ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

P.PORTO

Programação em Ambiente Web

Livraria

Licenciatura em Engenharia Informática

2021/2022

Grupo 13

Josué Natanael Cardoso de Freitas, nº8200308

Nuno Dinis Gonçalves Ribeiro, nº 8200309

Índice

Índice de figuras

Introdução

O âmbito deste projeto nasce do objetivo de modernizar e digitalizar uma empresa do ramo bibliotecário. Esta digitalização deverá manter o ADN da empresa sendo através do mesmo ser possível: gerir os livros, sendo que estes podem ser novos ou usados; registar e gerir clientes, onde o registo do cliente poderá acontecer de forma autónoma por parte do cliente ou por um funcionário de forma manual; registar compras e vendas, em que é necessário que a livraria tenha toda a informação relativa às vendas quer às compras (por compras entende-se a livraria a comprar o livro usado ao cliente e venda a compra é feita do cliente à livraria) no sistema, este registo poderá ser feita de forma manual no BackOffice pelo funcionário ou de forma autónoma pelo cliente numa compra na loja online da livraria; gerir um programa de fidelização de forma a atrair cliente e destacar-se das livraria tradicionais, em que é possível editar os valores de forma a aumentar ou diminuir o benefícios dos clientes.

Faz parte também desta renovação digital a criação e implementação de uma aplicação web que permita ao próprio cliente comprar, vender e avaliar livros, alterar e ver as suas compras e vendas ou mesmo subscrever a newsletter.

Como esta modernização pretende-se expandir o negócio de forma que os clientes possam comprar livros online, melhorar o processo de compra e venda de livros usados e venda de livros novos por parte dos funcionários, tornando o processo mais simples e menos propenso a falhas, assim como permitir a criação e gestão de um programa que permita recompensar os clientes pela quantidade de compras feitas.

Devido à complexidade e âmbito do projeto, o mesmo encontra-se dividido em duas *Milestones*. Na primeira *Milestone* consta o desenvolvimento do *backoffice* da aplicação*, ou seja,* a aplicação usada única e exclusivamente pelos funcionários. Na segunda *Milestone* foi desenvolvido o *frontoffice*, a aplicação que permitirá ao cliente realizar compras online.

Ao longo deste relatório será detalhado o trabalho desenvolvido em ambas as *Milestones*, abordando a forma de como é que cada uma destas funcionalidades foi desenvolvida, a sua contextualização e especificação incluindo também uma apreciação crítica e técnica.

Primeira *Milestone*

Identificação do projeto

Este projeto teve início com a análise e estudo do modelo de negócio com o objetivo de definir de forma mais clara o âmbito do projeto. A análise foi feita principalmente através da observação e conhecimento geral de plataformas semelhantes. Após a pesquisa pudemos concluir e classificar as restrições e especificações dos vários subsistemas existentes neste projeto.

Gestão de Funcionários

Como referido anteriormente, na primeira *Milestone* consta o desenvolvimento do *BackOffice*, sendo por isso apenas acessível pelos funcionários, fazendo deste o subsistema a base para todas as outras operações.

Foi identificado que cada funcionário só poderá ter acesso ao sistema e poder desempenhar o seu trabalho após o registo do mesmo por parte do Gerente de loja. Para além dos dados pessoais de cada funcionário, faz parte do seu registo o email e password que serão utilizadas para a sua autenticação. Considerando que a autenticação será feita a partir do email do funcionário este deverá ser único no sistema.

Para além de um funcionário poder ser facilmente identificado através do seu email, aquando do seu registo é deverá ser associado um ID único, inteiro e incremental, de forma que mesmo fora do sistema a sua identificação ocorra de uma maneira simples e eficaz. Devido a essa especificação, todos os dados de um funcionário podem sofrer alterações com exceção do seu ID.

Considerando que existe informação sensível que só poderá ser acedida por um trabalhador com um cargo mais elevado, é necessário existir dois tipos de funcionários correspondente à sua autorização no sistema: Funcionário e Gerente. O Funcionário com autorização de Funcionário só poderá ter acesso às funções mais básicas e apropriadas ao seu cargo, ou seja, registo e gestão de livros, registo e gestão de clientes e o registo de compra e venda de livros. O Funcionário com autorização de Gerente tem acesso a todas as funcionalidades do sistema, isto é, para além das funcionalidades associadas ao Funcionário, tem acesso à gestão de funcionários (que inclui as operações CRUD) e a gestão de pontos.

Gestão de Livros

Dado que o negócio em execução é uma livraria, temos que o principal e único produto, atualmente, são os livros, quer sejam estes para serem vendido ao publico como unidades novas, quer seja para serem comprados livros usados aos clientes que posteriormente serão vendidos na mesma condição de usados.

Para dar solução à especificação do domínio de negócio em que existe o comercio de livros usados, foi considerado que um livro poderá ter cinco tipos diferentes de

condições: novo, excelente, bom, médio e mau. Com esta solução, todos os livros poderão ser identificados segundo a sua condição, havendo a consequência de as diferentes condições apresentarem diferentes tipos de preços.

Para o cálculo dos diferentes tipos de preços, foi utilizado um preço de referência, o valor do livro novo para venda ao público, fruto deste preço os restantes são calculados segundo um critério de valorização, por exemplo, um livro em condição "excelente" será vendido por 80% do valor base e será comprado a um cliente por 60% do valor base. Com esta especificação é tendo em conta o lucro da livraria, bem como uma transparência com o cliente.

Todos os livros são únicos, onde o identificador é o ISBN, ou seja, a nível de armazenamento todos os livros iguais serão resumidos a uma entrada na base de dados, onde os dados que serão dinâmicos é o stock e o preço das diferentes condições.

As operações essenciais sobre a gestão do livro são as operações CRUD, onde passa por criar o livro, se houver necessidade editar as características do livro, visualizar as informações do livro e eliminar, se houver necessidade, o livro.

Dentro do backoffice, é garantido que todo o tipo de funcionário tem acesso livre às operações CRUD, não havendo restrições pelo nível de utilizador.

Gestão de Clientes

Os clientes são uma peça fundamental neste negócio e por isso é importante haver informações dos mesmos.

O cliente pode estar ligado às atividades do sistema de forma direta ou indireta, com acesso direto ao frontoffice, onde são estes que realizam as operações de forma autónoma, ou pelo backoffice com a inserção manual do funcionário, desde criar a conta de cliente a registar uma venda ou compra do cliente.

Numa lógica semelhante à forma como os livros são armazenados os clientes também possuem dois tipos de informação, uma que servirá para identificar o cliente com os dados pessoais e outra que irá possuir a informação sobre o programa de fidelização.

Gestão de Pontos

Faz parte do âmbito deste projeto a criação e gestão de um programa de fidelização de clientes que incentive, promova e impulsione as vendas e compras de livros. Este programa visa a recompensar os clientes pelas suas compras e vendas continuas angariando pontos à medida que o fazem.

De forma a puder responder a essa necessidade o Gerente de loja pode definir a quantidade de pontos ganhos por cada € gasto na loja, qual o desconto em € correspondente ao uso de cada 100 pontos, pontos necessários para o cliente ter portes grátis (através das compras online), pontos ganhos por cada livro que a loja compra ao cliente e a quantidade de pontos ganhos no registo do mesmo segundo o seu intervalo de idade.

Esta área de negócio não é essencial, ou seja, é possível fazer compras e vendas sem um programa de fidelização. Devido às características desta área, só é possível ter um programa de fidelização e faz parte dele as quatro operações *CRUD*.

Registo compra e venda de livros usados e venda de livros novos

O registo de compras e vendas é considerado a principal área e é a funcionalidade que vai juntar todos os subsistemas de forma a puder auxiliar no processo de modernização da livraria.

Segundo a regra de negócio, terá de ser possível registar compra e venda de livros usados e a venda de livros novos. Como especificado anteriormente a compra de livros usados é feita aos clientes e esses mesmos livros possuem as várias condições de estado que correspondem a preços tabelados. Essas duas transações também foram separadas a nível de documentos, ou seja, para as vendas o documento é sempre fatura, quando estamos perante as compras trata-se de uma nota de crédito.

No processo de compra/venda é necessário identificar sempre o cliente e como é claro os livros que fazem parte da transação e as suas quantidades especificas. Faz também parte da transação a utilização e aplicação do programa de fidelização (caso esteja em vigor) assim como a atualização dos stocks abrangidos.

Como se trata de dados de faturação, dado as suas características, apenas é possível criar e ver transação, não sendo possível alterar ou eliminar a mesma.

Desenvolvimento do Projeto

Considerando as especificações de cada subsistema e o âmbito do projeto, foi possível definir inicialmente as várias etapas e a sua ordem lógica de implementação. Na figura abaixo é detalhado essa ordem e a que membro ficou atribuído a responsabilidade da sua implementação.

TAREFA	ATRIBUÍDO A	INÍCIO PREVISTO	TÉRMINO PREVISTO
Análise e estudo do negócio	Ambos	7/4/22	10/4/22
Registo e gestão de funcionários	Josué Freitas	11/4/22	15/4/22
Registo e gestão de de livros	Nuno Ribeiro	11/4/22	17/4/22
Registo e gestão de clientes	Nuno Ribeiro	17/4/22	20/4/22
Gestão de pontos	Josué Freitas	16/4/22	17/4/22
Autenticação de utilizadores	Josué Freitas	18/4/22	20/4/22
Registo venda livros usados e novos	Nuno Ribeiro	20/4/22	26/4/22
Registo compra livros usados	Josué Freitas	20/4/22	26/4/22
Melhorias gerais	Ambos	26/4/22	3/5/22

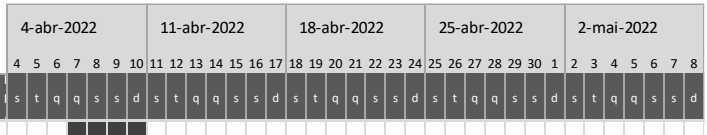


Figura 1 - Diagrama de desenvolvimento previsto inicialmente

Especificação geral

Livros

Numa primeira fase foi idealizado qual seria a melhor estrutura para que os dados dos livros pudessem ser armazenados na base dados e dar suporte a todas as atividades que interajam com estes. Para que tal acontecesse foi criada uma estrutura em que os dados são divididos em dois prepósitos, um destinado a fornecer informação sobre o livro em si e outro existe com o foco de monitorizar a parte logística do livro, por exemplo, fornecer o stock dos diferentes tipos de livros.

É apresentado de seguida a tabela que identifica o nome dos campos, bem como o seu domínio e se este é obrigatório ou não.

<table>
<tr><th colspan="3">Nome campo</th><th>Domínio do campo</th><th>Obrigatório</th></tr>
<tr><td colspan="3">Título</td><td>Apenas letras.
Tamanho: 1 até 60</td><td>Sim</td></tr>
<tr><td colspan="3">Edição</td><td>Apenas números superiores a 0.</td><td>Não</td></tr>
<tr><td colspan="3">Ano de publicação</td><td>Deve ser inferior ou igual ao ano atual.</td><td>Sim</td></tr>
<tr><td colspan="3">Nº páginas</td><td>Número inteiro superior a 1.</td><td>Sim</td></tr>
<tr><td colspan="3">ISBN</td><td>Número no formato ISBN-10 ou ISBN-13.</td><td>Sim</td></tr>
<tr><td colspan="3">Linguagem</td><td>Enumeração entre: "pt", "en", "de", "fr" e "es"</td><td>Sim</td></tr>
<tr><td rowspan="2">Autor</td><td colspan="2">Nome</td><td>Apenas letras.
Tamanho: 1 até 60</td><td>Sim</td></tr>
<tr><td colspan="2">Chave</td><td>String</td><td>Não</td></tr>
<tr><td colspan="3">Gêneros</td><td>Array de string, em que cada posição do array deve ter entre 1 a 60 caracteres.</td><td>Sim</td></tr>
<tr><td rowspan="2">Imagem</td><td colspan="2">Url público</td><td>Caminho público no servidor</td><td rowspan="2">Sim</td></tr>
<tr><td colspan="2">Tipo de ficheiro</td><td>Tipo de ficheiro.
Tipos aceites: png, jpg e jpeg</td></tr>
<tr><td rowspan="5">Stock</td><td colspan="2">Novo</td><td rowspan="5">Número inteiro superior ou igual a 0.</td><td rowspan="5">Sim</td></tr>
<tr><td colspan="2">Excelente</td></tr>
<tr><td colspan="2">Bom</td></tr>
<tr><td colspan="2">Médio</td></tr>
<tr><td colspan="2">Mau</td></tr>
<tr><td rowspan="5">Informação para venda</td><td rowspan="5">Preço</td><td>Novo</td><td rowspan="5">Número inteiro superior ou igual a 0.</td><td rowspan="5">Sim</td></tr>
<tr><td>Excelente</td></tr>
<tr><td>Bom</td></tr>
<tr><td>Médio</td></tr>
<tr><td>Mau</td></tr>
<tr><td rowspan="5">Informação para compra</td><td rowspan="5">Preço</td><td>Novo</td><td rowspan="5">Número inteiro superior ou igual a 0.</td><td rowspan="5">Sim</td></tr>
<tr><td>Excelente</td></tr>
<tr><td>Bom</td></tr>
<tr><td>Médio</td></tr>
<tr><td>Mau</td></tr>
</table>

Havendo chegado à estrutura do documento que irá ser capaz de dar suporte às atividades necessárias, é apresentado e seguida o diagrama de use case que relatas as principais atividades.

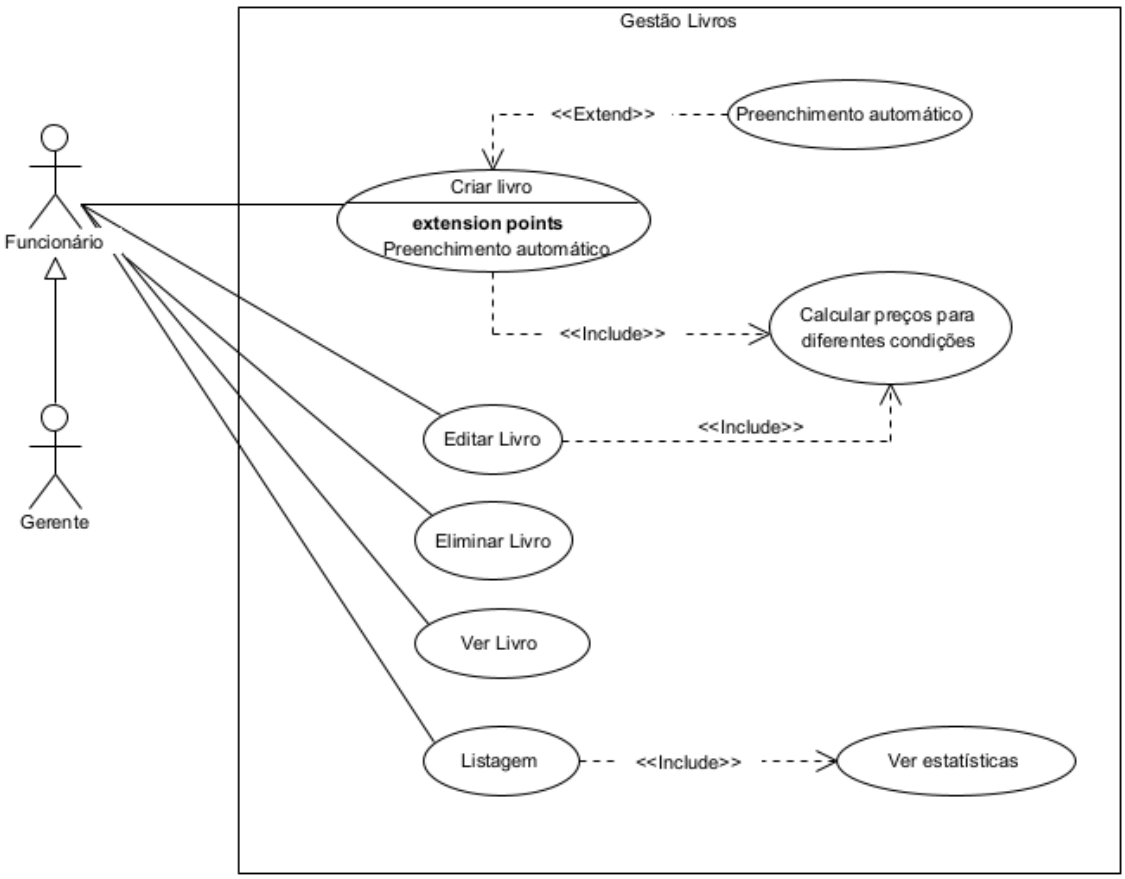


Figura 2 - Use case "Criar livro"

Podemos observar no diagrama que tanto o gerente como o funcionário podem interagir com o módulo sem restrições, pois dado o negócio não é justificado a restrição no contexto da gestão de livros.

É possível observar que no caso de uso “Criar livro” existe uma atividade que estende a atividade principal, sendo esta o “Preenchimento automático”, pois é uma atividade opcional ao cliente, sendo que a sua função é dada um ISBN faz uso de uma API que retém informação de livros com o prepósito de procurar o livro e caso seja encontrado informação será preenchido os campos correspondentes, deixando o campo imagem e a informação logística para o funcionário preencher.

De forma a dar informação visualmente fácil de ser consumida, foi criado dois gráficos. Os dados a serem pesquisados são os gêneros de cada livro e o autor de cada livro, onde é apresentado em formato *doughnut* os cinco autores com mais livros registados no sistema e os gêneros com mais livros.

Para que a implementação e a organização gráfica fossem simples foram criados *mockups* para a listagem e criação de livros com o objetivo de auxiliar nesse sentido.

Figura 3 - Mockup listagem de Livros

.

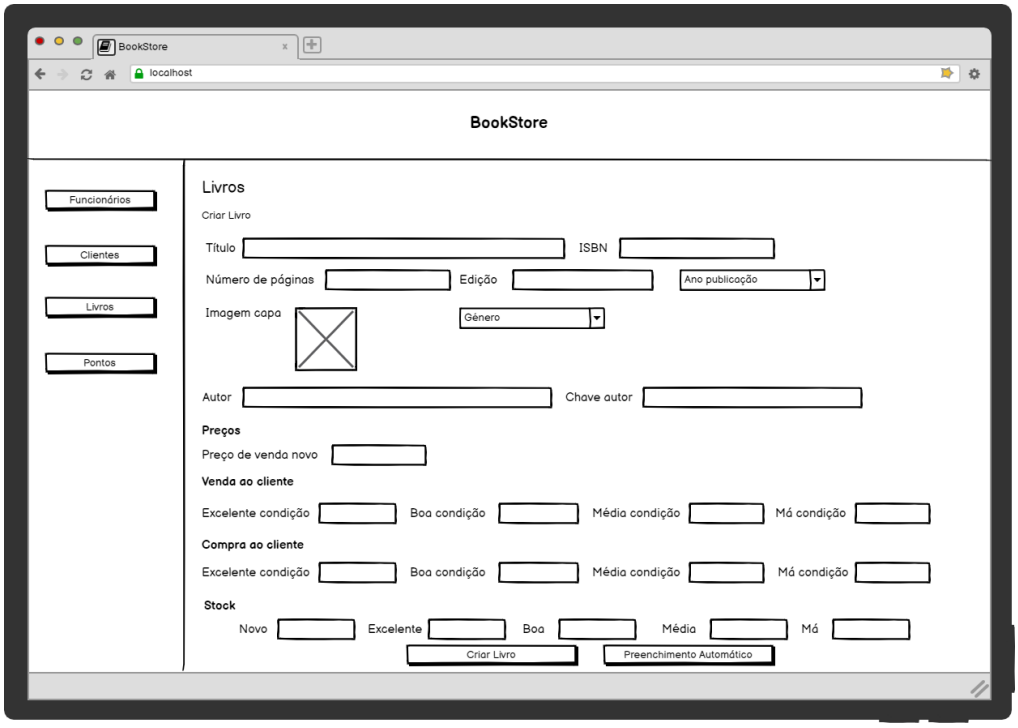


Figura 4 - Mockup criar livros

Funcionários

Considerando a identificação do projeto e também o desenvolvimento do mesmo, foi possível chegar à seguinte estrutura de dados associados a cada funcionário:

Nome campo	Validação	Obrigatório
Id	Identificador único	Não
Nome	*String* com tamanho mínimo de 3 caracteres.	Sim
Género	Apenas “Masculino”, “Feminino” ou “Outro”.	Sim
Data de Nascimento	Data valida no formato YYYY-MM-DD.	Sim
Número de telemóvel	Número inteiro correspondente a número registado em Portugal.	Sim
Rua	*String* com tamanho mínimo de 6 caracteres.	Sim
Cidade	*String* com tamanho mínimo de 3 caracteres.	Sim
Código postal	String no formato XXXX-XXX.	Sim
Tipo de utilizador	Apenas “Funcionário” ou “Gerente”.	Sim
Email	*String* correspondente a email válido	Sim
Palavra-passe	String com tamanho mínimo de 5 caracteres.	Sim

Relativamente ao campo Id estamos perante um dado inteiro e incremental, ou seja, não deverá ser introduzido pelo utilizador e sim atribuído automaticamente pelo sistema. Sobre o email é importante relembrar que aquando da inserção de um novo funcionário o sistema deverá verificar se o email introduzido já não está atribuído a outro funcionário, não permitindo existir dois funcionários com o mesmo email.

Segundo o levantamento de requisitos foi possivel construir o seguinte o use case com as funcionalidades que fazem parte da gestão de funcionários:

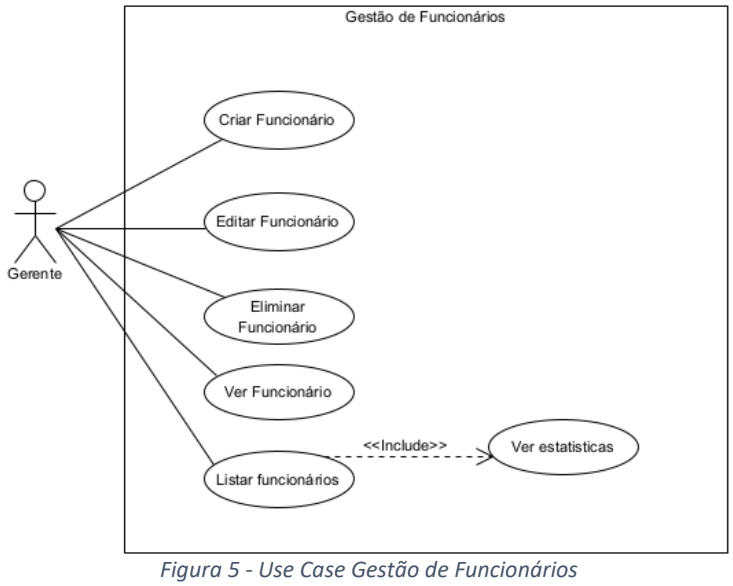


Figura 5 - Use Case Gestão de Funcionários

De forma a puder tirar mais informação dos dados do funcionário e poder auxiliar o gerente de loja, foi apontada a necessidade de criar gráficos estatísticos que informem a quantidade de funcionários por género e por cidade. A apresentação destas estatísticas deverá acontecer na listagem de funcionários, conforme representa o use case.

Para facilitar a implementação e organização dos elementos necessários, foram criados os seguintes *mockups* para as principais funcionalidades.

