



Universidade do Porto

FEUP Faculdade de
Engenharia

Financial Institute

Relatório Final

Carlos Babo - 080509118

Hélder Moreira - 080509170

30-05-2012

Introdução

Este relatório tem como objetivo descrever o projeto desenvolvido no âmbito da unidade curricular de Tecnologias de Distribuição e Integração do Mestrado Integrado em Engenharia Informática e Computação. Desta forma, é exposto o problema a resolver e a forma como a solução foi implementada, quer em termos de arquitetura escolhida, como também em termos de decisões consideradas relevantes.

O projeto sugerido tinha como objetivo desenvolver uma solução funcional de um sistema financeiro baseado numa arquitetura orientada aos serviços (SOA).

A solução implementada foi desenvolvida com recurso à tecnologia .NET da Microsoft, abordada nas aulas teóricas e ao modelo de programação WCF. Para além disso, foi também desenvolvido um *site* em ASP.NET. A solução é composta por vários componentes, que são descritos em detalhe no relatório, e que interagem entre si de determinadas formas.

Descrição do Projeto

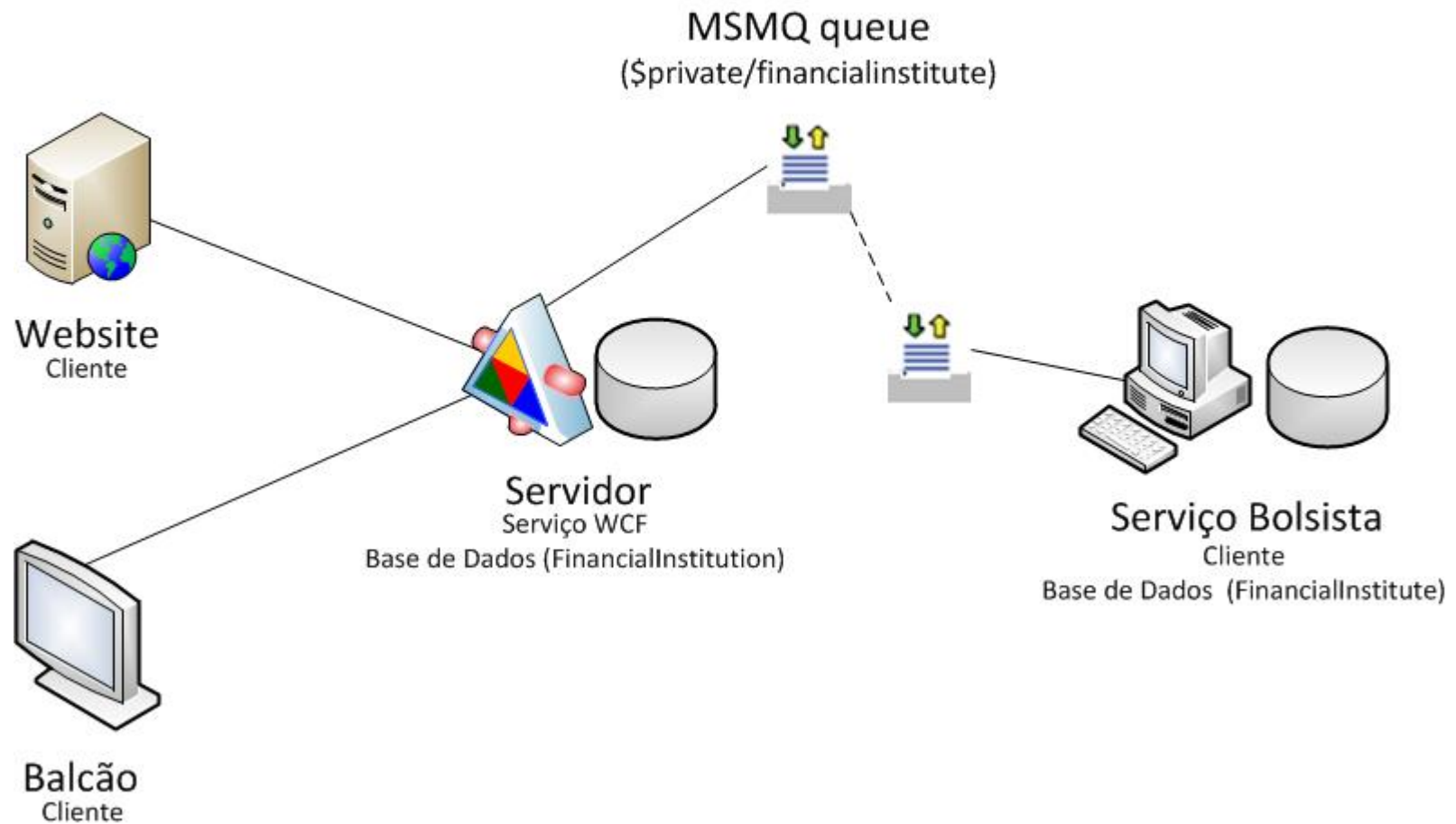
Neste projeto são implementadas quatro aplicações que trabalham em conjunto para simular um cenário que representa uma instituição financeira. Esta aceita dos seus clientes ordens de compra e venda de ações, que podem ser emitidas de dois locais distintos: balcões, onde o funcionário introduz as ordens de acordo com as instruções do cliente ou através de uma aplicação web que está disponível para os clientes e com a qual podem criar novas ordens ou mesmo consultar o estado das ordens já existentes.

