FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

iPortalDoc - Enterprise Service BUS (ESB)

Francisco Xavier Gomes Oliveira

MASTER THESIS

Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador Externo: Eng. Telma Salgueiro

Orientador Interno: Prof. Ana Cristina Aguiar

August 10, 2022

Resumo

O iPortaldoc é um Sistema de Gestão Documental e de Processos Corporativos que se encontra integrado no Sistema Operativo IPBRICK OS. O mesmo faz uso de fluxos de trabalho (Workflows) para a gestão dos documentos e processos, promovendo a eficiência do trabalho e a melhoria continua dos processos dentro da Intranet de uma empresa.

Apesar do grande desempenho da aplicação no que toca à gestão da informação dentro da Intranet, no que diz respeito à troca de informação com entidades externas, o iPortalDoc requer a criação de novas rotinas internas ao sistema e uma configuração complexa e trabalhosa, sendo essa uma solução de cariz ad hoc.

Este trabalho aborda o problema de diminuir a complexidade da comunicação do iPortalDoc, através da escolha e integração de uma ferramenta de middleware, o Enterprise Service Bus. Este deve ser Open Source e deve apresentar as funcionalidades necessárias para realizar as operações pretendidas. Pretende-se com essa ferramenta universalizar e centralizar os processos de comunicação mais complexos e permitir que os utilizadores possam fazer uso dessas operações através de uma configuração mais simples e intuitiva.

A ferramenta escolhida para solucionar este problema foi o WSO2 ESB que permite a criação de serviços que permitem utilizar diversos métodos de comunicação diferentes como Web Services, APIs, operações CRUD sobre a informação de uma base de dados, FTP para transferir ficheiros ou SMTP para enviar ou receber emails.

Para além das características descritas anteriormente, este ESB possui uma interface Web de administração bastante intuitiva, o que representa uma mais valia para o utilizador final, pois facilita o uso da ferramenta pelo mesmo. O ESB também possui um conjunto de Web Services de administração que permite o seu controlo através do iPortalDoc, sendo esta característica a de maior relevância no quesito de integração da ferramenta na plataforma.

A solução para a integração do ESB no iPortalDoc consistiu na criação de um sistema de gestão de contas de acesso ao ESB no backend do iPortalDoc através da utilização dos Web Services de administração. Ao nível do frontend foi desenvolvida uma interface para permitir o controlo do acesso dos utilizadores do iPortalDoc ao ESB pelo Administrador e foram desenvolvidos blocos de software para a interação do frontend com o backend. Para além disso, o ESB foi adicionado ao menu de aplicações do iPortalDoc permitindo um acesso direto à ferramenta pelos utilizadores autorizados e foi implementado um método de login automático no ESB.

Após a implementação da solução de integração, foi criado um caso de estudo onde o ESB, através de serviços definidos pelo utilizador, procede à recolha de informação de uma base de dados. Essa informação será usada em diferentes estados de um workflow para realizar o processo de "criação ou atualização de uma entidade no iPortalDoc", demonstrando assim a funcionalidade da ferramenta escolhida.

Abstract

iPortaldoc is a Documents and Corporate Processes' Management System, that is integrated into the IPBRICK Operating System. It makes use of workflows for the management of documents and processes, promoting work efficiency and the continuous improvement of processes within a company's Intranet.

Despite the great performance of the application in managing of information within the Intranet, the exchange of information requires the creation of new internal software routines to the system and a complex and laborious configuration, being this an ad hoc type of solution.

This project addresses the problem of reducing the communication's complexity in iPortalDoc, through the choice and integration of a middleware tool, the Enterprise Service Bus. This tool must be Open Source and must present a set of functionalities to carry out the intended operations. The tool is used with the goal of universalize and centralize the most complex communication processes and allow the users to use these operations through a simpler and more intuitive configuration.

The tool chosen to solve this problem was the WSO2 ESB, which allows the creation of services that allow the use of different communication methods such as Web Services, APIs, CRUD operations over a database information, FTP to transfer files or SMTP to send or receive emails.

In addition to the features described above, this ESB has a very intuitive Web administration interface, which represents extra value for the end user, as it eases the use of the tool by him. The ESB has also a set of Admin Web Services that allow its control through iPortalDoc, this being the most important feature in terms of integrating the tool into the platform.

