

## **Relatório**

# **“Libretto”**

*Enterprise Distributed System*

### **Grupo 10**

Carlos Manuel Carvalho Boavista Samouco

up201305187@fe.up.pt

Diogo Belarmino Coelho Marques

up201305642@fe.up.pt

José Alexandre Barreira Santos Teixeira

up201303930@fe.up.pt

22 de Maio de 2017

# Índice

<b>1</b>		Aplicação	2
1.1		<i>LibrettoClient</i>	3
1.2		<i>LibrettoInvoice</i>	4
1.3		<i>WarehouseClient</i>	6
1.4		<i>LibrettoOnline</i> (loja online)	8
<b>2</b>		Arquitetura	9
<b>4</b>		Serviços	10
4.1		<i>IBookstoreService</i>	10
4.2		<i>IWebstoreService</i>	15
4.3		<i>IWarehouseService</i>	18
4.4		<i>IBookstoreRemoting</i>	19
4.5		<i>IWarehouseRemoting</i>	19
<b>5</b>		Diagramas UML	21
5.1		<i>LibrettoClient</i>	22
5.2		<i>LibrettoCommon</i>	23
5.3		<i>LibrettoWCF</i>	24
5.4		<i>LibrettoInvoice</i>	25
5.5		<i>WarehouseClient</i>	26
5.6		<i>WarehouseServer</i>	26
<b>6</b>		Conclusão	27
<b>7</b>		Recursos	28
7.1		Referências bibliográficas	28
7.2		Software utilizado	28
<b>A</b>		Anexos	29

# Resumo

Este trabalho foi desenvolvido no âmbito da unidade curricular *Tecnologias de Distribuição e Integração* do Mestrado Integrado em Engenharia Informática e Computação (MIEIC) da Faculdade de Engenharia da Universidade do Porto (FEUP), e pretendeu-se desenvolver um sistema de facturação e gestão de stock baseado em *.NET WCF, Remoting e MSMQ*, na linguagem C# com interface gráfica (GUI) nos clientes utilizando Windows Forms. O sistema consiste em dois servidores, em que cada servidor é responsável por uma base de dados para armazenamento de informação relativa a vendas e encomendas. O sistema possui ainda três aplicações cliente com interface gráfica de *Windows Forms* para a gestão de stocks da loja e registo de vendas e encomendas, para a gestão de pedidos de encomendas e fornecimento de stock e para o registo das faturas relativas às vendas. Existe ainda uma aplicação web, que permite a clientes registarem-se e autenticarem-se na loja para poderem consultar os livros disponíveis e efetuar pedidos de compra e encomenda.

## 1 Aplicação

Uma editora vende livros nas suas instalações. Pretende desenvolver um sistema para coordenar as suas vendas, encomendas e realizar gestão de *stocks*. A editora é proprietária de duas instalações situadas em locais distintos: uma livraria com uma zona de exposição ao público, e um armazém para guardar livros em maiores quantidades. A editora quer ainda disponibilizar uma aplicação *web* que permita aos seus clientes consultar e encomendar livros.

### 1.1 *LibrettoClient*

Esta aplicação com *interface* gráfica em *Windows Forms*, operada por um funcionário ou pelo proprietário da mesma, permite a venda de livros disponíveis na loja a um cliente visitante, bem como a criação de uma encomenda (semelhante a uma encomenda realizada *online*) se esse livro apenas existir em *stock* no armazém. Uma venda guarda também o nome do cliente, o título do livro, o código de identificação único do cliente, o código de identificação único do livro, quantidade adquirida e valor total da compra, atualiza o *stock* na livraria e imprime uma fatura numa aplicação separada, que simula uma fila de impressão.

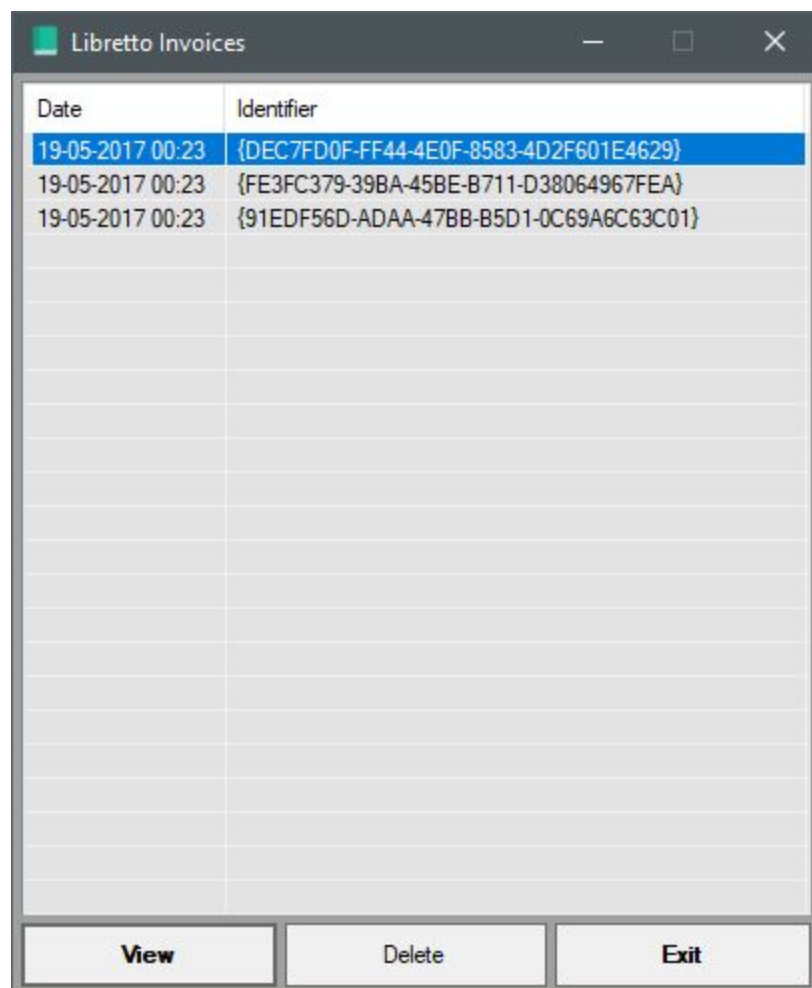
Quando um livro estiver esgotado na loja deve ser criada uma encomenda. As encomendas (com origem na loja *online* ou na aplicação GUI da loja pode apresentar um de três estados:

- **Waiting** (aguarda envio)  
*“waiting for expedition”*
- **Pending** (aguarda confirmação, mas será enviada em breve)  
*“dispatch will occur at ... (date)”*
- **Dispatched** (encomenda entregue)  
*“dispatch at ... (date)”*.

Quando a livraria recebe um pedido com os livros do armazém para determinada encomenda, um funcionário da mesma deve aceitá-los na *interface* da aplicação da loja, atualizando o *stock* do livro na loja e mudando o estado desta encomenda para **Dispatched**. Nesta situação é enviado um outro e-mail ao cliente.

