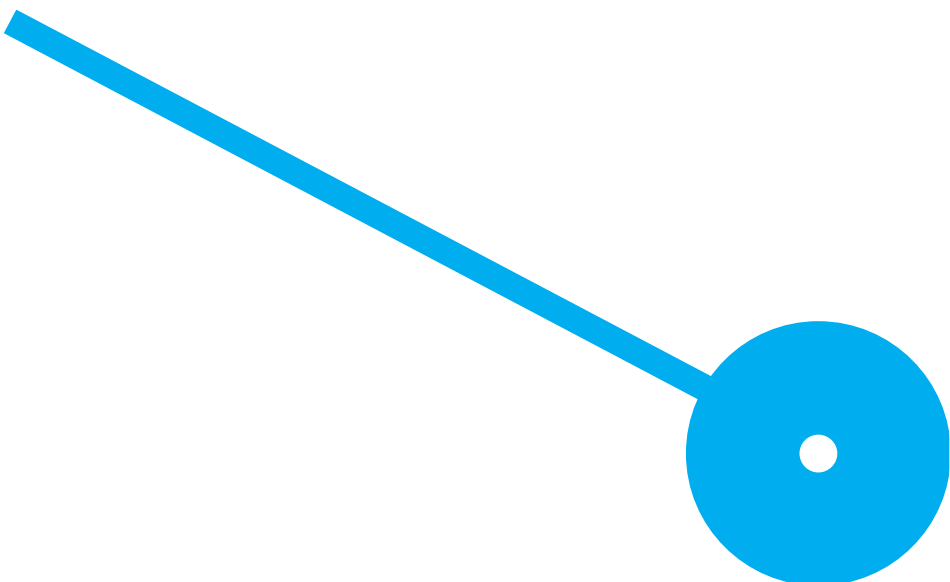


Plataforma de Vendas B2B

Nuno Filipe Oliveira Matos

08/2023





Plataforma de Vendas B2B

Nuno Filipe Oliveira Matos

8160207

Orientador

Prof. Doutor Ricardo Jorge da Silva Santos

Relatório de Estágio apresentado para cumprimento dos requisitos necessários à obtenção do grau de Licenciado em Engenharia Informática pela Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto.

08/2023

Este trabalho não inclui as críticas e sugestões feitas pelo Júri

Declaração de Integridade

Eu, Nuno Filipe Oliveira Matos, estudante nº 8160207, da Licenciatura de Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico do Porto, declaro que não fiz plágio nem auto-plágio, pelo que o trabalho intitulado “Plataforma de Vendas B2B” é original e da minha autoria, não tendo sido usado previamente para qualquer outro fim. Mais declaro que todas as fontes usadas estão citadas, no texto e na bibliografia final, segundo as regras de referência adotadas na instituição.

Este trabalho não inclui as críticas e sugestões feitas pelo Júri

Índice

CAP I – Contextualização e Motivação	10
Introdução	11
Apresentação da Empresa	11
Objetivos do Projeto	11
Resultados Obtidos	12
Organização do Documento	12
Fundamentação Teórica	14
Conceito de B2B	14
ERP <i>Macwin</i>	15
Plataforma Comércio Eletrónico <i>Magento</i>	16
Tecnologia PWA	17
CAP II – Conceptualização do Problema/Projeto	18
Requisitos do Projeto	19
Requisitos da API	19
Requisitos do Site B2B	25
Arquitetura da Solução	27
Arquitetura Tecnológica	27
Arquitetura Conceptual da API	29
CAP III – Metodologia de Operacionalização do Trabalho	37
Processo e Metodologia de Trabalho	38
Metodologia SCRUM	38
Testes Unitários e <i>Software Lint</i>	40
Controlo de Versões e Ferramentas de CI/CD	42
Desenvolvimento da Solução	46
Adaptação do Projeto	46
Gestão de Catálogos	47

Gestão de Artigos.....	50
Parametrização de Atributos	53
Validação de Artigos.....	54
Sincronização para Sites de Vendas.....	55
Desenvolvimento de Site de Vendas B2B	57
CAP IV – Discussão dos Resultados.....	59
Apresentação e Discussão dos Resultados.....	60
Apresentação e Discussão dos Impedimentos e Constrangimentos	60
CAP V – Conclusão	61
Reflexão Crítica dos Resultados.....	62
Conclusão e Trabalho Futuro.....	62
Referências	63

Índice de Tabelas

Tabela 1 - Requisitos da API - Criação de Catálogos por Temporada.	19
Tabela 2 - Requisitos da API - Remoção de Catálogos Independentes dos Artigos.	20
Tabela 3 - Requisitos da API - Edição de Catálogos.	20
Tabela 4 - Requisitos da API - Visualização de Artigos em Função de Filtros.	20
Tabela 5 - Requisitos da API - Edição de Artigos Individuais ou em Massa.	21
Tabela 6 - Requisitos da API - Seleção ou Desmarcação de Artigos para Sincronização.	21
Tabela 7 - Requisitos da API - Remoção de Artigos Individuais do Catálogo.	22
Tabela 8 - Requisitos da API - Validação de Artigos Consoante Requisitos Específicos.	22
Tabela 9 - Requisitos da API - Sincronização de Artigos para Sites de Vendas.	23
Tabela 10 - Requisitos da API - Remoção de Filas de Sincronização.	23
Tabela 11 - Requisitos da API - Criação de Relatórios de Erros de Filas de Sincronização.	23
Tabela 12 - Requisitos da API - Parametrização de Atributos da Aplicação <i>Macwin</i> para Atributos de Sites <i>Magento</i>	24
Tabela 13 - Requisitos da API - Atualização de Dados de Artigos.	24
Tabela 14 - Requisitos do Site B2B - Implementação do Tema Utero.	25
Tabela 15 - Requisitos do Site B2B - Traduções em Inglês e Português.	25
Tabela 16 - Requisitos do Site B2B - Capacidade de Selecionar o País de Acesso.	25
Tabela 17 - Requisitos do Site B2B - Obrigação de Login antes de Aceder ao Site.	26
Tabela 18 - Requisitos do Site B2B - Compra de Artigos em Matriz.	26

Índice de Figuras

Figura 1 - Exemplificação Visual da Arquitetura do Projeto.....	27
Figura 2 - Representação visual da arquitetura conceptual da API de gestão de catálogos.	29
Figura 3 - Rotas e Controladores relacionados com a Gestão de Pedidos e Gestão de Encomendas.....	30
Figura 4 - Rotas e Controladores relacionados com a Gestão de Catálogos.	30
Figura 5 - Exemplificação visual das operações de execução das filas de sincronização.....	34
Figura 6 - Lista de "Services".	36
Figura 7 - Backlog do projeto na ferramenta Jira. É a partir destes pontos que se planeiam o que deve ser feito para o sprint.....	39
Figura 8 - Sprint do projeto na ferramenta Jira. Representam as tarefas a serem realizadas com data de inicio e fim.....	39
Figura 9 - Exemplo de um teste unitário. O objetivo é verificar se, ao alterar os requisitos de um catálogo não existente, o erro ResourceNotFound é lançado.....	40
Figura 10 - Exemplo de uma verificação do código com o software Lint. Neste caso, a verificação não apresentou quaisquer falhas.	41
Figura 11 - Exemplo de algumas alterações efetuadas ao código presente no repositório BitBucket.....	42
Figura 12 - Através do Lens gerimos os Pods do projeto, que contém um ou mais contentores. O Pod relacionado com a API de Gestão de Catálogos encontra-se sublinhado a azul.....	43
Figura 13 - Etapas de integração de modificações do código.	44
Figura 14 - Ferramenta de Criação de Catálogos para a plataforma <i>Magento</i>	47
Figura 15 - Botão para atualização de artigos de um catálogo.....	48
Figura 16 - Filtros de artigos de um catálogo.	49
Figura 17 - Ferramenta de edição de catálogo.....	49
Figura 18 - Botão de remoção de catálogo.	49
Figura 19 - Seleção ou Desmarcação de Artigos e Configurações de Artigos.	50
Figura 20 - Botão de Atualização de Dados de Artigo Especifico.....	50
Figura 21 - Campos de Edição de um Artigo.....	51
Figura 22 - Botão de Remoção de um Artigo.	52
Figura 23 - Parametrização do Atributo Cor.	53

Figura 24 - Mensagens de Erros de Validação de um Artigo.	54
Figura 25 - Fila de Sincronização de um Catálogo.....	55
Figura 26 - Relatório de Erros de uma Fila de Sincronização.	56

Dicionário de Siglas

API → **A**pplication **P**rogramming **I**nterface;
B2B → **B**usiness **2** **B**usiness;
B2C → **B**usiness **2** **C**onsumer;
CI/CD → **C**ontinuous **I**ntegration and **C**ontinuous **D**eployment;
ECI → **E**l **C**orte **I**nglês;
ERP → **E**nterprise **R**esource **P**lanning;
IDE → **I**ntegrated **D**evelopment **E**nvironment;
MVC → **M**odel-**V**iew-**C**ontroller;
PHP → **P**ersonal **H**ome **P**age;
PWA → **P**rogressive **W**eb **A**pplication;
SCRUM → **S**ystematic **C**ustomer **R**esolution **U**nraveling **M**eeting;
SKU → **S**tock **K**eeping **U**nit;
SQL → **S**tructured **Q**uery **L**anguage;
SS23 → **S**pring/**S**ummer **2023**;
URL → **U**niform **R**esource **L**ocator;
YAML → **Y**et **A**nother **M**arkup **L**anguage;



CAP I – Contextualização e Motivação

Introdução

Apresentação da Empresa

A empresa envolvida no projeto de estágio é intitulada *Longratex*. Situada em Felgueiras, a *Longratex* foi estabelecida por uma família envolvida no negócio do vestuário há mais de 30 anos, e é especializada no fabrico de roupa de bebé, criança e senhora para as mais prestigiadas marcas mundiais.

Os principais mercados da empresa são o Reino Unido, França, Espanha, Estados Unidos e a sua capacidade de produção ultrapassa as sessenta mil unidades por mês. Opera a nível mundial através de uma vasta rede que permite a exportação de cerca de 85% do vestuário.

A empresa possui diversas marcas próprias que procuram responder à demanda de diversos mercados distintos. A marca diretamente relacionada com o projeto é chamada *Patachou*.

A *Patachou* é uma marca de moda de luxo acessível infantil, que oferece roupa e acessórios para bebés e crianças dos 0 aos 14 anos. Encontra-se acessível em 18 países e mercados através de conceituadas lojas multimarca em todo o mundo, desde os Estados Unidos até à Coreia do Sul, e tem presença em feiras de prestígio como a Pitti Bimbo na Itália, a *Bubble* no Reino Unido e a *Children's Club* nos Estados Unidos, entre outras.

Objetivos do Projeto

O principal objetivo do projeto incidia na criação de uma plataforma de vendas B2B. Para esse fim pretendia-se desenvolver uma API para a criação, manipulação e envio de catálogos de artigos para o site de vendas. A API tinha como objetivo ser aplicada no site onde se realizam e/ou monitorizam as operações diárias da empresa. Além disso, também se pretendia desenvolver um site de vendas B2B, através da plataforma de comércio eletrónico *Magento*.

Resultados Obtidos

Na altura da conclusão do estágio, a API anteriormente mencionada encontrava-se totalmente concluída e já se encontra em uso pela empresa no desenvolvimento do catálogo correspondente à temporada “Outono/Inverno 2023”.

Já o site de vendas B2B encontra-se ativo, e alcançável [1], porém ainda se encontra em estado de desenvolvimento, tendo a empresa demonstrado interesse em que o estagiário continue o seu desenvolvimento após o término do estágio.

Organização do Documento

Este documento encontra-se dividido em 5 capítulos, cada um separado por subcapítulos de modo a facilitar a leitura e compreensão do mesmo.

O primeiro capítulo, intitulado “Contextualização e Motivação”, tem como propósito explicar a necessidade do projeto dentro do contexto da empresa e proporcionar uma visão geral sobre o desenvolvimento do mesmo.

No segundo capítulo, denominado “Conceptualização do Problema/Projeto”, são explicados os requisitos dos componentes desenvolvidos, bem como a arquitetura sobre o qual esses componentes se baseiam, sendo feitas distinções entre as partes desenvolvidas pelo estagiário e aquelas realizadas por entidades externas ou por outros membros da empresa.

O terceiro capítulo, intitulado “Metodologia de Operacionalização do Trabalho”, são detalhadas as técnicas utilizadas no desenvolvimento do projeto. São abordados tópicos como a metodologia de desenvolvimento, a técnica de criação de testes e a gestão de versões. Além disso, são apresentados pormenores mais aprofundados sobre o que foi desenvolvido em relação a cada componente, como esses componentes foram implementados e como são utilizados.

No penúltimo capítulo, chamado “Discussão dos Resultados”, debate-se o estado final dos componentes no fim do estágio, bem como as dificuldades encontradas ao longo do desenvolvimento.

