



Universidade do Porto  
Faculdade de Engenharia  
**FEUP**

## **Tecnologias de Distribuição e Integração**

### **Trabalho prático #2 – Sistema distribuído de uma livraria**

Francisco Miguel Amaro Maciel - 201100692

Hugo Miguel Ribeiro de Sousa – 201100690

Ricardo Daniel Soares da Silva - 201108043

3 de Junho de 2015

## Índice

Introdução.....	3
Problema .....	3
Solução.....	4
Store.....	4
OrderStore .....	4
OrderSite .....	4
StoreGUI .....	4
ReceiptConsole .....	4
Warehouse .....	5
WarehouseService .....	5
WarehouseServer .....	5
WarehouseGUI .....	5
Interfaces / Cenários de Uso .....	6
Funcionalidades Implementadas .....	11
Instalação / Utilização .....	12
Conclusão .....	12

## Introdução

Este trabalho foi realizado no âmbito da unidade curricular de “Tecnologias de Distribuição e Integração” do Mestrado Integrado de Engenharia Informática e Computação. Consiste na elaboração de um sistema distribuído de uma livraria que se encontra dividida em duas localizações físicas que necessitam de comunicar entre elas: a loja e o armazém. Deve ainda ser possível encomendar livros através de um *website*. O objetivo deste trabalho é implementar um sistema distribuído aplicando os princípios SOA (Service Oriented Architecture), ou seja, as funcionalidades das aplicações devem ser disponibilizadas através de serviços.

## Problema

A empresa encontra-se dividida fisicamente em dois espaços: a loja (livraria) e o armazém. É possível ainda fazer encomendas *online*, que são processadas pela loja. Assume-se que o servidor da loja está sempre *online*, mas que o armazém apenas se encontra ligado nas horas de expediente.

A partir da loja, os clientes podem comprar livros. Caso este exista em stock, a venda é processada e é emitido um recibo com os dados do cliente. Caso contrário, é feito um pedido ao armazém para repor o stock em 10x a quantidade pedida pelo cliente. No armazém, assume-se por simplicidade do problema, que o stock é infinito. No armazém, aquando do envio dos livros, um trabalhador deve informar através da GUI o seu envio, atualizando o estado da encomenda. Quando os livros chegam à loja, o trabalhador deve também atualizar os novos stocks e o estado da encomenda. As ordens pendentes com os livros cujos stocks foram atualizados devem ser notificados via *e-mail*.

Nas encomendas online, o mesmo procedimento efetuado, mas o cliente é notificado por *e-mail* acerca do estado desta. O cliente pode também verificar o estado da sua encomenda no *website*, através do identificador único desta.

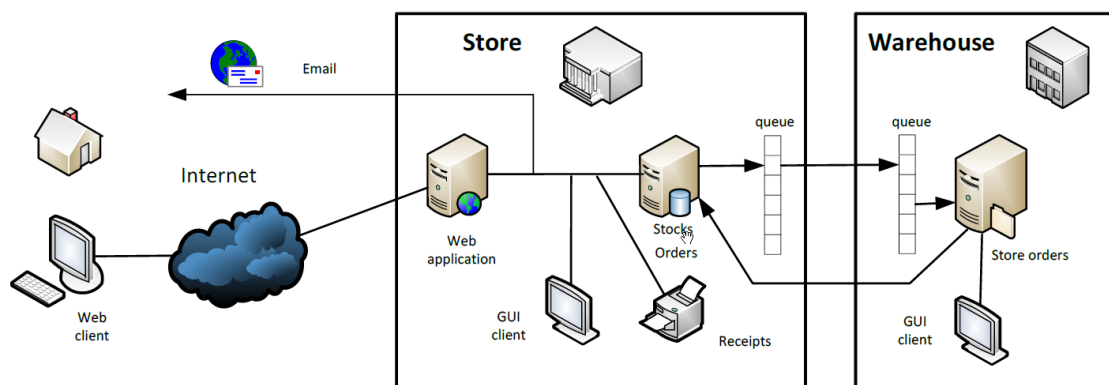


Fig. 1 – Representação esquematizada do sistema distribuído a implementar.

## Solução

O projeto foi realizado usando tecnologias .NET, com recurso à linguagem C# e serviços WCF. A solução é composta por 7 projetos (4 relativas à loja e 3 ao armazém), cujas funcionalidades serão descritas abaixo.

