**Faculdade de Engenharia da Universidade do Porto**



**Biblioteca de Jogos**

**Trabalho AEDA – Parte 2 - 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, 09 de janeiro de 2019

**Índice**

[**Descrição do tema do trabalho** 3](#_Toc530774202)

[**Solução Implementada** 4](#_Toc530774203)

[**Classes Implementadas** 8](#_Toc530774204)

[**Diagrama UML de classes** 10](#_Toc530774205)

[**Casos de Utilização** 11](#_Toc530774206)

[**Principais Dificuldades** 13](#_Toc530774207)

[**Esforço de Cada Membro** 15](#_Toc530774208)

# **Descrição do tema do trabalho**

O objetivo da segunda parte deste trabalho é o de acrescentar novas funcionalidades ao projeto previamente desenvolvido durante a primeira parte, a biblioteca virtual e multiplataforma de jogos de vídeo. Pretende-se que sejam usadas **estruturas de dados não lineares**, das quais árvores de pesquisa binária, filas de prioridade e tabelas de dispersão, assim como ilustrar operações básicas CRUD e listagens várias.

Em primeiro lugar, a biblioteca de jogos deve agora manter um registo de todas as **empresas** criadores de títulos que constam nela. Esta informação encontra-se guardada numa **árvore binária de pesquisa** organizada pelo número de títulos na plataforma e alfabeticamente. Cada empresa deverá possuir dados sobre a própria e estar associada aos seus títulos.

Seguidamente, pretende-se a criação de um sistema de **wishlists**, baseado numa **fila de prioridade** organizada por interesse. Os utilizadores passarão a guardar referências para os títulos que mais desejam comprar, sendo estas compostas pelo título em questão, o interesse do utilizador (de 1 a 10) e uma probabilidade representativa do quão provável se considera que o utilizador adquira o jogo se incitado a tal. Deve também ser possível selecionar, para cada utilizador, qual o item da sua wishlist que terá mais interesse em ser mostrada publicidade.

Finalmente, é pedida a implementação de uma **lista de ‘utilizadores adormecidos’** para cada título, baseada numa **tabela de dispersão**. Nas listas de cada título deverão constar os utilizadores que possuem uma probabilidade de compra acima de um valor residual mínimo. Estes utilizadores poderão ser removidos e readicionados conforme a sua atividade na biblioteca de jogos.

# **Solução Implementada**

**Registo de Empresas**

De modo a registar a informação de cada empresa na biblioteca de jogos, foi criada a classe “**Company**”. Esta classe possui o nome, NIF e contacto da entidade, assim como uma lista dos títulos criados pela mesma, que estão armazenados sob a forma de apontadores para referenciar as instâncias presentes na “**GameLibrary**”.

A listagem de todas as empresas encontra-se na classe “GameLibrary” sob a forma de uma árvore binária de pesquisa, organizada por número de títulos e alfabeticamente. Para este efeito foi usada a classe “**set**” da STL, que implementa uma forma de Árvore Vermelho-Preto.

É possível aplicar sobre esta lista de empresas as operações de criação, pesquisa, atualização e remoção, entre outras, cumprindo as operações básicas CRUD.

Implementamos também uma função que permite procurar pelo nome de *companies* recorrendo a uma wildstring, ou seja, uma palavra constituída por letras, \* (que correspondem a qualquer letra incluindo nenhuma ou várias) e ? (que faz de placeholder para qualquer letra), conseguindo assim fazer vários tipos de listagens e procuras alfabéticas.

**Sistema de Wishlists**

No sistema implementado cada utilizador possui a sua própria fila de prioridade representativa da sua wishlist. Esta contem objetos do tipo “**WishlistEntry”**, que possuem uma referência para o título que o utilizador deseja, um valor para o interesse de 1 a 10 e uma probabilidade de compra.

Esta probabilidade é calculada com base na similaridade dos outros jogos do utilizador a esse título, nos amigos que têm esse jogo na sua biblioteca, no número de cliques em publicidade do título e no número de pesquisas. A cada um destes fatores é atribuído um coeficiente, obtendo assim uma função linear de várias variáveis. Após calcular o respetivo valor da função, o resultado passa por uma função *sigmoid* para a distribuir no intervalo 0 < p < 1. Esta parte foi inspirada na estrutura de uma rede neuronal com uma função de ativação (neste caso a *sigmoid* porque pretendíamos uma probabilidade) com a diferença de que os parâmetros têm de ser ajustados manualmente, quer os coeficientes da função com diferentes variáveis quer da função ativação (esta normalmente é centrada na origem com ordenada a 0.5 e sobe rapidamente de 0 para 1 fazendo destes valores assintotas, a nós não nos interessava uma inclinação tão grande logo o *e* também é dividido por um e é somada uma constante para a deslocar para a direita no eixo dos xx).