## 1.2 LibrettoInvoice

Esta aplicação com *interface* gráfica em *Windows Forms* permite simular uma fila de impressão, implementando as funcionalidades mínimas. Permite ainda visualizar de forma intuitiva as faturas das compras realizadas pelos clientes à medida que estas vão sendo registadas no servidor da loja. Por ser implementado através de uma *message queue* privada, não é necessário que a aplicação se encontre em execução para receber as faturas enviadas pela loja. Pelo contrário, quando a aplicação é executada, a primeira ação realizada é inicializar a *message queue* `.\private$\InvoiceMsmq` (caso esta ainda não exista) ou estabelecer ligação com a existente. No caso em que a *message queue* já existe, a aplicação recebe todas as mensagens que foram publicadas na fila enquanto esta não esteve em execução, notificando a chegada das mesmas através de um sinal sonoro e um *toast* no canto inferior direito do ecrã.



Na janela principal da aplicação *LibrettoInvoice* encontramos uma lista com todas as faturas recebidas, bem como algumas das informações relativas às mesmas: código de identificação (GUID) e data de entrada no sistema. É possível realizar três ações nesta janela: **View**, que permite visualizar uma fatura com detalhe, abrindo uma nova janela que mostra de forma intuitiva todas as informações relevantes da mesma; **Delete**, que permite apagar uma ou mais faturas da fila de impressão; **Exit**, que permite sair da aplicação. É também possível visualizar uma fatura fazendo duplo clique sobre uma das faturas existentes na lista.



**View Invoice**

**Libretto**  
Rua Dr. Roberto Frias, s/n  
4200-391 Porto

**Invoice**

Identifier	{DEC7FD0F-FF44-4E0F-8583-4D2F601E4629}
Customer	Diogo Marques
Book	Dummy Book A

**Transaction**

Quantity	Status	Total
5	Purchased	9,99 €

**Date**

sexta-feira, 19 de maio de 2017 00:23:57

**Figura 2.** Janela de visualização de uma fatura, evidenciando as informações relevantes da mesma, tais como quantidade adquirida, valor total, nome do cliente, título do livro,

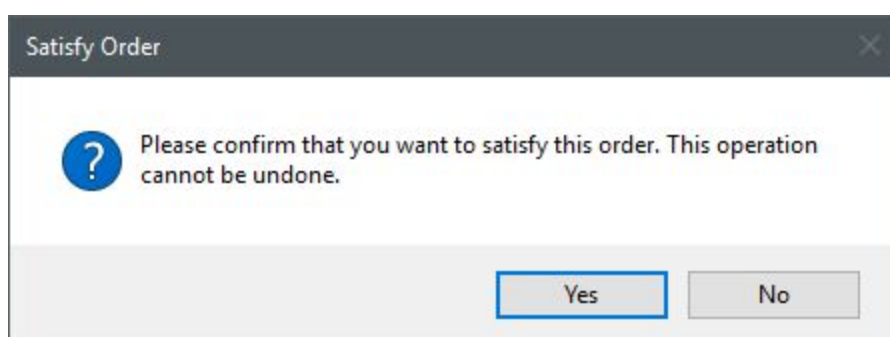
Na janela de visualização da fatura é possível consultar os detalhes da transação escolhida, tais como o código de identificação da compra, nome completo do cliente, título do livro adquirido, número de unidades adquiridas, valor total da compra, data de registo da compra no sistema. A aplicação persiste todas as faturas recebidas através da *message queue* em disco, guardando as transações num ficheiro XML, sendo que ao escolher a opção **Delete** na janela principal as faturas serão também apagadas deste ficheiro.

### 1.3 WarehouseClient

Esta aplicação com *interface* gráfica em *Windows Forms* permite a gestão das encomendas dos clientes que chegam ao armazém. Uma encomenda é registada no armazém sempre que a loja não possui *stock* suficiente para satisfazer a mesma, enviando um pedido pelo serviço *IWarehouseService* disponibilizado pelo servidor do armazém. A implementação deste serviço através de um *binding* do tipo *message queue* permite que as encomendas dêem entrada no servidor do armazém mesmo fora das suas horas de funcionamento, garantindo que não há perdas de informação se um dos servidores envolvidos na operação falhar.

[illegible]

Na janela principal da aplicação é possível visualizar uma listagem com todas as encomendas, filtrar encomendas por título do livro, ver as encomendas que chegaram ao servidor do armazém antes de determinada data, após determinada data e enquadrá-las entre duas datas. É possível realizar três ações nesta janela: **Satisfy Order**, que permite satisfazer um pedido, notificando o servidor da loja que em breve serão enviadas para a loja as unidades encomendadas no pedido; a ação **Refresh**, que permite atualizar manualmente a listagem das encomendas presentes no servidor do armazém (caso não seja realizada automaticamente); **Exit**, que permite abandonar a aplicação. É também possível satisfazer um pedido fazendo duplo clique sobre uma das encomendas existentes na lista.



**Figura 2.** Janela para confirmação da satisfação de um pedido no armazém.

Ao fazer **Satisfy Order** é apresentado um ecrã para confirmar a satisfação do pedido. A encomenda é removida da lista da aplicação, bem como do ficheiro XML que contém todas as encomendas que chegaram ao servidor do armazém. O seu estado no servidor da loja é atualizado para *Pending* e a sua data de entrega é definida para dois dias após a data em que a encomenda foi satisfeita. Os livros serão fisicamente transportados para a loja, assumindo que o armazém possui sempre a quantidade de livros pedida (*stock* infinito).

**NOTA:** Para esta aplicação funcionar corretamente, desempenhando funções mínimas, o servidor do armazém deve encontrar-se em execução. Para satisfazer uma encomenda de cliente é também necessário que o servidor da loja se encontre em execução, caso contrário será apresentada uma mensagem de erro ao utilizador.



## 1.4 *LibrettoOnline* (loja online)

Esta aplicação *web* foi desenvolvida na *framework* MVC (*model-view-controller*) Angular 2, de forma a simular uma loja *online* que permitisse aos clientes da livraria registarem um conta de utilizador, autenticarem-se, fazer encomendas/compras dos produtos disponibilizados pela mesma. Para além disso, permite ainda visualizar um histórico com todas as transações realizadas pelo cliente. A comunicação com o servidor da loja é assegurada através de uma API REST (*representational state transfer*) suportada por um serviço WCF.

Uma transação é efetuada a partir do momento em que um utilizador da aplicação *web* (depois de estar devidamente autenticado) seleciona um livro da lista de livros disponíveis e preenche um formulário que lhe é sugerido no *browser*, indicando se se trata de uma compra ou encomenda, bem como o número de unidades do livro que este deseja adquirir.