Por fim, no último capítulo, simplesmente denominado “Conclusão”, é realizada uma reflexão crítica sobre todo o projeto, e explicado os próximos passos no seguimento do desenvolvimento dos componentes.

Fundamentação Teórica

O projeto tem como objetivo a elaboração de um site de vendas B2B, utilizando a plataforma *Magento*, bem como a elaboração das ferramentas necessárias à criação de catálogos de artigos a serem enviados para dito site. Esses artigos são obtidos através do ERP *Macwin*, e são manipulados através do site de operações diárias da empresa. Um dos comportamentos desejados do site de vendas B2B era também o de este poder funcionar como uma aplicação nativa, tendo sido explorada a tecnologia de PWA para esse efeito, porém esta foi descartada por não ser viável.

Nesta secção, irá ser explicado o que é cada um destes conceitos e tecnologias, bem como a sua importância no contexto do problema.

Conceito de B2B

Business-to-business (B2B) é uma situação comercial em que uma empresa efetua uma transação com outra. Normalmente B2B refere-se à transação comercial entre o fabricante e o retalhista, ao invés do B2C, em que o retalhista fornece os bens ao consumidor [2]. No B2B existem empresas de ambos os lados, enquanto no B2C trata-se normalmente de uma empresa e um consumidor.

No caso da *Longratex*, e especificamente com a marca *Patachou*, isto ocorre normalmente quando:

- Se adquirem materiais para o processo de produção de artigos;
- São necessários serviços de outras empresas por razões operacionais.
- Agentes de revenda ou empresas comerciais externas queiram adquirir artigos para revender em lojas terceiras.

O projeto desenvolvido centra-se neste último ponto.

Em termos financeiros, as vendas B2B representam uma enorme percentagem das vendas totais da empresa, pois as entidades envolvidas normalmente adquirem produtos em massa, ao contrário das vendas B2C onde se compram artigos individuais. Por este motivo foi decidido que era necessário criar ferramentas modernas para facilitar este tipo de comércio.

ERP *Macwin*

Um ERP é um sistema de *software* que integra vários processos e funções empresariais de uma organização num sistema unificado e centralizado. O objetivo de um sistema ERP é simplificar e otimizar operações comerciais, aumentar a eficiência e proporcionar visibilidade de dados em tempo real de modo a que seja possível tomar decisões otimizadas. Os sistemas ERP abrangem por norma áreas como finanças, recursos humanos, produção, gestão da cadeias de fornecimento, gestão de relações com os clientes, entre outras [3].

No caso da *Longratex* o ERP utilizado é o *Macwin*, mais especificamente o *software* ERP de gestão têxtil. Este *software* permite gerir variadas etapas específicas do setor fabril em que se encontra a empresa, como por exemplo a gestão de clientes, encomendas, artigos de roupa, etc [4].

Para o desenvolvimento do projeto a função chave do ERP é a gestão de artigos. Diversos membros da empresa gerem a criação, edição e remoção de artigos diretamente no *Macwin*. Estes artigos são guardados numa base de dados SQL, e por defeito não existe maneira de exportar estes dados para outras plataformas. Deste modo, se fosse desejada a introdução de artigos do ERP para um site de vendas B2B isso implicaria a criação manual dos artigos no site, e quaisquer edições e/ou remoções teriam de ser efetuadas em ambos os lados. Por estes motivos entendeu-se que seria fundamental a criação de uma API que pudesse servir de ponte para estas duas plataformas, de modo a simplificar as operações.

Plataforma Comércio Eletrónico *Magento*

De modo a facilitar o comércio B2B, foi decidido por parte da empresa que se deveria proceder ao desenvolvimento de uma nova plataforma de comércio eletrónico.

Uma plataforma de comércio eletrónico é uma solução de *software* que permite às empresas criar, gerir e operar lojas online para vender artigos ou serviços através da Internet. Estas plataformas fornecem uma gama de ferramentas e funcionalidades que permitem às empresas estabelecer uma presença virtual, apresentar os seus produtos, gerir o seu inventário, processar pagamentos e gerir vários aspetos de vendas online e de interações com os clientes [5].

A *Longratex* já possuía uma presença em forma de comércio eletrónico B2B utilizando a plataforma *Magento*, porém este site era já bastante antigo por consequência muitas das ferramentas por ele utilizadas já não eram suportadas. Foi então decidido criar um site completamente novo, utilizando uma versão moderna do *Magento*.

O *Magento* é uma plataforma de comércio eletrónico que fornece às empresas uma solução versátil e personalizável para a criação e gestão de lojas online. É uma plataforma de código aberto, adquirida pela Adobe Inc. em 2018 [6], escrita em PHP, que utiliza elementos da *Framework Zend* [7] e que usa o sistema de gestão de bases de dados relacionais *MariaDB* ou *MySQL*.

Esta plataforma oferece uma gama de funcionalidades e ferramentas que permitem às empresas criar e operar os seus sites de comércio eletrónico, incluindo gestão de catálogos de produtos, funcionalidade de carrinho de compras, processamento de pagamentos, gestão de encomendas e muito mais. O *Magento* é conhecido pela sua flexibilidade, escalabilidade e extensas opções de personalização, o que a torna uma escolha popular para empresas de diversas dimensões [8].

Tecnologia PWA

No início do projeto, era do interesse da empresa que o site de vendas B2B pudesse servir como uma aplicação PWA.

Uma aplicação PWA é um tipo de aplicação web que utiliza tecnologias web modernas para fornecer aos utilizadores uma experiência semelhante a uma aplicação nativa, diretamente através de um *browser*. As PWAs têm como objetivo conjugar os pontos fortes do desenvolvimento de aplicações web e móveis, proporcionando funcionalidades como acesso offline, carregamento rápido e a capacidade de serem instaladas no ecrã inicial de um utilizador, assemelhando-se a uma aplicação convencional [9].

Essa uma tecnologia forneceria maior versatilidade ao site, porém continuava a ser uma tecnologia com adoção limitada, tendo o *browser* Firefox abandonou o suporte do mesmo [10], e os sistemas iOS ainda não suportavam a totalidade das suas funcionalidades. Por estes motivos, foi decidido que por enquanto não seria explorada esta tecnologia no projeto.



CAP II – Conceptualização do Problema/Projeto

Requisitos do Projeto

Requisitos de *software* consistem em descrições das capacidades necessárias para que um sistema ou aplicação possa ser devidamente executado. Estas descrições delineiam as funcionalidades, características, restrições e qualidades que o *software* deve incorporar, de forma a atender às exigências dos seus utilizadores, das partes envolvidas e do projeto.

O projeto compreendia o desenvolvimento de dois componentes de *software* diferentes, por este motivo esta secção encontra-se dividida em duas subsecções, que correspondem aos requisitos da API e aos requisitos do site de vendas B2B.

Requisitos da API

ID	RA0001
Nome do Requisito	Criação de Catálogos por Temporada
Descrição	<p>Deve ser possível ter a capacidade de criação de catálogos de artigos, com base nos itens registados na base de dados do ERP <i>Macwin</i>. Esses artigos têm de ser selecionados de acordo com a temporada específica associada ao catálogo.</p> <p>Por exemplo, se se pretendesse criar um catálogo para a Primavera/Verão 2023, seria necessário indicar que a temporada correspondente é "SS23", sendo apresentados apenas os artigos dessa temporada.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 1 - Requisitos da API - Criação de Catálogos por Temporada.

ID	RA0002
Nome do Requisito	Remoção de Catálogos Independentes dos Artigos
Descrição	<p>Pretende-se ter a capacidade de remover catálogos da base de dados sem que os artigos associados sejam eliminados simultaneamente.</p> <p>Essa autonomia é crucial devido à eventualidade de dois catálogos distintos partilharem os mesmos artigos. Consequentemente, ao eliminar um catálogo, não se pretende que o outro perdesse os artigos em comum.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 2 - Requisitos da API - Remoção de Catálogos Independentes dos Artigos.

ID	RA0003
Nome do Requisito	Edição de Catálogos
Descrição	<p>Possuir a capacidade de editar determinados dados do catálogo, nomeadamente o nome e a descrição do mesmo. Isso assegura que, em caso de erro de escrita durante a sua criação, não seja necessário começar um novo catálogo de raiz.</p>
Prioridade	Moderado
Estado	Implementado com Sucesso

Tabela 3 - Requisitos da API - Edição de Catálogos.

ID	RA0004
Nome do Requisito	Visualização de Artigos em Função de Filtros
Descrição	<p>Quando existir a necessidade de visualizar os artigos de um catálogo, deve ser possível aplicar determinados filtros para limitar os artigos visíveis aos elementos essenciais para as operações pretendidas.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 4 - Requisitos da API - Visualização de Artigos em Função de Filtros.

ID	RA0005
Nome do Requisito	Edição de Artigos Individuais ou em Massa
Descrição	A API deve possibilitar a edição de artigos, quer de forma individual quer em massa. Dessa forma, se surgir a necessidade de modificar informações relativas a um artigo, basta efetuar a alteração através da API, que, por sua vez, procede à atualização dos dados no ERP <i>Macwin</i> e nos sites de vendas.
Prioridade	Moderada
Estado	Implementado com Sucesso

Tabela 5 - Requisitos da API - Edição de Artigos Individuais ou em Massa.

ID	RA0006
Nome do Requisito	Seleção ou Desmarcação de Artigos para Sincronização
Descrição	Possibilidade de selecionar ou desmarcar os artigos de um catálogo que se pretende sincronizar ou não com os sites de vendas. Isso facilita uma filtragem mais eficaz do que deve ou não ser enviado para os sites de vendas, sem a necessidade de modificar diretamente os catálogos.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 6 - Requisitos da API - Seleção ou Desmarcação de Artigos para Sincronização.

ID	RA0007
Nome do Requisito	Remoção de Artigos Individuais do Catálogo
Descrição	<p>Possibilitar ao utilizador a remoção de artigos de um catálogo, sem que esses artigos sejam excluídos da base de dados. Isto é especialmente importante devido há possibilidade de dois catálogos distintos partilharem os mesmos artigos.</p> <p>O sistema <i>Macwin</i> não oferece a funcionalidade de eliminar artigos, e, portanto, por meio da API, é alcançado um maior controlo sobre o catálogo.</p>
Prioridade	Moderada
Estado	Implementado com Sucesso

Tabela 7 - Requisitos da API - Remoção de Artigos Individuais do Catálogo.

ID	RA0008
Nome do Requisito	Validação de Artigos Consoante Requisitos Específicos
Descrição	<p>Os artigos dos catálogos têm de possuir determinadas características, tais como um número específico de imagens e a necessidade de ter determinadas traduções, entre outros.</p> <p>Caso os artigos não cumpram esses requisitos, a sincronização com os sites de vendas não é permitida.</p> <p>Na altura da implementação do requisito, os requisitos dos catálogos eram estáticos; no entanto, existia interesse em permitir que os requisitos de validação fossem definidos através da API no futuro.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 8 - Requisitos da API - Validação de Artigos Consoante Requisitos Específicos.

ID	RA0009
Nome do Requisito	Sincronização de Artigos para Sites de Vendas
Descrição	Após a gestão do catálogo, quando os artigos selecionados estiverem preparados para ser exibidos nos sites de vendas, pretende-se possibilitar o envio dos mesmos através de filas de espera. Isso garante que, caso artigos idênticos sejam enviados por diferentes catálogos, estes não sejam inseridos múltiplas vezes nos sites.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 9 - Requisitos da API - Sincronização de Artigos para Sites de Vendas.

ID	RA0010
Nome do Requisito	Remoção de Filas de Sincronização
Descrição	Possuir a capacidade de remover filas de espera quando estas não estejam em funcionamento.
Prioridade	Moderada
Estado	Implementado com Sucesso

Tabela 10 - Requisitos da API - Remoção de Filas de Sincronização.

ID	RA0011
Nome do Requisito	Criação de Relatórios de Erros de Filas de Sincronização
Descrição	Quando uma fila de sincronização para de maneira inesperada, deve ser gerado automaticamente um relatório de erro com informações acerca do problema que tinha ocorrido.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 11 - Requisitos da API - Criação de Relatórios de Erros de Filas de Sincronização.

ID	RA0012
Nome do Requisito	Parametrização de Atributos da Aplicação <i>Macwin</i> para Atributos de Sites <i>Magento</i>
Descrição	<p>Determinados atributos dos artigos requeriam <i>Ids</i> para serem enviados para os sites <i>Magento</i>. Para garantir o correto envio destes atributos, é necessário estabelecer uma ligação entre os <i>Ids</i> dos atributos no ERP <i>Macwin</i> e os <i>Ids</i> correspondentes nos atributos dos sites.</p> <p>Por exemplo, a cor "Azul" é identificada no <i>Macwin</i> com o <i>Id</i> 10, mas no site do <i>Magento</i>, o <i>Id</i> correspondente a essa cor é o 50. Pretende-se possibilitar a indicação de que o atributo "Cor" com <i>Id</i> 10 no <i>Macwin</i> corresponde ao <i>Id</i> 50 do atributo "Cor" no <i>Magento</i>.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 12 - Requisitos da API - Parametrização de Atributos da Aplicação *Macwin* para Atributos de Sites *Magento*.