Figura 7 – Mockup listagem funcionários

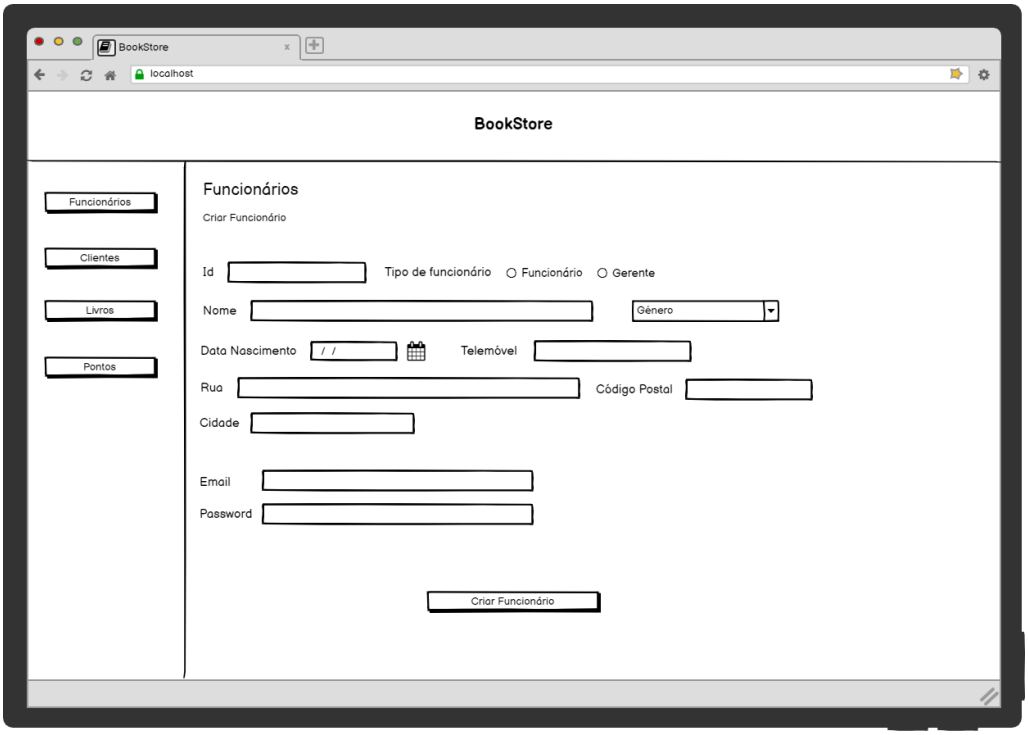


Figura 6 - Mockup criar funcionário

Clientes

Analisando o negócio e qual o papel do cliente perante a livraria foi chegado à seguinte estrutura de dados.

Nome Campo		**Validação**	**Obrigatório**
Informação Pessoal	Nome	Apenas letras. Tamanho: 1 até 60	Sim
	Gênero	Apenas “Masculino”, “Feminino” ou “Outro”.	Não
	Data de Nascimento	Deve ser inferior ou igual ao ano atual.	Sim
	Número de telemóvel	Número inteiro correspondente a número registado em Portugal.	Sim
	Email	*String* correspondente a email válido	Sim
	Rua	*String* com tamanho mínimo de 6 caracteres.	Sim
	Cidade	*String* com tamanho mínimo de 3 caracteres.	Sim
	Código postal	String no formato XXXX-XXX.	Sim
	NIF	Número inteiro com 9 dígitos.	Sim
Informação Programa de Fidelização	Data de Registo	String do tipo "AAAA-MM-DD"	Sim
	Nº Livros vendidos	Número inteiro que irá sendo atualizado	Sim
	Nº Livros comprados	Número inteiro que irá sendo atualizado	Sim
	Dinheiro gasto	Número decimal até 2 casas decimais que vai sendo atualizado	Sim
	Dinheiro em vendas à loja	Número decimal até 2 casas decimais que vai sendo atualizado	Sim
	Pontos atuais	Número inteiro	Sim

É possível observar que um documento cliente está divido em duas partes, em que uma será responsável por identificar o cliente com os seus dados pessoais e a outra parte irá armazenar a informação relativa ao plano de fidelização.

Os dados relativos à informação pessoal são inseridos na parte de criação do cliente, onde apenas poderá haver um NIF e um email único em toda a base de dados relativamente aos clientes.

A informação que rege o plano de fidelização é alterada sempre que haja uma compra ou uma venda por parte do cliente, onde é atualizado o valor gasto ou vendido à loja, bem como o número de livros e os pontos da sua conta corrente de pontos.

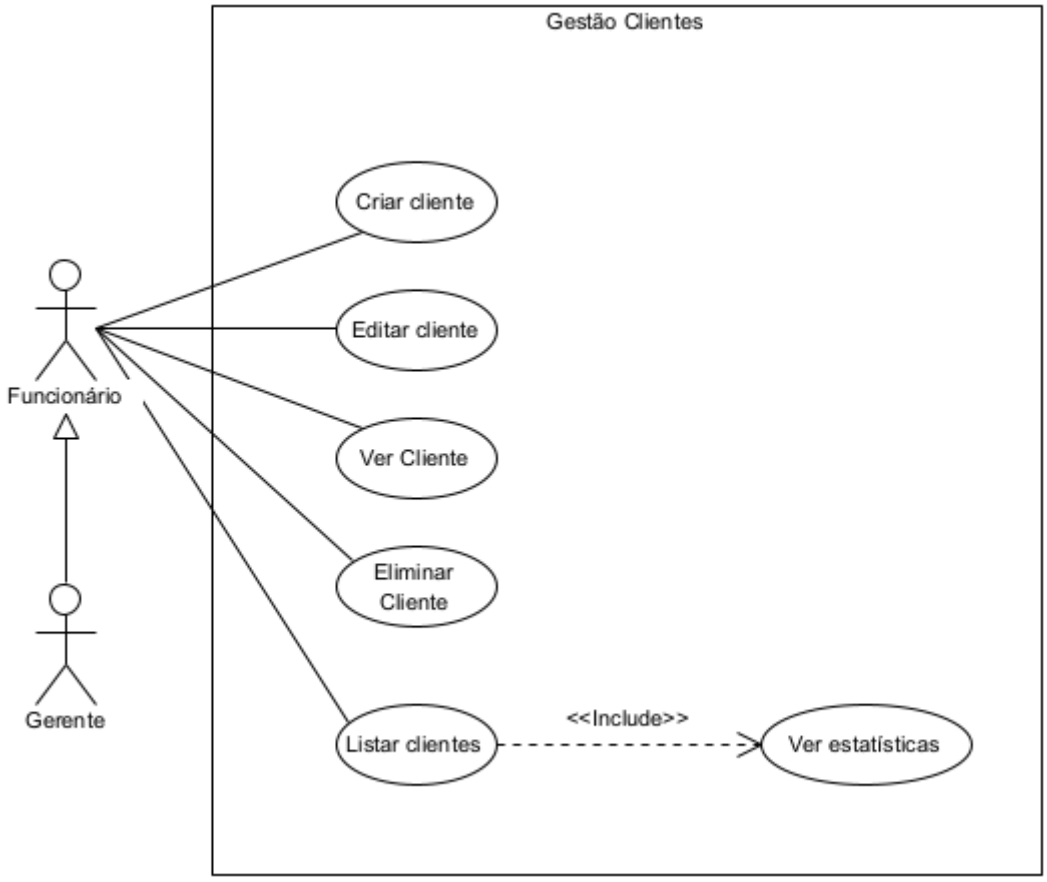


Figura 8 - Diagrama use case Clientes

Na figura acima é possível observar os casos de uso necessário para que haja uma gestão de clientes completa.

No que toca à gestão de clientes ambos perfis de utilizador poderão interagir com os casos de uso, tanto o gerente como o funcionário, pensando na possibilidade de um funcionário ter de registar um cliente, editar, entre outras operações.

Quando um funcionário ou gerente deseja ver a listagem dos clientes de forma automática é gerado as estatísticas.

De forma a auxiliar na implementação e organização gráfica foram criados os seguintes mockups que reúnem as principais funcionalidades.

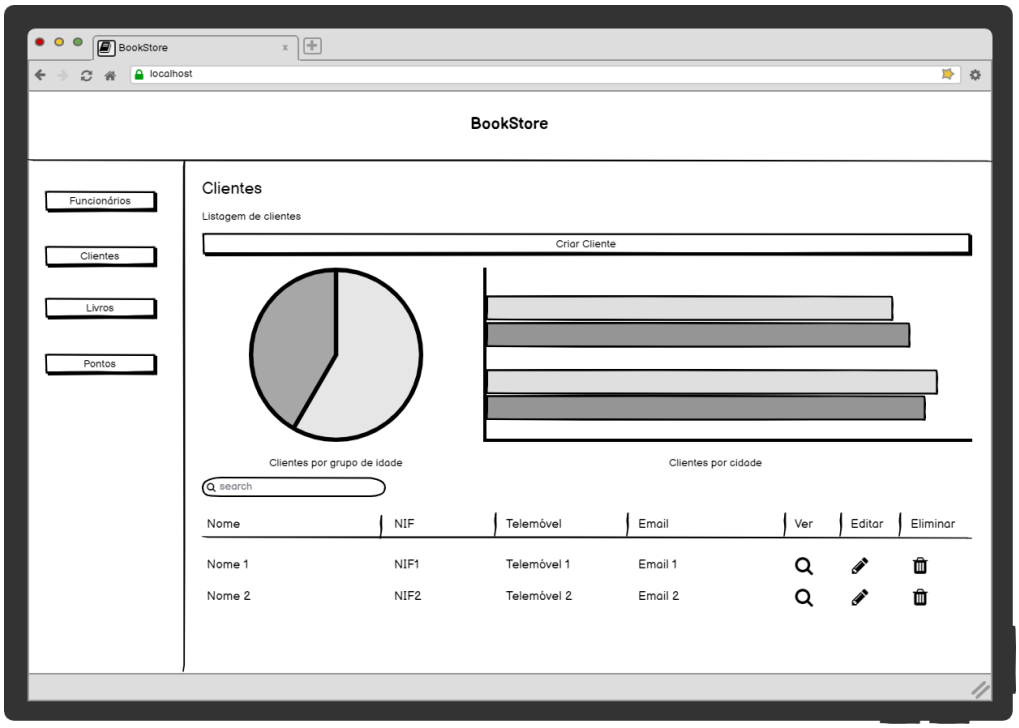


Figura 9 - Mockup listar clientes

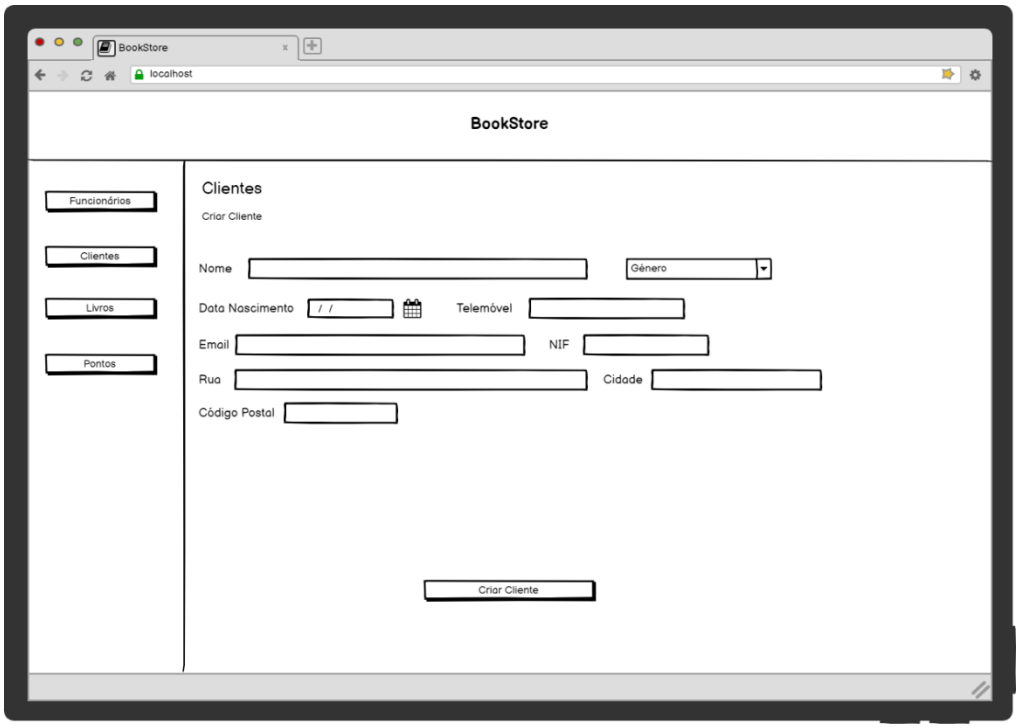


Figura 10 - Mockup criar cliente

Gestão de Pontos

Para gestão e criação de um programa de fidelização foram definidas várias métricas de atribuição e utilização de pontos, sendo elas:

- Pontos ganhos por cada euro gasto em loja;
- Desconto em euros a que corresponde a utilização de 100 pontos;
- Pontos necessários para que o cliente possa usufruir de uma encomenda com portes grátis;
- Pontos ganhos por cada livro vendido à loja.
- Pontos ganhos conforme a idade do cliente aquando do registo:
    - Infantil – idade inferior a 15 anos;
    - Juvenil – idade entre 15 e 23
    - Adulto – idade entre 24 e 50
    - Sénior – idade superior a 50

Nome campo	Validação	Obrigatório
Pontos ganhos por Euro	Inteiro com valor mínimo 0.	Sim
Desconto por 100 pontos		
Pontos porte grátis		
Pontos compra livro		
Pontos idade infantil		
Pontos idade juvenil		
Pontos idade adulto		
Pontos idade sénior		

Conforme as funcionalidades necessárias e as regras de negócio, foi possível reunir as funcionalidades a implementar num use case:

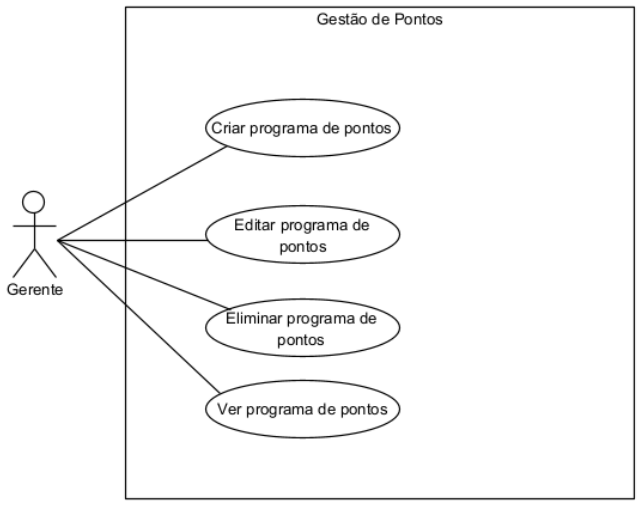


Figura 11 - Use Case Pontos

É importante relembrar que apenas pode existir um programa de pontos em vigor, ou seja, as quatro operações são relativas ao mesmo programa.

De forma a puder clarificar e auxiliar a implementação desta área foram criados *mockups* para as funcionalidades que deixariam mais duvidas a nível gráfico.

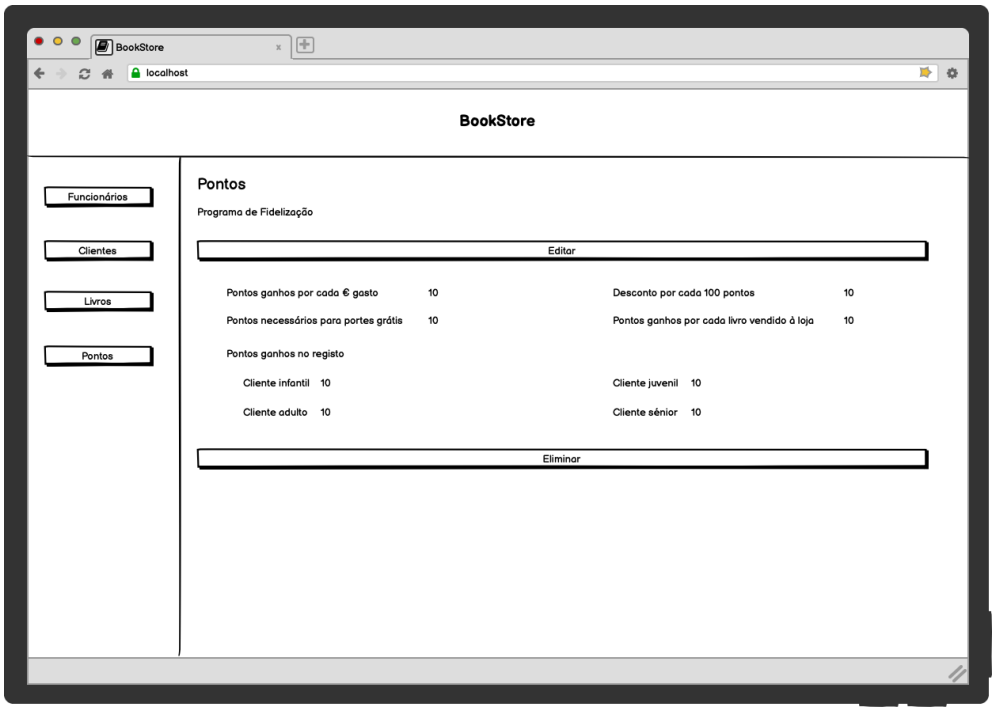


Figura 12 – Mockup ver programa de fidelização

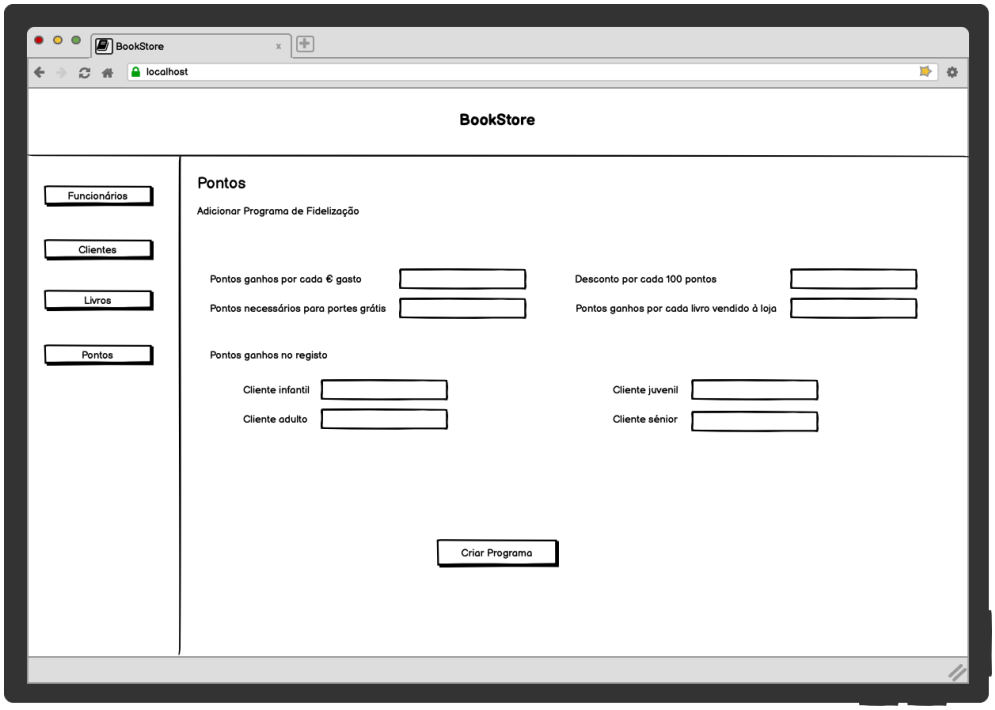


Figura 13 - Mockup criar programa de fidelização

Registo de uma venda

Após ter sido relatado a base do projeto, definição de clientes, funcionários, livros e o programa de fidelização, é possível então entrar nos componentes de registo de venda, sendo que neste local será cruzado toda a informação.

Temos que este registo diz respeito ao registo de uma compra pelo *BackOffice* da livraria, num cenário real, seria, por exemplo, uma compra numa caixa de pagamento da livraria.

Para que este registo seja possível é necessário que haja um funcionário, livros e um plano de fidelização em curso, sendo que o cliente poderá ser titulado de consumidor final ou associar a uma conta já registada no sistema.

<table>
<tr><th colspan="3">Nome Campo</th><th>Domínio do campo</th><th>Obrigatório</th></tr>
<tr><td></td><td colspan="2">id</td><td>Identificador único na base de dados</td><td>Sim</td></tr>
<tr><td rowspan="4">Cliente</td><td colspan="2">Nome</td><td>Nome do cliente associado</td><td>Sim</td></tr>
<tr><td colspan="2">Número de telemóvel</td><td>Número de telemóvel do cliente.</td><td>Sim</td></tr>
<tr><td colspan="2">Email</td><td>Email do cliente.</td><td>Sim</td></tr>
<tr><td colspan="2">Pontos antes da compra</td><td>Número inteiro que diz respeito ao número
de pontos antes de realizar a compra</td><td>Sim</td></tr>
<tr><td rowspan="13">Livros
[Array]</td><td colspan="2">id</td><td>Identificador único na base de dados</td><td>Sim</td></tr>
<tr><td colspan="2">Título</td><td>Título do livro</td><td>Sim</td></tr>
<tr><td colspan="2">ISBN</td><td>ISBN do livro</td><td>Sim</td></tr>
<tr><td rowspan="5">Quantidade</td><td>Novo</td><td rowspan="5">Número inteiro que indica quantas unidades o cliente leva da condição em causa</td><td>Sim</td></tr>
<tr><td>Excelente</td><td>Sim</td></tr>
<tr><td>Bom</td><td>Sim</td></tr>
<tr><td>Médio</td><td>Sim</td></tr>
<tr><td>Mau</td><td>Sim</td></tr>
<tr><td rowspan="5">Preço</td><td>Novo</td><td rowspan="5">Número decimal, máximo de 2 casas, que indica o preço dos livros na hora da compra</td><td>Sim</td></tr>
<tr><td>Excelente</td><td>Sim</td></tr>
<tr><td>Bom</td><td>Sim</td></tr>
<tr><td>Médio</td><td>Sim</td></tr>
<tr><td>Mau</td><td>Sim</td></tr>
<tr><td rowspan="2">Imagem</td><td>Url público</td><td>Caminho público no servidor</td><td>Sim</td></tr>
<tr><td>Tipo de ficheiro</td><td>String que representa o tipo de ficheiro.
Tipos aceites: png, jpg e jpeg</td><td>Sim</td></tr>
<tr><td colspan="2">Total</td><td>Valor decimal, máximo de 2 casas, que indica o valor total da linha</td><td>Sim</td></tr>
<tr><td colspan="3">Valor total da compra</td><td>Valor decimal, máximo de 2 casas, que indica o valor total da compra</td><td>Sim</td></tr>
</table>

Pontos a serem descontados	Número inteiro que indica os pontos a serem removidos da conta do cliente	Sim
Valor total com desconto	Valor decimal, máximo 2 casas, que indica o valor total com desconto	Sim
Desconto por cada 100 pontos	Valor a ser decontado por cada 100 pontos no momento da compra. Referência ao plano de fidelização	Sim
Data	Data da compra	Sim

Na tabela acima é possível ver como a informação é armazenada relativamente à venda, existe três partes distintas, onde uma parte irá guardar os dados referentes ao cliente, outra parte irá guardar a informação relativa aos livros, que poderá ser considerado uma linha de fatura e ainda os dados finais da compra.

Relativamente aos dados do cliente, é mantida a referência para o objeto cliente pelo parâmetro id, é armazenado a informação do cliente mais relevante para o registo de uma venda e ainda é acrescentado o campo sobre quantos pontos o cliente possuía antes da compra.

