U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Algoritmos e Estruturas de Dados
EIC0013

Ano Letivo 2014/2015

T2 - GESTÃO DE UMA APP STORE

FEUP_AEDA1415_2MIEIC03_D

Carlos Soares - up201305514@fe.up.pt

Diogo Marques - up201305642@fe.up.pt

Fábio Carneiro - ei11029@fe.up.pt

João Santos - ei10064@fe.up.pt

7 de Novembro de 2014

Índice

Introdução ao tema

Uma *App Store* é constituída por várias *apps* desenvolvidas por *developers* externos e que podem ser compradas pelos seus clientes, sendo uma parte do valor da venda retida pela *App Store*, e o restante devolvido aos developers.

Os *developers* podem ser do tipo individual ou empresa, recebendo 80% da venda das suas apps. As *apps* são obrigadas a ter uma categoria e podem ter qualquer valor de venda, inclusive ser gratuitas (quando o preço é 0,00). Os *developers* podem ainda consultar todas as suas vendas e saldo atual.

Os clientes podem comprar *apps* e têm acesso a um histórico completo das suas compras e das aplicações que possuem. Podem ainda carregar o saldo da conta e activar códigos promocionais (*vouchers*) que lhes garante um desconto de 5% numa compra.

É possível gerir toda a informação dos developers, clientes e *apps*, guardar toda a informação presente na mesma num ficheiro e ler a informação guardada.

Solucão Implementada

A *App Store* implementada conta com quatro vectores que guardam em memória as informações mais importantes da mesma: as *apps*, os clientes, os developers e as vendas. Além destas estruturas de dados mais "permanentes", existe ainda um nome que a identifica, e o carrinho de compras.

Developers

Existem dois tipos de *developers*: *developer* individual, que recebe duas strings: uma contendo o nome do *developer* e outra contendo a morada, ou empresa, que além das duas strings referidas anteriormente (nome oficial da empresa e morada) recebe também um número inteiro com exatamente 9 dígitos (número de identificação fiscal). Quando esta condição não se verificar, é lançada uma exceção pelo construtor da classe. O *developer*, assim que faz *login* no sistema, pode publicar uma app na loja, alterar informações relativas às *apps* que publicou, bem como retirar as apps publicadas. Caso o Developer tente aceder a uma *app* que não lhe pertença, é-lhe negado o acesso e o programa apresenta uma mensagem de erro. Este pode ainda pedir a listagem das *apps* que publicou, e conhecer o lucro que apurou com a venda.

Quando um *developer* é removido da loja pelo proprietário, o programa apaga também todas as *apps* do vetor "*apps*" associadas a esse *developer*, e tal como no caso anterior, tanto o registo de vendas das apps do developer como a coleção pessoal de cada cliente mantém-se.

Sale

A classe *Sale*, relativa às vendas realizadas dentro da *App Store*, regista o cliente que realizou a compra, a *app* ou o conjunto de *apps* compradas pelo cliente, e o valor total da compra. É a única classe implementada que não pode ser manipulada diretamente pelo proprietário da *App Store*.

Clientes

Quanto à classe Cliente, o construtor recebe dois argumentos: uma *string* contendo o nome do cliente, e um *float* que representa o saldo inicial. Na classe Cliente existe ainda um vetor que regista todas as *apps* compradas pelo cliente, evitando assim que o cliente compre duas vezes a mesma *app*. O cliente pode comprar apps da loja recorrendo a um carrinho de compras, que regista as *apps* compradas e o total a pagar numa transação. Os Clientes NÃO podem pedir reembolso das *apps* compradas. O total no carrinho de compras não pode exceder o saldo corrente do Cliente – nesse caso a transação não pode ser efectuada e uma mensagem de erro é apresentada ao utilizador. Caso isso aconteça, o Cliente não precisa de recomeçar a compra, pode a qualquer momento anular a compra de uma app que tenha adicionado recentemente ao carrinho, se o seu saldo não lhe permitir a aquisição do conjunto de *apps* escolhidas.

