Universidade do Porto

Faculdade de Engenharia

FEUP

Distribution and Integration Technologies

ASSIGNMENT 2: AN ENTERPRISE DISTRIBUTED SYSTEM

Professor António Miguel Pontes Pimenta Monteiro

Ruben Fernando Pinto Cordeiro – ei11097
Duarte Nuno Pereira Duarte – ei11101

June 3, 2015

Contents

Introduction

The following report describes the implementation of *an enterprise distributed system*. The system is a book order platform that allows end-users to buy books online.

Functionalities

The set of features of the system are represented in the use case diagram below.

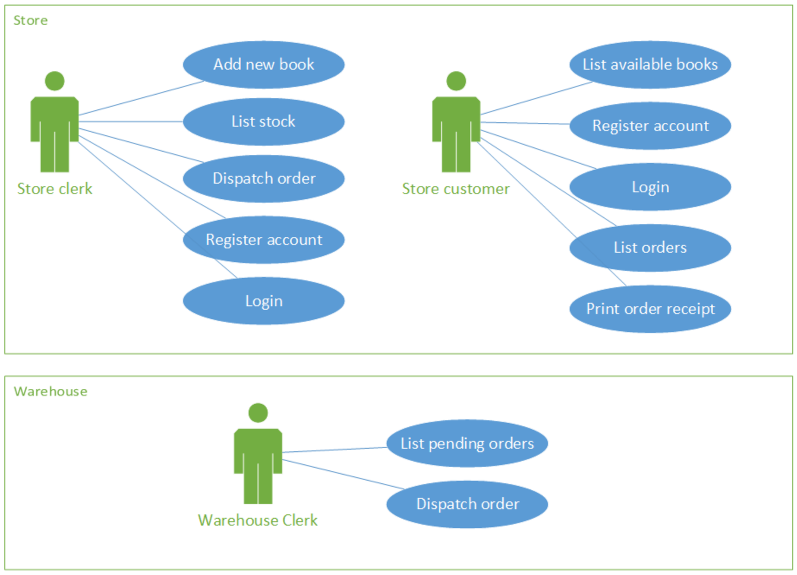


Figure 1: Use case diagram

Technologies

Both the store and warehouse servers are implemented in *Node.js*, with two different frameworks: *hapi.js* and S*ails.js*.

The store and warehouse clients are implemented in *Angular.js*.

There were several reasons that led us to choose Node.js over other alternatives for the back-end.

- Node.js really excels at I/O bound operations. In our application, data streaming and database accesses are the most common operations. There are barely any CPU intensive tasks that would block the main loop.
- The node package manager (NPM) has evolved into an extremely rich registry of modules that are ready to use.
- The fact that both the client and the server are written in *JavaScript* promotes some code reuse between the client and the server.

Instead of a set of static HTML pages, both of our client applications are *AngularJS* single page applications.

The adoption of this framework has several design goals:

- Decouple *DOM* manipulation from application logic. This substantially reduces boilerplate code and improves modularity and testability.
- Decouple the client side of the application from the server side. This reduces much of the burden on the backend and allows development work to progress in parallel with minor hiccups.
- Provide an incremental and iterative development process for the client, from the UI design to the writing of the business logic.

The store server and warehouse server communicate via a *RabbitMQ* remote data store. *RabbitMQ* provides robust messaging for applications with a high level of throughput.

The database instances of both the store and warehouse servers were built with *PostgreSQL*. The data domain fits the relational model perfectly. The fact that *Postgres* has high performance and is open source have also contributed to the choice of this technology.

Architecture

REMOTE QUEUE COMMUNICATION

The store server communicates with the warehouse server through a *RabbitMQ* remote data store.

The sequence diagram below describes a communication scenario for the whole system.