A instituição financeira possui um servidor onde é alojada a aplicação e onde está também alojado o *web service* que permite quer à aplicação web, quer à aplicação desktop, realizarem os seus pedidos.

Para além disto, existe um serviço bolsista nesta instituição financeira que gere as ordens que são criadas. Para esse efeito, foi criada uma aplicação desktop específica para o efeito, onde os funcionários deste serviço podem consultar as ordens que são criadas, e decidir quais as que serão executadas. Ao executar uma ordem, o funcionário deve inserir a cotação atual da ação e é comunicado ao servidor a mudança de estado (através do *web service* disponibilizado). Ao ser alterado o estado, o servidor envia uma mensagem de correio eletrónico ao cliente da ordem respetiva, informando-o da mudança. Esta aplicação é atualizada em tempo real, permitindo ao funcionário saber quais as novas ordens que são realizadas sem qualquer tipo de ação da sua parte.

O serviço bolsista não funciona em permanência, e como tal, é utilizada uma fila de mensagens assíncrona para manter a comunicação nesta situação. O servidor envia uma mensagem para a fila de mensagens e quando o serviço bolsista estiver disponível recebe-a, atualizando a sua base de dados de acordo com a mesma.

Diagrama da Arquitetura



Tal como é possível ver na figura, a arquitetura está dividida em quatro aplicações independentes: cliente de balcão, cliente *web*, servidor e serviço bolsista, tal como pedido no enunciado do projeto.

O cliente de balcão, o cliente *web* e o serviço bolsista contêm uma referência ao serviço disponibilizado pelo Servidor e fazem chamadas aos seus métodos.

O servidor comunica ainda com a fila de mensagens *financialinstitute* que permite informar o serviço bolsista de novas ordens.

Decisões relevantes de implementação

Tendo em conta a arquitetura orientada a serviços pretendida na resolução do problema, optamos por utilizar o modelo de programação Windows Communication Foundation (WCF), tendo em conta o seu modelo de programação unificado, o suporte a SOA (Service-oriented architecture), o facto de seguir os padrões de mercado e por permitir uma arquitetura flexível e extensível. Este modelo está presente no servidor, que disponibiliza um serviço de acesso remoto e nos diferentes clientes que o têm como referencia e o invocam. O *binding* escolhido para este serviço foi o netTcpBinding, que se baseia no protocolo TCP, permite ligações entre diferentes máquinas e que contém um mecanismo de segurança pré-definido (Windows authentication).

Para guardar os dados da aplicação, optou-se por utilizar duas bases de dados SQL (uma para o servidor, outra para o serviço bolsista), de forma a facilitar a manutenção da informação entre sessões. As bases de dados contêm os dados associados com as ações. Enquanto o servidor tem sempre na sua base de dados todas as ações criadas num dado momento, a base de dados do serviço bolsista pode não estar atualizada, tendo em conta que este serviço não está sempre em funcionamento. No entanto, quando a aplicação é ligada, é feito um pedido ao servidor de forma a atualizar a base de dados com os pedidos por executar.