ID	RA0013
Nome do Requisito	Atualização de Dados de Artigos
Descrição	<p>Devem ser disponibilizadas ferramentas para atualizar os dados dos artigos, tanto para um catálogo completo como para artigos individuais. Isso garante que as atualizações efetuadas nos artigos no ERP são refletidas no site de operações diárias.</p>
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 13 - Requisitos da API - Atualização de Dados de Artigos.

Requisitos do Site B2B

ID	RS0001
Nome do Requisito	Implementação do Tema Utero
Descrição	Deve-se implementar o tema Utero [11] de modo a modificar o aspeto e funcionalidade do site.
Prioridade	Alta
Estado	<u>Não implementado</u> . O tema não se encontrava atualizado para a versão corrente do <i>Magento</i> , e dado que a última atualização (na data em que este texto foi escrito) ocorreu em Setembro de 2022, optou-se por se utilizar outro temo, intitulado Supro [12].

Tabela 14 - Requisitos do Site B2B - Implementação do Tema Utero.

ID	RS0002
Nome do Requisito	Traduções em Inglês e Português
Descrição	O site deve encontrar-se disponível, para certas regiões, em Português e Inglês.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 15 - Requisitos do Site B2B - Traduções em Inglês e Português.

ID	RS0003
Nome do Requisito	Capacidade de Selecionar o País de Acesso
Descrição	O site serve para vendas a agentes de retalho, sendo que alguns destes adquirem artigos em nome de terceiros. Isso implicava que determinados utilizadores precisam aceder às versões dos sites do país de origem dos seus clientes e, portanto, têm de ser capazes de selecionar o seu país.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 16 - Requisitos do Site B2B - Capacidade de Selecionar o País de Acesso.

ID	RS0004
Nome do Requisito	Obrigação de Login antes de Aceder ao Site
Descrição	O acesso ao site não deve ser público, sendo reservado apenas a clientes oficiais de B2B. Nesse sentido, achou-se que era essencial limitar o acesso às páginas até que o utilizador estivesse devidamente registado.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 17 - Requisitos do Site B2B - Obrigação de Login antes de Aceder ao Site.

ID	RS0005
Nome do Requisito	Compra de Artigos em Matriz
Descrição	O comércio B2B implica a aquisição de produtos em massa. De modo a simplificar a venda de produtos desta forma, entendeu-se que os produtos deveriam ser dispostos em matriz, diferenciadas pelos seus tamanhos e cores, de modo a que se possa comprar várias variantes do mesmo artigo ao mesmo tempo.
Prioridade	Alta
Estado	Implementado com Sucesso

Tabela 18 - Requisitos do Site B2B - Compra de Artigos em Matriz.

Arquitetura da Solução

O projeto desenvolvido para a empresa *Longratex* exige a interação entre três componentes específicos: o ERP *Macwin*, o site de operações diárias da *Longratex* e o novo site de vendas B2B. Neste capítulo irá ser explorada a arquitetura tecnológica de cada um dos componentes, conforme representado na Figura 1, com ênfase nestes últimos dois. Irá também ser explorada a arquitetura conceptual da API desenvolvida.

Arquitetura Tecnológica

Pelo começo do desenvolvimento da solução, muitas das tecnologias e ferramentas a utilizar já se encontravam em uso ativo, não tendo sido alteradas ao longo do progresso do mesmo.

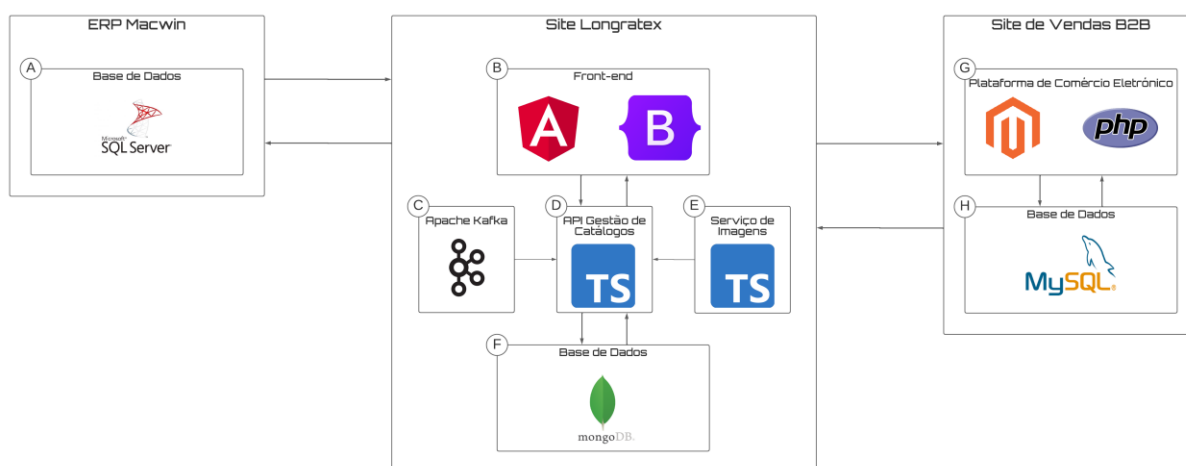


Figura 1 - Exemplificação Visual da Arquitetura do Projeto.

Um dos objetivos do projeto incidia na aquisição dos artigos que se encontram criados no ERP da empresa, de modo a se conseguir proceder à criação dos catálogos de vendas. Para esse propósito, estabelece-se uma ligação direta com a base de dados do ERP *Macwin*, que armazena os ditos artigos. O *software Macwin* utiliza o sistema de gestão de base de dados "*SQL Server*" (Figura 1A). A gestão do *software* encontra-se a cargo de uma empresa terceira denominada "*MacWin Tek, Lda.*". Como os dados retidos pelo *software* são vitais para as operações diárias da empresa, o estagiário não teve contato direto com eles. Em vez disso, o orientador do estágio da empresa forneceu as *queries*

necessárias para obter os artigos e os seus atributos da base de dados, de modo a que o estagiário pudesse manipular essas informações.

Os dados obtidos do *software Macwin* são depois utilizados no site de operações diárias da *Longratex* através de uma API criada para esse propósito. O desenvolvimento dessa API iniciou-se com a adaptação de um projeto já existente para uma outra plataforma de comércio eletrónico. A *Longratex* possui uma ferramenta de criação de catálogos para a empresa “El Corte Inglés”, porém esta empresa utiliza a plataforma *Mirakl*, cujos requisitos para catálogos e artigos diferiam dos que seriam usados na plataforma *Magento*. Era necessário então adaptar esse projeto de modo a que este continuasse a ser compatível com a plataforma *Mirakl* e com a plataforma *Magento*. Dado que o projeto a ser adaptado foi desenvolvido em *Typescript* (Figura 1D), e a sua base de dados foi criada em *MongoDB* (Figura 1F), essas mesmas ferramentas foram as utilizadas para o desenvolvimento.

A API utiliza o *Apache Kafka* (Figura 1C) para monitorizar alterações aos artigos guardados pelo ERP *Macwin*. Caso sejam identificadas alterações a artigos, prossegue-se à sua atualização automaticamente. É também utilizado um serviço de Imagens (Figura 1E) para atualizar as imagens dos artigos conforme estas sejam alteradas, e modificar os URLs dos mesmos, serviço este que foi desenvolvido em *Typescript*.

A API é depois utilizada pelo *front-end* (Figura 1B) do site de operações diárias de modo a poder utilizar as novas funcionalidades. Este *front-end* foi desenvolvido por um outro membro da equipa utilizando a *framework Angular* para controlar o seu comportamento e a *framework Bootstrap* para ajudar a definir o seu aspeto.

Os catálogos de vendas prosseguem depois para o site de vendas B2B. No desenvolvimento do site, a empresa manifestou interesse em utilizar a plataforma *Magento* (Figura 1G). Uma vez que esta plataforma é escrita em *PHP*, as funcionalidades do site foram criadas nessa mesma linguagem. O *Magento* permite a escolha entre o *MariaDB* e *MySQL* como sistema de gestão de base de dados relacional. Dado que o estagiário já tinha familiaridade com o *MySQL*, foi decidido utilizar esse sistema (Figura 1H).

É de destacar que o estagiário esteve ao controlo do desenvolvimento dos seguintes elementos: o desenvolvimento da API encarregue pela Gestão de Catálogos e da base de dados do site de operações diárias da empresa (Figura 1D e Figura 1F); desenvolvimento do site de vendas B2B (Figura 1G e Figura 1H). Já a integração com o *Apache Kafka*

(Figura 1C) e com o Serviço de Imagens (Figura 1E) foram desenvolvidas de forma colaborativa.

Arquitetura Conceptual da API

Um dos principais componentes desenvolvidos para o projeto era uma API que pudesse comunicar com o ERP utilizado pela empresa de modo a que fosse possível criar catálogos de vendas dos artigos por ele guardados. Para esse fim, adaptou-se um projeto existente de criação de catálogos para a plataforma *Mirakl*. Na data do fim do projeto, essa API encontrava-se estruturada da forma representada na Figura 2.

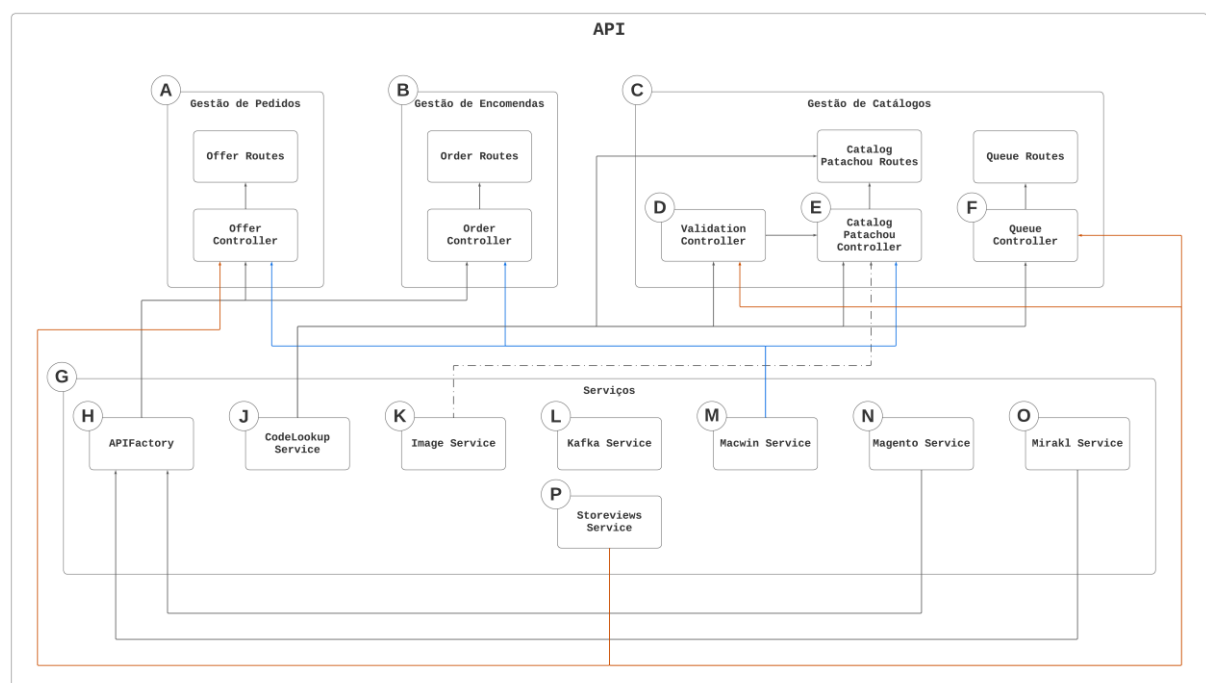


Figura 2 - Representação visual da arquitetura conceptual da API de gestão de catálogos.

A componente de Gestão de Pedidos (Figura 2, Figura 3) serve principalmente para gerir os *stocks* dos artigos consoante os pedidos dos clientes. É responsável por atualizar os *stocks* nas bases de dados e altera estados de pedidos. Já a componente de Gestão de Encomendas (Figura 2B, Figura 3) como o nome indica lida com as encomendas, guarda encomendas no ERP, gera etiquetas de envio, altera estados de encomendas e lida com o seu processamento.

Ambos a Gestão de Pedidos (Figura 2A, Figura 3) e Gestão de Encomendas (Figura 2B, Figura 3) já se encontravam desenvolvidos na altura do início do projeto, porém ambos encontravam-se ligados diretamente à API da plataforma *Miraki*. Foi substituída qualquer referência direta a essa API e passou-se a utilizar a “*APIFactory*” (Figura 2H) para indicar que API é que deve ser usada no momento de execução.