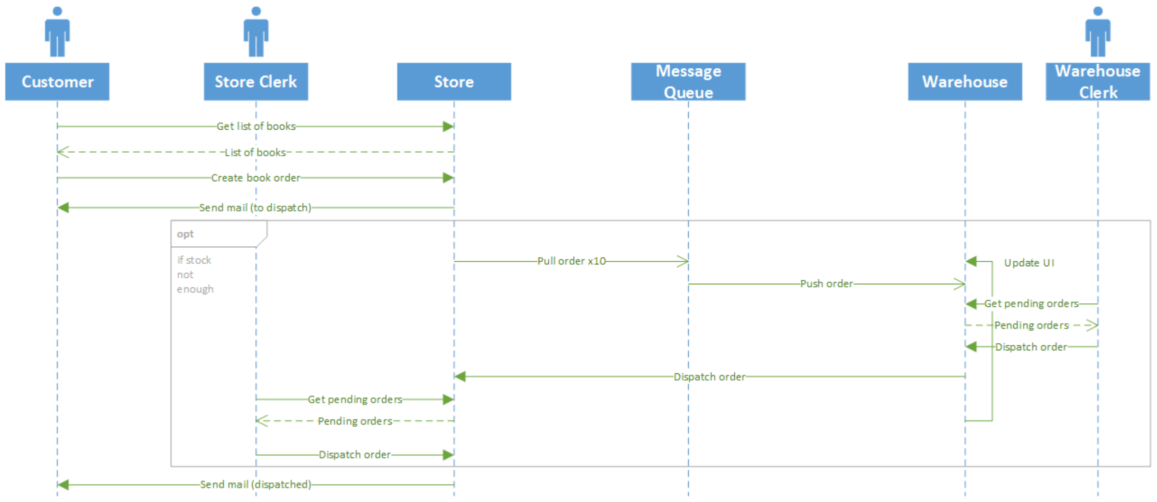


Figure 2: Book order creation sequence diagram

PHYSICAL ARQUITECTURE

The system is based on a client-server model. The client is a web application that communicates with the server via a REST API with the store server. The store and warehouse servers' data persistence is assured by *PostgreSQL* database.

A diagram with the physical architecture can be seen below:

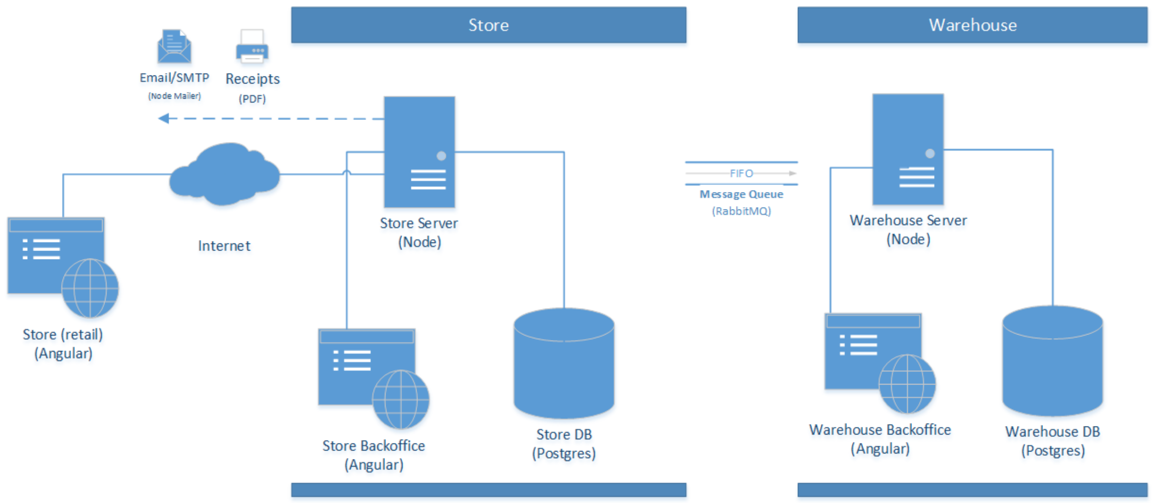


Figure 3: Physical Architecture Diagram

Domain model

STORE

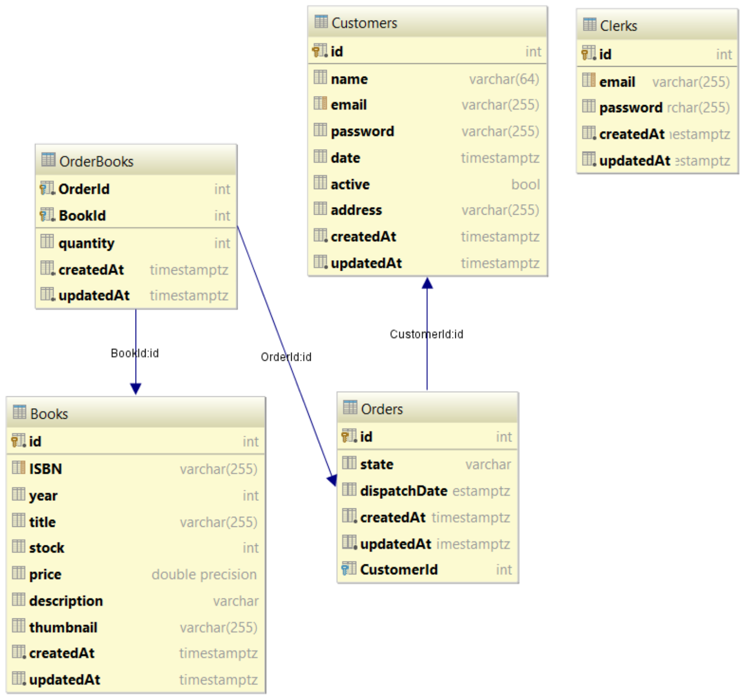


Figure 4: Store server data model

WAREHOUSE

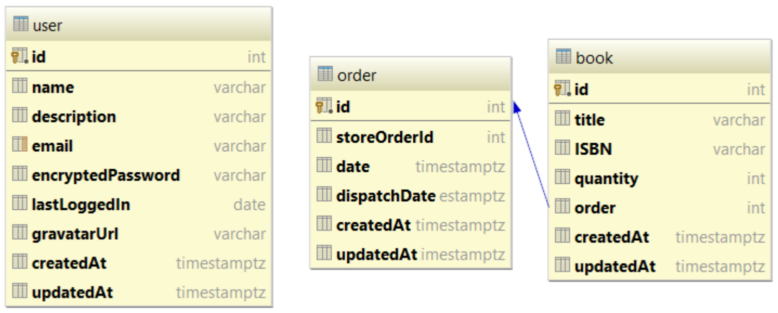
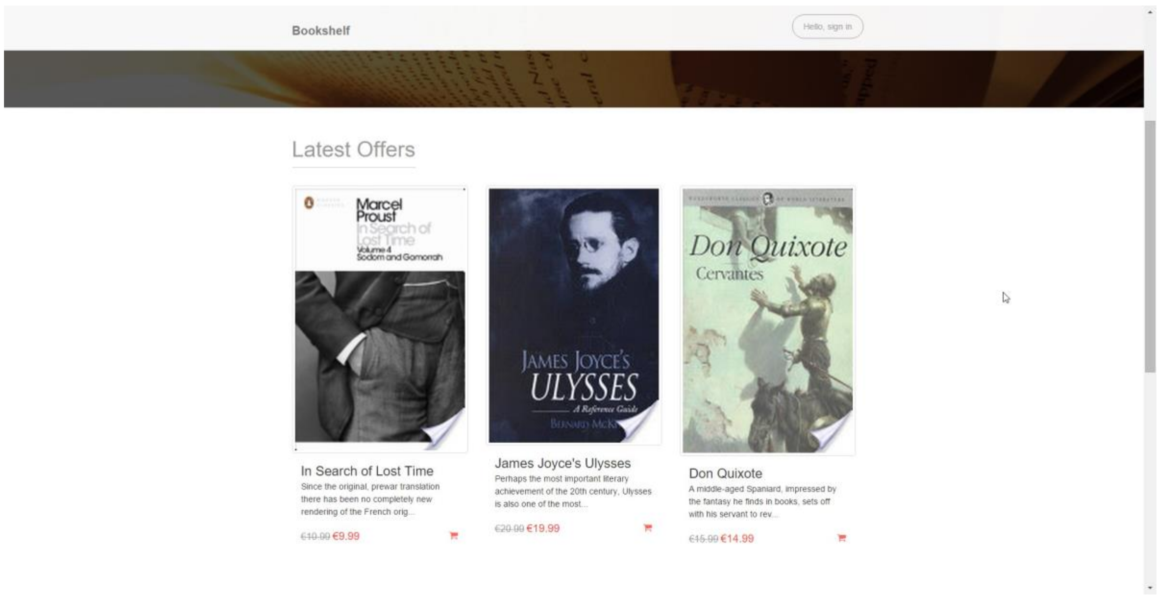