A fila de mensagens foi criada utilizando o protocolo MSMQ da Microsoft e permite enviar e receber mensagens de forma segura e assíncrona, tal como especificado no problema. Assim, a comunicação do servidor com o serviço bolsista é efetuado através da fila privada criada para o efeito (FinancialInstitute).

Para a construção do website, recorreu-se à tecnologia ASP.NET e ao ASP.NET Development Server para publicar o *website* para efeitos de teste.

Lista de funcionalidades implementadas

Servidor:

Classe:

FinancialOps (implementa a interface IFinancialOps, que define o contrato de utilização e é oferecida no serviço do servidor)

Métodos:

Boolean NewOrder(int client, string email, int op, int type, int quantity): permite adicionar uma nova ordem à base de dados do servidor. É também enviada uma mensagem para a fila de mensagens assíncrona para que o serviço bolsista seja avisado da nova ordem.

int GetStatus(int id): devolve o estado uma ordem (executada, ou por executar)-

Boolean ChangeOrder(int id, double cotation, String email): muda o estado de uma ordem de “por executar” para “executada”, de acordo com a cotação conseguida. Recebe também o e-mail do cliente para ser enviado o e-mail a notificar, sem ser preciso estar a procurar na base de dados qual o cliente da ordem com id passado como argumento. Este método só pode ser utilizado pelo serviço bolsista.

List<List<String>> GetRequestsByClient(int client): devolve todas as ordens de um cliente.

Cliente:

Apesar de terem sido implementadas duas aplicações cliente (*desktop* e *web*) a sua estrutura é bastante semelhante. Foram implementados 3 páginas/*forms* que dão ao utilizador a interface necessária para realizar as diferentes tarefas;

DeskForm.cs (desktop) / Default.aspx (web) - este é o ecrã inicial da aplicação e partir do qual o utilizador pode escolher se pretende consultar o estado de alguma ordem ou criar uma nova ordem;

ConsultarEstado.cs (desktop) / ConsultarOrdens.aspx (web) - neste ecrã é fornecida ao utilizador a interface necessária para que possa consultar os estados das ordens de um cliente, mediante a inserção do *id* do mesmo. Aqui é utilizada a função *GetRequestsByClient(int client)* fornecida pelo serviço e que permitirá obter as ordens do cliente escolhido pelo utilizador;

NovaOrdem.cs (desktop) / CriarOrdem.aspx (web) - aqui está implementada a interface relativa à criação de uma nova ordem. São pedidos ao utilizador os vários campos necessários para a criação da mesma e é efetuado tratamento de erros, não sendo possível ao utilizador inserir dados inválidos. A ordem é adicionada usando o método fornecido pelo serviço: *NewOrder(int client, string email, int op, int type, int quantity)*.

Serviço Bolsista:

Classe:

FinancialInstituteOps (definição da interface IFinancialInstituteOps, que define o contrato de utilização e é oferecida no serviço do serviço bolsista)

Métodos:

void ReportToInstitute(string time, int client, string email, int op, int type, int quantity, int id): este método, do tipo oneWay (ou seja, não espera valor de retorno por ser uma chamada assíncrona) permite reportar ao serviço bolsista que existe uma nova ordem. Se a aplicação estiver a correr, essa notificação é feita automaticamente. Caso esteja desligado, a fila de mensagens trata de guardar essa informação até que o serviço esteja novamente disponível. Ao adicionar uma nova ordem, a base de dados do serviço bolsista é atualizada automaticamente.

Conclusão

Com a realização deste trabalho foi possível colocar em prática a matéria abordada nas aulas sobre SOA (Service-Oriented Architecture), WCF, filas de mensagens, entre outros assuntos pertinentes e perceber como integrar tudo numa solução com vários componentes, sendo que, tal como no primeiro projeto, o facto de aplicar os conhecimentos a uma situação real permite uma melhor consolidação e interiorização dos conceitos que foram transmitidos.

