

**Terceiro Trabalho Prático**

|  |  |
| --- | --- |
| Alunos | Manuel Francisco Dias Marques, nº36836 |
|  | Oxana Dizdari, nº39278 |
|  | Beatriz Patusco Neto, nº39320 |

|  |  |
| --- | --- |
| Engenheiro | Luís Assunção |
|  |  |

Relatório do terceiro trabalho prático, realizado no âmbito de Sistemas Distribuídos,  
do curso de licenciatura em Engenharia Informática e de Computadores  
Semestre de Verão 2014/2015

Maio de 2015

**Índice**

[Lista de Figuras v](#_Toc423463356)

[1. Introdução 1](#_Toc423463357)

[1.1 Problema 1](#_Toc423463358)

[1.2 Pontos importantes 1](#_Toc423463359)

[1.3 Temas da unidade curricular 1](#_Toc423463360)

[2. Solução Implementada 2](#_Toc423463361)

[2.1 Serviço de Jogo 2](#_Toc423463362)

[2.1.1 Criação de Jogo 3](#_Toc423463363)

[2.1.2 Regras e funcionamento de Jogo 4](#_Toc423463364)

[2.1.3 Registo do Player 4](#_Toc423463365)

[2.3 Serviço de Tradução 4](#_Toc423463366)

[2.2 Serviço de Notificação 4](#_Toc423463367)

[3. Conclusões 5](#_Toc423463368)

[Referêcias 6](#_Toc423463369)

# Lista de Figuras

[Figura 1 – Escolha da directoria para dar acesso ao ficheiro de configuração. 2](#_Toc420051663)

[Figura 2 – UI para pesquisa de músicas 5](#_Toc420051664)

[Figura 3 – Música encontrada online 6](#_Toc420051665)

# Introdução

Neste documento, irá ser descrito o problema proposto neste terceiro trabalho de Sistemas Distribuídos. Irá também ser explicado, de forma mais explícita possível, as decisões tomadas perante cada problema enfrentado e a sua respectiva solução.

## 1.1 Problema

Neste trabalho foi proposta a implementação de um serviço *Jogo.* Este jogo consiste na procura de um tesouro num tabuleiro NxN. Este tabuleiro é preenchido por mortes, vidas e casa vazias, para além do tesouro. Este último é único e a sua descoberta dá a vitória ao jogador que o encontrou. O jogo é partilhado entre um ou mais participantes e após o tesouro ser descoberto todos os jogadores são notificados que o jogo acabou e que perderam, caso não tenham sido eles a descobrir o tesouro. Esta notificação é feita através de um serviço cujo contrato é disponibilizado pelo *Jogo*, tendo cada *player* que se reger pelo conjunto de métodos presente nesse contrato.

## 1.2 Pontos importantes

Para além do objecivo princpial explicado na secção anterior, irá ser referido como é feita a criação do jogo, nomeadamente o estado inicial do tabuleiro e de cada jogador presente nele. Quanto aos jogadores será explicado mais detalhadamente como estes se irão registar no jogo. Irá ser explicada a importância de cada uma das casas do tabuleiro, sendo o tesouro a casa principal, e vida, morte e *nothing* as secundárias. Relativamente à afetação do jogo irão também ser explicados aspectos importantes sobre esse tema, como os acontecimentos após cada jogada.

## 1.3 Temas da unidade curricular

Neste trabalho prático serão abordados e consolidados alguns dos temas de Sistemas Distribuídos que foram leccionados nas aulas. Alguns deles são: Arquitetura orientada aos Serviços (SOA); Windows Communication Fondation (WCF); Alguns aspetos fundamentais do WCF como Endpoints, Protocolos de Comunicação, Service Operation, Service Contract, Operation Contract, Bindings, Ficheiros de Configuração, entre outros.

# Solução Implementada

## 2.1 Serviço de Jogo

O Jogo é o elemento central do trabalho, sendo a parte mais importante. Está alojado num serviço *GameService* e suporta um jogo de procura de um tesouro em que podem participar múltiplos jogadores. É este que faz a gestão dos *players* e que os notifica quanto a resultados do jogo e publicidade neste presente(acho que não é este, é mesmo o GameManager)*.* A publicidade é emitida usando um serviço intermédio que irá ser explicado posteriormente.