Figure 5: Warehouse server data model

User interfaces

STORE

Customer

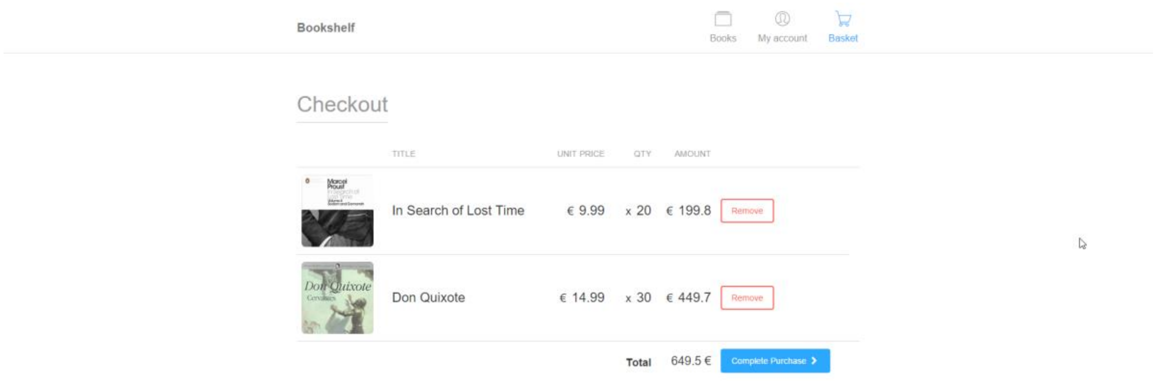


In Search of Lost Time
Since the original, prewar translation there has been no completely new rendering of the French orig...
~~€10.99~~ €9.99

James Joyce's Ulysses
Perhaps the most important literary achievement of the 20th century, Ulysses is also one of the most...
~~€20.99~~ €19.99

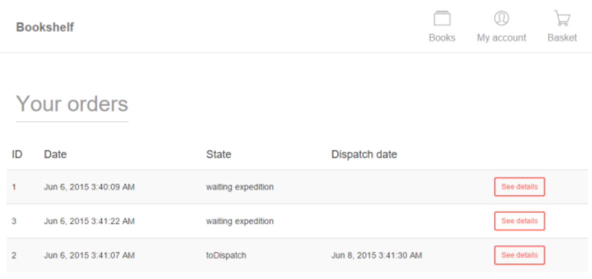
Don Quixote
A middle-aged Spaniard, impressed by the fantasy he finds in books, sets off with his servant to rev...
~~€15.99~~ €14.99

Figure 6: Store homepage



	TITLE	UNIT PRICE	QTY	AMOUNT	
	In Search of Lost Time	€ 9.99	x 20	€ 199.8	Remove
	Don Quixote	€ 14.99	x 30	€ 449.7	Remove
			Total	649.5 €	Complete Purchase

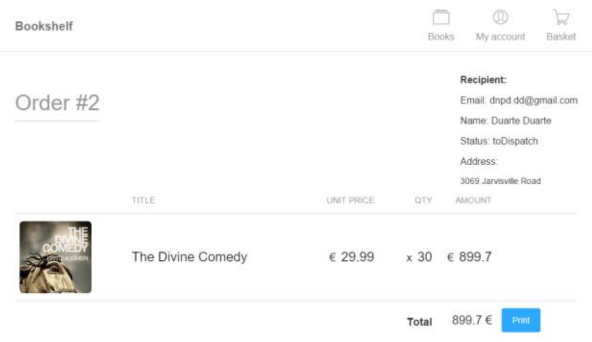
Figure 7: Store checkout page



Your orders

ID	Date	State	Dispatch date	
1	Jun 6, 2015 3:40:09 AM	waiting expedition		See details
3	Jun 6, 2015 3:41:22 AM	waiting expedition		See details
2	Jun 6, 2015 3:41:07 AM	toDispatch	Jun 8, 2015 3:41:30 AM	See details

