

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

1 Aplicação	2
1.1 <i>LibrettoClient</i>	3
1.2 <i>LibrettoInvoice</i>	9
1.3 <i>WarehouseClient</i>	11
1.4 <i>LibrettoOnline</i> (loja online)	13
2 Arquitetura	14
4 Serviços	15
4.1 <i>IBookstoreService</i>	15
4.2 <i>IWebstoreService</i>	20
4.3 <i>IWarehouseService</i>	23
4.4 <i>IBookstoreRemoting</i>	24
4.5 <i>IWarehouseRemoting</i>	24
5 Diagramas UML	26
5.1 <i>LibrettoClient</i>	27
5.2 <i>LibrettoCommon</i>	28
5.3 <i>LibrettoWCF</i>	29
5.4 <i>LibrettoInvoice</i>	30
5.5 <i>WarehouseClient</i>	31
5.6 <i>WarehouseServer</i>	31
6 Conclusão	32
7 Recursos	33
7.1 Referências bibliográficas	33
7.2 <i>Software</i> utilizado	33
A Anexos	34

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 *stocks* assente na plataforma *.NET Framework* da Microsoft, que compreende tecnologias tais como *Windows Communication Foundation (WCF)*, *.NET Remoting* e *Microsoft Message Queueing (MSMQ)*. Foi utilizada a linguagem C# e *Windows Forms* para a *interface* gráfica (GUI) das aplicações cliente.

Uma editora vende livros nas suas instalações. Pretende-se desenvolver um sistema para coordenar as vendas, encomendas, bem como gerir *stocks*. A editora é proprietária de duas instalações situadas em locais distintos: uma loja onde os livros estarão expostos aos clientes, e um armazém onde são guardados os livros em grande volume. A editora quer ainda disponibilizar uma aplicação *web* que permita aos seus clientes consultar e encomendar livros.

1 Aplicação

O sistema consiste em dois servidores, sendo cada servidor responsável por uma base de dados que armazena informação relativa às vendas/encomendas (*LibrettoServer*) e outro responsável pelo armazenamento dos pedidos de reposição de *stock* (*WarehouseServer*).

O sistema possui ainda três aplicações com *interface* gráfica para gerir os *stocks* da loja, registar vendas, registar encomendas (*LibrettoClient*), gerir pedidos de reposição de *stock* (*WarehouseClient*), e consultar os recibos relativos às vendas (*LibrettoInvoice*).

Foi ainda desenvolvida uma aplicação *web* que permite aos clientes registarem-se e autenticarem-se na loja *online* para poderem consultar os livros disponíveis, um histórico das suas transações, realizar compras e fazer pedidos de encomenda.

1.1 *LibrettoClient*

Esta aplicação, utilizada pelos funcionários ou pelo proprietário da loja, permite a venda imediata de livros que se encontrem em *stock* na loja a clientes visitantes, bem como a criação de encomendas (semelhantes às encomendas realizadas *online* através da aplicação *web*), gestão do catálogo de livros disponíveis para venda, gestão de clientes, etc..

[illegible]

Figura 2. Janela principal da aplicação GUI da loja, evidenciando algumas opções de filtragem, uma lista com as transações registradas na loja e botões para realizar diversas operações.

Na janela principal da aplicação é possível realizar as seguintes operações: **Register Purchase**, que abre uma janela contendo um formulário que permite a criação de uma venda; **Place Order**, que abre uma janela contendo um formulário que permite registar a encomenda de um cliente; **Cancel Order**, que permite cancelar uma encomenda em curso; **Delete**, que permite apagar uma transação (venda/encomenda) da lista, independentemente do estado em que se encontra; **Manage Books**, para abrir uma nova janela que permite a gestão do catálogo de livros disponíveis para venda.

O funcionário da loja pode ainda terminar sessão na aplicação a qualquer altura carregando na opção **Logout**, regressando à janela de autenticação onde é possível voltar a iniciar sessão com a mesma conta de utilizador ou com um utilizador diferente. Ao fazer duplo clique sobre um elemento da lista é possível abrir uma nova janela contendo informações relativas à transação seleccionada, ao cliente que a realizou e ao livro vendido. As encomendas apresentadas na janela principal podem ter origem na loja *online* ou na aplicação GUI. Na coluna **Status** é apresentado um dos seguintes estados:

- **Waiting** : número de unidades encomendadas pelo cliente ultrapassou *stock* disponível na loja, sendo que não foi possível satisfazer imediatamente a encomenda com *stock* da própria loja. Foi enviado um pedido de reposição de *stock* ao armazém, a loja aguarda agora pela chegada das mercadorias requisitadas;
- **Pending [DD/MM/YYYY]** : armazém confirmou expedição das mercadorias requisitadas no pedido de reposição de *stock*, aguarda que um funcionário confirme a sua chegada;
- **Dispatched [DD/MM/YYYY]** : a encomenda foi entregue com sucesso ao respetivo cliente ou encontra-se a caminho da sua morada;
- **Cancelled [DD/MM/YYYY]** : a encomenda foi cancelada por um funcionário da loja ou pelo próprio cliente através da aplicação *web*.

