Algoritmos e Estruturas de Dados

Relatório do Trabalho Prático – 1ª parte

GESTÃO DE UMA AGÊNCIA DE VIAGENS

(tela de entrada da aplicação)

100509065 João Pedro Santos Reis Leal – ei10065@fe.up.pt
100509018 Pedro José Leite da Cunha Melo Alves – ei10018@fe.up.pt
100509096 Pedro Vieira Lamares Martins – ei10096@fe.up.pt
Turma 2MIEIC6 – Grupo 2

Mestrado Integrado em Engenharia Informática e Computação
1º semestre – 2º ano
Data de Entrega: 06-11-2011

1. INTRODUÇÃO:

Este trabalho, realizado no âmbito da unidade curricular de Algoritmos e Estruturas de Dados, teve como principal propósito reputar a capacidade dos alunos em criar uma estrutura de base de dados, através de uma aplicação, que permitisse toda a sua gestão e manipulação. Este relatório, escrito como elemento de avaliação, pretende descrever todos os passos deste grupo que levaram à criação de tal aplicação.

O nosso grupo, após ponderar durante algum tempo, sujeitando os vários temas disponíveis a diversos critérios (tais como a dificuldade envolvida, o desafio inerente e/ou o interesse que daria, por exemplo), decidiu optar como 1ª opção pelo tema 8, denominado "Gestão de uma agência de viagens". Os objectivos pedidos, as sugestões de implementação presentes no enunciado e, claro, a ideia em si, foram cruciais para a decisão deste tema, que nos pareceu sempre acessível e ao mesmo tempo desafiante!

O objectivo de todo este projecto reside na criação de uma aplicação usando a linguagem C++ que permite gerir uma agência de viagens, agência esta que possui uma lista de clientes aos quais oferece serviços relacionados, tais como a divulgação de pacotes de viagem promocionais, a possibilidade de fazer o seu próprio pacote turístico, dentre outros.

A implementação decorreu em várias fases: numa fase inicial, preocupamo-nos essencialmente em conceptualizar uma solução que fosse ao pedido do enunciado e ao mesmo tempo que correspondesse às nossas expectativas, ou seja, visualizámos e criámos classes, delineamos objectivos futuros e, claro, procuramos rapidamente distribuir o trabalho de forma igual para todos os elementos do grupo. Numa fase intermédia, dedicamo-nos exclusivamente a todo o processo de implementação de código. Esta implementação, que será integralmente descrita e explicada ao longo do relatório, durou praticamente todo o tempo disponível para se fazer o projecto, e incluiu constantes remodelações ao modelo inicialmente estipulado. Por fim, já numa fase final perto da entrega, o grupo orientou-se para tentar robustecer o código, estilizar toda a aplicação, adicionar eventuais funções que permitissem uma bonificação adicional, documentar as classes e escrever o relatório.

2. CONCEPTUALIZAÇÃO DA SOLUÇÃO:

2.1 ESTRUTURA DAS CLASSES:

A ideia principal foi sempre modelar tudo fazendo uso total da propriedade de Programação Orientada a Objectos inerente ao C++, isto é, criando o máximo de classes possíveis, e trabalhar ao mais alto nível. Com isto em mente, numa fase inicial, surgiram as dúvidas habituais: que classes serão mais apropriadas? Que atributos usar? Que relações criar umas com as outras?

Numa tentativa de utilizar aquilo que aprendemos ao longo do tempo, por forma a adquirir experiência e a alargar o conhecimento, intencionámos de imediato usar na nossa estrutura de classes os conceitos de herança e polimorfismo. Assim, dado que toda a aplicação iria ser acedida por utilizadores, pensámos rapidamente em hierarquizar todos os possíveis clientes a partir de uma classe base abstracta, a que chamámos de User.

Segue-se o diagrama UML do modelo de classes concebido (com a relação que existe entre cada classe, bem como os respectivos métodos e atributos). A seguir, pretende-se dar uma descrição de cada classe, de modo a que seja possível entender toda a conceptualização, e relaciona-la com os objectivos pedidos pelo enunciado.

