

POLITÉCNICO DO PORTO
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Desenvolvimento de uma plataforma e-commerce para jogos de tabuleiro utilizando o Symfony

André Vieira Santos

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas



DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA
Instituto Superior de Engenharia do Porto

Dezembro, 2022

(Opcional) Poderá usar esta secção para dedicar o trabalho a alguém...

Agradecimentos

(Opcional) Agradecimentos que sejam devidos...

Resumo

Desde que a *World Wide Web* (WWW) foi inventada na década de 1990, o uso da Internet aumentou rapidamente. A Internet faz hoje parte do dia a dia de muitas pessoas e existem sites para quase todos os tipos de serviços. Atualmente, existem muitas tecnologias de software disponíveis para desenvolvimento de aplicações Web, todas diferentes nas suas capacidades. Assim surgiram várias *frameworks*, que reduzem e simplificam o processo de desenvolvimento de uma aplicação web. Um *framework* agiliza o desenvolvimento de aplicações automatizando muitos dos padrões empregados para uma determinada finalidade. Um *framework* também adiciona estrutura ao código, levando o desenvolvedor a escrever um código melhor, mais legível e mais sustentável. O comércio global cada vez mais se integra nas tecnologias emergentes em busca de simplicidade e acesso ao consumidor. Assim vários desenvolvedores de software web procuram desenvolver lojas *e-commerce* que se responsabilizem por tratar do comércio na internet. Este relatório apresenta e demonstra um exemplo de uma loja de jogos de tabuleiro utilizando ferramentas do panorama geral das funcionalidades da *framework* Symfony. A estrutura geral da *framework* é escrita em PHP 5+ e é compatível com muitos dos mecanismos de base de dados populares, incluindo MySQL utilizado neste projeto. Esta loja tem diversas funcionalidades quer seja administrador ou um consumidor, como editar, remover e adicionar produtos quando administrador e ver detalhadamente os produtos e adicionar ao carrinho para efetuar uma compra quando utilizador, entre muitas outras funcionalidades.

Palavras-Chave: Symfony, framework, PHP, WWW

Abstract

Since the *World Wide Web* (WWW) was invented in the 1990s, Internet usage has increased rapidly. The Internet is now part of many people's daily lives and there are websites for almost all types of services. Currently, there are many software technologies available for web application development, all different in their capabilities. Thus, several *frameworks* emerged, which reduce and simplify the development process of a web application. A *framework* streamlines application development by automating many of the patterns employed for a given purpose. A *framework* also adds structure to the code, leading the developer to write better, more readable and more maintainable code. Global commerce increasingly integrates with emerging technologies in pursuit of simplicity and consumer access. Thus, several web software developers seek to develop *e-commerce* stores that are responsible for dealing with Internet commerce. This report presents and demonstrates an example of a board games store using tools from the overview of *framework* Symfony's features. The general structure of the *framework* is written in PHP 5+ and is compatible with many of the popular database engines, including MySQL used in this project. This store has several features whether you are an administrator or a consumer, such as editing, removing and adding products when an administrator and seeing the products in detail and adding to the cart to make a purchase when a user, among many other features.

Keywords: Symfony, framework, PHP, WWW, e-commerce

Índice

1	Introdução	1
1.1	Problema	1
1.2	Requisitos	1
1.2.1	Atores	2
1.2.2	Histórias do Utilizador	3
	Como Administrador	3
	Como Utilizador	4
1.3	Tecnologias	4
1.4	Estrutura do Relatório	5
2	Desenvolvimento da Solução	7
2.1	Arquitetura	7
2.2	Modelo de Dados	8
2.2.1	Interpretação do Modelo de Dados	9
2.2.2	Aplicação do Modelo de Dados	10
2.3	Aplicação	11
2.3.1	Mapa de Aplicação	11
2.3.2	Funcionalidades	11
3	Testes	15
3.1	Debug	15
3.2	Funcionais	16
4	Conclusão	17
4.1	Resultados	17
4.2	Trabalho Futuro	17
5	Anexo	19
5.1	Demonstração	19
	Referências	23

Capítulo 1

Introdução

Este capítulo tem como principal objetivo dar a conhecer o que se pretende desenvolver neste projeto, assim como o problema emergente que determina o propósito central como as suas respetivas propostas de resolução acompanhado das metodologias que se pretendam utilizar.

1.1 Problema

Neste projeto pretende-se desenvolver uma interface web de e-commerce, no âmbito de jogos de tabuleiro. Então, é necessário conceber uma aplicação web que seja capaz de desempenhar as tarefas necessárias para um administrador de uma loja de tabuleiros dispor artigos da sua loja para venda cujos podem ser adquiridos por consumidores que pretendam visitar a loja.

