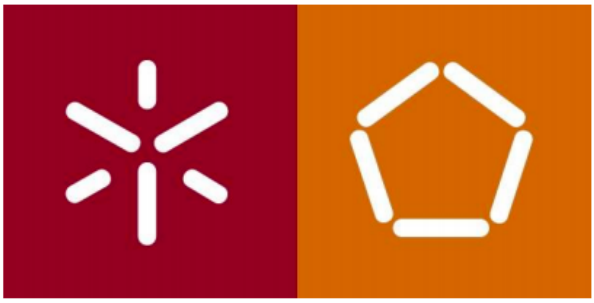


Universidade do Minho

Mestrado Integrado em Engenharia Informática

31/10/2020

Desenvolvimento de Sistemas de Software

Fase 2: Modelação conceptual da solução

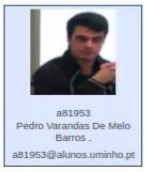
Grupo 26:


a80184
Filipe Pitrez Pereira Henriques .
a80184@alunos.uminho.pt


a82174
Nuno Miguel Freitas De Oliveira .
a82174@alunos.uminho.pt


a81761
João Duarte Vila Verde Silva .
a81761@alunos.uminho.pt


a81104
José Luís De Moura Marinho .
a81104@alunos.uminho.pt


a81953
Pedro Varandas De Melo Barros .
a81953@alunos.uminho.pt


a80872
Tiago Manuel Da Silva Rego Macedo Alves .
a80872@alunos.uminho.pt

Índice

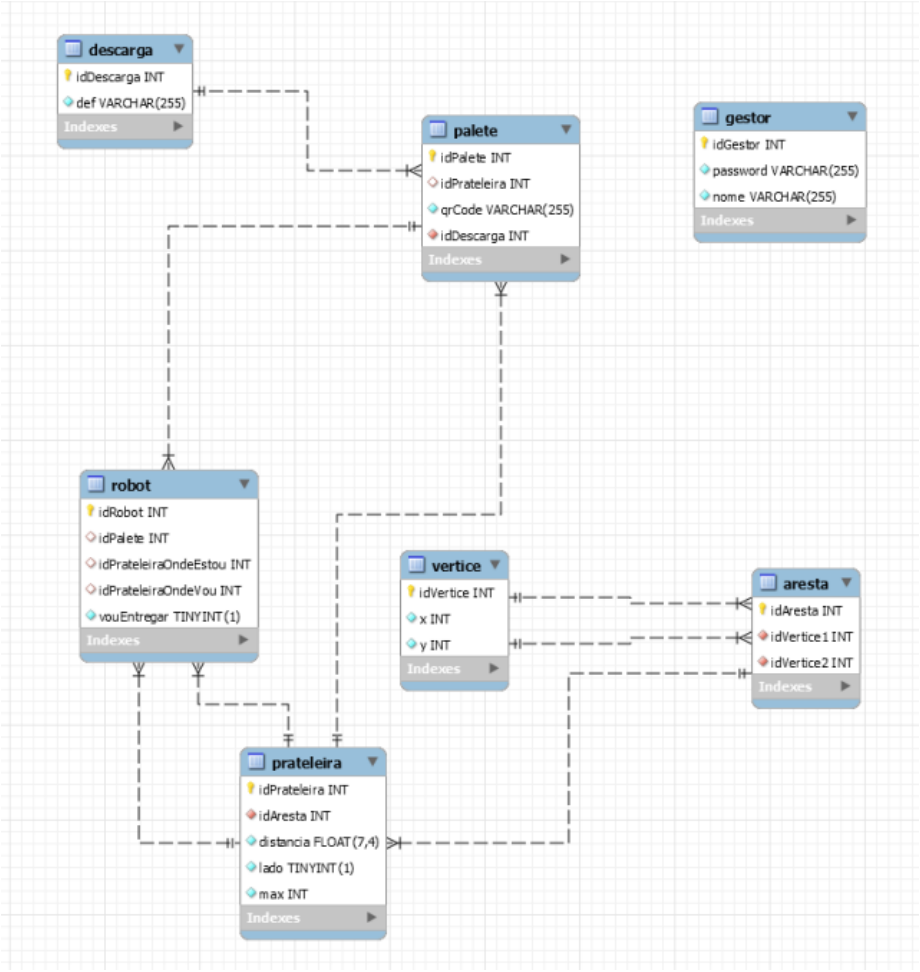
1 Introdução à 3ª fase do trabalho

Na terceira fase do trabalho começamos por ler o enunciado para percebermos quais os objetivos desta e se teríamos de acrescentar alguma coisa ao que tinhamos feito anteriormente. Analisando os Use Cases que a equipa docente marcou como necessário ter para esta fase, concluímos que já tinhamos implementado isso previamente no trabalho, com a pequena diferença de que o nosso "Sistema comunica ordem de transporte"chama-se "Pedido de recolha".

