Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Mestrado em Informática e Sistemas
Estágio/Projeto Industrial
Relatório Final

Desenvolvimento e teste de uma aplicação baseada em SOA

Carla Maria Silva Machado

Orientadores:

João Carlos Costa Faria da Cunha

Cristiana Manuela Afonso Areias

Instituto Superior de Engenharia de Coimbra

Coimbra, dezembro, 2015

Resumo

As arquiteturas orientadas a serviços, vulgarmente designadas por SOA (*Service Oriented Architecture*), são um tipo de arquitetura de sistemas que tem estado a ganhar notoriedade e relevância na atualidade, especialmente em soluções empresariais, uma vez que apregoa características bastante interessantes, nomeadamente uma boa capacidade de integração de diferentes sistemas e tecnologias juntamente com um elevado grau de adaptabilidade.
No entanto, apesar da popularidade crescente dos ambientes SOA, a informação disponível publicamente é ainda bastante limitada, resumindo-se na sua maioria a informação ao nível teórico sem se encontrar associada a uma concretização prática. Esta limitação de informação torna-se ainda mais notória no que se refere a exemplos concretos de ambientes SOA, não sendo comuns os exemplos que possam ser analisados em profundidade de forma a serem utilizados como referência ou como base para estudos ou testes sobre o tema SOA.
Considerando a relevância do tema e as limitações na informação disponível foi objetivo deste projeto fazer uma apresentação do conceito de SOA e a sua concretização num caso de estudo, pela idealização e implementação de um sistema baseado em SOA.
O caso de estudo desenvolvido foi o SOASales, no seu desenho e implementação procurou-se representar diversas características associadas a ambientes SOA, 1) recorrendo a uma diversidade tecnológica, tal como serviços SOAP e REST implementados em C# e Java; e 2) integrando e orquestrando diferentes sistemas de serviços por meio de um *service bus,* o Mule ESB, o que permitiu a adição de lógica de forma a acrescentar valor às funcionalidades disponibilizadas pelos diversos intervenientes.
Quando comparados diversos exemplos de ambientes SOA encontrados na literatura, o caso de estudo SOASales apresenta-se como o mais fiel aos princípios de um SOA, e o que apresenta uma maior diversidade de tecnologias envolvidas e intervenientes presentes no sistema.

Abstract

Service oriented architectures, commonly named SOA, are a style of system architecture that has been gaining relevance, particularly in business solutions, due to the proclamation of an interesting set of characteristics, such as the ability to easily promote integration between different systems and technologies in addition to a high level of adaptability.

However, despite the increase of popularity of SOA environments, the publicly available information is still limited, being in most cases restricted to theoretical information without a practical concretization. This limitation is even more noticeable when it comes to concrete examples of SOA systems, examples that can be analysed in depth so as to be used as reference for studies aren't common.

Taking into account the theme's relevance and the information limitations it was an objective of this project to present the concept as well as it's concretization on a case study by idealizing and implementing a SOA based system.

In the development of the case study, SOASales, it was sought to represent several characteristics associated with SOA environments, 1) using a variety of technologies like SOAP and REST services with implementation both in C# and Java; and 2) integrating and orchestrating different systems and services by use of a service bus, the Mule ESB, which allowed the addition of business logic and an increase of value for the available functionalities.

When comparing the SOASales with the examples found in the literature, the SOASales presents itself as the most loyal to the SOA principles as well as being the one showcasing a higher degree of technological diversity and participants in the system.

Palavras-Chave

SOA,
Orquestração de serviços,
ESB,
Mule ESB

Keywords

SOA,
Services,
Service Orchestration,
ESB,
Mule ESB

Glossário

Termo	Descrição
Aplicação	Software com uma finalidade específica
BPEL	*Business Process Execution Language – Standard* para a orquestração de serviços em fluxos processuais
ESB	*Enterprise Service Bus* – plataforma de integração
Especificação dos requisitos	Descrição do propósito e ambiente de um sistema a desenvolver
Estilo de Arquitetura	Padrões de alto nível utilizados em sistemas. Conjunto de princípios a utilizar na modelação de uma família de sistemas.
Hardware	Parte física que compõe um sistema
IIS	*Internet Information Services* – servidor web para disponibilização de recursos na internet
Interface	Abstração que define a forma de interação da entidade com o mundo exterior. Separa métodos de comunicação externa dos detalhes internos da operação
JMS	*Java Message Service* – Standard de mensagens que permite a troca de mensagens entre aplicações
json	*JavaScript Object Notation* – formato de troca de informação
Modelo de negócio	Forma pela qual uma empresa cria valor para os seus interesses
Plataforma de software	Ambientes de *software*, como um sistema operativo, sobre o qual outros *softwares* correm
Protocolo	Conjunto de regras sobre modo de comunicação entre entidades
REST	*Representational State Transfer* – tipo de arquitetura de alto nível
Sessão	Permite associar informação a um utilizador individual
SOA	*Service Oriented Architecture* – tipo de desenho de arquitetura
SOAP	*Simple Object Access Protocol* – protocolo para troca de informações. Serviços SOAP utilizam este protocolo.
SoapUI	*Software* para a realização de testes
Software	Programas que comandam o funcionamento de um computador
WSDL	*Web Services Description Language* – formato xml para a descrição de serviços como um conjunto de *endpoints* e mensagens

XML	*Extensible Markup Language* – linguagem para a descrição de informação empregue na partilha de informação que é independente do tipo de software e hardware envolvidos na interação

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

As arquiteturas orientadas a serviços são um tema que tem verificado um interesse crescente havendo, no entanto, uma escassez de exemplos de sistemas que evidenciem esta arquitetura. Estes fatores influenciaram o tema selecionado para o projeto que foi o desenvolvimento e teste de uma aplicação baseada numa arquitetura orientada a serviços.

1.1. Motivação

Arquitetura orientada a serviços, ou SOA para abreviar, é um conceito que se encontra muito em voga nos dias de hoje. Apesar da complexidade associada a este conceito e da evolução que sofreu desde os seus primórdios, com alguns autores como T.Erl [1] a identificarem um conceito de SOA primitivo ou fundamental e um contemporâneo, de uma forma simplista pode dizer-se que tal como o nome indica se trata de um tipo de arquitetura de *software* que recorre a serviços. De uma forma mais concreta e para estabelecer um ponto de partida que será elaborado numa secção posterior deste relatório, podemos dizer que se trata de um tipo de desenho de arquitetura que visa um baixo nível de acoplamento entre agentes de software que interagem entre si, procurando assim responder às necessidades de interligação entre sistemas que podem não seguir os mesmos standards e protocolos assegurando o fluxo de processos de negócio [1, 2, 3, 4].

Um exemplo de uma definição que pode ser encontrada na literatura é a apresentada por T.Erl[1]:

"SOA is a form of technology architecture that adheres to the principles of service-orientation. When realized through the Web services technology platform, SOA establishes the potential to support and promote these principles throughout the business process and automation domains of an enterprise."

Apesar do interesse gerado pelo conceito SOA e dos variados estudos e publicações existentes este nem sempre é conhecido ou compreendido na sua globalidade, sendo por vezes empregue de uma forma abusiva e como palavra-chave na promoção de sistemas e soluções informáticas que apenas apresentam um conjunto limitado das características associadas a estes ambientes.

As características normalmente associadas a ambientes que seguem esta arquitetura são variadas e no seu conjunto prometem um conjunto apetecível de benefícios, como a capacidade de integração de sistemas de protocolos e tecnologias diferentes. São estas características que procuram destacar ambientes SOA de ambientes que seguem outros tipos de arquiteturas e tornam SOA um assunto bastante relevante não só a nível académico, mas também empresarial, tendo vindo a ser alvo de um interesse crescente. No entanto, apesar do interesse crescente, dos benefícios prometidos e do seu tempo de vida, a informação disponível publicamente é algo limitada encontrando-se principalmente ao nível teórico.

A informação disponibilizada torna-se ainda mais escassa no que se refere a exemplos de sistemas que seguem este tipo de arquitetura. Os exemplos disponíveis são não só em número limitado como, por vezes, a informação fornecida é reduzida e superficial, não havendo uma descrição concreta da arquitetura do sistema ou das tecnologias empregues.

De uma forma geral não existem disponíveis para consulta por parte de arquitetos de sistemas ou investigadores concretizações de sistemas representativos de um SOA realista, e que apresentem diferentes características e com um grau de complexidade mais elevado. Seria uma mais-valia para a investigação nesta área a existência de exemplos práticos que pudessem ser reutilizados por diferentes projetos sem a necessidade de desenvolvimento de um exemplo mais limitado e específico para cada projeto.

1.2. Objetivos

Tendo em conta o tema selecionado, foram definidos os seguintes objetivos genéricos para o projeto:

- Idealização de um caso de estudo de um sistema com uma arquitetura orientada a serviços, definindo e apresentando uma organização e área de negócio de suporte ao caso de estudo;
- Especificação dos requisitos do sistema, devendo seguir os princípios subjacentes a SOA e usar uma grande diversidade de tecnologias;
- Implementação, do sistema especificado;
- Análise qualitativa e quantitativa do sistema tendo em conta as características atribuídas a ambientes SOA.

1.3. Enquadramento Académico

Este projeto foi realizado no contexto do Mestrado de Engenharia de Informática e Sistemas na área de especialização de Desenvolvimento de Software do Instituto Superior de Engenharia de Coimbra. Tendo sido realizado sob a orientação dos professores João Cunha e Cristiana Areias do departamento de informática desta instituição.
Uma vez que ao início deste projeto já me encontrava inserida no mercado de trabalho, inicialmente com vínculo à ITGrow [5] e posteriormente à Critical Software [6], este projeto foi realizado em regime pós-laboral, tendo sido iniciado em Outubro de 2014 e terminado em Dezembro de 2015.

1.4. Estrutura do documento

Este documento encontra-se dividido em 6 capítulos.

No capítulo 2, Arquitetura Orientada a Serviços, é feita uma descrição do conceito de SOA, suas características e princípios, as vantagens e desvantagens e exemplos de ambientes SOA . São ainda apresentados alguns conceitos associados como os serviços e *Enterprise Service Bus*.

No capítulo 3, Definição de um Caso de Estudo baseado em SOA - SOASales, é apresentado o caso de estudo idealizado como base do projeto, nomeadamente o modelo

de negócio, requisitos funcionais e tecnológicos e a especificação dos serviços e do sistema resultante da análise dos requisitos.

No capítulo 4, Desenvolvimento do SOASales, é descrita a implementação do caso de estudo sendo apresentados detalhes da implementação dos serviços das organizações Vendas Online e Livros Online, dos fluxos do *service bus* e das aplicações auxiliares.

No capítulo 5, Análise e teste, é realizada uma análise qualitativa e quantitativa do SOASales, com base nas suas características e em resultados de testes realizados ao sistema.

No capítulo 6, Conclusões, são feitas algumas considerações finais sobre o projeto.

Este documento possui como anexos os seguintes documentos e ficheiros:

- Anexo A – Especificação do Sistema, documento que apresenta a totalidade da especificação do sistema realizada no contexto deste projeto
- Anexo B – Fluxos do Mule ESB, documento que apresenta todos os fluxos do Mule ESB desenvolvidos na implementação do SOASales apresentando a visualização gráfica de cada fluxo acompanhada do conteúdo do ficheiro xml associado. Apresenta também o código das classes auxiliares implementadas;
- Anexo C – Testes ao Sistema, documento que apresenta a totalidade dos resultados dos testes executados ao sistema; Anexo D – Código Fonte Aplicações, pasta que contém os projetos de todas as aplicações desenvolvidas neste projeto;
- Anexo E – Aplicações, pasta que contém os binários de todas as aplicações desenvolvidas neste projeto

2. Arquitetura Orientada a Serviços

Nesta seção é realizada uma introdução a SOA. Neste sentido apresenta-se uma definição do conceito assim como de alguns conceitos complementares, nomeadamente o de *Enterprise service bus*. Adicionalmente apresenta-se também algumas das vantagens e desvantagens normalmente associadas a ambientes SOA e exemplos quer no âmbito académico quer no âmbito empresarial.

2.1. Serviços

Os serviços são considerados os blocos de construção de um SOA, sendo uma parte essencial neste tipo de arquitetura. Um serviço pode ser definido como uma aplicação de software que realiza operações e suporta interações diretas com outras aplicações pela troca de mensagens, empregando protocolos de internet. Um serviço apresenta-se como uma interface que descreve um conjunto de operações que são acessíveis pela internet, escondendo desta forma os seus detalhes de implementação [7, 8, 9].

Numa interação que envolva um serviço podem ser identificados diferentes papeis para os intervenientes, como esquematizado na Figura 2.1, tais como o provedor do serviço (*provider*), o consumidor (*requestor*) e o *registry*, sendo que o *registry* não se trata de um papel obrigatório, podendo não estar presente na interação.

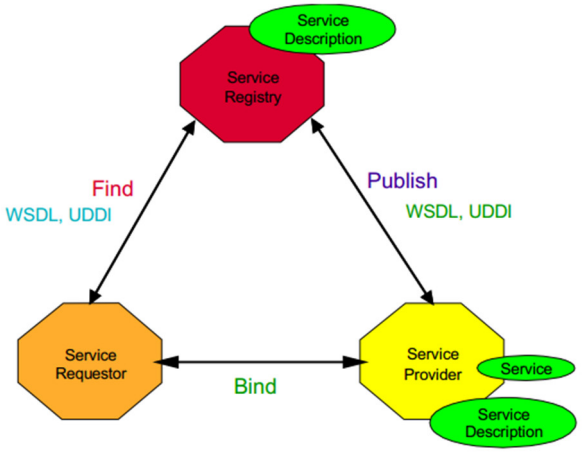

Figura 2.1: Papéis numa interação com um serviço [8]

O provedor do serviço é a aplicação/entidade que disponibiliza um serviço, ou seja, o dono do serviço. Para esse efeito disponibiliza a informação necessária para a sua invocação, a descrição do serviço. Esta informação pode ser disponibilizada diretamente para um consumidor ou para uma entidade intermediária, o *registry*.

O consumidor do serviço é uma aplicação/entidade que vai utilizar um serviço procedendo à sua invocação. De forma a proceder à invocação do serviço necessita da informação disponibilizada pelo provedor. No caso de não possuir essa informação localmente, pode obtê-la por meio de uma entidade de *registry*.

O *registry* é uma aplicação/entidade na qual os provedores de serviços podem publicar a descrição dos seus serviços. Os consumidores de serviços podem consultar e pesquisar estas entidades de forma a descobrirem as operações e serviços que necessitam. Esta entidade não é obrigatória na relação uma vez que o consumidor pode realizar uma

ligação ponto a ponto com o provedor, caso em que já possuiu a informação necessária para o estabelecimento da ligação, ou possuir uma um mecanismo próprio para obtenção da informação dos serviços e operações.

Para o estabelecimento de uma relação entre um consumidor e um provedor de serviços existem três etapas fundamentais.: 1) *publish,* etapa em que o provedor de um serviço deve fazer a publicação do mesmo de forma a que os seus consumidores o possam descobrir; 2) *find*, etapa em que os consumidores de serviços devem procurar e encontrar os serviços e operações que pretendem consumir de forma a determinar a sua localização e forma de invocação. Esta etapa pode ser realizada de forma estática, na fase de desenho, ou de forma dinâmica em tempo real. Podendo ser realizada por recurso ao *registry;* 3) *bind*, etapa em que o consumidor estabelece uma interação com o provedor, invocando o serviço [8, 10].

Idealmente um serviço deve apresentar as seguintes características [7]:

- Deve ser levemente acoplado de forma a ser reduzida a dependência entre serviços, permitindo que cada serviço envolvido numa interação possa evoluir sem que o outro seja afetado e que possam ser utilizados independentemente do hardware e plataforma de software em que assentam, assim como da sua linguagem de implementação.
- Deve ser definido por contrato, possuindo uma especificação da semântica do seu comportamento enquanto serviço. A especificação do serviço deve fornecer a informação necessária e suficiente para a sua utilização. A informação do contrato deve incluir os métodos disponíveis, as condições de utilização e o tipo de resposta do serviço.
- A informação que o serviço fornece ao exterior deve encontrar-se limitada ao seu contrato. Este fator ajuda na separação entre a lógica de implementação e os metadados permitindo que a implementação sofra alterações sem que o contrato seja afetado.
- Deve ser reutilizável (*reusable*) por diferentes clientes, permitindo a redução de redundância dentro de um sistema e complementarmente podendo implicar uma partilha de custos.
- Não devem ter noção de estado (*stateless*), e como tal não guardando informação entre chamadas. Serviços que guardem informação de estado entre chamadas implicam não só uma maior dependência como também dificultam o balanceamento de carga. Quando existe necessidade de serem substituídos não podem assegurar que a informação de estado é mantida.
- Deve ser passível de ser composto (*composable*), devendo ser implementado de forma a poderem ser utilizados por outros serviços, podendo assim ser utilizados por si só ou ser empregues por serviços compostos e oferecer uma funcionalidade mais abrangente e complexa.
- Deve ser detetável (d*iscoverable*), pois para ser possível a utilização de um serviço a sua localização deve ser conhecida.

2.2. Conceito SOA

Existem diferentes estilos de arquitetura que podem ser empregues no desenho soluções informáticas, como por exemplo arquiteturas orientadas a objetos, ou baseadas em componentes, entre outros. SOA é um estilo de arquitetura orientada a serviços. Esta é, no entanto, uma descrição muito básica e simplista do conceito. Nesta secção iremos abordar o tema em mais profundidade.

Um ambiente SOA é um ambiente que é desenhado com a intenção de ultrapassar diversos desafios [2]:

"An SOA is designed to allow developers to overcome many distributed enterprise computing challenges including application integration, transaction management, security policies, while allowing multiple platforms and protocols and leveraging numerous access devices and legacy systems"

No desenho de um sistema SOA é encorajada a utilização de unidades lógicas ou de trabalho autónomas, mas não completamente isoladas uma vez que deve ser assegurado um conjunto de regras e princípios comuns de forma a assegurar a existências de pontos comuns e standards ao mesmo tempo que é possibilitada a evolução independente de cada unidade [1, 2, 3, 4].

De um modo geral as unidades lógicas constituintes de um SOA são disponibilizadas na forma de serviços, normalmente utilizando dois tipos de serviços:

- SOAP, serviços que seguem o protocolo de troca de informação *Simple Object Access Protocol*;
- REST, serviços que seguem a arquitetura de alto nível *Representational State Transfer*.

No entanto, podem ser utilizadas outras tecnologias em particular tecnologias que respeitem as seguintes restrições:

- Implementem interfaces WSDL, que é um formato xml, Extensible Markup Language, para a descrição de serviços denominado *Web Services Description Language*;
- Comuniquem com recurso a mensagem com formato xml, ou seja, *Extensible Markup Language* que é uma linguagem de descrição de informação independente do hardware e software.

Cada serviço representa uma unidade de trabalho fornecida por um provedor e invocada por um consumidor [2].

Nos ambientes SOA e nas relações estabelecidas entre os intervenientes podem ser identificados diferentes papeis, devido à importância dos serviços nos ambientes SOA estes papeis sobrepõem-se aos apresentados no capítulo 2.1.

No capítulo 2.1 foram apresentados o provedor e o consumidor de serviços e o *registry* que continuam a estar presentes. Num ambiente SOA podem ser identificados dois papeis adicionais, o de agregador de serviços e o agente de serviços.

Um agregador de serviços (*service aggregator*) é uma aplicação/entidade que assume tanto o papel de provedor como de consumidor, como se pode observar na Figura 2.2. O agregador assume o papel de provedor de serviço para os seus clientes. Ao receber um pedido destes, irá assumir o papel de consumidor de forma a satisfazer o pedido. A satisfação do pedido poderá implicar apenas o consumo de um outro serviço e retornar o resultado, ou algo mais complexo que implique o consumo de diferentes serviços que podem pertencer a diferentes provedores. Um agregador procura muitas vezes oferecer aos seus clientes serviços compostos que oferecem funcionalidades acrescidas em relação a um serviço simples disponibilizado por um provedor [2, 11].

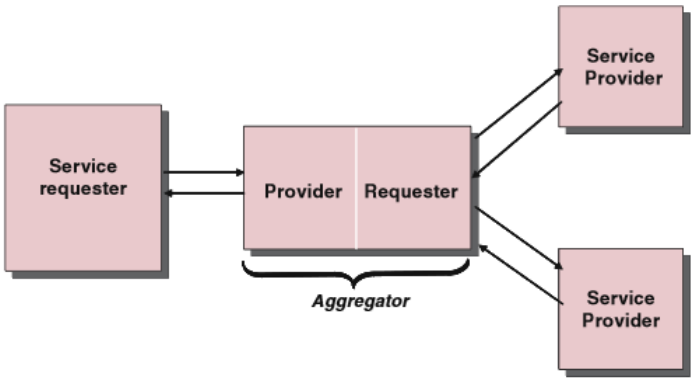

Figura 2.2: Agregador de serviços (service aggregator)[2]

Um agente de serviços (*service broker*) é uma entidade cujo papel engloba o de *registry* sendo o seu objetivo não apenas a disponibilização de uma listagem de serviços e operações mas também a geração de uma relação de maior confiança e segurança entre os diversos provedores e consumidores (Figura 2.3). Um agente enquanto elemento de confiança procura forçar os diferentes provedores a aderir e/ou cumprir determinadas práticas ou legislações. De forma a realizar a sua função o agente disponibiliza uma lista de provedores disponíveis podendo adicionar informação adicional, como por exemplo informação de qualidade, de segurança e confiança, informação esta que pode auxiliar os possíveis clientes na seleção de um provedor em detrimento de outro quando existe mais que um provedor a disponibilizar a mesma funcionalidade [2].

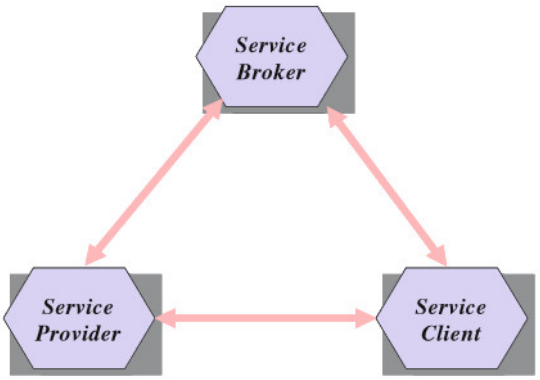

Figura 2.3: Agente de serviços (service broker) [2]

O aumento do interesse neste tipo de arquitetura encontra-se em grande parte relacionado com as características e benefícios a eles associados, que serão apresentados nas secções seguintes. Este interesse tem levado a que cada vez mais empresas afirmem possuir ambientes SOA ou planeiem a sua implementação e empresas a fornecer soluções SOA.

2.3. Princípios e Características

Uma SOA e os seus serviços constituintes devem preferencialmente respeitar certos princípios assim como possuir certas características de forma a ser assegurado um sistema funcional. Destes princípios e características alguns são fundamentais para o funcionamento do ambiente e para que este cumpra o seu propósito enquanto outras permitem a expansão e melhoramento do conceito base de SOA. Considerando a interligação entre SOA e serviços, alguns dos princípios e características sobrepõem-se aos apresentados na secção 2.1. Nesta secção iremos apresentar alguns destes princípios e características.

Um baixo nível de acoplamento entre os intervenientes do sistema deve ser uma das principais características de um SOA, permitindo uma minimização das interdependências entre os consumidores e provedores, podendo esta ser reduzida apenas à necessidade de terem conhecimento uns dos outros [1, 3, 4].

Os serviços intervenientes de um sistema SOA devem ser autónomos, cada serviço deve ser responsável e ter controlo sobre as suas funcionalidades. A mensagem enviada pelo consumidor do serviço a solicitar uma operação deve ser descritiva e não instrutiva, ou seja, não deve ter informação ou indicação no que diga respeito à forma de execução do pedido. O fornecedor de serviço é que deve ser o responsável nessa matéria [1, 2].

A lógica utilizada para a execução de pedidos deve encontrar-se abstraída da interface sendo os serviços opacos para os seus clientes, escondendo a lógica e disponibilizando apenas uma descrição das funcionalidades. A informação das funcionalidades deve ser fornecida preferencialmente numa linguagem descritiva, como por exemplo WSDL para serviços web, devendo descrever pelo menos o nome do serviço, nome das operações, os dados de entrada e os dados de saída. A funcionalidade disponibilizada deve ser encapsulada por uma interface independente da sua implementação[1, 3].

A reutilização de lógica deve ser promovida, a lógica deve encontrar-se dividida por diferentes serviços que representam pequenas unidades lógicas que podem ser reutilizadas em funcionalidades distintas [1, 2].

Os serviços disponibilizados podem ser compostos por uma coleção de serviços ou um serviço cuja lógica engloba funcionalidades disponibilizadas por vários serviços[1, 11].

Os serviços utilizados e consumidos pelo sistema não devem ter necessidade de armazenar informação de estado, de modo que a mensagem transmitida entre os intervenientes deve conter toda a informação necessária ao seu processamento, não havendo necessidade de guardar informação entre pedidos e permitindo que cada pedido seja tratado como genérico e que em caso de falhas a recuperação seja mais simples, aumentando a confiabilidade[1, 3].

Os serviços devem ser passiveis de ser descobertos e acedidos através de mecanismos de descoberta. A interface disponibilizada pelos serviços deve estar disponível para todos os intervenientes do sistema independentemente da sua localização e os seus mecanismos de invocação devem respeitar os standards[1, 2, 3].

O tratamento de um único pedido ou de pedidos duplicados deve produzir o mesmo resultado. Este comportamento permite que em caso de falha, a repetição do mesmo

pedido deverá originar o resultado pretendido. Esta característica pode provocar um melhoramento da confiabilidade dos serviços [3].

Uma característica que pode estar presente, apesar de ir contra algumas das que já foram mencionadas anteriormente é a partilha de sessão entre serviços, o que irá reduzir a escalabilidade ao mesmo tempo que aumenta o acoplamento entre os intervenientes. No entanto a partilha de sessão tem a vantagem de aumentar a eficiência, por exemplo pela partilha de informação de autenticação [3].

2.4. Vantagens e Desvantagens

São muitas as vantagens atribuídas a ambientes SOA, no entanto deve ser tido em conta que a sua adoção por si só não garante essas vantagens. As vantagens obtidas estão dependentes da situação em que é aplicado, dos objetivos traçados, das tecnologias empregues e da forma como é desenhado e implementado. Um ponto também a ter em conta é o facto de algumas das vantagens dos ambientes SOA apenas se manifestarem a médio e longo prazo.

Apesar destes condicionantes, algumas vantagens são mais generalizadas e comuns às diferentes implementações. Nesta secção iremos procurar apresentar algumas dessas vantagens mais generalizadas.

Tendencialmente a implementação de um ambiente SOA guiará o desenvolvimento no sentido da criação de uma solução interoperável trazendo a vantagem da diminuição do custo e esforço necessário para a integração de diferentes aplicações [1, 12].

O seguimento dos princípios associados a ambientes SOA promove a reutilização de lógica e por consequência a reutilização de serviços. Nesse sentido, os serviços são desenhados para dar resposta a uma necessidade específica e imediata, mas tendo sempre em vista a sua potencial reutilização no futuro por outras aplicações. A implementação de soluções com este intuito implica na generalidade um maior esforço e custo de desenvolvimento da solução atual. No entanto a reutilização da lógica de soluções já existentes deverá ter como consequência a diminuição do esforço e custo em soluções futuras, trazendo ainda uma flexibilidade acrescida ao sistema permitindo-lhe lidar com futuras mudanças de uma forma mais rápida e com menor esforço [1, 12].

O desenvolvimento e o crescente uso da tecnologia associada aos serviços web levou ao desenvolvimento de adaptadores abrindo as portas à utilização de aplicações pré-existentes em arquiteturas orientadas a serviços. Esta evolução trouxe consigo a possibilidade de integração de sistemas previamente isolados sem a necessidade de criar ligações ponto-a-ponto ou outras soluções de integrações, diminuindo o custo e esforço de integração e permitindo evitar ou pelo menos atrasar a substituição de sistemas pré-existentes que podem ter importância vital para a organização ou cuja substituição seja muito dispendiosa [1].

A adoção de um ambiente SOA origina a oportunidade de uniformizar o formato de apresentação de mensagens, pelo uso de uma linguagem standard como o xml, cujo conteúdo se tornará mais fácil de entender e de manter. Esta uniformização pode levar a uma redução da complexidade das aplicações envolvidas ao mesmo tempo que permite uma rede de comunicação mais adaptável e extensível. O uso de uma linguagem permite

o estabelecimento de fundações que mais uma vez irão promover a interoperabilidade do sistema [1].

Em arquiteturas orientadas a serviços uma assunção que deve estar bem presente é a de que as aplicações sofrem alterações, evoluem com o tempo. Um SOA bem desenhado e implementado oferece alguma proteção face a esta evolução uma vez que de acordo com os seus princípios os diferentes intervenientes do sistema podem evoluir independentemente uns dos outros sem provocar a necessidade de alterações uns nos outros [1].

Tendo sido mencionadas algumas das vantagens vamos agora olhar um pouco para os pontos menos positivos. Como acontece com qualquer tipo de arquitetura, a qualidade da solução final encontra-se intrinsecamente ligada ao processo de desenho e implementação da solução. Um mau desenho da arquitetura ou uma má implementação pode por em causa algumas das vantagens mencionadas.

Um exemplo de uma má implementação pode ser a implementação de uma solução distribuída em oposição a SOA, que pode em alguns casos ser provocado por um mau entendimento do conceito. Os principais problemas que podem advir desta situação são um mau particionamento das funcionalidades pelos serviços, a implementação de serviços simples em oposição a serviços compostos e se encontrarem de acordo com os standards [1].

A implementação de um ambiente SOA deve ser iniciada pelo estabelecimento das bases, nomeadamente na forma como a informação é estruturada e validada. No entanto, estas questões são muitas vezes negligenciadas. Uma organização pode deparar-se com um cenário em que possuiu diferentes sistemas SOA não compatíveis entre si por ter ocorrido o desenvolvimento de diferentes projetos isoladamente sem consideração de uma base comum previamente estabelecida [1].

Numa situação de migração para uma solução SOA uma transição de sucesso apenas é possível se esta se encontrar sustentada num trabalho de análise e planeamento prévio, caso isto não se verifique as repercussões ao nível da qualidade da solução e do custo da migração podem ser muito sérias [1].

Um mau desenho ou implementação pode ter um impacto bastante relevante na performance do sistema. Os requisitos devem ser analisados cuidadosamente e estabelecidos considerando o aumento das camadas lógicas de processamento que poderá advir da expansão do ambiente [1].

Um ambiente SOA promove a integração de diferentes sistemas e tecnologias. Neste sentido, é necessário um cuidado acrescido a vários níveis, como por exemplo questões de segurança nomeadamente ao nível das mensagens trocadas. Estas questões requerem atenção não só no imediato, mas também considerações relativas ao futuro do sistema com o desenvolvimento das tecnologias, protocolos e boas práticas. A implementação deste tipo de sistemas tem que ter em conta não apenas o próprio sistema e as tecnologias e plataformas empregues na sua implementação, mas também as tecnologias, plataformas e linguagens dos diversos componentes e intervenientes do sistema [1].

2.5. Enterprise Service Bus

A facilidade de integração de diferentes sistemas que muitas vezes se traduz em tecnologias, modelos de informação e protocolos distintos, é um dos pontos fortes atribuídos aos ambientes SOA. No entanto, esta integração não é um dado adquirido, mas sim uma componente do ambiente que deve ser estudada, planeada e executada e que também implica opções e custos.

Quando se trata de realizar a comunicação entre sistemas, tem-se a opção de criar ligações ponto a ponto entre o cliente e todos os sistemas que este invocará, o que implica a criação de interfaces específicos para cada conexão em conformidade com as características de cada sistema. Outra opção é a introdução no sistema de uma camada de integração que gere a comunicação entre os intervenientes e é responsável pela interoperabilidade entre os mesmos. Um exemplo de um intermediário que pode ser empregue nestas situações é um *Enterprise Service Bus* (ESB) do qual falaremos em mais detalhes nesta secção [2].

Um ESB é de uma plataforma de integração que recorre aos standards utilizados pelos serviços web de forma a oferecer suporte a uma variedade de comunicações através de diferentes protocolos promovendo o desacoplamento entre sistemas envolvidos numa integração e permitindo a quebra da lógica de integração em partes distintas e mais facilmente geridas. Neste sentido o ESB fornece uma infraestrutura sobre a qual se pode implementar um ambiente SOA, aplicando medidas de segurança e confiabilidade, e estabelecendo o controlo das mensagens trocadas entre os serviços controlando o seu fluxo e tradução [2].

Para o desempenho deste papel o ESB deve possuir diversas características:

- Deve ser capaz de interagir com sistemas pré-existentes que podem ser tecnologicamente obsoletos, mas que são vitais para as organizações que os empregam e que por variadas razões não podem ser substituídos ou modificados [2].
- Deve ser capaz de reencaminhar mensagens, sendo que este reencaminhamento pode incluir a transformação da mensagem de forma a garantir suporte para diferentes protocolos, modelos de mensagens e eventualmente também a transmissão de informação de contexto, como por exemplo de transação e de segurança. Este reencaminhamento pode ser baseado no conteúdo da mensagem ou por tópico, sendo esta uma forma de agrupamento de mensagens [2, 7].
- Deve ser capaz de realizar a monitorização dos serviços e log do sistema [2, 5].
- Idealmente deve também ser capaz de se ligar dinamicamente a serviços empregando a mesma API de ligação independentemente da tecnologia de implementação dos serviços, conseguindo descobrir, localizar e efetuar a ligação aos serviços podendo oferecer a possibilidade de seleção do serviço a utilizar de acordo com critérios de qualidade [2].

2.6. Aplicações SOA

Nos últimos anos a popularidade e utilização de ambientes SOA tem denotado um aumento. No entanto a informação sobre ambientes concretos não é abundante e a que existe nem sempre disponibiliza muitos detalhes, nomeadamente ao nível da arquitetura em si, tecnologias empregues ou pormenores de implementação. Apesar da escassez de

informação procura-se nesta secção apresentar alguns exemplos da sua utilização, quer em ambientes reais quer de investigação, iniciando pelos exemplos desenvolvidos para investigação, passando para os exemplos de ambientes reais, procurando no final fazer uma comparação entre os exemplos apresentados.

2.6.1. Ambientes para Investigação

Open SOALab – Reservas Aéreas

O Open SOALab [13] é uma iniciativa cujo objetivo se centra no fornecimento de um conjunto de aplicações SOA passíveis de serem utilizadas em estudos e investigações sobre ambientes SOA. Os exemplos disponibilizados são variados, como por exemplo uma aplicação de reservas de avião, uma de câmbios de moeda e uma de reservas de hotel apresentados de seguida.

A aplicação de reservas de viagens de avião, Figura 2.4, é um sistema que disponibiliza um conjunto de serviços simples passiveis de serem disponibilizados por uma companhia de aviação para que os seus utilizadores, por exemplo agências de viagem, realizem reservas. O sistema foi implementado recorrendo a um sistema Linux com base de dados MySQL e os servidores Apache e GlassFish, com diferentes linguagens, C# com utilização do Visual Studio, Java e PHP com o NetBeans, sendo as interfaces dos serviços definidos em WSDL e XSD [13].

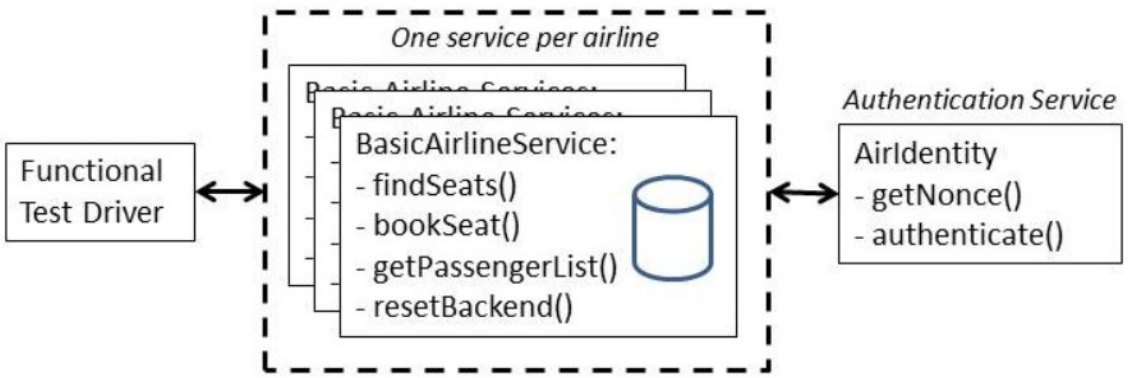

Figura 2.4: Exemplo SOA – Reservas aéreas [13]

Open SOALab – Câmbios

A aplicação de câmbios, Figura 2.5, representa um sistema online no qual participam vários bancos e corretores de moeda, sendo que estes últimos recebem pedidos de clientes para a realização do câmbio de moedas e recorrem aos diversos bancos para obtenção das cotações [13].

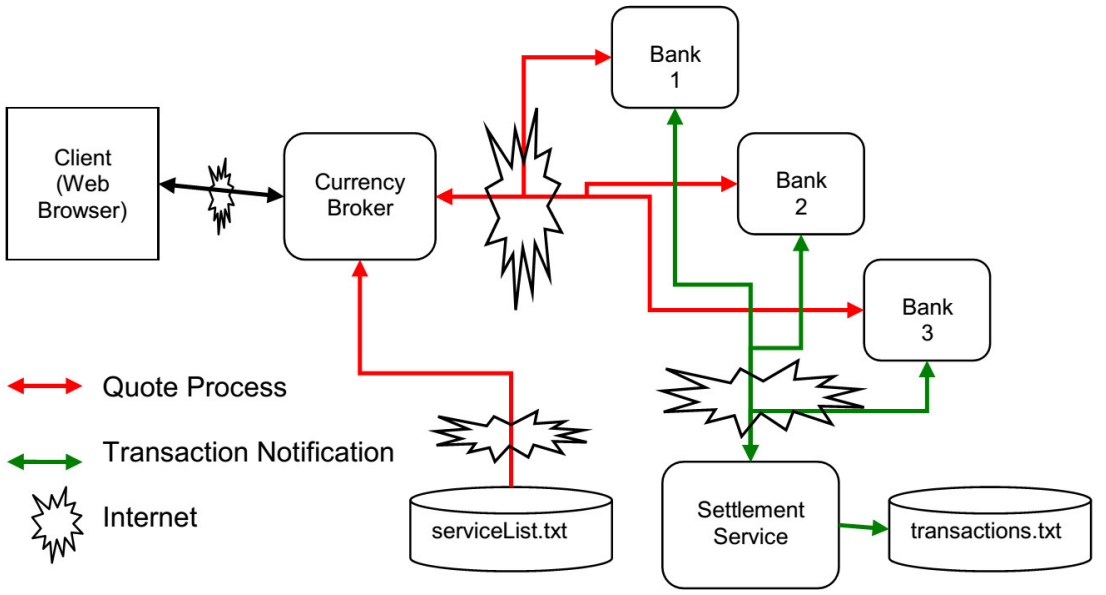

Figura 2.5: Exemplo SOA – Aplicação de Câmbio [13]

Open SOALab – Reservas de Hotel

A aplicação de reservas de hotel, Figura 2.6, representa um sistema que permite aos seus utilizadores a reserva de quartos de hotel, incluindo a funcionalidade de pagamento em diferentes moedas, recorrendo para isso à aplicação de câmbio mencionada no parágrafo anterior. Este sistema foi implementado em PHP com recurso a um servidor Linux, uma base de dados MySQL e um servidor http Apache [13].

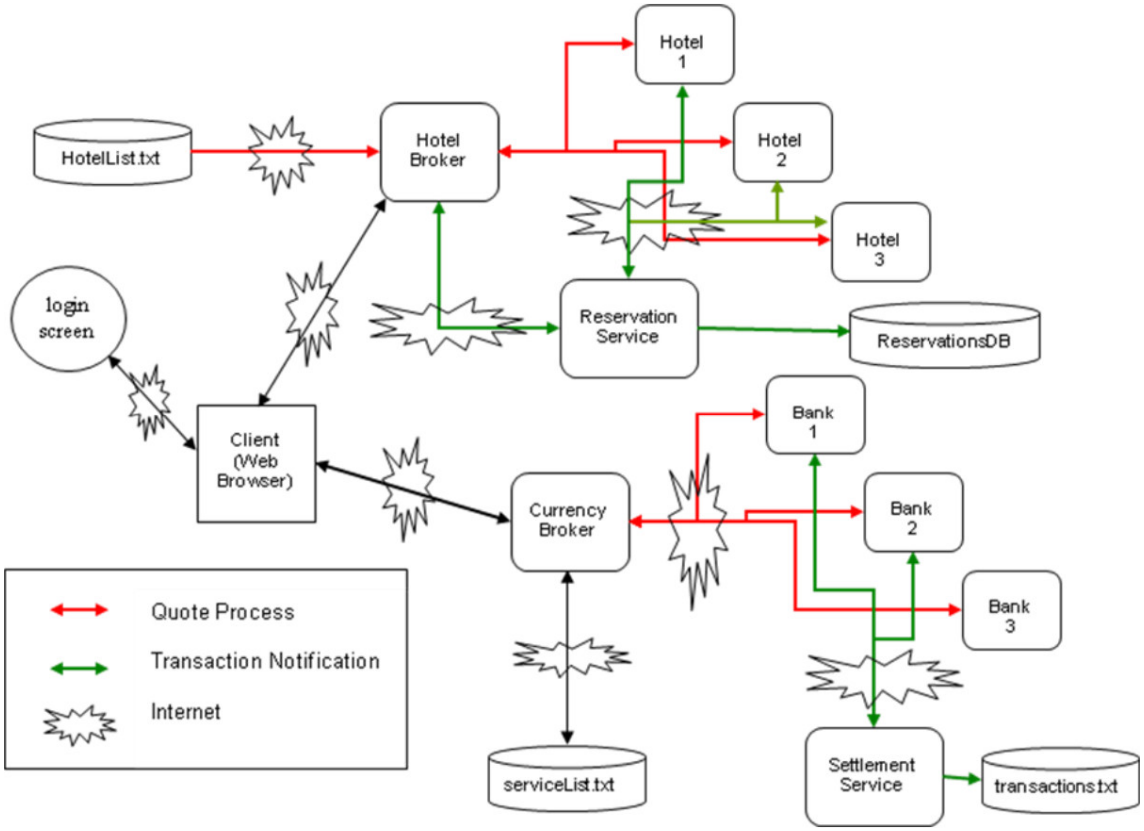

Figura 2.6: Exemplo SOA – Aplicação de Reservas de Hotel [13]

DEI-UC – Verificação e Validação (FMEA)

Uma das vertentes de estudo incidente sobre ambientes SOA consiste na sua verificação e validação devido aos desafios acrescidos neste tipo de ambiente face aos sistemas clássicos.

Os sistemas clássicos podiam ser testados mais facilmente e de forma mais completa num ambiente de qualidade ou pré-produção simulando as condições e cenários de produção. Num ambiente SOA a simulação das condições e cenários de produção pode não ser possível. Este facto deve-se quer à interação com serviços externos quer pela possibilidade de o comportamento do sistema poder ser determinado em *runtime* ou pelo facto de que ao contrário dos ambientes tradicionais poderem evoluir após se encontrarem em produção.

Em [14] são discutidos os desafios da aplicação de técnicas de validação e verificação em ambientes SOA, sendo igualmente estudada a adaptação do método *Failure Mode and Effects Analysis (FMEA)* de forma a propor uma técnica para a sua aplicação neste tipo de ambiente. No âmbito deste projeto foi considerada a arquitetura representada na Figura 2.7.

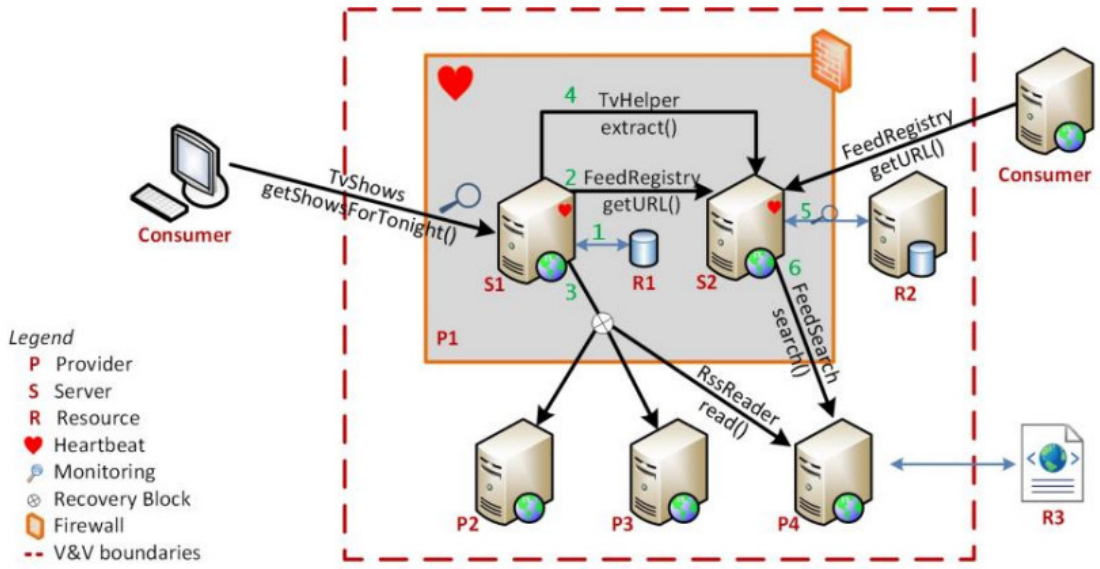

Figura 2.7: Exemplo SOA – Aplicação do método FMEA a ambiente SOA [14]

DEI-UC – Verificação e Validação (runtime)

Em [15] o objetivo da investigação centrou-se no desenvolvimento de uma abordagem de verificação e validação em tempo real para ambientes SOA. Tendo para o efeito sido empregue a arquitetura representada na Figura 2.8.

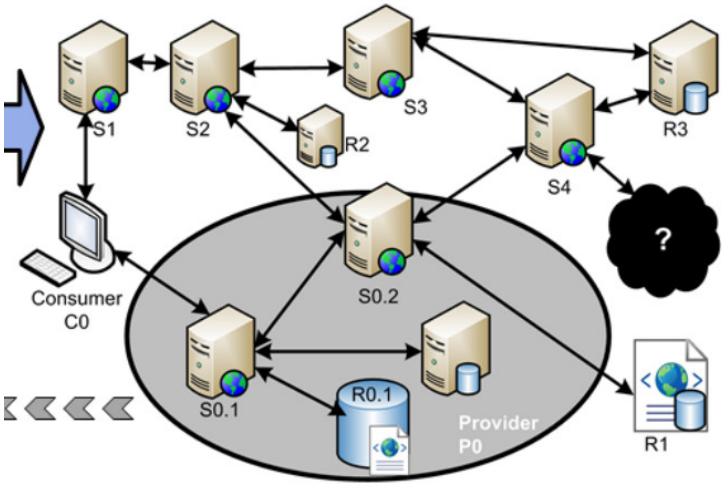

Figura 2.8: Exemplo SOA – Estudo Verificação e Validação de SOA em tempo real[15]

DCCUF[1]-CISUC[2] – A Service Discovery Approach for Testing Dynamic SOAs

Em [16] o objetivo do estudo consistiu na apresentação de uma solução para a descoberta automática de serviços constituintes de um SOA e a avaliação do seu comportamento durante a execução. Esta necessidade advém do facto de determinados serviços, por exemplo serviços críticos, terem de obedecer a determinadas regras e restrições e de um ambiente SOA ser composto por serviços que evoluem de forma transparente e que interagem entre si, não sendo possível que os seus utilizadores possuam um conhecimento completo dos serviços e da sua confiabilidade.

Na realização deste estudo foi utilizado um ambiente SOA de acordo com o esquema representado na Figura 2.9, que representa o sistema do ponto de vista do provedor de serviços. Os serviços constituintes do sistema são diferenciados em quatro tipos:

- Os serviços BPEL, *Business Process Execution Language* é um standard de orquestração de serviços em fluxos processuais, são os serviços do provedor e que controlam as atividades do sistema;
- Os serviços controlados que são também pertencentes ao provedor ou este tem acesso aos seus servidores;
- Os serviços atingíveis, que fazem parte do ambiente SOA, mas apenas são invocáveis pelo provedor não estando sob o seu controlo;
- Os serviços desconhecidos que fazem parte do ambiente SOA, mas a sua existência e detalhes não são do conhecimento do provedor.

[1] Department of Computer Science, University of Firenze
[2] Centro de Informática e Sistemas, Universidade de Coimbra

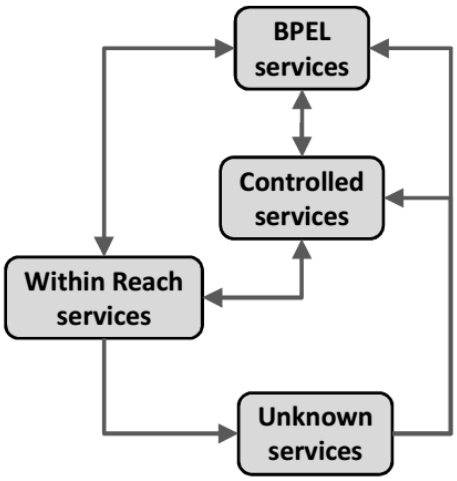

Figura 2.9: Exemplo SOA – Esquema Serviços de "A Service Discovery Approach for Testing Dynamic SOAs" [16]

DIOU[3]-CTICSTP[4] – Benchmarking of Web Services Platforms

O estudo [17] centra-se no benchmarking de *application servers* procurando apresentar formas passíveis de ser utilizadas para realização da avaliação recorrendo à implementação de um caso prático, representado na Figura 2.10, onde pode ser identificado o sistema a testar (SUT – System Under Test). Para a obtenção de resultados foram realizadas duas implementações utilizando *frameworks* diferentes, J2EE e .NET, e foi empregue o TCP-*App benchmark*.

