponto inicial, o grupo decidiu criar as seguintes funcionalidades:

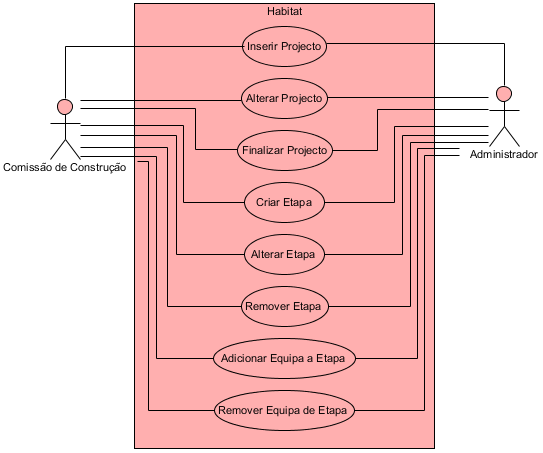


Figura 15 - Diagrama De Use Cases Gestão De Projectos

#### Descrições Textuais

De forma a explicar melhor as funcionalidades do programa, cada Use Case exposto neste relatório vem acompanhado de uma descrição tabular do diagrama em questão, de forma a mostrar de forma mais específica o comportamento da aplicação quando é utilizada essa funcionalidade. Este relatório vem acompanhado com o ficheiro de projeto (.vpp) onde constam todas as descrições dos Use Case até agora modelados.

### Diagrama de Classes

O Diagrama de Classes é o bloco principal da modelação Orientada aos Objetos. Este serve para descrever a estrutura de um sistema, mostrando as suas Classes, atributos, operações (ou métodos) e as relações entre objetos. É usado tanto para modelação conceptual da aplicação, como para transformação direta em código. No desenvolvimento de um sistema, um determinado número de classes são identificadas e agrupadas neste diagrama, o que ajuda a determinar de melhor forma as relações entre os objetos provenientes das mesmas. Com modelação mais detalhada, as classes dos diagramas mais conceptuais costumam frequentemente dividir-se em várias subclasses. Durante o desenvolvimento, o grupo chegou a uma primeira solução do diagrama de classes:

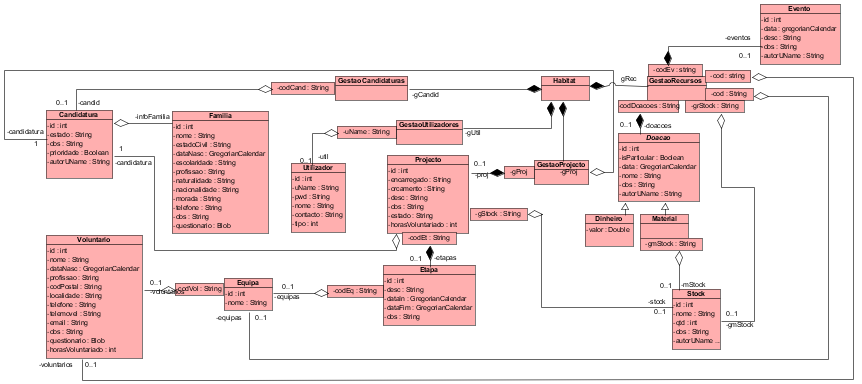


Figura 16 - Diagrama de Classes

Relembrando o diagrama de Modelo de Domínio, verificam-se algumas semelhanças na estrutura do mesmo, no entanto, este Diagrama de Classes apresenta-se mais elaborado e conciso. Este diagrama em particular provém de uma forma de modelação em que não era garantida a persistência dos dados, ou seja, ainda não foi modelada nenhuma interação com a Camada de Dados.

