

Mestrado Integrado em Engenharia Informática e Computação

Tecnologias de Distribuição e Integração (TDIN)

An enterprise distributed system

Relatório

Inês Carneiro - up201303501
Miguel Pereira - up201305998

Índice

Introdução	3
Arquitetura	3
Store	3
Warehouse	3
Comunicação entre Warehouse e Store/Website	3
Common	4
Website	4
Funcionalidades e testes	4
Venda de livros	4
Encomenda de livros	4
Ship de livros	4
Confirmação de encomenda	5
Visualização de pedido de encomenda	5
Modo de Funcionamento	6
Instruções para construir e usar todo o demonstrador	7
Conclusão	12
Referências	13
Referências bibliográficas	13
Software utilizado	13

Introdução

Este projeto, proposto no âmbito da unidade curricular de Tecnologias de Distribuição e Integração, consiste no desenvolvimento de um sistema para coordenar as vendas, encomendas e fazer a gestão do *stock* de uma livraria. Existem duas instalações físicas - a loja e um armazém - e um *website*, onde é possível visualizar e encomendar livros. A loja e o *website* estão sempre *online*, mas o armazém funciona apenas em horário laboral.

Arquitetura

O projeto está dividido em quatro módulos: *Store*, *Warehouse*, *Common* e *Website*. A *store* e o *website* estão ligados a um servidor *Node.js* e o *Warehouse* a outro. As bases de dados destes módulos são também independentes. Nas secções seguintes, será explicado em detalhe o que está implementado em cada módulo e de que maneira estes interagem.

Store

A *store* é uma aplicação desenvolvida em C# com a parte gráfica em *Windows Forms*. A comunicação com o servidor *Node.js*, onde está alojada a base de dados da loja, é feita através de *web sockets*. Para tal, foi utilizado o *package* “*SocketIoClientDotNet*”. Neste módulo, é possível fazer vendas de livros, fazer pedidos ao *warehouse* e aceitar os pedidos que chegam do mesmo.

Warehouse

Em termos de arquitetura, o *warehouse* é semelhante à *store*. A aplicação foi desenvolvida em C# com parte gráfica em *Windows Forms* e a comunicação com o servidor *Node.js* é feita, também, através de *web sockets*. Neste módulo, é possível visualizar todas as encomendas feitas ao *warehouse* e enviar essas mesmas encomendas para a *store*.

Comunicação entre Warehouse e Store/Website

A comunicação entre o *warehouse* e a *store/website* é feita através de uma *message queue*. Para implementar a *message queue* foi utilizado o *message broker* “*RabbitMQ*”. As novas encomendas, sejam elas feitas diretamente da *store* ou do *website*, quando não existe *stock* suficiente para satisfazer a encomenda, são enviadas através de uma *message queue*. Desta maneira, o servidor do *warehouse* não precisa de estar sempre *online* e receberá todas as mensagens assim que for inicializado.

Common

Este módulo inclui duas classes - *Book* e *Order* - que representam, respetivamente, um livro e uma encomenda. A classe *Book* contém o *id*, o nome, o autor, o ano, o *stock*, o preço e a imagem da capa do livro. Por sua vez, a classe *Order* contém o *id*, o nome do livro, o estado da encomenda e quantidade.

Website

O *backend* do *website* foi feito com *Node.js*, utilizando o módulo *Express.js*. O *frontend* foi desenvolvido utilizando HTML e CSS com a biblioteca *Bootstrap*. Como

template engine foi usado *Handlebars*. O *Website* vai permitir aos utilizadores unicamente verem e fazerem encomendas de livros. No *Website* os utilizadores podem ver todas as informações relativas aos livros (nome, autor, stock, imagem da capa do livro).

Funcionalidades e testes

Ao iniciar a aplicação, o utilizador, dependendo da interface com que está a operar, poderá ter acesso a diferentes funcionalidades.