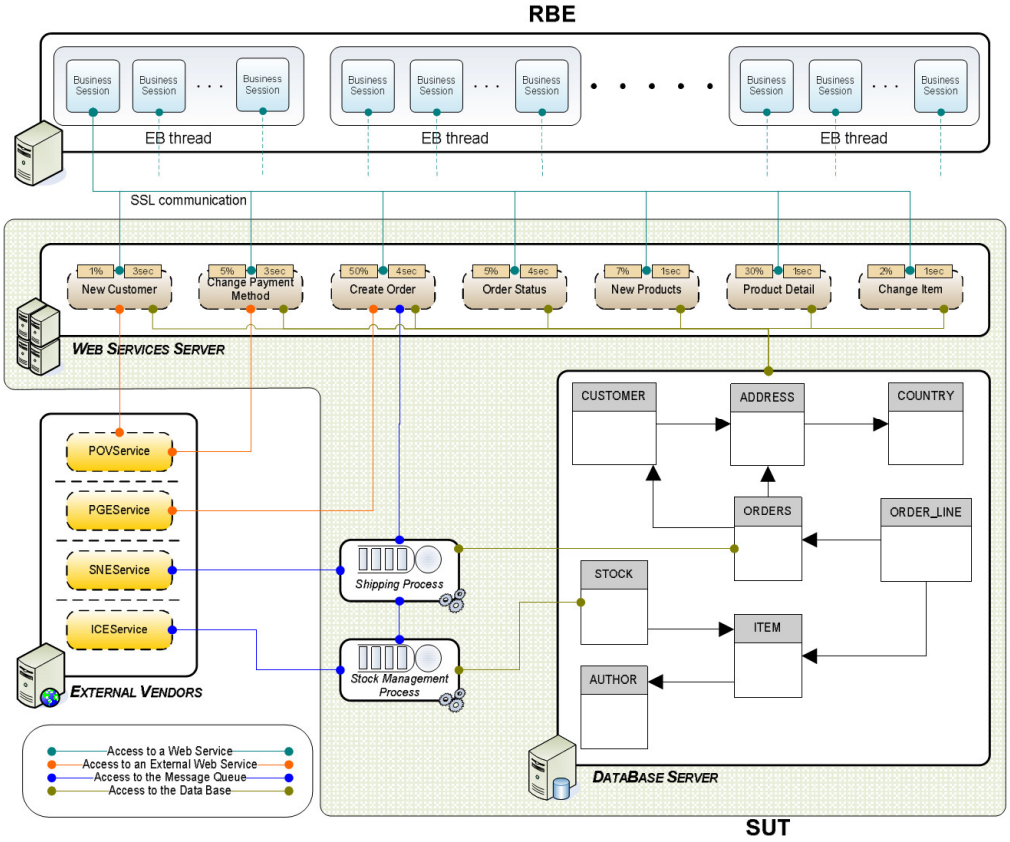

Figura 2.10: Exemplo SOA – TCP-App benchmark [17]

[3] Departamento de Informática, Oviedo University
[4] Centro Tecnológico de la Información y la Comunicación, Scientific and Technological Park

HSSP[5]– The practical Guide for SOA in Health Care

Um setor no qual a implementação de ambientes SOA pode trazer grandes vantagens é o da saúde, uma vez que coexistem diversos sistemas pré-existentes que implicaram investimentos consideráveis em hardware, software e aparelhos médicos, como por exemplo os diversos sistemas de diagnósticos adquiridos a fornecedores externos distintos que geram e guardam informação. A necessidade de evoluir tecnologicamente, de partilhar informação entre organizações distintas, como por exemplo entre organizações médicas e de investigação, são fatores incentivadores da mudança [18, 19, 20].

Em [20] o objetivo não é o estudo de ambientes SOA mas sim oferecer uma guia de orientação de alto nível para uma abordagem à implementação de sucesso de um ambiente SOA na área da saúde tendo em consideração as melhores práticas e fatores chave numa implementação SOA. Para o efeito deste estudo foi idealizada a instituição de saúde, a SampleHealth, cujo SOA é representado na Figura 2.11.

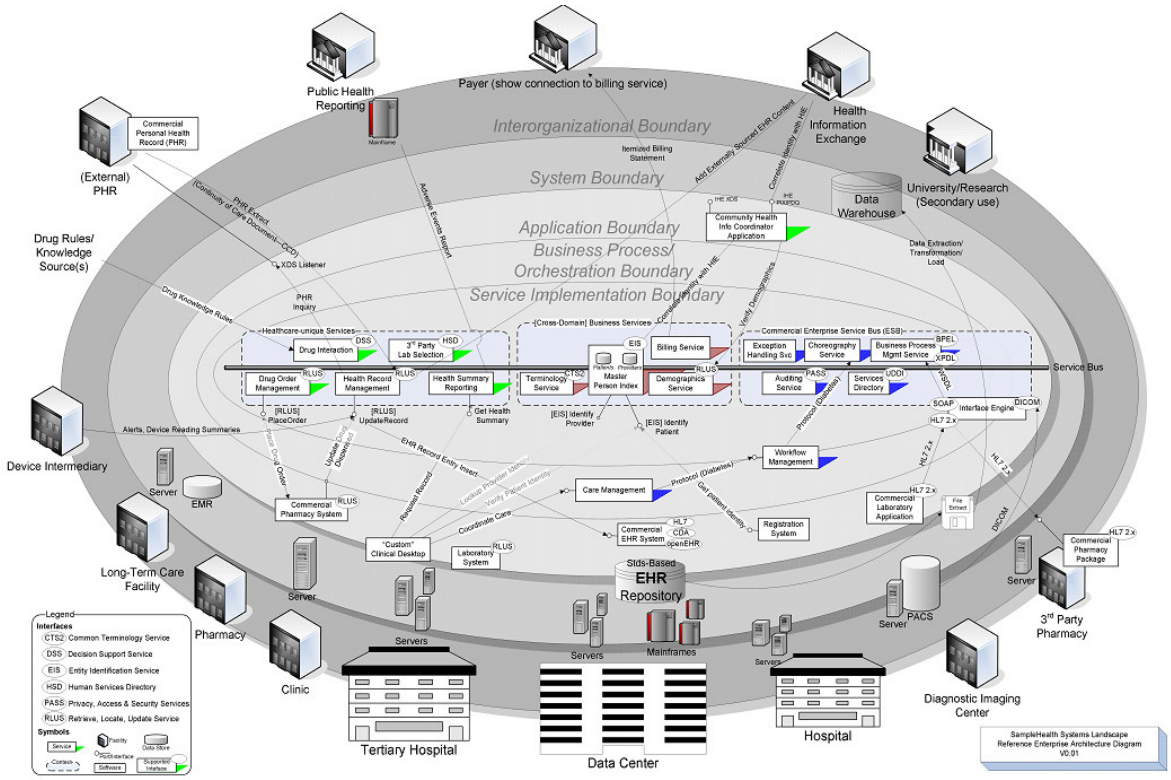

Figura 2.11: Exemplo SOA – Ambiente SOA da SampleHealth [20]

2.6.2. Ambientes de Organizações

UCM-Layer 7 Technologies

A Universidade do Centro Médico [21] em parceria com a empresa Layer 7 Technologies [22] adotou um ambiente SOA de forma a ser possível partilhar informação médica com a comunidade de investigação, assegurando, no entanto, todas as restrições inerentes a este tipo de informação, nomeadamente no que se refere à confidencialidade da informação. A infraestrutura implementada encontra-se representada na Figura 2.12 [19].

[5] Healthcare Services Specification Project – colaboração entre Health Level 7 e Object Management Group

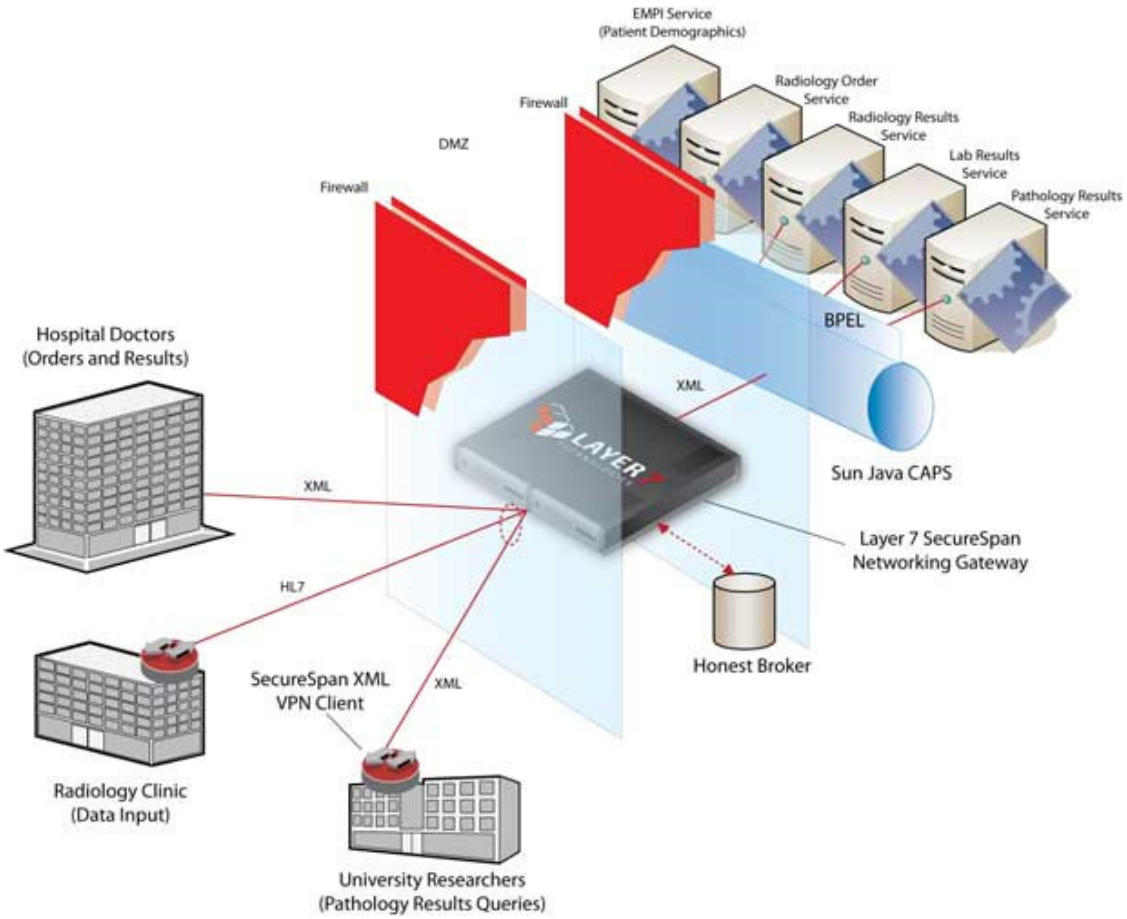

Figura 2.12: Exemplo SOA – Universidade do Centro Médico de Chicago [19]

Segurnet

O projeto Segurnet visa a partilha de informação entre companhias de seguros pertencentes à Associação Portuguesa de Seguradoras, sendo esta partilha efetuada através da troca de informação entre as diferentes companhias ou através de um repositório central de dados. Este é um exemplo de um sistema que tem evoluído para uma arquitetura SOA baseada em componentes *open source*. O projeto foi iniciado em 1999 tendo entrado em produção em 2000, tendo entre 2001 e 2009 sido alvo de várias evoluções incluindo a migração para uma arquitetura SOA [23].

Concur

A Concur, companhia online de gestão de despesas que possui vários sistemas pré-existentes, como por exemplo uma aplicação interna de recursos humanos e uma aplicação de gestão de relacionamento com os clientes, procurou melhorar a sua capacidade de análise das diferentes aplicações dos seus projetos internos. O sistema SOA implementado, Figura 2.13, foi executado de uma forma iterativa e flexível tendo recorrido a aplicações *open source* e com recursos internos. O processo foi iniciado com uma prova de conceito para apresentação aos executivos de forma a ganhar o seu apoio para o projeto [24].

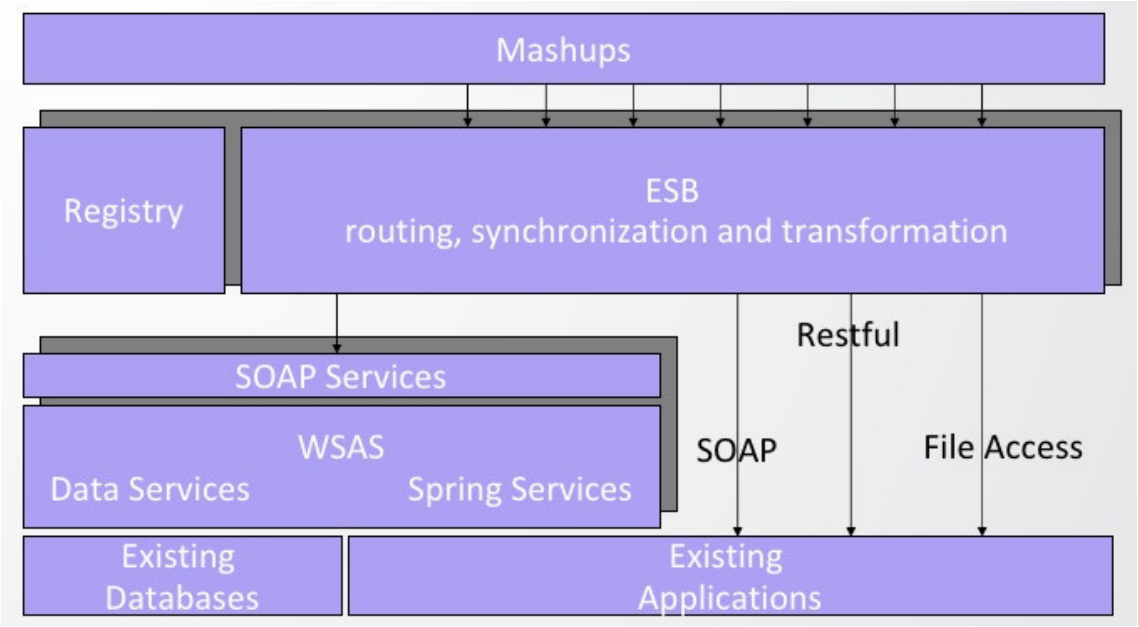

Figura 2.13: Exemplo SOA – Concur [24]

Governo Dinamarquês

O governo dinamarquês, com o intuito de simplificar os negócios eletrónicos, na perspetiva do negócio para o governo, procurando ao mesmo tempo realizar poupanças ao nível financeiro, recorreu a uma solução SOA. Os requisitos para a solução a implementar seriam que a entrega de mensagens fosse confiável e segura com mensagens encriptadas e assinadas e que conseguisse dar suporte a pequenos negócios. Algumas *frameworks* necessárias já se encontravam disponíveis, como foi o caso de uma rede de certificados digitais. Outros aspetos teriam de ser assegurados como o registro para pesquisa de serviços e um perfil de protocolos de transporte [24].

OVERSEE

O OVERSEE é um sistema de informação e suporte de operações marítimas que procura tornar mais eficientes e eficazes entidades públicas e privadas com responsabilidades no âmbito da segurança, salvaguarda da vida humana e proteção ambiental através da integração de um conjunto de fontes de dados que se encontram tipicamente em sistemas diferentes possibilitando uma tomada de decisão mais rápida e informada. Adicionalmente disponibiliza também funcionalidades associadas à fiscalização marítima, monitorização ambiental e capacidade de geração de conhecimento e antecipação de cenários de incidentes e ferramentas de apoio à decisão. Na sua implementação recorreu-se a serviços *RESTful* com mensagens XML e JSON e ao Apache Kafla como s*ervice bus* [25, 26].

First Citizens Bank

O First Citizens Bank é uma instituição que fornece vários serviços não só aos seus clientes como também a outras instituições, funcionando neste ultimo caso como uma empresa de serviços IT. De forma a disponibilizar os vários serviços e soluções para fora da sua rede interna e para novos mercados estes encontram-se assentes numa estrutura habilitada para SOA [18], [27].

OfficeMax

A OfficeMax implementou uma arquitetura centrada no *Enterprise Service Bus* tendo estabelecido uma política de priorização baseada no valor de negócio e não de IT que permitiu não só uma melhor performance como também uma redução de custos e obtenção de métricas em tempo real das transações de negócio [18, 28, 29].

2.6.3. Comparação e Análise dos exemplos

Na Tabela 2.1 são apresentadas as organizações, as áreas de negócio e o objetivo de cada ambiente apresentado nas secções anteriores.

Tabela 2.1: Exemplos SOA - Área de Negócio

Projeto/Organização	Investigação	Negócio	Objetivo
Open SOALab – Reservas Aéreas	Sim	Turismo	Fornecer ambiente de testes. Reservas aéreas
Open SOALab – Câmbios	Sim	Banca	Fornecer ambiente de testes. Câmbio de moeda
Open SOALab – Reservas de Hotel	Sim	Turismo	Fornecer ambiente de testes. Reservas de hotel
DEI-UC – Verificação e Validação (FMEA)	Sim		Estudo dos desafios da aplicação de técnicas de validação e verificação em SOA
DEI-UC – Verificação e Validação (runtime)	Sim		Desenvolvimento de abordagem de verificação e validação em tempo real para SOA
DCCUF-CISUC – A Service Discovery Approach for Testing Dynamic SOAs	Sim		Estudo sobre descoberta automática de serviços
DIOU-CTICSTP – Benchmarking of Web Services Platforms	Sim		Estudo sobre como realizar benchmarking de serviços web
HSSP– The practical Guide for SOA in Health Care	Sim	Saúde	Guia de orientação para abordagem à implementação de um SOA em saúde
UCM-Layer 7 Technologies	Não	Saúde	Plataforma de partilha de informação médica.
Segurnet	Não	Seguros	Plataforma de partilha de informação entre seguradoras
Concur	Não	Serviços	Sistema de análise
Governo Dinamarquês	Não	Governo	Simplificar os negócios eletrónicos, na perspetiva do negócio para o governo
OVERSEE	Não	Segurança	Plataforma de informação e suporte de operações marítimas
First Citizens Bank	Não	Banca	Fornecimento de serviços
OfficeMax	Não	Comércio	Sistema de análise em tempo real

Ao observar a informação presente na Tabela 2.1, podemos verificar que foi possível encontrar variados exemplos quer ao nível da investigação quer ao nível de ambientes reais pertencentes a organizações que optaram por este modelo como suporte ao seu negócio. Em termos de áreas de negócio, podemos verificar que são variadas, não sendo, contudo possível detetar uma tendência. Relativamente aos objetivos, podemos verificar a existência de vários casos que funcionam, pelo menos parcialmente, como plataformas de partilha de informações entre vários intervenientes, assim como o fornecimento de serviços.

Analisando e esquematizando a informação obtida dos exemplos no referente às características dos sistemas, Tabela 2.2 e Tabela 2.3,

Tabela 2.2: Exemplos SOA – Intervenientes

Projeto / Organização	**Serviços Externos**	**ESB**	**BPEL**	***Service Discovery***	**Servidor Mensagens**
Open SOALab – Reservas Aéreas	S				
Open SOALab – Câmbios	S			S	
Open SOALab – Reservas de Hotel	S			S	
DEI-UC – Verificação e Validação (FMEA)	S			S	
DEI-UC – Verificação e Validação (runtime)	S				
DCCUF-CISUC – A Service Discovery Approach for Testing Dynamic SOAs	S	N	S		
DIOU-CTICSTP – Benchmarking of Web Services Platforms	S	N			S
HSSP– The practical Guide for SOA in Health Care	S	S			
UCM-Layer 7 Technologies	S		S		
Segurnet					
Concur	N	S		S	
Governo Dinamarquês	S			S	
OVERSEE	S	S			
First Citizens Bank					
OfficeMax		S			

S – Característica presente | N – Característica ausente

Tabela 2.3: Exemplos SOA – Tecnologias

Projeto / Organização	SOAP	REST	XML	JSON	JAVA	.NET	PHP
Open SOALab – Reservas Aéreas					S	S	S
Open SOALab – Câmbios							S
Open SOALab – Reservas de Hotel							S
DEI-UC – Verificação e Validação (FMEA)							
DEI-UC – Verificação e Validação (runtime)							
DCCUF-CISUC – A Service Discovery Approach for Testing Dynamic SOAs							
DIOU-CTICSTP – Benchmarking of Web Services Platforms					S	S	
HSSP– The practical Guide for SOA in Health Care							
UCM-Layer 7 Technologies			S				
Segurnet			S		S		
Concur	S	S					
Governo Dinamarquês	S		S		S	S	
OVERSEE		S	S	S	S		
First Citizens Bank							
OfficeMax							

S – Característica presente	N – Característica ausente

Analisando a informação presente nas tabelas o ponto que mais salta à vista é a limitação da informação disponível sobre os sistemas implementados, nos quais apenas é afirmado que se trata de um ambiente SOA não sendo possível identificar as características do sistema ou quais as tecnologias envolvidas, o que impede a formação de uma opinião sobre se os sistemas podem na realidade ser considerados sistemas SOA ou se tratam apenas de sistemas que apresentam algumas características em comum com os ambientes SOA.

Um dos pontos fortes dos ambientes SOA é a capacidade de integração de diferentes sistemas e tecnologias. A integração de diferentes sistemas pode-se traduzir na integração de sistemas internos ou externos à organização. Observando os exemplos apresentados vemos que a integração de serviços externos é uma característica presente na maioria,

apenas quatro exemplos não recorrem a serviços externos ou essa informação não se encontra disponível.

Analisando os exemplos em relação à presença de diferentes tecnologias, a informação disponibilizada é limitada, pois a maioria dos exemplos não faz qualquer referência às tecnologias utilizadas e apenas quatro confirmam a utilização de um *service bus*.

Para conclusão deste capítulo podemos verificar que a informação disponibilizada ao público sobre sistemas SOA é bastante limitada, sendo que a partir da informação disponível não é possível tirar conclusões sobre os sistemas e a sua qualidade e se podem ser considerados como SOA. No entanto, analisando os exemplos e a informação disponibilizada verifica-se que os exemplos aparentemente mais completos apresentando um maior número de características associadas a SOA e diversidade tecnológica são os exemplos da Concur, do Governo dinamarquês e do OVERSEE.

3. Definição de um Caso de Estudo baseado em SOA - SOASales

Um dos objetivos deste projeto é a idealização de um caso de estudo de um ambiente SOA, nomeado de SOASales. A sua definição inclui não apenas a especificação do sistema, que deverá apresentar alguma complexidade e variedade de tecnologias, mas também a definição de uma organização com uma atividade numa área de negócio que justifique o sistema idealizado e providencie o SOASales de algum realismo.

Tendo em conta os objetivos definidos pensou-se numa área de negócio que permitiria a especificação de um sistema com as características pretendidas, tendo sido selecionada a área de vendas, em particular comércio eletrónico. Esta decisão baseou-se nos seguintes fatores:

- A organização, na qualidade de vendedor
    - poderia disponibilizar serviços aos seus clientes para a realização das diversas funcionalidades disponibilizadas;
    - internamente poderia recorrer a um *service bus* e poderiam ser utilizadas diferentes tecnologias de suporte para cada funcionalidade;
- A organização não assumindo apenas o papel de vendedor, mas também de revendedor
    - permitiria o consumo de serviços externos, sobre os quais a organização não tem conhecimento;
    - Adicionaria uma componente escalável, sendo possível adicionar novos fornecedores e serviços externos que podem representar implementações de tecnologias distintas.

A vertente de revendedor permite não só a integração de serviços externos sobre os quais não existe controlo ou conhecimento de implementação, mas também a idealização de outras organizações e serviços que possam ser controlados por exemplo para efeitos de testes do sistema, como foi o caso da organização Livros Online.

Partindo desta ideia inicial foi então definida uma organização principal, a Vendas Online, e optou-se por definir igualmente uma segunda organização que assumiria o papel de fornecer da Vendas Online, a Livros Online, assegurando assim não só pelo menos um fornecedor, mas também algum controlo sobre os serviços disponibilizados por este. Tendo as organizações definidas foi realizado o levantamento de requisitos para as organizações seguida da especificação dos serviços necessários e da arquitetura do sistema.

Nas secções seguintes será feita a apresentação das organizações, seguida pelo levantamento de requisito com recurso a casos de uso e por fim a especificação dos serviços e do sistema.

3.1. As Organizações

3.1.1. Vendas Online

A organização Vendas Online consiste numa empresa cuja área de negócio é a venda de artigos variados pela internet. Os produtos disponibilizados consistem não só em produtos da própria organização como também produtos de outras empresas, como é o caso da organização Livros Online, atuando como revendedor. A organização assume tanto o papel de revendedor para clientes particulares, vendendo diretamente a consumidores finais, como também de revendedor para outras organizações.

A organização disponibiliza uma aplicação web acessível aos seus clientes disponibilizando nesta um conjunto de funcionalidades complementares ao seu negócio, como sendo a pesquisa e visualização de produtos, o registro como clientes e a realização de encomendas.

Dos serviços disponibilizados pela organização alguns são privados e apenas acessíveis pela aplicação web ou outras aplicações internas à organização enquanto outros são públicos podendo ser consumido diretamente por clientes externos, nomeadamente outras organizações.

3.1.2. Livros Online

A organização Livros Online é de uma empresa cujo negócio é a venda de livros. De forma a dar suporte ao seu negócio, a organização disponibiliza vários serviços aos seus clientes para que estes possam pesquisar e consultar os livros disponíveis, realizar encomendas e consultar o estado das mesmas.

3.2. Análise de Requisitos Funcionais

Nesta secção será apresentada a análise de requisitos sob a forma de casos de uso para o sistema incluindo ambas as organizações mencionadas anteriormente, Vendas Online e Livros Online. Na Figura 3.1 encontram-se representados apenas os atores e as entidades envolvidas e as relações entre estes, sem ainda especificar casos de uso, que serão especificados nos diagramas seguintes.

Na figura pode ser identificada a organização Vendas Online que interage com vendedores externos, sendo um desses vendedores externos a organização Livros Online, e os atores principais, os clientes da organização Vendas Online que podem ser clientes web ou consumidores diretos dos serviços. A organização Vendas Online também se encontra representada como ator ao interagir como cliente da organização Vendas Online por outra via que não a extensão de um caso de uso.

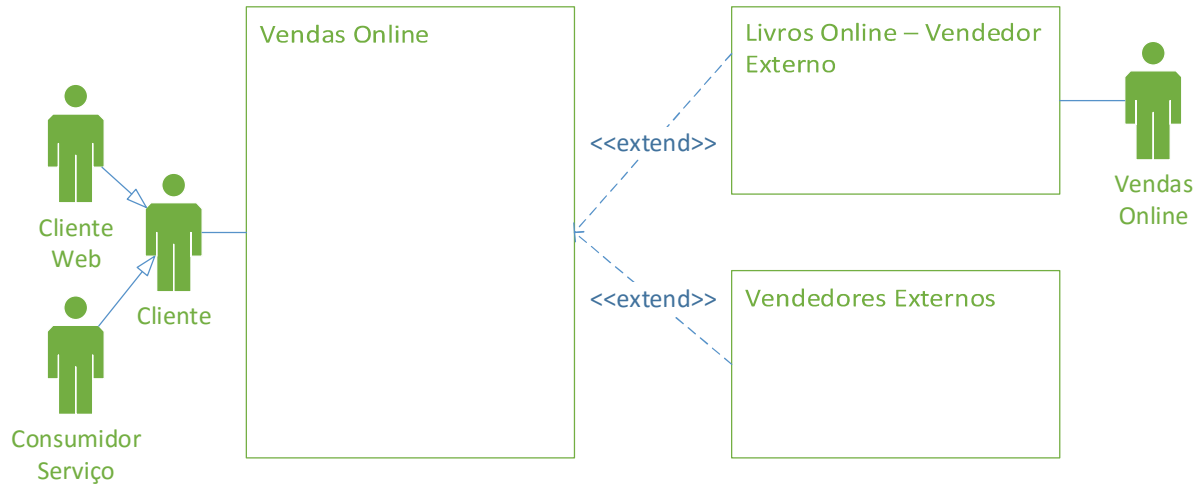

Figura 3.1: Casos de Uso – Visão Geral, atores e organizações

Nos diagramas seguinte são especificados os casos de uso, iniciando com uma visão geral, Figura 3.2.

O vendedor externo, Livros Online, é o único que se encontra representado nos diagramas seguintes por ser o único para o qual serão especificados requisitos. Adicionalmente do ponto de vista da entidade Vendas Online a organização Livros Online pode ser considerada representativa de outros vendedores externos.

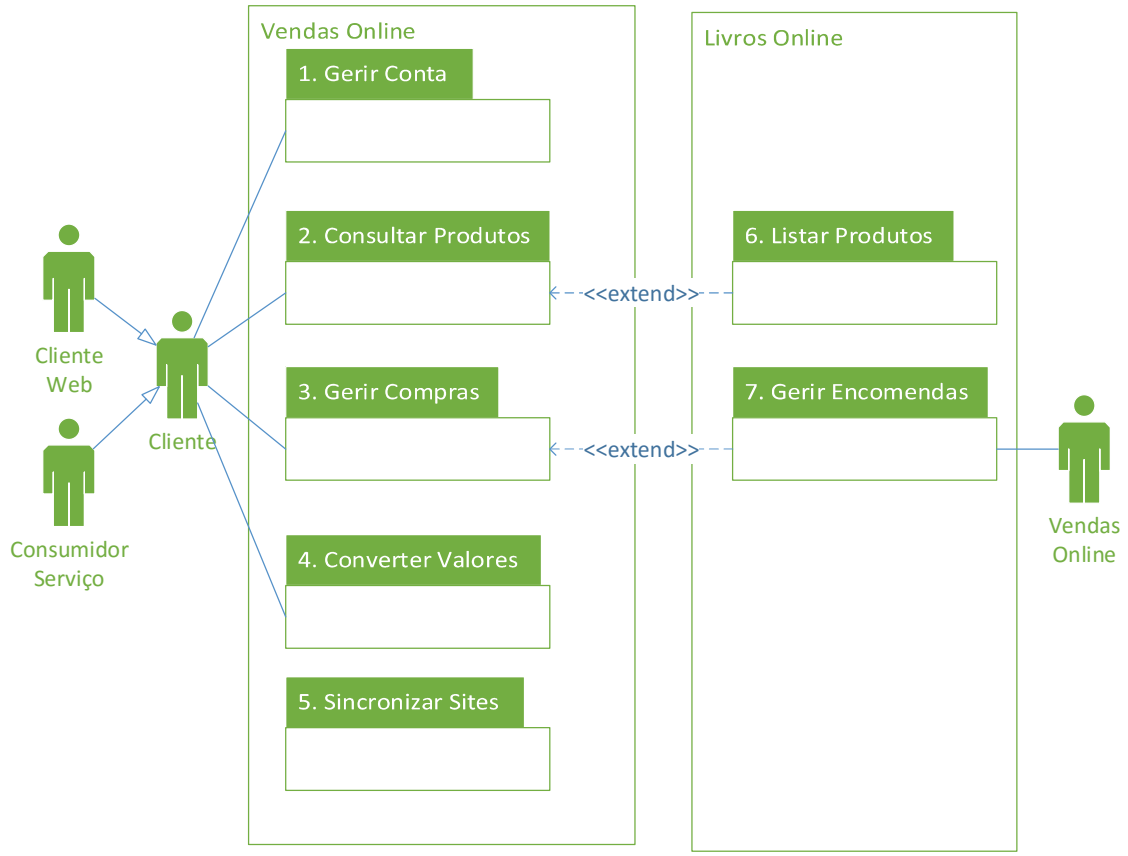

Figura 3.2: Casos de Uso – Visão Geral, casos de uso principais

De seguida serão concretizados e especificados com mais detalhes os diversos casos de uso sendo apresentada uma descrição de cada um e o diagrama do caso de uso.

1. **Gerir Conta** – O utilizador do sistema da organização Vendas Online deverá ter disponíveis funcionalidades que lhe permitam a gestão da sua conta de utilizador.
   - **1.1. Criar Conta** – O utilizador deverá ser capaz de criar uma conta de utilizador. O utilizador fornece a informação necessária à criação da conta e, após validação dos dados o sistema, criará a conta e retornará uma mensagem de sucesso; caso contrário retornará uma mensagem de erro. Esta funcionalidade apenas deve estar disponível para utilizadores da aplicação web.
   - **1.2. Atualizar Conta** – O utilizador deverá ser capaz de alterar/atualizar informação pessoal da sua conta. Esta funcionalidade apenas deve estar disponível para utilizadores da aplicação web.
   - **1.3. Visualizar Conta** – O utilizador deverá ser capaz de consultar a informação associada à sua conta de utilizador.
   - **1.4. Autenticar** – O utilizador deverá ser capaz de se autenticar usando a sua conta ganhando assim acesso funcionalidades adicionais.
   - **1.5. Visualizar Histórico** – O utilizador deverá ser capaz de consultar o histórico das compras realizadas.

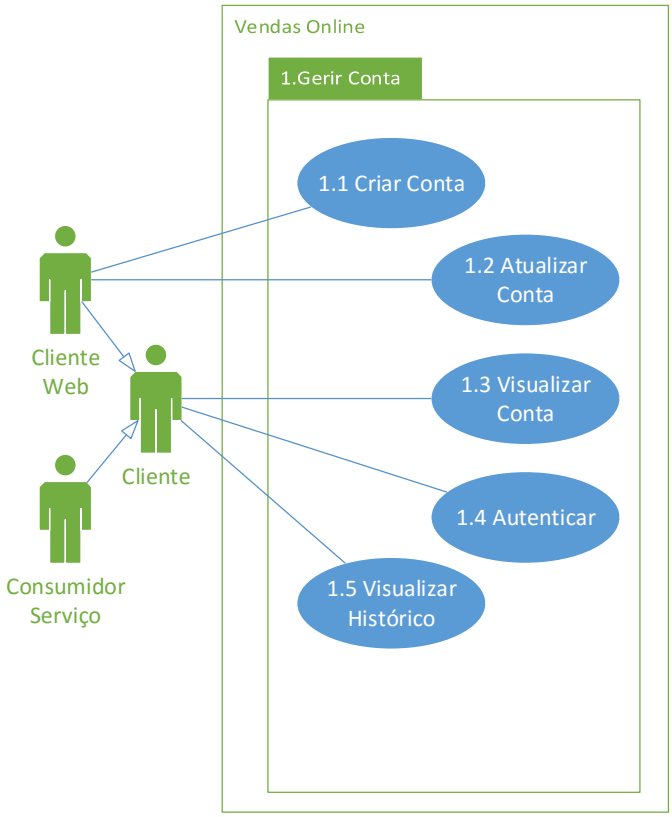

Figura 3.3: Casos de Uso – Gerir Conta

2. **Consultar Produtos** – O utilizador do sistema da organização Vendas Online deverá ter disponíveis funcionalidades que lhe permitam pesquisar e consultar produtos em que esteja interessado.
   - **2.1. Pesquisar Produtos** – O utilizador deverá ser capaz de pesquisar produtos com base num conjunto de palavras.
   - **2.2. Visualizar detalhes** – O utilizador deverá poder obter uma informação mais detalhada de um produto específico. O pedido para obtenção dos detalhes deve conter o identificador do produto e do seu vendedor.
   - **2.3. Comparar produtos** – O utilizador deverá poder selecionar até 4 produtos para obter informação mais detalhas dos mesmos para comparação, sendo a informação dos produtos devolvida numa única resposta.
   - **2.4. Pesquisar fontes em simultâneo** – O sistema deverá ser capaz de realizar pesquisas internas e em várias fontes externas em simultâneo.
   - **2.5. Combinar Resultados** – O sistema deverá ser capaz de combinar os resultados obtidos interna e externamente numa única resposta.
   - **2.6. Pesquisar Produtos Próprios** – O sistema deverá ser capaz de realizar pesquisas internamente, obtendo uma listagem de produtos próprios correspondentes às palavras inseridas na pesquisa. O sistema deverá retornar os produtos que possuam as palavras de pesquisa no nome, descrição ou no campo de informação genérica.
   - **2.7. Pesquisar Produtos Externos** – O sistema deverá ser capaz de aceder a fontes externas para obter produtos correspondentes aos critérios de pesquisa.
   - **2.8. Obter Detalhes Produto Próprio** – O sistema deverá ser capaz de obter os detalhes de um produto da própria organização.
   - **2.9. Obter Detalhes Produto Externo** – O sistema deverá ser capaz de obter os detalhes de um produto de uma fonte externa.
   - **2.10. Obter Detalhes em Simultâneo** – O sistema deverá ser capaz de disponibilizar em simultâneo os detalhes de até 4 produtos, podendo estes ser uma combinação de produtos da própria organização e de fontes externas. O sistema deverá realizar os pedidos de detalhes em simultâneo.

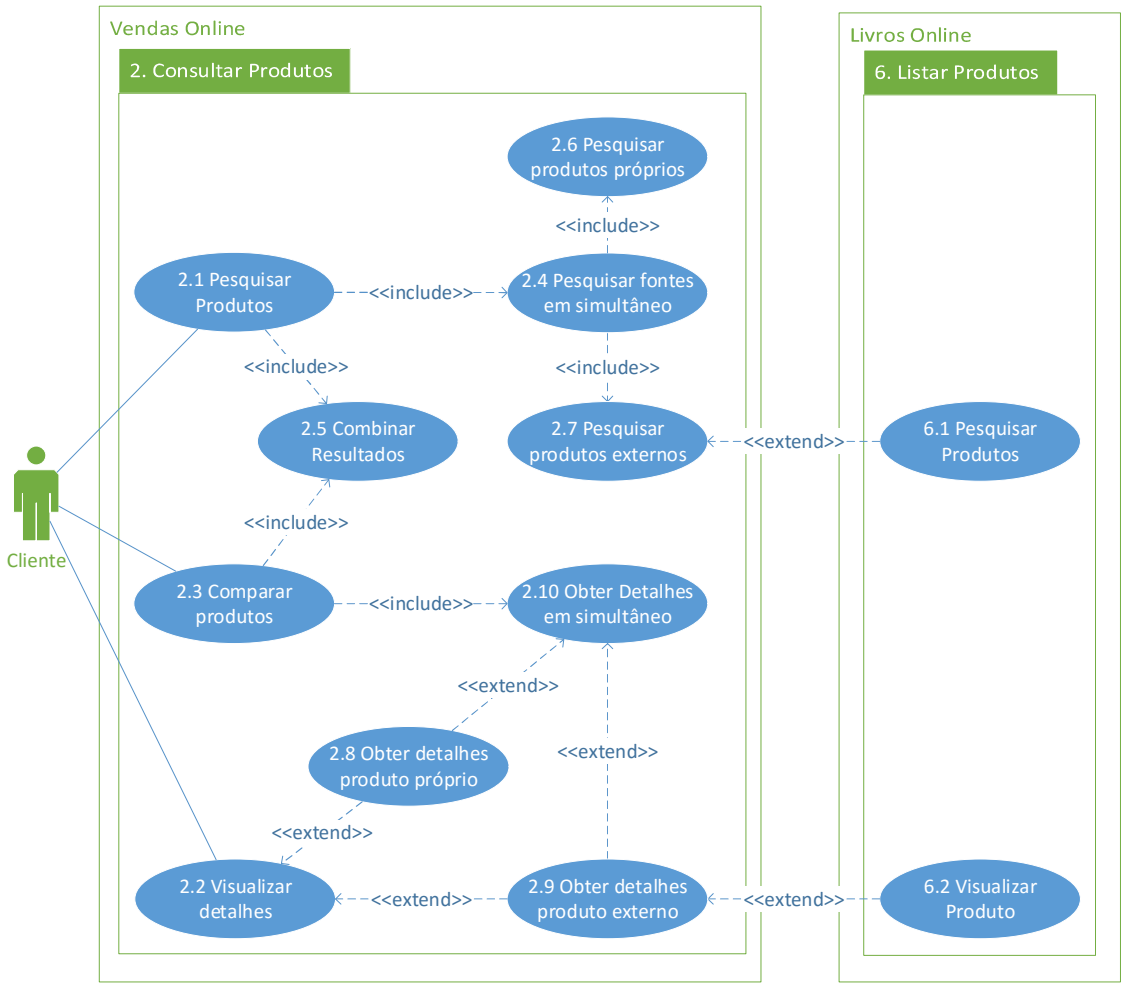

Figura 3.4: Casos de Uso – Consultar Produtos

3. **Gerir Compras** – O utilizador do sistema da organização "Vendas Online" deverá ter disponíveis funcionalidades que lhe permitam gerir e efetuar compras
   - **3.1. Selecionar produtos** – O utilizador deverá poder selecionar produtos para adicionar à sua lista de compras.
   - **3.2. Visualizar Produtos** – O utilizador deverá poder consultar a lista de produtos contidos na sua lista de compras.
   - **3.3. Eliminar Produto** – O utilizador deverá poder remover produtos da sua lista de compras que tenham sido adicionados anteriormente.
   - **3.4. Realizar Compra** – O utilizador deverá poder concretizar a compra dos produtos presentes na sua lista de compras realizando uma encomenda.
   - **3.5. Consultar Encomenda** – O utilizador deverá poder consultar o estado de uma encomenda que tenha efetuado.
   - **3.6. Cancelar Encomenda** – O utilizador deverá ter disponível uma opção para cancelar a encomenda efetuada. Esta opção apenas deverá estar disponível temporariamente, deixando de ser possível após o processamento e envio da mesma.

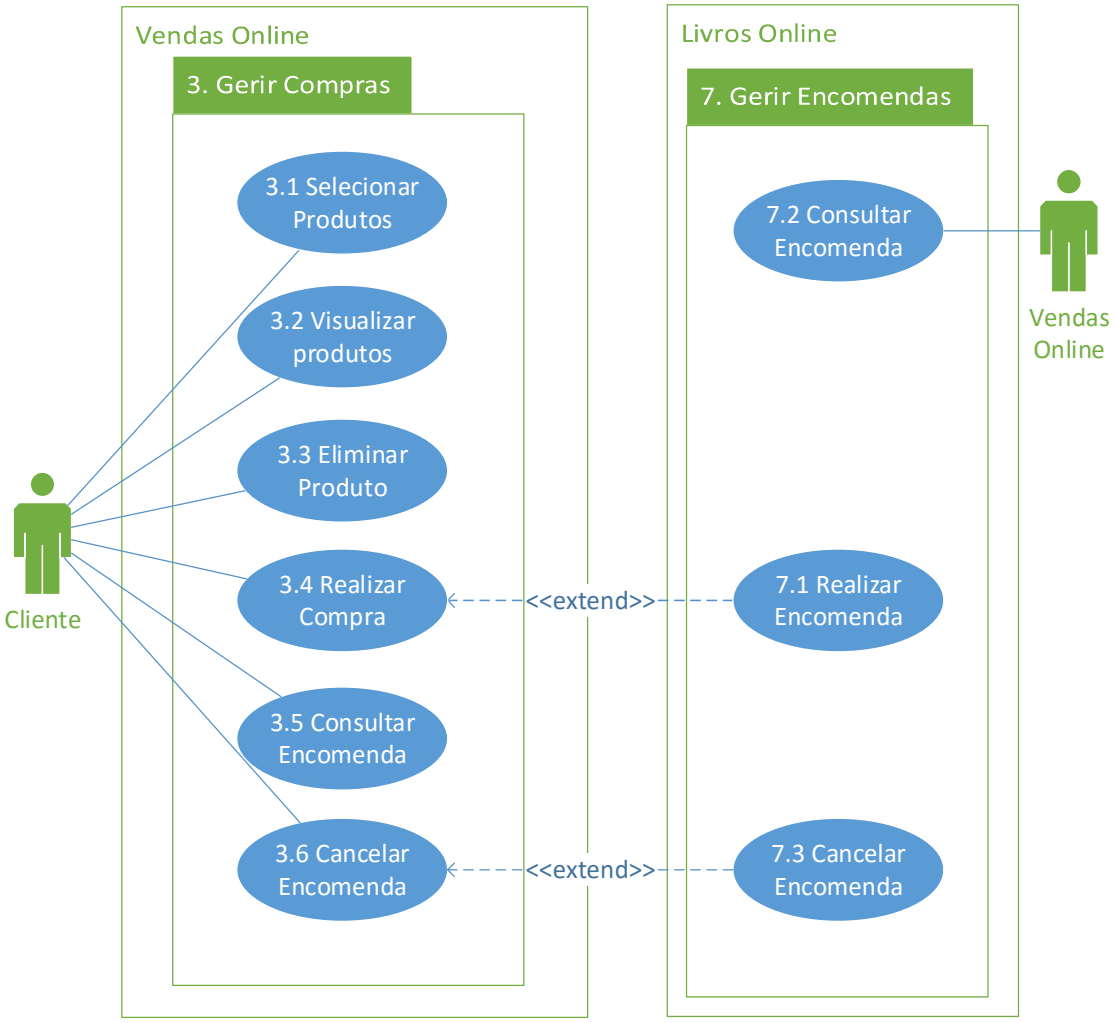

Figura 3.5: Casos de Uso – Gerir Compras

4. **Converter Valores –** O utilizador do sistema da organização Vendas Online deverá ter disponível uma funcionalidade que lhe permita realizar a conversão de valores para outras moedas.
   - **4.1. Converter Preços –** O utilizador deverá ter disponível uma funcionalidade que lhe permita fazer a conversão dos preços apresentados, de produtos individuais ou de um conjunto de produtos, para outras moedas.
   - **4.2. Consultar Registry** – O sistema deverá ser capaz de estabelecer ligações com serviços de *registry* e obter informação sobre serviços de conversão de moeda.
   - **4.3. Estabelecer Ligação** – O sistema deverá ser capaz de selecionar um dos serviços encontrados e estabelecer uma ligação com o serviço e fazer uso do serviço.
5. **Sincronizar Sites** – O sistema Vendas Online deverá possuir mecanismos de redundância ao nível dos seus serviços e base de dados. Para esse efeito a organização deverá duplicar os seus recursos em dois sites distintos.
   - **5.1. Detetar Estado** – O sistema deverá ser capaz de detetar o estado dos sites, identificando se os sites se encontram operacionais e acessíveis.
   - **5.2. Sincronizar Dados** – O sistema devera ser capaz de fazer a sincronização dos dados entre os sites. Deverá proceder à sincronização dos dados de uma forma periódica.

5.3. Alternar Site – O sistema deverá ser capaz de passar a operação para o site secundário quando deteta que o site primário não se encontra operacional.

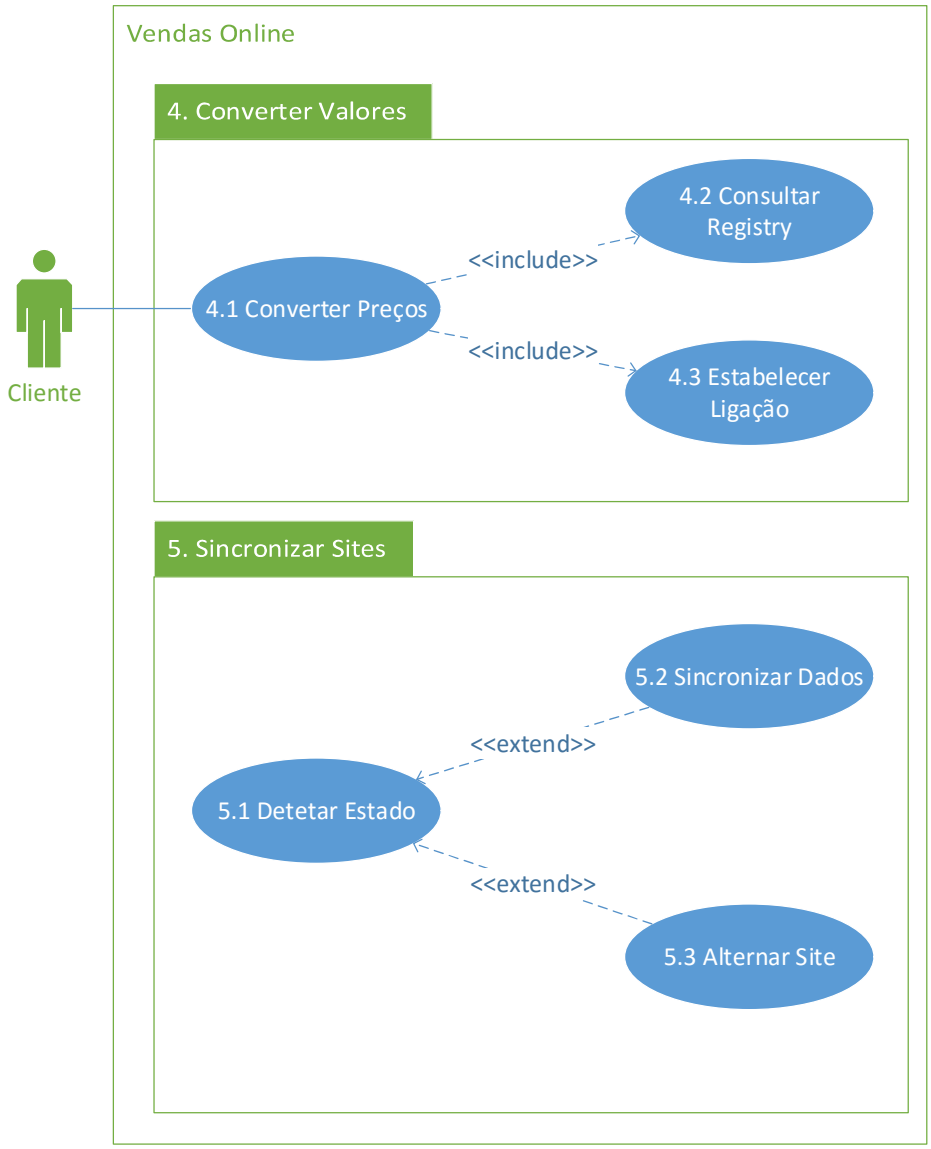

Figura 3.6: Casos de Uso – Converter Valores e Sincronizar Sites

A organização Livros Online deverá disponibilizar as seguintes funcionalidades, já representadas nos esquemas anteriores:

6. **Listar Produtos** – deverá ser possível aos clientes da organização Livros Online fazer a pesquisa de produtos e consulta de detalhes dos mesmos
   - **6.1. Pesquisar Produtos** – O Utilizador deverá ser capaz de pesquisar produtos, obtendo uma listagem de produtos correspondente aos critérios introduzidos
   - **6.2. Visualizar Produto** – O Utilizador deverá poder consultar informação detalhada de um produto específico
7. **Gerir Encomendas** – deverá ser possível aos clientes da organização Livros Online fazer a gestão das suas compras
   - **7.1. Realizar Encomenda** – O Utilizador deverá poder realizar uma encomenda de vários produtos

7.2. Consultar Encomenda – O Utilizador deverá poder consultar a sua encomenda e visualizar os detalhes e estado da mesma
7.3. Cancelar Encomenda – O Utilizador deverá poder cancelar uma encomenda que ainda não tenha sido processada

3.3. Requisitos Tecnológicos

Nesta secção serão apresentados os requisitos tecnológicos definidos para o sistema. Para a definição destes requisitos foram considerados o objetivo deste projeto, as características associadas a ambientes SOA e as características identificadas nos diferentes exemplos apresentados no capítulo 2.6.