No que toca aos dados dos livros é armazenada para cada livro a referência para o objeto real pelo id do documento, bem como também é armazenado a informação mais importante para a faturação, como a quantidade a ser comprada e o preço por unidade dos diferentes tipos de condições que o livro pode ter. É também calculado o custo final da linha, ou seja, do livro em questão.

No que toca aos dados relativos à compra em geral existe o valor total com e sem desconto, quantos pontos o cliente desejou descontar e o valor de desconto que cada 100 pontos valia na altura da compra.

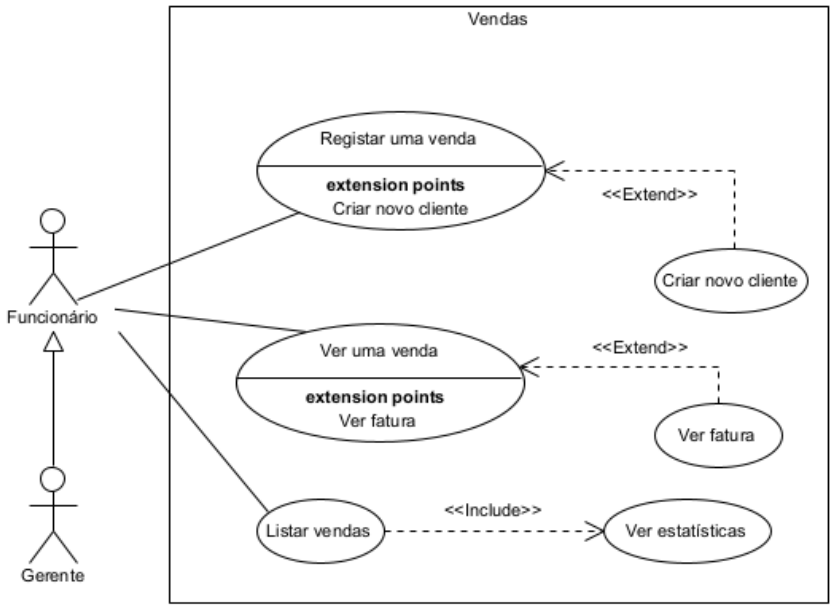


Figura 14-Use cases "Vendas"

Na figura acima é possível ver o diagrama de casos de uso, onde numa primeira análise é possível observar que não existe o caso de uso de eliminar ou de editar, dado que dentro do domínio após um registo de uma venda não deve ser possível manipular o registo.

É possível verificar ainda que o funcionário em interação com o caso de uso de “Registar uma venda” poderá criar um cliente na mesma operação.

No que toca ao caso de uso “Ver uma venda” existe uma operação opcional que é de ver uma fatura, onde caso seja útil o funcionário pode gerar uma fatura sobre a venda em questão.

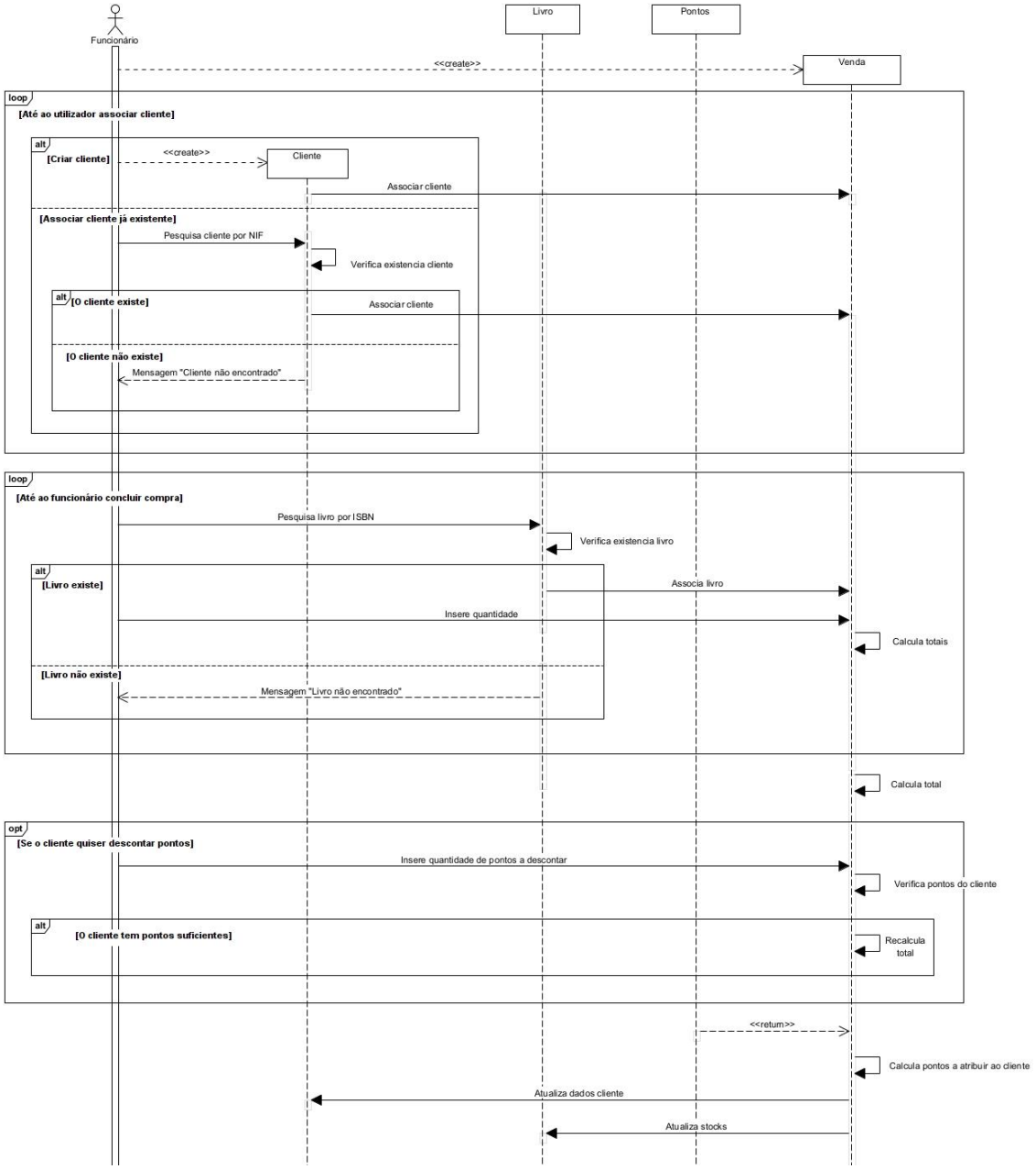


Figura 15-Diagrama de sequência "Registar Venda"

Como é possível observar na figura anterior, no diagrama de sequência, estão descritas as principais trocas de comunicação entre os módulos do sistema, sendo que é necessário a interação do sistema com os clientes, livros e sistema de pontos.

A interação inicia-se do lado do cliente, neste caso será o funcionário a interagir no início, sendo que, como anteriormente foi descrito pelo diagrama de casos de uso, poderá entre associar um cliente já criado ou registar no momento um cliente. Caso o funcionário deseje associar um cliente o sistema terá de analisar na base de dados se o cliente existe e em caso de sucesso irá ser associado à base de dados, mas caso o cliente não exista será informado o funcionário de forma a este puder tomar a decisão de pesquisar novamente, pois pode ter sido fruto de um erro ou de prosseguir à criação de um cliente.

Após o cliente estar associado à venda, o próximo passo é adicionar os livros à venda, onde serão as linhas da venda. Para esta tarefa o input poderia vir de várias formas, como por exemplo, leitor de código de barras, leitor de código QR, mas de momento o sistema apenas aceita inserção manual do ISBN. Com isto, o funcionário insere o ISBN, onde o sistema, semelhante ao processo de cliente, procura pelo livro e caso não encontre será lançado um erro, caso o livro seja encontrado o sistema verifica se este livro ainda não se encontra na lista de livros para a venda e conclui após esta análise.

Com os livros inseridos e as respetivas quantidades, o próximo e último passo do funcionário é descontar ou não pontos da carteira de pontos do cliente, sendo que os pontos devem ser superiores ou iguais a 100, sendo que é o mínimo no qual há desconto e não podem ser superiores aos pontos totais do cliente. No caso de os pontos excederem o total, ou seja, o cliente tem o direito de usufruir do livro de graça, o sistema irá calcular os pontos mínimos para que o livro seja gratuito de forma a serem otimizados os pontos do cliente. Após os pontos válidos inseridos, o sistema calcula o desconto que os pontos dão direito e apresenta o total.

No final de todo o processo do funcionário é necessário que o sistema faça as validações restantes, bem como a inserção da venda na base de dados. Numa primeira fase o sistema garante que todos os dados necessários foram submetidos e que existe stock dos livros em causa, em seguida irá atualizar os dados do cliente com a mudança a nível de pontos, se este usou a carteira de pontos, o número de livros comprados e o total dinheiro gasto em loja. Com a informação dos clientes atualizada é a vez dos livros sofrerem as alterações a nível de stock. Por último existe a inserção na base de dados.

Com uma certa quantidade de registos de vendas acaba por ser complicado analisar os dados e concluir algo sobre os mesmos, por isso, com o objetivo de melhorar a análise dos dados é apresentada um conjunto de estatísticas, em formato gráfico, bem como em formato textual.

As estatísticas focam-se na faturação e fornecem dados relevantes, tais como: o número de vendas no último mês, o total faturado no último mês e o número de livros nos últimos seis meses incluindo o mês atual.

Registo de uma compra

Semelhante às necessidades relativas à venda de livros usados e novos, temos a necessidade de registar as compras que a loja faz aos seus clientes de livros usados. A compra ocorria na loja física, sendo o estado do livro avaliado pelo funcionário. O registo dessa compra teria lugar perto do registo das vendas, sendo cada compra composta por:

Nome Campo		**Domínio do campo**	**Obrigatório**
Id		Identificador único da compra	Sim
Cliente	Id	Identificador único do cliente	Sim
	Nome	Nome do cliente associado	Sim
	NIF	Número de contribuinte do cliente	Sim
	Telemóvel	Número de telemóvel do cliente.	Sim
	Email	Email do cliente.	Sim
	Pontos antes da compra	A quantidade de pontos que o cliente tinha antes da compra	Sim
	Pontos depois da compra	Quantidade de pontos com que o cliente ficou após a compra	Sim
Livros [Array]	Id	Identificador único do livro	Sim
	Título	Título do livro	Sim
	ISBN	ISBN do livro	Sim
	Estado	Estado do livro. “Excelente”, “Bom”, “Médio” ou “Mau”	Sim
	Quantidade	Número de livros	Sim
	Preço	Preço de cada unidade de livro	Sim
	Total	Valor total da linha. (Quantidade * Preço)	Sim
Total da compra		Valor total de todos os livros comprados	Sim
Data		Data da compra	Sim

Como é possível verificar na tabela anterior para cada compra é guardada a informação de quantos pontos o cliente tinha antes da compra e depois da mesma, dados que são necessários caso queiramos acompanhar a evolução detalhada das compras do cliente.

Devido à complexidade associada à principal funcionalidade, registar compra, e as restrições envolvidas na faturação, foi possível chegar ao seguinte use case e diagrama de sequência.

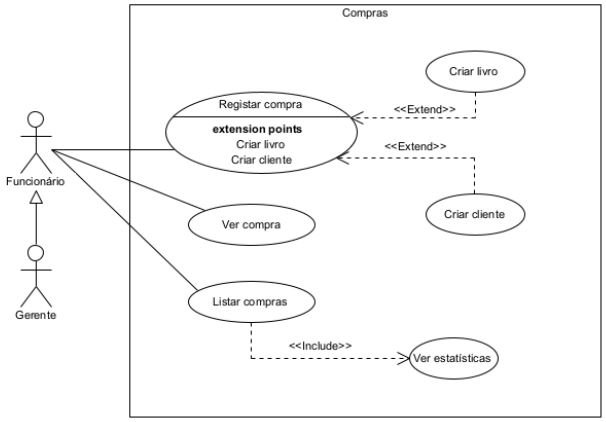


Figura 16 - Use case Compras

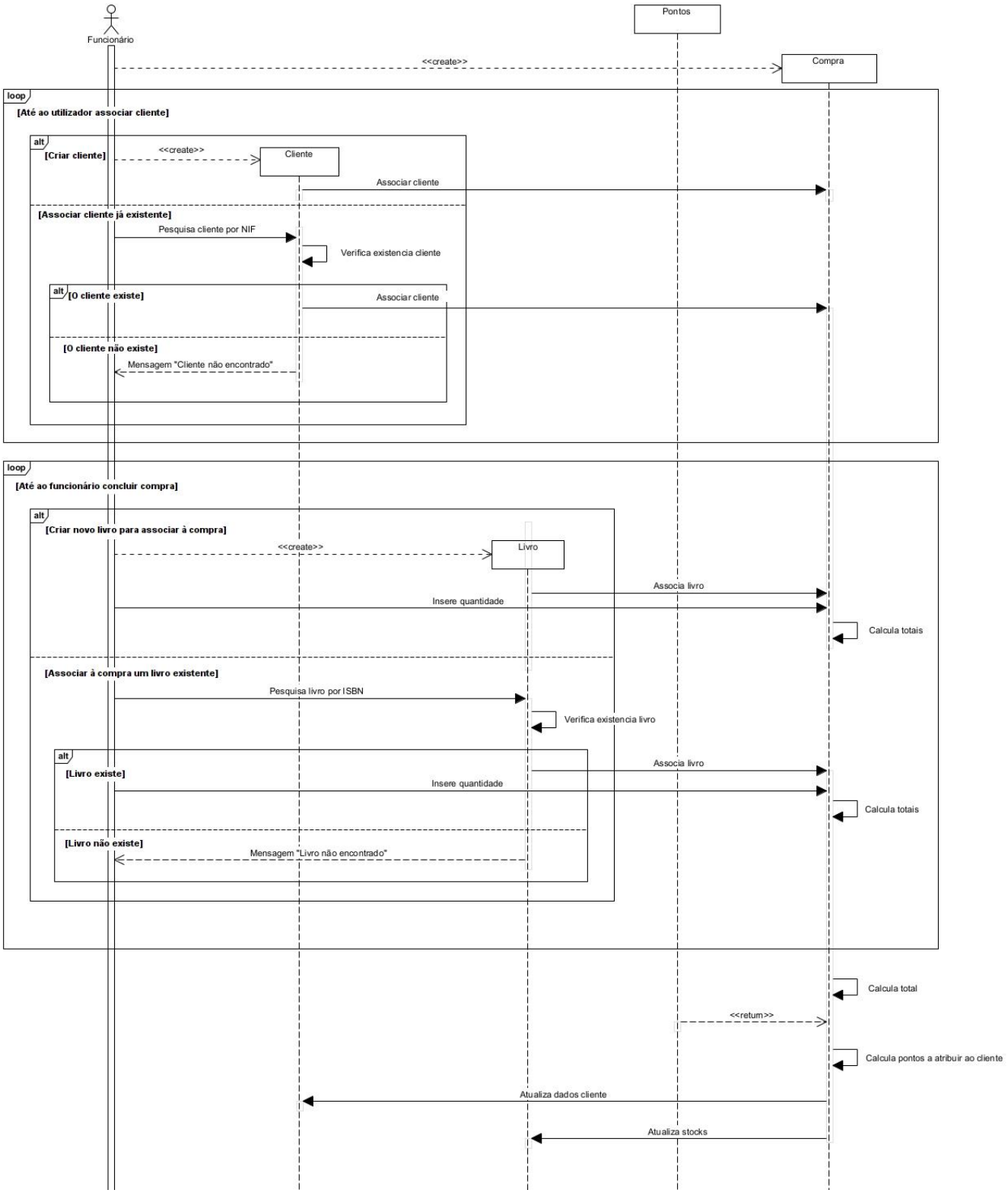


Figura 17 - Diagrama de sequência - Compra

Como o diagrama de sequência mostra, existe a junção dos vários subsistemas anteriormente referidos. Começamos inicialmente com associação do cliente à compra, essa associação pode ocorrer por um cliente que já tenha registo na loja ou sobre a forma de um novo registo.

Após a etapa da associação do cliente, associamos os livros que fazem parte da compra, no qual, tal como acontece nos clientes, é possível também criar um livro que até então não estava disponível em loja.

Numa fase final existe o calcular do custo total e atualização de toda a informação que sofreu alterações de forma a manter os dados atualizados e consistentes.

Análise da estrutura do projeto

Estrutura do projeto

De forma a organizar o projeto de forma mais escalável e de forma pradonizada foi escolhida a seguinte estrutura para o projeto:

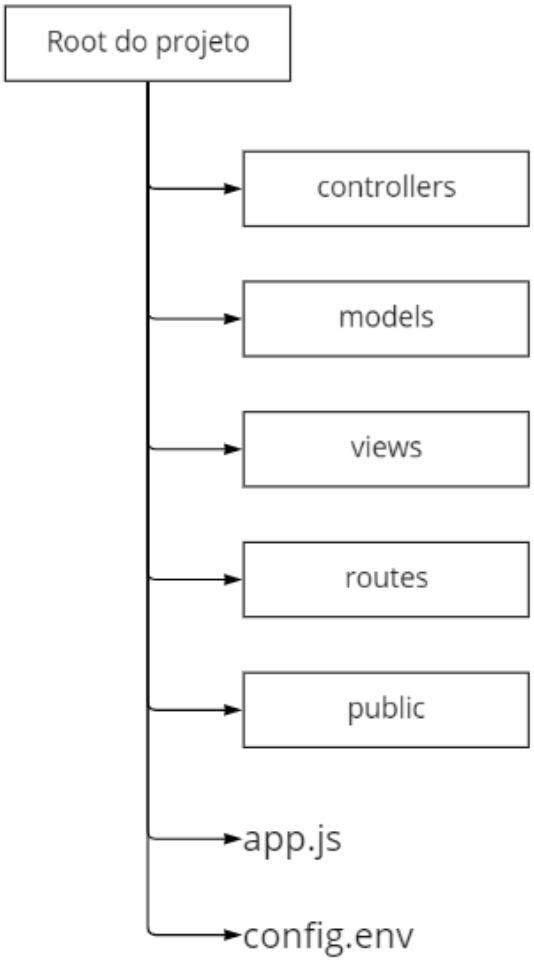


Figura 18-Estrutura do projeto

Como é possível observar na figura acima o projeto está dividido, entre outras, em cinco caminhos com responsabilidades distintas e com o motor do servidor, o ficheiro *app.js.*

Existe ainda outro ficheiro, *config.env*, que será responsável por armazenar a informação sensível da aplicação, tal como os dados de acesso à base de dados, bem como a chave de encriptação e desencriptação dos *tokens* de autenticação.

Será utlizado o esquema de rotas de forma a organizar a direção dos pedidos e a pasta *public* possuirá o conteúdo que pode ser consumido de forma estática, onde no caso deste domínio, por exemplo, são as imagens de livros, bem como documentos pdf gerados.

Analisando a estrutura é possível notar o uso do padrão MVC (*Model View Control*), pois com o uso desta organização é possível dividir os elementos da aplicação pela diferente lógica aplicacional.

Dentro da pasta *Model* ficará toda a informação relativa aos dados, como estes são estruturados, qual a validação dos diferentes tipos de dados dentro da estrutura.

Na pasta *View* estará os componentes visuais a serem projetados no lado do cliente, onde poderão ser páginas inteiras ou parte de uma página. O *template engine* usado é o *EJS*.

Por último, dentro dos *Controllers*, existe toda a parte lógica do *backend*, onde interage diretamente com os *modals* e com as *views* de forma a ser capaz de manipular os dados, comunicando com os *modals* e realizar o render da informação visualmente através da comunicação com os *views*.

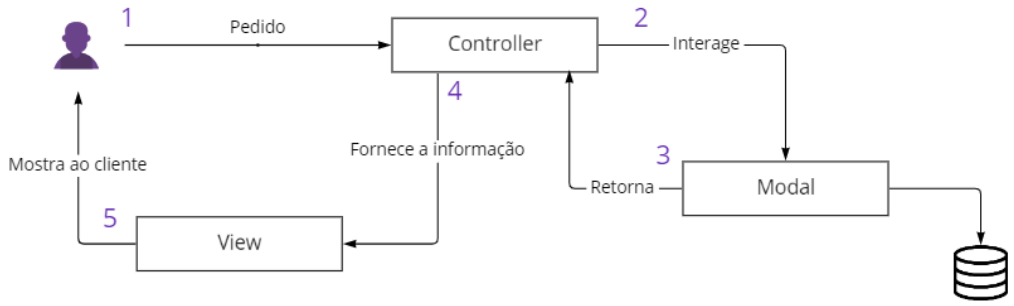


Figura 19-Ilustração WorkFlow MVC

Estrutura do template

O *template engine* a ser usado é o EJS, onde no intuito de não haver repetição de código, foi dividido uma página em seis componentes separados.

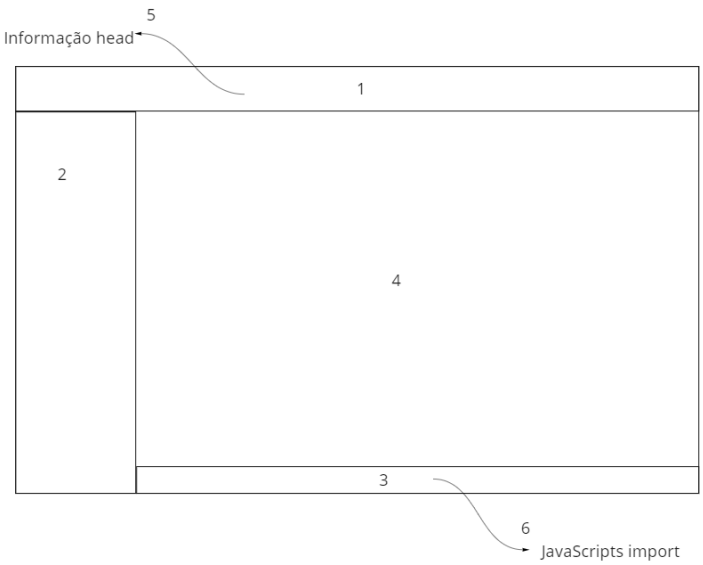


Figura 20-Divisão template

Como é possível observar na figura acima uma página é dividida em 5 parciais, são eles a informação do *head* (5), como os *imports* dos ficheiros *css* ou de *javascript* cruciais, o *header* (1) sendo a barra de horizontal do projeto, o menu (2) onde possui as ligações para as páginas do projeto, o *footer* (3) sendo o fim da página e por último os *imports* de ficheiros *javascriot* que podem ser carregados apenas no fim da página (6).

Após ser anunciado a funcionalidade dos 5 parciais fica a faltar o componente numerado por 4, que é o centro da página. Este componente será o conteúdo distinto de cada funcionalidade, tudo o resto mantém-se para que não haja duplicação de código e apenas este bloco na página é atualizado. Sempre que existe a necessidade de renderizar uma página é enviada numa variável chamada *loadContent.page* onde possuiu a direção para o conteúdo a ser apresentado no centro da página, onde o ficheiro *template* carrega os 5 parciais anteriormente anunciado e apenas o centro da página é dinâmico.

Middlewares

Para que seja possível realizar as operações é necessário uso de *middlewares*, para que estes possam auxiliar na resposta a um pedido. Os *middlewares* irão ter responsabilidades específicas em que juntando todo o trabalho realizado é então possível responder ao pedido.

No uso deste projeto foram utilizados *middlewares* já anteriormente criados por outros autores, bem como o uso de *midlewares* criados especificamente para o domínio em que o projeto estão inseridos.