O carrinho é reposto quando a compra é processada (assim que o cliente fizer *checkout*) e sempre que o cliente fizer *logout*, ou seja, se o cliente tiver *apps* no carrinho e não ter procedido ao *checkout* do mesmo, da próxima vez que entrar no sistema o seu carrinho de compras voltará a estar vazio e terá de repetir o processo. Isto acontece porque a *App Store* funciona através de sessões, e o carrinho de compras é, de facto, um vector "temporário" que guarda as *apps* que foram adicionadas ao carrinho pelo cliente durante a sua sessão.

No momento de *logout* de um utilizador (quer seja ele o proprietário, um *developer* ou um cliente), o programa garante que todas as informações são actualizadas nos ficheiros e que todos os atributos temporários são repostos.

O valor booleano “voucher” indica se o Cliente possui um *voucher*. Os *vouchers* são activados introduzindo um código promocional na loja (que para efeitos de teste é I-LOVE-APPS), no máximo um por cliente, e podem ser usados no máximo um em cada transação, isto é, o cliente não pode acumular os descontos de vários *vouchers* para conseguir um desconto superior numa transação, nem aplicar o desconto a apps individuais. Um voucher garante um desconto de 5% sobre o valor total da compra. No acto de compra, o programa confirma com o cliente se pretende ou não utilizar o voucher que possui naquela transação.

O cliente pode classificar e escrever um comentário na página da *app*, bem como ler os comentários deixados por outros utilizadores. O cliente pode ainda pedir listagens de *apps*, segundo vários critérios que tentam apelar ao maior número de utilizadores possível: por ordem alfabética, por categoria, por ordem crescente do preço (começando pelas grátis), por ordem decrescente da classificação (começando pelas melhores classificadas pelos outros utilizadores), etc...

Tratamento de exceções

Para o tratamento de exceções foram criadas seis classes: *AppInexistente*, *DeveloperInexistente* e *ClienteInexistente* (invocadas, por exemplo, quando o proprietário tenta apagar um utilizador que não existe, ou quando um utilizador tenta entrar com uma conta inválida); *JaExiste* (genérica, trata de todas as exceções do mesmo tipo para as classes *App*, *Cliente* e *Developer*) - é lançada, por exemplo, quando o proprietário tenta adicionar uma *app* quando uma com o mesmo nome já existe; por fim as exceções de entrada/saída de dados: *FileIOException* e *InvalidParameter* (sendo esta última a mais frequente).

"O proprietário"

O proprietário da *App Store* tem acesso às habituais operações de manutenção, podendo criar, alterar e apagar *apps*, clientes e developers. É também quem tem controlo sobre as vendas, podendo pedir listagens das vendas por volume (número de apps adquiridas em cada transação), montante e cliente.

O proprietário da *App Store* pode ainda distribuir *vouchers* de forma gratuita aos seus clientes, em ocasiões especiais. Neste caso, apenas 10% dos clientes, escolhidos aleatoriamente e que não possuam um *voucher*, recebem um e um só (visto que o máximo por pessoa é um). É também mostrada uma mensagem ao proprietário que indica quais os clientes que receberam o *voucher* (este pode consultar em qualquer altura os clientes que receberam *vouchers* ou que ativaram o código promocional).

Há duas formas de lançar um *app* na loja: através do proprietário e através do *developer*. Quando o proprietário da *App Store* cria uma *app*, além do nome é também pedido para associar a *app* a um developer (obrigatório). Quando um *developer*, após ter feito login no sistema, adiciona uma *app*, esta fica automaticamente associada a ele. Quando uma *app* é removida da loja, o programa apaga apenas a ocorrência dessa *app* dentro do vector “apps”, isto é, a *app* deixa de estar disponível para compra, no entanto o registo de vendas dessa *app* mantém-se, e esta continua na posse dos clientes que a tivessem comprado anteriormente..