Encontrados estes problemas, o grupo viu necessário modificar a estrutura geral do Diagrama de Classes, de forma a garantir a persistência de dados. Após algumas reuniões, este foi o resultado:

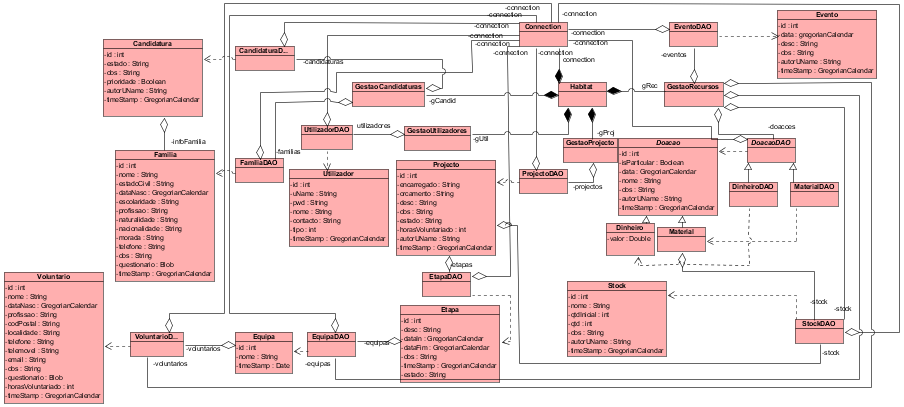


Figura 17 - Diagrama De Classes Com DAOs

Além de agora ser garantida a persistência de dados, através de classes DAO (Data Access Objects) foram corrigidos alguns problemas com o diagrama anterior, e portanto, o grupo pensa ter chegado a um estado final do diagrama. Neste momento, este diagrama pode ser transformado em código, juntamente com os métodos especificados nos vários Diagramas de Sequência com Subsistemas.

### Diagramas de Sequência

Os Diagramas de Sequência servem para mostrar como os elementos (Atores, Classes, Objetos) interagem entre si, estando organizados num cenário com uma boa perceção de tempo. Como meio de comunicação entre processos, estes usam mensagens, que podem ser caracterizadas por texto, ou num diagrama mais técnico, variáveis ou mesmo resultados de métodos. Diagramas de Sequência são normalmente associados às descrições textuais dos Use Case, e são considerados o passo seguinte no processo de Engenharia de Software baseado em UML.

#### Diagramas de Sequência de Sistema

Os Diagramas de Sequência de Sistema (DSS) são diagramas simples, que representam de forma fiel, a descrição textual dos Use Case. No entanto, devido à estrutura do Diagrama, esta transformação mostra de forma mais algorítmica, as trocas de mensagens entre o Sistema e o Ator. Como exemplo, vamos comparar a descrição textual do Use Case “Dar Saída de Material” com o respetivo diagrama de Sequência:

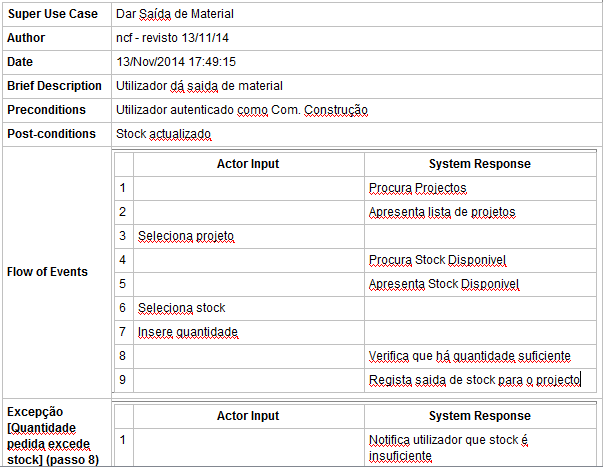


Figura 18 - Descrição Textual Do Use Case Dar Saída De Material

Como é possível ver na figura 19, existe um padrão definido de troca de mensagens durante o desenrolar do Use Case. No entanto, fica mais complicado perceber onde se encaixa o Fluxo de Exceção, e qual o comportamento a seguir caso ele seja selecionado. A transformação destadescrição em Diagrama de Sequência origina a seguinte figura:

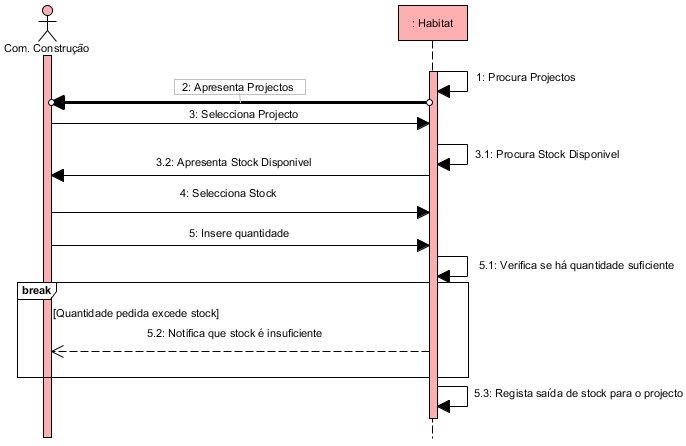


Figura 19 - Diagrama De Sequência De Sistema Dar Saída De Material

Este diagrama é a transformação direta da descrição textual. Nele é possível ter uma melhor perceção da troca de mensagens entre Ator e Sistema, bem como uma melhor noção dos fluxos de exceção e comportamentos alternativos.

#### Diagramas de Sequência com Subsistemas

Os Diagramas de Sequência com Subsistemas são uma implementação mais técnica dos Diagramas de Sequência por subdividirem (como o nome indica) o Sistema em subsistemas. Estes mostram claramente uma distinção entre a interação Ator-Sistema e a interação Sistema-Subsistemas. Estes diagramas são normalmente desenvolvidos em paralelo com o Diagrama de Classes, e podem ser reiterados para versões cada vez mais técnicas, chegando a um ponto em que é possível gerar código a partir dos mesmos.

O diagrama apresentado na Figura 15 originou o seguinte Diagrama de Sequência com Subsistemas:

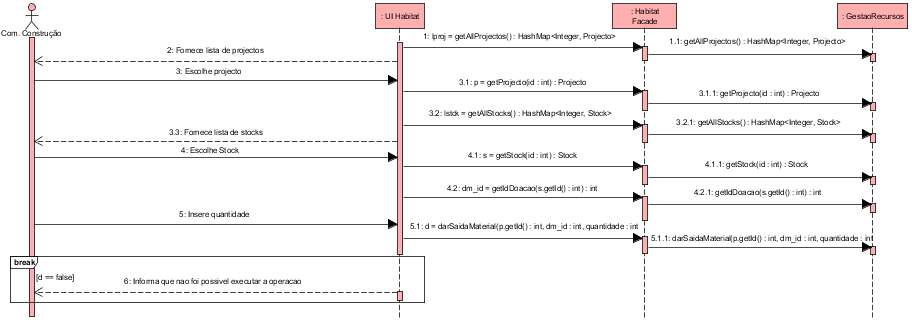
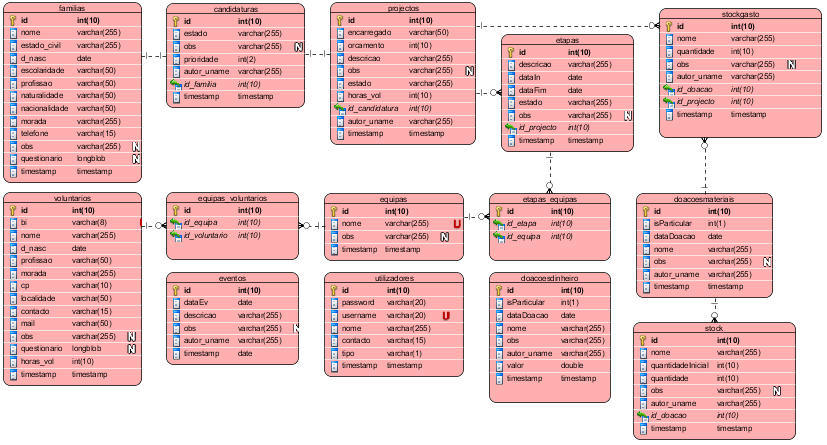


Figura 20 - Diagrama De Sequência Com Subsistemas Dar Saída De Material

Na Figura 21, pode-se verificar então que ainda está visível a interação entre o Sistema e o Utilizador, vista no Diagrama de Sequência de Sistema, com a adição da interação entre Sistema-Subsistemas. Nesta última, o conceito de mensagens já é mais restrito, sendo que estas são pedidos a funcionalidades (métodos) de outros subsistemas.