Middlewares da aplicação

Body-parser

É um *middleware* bastante útil para tratar os pedidos *http*, pois a sua funcionalidade é realizar o parse do body do pedido para que seja possível acedermos e manipularmos os campos do body do pedido como um objeto JSON. Após a execução deste *middleware* toda a informação do body é disponível na variável *req.body*.

É algo que é utilizado com frequência na utilização da aplicação, dado que em todos os pedidos de criação e edição existe a necessidade de ser acedido ao body.

Cookie parser

Semelhantemente ao *Body-parser*, o *Cookie-parser* tem como responsabilidade realizar o parse das Cookies por parte do cliente para uma variável no objeto *request* chamado de *req.cookies*.

É algo utilizado para a verificação da *token* de autenticação, pois em uso da aplicação via browser o cliente prova a sua autenticação pela posse de um *token* válido nas cookies e com este *middleware* o processo de analisar o *token* é simplificado.

Path

Responsável por interagir com os caminhos dos ficheiros, é possível obter os caminhos absolutos ou relativos, obter o caminho raiz do projeto entre outras manipulações uteis sobre caminhos.

Multer

Middleware responsável por processar ficheiros multipart de um pedido http, como, por exemplo, o envio de ficheiros.

Para o caso apresentado o Multer é usado para dar suporte à inserção de imagens dos livros.

A configuração do *multer* pode ter algum nível de especificidade, sendo que para o caso foi necessário localizar a pasta onde as imagens seriam guardadas e filtrar qual o tipo de ficheiro e o tamanhão do mesmo.

Cors

É um package que fornece um *middleware* capaz de ativar o CORS segundo várias opções.

A necessidade do uso do *Cors* nasce do facto de, por defeito, todos os pedidos realizados ao servidor fora do domínio são rejeitados pelos browsers, não permitindo a comunicação *cross-origin*, por isso é necessário dar esta liberdade para que o lado do cliente possa comunicar com liberdade.

Mongoose

É um driver responsável por modular os objetos da base de dados *MongoDB* de forma a trabalhar num formato assíncrono.

Trata-se de um componente muito importante no desenvolvimento da aplicação, pois é este o responsável por todas as interações diretas com a base de dados não relacional *MongoDB*. Com este driver é possível realizar pesquisas simples, bem como pesquisas mais complexas envolvendo a agregação, no qual o seu resultado é devolvido em formato JSON, o que facilita em muito a interação com o resultado. Temos que também será com o uso do *mangoose* que toda a manipulação de dados será feita, desde a inserção até à rua remoção.

A validação dos tipos de dados pode ser feita com recurso ao *mangoose*, onde temos assim a segurança que os dados que são introduzidos na base de dados estão dentro do domínio correto.

Pdf Creator Node

O Pdf Creator Node não é amplamente conhecido, mas teve um papel especial neste projeto pois é o middleware que permite a criação de um pdf através de uma página html fornecida.

Foi utilizado de forma a puder dar um significado maior à principal área deste sistema, o registo de compras e vendas, sendo possível gerar um documento-fatura através da inserção dos dados num template html. Posteriormente é obtido um pdf com todos os dados com as características do template html e pode ser facilmente impresso como qualquer outro pdf.

File-System

Usado para operações com o disco. Usado para inserir ou remover documentos, bem como as imagens dos livros ou os documentos de faturação.

Cons

Permite que seja executado tarefas em tempos determinados, tem a função de criar tarefas agendadas.

Será útil para remover todos os ficheiros de documentação que vão sendo gerados ao longo de um certo período de tempo, com o objetivo de não encher o disco do servidor.

Autenticação e Autorização

JWT

O middleware JWT é a base para a autenticação e autorização deste projeto. Trata-se de um padrão bastante conhecido que permite criar dados protegidos por criptografia, gerando algo vulgarmente chamado de *token.*

Um *token* trata-se de uma sequência de números e letras que quando descodificado contém os dados que foram armazenados. A sua criação baseia-se no conceito de chave publica e chave privada, no qual ao gerar um *token* é assinado com a chave privada, sendo depois verificado a legitimidade e conteúdo do *token* através da chave publica correspondente.

Ao mesmo tempo é possível definir o período de tempo que o token poderá estar válido, permitindo aumentar a segurança da aplicação.

Considerado que todas as rotas da aplicação BackOffice estão sujeitas a autenticação, este *middleware* será vulgarmente utilizado de forma a garantir que só é possível o

acesso depois de atribuído o *token* na fase de login e que passado o tempo de autenticação o utilizador deverá renovar o seu *token* através do login.

Bcrypt

Este middleware é o responsável pela criptografia das palavras-passes através do algoritmo hash.

Quando um utilizador é registado no sistema, a palavra-passe introduzida na base de dados é o resultado do hash baseado na palavra-passe introduzida. Desta forma não é possível ter um acesso fácil a essa informação sensível e encontra-se mais segura no caso de um possível ataque. No login é feita a comparação da palavra-passe introduzida com o hash existente na base de dados de forma a verificar a permissão de acesso, mantendo a segurança dos dados.

Principais pontos chave da aplicação

Criar livro

O processo de criar livro inicia-se por parte do cliente com o preenchimento de um formulário com os dados necessários para identificar um livro.

Neste formulário existe uma ajuda ao cliente de forma a ser possível preencher a informação do livro de forma automática através da identificação do ISBN do livro, sendo que o cliente ao usar esta feature deve introduzir o ISBN do livro e aguardar pela resposta.

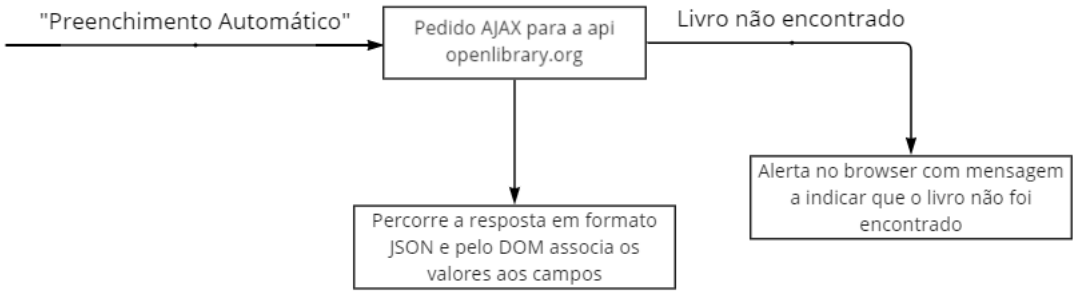


Figura 21-Workflow Pesquisa Automática

Como é possível observar na figura anterior, todo este processo é realizado na parte do cliente com recurso a pedido AJAX, onde dado a sua característica assíncrona o cliente poderá introduzir dados relativos aos stock e preço enquanto a resposta não é dada por parte da API, quando a resposta for recebida caso tenha havido sucesso com o uso de JavaScript pelas definições do DOM é atribuído os dados encontrados aos campos respetivos.

Independentemente se o cliente usou o preenchimento automático ou não, o importante é que o cliente deve enviar o pedido do formulário para o registo de um livro, onde após isso, o processo de criação passa por uma sequência de etapas no lado do servidor de forma a garantir a correta inserção ou a não duplicação de dados.

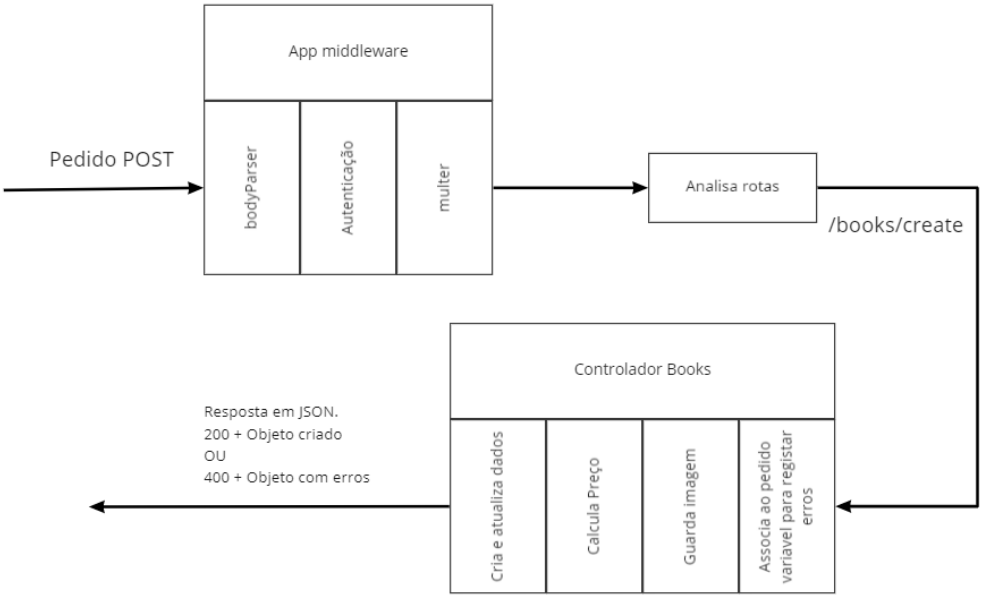


Figura 22-Middlewares create book

Como é possível observar na figura acima, o pedido nasce por parte do cliente com um pedido *HTTP POST*, onde é passado os dados necessário para criar um livro no *body* do pedido, sendo que estes dados são validados, dentro da limitação, no *frontend*, quer seja através do tipo de input imposto pelo formulário quer seja a nível de especificidade com o uso de *regex*, por exemplo. Após o cliente enviar o pedido, como na maioria dos pedidos, este passa pelos *middlewares* da aplicação, onde para este caso, os mais importantes são o *bodyParser*, a autenticação e o *multer*. Diferentemente aos outros pedidos, como há a necessidade de trabalhar com dados *multipart/form-data*, mais especificamente imagens, é necessário o uso do *middleware multer* para que seja possível obtermos a informação do ficheiro.

Após o pedido ter sido analisado e manipulado chega ao controlador, onde passa por 4 fases:

- Erros: a primeira fase é simples e a sua função é adicionar uma variável ao pedido que irá armazenar os erros ao longo do pedido.
- Guardar imagem: a segunda fase é garantir que foi recebido uma imagem válida e armazenar a mesma.
- Cálculo de preços: a terceira fase passa por calcular os preços, dado que dentro do domínio de negócio temos que os preços dos livros usados são calculados segundo uma percentagem de valorização, esta etapa serve para isso, fruto de um preço base, é feito uma pesquisa à base de dados para ser recolhida a

informação atual das percentagens e após isso é colocado numa variável *req.prices* todos os valores para os diferentes tipos de estado do livro.

- Fase final: nesta última fase acontece o registo na base de dados do livro, onde se todo o processo anterior ocorreu sem problemas o processo avança para o registo na base de dados. Caso a inserção tenha ocorrido sem problemas a resposta será de sucesso, com o código 200, e com o objeto criado, se porventura algum dado falhe na criação do objeto devido às validações realizadas no Schema é rejeitado o pedido, eliminado a imagem anteriormente inserida e é dada a resposta de erro, código 400, com a informação dos erros guardados.

Criar Funcionário

A funcionalidade de criar funcionário é desempenhada através da inserção dos dados num formulário semelhante ao mockup apresentado anteriormente. Estes dados são validados numa fase inicial através de atributos definidos nos inputs, por exemplo a validação de número mínimo de caracteres, padrões regex de forma a validar dados mais complexos (email, telemóvel, ...) e a validação referente à data de nascimento garantindo que o funcionário tem no mínimo 18 anos. Estas validações iniciais no *frontend* da aplicação vão garantir não só uma camada de segurança extra, mas também diminuir o número de pedidos entre o *backend* e o *frontend* visto que o pedido só é enviado quando os dados são corretos.

A passagem do pedido POST para criação do funcionário tem um comportamento semelhante ao pedido POST para criação de um livro, com exceção do *Multer*, este pedido utiliza o *middleware bodyParser* e o *jwt* para garantir a autenticação.

No backend numa fase inicial é criado o *hash* baseado na palavra-passe inserida e após essa criação o registo do funcionário é validado através do *middleware Mongoose* pelas restrições definidas no *schema* correspondente ao modal do funcionário. Esta validação é semelhante a validação que é feita no *frontend* com a exceção de que no schema é definido e restringido que cada email deverá ser único, garantido que não poderá haver dois utilizadores com o mesmo email.

Com estas duas validações para além de garantir a unicidade do email, garantimos que caso a aplicação seja acedida por um outro meio que não seja o browser a consistência e o cuidado dos dados estão garantidos.

Caso os dados inseridos estejam de acordo com as validações, o registo é inserido e o utilizador é redirecionado para a listagem de funcionários, se ocorrer algum erro o formulário é carregado novamente e enviado uma variável *error* a *true* que no *frontend* corresponde à apresentação de um modal que informa que ocorreu um erro ao tentar criar o funcionário.

Registar uma venda

Como foi anteriormente introduzido o processo de registar uma venda está disponível tanto ao funcionário como ao gerente e é realizada através de um preenchimento de formulário com grande interação a nível de *frontend* e posteriormente com um conjunto de validações e verificações a nível de servidor.

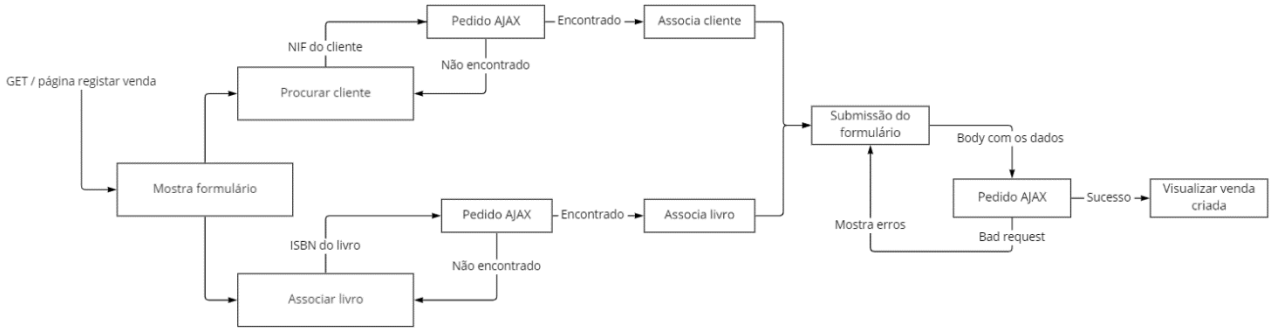


Figura 23-Workflow do lado do cliente em registar venda

Como é possível observar na figura acima existe uma grande comunicação entre o lado do cliente e o servidor através de pedidos AJAX de forma que a experiência do utilizador seja suavizada e tenha uma fluidez maior, sendo que o utilizador apenas é redirecionado para outra página no final do pedido caso este esteja correto.

Este processo é descrito anteriormente no documento sendo agora explorado com maior detalhe o fluxo de trabalho por parte do servidor e como o lado do cliente se comporta de forma a criar uma experiência suave ao utilizador sempre executando na mesma página.

Neste tipo de cenário foi optado pelo uso de pedidos AJAX de forma a haver comunicação com o servidor e perante a resposta dar um feedback ao utilizador sem haver mudança de página e dando liberdade para o utilizador poder realizar outra tarefa dentro do registo de venda sem ter de estar o processo bloqueado.

A forma como os erros são transmitidos para o utilizador é através de partes de código que estão com o atributo *hidden*, fazendo com que não sejam mostrados para o utilizador e quando houver a necessidade é retirado o atributo e mostrado o erro para que o utilizador consiga saber o que mudar.

Após o utilizador ter escolhido o cliente, associado os livros e ter atribuído quantidades válidas aos mesmos e no caso de o cliente desejar descontar pontos registar essa informação, o próximo passo é submeter a informação para o servidor processar no que resulta no seguinte fluxo de trabalho:

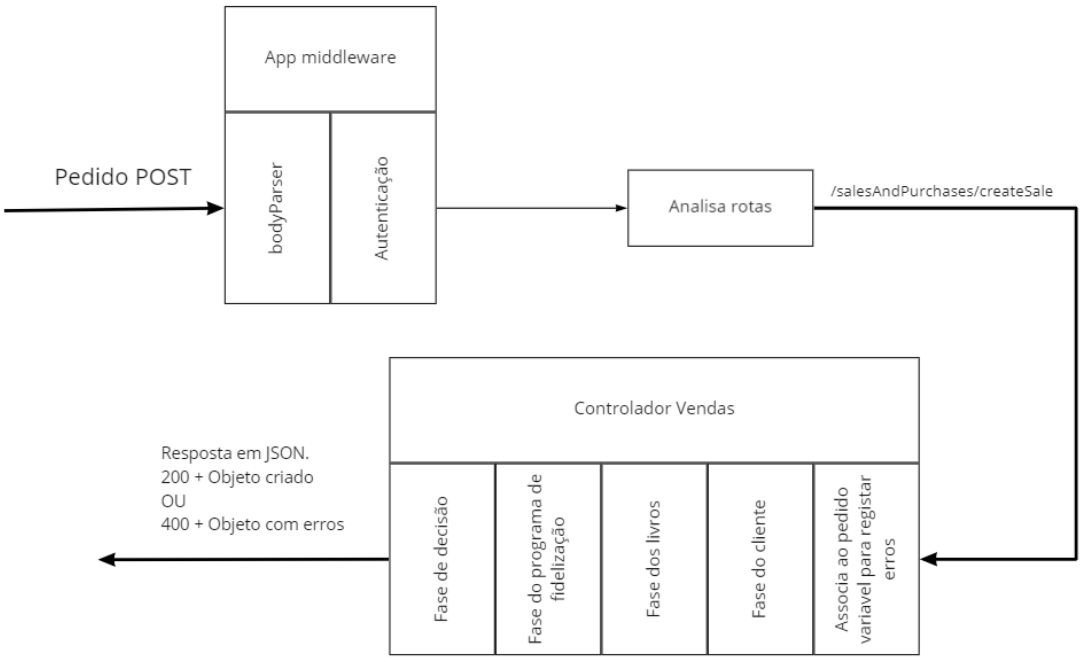


Figura 24-Workflow lado servidor Registar Venda

Após ter sido recebido o pedido POST o objeto request passa por um conjunto de middlewares da aplicação, sendo os mais uteis para este cenário o middleware que irá realizar o parse da informação no body para um objeto JSON.

Após ter sido garantido que o utilizador que deseja realizar este pedido está autenticado, é iniciado o processo dentro da rota com uso dos routes middlewares, onde ficam divididos em cinco fases:

- **Fase inicial:** Nesta fase semelhante a outros pedidos é criado uma variável no request que irá armazenar os erros ao longo do processo.
- **Fase do cliente:** Nesta parte será averiguado se o cliente recebido realmente existe na base de dados e em caso afirmativo associa este a uma variável no objecto request.
- **Fase dos livros:** Após ter sido validado o cliente é necessário validar os livros, onde existe a necessidade de para cada livro garantir que a quantidade que o cliente deseja comprar realmente é suportada pelo stock existente na base de dados. Além da verificação da quantidade é garantido também que os totais de cada livros batem certo com os cálculos realizados pelo servidor, para que não haja o erro de o funcionário anunciar ao cliente que o total será de, por exemplo, 40€ e momentos antes de o funcionário finalizar a venda, por algum motivo, o preço do livro foi atualizado para outro valor onde os 40€ anunciados pelo frontend passam a ser 49.99€, logo para evitar estes erros os totais são confirmados.
- **Fase do programa de fidelização:** Ambas duas fases anteriores eram obrigatórias, entrando nesta fase apenas será realizado o processo caso tenha

havido o uso de pontos para descontar. Em caso de desconto de pontos, o sistema irá realizar uma pesquisa à base de dados de forma a ler o programa de fidelização em curso e irá comparar com os valores recebidos, no mesmo sentido que nos livros para não haver o cenário do funcionário anunciar que, por exemplo, 100 pontos descontam 0.10€ e no lado do servidor a informação atualizada diz que são 0.05€ por 100 pontos. É também calculado o total com desconto.

- **Fase de decisão:** Na parte final o que resta é atualizar os dados, quer seja a nível de cliente, quer seja a nível de livros e registar a venda na base de dados. A resposta será de sucesso com o objeto criado ou de *bad request* com o objeto de erros.

Registar compra

Assim como o registar venda, o registar compra tem uma componente *frontend* muito trabalhada de forma que esse registo funcione e faça sentido ao utilizador. Tendo em conta que as duas transações são semelhantes, as validações ocorrem da forma sobre o mesmo objetivo.

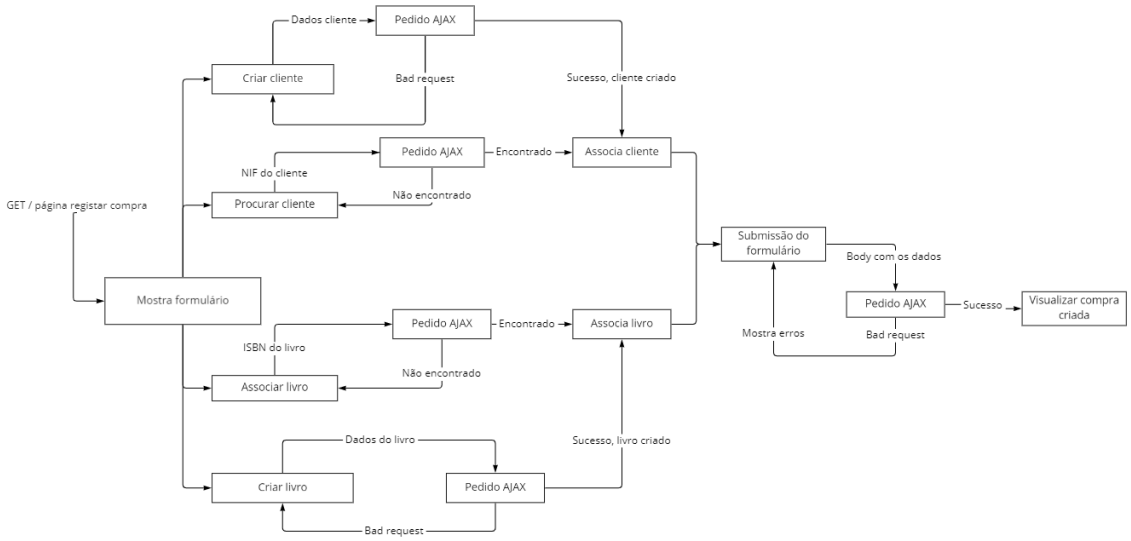


Figura 25 - Workflow do lado do cliente em registar compra

Assim como o registo de venda, a compra também é muito baseada em pedidos AJAX que permitem uma maior fluidez e dar mais sentido ao processo de registo de uma venda considerando as suas várias fases.

Numa fase inicial faz parte deste registo a associação do cliente à compra. Essa associação pode ocorrer através de um cliente já existente ou a partir da criação de um novo cliente a partir de um *modal*. Ambas as formas funcionam de maneira parecida, no caso de criar o cliente os dados são enviados para uma rota especialmente criada para o efeito, pois existe a necessidade de o retorno ser em JSON de forma que este informe ou o erro que ocorreu, ou retorne o cliente para que o mesmo seja associado à compra.

Algo semelhante acontece na associação só que apenas é feita a sua pesquisa na base de dados.

Como podemos estar perante um livro que até então não estava disponível em loja, existe a necessidade de criar um livro no processo de compra. Esse processo é relativamente parecido com os dos clientes.