A partir da **classe User**, que, tal como já foi dito, serviu para guardar a informação de um utilizador genérico que iria usar a aplicação, decidiu-se derivar duas outras classes, menos ambíguas, com finalidades diferentes na utilização do programa**.**

- **Classe Admin:**

Tal como o nome sugere, havia a necessidade de criar uma classe que permitisse gerir toda a base de dados. Como um administrador iria usar a aplicação como utilizador, e na necessidade de usar conceitos de hierarquia, juntou-se o útil ao agradável e criou-se esta classe. Um administrador tem completo e total controlo sob a agência de viagens, podendo adicionar e alterar pacotes, criar clientes, atribuir-lhes determinado serviço, lançar novidades, dentre outros.

- **Classe Costumer:**

A classe Costumer pretende armazenar a informação relativa a alguém que vá utilizar a aplicação enquanto cliente, isto é, escolhendo um dos pacotes de viagens disponíveis ou tendo total liberdade para fazer o seu, bem como aproveitar outros serviços oferecidos pela agência. Ao herdar as características de um User, um Costumer tem a possibilidade de registar uma conta na agência e de assim poder fazer parte da sua carteira de clientes, tendo acesso a descontos e a anúncios de pacotes promocionais!

Sugerido pelo enunciado foi também a distinção entre dois tipos de cliente, clientes particulares e clientes comerciais, com diferentes modos de utilização e tratamento por parte da agência. Nestas condições, deu-se a necessidade de continuar a hierarquia, derivando classes que fossem de encontro a este objectivo:

- **Classe Particular:**

Esta classe, derivada de Costumer, tem o intuito de especificar um cliente particular, isto é, uma única pessoa, que fará uso individual da aplicação. Ao receber uma data de nascimento, a agência vai poder calcular preços levando em consideração a idade (por exemplo, clientes com mais de 65 anos têm um desconto sénior de 10%, enquanto que vários tipos de alojamento possuem estadias grátis para crianças tipicamente abaixo dos 14 anos).

- **Classe Commercial:**

Um cliente comercial é uma entidade não pessoal (companhia, empresa, etc.) que a agência gere de forma diferente, nomeadamente ao nível de preços (tendo um atributo que diz respeito ao número de pessoas que constituem a empresa, o que vai influenciar a decisão de um preço), e a quem apresenta pacotes exclusivos.

A criação do produto base oferecido pela agência – uma viagem – surgiu logo após a conceptualização das classes em cima referidas. A tentativa seria criar uma plataforma a que nós denominamos de "pacote", que incluía o deslocamento e o alojamento, e que fosse a principal oferta da agência, não desconsiderando a compra individual de cada um destes serviços.

Nesta perspectiva, o grupo idealizou as seguintes classes:

- **Classe Transportation**

Classe que vai armazenar a informação relativa à forma de deslocamento, optada por um utilizador. Decidimos, a fim de evitar ambiguidades desnecessárias, que este meio de transporte poderia ser um de quatro tipos: avião (plane), embarcação (boat), comboio (train) e camioneta (bus). O preço era definido, a nível interno, com base no tipo de transporte e na classe escolhida para viajar (nos casos do avião e da embarcação). A nível externo, o número e/ou a idade da pessoa em questão, entraram nos cálculos do preço, influenciando-o de várias formas.

- **Classe Accommodation**

Esta classe é responsável pela criação de locais de alojamento, que poderão ser também comprados individualmente, conforme a vontade do cliente. Um local de alojamento não é muito complexo, contendo também ele um tipo, tal como hotel, aparthotel ou hostel, e um preço, que é novamente condicionado pelo número de pessoas a hospedar.

- **Classe Trip**

Esta viagem é um produto básico da agência, oferecida sob a forma de pacote, que contém um destino, uma duração e um transporte e alojamento associados. O preço total da viagem é calculado a partir dos preços correspondentes aos serviços que dela farão parte, podendo esta ser sempre influenciada pelo cliente que a irá comprar (respondendo assim ao objectivo de criar uma forte relação entre um utilizador e uma viagem, objectivo este pedido pelo enunciado). De modo a criar viagens promocionais, o grupo pensou numa ideia simples, porém engenhosa, que basicamente consistia na criação de um atributo que iria guardar se a viagem em questão era, ou não, promocional. Assim, foi possível implementar uma função que permitisse instalar um desconto automático, de forma rápida e sem complicações.