1.2 Requisitos

Para desenvolver uma aplicação de e-commerce na web, existem alguns requisitos que precisam de ser considerados a nível de desenvolvimento de software:

- Um servidor web para hospedar a aplicação e servir utilizadores da internet.
- Uma base de dados para armazenar informação sobre produtos, clientes e outros.

- Uma linguagem de programação para construir a lógica e funcionalidade da aplicação.
- Um design de front-end para implementar e embelezar a interface da aplicação.
- Uma framework para estruturar, organizar e simplificar o código da aplicação.

Este tópico irá ser abordado com as respetivas tecnologias da aplicação escolhidas no capítulo 1.3.

Relativamente às funcionalidades da aplicação, também podem ser enumerados alguns tópicos que enaltecem as principais funções que a aplicação deve conter:

1. Listar produtos para o consumidor ver os produtos disponíveis.
2. Contas de utilizador para os utilizadores da aplicação (administradores ou consumidores)
3. Carrinho de compras para o consumidor ver e modificar os artigos que pretende comprar assim como calcular o custo total da sua potencial compra.
4. Recibo para o consumidor observar se a encomenda já foi processada assim como detalhes de entrega
5. Ferramentas de controlo para o administrador poder disponibilizar artigos, remove-los, editá-los e gerir stock e preço.

Assim, justifica-se a implementação de histórias de utilizador para categorizar as várias tarefas que se pretendem desenvolver para esta aplicação. Isto serve como uma forma de capturar os requisitos funcionais da perspetiva do utilizador perante a aplicação.

1.2.1 Atores

Numa história de utilizador, um ator é uma pessoa ou entidade que interage com o sistema a ser desenvolvido. O ator representa o utilizador de um sistema e a história de utilizador descreve como o ator irá utilizar o sistema para completar um determinado objetivo.

Neste caso em particular, afim de simplificar a metodologia utilizada relativamente a funcionalidades da aplicação foram especificados dois tipos de ator: um utilizador comum que representa uma entidade consumidora que poderá utilizar a aplicação web para pesquisar e comprar produtos e um administrador que gere as várias funcionalidades.

O ator em uma história de utilizador é um elemento importante porque ajuda a definir as várias perspetivas do sistema e guia o desenvolvimento da aplicação na criação de um sistema que culmina nas necessidades e objetivos do utilizador.

1.2.2 Histórias do Utilizador

Uma história de utilizar é uma breve descrição de um recurso ou funcionalidade desejada por um utilizador ou cliente. Geralmente segue um formato específico, como apresentado abaixo:

- "Como *ator*, pretendo *executar alguma ação* para que eu possa *alcançar algum objetivo*"

Histórias de utilizadores são utilizadas em metodologias ágeis de desenvolvimento de software como forma de descrever a funcionalidade que deve ser implementada no produto. Estas devem ser curtas, simples e fáceis de entender e são usados para ajudar a hierarquizar o desenvolvimento de novos recursos. As histórias de utilizador geralmente são escritas por gestores de produto ou analistas de negócios e são utilizadas na orientação da equipa de desenvolvimento na criação de um produto que atenda às necessidades e desejos dos utilizadores finais. Tendo em conta os atores envolvidos e os requisitos funcionais da aplicação, categorizam-se as histórias de utilizador em administrador e utilizador nos subcapítulos seguintes.

Como Administrador

1. Como administrador, pretendo adicionar novos produtos à loja para poder oferecer uma gama mais ampla de produtos aos clientes.
2. Como administrador, pretendo editar os detalhes dos produtos existentes para poder manter as informações do produto atualizadas e precisas.
3. Como administrador, pretendo definir a disponibilidade de produtos para poder controlar quais produtos estão visíveis para os clientes.
4. Como administrador, pretendo definir o preço dos produtos para poder refletir com precisão o custo de cada produto.
5. Como administrador, pretendo fazer upload de imagens de produtos para que os clientes possam ver como são os produtos.
6. Como administrador, pretendo organizar os produtos em categorias para que os clientes possam encontrar facilmente os produtos que procuram.
7. Como administrador, pretendo ver uma lista de todos os produtos da loja para poder geri-los facilmente.
8. Como administrador, pretendo excluir produtos da loja se eles não estiverem mais disponíveis ou se estiverem esgotados.

9. Como administrador, pretendo definir a quantidade de produtos em stock para poder acompanhar com precisão o estoque de cada produto.
10. Como administrador, pretendo observar detalhadamente todos os preços e quantidades de um produto desde que foi adicionado.