Quando a loja 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**.

As principais diferenças entre uma venda e uma encomenda é que ao contrário do que acontece com as encomendas, ao realizar uma venda não é possível adquirir um número de unidades além das disponíveis em *stock* na loja; por outro lado, as encomendas apresentam um estado, enquanto que as vendas são realizadas de forma imediata.

Uma encomenda pode ser cancelada se ainda não tiver sido enviada ao cliente. Pela mesma razão, a opção **Cancel Order** aparece disponível apenas quando a encomenda se encontra no estado *Waiting*, sendo que muda para **Dispatch Order** quando passa ao estado *Pending*, e fica desativada nos estados *Dispatched* e *Cancelled*. A opção *Dispatch Order* permite confirmar a recepção das unidades pedidas ao armazém quando este notifica a loja do envio das mesmas.

A opção **Delete** permite apagar uma transação independentemente do estado em que se encontra. Ao apagar uma encomenda no estado *Pending*, todas as unidades enviadas pelo armazém contam como *stock* para a loja. Por exemplo, uma encomenda de 20 unidades gera um pedido de 20 + 10 unidades ao armazém. Quando o armazém informa a loja que as unidades foram expedidas, se um funcionário da loja confirmar a recepção da mercadoria e decidir enviar a encomenda ao cliente, 10 das unidades encomendadas ao armazém vão para *stock* da loja. No entanto, se nas mesmas condições o funcionário (ou o cliente) decidir cancelar a encomenda, as 30 unidades vão para *stock* da loja.

A principal diferença entre as operações *Cancel Order* e *Delete* para as encomendas reside no facto do cancelamento não remover a encomenda da base de dados, levando apenas à alteração do seu estado para *Cancelled*. Ou seja, a encomenda continuará a aparecer na listagem da janela principal, no entanto com um estado especial, bem como a data em que foi cancelada. Por outro lado, não é possível cancelar uma venda, mas é possível apagar. Apagar uma venda não provoca alterações no *stock* do respetivo livro, semelhante ao que acontece quando se apaga uma encomenda que se encontre no estado *Waiting*, *Dispatched* ou *Cancelled*.

Customer		Book	
GUID	D0679C4FA84D44FE86862C10514B56F7	GUID	B1867A4B7009466398080C6C85E19747
Name	Diogo Marques	Title	Dummy Book G
E-mail	marques999@gmail.com	Price	6,47 €
Location	Valongo, Porto	Stock	82 Un
Order			
Quantity	<input type="range"/>		12 Un
Total			77,67 €
Status	<input type="radio"/> Waiting Expedition <input type="radio"/> Pending <input checked="" type="radio"/> Dispatch Complete		
		Dispatch	Cancel

Figura 2. Janela para criação de uma nova encomenda de cliente.

Na janela de criação de uma nova encomenda é possível escolher o cliente ao qual se destina através de uma *combo box*. O botão com reticências (...) situado ao lado da *combo box* permite adicionar um novo cliente. Os clientes registados por este meio não têm acesso à loja *online*, pois não lhes é atribuída automaticamente uma *password* de acesso. No lado direito da janela existe uma *combo box* para escolher o livro a ser encomendado. É também possível visualizar informações do cliente e do livro escolhidos. Em baixo existe uma *track bar* para escolher o número de unidades encomendadas, o valor total da encomenda (calculado sempre que houver alteração na escolha do livro ou no número de unidades), bem como o estado inicial da encomenda. **NOTA:** A janela de criação de uma venda não irá ser abordada neste relatório devido às semelhanças que apresenta com a janela de criação de uma encomenda. No entanto existem duas diferenças: um funcionário não poder vender mais unidades de determinado livro do que as unidades em *stock* na loja, uma venda não apresentar estado, pelo que este campo não é apresentado na janela de vendas.

Publish Book	
GUID	B67C921AFFDF48F084F9A41264C379D3
Title	Sample Book
Price	14,99 €
Stock	20 Un
<div> <div>Confirm</div> <div>Cancel</div> </div>	

Figura 2. Janela para publicação de um livro, evidenciando um formulário com diferentes campos a serem preenchidos pelo funcionário da loja.

Ao carregar em **Confirm** na janela de publicação, este livro passa a estar imediatamente disponível para venda na aplicação, com o *stock* indicado no formulário preenchido.