Uma das principais características associadas aos SOA é a sua capacidade de integrar aplicações, que podem ser internas ou externas à organização, eventualmente desenvolvidas com recurso a diferentes tecnologias. Nos exemplos apresentados e recorrendo à Tabela 2.3 podemos ver que a maioria dos exemplos consome serviços externos. Nos exemplos que apresentam essa informação os serviços consumidos são REST ou SOAP desenvolvidos com recurso às linguagens JAVA ou C# .NET.

A utilização de um *service bus* é uma característica muito associada aos ambientes SOA, pelo menos na literatura, no entanto verifica-se que este nem sempre se encontra presente.

De acordo com esta informação e com o intuito de implementar um sistema abrangente e representativo foram definidos os seguintes requisitos:

1. **Tecnologias diferenciadas**
   1.1. A implementação do SOASales deverá apresentar tecnologias e plataformas diferenciadas.
2. **Serviços diferenciados**
   2.1. O sistema deverá consumir serviços internos, da própria organização, e externos, pertencentes a outras organizações. Pelo menos um dos serviços deverá estar hospedado num servidor distinto do sistema.
   2.2. Entre os serviços consumidos deverão estar presentes serviços REST e SOAP.
   2.3. Entre os serviços consumidos deverão estar presentes serviços desenvolvidos com recurso a C# .NET e Java.
3. ***Service Bus***
   3.1. O sistema deverá ter na sua constituição um *service bus*.
   3.2. O *service bus* deverá ser responsável por alguma lógica de negócio, não devendo ser responsável apenas por reencaminhamento de pedidos.

3.4. Especificação dos Serviços

Nesta secção serão apresentados o sistema e os serviços resultantes da concretização dos requisitos apresentados na secção anterior.

Considerando os casos de uso apresentado determinou-se quais os serviços e operações a disponibilizar pela organização Vendas Online que podem ser consultado na Tabela 3.1 e Tabela 3.2, apresentadas de seguida.

Tabela 3.1: Serviços e Operações Privados da organização Vendas Online

Serviço	Operação	Descrição da operação
AccountService	CreateAccount	Permite ao utilizadores registar-se como cliente da organização
	UpdateAcocunt	Permite ao utilizador atualizar informação da sua conta de utilizador

Tabela 3.2: Serviços e Operações Públicas da organização Vendas Online

Serviço	Operação	Descrição da operação
AccountInfoService	ViewAccount	Permite ao utilizador consultar a informação associada à sua conta
	ViewPurchaseHistory	Permite ao utilizador obter uma listagem de produtos adquiridos
ProductsService	Search	Permite ao utilizador a realização de pesquisas de forma a encontrar os produtos pretendidos. Este serviço apenas pesquisa os produtos da organização Vendas Online
	Get	Permite ao utilizador obter os detalhes de um produto específico da organização Vendas Online
PurchaseService	AddToCart	Permite ao utilizador selecionar um produto para compra
	ViewCart	Permite ao utilizador obter uma listagem de produtos selecionados para compra
	RemoveFromCart	Permite ao utilizador remover um ou mais itens da sua listagem de produtos para compra
	MakePurchase	Permite ao utilizador realizar a encomenda dos itens presentes na sua lista de compras
	TrackPurchase	Permite ao utilizador obter informação sobre compras efetuadas que ainda não se encontram concluídas
	CancelPurchase	Permite ao utilizador cancelar a sua encomenda

Para a organização Livros Online os serviços e operações determinados são os apresentados na Tabela 3.3.

Tabela 3.3: Serviços e Operações da organização Livros Online

Serviço	Operação	Descrição da operação
ListingService	SearchBook	Permite ao utilizador a realização de pesquisas de forma a encontrar os produtos da organização Livros Online pretendidos.
	ViewBook	Permite ao utilizador obter os detalhes de um produto específico da organização Livros Online
OrderService	MakeOrder	Permite ao utilizador realizar uma encomenda
	TrackOrder	Permite ao utilizador obter informação sobre as suas encomendas em progresso
	CancelOrder	Permite que o utilizador em determinadas condições cancele uma encomenda

3.5. Especificação do Sistema

Considerando a organização Vendas Online e os serviços e operações apresentados na Tabela 3.1 e Tabela 3.2, verificamos que estes não são suficientes para cumprir todos os casos de uso descritos, uma vez que apenas consideram produtos da própria organização e o objetivos é conjugar os produtos da própria organização com produtos de outras organizações.

Para responder a essas necessidades foi idealizado uma arquitetura orientada a serviços na qual o sistema vai ser responsável pela disponibilização das funcionalidades adicionais e a orquestração dos vários serviços. De seguida iremos apresentar uma visão geral do sistema seguida da especificação dos cenários implementados. Os restantes cenários podem ser consultados no anexo A.

Na Figura 3.7 podemos observar um esquema com uma visão geral do sistema idealizado para satisfazer as necessidades da organização Vendas Online.

A organização possui um *service bus* que serve de ponto de ligação entre a organização e o exterior, sendo o ponto de entrada do sistema. O *service bus* é responsável não só pelo reencaminhamento de pedidos, mas também por alguma da lógica de negócio.

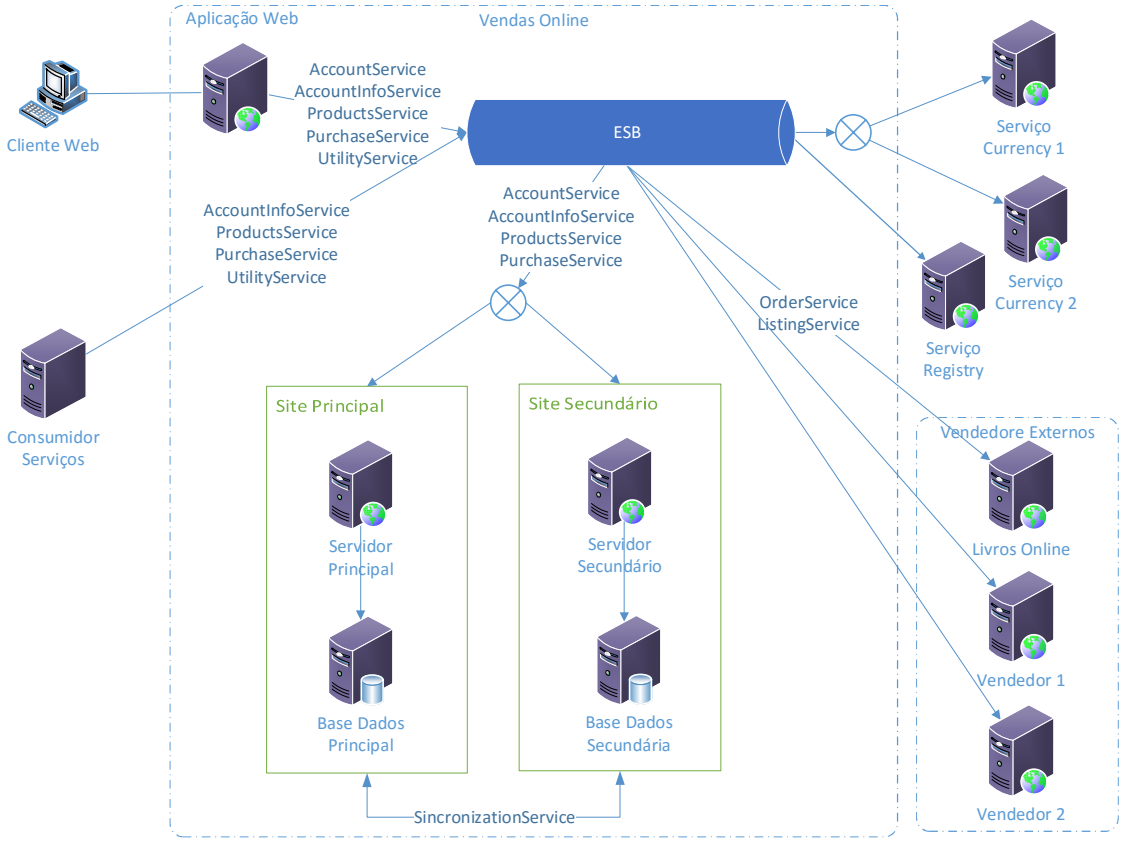

Figura 3.7: Ambiente SOA Vendas Online - Visão Geral

No esquema seguinte apresentamos uma versão mais detalhada do esquema anterior evidenciando os serviços de cada organização. Por uma questão de simplificação do esquema não se encontra representada a duplicação da estrutura por duas localizações.

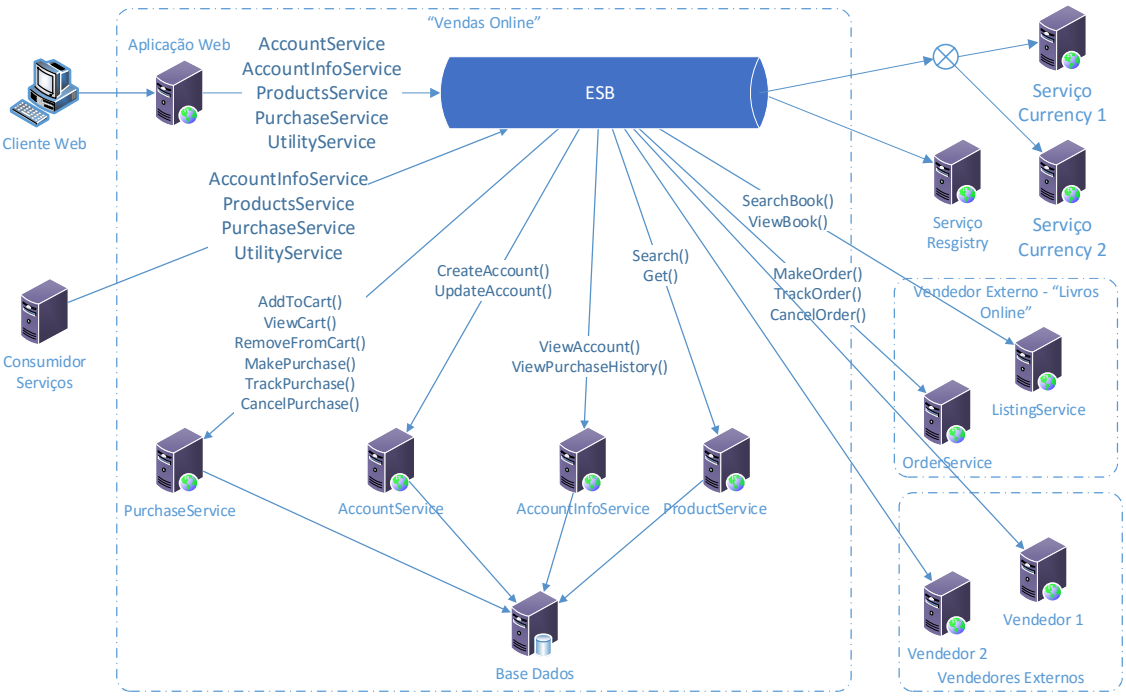

Figura 3.8: Ambiente SOA Vendas Online - Visão Geral Serviços

De seguida iremos particularizar alguns fluxos, nomeadamente para o serviço ProductsService e os cenários de Pesquisa, visualização e Comparação de produtos apresentando esquemas ilustrativos e descrição dos mesmos. Iniciando com o cenário da Pesquisa de produtos cujos esquemas são apresentados na Figura 3.9 e Figura 3.10.

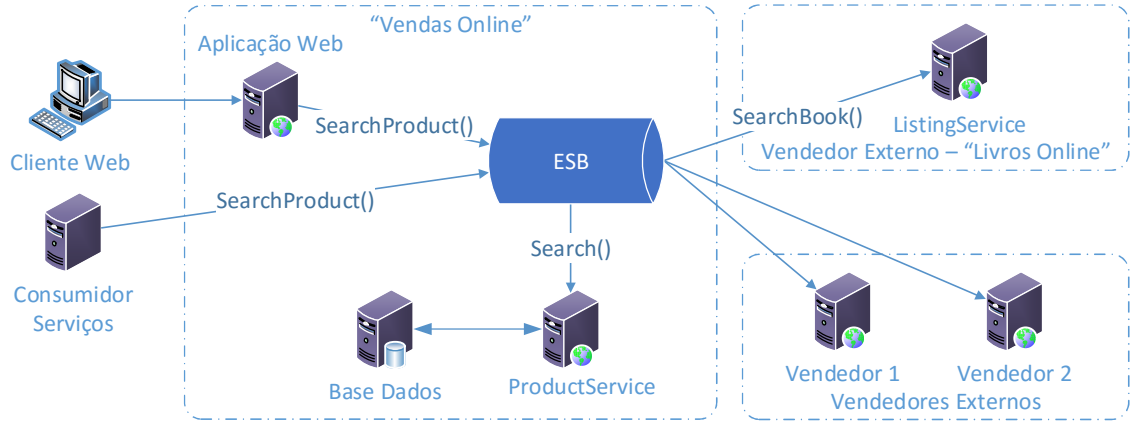

Figura 3.9: Orquestração de Serviços – Pesquisa de Produtos

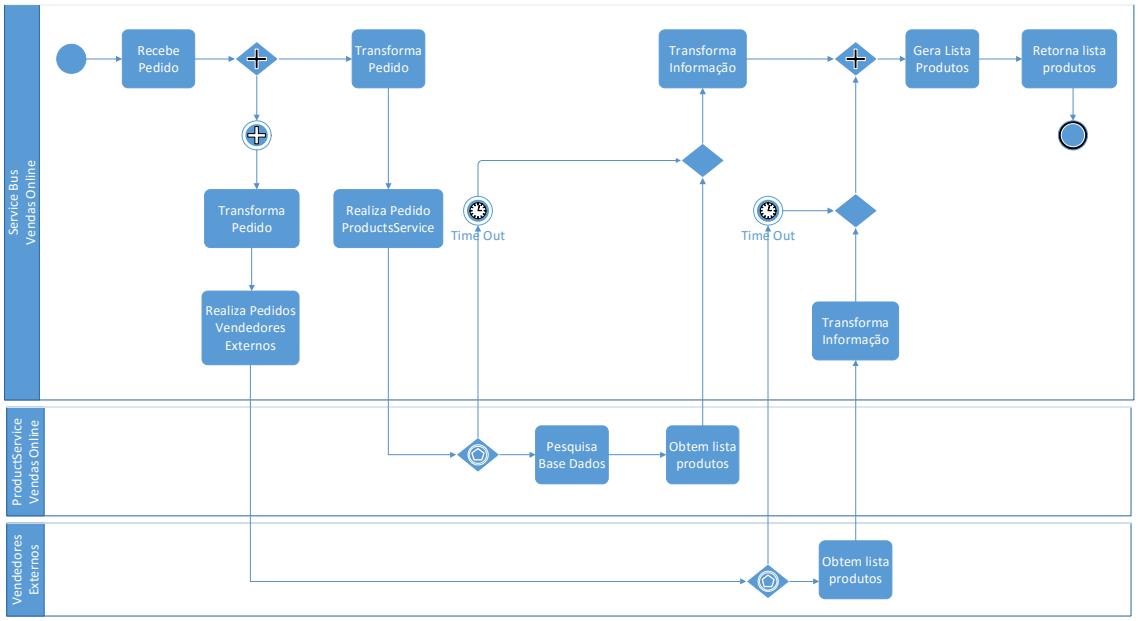

Figura 3.10: Diagrama BPMN – Pesquisa Produtos

Observando o cenário representado nos esquemas podemos verificar que os clientes da organização Vendas Online podem aceder à pesquisa de produtos de duas formas distintas quer por meio da aplicação web que realiza posteriormente o pedido ao *service bus* ou fazer o pedido diretamente a este, sendo que o pedido deverá conter as palavras a pesquisar.

O *service bus* ao receber o pedido realiza o seu processamento de forma paralela para o ProductsService da organização e para os vendedores externos incorporados no fluxo. O mesmo pedido será reencaminhado simultaneamente para todos os serviços fazendo o processamento necessário para que a mensagem tenha o formato correto para cada um.

O *service bus* aguarda pelas respostas de todos os serviços ou por um *time out*, agregando as respostas obtidas com sucesso numa única, tendo as respostas de cada serviço sido previamente transformadas para um formato comum.

A resposta com os resultados combinados da pesquisa realizada nos vários serviços é então retornada para o cliente.

Nos próximos esquemas, Figura 3.11 e Figura 3.12, está representado o cenário de visualização do detalhe de um produto.

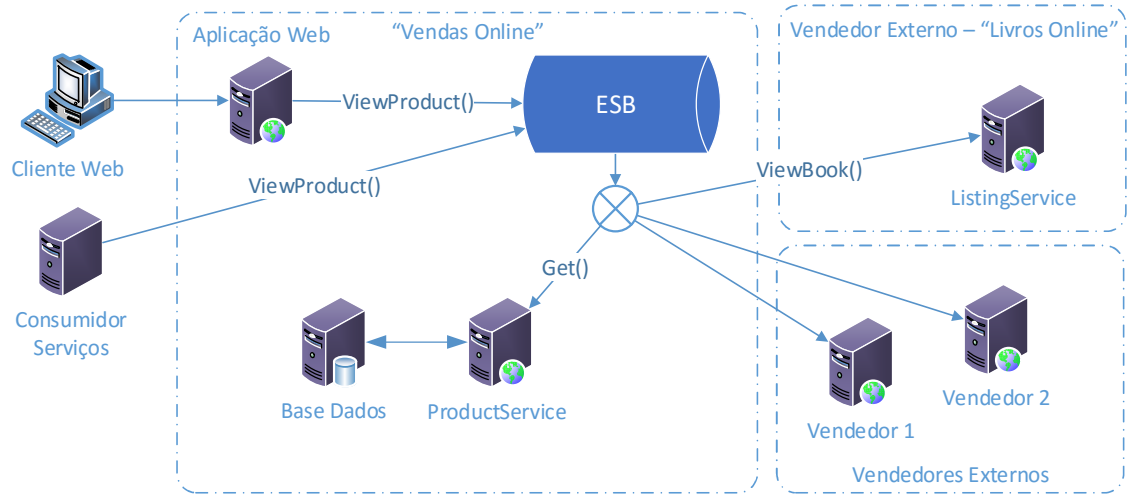

Figura 3.11: Orquestração de Serviços – Detalhe de Produto

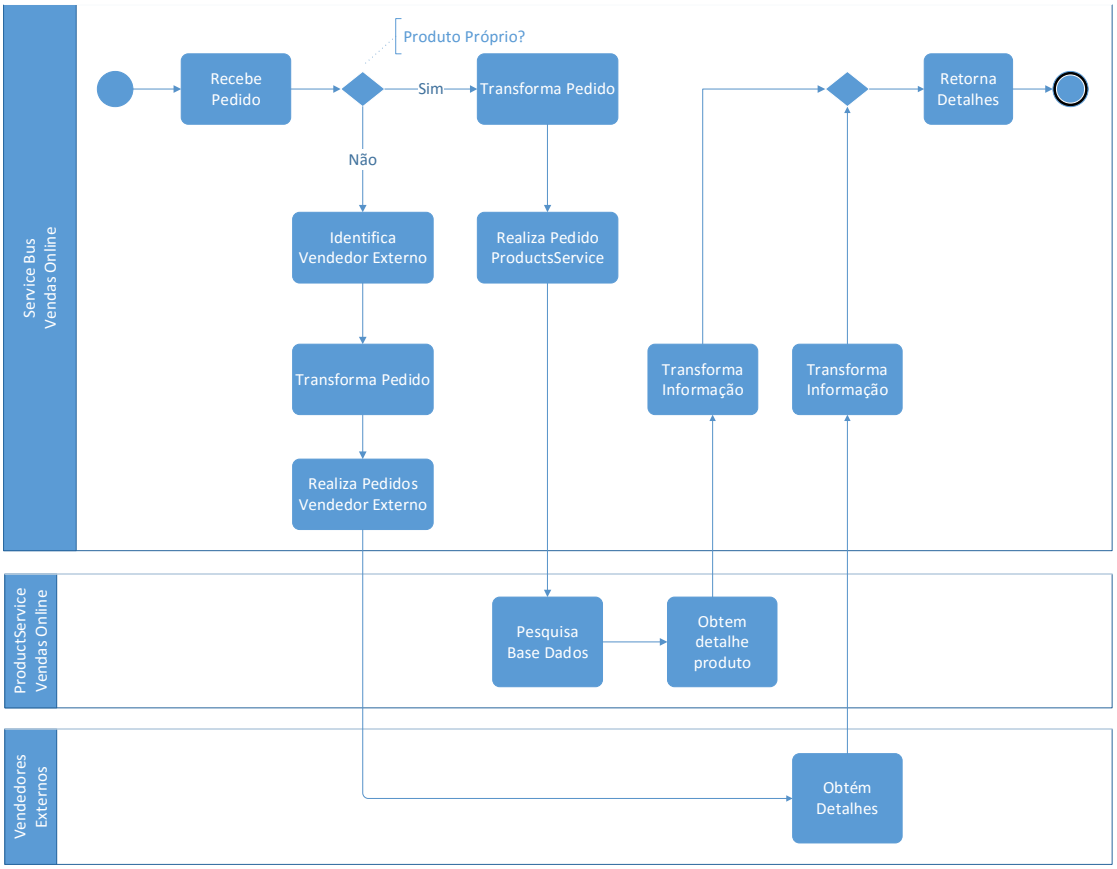

Figura 3.12: Diagrama BPMN – Detalhe de Produto

Observando os esquemas podemos verificar que tal como no caso anterior o pedido de detalhes de um produto pode ser feito por meio da aplicação web ou diretamente ao *service Bus*, o pedido deve conter a informação identificadora do produto e do seu vendedor.

O *service bus* ao receber um pedido começa por identificar o vendedor do produto, e se este se trata de um produto da própria organização ou de uma organização externa. De acordo com esta determinação o fluxo seguirá percursos alternativos.

Caso de trate de um produto próprio, o *service bus* irá realizar um pedido ao ProductsService da organização fazendo a transformação necessária ao pedido recebido para realizar a chamada ao método pretendido do ProductsService.

O ProductsService por sua vez ao receber um pedido de detalhe de produto vai pesquisar o identificador recebido na base de dados e retornar, caso exista, a informação do produto correspondente.

Caso se trate de um produto de uma organização externa, o *service bus* identifica a organização e transforma o pedido recebido para que este tenha o formato necessário para a realização da chamada à organização externa.

Após a *service bus* receber a resposta do serviço invocado processa e transforma a mensagem recebida e retorna a resposta.

Nos próximos esquemas, Figura 3.13 e Figura 3.14, está representado o cenário de visualização do detalhe de um produto.

Observando a Figura 3.13 e comparando com a Figura 3.11, verificamos que são semelhantes, pois apesar do ponto de entrada ser diferente os serviços envolvido na execução do pedido são os mesmos.

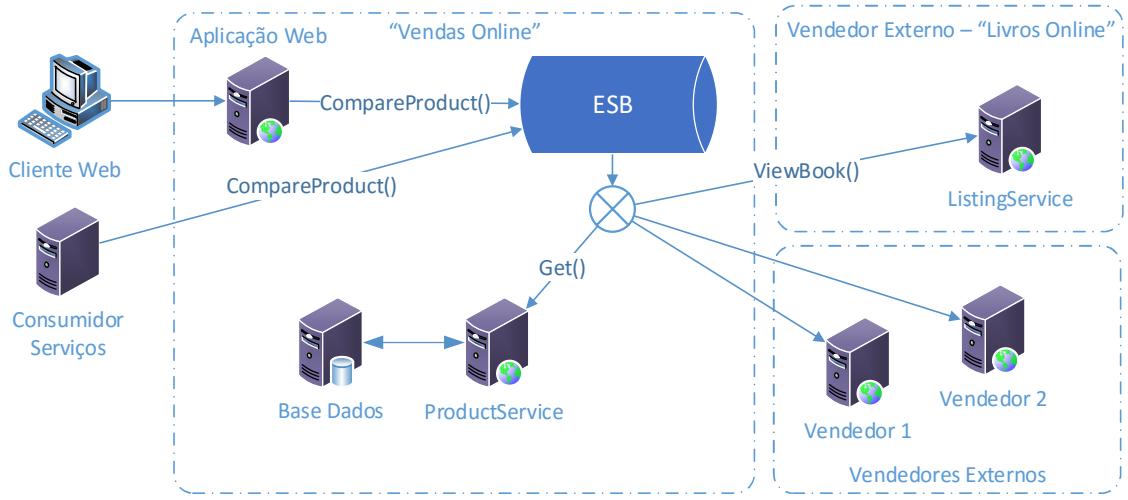

Figura 3.13: Orquestração de Serviços – Comparar Produtos

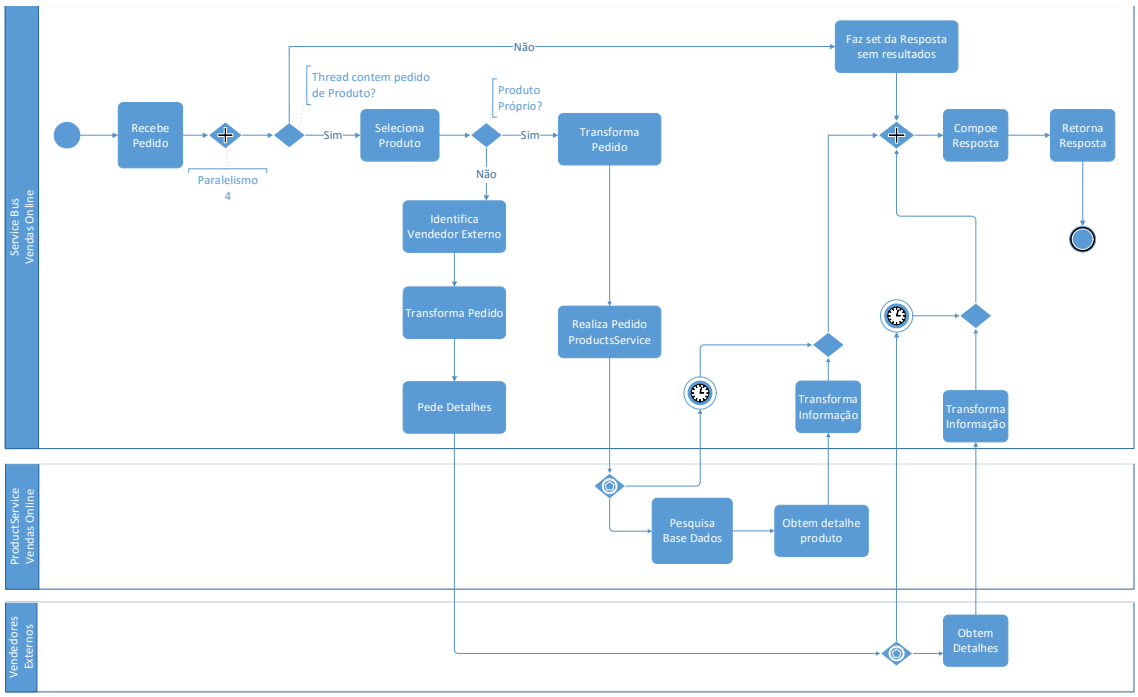

Figura 3.14: Diagrama BPMN – Comparar Produtos

Ao receber o pedido, o *service bus* vai iniciar um processamento em paralelo, utilizando quatro *threads*. Para cada *thread* é avaliado se o pedido contém o identificador de um produto. Caso este não exista, é feito o set da resposta para o equivalente de sem resultados; caso exista, será realizado o processamento necessário para pedir os seus detalhes, executando para esse efeito o mesmo fluxo do pedido de detalhes. No processamento de cada *thread* após a obtenção da resposta, esta será transformada para um formato próprio e comum a todas as *thread* e informação do vendedor adicionado.

O *service bus* aguarda que todas as *threads* sejam executadas, o que significa ter recebido respostas de todas ou ter atingido um time out, processando então todas as respostas obtidas com sucesso e agrupando os detalhes dos produtos obtidos isoladamente numa única resposta que é então retornada ao cliente.

4. Desenvolvimento do SOASales

Um dos objetivos deste projeto centra-se na idealização de um SOA, incluindo o desenho do sistema e a realização de uma implementação parcial. Nesta secção serão apresentados alguns aspetos referentes à implementação realizada, justificando algumas das opções tomada.

Numa primeira etapa da implementação procurou identificar-se um *service bus* para ser utilizado no sistema, tendo sido realizada uma pesquisa de forma a identificar as diversas opções disponíveis. Foi verificado que existem várias opções no mercado entre soluções proprietárias e *open source*, pagas e gratuitas, ou pelo menos com uma versão gratuita ainda que mais limitada, entre as soluções disponíveis podem-se encontrar soluções implementadas em Java e em .Net. Algumas das soluções encontradas foram o Enterprise Service Bus da Microsoft, o Oracle Service Bus, o Mule ESB, o Open ESB entre outros.

Após análise das soluções disponíveis a escolha recaiu sobre o Mule ESB. Trata-se de uma ferramenta *open source*, disponibilizando uma versão gratuita e uma versão comercial, sendo disponibilizada em diferentes formatos como o *standalone*, que pode ser instalado como serviço do Windows. A existência de um IDE para auxiliar no desenvolvimento do sistema e dos fluxos, o Anypoint Studio (Figura 4.1), oferece suporte para diferentes protocolos e tecnologias, como REST, SOAP, JMS, Email entre outros além de permitir um elevado grau de flexibilidade e customização. Adicionalmente, apesar do Mule ESB se tratar de uma ferramenta *open source,* usufrui de uma larga comunidade de desenvolvimento e suporte, existindo ainda uma alargada base de documentação que inclui tutoriais e exemplos de implementações, para além de ser possível adquirir suporte comercial [30, 31, 32].

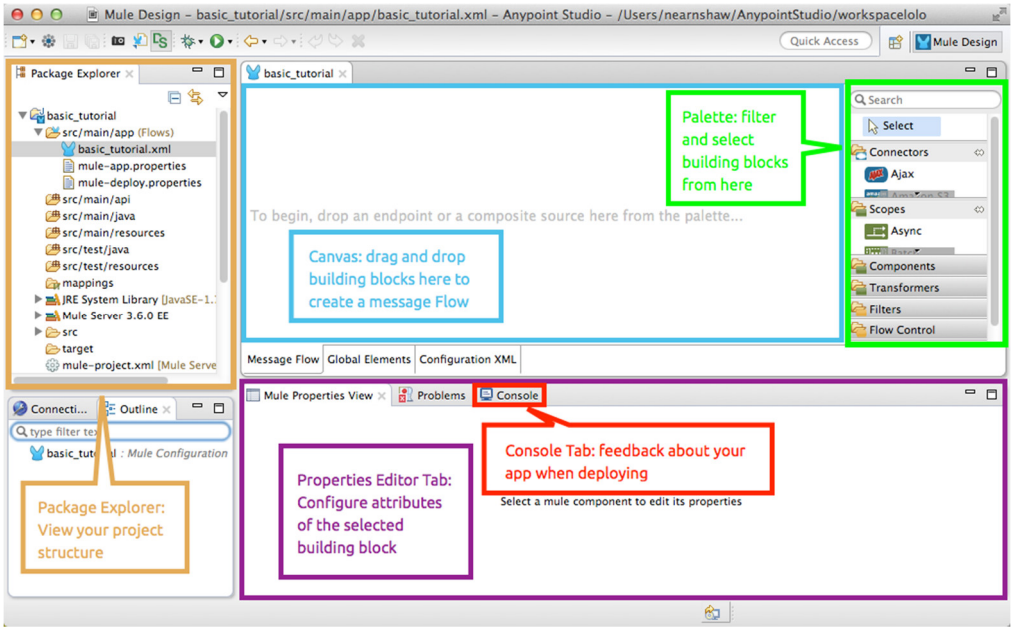

Figura 4.1: AnyPoint Studio - Interface [30]

Falando em mais detalhe acerca do Anypoint Studio, cujo interface pode ser visualizado na Figura 4.1, este disponibiliza um editor de xml e um editor gráfico para auxiliar o processo de desenvolvimento, Figura 4.2, disponibilizando um conjunto variado de componentes, alguns dos quais podem ser usados diretamente, outros necessitando de

alguma configuração básica e outros permitindo a customização do comportamento. Esta customização pode ser realizada diretamente no componente ou com recurso a lógica implementada em JAVA que pode ser reutilizada em diferentes componentes. O Anypoint Studio permite o teste dos fluxos criados pela sua execução, disponibilizando um servidor para a sua execução e funcionalidades de *debug*.

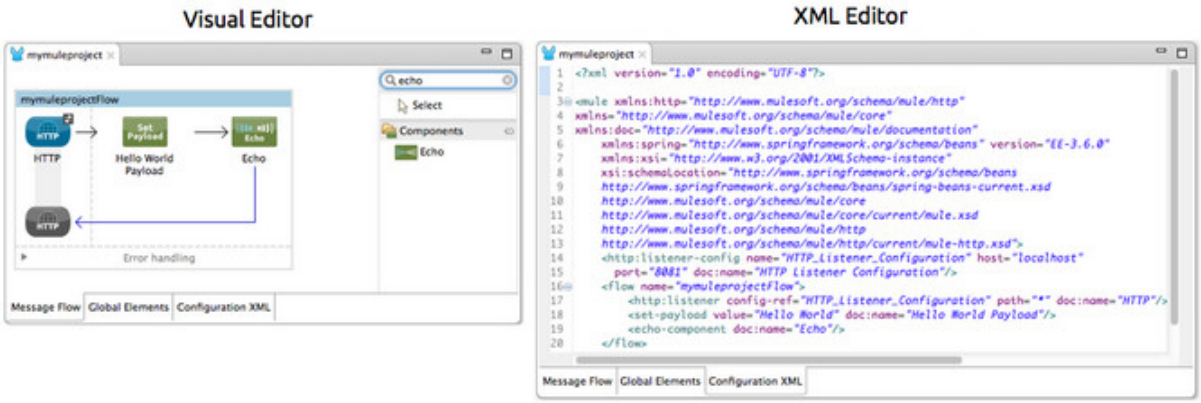

Figura 4.2: AnyPoint Studio – Editores [30]

Outro fator que contribuiu para esta escolha são os exemplos de organizações que recorreram ao Mule ESB nas suas soluções, como Ebay [33], The University of Witwatersrand [34], TiVo [35], UCSF Medical Center [36], Unicef [37] entre outras. Em algumas destas situações foram criados casos de estudo que se encontram disponíveis para consulta na página web da mule soft[30].

Tendo selecionado o Mule ESB foram identificadas as necessidades ao nível do software para a execução do Anypoint Sudio e para montagem do sistema a desenvolver. A máquina selecionada para o desenvolvimento dos fluxos do *service bus* e para hospedar a solução e os serviços apresenta a seguinte especificação

- Sistema Operativo: Windows 7 64bit,
- RAM: 8GB
- Processador: 2.20GHz, dual-core
- Disco: 50GB

Após a seleção do *service bus* e da preparação da máquina de desenvolvimento foi iniciado o processo de implementação do sistema. Nesse sentido foram identificados alguns cenários e serviços de entre os apresentados no capítulo 3 para iniciar a implementação. Os cenários selecionados foram os relacionados com a pesquisa e consulta de produtos uma vez que estes cenários eram os mais complexos e variados, contendo composição de serviços, consumo de serviços internos e externos assim como cenários em que o serviço a chamar é decido de acordo com o conteúdo do pedido.

Após esta fase inicial foi iniciada a implementação das diversas aplicações necessárias ao sistema. Nas seções seguintes será feita uma apresentação mais detalhada das diversas aplicações e implementações realizadas.

4.1. Serviços das organizações

O SOASales apresenta a especificação para serviços de duas organizações: a Vendas Online e a Livros Online, tendo sido definidos diversos serviços e operações para cada uma delas.

Considerando os cenários selecionados, foram identificados os seguintes serviço e operações da Vendas Online a serem implementados:

- ProductsService
  - Search
  - Get

Para a organização Livros Online por sua vez foi identificado o seguinte serviço e operações:

- ListingService
  - SearchBook
  - ViewBook

Considerando que seriam implementados dois serviços pertencentes a organizações diferentes optou-se por realizar a implementação dos mesmos recorrendo a diferentes tecnologias e com suporte em fontes de dados distintas.

O ProductsService, sendo o serviço interno à organização que implementa o SOA com recurso ao Mule ESB, baseado em Java, foi implementado com recurso a C# .NET e como serviço REST. A fonte de dados selecionada foi o MySQL com uma tabela Products, Figura 4.3, com informação sobre os produtos da empresa. Inicialmente a informação presente na tabela era mais limitada, com informação como o nome, descrição e preço, tendo sido adicionados novos campos numa fase posterior não estando, no entanto, estes campos a ser utilizados ou apresentados pelo SOA, como por exemplo a origem do fabrico ou local de armazenamento. O código fonte e o script de criação da base de dados podem ser encontrados no Anexo D e os binários do serviço no Anexo E.

Figura 4.3: Tabela Products - Vendas Online

O ListingService, pertencente a uma organização externa, foi implementado com recurso a Java e SOAP. Inicialmente a fonte de dados foi um simples ficheiro csv (*Comma-separated values*), tendo mais tarde sido substituído por uma fonte de dados sqlite. O código fonte no Anexo D e os binários do serviço no Anexo E, incluindo o ficheiro sqlite.

Após a obtenção de uma versão funcional dos serviços, estes foram instalado na máquina de desenvolvimento. Para esse efeito foi ativado o *Internet Information Service*, IIS, disponibilizado como funcionalidade do Windows para a instalação do ProductsService, enquanto que para o ListingService foi necessária a instalação de um servidor adicional tendo sido selecionado o Apache Tomcat,

4.2. Service Bus

O sistema desenhado inclui na sua arquitetura um *service bus* que é responsável por parte da lógica de negócio. Tendo sido selecionado o Mule ESB como *service bus* a implementação da lógica traduz-se na construção de fluxos. Cada fluxo define e instancia um ponto de entrada no sistema.

Na implementação realizada foram criados quatro fluxos principais, um para cada funcionalidade disponibilizada pela organização: a pesquisa de produtos, a obtenção de detalhes, a comparação de produtos e a disponibilização uma interface alternativa para cada operação, tendo este quarto fluxo sido apenas parcialmente implementado.

A criação destes fluxos foi um processo iterativo, uma vez que com o decorrer do projeto e o consequente aprofundamento do conhecimento sobre o Mule ESB, foram feitas algumas alterações às implementações iniciais, nomeadamente a utilização de fluxos principais e secundários de forma a tornar mais legível a o fluxo e em algumas situações possibilitar uma reutilização dos fluxos secundários.

De seguida iremos apresentar com algum detalhe um dos fluxos implementados, o de obtenção de detalhes de um produto, cuja visualização gráfica se encontra na Figura 4.4, Figura 4.5 e na Figura 4.6. Nestas imagens podem ser observados diversos componentes disponíveis no Mule ESB para construção dos fluxos. No anexo B encontram-se os vários diagramas implementados na sua totalidade juntamente com os ficheiros xml associados e as classes java implementadas. O projeto do Anypoint Studio é disponibilizado no anexo C e a exportação deste projeto para instalação do sistema no anexo D.

Na Figura 4.4 podemos observar o fluxo principal da obtenção dos detalhes, sendo neste fluxo que se encontra definido o ponto de entrada e de saída.

Com base nos parâmetros de entrada são definidas duas variáveis que serão utilizadas posteriormente no fluxo. Após a definição destas variáveis o processamento é para um fluxo secundário, o getProductFlow apresentado na Figura 4.5, ficando o fluxo principal fica a aguardar pela resposta.

O fluxo principal após receber a resposta realiza a sua transformação para xml e reencaminhamento para um servidor de mensagens e para o cliente que realizou o pedido.

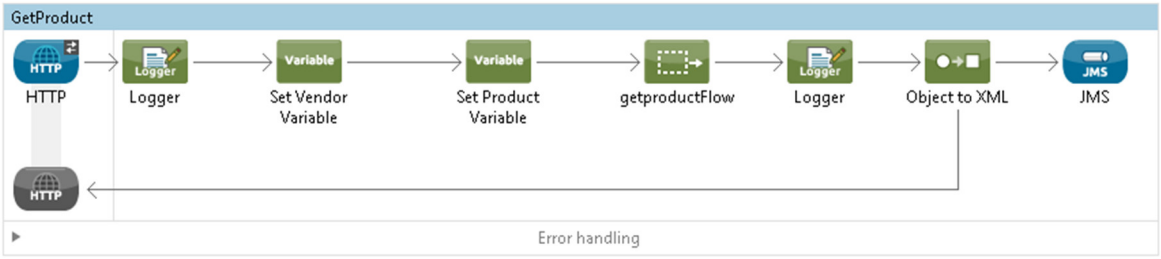

Figura 4.4: GetProduct - Fluxo principal

Como foi dito anteriormente, o Anypoint disponibiliza um editor gráfico e um editor de xml, sendo que ambos podem ser utilizados na criação do fluxo e a alteração de um deles vai ser refletida no outro. De seguida é apresentada uma pequena secção do xml deste fluxo correspondente apenas ao fluxo principal, Figura 4.4.

```
<flow name="GetProduct">
    <http:listener config-ref="HTTP_Listener_Configuration"
path="/api/ProductsService/Get" doc:name="HTTP"/>
    <logger message="GetProduct Request -  vendor:
#[message.inboundProperties.'http.query.params'.vendor]
Id=#[message.inboundProperties.'http.query.params'.id] " level="INFO"
doc:name="Logger"/>
    <set-variable variableName="vendor"
value="#[message.inboundProperties.'http.query.params'.vendor]" doc:name="Set Vendor
Variable"/>
    <set-variable variableName="product"
value="#[message.inboundProperties.'http.query.params'.id]"
doc:name="SetProductVariable"/>
    <flow-ref name="getproductFlow" doc:name="getproductFlow"/>
    <logger message="Get Product Result - Payload:  #[payload]" level="INFO"
doc:name="Logger"/>
    <mulexml:object-to-xml-transformer doc:name="Object to XML"/>
    <jms:outbound-endpoint  connector-ref="Active_MQ" transformer-
refs="Object_to_JMSMessage" doc:name="JMS" topic="RecentlyViewd">
        <jms:transaction action="NONE"/>
      </jms:outbound-endpoint>
</flow>
```

No fluxo secundário, getProductFlow, é realizada a identificação do pedido a fazer para obtenção dos detalhes do produto. A variável vendor definida no fluxo principal é utilizada para a identificação do vendedor e consequentemente qual o pedido a realizar.

Após identificação do vendedor é realizada a chamada ao serviço correspondente, se necessário contruindo previamente a mensagem.

No caso de uma chamada ao serviço ProductsService Vendas Online apenas é necessário construir a *query string* com os parâmetros adequados. No caso dos vendedores Livros Online e Ebay por se tratarem de pedidos a serviços SOAP é necessário construir o envelope SOAP esperado pelos serviços. No caso do Ebay é ainda necessário definir alguns parâmetros não ao nível da mensagem mas do pedido, Figura 4.6.

Após a obtenção de resposta do vendedor esta é processada, por meio de implementações Java, de forma a que a informação seja convertida para um formato comum e retornada ao fluxo principal.

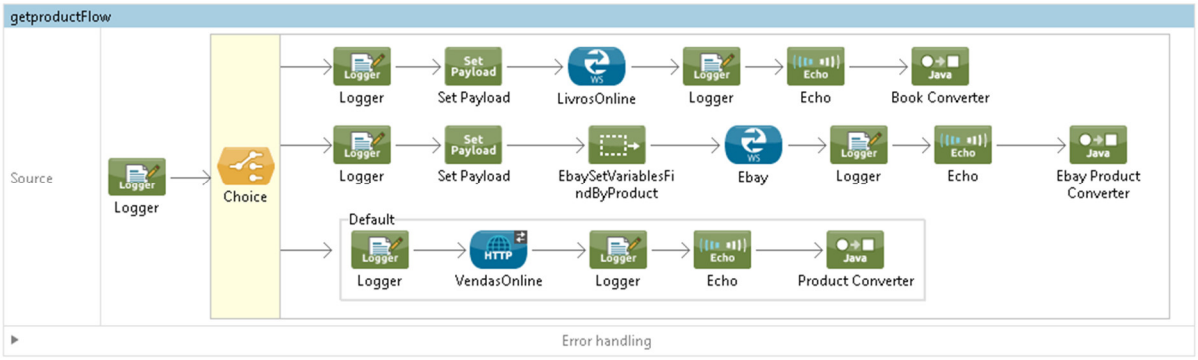

Figura 4.5: GetProduct, identificação vendedor e realização do pedido - Fluxo secundário

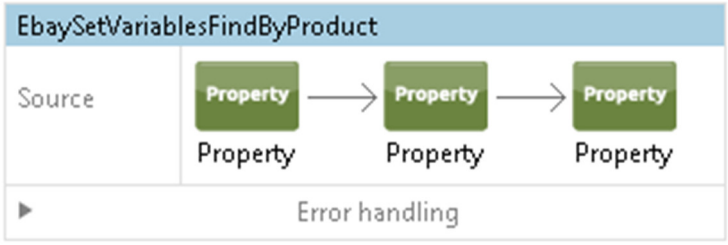

Figura 4.6: GetProduct, definição variáveis - Fluxo secundário

Como foi referido anteriormente, estão a ser publicadas mensagens para um servidor de mensagens sob a forma de tópico, ao qual foi dado o nome de RecentlyViewd, sempre que os detalhes de um produto são consultados. Estas mensagens ficam disponíveis para qualquer entidade que queira subscrever o tópico.

Esta funcionalidade foi introduzida após a fase de desenho do sistema com o intuito de adicionar mais uma tecnologia ao sistema e pelo facto de a comunicação por troca de mensagens no formato de *publish-subscrive* ser uma forma de comunicação comum em cenários de integração de aplicações.

O cenário implementado foi um cenário simples com o intuito não de adicionar valor de negócio, mas de ter representação de troca de mensagens por *publish-subscrive* no sistema.

Para a implementação deste cenário foi necessária a seleção e instalação de um servidor de mensagens tendo sido selecionado o ActiveMQ [38] pela facilidade de integração com o Mule ESB, por suportar mensagem por Tópicos (*Topics*) e Filas (*Queues*) e por disponibilizar uma aplicação web de administração.

Sem fazer uma apresentação completa dos restantes fluxos, que podem ser consultados em anexo, iremos salientar alguns pontos dos mesmos.

No fluxo de pesquisa de produtos, SearchProducts, no fluxo secundário, searchProductsFlow que pode ser visualizado na Figura 4.7, é empregue um processamento em paralelo onde se pretende obter e agregar a resposta de vários pedidos, sendo que num dos processamentos se verifica uma situação em que tendo dois pedido se pretende o resultado do primeiro a retornar uma resposta com sucesso.

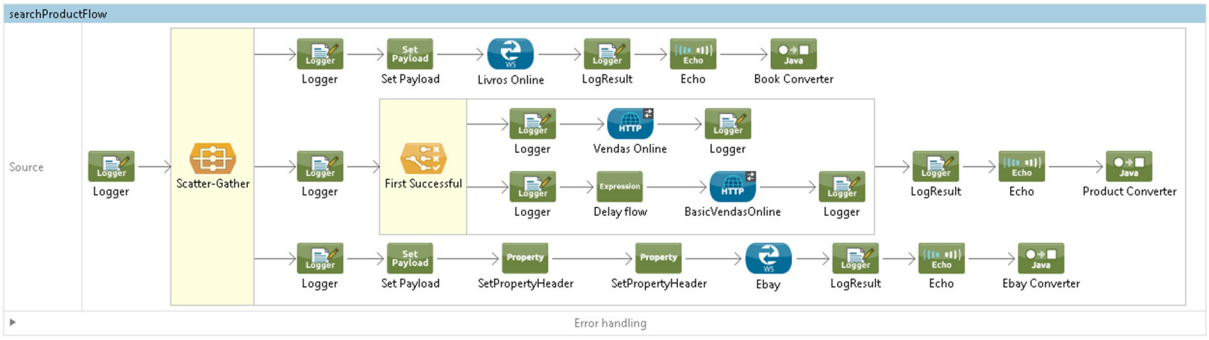

Figura 4.7: Search Product - Fluxo secundário

O pedido recebido contém a informação a pesquisar. No fluxo, esta informação vai ser enviada na sua totalidade para os três caminhos pré-definidos, cada um deles correspondente a um vendedor. O fluxo fica a aguardar até receber a reposta de todos os caminhos ou um máximo de 60s, altura em que vai processar as mensagens recebidas de cada caminho, ignorando caminhos que tenham falhado, e as agrega numa única resposta.

O fluxo apresentado na Figura 4.7 representa a versão final do fluxo, tendo sido adicionado em relação à versão original lógica de redundância para o serviço de pesquisa da organização Vendas Online. Esta foi uma decisão tomada no sentido de acrescentar complexidade ao sistema assim como um exemplo de redundância em caso de falha de resposta de um serviço. Nesse sentido foi criado um novo serviço semelhante ao serviço de pesquisa definido inicialmente para a organização, mas mais limitado, o serviço foi desenvolvido com a mesma tecnologia do original sendo mais limitado por pesquisar em menos campos da base de dados e retornar um número limitado de resultados. Este serviço foi instalado num site distinto de forma a que os serviços fossem independentes.

A lógica de redundância implementada baseou-se num mecanismo de primeira resposta. No caminho correspondente à organização Vendas Online foi adicionado um componente "*First Successfull*" com dois caminhos, um para o serviço de pesquisa principal e outro para o serviço secundário, sendo utilizada apenas a primeira resposta a ser obtida.

Uma vez que serviço de pesquisa principal realiza uma pesquisa mais completa o seu tempo de resposta deverá ser mais lento que o do serviço simplificado. Considerando que apenas se pretende recorrer ao serviço simplificado em caso de falha ou problemas de *performance* do serviço principal foi implementado um compasso de espera antes da chamada ao serviço simplificado.