Como Utilizador

1. Como utilizador, pretendo navegar pelos produtos da loja para poder ver o que está disponível para compra.
2. Como utilizador, pretendo pesquisar produtos específicos por nome ou categoria para encontrar rapidamente o que estou procurando.
3. Como utilizador, pretendo visualizar os detalhes de um produto, incluindo preço, descrição e imagens, para poder tomar uma decisão de compra informada.
4. Como utilizador, pretendo adicionar produtos ao meu carrinho de compras para poder comprar vários itens de uma só vez.
5. Como utilizador, pretendo visualizar os itens em meu carrinho de compras e ver o custo total de minha compra para poder revisar meu pedido antes de finalizar a compra.
6. Como utilizador, pretendo inserir minhas informações de envio e pagamento e concluir minha compra para poder comprar facilmente os produtos que desejo.
7. Como utilizador, pretendo ver o meu histórico de pedidos para poder rever as compras anteriores.
8. Como utilizador, pretendo deixar avaliações e análises de produtos para que outros usuários possam ver meu feedback.
9. Como utilizador, pretendo salvar produtos em uma lista de desejos para que possa encontrá-los facilmente mais tarde.
10. Como utilizador, pretendo receber notificações sobre promoções, novos produtos e outras atualizações para me manter informado sobre a loja.

1.3 Tecnologias

As tecnologias que se pretendem utilizar no desenvolvimento deste projeto têm de coincidir com os requisitos a nível de software retratados no capítulo 1.2. A *framework* Symfony será responsável por estruturar, organizar o código da aplicação. Nomeadamente, esta tecnologia será o cerne da aplicação, ou seja, todas as restantes

escolhas de tecnologias serão escolhidas tendo em conta esta seleção. Um dos fatores que levou à escolha desta tecnologia foi a quantidade de ferramentas e bibliotecas disponíveis que contribuem para um rápido desenvolvimento de uma aplicação web [1]. Também providencia um servidor web.

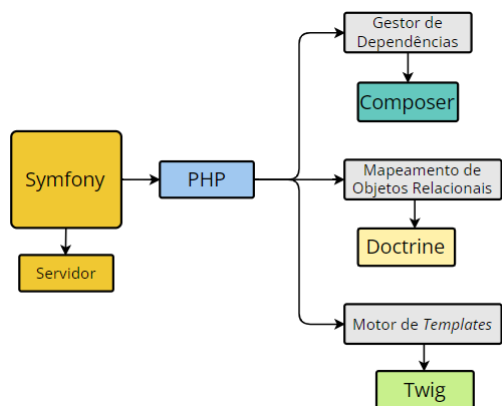


Figura 1.1: Esquema das tecnologias do Symfony da linguagem PHP

A linguagem de programação escolhida foi PHP [2] pois é a linguagem que o Symfony se baseia. No que toca ao design de *front-end*, Twig [3] coopera com o Symfony e com as linguagens base de arquitetura de *front-end* como HTML, CSS e JavaScript. Como base de dados, existem múltiplas opções que satisfaziam este projeto e tinham compatibilidade com o Symfony, contudo foi escolhido MySQL. Esta aplicação foi desenvolvida e estruturada com o apoio de uma plataforma de programação denominada Visual Studio Code. É necessário evidenciar esta plataforma visto que oferece um terminal integrado, comunicação direta com a base de dados para além de uma extensão flexível que permite visualizar e editar os dados facilmente e ligação com o GitHub, uma plataforma a plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git.

1.4 Estrutura do Relatório

Este relatório está estruturado em 4 capítulos principais. No primeiro capítulo (1) é referida a introdução, o problema inicial e as tecnologias utilizadas no projeto. O segundo capítulo (2) envolve a metodologia e desenvolvimento do projeto. O terceiro capítulo (3) engloba uma secção onde são feitos testes à aplicação. Por fim, o ultimo capítulo 4 conclui o projeto com umas notas relativamente a projetos futuros.

Capítulo 2

Desenvolvimento da Solução

Neste capítulo pretende-se abordar a solução escolhida e como será desenvolvida. As decisões e a metodologia abordada no projeto englobar-se-á na totalidade deste capítulo.

2.1 Arquitetura

A arquitetura de software para uma aplicação web refere-se para a estrutura fundamental de uma aplicação, incluindo os seus componentes e as suas relações e os princípios que guiam a arquitetura. Na secção 1.3 foram referidas as tecnologias utilizadas que irão ditar a arquitetura de software da aplicação.