Dado que vários elementos nesta estrutura de classes estavam dependentes de datas, decidimos criar uma simples classe que pudesse facilmente manipula-las, a que chamamos de classe **Date**, com atributos óbvios (o dia, o mês e o ano) e várias funções que permitiam operações sobre este tipo de objectos. Apesar da sua simplicidade, esta classe foi exaustivamente usada em muitas outras, dado que, como já nos tínhamos apercebido até então, durações e idades entravam em jogo bastantes vezes na gestão de viagens.

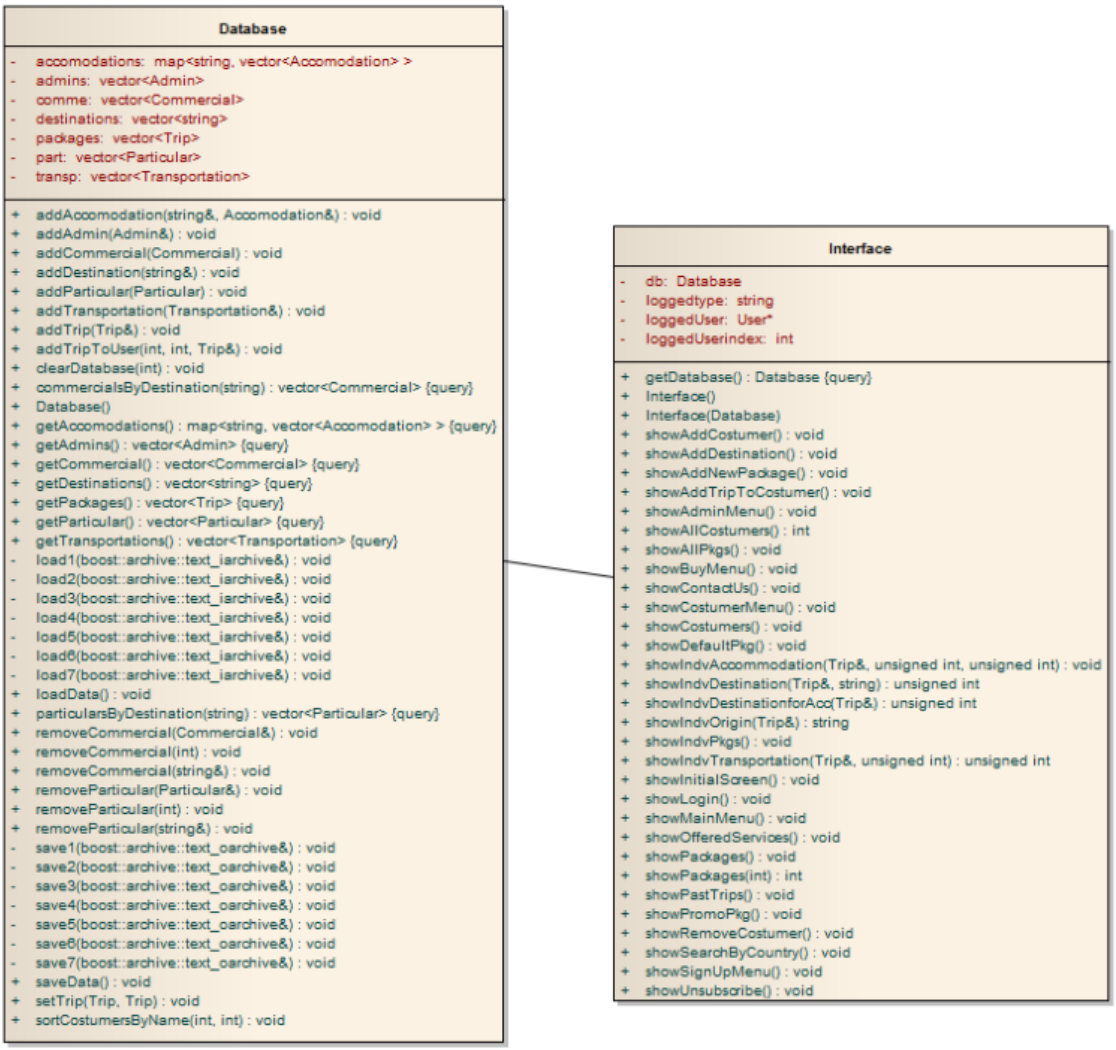
Estas classes representavam elementos majores de elevada importância, que, juntos, criavam uma agência de viagens funcional. Havia, porém, que armazenar toda esta informação de alguma forma. A criação de uma base dados foi fundamental a certo ponto:

- **Classe Database**

Classe que faz uso quase total de vectores da *STL* para armazenar toda a informação registada na agência de viagens, tais como os clientes (particulares e comerciais) registados, as viagens disponíveis, etc.