Os objetivos propostos foram alcançados com sucesso, sendo prova disso o facto da solução encontrada estar completamente operacional e com todas as funções propostas implementadas.

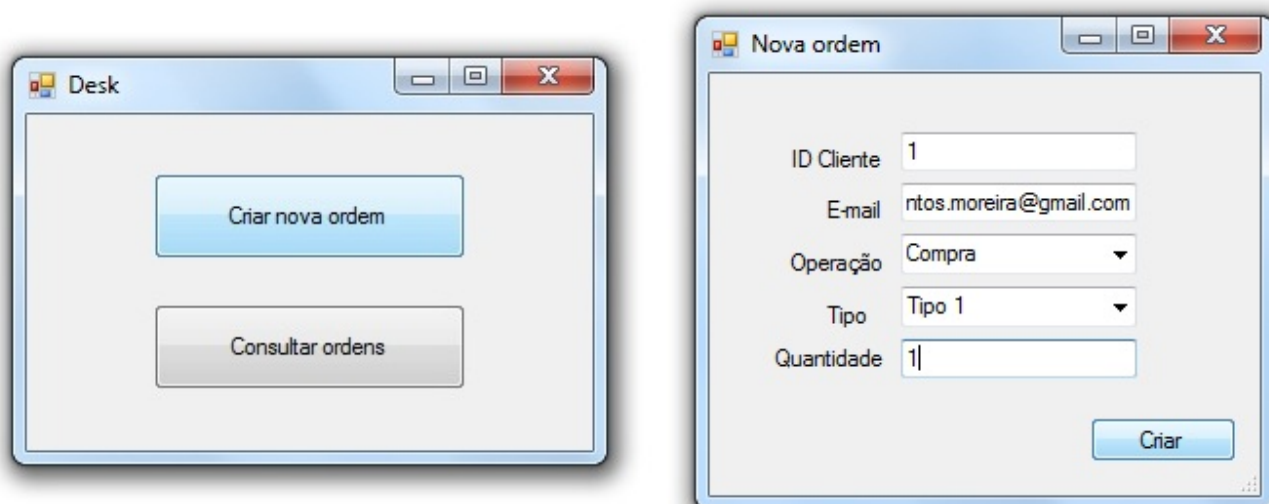
Apesar do exercício não representar o funcionamento real de um serviço bolsista, os conhecimentos adquiridos com o trabalho traduzem-se numa mais-valia e poderiam ser facilmente utilizados na sua expansão ou criação de outros cenários reais.

Exemplos de utilização

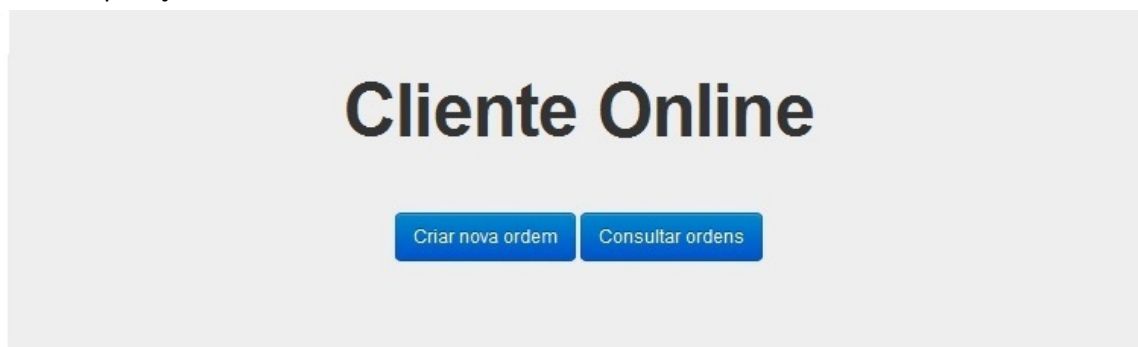
Criar nova ordem

Para criar uma nova ordem, podemos fazê-lo na aplicação desktop ou na aplicação web tal como é demonstrado nos seguintes exemplos de utilização.