### Store

#### *OrderStore*

Projeto que implementa o serviço das funcionalidades da loja, tal como atualização de stocks, processamento de vendas, listagem de encomendas e livros, entre outros. Este serviço é duplex, por dois motivos:

- Atualização automática da GUI, após processamento de vendas ou atualização de stocks, sem necessidade de a fazer *refresh* manualmente.
- Impressão dos recibos é feita através de *callback*, ou seja, após o trabalhador escolher qual das impressoras em que pretende imprimir o recibo da venda, é chamado um método nessa impressora apenas.

#### *OrderSite*

Implementa o *website/webserver* em ASP.NET. Este website é composto por duas páginas: uma para criar uma encomenda e outra para rastrear o estado de uma encomenda já efetuada. A base de dados da loja é armazenada neste projeto e são usados os serviços referidos em OrderStore.

#### *StoreGUI*

Interface gráfica da loja. É possível, através desta, a realização de uma venda (ou pedido de encomenda, caso não haja stock suficiente) e atualização de stocks.

#### *ReceiptConsole*

Este projeto pretende simular uma impressora. No início da sua execução é pedido para atribuir um nome à impressora. A impressora regista-se no serviço. Durante uma venda na loja, é perguntado ao trabalhador em qual das impressoras pretende imprimir o recibo, caso alguma impressora esteja ligada. O serviço chama então o método de impressão na impressora respetiva, através de *callback*.

## Warehouse

### *WarehouseService*

Implementa o serviço usado no armazém. Este serviço lista as encomendas pendentes, adiciona novas encomendas à base de dados e atualiza os pedidos aquando do envio para a loja. Também este serviço é duplex, de forma a que a GUI seja automaticamente atualizada, sem necessidade de fazer *refresh* manualmente.

### *WarehouseServer*

Este servidor trata apenas de ler a lista de mensagens, de forma síncrona, invocando o serviço anterior para adicionar o pedido de encomenda quando uma nova mensagem é lida. Assim sendo, para obter novos pedidos de encomenda, este servidor deverá ser inicializado, mas é independente da GUI.

### *WarehouseGUI*

Interface gráfica do armazém. Nesta interface é possível verificar os pedidos pendentes, qual o livro e as quantidades associadas, assim como os tempos de envio e receção da respetiva encomenda. É possível proceder ao envio de um pedido, através de um único clique, que atualiza o estado da encomenda, tanto no armazém como na loja.

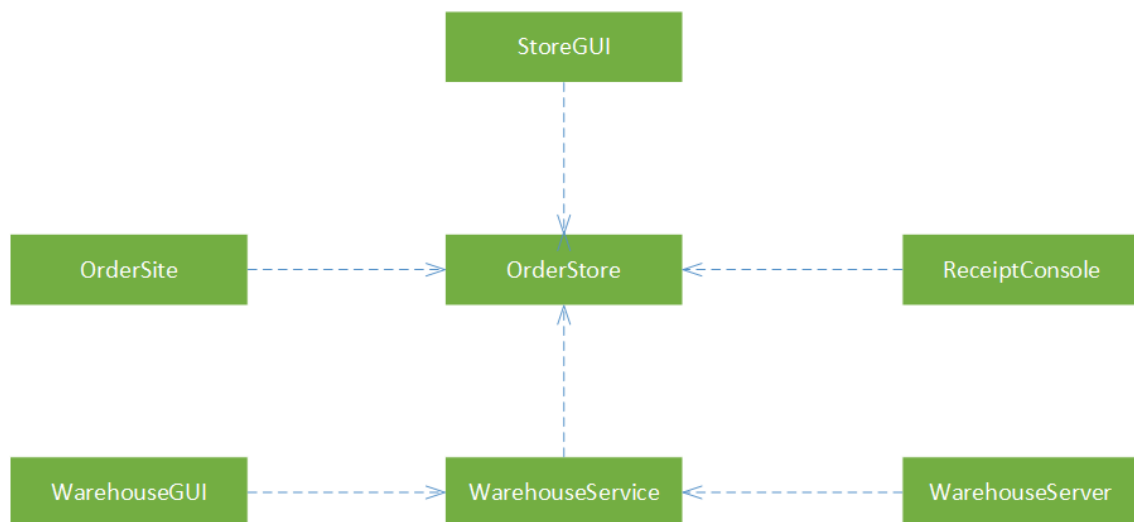


Fig. 2 – Dependências entre os diferentes projetos da solução.

## Interfaces / Cenários de Uso

The screenshot shows a window titled "Book Store" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are two tabs: "Sell" (which is active) and "Add Stock".