- **Classe Interface**

Esta classe, sem dúvida aquela que mais trabalho deu e dificuldades gerou ao longo de toda a implementação do projecto, é a responsável pela associação de todos os elementos da agência de viagens. Através da criação de menus, permite, tal como o nome indica, a existência de uma interface onde, através dela, se fazem todas as acções para que a agência possa efectivamente funcionar.



Para mais informações sobre as classes, tal como um detalhe mais aprofundado das funções e/ou descrição de cada atributo associado a cada classe, pede-se a consulta da documentação do projecto, gerada pelo Doxygen.

2.2 NOTAS SOBRE A IMPLEMENTAÇÃO

Ao implementar o projecto, numa tentativa constante de pôr em prática tudo aquilo que tínhamos aprendido até agora, quisemos usar todos os conceitos e todos os bons métodos de programação (tal como o já referido uso de hierarquia para modelizar toda a nossa estrutura de classes) de modo a que o nosso código pudesse parecer o mais profissional possível, nomeadamente no seguinte:

- Fizemos overload de vários operadores na classe Date para que fosse possível haver uma maneira intuitiva de fazer operações sobre estas. Além disso, fizemos também overload de algumas funções na classe Database.
- Ao criar um apontador para um objecto do tipo User, procuramos fazer uso do polimorfismo para aceder a funções virtuais.
- Criamos uma variável estática na classe User, *numOfUsers*, que conta o número total de utilizadores na aplicação (incluindo administradores).
- Fizemos uso da directiva #ifndef para evitar múltiplas criações de headers, que sobrecarregariam o projecto e pesavam na hora da compilação.
- Quisemos implementar o máximo de funcionalidade possível em classes, de modo a que o ficheiro de execução (main.cpp) não dependesse de a escrita de um grande número de linhas de código.
- Usámos algoritmos de pesquisa sequencial para trabalhar com os vectores, uma vez que, dado que sabíamos à partida que a nossa agência não iria suportar um grande número de utilizadores (ao contrário de uma aplicação que gere uma agência de viagens real), achamos que implementar métodos de pesquisa binária não seriam recomendados.
- Optamos por usar o algoritmo de ordenação de vectores disponibilizado pela *STL*, *sort*, que trabalha com iteradores e aceita uma função de ordenação simples de implementar.
- Tivemos sempre em conta o princípio do privilégio mínimo, declarando todas as funções que não alteram o objecto como constantes, por exemplo.
- Demos uma enorme importância a todo o processo de encapsulamento de código, declarando como private todos os membros-dado, implementando os gets e sets necessários à manipulação destes e excluindo de imediato a hipótese de usar o qualificador friend para declarar classes e/ou funções que não as estritamente essenciais.
- Intencionamos sempre escrever um código legível e agradável de se ler, indentando-o na totalidade, e tentando sempre reduzir o máximo de linhas possíveis, dispensando código desnecessário.

A criação de funções adicionais para efeitos de ordenação de vectores exigiu a criação de um ficheiro *header* adicional, com as declarações pretendentes, e um ficheiro *source .cpp* com as definições respectivas.

2.3 CONSIDERAÇÕES GERAIS E DIFICULDADES SENTIDAS

À medida que íamos fazendo o projecto, e implementando função atrás de função, fomos chegando à conclusão de que, afinal, não havia necessidade da inclusão de algumas, e que haviam situações em que se podia inserir a funcionalidade dentro de outras funções já existentes.

A nível de dificuldades que tenhamos encontrado (e ultrapassado), destacou-se, sem dúvida, a decisão que nós tivemos de tomar em relação ao processo que iriamos usar para gravar em ficheiro os dados armazenados na agência. Inicialmente pensamos no carregamento membro a membro em cada objecto de um vector, tal como já tínhamos usado na unidade curricular de Programação no ano passado, mas rapidamente concluímos de que isto seria martirizante e descartamos essa solução. Tentamos, então, procurar alternativa possível, e após uma breve pesquisa na Internet, encontramos uma solução que fazia uso de uma biblioteca incluída no *boost* do Eclipse, a biblioteca *boost_serialization*, que gravava informação de objectos de forma intuitiva, para isso tendo apenas que usar código exigido pela mesma.