O *Model-View-Controller* (MVC) [4] é um design que agrupa a arquitetura de software de uma aplicação web em três componentes distintos: o modelo, a vista e o controlador. A *framework* Symfony segue este design. No contexto do Symfony, o modelo representa a informação e a lógica da aplicação e é implementada neste projeto com a ajuda da tecnologia Doctrine [5] (ver 2.2.2). A vista representa a interface do utilizador na aplicação e é implementada com o apoio do motor de templates Twig. Twig permite definir *templates* reutilizáveis para renderizar HTML e outro tipo de conteúdo e simplifica a passagem de informação do controlador para a vista. O controlador encontra-se entre o modelo e a vista e é responsável por controlar pedidos do utilizador, interagir com o modelo para buscar ou receber dados quando necessário e determinar que vista é mostrada ao utilizador baseando-se nas suas ações.

Num projeto Symfony, os diretórios são organizados de maneira específica para separar diferentes tipos de arquivos e seguir o padrão de design Model-View-Controller (MVC). Este projeto seguirá a mesma topologia para evitar complicações:

- `bin/`: Contém a aplicação da consola Symfony e outros ficheiros executáveis.
- `config/`: Contém os ficheiros de configuração da aplicação e dos seus pacotes.
- `public/`: Contém o controlador frontal (`index.php`) e outros *assets* como imagens (no diretório `uploads/`) .
- `src/`: Contém o código-fonte da aplicação. Neste local é realizado todo o código e lógica relativa à loja. Também inclui os controladores (`controller/`), modelos (`entity/` e `repository/`) e outros serviços como construtores de formulários (`form/`), arranjos de dados na base de dados (`dataFixtures/`) e segurança (`security/`).
- `templates/`: Contém as vistas separadas por diretórios tendo em conta as rotas e a lógica definidas nos controladores.
- `test/`: Contém os casos de teste da aplicação
- `var/`: Contém os ficheiros automaticamente gerados como *cache*, *logs*, etc.
- `vendor/`: Contém as bibliotecas *third-party* e as *frameworks* instaladas pelo Composer, o gestor de dependências PHP.

Por fim, engloba também alguns ficheiros relativos no diretório inicial nomeadamente ficheiros do Composer , que gerem as dependências do projeto, ficheiros relativos ao Docker e um ficheiro com as variáveis de ambiente do projeto, como a ligação à base de dados MySQL.

2.2 Modelo de Dados

Um modelo de dados é um design ou *blueprint* que especifica como a informação é organizada e estruturada e como pode ser acedida e modificada. No modelo de dados relacional a informação é organizada em tabelas com linhas e colunas e são especificadas relações entre a informação de uma tabela para outra. Estas relações são evidenciadas através de regras e características e são estabelecidas utilizando chaves primárias e secundárias e podem ser de três tipos.

- Um para um (*One-to-One*): cada registo em uma tabela é relacionado com exatamente um único registo noutra tabela
- Um para muitos (*One-to-Many*): cada registo em uma tabela é relacionado com um ou mais registos noutra tabela.

- Muitos para muitos (*Many-to-Many*): cada registo em uma tabela é relacionado com um ou mais registos noutra tabela e cada registo na segunda tabela é relacionado com um ou mais registos na primeira tabela.

Estas relações são especificadas no esquema do capítulo 2.2.1.

2.2.1 Interpretação do Modelo de Dados

No que toca ao modelo de dados específico no caso desta aplicação, foi preciso ter em consideração os dados que serão ser necessários armazenar na base de dados para desenvolver a aplicação.