À medida que os livros e as quantidades consoante as várias condições são inseridos, o total e sub-totais vão sendo atualizados, dando por finalizado o processo de compra quando o utilizador carrega em criar a compra, passando o trabalho para o backend:

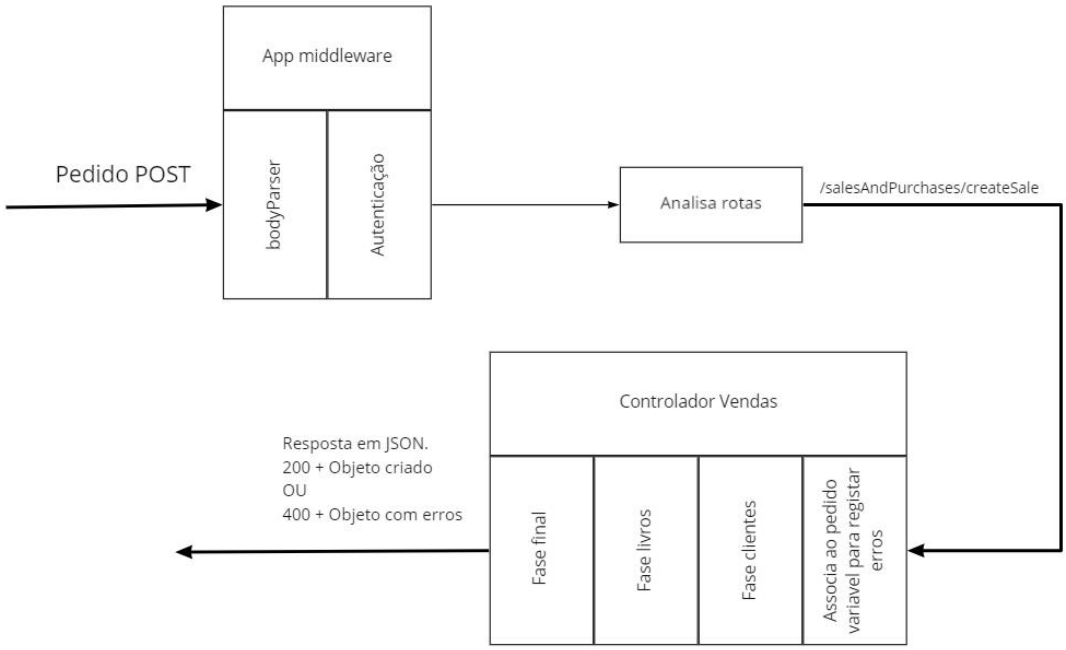


Figura 26 - Figura -Workflow lado servidor Registar Compra

Assim como no registo de venda, a compra passa por uma fase inicial de criação de variável que conterá os erros que ocorram e de seguida é verificado o cliente.

Na fase de livros existe a verificação da existência do livro e a validação dos preços enviados a partir do backend, visto que eles podem ter sofrido alteração até então.

Numa fase final é verificado a existência de algum erro na variável passada pelo *request* desde o início do processo, se não existir qualquer erro a compra é registada. Após ser registada existe a necessidade de atualizar os stocks dos livros afetados, calcular a quantidade de pontos que o cliente ganhou com aquela compra e atualizar esses dados no registo do cliente.

Consoante a resposta recebida pelo *frontend* no pedido AJAX a página ou é redirecionada para a visualização da compra que acabou de ser registada, ou caso tenha ocorrido erros é retirado o atributo *hidden* do modal correspondente.

Listagens

Para que haja uma listagem que ajude o utilizador a realizar uma melhor análise dos dados ou a otimizar tempo a ser encontrado um certo tipo de dados foi elaborado uma estratégia para as listagens, sendo que o processo é semelhante ao longo das diferentes listagens.

No início havia a dúvida sobre se o ideal seria a listagem acontecer toda no lado do frontend, ou seja, através de um pedido ao servidor era retornado a lista completa dos dados e era manipulado a listagem na parte do cliente através de javascript, porém o problema nasce quando estamos na presença de muitos dados, que por sua vez irá se traduzir num tráfego muito grande de rede o que poderá fazer com que a experiencia do utilizador seja comprometida e o servidor demore muito tempo a dar a resposta.

Dado isto foi pensada noutra ideia que envolve especificarmos a quantidade de dados que é suposto ser retornado pela base de dados, esta informação será enviada através do *url* com os parâmetros necessários. Exemplo: *…?page=1&perPaage=20*, como é possível observar o cliente especifica quantos dados devem ser apresentados e qual a página. Após o servidor obter estes dados, número de página bem como a quantidade de itens por página, é possível com recurso a uma funcionalidade do *MongoDB* fazer *skip* a um número de dados e é isso que acontece sabendo o número da página e a quantidade de dados por página.

No que toca ao tipo de dados a ser pesquisado é acrescentado ao *url* esta informação, obtendo assim um resultado do gênero: *…?searchType=NIF&searchValue=999999990&perPage=200&page=1,* é necessário especificar qual o tipo de dados a ser procurado e o valor que irá assumir na pesquisa, neste caso a pesquisa irá filtrar todas as entradas que possuam o NIF 999999990.

Estatísticas

Como foi anunciado ao longo do documento, foi procurado apresentar um conjunto de estatísticas para que fosse transmitido de forma prática a situação do negócio.

Ao longo do projeto existem estatísticas fornecidas através de gráficos outras através de texto, como ilustrado nos *mockups* acima referidos.

Para a apresentação dos gráficos foi utilizada a biblioteca chartsjs (https://www.chartjs.org/), que possuiu um conjunto vasto de funcionalidades que através de um objeto é possível gerar gráficos intuitos.

Estes gráficos estão representados nas listagens, como foi representado nos diagramas de casos de uso, as estatísticas estão como incluídas pois são tarefas que acontecem ao mesmo tempo, a nível técnico funcionam através de pedidos AJAX feitos ao servidor, de forma a não bloquear a página enquanto é esperado os resultados, caso o resultado demore o utilizador pode utilizar a página sem entraves, daí ser preferido a lógica assíncrona dos pedidos AJAX neste tipo de casos.

Do lado do servidor temos *endpoints* que irão funcionar como pedidos REST, onde é feito o pedido e devolvido os dados em formato JSON, onde no lado do cliente é feita uma ligeira manipulação para que se consiga distingue quais são os campos que irão conter os valores e quais os que irão conter o título da informação, por exemplo, em gráfico de barras é necessário dividir a informação entre os eixos do x e do y.

Geração de documentos de faturação

O processo de faturação passa por criar um documento *pdf*, fruto de um *template* em html com o envio de variáveis chaves para a criação do documento.

Por questões de otimização de espaço os documentos apenas são criados caso o cliente deseja ver o documento, dado que poderá ser um processo não comum, foi projetado esta estratégia pois o processo de criação de um documento *pdf* é custoso e assim apenas é realizado caso seja realmente necessário.

O plano para contrariar o espaço em disco foi com a criação de uma tarefa agendada para ser executada todos os domingos pelas 5h00, com o objetivo de eliminar todos os ficheiros que possam ter sido criados pelos funcionários durante a semana.

Segunda *Milestone*

Identificação do projeto

A segunda *milestone* foi tomada como um projeto em si, no sentido de adicionar valor e âmbito ao projeto no seu todo. Com a construção da segunda *milestone* foi possível criar uma aplicação web para que a loja esteja disponível para os clientes de forma online.

Para que este projeto seja um incremento ao projeto, este vai utilizar todas as funcionalidades anteriormente implementadas e adicionar novas para que a aplicação cumpra com os requisitos, por exemplo, temos que os clientes criados em loja poderão agora completar o seu registo e aceder à loja online, todo o plano de fidelização é estendido e melhorado para a loja online, é dado um poder maior a nível de marketing à loja com o uso de newsletters, será possível avaliar os livros, entre outras funcionalidades.

O desenvolvimento desta *milestone* teve início com pesquisa e aprendizagem de websites que partilham o mesmo objetivo de forma a obtermos uma melhor perceção das funcionalidades que seriam implementadas.

No que toca à parte técnica será utilizado a *framework Angular* para a parte do cliente e a comunicação realizada ao servidor será através pedidos *REST*.

BackOffice

Cupões

Para que haja um aumento de interesse por parte dos clientes na compra de livros desta livraria e para serem cativados novos clientes, foi criado um sistema de cupões onde o objetivo é oferecer descontos a todo o tipo de cliente, quer sejam clientes novos, não clientes ou clientes com um historial de compras considerável.

O identificador de um cupão será o seu código, onde é atribuído uma percentagem a descontar sobre o valor total da compra, é ainda dado um intervalo de tempo onde o cupão é válido, excedendo a data-limite passa a estar expirado e sem utilidade.

A gestão dos cupões é realizada na parte do *backoffice*, sendo que cabe aos funcionários ou gestor inserir e gerir a informação dos mesmos.

Newsletter

Com o objetivo de impulsionar e cativar os clientes na compra e venda de livros, foi implementado um serviço de Newsletter. Este serviço não só permitirá automaticamente informar os clientes que existe um novo livro disponível em loja, como também deverá ser possível informar possíveis campanhas ou avisos.

A Subscrição e desinscrição deste serviço estará disponível no *frontoffice*, sendo o envio da newsletter gerido pelo *backoffice* da aplicação.

Loja

De forma a haver uma uniformização de todos os dados referentes à loja foi criado um espaço no *backoffice* para que seja possível alterar os dados referentes a toda a informação da loja, nomeadamente a morada, telefone e email e que de forma prática e automática os novos dados sejam replicados de forma instantânea nos restantes processos que necessitam da informação da loja.

Toda a gestão dos dados fica a cabo do gestor da loja.

Compra do cliente

No que toca ao processo de compra do cliente não houve grandes alterações a nível de *backoffice*, foi necessário adicionar informação que antes não era considerada, como é o caso dos de cupões por parte do cliente, os portes de envio e dados respetivos, dado que anteriormente todas as compras eram feitas na loja não havia necessidade de considerar os mesmos.

Venda do cliente

Dado que que até à segunda *milestone* todas as vendas por parte do cliente eram realizadas na loja física, onde havia uma inspeção rápida por parte dos funcionários sobre o real estado do livro, com a expansão para um método online o processo de avaliação não é instantâneo e por isso as vendas passam a ter um processo assíncrono, no sentido que o cliente irá iniciar o processo online, o processo após iniciado fica em estado pendente até que a loja receba o livro. Na loja, ou seja, no *backoffice*, o gestor ou funcionário, quando receber o livro, poderá aceitar o concordado ou recusar caso o livro não esteja na condição anunciada.

Avaliação de livros

Esta nova funcionalidade tem como objetivo criar mais confiança na compra por parte do cliente, pois com o uso de avaliações é possível verificar as opiniões e experiências de outros clientes.

A nível de *backoffice*, é dado a possibilidade de, tanto o gestor como os funcionários, de visualizarem e gerirem os dados referentes à avaliação, onde poderão eliminar caso seja concluído que o comentário não é apropriado e visualizar o conteúdo.

Uma avaliação terá de possuir o autor, a avaliação numa escala de 0-5, a data da realização e um comentário textual.

FrontOffice

Gestão de cliente

Com o registo feito no website, o cliente deverá ter a sua disposição todos os seus dados, tanto pessoais, como relativos a vendas e compras que já efetuou. Deverá ainda também puder alterar a sua informação pessoal de forma a mantê-la atualizada.

Compras

No que toca à compra de um livro na loja online, temos que é um processo semelhante ao registo de uma compra em loja física, com algumas alterações no sentido de ser acrescentado informações e funcionalidades.

Foi acrescentado o uso de cupões, sendo que os cupões apenas podem ser utilizados nas compras online e apenas poderá ser utlizado um cupão por compra, mas em contrapartida um cliente pode utilizar o mesmo cupão em diversas compras diferentes desde que este esteja válido.

Também foi acrescentado o parâmetro de custo de transporte e morada, sendo que existem três tipos de transporte diferente: recolha na loja, recolha na morada atual do cliente ou recolha em outra morada. O preço de custo de transporte é ditado pela empresa, sendo um valor fixo, sendo que é dada a possibilidade de não ser cobrado custos de envio, caso o cliente escolha recolher o livro em loja. Com isto é possível obter os dados de envio para que seja possível à livraria enviar o livro para a casa do cliente.

Por último, foi adicionado a funcionalidade de pagar o livro online, para isso foi utilizado a *Stripe API*, dando o conforto ao cliente de pagar com cartão de crédito, cartão de débito e *apple pay*, de forma rápida e segura.

Vendas

O processo de venda de um livro usado por parte do cliente é muito semelhantemente à venda por parte do funcionário em loja física, sendo que o processo se estabelece de forma igual, o cliente pode vender um livro que a loja tenha registado e perante uma tabela de preços é feita a conjunção da qualidade do livro para com o preço novo resultado no preço a pagar ao cliente.

Considerando que em tempo real a condição do livro não pode ser atestada, o processo de venda não é instantâneo e sim sujeito à aprovação por parte da equipa em loja. Enquanto o processo se encontra pendente o cliente tem à sua disposição um documento para poder efetuar o envio dos produtos para a loja.

Desenvolvimento do projeto

Após consideradas todas as especificidades e funcionalidades a ser integradas nesta segunda *milestone* do projeto, foi possível obter o seguinte diagrama de desenvolvimento previsto. É apresentado no diagrama a seguir, as atividades a ser realizadas, as datas previstas de início e fim assim como a pessoa responsável pela implementação.

TAREFA	ATRIBUÍDO A	INÍCIO PREVISTO	TÉRMINO PREVISTO
Análise e estudo do negócio	Ambos	16/5/22	19/5/22
Página inicial	Josué Freitas	20/5/22	22/5/22
Carrinho	Nuno Ribeiro	22/5/22	23/5/22
Compras	Ambos	23/5/22	28/5/22
Gestão de cliente (FrontOffice)	Nuno Ribeiro	28/5/22	31/5/22
Vendas	Josué Freitas	27/5/22	30/5/22
Avaliações Livros	Nuno Ribeiro	1/6/22	4/6/22
Newsletter	Josué Freitas	31/5/22	3/6/22
Melhorias gerais	Ambos	4/6/22	12/6/22

Figura 27 - Diagrama de desenvolvimento previsto inicialmente - Segunda milestone

Especificação geral

Compras

Com a nova necessidade de integrar as compras realizadas no website, a estrutura inicialmente feita na primeira *milestone* sofreu alterações, sendo os novos campos referidos a seguir:

<table>
<tr><th colspan="3">Nome campo</th><th>Validação</th><th>Obrigatório</th></tr>
<tr><td colspan="3">Percentagem do cupão</td><td>Número inteiro igual ou superior a 0.</td><td>Não</td></tr>
<tr><td colspan="3">Nome do cupão</td><td>String com tamanho mínimo de 3 caracteres.</td><td>Não</td></tr>
<tr><td rowspan="6">Informações de envio</td><td colspan="2">Portes grátis</td><td>Valor booleano.</td><td>Sim</td></tr>
<tr><td colspan="2">Custo</td><td>Valor igual ou superior a 0.</td><td>Sim</td></tr>
<tr><td colspan="2">Tipo de envio</td><td>Apenas “storeAddress”, “clientAddress” ou “otherAddress”</td><td>Sim</td></tr>
<tr><td rowspan="3">Morada</td><td>Rua</td><td>String com tamanho mínimo de 1 caracter.</td><td>Sim</td></tr>
<tr><td>Cidade</td><td>String com tamanho mínimo de 1 caracter.</td><td rowspan="2">Sim</td></tr>
<tr><td>Código Postal</td><td>String no formato XXXX-XXX.</td></tr>
<tr><td colspan="3">Estado</td><td>Apenas “Aguarda Pagamento” ou “Pago”.</td><td>Não</td></tr>
</table>

Como é possível reparar, uma compra é agora composta por um tipo de envio. Este tipo de envio representa as várias opções que o cliente possui de envio da compra, “*storeAddress*” representa o envio da compra para a própria loja, ou seja, podemos concluir que o valor dos portes seriam grátis. A opção “*clientAddress*” e “*otherAddress*” representa respetivamente a opção de enviar a compra a morada do cliente ou para outra morada que não seja a do cliente, onde é assumido o valor de portes definidos pela loja.

O estado de pagamento de uma compra é definido inicialmente como “Aguarda Pagamento” visto que a compra é criada quando o cliente avança para o pagamento. Após o pagamento esse estado é alterado para “Pago” e passa a ser considerada uma compra válida. Os detalhes relativos a implementação e utilização deste atributo “Estado” será explicado de um ponto de vista mais técnico mais à frente neste relatório.

Como mostra o diagrama de use case seguinte o cliente criar uma compra não significa necessariamente que a mesma é valida, visto que a mesma só é validada depois do pagamento ter sido aprovado.

Também é possível verificar que na aplicação *frontoffice* o cliente terá acesso as suas compras assim como as respetivas faturas.

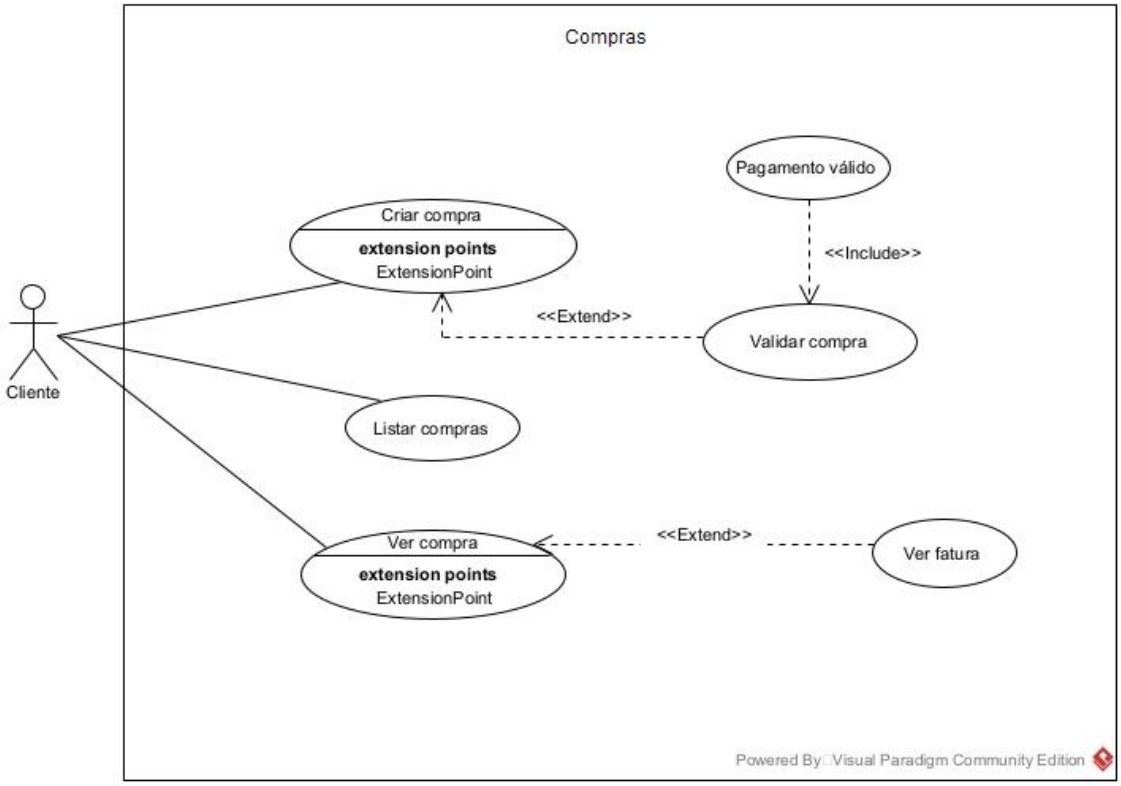


Figura 28 - Use case "Compras" - Frontoffice

Com objetivo de ter uma melhor noção das funcionalidades que compõem as compras, foram criados *mockups* para auxiliar nesse sentido.

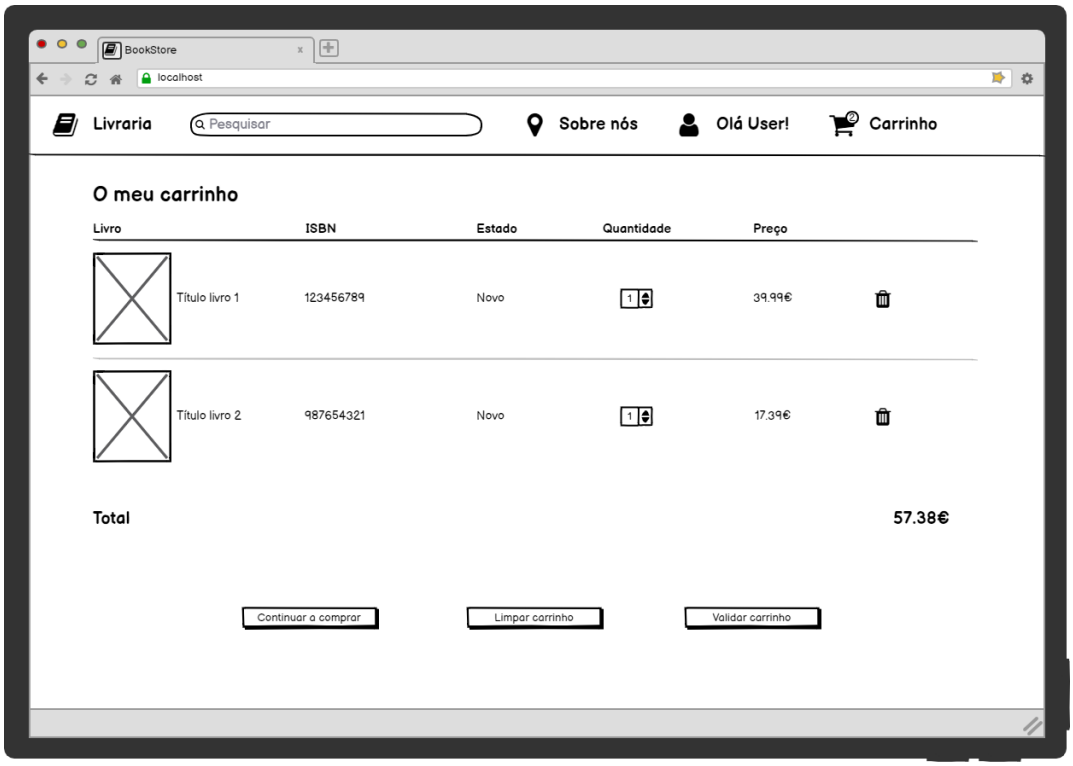


Figura 29 - Mockup carrinho

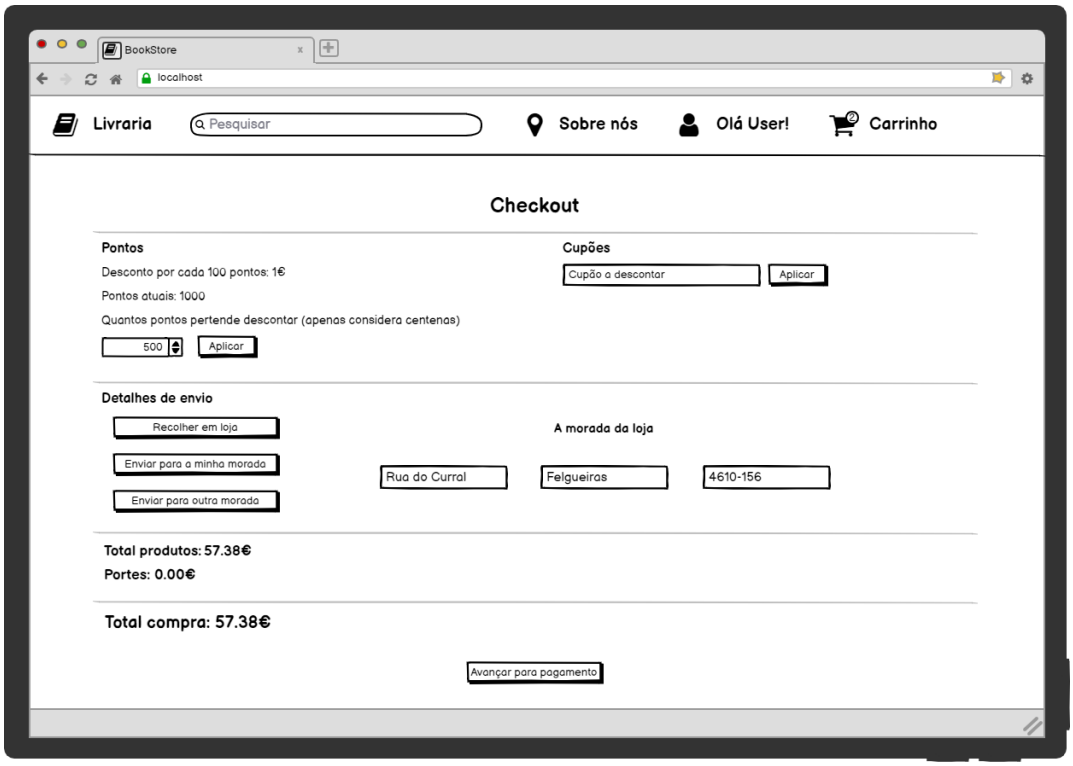


Figura 30 - Mockup checkout

É necessário esclarecer que a compra é criada quando o cliente carrega no botão "Avançar para pagamento" presente no checkout, sendo o cliente reencaminhado para a sessão STRIPE para puder realizar o pagamento.