Figure 8: Store customer's orders



Order #2

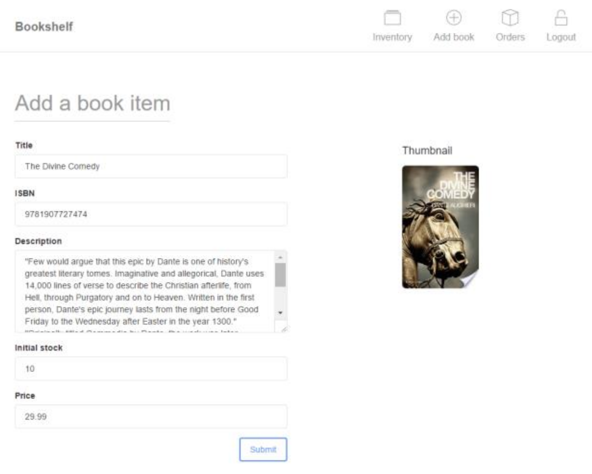
Recipient:
Email: dnpd.dd@gmail.com
Name: Duarte Duarte
Status: toDispatch
Address:
3069 Jarvisville Road

	TITLE	UNIT PRICE	QTY	AMOUNT
	The Divine Comedy	€ 29.99	x 30	€ 899.7

Total 899.7 €

Figure 9: Customer's order details

Store Clerk



Add a book item

Title: The Divine Comedy

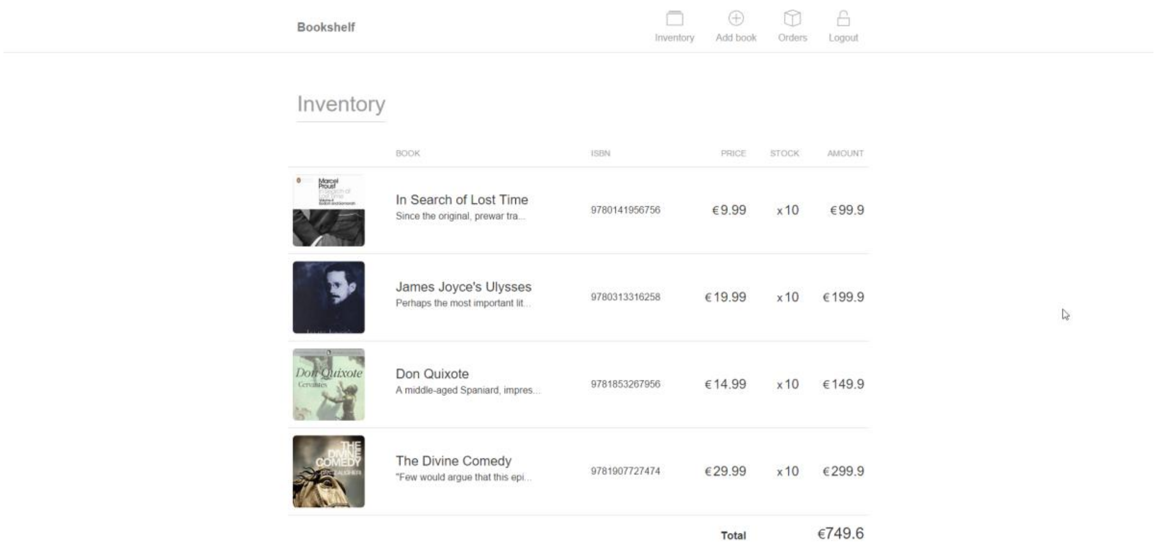
ISBN: 9781907727474

Description: "Few would argue that this epic by Dante is one of history's greatest literary tomes. Imaginative and allegorical, Dante uses 14,000 lines of verse to describe the Christian afterlife, from Hell, through Purgatory and on to Heaven. Written in the first person, Dante's epic journey lasts from the night before Good Friday to the Wednesday after Easter in the year 1300."