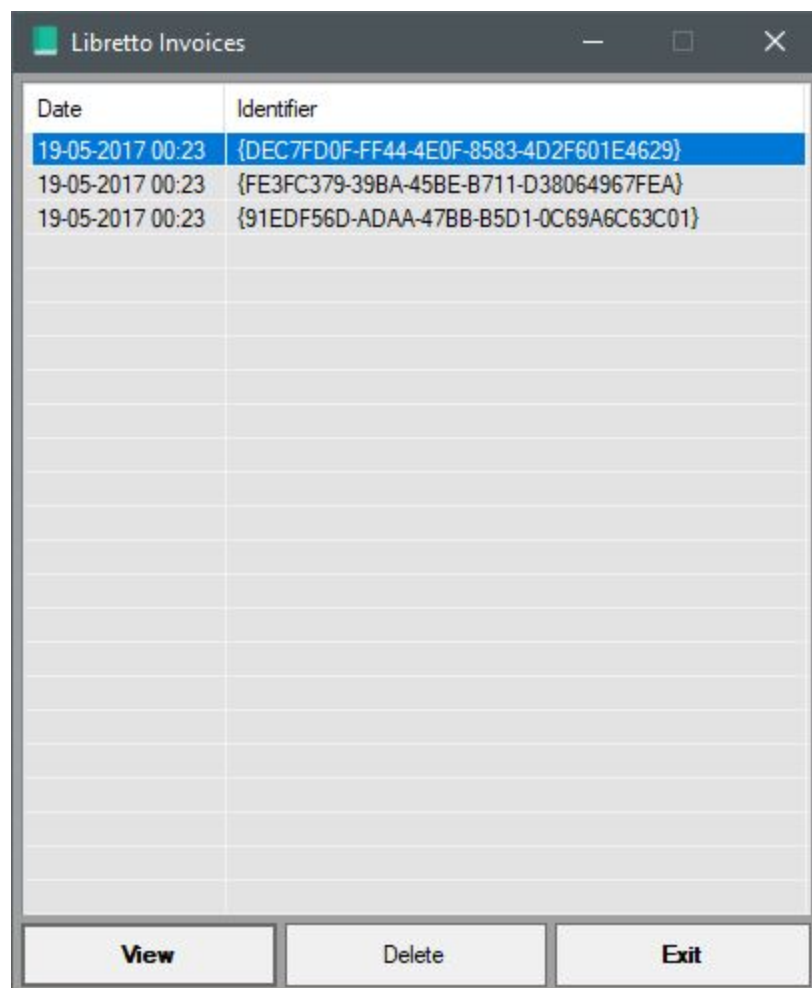
Register Customer	
GUID	D3B5B5F6F99949E399CC2EE214DDCE60
Name	Carlos Samouco
E-mail	carlosamouco@gmail.com
Location	Porto, Portugal
<div> <div>Confirm</div> <div>Cancel</div> </div>	

Figura 2. Janela para registo de um cliente, evidenciando um formulário com diferentes campos a serem preenchidos pelo funcionário da loja.

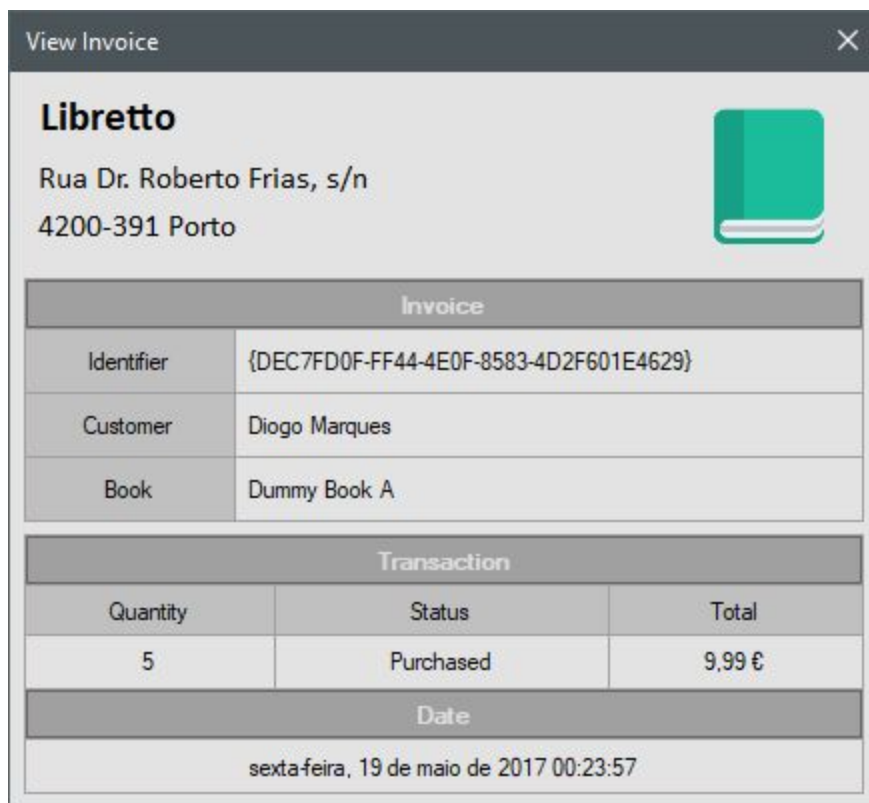
Ao carregar em **Confirm** na janela de registo, este passa a estar imediatamente disponível na *combobox* “Name” existente janela de criação da encomenda.