Considerando que o processo de compra é um processo complexo, é mostrado o seguinte diagrama de atividades que descreve as atividades que compõem uma compra. No diagrama é possível notar as várias entidades que participam no processo de compra e o seu papel no processo.

Este processo tem início com a adição de livros ao carrinho do cliente, estes livros podem ser adicionados através da página inicial ou através de uma pesquisa no website. Com pelo menos um livro no carrinho, o cliente pode avançar para o checkout. No checkout o cliente terá a oportunidade de descontar os pontos que possui por desconto no preço total e utilizar cupões de desconto.

O cliente não deverá poder avançar para a fase de pagamento até que tenha selecionado para onde pretende o envio dos seus livros. Este envio pode ocorrer para a própria loja, a morada do cliente ou outra morada. Ao avançar para o pagamento a compra será registada com o estado de a "Aguardar Pagamento" até que o mesmo seja bem-sucedido na sessão do stripe.

Durante o checkout o cliente é informado dos descontos que estão a ocorrer e a adição do valor de portes de envio.

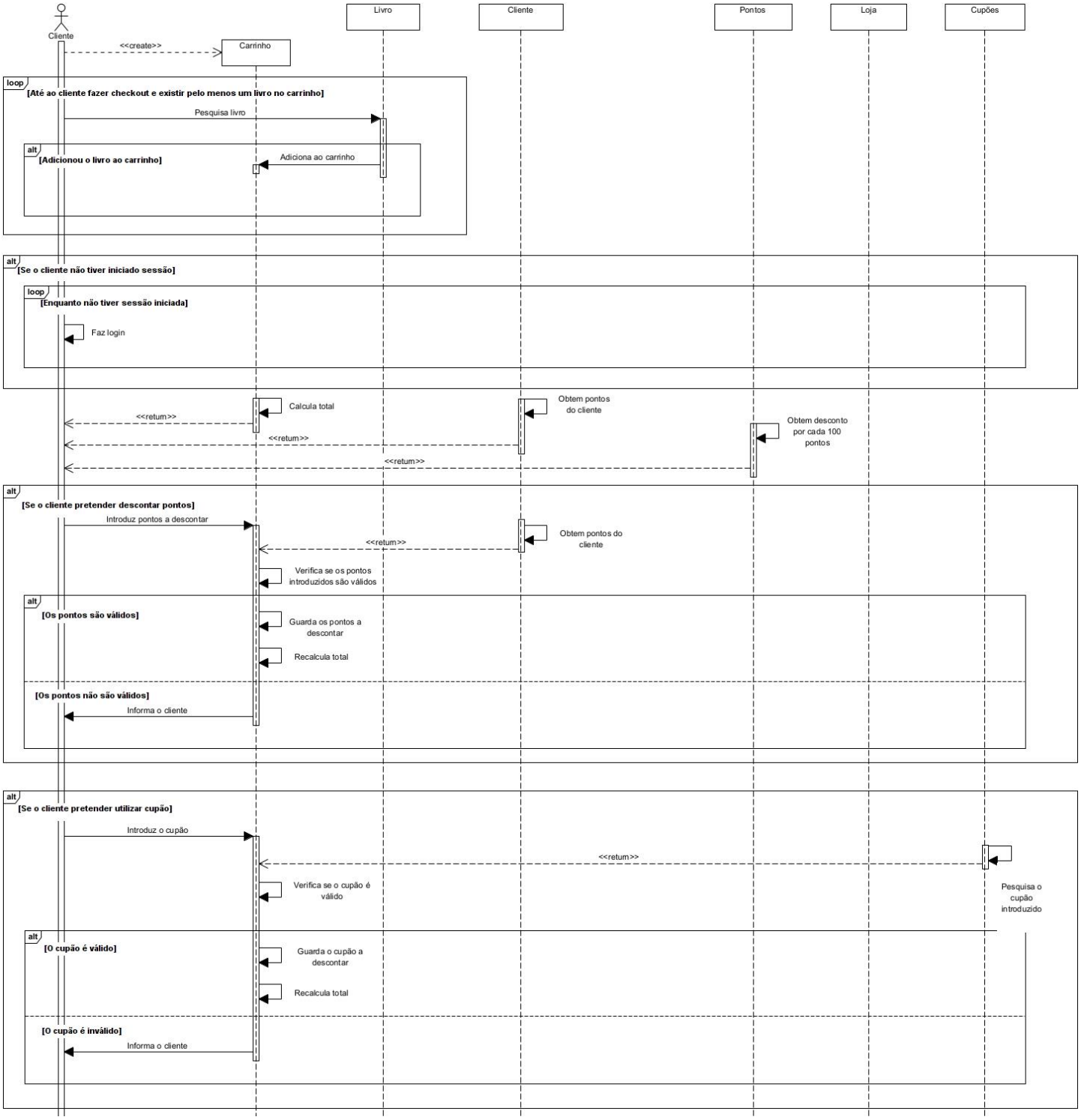


Figura 31 - Diagrama de sequência "Criar compra"

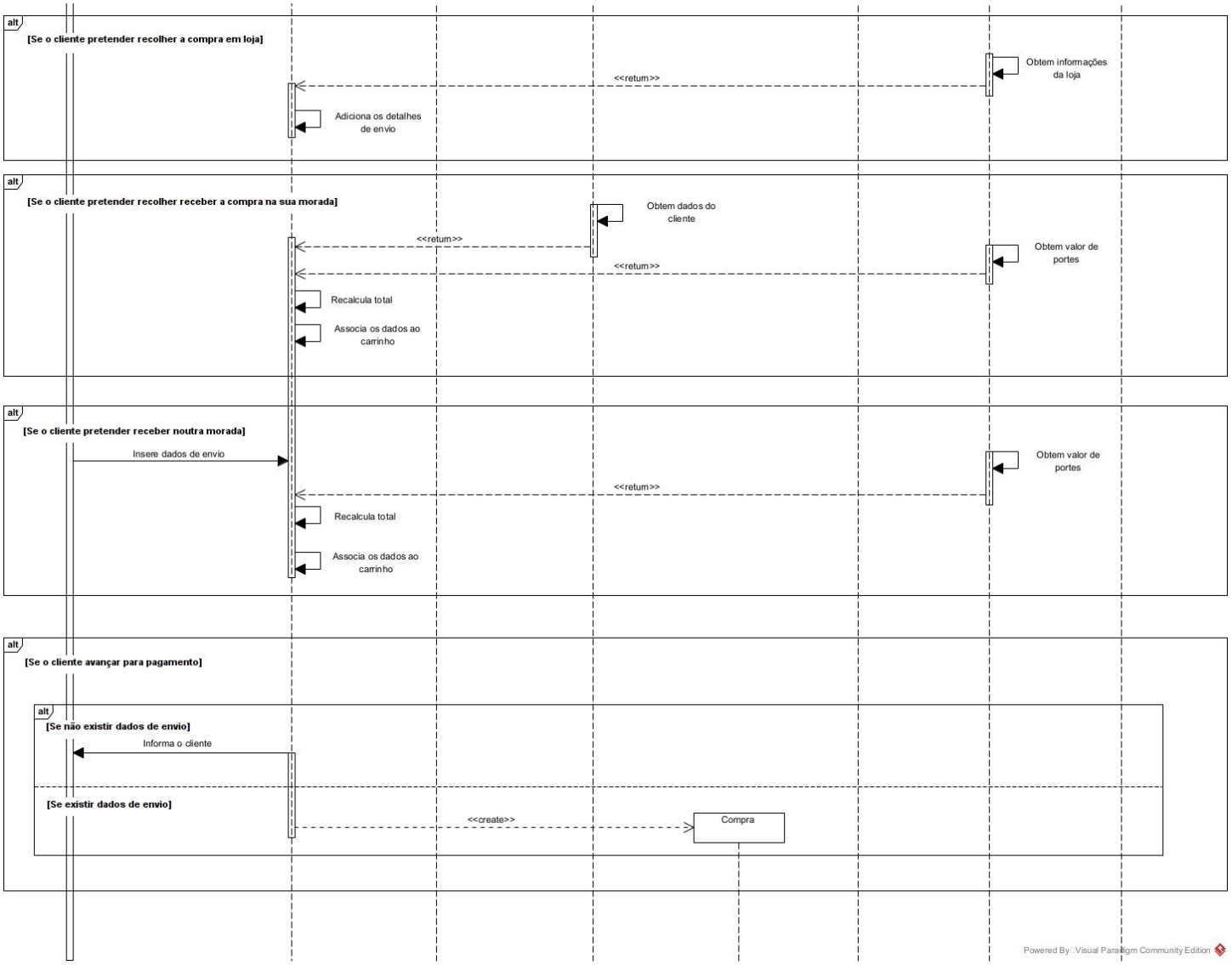


Figura 32 - Diagrama de sequência "Criar compra"

Devido à nova estrutura da compra, a aplicação *backoffice* também sofreu alterações de forma a mostrar aos funcionários e gerentes a proveniência das compras e os seus dados de envio e cupões utilizados numa compra.

Cupões

Como anunciado anteriormente foi acrescentado a funcionalidade do uso cupões, onde para dar suporte a esta funcionalidade foi definido o seguinte modelo de dados:

Nome campo	Validação	Obrigatório
Código	*String* com valor mínimo de 3 caracteres.	Sim
Data de início	Data de início onde o cupão passa a estar disponível.	Sim
Data de expiração	Data onde o cupão passa a estar inválido.	Sim
Percentagem a descontar	Valor inteiro de 1 até 100, que dita a percentagem a descontar sobre o valor total.	Sim

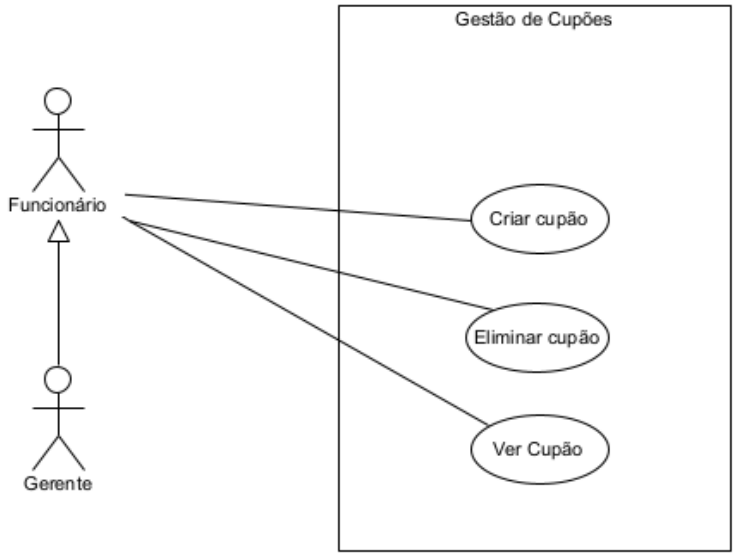


Figura 33- Use case "Gestão de cupões"

Como é ilustrado na figura acima tanto o funcionário como o gerente podem gerir os cupões, sendo que é dado a opção de criar um cupão, eliminar ou ver os detalhes de um cupão. O uso do cupão por parte do cliente, esta relacionado com o momento da compra.

Cada cupão é único, sendo que não há a possibilidade de haver dois cupões com o mesmo código mesmo que as datam sejam diferentes, como é esperado as datas devem criar um intervalo de tempo válido, para isso da data de fim deve ser superior à data de início.

Para a sua utilização é necessário que o cliente insira o código e será verificado o seu valor de desconto.

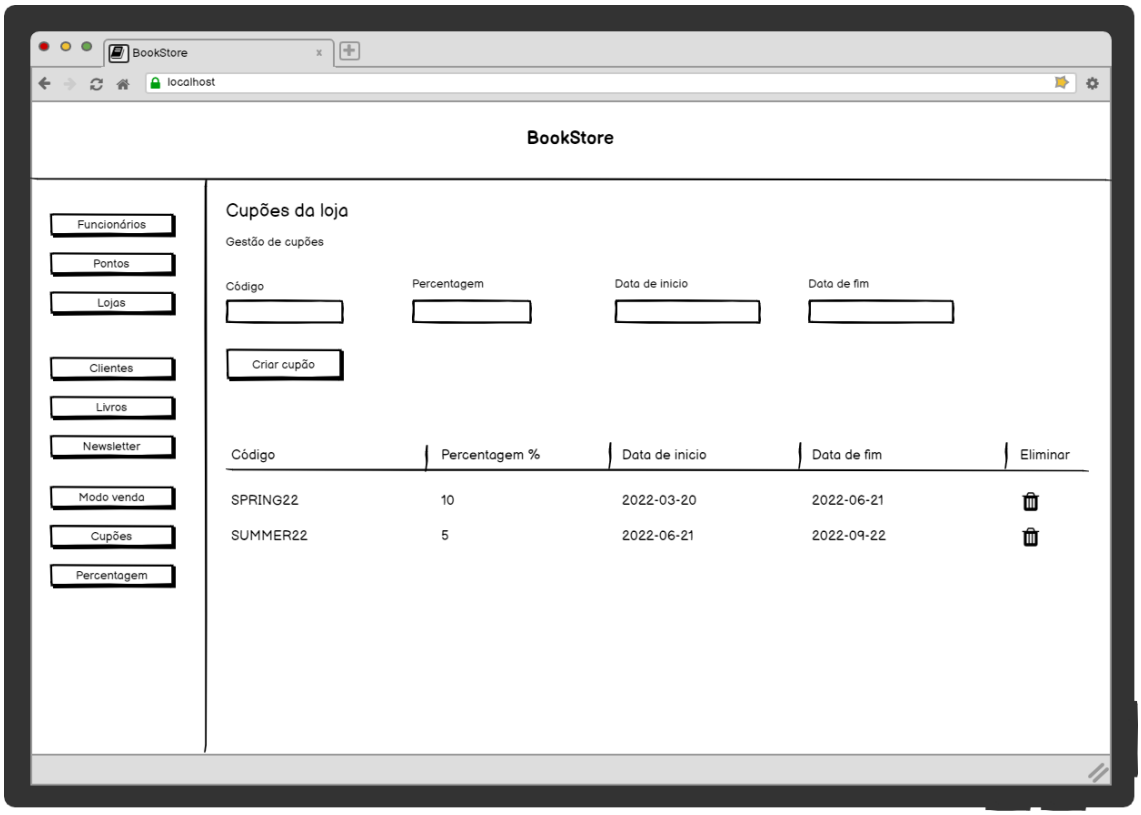


Figura 34-Mockup "Cupões"

Vendas

Como referido anteriormente as vendas sofreram uma pequena alteração estrutural de forma a serem integradas no sistema. Considerando a venda de livros à livraria está sujeita a aprovação, quando um pedido de uma venda é feito esta é criada com o estado de “Pendente”.

Depois perante a avaliação dos funcionários em loja esse estado pode alterar para “Aprovado” ou “Recusado”. Durante o processo o cliente poderá acompanhar o estado atual na sua página de gestão, estando disponível enquanto o pedido for “Pendente” um documento que contém a informação necessária para o envio para a loja através de uma transportadora, também conhecido como “*shipping sticker*”, caso a venda tenha sido aprovada ou em lojas física o cliente poderá ver a nota de crédito.

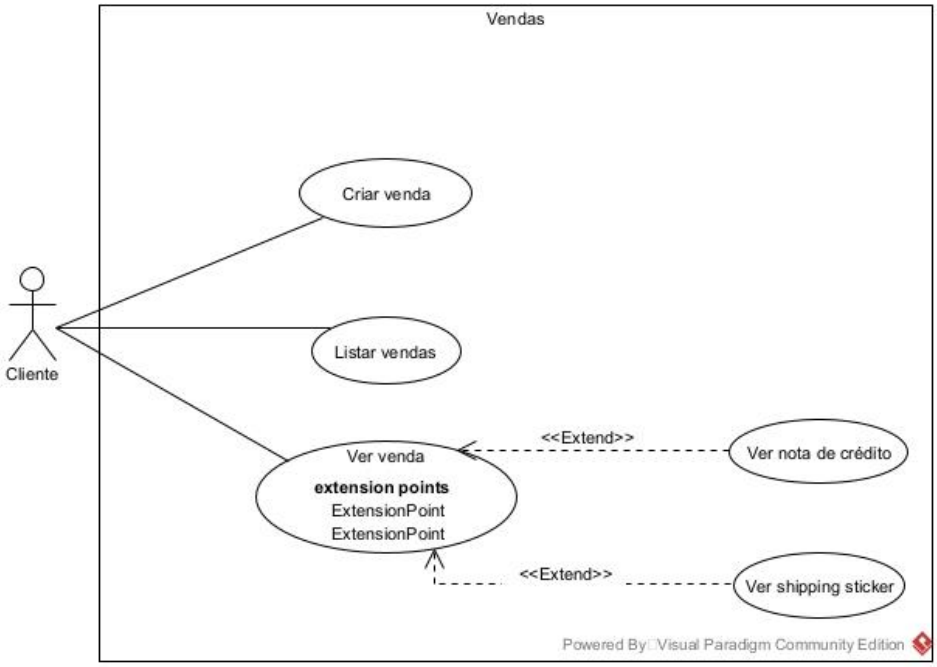


Figura 35 - Use case Vendas

De forma a auxiliar na implementação, foi criado o *mockup* para criar venda.

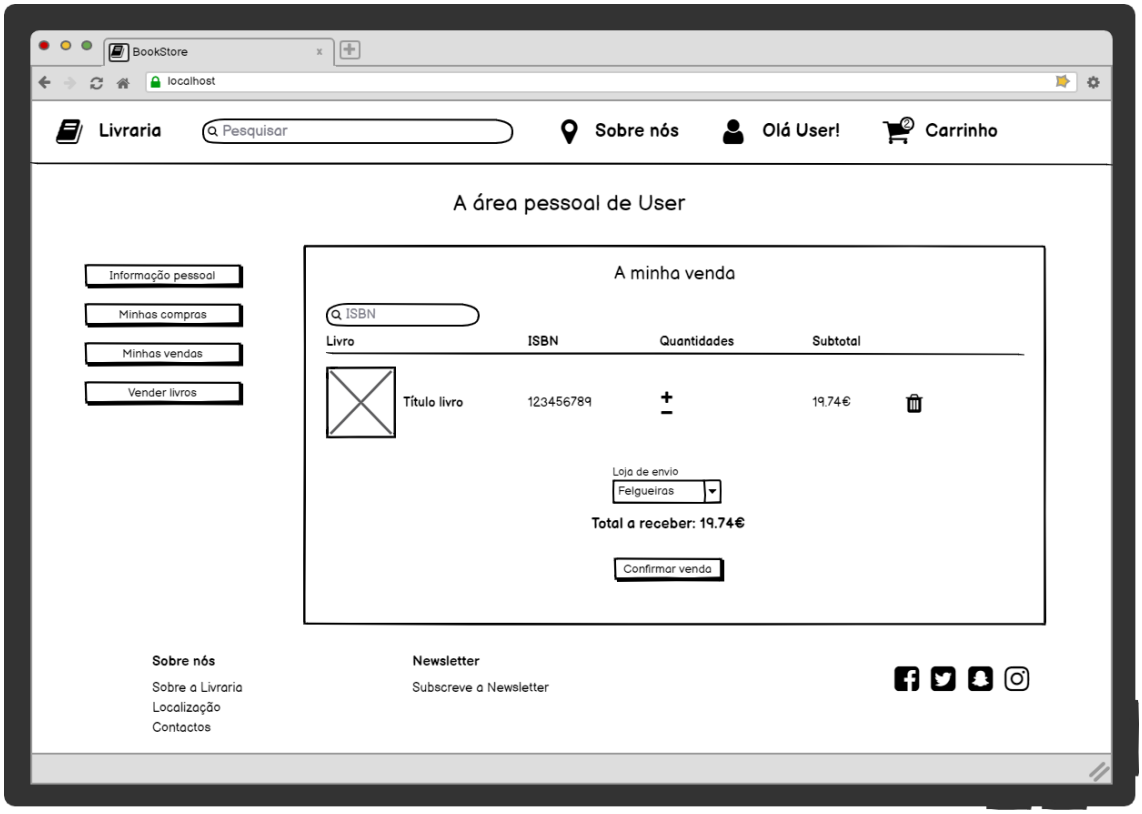


Figura 36 - Mockup criar venda

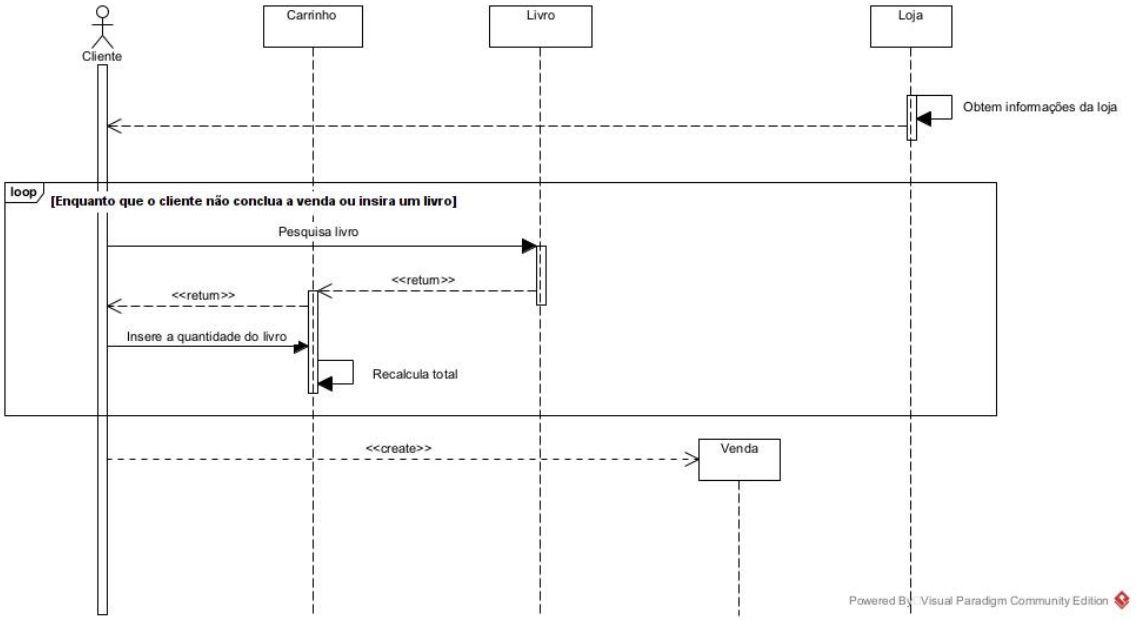


Figura 37 - Diagrama de sequência criar venda

Como mostra o diagrama de sequência anterior, a venda consiste apenas na adição de livros que o cliente pertende vender e a inserção da sua quantidade pelo estado que o cliente pensa estar o livro. As informações da loja são obtidas de forma a reforçar ao cliente para onde deverá enviar o livro.