Assim, tomamámos em conta aquilo que iríamos necessitar para desenvolver a nossa aplicação, como uma base de dados, um interface, etc... e passámos à criação desta. Ao longo destas 3 fases, o processo passou pelos seguintes passos exemplificados a seguir no use case "Notificar recolha de paletes".

É necessário mencionar que cada use case é diferente e que envolveram diferentes níveis de trabalho, o exemplo a seguir foi desenvolvido em mais passos do que aqueles que são apresentados neste relatório. No entanto, achamos por bem apenas relatar aqueles que consideramos os mais importantes, de forma a tornar a idealização do nosso método de trabalho o mais simples possível.

2 Modelo lógico da base de dados

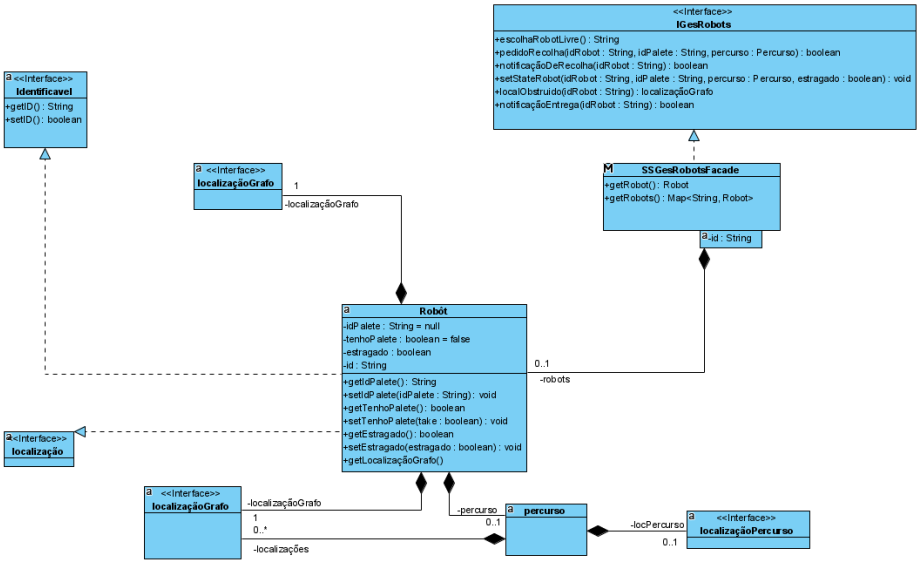


3 Escalonamento do trabalho

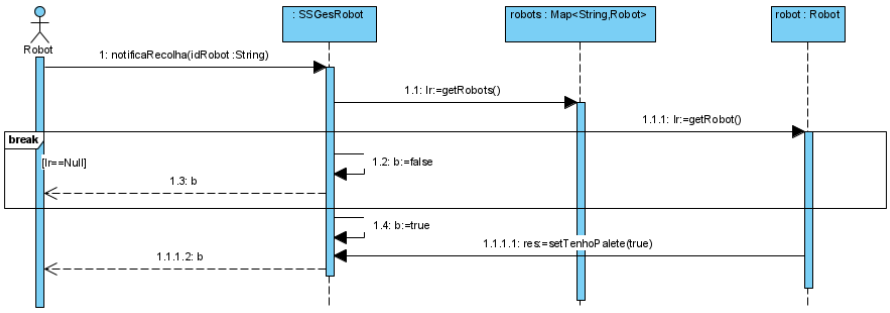
3.1 Definição do Use Case

Use case:	Notificação da Recolha	
Ator:	Robot e Encarregado	
Cenário:	1 e 2	
Pré-condição:	Robot recebe percurso e ID da palete	
Pós-condição:	Robot notifica recolha da palete e Sistema atualiza localização da palete	
Fluxo normal:	Ator input	Resposta do sistema
	1. Robot dirige-se até ao local de recolha	
	2. Robot recolhe palete com ID pretendido	
	3. Robot notifica Sistema da recolha da palete	
Fluxo alternativo 1:[Robot não consegue efetuar percurso] (passo 1)	1.1 Robot informa sistema de percurso obstruído.	
		1.2 Sistema notifica Encarregado do sucedido e guarda obstrução
		1.3 Sistema envia novo percurso
	1.4 Robot notifica sistema da receção da informação	
		...(Fluxo continua no Fluxo Normal – passo 1)
Fluxo alternativo 2:[Não existe outro percurso] (alternativa 1 – passo 1.3)		1.3.1 Sistema notifica(modo urgente) Encarregado do sucedido
	1.3.2 Robot aguarda desobstrução do percurso	
Fluxo exceção 1:[Sistema não recebe notificação da recolha no tempo esperado] (passo 3)		3.1. Sistema notifica Encarregado de possível falha
		3.2 Sistema envia sinal de paragem e tenta reboot

3.2 Atribuição de um subsistema