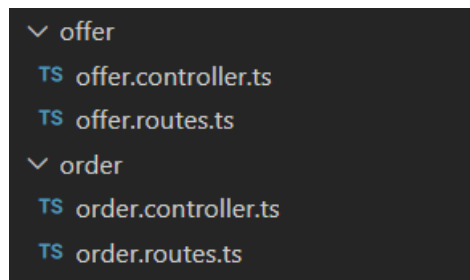


Figura 3 - Rotas e Controladores relacionados com a Gestão de Pedidos e Gestão de Encomendas.

A Gestão de Catálogos (Figura 2C, Figura 4) é composta pelas rotas e controladores necessárias à criação, remoção, manutenção e envio de catálogos. O ficheiro “*Validation Controller*” (Figura 2D) contém os métodos necessários para a validação dos artigos consoante os requisitos obrigatórios do catálogo onde eles se encontram. Permite a validação de um catálogo inteiro, de um artigo individual com todas as suas configurações de tamanhos e cores, ou de um artigo individual com um tamanho e cor específico. Cria também os relatórios de erros de validação de modo a informar o utilizador final sobre quais as correções que são necessárias efetuar. Utiliza o “*CodeLookup Service*” (Figura 2J) para validar os códigos dos atributos da plataforma *Magento* em comparação com os códigos dos atributos do ERP. Utiliza também o “*Storeviews Service*” (Figura 2P) para verificar os preços que os artigos são obrigados a ter consoante as vistas da plataforma *Magento*.

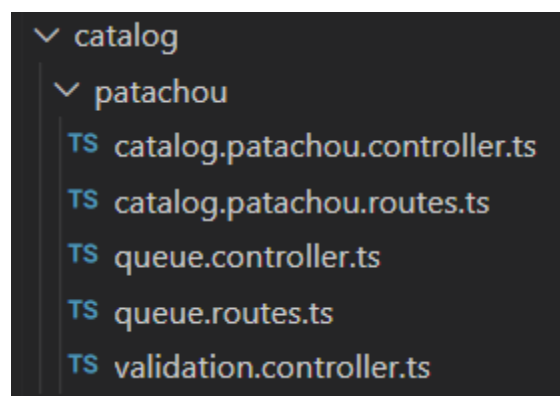


Figura 4 - Rotas e Controladores relacionados com a Gestão de Catálogos.

O ficheiro “*Catalog Patachou Controller*” (Figura 2E) contém os métodos necessários para a gestão dos catálogos de vendas. Permite realizar as seguintes operações:

- Criar, eliminar e editar catálogos de vendas;
- Atribuir novos requisitos a catálogos;
- Editar artigos individualmente ou em massa;
- Eliminar artigos de um catálogo de forma individual;
- Selecionar ou desseleccionar artigos e as suas configurações, nomeadamente os tamanhos e cores desejadas;
- Recarregar os dados de artigos de forma individual ou de todos os artigos de um catálogo;

O “*Catalog Patachou Controller*” utiliza também outros ficheiros da seguinte forma:

- O “*Macwin Service*” (Figura 2M) é usado para obter os artigos do ERP *Macwin*, e para editar os artigos desejados.
- O “*Validation Controller*” (Figura 2D) é utilizado para validar os artigos quando estes são criados ou recarregados de alguma forma, ou quando artigos são editados.
- O “*CodeLookup Service*” (Figura 2J) é utilizado para calcular o *score* de um produto. Os produtos têm *score* positivo caso todas as suas configurações sejam vendidos de forma repartida, sem que haja uma discrepância muito grande entre *stocks* de configurações diferentes.
- O “*Image Service*” (Figura 2K) é usado para obter os URLs das imagens de um artigo.

O “*Catalog Patachou Controller*” é depois usado pelo “*Catalog Patachou Routes*” de forma a proporcionar ao *Front-end* do site as ferramentas necessárias à gestão dos catálogos.

A Gestão de Catálogos contém também o “*Queue Controller*” (Figura 2F), que se encontra encarregue de gerir as filas de envio de catálogos para os sites de vendas da plataforma *Magento*. Possui os métodos necessários para criar e eliminar filas, bem como alterar o estado de filas para “parado”, “em operação” ou “em espera”. Este controlador está encarregue de correr as filas. Dez minutos após o ter corrido a última fila de sincronização, verifica-se se existem novas filas de sincronização para correr. Caso alguma seja identificada, essa fila percorre a ordem de operações seguinte (Figura 5):

1. Obtém-se a lista de artigos que pertencem a um catálogo;
2. Começa-se a tratar das configurações de um artigo individualmente;
 - 2.1. Caso uma configuração de um artigo ainda não se encontre na plataforma *Magento*, cria-se um objeto que contém as características dessa configuração que irá ser utilizado para enviar um pedido à *REST API* da plataforma de modo a poder ser lá introduzido;
 - 2.2. Se uma configuração de um artigo se encontrar na plataforma, então primeiro verifica-se se os dados do artigo lá contidos coincidem com os dados que este tem na base de dados da API de Gestão de Catálogos;
 - 2.2.1. Se os dados não coincidirem, então cria-se um objeto com os dados atualizados, e utiliza-se esse objeto para enviar um pedido à *REST API* da plataforma de modo a atualizar os dados no site;
 - 2.2.2. Se os dados coincidirem então não se modifica nada e passa-se para a configuração seguinte;
3. Repete-se o passo 2 até todas as configurações do artigo que estiverem selecionadas para envio se encontrarem no site de vendas;
4. Começa-se a tratar do envio do artigo geral;
 - 4.1. Caso um artigo geral (sem dados de tamanho ou cor) ainda não se encontre na plataforma *Magento*, cria-se um objeto que contém as características desse artigo que irá ser utilizado para enviar um pedido à *REST API* da plataforma de modo a poder ser lá introduzido;
 - 4.2. Se um artigo geral se encontrar na plataforma, então verifica-se se os dados do artigo lá contidos coincidem com os dados que este tem na base de dados da API;
 - 4.2.1. Se os dados não coincidirem, então cria-se um objeto com os dados atualizados, e utiliza-se esse objeto para enviar um pedido à *REST API* da plataforma de modo a atualizar os dados no site;

4.2.2. Se os dados coincidirem então não se modifica nada e passa-se para o artigo seguinte;

5. Repete-se os passos 2 a 4 até todos os artigos e as suas configurações se encontrarem no site de vendas;
6. No final a fila de sincronização passa para o estado “parado” e, caso haja outras filas em espera, começa-se a tratar de uma nova.

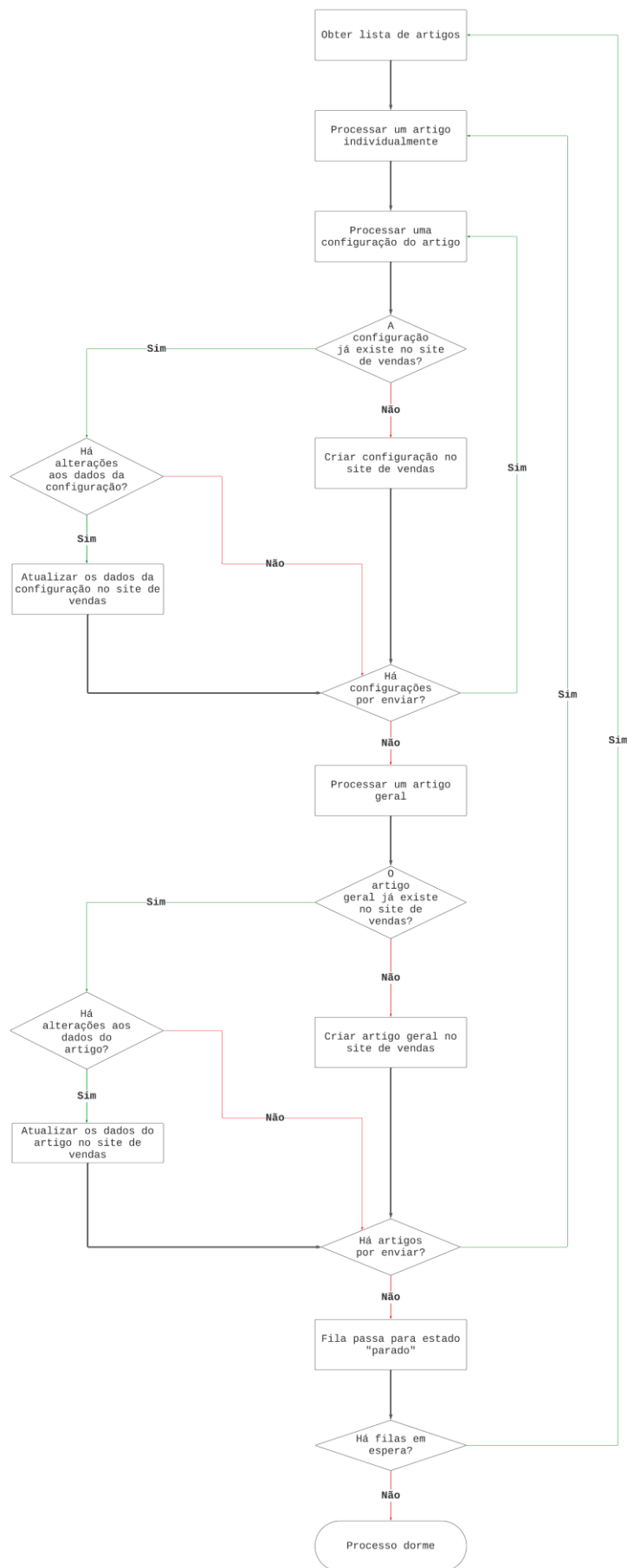


Figura 5 - Exemplificação visual das operações de execução das filas de sincronização.

O *Queue Controller* (Figura 2F) utiliza o *CodeLookup Service* (Figura 2J) para obter os códigos dos atributos da plataforma *Magento* que são compatíveis com os códigos dos atributos do ERP que estejam associados a um artigo. Utiliza também o *Storeviews Service* (Figura 2P) para enviar as versões corretas dos preços aos vistas corretas do site de vendas, e para obter as versões corretas URLs que devem ser utilizados para o envio de pedidos à *REST API* da plataforma *Magento* dependendo das vistas.

O *Queue Controller* é depois usado pelo *Queue Routes* de forma a proporcionar ao *Front-end* do site as ferramentas necessárias à gestão das filas de sincronização dos catálogos.

Os controladores e as rotas utilizam também alguns ficheiros aos quais chamamos *Services* (Figura 2G), de modo a proporcionar mais ferramentas aos anteriormente mencionados. Os *Services* disponíveis são os seguintes (Figura 6):

- A *APIFactory* (Figura 2H) é utilizada para identificar que API deve ser usada pelos controladores. O service devolve uma instância do *Mirakl Service* (Figura 2O) ou do *Magento Service* (Figura 2N) dependendo do tipo de serviço que se quer prestar.
- O *CodeLookup Service* (Figura 2J) serve para obter os dados de certos atributos do *Magento* e do ERP *Macwin*, estabelecer elos de ligação entre os dois, calcular o *score* de produtos. Alguns dos seus métodos são utilizados pelo *Catalog Patachou Routes* de modo a ser possível criar as ligações no site de operações diárias, bem como ver os dados dos atributos;
- O *Image Service* (Figura 2K) obtém os URLs das imagens dos produtos;
- O *Kafka Service* (Figura 2L) estabelece uma conexão com a ferramenta *Apache Kafka*. É através dessa ferramenta que verificamos se um artigo sofreu alguma alteração no lado do ERP, e caso alguma alteração seja identificada, se altera também os dados do artigo no lado da base de dados da API.
- O *Macwin Service* (Figura 2M) estabelece uma ligação com o ERP *Macwin*. Através deste *Service*, obtemos os artigos guardados no ERP, os dados dos atributos que um artigo pode conter, as encomendas e pedidos, e permite-nos editar os dados de artigos.
- O *Mirakl Service* (Figura 2O) e o *Magento Service* (Figura 2N) contém os mesmos tipos de métodos, e servem principalmente para gerir as ofertas e encomendas.

- O “Storeviews Service” (Figura 2P) serve para obter os dados das vistas dos sites de vendas da plataforma *Magento*, tais como os URLs de cada vista, ou o tipo de preço associado a cada um.

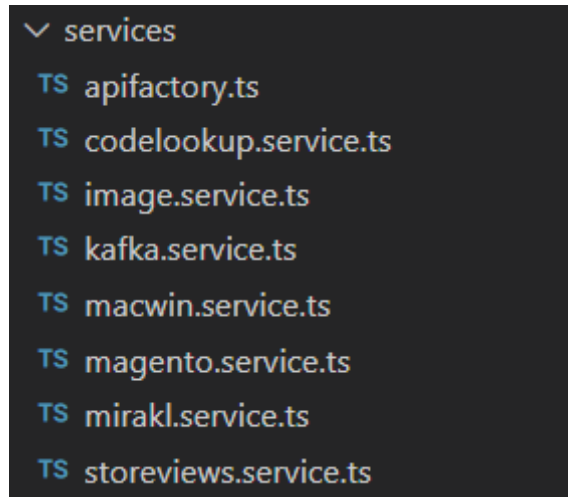


Figura 6 - Lista de "Services".