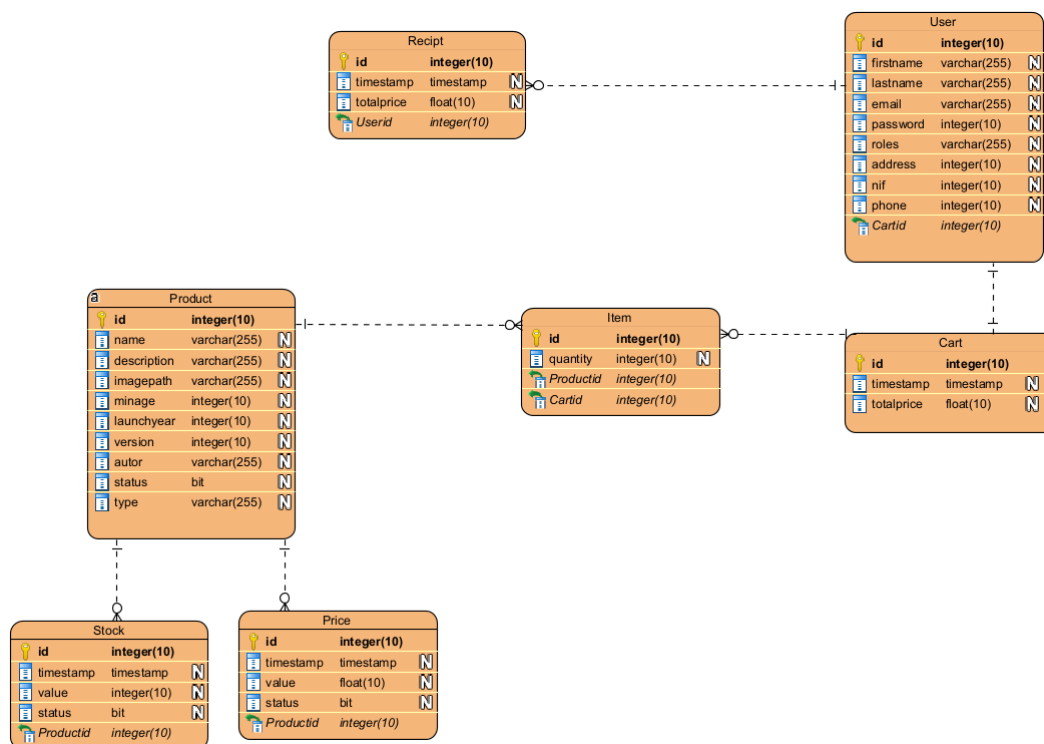


Figura 2.1: Modelo de dados

A tabela **Product** representa os artigos da loja e as suas características. O parâmetro **Imagepath** refere-se à localização (diretório) da imagem respetiva ao artigo no contentor da aplicação. O parâmetro **Status** é comum a muitas outras tabelas e simboliza neste caso em particular se o produto se encontra disponível ou indisponível para o consumidor comprar.

As tabelas **Price** e **Stock** são similares nas suas características e simbolizam o preço e a quantidade de um artigo respetivamente. Tem uma relação de um para um com a tabela **Product** visto que cada produto tem somente um valor de preço e quantidade. Estas tabelas foram criadas para ser possível observar a oscilação de preço/quantidade de um artigo numa escala temporal. O parâmetro **status** serve

para simbolizar que o preço/quantidade que tiver o Status ativo será o preço/quantidade atual do produto (só existirá um único).

A tabela User engloba todos os parâmetros de um utilizador quando é registado na loja. O parâmetro Roles determina se um utilizador é utilizador comum ou é administrador. É necessário ressaltar que um administrador também tem a permissão de um utilizador comum.

A tabela Cart armazena informação relativa a um carrinho de um utilizador. Estabelece uma relação de um para um com a tabela User. Um utilizador pode ter múltiplos produtos no seu carrinho, mas o mesmo produto pode ser adicionado ao carrinho de muitos utilizadores. Esta relação é denominada de muitos para muitos. No entanto, esta relação foi intermediada com uma tabela Item, fazendo uma relação de um para muitos entre a tabela Product e Item e uma relação de muitos para um entre a tabela Item e Cart.

A tabela Item representa uma instância temporária de adição de um produto a um carrinho, isto é, quando é adicionado um produto ao carrinho, no fundo é adicionado um item que contém uma referência ao produto associado da quantidade do mesmo. Um utilizador pode ter múltiplos produtos no seu carrinho com diversas quantidades. Caso um item seja removido do carrinho, ou uma compra seja finalizada, o carrinho é limpo e os itens são removidos da base de dados.

Por fim a tabela Receipt trata os recibos quando uma compra é finalizada e é similar à tabela Cart nas suas características. No entanto, tem uma relação de muitos para um com a tabela User porque um utilizador pode ter múltiplos recibos com os seus dados.

2.2.2 Aplicação do Modelo de Dados

No Doctrine, uma classe ou entidade pode ser convertida a um esquema de base de dados utilizando um processo denominado de "mapping". A entidade é elaborada tendo em conta os parâmetros necessários da base de dados a ser gerada. O esquema é criado sabendo a ligação à base de dados. Quando o esquema é criado, o Doctrine pode ser utilizado para persistir, remover e encontrar informação da base de dados utilizando um gestor de entidades (*EntityManager*) que coopera na integração da lógica do código da aplicação. O Symfony CLI acompanhado do pacote *maker-bundle* possui comandos que possibilitam criar um ficheiro de migração com as *queries* correspondentes às últimas atualizações executadas nas entidades. Posteriormente, pode ser efetuado um comando que efetivamente transfere essas *queries* para a base de dados, neste caso MySQL.