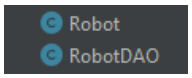


3.3 Diagrama de implementação

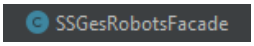


3.4 Implementação das classes necessárias

Classes de definição do ator



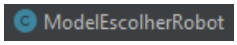
Classes de interação do ator com o sistema



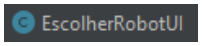
Neste caso, implementámos classes relacionadas com o percurso que o robot vai ter de percorrer e criámos ainda um algorítmo do melhor caminho.

3.5 Arquitetura MVC

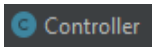
Model



View



Controller



4 Algorítmo

No contexto do problema e como forma de cálcular o caminho mais curto baseamo-nos no algorítmo de Dijkstra. Este algorítmo consiste na definição de um ponto inicial X, e vários pontos Y cuja distância é considerada o espaçamento entre X e Y.

1. Todos os nodos são maracados como "Por visitar".
2. Definimos como distancia de X o valor de 0 e infinito para os restantes.
3. Para o nodo atual, temos em consideração todos os nodos vizinhos e cálculamos as suas distâncias relativamente ao nodo atual. Comparamos o valor cálculado das distâncias selecionando o menor.
4. Após considerar todos os vizinhos por visitar do nodo atual, marcamos este como visitado, removendo-o da lista de nodos por visitar de forma a nunca mais o verificar.
5. Se o nodo de destino estiver marcado como visitado (enquato planemos uma rota entre dois nodos específicos) ou se a menor distância entre os nodos da lista de nodos por visitar for infinito (quando planeamos uma travessia completa) (ocorre quando nao existe conexão entre o nodo inicial e os restante nodos por visitar), então paramos e damos como finalizado o algorítmo.
6. Se o ponto anterior não se verificar selecionamos o nodo por visitar que esta marcado com a menor distância e passamos a considerá-lo o nodo atual e regressamos ao passo 3.

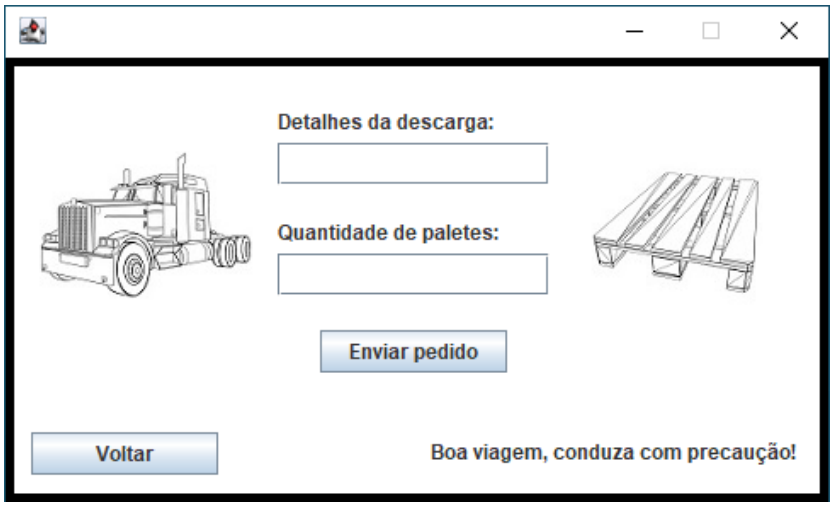
4.1 Pseudocódigo do algorítmo

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]