Ao autenticar-se é apresentada uma mensagem de boas-vindas ao utilizador no canto superior direito da página, juntamente com o seu nome. A partir desta barra de navegação é possível realizar uma de três ações: **Books**, que permite consultar um catálogo com os livros disponíveis para compra; **Transactions**, que permite ao utilizador consultar um histórico das suas transações, tal como foi referido anteriormente. Os utilizadores podem ainda terminar a sua sessão na aplicação *web* carregando no botão **Logout**.

## 2 Arquitetura

Na livraria existe um servidor que hospeda a aplicação, e disponibiliza uma série de serviços (tanto à aplicação *web* como à aplicação cliente da loja) e mantém um registo persistente dos livros disponíveis, preços e quantidade disponível (*stock*) na livraria. Este servidor encontra-se sempre ativo e ligado à Internet. Entre a loja e o armazém existe ainda uma ligação de rede, mas os computadores do armazém normalmente encontram-se ativos apenas nas horas de trabalho, incluindo o servidor.

O servidor da loja aceita encomendas de clientes pela Internet através da *web* app. Cada encomenda possui um título (assumindo que uma encomenda/aquisição diz respeito apenas a um livro), quantidade encomendada pelo cliente, nome do cliente, código de identificação do livro, código de identificação do cliente, valor total da encomenda. Para cada transação é também gerado um código de identificação único (GUID/UUID). Todas as transações registadas neste servidor são persistidas em disco e possuem um estado associado. Por razões de simplicidade, ignoramos o processo de pagamento neste cenário e consideramos um número relativamente reduzido de livros disponíveis.

O servidor do armazém recebe estas chamadas de forma assíncrona através de uma fila de mensagens (*message queue*) associada a um serviço WCF (utiliza *MsmqNetBinding*), persistindo localmente esta informação num ficheiro XML sempre que houver alguma alteração, para posterior consulta ou processamento (nas situações em que uma nova encomenda é registada no armazém e uma encomenda em curso é enviada/cancelada pelo cliente ou pelo funcionário da livraria. A persistência de dados foi implementada recorrendo à biblioteca de serialização XML nativa de objetos da linguagem C#.

## 3 Serviços

Esta secção do relatório servirá para descrever com detalhe os diferentes serviços presentes nas aplicações desenvolvidas, as funcionalidades implementadas, as tecnologias utilizadas (*Windows Communication Foundation*, *.NET Remoting*, *Message Queueing*, *REST*), bem como a contribuição individual dos serviços para a organização e correto funcionamento do sistema proposto.

### 3.1 *IBookstoreService*

Este serviço WCF implementado no **servidor** da loja *LibrettoWCF* define um conjunto de métodos invocados remotamente pela **aplicação cliente** da loja que permite ao proprietário (administrador) e aos funcionários da mesma a gestão do catálogo de livros, gerir clientes, registar e gerir compras, bem como registar e gerir as encomendas dos clientes. Este serviço confere as seguintes funcionalidades à aplicação da loja:

- Adicionar/remover livros
- Registar/remover compras
- Registar/remover encomendas
- Atualizar encomendas
- Cancelar encomendas
- Consultar clientes
- Consultar livros disponíveis
- Consultar compras/encomendas
- Autenticar funcionários/proprietário

### 3.1.1 Configuração

```
<service behaviorConfiguration="StoreServiceBehavior" name="LibrettoWCF.StoreService">
  <endpoint binding="netTcpBinding" bindingConfiguration="StoreTcpConf" name="TcpEndpoint"
    listenUriMode="Explicit" contract="LibrettoWCF.IStoreService" />
  <endpoint address="mex" binding="mexHttpBinding"
    name="ServiceInfo" contract="IMetadataExchange" />
  <host>
    <baseAddresses>
      <add baseAddress="http://localhost:8733/Design_Time_Addresses/LibrettoBookstore/" />
      <add baseAddress="net.tcp://localhost:9000/LibrettoBookstore/" />
    </baseAddresses>
  </host>
</service>
```

### 3.1.2 Behaviors

```
<behavior name="StoreServiceBehavior">
  <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
  <serviceDebug includeExceptionDetailInFaults="true" />
  <serviceCredentials>
    <serviceCertificate findValue="CN=FeupEnterprise" />
    <userNameAuthentication userNamePasswordValidationMode="Custom"
      customUserNamePasswordValidatorType="LibrettoWCF.Authentication.ClerkValidator,LibrettoWCF"
      />
  </serviceCredentials>
  <serviceAuthorization principalPermissionMode="Custom">
    <authorizationPolicies>
      <add policyType="LibrettoWCF.Authentication.AuthorizationPolicy,LibrettoWCF" />
    </authorizationPolicies>
  </serviceAuthorization>
</behavior>
```

### 3.1.3 Bindings

```
<netTcpBinding>
  <binding name="StoreTcpConf">
    <security mode="Message">
      <message clientCredentialType="UserName" />
    </security>
  </binding>
</netTcpBinding>
```

### 3.1.4 Contract <Authentication>

Clerk Profile()	
<b>Descrição</b>	Permite ao funcionário/administrador com sessão iniciada na aplicação cliente consultar as suas informações de perfil (nome completo, endereço de correio eletrónico). Permite ainda à aplicação identificar as permissões ( <i>role</i> ) desse utilizador, ativando/desativando certas funcionalidades nela própria.

### 3.1.5 Contract <Books>

List<Book> ListBooks()	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja obter uma lista com todos os livros disponíveis para venda.
Response InsertBook( <b>Book</b> bookInformation)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja lançar um novo livro.
<b>Parâmetros</b>	bookInformation. <b>Title</b> : título completo do livro bookInformation. <b>Price</b> : preço de venda recomendado bookInformation. <b>Stock</b> : unidades inicialmente em <i>stock</i> bookInformation. <b>Identifier</b> : código de identificação do livro (GUID)
Response UpdateBook( <b>Book</b> bookInformation)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja editar as informações relativas a determinado livro publicado na loja.
<b>Parâmetros</b>	bookInformation. <b>Title</b> : título completo do livro bookInformation. <b>Price</b> : preço de venda recomendado bookInformation. <b>Stock</b> : unidades inicialmente em <i>stock</i>
Response DeleteBook( <b>Guid</b> bookIdentifier)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja retirar um livro da loja.
<b>Parâmetros</b>	<b>bookIdentifier</b> : código de identificação do livro (GUID)

### 3.1.6 Contract <Customers>

List<Customer> ListCustomers()	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja obter uma lista com todos os clientes (tanto os registados na aplicação <i>web</i> como os clientes ocasionais da loja física).
Response InsertCustomer( <b>Customer</b> customerInformation)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja registar um novo cliente para a criação de uma compra/encomenda.
<b>Parâmetros</b>	customerInformation. <b>Email</b> : endereço de correio eletrónico do cliente customerInformation. <b>Identifier</b> : código de identificação do cliente (GUID) customerInformation. <b>Name</b> : nome completo (primeiro + último) do cliente customerInformation. <b>Location</b> : morada completa do cliente

