Faculdade de Engenharia da Universidade do Porto

Biblioteca de Jogos

Trabalho AEDA – Parte 1 - 2018/19 - MIEIC

Coordenadores: Luís Paulo Reis e Ana Paula Rocha

Turma 2MIEIC05 Grupo 5

Professor Práticas: Luís Paulo Gonçalves dos Reis

Autores

Gonçalo Oliveira, up201705494 (up201705494@fe.up.pt)

Manuel Coutinho, up201704211 (up201704211@fe.up.pt)

Mário Mesquita, up201705723 (up201705723@fe.up.pt)

Porto, 23 de novembro de 2018

Índice

Descrição do tema do trabalho

O objetivo deste trabalho é o de criar a estrutura de dados e de gestão para **uma biblioteca virtual e multiplataforma de jogos de vídeo**, com o intuito de nos familiarizarmos com o uso de classes, herança, polimorfismo, exceções, métodos de pesquisa e ordenação, entre outros.

Esta biblioteca deveria, segundo o tema com que ficámos, ser constituída por um conjunto de **títulos** e de **utilizadores**.

Dentro dos títulos existem dois tipos: os chamados "**Home**" e os "**Online**". Estes distinguem-se pelos seguintes fatores: os primeiros vão tendo "**Updates**", cujo preço deverá ser cobrado aos utilizadores no ato da atualização, mantendo informação sobre a data de todas as atualizações em que cada utilizador fez download do título; os segundos possuem uma **subscrição** que indica o preço a pagar pelo serviço, guardando também dados sobre os utilizadores que possuem o próprio título, o total de horas jogadas e um historial de sessões de jogo.

Há ainda dois tipos de **subscrição** para os títulos online: as **fixas**, em que os jogadores pagam um certo valor por cada sessão, e as **dinâmicas**, em que os jogadores pagam por hora de uso.

Cada utilizador possui a sua **própria biblioteca**, sendo possível **comprar e adicionar títulos** se tiver fundos suficientes. O custo dos jogos poderá variar em alturas de promoções, pelo que estes guardam um historial de preços.

A biblioteca de jogos deverá ser capaz de construir várias **listagens e rankings** relativas aos seus utilizadores e títulos, nomeadamente sobre popularidade, rentabilidade, número de títulos por biblioteca, plataformas preferidas, custo de construção das bibliotecas e hábitos de consumo.

Solução Implementada

GameLibrary

De modo a gerir o sistema da Biblioteca de Jogos foi criada uma classe "**GameLibrary**", responsável por manter toda a informação sobre os utilizadores e títulos, coordenar as interações entre estas duas entidades e construir as listagens e rankings baseadas nos dados que possui. Todos estes dados estão organizados em estruturas de dados adequadas, promovendo a eficiência da pesquisa de informação.

Esta classe possui a seguinte informação:

- O conjunto de Títulos pertencentes à Biblioteca, guardados num set, de modo a se manterem organizados (por ID) e prevenir duplicados;
- Um map de Utilizadores para set de Títulos, mantendo de forma organizada o conjunto de todos os utilizadores (por ID) e respetivas bibliotecas de jogos;
- Um map que guarda para cada Título o respetivo rendimento total.

Além de permitir adicionar / remover títulos e utilizadores, a GameLibrary possui como funcionalidades:

- Gravar toda a informação do sistema num ficheiro;
- Carregar a informação de uma GameLibrary a partir de um ficheiro;
- Atualizar um título "Home", adicionando-lhe uma nova atualização caso seja mais recente;
- Determinar o tempo total gasto por um utilizador num determinado título "Online";
- Determinar o número médio de títulos por utilizador;
- Determinar a plataforma preferida de um utilizador;
- Determinar o custo de construção da biblioteca de um utilizador, incluindo o preço de jogos adquiridos, atualizações realizadas e subscrições pagas;
- Determinar o custo médio de construção das bibliotecas dos utilizadores, incluindo o preço de jogos adquiridos, atualizações realizadas e subscrições pagas;
- Construir um ranking global de popularidade dos títulos, sendo possível filtrar os resultados apresentados com base em plataforma, género e intervalo de idades;
- Construir um ranking global de rendimento dos títulos, sendo possível filtrar os resultados apresentados com base em plataforma, género e intervalo de idades;
- Construir um ranking de títulos mais jogados por um utilizador, sendo possível filtrar os resultados apresentados com base em plataforma e género;
- Construir uma lista detalhada de hábitos de consumo de um utilizador, sendo possível filtrar os resultados apresentados com base em compras de títulos, atualizações e subscrições pagas.

Utilizador

O Utilizador implementado no nosso sistema é inicializado com um ID único para permitir uma distinção clara entre diferentes instâncias, sendo também guardada a data de criação da conta.

Para além da informação básica sobre a própria pessoa, possui:

- O conjunto de títulos adquiridos, sob a forma de um apontador para o respetivo set de Títulos, presente na classe GameLibrary;
- Uma lista de utilizadores amigos, guardada num set de utilizadores, de modo a se manter organizada (por ID de utilizador) e prevenir duplicados;
- Uma lista de transações efetuadas, guardando num vetor, de modo a manter a ordem temporal, objetos do tipo "Transaction" que possuem informação sobre uma transação dentro do sistema.

O utilizador pode comprar novos títulos e atualizar ou jogar os que já possui. Destaca-se que para jogar um título "Home", este deverá estar atualizado. Similarmente, para jogar um título "Online", deverá ser paga uma subscrição.

O valor das transações que poderão ser feitas, seja a compra de jogos, a atualização ou o pagamento de subscrições, é cobrado aos cartões de crédito do utilizador.

Título

Os títulos são inicializados com um ID único para permitir uma distinção clara entre diferentes instâncias. A eles está também associada uma plataforma e um género que seja válido no sistema (plataformas e géneros aceites são definidos previamente pela GameLibrary).

Podem sofrer mudanças de preço em alturas de promoção, razão pela qual guardam um historial de promoções num vetor, definidas no intervalo de datas em que o jogo fica mais barato segundo um desconto percentual.

Existem dois tipos de títulos na GameLibrary, "Home" e "Online", distintos em vários aspetos.

- ## Título "Home"

  Este tipo de título é caraterizado por possuir um vetor de "**Updates**", cada um com um determinado custo que deverá ser cobrado aos utilizadores no ato da atualização. Guardam, portanto, uma lista de atualizações do próprio título, sob a forma de um vetor para manter a ordem temporal (ordem de inserção).

  Decidimos ser possível cada atualização especificar o próprio custo, em vez de ter um valor fixo de 1 euro como sugerido na especificação do projeto.

Além dessa informação, possuem também um map entre utilizadores e o correspondente vetor com apontadores para "Updates", registando quais as atualizações efetuadas por cada utilizador, para efeitos estatísticos e contabilísticos.

- **Título "Online"**

  Os títulos online distinguem-se por guardarem informação sobre as **sessões de jogo** de cada utilizador. Através de um mapa entre Utilizador e vetor de Sessões (vetor organizado de modo a respeitar a ordem temporal de inserção), é guardada a informação sobre cada data em que o título foi jogado, por quem e por quanto tempo.

  Possuem ainda uma **subscrição**, que pode ser de dois tipos: as **fixas**, em que os jogadores pagam um certo valor por cada sessão, e as **dinâmicas**, em que os jogadores pagam por hora de uso. Esta é implementada através de uma interface responsável por determinar os custos de jogo.

De modo a permitir que a GameLibrary apenas trabalhe com apontadores para Título, não se preocupando qual o tipo exato do objeto, a classe título possui vários métodos virtuais puros que são redefinidos nas suas classes derivadas. Métodos apenas aplicáveis a títulos "Online", quando chamados em títulos "Home", lançam uma exceção indicando que foram chamados no tipo incorreto de título (e vice-versa), efetivamente criando uma maneira simples e intuitiva da GameLibrary interagir com os seus títulos.