No backoffice esta nova faceta da venda de livros também teve de sofrer alterações, tendo seguido o modelo do próximo mockup que representa a listagem de pedidos.

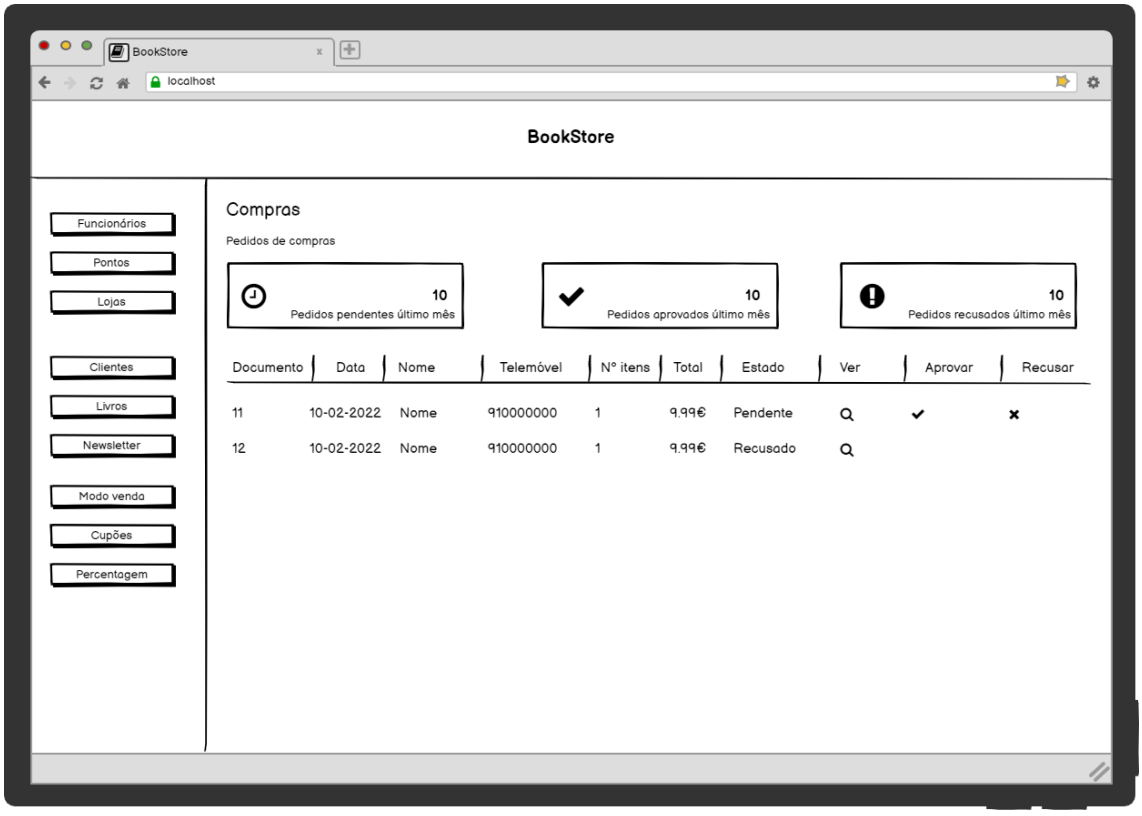


Figura 38 - Mockup listagem pedidos - Backoffice

Livros

No que toca à parte de livros houve um aumento no âmbito do mesmo, no sentido de ser suportado avaliações por parte de clientes registados aos livros. Para que seja possível este suporte foi alterado o modelo de dados com o seguinte acréscimo:

Nome campo			**Validação**	**Obrigatório**
Reviews	Avaliação média		Valor com até duas casas decimais entre 0 e 5.	Sim
	Avaliações totais		Número inteiro que corresponde ao número de avaliações realizadas	Sim
	Contadores	5 estrelas	Número de 5 estrelas dadas ao livro.	Sim
		4.5 Estrelas	Número de 4.5 estrelas dadas ao livro.	Sim
		4 Estrelas	Número de 4 estrelas dadas ao livro.	Sim
		3.5 Estrelas	Número de 3.5 estrelas dadas ao livro.	Sim
		3 Estrelas	Número de 3 estrelas dadas ao livro.	Sim
		2.5 Estrelas	Número de 2.5 estrelas dadas ao livro.	Sim
		2 Estrelas	Número de 2 estrelas dadas ao livro.	Sim
		1.5 Estrelas	Número de 1.5 estrelas dadas ao livro.	Sim
		1 Estrela	Número de 1 estrelas dadas ao livro.	Sim
	Lista de comentários [Array]	Nif	Número com 9 dígitos que identifique o cliente.	Sim
		Nome	String de apenas letras com o nome do cliente.	Sim
		Data	Data da avaliação.	Sim
		Avaliação	Um dos seguintes valores: 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5 ou 5.	Sim
		Comentário	Texto com o comentário do cliente.	Sim

Com esta nova estrutura é possível aprimorar uma das novas funcionalidades que nasce com o desenvolvimento do *frontoffice* que é a apresentação dos diversos livros na página inicial da aplicação, onde é possível observar qual a avaliação do livro.

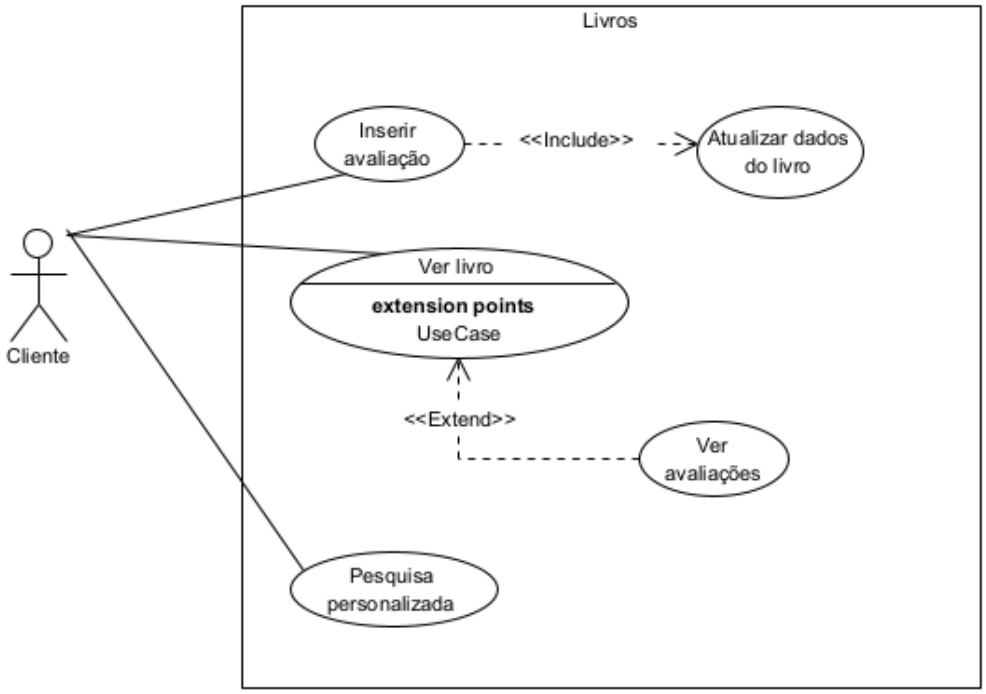


Figura 39 - Use case "Livros"

Como é possível verificar no diagrama de use case, o cliente tem a possibilidade de inserir uma nova avaliação, onde é criado um espaço reservado para esta nova funcionalidade na aplicação, onde de forma indireta e obrigatória é necessário atualizar os dados dos livros no sentido de inserir um novo comentário e atualizar todos os contadores e médias relativas às avaliações relativas aquele livro em específico. A outra atividade importante é a visualização de um livro, onde está relacionada a visualização das avaliações dos clientes ao livro, sendo que é um conteúdo extra poderá ser escolha do cliente ou não visualizar. O utilizador poderá ainda pesquisar de forma personalizada por um conjunto de livros segundo os seguintes critérios: pesquisa por título, por nome do autor ou pelo ISBN.

As funcionalidades acima relatadas ocorrem sempre que o utilizador acede à página inicial e é possível visualizar todos os livros existentes na loja, quando vê detalhes de livro ou quando insere uma avaliação referente a um determinado livro.

Figura 40 - Mockup "Detalhes do livro 1"

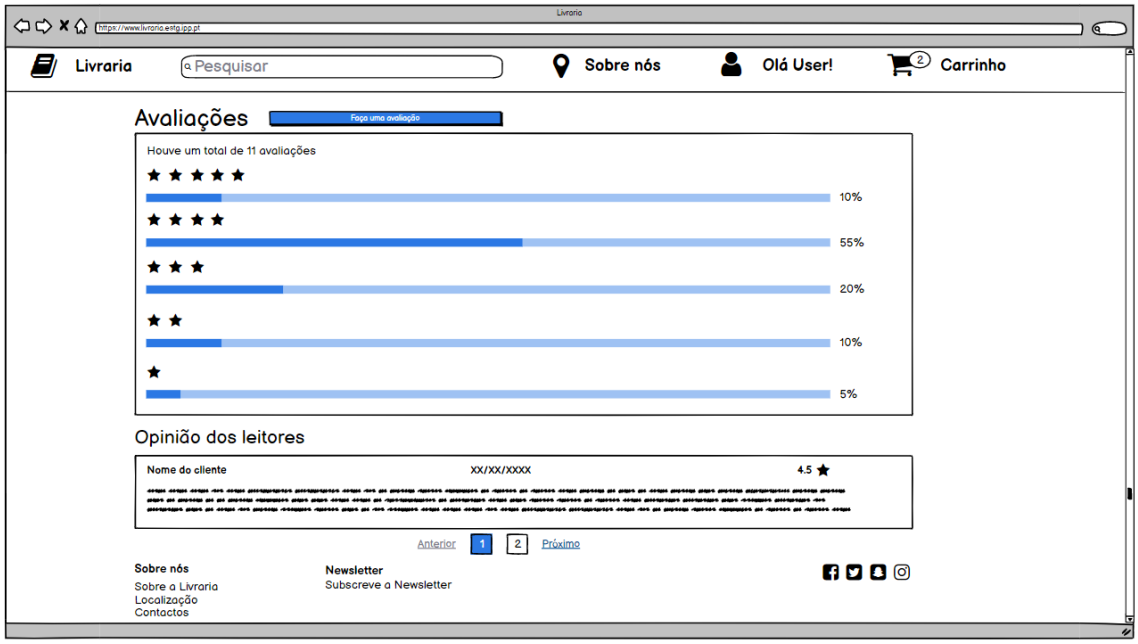


Figura 41 - Mockup "Avaliações dos livros"

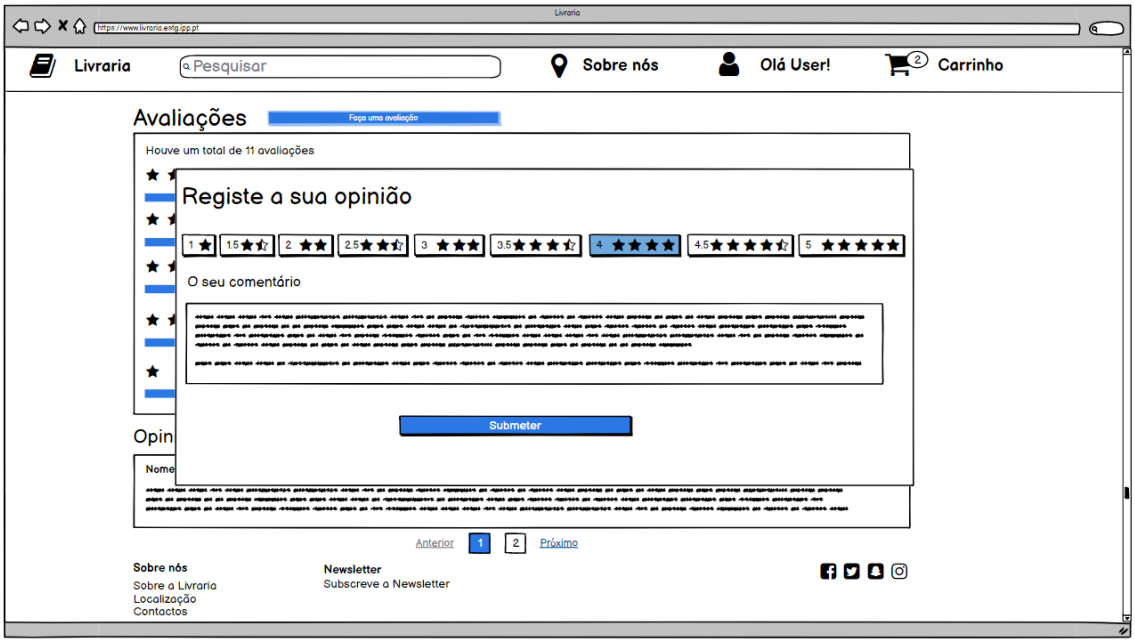


Figura 42 - Mockup "Registar uma avaliação"

Newsletter

Como referido a Newsletter vem combater a necessidade de anunciar e promover a livraria. A Newsletter é enviada a partir do *backoffice* por um funcionário ou gerente, mas é a partir do *frontoffice* que o cliente pode aderir à newsletter. A adesão à newsletter não é vitalícia, tendo o cliente a possibilidade de se desincrever quando quiser.

Foi identificado como necessário criar dois tipos de newsletter. Existe a newsletter considerada normal que seria utilizada para avisos da livraria ou promoções, essa newsletter segue o formato do mockup apresentado a seguir.

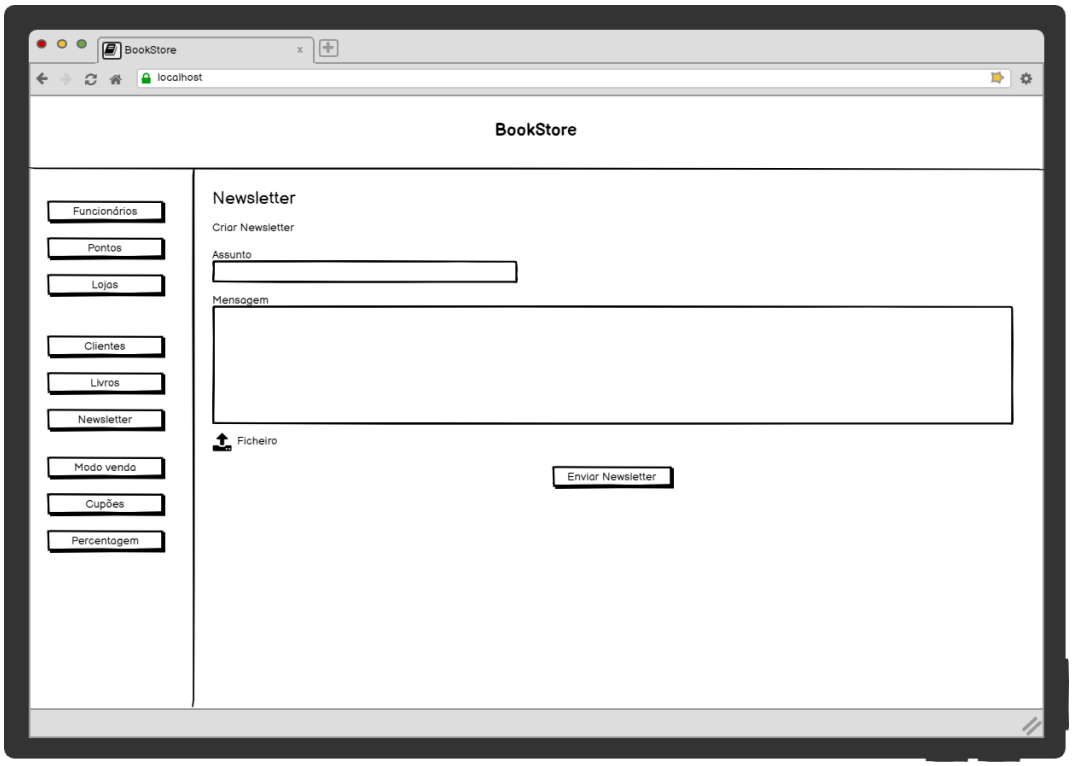
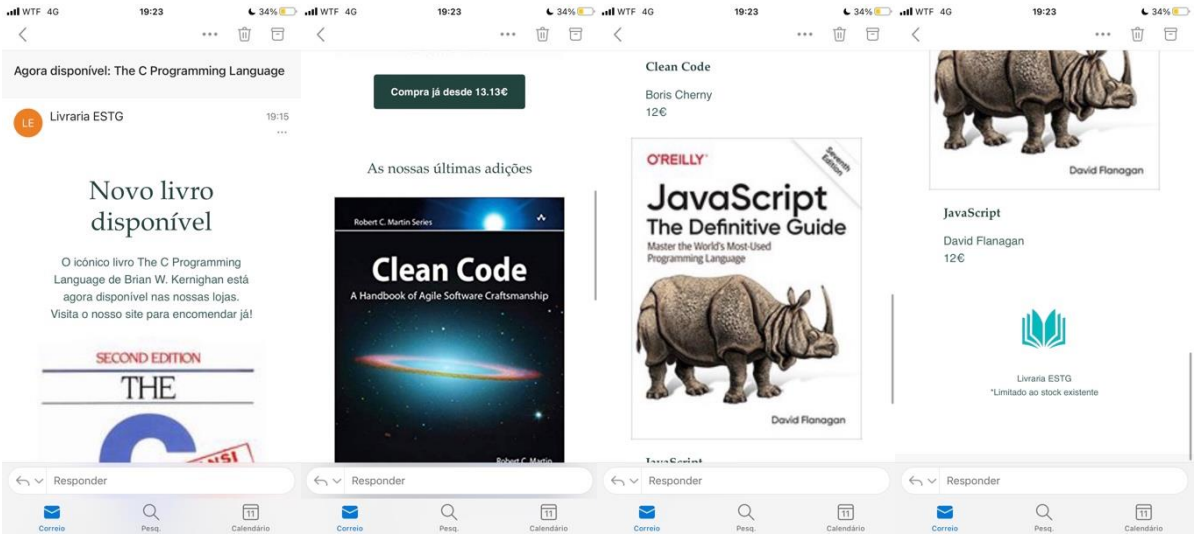


Figura 43 - Mockup newsletter

Para além dessa newsletter, foi implementado também uma newsletter para promover a adição de um novo livro ao sistema, sendo ativada através de uma check-box no processo de criação do livro. Nesse caso é utilizado como template um html que conterá a fotografia do livro, preço, autor e nome, assim como os últimos livros inseridos no sistema. Considerando que grande parte das pessoas verifica o email a partir do telemóvel, o template utilizado foi vocacionado para esse propósito, como mostra as imagens seguintes:



Análise da estrutura do projeto

Neste capítulo é analisado a estrutura do projeto de forma geral e sem grande detalhe, de modo que seja introduzido a forma de organização e de implementação num nível técnico. Serve de introdução para o próximo capítulo, onde é apresentado de forma detalhada os diferentes pontos mais importantes da aplicação.

Informações gerais do projeto

Com o início deste novo projeto, a loja online, foi necessário reformular a estrutura do projeto para que haja uma maior organização, resultando na criação de três repositórios individuais, dois deles responsáveis pelo *frontend* e *backend*, um terceiro de menor dimensão onde é armazenado os ficheiros para a extensão para o *browser google chrome*, onde será abordada mais à frente no documento.

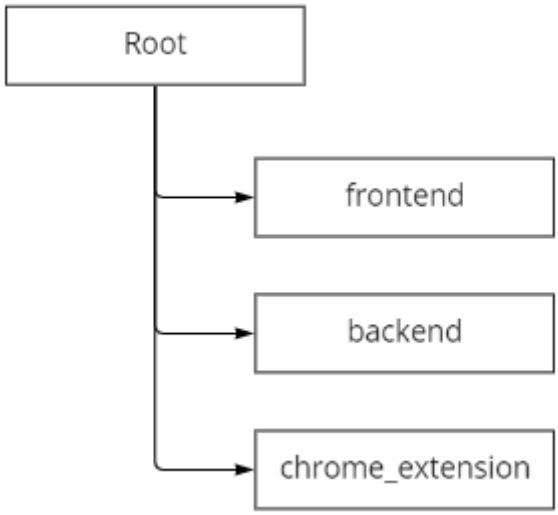


Figura 44-Estrutura root do projeto

De seguida é focado a organização da parte do *frontend*, dado que a parte de *backend* foi abordada na parte referente ao primeiro *milestone*.

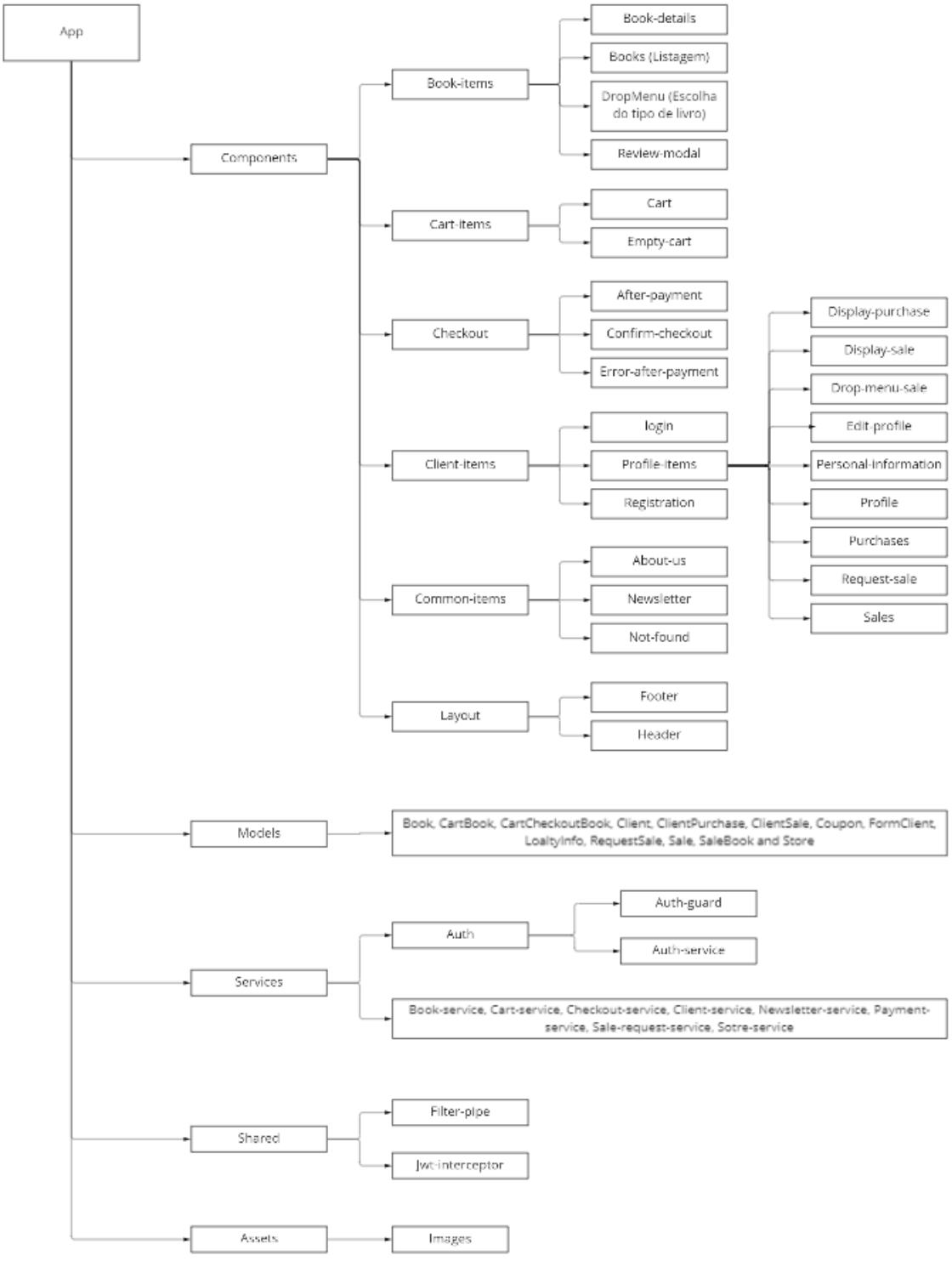


Figura 45- organização do projeto frontend

Analisando a organização é possível reparar que houve uma divisão sobre os grandes módulos:

- Components

  Nesta pasta é possível encontrar os componentes que a aplicação necessita, sendo que estão divididos pelos grandes módulos da aplicação. Cada nome de componente

acaba por ser explicito sobre a sua aplicação, mas serão mais à frente detalhados quando for abordado os principais pontos chave da aplicação.