Under the "Sell" tab, there is a table with three columns: "Title", "Stock", and "Price". The table contains three rows of data:

Title	Stock	Price
Titulo1	0	10.5
Titulo2	3	12.3
Titulo3	0	8.99

Below the table, there are three input fields for transaction details:

- "Client Name:" followed by a text input box.
- "Quantity:" followed by a text input box.
- "Price:" followed by a text input box.

At the bottom of the window, there is a large, light gray button labeled "Sell".

Fig. 3 – Interface da loja, relativa à venda de livros.

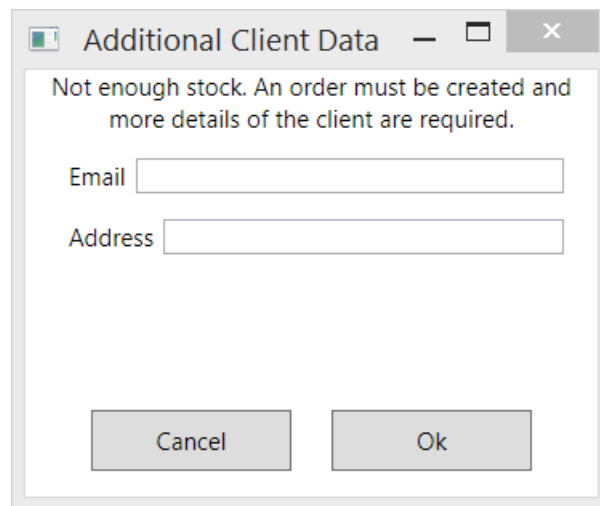


Fig. 4 – Interface que surge a requisitar informações adicionais do cliente, no caso de stock insuficiente.

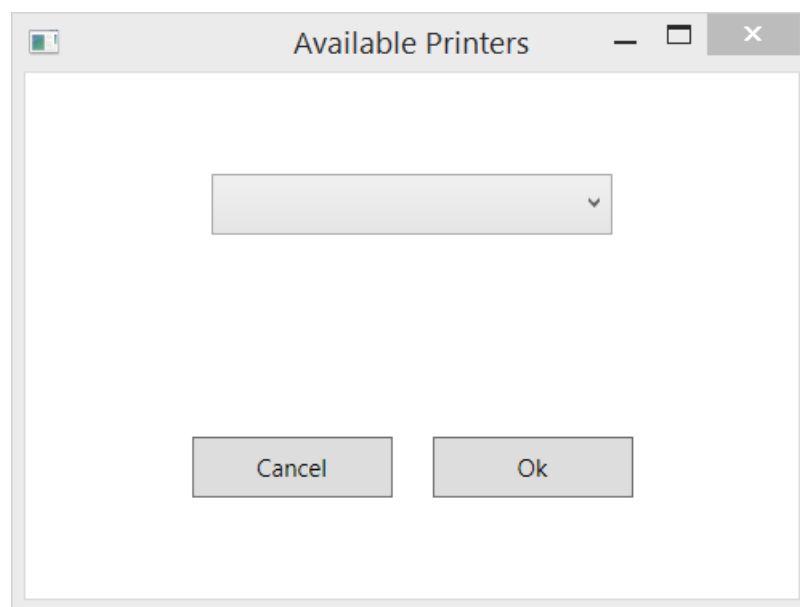
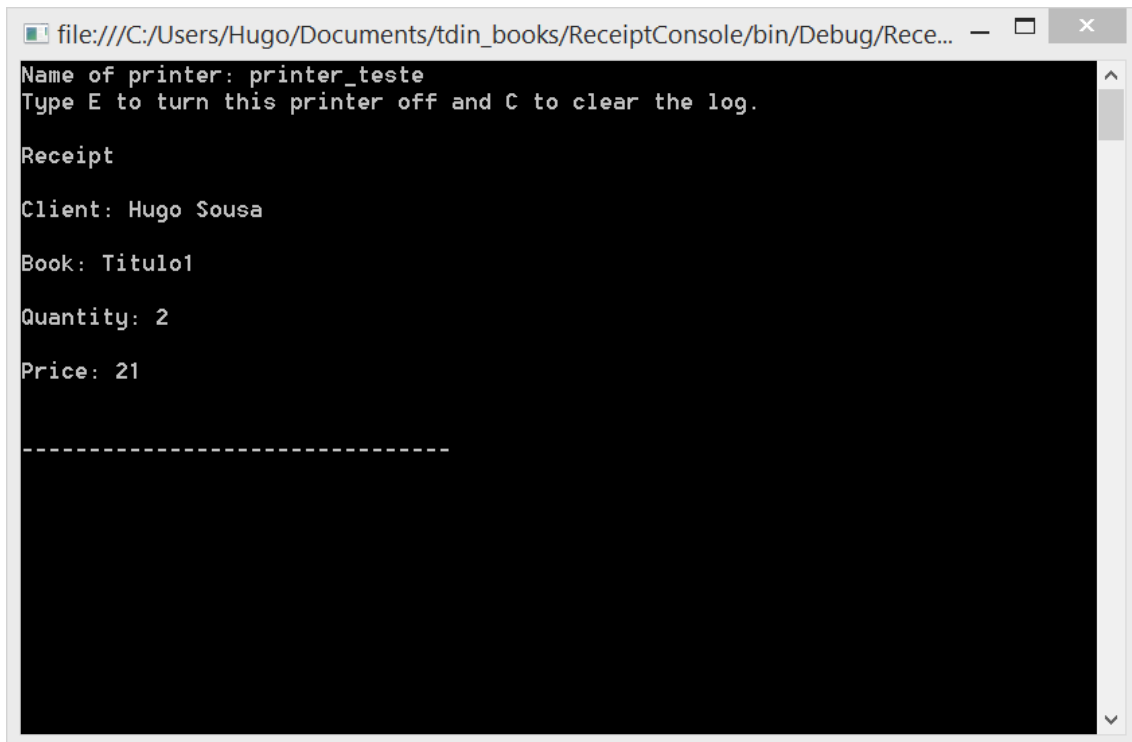