Optou-se ainda por implementar um destrutor parcial da GameLibrary, ou seja, quando invocado, o destrutor apaga apenas a informação de todos os Users deixando guardados os jogos. Esta decisão foi tomada, pois imaginamos que num contexto real se uma Biblioteca destas deixasse de existir toda a informação sobre os seus utilizadores seria (ou pelo menos deveria ser) perdida, já os jogos e toda a estatística sobre eles continuam a existir uma vez que as empresas criadoras devem ter acesso a eles.

Interface

Para permitir o teste e uso desta solução implementou-se uma interface simples (descrita em maior detalhe no capítulo em baixo) que, por meio de uma navegação através de menus, deixa adicionar e remover Titles, Users e objetos de todas as outras classes auxiliares (adicionar e remover Sales, Updates…). É também possível fazer listagens e rankings, globais e de User, bem como as ações normais num Title (jogar, comprar, update…)

Optou-se por não adicionar a Interface e as suas funções auxiliares ao documento Doxygen, pois se considerou que o objetivo do trabalho era fazer uma espécie de base de dados que pudesse hipoteticamente ser utilizada por quem quisesse, pois já existira uma estrutura por trás. Acoplar a isto uma interface era tornar o trabalho "fechado" e redutor, uma vez que qualquer pessoa que quisesse fazer uso da mesma teria de trabalhar com as restrições por nós impostas para que pudéssemos fazer uma pequena demonstração na sala de aula (problema da biblioteca temporal-atemporal explicado nas dificuldades).

Conceitos Utilizados

- Classes
- Herança de classes
- Polimorfismo (overloading de funções, métodos virtuais, entre outros)
- Classes interface
- Membros e métodos Static
- Exceções (lançamento e tratamento de diversos tipos)
- Estruturas de dados lineares (vetores)
- Estruturas de dados não lineares (sets e maps)
- Algoritmos de pesquisa e ordenação
- Documentação do código usando Doxygen
- Leitura e escrita de dados em ficheiros

Notas:

É aconselhada fazer a leitura deste capítulo em conjunto com o visionamento do diagrama UML de classes presente neste relatório, de modo a perceber melhor as várias ligações entre entidades.

Foram implementadas várias classes "Utilitárias" de modo a simplificar a gestão e implementação de todo o sistema, as quais podem ser vistas na secção seguinte deste relatório.

Classes Implementadas

GameLibrary

A classe GameLibrary representa a Biblioteca de Jogos em si, responsável por guardar todos os títulos e utilizadores. Armazena também informação sobre o rendimento total de cada título e sobre o conjunto de títulos que cada utilizador possui. Além de gerir as interações entre entidades, permite ainda a criação de várias listagens e rankings relevantes sobre os seus jogos e utilizadores.

Date

A classe Date representa uma data no formato DD/MM/AAAA. É utilizada ao longo de todo o programa para registar diversos acontecimentos.

User

A classe User representa um utilizador na nossa Biblioteca de Jogos, capaz de adquirir, atualizar e jogar títulos. Possui, portanto, a sua própria coleção de jogos. Cada utilizador possui um ID e email únicos. Tem também uma lista de utilizadores amigos e um registo de todas as transações que efetua.

Address

A classe Address representa uma morada, utilizada para o registo de um novo utilizador. É caraterizada por um número de casa, uma rua, uma cidade e um país.

CreditCard

A classe CreditCard representa um cartão de crédito, utilizada pelos utilizadores para efetuarem transações na Biblioteca de Jogos.

Transaction

A classe Transaction representa uma transação na Biblioteca de Jogos, podendo ser uma compra de um título, uma atualização ou o pagamento de uma subscrição.