Venda de livros

Quando o utilizador, trabalhador na livraria, acede à aplicação da *store* poderá visualizar os livros existentes em loja e fazer vendas. Quando o utilizador acede à página da livraria online poderá visualizar todos os livros existentes no sistema, com o nome, autor, ano e capa de livro. Caso haja *stock* para a quantidade pretendida, o utilizador poderá comprar livros indicando o seu nome, e-mail, morada e a quantidade de livros que pretende comprar. Esta funcionalidade foi testada com diversos livros, com *stocks* diferentes.

Encomenda de livros

Ao aceder à aplicação da *store* ou ao *website* o utilizador terá acesso a todos os livros existentes na aplicação onde poderá encomendar os livros. Caso o acesso seja à aplicação da *store*, essa encomenda é feita ao *warehouse*. Caso seja no *website*, o sistema pedirá ao utilizador para indicar o seu nome, e-mail, morada e a quantidade de livros que pretende encomendar. Assim, será feita uma encomenda ao *warehouse*. Esta funcionalidade foi testada com diversos livros, com *stocks* diferentes.

Ship de livros

Ao aceder à aplicação do *warehouse* o utilizador terá acesso a todos os pedidos de encomendas pendentes, com o seu ID, título do livro para encomenda, quantidade e o estado dessa encomenda (*dispatched* ou *waiting expedition*). Ao selecionar uma encomenda o utilizador poderá dar *ship* da encomenda, ou seja, enviá-la para a loja correspondente. Para testar esta funcionalidade foram feitas diversas encomendas da *store* para o *warehouse*.

Confirmação de encomenda

Ao aceder à aplicação da *store* o utilizador terá acesso a todas as encomendas pendentes, com o seu ID, título do livro para encomenda e quantidade. Ao selecionar uma encomenda o utilizador poderá aceitar a encomenda, ou seja, adicioná-la ao stock existente e responder aos pedidos de livros pendentes. Para testar esta funcionalidade foram aceites diversas encomendas do *warehouse* para a *store*.

Visualização de pedido de encomenda

Ao aceder ao *website* da loja um utilizador que tenha uma encomenda pendente poderá visualizar o estado da sua encomenda. Para tal terá de indicar o ID da sua

encomenda que lhe foi previamente enviado para o e-mail aquando do pedido de encomenda e ser-lhe-á mostrado o estado atual da encomenda, com o ID da encomenda, nome do livro, quantidade encomendada e estado atual da encomenda do livro. Para testar esta funcionalidade foram efetuadas diversas encomendas de livros.

Modo de Funcionamento

Dependendo do tipo de utilizador a aplicação e o seu modo de funcionamento diferem um pouco. Começando com o utilizador do *website*, este poderá visualizar todos os livros existentes no sistema e poderá fazer encomendas que mais tarde poderão ser consultadas no mesmo *website*.

Com esta lista de livros o utilizador poderá seleccionar um para encomendar ou comprar. Quando é feito um pedido para encomenda de livros é enviado um email para o utilizador com o título do livro, data em que o livro vai ser enviado, quantidade, preço por livro e preço da encomenda e ainda um ID da encomenda para que possa ser mais tarde consultada no *website*.

Caso a aplicação acedida seja a da *store*, o utilizador terá uma lista de livros de entre os quais poderá fazer encomendas ao *warehouse* ou vendas diretas a clientes, onde terá de registar essa venda com os dados do cliente. Quando a loja efetua uma encomenda ao *warehouse* é enviada uma mensagem através de uma *message queue* ao *warehouse*. Nesta aplicação será permitido também ao utilizador aceitar pedidos de encomenda pendentes. Ao aceitar as encomendas por parte do *warehouse* estas serão adicionadas ao stock da loja e serão também satisfeitos pedidos de livros pendentes.

Quando a aplicação acedida é a da *warehouse* o utilizador apenas terá disponível uma lista de todos os pedidos de encomenda, sendo que pode enviar para a loja os que ainda não foram enviados. Nesta aplicação o utilizador também poderá utilizar diferentes filtros ("All", "Dispatched", "Waiting Expedition") de visualização de dados para facilitar a pesquisa de informação.