### 3.1.7 Contract <Purchases>

List<Purchase> ListPurchases()	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja visualizar uma lista com todas as compras realizadas pelos clientes.
Purchase LookupPurchase( <b>Guid</b> purchaseIdentifier)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja consultar informações relativas a uma compra.
<b>Parâmetros</b>	<b>purchaseIdentifier</b> : código de identificação da compra (GUID)
Response DeletePurchase( <b>Guid</b> purchaseIdentifier)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja anular uma compra.
<b>Parâmetros</b>	<b>purchaseIdentifier</b> : código de identificação da compra (GUID)

<b>Response InsertPurchase(Purchase purchaseInformation)</b>	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja registar uma compra.
<b>Parâmetros</b>	<p>purchaseInformation.<b>BookId</b> : código de identificação do livro adquirido</p> <p>purchaseInformation.<b>BookTitle</b> : título completo do livro adquirido</p> <p>purchaseInformation.<b>CustomerId</b> : código de identificação do cliente</p> <p>purchaseInformation.<b>CustomerName</b> : nome completo do cliente</p> <p>purchaseInformation.<b>Quantity</b> : número de unidades adquiridas</p> <p>purchaseInformation.<b>Total</b> : valor total da compra (derivado de preço unitário do livro * número de unidades adquiridas pelo cliente)</p>

### 3.1.8 Contract <Orders>

<b>List&lt;Order&gt; ListOrders()</b>	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja obter uma lista com todas as encomendas realizadas pelos clientes.
<b>Order LookupOrder(Guid orderIdentifier)</b>	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja consultar informações relativas a uma encomenda de cliente.
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)
<b>Response InsertOrder(OrderTemplate orderForm)</b>	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja registar uma encomenda de cliente.
<b>Parâmetros</b>	<p>orderForm.<b>BookId</b> : código de identificação do livro encomendado (GUID)</p> <p>orderForm.<b>BookTitle</b> : título completo do livro encomendado</p> <p>orderForm.<b>CustomerId</b> : código de identificação do cliente (GUID)</p> <p>orderForm.<b>CustomerName</b> : nome completo do cliente</p> <p>orderForm.<b>Quantity</b> : número de unidades encomendadas</p> <p>orderForm.<b>Total</b> : valor total da encomenda (derivado de preço unitário do livro * número de unidades encomendadas pelo cliente)</p>
<b>Response CancelOrder(Guid orderIdentifier)</b>	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja cancelar uma encomenda de cliente (alterar estado para <i>CANCELLED</i> ).
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)

Response DeleteOrder(Guid orderIdentifier)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja apagar de forma permanente uma encomenda de cliente.
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)
Response DispatchOrder(Order orderInformation)	
<b>Descrição</b>	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja aceitar a chegada das unidades do armazém associado a uma encomenda.
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)

## 3.2 IWebstoreService

Este serviço WCF REST (*representational state transfer*) implementado no **servidor** da loja *LibrettoWCF* define um conjunto de métodos invocados pelo protocolo HTTP na aplicação *web* (loja *online*) que conferem as seguintes funcionalidades à mesma:

- Registar utilizador
- Autenticar utilizador
- Realizar compras
- Realizar encomendas
- Cancelar encomendas
- Consultar livros disponíveis na loja *online*
- Consultar informações relativas às compras/encomendas efetuadas

### 3.2.1 Behaviors

```
<behavior name="WebstoreServiceBehavior">
  <serviceMetadata httpGetEnabled="True" httpsGetEnabled="True" />
  <serviceDebug includeExceptionDetailInFaults="False" />
</behavior>
```



### 3.2.2 Endpoints

```
<service behaviorConfiguration="WebstoreServiceBehavior" name="LibrettoWCF.WebstoreService">
  <endpoint address="" behaviorConfiguration="restfulBehavior"
    binding="webHttpBinding" contract="LibrettoWCF.IWebstoreService">
    <identity>
      <dns value="localhost" />
    </identity>
  </endpoint>
  <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
  <host>
    <baseAddresses>
      <add
baseAddress="http://localhost:8733/Design_Time_Addresses/LibrettoWebstore/ServiceProvider.s
vc/" />
      </add>
    </baseAddresses>
  </host>
</service>
```

### 3.2.3 Contract <Authentication>

Customer Login(LoginTemplate loginForm)	
Descrição	Permite ao cliente autenticar-se na aplicação <i>web</i> .
Parâmetros	<b>loginForm</b> : objeto contendo as credenciais (endereço de correio eletrónico, <i>password</i> ) do utilizador

### 3.2.4 Contract <Purchases>

Response InsertPurchase(PurchaseTemplate purchaseForm)	
Descrição	Permite ao cliente com sessão iniciada na loja <i>online</i> realizar uma compra.
Parâmetros	purchaseForm. <b>BookId</b> : código de identificação do livro adquirido purchaseForm. <b>BookTitle</b> : título completo do livro adquirido purchaseForm. <b>CustomerId</b> : código de identificação do cliente purchaseForm. <b>CustomerName</b> : nome completo do cliente purchaseForm. <b>Quantity</b> : número de unidades adquiridas pelo cliente purchaseForm. <b>Total</b> : valor total da compra (derivado de preço unitário do livro * número de unidades adquiridas pelo cliente)

<b>List&lt;Purchase&gt; GetPurchasesByUser()</b>	
<b>Descrição</b>	Permite ao cliente com sessão iniciada na aplicação <i>web</i> visualizar uma lista com todas as compras que realizou, quer na loja <i>online</i> , quer na loja física.

### 3.2.5 Contract <Purchases>

<b>List&lt;Order&gt; GetOrdersByUser()</b>	
<b>Descrição</b>	Permite ao cliente com sessão iniciada na aplicação <i>web</i> visualizar uma lista com todas as encomendas efetuadas, quer na loja <i>online</i> , quer na loja física.
<b>Response InsertOrder(OrderTemplate orderForm)</b>	
<b>Descrição</b>	Permite ao cliente com sessão iniciada na aplicação <i>web</i> realizar uma encomenda.
<b>Parâmetros</b>	orderForm. <b>BookId</b> : código de identificação do livro encomendado orderForm. <b>BookTitle</b> : título completo do livro encomendado orderForm. <b>CustomerId</b> : código de identificação do cliente orderForm. <b>CustomerName</b> : nome completo do cliente orderForm. <b>Quantity</b> : número de unidades encomendadas orderForm. <b>Total</b> : valor total da encomenda (derivado de preço unitário do livro * número de unidades encomendadas pelo cliente)
<b>Response DeleteOrder(Guid orderIdentifier)</b>	
<b>Descrição</b>	Permite ao cliente com sessão iniciada na aplicação <i>web</i> cancelar uma encomenda que realizou (cujo pedido ainda não tenha sido satisfeito).
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda

### 3.2.6 Contract <Books>

<b>List&lt;Book&gt; ListBooks()</b>	
<b>Descrição</b>	Permite ao cliente com sessão iniciada na aplicação <i>web</i> visualizar uma lista com todos os livros disponíveis na loja <i>online</i> .