Nos fluxos de pesquisa e obtenção de detalhes os pontos de entrada definidos são pedidos HTTP em que a informação necessária deve vir na *query string*. No fluxo de comparação de produtos, Figura 4.8, optou-se por uma abordagem diferente expondo o ponto de entrada como um serviço SOAP, sendo exposto um WSDL.

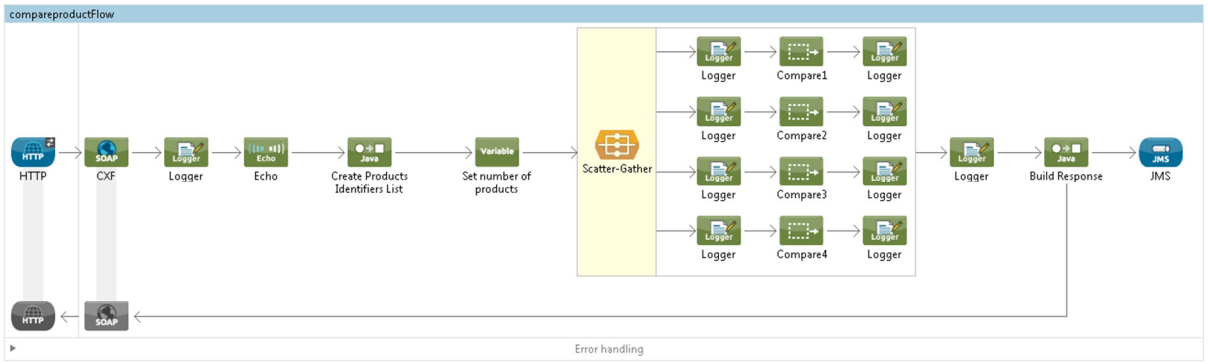

Figura 4.8: CompareProducts - Fluxo Principal

Analisando os fluxos apresentados podemos verificar que o sistema conta com três organizações distintas: a própria organização, Vendas Online, e duas organizações externas que são fornecedores da organização, a Livros Online e o Ebay.

A adição da organização Ebay foi efetuada estando já as três operações a funcionar com as organizações Vendas Online e Livros Online, a decisão de a adicionar foi justificada pela inclusão de um novo elemento, externo à máquina de desenvolvimento e do qual apenas é conhecido a interface disponibilizada e a informação que a organização disponibilize sobre o mesmo.

Após a implementação dos fluxos para as três operações, foi iniciada a implementação de um novo fluxo cujo objetivo seria disponibilizar uma interface alternativa para as operações, ou seja, cada operação estaria disponível em mais que um ponto de entrada do sistema em formatos distintos.

Este fluxo, presente na Figura 4.9, expõe um ponto de entrada SOAP com três operações disponíveis correspondentes às funcionalidades já mencionadas. Este fluxo apenas foi implementado na sua totalidade para a obtenção de detalhes, sendo que as outras operações retornam uma mensagem a indicar que não se encontra implementado.

Neste fluxo, após a receção do pedido este é processado de forma a identificar a operação do pedido. Analisando este fluxo para a operação de obtenção de detalhes, o pedido SOAP é recebido pelo sistema sendo a informação do pedido analisada e guardada em diversas variáveis a usar posteriormente no fluxo. Com base nas variáveis definidas é identificada a operação pretendida e selecionado o caminho a seguir. Tendo identificado a operação de obtenção de detalhes o fluxo é passado para um fluxo secundário, vendasonlineViewProductFlow apresentado na Figura 4.10, onde é feito a definição de variáveis em preparação para a chamada ao fluxo secundário getproductFlow definido no fluxo apresentado anteriormente na Figura 4.5.

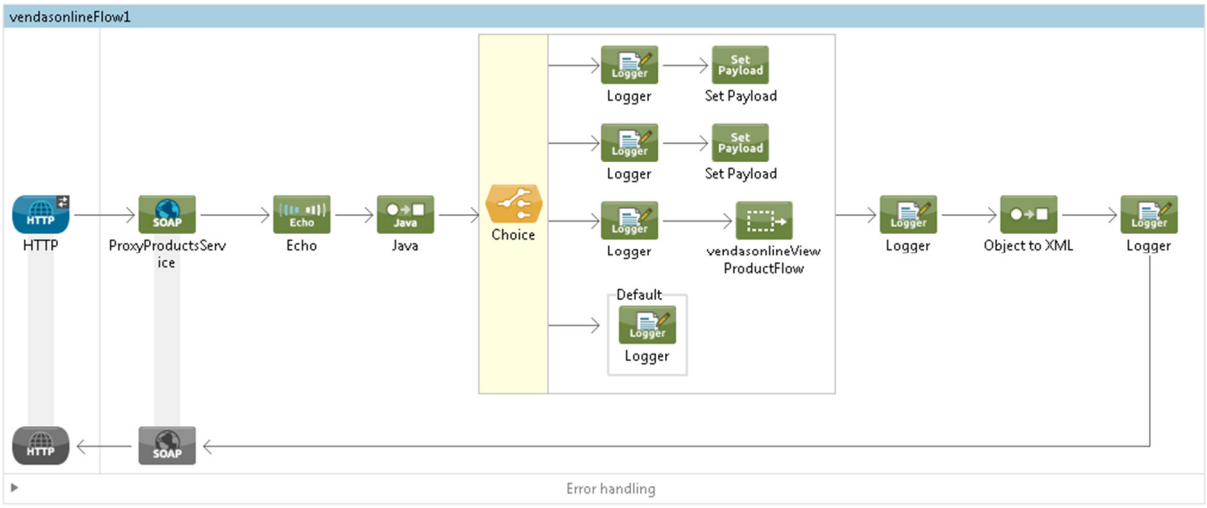

Figura 4.9: Mule ESB - SOAP Interface, fluxo principal

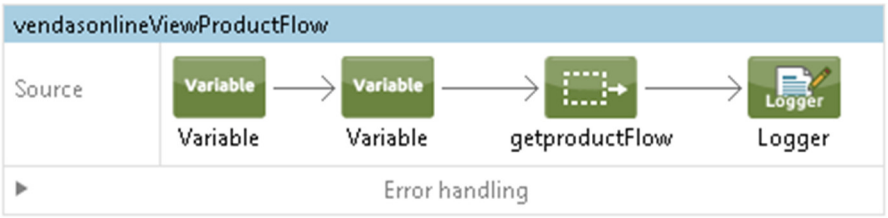

Figura 4.10: Mule ESB - SOAP Interface, fluxo secundário

4.3. Aplicações Auxiliares

O objetivo principal deste projeto centrava-se na idealização e implementação de um sistema SOA. Adicionalmente foram implementadas duas aplicações auxiliares, tendo a decisão de as implementar surgido no sentido de enriquecer o trabalho realizado.

Uma das aplicações consiste numa aplicação web, tendo surgido de forma a garantir a existência de um cliente do sistema, ou seja, um cliente que consume os serviços e operações disponibilizadas pelo SOA. A aplicação foi desenvolvida com recurso a várias tecnologias e *frameworks* como por exemplo C#, MVC, JQuery, nomeadamente o plug-in DataTables [39], bootstrap [40], sendo algo muito simples apenas para consumir os serviços e apresentar as respostas.

A aplicação web disponibiliza funcionalidade para o consumo das três operações implementadas no sistema, a pesquisa de produtos, a obtenção e comparação de produtos.

A pesquisa de produtos, Figura 4.11, permite ao utilizador realizar pesquisas de produtos pela inserção do texto a pesquisar e apresentando os resultados numa tabela com paginação, ordenação e pesquisa nos resultados.

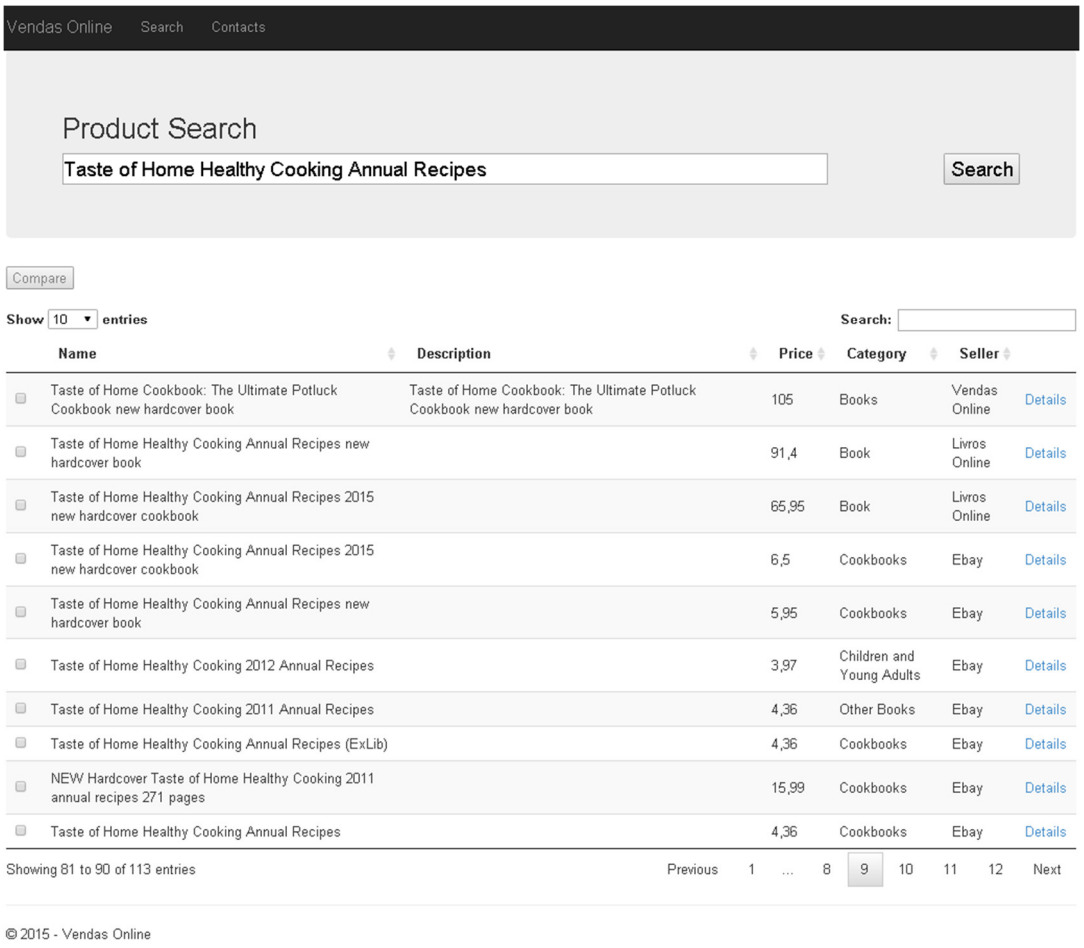

Figura 4.11: Aplicação Web Cliente – Pesquisa de produtos

A partir dos resultados da pesquisa é possível realizar a comparação de produtos, permitindo a seleção de até quatro produtos para comparar e sendo as características destes apresentadas lado a lado, Figura 4.12.

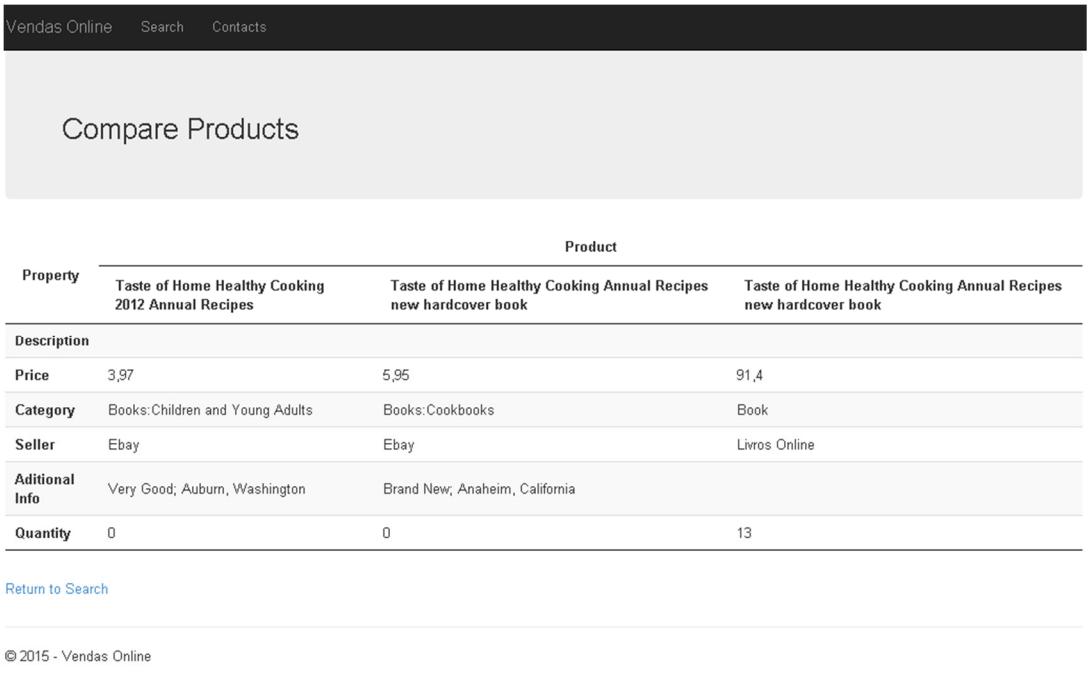

Figura 4.12: Aplicação Web Cliente - Comparação de produtos

Também a partir dos resultados da pesquisa é possível a visualização dos detalhes de um produto em particular, Figura 4.13.

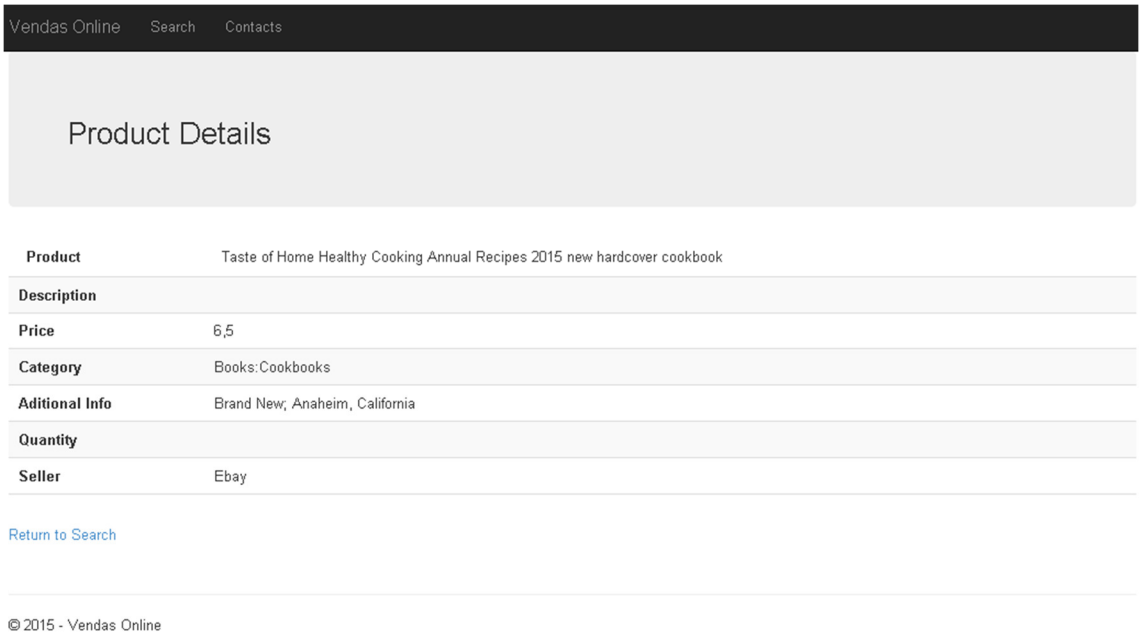

Figura 4.13: Aplicação Web Cliente - Visualização de detalhes

Em adição à aplicação web foi também implementada uma simples aplicação de consola, em C# e Web forms, Figura 4.14. Esta aplicação surgiu no seguimento da decisão de implementar um cenário JMS com publicação de mensagens de forma a ter um cliente a subscrever as mensagens publicadas.

A aplicação cliente estabelece uma ligação ao servidor de mensagens, o ActiveMQ, e realiza uma subscrição ao tópico “RecentlyViewd”. Ao ser publicada uma mensagem para o servidor de mensagens a aplicação estando à escuta recebe a mensagem e apresenta o seu conteúdo.

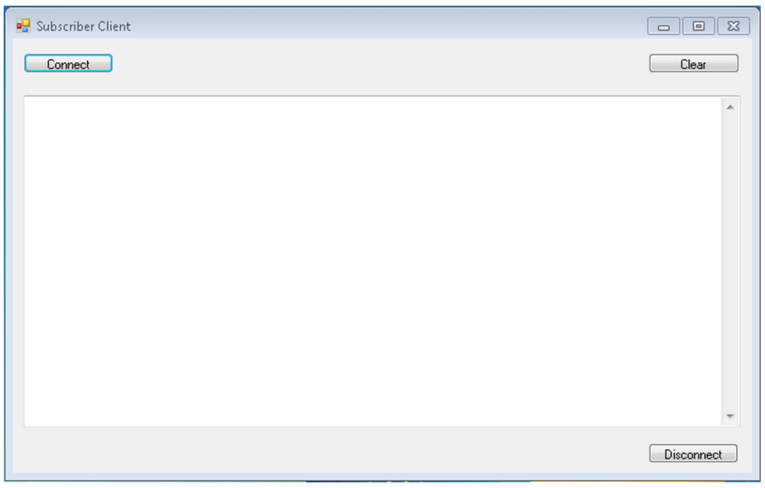

Figura 4.14: Aplicação de consola - Cliente ActiveMQ

5. Análise e teste

Nesta secção pretende-se fazer uma análise ao SOASales, tendo como objeto não só o sistema implementado, mas também o sistema idealizado na sua globalidade, realizando uma análise crítica às suas características quando comparadas com as características atribuídas a outros ambientes SOA. Sendo esta análise de um carácter qualitativo e não quantitativo, foram definidos e realizados alguns testes de forma a obter alguns valores da performance do sistema de forma a realizar também uma análise quantitativa.

5.1. Tecnologias e Características do Sistema

A avaliação qualitativa de um sistema encontra-se intrinsecamente ligada ao objetivo que originou a sua conceção e aos atributos e requisitos mais relevantes para os *stakeholders* do sistema, sendo que muitas vezes tem que haver um equilíbrio ou compromisso quando existem requisitos contraditórios, como por exemplo uma melhor performance em detrimento de uma maior adaptabilidade do sistema.

Considerando que o sistema foi idealizado com o intuito de ser representativo de um ambiente SOA com alguma complexidade, será realizada uma análise das características presentes no sistema comparativamente a características atribuídas a este tipo de ambiente de forma a determinar se o sistema pode ser considerado representativo. Para a realização desta análise será considerada informação reunida de várias fontes, nomeadamente a apresentada no capítulo 2 deste documento e de fontes adicionais, como o relatório técnico Evaluating a Service-Oriented Architecture [41]. Em adição às características iremos também analisar as tecnologias presentes no sistemas utilizando como termos de comparação os exemplos apresentados no capítulo 2.6.

No decorrer desta avaliação será considerado o ambiente idealizado na sua totalidade fazendo particular atenção às secções implementadas.

Algumas das características associadas a ambientes SOA que se encontram presentes no sistema que iremos discutir neste capítulo são:

- Baixo nível de acoplamento entre os intervenientes
- Serviços autónomos
- Reutilização de lógica
- Disponibilização de serviços simples e compostos, orquestração de serviços
- Adaptabilidade do sistema
- Interoperabilidade do sistema com diferentes tecnologias e sistemas

Baixo nível de acoplamento

O baixo nível de acoplamento entre os intervenientes e os serviços autónomos traduzem-se numa certa independência entre estes, sendo que cada um deverá poder sofrer alterações sem que estas tenham repercussões nos outros intervenientes ou serviços, desde que as alterações não sejam ao nível da interface de comunicação.

Analisando o sistema em busca desta característica podemos considerar que esta se encontra presente, sendo que esta afirmação é suportada pelas diversas alterações de que os alguns dos intervenientes do sistema foram alvo no decorrer do processo de desenvolvimento. Exemplos destas alterações foram as alterações ao nível da fonte de

dados tanto do ProductsService como do ListingService. No caso do ProductsService a fonte de dados sofreu alterações ao nível de uma tabela, tendo sido adicionadas novas colunas, enquanto que no caso do ListingService a alteração foi a troca da fonte de dados que passou de um ficheiro csv para Sqlite. Em ambos os casos as alterações nos serviços não tiveram repercussões na lógica implementada no *service bus.*

Reutilização de lógica

A reutilização de lógica é promovida nos ambientes SOA por normalmente se encontrar associada a uma redução de custos, pelo menos a longo prazo. No sistema podemos verificar que existe reutilização de lógica para a disponibilização de funcionalidades distintas. As operações de obtenção de detalhes e de comparação de produtos utilizam ambas os mesmos serviços e operações das organizações envolvidas. Em ambos os casos é utilizada a operação de cada organização que, recebendo um identificador, retorna os detalhes de um produto.

Serviços compostos e orquestração de serviços

As operações mencionadas são também exemplos da composição ou orquestração de serviços uma vez que o *service bus* adiciona lógica de negócio e combina diferentes inputs de forma a acrescentar valor para os seus clientes. Em ambas as situações existe lógica de forma a determinar a que organizações realizar os pedidos e no caso da comparação de produtos existe ainda a agregação de resultados de diferentes pedidos. A agregação de resultados de diferentes pedidos está igualmente presente na operação de pesquisa.

Associado a serviços compostos e orquestração de serviços, encontra-se muitas vezes associado um elemento de integração, responsável não só pela integração entre os diferentes intervenientes, mas também por lógica de negócio. Este elemento de integração pode por exemplo realizar-se por meio de módulos de BPEL. No entanto, no sistema implementado este tipo de elementos não se encontra representado uma vez que se optou por realizar a integração por meio de um *service bus*.

Registry

Quando se fala de SOA ou interação com serviços, pode existir um terceiro elemento na relação, o *registry*, servindo como repositório de serviços que pode ser consultado de forma a identificar serviços que disponibilizem determinada funcionalidade. Quando um *registry* se encontra envolvido na interação, a ligação é normalmente estabelecida dinamicamente em oposição a uma ligação ponto-a-ponto definida previamente no sistema. Ambas as opções têm vantagens e desvantagens, sendo que no SOASales se decidiu usar ligações ponto-a-ponto, tendo em conta a melhor performance associada a este tipo de ligação, assim como uma maior simplicidade em termos de implementação.

Apesar de um cenário com utilização de um *registry* e de estabelecimento de ligação dinâmica não ter sido implementado, olhando para o sistema idealizado na sua totalidade este cenário encontra-se contemplado na funcionalidade de conversão de moeda, podendo vir a ser implementado no futuro.

Um padrão comum de comunicação entre sistemas é o *publish-subscribe*. O SOASales contém um exemplo simples deste tipo de comunicação recorrendo para isso a um servidor de mensagens.

Adaptabilidade do sistema

A adaptabilidade de ambientes SOA é um dos seus dos pontos fortes, nomeadamente na facilidade de adaptação a novas necessidades. A adaptabilidade do sistema implementada pode ser observada em algumas das alterações realizadas. Exemplo destas alterações foi a adição de uma nova organização. Inicialmente apenas duas organizações faziam parte do sistema, a Vendas Online e a Livros Online, tendo numa segunda fase sido adicionada uma nova organização, o Ebay. Esta adição exigiu a alteração dos fluxos já implementados de forma a adicionar a lógica e as chamadas aos serviços da organização.

Inicialmente apenas o serviço FindingService da organização Ebay foi incorporado no sistema, sendo operações distintas deste serviço utilizadas para a pesquisa, e obtenção de detalhes de produtos, posteriormente o serviço FindingService foi substituído pelo ShoppingService na obtenção de detalhes de produtos.

Interoperabilidade do sistema

Uma das principais características atribuídas a um ambiente SOA é a sua interoperabilidade que é demonstrada pela capacidade de integração de diferentes sistemas, mesmo quando as tecnologias são compatíveis entre si.

Analisando o SOASales e em particular a implementação realizada no referente às tecnologias, podemos verificar uma certa diversidade, como por exemplo implementações Java e .NET, serviços REST e SOAP. Tomando como referência a Tabela 2.2 e Tabela 2.3 apresentada no capítulo 2.6 que contém um resumo das características de diversos exemplos de SOA, foi adicionada uma nova entrada para o SOASales, tendo resultado nas Tabela 5.1 e Tabela 5.2. Aqui podemos observar que o SOASales inclui grande parte das características identificadas no conjunto dos exemplos estudados. Nesta comparação deve ser tido em conta a falta de informação de alguns exemplos que não permitem concluir que tecnologias se encontram presentes em determinados exemplos.

Tabela 5.1: SOA – Intervenientes

Projeto / Organização	**Serviços Externos**	**ESB**	**BPEL**	***Service Discovery***	**Servidor Mensagens**
Open SOALab – Reservas Aéreas	S				
Open SOALab – Câmbios	S			S	
Open SOALab – Reservas de Hotel	S			S	
DEI-UC – Verificação e Validação (FMEA)	S			S	
DEI-UC – Verificação e Validação (runtime)	S				
DCCUF-CISUC – A Service Discovery Approach for Testing Dynamic SOAs	S	N	S		
DIOU-CTICSTP – Benchmarking of Web Services Platforms	S	N			S
HSSP– The practical Guide for SOA in Health Care	S	S			
UCM-Layer 7 Technologies	S		S		
Segurnet					
Concur	N	S		S	
Governo Dinamarquês	S			S	
OVERSEE	S	S			
First Citizens Bank					
OfficeMax		S			
SOASales	S	S	N	NI	S

S – Característica presente | NI – Não implementada | N – Característica ausente

Tabela 5.2: SOA – Tecnologias

Projeto / Organização	SOAP	REST	XML	JSON	JAVA	.NET	PHP
Open SOALab – Reservas Aéreas					S	S	S
Open SOALab – Câmbios							S
Open SOALab – Reservas de Hotel							S
DEI-UC – Verificação e Validação (FMEA)							
DEI-UC – Verificação e Validação (runtime)							
DCCUF-CISUC – A Service Discovery Approach for Testing Dynamic SOAs							
DIOU-CTICSTP – Benchmarking of Web Services Platforms					S	S	
HSSP– The practical Guide for SOA in Health Care							
UCM-Layer 7 Technologies			S				
Segurnet			S		S		
Concur	S	S					
Governo Dinamarquês	S		S		S	S	
OVERSEE		S	S	S	S		
First Citizens Bank							
OfficeMax							
SOASales	S	S	S	S	S	S	N

S – Característica presente	N – Característica ausente

5.2. Testes ao sistema

Após a implementação do sistema este foi sujeito a alguns testes, nomeadamente testes de performance e de carga. Ao realizar testes a um sistema com a intenção de obter valores de performance um ponto muito importante é o ambiente em que o sistema se encontra hospedado, uma vez que o ambiente influencia a performance.

O sistema foi implementado numa máquina virtual hospedada num servidor de máquinas virtuais disponibilizado pelo Instituto Superior de Engenharia de Coimbra com as seguintes características:

- Sistema Operativo: Windows 7 64bit,
- RAM: 8GB
- Processador: 2.20GHz, dual-core
- Disco: 50GB

Nesta máquina foram instalados os softwares necessários ao funcionamento do sistema, como o Apache Tomcat [42], o ActiveMQ [38], o Mule ESB e o Anypoint Studio [30]. Na máquina foram igualmente instalados os serviços das organizações Vendas Online e Livros Online.
Foi ainda instalada a ferramenta SoapUI [43] que foi utilizada para a realização dos testes ao ambiente que foram efetuados a partir da própria máquina.
Foram criados no SoapUI projetos e testes para cada uma das operações, incluindo testes de carga. A cada teste foram associadas algumas asserções de forma a validar o sucesso do mesmo, como por exemplo o status e o tempo máximo de resposta.

Os testes realizados e os resultados obtidos são meramente informativos uma vez que não foram estabelecidos requisitos do sistema ao nível da performance e não possuindo valores de um sistema e ambiente semelhantes não são realmente comparáveis.

Nesta secção será apresentado um resumo dos resultados dos testes podendo estes ser consultados na sua íntegra no Anexo C.

5.2.1. Pesquisa de produtos

Uma das funcionalidades disponibilizadas pelo sistema é a pesquisa de produtos. Este pedido vai procurar em todos os fornecedores disponíveis, por realização de chamadas aos serviços que estes disponibilizam, produtos que correspondam aos parâmetros de pesquisa definidos pelo cliente.
Para a obtenção de tempos de resposta deste pedido foram criados quatro testes com quatro pedidos distintos:

- Cerca 100 – pedido de pesquisa com cerca de 100 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “Statistics”;
- Acima 100 – pedido de pesquisa com mais de 100 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “cooking”;
- Acima 300 – pedido de pesquisa com mais de 300 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “Potter”;
- 0 Resultados – pedido de pesquisa que não deverá retornar qualquer resultado
    - Parâmetro searchParameters – “«««»»»”;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200,
- Tempo máximo de resposta – 90000ms,
- Conteúdo da mensagem – a palavra product, parte do xml retornado na resposta,
- Conteúdo da mensagem – parâmetro de pesquisa.

Para cada teste descrito foram realizadas 30 execuções, podendo na Figura 5.1 ser visualizado o resultado apresentado pelo SoapUI de uma das execuções, onde se pode visualizar o sucesso dos pedidos e os tempos de resposta. Analisando os resultados de

todas as execuções realizadas verificou-se que todas foram executadas com sucesso e no gráfico da Figura 5.2 pode ser visualizado o resumo dos resultados obtidos.

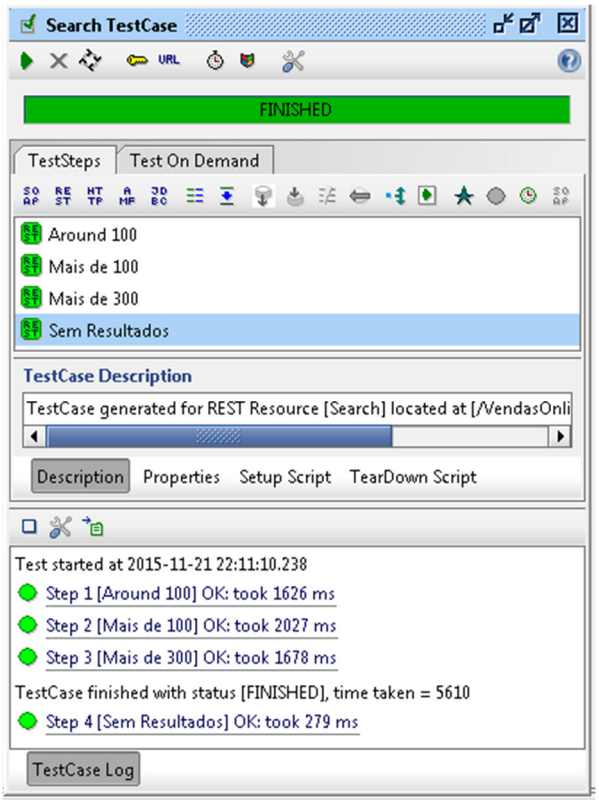

Figura 5.1: Execução testes de pesquisa [43]

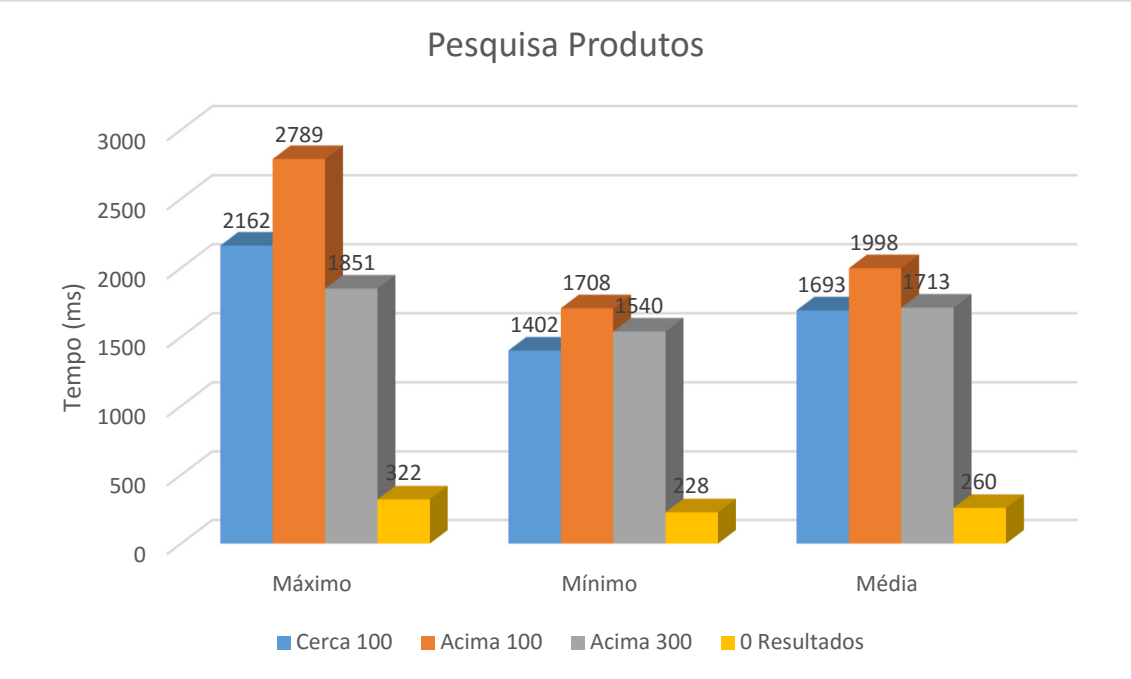

Figura 5.2: Gráfico - Pesquisa Produtos

Analisando os resultados obtidos podemos verificar que, como seria de esperar, o pedido que não retorna resultados é o mais rápido. No entanto e ao contrário do que se poderia esperar, o pedido mais lento não é o que retorna mais resultados (Acima de 300), mas o que retorna mais de 100 (Acima de 100). Este desvio aos resultados esperados pode estar

relacionado com a proporção de resultados dos diferentes fornecedores, e o tempo de resposta dos serviços dos fornecedores assim como o tempo de processamento das respostas de cada fornecedor.

Nos resultados obtidos as proporções em percentagem de produtos de cada vendedor foram:

Tabela 5.3: Resultados Pesquisa Produtos - Proporções produtos de vendedores

	Vendas Online	Livros Online	Ebay
Cerca 100	92.59	51.28	25.12
Acima de 100	5.56	18.97	1.51
Acima de 300	1.85	29.74	73.37

Analisando os resultados verifica-se que o pedido mais lento é o que apresenta uma maior proporção de produtos do vendedor Livros Online, podendo por isso estar relacionado com uma pior performance da fonte de dados.

Tendo executado os testes isoladamente realizou-se então um teste de carga, tendo o teste sido executado durante um período de 120 segundos. Na Figura 5.3 podemos ver uma imagem dos resultados apresentados pelo SoapUI após a execução dos testes.

Observando os resultados dos testes de carga verifica-se a mesma tendência dos testes anteriores nos tempos de resposta dos diferentes pedidos, o pedido mais lento é o que retorna mais de 100 resultados (Acima 100) e o mais rápido o pedido que não retorna resultados.

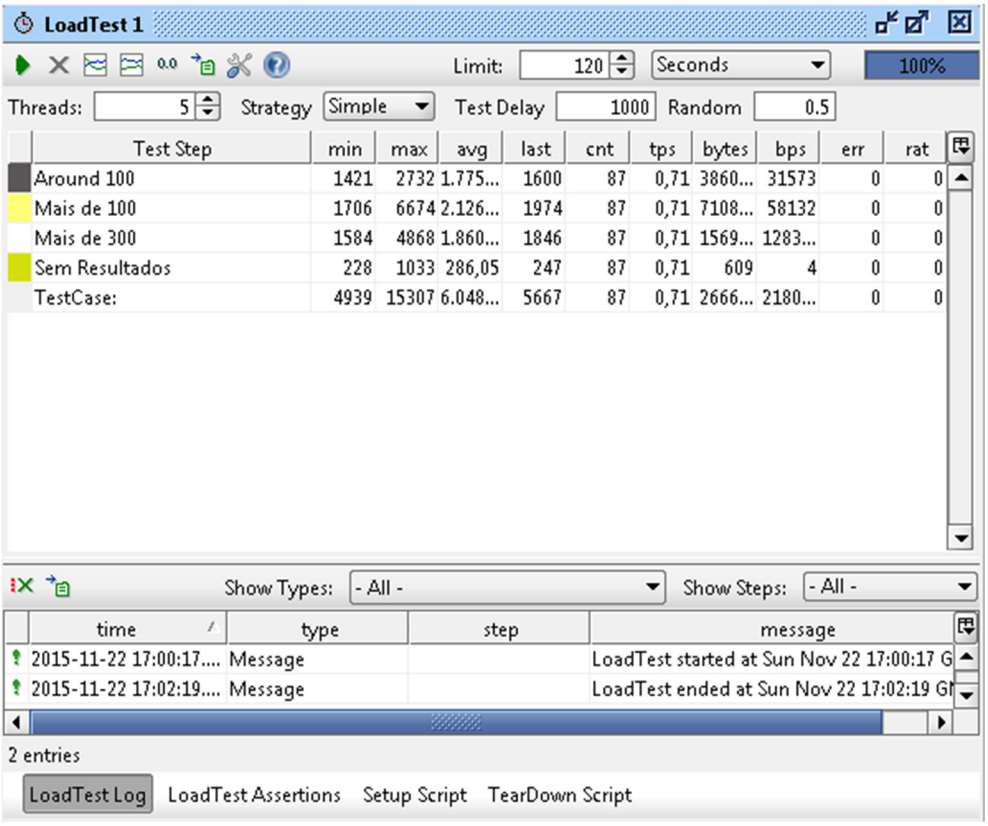

Figura 5.3: SoapUI Resultados teste carga Pesquisa Produtos [43]

5.2.2. Detalhe de produto

Uma das funcionalidades disponibilizadas pelo sistema é a obtenção dos detalhes de um produto em particular. Esta operação vai requisitar os detalhes do produto pretendido ao seu fornecedor, realizando a chamada ao serviço correspondente.

Para a obtenção de tempos de resposta desta operação foram criados quatro testes com quatro pedidos distintos:

- Ebay – pedido de detalhes de um produto da organização Ebay
  - Parâmetro vendor – 3;
  - Parâmetro id – 131604043953;
- Livros Online – pedido de detalhes de um produto da organização Livros Online
  - Parâmetro vendor – 2;
  - Parâmetro id – 6254;
- Vendas Online – pedido de detalhes de um produto da organização Vendas Online
  - Parâmetro vendor – 1;
  - Parâmetro id – 56;
- Vendedor Inválido – pedido de detalhes de um produto com um identificador de organização inválido
  - Parâmetro vendor – 4;
  - Parâmetro id – 1;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200
- Tempo máximo de resposta – 30000ms
- Conteúdo da mensagem – Nome do vendedor
- Conteúdo da mensagem – Nome do produto

Os resultados obtidos foram analisados de forma a identificar os tempos máximos e mínimos de resposta assim como a sua média. Estes valores encontram-se representado no gráfico da Figura 5.4.

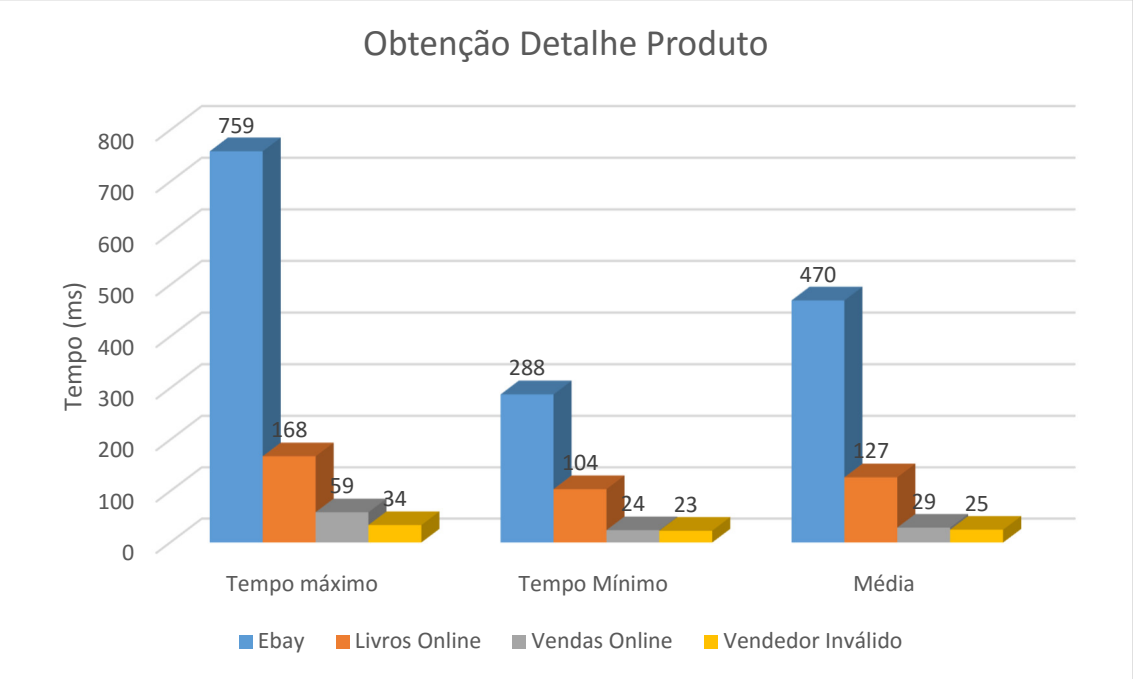

Figura 5.4: Gráfico - Detalhes de Produto

Fazendo uma análise dos valores podemos verificar que como expectável existem variações dos tempos de resposta dos pedidos entre as várias execuções assim como entre os diferentes pedidos realizados. Tendencialmente o pedido para o vendedor Ebay é o mais lento e o pedido para o Vendas Online o mais rápido de entre os pedidos com vendedores válidos.

Uma possível razão para o pedido para o Vendas Online ser mais rápido pode ser por a chamada ao ProductsService ser mais rápida que a chamada aos restantes serviços quer pela quantidade de registos a pesquisar, o Ebay deve possuir uma quantidade largamente superior de registos, quer por uma pior performance da fonte de dados, o sqlite do LivrosOnline. Outro fator que pode influenciar este resultado é o tempo de conversão de resposta, uma vez que a resposta do ProductsService é um json enquanto que dos outros serviços é um xml.

O pedido para um vendedor inválido é o mais rápido de todos, como esperado uma vez que neste caso o *service bus* identifica que se trata de um vendedor inválido retornando logo a resposta sem realizar um pedido a um novo serviço para obtenção dos detalhes.

Os resultados do teste de carga, que executou os 4 pedidos no decorrer de 120 segundos, podem ser visualizados na Figura 5.3.

Analisando os valores obtidos verificam-se as mesmas tendências que nos testes de performance, sendo o mais lento o pedido para o Ebay e o mais rápido o do vendedor inválido, seguido do pedido para o Vendas Online. No entanto nos testes de carga verifica-se que o pedido para o vendedor Livros Online apresenta uma maior variação dos tempos de resposta, sendo o seu tempo máximo de resposta mais lento do que o pedido para o Ebay mantendo, no entanto, uma média mais baixa.

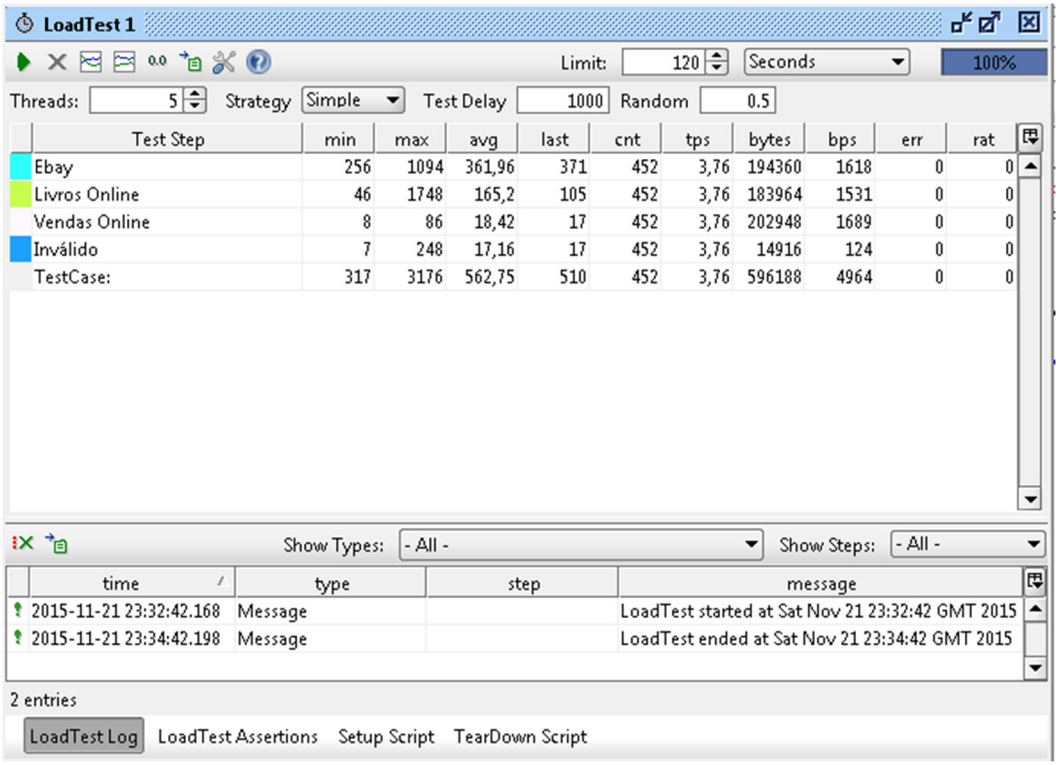

Figura 5.5: SoapUI Resultados teste carga Obtenção Detalhes Produto [43]

5.2.3. Comparação de produtos

Uma das funcionalidades disponibilizadas pelo sistema é a comparação dos detalhes de até quatro produtos. Este pedido vai obter os detalhes de até um máximo de quatro produtos, pedindo os detalhes de cada produto pretendido ao respetivo fornecedor, realizando para cada produto um pedido ao serviço disponibilizado pelo seu fornecedor. Para a obtenção de tempos de resposta deste pedido foram criados quatro testes com quatro pedidos distintos:

- 4 Vendedores diferentes – 4 produtos, 1 de cada vendedor e um vendedor repetido,
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 2;
        - Parâmetro ProductId – 6254;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 3;
        - Parâmetro ProductId – 131604043953;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- 3 Vendedores diferentes – 3 produtos, 1 de cada vendedor
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 2;
        - Parâmetro ProductId – 6254;
    - Parâmetro ProductIdentifierDataContract

        - Parâmetro VendorId – 3;
        - Parâmetro ProductId – 131604043953;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
- 1 Produto – 1 produto, vendedor Vendas Online
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- 4 Mesmo Vendedor – 4 produtos, todos do mesmo vendedor Vendas Online
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- Vendedor Inválido – 1 produto, vendedor inválido
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 4;
        - Parâmetro ProductId – 6254;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200,
- Tempo máximo de resposta – 90000ms,
- Resposta SOAP – válida, deve ser uma resposta SOAP válida,
- Conteúdo da mensagem – a palavra ProductsDataContract, que faz parte da resposta SOAP retornada pelo sistema.

Para a obtenção dos tempos de resposta da operação de obtenção de detalhes, os pedidos descritos foram executados sequencialmente por 30 vezes, os testes foram todos executados com sucesso e o resumo dos valore obtidos podem ser consultados no gráfico da Figura 5.6.

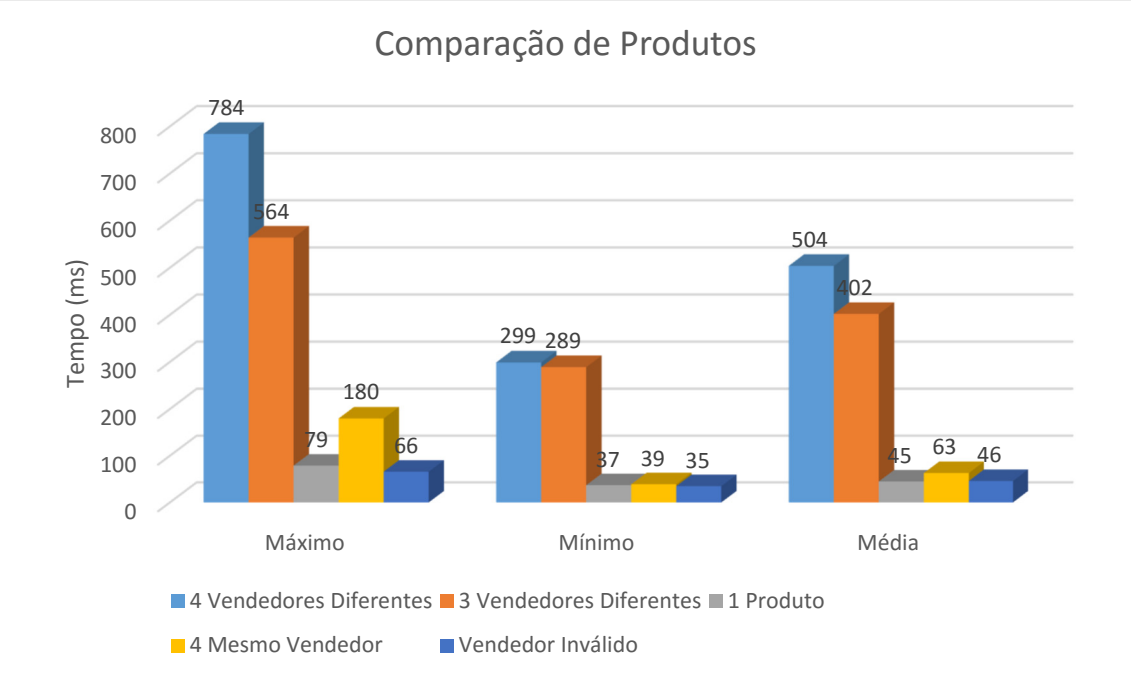

Figura 5.6: Gráfico - Comparação Produtos

Analisando os resultados verifica-se que o pedido mais lento é o dos quatro produtos (4 Vendedores diferentes) e que o mais rápido é o de apenas 1 produto de um vendedor inválido (Vendedor Inválido), neste caso a resposta é retornada pelo *service bus* sem a necessidade da realização de pedidos a outros serviços.