The solution for integrating the ESB into iPortalDoc consists of creating an account system for accessing the ESB through the backend of iPortalDoc using the Admin Web Services. At the frontend level, an interface was developed to allow the control of iPortalDoc users' access to the ESB by the System Administrator and software blocks were developed to connect the frontend and the backend. Furthermore, the ESB was added to the iPortalDoc apps menu, allowing direct access to the tool by authorized users and an automatic login method to the ESB was implemented.

After implementing the integration solution, a case study was created where the ESB, through user-defined services, collects information from a database. This information will be used in different states of a workflow to carry out the process of "creating or updating an entity in iPortalDoc", thus demonstrating the functionality of the chosen tool.

Agradecimentos

Em primeiro lugar gostaria de agradecer aos meus pais, em especial à minha mãe por sempre acreditar que eu seria capaz de realizar concluir com sucesso este objetivo que eu tenho e por apesar das dificuldades que isso acatou, ter sempre feito os sacrificios que me permitem hoje dizer que consegui. Gostaria também de agradecer aos bons amigos que esta etapa da minha vida me proporcionou, amigos esses que levo para vida e que tornaram esta jornada na FEUP muito mais cativante e que me inpiraram e inspiram a ser cada dia um pouco melhor.

Um grande obrigado aos meus melhores amigos Gonçalo e Yukari por estarem lá nos bons momentos e especialmente por me apoiarem nos maus e por me relembrarem do porquê de me levantar todos os dias. Um agradecimento a todos os meus outros amigos próximos e de longa data por permanecerem ao meu lado durante esta aventura.

Gostaria também de agradecer à Prof.ª Ana Aguiar, que forneceu ideias e conselhos que enriqueceram esta dissertação, sou muito grato pela sua orientação. Por fim gostaria de agradecer à Equipa IDI iPortalDoc da IPBRICK e em especial à Engenheira Telma Salgueiro, pela forma como me receberam de braços abertos na empresa e por estarem sempre dispostos a ajudar nos mais variados problemas que surgiram durante o projeto.

Francisco Oliveira

“It is hard to fail, but it is worse never to have tried to succeed."

Theodore Roosevelt

Contents

List of Figures

List of Tables

Abreviaturas e Símbolos

API	Application Programming Interface
ESB	Enterprise Service Bus
GUI	Graphical User Interface
IoT	Internet of Things
IP	Internet Protocol
JSON	JavaScript Object Notation
OSGi	Open Service Gateway Initiative
REST	Representational State Transfer
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
TI	Tecnologia da Informação
XML	Extensible Markup Language

Chapter 1

Introdução

1.1 Contexto

Nos dias de hoje, as arquitecturas orientadas a serviços são aceites como um standard nos processos de integração de sistemas nas organizações. Neste padrão de arquitetura de software as funcionalidades implementadas nas aplicações são disponibilizadas na forma de serviços, acessíveis normalmente através de web services. É baseada nos princípios da computação distribuída pois é constituida por aplicações que se encontram espacialmente distantes umas das outras ligadas por uma rede de comunicação e utiliza o paradigma *request/reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas dos serviços.

A utilização desta arquitetura apresenta determinadas vantagens. A confiabilidade devido à facilidade de teste e debugging de serviços pequenos e independentes. a independência de localização, a independência de plataforma, a reutilização de componentes de software e a agilidade no que toca ao desenvolvimento de novas aplicações para integrar o sistema [1].

No entanto existem alguns problemas associados ao uso desta arquitetura, no que diz respeito à rede de comunicação. Se a comunicação entre as diversas aplicações que integram o sistema for realizada ponto-a-ponto, em que todas as aplicações precisem de se ligar entre si, ou a grande maioria, torna-se bastante complexa e trabalhosa, a integração de novas aplicações. Cada aplicação nova teria de ser ligada ponto-a-ponto a todas as outras do sistema, sendo que para cada ligação teriam de ser respeitadas as especificidades que permitem a comunicação e a troca de informação entre as 2 aplicações. Isto cria problemas, como o aumento da complexidade de manutenção e a dificuldade em escalar o sistema.

Para resolver tais problemas as empresas de TI decidiram dotar os seus sistemas orientados a serviços com uma ferramenta de Middleware conhecida como Enterprise Service Bus (ESB), ferramenta essa que surgiu em 2002 [2] e a qual tem tido um aumento considerável de adesão.

O ESB funciona como um *broker* na comunicação entre aplicações e promove uma fácil manutenção e integração de novas aplicações. Para além disso também realiza o roteamento,

transformação de mensagens e gestão de transações, aumentando a segurança e fiabilidade do sistema.