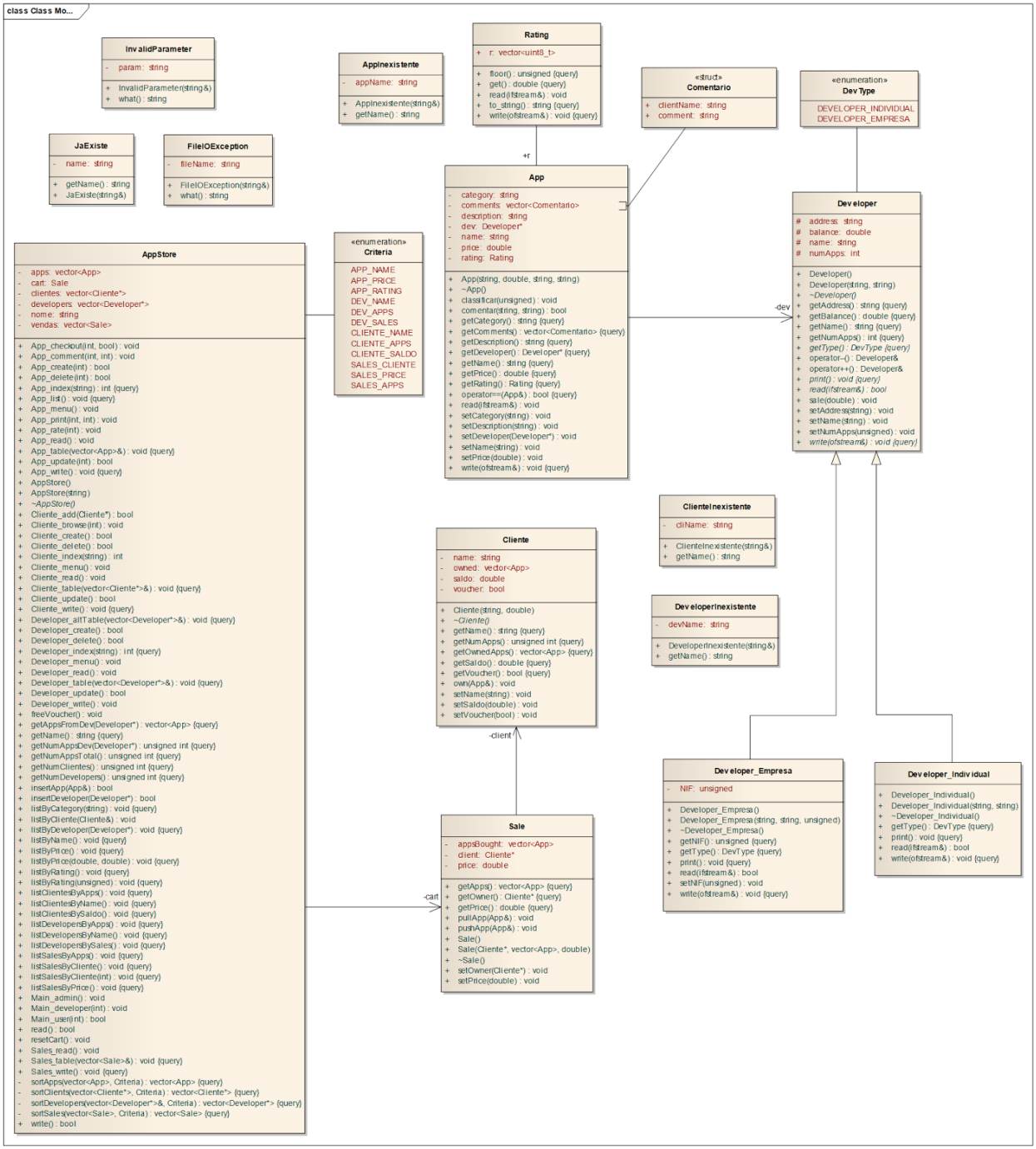
Estrutura dos ficheiros

A informação das aplicações é guardada no ficheiro “Apps.bin”, e tem a seguinte estrutura: nome, developer, categoria, descrição, preço, classificação, comentários. As classificações são constituídas pelo número de elementos do vetor e pelo vetor das classificações em si, enquanto que cada entrada do vetor de comentários apresenta o nome do cliente e o comentário.

Existe também um ficheiro que guarda os developers, "Developers.bin", igualmente em formato binário, com a seguinte estrutura: um byte no início de cada entrada que identifica o tipo de *developer* (0 para individual ou 1 para empresa), o nome do *developer*/empresa, a morada, no caso de ser uma empresa o número de identificação fiscal, e a quantia total que recebeu com a venda das *app*.