Title

A classe Title representa um título da Biblioteca de Jogos, podendo ser de dois tipos, “Home” e “Online”. Todos os títulos possuem um ID único e não são permitidos títulos “sem tipo”. A plataforma e género associados devem pertencer às plataformas e géneros aceites pela GameLibrary. Possui ainda um registo de promoções.

Sale

A classe Sale representa uma promoção associada a um título, caraterizada por um intervalo de Datas de aplicação e de um valor de desconto percentual.

HomeTitle

A classe HomeTitle representa um título do tipo “Home”, caraterizado por possuir atualizações e registar quais as versões de cada utilizador.

Update

A classe Update representa uma atualização de um título Home, caraterizada por uma data, uma descrição, uma versão e um preço.

OnlineTitle

A classe OnlineTitle representa um título do tipo "Online", caraterizado por guardar informação acerca das sessões de jogo de cada utilizador e por possuir uma subscrição.

Session

A classe Session representa uma sessão de jogo de um título "Online", caraterizada por uma data e uma duração.

Subscription

A classe Subscription representa uma subscrição de um título "Online", indicativa do preço a pagar por sessão de jogo. É uma classe interface que pode ser de dois tipos, fixa e dinâmica.

DynamicSubscription

A classe DynamicSubscription representa uma subscrição dinâmica, caraterizada por cobrar um valor que depende das horas de jogo do utilizador.

FixedSubscription

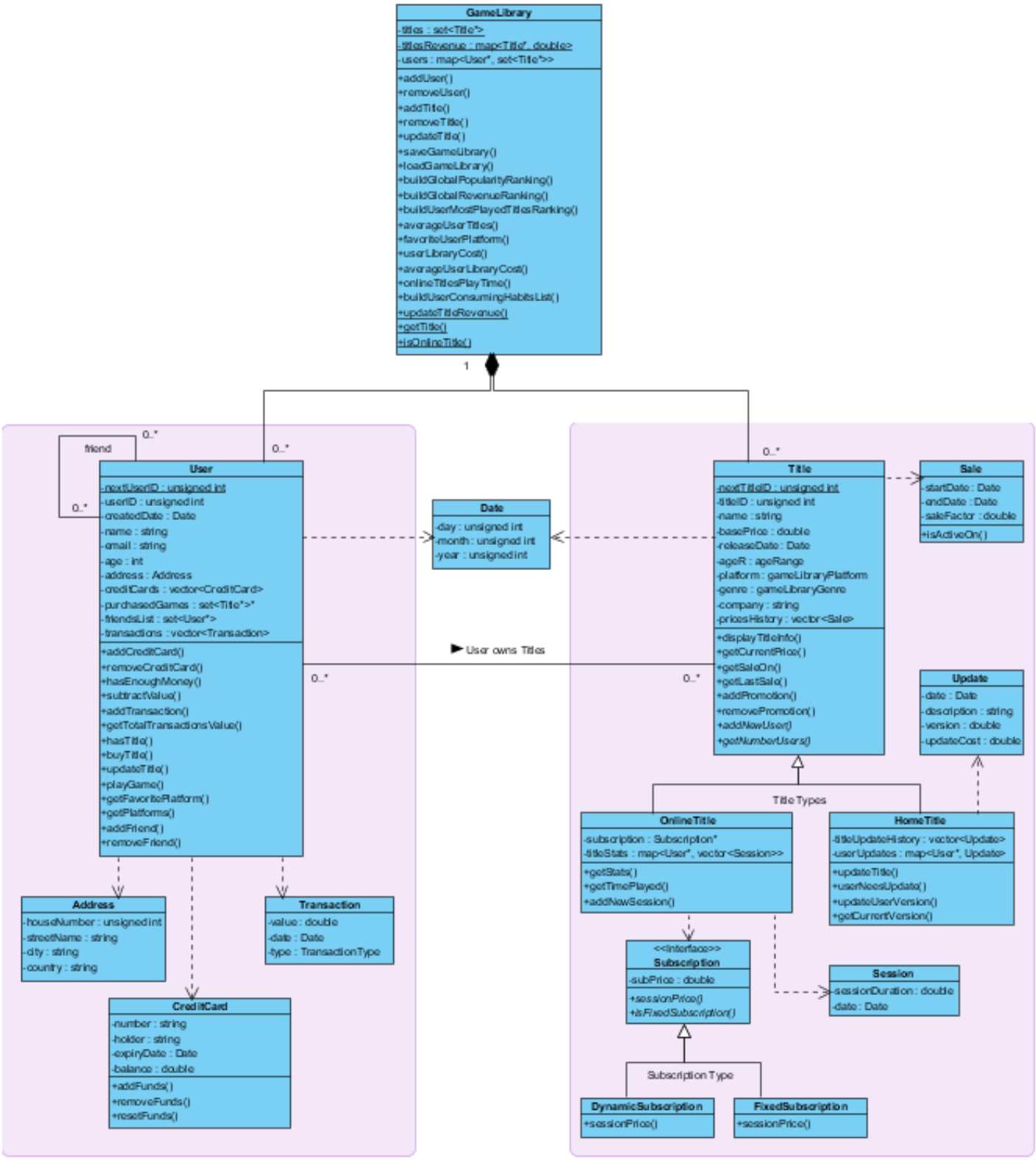
A classe FixedSubscription representa uma subscrição fixa, caraterizada por cobrar um valor fixo por sessão de jogo.