Desde 2010 a investigação no tema tem aumentado consideravelmente [2], tanto que grandes empresas de TI como IBM, Oracle e Microsoft lançaram as suas próprias versões deste tipo de ferramenta [3].

A IPBRICK SA é uma empresa fabricante e distribuidora especializada em Soluções de Comunicação Corporativas, que tem como principal produto uma plataforma de Comunicações para Empresas, o sistema operativo IPBRICK OS, baseado em Linux Debian e que contém uma solução integrada para a Gestão de Documentos e Processos baseada em fluxos de trabalho, o iPortalDoc. O iPortalDoc permite modelar e implementar processos de todas as áreas das organizações, dando acesso a um histórico dos intervenientes e intervenções ocorridas nos processos, assim como a documentos e emails associados, constituindo assim uma ferramenta que facilita a pesquisa e evita perdas de tempo e de informação.

O iPortalDoc encarrega-se também da comunicação entre as diferentes aplicações que intervêm nos processos. Num cenário em que seja necessária uma integração multi aplicação em que as aplicações apresentam um caráter heteorogéneo entre si, o estabelecimento das comunicação entre elas torna-se complexo, devido às especificações requeridas por cada aplicação no que toca à formatação de mensagens, a métodos Web-Service utilizados para interação, entre outras variáveis que intervêm na comunicação.

Através do uso de um Enterprise Service Bus (ESB) é possível universalizar essa tarefa de comunicação entre aplicações. Esta dissertação vai incidir na escolha e integração de funcionalidades de um ESB open-source no iPortalDoc e na criação de uma interface GUI e APIs para interação com o ESB.

1.2 Motivação

Na atual abordagem usada para a comunicação entre as aplicações e os processos no iPortalDoc é necessário atender às dependencias especificas de cada interação aplicação-cliente e no caso de ser necessário escalar o sistema e adicionar novas ligações, essas ligações terão de ser adicionadas de forma manual e atendendo às necessidades da aplicação, tudo isto antes de ser possivel estabelecer a interação. Ao nivel de uma organização com inumeras aplicações que precisam de interagir com processos, estabelecer novas interações constitui um trabalho complexo, exaustivo e que toma imenso tempo.

Um ESB é uma ferramenta promissora no que toca à integração de aplicações em sistemas distribuidos com aplicações de caráter heterogéneo. Esta ferramenta de middleware permite que

as aplicações troquem informação e oferece suporte de infraestrutura para a transformação de mensagens e de dados, assim como roteamento inteligente.

A ideia do ESB é não depender de um padrão de integração que seja dificil de gerir com o passar do tempo, mas sim de uma abordagem que melhora a agilidade da infraestrutura de comunicações e a interoperabilidade entre aplicações permitindo uma gestão mais fácil e o escalamento do sistema com mais aplicações e de caráter mais variado com uma redução acentuada no tempo despendido em tais tarefas. Um ESB é ideal para a implementação de uma arquitetura orientada a serviços (SOA), pois oferece um mecanismo que integra todas as aplicações necessárias às organizações como a Contabilidade e Finanças (AF), Gestão de Relações com o Cliente (CRM), Marketing e Vendas (SM), Gestão de Recursos Humanos (HRM), entre outros. Tudo isso sem perder fiabilidade, performance e segurança.

A escolha de um produto open-source é uma mais valia em termos de custo, facilidade de instalação e de integração. A criação de uma interface gráfica (GUI) simplifica ainda mais a gestão do ESB através do iPortalDoc.

A importância do tema é ainda outro fator motivacional, economicamente falando, pois a implementação desta tecnologia permite acrescentar valor económico ao software disponibilizado pela IPBRICK, na medida em que o mesmo melhora o seu desempenho.

1.3 Objetivos

O objetivo global do projeto é permitir o acesso dos utilizadores ao ESB a partir do iPortalDoc e apresentar uma interface de configuração intuitiva que permite que os mesmos consigam definir serviços que possam ser invocados a partir do iPortalDoc e usados nos seus Workflows de maneira a suprir as necessidades dos utilizadores.

No entanto o projeto pode ser dividido em objetivos mais pequenos:

- Gerir os utilizadores do ESB a partir do iPortalDoc;
- Permitir que os utilizadores possam definir e invocar diferentes tipos de serviços, como APIs e Data Services;
- Permitir que os utilizadores possam aceder diretamente à interface do ESB através do menu de Aplicações do iPortalDoc;
- Permitir que os utilizadores possam aceder à interface do ESB sem ter de efetuar autenticação manual.