1.2 LibrettoInvoice

Esta aplicação com *interface* gráfica em *Windows Forms* simula uma fila de impressão, permitindo visualizar de forma intuitiva os recibos das vendas efetuadas na loja à medida que estas vão sendo registadas no servidor da loja. Por ser implementado através de uma *message queue*, não é necessário que esta aplicação se encontre em execução para receber em tempo real os recibos enviados pela loja. Pelo contrário, quando a aplicação é executada, a primeira tarefa que esta realiza 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 todos os recibos que foram publicados na fila enquanto esta não esteve em execução, notificando a chegada dos mesmos através de um sinal sonoro e um *toast* no canto inferior direito do ecrã.



Na janela principal desta aplicação encontramos uma listagem de todos os recibos que chegaram à *message queue*, bem como algumas informações relativas às mesmas: código de identificação, data de criação. É possível realizar três ações nesta janela: **View**, para visualizar um recibo com detalhe, abrindo uma nova janela que mostra todas as informações relevantes do mesmo; **Delete**, para apagar recibos da fila de impressão; **Exit**, para sair da aplicação. É também possível visualizar um recibo fazendo duplo clique sobre um dos recibos da lista.



The screenshot shows a window titled "View Invoice" with a close button (X) in the top right corner. The window displays the following information:

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

Next to the address is a green icon of a book.

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 um recibo, evidenciando informações relevantes do mesmo, tais como o número de unidades adquiridas, valor total, nome do cliente, título do livro,

Na janela de visualização do recibo é possível consultar informações sobre a venda selecionada, tais como: código de identificação único, nome completo do cliente, título do livro adquirido, número de unidades adquiridas, valor total, data de registo. A aplicação persiste em disco todos os recibos que chegaram através da *message queue*, gravando as vendas num ficheiro XML, sendo que ao escolher a opção **Delete** na janela principal os recibos serão também apagados deste ficheiro.

1.3 WarehouseClient

Esta aplicação permite a gestão dos pedidos de reposição de *stock* que chegam ao armazém. Um pedido é registado no armazém sempre que a loja não possui *stock* suficiente para satisfazer uma encomenda de um cliente, enviando um pedido através do serviço *IWarehouseService* disponibilizado pelo servidor do armazém. A implementação deste serviço através de um *message queue binding* permite que as encomendas dêem entrada no servidor do armazém mesmo fora das suas horas de funcionamento, garantindo que não se perde informação se um dos servidores envolvidos na operação falhar.

[illegible]

Na janela principal da aplicação é possível visualizar uma listagem com todos os pedidos, filtrar pedidos por título do livro, ver os pedidos que chegaram ao servidor do armazém antes de determinada data, após determinada data e enquadrá-los 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 pelo cliente; **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 uma encomenda fazendo duplo clique sobre um dos pedidos existentes na lista.

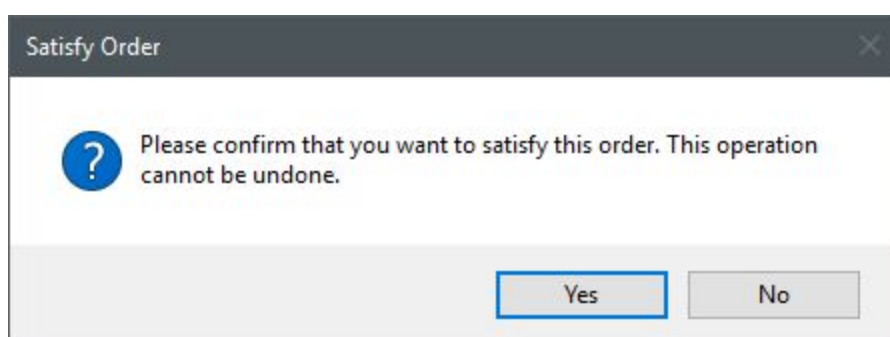


Figura 2. Janela para confirmação da satisfação de um pedido.

Ao fazer **Satisfy Order** é apresentado uma janela para confirmar a expedição do pedido. O pedido é removido da lista, bem como do ficheiro XML que guarda em disco todos os pedidos que chegaram ao servidor do armazém e que ainda não foram confirmados. O estado da respetiva encomenda no servidor da loja é atualizado para *Pending* e a sua data prevista de entrega é definida para dois dias após a data em que a mercadoria foi enviada. Neste intervalo de tempo os livros são fisicamente transportados para a loja, assumindo que o armazém possui sempre a quantidade de livros pedida (*stock* infinito).

NOTA: Para esta aplicação funcionar corretamente, pelo menos 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 quando este tentar executar a ação **Satisfy Order**.

1.4 *LibrettoOnline* (loja online)

Esta aplicação *web* foi desenvolvida na *framework* 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 venda; **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 loja existe um servidor que hospeda a aplicação e disponibiliza uma série de serviços (tanto à aplicação *web* como à aplicação da loja) e mantém um registo persistente dos livros disponíveis, preços e quantidade de *stock* existente na livraria. Este servidor encontra-se sempre ativo e ligado à Internet. Os computadores do armazém normalmente encontram-se ativos apenas durante as horas de trabalho, incluindo o servidor.