2.3 Aplicação

A aplicação será abordada e estruturada através do seu mapa e as funcionalidades derivadas das histórias de utilizador

2.3.1 Mapa de Aplicação

A estrutura da aplicação no que toca a "routing" envolve muitas páginas e sub-páginas. Essencialmente, pode-se dividir entre seis grandes páginas:

1. Página de Register: Aqui é efetuado um registo de um novo utilizador com os respetivos parâmetros necessários para completar a inscrição, como um e-mail válido e os detalhes de envio. (/register)
2. Pagina de Login: Nesta página efetua-se a ligação de um utilizador á loja que foi previamente registado. (/login)
3. Página Principal: Nesta página visualiza-se os produtos da loja.
4. Página dos Detalhes: Aqui visualiza-se os detalhes tendo em conta o produto selecionado. (/product/<id do produto>)
5. Página do Carrinho: Nesta página visualiza-se o carrinho atual do utilizador.
6. Página do Recibo: Esta página representa uma simulação de um recibo que contém as informações do produto e os detalhes para envio do utilizador. (checkout/)

Posteriormente, cada funcionalidade ou método que não requer uma vista redireciona para a rota pretendida após executar a sua funcionalidade. Estas rotas geralmente possuem o nome do método acompanhado do id do produto. Por exemplo, para desativar um produto como administrador, ao direcionar para a rota (/disable/<id do produto>) executa no controlador a lógica necessária para desativar o produto e por fim redireciona para a pagina principal.

2.3.2 Funcionalidades

As funcionalidades da loja são guiadas tendo em conta o tipo de utilizador que se encontra autenticado na loja. Assim, seguindo as histórias de utilizador enunciadas no capítulo 1.2.2, é possível identificar os passos a seguir de modo a desenvolver a loja. O esquema abaixo representa as funcionalidades completas da loja.

A numeração efetuada no capítulo 1.2.2 servirá como guia para enumerar as tarefas realizadas na loja.

- **Como utilizador comum:**

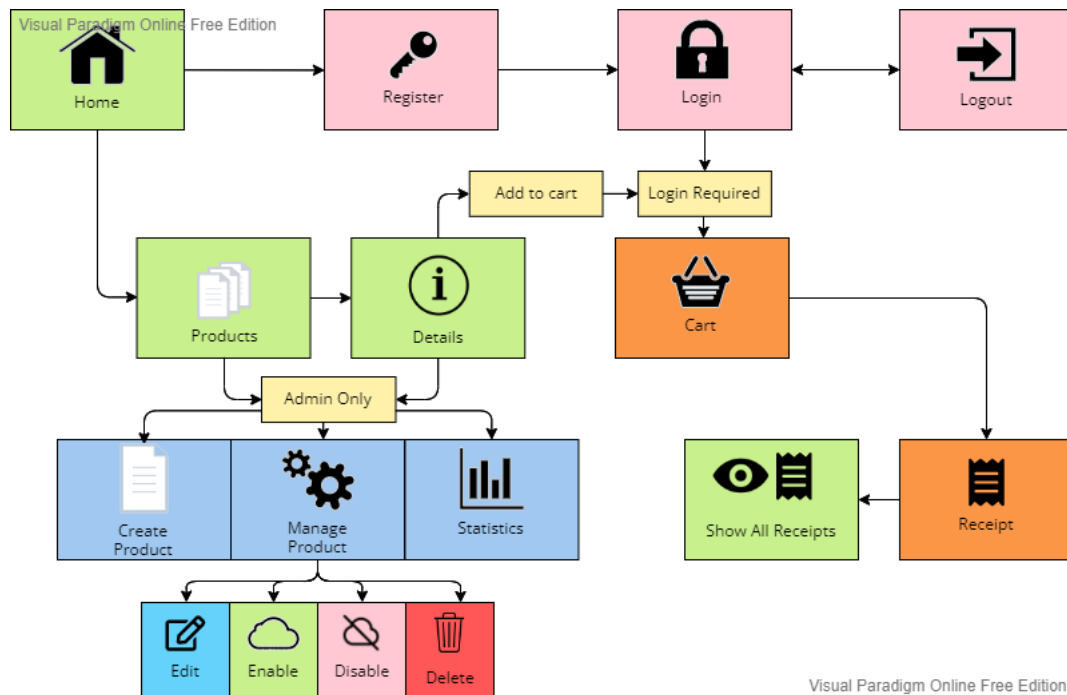


Figura 2.2: Esquema das funcionalidades

A visualização dos produtos é observada na rota principal, no qual se consegue observar apenas todos os produtos (1). Para se dirigir aos detalhes de um produto, basta pressionar o botão "View Details" que direciona o utilizador para a página dos detalhes do produto (3). Aqui consegue-se observar as características do produto assim como uma visão aprimorada do produto. Pode-se também adicionar ao carrinho e visualizar os produtos no carrinho seguindo para a página do carrinho (4 e 5). As informações de envio e pagamento estão diretamente relacionadas com as informações no registo do utilizador (6).