- Models

Esta pasta é responsável por possuir todos os modelos de dados que a aplicação necessita.

- Services

Numa vertente relacionada com os *models* e componentes, foram criados os serviços essenciais que ligam os componenetes e os *models* para ser criado um bom funcionamento.

- Shared

Nesta paste é possível encontrar ficheiros que são partilhados por todo o projeto, como o caso do *jwt-interceptor*, onde por cada pedido HTTP realizado é ativado a sua funcionalidade, bem como o *filter-pipe*, onde poderá ser usado em múltiplos componentes, mas para este caso apenas é usado num.

- Assets

Na seguinte localização é possível encontrar todos os ficheiros auxiliares, no caso atual são guardadas as imagens.

Para uma maior escalabilidade o ideal era a criação de um ficheiro que fosse capaz de guardar o domínio atual do *backend* e do *frontend*, gênero de um ficheiro *.env*, de forma a ser mais fácil a alteração de domínio. Apesar da não implementação existe a consciência que a forma ideal necessitaria deste ficheiro.

Organização do template

De forma a haver reutilização de componentes, foi implementado o seguinte esquema:

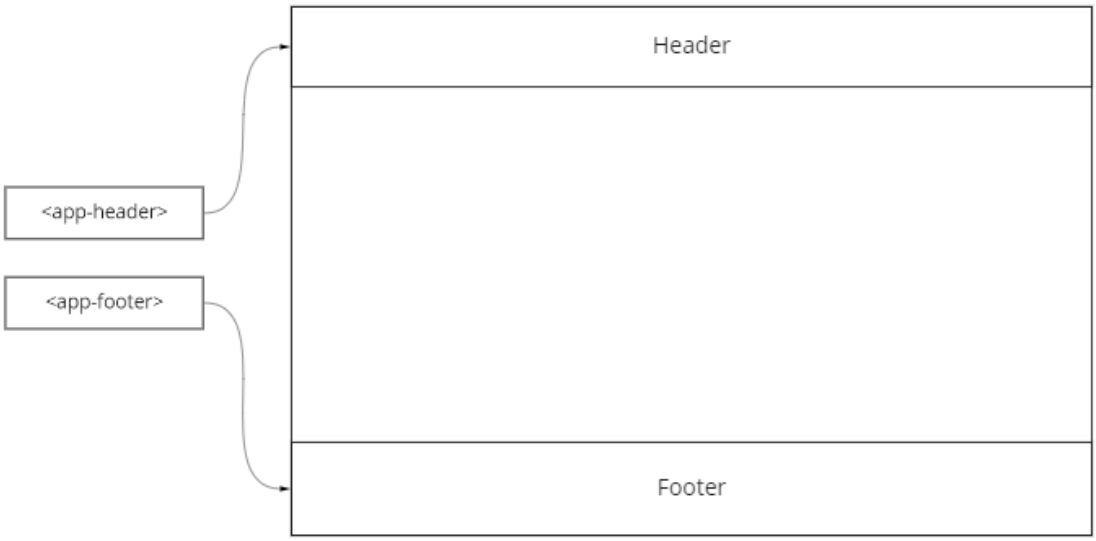


Figura 46-Esquema template segunda milestone

Com a inserção dos dois componentes *app-header* e *app-footer* no ficheiro app-component.html garantimos que em qualquer página será possível visualizar o *header* e o *footer*, sendo que o conteúdo da página será gerido consoante as necessidades e será colocado sempre entre o *header* e o *footer*, dado que em alguns casos o conteúdo da página será um conjunto de agregações de componentes.

Imports necessários

De seguida será enumerado e descrito os packages, bibliotecas ou serviços mais usados ao longo do desenvolvimento.

Auth0/Angular-jwt

Para que haja uma gestão mais eficaz sobre os *tokens* de autenticação gerados, ao invés de ser deixado todo o processo de validação do *token* do lado do servidor, é usada este módulo no sentido de gerir a *token* também no lado do cliente.

Com o uso desta ferramenta há a possibilidade de antes de algum pedido, ou no acesso a algum componente restrito a cliente com login realizado, ser verificado se o *token* está atualmente válido. É também dado a possibilidade de ser conhecido os dados que foram usados para a criação do *token*.

Referência: https://www.npmjs.com/package/@auth0/angular-jwt.

Ngx-pagination

Ao longo da aplicação existem listagens de dados, onde para uma utilização mais amigável da aplicação é implementado paginação.

No *frontoffice* é aplicado uma outra estratégia de paginação, vale recordar que no *backoffice* a paginação é feita com o auxílio do servidor, onde é enviado nos parâmetros qual o número de página a disponibilizar e o servidor trata de escolher os dados, no caso do *frontend* haverá um pedido ao servidor sobre todos os dados existentes e de seguida na parte do cliente é gerida a paginação. Esta abordagem difere do *backoffice* pois no *backoffice* quando era necessário apresentar as encomendas eram todas as encomendas de todos os clientes, o que seria uma grande sobrecarga, neste caso será apenas as encomendas de um cliente específico, o mesmo se aplica para as vendas, no que toca aos livros, o número de livros é o mesmo, mas se recebêssemos apenas uma quantidade reduzida de livros o pesquisar não seria eficiente, pois iriamos estar a pesquisar sobre um número reduzido de livros.

Referência: https://www.npmjs.com/package/ngx-pagination.

Ng-toast

Devido a quantidade de operações existentes e as possíveis respostas por parte do sistema, foi utilizada a biblioteca *Ng-toast* que tem como objetivo simplificar as notificações feitas ao utilizador na forma de pop-up.

Referência: https://www.npmjs.com/package/ng-toast.

Angular-Material e Bootstrap

De forma a ser apresentado um aspeto agradável ao cliente, foi utilizado o angular-material juntamente com o *bootstrap*.

Principais pontos chave da aplicação

Será agora abordado as questões técnicas referentes aos pontos mais importantes da aplicação.

Processo de venda de um livro ao cliente

O processo da venda é a peça mais importante de todo o sistema, tanto a nível de negócio, pois é o responsável pela faturação da empresa, mas bem como a nível de segurança, dado que é importante que a implementação consiga proteger bem a rota de forma a não ser possível prejudicar a livraria.

Este processo irá usar parte da lógica que tinha sido anteriormente implementada para o registo de uma venda pelo *backoffice*. Recordamos que um processo de registo em loja precisava de verificar apenas os seguintes tópicos: autenticação do funcionário, verificação do cliente, verificação dos livros, verificação dos pontos e a fase final. Num processo de compra na loja online é necessário acrescentar uma fase dedicada aos cupões, criar um mecanismo de modo a controlar a questão do envio do livro, ou seja, o servidor deve receber as possibilidades de envio e agir perante isso, bem como os portes grátis que os cientes poderão ter acesso se tiverem um determinado número de pontos.

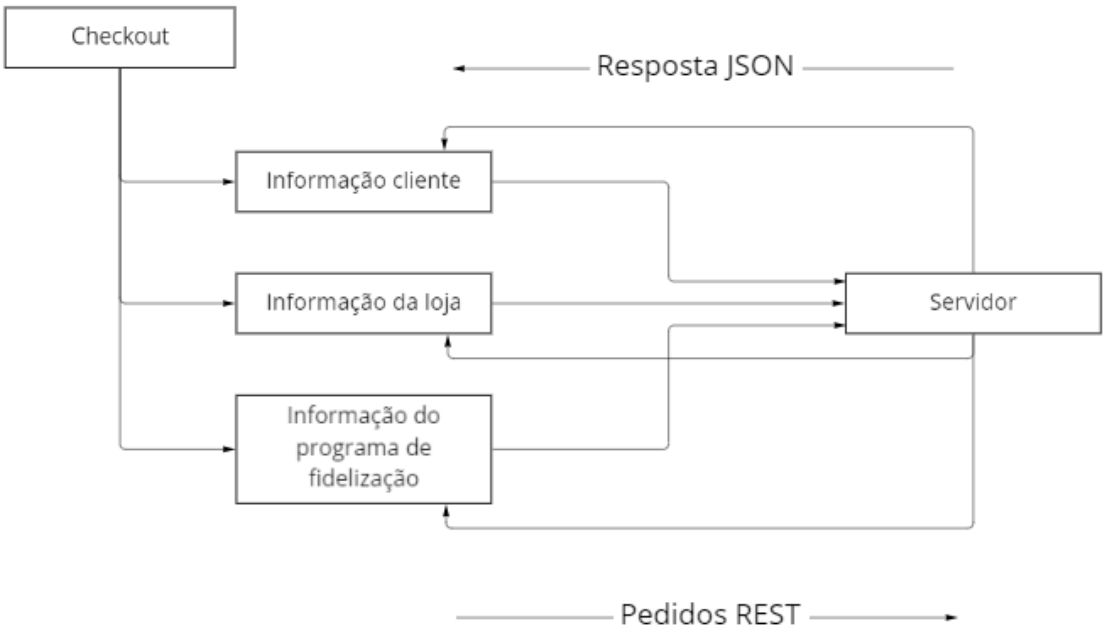


Figura 47- Pedidios iniciais do "Checkout - Website"

A imagem acima ilustra o fluxo de trabalho quando o componente *Checkout* é apresentado. É necessário que seja atualizada toda a informação referente ao cliente, à loja e ao programa de fidelização, de forma a ser garantido que as verificações ocorrerão

com dados atualizados, em casos raros poderá haver o uso do checkout com dados desatualizados que numa primeira visão não haverá problemas para o utilizador, mas que serão bloqueados pelo servidos quando houver intenção de compra por parte do cliente, e para prevenir isso é utilizado pedidos REST.

Pontos da compra:

- Pontos

O cliente poderá usufruir do programa de pontos, onde o funcionamento se mantém igual comparado à forma como ocorrida no backoffice.

Para esta parte o servidor necessita de ter armazenada a informação relativa ao cliente de forma a ser possível obter o número de pontos atualizados do cliente e também saber qual o desconto que a livraria no momento está a oferecer por cada 100 pontos.

É dado também a oportunidade ao cliente de usufruir de portes gratuitos e para isso o cliente deve escolher a opção caso tenha pontos para tal, a quantidade que dita se o cliente pode usufruir de portes grátis ou não vêm do lado do servidor, semelhantemente ao que acontece com o desconto por 100 pontos.

- Cupões

A parte dos cupões é algo exclusivo de compras online, onde tem o seguinte processo de trabalho:

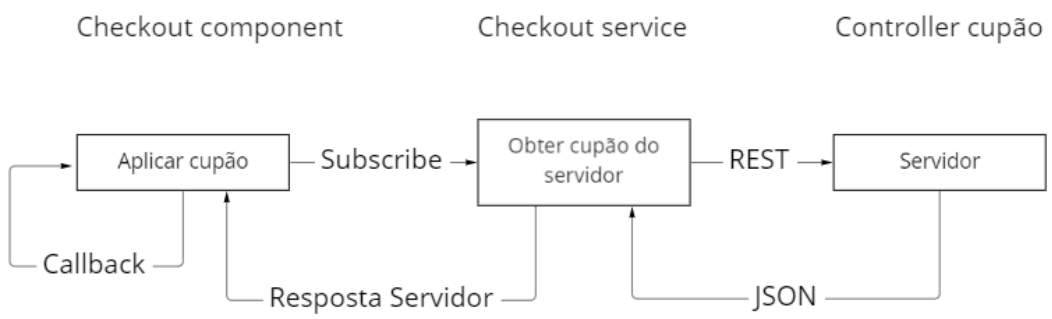


Figura 48-Fluxo da parte do cupão em Checkout

Como é possível verificar no diagrama acima, quando o cliente insere um cupão o sistema necessita de verificar se o cupão é válido, para isso o componente faz uso do serviço checkout que irá comunicar com o servidor através de um pedido REST, dado que o componente "fez *subscribe*" logo que o servidor responda o serviço irá notificar o componente onde será ativado a *callback* definida. Nesta *callback* é realizada toda a logica por trás do desconto do cupão, onde é verificado qual o valor total e é aplicado um desconto sobre uma percentagem.

- Detalhes de envio

Nesta secção é dado três opções de escolha, sendo que o cliente obrigatoriamente deve escolher uma delas: recolha em loja, recolha na morada atual do cliente ou recolha noutra morada a definir.

A opção de recolha em loja dá ao cliente o acesso a portes gratuitos, todas as outras terão de passar por processos de transporte e por isso é definido uma taxa fixa a ser cobrada ao cliente.

Caso o utilizador escolha a recolha em loja ou recolha na sua morada atual é mostrado os dados que foram carregados no início do componente que por sua vez podem não dizer respeito à realidade, mas com o registo da venda os dados são verificados e são registados corretamente. Com isto pode induzir em erro o cliente, mas a probabilidade de acontecimento é muito baixa, dado que se a morada do cliente for alterada, entretanto, será ele próprio a alterar e a alteração de morada da loja é pouco provável e caso aconteça a empresa fará questão de anunciar de forma clara.

- Pagamento

Após a validação dos dados introduzidos pelo cliente para finalizar a compra, ao carregar em avançar para pagamento a compra é criada num registo "temporário" com o estado de "Aguarda pagamento". Com a correta criação da compra a fase de pagamento é apresentada ao cliente.

O pagamento é possível através da *API Stripe*. Para a sua utilização foi criado um criado um *endpoint* "*checkoutSession*" que lida com a *Stripe API*. A premissa é simples: o pagamento é realizado numa sessão fora do domínio atual de forma a manter os dados de pagamento seguros. Essa sessão é criada no *endpoint* "*checkoutSession*" que através da *API key* obtida para o efeito vai criar e registar pedidos associados à conta da livraria.

A validação dos métodos de pagamento é da responsabilidade da sessão e é apenas enviado para a mesma o valor total a ser pago pelo cliente e o ID da compra que foi criada temporariamente. Visto que é necessário configurar o endereço de destino quando o pagamento for bem-sucedido, o ID da compra assume uma especial importância nesse endereço. O endereço de sucesso trata-se de um *endpoint* que tem como parâmetro o id da compra que é responsável por validar a compra que foi feita temporariamente, ou seja, alterar o seu estado e alterar stocks e informações do cliente.

Após a confirmação da compra o utilizador seria reencaminhado para uma página de sucesso ou erro dependendo da existência de erros na atualização da compra.

Podemos verificar que a solução implementada não é ideal por dois motivos: o retrocesso e avanço no pagamento causaria múltiplas criações de compras quando estaríamos perante uma única compra válida e também o facto de que facilmente se poderia confirmar uma compra sem efetuar o pagamento, passando o ID da compra pelo *endpoint* que confirma a compra.

A solução ideal passaria pela utilização de *webhooks*, mais propriamente o *webhook* disponibilizado pelo *Stripe*. Um *webhook* consiste numa ferramenta de escuta que pode ter várias ações conforme o despoletar de certos eventos.

Embora o *Stripe* disponibilize a utilização simples de *webhooks* a mesma exige que o domínio do site seja público e devido ao cronograma inicial deste projeto essa solução não era possível. No entanto, o *Stripe* também disponibiliza a versão local de *webhooks*, mas após a pesquisa e análise das ferramentas necessárias foi possível concluir que essa solução não era de todo viável. Essa implementação local exigiria a configuração de dados de acesso ao *stripe*, a criação de *triggers*, instalação de bibliotecas e ferramentas, tudo métodos que não seriam exportadas apenas num clique ou num comando, dificultando dessa forma a utilização deste projeto noutras máquinas.

Mesmo não possuindo a implementação ideal, podemos concluir que a mesma passaria pela utilização de *weebhooks* definidos na *dashboard* da *Stripe* que despoletaria um *endpoint* presente no projeto (que estaria num domínio publico), onde seria criada a compra.

Newsletter

A implementação do serviço de newsletter tem como base a utilização da biblioteca *nodemailer*. Esta biblioteca permite o envio fácil de emails através no Node.js. Para que fosse possível utilizar esta funcionalidade foi criado um email Outlook e guardado os seus dados de acesso no ficheiro env.

O *nodemailer* é configurado através do *transport*. O *transport* é um objecto de configuração do serviço de email, que entre outras informações, é no *transport* que é feita a configuração SMTP para a utilização do Outlook como remente do email. Visto que a autenticação é feita apenas com email e password fora do serviço oficial de email é importante realçar o atributo *rejectUnauthorized* presente na configuração do *transport* que permite que o mesmo se autentique de forma não segura como a presente.

Através do objeto *transporter* é enviado o email com as informações do remetente, destinatários, assunto, mensagem e ficheiros anexados no mesmo.

No caso de envio do email ser o anúncio de um novo livro é utilizado um *template* html. Este *template* html é na verdade um ficheiro *handlebars*. A biblioteca e tipo de ficheiro *handlebars* permite a compilação de *template* dinâmicos através da passagem de parâmetros para diversos pontos do *template*. Antes do envio do email o *handlebars* é configurado de forma a puder compilar o *template* com os dados do novo livro. Aquando do envio é enviado o *template* compilado com dados e ficheiros necessários.

Avaliações de livros

No sentido de aumentar a confiança no cliente foi criado um sistema de avaliação sobre os livros.

O desenvolvimento desta funcionalidade acaba por ser semelhante a outras que que envolvam as operações básicas de criação, visualização e eliminação.

Para a criação de uma venda é necessário que o cliente esteja com o login efetuado, após isto nos detalhes de livros o processo é explicito, onde o cliente deve escolher qual a pontuação de 0 a 5, em intervalos de 0.5, e qual o comentário a realizar.

Por questões de controlo é dada a oportunidade ao funcionário e gestor de eliminar qualquer avaliação que seja considerada necessária de ser removida.

Em cada remoção os dados gerais das avaliações, nomeadamente a quantidade de avaliações e as médias são atualizadas e o mesmo processo acontece no método de inserir. Foi escolhido realizar os cálculos a cada inserção e remoção para que na apresentação de dados seja prático, pois é praticamente dado como certo que um livro será muitas mais vezes visualizado do que avaliado.

Extensão Google Chrome

De forma a ser implementado algo além daquilo lecionado, não no sentido de linguagem, mas de resultado, o grupo teve por ambição a criação de uma extensão para o browser Google Chrome, mesmo que a sua funcionalidade seja algo simples poderá ser algo que num projeto real pudesse ajudar a angariar algumas vendas face a concorrentes.

A ideia por trás desta extensão é criar algo que esteja disponível a qualquer momento e a qualquer hora ao cliente, sem exigir que este esteja a navegar na loja online da livraria. Com esta funcionalidade, caso o cliente esteja numa loja online concorrente e encontre um livro que pretende comprar, a função da extensão é fazer com que o utilizador antes de finalizar a compra na loja concorrente, abra a extensão, copie o ISBN do livro e procure pelo livro na extensão, esta irá anunciar se o livro encontra-se disponível ou não, caso o livro exista será anunciado o menor preço em que se pode comprar o livro e caso o utilizador deseje será reencaminhado para a página, com isto é aumentado a probabilidade de compra com o utilizador a navegar fora do domínio da loja online.

No que toca à parte de desenvolvimento, o maior desafio foi como preparar a extensão e perceber os conceitos relativos aos *scripts* de *background*, funcionalidade essa que para este tipo de extensão não é necessário, após a extensão ter o documento *manifest* criado, é necessário desenvolver a parte funcional que acaba por ser um projeto com uso das linguagens básicas de desenvolvimento *web*: *html*, *javascript* e *css*.

Referência: https://developer.chrome.com/docs/extensions/mv3/getstarted/.

Carrinho de compras

Para o carrinho de compras foi utilizado uma instância de *BehaviorSubject,* com o objetivo de ser tratado nos diferentes componentes como um mensageiro. Com o uso desta instância, sempre que houver alterações na mesma, todos os componentes que subescreveram irão ser notificados e irá ser executado a função de *callback* estipulada.

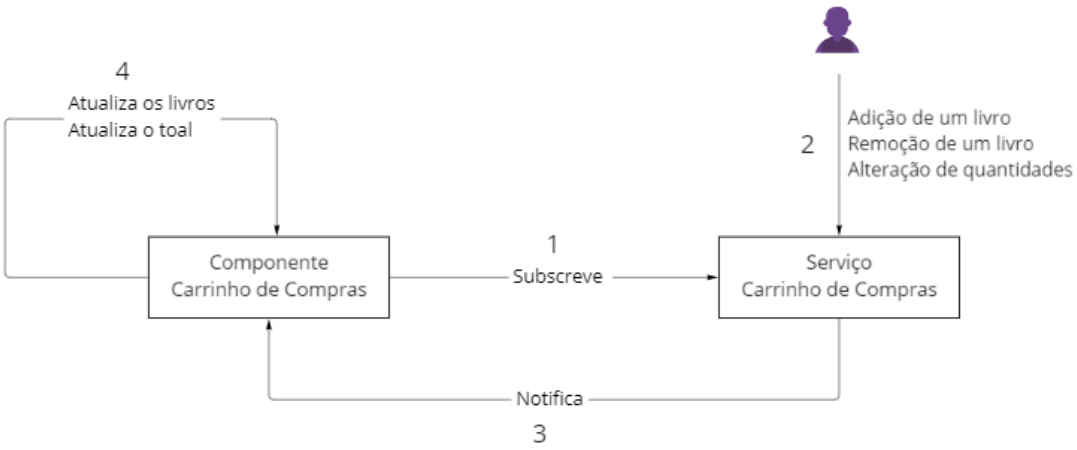


Figura 49-Comunicação entre componente de compras e serviço de compras

Como é possível observar na imagem acima o primeiro passo é a subscrição por parte do componente à instância de *BehaviorSubject* do serviço de compras*,* onde a cada movimento do utilizador perante os livros, como o inserir um novo livro, remover um livro, alterar quantidades, remover todo o carrinho, o componente é notificado e é ativado a função de *callback*, sendo que será atualizado os livros que estão no carrinho e calculado de novo o total a ser cobrado.

A retenção de informação tem como auxílio o *localstorage*, para que seja possível ao utilizador continuar a ter os mesmos livros no carrinho independentemente do tempo passado.