### 3.3 *IWarehouseService*

Este serviço implementado na **aplicação servidor** do armazém (*WarehouseServer*) define um conjunto de métodos que processam as mensagens recebidas pela *message queue* estabelecida entre este e o servidor da loja. Estas mensagens estão relacionadas com a gestão das encomendas dos clientes, permitindo ao servidor da loja notificar o servidor do armazém sempre que uma nova encomenda for registada no sistema, atualizada ou cancelada.

#### 3.3.1 *Bindings*

```
<bindings>
  <netMsmqBinding>
    <binding name="NoMsmqSecurity">
      <security mode="None" />
    </binding>
  </netMsmqBinding>
</bindings>
```

#### 3.3.2 *Endpoints*

```
<service name="Libretto.WarehouseService" behaviorConfiguration="WarehouseBehavior">
  <host>
    <baseAddresses>
      <add baseAddress="http://localhost:8733/Design_Time_Addresses/LibrettoWarehouse/" />
      <add baseAddress="net.msmq://localhost/private/warehouseMsmq" />
    </baseAddresses>
  </host>
  <endpoint address="" binding="netMsmqBinding" contract="Libretto.IWarehouseService"
bindingConfiguration="NoMsmqSecurity" />
  <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
</service>
```

#### 3.3.3 *Contract*

void CancelOrder( <b>MessageCancel</b> messageCancel)	
<b>Descrição</b>	Permite processar uma mensagem do tipo <i>MessageCancel</i> , para informar que determinada encomenda foi cancelada pelo cliente ou pelo funcionário.
<b>Parâmetros</b>	<b>orderIdIdentifier</b> : código identificador da encomenda (GUID)

<b>void InsertOrder(WarehouseOrder warehouseOrder)</b>	
<b>Descrição</b>	Permite processar uma mensagem do tipo <i>WarehouseOrder</i> , para informar que uma nova encomenda foi registada no sistema, realizada quer por um cliente na aplicação <i>web</i> , quer por um funcionário na aplicação da loja.
<b>Parâmetros</b>	warehouseOrder. <b>DateCreated</b> : data de registo da encomenda warehouseOrder. <b>DateModified</b> : data da última atualização warehouseOrder. <b>Identifier</b> : código identificador da encomenda (GUID) warehouseOrder. <b>Status</b> : estado da encomenda warehouseOrder. <b>Title</b> : nome do livro encomendado warehouseOrder. <b>Total</b> : valor total da encomenda (derivado de preço unitário do livro * número de unidades encomendadas pelo cliente)

### 3.4 IBookstoreRemoting

Esta *interface* implementada na **aplicação servidor** da loja (*BookstoreServer*) e invocada pela **aplicação servidor** do armazém (*WarehouseServer*) define um conjunto de métodos invocados remotamente que permitem ao servidor do armazém gerir as encomendas existentes na loja, estabelecendo um meio de comunicação entre ambos.

<b>bool DispatchOrder(Guid orderIdentifier)</b>	
<b>Descrição</b>	Permite ao servidor armazém satisfazer uma encomenda, notificando o servidor da loja assim que os produtos da encomenda estiverem prontos a serem enviados.
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)

### 3.5 IWarehouseRemoting

Esta *interface* implementada na **aplicação cliente** do armazém (*WarehouseClient*) e invocada pela **aplicação servidor** do armazém (*WarehouseServer*) define um conjunto de métodos invocados remotamente que permitem à aplicação realizar operações relativas à gestão das encomendas existentes no servidor do armazém.

### 3.5.1 Eventos

<b>event WarehouseUpsertHandler OnUpsert</b>
Evento subscrito pela aplicação cliente do armazém para receber notificações sempre que uma encomenda chega ao servidor do armazém. Ao ser invocado remotamente permite adicionar a encomenda à lista de encomendas presente na <i>interface</i> da mesma.
<b>event WarehouseDeleteHandler OnDelete</b>
Evento subscrito pela aplicação cliente do armazém para receber notificações sempre que uma encomenda é cancelada ou removida do servidor do armazém, quer através da loja <i>online</i> , quer através de um funcionário a utilizar a aplicação da loja. Ao ser invocado permite remover elementos da lista de encomendas presente na <i>interface</i> da mesma.

### 3.5.2 Métodos

<b>List&lt;WarehouseOrder&gt; ListOrders()</b>	
<b>Descrição</b>	Permite à aplicação cliente do armazém obter uma lista atualizada com todas as encomendas que chegaram ao servidor do armazém.
<b>bool DispatchOrder(Guid orderIdentifier)</b>	
<b>Descrição</b>	Permite à aplicação cliente do armazém satisfazer uma encomenda, notificando o servidor do armazém assim que os produtos da encomenda estiverem prontos a serem enviados. Posteriormente, o servidor do armazém notifica a loja invocando um método numa segunda <i>interface</i> remota.
<b>Parâmetros</b>	<b>orderIdentifier</b> : código de identificação da encomenda (GUID)