Chapter 2

Estado da Arte

Para compreender melhor o problema e montar um plano de resolução é necessário conhecer o leque de soluções diferentes que existem e podem solucionar o mesmo, as vantagens e desvantagens de cada um, assim como o contexto em que o problema se encontra inserido.

Neste capítulo é descrito o estado atual da tecnologia relacionada com o tema tendo como referências livros, artigos científicos provinientes de fontes académicas, assim como foruns online.

De modo a resolver o problema é necessário entender o contexto onde ele se encaixa. Sendo que o problema afeta um sistema com Arquitetura Orientada a Serviços como o iPortalDoc, esse será um ponto de partida para a resolução do mesmo.

2.1 Arquitetura Orientada a Serviços

2.1.1 Conceito

A era moderna da computação é projetada em torno de sistemas orientados a serviços que integram vários serviços para fornecer serviços sofisticados por meio de interfaces bem conhecidas. Isso pode ser feito com eficiência usando recursos da Arquitetura Orientada a Serviços (SOA) [4].

A arquitetura orientada a serviços (SOA) é um paradigma que integra componentes de software distribuídos, mantidos separadamente e implantados e organiza-os como serviços interoperáveis que podem facilmente ser reutilizados e compartilhados entre aplicações e empresas [5].

Esses serviços ficam disponíveis numa rede para serem invocados por meio de interfaces definidas e combinados em soluções para problemas de negócio [6]. SOA é habilitada por tecnologias e padrões que facilitam a comunicação e cooperação dos componentes em uma rede,

especialmente numa rede IP [7]. Através desse paradigma é possivel vários utilizadores colaborarem entre si, utilizando dados, arquivos e ficheiros que se encontram em locais diferentes do globo, de forma a produzirem os resultados que desejam.

De um modo essencial, SOA é uma coleção de serviços que comunicam entre si. A comunicação pode envolver apenas uma troca de dados ou pode envolver vários serviços a coordenar alguma atividade[8]. Baseia-se na Orientação a Serviços que é uma metodologia que faz uso dos serviços e dos resultados por eles gerados[1].

Uma tendência dominante passa por migrar de um sistema legado para um sistema baseado em SOA para modernizar o sistema de software das organizações. Muitos estudos destacam os benefícios do emprego de SOA no desenvolvimento de novas tecnologias , como a Internet das Coisas (IoT) e CloudComputing e microsserviços [9].

Em relação aos setores industriais, até agora, SOA provou ser um paradigma chave em vários setores, como bancário, saúde, transporte e de tecnologia [9]. SOA insere-se num processo de reorganização dos departamentos de tecnologia da informação das organizações, permitindo um melhor relacionamento entre as áreas que dão suporte tecnológico à empresa e as áreas responsáveis pelo negócio propriamente dito [5].

Fazer uso de SOA para resolver um problema de negócios requer a existência prévia de pelo menos uma solução para o problema de negócios. SOA fornece um meio de usar mecanismos padronizados para descoberta, invocação e comunicação para acessar recursos subjacentes. Se o problema de negócios não pode ser resolvido sem SOA, então SOA por si só não pode fornecer uma solução. Esses serviços devem existir antes que SOA possa fornecer um meio de aceder e interagir com eles. O único problema para o qual SOA fornece uma solução é o problema da facilidade de interação eficiente entre consumidores e fornecedores de soluções para problemas de negócios [6].

2.1.2 Características

Numa Arquitetura Orientada a Serviços a funcionalidade de negócios é disponibilizada na forma de serviços que podem funcionar de maneira granular. Um serviço é uma unidade discreta de funcionalidade que pode ser acessada remotamente e controlada e atualizada de forma independente [7].

A arquitetura orientada a serviços é caraterizada por possuir:

- **Fraco acoplamento**: Devido à independência entre serviços que dela fazem parte;

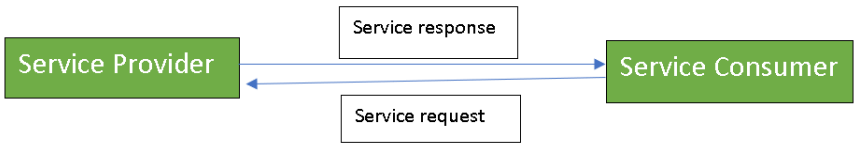

Figure 2.1: Consumidor e fornecedor de serviços e correspondentes interações.