As filas de prioridade são organizadas pelo interesse do utilizador e, em segundo caso, pela probabilidade de compra. É possível aplicar todas as operações básicas CRUD, assim como obter o próximo título que deverá ser apresentado ao utilizador de modo a conduzi-lo a uma transação. Visto que a probabilidade é calculada internamente, não pode ser alterada pelo user.

**Lista de ‘Utilizadores Adormecidos’**

Este foi o ponto mais complicado do trabalho, não por ser tecnicamente difícil, mas por causa do seu enunciado e da compatibilização de tudo o que nos era pedido (prova disso é que as funções para cada objetivo específico foram todas feitas rapidamente, tendo demorado apenas a interface que as teria de juntar e fazer delas sentido).

Tendo cada user o seu endereço de email, a hash function para a tabela de dispersão é feita tendo este por input. Começamos por tentar desenvolver nós a nossa própria função, mas, após lermos sobre o assunto, chegamos à conclusão de que esta não era a melhor ideia e, depois de alguma pesquisa, acabamos por escolher um algoritmo chamado djb2 segundo o nome Daniel J. Bernstein, que consiste em multiplicar cada letra por 33 e somar uma constante. O número 33 deve-se não só ao facto ser ímpar (que permite uma melhor distribuição que pares), mas também pela facilidade de o computar (bastam um shift e uma adição) tornando um algoritmo com uma rapidez bastante razoável em strings.

Como nada é diretamente dependente destas tabelas e visto o seu custo de atualização ser alto (tem de se atualizar probabilidades, rever os users que ficaram ativos…), esta é como o gato de Schrodinger: até se abrir a caixa (até se quiser consultar a tabela, que é o único propósito dela) ele está morto e vivo ao mesmo tempo. Ou seja, a tabela mantém um estado interno desatualizado até que seja consultada. Com a melhoria de que quando a nossa caixa é aberta o nosso gato está sempre vivo.

Como estas tabelas são algo calculável a partir dos dados que temos, optámos por não ocupar espaço desnecessário a guardá-la quando se faz save. Também consideramos que as operações CRUD não se aplicam aqui de forma direta, ou seja, ao user não pode nem deve ser deixada a possibilidade de alterar a tabela. Existem funções de remoção e adição, cujas partes do enunciado a que dizem respeito estão indicadas na documentação, mas estas nunca estão acessíveis ao utilizador.

**Outras Mudanças**

Além disto, foi também preciso adaptar algumas das coisas já existentes: adicionar um contador de publicidades vistas e pesquisas feitas (através de um tuplo) que vai sendo incrementado consoante se pesquise ou veja uma publicidade de certo jogo; tivemos de adicionar uma data interna à biblioteca para poder controlá-la e respeitar a condição de um jogador ser retirado se passado X meses não comprar um jogo; para verificar se algum jogo foi comprado nesse mesmo tempo foi ainda adicionado o ID do título às transações.

**Conceitos Utilizados**

* Classes
* Polimorfismo (overloading de funções, métodos virtuais, entre outros)
* Membros e métodos Static
* Exceções (lançamento e tratamento de diversos tipos)
* Árvores binárias de pesquisa
* Filas de prioridade
* Tabelas de dispersão
* Documentação do código usando Doxygen
* Leitura e escrita de dados em ficheiros
* Algoritmos de cálculo da Hash

# **Classes Implementadas**

**Company**

A classe Company representa uma empresa na Biblioteca de Jogos, contendo informações sobre a mesma e uma lista dos seus títulos.

**WishlistEntry**

A classe WishlistEntry representa uma entrada numa lista de desejos, contendo um apontador para o título desejado, um valor representativo do interesse do utilizador no título (de 1 a 10) e um float que representa o quão provável é o utilizador comprar o título se incitado a tal.

**Exceções**

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

* NoMatchingWishlistEntry: lançada quando uma WishlistEntry com probabilidade de compra acima de um determinado valor não existe;
* InvalidProbability: quando é introduzida uma probabilidade errada (acima de 1 ou abaixo de 0);
* NoMatchingWishlistEntryUser: lançada quando não existe nenhuma publicidade para ser mostrada

**CompareObject**

Foram ainda adicionadas várias classes de comparação de objetos para permitir ordenação e reordenação de novos elementos:

* CompareUsr: Que permite ordenar os users da tabela de dispersão por ID, pelo número de publicidade vistas, pelo número de pesquisas ou pela probabilidade de compra (esta escolha é feita por meio de um enum);
* CompareCompany: Para ordenar as *companies* da maneira descrita no enunciado;
* CompareCompanyByName: Para ordenar as *companies* por nome.

