

**Sistemas de Distribuidos**

**Trabalho prático 1**

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Autores | Daniel Abreu nº 38539  João Pereira nº 37628 |
|  | Luís Lucas nº 38849 |

Semestre de Verão 2015/2016

Maio de 2015

**Índice**

[1. Introdução 1](#_Toc417484093)

[2. Formulação do Problema 3](#_Toc417484099)

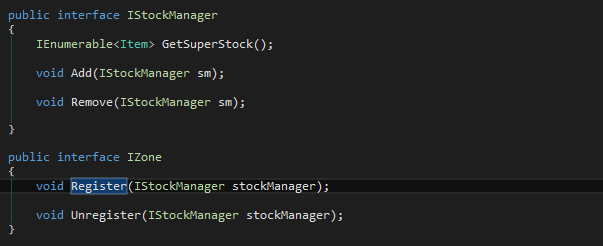
# 

# Introdução

# Desenvolvimento

Podemos separar a solução em duas grandes partes, o cliente que no contexto do problema será o projecto *Super*, este projecto é uma aplicação de consulta de stocks de produtos de supermercados, para que esta consulta seja abrangente a todos os supermercados de uma hipotética cadeia de supermercados esta aplicação terá que comunicar com um servidor que lhe transmitirá essa informação. O projecto *Zone* reflecte o comportamento do servidor, este projecto tem a capacidade de poder comunicar com outros servidores de forma a ter conhecimento do stock global de todos os supermercados registados nos servidores de zona. Este servidor realiza pedidos a outros servidores obtendo informação do stock disponível, e fornece essa mesma informação ao seu cliente, fazendo deste componente um *peer-to-peer*.

De forma a existir um contrato conhecido e respeitado entre estas duas entidades foi também criado um projecto que é uma biblioteca *.DLL* necessária tanto na compilação do projecto *Zone* como do projecto *Super*, neste projecto estão definidas as interfaces *IZone* e *IStockManager* que serão as entidades necessárias de conhecimento global para a implementação da solução adoptada.



A interface *IZone* tem um método *Register* que permite a um *Super* registar um objecto do tipo *IStockManager*. Cada servidor de zona comunica com um outro servidor de zona obtendo informação e enviando informação, no momento do registo de um *Super* numa zona esta comunica com a zona que esta conhece, registando também o objecto IStockManager. Este procedimento permite que a próxima zona se registe na sua zona conhecida, e a próxima zona numa outra e este procedimento irá continuar até que uma zona feche um círculo registando na zona inicial onde o *Super* se registou inicialmente. Desta forma todas as zonas têm conhecimento de todos os objectos *IStockManager* presente no sistema todos os *Super* registados.

Também presente na interface *IZone* está um método *Unregister* que recebe um objecto *IStockManager* eliminando-o da sua zona, este método tem um comportamento idêntico ao *Register*, que propaga a chamada a este método a sua zona conhecida eliminando o objecto *IStockManager* totalmente do sistema.

O método *GetItemStock* está também presente na interface *IZone* este método recebe o nome do produto que se quer saber o stock e retorna todo o stock desse mesmo produto presente no sistema.

Convém referir que o objecto *Zone* que implementa *IZone* é registado pelo servidor *Zone* em modo *Singleton* ou seja, cada vez que é obtida uma instancia deste tipo remotamente será sempre a mesma instância que será partilhada por todos os *Super* que se queiram registar no servidor, assim como a zona que tenha uma instância de outra, esta instância será a mesma que os *Super* obtêm.

A interface *IStockManager* tem dois métodos controladores da sua lista interna que contém todos os StockManagers presentes no sistema(*Add* e *Remove*). Além destes contém ainda um método *GetSuperStock* que devolve o stock do *Super* que o instanciou.

Esta lista de StockManagers é atualizada de cada vez que são chamados os métodos *Register* e *Unregister* de *IZone.* Assim todos os StockManagers conhecem todos os StockManagers do sistema, por isso, numa eventual falha do sistema de servidores, os StockManagers vão ser autossuficientes para devolver o stock presente em todo o sistema.

# Conclusão