- **Como administrador :**

É possível criar um novo produto através do botão da barra de navegação "Create Product"(1). A secção de controlar as características de um produto pode ser acedida através do botão "Manage Product" para cada produto. Neste botão, observa-se uma lista que enumera as várias opções gerais para controlar o produto: "Edit Product" para editar as características do produto (2)(4)(5)(10), "Disable/Enable Product" para remover/adicionar o produto à visibilidade do utilizador (3) e "Delete Product" para remover completamente o produto (9). Pode-se também observar uma tabela detalhada com todos os antigos preços e quantidades de cada produto na rota que direciona o botão "Statistics".

- **Notas finais sobre as histórias de utilizador incompletas:**

Visto que a loja apenas integra artigos de um só tipo, não é possível categorizar artigos em tipos. A pesquisa de artigos, a integração de um método de pagamento,

a lista de favoritos e as notificações de promoções foram tópicos não desenvolvidos no decorrer desta aplicação.

Capítulo 3

Testes

A *framework* Symfony contribui com um pacote denominado de *symfony/profiler-pack* que é uma poderosa ferramenta para desenvolvedores que fornece informações detalhadas sobre a execução de qualquer pedido HTTP. É considerada uma ferramenta de *debug* porém pode ser utilizada para efetuar simples testes na aplicação. No entanto, o Symfony também possui um pacote específico para testes que pode ser englobado no mapa da aplicação no diretório "tests/". Outra ferramenta que também pode ser utilizada para efetuar testes na API é o Postman.

3.1 Debug

Estes testes simples são apenas para demonstrar algumas estatísticas relativamente ao tempo de acesso à página, a memória máxima obtida, o cache, o tempo de renderização do template twig, e o tempo de execução das *queries* na base de dados.

Rota	Tempo	Memória Máxima	Cache	Twig	Base de Dados
@home	230 ms	20.0 MiB	4 em 0,54 ms	13 ms	3 em 1,62 ms
@registration.register	185 ms	20.0 MiB	4 em 0,56	9 ms	1 em 1,25 ms
@app.login	190 ms	24.0 MiB	290 em 12,06 ms	7 ms	-
@product.detail	184 ms	20 MiB	4 em 0,20 ms	11 ms	3 em 1,38 ms
@app_cart	220 ms	20 MiB	4 em 0,97 ms	7 ms	4 em 2,15 ms
@cart.receipt	298 ms	20 MiB	4 em 0,89 ms	9 ms	21 em 15,46 ms

3.2 Funcionais

Esta secção envolve os testes funcionais da aplicação. Contudo, as ferramentas utilizadas para executar esta tarefa contiveram erros na sua inicialização devido a incoerências na base de dados que não puderam ser resolvidas no *symfony/testing*.

O acesso à autenticação do utilizador através do Postman fica inabilitado devido a conflitos no acesso ao token de segurança csrf. Foram efetuadas tentativas para simular a autenticação do utilizador mas sem sucesso. Assim, não foi possível executar os testes pretendidos que envolvam autenticação (que são praticamente todos). Assim segue abaixo os testes que foram possíveis realizar com o Postman dado as limitações na autenticação:

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	1s 55ms	3	265 ms

All Tests	Passed (3)	Failed (0)	Skipped (0)	View Summary
-----------	------------	------------	-------------	------------------------------

Iteration 1					
GET	Home page	http://127.0.0.1:8000/	/ Home page	200 OK	304 ms 88.117 KB
	Pass	Status code is 200			
GET	Details Page	http://127.0.0.1:8000/product/41	/ Details Page	200 OK	310 ms 70.284 KB
	Pass	Status code is 200			
GET	Login Page	http://127.0.0.1:8000/login	/ Login Page	200 OK	181 ms 67.423 KB
	Pass	Status code is 200			

Figura 3.1: Testes efetuados

Capítulo 4

Conclusão

Este projeto tem como principal objetivo demonstrar um exemplo de uma loja *e-commerce* de jogos de tabuleiro funcional. Também enalteceu algumas das capacidades que a tecnologia Symfony oferece. A loja tem diversas funcionalidades e oferece uma experiência simples para o administrador gerir os produtos da loja e o utilizador de efetuar compras na loja.

4.1 Resultados

As funcionalidades da loja enunciadas através das histórias de utilizador foram realizadas numa perspetiva funcional. A *framework* Symfony coopera e oferece inúmeras ferramentas para o desenvolvimento da loja. De um modo geral, as histórias de utilizador mais relevantes foram completas para uma loja atual *e-commerce*. No Anexo 5 é possível observar alguns recortes da loja.