# **Casos de Utilização**

Neste capítulo apenas serão explicadas as novas funcionalidades da interface já desenvolvida para a parte anterior.

Assim sendo, a interface começa logo com uma novidade: é necessário o setup do número de meses para que um jogador seja considerado inativo pela biblioteca. A data é começada automaticamente para o dia 29 de novembro de 1972 (dia de “nascimento” do jogo Pong).

Após este input avançamos para um menu já conhecido de onde podemos prosseguir para o **Menu Principal**. Aqui existem duas novas opções: **Fast Forward** que permite andar com a data do jogo para a frente levando nos para um menu que nos deixa escolher entre andar X meses para a frente, X anos, X dias ou mesmo selecionar a data para que se quer ir (este X é escolhido pelo utilizador). Quando andamos para a frente não só são recalculados os users adormecidos, como também a idade dos mesmos. A segunda opção é o **Manage Companies** que nos leva a um menu que permite fazer todas as operações sobre este novo tipo de objeto: criar um novo, remover, listar brevemente, listar a informação completa entre outros. Aqui destacamos o algoritmo de pesquisa de *companies* por nome que permite pesquisas por wildstrings, ou seja, se um utilizador não se lembra do nome exato, mas apenas de alguns carateres, pode fazer pesquisas usando os caracteres especiais \* (significam 0 ou mais caracteres neste lugar) e ? (placeholder para um caracter que não o vazio).

Exemplo: a?a\* daria companhias como “amazon”, “atari”, “ana”…

Na criação de jogos foi adicionado um novo parâmetro a pedir a probabilidade a partir da qual um jogo quer ser notificado de que um user está inativo e, aquando da inserção do nome do desenvolvedor, caso este ainda não exista, é-lhe pedido para criar uma *company* com este nome (no momento, não sendo preciso sair da criação do jogo e criar a *company*).

No **Menu de Info de Títulos** é nos dada a opção de ver a sua tabela de dispersão, quer completa quer apenas dos user acima de um certo valor de: probabilidade se pedirmos para listar por probabilidade; número de publicidades vistas se for este o fator de ordenação; número de pesquisas pelo jogo ou pelo ID dos users (neste caso a listagem é sempre completa).

Já no **Menu do User**(menu após já se ter selecionado um user específico) este pode selecionar a 5 opção indo para o **Menu Wishlist** que permite ver um sumário de todos os itens na sua wishlist, adicionar um novo, remover, editar ou ver o próximo anúncio.

Foi adicionada a opção de um user procurar por título (para se incrementar o número de pesquisas). Esta pode ser feita através de vários parâmetros, permitindo várias espécies de listagens parciais ou totais (pode ser especificada plataforma, tipo, género, idade…). Foi também integrada publicidade (apesar de não fazer muito sentido, mas ser necessária segundo o enunciado) quando se joga, compra ou procura por um título.

# **Principais Dificuldades**

Como foi referido acima, as dificuldades começaram quando nos falam do cálculo de uma probabilidade no ponto 2 e culminam no ponto 3.

Começando pela probabilidade: “Este valor é calculado com base em várias outras métricas internas como número de cliques no título, número de vídeos/elementos promocionais do título foram visualizados, etc.”. O problema aqui vem de uma aparente desconexão do enunciado com o trabalho que já tinha sido feito para a primeira parte: se o objetivo é fazer uma estrutura que consiga guardar informação sobre uma biblioteca, qual o sentido que nesta sejam mostradas publicidades aos seus títulos? Algo não completamente problemático (até porque também já dava para “jogar” um título), mas que começa a lembrar o enunciado do terceiro teste…

A resolução deste problema não foi difícil: criou-se um tuplo por jogo por *user* com estas duas novas informações (o número de pesquisas e o número de publicidades vistas).

O verdadeiro problema vem com o enunciado da tabela de dispersão: “Pretende-se também manter uma lista “utilizadores adormecidos” para cada título numa tabela de dispersão.”. Lendo esta frase pensamos que era claro que isto seria um parâmetro dos jogos pois “cada título” teria uma. Acontece que parece não ser exatamente isto que o enunciado pede. Continuemos a analisar (esta análise está a ser feita não como um insulto às capacidades de interpretação do leitor, mas uma evidente falha na nossa, pois foi mesmo neste ponto – interpretação do enunciado – que surgiram a maior parte dos problemas).