Fig. 5 – Interface que surge após uma venda bem sucedida e no caso de haverem impressoras disponíveis.



```
file:///C:/Users/Hugo/Documents/tdin_books/ReceiptConsole/bin/Debug/Rece...
Name of printer: printer_teste
Type E to turn this printer off and C to clear the log.

Receipt

Client: Hugo Sousa
Book: Titulo1
Quantity: 2
Price: 21

-----
```

Fig. 6 – Exemplo de impressão de recibo de uma venda numa impressora.



Book Store

Sell

Add Stock

Title	Stock	Price
Titulo1	0	10.5
Titulo2	3	12.3
Titulo3	0	8.99

Quantity:

Add Stock

Fig. 7 – Interface da loja, relativa à reposição de stock.

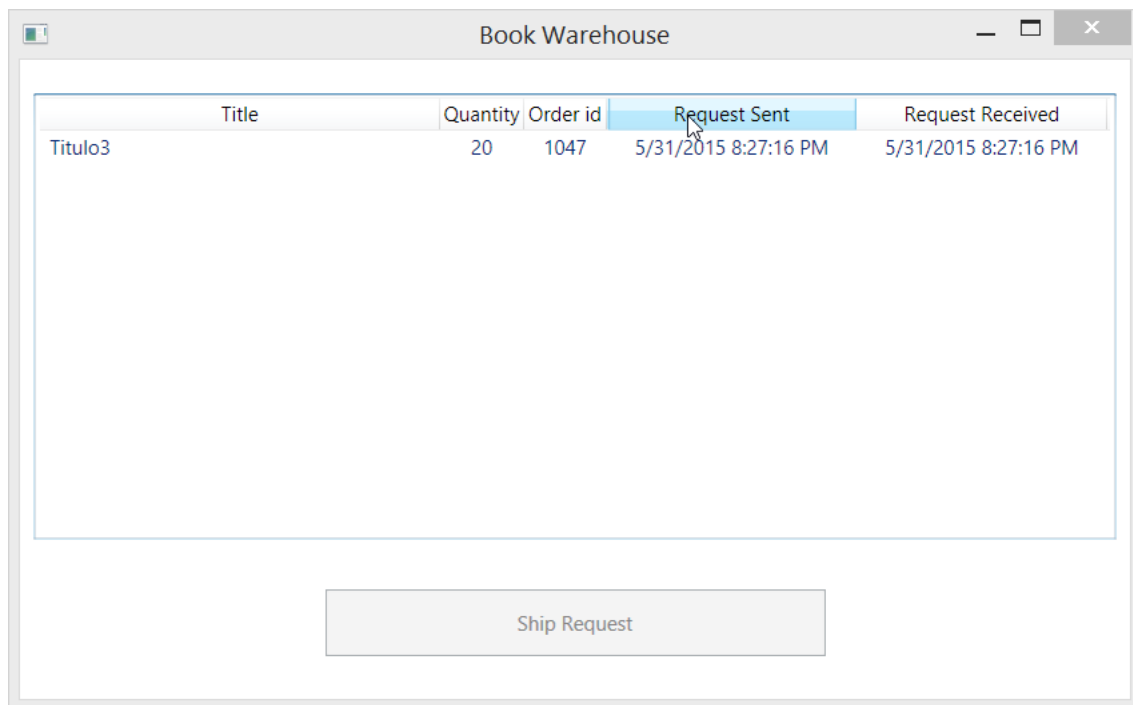


Fig. 8 – Interface do armazém.