### Base de Dados

A Base de Dados de um sistema é o ponto fulcral para garantir a persistência de Informação, e como tal, é um ponto de grande relevo no processo de Modelação. Durante o processo, o grupo seguiu as regras básicas de conceção de tabelas (Formas Normais) e pensa ter conseguido relacionar as várias tabelas de forma simples e de fácil interpretação. Como resultado, surge o seguinte Modelo Conceptual:

Figura 21 - Modelo Da Base De Dados

# Dificuldades encontradas

Durante o levantamento de requisitos surgiram algumas dificuldades referentes ao arranque do projeto. Foi difícil para o grupo iniciar a caracterização do domínio da aplicação, isto é, perceber o contexto em que a Habitat estava inserido. Após alguma pesquisa e já mais elucidado em relação à organização, esta tarefa tornou-se mais fácil, e foi então possível iniciar o desenvolvimento do modelo de Domínio da aplicação. Depois de iniciado o processo de desenvolvimento, a progressão para o Diagrama de Use Cases foi suave, havendo apenas alguns pontos “esquecidos” que prontamente foram readicionados ao Modelo de Domínio da aplicação. A partir daqui, com as várias iterações do par Diagrama de Classes/Diagrama de Sequência com Subsistemas, o processo de desenvolvimento foi mais simples.

Quanto ao Interface com Utilizador, surgiram imensas dificuldades relacionadas com a tecnologia a implementar para permitir cumprir alguns dos requisitos. No entanto, o grupo pensa que escolher uma Interface com Utilizador baseada em Web foi a melhor, embora mais trabalhosa opção. Após algum esforço, o grupo pensa ter um trabalho ambicioso que cumpre os requisitos pedidos pelo cliente. Todas as dificuldades encontradas foram então superadas, algo que o grupo considera bastante positivo, aumentando assim a capacidade de trabalho em equipa.

# Conclusões e Trabalho Futuro

Após a fundamentação, o grupo pensa ter atingido os objetivos pretendidos no que respeita ao levantamento de requisitos e estruturação de um modelo de Domínio funcional e interessante. Esta etapa foi marcada por uma forte componente de comunicação entre os elementos do grupo e também através de várias reuniões para recolha e avaliação de informação (sendo que algumas contaram com a presença do representante da Habitat) sobre o modelo de Negócio a implementar.

Depois de todos os requisitos levantados, o grupo iniciou o processo de desenvolvimento da aplicação, onde foram superados alguns problemas referidos na secção anterior.

Numa análise referente à primeira entrega do trabalho realizado, é possível concluir que neste relatório estão os modelos e diagramas que o grupo pensa serem os necessários para que seja possível iniciar o processo de implementação da aplicação sem dificuldades. Neste sentido o grupo foi capaz de proceder à implementação da aplicação sem grandes problemas à exceção da tecnologia a utilizar na camada de Interface.

No que diz respeito à camada de Interface, surgiram algumas dificuldades onde foi necessário tomar algumas decisões comprometedoras. No entanto, a avaliação do grupo em relação ao produto final é positivo.

Outra conclusão retirada pelo grupo durante a aplicação do modelo de desenvolvimento em cascata, foi que este sistema é difícil de ser seguido da forma como está definido, nomeadamente no progresso sequencial de etapas, sendo necessário voltar atrás em alguns pontos para adicionar elementos que até àquele momento não foram considerados ou descobertos.

No que toca a trabalho futuro, o grupo pensa que desenvolveu um sistema capaz de ser expandido, sendo então possível adicionar novas funcionalidades, desde que estas sejam devidamente modeladas, de forma a tornar a sua implementação e integração no sistema, o mais suave possível. Com isto, o grupo conclui que o balanço final d trabalho é extremamente positivo e que além das competências associadas a todo o processo de desenvolvimento, o grupo acredita que foram também desenvolvidas algumas soft-skills, como a comunicação, trabalho em equipa e capacidade de adaptação face às dificuldades encontradas.