Exceções

Para o funcionamento da GameLibrary foram também implementadas várias classes representativas de exceções:

- InvalidDay: lançada quando é passado como parâmetro um dia inválido
- InvalidMonth: lançada quando é passado como parâmetro um mês inválido
- InvalidYear: lançada quando é passado como parâmetro um ano inválido
- InvalidDateFormat: lançada quando uma data foi especificada num formato inválido
- NegativeFunds: lançada quando um cartão de crédito possui fundos negativos
- NotEnoughFunds: lançada quando um user tenta comprar algo sem fundos suficientes
- InexistentUser: lançada ao aceder aos dados de um utilizador num título que não possui
- DuplicatedUser: lançada ao tentar registar um utilizador num título que já possui
- NotHomeTitle: lançada ao tentar usar métodos de títulos "Online" em títulos "Home"
- NotOnlineTitle: lançada ao tentar usar métodos de títulos "Home" em títulos "Online"
- InexistentSale: lançada ao tentar aceder a uma promoção numa data em que não existe
- OverlappingSales: lançada quando existem promoções para um título na mesma data
- ExpiredSale: lançada quando se tenta usar uma promoção que já expirou
- SaleStarted: lançada quando se tenta remover uma promoção que já começou
- OldUpdate: lançada quando se tenta atualizar um título para uma versão anterior
- TitleUpToDate: lançada ao tentar atualizar uma versão de utilizador já atualizada

Diagrama UML de classes



Nota: Foram implementadas várias exceções, especificadas na secção acima, que não constam no diagrama UML, de modo a não o sobrecarregar demasiado. Com o mesmo objetivo em mente, apesar da classe Data ser usada por muitas outras classes, apenas está indicada no diagrama como dependência de User e Title.

Casos de Utilização

Quando se inicia o programa, o utilizador (diferente de User, o primeiro é o criador da biblioteca que está a interagir com a interface, enquanto o segundo é um elemento da própria biblioteca) depara-se com um menu que lhe permite escolher de entre 4 opções: criar uma biblioteca de raiz, continuar uma biblioteca previamente criada, guardar a presente GameLibrary e sair da interface (opção 0 – há uma opção semelhante em cada menu que permite voltar para o imediatamente anterior, por isso, daqui para a frente será omitida).