Instruções para construir e usar todo o demonstrador

Inicialmente, o utilizador deve correr os três servidores - *website*, *warehouse* e *store* - com os seguintes comandos:

- *website*: npm start
- *store*: node store-server.js
- *warehouse*: node warehouse-server.js

De seguida, deve correr os executáveis “Store.exe” e “Warehouse.exe” que abrem, respetivamente, as aplicações da *store* e do *warehouse*. O *website* deve ser acedido através do endereço <http://localhost:3000>.

Na *store* estarão listados todos os livros existentes e todas as encomendas pendentes, isto é, que têm de ser aceites para serem adicionadas ao *stock*. Carregando num livro, aparece, no lado direito, a informação relativa a esse livro e é possível efetuar a compra do mesmo ou fazer um pedido de *stock* ao *warehouse*. Estas ações podem ser realizadas clicando nos respectivos botões.

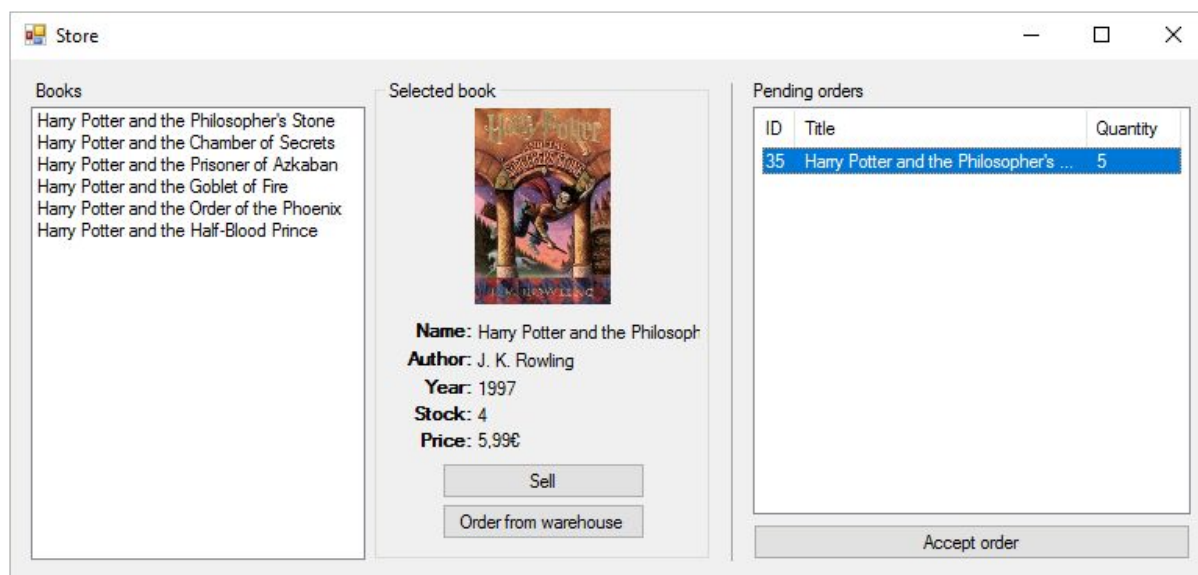


Fig. 1: *Store* com um livro selecionado e uma encomenda pendente

Carregando no botão “sell”, é aberto um *form* onde é possível colocar o nome do cliente e a quantidade de livros pretendida.

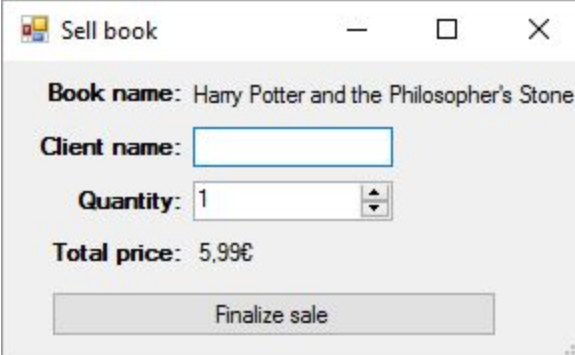
A screenshot of a Windows-style window titled "Sell book". Inside the window, there are four labels with corresponding input fields: "Book name:" followed by the text "Harry Potter and the Philosopher's Stone"; "Client name:" followed by an empty text box; "Quantity:" followed by a spinner box containing the number "1"; and "Total price:" followed by the text "5,99€". At the bottom of the window is a button labeled "Finalize sale".