- Aplicação Desktop



- Aplicação Web



Criar nova ordem

ID do cliente:

Email:

Operação: ☐ Compra ☐ Venda

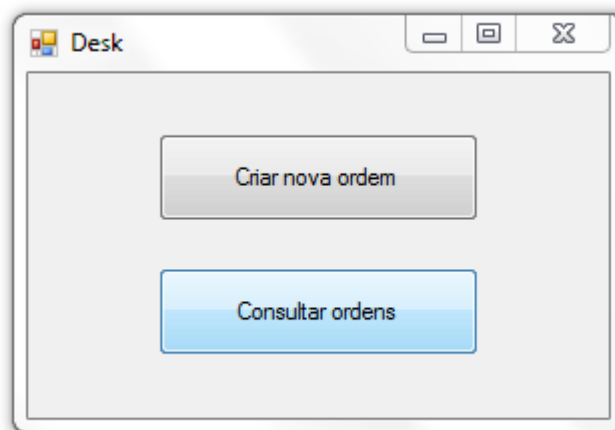
Tipo: ☐ Tipo 1 ☐ Tipo 2

Quantidade:

Em ambas as aplicações, o procedimento é semelhante. Começamos por escolher a opção “Criar nova ordem” que irá dar origem a um segundo ecrã/página. Nesta preenchemos os vários campos para a nova ordem e terminamos o processo de criação escolhendo a opção “Criar”.

Consultar ordens já existentes

- Aplicação Desktop



Consultar estado

ID do cliente:

Data	Email	Operação	Tipo	Quantidade	Cotação	Valor	Estado
20-05-2012 15...	heldersantos.moreira@...	Compra	Preferencial	1	12,0000	12,0000	Executado
28-05-2012 17...	heldersantos.moreira@...	Compra	Ordinária	1	5,0000	5,0000	Executado
28-05-2012 20...	heldersantos.moreira@...	Compra	Preferencial	5	-	-	Por executar

- Aplicação Web

Cliente Online

Consultar ordens

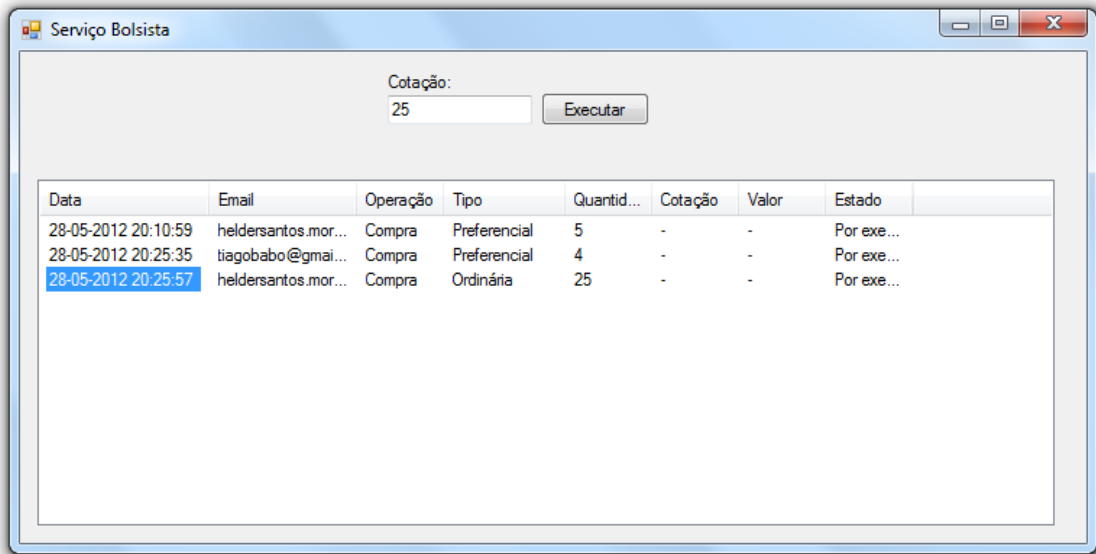
ID do cliente:

Atualizar

Data	E-Mail	Operação	Tipo	Quantidade	Cotação	Valor	Estado
20-05-2012 15:13:18	heldersantos.moreira@gmail.com	Compra	Preferencial	1	12,0000	12,0000	Executado
28-05-2012 17:17:20	heldersantos.moreira@gmail.com	Compra	Ordinária	1	5,0000	5,0000	Executado
28-05-2012 20:10:58	heldersantos.moreira@gmail.com	Compra	Preferencial	5	-	-	Por executar

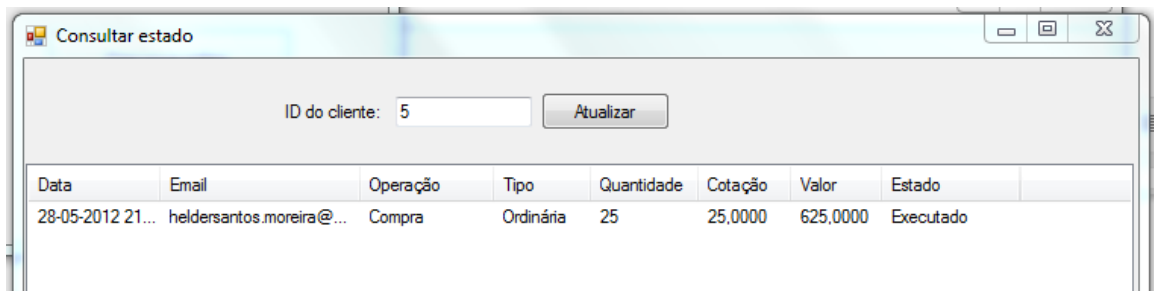
Em ambas as aplicações, o procedimento é semelhante. Começamos por escolher a opção “Consultar ordens” que irá dar origem a um segundo ecrã/página. Nesta preenchemos o id do cliente para o qual queremos consultar as ordens e clicamos na opção “Atualizar”. Será então mostrada uma lista com as ordens que foram encontradas para o id escolhido.

Executar uma ordem



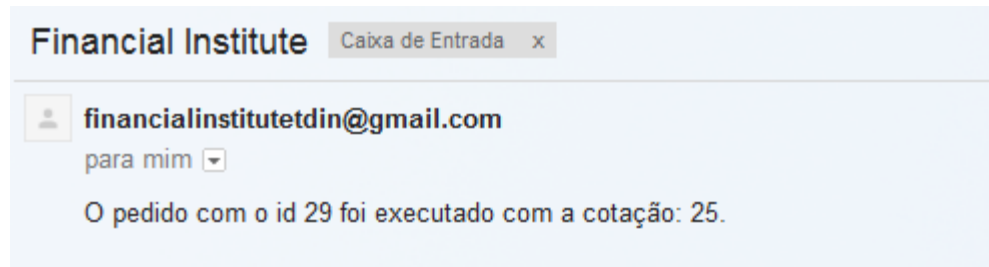
Data	Email	Operação	Tipo	Quantid...	Cotação	Valor	Estado
28-05-2012 20:10:59	heldersantos.mor...	Compra	Preferencial	5	-	-	Por exe...
28-05-2012 20:25:35	tiagobabo@gmail...	Compra	Preferencial	4	-	-	Por exe...
28-05-2012 20:25:57	heldersantos.mor...	Compra	Ordinária	25	-	-	Por exe...

Para executar uma ordem devemos, na aplicação do serviço bolsista, seleccionar uma das ordens que são apresentadas no ecrã. De seguida, introduzimos a cotação com a qual queremos executar a ordem e escolhemos a opção “Executar”.



Data	Email	Operação	Tipo	Quantidade	Cotação	Valor	Estado
28-05-2012 21:...	heldersantos.moreira@...	Compra	Ordinária	25	25,0000	625,0000	Executado

Podemos depois então verificar que consultando esta ordem, ela aparece já como executada e com o valor de cotação e valor atual atualizados. O mesmo resultado seria visível no website.



É também possível verificar que tal como era esperado, foi enviado pelo servidor um email, notificando o utilizador da execução da sua ordem.