

**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](#_Toc420006162)

[1. Introdução 1](#_Toc420006163)

[1.1 Problema 1](#_Toc420006164)

[1.2 Pontos importantes 1](#_Toc420006165)

[1.3 Temas da unidade curricular 1](#_Toc420006166)

[2. Solução Implementada 2](#_Toc420006167)

[2.1 Registo Remoto do Peer 2](#_Toc420006168)

[2.1.1 Ficheiro de configuração xml 2](#_Toc420006169)

[2.1.2 Peer 3](#_Toc420006170)

[2.2 Interface IPeer 4](#_Toc420006171)

[2.3 Pesquisa de Música 4](#_Toc420006172)

[2.3.1 Pesquisa Local 4](#_Toc420006173)

[2.3.2 Pesquisa Online 5](#_Toc420006174)

[2.4 Visualização de pedidos feitos 5](#_Toc420006175)

[2.5 Tratamento de falhas 5](#_Toc420006176)

[3. Correr o Sistema Distribuído 6](#_Toc420006177)

[4. Conclusões 7](#_Toc420006178)

[Referêcias 8](#_Toc420006179)

# 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

A primeira coisa que fazemos quando lançada a aplicação, é registar remotamente o Peer. Segue-se os passos realizados para tal, bem como breves comentários sobre as decisões tomadas.

### 2.1.1 Ficheiro de configuração xml

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

Para registar o peer, ao iniciar a aplicação, é pedido ao utilizador que indique a diretoria de um dos ficheiros de configuração *.xml*. Depois de aberto o ficheiro para leitura, com o auxílio do XML Serializer, 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.

**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>

### 2.1.2 Peer

Peer é um objeto remoto, ou seja, pode ser acedido por outros domínios. Este objeto deriva de *System.MarshalByRefObject,* o lhe dá a capacidade de ser acedido remotamente.

O facto de Peer derivar de *System.MarshalByRefObject* não disponibiliza automaticamente o objeto remotamente, sendo necessário registar um canal para o mesmo que possibilita também a troca de mensagens entre Peer’s.

Na nossa aplicação foi registado um canal HTTP disponibilizado pelo .NET, *HttpChannel ,que* utiliza SOAP (Simple Object Access Protocol) , no porto especificado no ficheiro XML do Peer que se pretende registar.

De seguida é necessário registar o tipo do objeto que queremos disponibilizar remotamente usando o método RemotingConfiguration.RegisterWellKnownServiceType(..). Neste método é especificado também o tipo de ativação do objeto, escolhendo no nosso caso a utilização do modo de ativação **Singleton**. Geralmente os objetos SingleCall são perfeitos para aplicações que querem simplesmente expor recursos de servidores. Não podem ser usados em aplicações Peer-to-Peer, onde a base é a comunicação bidirecional entre aplicações long-runtime, pois esta define um objeto statless, criado por cada invocação de um método do objeto e destruída no final do mesmo.

Nestas aplicações é necessário utilizar objetos Singleton, que são associados a um endpoint com uma única instância do objeto. Independentemente do número de clientes que se conectam ao mesmo, apenas existirá um instância do objeto remoto.

Estando o Peer já registado no seu canal, é criado um proxy associado ao URL especificado na informação do Peer, com o auxilio do método GetObject(..) de *System.Activator*, indicando para além do URL, também o tipo. 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.

Depois de finalizados estes passos, pode ser iniciada a pesquisa de musicas.

## 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

Como referido anteriromente, na Secção 2.1.2, depois de respeitados os passos de registo do Peer, sstamos aptos, a pesquisar músicas. A pesquisa pode ser feita por Titulo, Álbum ou Artista, e inicialmente é feita localmente e só depois online, fazendo pedidos aos Peers conhecidos.

### 2.3.1 Pesquisa Local

Pesquisa local é realizada apenas na biblioteca local do peer. Ou seja, na lista que foi armazenada através da leitura do ficheiro xml. A interação com o utilizador faz-se através do WindowsForm, onde é possível inserir dados pelos quais a pesquisa deve ser feita. Se encontrada localmente, a referência para a música é retornada e o utilizador pode visualizar os seus dados.

### 2.3.2 Pesquisa Online

Caso não tenha sido encontrada a música por pesquisa loal, é necessário pesquisar pela música nos peers conhecidos. Para que isso seja possível, é ncessário estabelecer a conexão com os peers conhecidos em questão. Sendo disponibilizada a interface *IPeer* e sendo conhecido o url de cada um dos Peers conhecidos, recorre se de novo a do método GetObject(..) de *System.Activator*.

**IPeer peer = ((IPeer) Activator.GetObject(typeof (IPeer), url));**

É criado e guardado em *peer* o proxy para um Peer que esteja a escuta de pedidos no url dado.

## 2.4 Visualização de pedidos feitos

Fef

## 2.5 Tratamento de falhas

sddv

# 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

<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>.

|  |  |
| --- | --- |
| [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. |