“Sempre que um utilizador tem uma probabilidade de compra acima de um valor residual mínimo para um determinado título, deve ser adicionado à tabela do título correspondente.” Daqui deduzimos que afinal só há hashtable para os itens que têm uma probabilidade de compra, ou seja, para os itens que estão na wishlist de alguém já que esta era nos descrita como um parâmetro de uma entrada desta fila de prioridade. Aqui a lista de “utilizadores adormecidos” começa quase a ganhar outra forma: com o aparecimento de um mínimo (aquele que o fabricante do jogo define) para que se entre para a tabela, isto faz nos lembrar uma espécie de tabela com os potenciais compradores (toda a história das publicidades, pesquisas e da probabilidade a aumentar com esta sendo relevante para o próximo anúncio leva nos a esta dúvida).

Os problemas culminam na última frase: “Quando o utilizador compra este título, ou outro título da mesma plataforma dentro, passa a ser considerado ativo e é retirado da tabela de dispersão - devendo voltar a ser adicionado se não comprar nenhum título dentro de X meses.”. De facto, cada restrição individualmente faz sentido e são bastante fáceis de fazer (na documentação foi transcrita a parte do enunciado que cada função respeita). Até esta frase sozinha consegue ser coerente, apesar de não percebermos o “dentro” da parte entre vírgulas (remove-se quando há uma compra, adiciona-se quando fica inativo, fazendo da tabela uma verdadeira lista de “utilizadores adormecidos”).

A dificuldade foi em juntar tudo na interface porque a segunda e terceira frase são quase contraditórias: é para fazer uma lista de users adormecidos ou uma com base na probabilidade que se assemelha aos potenciais compradores? Imaginemos que um utilizador vai para a tabela porque apresenta uma alta probabilidade de compra do jogo. Acontece que este jogador compra um jogo da mesma plataforma, logo é removido da hashtable perdendo esta todo o valor de “potencial buyer” que poderia ter. Situação contrária: o que acontece quando passa muito tempo? A probabilidade decresceria, mas ele seria reinserido… Estas inserções e o seu contrário fazem com que este parâmetro perca qualquer valor e que esta passe a ser uma tabela puramente de users adormecidos.

Assim sendo, porque é que este é um parâmetro de adição inicial? Se há imensos users que não fizeram compra nenhuma nos últimos X meses, mas não têm uma probabilidade associada porque se calhar nem wishlist têm, porque é que estes não são considerados adormecidos? Não seria uma melhor métrica inicial se um user tem algum jogo para a mesma consola, para saber se a detêm e está realmente adormecido? Qual o sentido de juntar um parâmetro da wishlist, que transmite desejo de algo, com o facto de estar adormecido? Que informação ganha a biblioteca com isto? Os jogadores adormecidos para o jogo já vimos que não era, apesar de ser a que faria mais sentido fazer. Será os jogadores que desejaram o jogo há muito tempo?

Estas foram algumas das dúvidas que nos surgiram ao tentar compatibilizar a probabilidade com as restrições da terceira fase e a tentar fazer sentido do que havia sido escrito. Parece que fica a meio caminho entre *potencial buyer* e *asleep users* sem cumprir nenhuma: a primeira porque há users com probabilidade alta que não estão na tabela porque se encontram ativos, a segunda porque existem users verdadeiramente adormecidos que não estão na tabela por causa da probabilidade.

# 

# **Esforço de Cada Membro**

O trabalho foi realizado com sucesso, tendo conseguido implementar o que pensamos ser o requisitado. Salienta-se, no entanto, esta incerteza perante o enunciado, especificamente no terceiro tópico.

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.

Consideramos que todos ajudaram e contribuíram igualmente, cada um responsável pela sua parte e trabalhando ao seu ritmo.

As contribuições individuais de cada membro foram as seguintes:

**Manuel Coutinho**

* Estruturas de dados implementadas:
  + Tabela de Dispersão
* Contribuições adicionais:
  + Adição de exceções
  + Desenvolvimento da Interface
  + Temporalidade da Biblioteca
  + Contributos para a Wishlist
  + Função probabilidade
  + Reformulação das Transações
* Documentação Doxygen do projeto
* Escrita do relatório

**Mário Mesquita**

* Estruturas de dados implementadas:
  + Árvore de pesquisa binária
  + Fila de prioridade
* Classes implementadas:
  + Company
  + WishlistEntry
  + CompareUsr
* Contribuições adicionais:
  + Exceções relativas às classes implementadas
  + Função probabilidade
* Documentação Doxygen do projeto
* Escrita do relatório

**Gonçalo Oliveira**

* Contributos para a classe WishlistEntry e User
* Implementação de uma máquina de estados para escrita e leitura de ficheiros que representam o estado da biblioteca de jogos num dado instante. Esta implementação inclui manipulação de hash tables, filas de prioridade, sets, entre outras estruturas de dados. Parte desta teve que ser refeita devido a mudanças nos conteúdos a gravar.
* Desenvolvimento da interface entre o utilizador e o programa.
* Documentação Doxygen do projeto
* Escrita do relatório