Fig. 2: *Form* de venda

Carregando no botão “order from warehouse”, é aberto um *form* onde é possível escolher a quantidade que se pretende pedir.

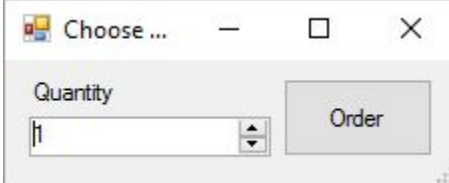
A screenshot of a Windows-style window titled "Choose ...". Inside the window, there is a label "Quantity" above a spinner box containing the number "1". To the right of the spinner box is a button labeled "Order".

Fig. 3: *Form* de encomenda

No *warehouse* estarão listadas todas as encomendas feitas. É possível selecionar uma encomenda e enviá-la para a *store*, carregando no botão “ship order” e filtrar as encomendas pelo seu estado - *dispatched*, *waiting expedition*.

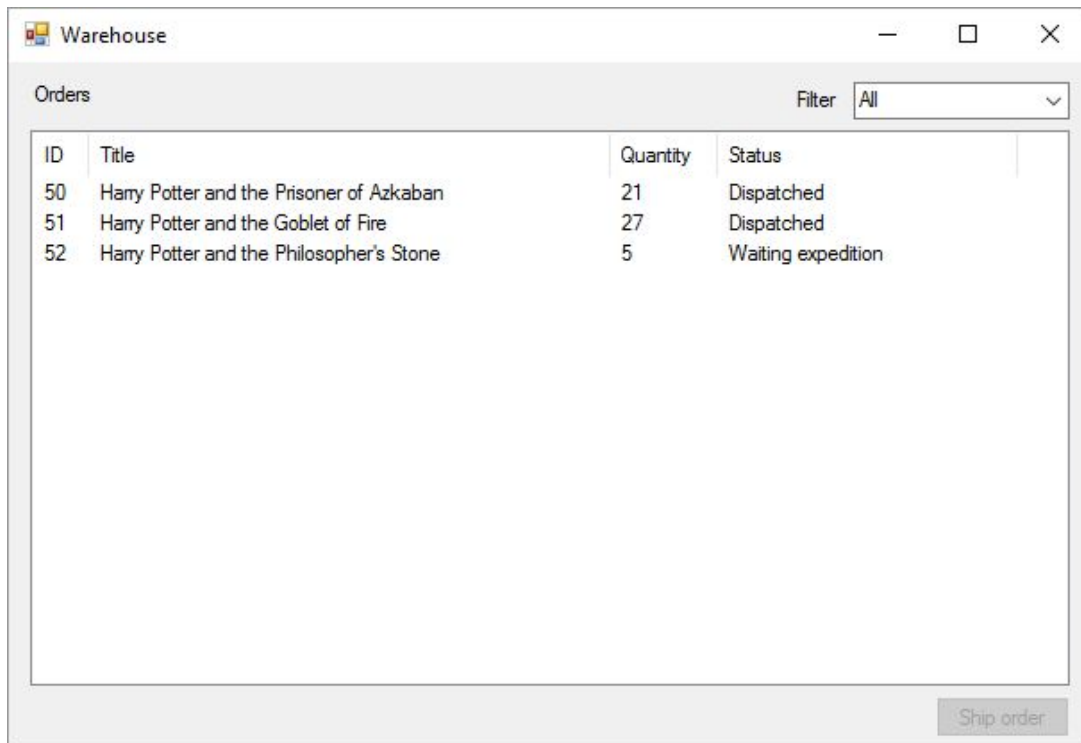


Fig. 4: Warehouse com o filtro a mostrar todas as encomendas

A página inicial do website permite visualizar todos os livros disponíveis para compra.