Daqui seguimos para o **Menu Principal** onde há 3 opções principais: Manage Games, que permitirá fazer todas as operações relacionadas com eles (desde adicionar a remover, passando por listar atributos e adicionar Sales entre outros); Manage Users que, à semelhança do anterior, nos leva a um menu que possibilita fazer todas as ações relacionadas com um User; e, por fim, a opção Lists and Rankings que, como o próprio nome indica, deixa fazer rankings parciais e totais tendo em conta filtros definidos pelo utilizador.

Menu Jogos: É neste menu onde a adição e remoção de jogos é feita. É também possível listar um breve sumário de todos os jogos e aceder a um novo submenu por meio do titleID adaptado ao jogo. Neste **Menu Específico de Jogo**, o utilizador pode consultar as informações detalhadas de um jogo ir para o **Menu Sales** e dependendo de se tratar de um título Online ou Home ir para o **Menu Sessions** (primeiro caso) ou para o **Menu Updates.**

No **Menu Sales** podemos fazer o display de um breve sumário de cada promoção, adicionar e remover Sales (tendo parâmetros que deve respeitar: como não ser possível adicionar uma promoção passada ou que se sobreponha a outra) e ainda consultar a promoção atual.

O **Menu Sessions** é um pouco diferente dos anteriores, pois não permite adicionar ou remover Sessions (uma vez que uma vez jogado, um jogo não deve deixar de o poder ter sido, e não cabe ao jogo decidir quando se o vai jogar), mas sim uma série de listagens diferentes: listar as últimas N sessões de cada User do jogo (N este definido pelo utilizador); listar as sessões dos top 3 Users (aqueles que usufruem mais do Title em questão); e mostrar todos as sessões de todos os Users que têm o Title (pode ser muito extensivo, por isso, não é aconselhado o uso frequente).

Já o **Menu Updates** é bastante parecido com o das Sales, deixando adicionar um Update para uma data futura, consultar o último update e listar um pequeno sumário de todos os updates por ordem cronológica.

Passamos então ao segundo submenu do **Menu Principal**, o **Menu Jogadores**. Este Menu é praticamente uma cópia do **Menu Jogos**, com a possibilidade de listar um breve sumário de todos Users, de adicionar e remover Users e ainda de ir para um **Menu Específico de User,** por meio do seu endereço de e-mail que deverá ser único e num formato válido avaliado por meio de um teste de linguagem regular.

Este **Menu Específico de User** tem uma opção para fazer a listagem da informação detalhada do determinado User e 3 opções que vão dar a novos submenus: o **Menu Credit Card**, o **Menu Friends** e o **Menu User Titles.**

Os dois primeiros submenus têm as operações básicas: listar sumariamente o tipo de objetos escolhido (fazer uma lista dos Credit Cards ou dos Friends) e adicionar e remover estes elementos. O **Menu Credit Card** destaca-se pelas opções adicionais que tem para adicionar dinheiro a um dos cartões e ver a lista de transações que o User fez.

Um User pode ainda jogar, comprar e fazer o Update de um Title através do **Menu User Titles** que também deixa fazer uma listagem dos jogos que este tem e ir para um novo menu chamado **User Ranking Menu**. Este último permite mostrar os Hábitos de Consumo de um dado User, ver os Title que mais joga (aqui só entram os Online uma vez que na especificação apenas pedia para guardar o tempo de jogo destes), consultar as últimas N Sessions (este N deve ser introduzido pelo utilizador) e, ainda, mostrar outras estatísticas como a consola favorita.

Chegamos por fim ao último submenu do **Menu Principal**, o **Menu Lists & Rankings**, que permite escolher entre mostrar as estatísticas médias e fazer um ranking (total ou parcial através de filtros que devem ser escolhidos) por popularidade e por rendimento.

Principais Dificuldades

A maior dificuldade neste trabalho foi em compreender o tema que escolhemos. Tendo em conta exemplos reais de bibliotecas de jogos, alguns aspetos estavam a criar confusões para as quais não estávamos a conseguir chegar a um consenso internamente no grupo. Assim, e de modo a assegurar-nos que o nosso trabalho respeitaria o pretendido, contactamos e fomos de encontro aos monitores e professores coordenadores.