Para os clientes, a estrutura é a seguinte: nome do cliente, saldo actual, número de *apps* que comprou, e uma lista com os nomes das apps compradas. Toda esta informação é guardada no ficheiro "Clientes.bin". Com uma estrutura semelhante ao anterior temos ainda o ficheiro "Vendas.bin", que regista todas as transações efectuadas por todos os clientes da loja. O ficheiro encontra-se organizado da seguinte forma: nome do cliente, valor total da compra, número de *apps* que comprou, e uma lista com os nomes das *apps* compradas.

Diagrama UML



Casos de utilização

Um cliente pode:

- Listar apps disponívies na App Store por ordem alfabética
- Listar apps disponívies na App Store publicadas por um determinado developer
- Listar apps por ordem decrescente da classificação (uma espécie de "*best of*")
- Listar apps por ordem crescente do preço
- Comprar uma ou várias apps, tendo acesso a um carrinho de compras
- Adicionar e retirar apps do carrinho
- Fazer checkout do carrinho e processar a transação
- Comentar e/ou classificar uma app
- Visualizar informação sobre as apps, inclusive comentários feitos por outros utilizadores
- Visualizar todas as transações que realizou
- Visualizar todas as apps que comprou
- Visualizar o saldo actual
- Introduzir um código de desconto para obter um voucher

Um developer pode:

- Publicar uma nova app na App Store
- Alterar informações relativas a uma app publicada
- Remover uma app sua existente na App Store
- Obter informações sobre o lucro total das vendas e o número de apps publicadas
- Listar as suas apps por ordem alfabética

O proprietário da App Store pode ainda:

- Adicionar, modificar e apagar apps da App Store (operações de manutenção)
- Adicionar, modificar e apagar developers da App Store (operaçoes de manutenção)
- Adicionar, modificar e apagar clientes da App Store (operações de manutenção)
- Associar developers às apps
- Atribuir vouchers aleatoriamente aos clientes, numa espécie de sorteio
- Listar clientes por ordem alfabética do nome
- Listar clientes por ordem decrescente do número de apps e saldo
- Listar developers por ordem alfabética
- Listar developers por ordem decrescente do número de apps publicadas na App Store
- Listar developers por ordem decrescente do lucro em vendas de apps
- Listar vendas por cliente, por volume (número de apps/transação) e montante
- Visualizar informação sobre o número de apps existentes na loja
- Visualizar informação sobre o número de clientes e developers inscritos

Dificuldades

No entender do grupo e devido à entrada de elementos já depois de o tema estar atribuído, a principal dificuldade foi a gestão do tempo e atribuição de tarefas, dado que nos foi causando alguma inquietação. No entanto tudo deu para o certo.

Em relação à implementação do código, algumas das principais dificuldades foram: a correção de erros de compilação asociados à dependencia mútua das classes App <=> Cliente <=> Developer, o uso adequado de "cin" versus "getline" no processamento de entrada do utilizador, e em determinar o tipo de dados com os quais estávamos a trabalhar em classes Polifórmicas. Conseguimos ultrapassar esse obstáculo criando um método virtual na classe Developer para retornar o tipo de developer para o qual o elemento do vector apontava, e recorrendo a um *dynamic_cast* para invocar de forma segura o método *setNIF()* da classe *Developer_Empresa.*

Gostariamos de ter ido mais além, sendo que ideias não nos faltou. Um dos nossos objectivos era construir um programa para gerir múltiplas App Stores, o qual infelizmente não conseguimos concretizar.

Distribuição das tarefas pelos elementos do grupo

As tarefas relacionadas com o nosso trabalho começaram por ser divididas, inicialmente, por 3 elementos do grupo quer com reuniões quer com encontros online. Com a adição atrasada de um novo elemento ao grupo, as tarefas restantes tiveram que ser redistribuídas perto do final do prazo de entrega do trabalho.

No entanto, as tarefas foram distribuídas da melhor maneira (de acordo com o grupo) e cada um deu o seu melhor para alcançar as metas pré-definidas. De uma maneira geral, todos participaram ativamente sendo notável uma maior dedicação por parte de um dos elementos do grupo, Diogo Marques.

Com tudo isto, o trabalho em grupo acabou por ser uma experiencia positiva onde houve, principalmente, cooperação e entendimento entre os elementos do grupo.