        remove u from Q

        for each neighbor v of u:
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

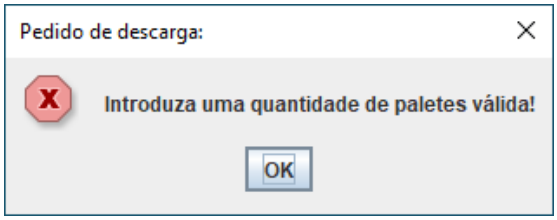
Com base nisto e aplicando-o ao contexto do nosso sitema, foi implementado um algorítmo do caminho mais curto, em Java, para otimizar o processo de recolha e entrega de paletes por parte dos robots.

5 Interface - Menu Inicial

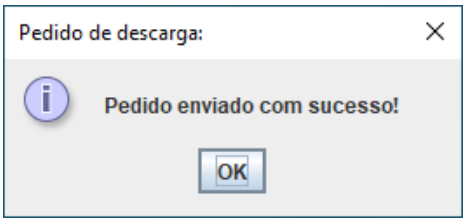
5.1 Motorista



Caso seja introduzido um número inválido de paletes:



Caso os dados sejam inseridos corretamente:

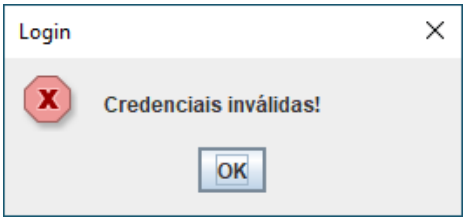


5.2 Gestor

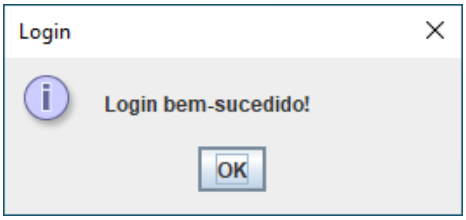
5.2.1 Login

ID/Nome -> 1 Password -> admin

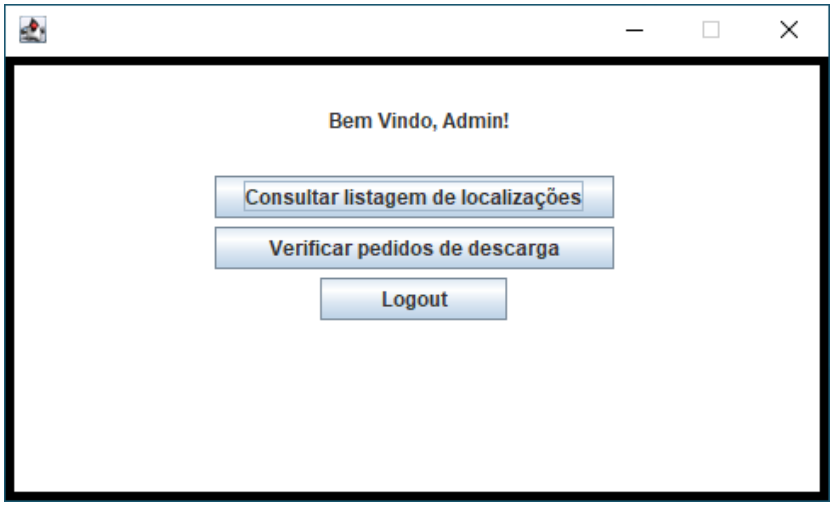
Caso sejam introduzidas credencais incorretas:



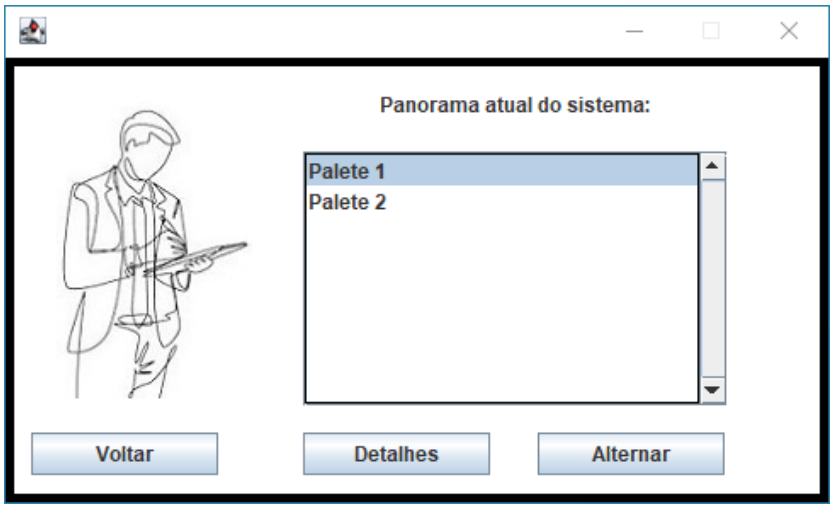
Caso o login seja bem sucedido:



5.2.2 Após o Gestor fazer login:



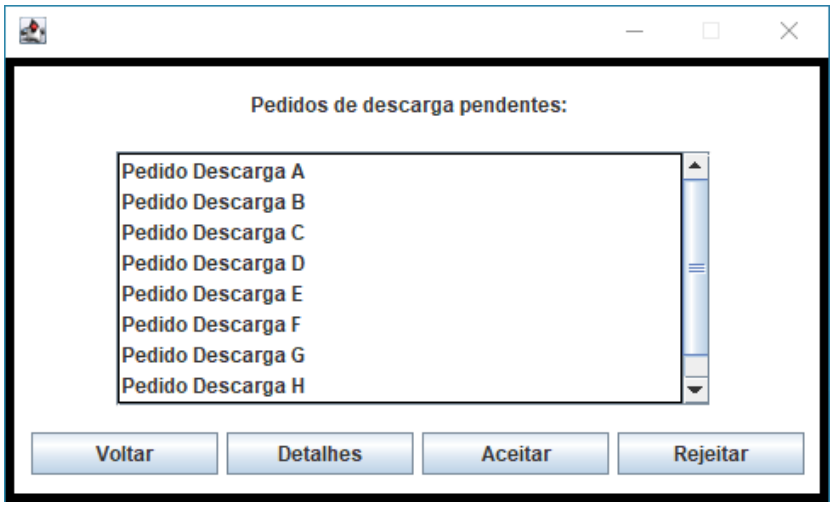
5.2.3 Listagem de localizações



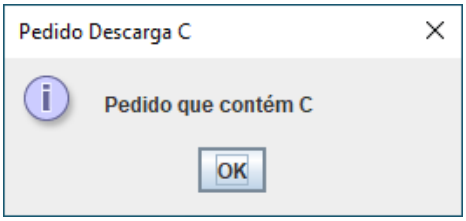
Caso se carrega no detalhe de uma palete, é-nos dito em que prateleira esta está situada ou, excecionalmente, que robot a está a transportar.

Caso se alterne a listagem de localizações, será apresentado uma lista com todas as prateleiras ocupadas no sistema. Se carregarmos nos detalhes de uma prateleira, temos acesso a todas as paletes que estão inseridas nessa prateleira.

5.2.4 Pedidos de descarga



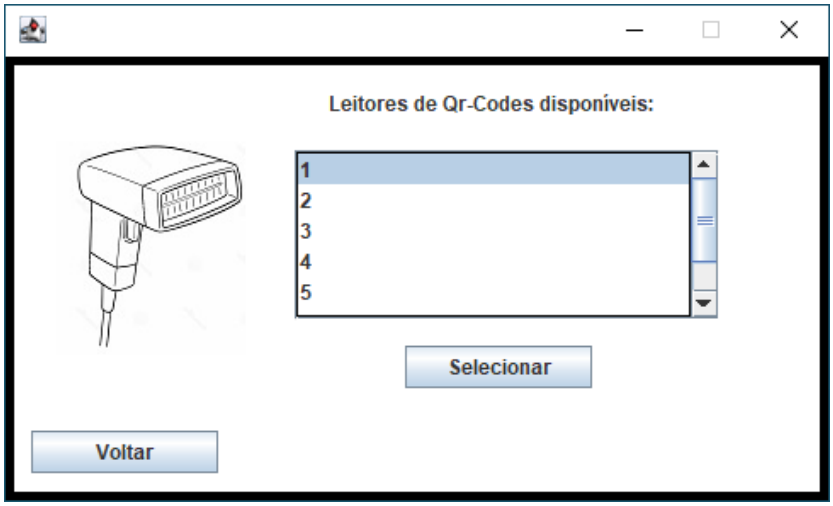
Caso se queira ver o conteúdo/detalhes do pedido de descarga:



Escolher um pedido de descarga pendente e selecionar a opção de "Aceitar"ou "Rejeitar"remove-os da lista.