Os resultados do teste de carga podem ser consultados na Figura 5.7, estes foram obtidos pela criação de um teste de carga a partir dos pedidos apresentados anteriormente ,que foi executado no decorrer de 120 segundos.

Ao analisar os resultados verificam-se as mesmas tendências que nos testes de performance, sendo que o pedido mais lento é a comparação dos 4 produtos de vendedores distintos e o mais rápido o do vendedor inválido.

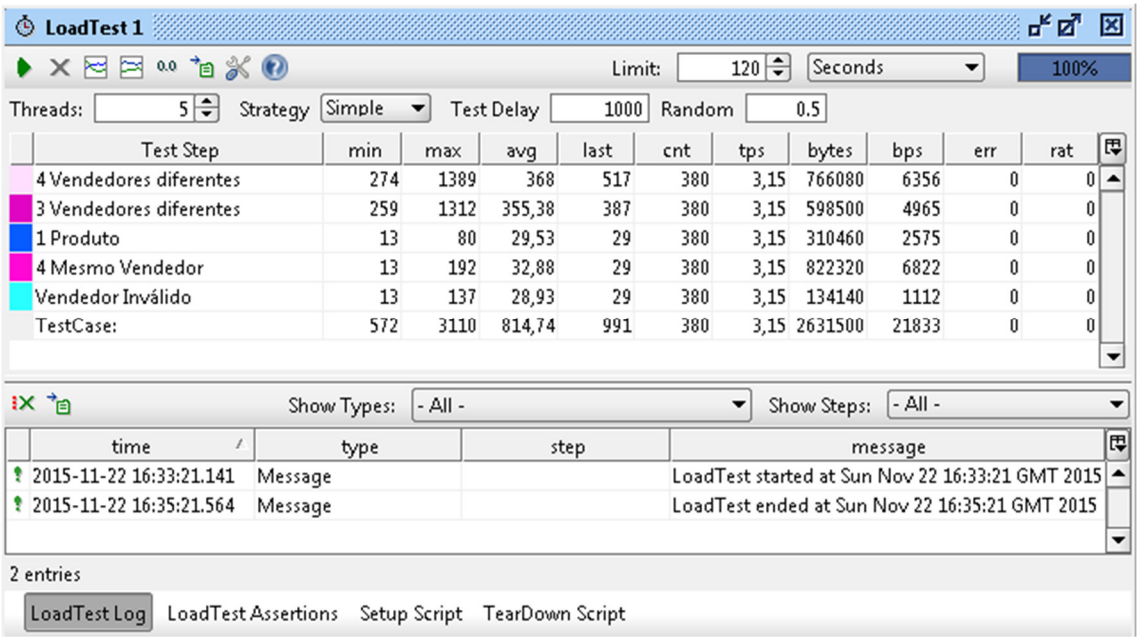

Figura 5.7: SoapUI Resultados teste carga Comparação de Produtos [43]

5.2.4. Comparação entre Operações

Para a realização da comparação dos tempos de respostas das várias operações, iremos analisar os valores obtidos para os testes de performance para a totalidade de pedidos definidos e para as várias execuções de cada um de forma a identificar o tempo de resposta mais lento, o mais rápido e a média. Para esta análise serão igualmente considerados os pedidos dos quais não se espera obter resultados. Os resultados desta análise podem ser consultados na Tabela 5.4 e no gráfico representado na Figura 5.8.

Tabela 5.4: Tempos de resposta do sistema

Operação	Tempo (ms)		
	Máximo	Mínimo	Média
Pesquisa	2789	228	1691
Detalhes	759	23	82
Comparação	784	39	212

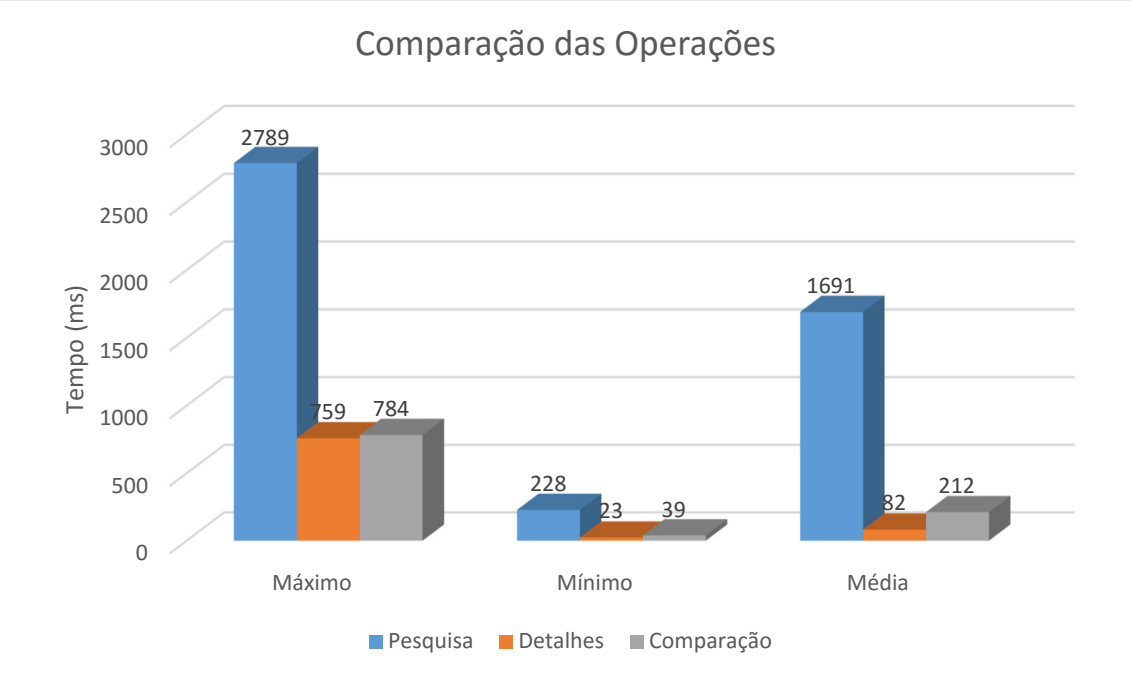

Figura 5.8: Gráfico - Comparação Operações

Observando estes valores podemos verificar que a operação de pesquisa de produtos é a mais lenta, como seria de esperar uma vez que esta operação tendencialmente processa mais informação. A mais rápida é a de obtenção de detalhes, pois processa menos informação e tem o fluxo mais simples.

Olhando para os valores dos tempos de resposta de todas as operações verifica-se que a pesquisa de produtos é a operação mais lenta e o valor máximo desta operação registado durante os testes foi de 2789 milissegundos, ou seja, menos de 3 segundos. Mesmo considerando que a adição de novos fornecedores e a dependência da resposta do sistema da performance dos serviços dos fornecedores, considerando igualmente que ao serem

retornado um maior número de resultados na pesquisa o tempo de resposta também deverá aumentar pode-se considerar que o sistema é eficiente ao responder aos pedidos efetuados, podendo os tempos apresentados ser considerados aceitáveis.

5.3. Síntese

Fazendo uma síntese da análise do SOASales e dos testes realizados ao sistema, pode afirmar-se que o sistema apresenta várias das características associadas a ambientes SOA como por exemplo:

- Diversidade de tecnologias, utilizando serviços REST e SOAP e trocando mensagens em json e xml;
- Integração de diferentes sistemas, no sistema podem ser identificadas três organizações que possuem sistemas distintos e independentes;
- Orquestração de serviços, combinando serviços de forma a providenciar uma funcionalidade crescida;
- Reutilização de lógica, reutilizando serviços e adicionando lógica de forma a fornecer funcionalidades distintas;
- Baixo nível de acoplamento entre os intervenientes, os intervenientes do sistema podem sofrer alterações interna sem que essas alterações afetem o sistema.

De salientar a utilização no sistema de um *service bus* como elemento de integração, adição de lógica de negócio e controlo dos fluxos das operações.

Observando os resultados dos testes realizados pode verificar-se que, como seria de esperar, as diferentes operações apresentam diferentes tempos médios de resposta,

- Pesquisa de Produtos – 1691s;
- Comparação de Produtos – 212s;
- Detalhes de Produto – 82s.

A pesquisa de produtos é a operação com os tempos de resposta mais lentos, uma vez que é a operação que tendencialmente processa uma maior quantidade de informação.

Considerando que a operação mais lenta teve um tempo máximo abaixo dos 3s e um tempo médio abaixo dos 2s, não existindo requisitos de performance definidos nem valores de referência ou de outros sistemas para comparação, isto é, analisando os valores apenas por si só, pode considerar-se que os tempos de resposta do sistema são aceitáveis.

6. Conclusões

Neste capítulo será realizada uma análise ao trabalho realizado com apresentação de algumas das dificuldades encontradas no decorrer do projeto terminando com algumas sugestões de trabalho futuro para a evolução do SOASales.

6.1. Revisão do trabalho realizado

Neste projeto procurou-se idealizar um sistema SOA que apresentasse um conjunto alargado de características associadas a este tipo de arquitetura de forma a que pudesse ser considerado representativo. Adicionalmente, pretendia-se a sua implementação, mesmo que parcial.

Após um trabalho inicial de pesquisa e familiarização com o tema foram identificadas algumas características deste tipo de sistema tendo então sido iniciado o trabalho de desenho do SOASales, que engloba as organizações Vendas Online e Livros Online. No processo de desenho foram sendo identificadas características e tecnologias a representar no sistema, sendo algumas decisões tomadas apenas na fase de implementação, como por exemplo o ESB a utilizar ou as linguagens de implementação dos serviços.

Após a especificação do sistema, foi iniciado o processo de implementação, tendo sido tomada a decisão de que este seria iterativo, ou seja, seriam selecionados alguns cenários para iniciar a implementação, os de pesquisa e consulta de produtos, e só após a conclusão destes seria tomada a decisão de implementar ou não cenários adicionais.

Após a implementação da primeira versão dos cenários e em diversas alturas da implementação o sistema foi analisado com o intuito de identificar melhorias do ponto de vista das características e funcionalidades do sistema que acrescentassem valor ao projeto. Foi a partir destas análises que algumas decisões de implementação foram tomadas. Após a implementação ter sido dada como terminada foram realizados alguns testes de forma a obter alguns valores em termos de tempos de resposta do sistema.

Em conclusão, considerando o objetivo deste projeto, o SOASales idealizado e a implementação realizada tendo como base de comparação os exemplos encontrados e apresentados no capítulo 2.6, somos em crer que o SOASales vai de encontro ao objetivo proposto e que pode ser considerado representativo de um SOA.

6.2. Lições Aprendidas

No decorrer deste projeto com o progresso do trabalho, o contacto com novas tecnologias, as dificuldades encontradas e a sua superação foram-se aprendendo algumas lições. Nesta secção iremos apresentar algumas dessas lições.

Ao iniciar este projeto uma das primeiras tarefas foi a familiarização com o tema das arquiteturas orientadas a serviços o que incluiu não só a procura de informação generalizada sobre o tema, mas também de informação mais concreta como casos de estudo e exemplos de organizações e sistemas com este tipo de arquitetura. Esta tarefa permitiu o contacto com muita informação e um conhecimento considerável sobre o assunto. No entanto apresentou algumas dificuldades, por um lado pela quantidade de informação disponível tornando-se moroso selecionar a informação relevante, e por outro

pela pouca informação referente a exemplos concretos acompanhados de exemplificação ou descrição do sistema, da sua arquitetura e das tecnologias envolvidas.

O contacto com novas tecnologias e ferramentas, como o Mule ESB que foi selecionado para base do sistema, foi um processo de aprendizagem que permitiu a aquisição de novas competências tecnológicas. No entanto como seria de esperar no contacto com uma nova ferramenta e/ou tecnologia, existe sempre uma curva de aprendizagem e por abundante e de qualidade que seja a documentação, esta tem sempre que ser entendida e adaptada para o caso particular pretendido. O processo de implementação dos fluxos do ESB foi evolutivo tendo sido iniciado com algumas experiências para perceber o funcionamento, só depois passando à implementação em si, tendo em algumas ocasiões os fluxos sido refeito ou alterados com o avançar do projeto e um melhor entendimento do seu funcionamento.

Por razões profissionais a disponibilidade para a realização do projeto não era total, tendo originado algumas dificuldades ao nível de tempo disponível para a execução das tarefas, ritmo de trabalho e motivação. A gestão de tempo foi uma componente bastante importante para a execução deste projeto.

6.3. Trabalho futuro

O sistema idealizado na fase inicial deste projeto não foi implementado na sua totalidade, tendo apenas sido implementados os cenários de pesquisa e obtenção de detalhes de produtos. A implementação dos restantes cenários, de realização de transações de compra e de gestão de conta, seria uma mais-valia para o sistema por lhe acrescentar complexidade e uma maior diversidade de cenários, tais como a autenticação e a utilização de um serviço de *registry*.

Os serviços implementados, ProductsService e ListingService, apesar de pertencerem a organizações distintas, encontram-se instalados na mesma máquina de desenvolvimento. A deslocação do ListingService para uma máquina distinta, possivelmente com um sistema operativo também diferente, tornaria o sistema mais realista e adicionaria uma nova componente ao nível das tecnologias em uso.

A adição de novas organizações, vendedores, ao sistema seria também uma mais-valia para o sistema.

Adicionalmente podem ser idealizados novos cenários de forma a adicionar novas tecnologias e características comuns de ambientes SOA.

Referências Bibliográficas

[1] T. Erl, *Service-oriented architecture: concepts, technology, and design*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005.

[2] M. P. Papazoglou e W.-J. van den Heuvel, «Service oriented architectures: approaches, technologies and research issues», *The VLDB Journal*, vol. 16, n. 3, pp. 389–415, Jul. 2007.

[3] H. He, «What is service-oriented architecture», *Publicação eletrônica em*, vol. 30, pp. 1–5, 2003.

[4] T. Yoon e P. Carter, «Investigating the antecedents and benefits of SOA implementation: a multi-case study approach», *AMCIS 2007 Proceedings*, p. 195, 2007.

[5] «iTGrow». [Em linha]. Disponível em: http://www.itgrow.pt/pt/inicio. [Acedido: 09-Dez-2015].

[6] «Dependable Technologies for Critical Systems», *CRITICAL Software*. [Em linha]. Disponível em: http://www.criticalsoftware.com/pt/homepage. [Acedido: 09-Dez-2015].

[7] W. Vegter, «Critical success factors for a SOA implementation», em *11th Twente Student Conference on IT, Enschede, June 29th*, 2009.

[8] «Web Services Architecture - networkservicesandlayeredarchitecturenotes2.pdf». .

[9] Y. V. Natis, *Service-oriented architecture scenario*. 2003.

[10] «Web Services Architecture». [Em linha]. Disponível em: http://www.w3.org/TR/ws-arch/#whatis. [Acedido: 25-Fev-2015].

[11] M. P. Papazoglou e W.-J. van den Heuvel, «Service-Oriented Computing: State-of-the-Art and Open Research Issues», *IEEE Computer. v40 i11*, 2003.

[12] M. Greiler, H.-G. Gross, e K. A. Nasr, «Runtime integration and testing for highly dynamic service oriented ICT solutions–An Industry Challenges Report», em *Testing: Academic and Industrial Conference-Practice and Research Techniques, 2009. TAIC PART'09.*, 2009, pp. 51–55.

[13] «Open SOALab - Supporting Research and Teaching of Services Oriented Architecture». [Em linha]. Disponível em: http://uwf.edu/nwilde/soaResources/. [Acedido: 23-Mar-2015].

[14] C. Areias, N. Antunes, e J. C. Cunha, «On Applying FMEA to SOAs: A Proposal and Open Challenges», em *Software Engineering for Resilient Systems*, Springer, 2014, pp. 86–100.

[15] C. Areias, N. Antunes, J. Cunha, e M. Vieira, «Towards Runtime V&V for Service Oriented Architectures».

[16] A. Ceccarelli, M. Vieira, e A. Bondavalli, «A Service Discovery Approach for Testing Dynamic SOAs», 2011, pp. 133–142.

[17] D. F. García, J. García, M. García, I. Peteira, R. García, e P. Valledor, «Benchmarking of web services platforms», em *Proceedings of 2nd International Conference on Web Information Systems and Technologies, WEBIST*, 2006, pp. 75–80.

[18] «Ten examples of SOA at work in 2010 | ZDNet». [Em linha]. Disponível em: http://www.zdnet.com/article/ten-examples-of-soa-at-work-in-2010/. [Acedido: 03-Jan-2015].

[19] «Why Health Care Needs SOA | K. Scott Morrison's Blog». [Em linha]. Disponível em: http://kscottmorrison.com/2010/08/11/why-health-care-needs-soa/. [Acedido: 31-Jan-2015].

[20] «HSSP - PracticalGuide». [Em linha]. Disponível em: http://hssp.wikispaces.com/PracticalGuide. [Acedido: 14-Mar-2015].
[21] «The University of Chicago Medicine». [Em linha]. Disponível em: http://www.uchospitals.edu/index.shtml. [Acedido: 05-Dez-2015].
[22] «Layer 7 Technologies is Now the CA API Management Suite - CA Technologies». [Em linha]. Disponível em: http://www.ca.com/us/lpg/layer-7-redirects.aspx. [Acedido: 05-Dez-2015].
[23] «Semana Informática». [Em linha]. Disponível em: http://www.semanainformatica.xl.pt/sector-financeiro/sector-financeiro/plataforma-segurnet-promove-cooperacao-no-sector-dos-seguros. [Acedido: 04-Mar-2015].
[24] «Three SOA Case Studies». [Em linha]. Disponível em: http://www.slideshare.net/pizak/three-soa-case-studies?related=2. [Acedido: 03-Jan-2015].
[25] «CRITICAL Software - Oversee». [Em linha]. Disponível em: http://www.criticalsoftware.com/pt/products/p/oversee. [Acedido: 10-Mar-2015].
[26] «Home - Oversee». [Em linha]. Disponível em: http://www.oversee-solutions.com/. [Acedido: 10-Mar-2015].
[27] «How SOA enables transition of companies into cloud service providers | ZDNet». [Em linha]. Disponível em: http://www.zdnet.com/article/how-soa-enables-transition-of-companies-into-cloud-service-providers/. [Acedido: 31-Jan-2015].
[28] «The “two-second advantage”: real time gets real for many organizations | ZDNet». [Em linha]. Disponível em: http://www.zdnet.com/article/the-two-second-advantage-real-time-gets-real-for-many-organizations/. [Acedido: 31-Jan-2015].
[29] «@ Tibco TUCON: Real-time Retail, Using Information to Improve Customer Satisfaction - Business-Driven Architect». [Em linha]. Disponível em: http://www.ebizq.net/blogs/bda/2010/05/_tibco_tucon_real-time_retail.php. [Acedido: 31-Jan-2015].
[30] «MuleSoft», *MuleSoft*. [Em linha]. Disponível em: https://developer.mulesoft.com/. [Acedido: 25-Out-2015].
[31] V. Graaff, «Towards distributed information access: possibilities and implementation», 2009.
[32] «Mule: A Case Study». [Em linha]. Disponível em: http://www.theserverside.com/news/1365047/Mule-A-Case-Study. [Acedido: 12-Ago-2015].
[33] «Our Company - eBay Inc.» [Em linha]. Disponível em: https://www.ebayinc.com/our-company/. [Acedido: 05-Dez-2015].
[34] «Wits University - Wits University». [Em linha]. Disponível em: http://www.wits.ac.za/. [Acedido: 05-Dez-2015].
[35] «About TiVo Inc. | Pioneer DVR and TV Streaming». [Em linha]. Disponível em: https://www.tivo.com/about. [Acedido: 05-Dez-2015].
[36] «UCSF Medical Center». [Em linha]. Disponível em: http://www.ucsfhealth.org/. [Acedido: 05-Dez-2015].
[37] «Unicef». [Em linha]. Disponível em: http://www.unicef.pt/. [Acedido: 05-Dez-2015].
[38] «Apache ActiveMQ ™ -- Index». [Em linha]. Disponível em: http://activemq.apache.org/. [Acedido: 22-Jan-2015].
[39] «DataTables | Table plug-in for jQuery». [Em linha]. Disponível em: https://www.datatables.net/. [Acedido: 06-Nov-2015].

[40] «Bootstrap · The world’s most popular mobile-first and responsive front-end framework.» [Em linha]. Disponível em: http://getbootstrap.com/. [Acedido: 06-Nov-2015].
[41] A. P. Moore, R. J. Ellison, e R. C. Linger, «Attack modeling for information security and survivability», DTIC Document, 2001.
[42] «Apache Tomcat - Welcome!» [Em linha]. Disponível em: http://tomcat.apache.org/. [Acedido: 04-Dez-2015].
[43] «SoapUI | Functional Testing for SOAP and REST APIs». [Em linha]. Disponível em: http://www.soapui.org/. [Acedido: 04-Dez-2015].

Anexos

- Anexo A – Especificação do Sistema
- Anexo B – Fluxos do Mule ESB
- Anexo C – Testes ao Sistema
- Anexo D – Código Fonte Aplicações
- Anexo E – Aplicações

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Mestrado em Informática e Sistemas
Estágio/Projeto Industrial
Relatório Final

Anexo A – Especificação do Sistema

Carla Maria Silva Machado

Orientadores:

João Carlos Costa Faria da Cunha

Cristiana Manuela Afonso Areias

Instituto Superior de Engenharia de Coimbra

Coimbra, dezembro, 2015

Índice

Índice de Figuras

1. Introdução

A idealização do caso de estudo, SOASales, incluiu a definição de uma área de negócio, de organizações envolvidas, da especificação dos serviços de cada organização e especificação do sistema.

Neste documento apresenta-se a especificação do sistema sendo apresentada uma visão geral de sistema e as várias funcionalidades disponibilizadas pelo mesmo. A apresentação de cada funcionalidade é realizada com recurso a diagramas e à descrição de um cenário.

2. Visão geral do Sistema

Neste capítulo será apresentado uma visão geral do sistema idealizado para satisfazer as necessidades da organização Vendas Online, Figura 2.1.

A organização possui um *service bus* que serve de ponto de ligação entre a organização e o exterior, sendo o ponto de entrada do sistema. O *service bus* é responsável não só pelo reencaminhamento de pedidos, mas também por alguma da lógica de negócio.

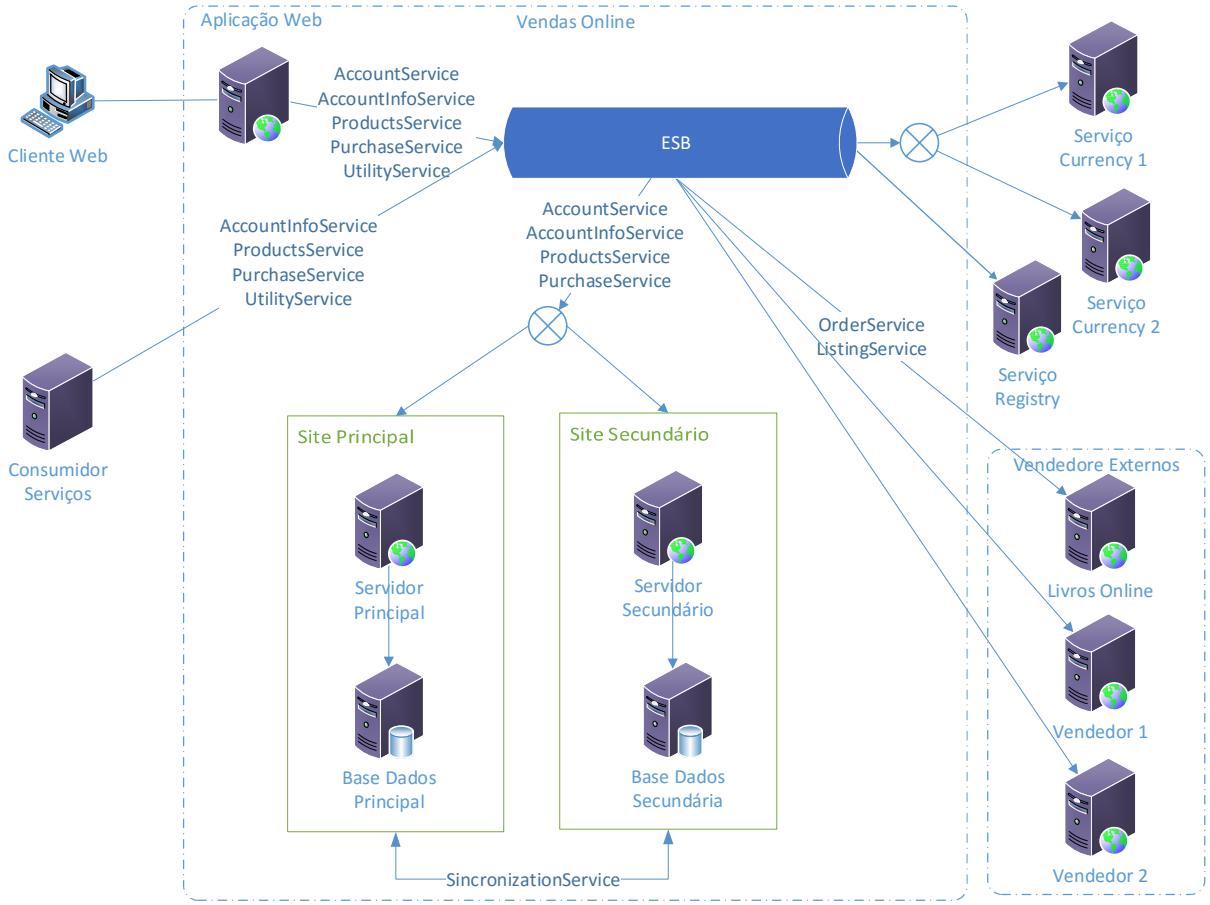

Figura 2.1: Ambiente SOA Vendas Online - Visão Geral Serviços

Na Figura 2.2 apresentamos uma versão mais detalhada do esquema anterior evidenciando os serviços de cada organização. Por uma questão de simplificação do esquema não se encontra representada a duplicação da estrutura por duas localizações.

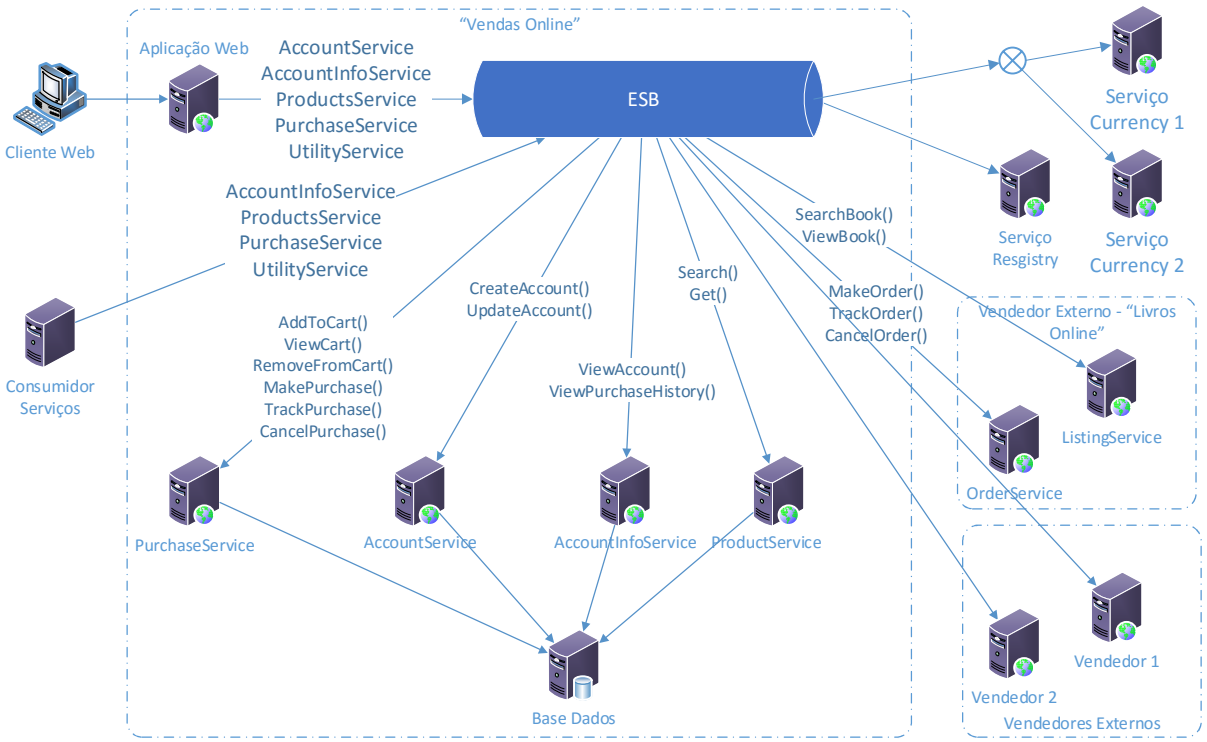

Figura 2.2: Ambiente SOA Vendas Online - Visão Geral Serviços

Observado os esquemas podemos observar a disponibilização de 5 serviços pela organização a partir do Service Bus, sendo que destes apenas 4 deverão ser acessíveis por entidades externas à organização.

3. Funcionalidades

De seguida iremos particularizar e especificar as diferentes funcionalidades do sistema apresentando esquemas ilustrativos e descrições dos cenários.

3.1. Pesquisa de Produtos

Neste capítulo serão apresentados os esquemas e diagramas, Figura 3.1 e Figura 3.2, relativos ao cenário da Pesquisa de produtos.

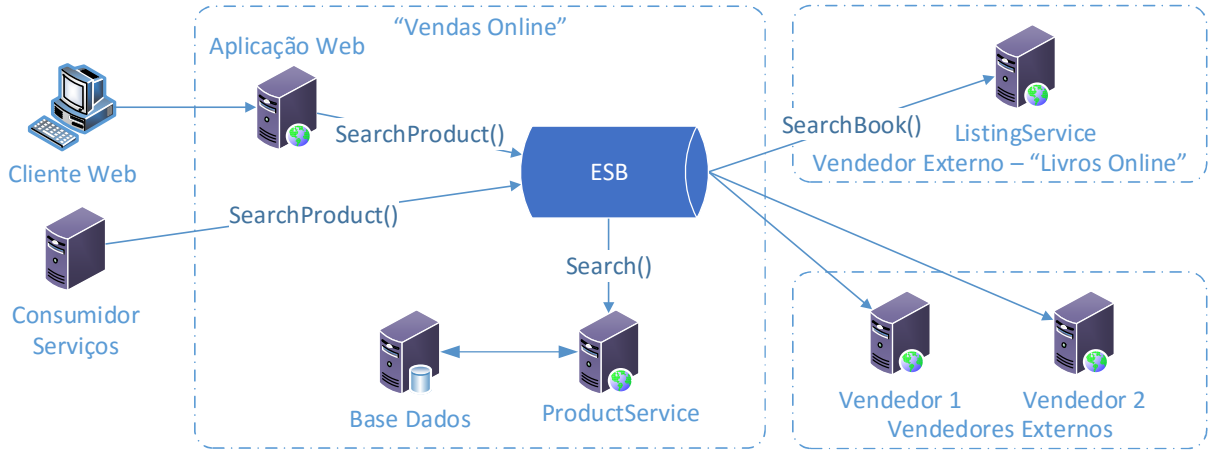

Figura 3.1: Orquestração de Serviços – Pesquisa de Produtos

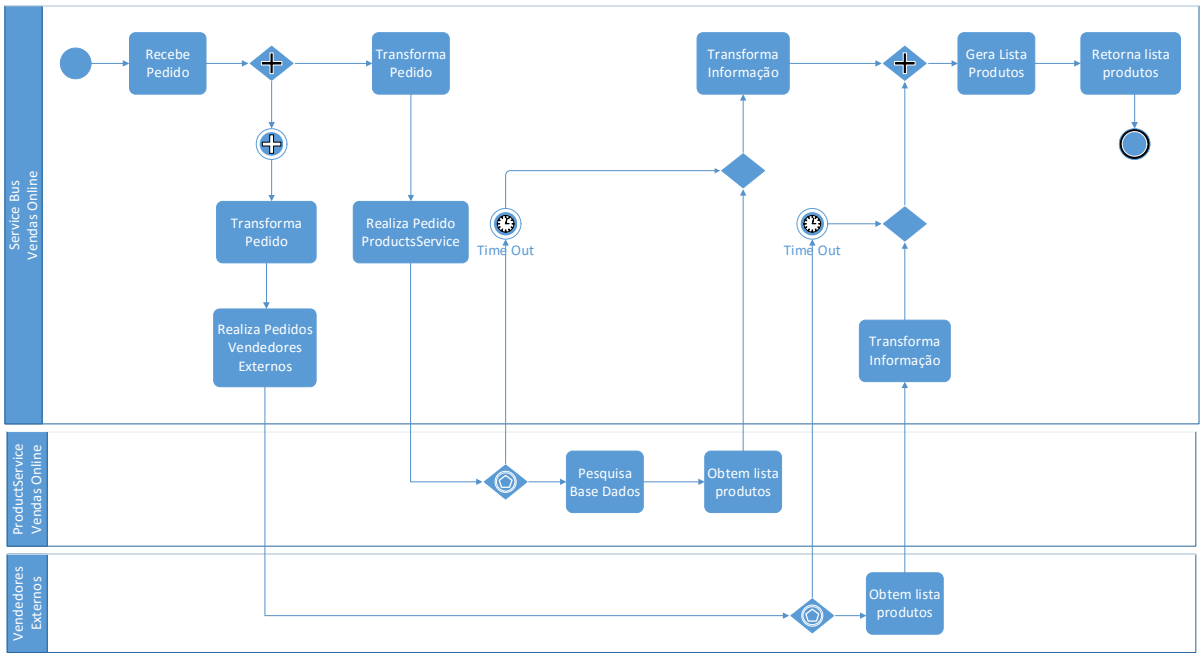

Figura 3.2: Diagrama BPMN – Pesquisa Produtos

Observando o cenário representado nos esquemas podemos verificar que os clientes da organização Vendas Online podem aceder à pesquisa de produtos de duas formas distintas quer por meio da aplicação web que realiza posteriormente o pedido ao *service bus* ou fazer o pedido diretamente a este, sendo que o pedido deverá conter as palavras a pesquisar.

O *service bus* ao receber o pedido realiza o seu processamento de forma paralela para o ProductsService da organização e para os vendedores externos incorporados no fluxo. O mesmo pedido será reencaminhado simultaneamente para todos os serviços fazendo o processamento necessário para que a mensagem tenha o formato correto para cada um.

O *service bus* aguarda pelas respostas de todos os serviços ou por um *time out*, agregando as respostas obtidas com sucesso numa única, tendo as respostas de cada serviço sido previamente transformadas para um formato comum.

A resposta com os resultados combinados da pesquisa realizada nos vários serviços é então retornada para o cliente.

3.2. Comparação de Produtos

Neste capítulo serão apresentados os esquemas e diagramas, Figura 3.3 e Figura 3.4 , relativos ao cenário de comparação de produtos.
Ao observar a Figura 3.3 e comparando com a Figura 3.5 uma constatação que pode ser realizada é o facto de os serviços das organizações consumidos na obtenção de detalhes e na comparação são os mesmos, no entanto o ponto de entrada dos fluxos e a funcionalidade disponibilizada são diferentes.

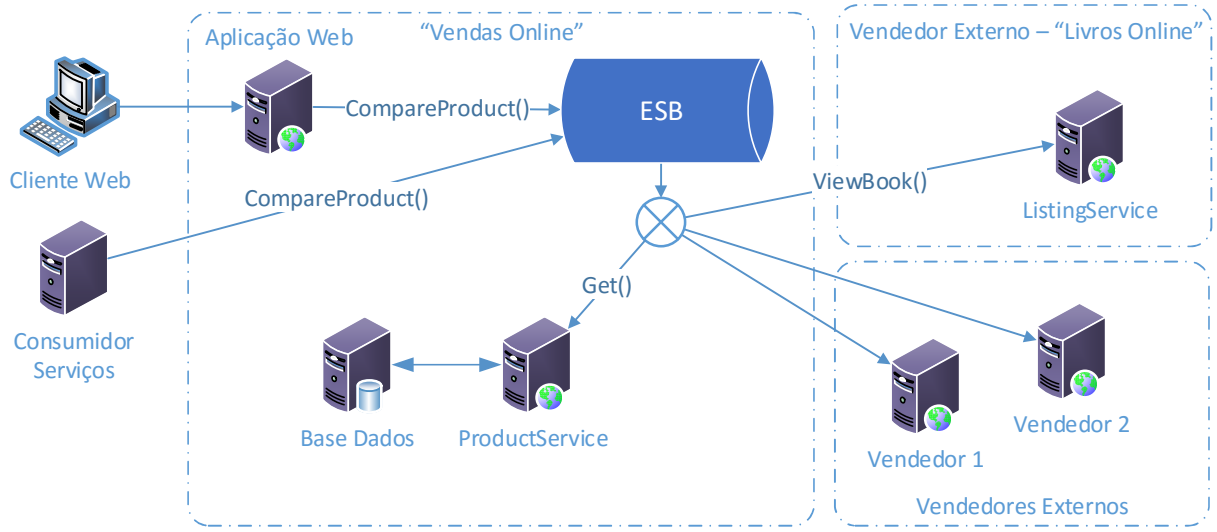

Figura 3.3: Orquestração de Serviços – Comparar Produtos

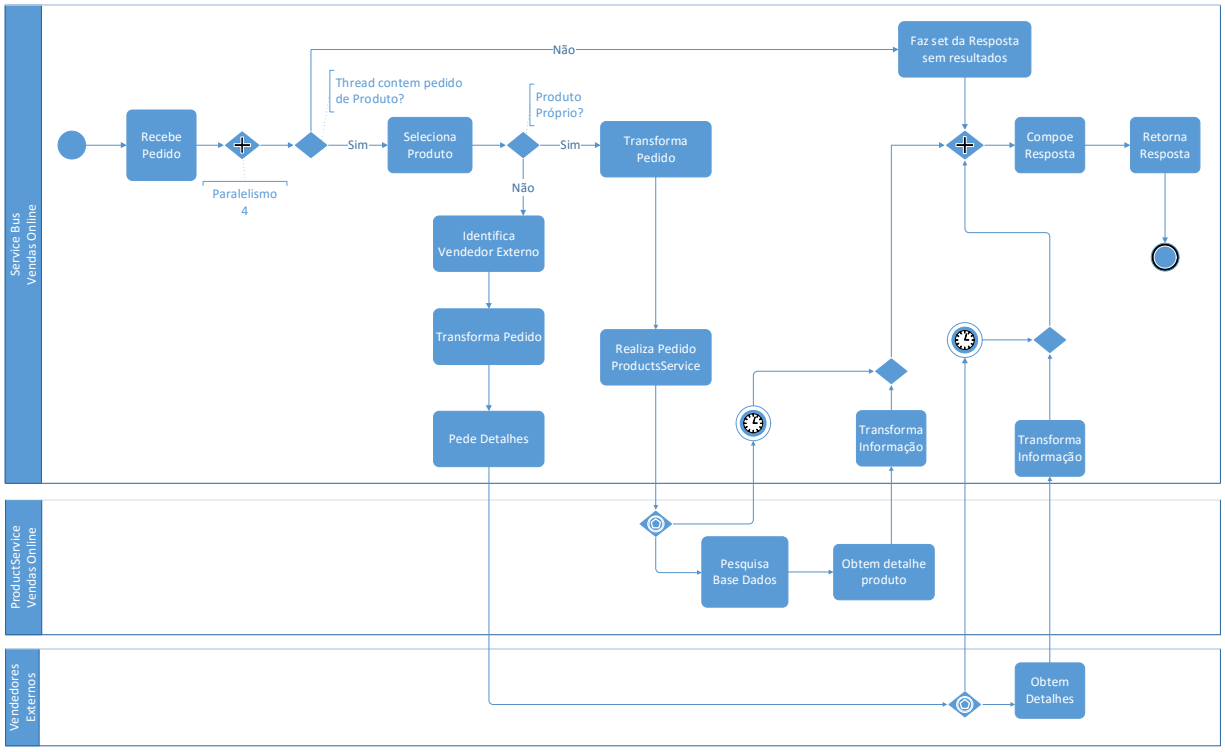

Figura 3.4: Diagrama BPMN – Comparar Produtos

Observando os esquemas podemos verificar que tal como no caso anterior o pedido de detalhes de um produto pode ser feito por meio da aplicação web ou diretamente ao *service Bus*, o pedido deve conter a informação identificadora do produto e do seu vendedor.

O *service bus* ao receber um pedido começa por identificar o vendedor do produto, e se este se trata de um produto da própria organização ou de uma organização externa. De acordo com esta determinação o fluxo seguirá percursos alternativos.

Caso de trate de um produto próprio, o *service bus* irá realizar um pedido ao ProductsService da organização fazendo a transformação necessária ao pedido recebido para realizar a chamada ao método pretendido do ProductsService.

O ProductsService por sua vez ao receber um pedido de detalhe de produto vai pesquisar o identificador recebido na base de dados e retornar, caso exista, a informação do produto correspondente.

Caso se trate de um produto de uma organização externa, o *service bus* identifica a organização e transforma o pedido recebido para que este tenha o formato necessário para a realização da chamada à organização externa.

Após a *service bus* receber a resposta do serviço invocado processa e transforma a mensagem recebida e retorna a resposta.

3.3. Obtenção de detalhe de Produto

Neste capítulo serão apresentados os esquemas e diagramas, Figura 3.5 e Figura 3.6, relativos ao cenário de obtenção de detalhes de um produto.

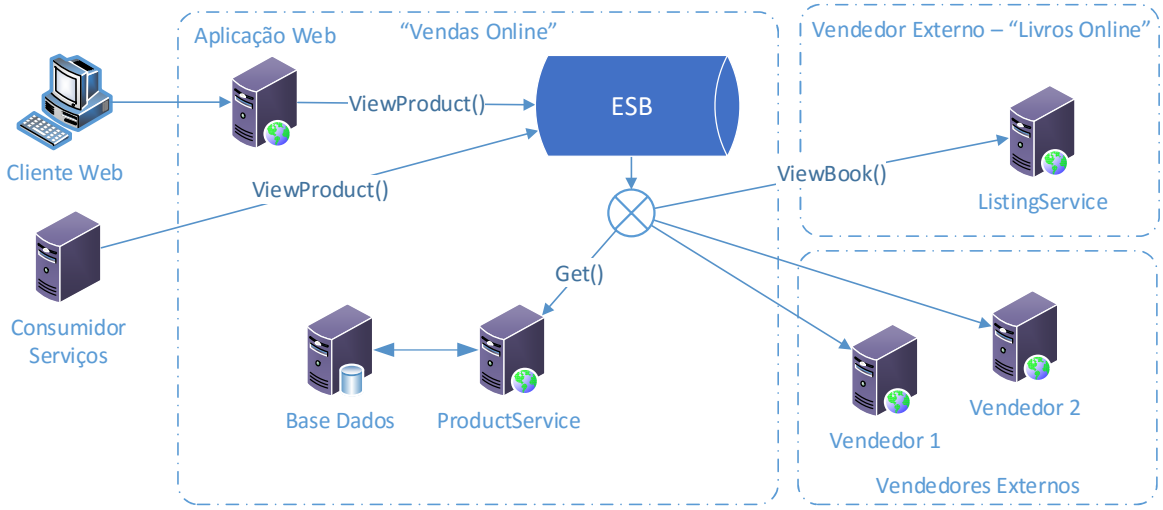

Figura 3.5: Orquestração de Serviços – Detalhe de Produto

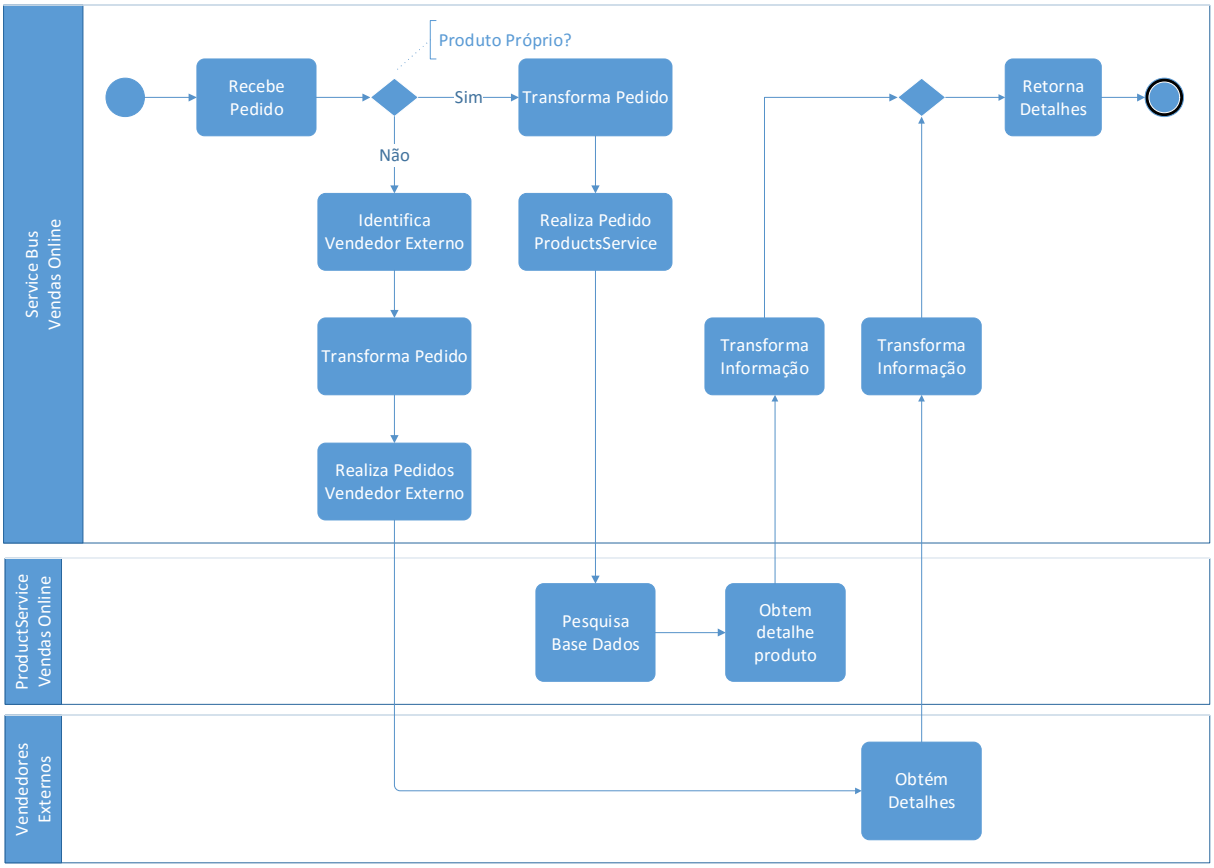

Figura 3.6: Diagrama BPMN – Detalhe de Produto

Ao receber o pedido, o *service bus* vai iniciar um processamento em paralelo, utilizando quatro *threads*. Para cada *thread* é avaliado se o pedido contém o identificador de um produto. Caso este não exista, é feito o set da resposta para o equivalente de sem resultados; caso exista, será realizado o processamento necessário para pedir os seus detalhes, executando para esse efeito

o mesmo fluxo do pedido de detalhes. No processamento de cada *thread* após a obtenção da resposta, esta será transformada para um formato próprio e comum a todas as *thread* e informação do vendedor adicionado.

O *service bus* aguarda que todas as *threads* sejam executadas, o que significa ter recebido respostas de todas ou ter atingido um time out, processando então todas as respostas obtidas com sucesso e agrupando os detalhes dos produtos obtidos isoladamente numa única resposta que é então retornada ao cliente.

3.4. Criação de Conta

Neste capítulo será apresentada a funcionalidade de criação de conta representado nas Figura 3.7 e Figura 3.8. Esta funcionalidade vai consumir um serviço privado da organização, por essa razão só se encontrará disponível através de aplicações da organização, como é o caso da aplicação web disponibilizada pela organização.