- **Orquestração de serviço**: Devido à capacidade dos serviços poderem agregar dados de outros serviços e criarem fluxos de trabalho para atender à solicitação de um determinado consumidor de serviço. Esta carateristica contribui para a composibilidade deste tipo de Arquitetura;
- **Agilidade de integração** e **capacidade de reutilização** de serviços: Por se tratar de uma arquitetura modular e assim permitindo reduzir custos e tempo de desenvolvimentos de novos serviços;
- **Transparência** e **abstração**: Encapsulando várias aplicações e fontes de dados, ocultando a sua lógica;
- **Autonomia**: Encapsulando várias aplicações e fontes de dados, ocultando a sua lógica;
- **Registo/Catálogo**: Que permite localizar cada serviço implantado no sistema e fornece informação sobre os parâmetros de invocação do mesmo;

Existem duas funções principais na Arquitetura Orientada a Serviços:

- **Fornecedor de serviços**: o fornecedor de serviços é o gerente do serviço e a organização que disponibiliza um ou mais serviços para uso de terceiros. Para anunciar serviços, o fornecedor pode publicá-los em um registo, juntamente com um contrato de serviço que especifica a natureza do serviço, como usá-lo, os requisitos para o serviço e as taxas cobradas.
- **Consumidor de serviço**: O consumidor de serviço pode localizar os metadados de serviço no registo e desenvolver os componentes do cliente necessários para ligar e usar o serviço [10].

Um sistema construído no paradigma SOA deve fornecer visibilidade das necessidades e recursos; deve incluir um meio de interação entre consumidores e fornecedores.

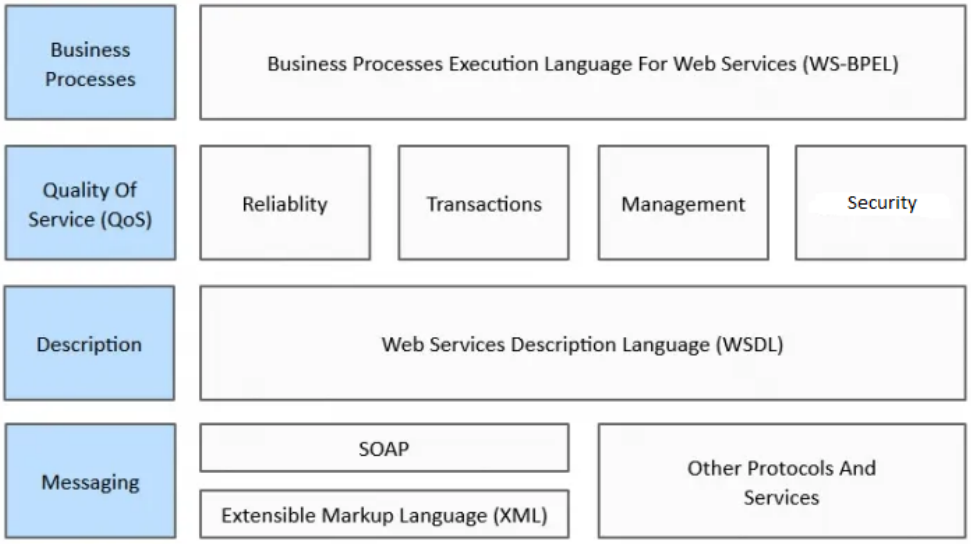

Figure 2.2: Pilha protocolar de SOA. [1]

Visibilidade refere-se à capacidade dos consumidores com necessidades e fornecedores de recursos que atendem às necessidades de se encontrarem e se prepararem para interagir. Isto inclui o estabelecimento de uma consciência de pelo menos um participante do outro , uma vontade por parte dos participantes de interagir e a acessibilidade necessária para trocar informações [6].

Um meio comum de obter conscientização e **detectabilidade** de serviços é através do **contrato de serviço padronizado** que é um registo de serviço que lista por meio de um ou mais documentos descrições dos serviços disponíveis [10]. Para que um consumidor avalie se um serviço atende às suas necessidades e, eventualmente, interaja com o serviço, a descrição deve incluir as funções desempenhadas; formatos de entrada e saída; restrições e políticas; protocolos, semântica, mecanismos de acesso e uso, que descrevem como um serviço pode ser usado [6].