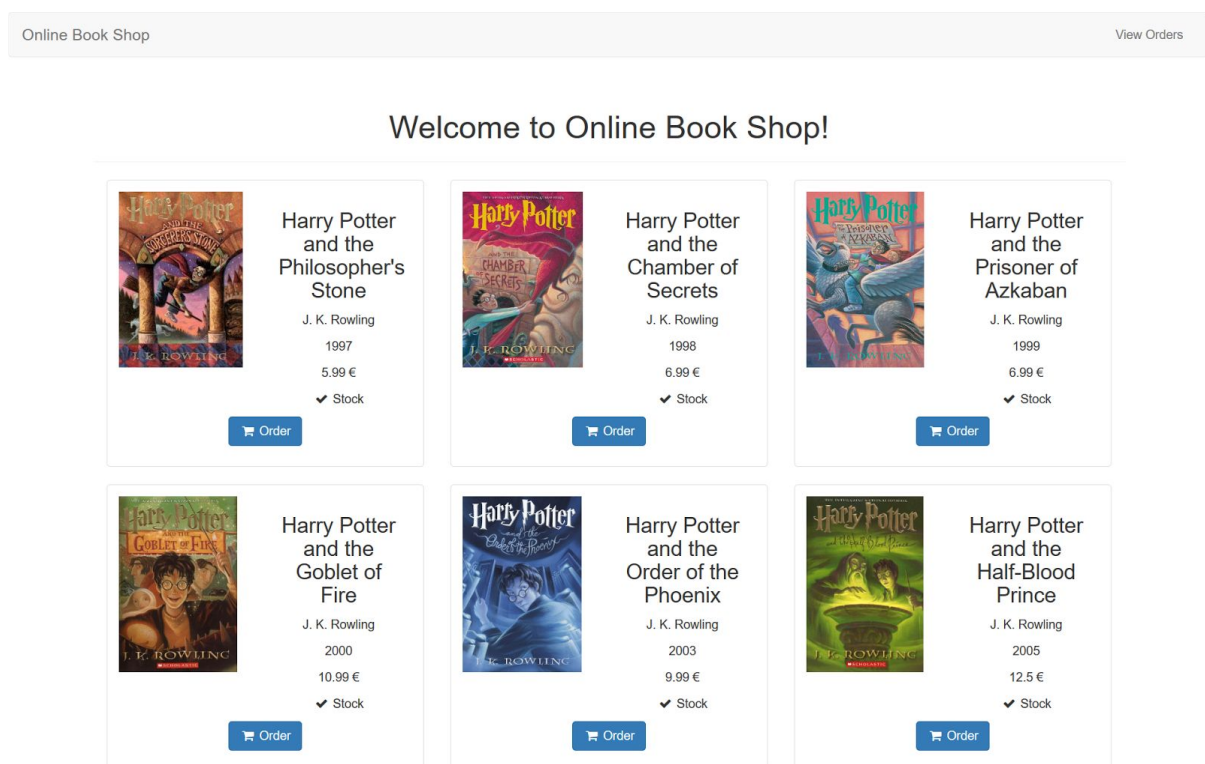


Fig. 5: Página home do website

Carregando no botão “order”, é aberto um formulário onde é pedido para inserir o nome, a morada e o e-mail do cliente, bem como a quantidade de livros pretendida.

The screenshot shows a web application titled "Online Book Shop" with a "View Orders" link in the top right. A modal window titled "Complete your order for Harry Potter and the Philosopher's Stone :" is centered. The form contains the following fields: "Full Name:" with a text input, "Address:" with a text input, "Mail:" with a text input, "Price: 5.99 €", a checked "Stock" status, and a quantity selector set to "1". At the bottom right of the modal are "Order" and "Cancel" buttons. The background shows a grid of book listings, including "Harry Potter and the Philosopher's Stone", "Harry Potter and the Prisoner of Azkaban", "Harry Potter and the Goblet of Fire", "Harry Potter and the Order of the Phoenix", and "Harry Potter and the Half-Blood Prince".

Fig. 6: Formulário de encomenda no *website*

Caso exista *stock* suficiente, carregando no botão “order” o utilizador receberá um e-mail a dizer que a encomenda será enviada. Se não existir *stock* suficiente, o e-mail irá conter um ID que pode ser utilizado para visualizar a encomenda no *website*. Para tal, o utilizador deve carregar em “view order”, no canto superior direito e inserir o ID da encomenda.