Initial stock: 10

Price: 29.99

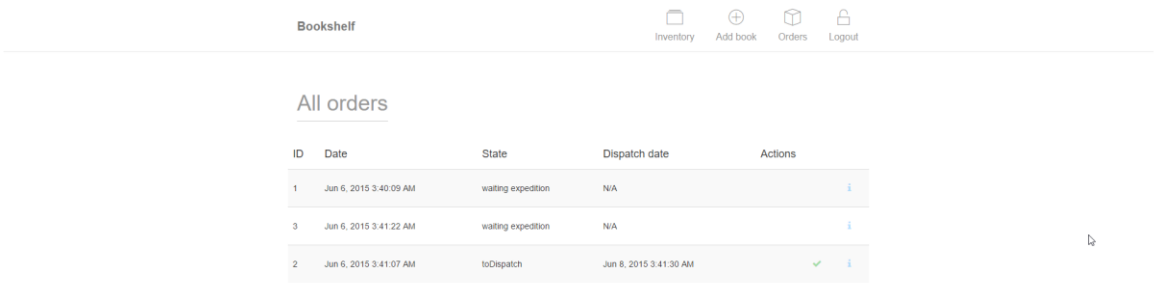
Figure 10: Clerk's add book



Inventory

	BOOK	ISBN	PRICE	STOCK	AMOUNT
	In Search of Lost Time Since the original, prewar tra...	9780141956756	€9.99	x10	€99.9
	James Joyce's Ulysses Perhaps the most important lit...	9780313316258	€19.99	x10	€199.9
	Don Quixote A middle-aged Spaniard, impres...	9781853267956	€14.99	x10	€149.9
	The Divine Comedy "Few would argue that this epi...	9781907727474	€29.99	x10	€299.9
			Total		€749.6

Figure 11: Clerk's inventory/stock view

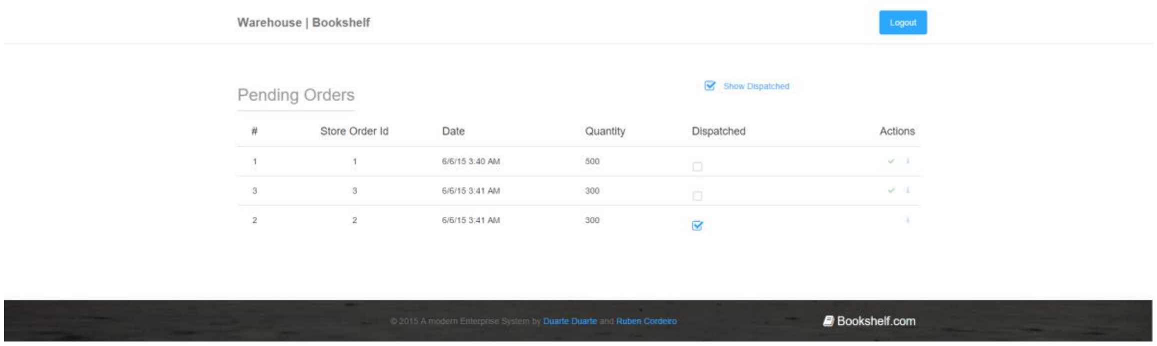


All orders

ID	Date	State	Dispatch date	Actions
1	Jun 6, 2015 3:40:09 AM	waiting expedition	N/A	i
3	Jun 6, 2015 3:41:22 AM	waiting expedition	N/A	i
2	Jun 6, 2015 3:41:07 AM	toDispatch	Jun 8, 2015 3:41:30 AM	✓ i