## Order an Item

Title	Stock	Price
Titulo1	8	10.5
Titulo2	22	12.3
Titulo3	0	8.99

Quantity

[Already have an Order? Check their state!](#)

Fig. 9 – Página do website onde é possível encomendar um livro.

## Track Order

Track Order

Order Number : 1046

**Book**

Name	Quantity	Price
Titulo2	1	12.3

**Client**

Name	Address	Email
Rui Castro	Lugar de Teste, 122	77hugosousa@gmail.com

Dispatched at 01/06/2015

Fig. 10 – Página do website onde é possível verificar o estado de uma encomenda.

## Funcionalidades Implementadas

Todas as funcionalidades básicas requeridas no enunciado foram implementadas, das quais se enunciam:

<i>Funcionalidade</i>	<i>Teste</i>
<i>Venda de livro</i>	Na GUI da loja, após preencher os dados do cliente, vender livro numa quantidade inferior ao seu stock. A venda deve ser bem sucedida. Caso haja impressoras disponíveis, deve aparecer um <i>dialog</i> para escolher a impressora desejada.
<i>Encomenda de livro a partir da loja</i>	Na GUI da loja, após preencher os dados do cliente, vender livro numa quantidade superior ao seu stock. Deverá aparecer um <i>dialog</i> , para preencher com mais dados do cliente, para se poder proceder com a encomenda.
<i>Atualização de stock manual na loja</i>	Na GUI da loja, selecionar a <i>tab</i> “Add Stock”, selecionar um livro e a quantidade de stock a adicionar e clicar no botão “Add Stock”. O stock é atualizado, tentando satisfazer as encomendas possíveis.
<i>Encomenda de livro a partir do website</i>	Na página default do <i>website</i> , selecionar o livro pretendido e os restantes dados. A encomenda deve ser encaminhada e um id informado ao comprador, para que possa rastrear a sua encomenda.

<i>Tracking da encomenda no website</i>	Na página /TrackOrder.aspx do website, preencher o id da encomenda que se pretender rastrear. Caso seja um identificador válido, devem surgir os dados e o estado da encomenda respetiva.
<i>Leitura de mensagens provenientes da loja</i>	Executar o servidor do armazém. Este deverá ler as mensagens da fila de espera sincronamente e adicioná-las à base de dados do armazém. Na consola, surge a indicação destes eventos.
<i>Embarque de encomendas a partir do armazém</i>	Na GUI do armazém, selecionar a encomenda pretendida e clicar "Ship Request". O estado da encomenda deve ser atualizado.

## Instalação / Utilização

O sistema MSMQ deve ser ativado no sistema operativo, e deve ser criada manualmente uma fila de mensagens privada com o nome *warehouse\_books*.

Os projetos *OrderSite* e *WarehouseService* devem estar a correr no IIS. Para evitar a configuração manual de portas (5440 até 5444) que são usadas para comunicação de canal duplex dos serviços, deve-se executar as aplicações (ou o Visual Studio, caso seja executado a partir deste) em modo de administrador. Caso não seja corrido em modo de administrador, nem as portas sejam abertas manualmente, serão lançadas exceções relativas a estas portas.

No lado do armazém, caso o servidor não esteja a correr, as mensagens da fila não serão lidas e não chegarão novas encomendas, pelo que esta aplicação deve estar a correr simultaneamente com a GUI.

## Conclusão

Uma das principais dificuldades que sentimos na realização deste projeto foi a configuração de serviços WCF. Por vezes, pormenores podem resultar em bastante tempo perdido a tentar perceber o que se encontra de errado nas configurações destes serviços.

Este projeto foi interessante de realizar. Por um lado, por ser uma solução distribuída com bastantes projetos e fazer com que comuniquem se entendam. Por outro, por ser uma abordagem a tecnologias interessantes como o WCF e .ASP NET, sendo a primeira experiência para nós.