*GameService* disponibiliza dois contratos. O primeiro é o *IGameManager* cujas operações permitidas são *StartGame* – responsável por iniciar o jogo, especificando o tamanho do tabuleiro, *EndGame –* que serve para fechar o jogo e o *SetAdv* – que é utilizado para enviar publicidade aos jogadores. O segundo é *IGamePlayer* que permite ao jogador juntar-se ao jogo, especifinado o seu nome e a sua língua (esta útlima irá servir para a publicidade enviada pelo *Manager* ser traduzida para a língua do jogador em questão e só depois ser enviada) – *JoinGame*. Permite também fazer jogadas – *MakeMove*, especificando a casa onde o mesmo quer jogar, e sair do Jogo – *ExitGame.*

O contrato *IGamePlayer* do *GameSerice* disponibiliza também um *CallbackContract* do tipo  *IGamePlayerReceiverCallback,* constituído por dois métodos, o *NewAnnounce* – utilizado para notificar os jogadores, por exemplo sobre o fim do jogo caso alguém já tenha ganho, e o *NewAdvertisement* – para enviar a publicidade traduzida aos jogadores.

Para o *GameService* foi escolhido o *InstanceContextMode* como *Single*, visto que o pretendido é que a mesma instância seja partilhada por vários clientes, neste caso múltiplos jogadores. O *ConcurrencyMode* escolhido é *Single*, de maneira a não permitir múltiplos pedidos serem processados ao mesmo tempo, causando problemas de concorrência. (No trabalho temos Multiple, mudo? Podem mudar esta frase se houver melhor maneira de dizer :) )

### 2.1.1 Criação de Jogo

Uma das regras do jogo é que é o Gestor que o deve iniciar. Para isso, foi criada uma aplicação *GameManager* que tem como responsabilidade indicar o tamanho do tabuleiro e começar o jogo em si. Este *GameManager* utiliza o contrato *IGameManager,* específico para utilizadores com privilégio sobre o jogo. Estes privilégios abrangem o início e o fim de um jogo tal como a gestão de anúncios. Ou seja, tendo controlo sobre qual o anuncio inicial e podendo mudar e enviar vários anúncios ao longo do jogo.

O jogo decorre sobre um tabuleiro NxN definido pelo *GameManager*. Aqui é também definido o anúncio inicial que vai estar presente na *user interface* de cada participante. Ao clicar no botão *StartGame* (Figura 1), depois de definido o tamanho do tabuleiro, é invocado o méotodo *StartGame* do respectivo contrato e a partir deste momento, o serviço de jogo está pronto a ser utilizado.

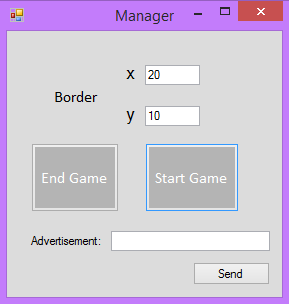


Figura 1 - Início de jogo com o tabuleiro de 20:10

### 2.1.2 Regras e funcionamento de Jogo

Cada jogo é composto por um Board com n por n Cells. Cada Cell tem características que influenciam a duração do jogo. Uma destas possui o tesouro, que ao ser descoberto por um dos jogadores faz com que seja vencedor, e de seguida o jogo notifica todos os outros participantes que perderam. Além do tesouro as Cells ainda podem ser zero, vida ou morte. Estas, ao contrário do tesouro, podem estar repetidas por todo o tabuleiro de uma maneira aleatória.

Cada jogador tem uma contagem inicial de três vidas, que quando chega a zero significa a derrota do participante. A casa morte decrementa o contador de vidas, enquanto que a casa vida o incrementa. A característica zero não influência nada no jogo.

Depois de feita a jogada pelo *Player*, é modificado o conteúdo da *cell* (casa) de uma forma aleatória, de maneira a não permitir ao jogador coleccionar um número muito elevado de vidas, caso ele descubra que lá estejam. Assim, se o jogador apostou na casa 1:1 e ganhou uma vida, é feito reset a esta casa e na próxima jogada pode conter morte, outra vida ou nada.