CAP III – Metodologia de Operacionalização do Trabalho

Processo e Metodologia de Trabalho

Nesta secção, são abordadas as metodologias utilizadas para o desenvolvimento do projeto. Mais concretamente, irá falar-se sobre a metodologia SCRUM, que foi adotada para o desenvolvimento do projeto. Além disso, será detalhado o desenvolvimento dos testes unitários e a utilização do *software Lint*. Também será discutido o sistema de controlo de versões *BitBucket*, bem como as ferramentas de suporte a CI/CD, *Docker* e *Kubernetes*.

Metodologia SCRUM

No desenvolvimento de *software*, as metodologias representam os processos estruturados que abrangem o planeamento, execução e gestão de um projeto. Essas metodologias oferecem uma forma estruturada para as equipas de desenvolvimento colaborarem, definirem objetivos de projeto, gerirem tarefas, alocarem recursos e entregarem produtos finais que cumpram os requisitos e padrões de qualidade desejados.

A metodologia de desenvolvimento de *software* utilizada pela equipa de desenvolvimento já se encontrava em vigor na empresa no começo do projeto, metodologia esta que era a metodologia Ágil SCRUM.

A metodologia Ágil representa uma abordagem ao desenvolvimento de *software* e à gestão de projetos que privilegia a flexibilidade, a colaboração e o no cliente. Esta metodologia segue uma estrutura iterativa, priorizando a entrega de *software* funcional em ciclos mais pequenos e incrementais, geralmente designados como *sprints*, em vez de tentar entregar um produto completo de uma só vez [13].

Cada *sprint* era normalmente executado dentro de um período de uma a três semanas. No final de cada *sprint*, a equipa participava numa reunião de planeamento do *sprint* seguinte, onde se seleccionava um conjunto de itens do *backlog* do projeto (Figura 7) que deveriam ser concluídos dentro da data estipulada. No caso de um *sprint* ultrapassar a duração considerada "normal", realizava-se uma reunião menos formal para avaliar o progresso do *sprint*, possíveis contratempos e aspetos a melhorar.

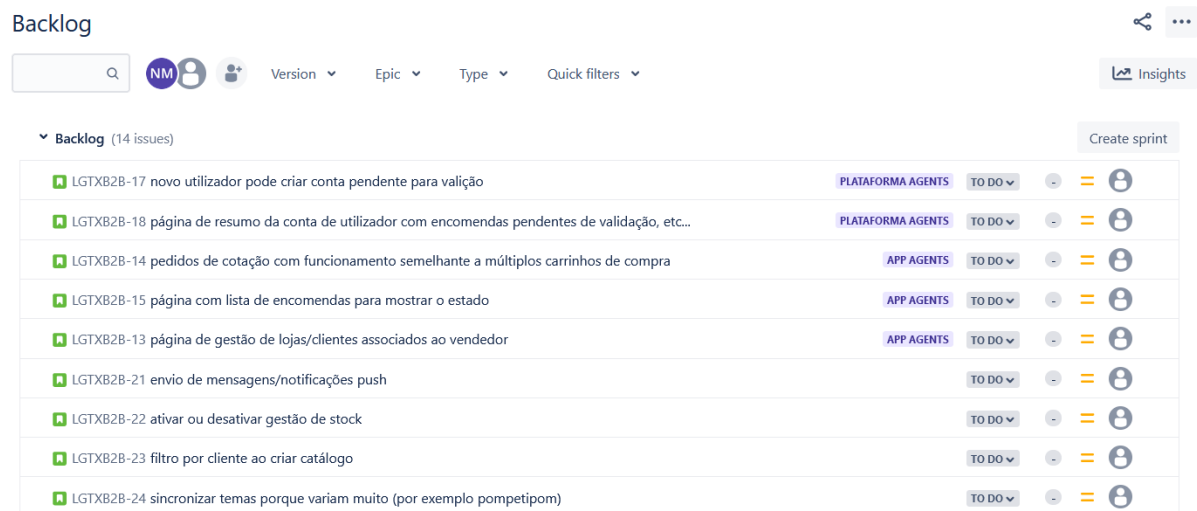


Figura 7 - Backlog do projeto na ferramenta Jira. É a partir destes pontos que se planeiam o que deve ser feito para o *sprint*.

Para permitir que a equipa monitorasse a evolução do projeto, foi adotada a ferramenta *Jira* [14]. O *Jira* é uma ferramenta de gestão de projetos e acompanhamento de problemas, desenvolvida pela *Atlassian*. Através do *Jira* realizava-se a gestão do *backlog* do projeto (Figura 7), e o planeamento dos *Sprints* (Figura 8).

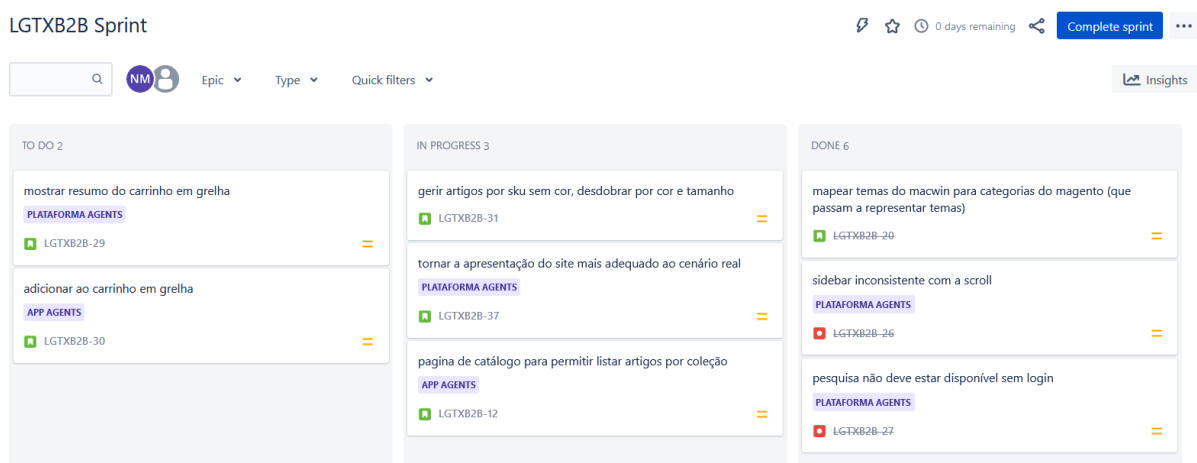


Figura 8 - Sprint do projeto na ferramenta Jira. Representam as tarefas a serem realizadas com data de início e fim..

A utilização da metodologia SCRUM foi uma mais-valia no desenvolvimento do projeto. A subdivisão do trabalho em *Sprints* ajudou a manter o foco nos requisitos prioritários do projeto, evitando distrações com requisitos menos significativos. Para além disso, as reuniões de final de *sprint* permitiram uma avaliação dos métodos e hábitos de desenvolvimento, de modo que esses pudessem evoluir de forma positiva.

Testes Unitários e Software Lint

Para assegurar a qualidade do *software* criado, os testes unitários aos componentes a desenvolver deveriam ser elaborados em primeiro lugar. Além disso, todo o código desenvolvido deveria ser analisado através da ferramenta *Lint*.

Os testes unitários representam um aspeto essencial no processo de desenvolvimento de *software*. Estes são aplicados de modo a avaliar isoladamente componentes individuais ou unidades de uma aplicação de *software*. Geralmente, essas unidades são as porções mais pequenas do código que podem ser examinadas, tais como funções, métodos ou classes. O principal objetivo dos testes unitários consiste em verificar se cada unidade de código opera corretamente e gera o resultado esperado para um conjunto específico de entradas (Figura 9).

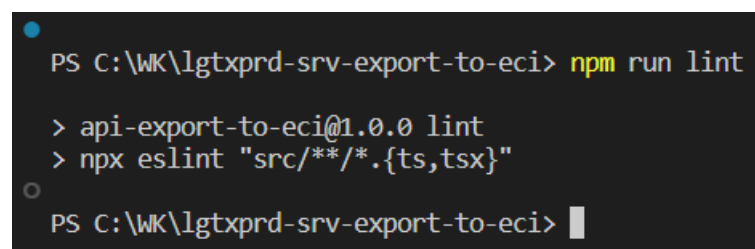
```
test('throw ResourceNotFound() if catalog is not found', async () => {
  let error: any = null;
  try {
    await catalogB2CController.assignRequirementsToCatalog("63eb751623ec6d246b7db00b", {} as IRequiredValidations);
  } catch (e: any) {
    error = e;
  }
  expect(error).not.toBeNull();
  expect(error?.message).toEqual("Catalogo com requisitos a alterar não foi encontrado");
});
```

Figura 9 - Exemplo de um teste unitário. O objetivo é verificar se, ao alterar os requisitos de um catálogo não existente, o erro *ResourceNotFound* é lançado.

Os testes unitários constituíam sempre a primeira etapa no desenvolvimento do código do projeto. Antes de escrever qualquer função, por exemplo, era realizado uma reflexão sobre o seu propósito, o resultado esperado tanto em caso de funcionamento adequado como em caso de falha, e só depois se procedia à escrita da função. Desse modo garantia-se que a função se comportaria conforme o planeado, evitando comportamentos anómalos. Naturalmente, isso implicava que, ao longo do desenvolvimento das funções, caso fosse feito qualquer ajuste no seu comportamento para melhor se alinhar com o desejado, primeiro os testes deveriam ser adaptados e, em seguida, as funções seriam modificadas.

No caso do *software Lint*, refere-se a ferramentas e processos que analisam o código desenvolvido de modo a detetar e assinalar possíveis problemas, violações das normas de boas práticas e de sintaxe, bem como outras questões não diretamente relacionadas com a sintaxe. Essas ferramentas são especialmente úteis em projetos colaborativos nos quais múltiplos programadores com diferentes estilos de programação trabalham em conjunto. A prática de *linting* ajuda a manter um estilo de código coerente em toda a base do projeto, o que torna o código mais legível e mais fácil de manter.

Antes de sincronizar quaisquer alterações ao código do projeto com o repositório do *BitBucket*, efetua-se uma verificação *Lint* (como apresentado na Figura 10) e, caso seja identificado algum erro pelo *software*, prossegue-se à sua correção e retoma-se a verificação dos testes e *linting*.

A screenshot of a Windows command prompt window with a dark background. The text is white. It shows a series of commands being entered and executed. The first line is 'PS C:\WK\lgtxprd-srv-export-to-eci> npm run lint'. The second line is '> api-export-to-eci@1.0.0 lint'. The third line is '> npx eslint "src/**/*.{ts,tsx}"'. The fourth line shows the prompt again: 'PS C:\WK\lgtxprd-srv-export-to-eci>'. There is a small blue dot at the start of the first line and a small grey circle at the start of the fourth line.

```
PS C:\WK\lgtxprd-srv-export-to-eci> npm run lint

> api-export-to-eci@1.0.0 lint
> npx eslint "src/**/*.{ts,tsx}"

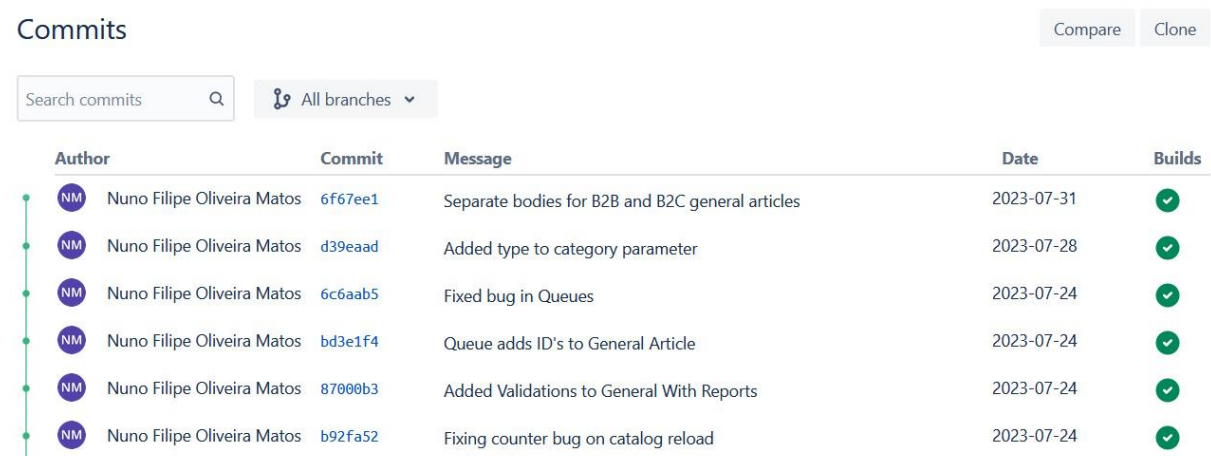
PS C:\WK\lgtxprd-srv-export-to-eci>
```

Figura 10 - Exemplo de uma verificação do código com o *software Lint*. Neste caso, a verificação não apresentou quaisquer falhas.