## 4 Diagramas UML

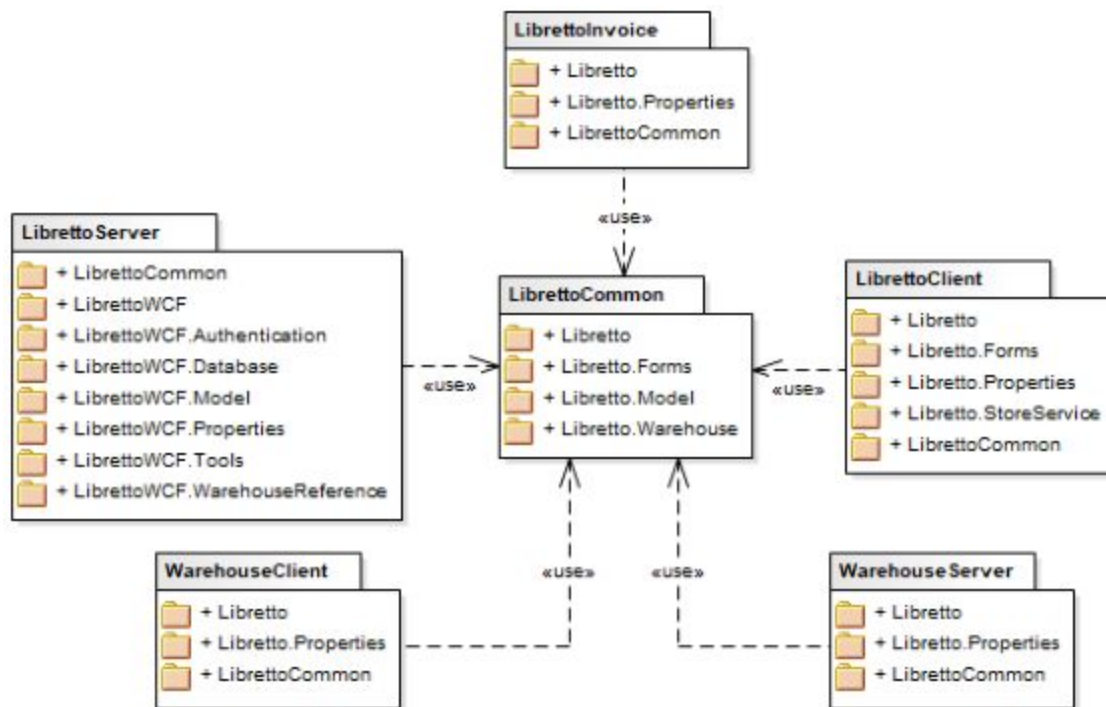


Diagrama de packages do sistema, evidenciando as relações de dependência entre os vários projetos, bem como os namespaces existentes em cada um.

O sistema encontra-se estruturado em quatro *solutions* diferentes, cada uma constituída por um ou mais projetos de Visual Studio, para além de um projeto Node/npm para a aplicação web da loja *online* que no conjunto constituem a aplicação *Libretto*.

- **LibrettoBookstore** : aplicações que implementam e permitem a gestão da livraria
  - **LibrettoClient** : aplicação cliente da livraria com *interface* gráfica
  - **LibrettoWCF** : aplicação do servidor da livraria
- **LibrettoCommon** : *class library* partilhada por todas as aplicações
- **LibrettoInvoice** : aplicação de faturação com *interface* gráfica
- **LibrettoWarehouse** : aplicações que implementam e permitem a gestão do armazém
  - **WarehouseClient** : aplicação cliente do armazém com *interface* gráfica
  - **WarehouseServer** : aplicação do servidor do armazém
- **libretto-online** : projeto da aplicação web

## 4.1 *LibrettoClient*

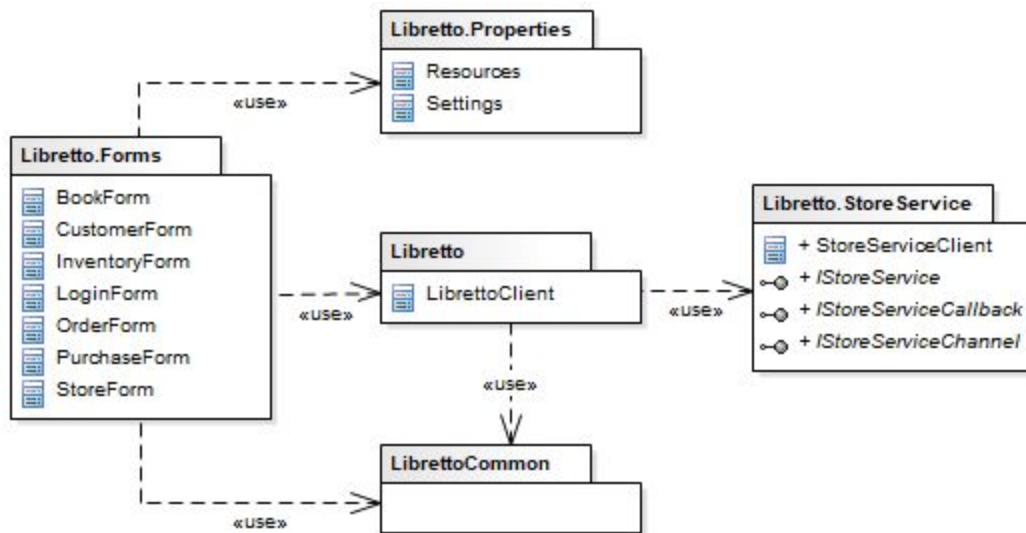


Diagrama de classes do projeto **LibrettoClient**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>Libretto</i>	Funcionalidades necessárias ao funcionamento da aplicação cliente; contém ainda uma classe <i>singleton</i> que guarda informações relativas ao estado da mesma.
<i>LibrettoCommon</i>	Define constantes, operações, <i>interfaces</i> , classes e <i>enums</i> , entre funcionalidades partilhadas por todas as aplicações constituintes deste sistema, nomeadamente as estruturas de dados utilizadas nos serviços, os estados de uma encomenda, as respostas devolvidas pelos serviços ao utilizador, a localização das <i>messages queues</i> , os portos utilizados na comunicação com <i>remoting</i> , etc...
<i>Libretto.Forms</i>	<i>Forms</i> utilizados na <i>interface</i> gráfica da aplicação, tais como a janela de <i>login</i> , a janela principal da aplicação, janelas para criação de uma nova encomenda/compra/cliente, gestão de livros etc...
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia, textuais utilizados na interface <i>gráfica</i> da aplicação, tais como ícones, <i>bitmaps</i> e <i>strings</i> .
<i>Libretto.StoreService</i>	Implementa uma classe <i>proxy</i> com os diferentes métodos/operações e estruturas de dados que permitem a comunicação com o serviço WCF com o mesmo nome implementado no servidor da loja.

## 4.2 LibrettoCommon

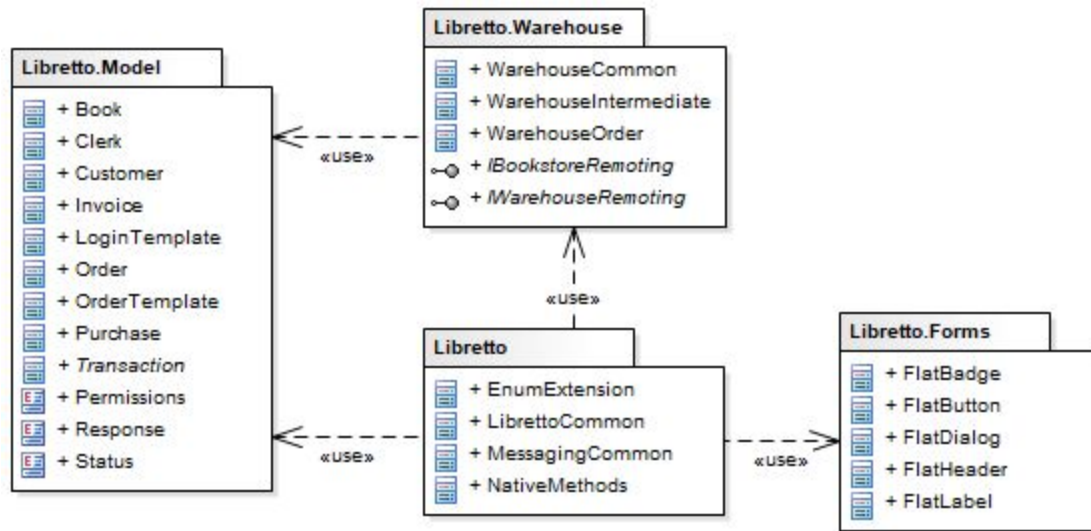


Diagrama de classes do projeto **LibrettoCommon**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>Libretto</i>	Implementa funcionalidades comuns tanto à aplicação cliente como à aplicação servidor.
<i>Libretto.Forms</i>	Implementa controlos de <i>interface</i> gráfica personalizados utilizados nos <i>forms</i> das diversas aplicações.
<i>Libretto.Model</i>	Define estruturas de informação utilizadas na comunicação através de <i>chamadas remotas</i> , serviços WCF, serviços REST e filas de mensagens, formulários de encomenda, autenticação, estruturas para registo de clientes, livros, funcionários etc...
<i>Libretto.Warehouse</i>	Define <i>interfaces de remoting</i> , estruturas de informação e constantes comuns ao servidor o armazém, bem como às diversas aplicações que interagem com este,



