**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](mailto:up201705494@fe.up.pt))

Manuel Coutinho, up201704211 ([up201704211@fe.up.pt](mailto:up201704211@fe.up.pt))

Mário Mesquita, up201705723 ([up201705723@fe.up.pt](mailto:up201705723@fe.up.pt))

Porto, 23 de novembro de 2018

**Índice**

[Descrição do tema do trabalho 3](#_Toc530734435)

[Solução Implementada 4](#_Toc530734436)

[Classes Implementadas 7](#_Toc530734437)

# Descrição do tema do trabalho

O objetivo deste trabalho é de criar a estrutura de dados e de gestão para **uma biblioteca virtual e multiplataforma de jogos de vídeo**, no intuito de nos familiarizar ao 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 **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.

**SINTETIZAR ISTO ???**

# 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, 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 “**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 mapa entre utilizadores e “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.

**FALAR DA INTERFACE DA GAMES LIBRARY AQUI ?!?!?**

**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

Notinha sobre exceções não incluídas e sobre ligações a Date

# Casos de Utilização

# Principais dificuldades encontradas no desenvolvimento do trabalho

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.

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. Destaca-se ainda que um membro do nosso grupo sofreu um acidente e ficou incapacitado da sua mão direita até à última semana de entrega do projeto, reduzindo a capacidade de trabalho do grupo durante um longo período.

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 na divisão igualitária das tarefas.

# Esforço dedicado por cada membro

O trabalho foi realizado com sucesso, tendo conseguido implementar o sistema da Biblioteca de Jogos com todas as funcionalidades requeridas e ainda algumas extra que consideramos 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>

As contribuições individuais de cada membro foram as seguintes:

**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