Controlo de Versões e Ferramentas de CI/CD

O controlo de versões consiste num sistema e num processo utilizado no desenvolvimento de *software* para monitorizar e gerir as modificações feitas no código-fonte e em outros ficheiros ao longo do tempo. O seu principal objetivo é facilitar a colaboração entre programadores, manter um histórico das alterações efetuadas ao código e simplificar a gestão de diferentes versões de um projeto de *software*. Os sistemas de controlo de versões fornecem uma forma estruturada de acompanhar as modificações, reverter para estados anteriores e trabalhar simultaneamente no código, minimizando conflitos.

A ferramenta utilizada para gerir o controlo de versões foi o *BitBucket*. O *Bitbucket* é uma plataforma baseada na Web destinada ao controlo de versões, gestão de código-fonte e colaboração, destinada principalmente a equipas de desenvolvimento de *software*. Esta fornece funcionalidades para alojar e gerir repositórios *Git*, permitindo que os programadores trabalhem de forma colaborativa no código, acompanhem as modificações (Figura 11) e mantenham os projetos de maneira mais eficiente.



The screenshot shows the 'Commits' section of a BitBucket repository. At the top right are 'Compare' and 'Clone' buttons. Below them is a search bar labeled 'Search commits' and a dropdown menu for 'All branches'. The main content is a table of commits with columns: Author, Commit, Message, Date, and Builds. A vertical timeline on the left shows the commit history. All commits are by 'Nuno Filipe Oliveira Matos' and have a green checkmark in the 'Builds' column.

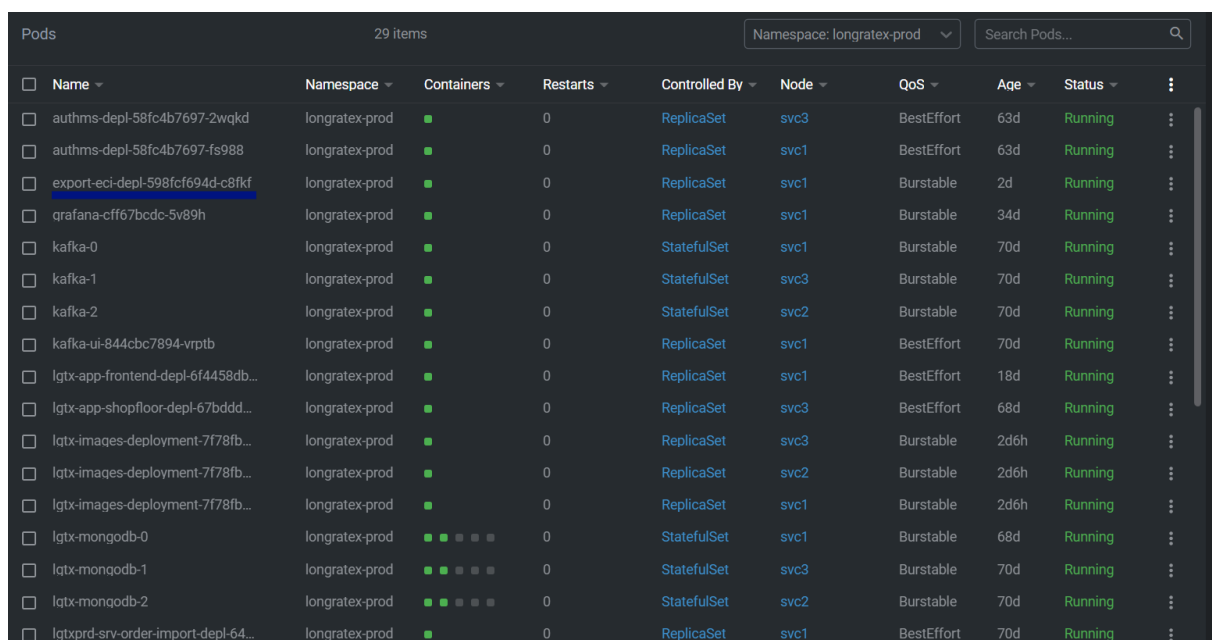
Author	Commit	Message	Date	Builds
Nuno Filipe Oliveira Matos	6f67ee1	Separate bodies for B2B and B2C general articles	2023-07-31	✓
Nuno Filipe Oliveira Matos	d39eaad	Added type to category parameter	2023-07-28	✓
Nuno Filipe Oliveira Matos	6c6aab5	Fixed bug in Queues	2023-07-24	✓
Nuno Filipe Oliveira Matos	bd3e1f4	Queue adds ID's to General Article	2023-07-24	✓
Nuno Filipe Oliveira Matos	87000b3	Added Validations to General With Reports	2023-07-24	✓
Nuno Filipe Oliveira Matos	b92fa52	Fixing counter bug on catalog reload	2023-07-24	✓

Figura 11 - Exemplo de algumas alterações efetuadas ao código presente no repositório BitBucket.

O *BitBucket* também permitiu à equipa de desenvolvimento assegurar que o projeto aderisse às práticas de CI/CD, através da interação com outras ferramentas, sendo o *Docker* uma dessas ferramentas neste caso. As práticas e os processos de CI/CD visam automatizar e simplificar a criação, teste e implementação de modificações do código. Estas práticas têm como propósito aprimorar a eficiência e a fiabilidade no desenvolvimento de *software*, automatizando tarefas repetitivas, reduzindo intervenções manuais e garantindo que as alterações de código sejam integradas e implementadas no ambiente de produção da maneira mais simples possível.

Antes de explicar os passos relacionados com a integração das modificações no código-fonte, é essencial compreender a utilidade das ferramentas *Docker* e *Kubernetes*:

- O *Docker* consiste numa plataforma e num conjunto de ferramentas que possibilitam a criação, implementação e administração de aplicações por meio de contentores. Os contentores são unidades leves e portáteis que encapsulam uma aplicação juntamente com as suas dependências, permitindo a sua execução de forma uniforme em diversos ambientes. O *Docker* estabelece um método padrão para empacotar, distribuir e executar aplicações, simplificando o desenvolvimento, teste e implementação em vários ambientes, desde máquinas locais de desenvolvimento até servidores de produção.
- O *Kubernetes* é uma plataforma que ajuda na automatização da implementação, escalabilidade e gestão de aplicações em contentores. Enquanto o *Docker* fornece a tecnologia para criar e gerir contentores, o *Kubernetes* fornece uma estrutura que lida com a gestão de grupos de contentores e aborda tarefas complexas relacionadas com a implementação e escalabilidade de aplicações. Para o desenvolvimento foi usada a ferramenta *Lens* (Figura 12), que é um IDE que permite conectar e gerir múltiplos contentores.



Name	Namespace	Containers	Restarts	Controlled By	Node	QoS	Age	Status
authms-depl-58fc4b7697-2wqkd	longratex-prod	1	0	ReplicaSet	svc3	BestEffort	63d	Running
authms-depl-58fc4b7697-fs988	longratex-prod	1	0	ReplicaSet	svc1	BestEffort	63d	Running
export-eci-depl-598fcf694d-c8fkf	longratex-prod	1	0	ReplicaSet	svc1	Burstable	2d	Running
grafana-cff67bcd-5v89h	longratex-prod	1	0	ReplicaSet	svc1	Burstable	34d	Running
kafka-0	longratex-prod	1	0	StatefulSet	svc1	Burstable	70d	Running
kafka-1	longratex-prod	1	0	StatefulSet	svc3	Burstable	70d	Running
kafka-2	longratex-prod	1	0	StatefulSet	svc2	Burstable	70d	Running
kafka-ui-844cbc7894-vrptb	longratex-prod	1	0	ReplicaSet	svc1	BestEffort	70d	Running
lgtx-app-frontend-depl-6f4458db...	longratex-prod	1	0	ReplicaSet	svc1	BestEffort	18d	Running
lgtx-app-shopfloor-depl-67bdd...	longratex-prod	1	0	ReplicaSet	svc3	BestEffort	68d	Running
lgtx-images-deployment-7f78fb...	longratex-prod	1	0	ReplicaSet	svc3	Burstable	2d6h	Running
lgtx-images-deployment-7f78fb...	longratex-prod	1	0	ReplicaSet	svc2	Burstable	2d6h	Running
lgtx-images-deployment-7f78fb...	longratex-prod	1	0	ReplicaSet	svc1	Burstable	2d6h	Running
lgtx-mongodb-0	longratex-prod	5	0	StatefulSet	svc1	Burstable	68d	Running
lgtx-mongodb-1	longratex-prod	5	0	StatefulSet	svc3	Burstable	70d	Running
lgtx-mongodb-2	longratex-prod	5	0	StatefulSet	svc2	Burstable	70d	Running
lgtxprd-srv-order-import-depl-64...	longratex-prod	1	0	ReplicaSet	svc1	BestEffort	70d	Running

Figura 12 - Através do *Lens* gerimos os Pods do projeto, que contém um ou mais contentores. O Pod relacionado com a API de Gestão de Catálogos encontra-se sublinhado a azul.

Para que as modificações efetuadas no código-fonte fossem devidamente aceites, era necessário seguir determinados procedimentos (Figura 13). Caso alguma das fases não fosse bem-sucedida, assegurava-se que o código que se encontrava em produção permaneceria intacto.

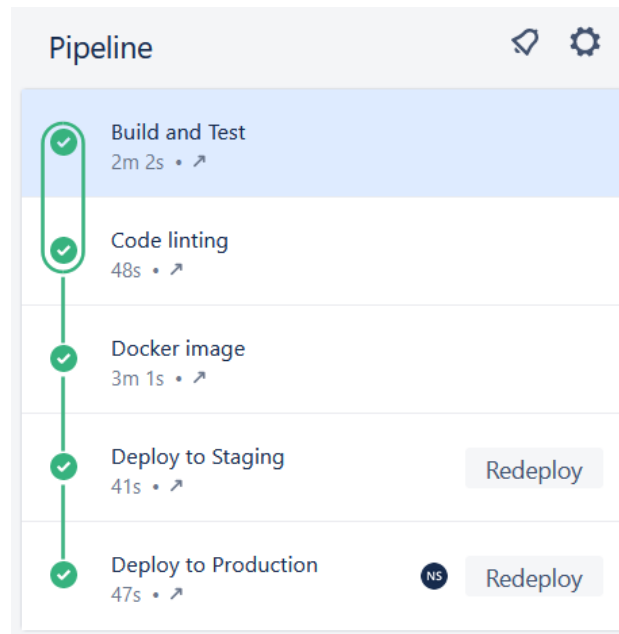


Figura 13 - Etapas de integração de modificações do código.

A primeira etapa envolvia a verificação do código-fonte por meio do *software Lint*, bem como a validação dos testes unitários. Esses dois processos eram realizados simultaneamente.

A segunda etapa compreendia a criação de uma imagem *Docker* da aplicação com as modificações efetuadas. As imagens *Docker* são pacotes independentes que englobam o código da aplicação, bem como suas dependências e configurações.

Posteriormente, utilizando o *Kubernetes*, e através de um ficheiro de configuração (manifestos YAML) para definir como a imagem deveria ser executada, a imagem *Docker* era implementada como a nova versão da aplicação ativa.

Inicialmente, a imagem *Docker* era transferida para um ambiente de “*staging*”, que servia para testes e deteção de erros. Isso permitia que quaisquer problemas fossem identificados e corrigidos antes de esta versão ser utilizada globalmente.

Após a verificação e correção dos eventuais erros encontrados, esta versão seria transferida para o ambiente de produção, permitindo que os restantes membros da empresa trabalhassem com a nova versão do projeto.

Desenvolvimento da Solução

Nesta secção, são abordadas as fases do desenvolvimento do projeto. Por ordem, são discutidos os seguintes tópicos: a adaptação do projeto de gestão de catálogos existente para o site da empresa “El Corte Inglés”, o desenvolvimento das ferramentas de gestão de catálogos para a plataforma *Magento*, a criação das ferramentas de gestão de artigos, a implementação dos métodos de ligação entre os atributos do ERP *Macwin* e a plataforma *Magento*, a introdução de instrumentos de validação de artigos, a incorporação das ferramentas de sincronização entre a API e a plataforma *Magento* e, por último, o desenvolvimento do site de vendas B2B.

Adaptação do Projeto

O primeiro passo na construção da solução do projeto envolveu a adaptação do projeto existente para a criação de catálogos e gestão de pedidos e encomendas na plataforma *Mirakl*. Foi criada uma classe para distinguir entre as plataformas de catálogos consoante o pedido ao servidor. Quaisquer referências diretas à API da plataforma *Mirakl* foram substituídas por uma referência abstrata à API necessária para esse contexto.

