

**Primeiro Trabalho Prático**

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

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

Relatório do primeiro 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](#_Toc419970661)

[1. Introdução 1](#_Toc419970662)

[1.1 Problema 1](#_Toc419970663)

[1.2 Pontos importantes 1](#_Toc419970664)

[1.3 Temas da unidade curricular 1](#_Toc419970665)

[2. Solução Implementada 2](#_Toc419970666)

[2.1 Interface IPeer 2](#_Toc419970667)

[2.2 Registo Remoto do Peer 2](#_Toc419970668)

[2.2.1 Ficheiro de configuração xml 2](#_Toc419970669)

[2.2.2 Objecto Singleton 2](#_Toc419970670)

[2.2 Pesquisa de Música 2](#_Toc419970671)

[2.2.1 Pesquisa Local 2](#_Toc419970672)

[2.2.2 Pesquisa Online 2](#_Toc419970673)

[3. Correr o Sistema Distribuído 3](#_Toc419970674)

[4. Conclusões 4](#_Toc419970675)

[Referências 5](#_Toc419970676)

[A.1 Diagramas da Aplicação 6](#_Toc419970677)

# Lista de Figuras

[Figura 2 - Legenda da figura com o logotipo do ISEL – versão 2. 4](#_Toc419918416)

[Figura 3 – Diagrama de casos de utilização. 14](#_Toc419918417)

# Introdução

Neste documento, iremos descrever brevemente o problema proposto no âmbito do primeiro trabalho prático da unidade curricular Sistemas Distribuídos. Iremos também explicar de forma mais explícita possível, as decisões tomadas perante cada um dos dilemas e a solução implementada.

## 1.1 Problema

Pretende-se desenvolver um sistema distribuído, em que cada utilizador é um *Peer*, isto é, servidor e cliente ao mesmo tempo. Este *peer* deve gerir informação, através de uma aplicação. A informação consiste numa colecção de referências musicais (Artista, Álbum, Título, Ano, Formato, etc.).

Tendo cada um dos utilizadores a sua própria colecção de referências musicais, o sistema desenvolvido deve permitir, como resultado final, a pesquisa de músicas (e.g.: por título, artista ou álbum). Deve ser tido em conta que, no caso de o utilizador em questão não conter a música pretendida, há também a possibilidade de existir conexão entre este e outros *peers* online.

## 1.2 Pontos importantes

Para além do objecivo princpial explicado na secção anterior, devem ser tidos em conta vários pormenores igualmente relevantes. Esses pormenores são o aumento do registo de outros *Peers* online, de maneira a aumentar a rede de contactos para futuras pesquisas; A pesquisa de uma música que não existe na biblioteca local; Tratar um pedido de maneira a não demorar tempo indetrimando; Tratamento de falhas; Visualização de pedidos feitos ao *Peer* em questão.

## 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: Modelos e Arquitecturas de Sistemas Distribuídos, nomeadamente Modelo *Peer to Peer*;

# Solução Implementada

## 2.1 Registo Remoto do Peer

### 2.2.1 Ficheiro de configuração xml

O ficheiro XML contém a informação necessária para a construção de um Peer. A lista de músicas e o URL dos Peers que conhece estão indicados no mesmo, bem como o porto onde fica alojado remotamente.

Ao iniciar a aplicação é pedido que seja indicada a diretoria deste ficheiro e com o auxílio do XML Serializer, é obtida uma referencia para uma instancia de *PeerInfo* que contem toda a informação presente no ficheiro XML.

***Exemplo :***

*<?xml version="1.0" encoding="utf-16"?>*

*<PeerInfo*

*xmlns:xsd="http://www.w3.org/2001/XMLSchema"*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">*

*<port>8000</port>*

*<portname>nameport</portname>*

*<name>Peer3</name>*

*<musics>*

*<Music>*

*<Title>Solteiro</Title>*

*<Artist>Regula</Artist>*

*<Album>Album1</Album>*

*<Year>2000</Year>*

*<Format>CD</Format>*

*<Owner>http://localhost:8000/RemotePeer.soap</Owner>*

*</Music>*

*<Music>*

*<Title>Poetas de karaoke </Title>*

*<Artist>Sam the kid</Artist>*

*<Album>Album2</Album>*

*<Year>2001</Year>*

*<Format>MP3</Format>*

*<Owner>http://localhost:8000/RemotePeer.soap</Owner>*

*</Music>*

*</musics>*

*<friends>*

*<Url>http://localhost:8888/RemotePeer.soap</Url>*

*</friends>*

*</PeerInfo>*

### Peer

Peer é uma componente Singleton do nosso projeto. Esta decisão baseou-se no facto de nas componentes SingleCall ser criada uma instância por pedido.