### 4.3 LibrettoWCF

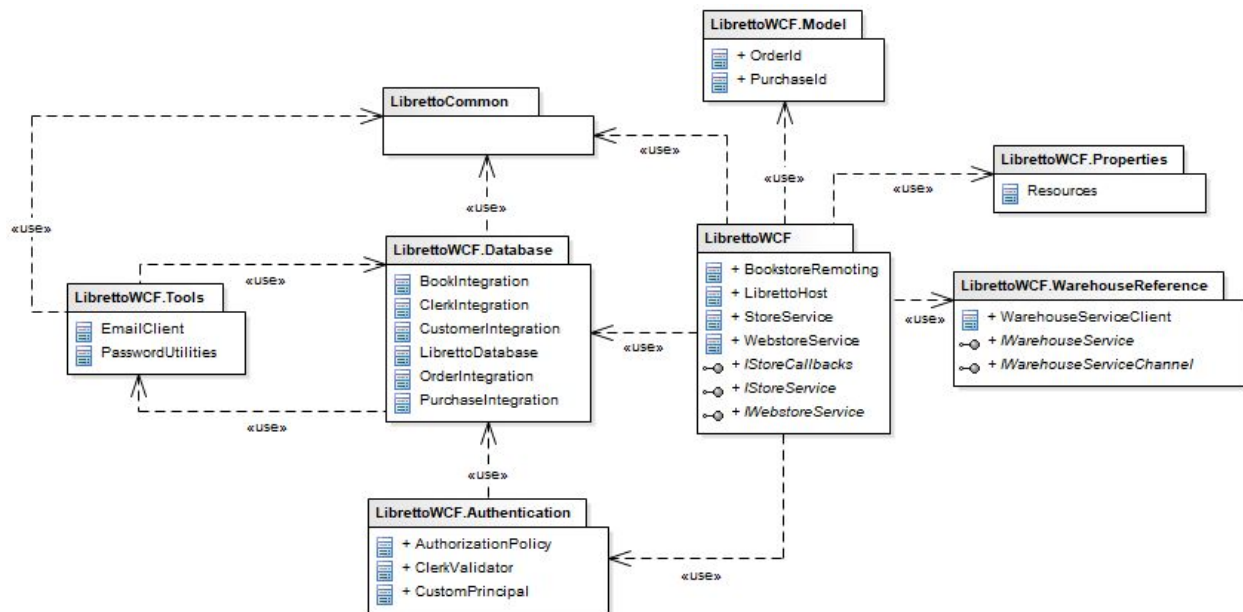


Diagrama de classes do projeto **LibrettoWCF**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>LibrettoWCF</i>	Define um conjunto de funcionalidades necessárias ao funcionamento do servidor; contém uma classe <i>singleton</i> que guarda informações relativas ao estado da mesma;
<i>LibrettoWCF.Authentication</i>	Define um conjunto de classes que implementam autenticação segura do tipo <i>custom</i> para os funcionários e proprietário da loja autenticarem-se na aplicação cliente através de um certificado, <i>username</i> e <i>password</i> .
<i>LibrettoWCF.Database</i>	Define uma <i>interface</i> intuitiva para acesso e execução de <i>queries</i> sobre a base de dados SQL Server da aplicação, recorrendo ao ORM ( <i>object-relational mapping</i> ) <i>Entity Framework</i> da Microsoft, implementando também parte da lógica associada às encomendas (comunicação com armazém, envio de e-mails aos clientes, atualização do estado).
<i>LibrettoWCF.Model</i>	Define estruturas de dados personalizadas para receber pedidos <i>HTTP</i> através do servidor REST.

<i>LibrettoWCF.Properties</i>	Define um conjunto de recursos multimédia, textuais utilizados na interface <i>gráfica</i> da aplicação, tais como ícones, <i>bitmaps</i> e <i>strings</i> .
<i>LibrettoWCF.Tools</i>	Implementa um cliente de <i>e-mail</i> para notificar os clientes sempre que se verificarem atualizações no estado das suas encomendas ( <i>EmailService</i> ), bem como encriptação e verificação de <i>passwords</i> ( <i>PasswordUtilities</i> ) através de um algoritmo de <i>hashing</i> .
<i>LibrettoWCF.WarehouseReference</i>	Implementa uma classe <i>proxy</i> com os diferentes métodos/operações e estruturas de dados que permitem a comunicação com o serviço WCF <i>WarehouseService</i> implementado no servidor do armazém.

#### 4.4 LibrettoInvoice



Diagrama de classes do projeto **LibrettoInvoice**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>Libretto</i>	Forms utilizados na <i>interface</i> gráfica da aplicação das faturas, tais como a janela inicial, que permite a listagem das faturas recebidas por <i>message queueing</i> , ou a janela de visualização da fatura, que permite visualizar detalhes relevantes da mesma.
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia, textuais utilizados na interface <i>gráfica</i> da aplicação, tais como ícones, <i>bitmaps</i> e <i>strings</i> .
<i>LibrettoCommon</i>	Define constantes, operações, <i>interfaces</i> , classes e <i>enums</i> , entre funcionalidades partilhadas por todas as aplicações constituintes deste sistema, nomeadamente as estruturas de dados utilizadas nos serviços, os estados de uma encomenda, as respostas devolvidas pelos serviços ao utilizador, a localização das <i>messages queues</i> , os portos utilizados na comunicação com <i>remoting</i> , etc..

## 4.5 WarehouseClient



Diagrama de classes do projeto **WarehouseClient**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>Libretto</i>	Forms utilizados na <i>interface</i> gráfica da aplicação do armazém, tais como a janela principal, que permite consultar encomendas e gerir as encomendas registadas no servidor do armazém.
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia, textuais utilizados na interface <i>gráfica</i> da aplicação, tais como ícones, <i>bitmaps</i> e <i>strings</i> .
<i>LibrettoCommon</i>	Define constantes, operações, <i>interfaces</i> , classes e <i>enums</i> , entre funcionalidades partilhadas por todas as aplicações constituintes deste sistema, nomeadamente as estruturas de dados utilizadas nos serviços, os estados de uma encomenda, as respostas devolvidas pelos serviços ao utilizador, a localização das <i>messages queues</i> , os portos utilizados na comunicação com <i>remoting</i> , etc..