### 2.1.3 Registo do *Player* e como se joga

Foi criada uma aplicação *GamePlayer* que utiliza o contrato *IGamePlayer* do *GameService* e é usada pelos jogadores. Quando esta aplicação corre pela primeira vez, o jogador não fica imediatamento apto a jogar, visto que necessita de se registar no jogo. Para isso deve inserir o seu nome e a sua língua (Figura 2) e clicar no botão *Regist*.

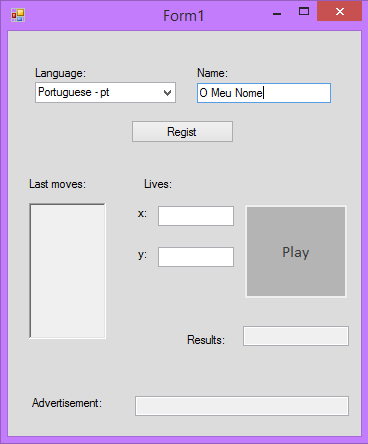


Figura 2 - Registo do utilizador

Depois de ter sido feito o registo, o jogador está apto a jogar. Deve indicar o valor do x e do y que corresponderão à casa onde o jogador quer apostar, podendo clicar assim no botão *Play*. À esquerda serão registadas as útlimas jogadas e na *TextBox* de *Results* o resultado da jogada feita.

Poderá ver um exemplo de uma jogada feita na figura 3. Neste caso o jogador apostou em 1:1 e não lhe calhou nada.

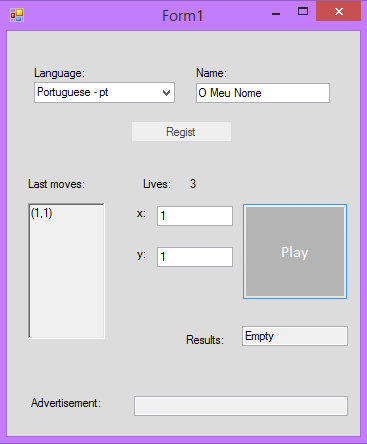


Figura 3 - Jogada feita em 1:1

Caso o jogador efectuar uma jogada sem o Gestor ter iniciado o jogo, é lançada uma exceção.

Esta aplicação aloja também um serviço de notificações, criando uma classe *Receiver* que implementa a interface *IGamePlayerCallback* definida no contrato. Esta parte irá ser explicada em detalhe na Secção 2.2.

## 2.3 Serviço de Tradução

## 2.2 Serviço de Notificação

# Conclusões

Concluímos que este trabalho é muito útil para começar a entender determinados conceitos dos Sistemas Distribuidos e o quão importante podem ser no nosso futuro profissional, dando-nos uma ideia de como determinadas aplicações bastante conhecidas são implementadas . Apesar de o trabalho no geral correr bem, as maiores dificuldades encontradas no mesmo foram arranjar soluções para trabalhar com as limitações dos objetos serializáveis e entender correctamente a diferença entre SingleCall e Singleton.

Tivemos a possibilidade de também exercitar várias técnicas aprendidas ao longo do curso (por exemplo modos de pesquisa assíncrona), o que se torna bastante benéfico.

# Referêcias

|  |  |
| --- | --- |
| [1] | “Peer-to-peer with VB .Net,” 21 05 2015. [Online]. Available: https://books.google.pt/books?id=MBrYAiEikR0C&pg=PA38&lpg=PA38&dq=peer+to+peer+singleton&source=bl&ots=s1bJKVSFPh&sig=UXOk8Xn7fN3W0P2sgubEC1gZjuk&hl=pt-PT&sa=X&ei=1Q1eVZ\_JMsH0UIHwgKAP&ved=0CC4Q6AEwAg#v=onepage&q=peer%20to%20peer%20singleton&f=false.. |
| [2] | “Remoting in C#,” 21 05 2015. [Online]. Available: http://www.jot.fm/issues/issue\_2004\_01/column8.pdf. |