O servidor da loja aceita tanto encomendas como compras dos clientes pela Internet através da *web* app. Cada transação possui um título, número de unidades encomendadas 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 apresentam um estado associado. Por razões de simplicidade, apenas se pode adquirir um livro, mas várias unidades do mesmo. Ignoramos também o processo de pagamento no cenário de compra e consideramos um número relativamente reduzido de livros disponíveis. Ao confirmar uma compra, o servidor atualiza o *stock* na livraria e imprime um recibo numa aplicação separada, que simula uma fila de impressão.

O servidor do armazém recebe chamadas do servidor da loja de forma assíncrona através de uma fila de mensagens associada a um serviço WCF (utiliza *MsmqNetBinding*), persistindo esta informação num ficheiro XML 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#.

No caso do servidor da loja, a persistência dos dados é assegurada por uma base de dados relacional SQL assente na tecnologia *SQL Server* da Microsoft. Para criar uma abstração orientada a objetos que facilitasse as operações de acesso à base de dados (*SELECT*, *INSERT*, *DELETE*, *UPDATE*, etc..) recorreu-se ao *Entity Framework*, uma biblioteca de ORM (*object-relational mapping*) da Microsoft para linguagens *NET*. Conseguiu-se assim desenvolver um sistema empresarial altamente integrado e robusto dentro do ecossistema *Windows / .NET Framework*.

3 Serviços

Esta secção do relatório servirá para descrever com detalhe os diferentes serviços implementados nas aplicações desenvolvidas, as tecnologias utilizadas (*WCF*, *NET Remoting*, *MSMQ*, *REST*), bem como a contribuição individual destes serviços para a organização e correto funcionamento do sistema que se pretendeu desenvolver..

3.1 *IBookstoreService*

Este serviço WCF implementado no **servidor da loja** define um conjunto de métodos invocados pela **aplicação cliente** da mesma que permite ao proprietário (administrador) e aos funcionários fazer a gestão do catálogo de livros, dos clientes bem como das transações (todas as vendas/encomendas realizadas pelos clientes) de forma intuitiva. Este serviço confere as seguintes funcionalidades à aplicação da loja:

- Consultar clientes
- Consultar transações
- Consultar livros disponíveis
- Adicionar livros ao catálogo
- Remover livros do catálogo
- Registar vendas
- Registar encomendas
- Cancelar encomendas
- Satisfazer encomendas
- Apagar registo das vendas
- Apagar registo das encomendas
- Confirmar reposição de *stock*
- Enviar pedidos de reposição de *stock*
- 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()	
Descrição	Permite ao funcionário/administrador com sessão iniciada na aplicação da loja 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.

3.1.5 Contract <Books>

List<Book> ListBooks()	
Descrição	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja obter uma lista com todos os livros disponíveis no catálogo.
Response InsertBook(Book bookInformation)	
Descrição	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja lançar um novo livro.
Parâmetros	bookInformation. Title : título completo do livro bookInformation. Price : preço de venda recomendado bookInformation. Stock : unidades inicialmente em <i>stock</i> bookInformation. Identifier : código de identificação do livro (GUID)
Response UpdateBook(Book bookInformation)	
Descrição	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja editar as informações relativas a determinado livro publicado na loja.
Parâmetros	bookInformation. Title : título completo do livro bookInformation. Price : preço de venda recomendado bookInformation. Stock : unidades inicialmente em <i>stock</i>
Response DeleteBook(Guid bookIdentifier)	
Descrição	Permite ao funcionário/proprietário com sessão iniciada na aplicação da loja retirar um livro do catálogo.
Parâmetros	bookIdentifier : código de identificação do livro (GUID)

3.1.6 Contract <Customers>

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

3.1.7 Contract <Purchases>

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

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

3.1.8 Contract <Orders>

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

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

3.2 *IWebstoreService*

Este serviço WCF REST implementado no **servidor da loja** define um conjunto de operações invocadas na aplicação *web* que lhe conferem as seguintes funcionalidades:

- Registar utilizador
- Autenticar utilizador
- Realizar compra
- Realizar encomenda
- Cancelar encomenda
- Consultar livros disponíveis no catálogo da 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/" />
      </baseAddresses>
    </host>
  </service>
```

3.2.3 Contract <Authentication>

Customer Login(LoginTemplate loginForm)	
Descrição	Permite ao cliente autenticar-se na aplicação <i>web</i> (loja <i>online</i>).
Parâmetros	loginForm. Email : endereço de correio eletrónico do utilizador loginForm. Password : palavra-passe 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. BookId : código de identificação do livro adquirido purchaseForm. BookTitle : título completo do livro adquirido purchaseForm. CustomerId : código de identificação do cliente purchaseForm. CustomerName : nome completo do cliente purchaseForm. Quantity : número de unidades adquiridas pelo cliente purchaseForm. Total : valor total (derivado do preço unitário do livro * número de unidades adquiridas pelo cliente)

List<Purchase> GetPurchasesByUser()	
Descrição	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>

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

3.2.6 Contract <Books>

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

3.3 *IWarehouseService*

Este serviço WCF implementado no **servidor do armazém** (*WarehouseServer*) define um conjunto de métodos que processam as mensagens enviadas pelo servidor da loja para a *message queue* do armazém. Estas mensagens estão relacionadas com a gestão dos pedidos de reposição de *stock* efetuados pela loja, permitindo ao servidor da loja notificar o servidor do armazém sempre que é necessário enviar *stock* para satisfazer uma encomenda de um cliente.

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 DeleteOrder (Guid orderIdentifier)	
Descrição	Permite ao servidor da loja informar ao servidor do armazém que determinada encomenda foi cancelada ou apagada pelo cliente/funcionário, removendo os pedidos de reposição de <i>stock</i> associados.
Parâmetros	orderIdentifier : código de identificação da encomenda (GUID)

void InsertOrder(WarehouseOrder warehouseOrder)	
Descrição	Permite ao servidor da loja informar ao servidor do armazém que uma nova encomenda (realizada por um cliente na aplicação <i>web</i> ou por um funcionário na aplicação da loja) foi registada no sistema.
Parâmetros	warehouseOrder. Identifier : código de identificação da encomenda (GUID) warehouseOrder. Status : estado da encomenda (<i>Waiting</i> ou <i>Dispatched</i>) warehouseOrder. Timestamp : data de registo da encomenda warehouseOrder. Title : nome do livro encomendado warehouseOrder. Total : valor total da encomenda (derivado do preço unitário do livro * número de unidades encomendadas pelo cliente)