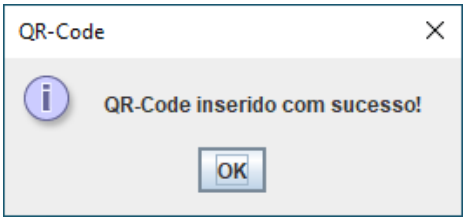
5.3 Leitor Qr-Code

5.3.1 Escolher leitor de Qr-Code

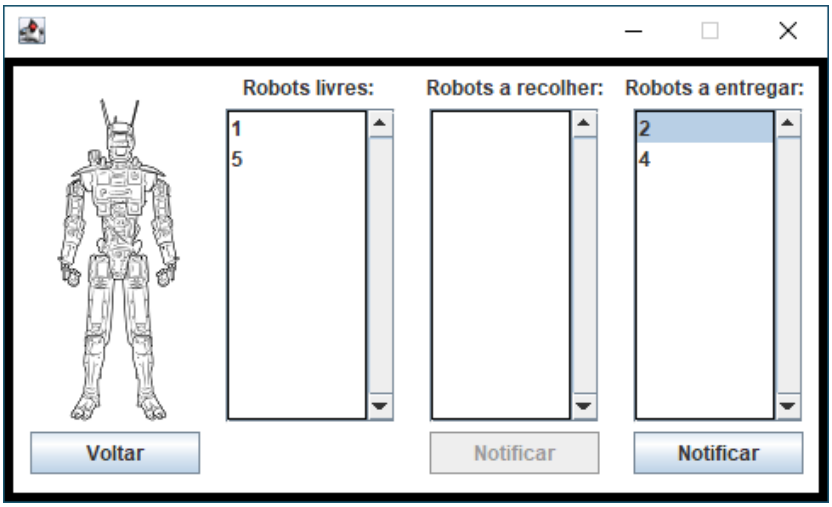


5.3.2 Introdução de Qr-Code

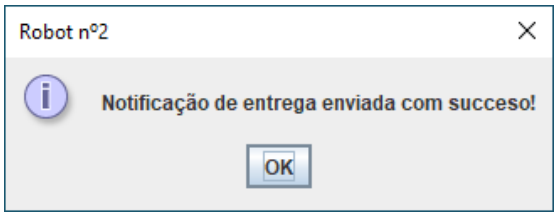
Após inserir um QR-Code:



5.4 Robot



Quando um robot notifica uma entrega:



Caso haja robots a recolher e estes notifiquem uma recolha, um pop up semelhante, só que para uma recolha, é ativado.

6 Conclusões finais

Em suma, numa fase inicial do projeto foi-nos pedido para analisar e desenvoler a ideia da nossa futura aplicação. Para tal, começamos por desenvolver o nosso modelo de domínio, entidades e relações entre elas. Esta modulação de domínio foi o que nos auxíliou na visualização do futuro e consequente planeamento do nosso projeto de modo menos abstrato. Estando ainda numa fase bastante embrionária do projeto, optamos por mantê-lo abrangente e pouco definido de modo a possibilitar possiveis alteração futuras.

Numa segunda fase, dando seguimento à primeira, optamos por uma abordagem mais discriminativa na qual começamos o desenvolvimento de todos os diagramas necessários para a antevisão do projeto. Tal processo ficou descrito mais aprofundadamente nos realatórios das etapas anteriores.

Esta terceira e ultima fase, foi dedicada à consolidação do trabalho, implementação dos diagramas e, por fim, ao planeamento e elaboração da interface de utilização do programa. Para isso, servimo-nos de um model-view-controller mais explorado no quarto tópico deste relatório.

Em forma de conclusão, consideramos que este projecto foi bastante representativo dos assuntos abordados na unidade curricular e preponderante no que toca à sua consolidação. No futuro colocamos como hipótese a utilização de multithreading como forma de melhorar o nosso projeto.