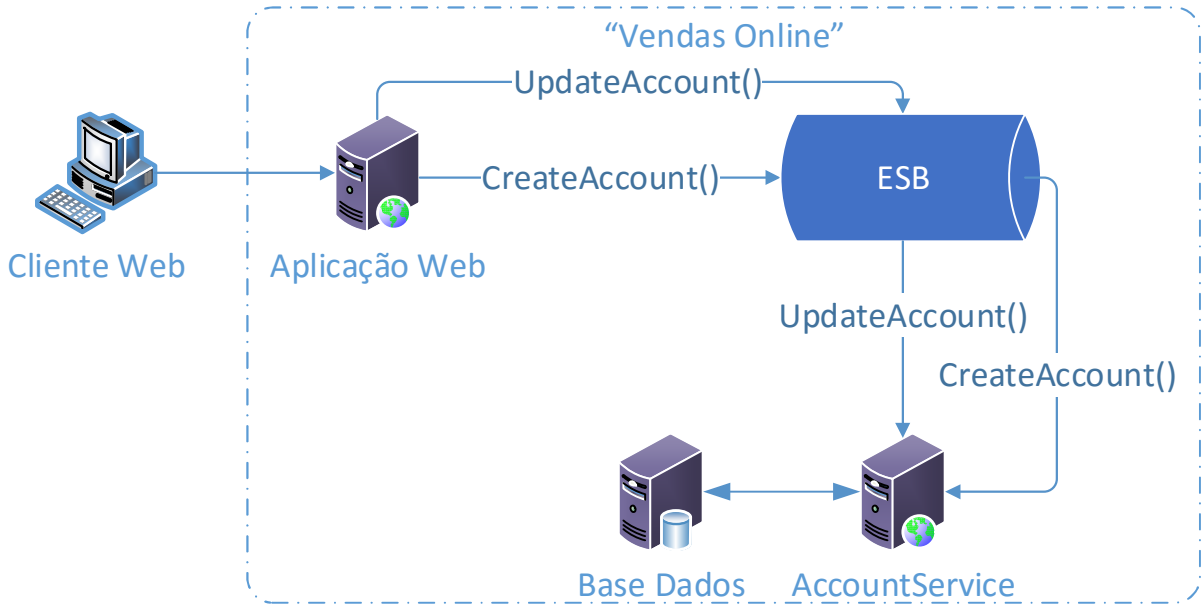

Figura 3.7: Orquestração de Serviços – Criar e Atualizar Conta

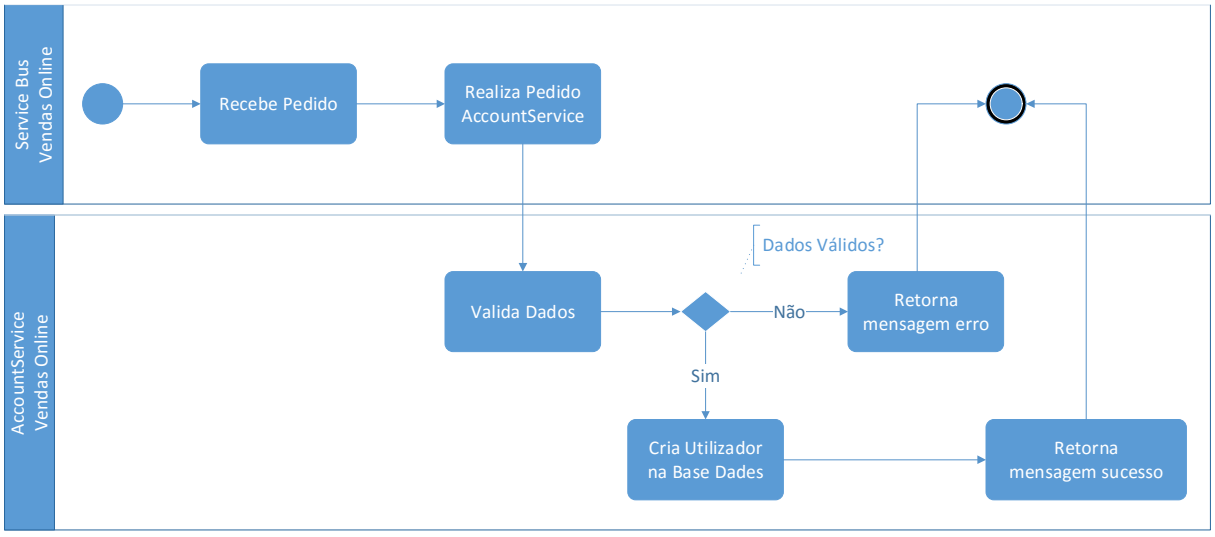

Figura 3.8: Diagrama BPMN – Criar Conta

Um utilizador acede à aplicação web e tem disponível a opção de criação de conta, ao submeter um pedido a aplicação web realiza o seu reencaminhamento para service bus.

O *service bus* por sua vez ao receber o pedido realiza o seu reencaminhamento para o AccountService da organização.

O AccountService ao receber o pedido realiza as validações necessárias à informação contida no pedido. Se o pedido for validado o serviço acede à base de dados inserindo um novo registo correspondente ao novo utilizador e retorna uma mensagem de sucesso caso contrário é retornada uma mensagem de erro.

3.5. Atualização de Conta

Neste capítulo será apresentada a funcionalidade de atualização de conta representado nas Figura 3.7 e Figura 3.9. Tal como a criação de conta esta funcionalidade apenas esta disponível por meio de aplicações da organização.

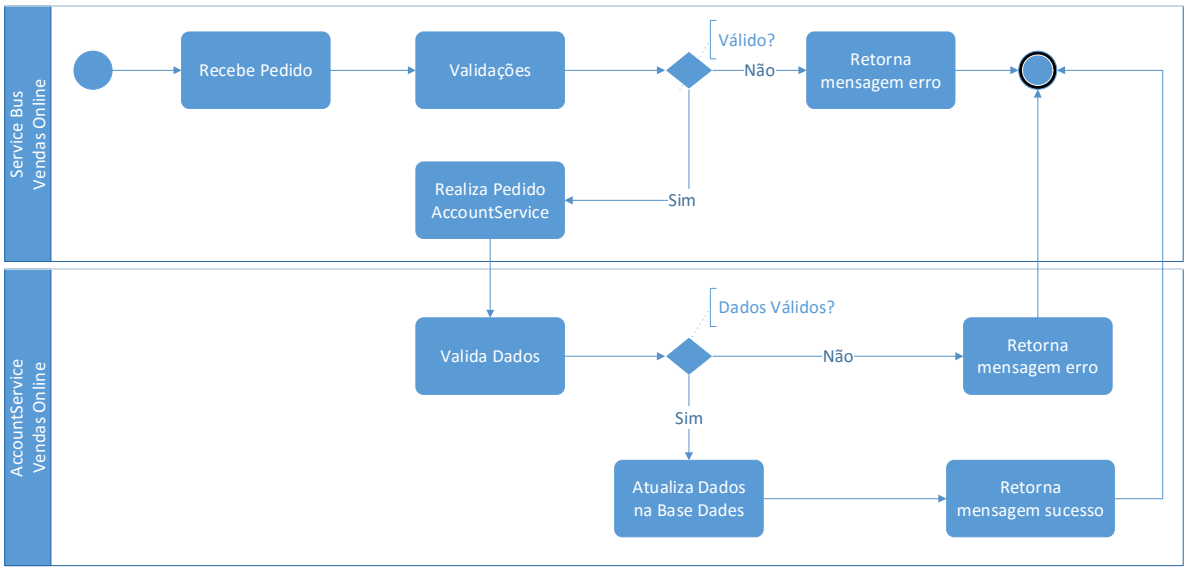

Figura 3.9: Diagrama BPMN – Atualizar Conta

Um utilizador que já possua uma conta de utilizador ao aceder à aplicação web tem a opção de se autenticar, ficando então com a funcionalidade de atualização de conta disponível. Ao submeter um pedido de atualização a aplicação vai reencaminhar o pedido para o *service bus*.

O *service bus* ao receber um pedido de atualização de conta vai realizar algumas validações nomeadamente de permissões, se o pedido não for válido retorna uma mensagem de erro se for válido realiza uma chamada ao AccountService.

O AccountService ao receber o pedido vai validar a informação contida no pedido e caso seja válida vai aceder à base de dados e atualizar o registo retornando então uma mensagem de sucesso, caso contrário retorna uma mensagem de erro.

3.6. Visualização de informação da Conta

Neste capítulo iremos apresentar o cenário da visualização de informação da conta, Figura 3.10 e Figura 3.11.

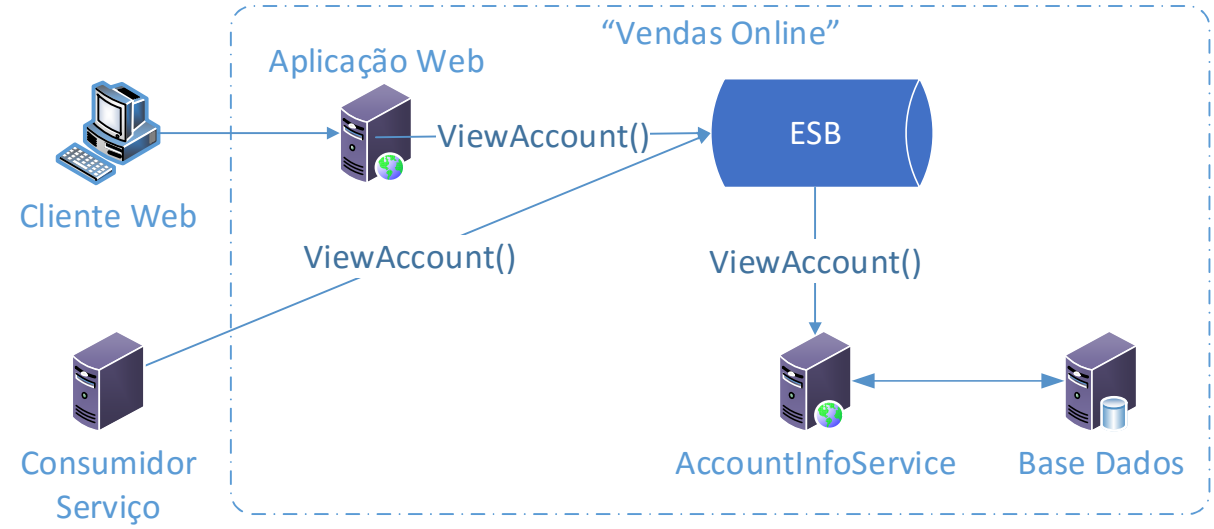

Figura 3.10: Orquestração de Serviços – Visualizar Conta

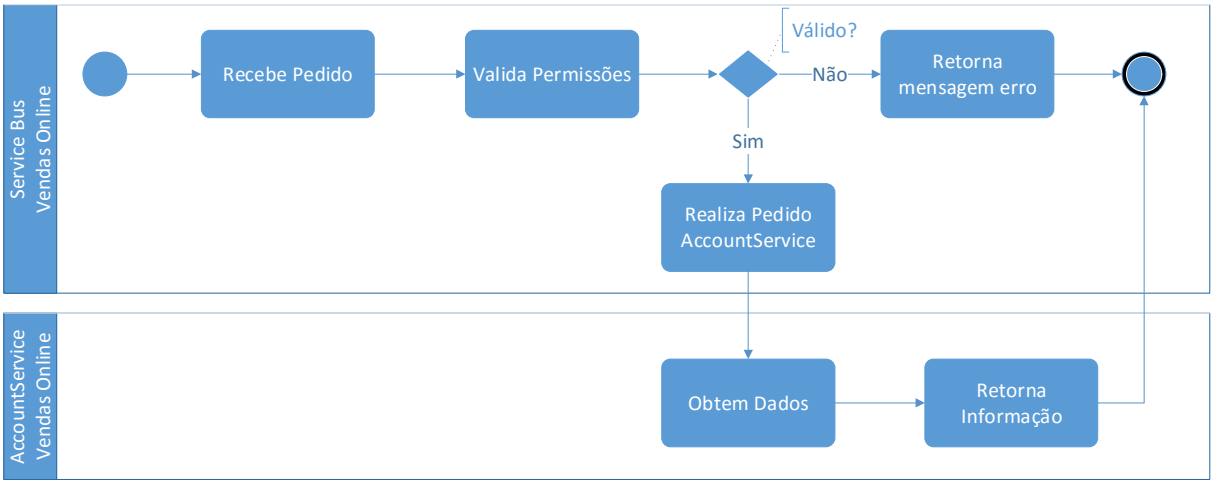

Figura 3.11: : Diagrama BPMN – Visualizar Conta

Como pode ser observado nos esquemas a visualização de informação de conta pode ser acedida pela aplicação web ou diretamente pelo service bus.

Um utilizador que possui conta acede à aplicação web e após realizar a autenticação tem disponível a funcionalidade de visualizar a sua informação de conta. A aplicação web ao ser submetido o pedido realiza o seu reencaminhamento para o service bus.

O service bus ao receber o pedido realiza uma validação de permissões de forma a confirmar que o utilizador pode aceder à informação. Se o utilizador tiver permissões para aceder à conta o service bus realiza um pedido ao serviço AccountInfoService da organização vendas online, caso contrário retorna uma mensagem de erro ao utilizador.

O AccountInfoService ao receber um pedido acede à base de dados de forma a obter a informação pretendida e retorna-a ao service bus que por sua vez a retorna ao utilizador.

3.7. Visualização do histórico de compras

Neste capítulo iremos apresentar o cenário da visualização de informação da conta, Figura 3.12 e Figura 3.13.

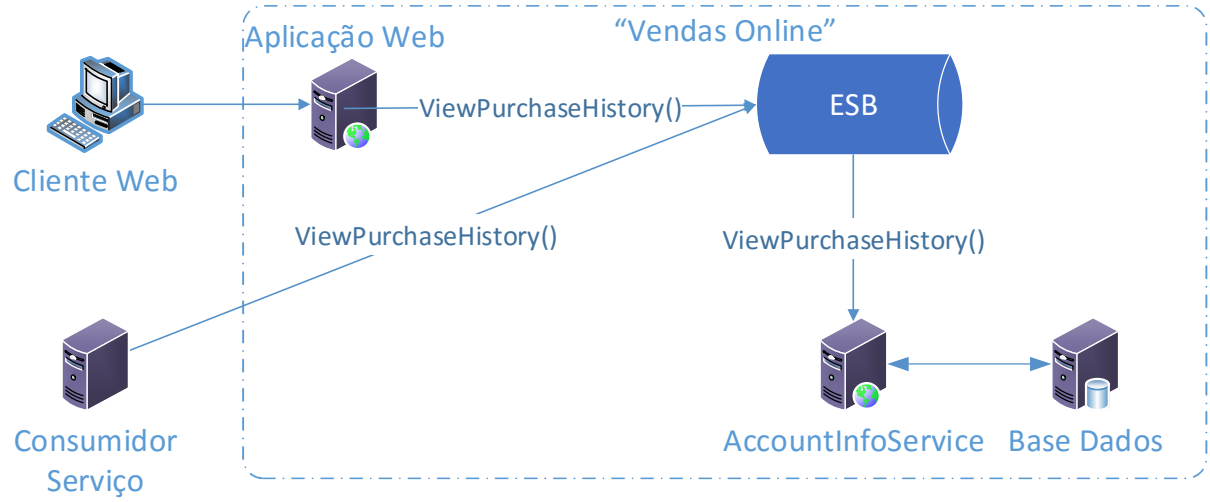

Figura 3.12: Orquestração de Serviços – Visualizar Histórico de Compras

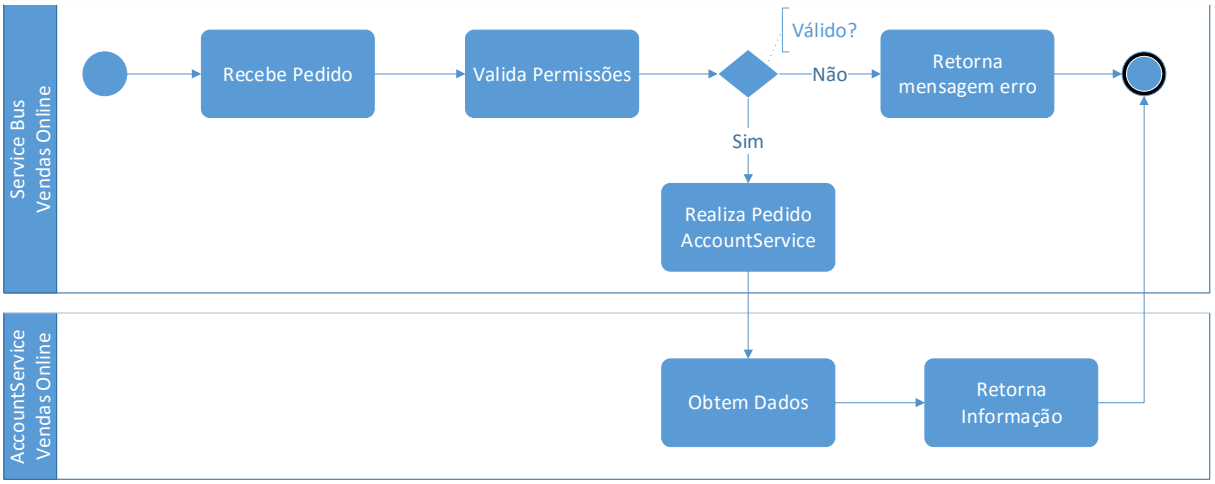

Figura 3.13: Diagrama BPMN – Visualizar Histórico de Compras

Observando os esquemas e comparando com os da visualização de informação de conta, verifica-se que são semelhantes, sendo a diferença a informação obtida da base de dados e retornada ao utilizador, no cenário anterior tratava-se de informação de conta e neste cenário trata-se de uma listagem de compras efetuadas anteriormente pelo utilizador.

Um utilizador que possui conta acede à aplicação web e após realizar a autenticação tem disponível a funcionalidade de visualizar a listagem de compras realizadas à organização. A aplicação web ao ser submetido o pedido realiza o seu reencaminhamento para o service bus.

O service bus ao receber o pedido realiza uma validação de permissões de forma a confirmar que o utilizador pode aceder à informação. Se o utilizador tiver permissões para aceder à conta o service bus realiza um pedido ao serviço AccountInfoService da organização vendas online, caso contrário retorna uma mensagem de erro ao utilizador.

O AccountInfoService ao receber um pedido acede à base de dados de forma a obter a informação pretendida e retorna-a ao service bus que por sua vez a retorna ao utilizador.

3.8. Adição de produto à lista de compras

Neste capítulo iremos apresentar o cenário da adição de um produto à lista de compras, Figura 3.14 e Figura 3.15.

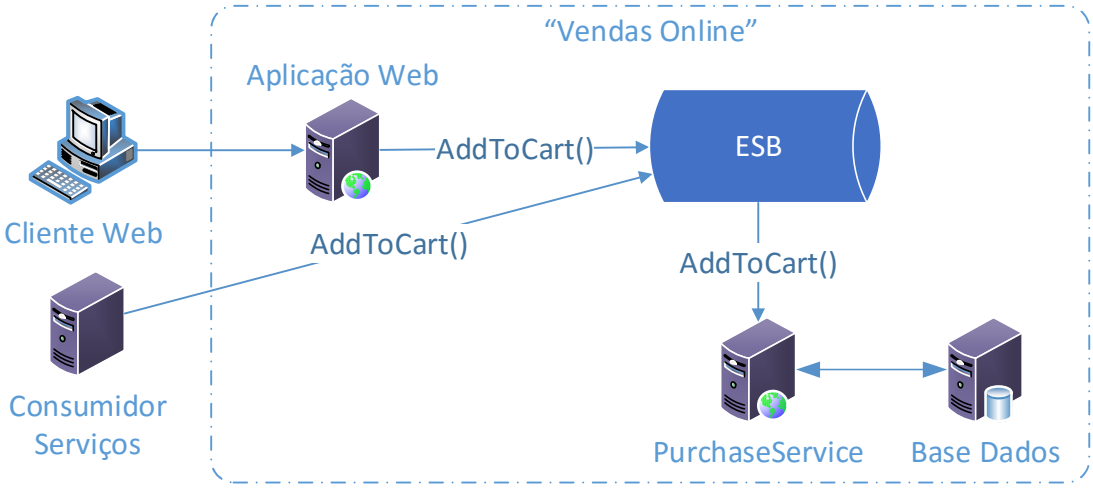

Figura 3.14: Orquestração de Serviços – Adicionar produto à lista de compras

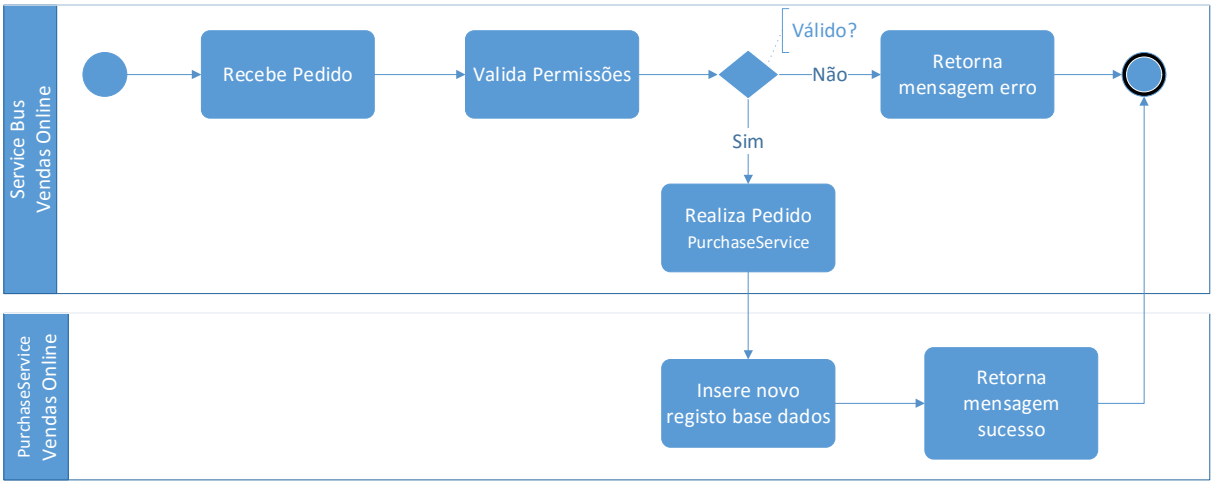

Figura 3.15: Diagrama BPMN – Adicionar produto à lista de compras

Um utilizador da aplicação web autentica-se na mesma e realiza uma pesquisa por produtos. Após a obtenção dos resultados, pode a partir da listagem ou dos detalhes de um dos produtos realizar a adição de um produto à lista de compras

A aplicação web ao ser submetido um pedido realiza uma chamada ao service bus.

O service bus ao receber um pedido valida as permissões do utilizador, se o pedido passar nas validações é realizado um pedido ao PurchaseService, caso contrário é retornada uma mensagem de erro.

O PurchaseService ao receber o pedido acede a base e insere um novo registo na listagem de produtos do utilizador e retorna uma mensagem de sucesso.

3.9. Visualização lista de compras

Neste capítulo iremos apresentar o cenário da adição de visualização de lista de compras, Figura 3.16 e Figura 3.17.

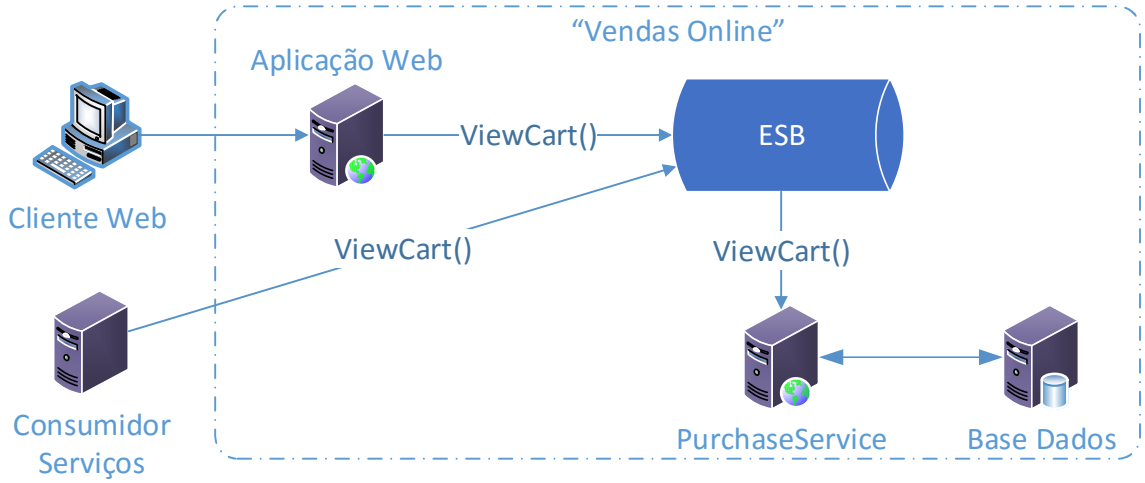

Figura 3.16: Orquestração de Serviços – Visualizar lista de compras

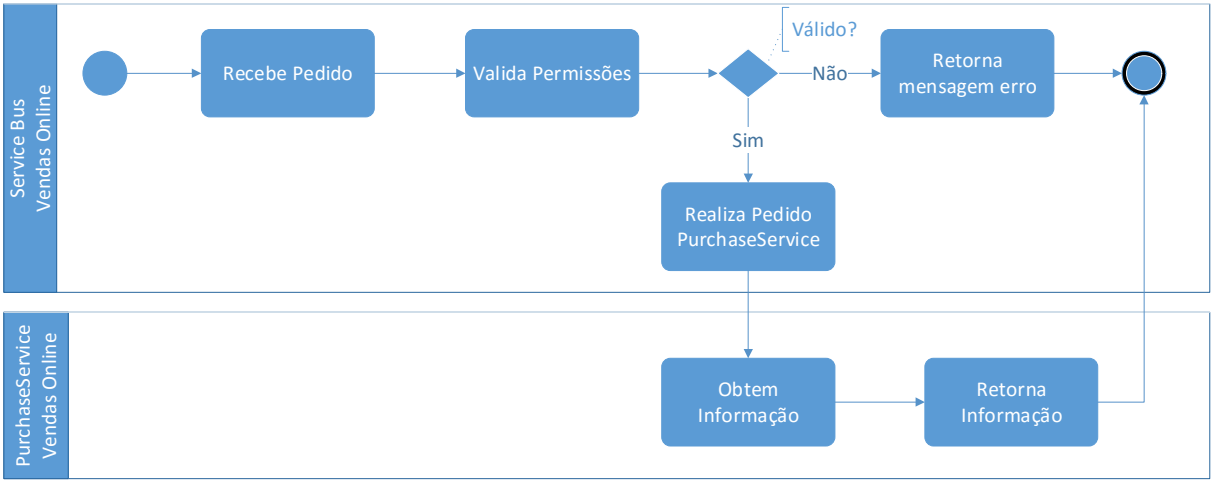

Figura 3.17: Diagrama BPMN – Visualizar lista de compras

Um utilizador da aplicação web autentica-se na mesma e tem a possibilidade de aceder à sua lista de compras, de forma a consultar os produtos incluídos.
A aplicação web ao ser submetido um pedido realiza uma chamada ao service bus.
O service bus ao receber um pedido valida as permissões do utilizador, se o pedido passar nas validações é realizado um pedido ao PurchaseService, caso contrário é retornada uma mensagem de erro.
O PurchaseService ao receber o pedido acede à base de dados e obtém a lista de compras do utilizador retornando à informação.

3.10. Remover produtos da lista de compras

Neste capítulo iremos apresentar o cenário da remoção de um produto à lista de compras, Figura 3.18 e Figura 3.19.

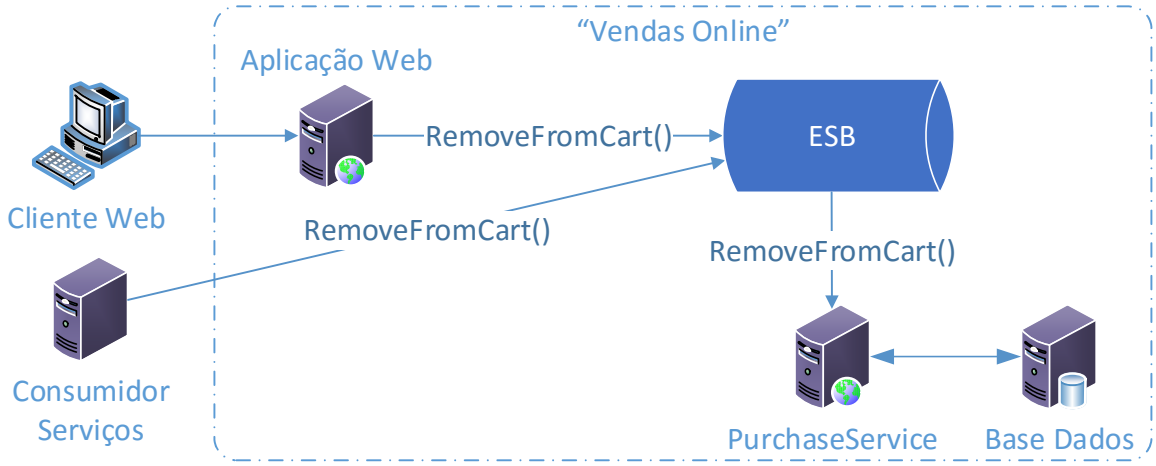

Figura 3.18: Orquestração de Serviços – Remover Produto da lista de compras

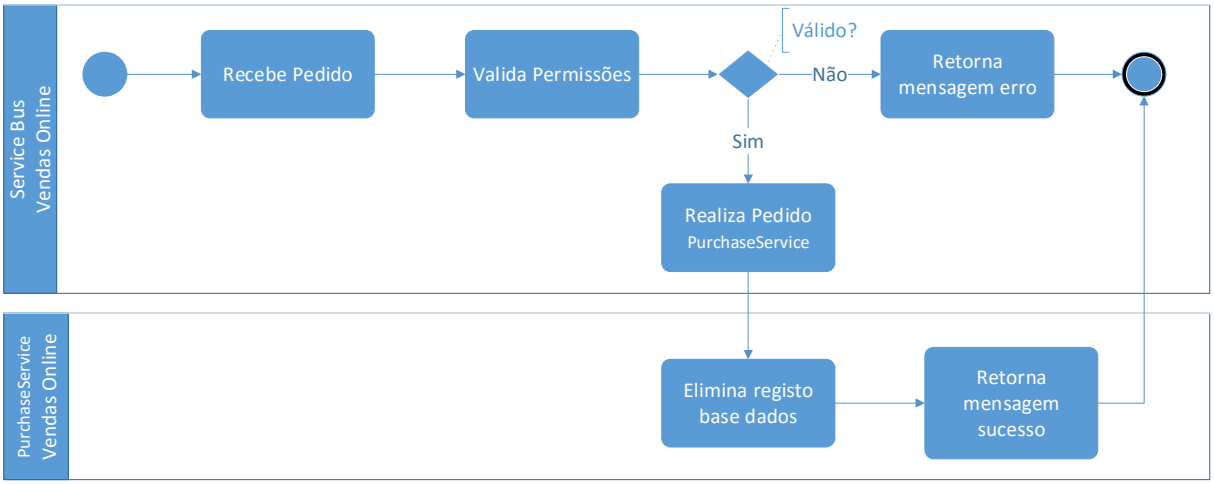

Figura 3.19: Diagrama BPMN – Remover produto da lista de compras

Um utilizador da aplicação web autentica-se na mesma e acede à sua lista de compras, a partir da listagem tem a possibilidade de remover produtos da listagem.
A aplicação web ao ser submetido um pedido realiza uma chamada ao service bus.
O service bus ao receber um pedido valida as permissões do utilizador, se o pedido passar nas validações é realizado um pedido ao PurchaseService, caso contrário é retornada uma mensagem de erro.
O PurchaseService ao receber o pedido acede a base e elimina o registo correspondente da listagem de produtos do utilizador retornando uma mensagem de sucesso.

3.11. Realizar compra

Neste capítulo iremos apresentar o cenário da remoção de um produto à lista de compras, Figura 3.20 e Figura 3.21.

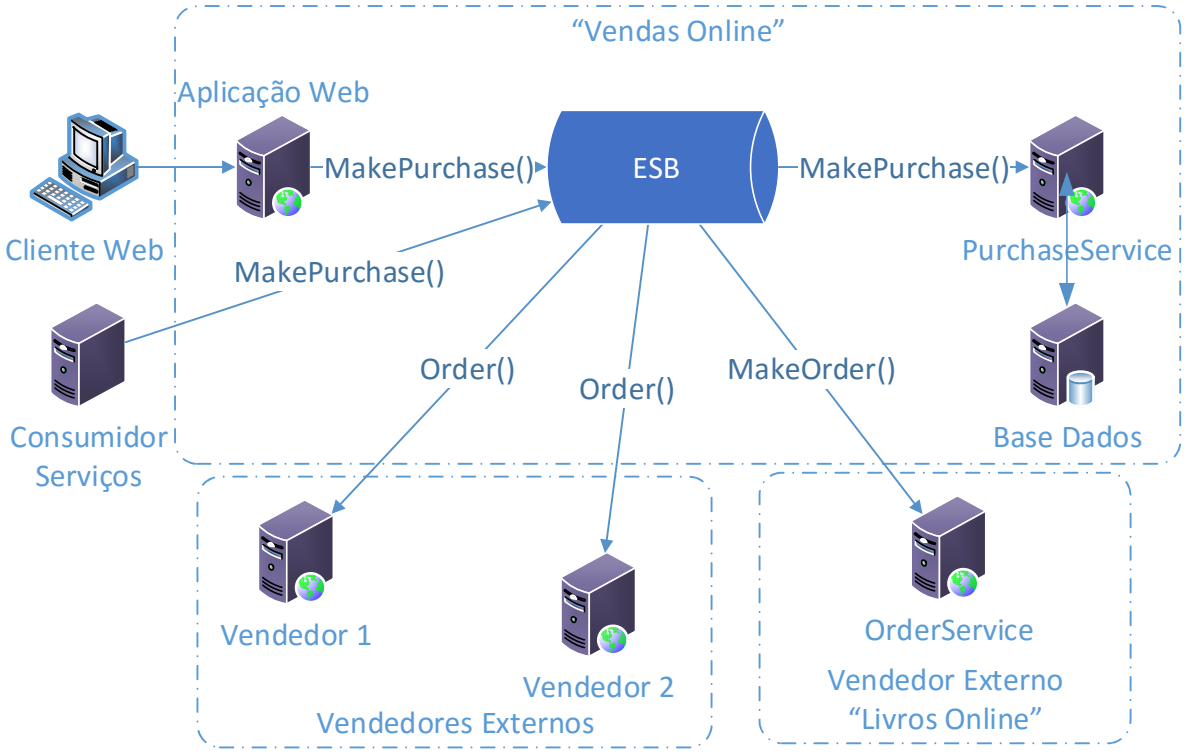

Figura 3.20: Orquestração de Serviços – Realizar Compra

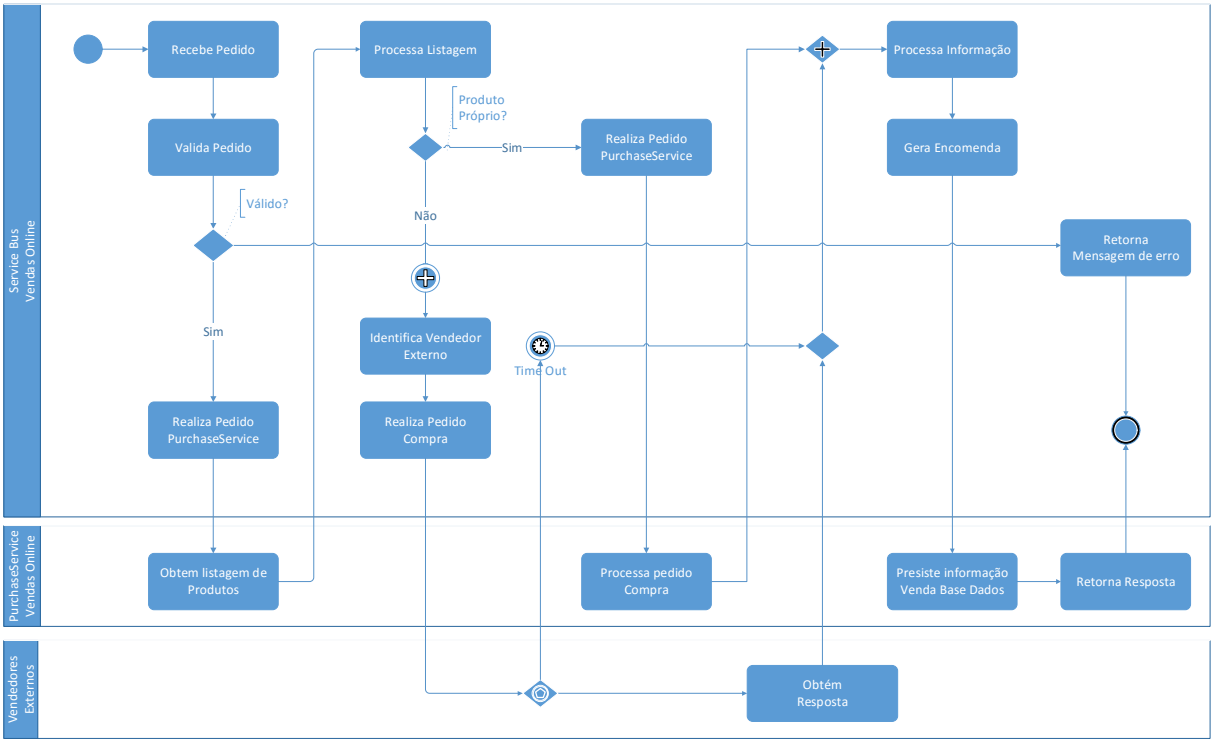

Figura 3.21: Diagrama BPMN – Realizar Compra

Um utilizador da aplicação web autentica-se na mesma e seleciona a opção para consultar a sua lista de compras. A partir da sua lista de compras o utilizador seleciona a opção de efetuar a compra.
A aplicação web ao ser submetido o pedido realiza uma chamada ao service bus. Este realiza validações ao pedido, se o pedido não for válido retorna uma mensagem de erro, caso contrário realiza um pedido ao PurchaseService de forma a obter a lista de compras. Após obter a lista de compras o service bus vai processar cada pedido identificando se o produto é da organização ou de uma organização externa, caso se trate de um produto próprio realiza uma chamada ao PurchaseService, caso se trate de um vendedor externos realiza um pedido para o vendor em causa construindo pedidos adequados.
Após obtenção das respostas dos vários pedidos processa a informação recebida, gera a nota de encomenda e realiza nova chamada ao PurchaseService de forma a persistir a informação para a base de dados, sendo depois retornada uma mensagem de sucesso.

3.12. Conversão de Preço

Neste capítulo será apresentada a funcionalidade de conversão de preços representado na Figura 3.22 e Figura 3.23.

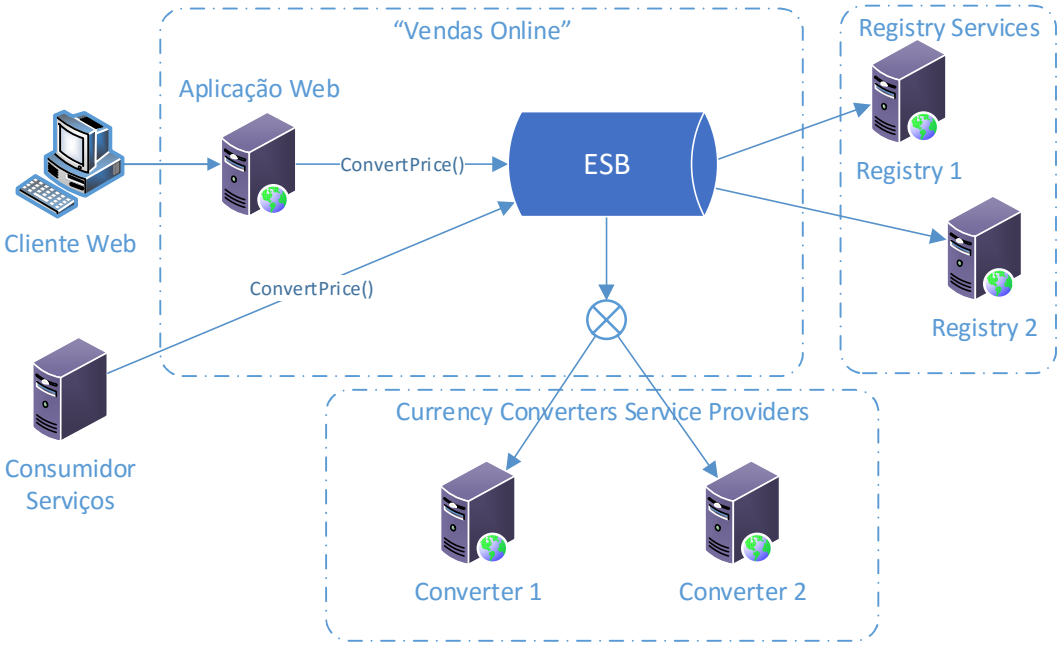

Figura 3.22: Orquestração de Serviços – Conversão de Preços

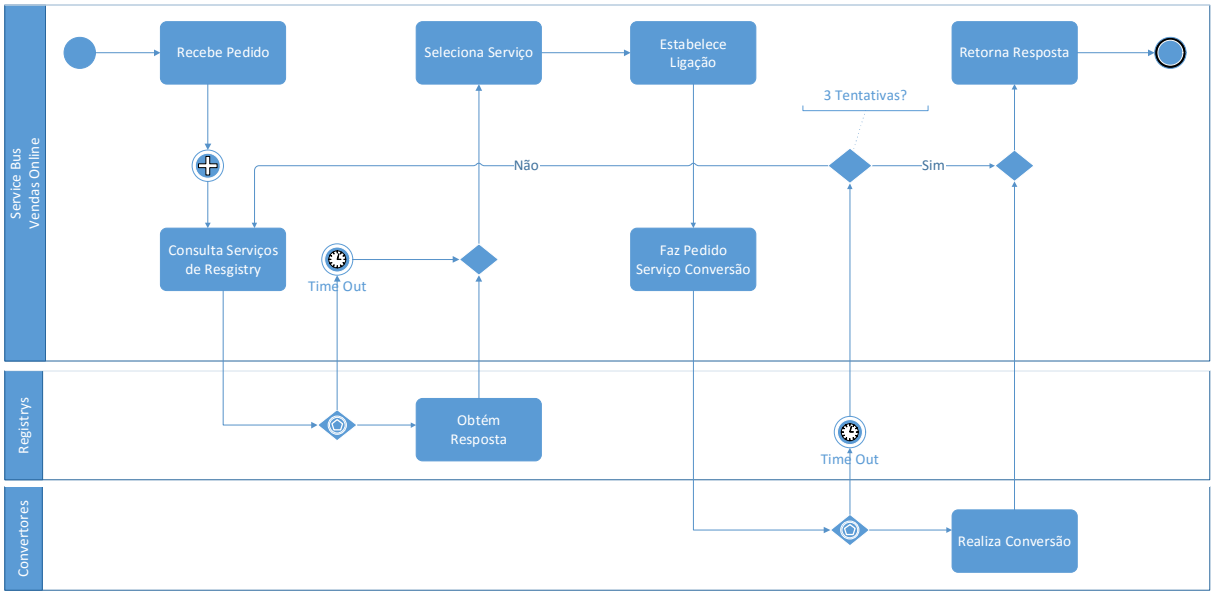

Figura 3.23: Diagrama BPMN – Conversão de Preços

Um utilizador da aplicação web seleciona a opção de realizar conversão dos preços, que deverá estar disponível para um produto isolado e para a listagem.

Ao ser submetido um pedido a aplicação web realiza uma chamada service bus. Este vai consultar serviços de *registry* de forma a obter uma lista de serviços que realizem conversão de moedas. Após descobrir alguns serviços processas as respostas e seleciona um ao qual se ligar. A ligação ao serviço e realizada e chamado o método para realizar a conversão. Após receber a resposta o service bus constrói a resposta e retorna-a à aplicação web que apresenta o resultado ao utilizador.

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Mestrado em Informática e Sistemas
Estágio/Projeto Industrial
Relatório Final

Anexo B – Fluxos do Mule ESB

Carla Maria Silva Machado

Orientadores:

João Carlos Costa Faria da Cunha

Cristiana Manuela Afonso Areias

Instituto Superior de Engenharia de Coimbra

Coimbra, dezembro, 2015

Índice

Índice de Figuras

1. Introdução

A utilização de um *service bus* no sistema foi acompanha de implementação de lógica de negócio ao nível do *service bus*. No caso do Mule ESB, que foi o *service bus* utilizado a lógica é implementada na forma de fluxo, sendo que se estes forem implementados com recurso ao AnyPoint Studio podem ser implementados com recurso a um editor gráfico ou de xml. Para uma maior liberdade de implementação o Mule ESB permite a utilização de classes Java.

Neste documento são apresentados os fluxos desenvolvidos, apresentando os diagramas de cada fluxo acompanhados do xml associado. São também apresentadas as classes auxiliares implementadas.

2. Fluxos Mule ESB

Neste capítulo serão apresentados os fluxos contruídos no Mule ESB para o sistema, serão apresentados os diagramas representativos dos fluxos e o ficheiro xml associado a cada fluxo.

2.1. Fluxo Search Product

Neste capítulo é apresentado o fluxo do método de pesquisa de produtos, SearchProduct.