Uma das respostas dos coordenadores, levou-nos a optar por um destrutor apenas parcial: quando foi posta a dúvida de porquê apenas os jogos online guardarem sessões e porque é que deveria ser este (os Titles) a guardarem este tipo de informação, foi nos apresentado como exemplo os jogos online de hoje que guardam as informações dos jogadores (muitas vezes sem conhecimento dos mesmos) enquanto que se eu tiver a jogar Pokémon na minha Nintendo DS ninguém saberá quanto tempo foi. Isto pareceu-nos fazer sentido e foi exatamente por isso que consideramos fazer algo semelhante ao que se passa. Mesmo quando a biblioteca é apagada, apesar do registo de todos os jogadores o serem, os jogos continuam a existir e guardar alguma informação.

Esta dificuldade, que deveríamos ter resolvido logo no início, levou a que todo o desenvolvimento do trabalho se atrasasse, efetivamente criando um problema de tempo.

Ironicamente este problema temporal estendeu-se também à própria biblioteca de jogos: o facto de ser necessário fazer uma demonstração num dia específico trouxe-nos alguns problemas a nível da temporalidade da biblioteca.

Como vemos no UML e na nota que o acompanha, a classe Date acaba por ter muita preponderância no contexto da base de dados, visto que quase todas as outras classes a usam. Sabendo que a inclusão de data na maior parte dos constituintes era um requerimento do próprio tema (sessões, updates, promoções, transições entre outros deviam ter sempre associadas pelo menos uma data e nem se tratavam das classes nucleares) tivemos de fazer um compromisso entre a temporalidade da biblioteca e a sua intemporalidade para que fosse possível o seu uso num dia específico e não ao longo do tempo como é suposta ser usada uma estrutura destas.

Isto causa problemas específicos como o de adicionar uma promoção a dias que já passaram ser ilógico de um ponto de vista real e, por isso, não permitido pela base de dados, apesar de que pudesse dar jeito para criar maior quantidade de informação para uma melhor demonstração no dia. O contrário acontece com as sessões: num contexto real apenas se poderia jogar no dia, e não adicionar sessões posteriores ou anteriores, mas estas duas restrições seriam extremamente redutoras para a demonstração e, então, optou por não se colocar (só ter sessões do próprio não faria sentido para listagens como as últimas 10 sessões de cada jogador).

Passados os primeiros problemas estruturais gerais, ainda tivemos problemas com que estruturas de dados usar para cada organização informação. Vinha no relatório que vetores era um dos requerimentos, por isso, esse era evidente que teríamos de usar, mas também nos foi aconselhado por monitores a usar estruturas mais eficazes como sets e maps.

O grande debate interno do grupo foi relativamente a como guardar a informação de um User de um Title (construção frásica propositadamente forçada). Havia dois tipos de informação específica de cada Title para cada User caso o primeiro fosse Online ou Home

(traria associado as sessões ou os updates feitos), por isso, pareceu-nos evidente que teríamos de fazer proveito de certo tipo de polimorfismo, mas as adversidades estavam apenas a começar. Haver este tipo de informação que estava associada a dois lados causou um debate de redundância e eficácia e do comprometimento de uma em relação a outro. A solução que nos pareceu ideal era uma espécie de HashTable de duas chaves (inspirada em Java) que permitisse ir buscar informação com base em apenas uma. Ou seja, um espécie de função de duas entradas f(x, y) = z que permitisse ir buscar a partir de uma só chave x a soma de todos yy e vice-versa, permitindo assim especificar apenas o jogador e ir buscar todas as sessões online de todos os jogos, apenas o jogo e ir buscar todas as sessões online de todos os jogadores ou mesmo as sessões específicas de um jogador num jogo (semelhante para os Home e correspondentes updates).