A seguir, com base nos métodos utilizados no serviço da plataforma *Mirakl*, foi desenvolvido um serviço para a plataforma *Magento* para lidar com encomendas e pedidos.

Por fim verificou-se se as funcionalidades do projeto para a plataforma *Mirakl* continuavam inalteradas depois das modificações, e após essa confirmação prosseguiu-se para a próxima etapa do projeto.

Gestão de Catálogos

Nesta fase, foram criadas as ferramentas necessárias para a gestão dos catálogos a serem enviados para a plataforma *Magento*. Começando pela criação dos catálogos (Requisito RA0001, Figura 14), o utilizador necessita obrigatoriamente de indicar o nome do catálogo (nome este que devia ser único), os utilizadores são obrigados a fornecer o nome único do catálogo, seleccionar a estação correspondente e as plataformas para as quais deve ser enviado. Embora opcional, também é possível fornecer uma descrição do catálogo para maior clarificação do seu propósito.



O formulário, intitulado "Detalhes do catálogo", contém os seguintes campos:

- Nome:** Campo de texto obrigatório.
- Descrição:** Campo de texto opcional.
- Estação:** Campo de seleção com uma seta para cima e uma seta para baixo.
- Plataformas:** Campo de seleção com uma seta para cima e uma seta para baixo.

Na base do formulário, há dois botões: "Cancelar" (cinza) e "Guardar" (laranja).

Figura 14 - Ferramenta de Criação de Catálogos para a plataforma Magento.

Através dos atributos obtidos do site, verifica-se se o catálogo possui um nome único e, após essa validação, o novo catálogo é armazenado no *MongoDB*. Após a conclusão da criação do catálogo, o processo passa para a fase de criação dos artigos.

Para esse objetivo, é utilizado um serviço responsável por interagir com o ERP *Macwin*. Esse serviço solicita todos os artigos armazenados na base de dados do ERP que correspondem à temporada de vendas desejada. Ao obter esses artigos, são criados dois objetos para cada um: um que representa o artigo com todos os seus atributos e outro que estabelece a ligação entre o artigo e o catálogo gerado. Esses objetos são criados com o propósito de assegurar que nenhum artigo se encontra diretamente ligado ao catálogo. É fundamental garantir que não existe ligação direta entre catálogos e artigos, pois vários catálogos podem conter artigos em comum, e assim ao ter um documento próprio para conectar esses mesmos itens garante-se que qualquer alteração específica de um artigo de um catálogo não altera informações partilhadas. Quando um artigo já existia na base de dados da API, apenas se estabelece a ligação entre ambos.

Após a ser concebida a funcionalidade de criação de catálogos, procedeu-se à criação de um método para atualizar os detalhes dos artigos de um catálogo (RA0013, Figura 15). Dado que os artigos podem ser modificados diretamente no ERP, é essencial garantir que os utilizadores disponham de uma maneira de atualizar os artigos de um catálogo através da API, assegurando assim que essas alterações também se refletissem na base de dados da API. Nesse sentido, foi implementado um método que inicialmente obtém os artigos de um catálogo a partir da base de dados do ERP, compara esses artigos com aqueles armazenados na base de dados da API e, caso sejam detetadas alterações, procede-se à respetiva atualização do artigo.

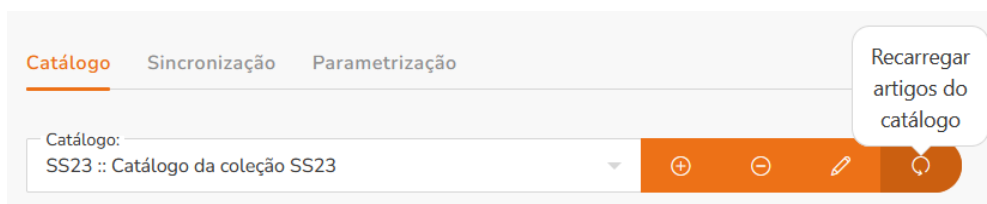


Figura 15 - Botão para atualização de artigos de um catálogo.

Com o objetivo de facilitar a visualização dos artigos de um catálogo, foi explorada a aplicação de filtros aos artigos (RA0004, Figura 16). O utilizador tem a possibilidade de ajustar a quantidade de artigos a serem exibidos por página, efetuar pesquisas com base em *SKUs* específicos, optar por visualizar tanto artigos válidos como inválidos, ou apenas artigos que estejam numa dessas categorias. Caso a opção seja por artigos inválidos, é possível especificar as razões pelas quais esses artigos são considerados inválidos. Outras opções de filtragem são implementadas através do *Front-end* do site.

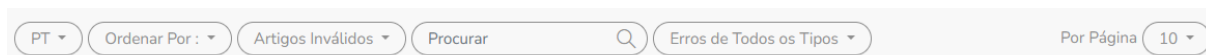


Figura 16 - Filtros de artigos de um catálogo.

Desenvolveu-se uma funcionalidade para a edição dos dados de um catálogo (RA0003, Figura 17). Nesse processo, permite-se efetuar alterações apenas no nome e na descrição do catálogo, respeitando a condição de que o nome do catálogo continua a ser único.

Figura 17 - Ferramenta de edição de catálogo.

Por último, foi desenvolvido um método para eliminar catálogos (RA0002, Figura 18). Primeiro, é realizada uma verificação para garantir que o catálogo não se encontre associado a nenhuma operação em desenvolvimento. Posteriormente, procede-se à remoção das ligações entre o catálogo e os artigos, seguida da eliminação do próprio catálogo.

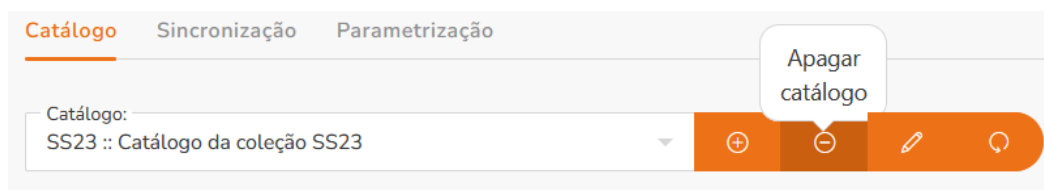


Figura 18 - Botão de remoção de catálogo.

Gestão de Artigos

A primeira funcionalidade a ser desenvolvida para a gestão de artigos de um catálogo consistiu na seleção ou desmarcação de artigos para sincronização com a plataforma *Magento* (RA0006, Figura 19). Visto que é por vezes necessário enviar apenas determinados artigos ou configurações de um artigo para a plataforma, como uma cor específica ou um artigo com determinados tamanhos, torna-se essencial a capacidade de fazer essa distinção. Nos objetos que ligam os artigos aos catálogos, foram introduzidos *arrays* para as cores e tamanhos dos artigos, que guardam o estado de seleção dessas configurações, além do estado de seleção geral do artigo. Durante a sincronização do catálogo com a plataforma *Magento*, são verificados os artigos e combinações que estão selecionados, enviando apenas esses.

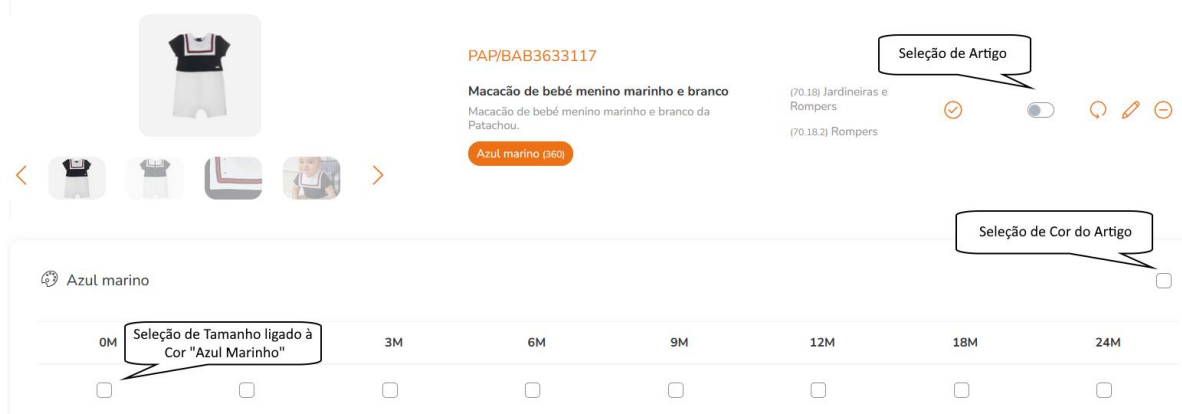


Figura 19 - Seleção ou Desmarcação de Artigos e Configurações de Artigos.

Assim como na gestão de catálogos, considerou-se essencial implementar um método de atualização dos dados de um artigo (RA0013, Figura 20). No caso de haver diferenças entre os dados armazenados no ERP e os dados na base de dados da API, os dados da API são atualizados para refletir as informações mais recentes.

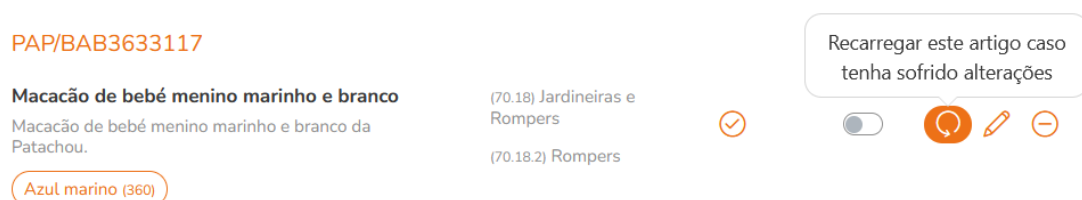


Figura 20 - Botão de Atualização de Dados de Artigo Específico.

Um dos objetivos da API também inclui a capacidade de realizar edições de artigos através do mesmo (RA0005, Figura 21), sem a necessidade de intervenção direta no ERP. Atualmente, é possível editar apenas os atributos relacionados com o nome e a descrição de um artigo em três línguas diferentes: Português, Inglês e Espanhol. Contudo, a perspectiva é que, no futuro, se torne possível editar todos os atributos de um artigo por meio da API. Adicionalmente, foi desenvolvido um método de edição em massa, embora este ainda não esteja operacional no site de operações diárias.

Editar Artigo PAP/BAB3633117

×

Título em PT

Macacão de bebé menino marinho e branco

Descrição em PT

Macacão de bebé menino marinho e branco da Patachou.

Título em EN

Navy and white baby boy romper

Descrição em EN

Patachou navy and white baby boy jumpsuit.

Título em ES

Mono de bebé niño marino y blanco

Descrição em ES

Mono de bebé niño Patachou marino y blanco.

Cancelar

Guardar

Figura 21 - Campos de Edição de um Artigo.

A última funcionalidade implementada para a gestão de artigos envolveu a capacidade de remover artigos (RA0007, Figura 22) de um catálogo. Para que um artigo fosse removido existem duas pré-condições a cumprir: o catálogo não pode estar a sofrer alterações através de outra operação (por exemplo, não pode estar a ser recarregado) e a criação de artigos para o catálogo tem de estar concluída. Quando um artigo é marcado para remoção do catálogo, o seu SKU é adicionado a uma lista negra. Esta ação é necessária porque o ERP não permite a eliminação de artigos. Por este motivo, um artigo está sempre associado a uma estação, e qualquer atualização do catálogo resulta na recriação desse artigo. Assim, antes de criar os artigos, verifica-se se o seu SKU consta na lista negra. Em caso afirmativo, o artigo não é processado. Depois de ser adicionado à lista negra, a ligação entre o catálogo e o artigo é removida.

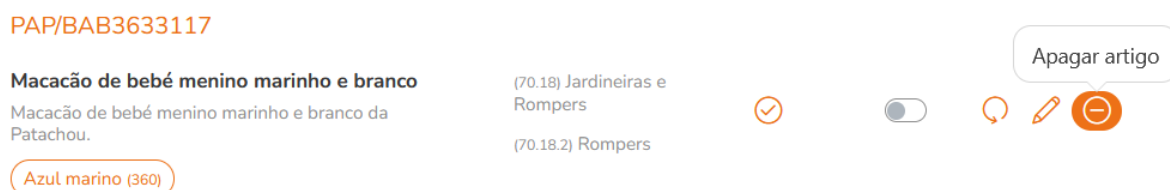


Figura 22 - Botão de Remoção de um Artigo.