The screenshot shows the same "Online Book Shop" interface. A modal window titled "View your order:" is centered. It contains an "Order ID:" label and a text input field with a small search icon on the right. At the bottom right of the modal are "View" and "Cancel" buttons. The background shows the same grid of book listings as in Figure 6.

Fig. 7: Formulário de visualizar encomenda

Inserindo o ID e carregando em “view”, o utilizador será redirecionado para uma página com os detalhes da encomenda.

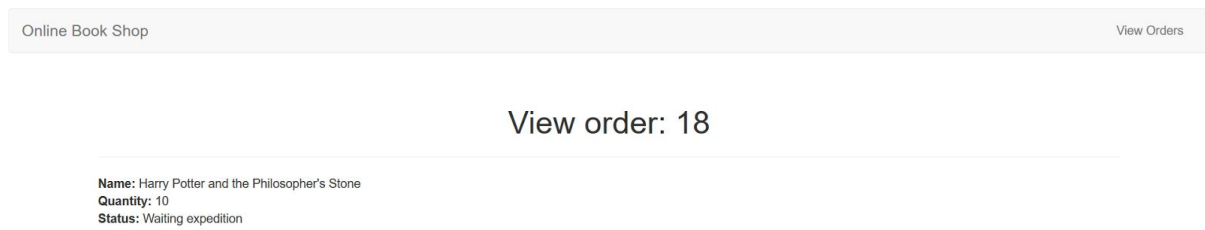


Fig. 8: Detalhes da encomenda

Conclusão

Com este projeto o grupo acredita ter adquirido competências básicas do domínio de aplicações com uma arquitetura *service-oriented*, bem como da utilização de *web sockets* para comunicação entre aplicações C# e, neste caso, servidores *Node.js*.

A aplicação desenvolvida permite vender livros, através do *website* ou diretamente da loja e, ainda, fazer encomendas ao armazém. Os utilizadores que fazem a compra pelo *website* recebem o e-mail com o estado da sua encomenda. Por sua vez, o armazém pode visualizar todas as encomendas feitas e enviar para a loja as que ainda não foram enviadas.

Todas as funcionalidades descritas no enunciado do projeto foram implementadas.

Referências

Referências bibliográficas

- Apontamentos das aulas teórico-práticas disponibilizados na página oficial da unidade curricular (<https://paginas.fe.up.pt/~apm/TDIN/main.html>)
- Instruções de utilização disponibilizadas na página do GitHub do “SocketIoClientDotNet” (<https://github.com/Quobject/SocketIoClientDotNet>)
- Tutorial para Javascript disponibilizado na página oficial do RabbitMQ (<https://www.rabbitmq.com/tutorials/tutorial-one-javascript.html>)

Software utilizado

- **“Visual Studio Enterprise 2015”** (<https://www.visualstudio.com/vs>)
Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft, used to develop computer programs for Windows, as well as websites, web applications, web services and mobile applications. Visual Studio integrates Microsoft software development platforms such as Windows API, Windows Forms, Windows Communication Foundation (WCF), Windows Presentation Foundation (WPF), Windows Store and Microsoft Silverlight. It can produce both native code and managed code.
- **“IntelliJ IDEA”** (<https://www.jetbrains.com/idea/>)
IntelliJ IDEA is a Java integrated development environment (IDE) for developing computer software. It is developed by JetBrains (formerly known as IntelliJ).
- **“RabbitMQ”** (<https://www.rabbitmq.com/>)
RabbitMQ is open source message broker software (sometimes called message-oriented middleware) that implements the Advanced Message Queuing Protocol (AMQP). The RabbitMQ server is written in the Erlang programming language and is built on the Open Telecom Platform framework for clustering and failover. Client libraries to interface with the broker are available for all major programming languages.
- **“SocketIoClientDotNet”** (<https://github.com/Quobject/SocketIoClientDotNet>)
Socket.IO Client Library for C#, which is ported from the JavaScript client version 1.1.0. This library supports all of the features the JS client does, including events, options and upgrading transport.