3. CONCLUSÕES

Ao realizar este projecto, achamos que conseguimos atingir um patamar que já nos permite conceber soluções bastante capazes para lidar com os problemas que nos são propostos. Não só a construção de raiz de uma aplicação que gere uma agência de viagens, mas também todo o conhecimento adquirido ao nível da linguagem que nós usamos (c++), tratamento de excepções, hierarquia, polimorfismo e outros aspectos, a nosso ver, essenciais para o nosso futuro enquanto engenheiros informáticos. O conhecimento de um outro IDE, o envolvimento de ferramentas nunca antes usadas (tais como o Doxygen), e mesmo a escrita de um relatório que se espera mais completo e detalhado foram experiências e evoluções bastante benéficas.

Relativamente ao impacto que cada membro teve no grupo, há que realçar a preocupação que todos nós sempre tivemos em dosear as tarefas de igual maneira, de modo a que, após as saídas das notas, ninguém se sentisse injustiçado em relação a ninguém. Aliado a isso, o facto de todos nós sermos bons amigos, ajudou em todo o companheirismo e ambiente de trabalho que se fez sentir, onde a colmatação de falhas, a certificação de que não haviam dúvidas no ar e/ou de que ninguém passava um mau bocado à volta de um problema aparentemente irresolúvel sempre foram regras a respeitar após a formação do nosso grupo, por forma a garantir uma experiência de trabalho memorável e única.

A ideia sempre foi procurar a solução mais rápida, elegante e interessante para resolver problemas. Nunca desdenhamos do facto de estarmos apenas no 2º ano de curso, e de portanto o nosso conhecimento ser ainda de certa forma limitado, mas ao encararmos aquilo que foi pedido com naturalidade, humildade e dedicação, conseguimos atingir o objectivo no tempo certo e de forma extremamente satisfatória, e portanto ficamos orgulhosos com isso.

4. BIBLIOGRAFIA

[1] Serialization – Boost http://www.boost.org/doc/libs/release/libs/serialization/

[2] C++ Language Tutorial – C++ Language Documentation http://www.cplusplus.com

[3] C++ Reference [C++ Reference] http://www.cplusplus.com

ANEXO - LISTA DE ALGUNS CASOS DE UTILIZAÇÃO

Caso de utilização	Descrição
Signup no sistema	Caso inicia-se quando o utilizador escolhe a opção 1 do menu. O programa devolve um outro menu que permite o registo do utilizador enquanto cliente na agência de viagens, podendo optar entre particular ou comercial, colocando os respectivos dados de registo.
Serviços oferecidos pelo sistema	Caso inicia-se quando o utilizador escolhe a opção 2 do menu. Através desta opção, o utilizador pode ver quais os serviços oferecidos pela agência de viagens.
Login no sistema	Caso inicia-se quando o utilizador escolhe a opção 3 do menu. A partir desta opção, após fazer um login correcto e ser aceite pelo sistema, o utilizador tem acesso aos serviços visualizados na opção anterior.
Ver os clientes registados	Caso inicia-se enquanto na área de administração, ao escolher a opção 1 do menu. Permite averiguar a informação de cada utilizador da aplicação em forma de lista.
Ver os pacotes de viagem disponíveis	Caso inicia-se ao escolher a 2ª opção do menu de administração. Permite ver todos os pacotes de viagens disponíveis, incluindo os promocionais.
Associar Viagens/Clientes	Caso inicia-se ao escolher a 3ª opção do menu de administração. Permite associar pacotes ou produtos isolados (alojamento, deslocamento) a clientes registados.
Adicionar/Remover clientes	Caso inicia-se ao escolher a 4ª e a 5ª opções do menu de administração. Permite adicionar e/ou remover clientes ao sistema
Sair do Programa	Caso inicia-se ao escolher a 4ª opção do menu principal, saindo da aplicação e certificando-se de guardar os dados em ficheiro.