2.1.1. Diagrama

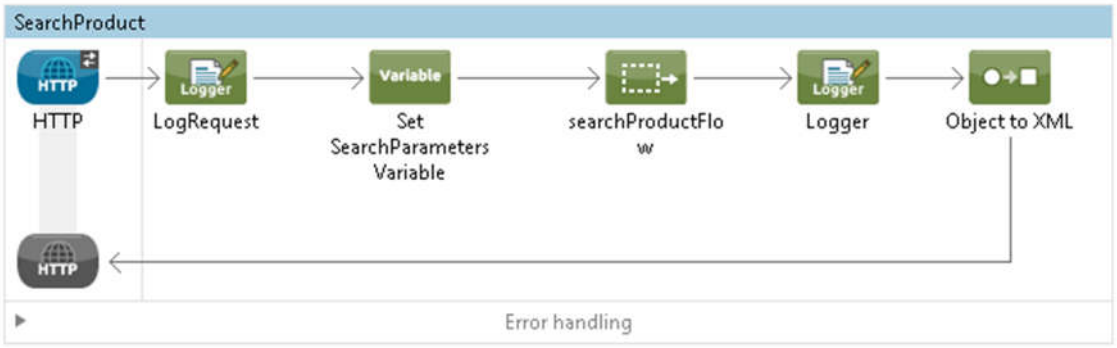

Figura 2.1: SearchProduct – Fluxo principal

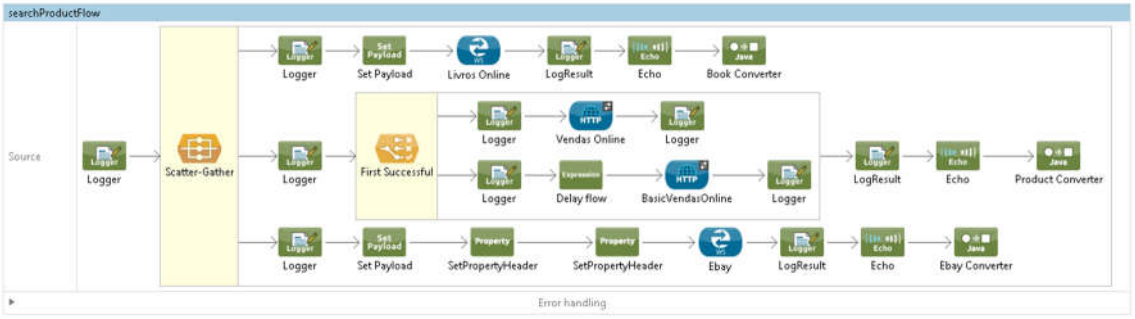

Figura 2.2: SearchProduct – Realização dos pedido – Fluxo secundário

2.1.2. Ficheiro xml

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:ws="http://www.mulesoft.org/schema/mule/ws"
      xmlns:http="http://www.mulesoft.org/schema/mule/http"
      xmlns:mulexml="http://www.mulesoft.org/schema/mule/xml"
      xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
      xmlns:spring="http://www.springframework.org/schema/beans"
version="CE-3.6.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.mulesoft.org/schema/mule/ws
http://www.mulesoft.org/schema/mule/ws/current/mule-ws.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/xml
http://www.mulesoft.org/schema/mule/xml/current/mule-xml.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
```

```
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd">
    <http:request-config    name="Ebay"    host="http://svcs.ebay.com"
port="80" doc:name="HTTP Request Configuration"/>

  <flow name="SearchProduct">
        <http:listener        config-ref="HTTP_Listener_Configuration"
path="/api/ProductsService/Search" doc:name="HTTP"/>
        <logger    message="Search    Request:    Query    params:    +
#[message.inboundProperties.'http.query.params'.searchParameters]     "
level="INFO" doc:name="LogRequest"/>
        <set-variable                    variableName="searchParameters"
value="#[message.inboundProperties.'http.query.params'.searchParameter
s]" doc:name="Set SearchParameters Variable"/>
        <flow-ref                                  name="searchProductFlow"
doc:name="searchProductFlow"/>

        <logger   message="Search   RESULT   -   Payload:     #[payload]"
level="INFO" doc:name="Logger"/>
        <mulexml:object-to-xml-transformer doc:name="Object to XML"/>
    </flow>

    <flow name="searchProductFlow">
        <logger   level="INFO"   doc:name="Logger"   message="Search   -
Parameters: #[flowVars.searchParameters]"/>
        <scatter-gather doc:name="Scatter-Gather" timeout="60000">
            <custom-aggregation-strategy
class="aggregators.SearchResponseAggregator"/>
            <processor-chain>
                <logger message="Flow WCF - LivrosOnline" level="INFO"
doc:name="Logger"/>
                <set-payload  value="&lt;?xml  version=&quot;1.0&quot;
encoding=&quot;UTF-8&quot;?&gt;&lt;ser:SearchBook
xmlns:ser=&quot;http://Services/&quot;&gt;&lt;arg0&gt;#[flowVars.searc
hParameters]&lt;/arg0&gt;&lt;/ser:SearchBook&gt;"         doc:name="Set
Payload"/>
                <ws:consumer                config-ref="ListingService"
operation="SearchBook" doc:name="Livros Online"/>
                <logger  message="Livros  Online  RESULT  -  Payload:
#[message.payload]" level="INFO" doc:name="LogResult"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.BookListToProductListTransformer"    doc:name="Book
Converter"/>
            </processor-chain>
            <processor-chain>
                <logger message="Flow REST  - VendasOnline" level="INFO"
doc:name="Logger"/>
                <first-successful doc:name="First Successful">
                    <processor-chain>
                        <logger       message="Flow       VendasOnline"
level="INFO" doc:name="Logger"/>
                        <http:request      config-ref="ProductsService"
path="/Search" method="GET" doc:name="Vendas Online">
                            <http:request-builder>
                                <http:query-param
paramName="searchParameters" value="#[flowVars.searchParameters]"/>
                            </http:request-builder>
                        </http:request>
```

```
                        <logger   message="Flow   VendasOnline   RESULT
#[payload]" level="INFO" doc:name="Logger"/>
                    </processor-chain>
                    <processor-chain>
                        <logger    message="Flow    BasicVendasOnline"
level="INFO" doc:name="Logger"/>
                        <expression-component           doc:name="Delay
flow"><![CDATA[Thread.sleep(40000);]]></expression-component>
                        <http:request  config-ref="BasicProductService"
path="/SimpleSearch" method="GET" doc:name="BasicVendasOnline">
                            <http:request-builder>
                                <http:query-param
paramName="searchParameters" value="#[flowVars.searchParameters]"/>
                                <http:query-param
paramName="numberOfProducts" value="15"/>
                            </http:request-builder>
                        </http:request>
                        <logger message="Flow BasicVendasOnline RESULT
#[payload]" level="INFO" doc:name="Logger"/>
                    </processor-chain>
                </first-successful>
                <logger   message="Vendas   Online   RESULT   -   Payload:
#[message.payload]" level="INFO" doc:name="LogResult"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.ProductListToProductListTransformer"
doc:name="Product Converter"/>
            </processor-chain>
            <processor-chain>
                <logger   message="Flow   WCF   -   Ebay"   level="INFO"
doc:name="Logger"/>
                <set-payload  value="&lt;ser:findItemsByKeywordsRequest
xmlns:ser=&quot;http://www.ebay.com/marketplace/search/v1/services&quo
t;&gt;&lt;ser:paginationInput&gt;&lt;ser:pageNumber&gt;1&lt;/ser:pageN
umber&gt;&lt;ser:entriesPerPage&gt;30&lt;/ser:entriesPerPage&gt;&lt;/s
er:paginationInput&gt;&lt;ser:keywords&gt;#[flowVars.searchParameters]
&lt;/ser:keywords&gt;&lt;/ser:findItemsByKeywordsRequest&gt;"
doc:name="Set Payload"/>
                <set-property  propertyName="X-EBAY-SOA-OPERATION-NAME"
value="findItemsByKeywords" doc:name="SetPropertyHeader"/>
                <set-property        propertyName="X-EBAY-SOA-SECURITY-
APPNAME" value="*****" doc:name="SetPropertyHeader"/>
                <ws:consumer            config-ref="EbayFindingService"
operation="findItemsByKeywords" doc:name="Ebay"/>
                <logger     message="EBAY     RESULT     -     Payload:
#[message.payload]" level="INFO" doc:name="LogResult"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.EbayProductListToProductList"        doc:name="Ebay
Converter"/>
            </processor-chain>
        </scatter-gather>
    </flow>

</mule>
```

2.2. Fluxo Get Product

Neste capítulo é apresentado o fluxo do método de obtenção de detalhes de um produto, GetProduct, no qual é recebido um pedido com a informação do vendedor e produto que se pretende consultar na query string e de acordo com esta informação vai ser determinado a qual dos vendedores disponíveis deve ser realizado o pedido os detalhes.

2.2.1. Diagrama

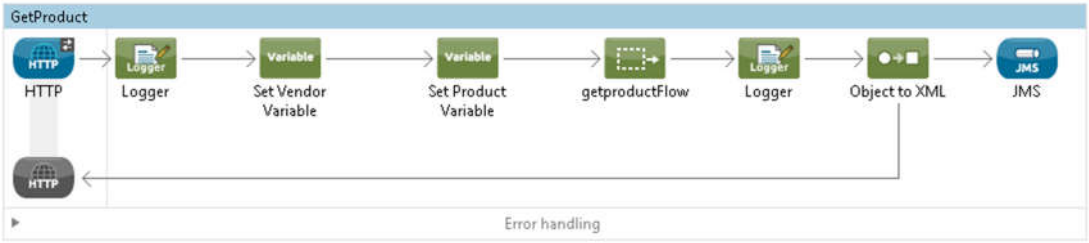

Figura 2.3: GetProduct – Fluxo principal

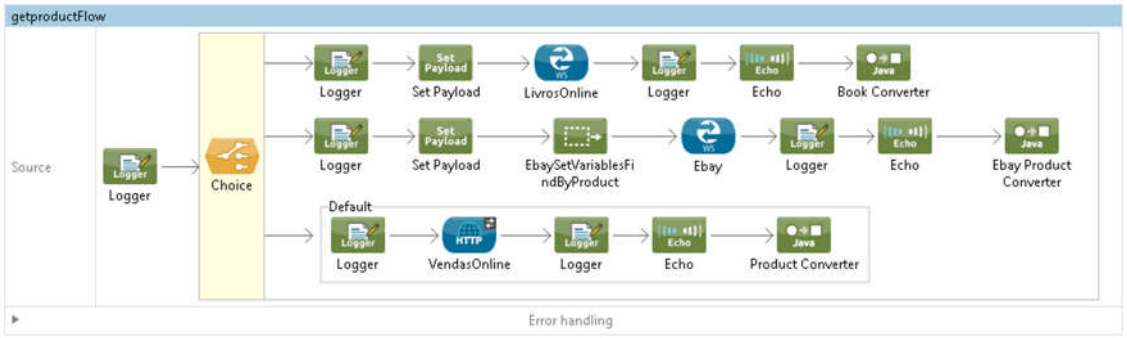

Figura 2.4: GetProduct – Realização do pedido – Fluxo secundário

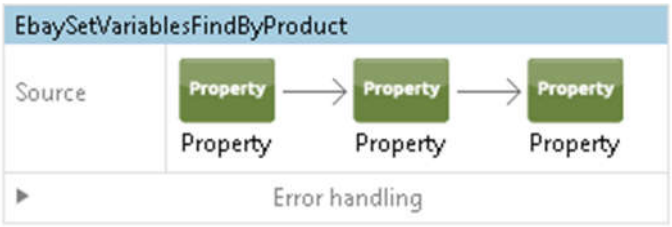

Figura 2.5: GetProduct – Definição variáveis - Fluxo secundário

2.2.2. Ficheiro xml

```
<?xml version="1.0" encoding="UTF-8"?>

<mule                 xmlns:jms="http://www.mulesoft.org/schema/mule/jms"
xmlns:json="http://www.mulesoft.org/schema/mule/json"
xmlns:ws="http://www.mulesoft.org/schema/mule/ws"
      xmlns:http="http://www.mulesoft.org/schema/mule/http"
      xmlns:mulexml="http://www.mulesoft.org/schema/mule/xml"
      xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
      xmlns:spring="http://www.springframework.org/schema/beans"
version="CE-3.6.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
      xsi:schemaLocation="http://www.mulesoft.org/schema/mule/ws
http://www.mulesoft.org/schema/mule/ws/current/mule-ws.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/xml
http://www.mulesoft.org/schema/mule/xml/current/mule-xml.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/json
http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/jms
http://www.mulesoft.org/schema/mule/jms/current/mule-jms.xsd">
    <jms:object-to-jmsmessage-transformer   name="Object_to_JMSMessage"
doc:name="Object to JMSMessage"/>

 <flow name="GetProduct">
        <http:listener         config-ref="HTTP_Listener_Configuration"
path="/api/ProductsService/Get" doc:name="HTTP"/>
        <logger     message="GetProduct     Request     -          vendor:
#[message.inboundProperties.'http.query.params'.vendor]
Id=#[message.inboundProperties.'http.query.params'.id]   "   level="INFO"
doc:name="Logger"/>
         <set-variable                              variableName="vendor"
value="#[message.inboundProperties.'http.query.params'.vendor]"
doc:name="Set Vendor Variable"/>
        <set-variable                              variableName="product"
value="#[message.inboundProperties.'http.query.params'.id]"
doc:name="Set Product Variable"/>
        <flow-ref name="getproductFlow" doc:name="getproductFlow"/>

        <logger  message="Get  Product  Result  -  Payload:    #[payload]"
level="INFO" doc:name="Logger"/>
        <mulexml:object-to-xml-transformer doc:name="Object to XML"/>
        <jms:outbound-endpoint   connector-ref="Active_MQ"  transformer-
refs="Object_to_JMSMessage" doc:name="JMS" topic="RecentlyViewd">
            <jms:transaction action="NONE"/>
        </jms:outbound-endpoint>
    </flow>
    <flow name="getproductFlow">
        <logger  message="Get  SubFlow    Vendor  #[flowVars.vendor]  and
product #[flowVars.product]" level="INFO" doc:name="Logger"/>
            <choice doc:name="Choice">
            <when expression="#[flowVars.vendor == '2']">
                <logger       message="LivrosOnline"       level="INFO"
doc:name="Logger"/>
                <set-payload                      value="&lt;ser:ViewBook
xmlns:ser=&quot;http://Services/&quot;&gt;&lt;arg0&gt;#[flowVars.produ
ct]&lt;/arg0&gt;&lt;/ser:ViewBook&gt;" doc:name="Set Payload"/>
                <ws:consumer                  config-ref="ListingService"
operation="ViewBook" doc:name="LivrosOnline"/>
                <logger  message="Livros  Online  RESULT"  level="INFO"
doc:name="Logger"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.BookListToProductListTransformer"    doc:name="Book
Converter"/>
            </when>
            <when expression="#[flowVars.vendor == '3']">
```

```
                <logger message="Ebay" level="INFO" doc:name="Logger"/>
                <set-payload        value="&lt;urn:GetSingleItemRequest
xmlns:urn=&quot;urn:ebay:apis:eBLBaseComponents&quot;&gt;&lt;urn:ItemI
D&gt;#[flowVars.product]&lt;/urn:ItemID&gt;&lt;/urn:GetSingleItemReque
st&gt;" doc:name="Set Payload"/>
                <flow-ref          name="EbaySetVariablesFindByProduct"
doc:name="EbaySetVariablesFindByProduct"/>
                <ws:consumer           config-ref="EbayShoppingService"
operation="GetSingleItem" doc:name="Ebay"/>
                <logger  level="INFO"  doc:name="Logger"  message="Ebay
RESULT"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.EbayProductToProductTransformer"     doc:name="Ebay
Product Converter"/>
            </when>
            <otherwise>
                <logger     message="DEFAULT     REST     VendasOnline"
level="INFO" doc:name="Logger"/>
                <http:request              config-ref="ProductsService"
path="/#[flowVars.product]" method="GET" doc:name="VendasOnline">
                    <http:request-builder>
                    </http:request-builder>
                </http:request>
                <logger     message="REST     RESULT"     level="INFO"
doc:name="Logger"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.ProductToProductTransformer"      doc:name="Product
Converter"/>
            </otherwise>
        </choice>
    </flow>

    <flow name="EbaySetVariablesFindByProduct">
        <set-property              propertyName="X-EBAY-API-CALL-NAME"
value="GetSingleItem" doc:name="Property"/>
        <set-property  propertyName="X-EBAY-API-APP-ID"  value="*****"
doc:name="Property"/>
        <set-property   propertyName="X-EBAY-API-VERSION"   value="935"
doc:name="Property"/>
    </flow>
</mule>
```

2.3. Fluxo Compare Products

Neste capítulo é apresentado o fluxo do método de comparação de produtos, CompareProducts, no qual um pedido é recebido com a informação do vendedor e produto de até um máximo de quatro produtos. São gerados quatro caminhos em paralelo que irão verificar se necessitam fazer um pedido e em caso positivo irão determinar a que vendedor fazer o pedido. Após o termino dos quatro caminhos paralelos as várias respostas são agregadas em apenas uma resposta.

2.3.1. Diagrama

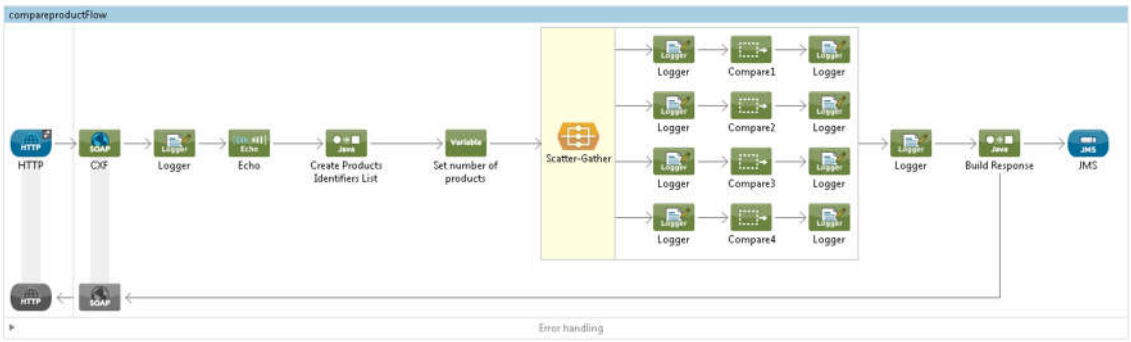

Figura 2.6: CompareProducts - Fluxo principal

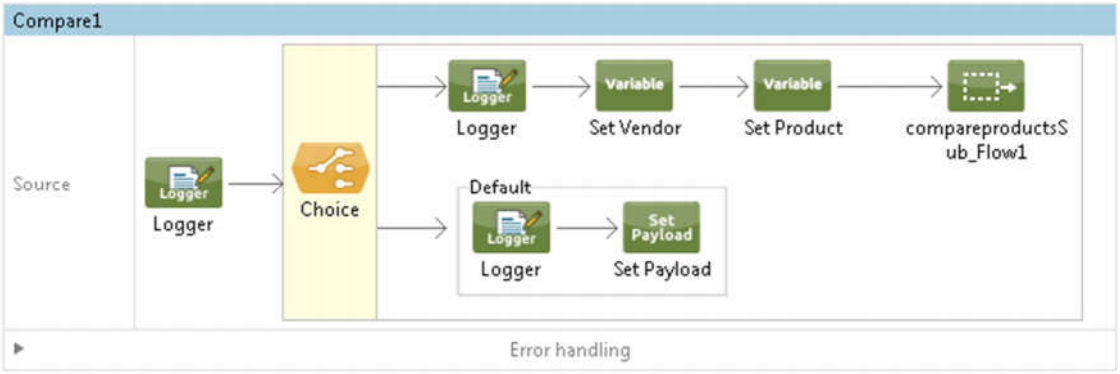

Figura 2.7: CompareProducts – Definição de variáveis - Fluxo secundário

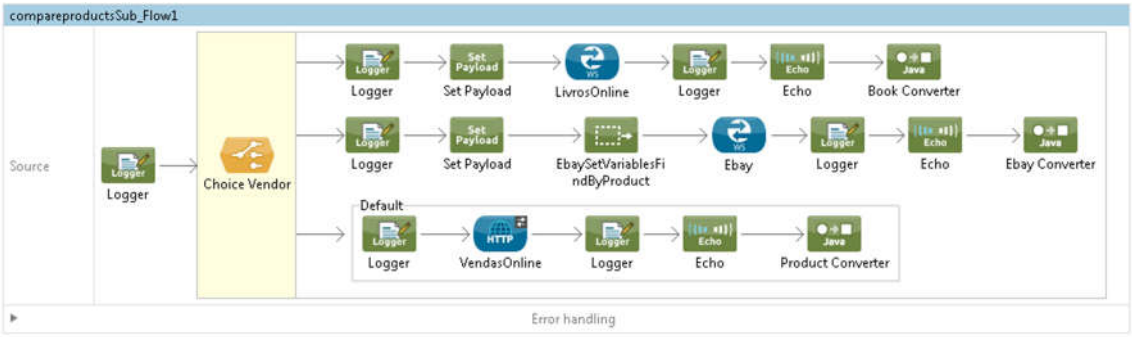

Figura 2.8: CompareProducts –Realização do pedido – Fluxo secundário

2.3.2. Ficheiro xml

```
<?xml version="1.0" encoding="UTF-8"?>

<mule                xmlns:jms="http://www.mulesoft.org/schema/mule/jms"
xmlns:mulexml="http://www.mulesoft.org/schema/mule/xml"
      xmlns:ws="http://www.mulesoft.org/schema/mule/ws"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:cxf="http://www.mulesoft.org/schema/mule/cxf"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
      xmlns:spring="http://www.springframework.org/schema/beans"
version="CE-3.6.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.mulesoft.org/schema/mule/xml
http://www.mulesoft.org/schema/mule/xml/current/mule-xml.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
```

```
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ws
http://www.mulesoft.org/schema/mule/ws/current/mule-ws.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/cxf
http://www.mulesoft.org/schema/mule/cxf/current/mule-cxf.xsd
http://www.mulesoft.org/schema/mule/jms
http://www.mulesoft.org/schema/mule/jms/current/mule-jms.xsd">
    <ws:consumer-config                              name="ProductService"
wsdlLocation="http://localhost/ProductService/Services/ProductsService
.svc?wsdl"                                      service="ProductsService"
port="BasicHttpBinding_IProductsService"
serviceAddress="http://localhost/ProductService/Services/ProductsServi
ce.svc" doc:name="Web Service Consumer"/>
    <flow name="compareproductFlow">
        <http:listener         config-ref="HTTP_Listener_Configuration"
path="/CompareProducts" doc:name="HTTP"/>
        <cxf:proxy-service              namespace="http://tempuri.org/"
service="CompareProductsService"                          payload="body"
wsdlLocation="http://localhost/ProductService/Services/CompareProducts
Service.svc?Wsdl" doc:name="CXF"/>
         <logger  message="Compare  Request  -  Payload:  #[payload]"
level="INFO" doc:name="Logger"/>
        <echo-component doc:name="Echo"/>
         <custom-transformer
class="Transformers.ProductIdentifierTransformer"      doc:name="Create
Products Identifiers List"/>
        <set-variable     variableName="productsToCompare"     value="
#[payload.size()]" doc:name="Set number of products"/>
        <scatter-gather doc:name="Scatter-Gather" timeout="90000">
            <custom-aggregation-strategy
class="aggregators.CompareResponseAggregator"/>
            <processor-chain>
                <logger       message="Flow       1"       level="INFO"
doc:name="Logger"/>
                <flow-ref name="Compare1" doc:name="Compare1"/>
                <logger   message="Flow   1   Response"   level="INFO"
doc:name="Logger"/>
            </processor-chain>
            <processor-chain>
                <logger       message="Flow       2"       level="INFO"
doc:name="Logger"/>
                <flow-ref name="Compare2" doc:name="Compare2"/>
                <logger   message="Flow   2   Response"   level="INFO"
doc:name="Logger"/>
            </processor-chain>
            <processor-chain>
                <logger       message="Flow       3"       level="INFO"
doc:name="Logger"/>
                <flow-ref name="Compare3" doc:name="Compare3"/>
                <logger   message="Flow   3   Response"   level="INFO"
doc:name="Logger"/>
            </processor-chain>
            <processor-chain>
                <logger       message="Flow       4"       level="INFO"
doc:name="Logger"/>
                <flow-ref name="Compare4" doc:name="Compare4"/>
```

```
                <logger message="Flow 4 Response" level="INFO"
doc:name="Logger"/>
            </processor-chain>
        </scatter-gather>
        <logger level="INFO" doc:name="Logger" message="Compare Result:
#[payload]"/>
        <custom-transformer
class="Transformers.CompareResponseTransformer" doc:name="Build
Response"/>
        <jms:outbound-endpoint doc:name="JMS" transformer-
refs="Object_to_JMSMessage" connector-ref="Active_MQ"
topic="RecentlyViewd">
            <jms:transaction action="NONE"/>
        </jms:outbound-endpoint>
    </flow>

    <flow name="Compare1">
        <logger level="INFO" doc:name="Logger" message="FlowCompare 1
size: #[flowVars.productsToCompare] payload #[payload.size()] ///
#[message.payload.size() &gt; 0] / #[message.payload.size()]"/>
        <choice doc:name="Choice">
            <when expression="#[message.payload.size() &gt; 0]" >
                <logger message="Compare 1 Flow vendor" level="INFO"
doc:name="Logger"/>
                 <set-variable variableName="vendor"
value="#[payload[0].getVendorId()]" doc:name="Set Vendor"/>
                <set-variable variableName="product"
value="#[payload[0].getProductId()]" doc:name="Set Product"/>
                <flow-ref name="compareproductsSub_Flow1"
doc:name="compareproductsSub_Flow1"/>
            </when>
            <otherwise>
                <logger message="Compare 1 Default - No product request"
level="INFO" doc:name="Logger"/>
                <set-payload value="#[null]" doc:name="Set Payload"/>
            </otherwise>
        </choice>
    </flow>

    <flow name="Compare2">
        <logger level="INFO" doc:name="Logger" message="FlowCompare2
size: #[flowVars.productsToCompare] payload #[payload.size()]"/>
        <choice doc:name="Choice">
            <when expression="#[message.payload.size() &gt; 1]" >
                <logger message="Compare 2 Flow vendor" level="INFO"
doc:name="Logger"/>
                <set-variable variableName="vendor"
value="#[payload[1].getVendorId()]" doc:name="Set Vendor"/>
                <set-variable variableName="product"
value="#[payload[1].getProductId()]" doc:name="Set Product"/>
                <flow-ref name="compareproductsSub_Flow1"
doc:name="compareproductsSub_Flow1"/>
            </when>
            <otherwise>
                <logger message="Compare 2 Default - No product request"
level="INFO" doc:name="Logger"/>
                <set-payload value="#[null]" doc:name="Set Payload"/>
            </otherwise>
        </choice>
    </flow>
```

```
    <flow name="Compare3">
        <logger level="INFO" doc:name="Logger" message="FlowCompare 3
size:  #[flowVars.productsToCompare] payload  #[payload.size()]"/>
        <choice doc:name="Choice">
            <when expression="#[message.payload.size() &gt; 2]" >
                <logger message="Compare 3 Flow vendor" level="INFO"
doc:name="Logger"/>
                <set-variable                    variableName="vendor"
value="#[payload[2].getVendorId()]" doc:name="Set Vendor"/>
                <set-variable                   variableName="product"
value="#[payload[2].getProductId()]" doc:name="Set Product"/>
                <flow-ref               name="compareproductsSub_Flow1"
doc:name="compareproductsSub_Flow1"/>
            </when>
            <otherwise>
                <logger message="Compare 3 Default - No product request"
level="INFO" doc:name="Logger"/>
                <set-payload value="#[null]" doc:name="Set Payload"/>
            </otherwise>
        </choice>
    </flow>

    <flow name="Compare4">
        <logger level="INFO" doc:name="Logger" message="FlowCompare 4
size:  #[flowVars.productsToCompare] payload  #[payload.size()]"/>
        <choice doc:name="Choice">
            <when expression="#[message.payload.size() &gt; 3]" >
                <logger message="Compare 4 Flow vendor" level="INFO"
doc:name="Logger"/>
                <set-variable                    variableName="vendor"
value="#[payload[3].getVendorId()]" doc:name="Set Vendor"/>
                <set-variable                   variableName="product"
value="#[payload[3].getProductId()]" doc:name="Set Product"/>
                <flow-ref               name="compareproductsSub_Flow1"
doc:name="compareproductsSub_Flow1"/>
            </when>
            <otherwise>
                <logger message="Compare 4 Default - No product request"
level="INFO" doc:name="Logger"/>
                <set-payload value="#[null]" doc:name="Set Payload"/>
            </otherwise>
        </choice>
    </flow>

    <flow name="compareproductsSub_Flow1">
        <logger message="Choice Flow" level="INFO" doc:name="Logger"/>
        <choice doc:name="Choice Vendor">
            <when expression="#[flowVars.vendor == '2']">
                <logger message="Choice Vendor:   #[flowVars.vendor]"
level="INFO" doc:name="Logger"/>
                <set-payload                  value="&lt;ser:ViewBook
xmlns:ser=&quot;http://Services/&quot;&gt;&lt;arg0&gt;#[flowVars.produ
ct]&lt;/arg0&gt;&lt;/ser:ViewBook&gt;" doc:name="Set Payload"/>
                <ws:consumer               config-ref="ListingService"
operation="ViewBook" doc:name="LivrosOnline"/>
                <logger  message="LivrosOnline  RESULT"  level="INFO"
doc:name="Logger"/>
                <echo-component doc:name="Echo"/>
```

```
                <custom-transformer
class="Transformers.BookToProductTransformer"            doc:name="Book
Converter"/>
            </when>
            <when expression="#[flowVars.vendor == '3']">
                <logger message="Ebay" level="INFO" doc:name="Logger"/>
                <set-payload        value="&lt;urn:GetSingleItemRequest
xmlns:urn=&quot;urn:ebay:apis:eBLBaseComponents&quot;&gt;&lt;urn:ItemI
D&gt;#[flowVars.product]&lt;/urn:ItemID&gt;&lt;/urn:GetSingleItemReque
st&gt;" doc:name="Set Payload"/>
                <flow-ref          name="EbaySetVariablesFindByProduct"
doc:name="EbaySetVariablesFindByProduct"/>
                <ws:consumer           config-ref="EbayShoppingService"
operation="GetSingleItem" doc:name="Ebay"/>
                <logger  level="INFO"  doc:name="Logger"  message="EBAY
RESULT"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.EbayProductToProductTransformer"     doc:name="Ebay
Converter"/>
            </when>
            <otherwise>
                <logger message="DEFAULT REST id: #[flowVars.product]"
level="INFO" doc:name="Logger"/>
                <http:request              config-ref="ProductsService"
path="/#[flowVars.product]" method="GET" doc:name="VendasOnline">
                    <http:request-builder/>
                </http:request>
                <logger   message="VendasOnline   RESULT"   level="INFO"
doc:name="Logger"/>
                <echo-component doc:name="Echo"/>
                <custom-transformer
class="Transformers.ProductToProductTransformer"      doc:name="Product
Converter"/>
            </otherwise>
        </choice>
    </flow>

</mule>
```

2.4. Fluxo SOAP Interface

Neste capítulo é apresentado o fluxo para a interface SOAP para todas as operações disponibilizadas nos fluxos anteriores. Neste fluxo apenas o caminho para a obtenção de detalhes foi implementado na sua totalidade.

Ao ser recebido um pedido é identificada a operação e um caminho seguido de acordo com a operação, nesse caminho é preparada a mensagem de acordo com as necessidades e realizado o pedido correspondente às organizações envolvidas, sendo que para a realização dos pedidos e tratamento das respostas recebidas são reutilizados os fluxos secundários descritos nas secções anteriores para cada operação.

2.4.1. Diagrama

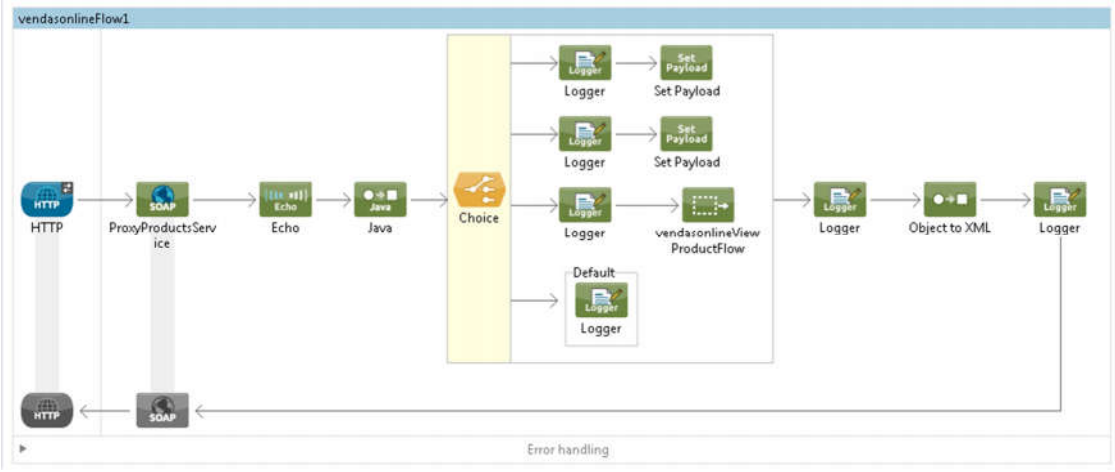

Figura 2.9: SOAP Interface – Fluxo principal

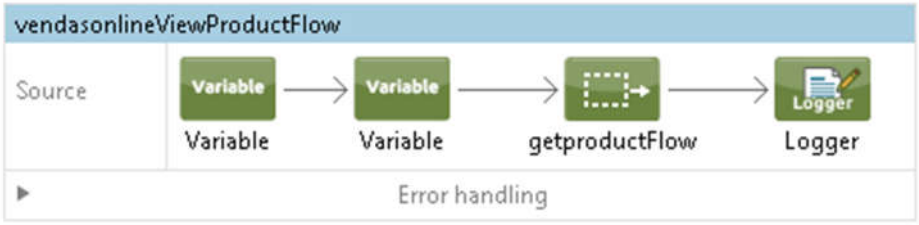

Figura 2.10: SOAP Interface – Fluxo secundário Obtenção Detalhes de Produto

2.4.2. Ficheiro xml

```
<?xml version="1.0" encoding="UTF-8"?>

<mule xmlns:jms="http://www.mulesoft.org/schema/mule/jms"
xmlns:ws="http://www.mulesoft.org/schema/mule/ws"
xmlns:json="http://www.mulesoft.org/schema/mule/json"
xmlns:file="http://www.mulesoft.org/schema/mule/file"
xmlns:mulexml="http://www.mulesoft.org/schema/mule/xml"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:cxf="http://www.mulesoft.org/schema/mule/cxf"
xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
      xmlns:spring="http://www.springframework.org/schema/beans"
version="CE-3.6.1"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-current.xsd
http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/cxf
http://www.mulesoft.org/schema/mule/cxf/current/mule-cxf.xsd
http://www.mulesoft.org/schema/mule/ws
http://www.mulesoft.org/schema/mule/ws/current/mule-ws.xsd
http://www.mulesoft.org/schema/mule/file
http://www.mulesoft.org/schema/mule/file/current/mule-file.xsd
http://www.mulesoft.org/schema/mule/xml
http://www.mulesoft.org/schema/mule/xml/current/mule-xml.xsd
```

```
http://www.mulesoft.org/schema/mule/json
http://www.mulesoft.org/schema/mule/json/current/mule-json.xsd
http://www.mulesoft.org/schema/mule/jms
http://www.mulesoft.org/schema/mule/jms/current/mule-jms.xsd">
    <spring:beans>
        <spring:bean id="objectStore"
class="org.mule.util.store.SimpleMemoryObjectStore"/>
    </spring:beans>
    <http:listener-config name="HTTP_Listener_Configuration"
host="0.0.0.0" port="8081" basePath="/VendasOnline" doc:name="HTTP
Listener Configuration"/>
    <http:request-config name="ProductsService" host="localhost"
port="80" doc:name="HTTP Request Configuration"
basePath="/VendasOnline/api/products"/>
    <ws:consumer-config name="ListingService"
wsdlLocation="http://localhost:9090/LivrosOnline/services/ListingServi
ce?wsdl" service="ListingServiceService" port="ListingServicePort"
serviceAddress="http://localhost:9090/LivrosOnline/services/ListingSer
vice" doc:name="Web Service Consumer"/>
    <ws:consumer-config name="EbayFindingService"
wsdlLocation="https://developer.ebay.com/webservices/finding/latest/Fi
ndingService.wsdl" service="FindingService"
port="FindingServiceSOAPPort"
serviceAddress="https://svcs.ebay.com/services/search/FindingService/v
1" doc:name="Web Service Consumer"/>
    <jms:activemq-connector name="Active_MQ" username="system"
password="manager" brokerURL="tcp://localhost:61616"
validateConnections="true" doc:name="Active MQ" durable="true"/>
    <http:request-config name="BasicProductService" host="localhost"
port="80" basePath="/BasicVendasOnline/api/products" doc:name="HTTP
Request Configuration"/>
    <ws:consumer-config name="EbayShoppingService"
wsdlLocation="http://developer.ebay.com/webservices/935/ShoppingServic
e.wsdl" service="Shopping" port="Shopping"
serviceAddress="http://open.api.ebay.com/shopping" doc:name="Web
Service Consumer"/>
    <flow name="vendasonlineFlow1">
        <http:listener config-ref="HTTP_Listener_Configuration"
path="/Test" doc:name="HTTP">
            <http:response-builder statusCode="200"/>
        </http:listener>
        <cxf:proxy-service namespace="http://tempuri.org/"
service="ProductsService" payload="body"
wsdlLocation="http://localhost/ProductService/Services/ProductsService
.svc?Wsdl" doc:name="ProxyProductsService"/>
        <echo-component doc:name="Echo"/>
        <custom-transformer
class="Transformers.SoapRequestTransformer" doc:name="Java"/>
        <choice doc:name="Choice">
            <when expression="#[payload.getOperation() ==
'SearchProduct']">
                <logger message="Search Product" level="INFO"
doc:name="Logger"/>
                <flow-ref name="vendasonlineSearchProductsFlow"
doc:name="vendasonlineSearchProductsFlow"/>
            </when>
            <when expression="#[payload.getOperation() ==
'CompareProducts']">
                <logger message="Compare Products" level="INFO"
doc:name="Logger"/>
```

```
            </when>
            <when expression="#[payload.getOperation() ==
'ViewProduct']">
                <logger message="View Product" level="INFO"
doc:name="Logger"/>
                <flow-ref name="vendasonlineViewProductFlow"
doc:name="vendasonlineViewProductFlow"/>

            </when>
            <otherwise>
                <logger message="Invalid Operattion" level="ERROR"
doc:name="Logger"/>
            </otherwise>
        </choice>
        <mulexml:object-to-xml-transformer doc:name="Object to XML"/>
        <logger message="RESULT: #[payload]" level="INFO"
doc:name="Logger"/>
    </flow>
    <flow name="vendasonlineViewProductFlow">
       <set-variable variableName="vendor"
value="#[payload.getVendor()]" doc:name="Variable"/>
                <set-variable variableName="product"
value="#[payload.getProduct()]" doc:name="Variable"/>
                <flow-ref name="getproductFlow"
doc:name="getproductFlow"/>
                <logger message="RESULT View Product #[payload]"
level="INFO" doc:name="Logger"/>
    </flow>
    <flow name="vendasonlineSearchProductsFlow">
        <set-variable variableName="searchParameters"
value="#[message.inboundProperties.'http.query.params'.searchParameter
s]" doc:name="SetSearchParametersVariable"/>
        <flow-ref name="searchProductFlow"
doc:name="searchProductFlow"/>
        <logger level="INFO" doc:name="Logger"/>
    </flow>
</mule>
```

3. Classes Auxiliares

Neste capítulo serão apresentadas as várias classes de JAVA implementadas para utilização nos fluxos do Mule ESB.

Fazendo uma breve introdução às classes implementadas foram implementadas classes de entidades auxiliares aos fluxos

- product – entidade que representa os produtos vendidos pela organização, contento a informação das suas propriedades e características
- ProductIdentifier – entidade que representa a identificação de um produto, contendo o identificador do vendedor e do próprio produto
- SoapRequest – entidade empregue na construção na descodificação de um pedido SOAP e definir varáveis a usar posteriormente nos fluxos

Foram igualmente implementadas classes empregues na conversão da informação recebida para um formato comum, ou seja, conversão a informação recebida para a entidade product

- BookListToProductListTransformer – Transforma uma lista de produtos do vendedor Livros Online numa lista de objetos da entidade product
- ProductListToProductListTransformer – Transforma uma lista de produtos do vendedor Vendas Online numa lista de objetos da entidade product
- EbayProductListToProductList – Transforma uma lista de produtos do vendedor Ebay numa lista de objetos da entidade product
- EbayProductToProductTransformer
- ProductToProductTransformer
- ProductIdentifierTransformer
- BookToProductTransformer

Por fim foram também implementadas classes com a função de personalizar a agregação das respostas nos casos em que o processamento é realizado em paralelo e se pretende o conjunto das respostas de todos os caminhos

- SearchResponseAggregator – agrega a respostas no método de pesquisa de produtos, ou seja, agrega as respostas dos vários pedidos de pesquisa numa única resposta, ignorando os caminhos que eventualmente não tenham retornado resposta ou tenha ocorrido alguma falha
- CompareResponseAggregator – agrega a respostas no método de comparação de produtos, ou seja, agrega as respostas dos vários pedidos de detalhes de produtos numa única resposta, ignorando os caminhos que eventualmente não tenham retornado resposta ou tenha ocorrido alguma falha

3.1. Entidades

3.1.1. product

```
package entities;

import org.codehaus.jackson.map.annotate.JsonDeserialize;

import converters.*;

@JsonDeserialize(using = ProductConverter.class)
public class product {
    public long ProductId;
    public int VendorId;
    public String Name;
    public String Description;
    public float Price;
    public String Quantity;
    public String Category;
    public String Other_Info;
    public String VendorName;

    public product() {
            super();
      }

      public product(long productId, int vendorId, String name,
String description, float price, String quantity, String category,
String other_Info, String vendorName) {
            super();
            ProductId = productId;
            VendorId = vendorId;
            Name = name;
            Description = description;
            Price = price;
            Quantity = quantity;
            Category = category;
            Other_Info = other_Info;
            VendorName = vendorName;
      }

      public product(long productId, String name, String description,
float price, String quantity, String category, String other_Info) {
            super();
            ProductId = productId;
            Name = name;
            Description = description;
            Price = price;
            Quantity = quantity;
            Category = category;
            Other_Info = other_Info;
      }

      public long getProductId() {
            return ProductId;
      }
      public void setProductId(long productId) {
            ProductId = productId;
      }
```

```
	public String getName() {
		return Name;
	}
	public void setName(String name) {
		Name = name;
	}
	public String getDescription() {
		return Description;
	}
	public void setDescription(String description) {
		Description = description;
	}
	public float getPrice() {
		return Price;
	}
	public void setPrice(float price) {
		Price = price;
	}
	public String getQuantity() {
		return Quantity;
	}
	public void setQuantity(String quantity) {
		Quantity = quantity;
	}
	public String getCategory() {
		return Category;
	}
	public void setCategory(String category) {
		Category = category;
	}
	public String getOther_Info() {
		return Other_Info;
	}
	public void setOther_Info(String other_Info) {
		Other_Info = other_Info;
	}

	public int getVendorId() {
		return VendorId;
	}

	public void setVendorId(int vendorId) {
		VendorId = vendorId;
	}

	public String getVendorName() {
		return VendorName;
	}

	public void setVendorName(String vendorName) {
		VendorName = vendorName;
	}
}
```

3.1.2. ProductIdentifier

```
package entities;

public class ProductIdentifier {

	public long ProductId;
	public int VendorId;

	public ProductIdentifier(long productId, int vendorId) {
		super();
		ProductId = productId;
		VendorId = vendorId;
	}

	public long getProductId() {
		return ProductId;
	}

	public void setProductId(long productId) {
		ProductId = productId;
	}

	public int getVendorId() {
		return VendorId;
	}

	public void setVendorId(int vendorId) {
		VendorId = vendorId;
	}
}
```

3.1.3. SoapRequest

```
package entities;

import java.util.List;

public class SoapRequest {

	public String operation;
	public List<ProductIdentifier> products;
	public String parameters;
	public String vendor;
	public String product;


	public SoapRequest() {
		super();
	}

	public SoapRequest(String operation, List<ProductIdentifier>
products, String parameters) {
		super();
		this.operation = operation;
		this.products = products;
		this.parameters = parameters;
	}
```

```
	public String getVendor() {
		return vendor;
	}

	public void setVendor(String vendor) {
		this.vendor = vendor;
	}

	public String getProduct() {
		return product;
	}

	public void setProduct(String product) {
		this.product = product;
	}

	public String getOperation() {
		return operation;
	}

	public void setOperation(String operation) {
		this.operation = operation;
	}

	public List<ProductIdentifier> getProducts() {
		return products;
	}

	public void setProducts(List<ProductIdentifier> products) {
		this.products = products;
	}

	public String getParameters() {
		return parameters;
	}

	public void setParameters(String parameters) {
		this.parameters = parameters;
	}
}
```

3.2. Constantes

3.2.1. Constants

```
package Common;

public class Constants {

	public static int VendasOnlineId = 1;
	public static String VendasOnlineName = "Vendas Online";

	public static int LivrosOnlineId = 2;
	public static String LivrosOnlineName = "Livros Online";

	public static int EbayId = 3;
	public static String EbayName = "Ebay";
}
```

3.2.2. SoapOperations

```
package Common;

public class SoapOperations {

	public static String SearchProduct = "SearchProduct";
	public static String ViewProduct = "ViewProduct";
	public static String CompareProduct = "CompareProducts";
}
```

3.3. Conversores

3.3.1. ProductConverter

```
package converters;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.List;

import org.codehaus.jackson.JsonNode;
import org.codehaus.jackson.JsonParser;
import org.codehaus.jackson.JsonProcessingException;
import org.codehaus.jackson.map.DeserializationContext;
import org.codehaus.jackson.map.JsonDeserializer;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.node.IntNode;
import org.mule.api.MuleMessage;

import Common.Constants;
import entities.product;

public class ProductConverter extends JsonDeserializer<product>   {

	@Override
	public product deserialize(JsonParser jp, DeserializationContext
ctx) throws IOException, JsonProcessingException {

		JsonNode node = jp.getCodec().readTree(jp);
		 long id = node.get("ProductId").asLong();;
		 String name = node.get("Name").asText();
		 String description = node.get("Description").asText();
		 String category = node.get("Category").asText();
		 String otherInfo = node.get("OtherInfo").asText();
		 String priceStr = node.get("Quantity").asText();
		 Float price = Float.parseFloat(priceStr);
		 String quantity = ((IntNode)
node.get("Quantity")).getNumberValue().toString();

		return new product(id, Constants.VendasOnlineId, name,
description, price, quantity, category, otherInfo,
Constants.VendasOnlineName);
	}
```

```
	public List<product> ConvertMessageProductResponse(MuleMessage
message){
		List<product> productsList = new ArrayList<>();
		ObjectMapper mapper = new ObjectMapper();

		String  payload = ReadInputStream(message.getPayload());

		try {
			payload = payload.replace("[", "").replace("]", ",");

			String[] jsonObjects = payload.split("},");

			for(int i=0; i< jsonObjects.length; i++){
				product prod =
mapper.readValue(jsonObjects[i]+"}", product.class);
				productsList.add(prod);
			}
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		return productsList;
	}

	public String ReadInputStream(Object object) {
		InputStream content = (InputStream)object;

		  int numRead;
		final int bufferSize = 1024;
		byte[] buffer = new byte[bufferSize];
		ByteArrayOutputStream outString = new
ByteArrayOutputStream();
		try{
		   while ((numRead = content.read(buffer)) != -1) {
			outString.write(buffer, 0, numRead);
		    }
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();

		} finally {
		  try {
			content.close();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		return new String(outString.toByteArray());
	}

	public List<product> ConvertStringProductResponse(String
payload){
		List<product> productsList = new ArrayList<>();
		ObjectMapper mapper = new ObjectMapper();

		try {
			payload = payload.replace("[", "").replace("]", ",");
```

```
                String[] jsonObjects = payload.split("},");

                for(int i=0; i< jsonObjects.length; i++){
                      product prod =
mapper.readValue(jsonObjects[i]+"}", product.class);
                      productsList.add(prod);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return productsList;
      }
}
```

3.3.2. ProductListToProductListTransformer

```
package Transformers;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

import org.codehaus.jackson.map.ObjectMapper;
import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;

import entities.product;

public class ProductListToProductListTransformer extends
AbstractMessageTransformer{

      @Override
      public Object transformMessage(MuleMessage message, String
outputEncoding) throws TransformerException {

            String payload = (String)message.getPayload();
            List<product> productsList = new ArrayList<>();
            ObjectMapper mapper = new ObjectMapper();

            try {
                payload = payload.replace("[", "").replace("]", ",");
                String[] jsonObjects = payload.split("},");

                for(int i=0; i< jsonObjects.length; i++){
                      product prod =
mapper.readValue(jsonObjects[i]+"}", product.class);
                      productsList.add(prod);
                }

            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return productsList;
      }
}
```

3.3.3. ProductToProductTransformer

```
package Transformers;

import java.io.IOException;

import org.codehaus.jackson.map.ObjectMapper;
import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;

import entities.product;

public class ProductToProductTransformer extends
AbstractMessageTransformer{

	@Override
	public Object transformMessage(MuleMessage message, String
outputEncoding) throws TransformerException {

		String payload = (String)message.getPayload();
		product newProduct = null;
		ObjectMapper mapper = new ObjectMapper();

		try {
			payload = payload.replace("[", "").replace("]", ",");

			String[] jsonObjects = payload.split("},");

			for(int i=0; i< jsonObjects.length; i++){
				newProduct =
mapper.readValue(jsonObjects[i]+"}", product.class);
			}

		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return newProduct;
	}
}
```

3.3.4. EbayProductListToProductList

```
package Transformers;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import Common.Constants;
import entities.product;

public class EbayProductListToProductList extends
AbstractMessageTransformer {

	@Override
	public Object transformMessage(MuleMessage message, String
outputEncoding)
			throws TransformerException {

		String payload = (String) message.getPayload();
		List<product> productsList = new ArrayList<>();

		DocumentBuilder db;
		try {
			db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

			InputSource is = new InputSource();
			is.setCharacterStream(new StringReader(payload));

			Document doc = db.parse(is);
			NodeList nodes = doc.getElementsByTagName("item");

			for (int i = 0; i < nodes.getLength(); i++) {
				Element element = (Element) nodes.item(i);
				NodeList productIdNodes = element
						.getElementsByTagName("itemId");

				if (productIdNodes.getLength() > 0) {

					String productIdStr =
productIdNodes.item(0)
							.getTextContent();
					long id = Long.parseLong(productIdStr);
					String name =
element.getElementsByTagName("title").item(0)
							.getTextContent();
					if (name != "") {
						name = name.replace("&", "and");
					}
					String description = "";
					String quantity = "";

					NodeList categoryElements = element

	.getElementsByTagName("categoryName");
					String category = "";
					for (int cat = 0; cat <
categoryElements.getLength(); cat++) {
						category +=
categoryElements.item(cat).getTextContent()
								+ " ";
					}
```

```
                              if (category != "") {
                                    category = category.replace("&",
"and");
                              }

                              String priceStr = element

      .getElementsByTagName("currentPrice").item(0)
                                          .getTextContent();
                              Float price = Float.parseFloat(priceStr);

                              String otherInfo="";
                              NodeList nodeInfo =
element.getElementsByTagName("conditionDisplayName");
                              if(nodeInfo.getLength() > 0){
                                    otherInfo +=
nodeInfo.item(0).getTextContent();
                              }

                              nodeInfo =
element.getElementsByTagName("location");
                              if(nodeInfo.getLength() > 0){
                                    otherInfo +=
nodeInfo.item(0).getTextContent();
                              }
                              if (otherInfo != "") {
                                    otherInfo = otherInfo.replace("&",
"and");
                              }

                              productsList.add(new product(id,
Constants.EbayId, name,
                                          description, price, quantity,
category, otherInfo,
                                          Constants.EbayName));
                        }
                  }
            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return productsList;
      }
}
```

3.3.5. EbayProductToProductTransformer

```
package Transformers;

import java.io.StringReader;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import Common.Constants;
import entities.product;

public class EbayProductToProductTransformer extends
AbstractMessageTransformer {

	@Override
	public Object transformMessage(MuleMessage message, String
outputEncoding)
			throws TransformerException {

		product newProduct = null;
		String payload = (String) message.getPayload();

		DocumentBuilder db;
		try {
			db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

			InputSource is = new InputSource();
			is.setCharacterStream(new StringReader(payload));

			Document doc = db.parse(is);
			NodeList nodes = doc.getElementsByTagName("Item");

			if (nodes.getLength() > 0) {
				Element element = (Element) nodes.item(0);
				NodeList productIdNodes = element
						.getElementsByTagName("ItemID");

				if (productIdNodes.getLength() > 0) {

					String productIdStr =
productIdNodes.item(0)
							.getTextContent();
					long id = Long.parseLong(productIdStr);
					String name =
element.getElementsByTagName("Title").item(0)
							.getTextContent();
					if (name != "") {
						name = name.replace("&", "and");
					}
					String description = "";
					String quantity = "";

					NodeList categoryElements = element

	.getElementsByTagName("PrimaryCategoryName");
					String category = "";
					for (int cat = 0; cat <
categoryElements.getLength(); cat++) {
						category +=
categoryElements.item(cat).getTextContent()
								+ " ";
					}
```

```
                              if (category != "") {
                                    category = category.replace("&",
"and");
                              }

                              String priceStr = element

      .getElementsByTagName("ConvertedCurrentPrice")
                                          .item(0).getTextContent();
                              Float price = Float.parseFloat(priceStr);

                              String otherInfo = element

      .getElementsByTagName("ConditionDisplayName")
                                          .item(0).getTextContent();
                              otherInfo += "; "
                                          +
element.getElementsByTagName("Location").item(0)

      .getTextContent();
                              if (otherInfo != "") {
                                    otherInfo = otherInfo.replace("&",
"and");
                              }

                              newProduct = new product(id,
Constants.EbayId, name,
                                          description, price, quantity,
category, otherInfo,
                                          Constants.EbayName);
                        }
                  }
            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return newProduct;
      }
}
```

3.3.6. BookListToProductListTransformer

```
package Transformers;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
```

```
import org.xml.sax.InputSource;

import Common.Constants;
import entities.product;

public class BookListToProductListTransformer extends
AbstractMessageTransformer {

      @Override
      public Object transformMessage(MuleMessage message, String
outputEncoding)
                  throws TransformerException {

            List<product> productsList = new ArrayList<>();
            String payload = (String)message.getPayload();

            DocumentBuilder db;
            try {
                  db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

                  int initialPosition = payload.indexOf("<ns2:");
                String bookXML = payload.substring(initialPosition);


                InputSource is = new InputSource();
                is.setCharacterStream(new StringReader(bookXML));

                  Document doc = db.parse(is);
                  NodeList nodes = doc.getElementsByTagName("return");

                  for(int i=0; i<nodes.getLength(); i++){
                        Element element = (Element)nodes.item(i);

                        String productIdStr =
element.getElementsByTagName("bookId").item(0).getTextContent();
                        int id = Integer.parseInt(productIdStr);
                      String name =
element.getElementsByTagName("title").item(0).getTextContent();

                      NodeList descriptionNode =
element.getElementsByTagName("description");
                      String description= "";
                      if(descriptionNode.getLength() > 0){
                        description =
descriptionNode.item(0).getTextContent();
                      }
                      String category = "Book";
                      String priceStr =
element.getElementsByTagName("price").item(0).getTextContent();
                      Float price = Float.parseFloat(priceStr);
                      String quantityStr =
element.getElementsByTagName("quantity").item(0).getTextContent();

                      String otherInfo = "";
                      NodeList authorsList =
element.getElementsByTagName("authors");

                      for(int j=0; j<authorsList.getLength(); j++){
```

```
                        otherInfo +=
authorsList.item(j).getTextContent();
                    }
                    productsList.add(new product(id,
Constants.LivrosOnlineId, name, description, price, quantityStr,
category, otherInfo, Constants.LivrosOnlineName));
                }

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            return productsList;
      }
}
```

3.3.7. BookToProductTransformer

```
package Transformers;

import java.io.StringReader;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import Common.Constants;
import entities.product;

public class BookToProductTransformer extends
AbstractMessageTransformer {

      @Override
      public Object transformMessage(MuleMessage message, String
outputEncoding)
                  throws TransformerException {

            product newProduct = null;
            String payload = (String)message.getPayload();

            DocumentBuilder db;
            try {
                  db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

                  int initialPosition = payload.indexOf("<ns2:");
                String bookXML = payload.substring(initialPosition);


                InputSource is = new InputSource();
```

```
                is.setCharacterStream(new StringReader(bookXML));

                  Document doc = db.parse(is);
                  NodeList nodes = doc.getElementsByTagName("return");

                  for(int i=0; i<nodes.getLength(); i++){
                        Element element = (Element)nodes.item(i);