4.2 Trabalho Futuro

A loja encontra-se funcional e aborda as funcionalidades mais relevantes de uma loja de jogos de tabuleiro. No entanto, existem alguns aspetos que poderiam complementar o projeto num trabalho futuro, por exemplo, a integração de uma ferramenta para adicionar métodos de pagamento e efetuar o pagamento quando é concluída uma compra.

Capítulo 5

Anexo

5.1 Demonstração

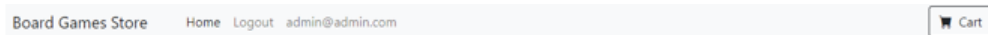


Figura 5.1: Barra de Navegação



Figura 5.2: Produto na vista de um utilizador



Figura 5.3: Produto na vista de um administrador

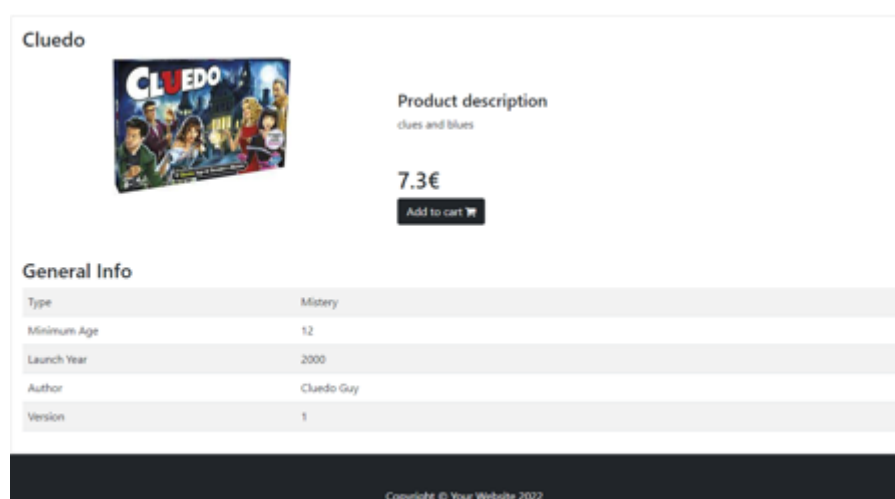



Figura 5.4: Detalhes de um produto

Shopping Cart

3 items



Codenames


Codigo de nomes

1

9.99

×

+



Snakes and Ladders


cobrinhas e escadas

1

6.99

×

+



Mysterium

Jogo misterioso

1

24.99

×

+

Summary

3 ITEMS41.97 €

SHIPPING2.5 €

TOTAL PRICE44.47 €

Buy All

◀ | Back to shop

Figura 5.5: Exemprio de um carrinho




Your order is confirmed!				
Hello André Santos.				
Your order has been confirmed and will be shipped in two days				
Order date	Order number	Payment method	Shipping Address	
2022-12-29 21:39:44	40	Credit card	Rua do Andre	
	Codenames Codigo de nomes	Qty: 1	9.99 €/unit	
			Total: 9.99 €	
	Snakes and Ladders cobrinhas e escadas	Qty: 1	6.99 €/unit	
			Total: 6.99 €	
	Mysterium Jogo misterioso	Qty: 1	24.99 €/unit	
			Total: 24.99 €	
Subtotal		41.97 €		
Shipping fee		2.5 €		
Total		44.47 €		
Expected delivery date		31/12/2022		
We will be sending a shipping confirmation email when the item is shipped!				
Thanks for shopping Board Games Store			Need Help ? Call - 123456789	

Figura 5.6: Exemplo de um recibo

Stock table

Stock Value	Date of Stock Change	Status
999 units	21/12/2022 12:55:35	
987 units	21/12/2022 01:08:35	
986 units	21/12/2022 01:22:17	
985 units	29/12/2022 04:42:52	1

Price table

Price Value	Date of Price Change	Status
10 €	19/12/2022 10:45:56	1

Figura 5.7: Tabelas indicativas da oscilação de preço e stock de um produto

Referências

- [1] SensioLabs, “Symfony docs.” Available at <https://symfony.com/doc>. [Citado na página 5]
- [2] PHP, “Php.” Available at <https://www.php.net/>. [Citado na página 5]
- [3] Twig, “Twig.” Available at <https://twig.symfony.com/>. [Citado na página 5]
- [4] N. A. RF Olanrewaju, T Islam, “An empirical study of the evolution of php mvc framework,” *Springer*, 2015. [Citado na página 7]
- [5] Doctrine, “Doctrine.” Available at <https://www.doctrine-project.org/>. [Citado na página 7]