Figure 12: Clerk's orders view

WAREHOUSE

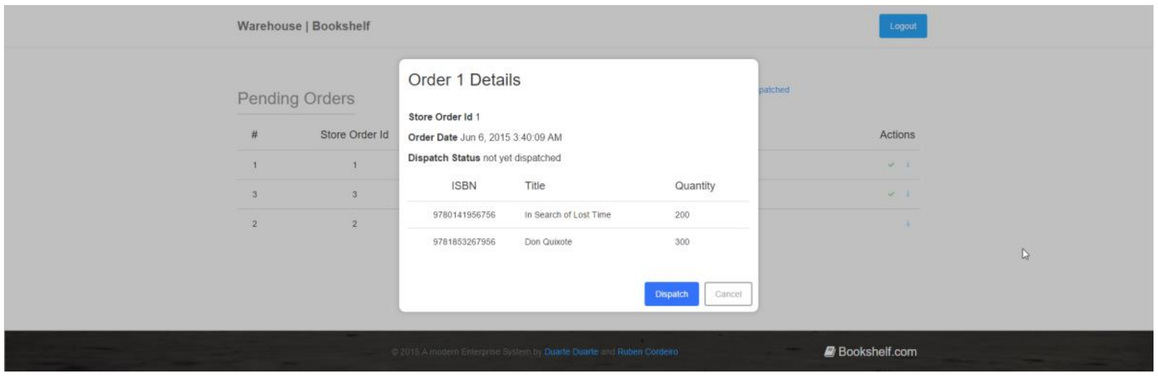


Pending Orders

#	Store Order Id	Date	Quantity	Dispatched	Actions
1	1	6/6/15 3:40 AM	500	☐	
3	3	6/6/15 3:41 AM	300	☐	
2	2	6/6/15 3:41 AM	300	☑	

© 2015 A modern Enterprise System by Duarte Duarte and Ruben Cordeiro

Figure 13: Pending orders to dispatch view



Order 1 Details

Store Order Id 1

Order Date Jun 6, 2015 3:40:09 AM

Dispatch Status not yet dispatched

ISBN	Title	Quantity
9780141956756	In Search of Lost Time	200
9781853267956	Don Quixote	300

© 2015 A modern Enterprise System by Duarte Duarte and Ruben Cordeiro

Figure 14: Order details view

Testing

There are several unit tests for both the store and warehouse servers. Fake *http* requests are injected into the server for the test suites.

Several integration and end-to-end tests were made for the clients. Two technologies were used: *protractor and karma*.

KARMA

This running environment is used for the unit testing of small "units" of code, such as isolated views, controllers and services.

Karma is a tool which spawns a web server that executes source code against test code for each of the browsers connected.

Each time a test is run, Karma records its status and then tallies up which browsers have failed for each test and which ones passed and timed out. This makes each test work 100% natively in each browser without the need to test individually.

PROTRACTOR

Karma provides a bundled AngularJS Scenario Runner, however, it was deprecated in favor of a more robust end-to-end testing platform: Protractor. End-to-end tests are equivalent to acceptance tests.

By translating the formal tests to a Protractor testing suite, we are ensuring that the application respects the specification requirements. It does not replace the traditional deployment of the application to the end user for testing, but it does help in the development process as an additional insurance.

Instructions

Required software

- Node.js (https://nodejs.org/)
- npm (https://www.npmjs.com/) (included with Node)
- Bower (http://bower.io/) (`npm install bower -g`)
- Sails.js (http://sailsjs.org/) (`npm install sails -g`)
- PostgreSQL (http://www.postgresql.org/)
- RabbitMQ (https://www.rabbitmq.com/)

Store

1. Edit `store-server/config/default.json` (namely database access and email account).
2. `cd store-server`
3. `npm install`
4. `cd public && bower install`
5. `cd store-server && node server.js`
6. `node server.js`
7. ..
8. Browse to http://localhost:8000
9. Register/Login
   a. Clerk: username: clerk@bookshelf.com – password: bookshelf
   b. Customer: create an account
10. ..
11. Profit!

Warehouse

1. Edit `warehouse/configs/connections.js`
2. `cd warehouse`
3. `npm install`
4. `bower install`
5. `node app.js`
6. ..
7. Browse to http://localhost:1337
8. Create an account
9. Login
10. ..
11. Profit!