Infelizmente, o mais perto que chegámos foi um map com uma chave dupla em forma de pair em que se poderia ir buscar toda a informação da primeira chave sem se saber a segunda, mas não o contrário, o que levaria a uma obrigatória redundância para que a operação inversa fosse possível.

Rendidos a uma eminente inevitabilidade de existência de informação repetida (para evitar que a biblioteca fosse muito lenta com grandes quantidades de informação) optámos por simplificar e usar a estruturação descrita em capítulos anteriores.

Relativamente à implementação e escrita de código, não foram encontradas dificuldades significativas. Após perceber o tema e estruturar a nossa abordagem, internamente conseguimos resolver qualquer adversidade que surgiu.

Salienta-se, no entanto, que para esta primeira parte do projeto houve alguma dificuldade no desenvolvimento conjunto de código, pois ao contrário do google Docs o GitHub ainda não permite um visionamento em tempo real de quem está a editar o quê tornando difícil saber se vamos estar a alterar as mesmas partes.

Esforço de Cada Membro

O trabalho foi realizado com sucesso, tendo conseguido implementar o sistema da Biblioteca de Jogos com todas as funcionalidades requeridas e ainda alguma, as que tivemos tempo, porque havia ideias para mais, extra que consideramos mais relevantes.

Todos os membros contribuíram igualmente na discussão do tema e na procura de soluções para os problemas encontrados, tendo sido realizadas reuniões de grupo presenciais para este efeito. Mantivemos um diálogo constante e ajudamo-nos mutuamente.

De modo a facilitar o nosso processo de desenvolvimento, utilizamos o sistema de controlo de versões "git". Para isso criamos um repositório no *Github,* que pode ser acedido no link disponibilizado em baixo. Para disponibilizar informação mais detalhada sobre a nossa divisão de tarefas, processo de desenvolvimento e contribuições individuais, este encontra-se agora público (anteriormente à data de entrega encontrava-se privado, de modo a não permitir qualquer plágio).

Repositório no *Github*: https://github.com/GambuzX/Games_Library

Consideramos que todos ajudaram e contribuíram com o que conseguiram para a realização deste trabalho e, apesar, de se calhar as tarefas acabarem por não ser tão equitativamente distribuídas como gostaríamos, achamos que recaí sobre todos nós essa "culpa" uma vez que fomos nós que fizemos a divisão e subvalorizamos ou sobrevalorizamos partes que acabaram por não o justificar.

As contribuições individuais de cada membro foram as seguintes:

Manuel Coutinho

- Classes implementadas:
  - Title
  - Online Title
  - Home Title
  - Date
  - Address
  - Subscription
  - Dynamic Subscripion
  - Fixed Subscription
- Classes para as quais contribuiu:
  - Sale
  - Updates
  - Session
- Contribuições adicionais:
  - Exceções relativas às classes implementadas
  - GameLibraryInfo.cpp
  - Interface e derivados (ConsoleFunctions e Inputs)
- Documentação Doxygen do projeto
- Escrita do relatório

Mário Mesquita

- Criação do diagrama UML de classes

- Classes implementadas:
  - User
  - GameLibrary
  - CreditCard
  - Transaction
  - Sale
  - Update
  - Session
- Classes para as quais contribuiu:
  - Title
  - OnlineTitle
  - HomeTitle
- Contribuições adicionais:
  - Exceções relativas às classes implementadas
  - Header GameLibraryInfo.h
- Documentação Doxygen do projeto
- Escrita do relatório

Gonçalo Oliveira

- Criação do diagrama UML de classes
- Implementação da fase inicial da interface
- Implementação de um sistema de gravação da biblioteca de jogos em ficheiros e posterior leitura.
  - Criação de operadores de inserção (operator<<) Sale, Transaction, Credit Card e funções de escrita num ostream adequadas à escrita de informação.
  - Implementação de uma máquina de estados para suportar todos os tipos de Title e informação que o User pode ter.
- Contribuições adicionais:
  - Documentação Doxygen do projeto