3.4 IBookstoreRemoting

Esta *interface* implementada no **servidor da loja** (*BookstoreServer*) e invocada remotamente pelo **servidor do armazém** (*WarehouseServer*) define um único método de *remoting* que permite ao servidor do armazém satisfazer os pedidos de reposição de *stock* enviados pelo servidor da loja, estabelecendo assim um meio de comunicação entre ambos.

bool DispatchOrder(Guid orderIdentifier)	
Descrição	Permite ao servidor armazém responder ao pedido de reposição de <i>stock</i> enviado pelo servidor da loja, notificando este assim que os produtos da encomenda estiverem prontos a serem enviados para a loja.
Parâmetros	orderIdentifier : código de identificação da encomenda (GUID)

3.5 IWarehouseRemoting

Esta *interface* implementada no **servidor do armazém** (*WarehouseClient*) e invocada remotamente pela aplicação cliente do armazém (*WarehouseServer*), define um conjunto de métodos de *remoting* que permitem a esta aplicação realizar operações relativas à gestão dos pedidos de reposição de *stock* recebidos pelo servidor do armazém.

3.5.1 Eventos

event WarehouseUpsertHandler OnUpsert

Evento subscrito pela aplicação cliente do armazém para receber notificações sempre que um pedido de reposição de *stock* chega ao armazém. Ao ser invocado remotamente, este evento executa um *delegate* na janela principal da aplicação que permite adicionar esse pedido à lista de pedidos presente na *interface* da mesma.

event WarehouseDeleteHandler OnDelete

Evento subscrito pela aplicação cliente do armazém para receber notificações sempre que uma encomenda é cancelada ou removida, quer através da loja *online*, quer através de um funcionário na aplicação da loja (removendo assim os respetivos pedidos de reposição de *stock* registados no armazém). Ao ser invocado remotamente, este evento executa um *delegate* na janela principal da aplicação que permite remover elementos da lista de pedidos.

3.5.2 Métodos

List<WarehouseOrder> ListOrders()	
Descrição	Permite à aplicação cliente do armazém visualizar uma lista com todos os pedidos de reposição de <i>stock</i> que chegaram ao servidor do armazém.
bool DispatchOrder(Guid orderIdentifier)	
Descrição	Permite à aplicação cliente do armazém responder a um pedido de reposição de <i>stock</i> , 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 remotamente um outro método existente numa segunda <i>interface</i> , estabelecida entre ambos os servidores.
Parâmetros	orderIdentifier : código de identificação da encomenda (GUID)

4 Diagramas UML

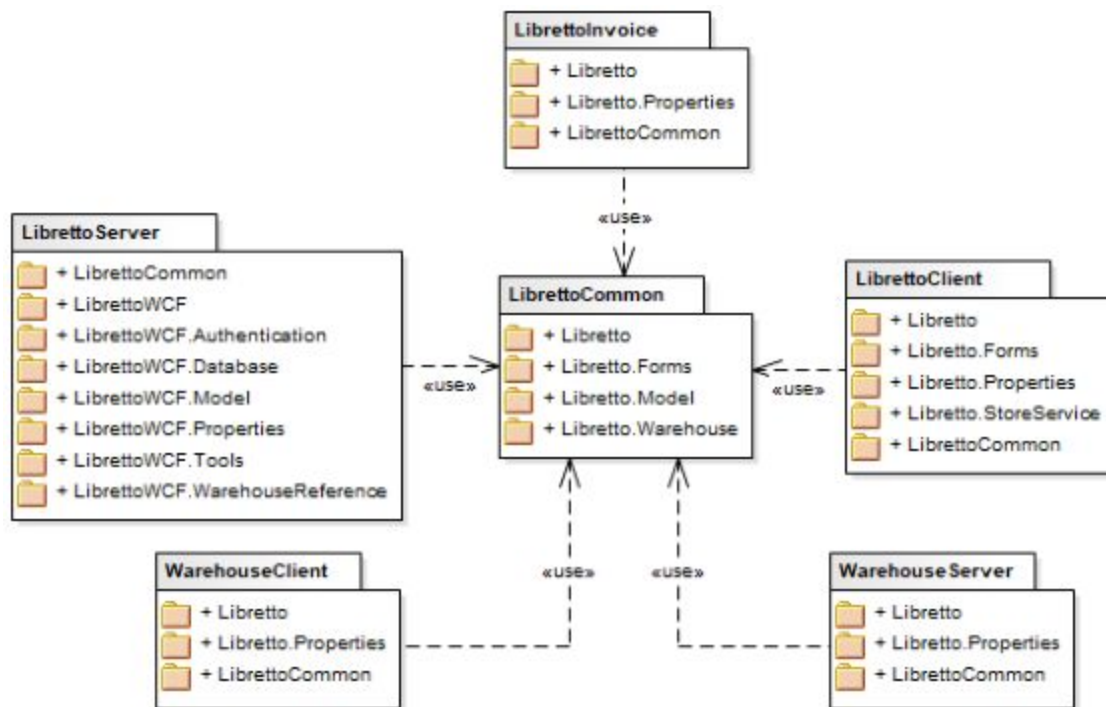


Diagrama de packages do sistema, evidenciando as relações de dependência entre os vários projetos, bem como os namespaces existentes em cada um.

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 loja com *interface* gráfica