Parametrização de Atributos

Uma das prioridades do projeto envolvia a ligação dos atributos presentes no ERP aos atributos disponíveis para seleção na plataforma *Magento* (RA0012, Figura 23). Tanto o ERP quanto o *Magento* utilizam *Ids* internos para identificar atributos (por exemplo, a cor azul é identificada pelo *Id* 50 no ERP e pelo *Id* 29 no *Magento*). Como é necessário relacionar esses *Ids* entre as duas plataformas, desenvolveu-se uma ferramenta que permite aos utilizadores associar um *Id* de um atributo do *Magento* a um *Id* de um atributo do ERP. Para um atributo específico, os *Ids* relacionados a esse atributo eram obtidos a partir do ERP e, em seguida, do *Magento*. Esses *Ids* são disponibilizados ao utilizador para que pudesse escolher os atributos correspondentes. Posteriormente, quando era necessário verificar os *Ids* associados, um método criado para esse fim era utilizado.

Cor

^ ? Guardar

Cod. Macw	Desc. Macw	Magento
001	Amarelo	<div>Cor</div> <div>Amarelo</div> <div>Azul</div>
002	Azul	<div>Azul Claro</div> <div>Azul Marinho</div>
003	Azul Claro	<div>Bege</div> <div>Bordô</div>
004	Azul Índigo	<div>Branco</div> <div>Castanho</div>
005	Azul Marinho	<div>Cor</div> <div>Azul Marinho</div>

Figura 23 - Parametrização do Atributo Cor.

Validação de Artigos

Para que um catálogo pudesse ser enviado à plataforma de vendas, era necessário validar os artigos nele contidos com base em critérios específicos (RA0008, Figura 24). Atualmente, esses critérios de validação são fixos; no entanto, a API dispõe de um método para modificar os requisitos de validação de um catálogo.

Foi desenvolvido um método que avalia se um artigo satisfaz os requisitos do catálogo ao qual pertence. Se o artigo não cumprir algum requisito, é criado um relatório no objeto de ligação entre o catálogo e o artigo. Esse relatório contém informações identificativas sobre a configuração do artigo onde ocorreram os erros (como por exemplo o seu tamanho e cor), juntamente com a mensagem correspondente ao requisito que não foi cumprido. Além disso, os contadores de artigos inválidos do catálogo e do artigo são atualizados.

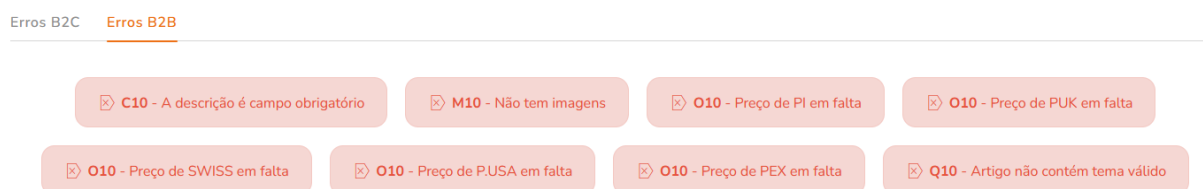


Figura 24 - Mensagens de Erros de Validação de um Artigo.

Sincronização para Sites de Vendas

Para que os artigos de um catálogo possam ser enviados para o site de vendas (RA0009, Figura 25), é criada uma fila contendo os artigos válidos e selecionados. No momento da criação, as filas ficam inativas. Quando o utilizador decidir iniciar o envio dos artigos para o site, basta ativar a fila. Se houver mais de uma fila pronta para iniciar o envio de artigos, a primeira fila a ser tratada é escolhida com base na sua data de criação. Somente após essa fila ser processada é que a próxima entra em funcionamento.

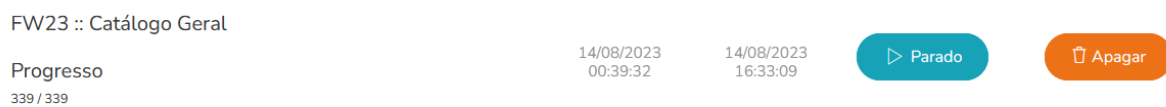


Figura 25 - Fila de Sincronização de um Catálogo.

Através de um serviço que realiza verificações periódicas de filas prontas para serem processadas, inicia-se o tratamento das mesmas, convertendo os artigos em vários objetos. Esses objetos são então inseridos nos pedidos que são enviados para a API da plataforma *Magento* [15], possibilitando assim a inclusão dos artigos no site.

Primeiramente, as configurações de um artigo da API são convertidas em objetos usados para criar artigos simples no *Magento* (ou seja, artigos que não incluem outros artigos). Esses artigos simples representam os produtos disponíveis para compra no site e são associados a tamanhos e cores específicos. Porém, esses artigos simples não se encontram visíveis ao cliente no site de vendas.

Após a inclusão dos artigos simples, o artigo da API é convertido num objeto utilizado para criar um artigo configurável no *Magento*. Um artigo configurável abrange todos os artigos simples de um item, não se encontra associado a nenhuma cor nem tamanho e é este que os clientes podem visualizar no site de vendas. A partir dele, são selecionadas as configurações desejadas de um produto, permitindo assim a aquisição dos artigos simples correspondentes.

No caso de um artigo simples ou configurável já estar presente no site de vendas, solicita-se à plataforma *Magento* que forneça a versão existente do artigo armazenado. A seguir, é efetuada uma comparação entre o artigo atual na API e o artigo no site. Caso haja

Desenvolvimento de Site de Vendas B2B

Esta fase do desenvolvimento da solução teve início com a instalação da plataforma *Magento* e das ferramentas associadas numa máquina Linux hospedada num servidor pertencente à empresa.

Após concluir a configuração dos URLs do site de vendas B2B e assegurar a acessibilidade e instalação adequada da plataforma, deu-se continuidade à etapa seguinte, que foi a instalação do tema para o site (Requisito RS0001). Uma vez que a versão do tema originalmente desejada pela empresa se encontrava desatualizada, optou-se pela instalação de um tema alternativo, também escolhido pela empresa, intitulado Supro [12]. Foi realizada a configuração do tema de forma a alinhar-se melhor com a visão da empresa, sendo necessário efetuar correções em erros visuais e de funcionalidade que foram identificados.

Com o intuito de garantir a disponibilidade do site tanto em Português como em Inglês, optou-se pela instalação de um dicionário de traduções desenvolvido pela empresa *Mageplaza* [16] (RS0002). Dessa forma, quando a vista da loja fosse definida como sendo em língua portuguesa, a plataforma procederia à verificação da existência de traduções correspondentes no dicionário, apresentando-as se encontrasse correspondências. Sempre que fosse necessário adicionar novas traduções, estas seriam adicionadas manualmente no dicionário.

Criou-se depois uma página inicial que servisse como uma página de seleção do país onde se encontra o cliente (RS0003). Uma vez que diferentes países acedem à loja através de vistas distintas da loja, ao selecionar um país, o utilizador seria redirecionado para a vista correspondente à seleção efetuada.

Com a intenção de obrigar os clientes a efetuar o registo no site antes de navegarem pelo mesmo (RS0004), e dado que esta funcionalidade não estava nativamente disponível na plataforma *Magento*, optou-se pela aquisição e instalação da extensão “*Required Login*” [17] da empresa *Mageplaza*. Adicionalmente, foram limitados os dados apresentados nos cabeçalhos e rodapés do site na página de Registo e Login, a fim de assegurar a restrição dos dados do site.

Por último, devido à limitação inerente à plataforma *Magento* que impede a compra de mais do que uma configuração de um artigo, e considerando o desejo da empresa de permitir que os clientes adquirissem várias configurações através de uma grelha onde pudessem inserir quantidades para cada configuração, visando facilitar a compra em grande volume (RS0005), optou-se pela aquisição e implementação da extensão “*Product Matrix Variants*” [18] desenvolvida pela empresa Webkul.



CAP IV – Discussão dos Resultados

Apresentação e Discussão dos Resultados

No final do estágio, a API encontra-se completamente funcional, e era inclusivamente utilizada pela empresa no desenvolvimento do catálogo correspondente à temporada "Outono/Inverno 2023". Também era utilizada na produção de catálogos para o site B2C da empresa. Está prevista a incorporação de novas funcionalidades, e espera-se que o desenvolvimento seja continuado pelo estagiário.

O site de vendas B2B encontra-se funcional e acessível. No entanto, permanece num estado ativo de desenvolvimento, não se encontrando pronto para ser lançado oficialmente.

Apresentação e Discussão dos Impedimentos e Constrangimentos

Uma parte substancial do desenvolvimento da API envolveu a adaptação de um projeto já existente, exigindo do aluno a interpretação do código prévio para realizar as devidas adaptações, o que representou um desafio por si próprio.

Houve um contratempo com o servidor onde o site B2B estava hospedado, resultando na corrupção de alguns dos seus discos, inclusive o disco que abrigava o site. Isso levou o estagiário a recomençar o desenvolvimento do site de raiz. Esse incidente causou um atraso de várias semanas no desenvolvimento do site.



CAP V – Conclusão

Reflexão Crítica dos Resultados

As componentes desenvolvidas para o projeto encontram-se num estado positivo. A API cumpre os objetivos atuais da empresa, e a sua simplicidade de utilização faz com que esta seja eficaz para o uso de outros membros da empresa. Algumas melhorias de código ainda podem ser implementadas de modo a tornar a API mais eficiente, porém atualmente não é o foco principal no desenvolvimento.

O site de vendas B2B por sua vez poderia encontrar-se num estado de desenvolvimento mais avançado, pois embora os requisitos tenham sido cumpridos de modo satisfatório, os contratempos ocorridos atrasaram o avanço deste componente.

Conclusão e Trabalho Futuro

Em conclusão, o trabalho desenvolvido ao longo do projeto levou a um aumento considerável do conhecimento do estagiário, a interação com diversas ferramentas de desenvolvimento e metodologias de trabalho garantem que no futuro, caso este lidere um projeto de desenvolvimento de *software*, este esteja melhor preparado para aplicar estas mesmas técnicas de modo a garantir uma evolução eficaz do mesmo.

Num futuro próximo o desenvolvimento do projeto irá centrar-se na evolução do estado do site B2B, e do lado da API, na implementação de mais opções de edição de atributos de artigos, correção de *bugs*, e melhoria do tempo de resposta da API.

Referências

- [1] Longratex – Fábrica de Confecções, Lda, “Patachou B2B Platform,” [Online]. Available: <https://agents-dev.patachou.com/>.
- [2] V. Kumar e G. Raheja, Business to business and business to consumer management.
- [3] T. H. Davenport, “Putting the Enterprise into the Enterprise System,” 1998.
- [4] MacWin Tek, Lda, “Software de gestão ITV *Macwin*,” [Online]. Available: <https://www.macwin.pt/pt/1-erp-macwin/3-solucoes-gm-textil/>.
- [5] A. Manzoor, E-commerce: an introduction, 2010.
- [6] R. Miller, “Adobe to acquire *Magento* for \$1.68B,” 21 Maio 2018. [Online]. Available: <https://tcrn.ch/2lCqgLT>.
- [7] Zend, “Varien and the *Magento* eCommerce Platform,” 12 Janeiro 2015. [Online]. Available: <https://web.archive.org/web/20150112072406/http://www.zend.com/topics/Magento-CS.pdf>.
- [8] Adobe Inc., “*Magento*, now Adobe Commerce | eCommerce Software,” [Online]. Available: <https://business.adobe.com/products/magento/magento-commerce.html>.
- [9] S. Richard e P. LePage, “What Are Progressive Web Apps?,” 9 Junho 2020. [Online]. Available: <https://web.dev/i18n/en/what-are-pwas/>.
- [10] I. Elliot, “Firefox Drops Support For PWA,” 6 Janeiro 2021. [Online]. Available: <https://www.i-programmer.info/news/87-web-development/14261-firefox-drops-support-for-pwa.html>.
- [11] Envato, “Utero - Clean, minimal *Magento* 2 theme!,” [Online]. Available: <https://themeforest.net/item/utero-clean-minimal-magento-2-theme/34242190>.
- [12] Envato, “Supro - Minimalist AJAX Theme,” [Online]. Available: <https://themeforest.net/item/supro-minimalist-ajax-magento-2-theme/23476627>.
- [13] Agile Alliance, “What is Agile?,” [Online]. Available: <https://www.agilealliance.org/agile101/>.
- [14] Atlassian Corporation, “Jira | Issue & Project Tracking Software | Atlassian,” [Online]. Available: <https://www.atlassian.com/software/jira>.

- [15] Adobe Inc., “REST endpoints,” [Online]. Available: <https://developer.adobe.com/commerce/webapi/rest/quick-reference/>.
- [16] Mageplaza, “Portuguese Language Pack for *Magento 2*,” [Online]. Available: <https://www.mageplaza.com/magento-2-portuguese-language-pack.html>.
- [17] Mageplaza, “Required Login for *Magento 2*,” [Online]. Available: <https://www.mageplaza.com/magento-2-required-login/>.
- [18] Webkul, “*Magento 2* Product Matrix Module | Configurable Variants Grid Extension,” [Online]. Available: <https://store.webkul.com/magento2-product-matrix.html>.
- [19] Scrum Alliance, “What Is Scrum: A Guide to the Most Popular Agile Framework,” [Online]. Available: <https://www.scrumalliance.org/about-scrum>.