                        String productIdStr =
element.getElementsByTagName("bookId").item(0).getTextContent();
                        int id = Integer.parseInt(productIdStr);
                      String name =
element.getElementsByTagName("title").item(0).getTextContent();

                      NodeList descriptionNode =
element.getElementsByTagName("description");
                      String description= "";
                      if(descriptionNode.getLength() > 0){
                        description =
descriptionNode.item(0).getTextContent();
                      }
                      String category = "Book";
                      String priceStr =
element.getElementsByTagName("price").item(0).getTextContent();
                      Float price = Float.parseFloat(priceStr);
                      String quantityStr =
element.getElementsByTagName("quantity").item(0).getTextContent();

                      String otherInfo = "";
                      NodeList authorsList =
element.getElementsByTagName("authors");

                      for(int j=0; j<authorsList.getLength(); j++){
                        otherInfo +=
authorsList.item(j).getTextContent();
                      }

                      newProduct = new product(id,
Constants.LivrosOnlineId, name, description, price, quantityStr,
category, otherInfo, Constants.LivrosOnlineName);
                  }

            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return newProduct;
      }
}
```

3.3.8. SoapRequestTransformer

```
package Transformers;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import Common.SoapOperations;
import entities.ProductIdentifier;
import entities.SoapRequest;

public class SoapRequestTransformer extends AbstractMessageTransformer
{

	@Override
	public Object transformMessage(MuleMessage message, String
outputEncoding)
			throws TransformerException {

		SoapRequest request = new SoapRequest();
		String payload = (String) message.getPayload();

		DocumentBuilder db;
		try {
			db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

			InputSource is = new InputSource();
			is.setCharacterStream(new StringReader(payload));

			String searchOperationNode = "tem:SearchProduct";
			String compareOperationNode = "tem:CompareProducts";
			String viewOperationNode = "tem:ViewProduct";

			Document doc = db.parse(is);

			if
(doc.getElementsByTagName(searchOperationNode).getLength() > 0) {


	request.setOperation(SoapOperations.SearchProduct);

			} else if
(doc.getElementsByTagName(viewOperationNode).getLength() > 0) {


	request.setOperation(SoapOperations.ViewProduct);
```

```
                        request = GetProductsIdentifiers(doc, request);

                  } else if
(doc.getElementsByTagName(compareOperationNode).getLength() > 0) {


      request.setOperation(SoapOperations.CompareProduct);

      request.setProducts(GetProductsIndentifierList(doc));
                  }

            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return request;
      }

      public String GetSearchParameters(Document doc) {

            String searchParameters = "";

            try {

                  String parameterNodeName = "arr:string";

                  NodeList nodes =
doc.getElementsByTagName(parameterNodeName);

                  if (nodes.getLength() == 0) {
                        nodes = doc.getElementsByTagName("string");
                  }

                  for (int i = 0; i < nodes.getLength(); i++) {

                        searchParameters +=
nodes.item(i).getTextContent();
                  }

            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return searchParameters;
      }


      public List<ProductIdentifier>
GetProductsIndentifierList(Document doc) {

            List<ProductIdentifier> productsIdentifiers = new
ArrayList<>();

            try {

                  String productIdNodeName = "ven:ProductId";
                  String vendorIdNodeName = "ven:VendorId";
```

```
                  NodeList nodes = doc

      .getElementsByTagName("ven:ProductIdentifierDataContract");

                  if (nodes.getLength() == 0) {
                        nodes = doc

      .getElementsByTagName("ProductIdentifierDataContract");
                        productIdNodeName = "ProductId";
                        vendorIdNodeName = "VendorId";
                  }

                  for (int i = 0; i < nodes.getLength(); i++) {
                        Element element = (Element) nodes.item(i);

                        if
(element.getElementsByTagName(productIdNodeName).getLength() > 0
                                    &&
element.getElementsByTagName(vendorIdNodeName)
                                                .getLength() > 0) {
                              String productIdStr = element

      .getElementsByTagName(productIdNodeName).item(0)
                                          .getTextContent();
                              int productId =
Integer.parseInt(productIdStr);

                              String quantityStr = element

      .getElementsByTagName(vendorIdNodeName).item(0)
                                          .getTextContent();
                              int vendorId =
Integer.parseInt(quantityStr);

                              productsIdentifiers.add(new
ProductIdentifier(productId,
                                          vendorId));
                        }
                  }

            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            return productsIdentifiers;
      }

      private SoapRequest GetProductsIdentifiers(Document doc,
SoapRequest request) {

            try {

                  String productIdNodeName = "tem:productId";
                  String vendorIdNodeName = "tem:vendorId";

                  NodeList productNode =
doc.getElementsByTagName(productIdNodeName);
```

```
                    if (productNode.getLength() == 0) {
                          productIdNodeName = "ProductId";
                          productNode =
doc.getElementsByTagName(productIdNodeName);
                    }

                    NodeList vendorNode =
doc.getElementsByTagName(vendorIdNodeName);

                    if (vendorNode.getLength() == 0) {
                          vendorIdNodeName = "VendorId";
                          vendorNode =
doc.getElementsByTagName(vendorIdNodeName);
                    }

                    String productIdStr =
productNode.item(0).getTextContent();
                    request.setProduct(productIdStr);

                    String vendorStr =
vendorNode.item(0).getTextContent();
                    request.setVendor(vendorStr);

              } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
              }

              return request;
        }
}
```

3.3.9. ProductIdentifierTransformer

```
package Transformers;

import java.io.StringReader;
import java.util.ArrayList;
import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

import entities.ProductIdentifier;

public class ProductIdentifierTransformer extends
AbstractMessageTransformer {
```

```
	@Override
	public Object transformMessage(MuleMessage message, String
outputEncoding) throws TransformerException {

		List<ProductIdentifier> productsIdentifiers = new
ArrayList<>();
		String payload = (String)message.getPayload();

		DocumentBuilder db;
		try {
			db =
DocumentBuilderFactory.newInstance().newDocumentBuilder();

			InputSource is = new InputSource();
			is.setCharacterStream(new StringReader(payload));

			String productIdNodeName = "ven:ProductId";
			String vendorIdNodeName = "ven:VendorId";

			  Document doc = db.parse(is);
			  NodeList nodes =
doc.getElementsByTagName("ven:ProductIdentifierDataContract");

			if(nodes.getLength() ==0){
				nodes =
doc.getElementsByTagName("ProductIdentifierDataContract");
				productIdNodeName = "ProductId";
				vendorIdNodeName ="VendorId";
			}

			for(int i=0; i<nodes.getLength(); i++){
				Element element = (Element)nodes.item(i);

	if(element.getElementsByTagName(productIdNodeName).getLength()
>0 && element.getElementsByTagName(vendorIdNodeName).getLength() > 0)
				{
					  String productIdStr =
element.getElementsByTagName(productIdNodeName).item(0).getTextContent
();
					  long productId =
Long.parseLong(productIdStr);

					String quantityStr =
element.getElementsByTagName(vendorIdNodeName).item(0).getTextContent(
);
					int vendorId =
Integer.parseInt(quantityStr);

					productsIdentifiers.add(new
ProductIdentifier(productId, vendorId));
				}
			}

		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		return productsIdentifiers;
	}
}
```

3.3.10. CompareResponseTransformer

```
package Transformers;

import java.util.List;

import org.mule.api.MuleMessage;
import org.mule.api.transformer.TransformerException;
import org.mule.transformer.AbstractMessageTransformer;

import entities.product;

public class CompareResponseTransformer extends
AbstractMessageTransformer {

      @Override
      public Object transformMessage(MuleMessage message, String
outputEncoding)throws TransformerException {

            String soapResponse = "<?xml version=\"1.0\"
encoding=\"UTF-8\"?><CompareProductsResponse
xmlns=\"http://tempuri.org/\">"
                        + "<CompareProductsResult
xmlns:a=\"http://schemas.datacontract.org/2004/07/VendasOnline.WCFServ
ices.DataContracts\" xmlns:i=\"http://www.w3.org/2001/XMLSchema-
instance\">";

            List<product> payload = (List<product>)
message.getPayload();

            for (product prod : payload) {
                  soapResponse +=
"<a:ProductsDataContract><a:Category>"
                              + prod.getCategory() + "</a:Category>";
                  if (prod.getQuantity() != null && prod.getQuantity()
!= "") {
                        soapResponse += "<a:Description>" +
prod.getDescription()
                                    + "</a:Description>";
                  }
                  soapResponse += "<a:Name>" + prod.getName()
                              + "</a:Name><a:Other_Info>" +
prod.getOther_Info()
                              + "</a:Other_Info><a:Price>" +
prod.getPrice()
                              + "</a:Price><a:ProductId>" +
prod.getProductId()
                              + "</a:ProductId>";
                  if (prod.getQuantity() != null && prod.getQuantity()
!= "") {
                        soapResponse += "<a:Quantity>" +
prod.getQuantity()
                                    + "</a:Quantity>";
                  }
                  soapResponse += "<a:VendorId>" + prod.getVendorId()
                              + "</a:VendorId><a:VendorName>" +
prod.getVendorName()
```

```
                                + "</a:VendorName>" +
"</a:ProductsDataContract>";
            }

            soapResponse +=
"</CompareProductsResult></CompareProductsResponse>";

            return soapResponse;
      }
}
```

3.4. Agregadores

3.4.1. SearchResponseAggregator

```
package aggregators;

import java.util.ArrayList;
import java.util.List;

import org.mule.DefaultMuleEvent;
import org.mule.api.MuleEvent;
import org.mule.api.MuleException;
import org.mule.api.MuleMessage;
import org.mule.api.routing.AggregationContext;
import org.mule.routing.AggregationStrategy;

import entities.product;

public class SearchResponseAggregator implements AggregationStrategy {

      @Override
      public MuleEvent aggregate(AggregationContext context) throws
MuleException {

            // mule event that will be rewritten
            MuleEvent originalEvent = context.getOriginalEvent();
            // message which payload will be rewritten
            MuleMessage message = originalEvent.getMessage();

            List<product> productsList = new ArrayList<product>();

          for (MuleEvent event :
context.collectEventsWithoutExceptions()) {
            List<product> resultProductsList =
(List<product>)event.getMessage().getPayload();
            productsList.addAll(resultProductsList);
          }

          message.setPayload(productsList);

          return new DefaultMuleEvent(message, originalEvent);
      }
}
```

3.4.2. CompareResponseAggregator

```
package aggregators;

import java.util.ArrayList;
import java.util.List;

import org.mule.DefaultMuleEvent;
import org.mule.api.MuleEvent;
import org.mule.api.MuleException;
import org.mule.api.MuleMessage;
import org.mule.api.routing.AggregationContext;
import org.mule.routing.AggregationStrategy;
import entities.product;

public class CompareResponseAggregator implements AggregationStrategy
{

      @Override
      public MuleEvent aggregate(AggregationContext context) throws
MuleException {

            // mule event that will be rewritten
            MuleEvent originalEvent = context.getOriginalEvent();
            // message which payload will be rewritten
            MuleMessage message = originalEvent.getMessage();

            List<product> productsList = new ArrayList<product>();
           // MuleEvent result = null;

          for (MuleEvent event :
context.collectEventsWithoutExceptions()) {
            try{
                  MuleMessage eventMessage = event.getMessage();
                  if(eventMessage.getPayload() !=
org.mule.transport.NullPayload.getInstance() ){
                        product resultProduct =
(product)eventMessage.getPayload();
                        productsList.add(resultProduct);
                  }
            }catch(Exception exc){
                  exc.printStackTrace();
            }
          }

          message.setPayload(productsList);

          return new DefaultMuleEvent(message, originalEvent);
      }
}
```

Instituto Politécnico de Coimbra

Instituto Superior de Engenharia de Coimbra

Departamento de Engenharia Informática e de Sistemas

Mestrado em Informática e Sistemas
Estágio/Projeto Industrial
Relatório Final

Anexo C – Testes ao Sistema

Carla Maria Silva Machado

Orientadores:

João Carlos Costa Faria da Cunha

Cristiana Manuela Afonso Areias

Instituto Superior de Engenharia de Coimbra

Coimbra, dezembro, 2015

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

O sistema implementado foi sujeito a alguns testes de forma a obter alguns valores referentes à performance do sistema, em particular de cada operação. Para cada operação implementada foram criados testes, com diferentes parâmetros de entrada, sendo executados testes de performance e de carga.

Neste documento são apresentados os pedidos definidos para cada teste assim como as condições de sucesso dos testes, sendo apresentados na íntegra os resultados dos testes realizados ao sistema assim como a análise realizada aos valores obtidos.

2. Testes ao Sistema

Neste capítulo será apresentado o ambiente de testes e também serão apresentados os testes realizados ao sistema, especificando para cada operação os diferentes pedidos e as condições de sucesso, sendo os resultados das execuções dos testes apresentados e analisados.

2.1. Ambiente de testes

Sempre que são efetuados testes de performance um ponto muito importante é o ambiente em que estes são realizados, uma vez que a capacidade das máquinas tem influência na performance do sistema, uma vez que uma máquina com pouco poder de processamento pode limitar a performance dos sistemas nela hospedados.
O sistema foi implementado numa máquina com as seguintes características:

- Sistema Operativo: Windows 7 64bit,
- RAM: 8GB
- Processador: 2.20GHz, dual-core
- Disco: 50GB

Esta máquina encontra-se hospedada num servidor de máquinas virtuais disponibilizado pelo Instituto Superior de Engenharia de Coimbra.
Nesta máquina foram instalados os diversos softwares necessários ao funcionamento do sistema, como o Apache Tomcat, o Mule ESB e o ActiveMQ. Na máquina foram igualmente instalados os serviços das organizações Vendas Online e Livros Online.
Nesta máquina foi também instalada a ferramenta SoapUI que foi utilizada para a realização dos testes ao ambiente que foram efetuados a partir da própria máquina.

2.2. Pesquisa de Produtos

Uma das funcionalidades disponibilizadas pelo sistema é a pesquisa de produtos. Este pedido vai procurar em todos os fornecedores disponíveis, por realização de chamadas aos serviços que estes disponibilizam, produtos que correspondam aos parâmetros de pesquisa definidos pelo cliente.
Para a obtenção de tempos de resposta deste pedido foram criados quatro testes com quatro pedidos distintos:

- Cerca 100 – pedido de pesquisa com cerca de 100 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “Statistics”;
- Acima 100 – pedido de pesquisa com mais de 100 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “cooking”;
- Acima 300 – pedido de pesquisa com mais de 300 resultados dos três fornecedores disponíveis
    - Parâmetro searchParameters – “Potter”;
- 0 Resultados – pedido de pesquisa que não deverá retornar qualquer resultado

- Parâmetro searchParameters – “«««»»»”;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200,
- Tempo máximo de resposta – 90000ms,
- Conteúdo da mensagem – a palavra product, parte do xml retornado na resposta,
- Conteúdo da mensagem – parâmetro de pesquisa.

Os testes de performance e de carga foram realizados com base nestes pedidos, sendo os resultados obtidos apresentados nas secções seguintes.

2.2.1. Testes de Performance

Para a obtenção dos tempos de resposta da operação de pesquisa de produtos os pedidos descritos foram executados 30 vezes.

A Figura 2.1 apresenta o resultado apresentado pelo SoapUI de uma das execuções, onde se pode visualizar o sucesso dos pedidos e os tempos de resposta. Nas 30 execuções os testes foram sempre executados com sucesso e os tempos de resposta podem ser consultados na Tabela 2.1.

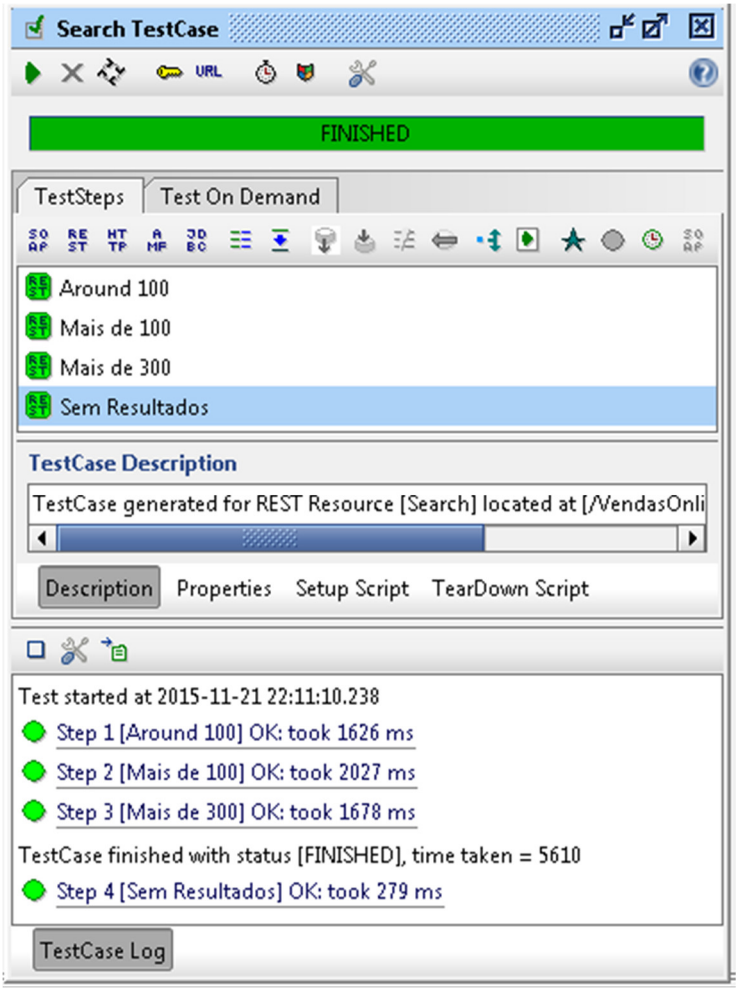

Figura 2.1: Execução testes de pesquisa

Tabela 2.1: Resultados Testes Pesquisa de Produtos

	Tempo (ms)			
	Cerca 100	Acima 100	Acima 300	0 Resultados
1	1487	1887	1650	228
2	1924	1895	1649	232
3	1450	2789	1689	239
4	1402	1708	1823	232
5	1438	1790	1630	240
6	1595	2075	1707	284
7	1657	2289	1812	311
8	1718	1984	1697	269
9	1600	1847	1704	289
10	1626	2027	1678	279
11	1795	2179	1799	256
12	2162	1811	1766	257
13	1727	1908	1667	289
14	1987	1894	1703	249
15	1937	1988	1682	232
16	1897	2155	1624	247
17	1885	1736	1793	322
18	1645	1981	1735	256
19	1728	1914	1705	313
20	1474	1934	1782	233
21	1615	1886	1771	236
22	1630	1836	1540	231
23	1616	1988	1596	229
24	1667	2022	1712	265
25	1758	2084	1766	280
26	1682	2041	1851	285
27	1694	2027	1686	258
28	1651	1996	1740	253
29	1679	2119	1749	264
30	1678	2166	1674	258

Na Tabela 2.2 podemos visualizar o resultado da análise aos resultados obtidos, com a apresentação do valor médio do tempo de resposta, o maior e o menor tempo de resposta registado. Estes valores encontram-se também representados em forma de gráfico e podem ser consultados na Figura 2.2.

Tabela 2.2: Análise resultados Pesquisa de produtos

	Tempo (ms)		
	Máximo	Mínimo	Média
Cerca 100	2162	1402	1693
Acima 100	2789	1708	1998
Acima 300	1851	1540	1713
0 Resultados	322	228	260

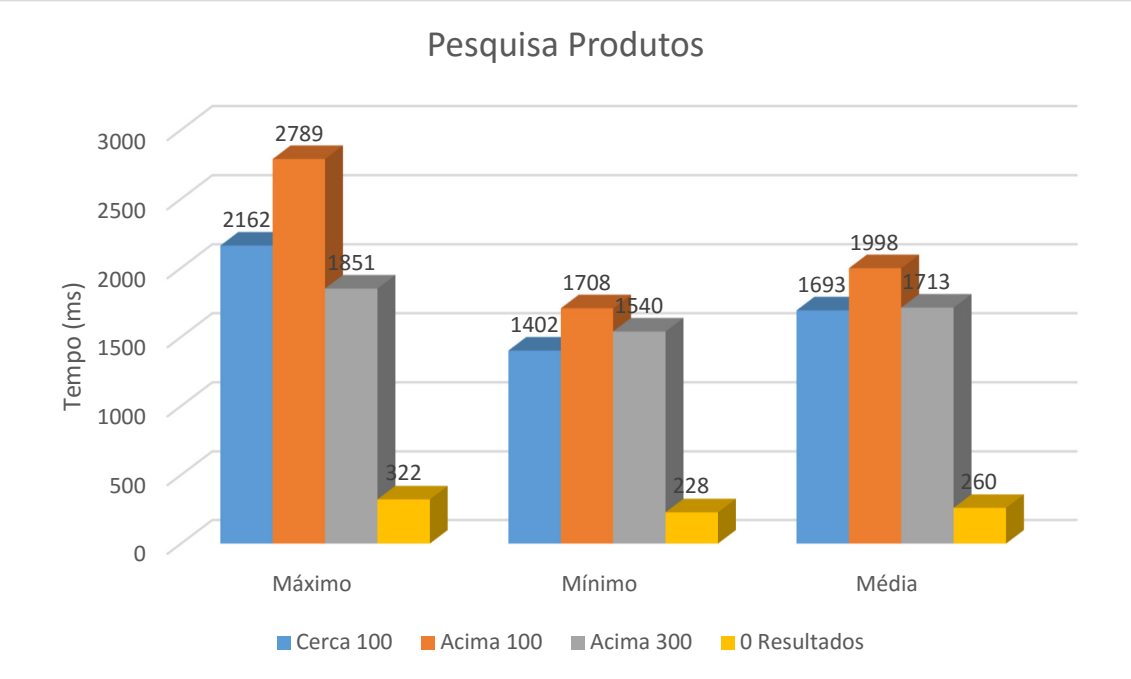

Figura 2.2: Gráfico - Pesquisa Produtos

Analisando os resultados obtidos podemos verificar que, como seria de esperar, o pedido que não retorna resultados é o mais rápido. No entanto e ao contrário do que se poderia esperar, o pedido mais lento não é o que retorna mais resultados (Acima de 300), mas o que retorna mais de 100 (Acima de 100). Este desvio aos resultados esperados pode estar relacionado com a proporção de resultados dos diferentes fornecedores, e o tempo de resposta dos serviços dos fornecedores assim como o tempo de processamento das respostas de cada fornecedor.

Nos resultados obtidos as proporções em percentagem de produtos de cada vendedor foram:

Tabela 2.3: Resultados Pesquisa Produtos - Proporções produtos de vendedores

	Vendas Online	Livros Online	Ebay
Cerca 100	92.59	51.28	25.12
Acima de 100	5.56	18.97	1.51
Acima de 300	1.85	29.74	73.37

Analisando os resultados verifica-se que o pedido mais lento é o que apresenta uma maior proporção de produtos do vendedor Livros Online, podendo por isso estar relacionado com uma pior performance da fonte de dados.

2.2.2. Testes de Carga

Tendo executado os testes isoladamente realizou-se então um teste de carga, tendo o teste sido executado durante um período de 120 segundos. Na Figura 2.3 podemos ver uma imagem dos resultados apresentados pelo SoapUI após a execução dos testes.

Observando os resultados dos testes de carga verifica-se a mesma tendência dos testes anteriores nos tempos de resposta dos diferentes pedidos, o pedido mais lento é o que retorna mais de 100 resultados (Acima 100) e o mais rápido o pedido que não retorna resultados.

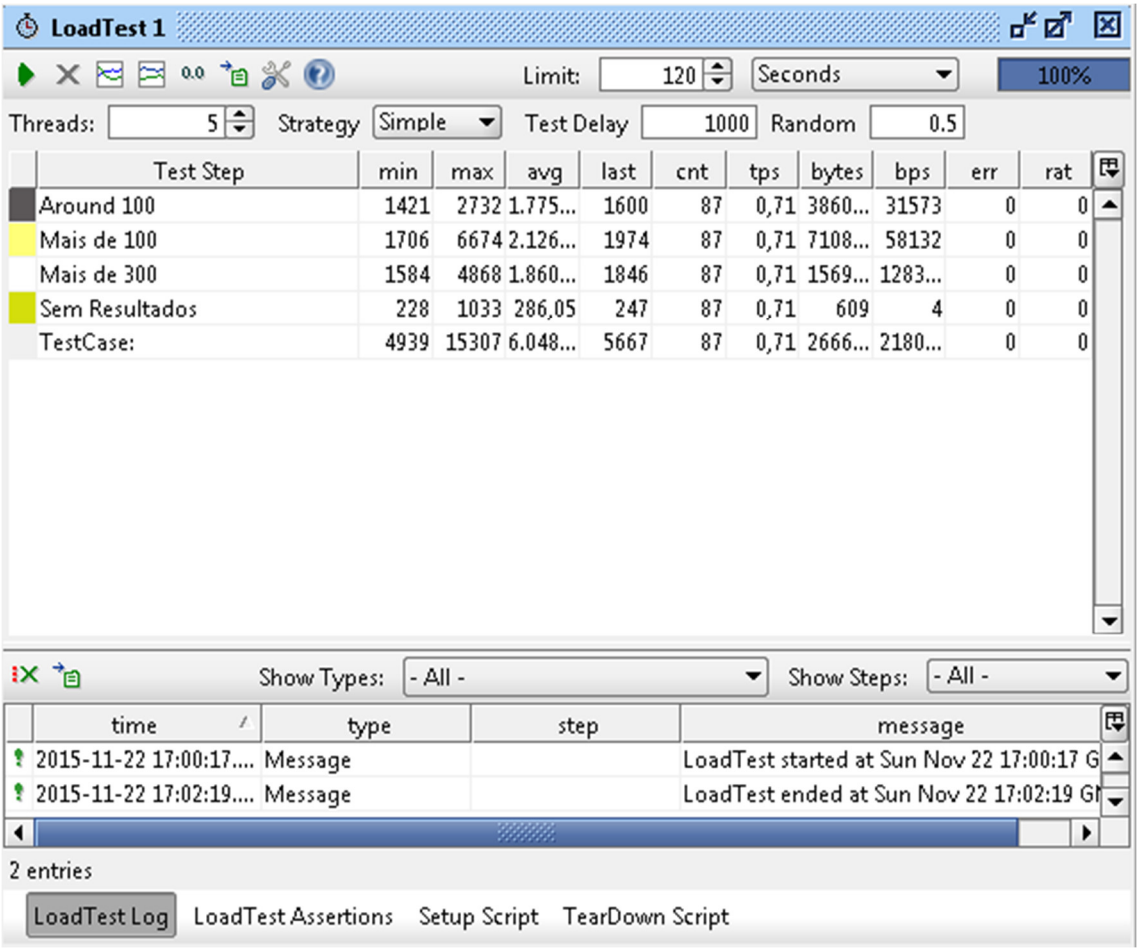

Figura 2.3: SoapUI Resultados teste carga Pesquisa Produtos

2.3. Obtenção de Detalhes de Produto

Uma das funcionalidades disponibilizadas pelo sistema é a obtenção dos detalhes de um produto em particular. Esta operação vai requisitar os detalhes do produto pretendido ao seu fornecedor, realizando a chamada ao serviço correspondente.
Para a obtenção de tempos de resposta desta operação foram criados quatro testes com quatro pedidos distintos:

- Ebay – pedido de detalhes de um produto da organização Ebay
  - Parâmetro vendor – 3;
  - Parâmetro id – 131604043953;
- Livros Online – pedido de detalhes de um produto da organização Livros Online
  - Parâmetro vendor – 2;
  - Parâmetro id – 6254;
- Vendas Online – pedido de detalhes de um produto da organização Vendas Online
  - Parâmetro vendor – 1;
  - Parâmetro id – 56;
- Vendedor Inválido – pedido de detalhes de um produto com um identificador de organização inválido
  - Parâmetro vendor – 4;
  - Parâmetro id – 1;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200
- Tempo máximo de resposta – 30000ms
- Conteúdo da mensagem – Nome do vendedor
- Conteúdo da mensagem – Nome do produto

Os testes de performance e de carga foram realizados com base nestes pedidos, sendo os resultados obtidos apresentados nas secções seguintes.

2.3.1. Testes de Performance

Para a obtenção dos tempos de resposta da operação de obtenção de detalhes os pedidos descritos foram executados 30 vezes, os testes foram todos executados com sucesso e os tempos de resposta podem ser consultados na Tabela 2.4.

Tabela 2.4: Resultados Testes Obtenção Detalhes de Produto

	Tempo (ms)			
	Ebay	Livros Online	Vendas Online	Vendedor Inválido
1	300	139	59	23
2	500	161	29	24
3	288	136	25	23
4	288	136	26	26

5	759	147	27	24
6	590	155	27	23
7	629	131	26	24
8	512	127	24	24
9	312	141	25	23
10	568	140	30	24
11	726	126	36	32
12	521	168	29	27
13	334	111	28	34
14	308	124	27	32
15	530	121	31	24
16	319	113	25	28
17	507	116	28	28
18	304	114	27	27
19	709	125	29	24
20	719	122	28	23
21	425	120	38	25
22	319	124	25	26
23	533	117	29	26
24	330	114	26	23
25	327	104	26	25
26	532	126	29	26
27	525	127	24	23
28	474	113	28	25
29	383	109	24	23
30	527	116	27	23

Na Tabela 2.5 podemos visualizar o resultado da análise aos resultados obtidos, com a apresentação do valor médio do tempo de resposta, o maior e o menor tempo de resposta registados. Estes valores encontram-se também representados em forma de gráfico e podem ser consultados na Figura 2.4.

Tabela 2.5: Análise resultados Obtenção Detalhes de Produto

	Tempo (ms)		
	Máximo	Mínimo	Média
Ebay	759	288	470
Livros Online	168	104	127
Vendas Online	59	24	29
Vendedor Inválido	34	23	25

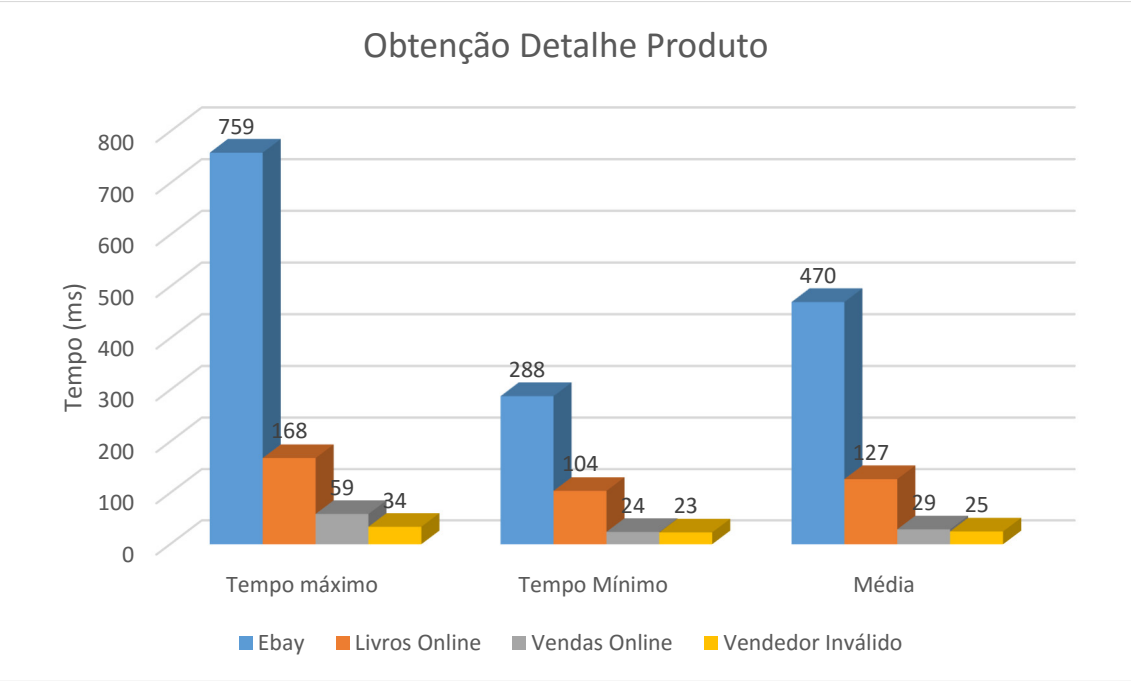

Figura 2.4: Gráfico - Detalhes de Produto

Fazendo uma análise dos valores podemos verificar que como expectável existem variações dos tempos de resposta dos pedidos entre as várias execuções assim como entre os diferentes pedidos realizados. Tendencialmente o pedido para o vendedor Ebay é o mais lento e o pedido para o Vendas Online o mais rápido de entre os pedidos com vendedores válidos.

Uma possível razão para o pedido para o Vendas Online ser mais rápido pode ser por a chamada ao ProductsService ser mais rápida que a chamada aos restantes serviços quer pela quantidade de registos a pesquisar, o Ebay deve possuir uma quantidade largamente superior de registos, quer por uma pior performance da fonte de dados, o sqlite do

LivrosOnline. Outro fator que pode influenciar este resultado é o tempo de conversão de resposta, uma vez que a resposta do ProductsService é um json enquanto que dos outros serviços é um xml.
O pedido para um vendedor inválido é o mais rápido de todos, como esperado uma vez que neste caso o *service bus* identifica que se trata de um vendedor inválido retornando logo a resposta sem realizar um pedido a um novo serviço para obtenção dos detalhes.

2.3.2. Testes de Carga

Os resultados do teste de carga, que executou os 4 pedidos no decorrer de 120 segundos, podem ser visualizados na Figura 2.5.

Analisando os valores obtidos verificam-se as mesmas tendências que nos testes de performance, sendo o mais lento o pedido para o Ebay e o mais rápido o do vendedor inválido, seguido do pedido para o Vendas Online. No entanto nos testes de carga verifica-se que o pedido para o vendedor Livros Online apresenta uma maior variação dos tempos de resposta, sendo o seu tempo máximo de resposta mais lento do que o pedido para o Ebay mantendo, no entanto, uma média mais baixa.

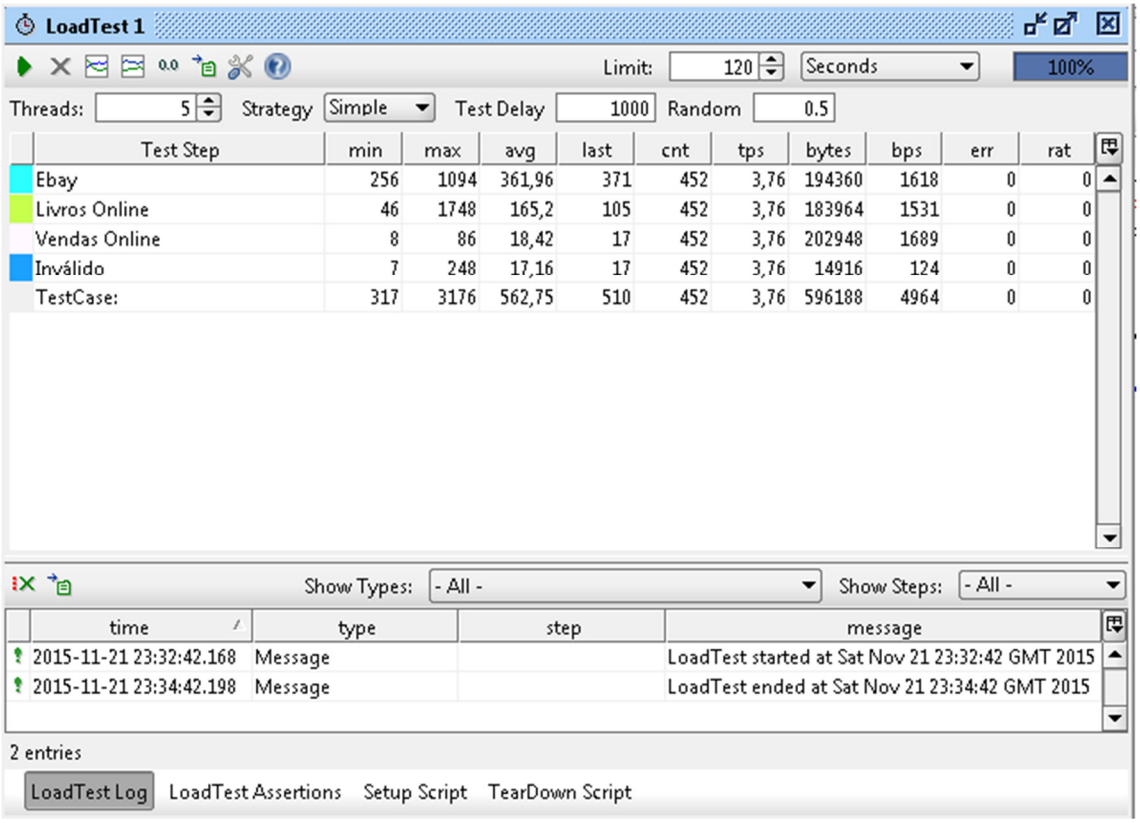

Figura 2.5: SoapUI Resultados teste carga Obtenção Detalhes Produto

2.4. Comparação de Detalhes de Produto

Uma das funcionalidades disponibilizadas pelo sistema é a comparação dos detalhes de até quatro produtos. Este pedido vai obter os detalhes de até um máximo de quatro produtos, pedindo os detalhes de cada produto pretendido ao respetivo fornecedor, realizando para cada produto um pedido ao serviço disponibilizado pelo seu fornecedor.
Para a obtenção de tempos de resposta deste pedido foram criados quatro testes com quatro pedidos distintos:

- 4 Vendedores diferentes – 4 produtos, 1 de cada vendedor e um vendedor repetido,
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 2;
        - Parâmetro ProductId – 6254;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 3;
        - Parâmetro ProductId – 131604043953;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- 3 Vendedores diferentes – 3 produtos, 1 de cada vendedor
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 2;
        - Parâmetro ProductId – 6254;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 3;
        - Parâmetro ProductId – 131604043953;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
- 1 Produto – 1 produto, vendedor Vendas Online
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- 4 Mesmo Vendedor – 4 produtos, todos do mesmo vendedor Vendas Online
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 56;

    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 1;
        - Parâmetro ProductId – 48;
- Vendedor Inválido – 1 produto, vendedor inválido
    - Parâmetro ProductIdentifierDataContract
        - Parâmetro VendorId – 4;
        - Parâmetro ProductId – 6254;

Para cada pedido foram adicionadas algumas asserções de forma a determinar se o teste foi executado com sucesso. As asserções definidas para os testes foram as seguintes:

- Status da resposta – 200,
- Tempo máximo de resposta – 90000ms,
- Resposta SOAP – válida, deve ser uma resposta SOAP válida,
- Conteúdo da mensagem – a palavra ProductsDataContract, que faz parte da resposta SOAP retornada pelo sistema.

Os testes de performance e de carga foram realizados com base nestes pedidos, sendo os resultados obtidos apresentados nas secções seguintes.

2.4.1. Testes de Performance

Para a obtenção dos tempos de resposta da operação de obtenção de detalhes, os pedidos descritos foram executados sequencialmente por 30 vezes, os testes foram todos executados com sucesso e o resumo dos valore obtidos podem ser consultados no gráfico da Tabela 2.6.

Tabela 2.6: Resultados Testes Comparação Produtos

	Tempo (ms)				
	4 Vendedores Diferentes	3 Vendedores Diferentes	1 Produto	4 Mesmo Vendedor	Vendedor Inválido
1	525	316	55	56	47
2	784	549	79	138	61
3	468	347	61	72	58
4	562	564	57	68	56
5	455	373	56	105	50
6	548	537	39	43	41
7	351	381	49	180	46
8	575	349	51	58	52
9	660	368	50	66	46
10	747	400	48	60	45
11	367	322	49	101	50
12	544	326	46	59	66
13	525	521	43	106	49

14	332	380	43	47	49
15	540	527	39	48	40
16	336	544	40	47	58
17	324	500	40	42	43
18	507	497	39	46	38
19	305	305	39	42	42
20	502	366	39	49	41
21	503	513	38	44	43
22	505	306	39	39	35
23	630	303	42	51	49
24	480	505	41	44	42
25	580	303	39	46	40
26	299	289	37	47	43
27	729	398	40	53	40
28	602	326	42	42	47
29	311	330	39	41	41
30	528	319	37	40	36

Analisando os valores obtidos em termos do valor médio do tempo de resposta, o maior e o menor tempo de resposta, foram obtidos os valores apresentados na Tabela 2.7.

Tabela 2.7: Análise resultados Comparação Produtos

	Tempo (ms)		
	Máximo	Mínimo	Média
4 Vendedores Diferentes	784	299	504
3 Vendedores Diferentes	564	289	402
1 Produto	79	37	45
4 Mesmo Vendedor	180	39	63
Vendedor Inválido	66	35	46

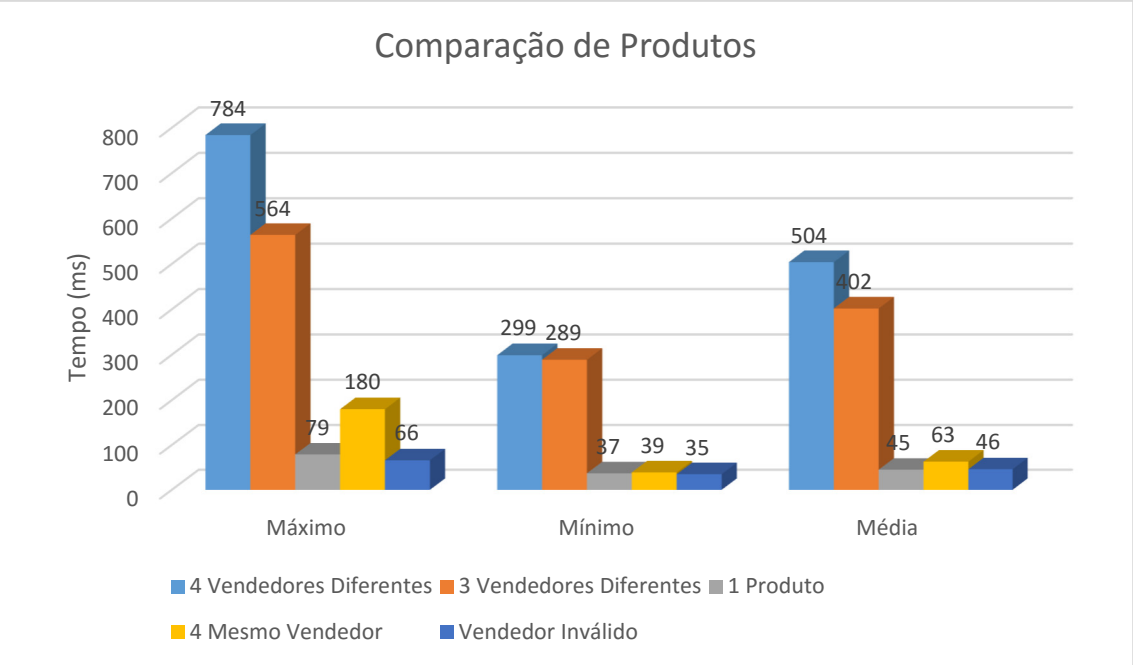

Figura 2.6: Gráfico - Comparação Produtos

Analisando os resultados verifica-se que o pedido mais lento é o dos quatro produtos (4 Vendedores diferentes) e que o mais rápido é o de apenas 1 produto de um vendedor inválido (Vendedor Inválido), neste caso a resposta é retornada pelo *service bus* sem a necessidade da realização de pedidos a outros serviços.

2.4.2. Testes de Carga

Os resultados do teste de carga podem ser consultados na Figura 2.7, estes foram obtidos pela criação de um teste de carga a partir dos pedidos apresentados anteriormente ,que foi executado no decorrer de 120 segundos.

Ao analisar os resultados verificam-se as mesmas tendências que nos testes de performance, sendo que o pedido mais lento é a comparação dos 4 produtos de vendedores distintos e o mais rápido o do vendedor inválido.

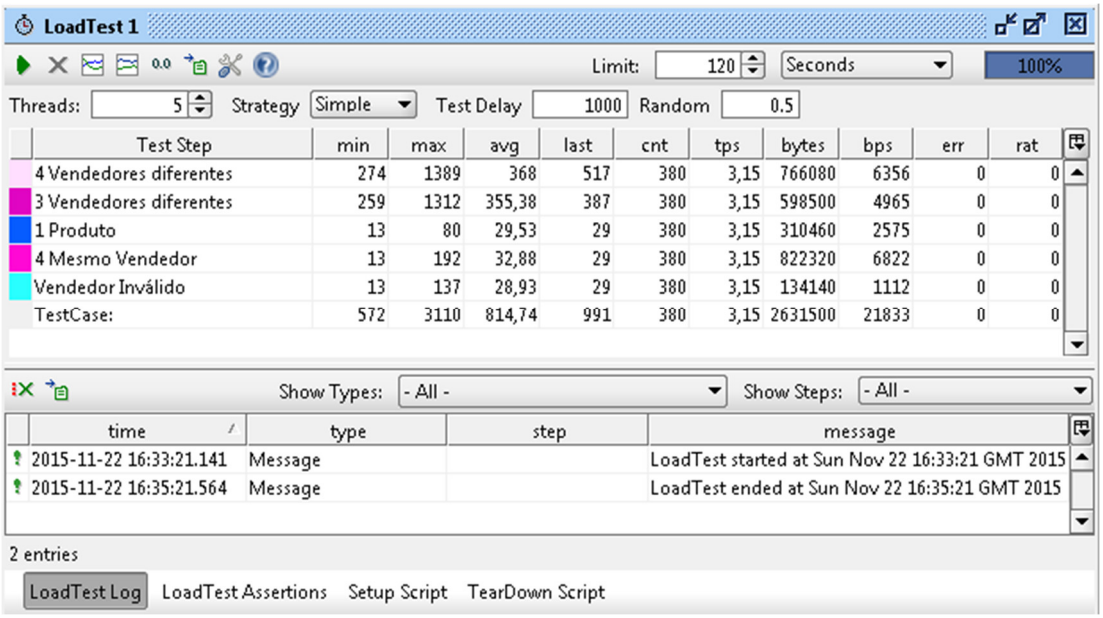

Figura 2.7: SoapUI Resultados teste carga Comparação de Produtos

3. Conclusão

Neste capítulo iremos analisar os valores obtidos nos testes de cada operação de forma a realizar uma comparação dos seus tempos de resposta, assim com tirar uma conclusão dos valores obtidos.

Para a realização da comparação dos tempos de respostas das várias operações, iremos analisar os valores obtidos para os testes de performance para a totalidade de pedidos definidos e para as várias execuções de cada um de forma a identificar o tempo de resposta mais lento, o mais rápido e a média. Para esta análise serão igualmente considerados os pedidos dos quais não se espera obter resultados. Os resultados desta análise podem ser consultados na Tabela 3.1 e na Figura 3.1.

Tabela 3.1: Tempos de resposta do sistema

Operação	Tempo (ms)		
	Máximo	Mínimo	Média
Pesquisa	2789	228	1691
Detalhes	759	23	82
Comparação	784	39	212

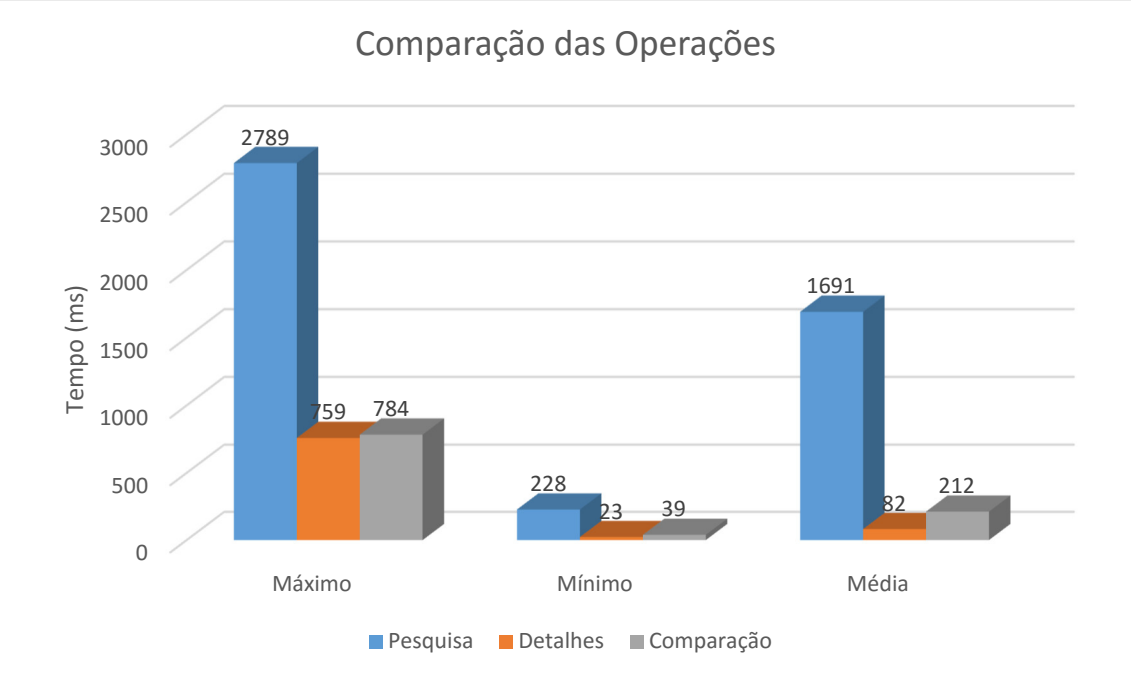

Figura 3.1: Gráfico - Comparação Operações

Observando estes valores podemos verificar que a operação de pesquisa de produtos é a mais lenta, como seria de esperar uma vez que esta operação tendencialmente processa mais informação. A mais rápida é a de obtenção de detalhes, pois processa menos informação e tem o fluxo mais simples.

Olhando para os valores dos tempos de resposta de todas as operações verifica-se que a pesquisa de produtos é a operação mais lenta e o valor máximo desta operação registado durante os testes foi de 2789 milissegundos, ou seja, menos de 3 segundos. Mesmo considerando que a adição de novos fornecedores e a dependência da resposta do sistema da performance dos serviços dos fornecedores, considerando igualmente que ao serem retornado um maior número de resultados na pesquisa o tempo de resposta também deverá aumentar pode-se considerar que o sistema é eficiente ao responder aos pedidos efetuados, podendo os tempos apresentados ser considerados aceitáveis.