 - **LibrettoWCF** : aplicação do servidor da loja
- **LibrettoCommon** : *class library* partilhada por todas as aplicações
- **LibrettoInvoice** : aplicação de recibos 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*

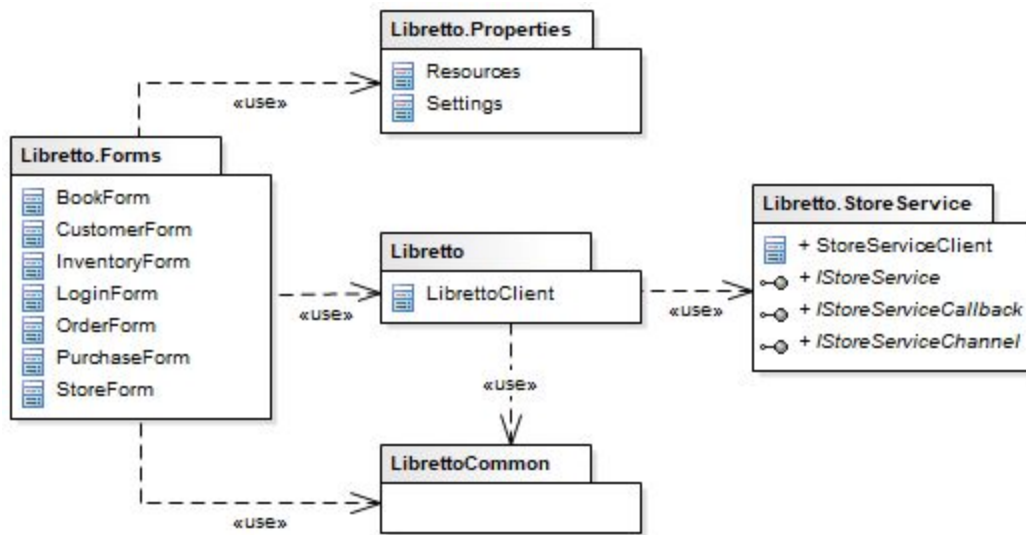


Diagrama de classes do projeto **LibrettoClient**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

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 outras funcionalidades partilhadas pelas aplicações que constituem este 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/venda/cliente, gestão de livros etc...
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia e 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

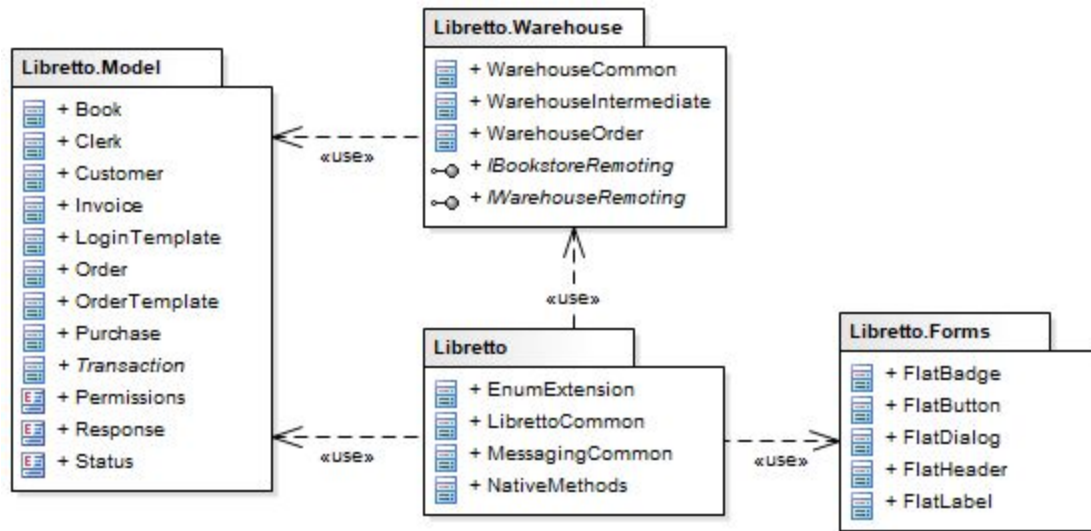


Diagrama de classes do projeto **LibrettoCommon**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

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

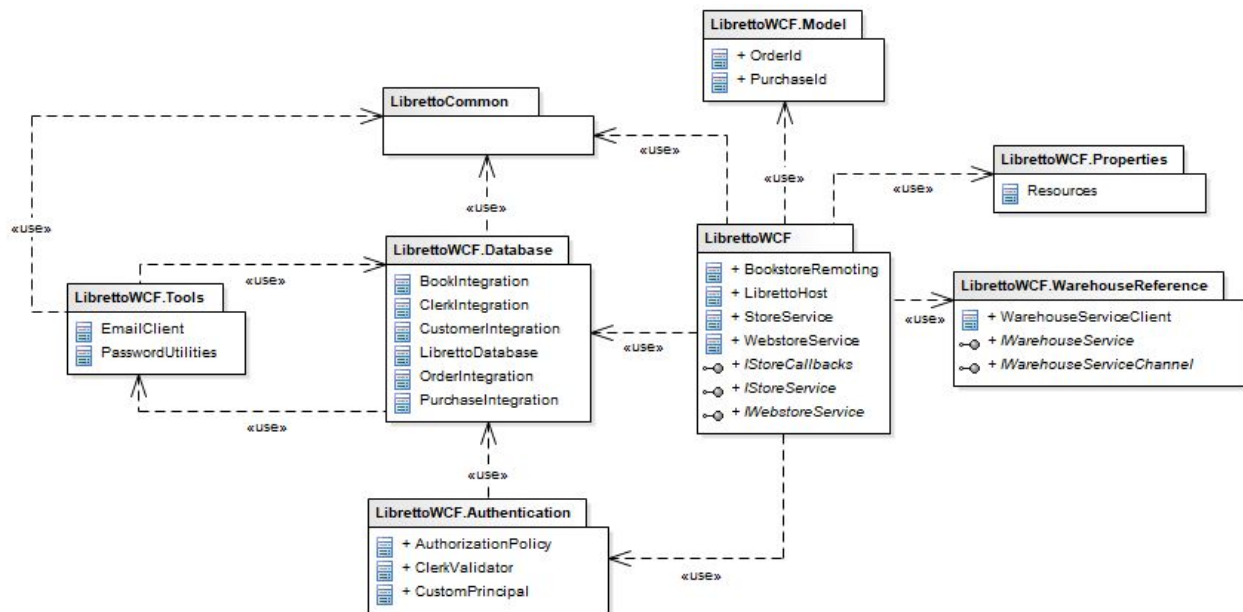


Diagrama de classes do projeto **LibrettoWCF**, evidenciando as relações de dependência entre os vários namespaces, bem como a sua organização em classes.

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 à biblioteca 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 das encomendas).
<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 e 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 várias classes <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>	<i>Forms</i> utilizados na <i>interface</i> gráfica da aplicação de recibos, tais como a janela inicial, que permite a listagem dos recibos que chegaram por <i>message queueing</i> , ou a janela de visualização do recibo, que permite consultar informações relevantes do mesmo.
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia e 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 outras funcionalidades partilhadas pelas aplicações que constituem este 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 e gerir os pedidos de reposição de <i>stock</i> registados no servidor do armazém.
<i>Libretto.Properties</i>	Define um conjunto de recursos multimédia e 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 outras funcionalidades partilhadas pelas aplicações que constituem este 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 e 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 outras funcionalidades partilhadas pelas aplicações que constituem este 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)
- ❖ **“Duplex Service in WCF - CodeProject”**
(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/)

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						
Books Transactions						
Welcome, Diogo Marques Logout						
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.