A diferença entre SingleCall e Singleton baseia-se no facto de, no mdo

The difference between SingleCall and Singleton is that, for SingleCall , every incoming request will get a new instance of the defined type created to handle that call. Each instance will have its own memory space and instance variables, but they can still share static and global variables, external resources, files, network connections, etc. If the SingleCall class is coded to access any shared memory state in a thread-unsafe manner, then you will have issues.

A Singleton, on the other hand, only gets one instance created for ALL incoming requests, so by definition, every instance variable in use within that singleton is, in fact, shared among all incoming requests. A good example might be a message publisher, that all code in the server needs to access to send messages to one or more subscribed clients....

## 2.2 Interface IPeer

As interfaces são essenciais para aceder a objetos remotos do .NET. Estas permitem que exista uma separação completa entre a componente Servidora e Cliente do Peer.

Todos os Peer’s têm de implementar esta interface e os métodos da mesma são essenciais para a componente Servidora dos Peer’s do nosso projeto. Posto isto um Peer conecta-se com outro, obtendo a referência para um IPeer com especifico URL, conhecendo assim parte da implementação do mesmo e facilitando a comunicação entre ambos.

## 2.3 Pesquisa de Música

Para registar o peer, o utilizador introduz o ficheiro de configuração xml, que contém o porto e o nome do peer de maneira a saber onde registá-lo. Contém, também a colecção de referências musicais do Peer em questão. Depois de definido o caminho para o ficheiro de configuração, desseriazamos o seu conteúdo para uma instância de PeerInfo. A classe PeerInfo é uma classe auxiliar que contém os campos necessários para uma correcta desserialização, de maneira a guardar em memória tanto a colecção de músicas, como o conjunto de Peers ja conhecidos.

Estando o Peer já registado no seu canal, é craido o proxy, para o objecto Well-Known, através de Activator.GetObject, indicando o tipo, neste caso IPeer, e o seu URL. Este deve poder armazenar tanto as músicas como os peers em sua própria lista, por isso o próximo passo é dar-lhe a conhecer os dados que foram desserializados e guardados na instância de PeerInfo. Visto que o Serializador Soap não suporta a serialização de tipos genéricos, a lista completa não pode ser passada ao Peer, por isso as músicas e os links dos peers conhecidos devem ser passados um a um.

Estamos aptos, assim, a pesquisar músicas. A pesquisa pode ser feita por Titulo, Album ou Artista, e inicialmente é feita localmente e só depois online, fazendo pedidos aos Peers conhecidos.

### 2.3.1 Pesquisa Local

### 2.3.2 Pesquisa Online

## 2.4 Visualização de pedidos feitos

## 2.5 Tratamento de falhas

# Correr o Sistema Distribuído

Codigo:

**namespace ps;**

**public static void main() {**

**System.out.println(“PS - Projecto e Seminário”);**

**}**

# Conclusões

Neste trabalho tratou-se o problema. Foi formulada a solução que assenta nos princípios de boas práticas aprendidos ao longo do curso.

A solução obtida atingiu resultados satisfatórios.

# Referêcias

|  |  |
| --- | --- |
| [1] | Wikipedia, “Big data --- Wikipedia, The Free Encyclopedia,” http://en.wikipedia.org/w/index.php?title=Big\_data&oldid=648786139, 2015. |
| [2] | J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong e J. Zhang, “What Will 5G Be?,” *IEEE Journal on Selected Areas in Communications,* vol. 32, n.º 6, pp. 1065-1082, 2014. |
| [3] | L. Boytsov, “Indexing Methods for Approximate Dictionary Searching: Comparative Analysis,” *J. Exp. Algorithmics,* vol. 16, n.º may, p. 1.81, 2011. |
| [4] | X. Ding, X. Zhu e G. Wu, “Data mining with big data,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, n.º 1, pp. 97-107, 2014. |
| [5] | T. Jurkiewicz e K. Mehlhorn, “On a Model of Virtual Address Translation,” *J. Exp. Algorithmics,* vol. 19, n.º jan, pp. 1-18, 2015. |
| [6] | B. Kernighan e P. Plauger, The Elements of Programming Style, New York, NY, USA: McGraw-Hill, Inc., 1982. |
| [7] | J. Neumann, The Computer and the Brain, New Haven, CT, USA: Yale University Press, 1958. |

# A.1 Diagramas da Aplicação

Estamos no início do anexo 1. Nalguns casos, é conveniente colocar anexos de forma a complementar os resultados. Por vezes, em casos excepcionais devido à sua dimensão, as figuras têm que ser apresentadas de forma a ocupar toda a página, na forma de paisagem (*landscape*). Podemos fazer isso da forma que se apresenta na figura 3.



Figura 3 – Diagrama de casos de utilização.