## 4.6 WarehouseServer



Diagrama de classes do projeto **WarehouseServer**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

Namespace	Descrição
<i>Libretto</i>	Funcionalidades necessárias ao funcionamento do servidor do armazém; contém uma classe <i>singleton</i> que guarda informações relativas ao estado da mesma e inicializa os serviços associados; implementa ainda operações que permitem carregar, guardar em disco (armazenamento persistente) um registo das encomendas que chegam ao servidor através do serviço de <i>message queueing</i> .
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia, de imagem e textuais da aplicação, tais como ícones, <i>bitmaps</i> e algumas <i>strings</i> utilizadas na <i>interface</i> gráfica.
<i>LibrettoCommon</i>	Define constantes, operações, <i>interfaces</i> , classes e <i>enums</i> , entre funcionalidades partilhadas por todas as aplicações constituintes deste sistema, nomeadamente as estruturas de dados utilizadas nos serviços, os estados de uma encomenda, as respostas devolvidas pelos serviços ao utilizador, a localização das <i>messages queues</i> , os portos utilizados na comunicação com <i>remoting</i> , etc..

## 5 Conclusão

No final do desenvolvimento deste projeto foi possível adquirir competências básicas no domínio da *framework Windows Communication Foundation*, *message queueing* (MSMQ.Net) e sistemas distribuídos seguindo uma arquitetura orientada a serviços, bem como aprofundar competências previamente adquiridas sobre linguagem C#, .NET Remoting e desenvolvimento de aplicações *web* modernas. Como resultado final, o sistema desenvolvido implementa as regras de negócio pedidas no enunciado, sendo capaz de comunicar de forma eficiente entre as várias aplicações nele integradas.

Como principais dificuldades encontradas durante a realização deste trabalho podemos salientar a configuração dos serviços, nomeadamente dos seus *endpoints*; a implementação do mecanismo de *callbacks* nos serviços WCF de forma a notificar a aplicação da loja com um mecanismo semelhante a eventos sempre que um cliente realiza compras na loja *online* ou sempre que um funcionário do armazém confirma o envio das unidades encomendadas; a implementação de um mecanismo seguro de autenticação na aplicação *web*.

## 6 Recursos

Segue-se uma lista de todas as referências bibliográficas consultadas, bem como do *software* utilizado ao longo do desenvolvimento deste trabalho:

### 6.1 Referências bibliográficas

- ❖ **“A Beginner's Tutorial on Creating WCF REST Services - CodeProject”**  
([www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services](http://www.codeproject.com/Articles/571813/A-Beginners-Tutorial-on-Creating-WCF-REST-Services))
- ❖ **“Duplex Service in WCF - CodeProject”**  
([www.codeproject.com/Articles/660479/Duplex-Service-in-WCF](http://www.codeproject.com/Articles/660479/Duplex-Service-in-WCF))
- ❖ **“Microsoft Developer Network - Configuring Services Using Configuration Files”**  
(<https://msdn.microsoft.com/en-us/library/ms733932%28v=vs.110%29.aspx?f=255&MSPPErr=-2147217396>)
- ❖ **“Microsoft Developer Network - System Provided Bindings”**  
([https://msdn.microsoft.com/en-us/library/ms730879\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/ms730879(v=vs.110).aspx))
- ❖ **“Microsoft Developer Network - Using Message Queueing”**  
([https://msdn.microsoft.com/en-us/library/ms705205\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms705205(v=vs.85).aspx))
- ❖ **“Using the Microsoft Message Queue MSMQ and C# ASP .NET | Primary Objects”**  
([www.primaryobjects.com/2007/08/13/using-the-microsoft-message-queue-msmq-and-c-asp-net/](http://www.primaryobjects.com/2007/08/13/using-the-microsoft-message-queue-msmq-and-c-asp-net/))

### 6.2 *Software* utilizado

- ❖ **“Visual Studio Enterprise 2017”** (<https://www.visualstudio.com/vs>)  
Microsoft Visual Studio is an integrated development environment from Microsoft, used to develop computer programs for Windows, as well as websites, web applications, web services and mobile applications. Visual Studio integrates software development platforms such as Windows API, Windows Forms, Windows Communication Foundation Windows Presentation Foundation, Windows Store and Microsoft Silverlight.

## A Anexos

Libretto Store <a href="#">Books</a> <a href="#">Transactions</a> <span>Welcome, Diogo Marques</span> <a href="#">Logout</a>						
Orders						
Book Title	Price	Quantity	Sub-Total	Status	Last Updated	Date Created
Dummy Book J	\$83.61	55	\$4,598.29		21/05/2017 @ 3:10PM	21/05/2017 @ 3:10PM
Purchases						
Book Title	Price	Quantity	Sub-Total	Date Created		
Dummy Book I	\$155.62	1	\$155.62	21/05/2017 @ 3:07PM		
Dummy Book A	\$5.00	2	\$9.99	13/05/2017 @ 7:44PM		

**Figura 1.** Visualização de encomendas e compras do utilizador autenticado


Libretto Store

Books

Transactions

Welcome, Diogo Marques


Logout



Dummy Book G

\$6.47


Units Available: 94



Dummy Book J

\$83.61


Units Available: 38



Dummy Book I

\$155.62


Units Available: 12



Dummy Book A

\$117.84


Units Available: 4



Dummy Book F

\$199.46


Units Available: 23



Dummy Book E

\$76.46


Units Available: 21



Dummy Book H

\$153.05


Units Available: 81



Dummy Book B

\$190.40

Units Available: 7



Dummy Book C

\$97.22

Units Available: 9

**Figura 2.** Visualização dos livros que a aplicação Libretto disponibiliza, com informação relativa ao título, unidades disponíveis e o respetivo preço.

Libretto Store Books Transactions

Welcome, Diogo Marques Logout

### Dummy Book F

**Product Information**

Price  
\$199.46

Stock  
23

**Buyer Information**

Full Name  
Diogo Marques

Address  
Valongo, Porto

**Transaction Type**  
Purchase

**Quantity**  
1

Submit

**Dummy Book I** \$155.62  
Units Available: 12

**Dummy Book E** \$76.46  
Units Available: 21

**Dummy Book H** \$153.05  
Units Available: 81

**Dummy Book B** \$190.40  
Units Available: 7

**Dummy Book C** \$97.22  
Units Available: 9

**Figura 3.** Janela para criação de encomenda ou compra de um livro por parte do utilizador autenticado, com informação relativa ao comprador e ao livro a ser comprado.