Uma vez que a visibilidade é estabelecida, a interação pode prosseguir. Na maioria das vezes, isso é realizado pela troca de mensagens por meio de interfaces de serviço bem definidas [6]. Atualmente, a implementação mais comum de SOA são os Web Services, que conferem uma **integração de linguagem neutra**, pois constituem um mecanismo comum para a invocação dos serviços, independentemente da linguagem de desenvolvimento usada [11].

A maioria desses padrões usa o Extensible Markup Language (XML) como sintaxe básica e fornece uma ligação ao Hypertext Transfer Protocol (HTTP) para o protocolo de mensagem subjacente. As principais especificações de mensagem de Web Services são SOAP para definir

[1] Imagem retirada de https://www.w3schools.in/service-oriented-architecture

o envelope de mensagem de Web Services e Web Services Description Language (WSDL) para descrever elementos da interface de Web Services [6].

Na figura 2.2 está representada a pilha protocolar de SOA mostrando os protocolos que atuam em cada camada. Esses componentes são escritos em BPEL (Business Process Execution Languages), Java, C , XML etc [11].

Na figura 2.3 é apresentada uma hierarquia dos elementos constituintes de uma Arquitetura Orientada a Serviços, do mais abstrato ao mais detalhado.

Esta padrão de software é constituido essencialmente por 4 elementos: Application frontend, serviço, repositório de serviços e Service Bus que será posteriormente abordado na secção 2.2.

Um serviço pode ser dividido em 3 componentes.

- A interface define como um fornecedor de serviços solicitará o input de um consumidor de serviço, de forma a poder gerar a resposta desejada.
- O contrato define como o fornecedor de serviço e o consumidor de serviço devem interagir.
- A implementação é o código que define a execução do serviço.

Como a interface se encontra separada da sua implementação, o fornecedor de serviço pode executar um pedido sem o consumidor saber como tudo se processa.

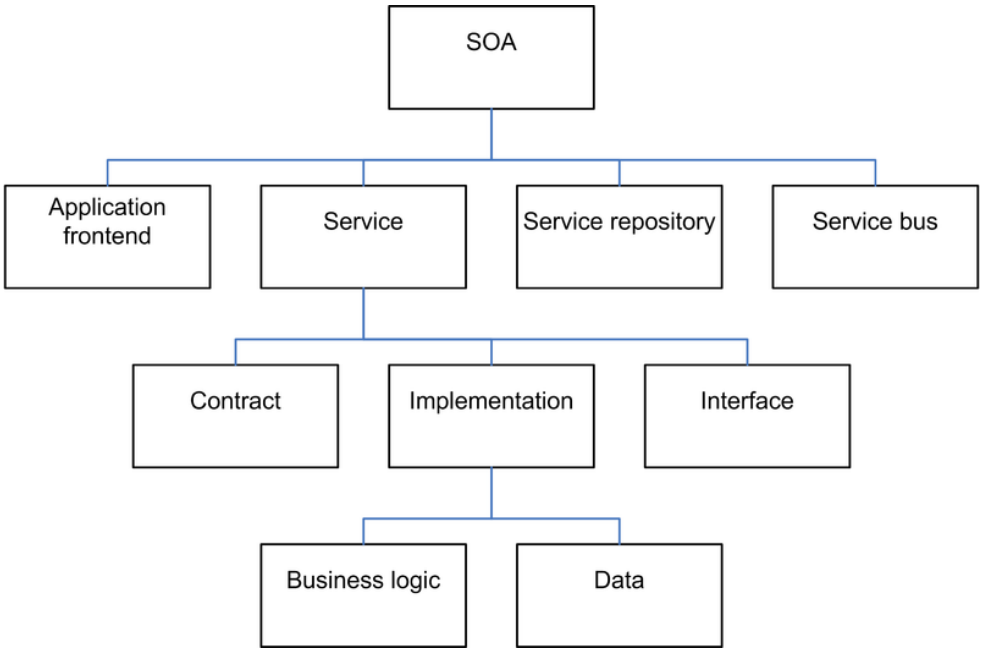

Figure 2.3: Hierarquia de elementos constituintes de SOA.[2]

2.1.3 Vantagens e Limitações

Nesta secção vão ser discutidas as vantagens da adesão a uma Arquitetura Orientada a Serviços e as limitações que esse modelo de Arquitetura apresenta e verificar o trade-off obtido.

A utilização de uma arquitetura SOA permite obter vantagens como [10, 12]:

- **Confiabilidade**: Com serviços pequenos e independentes no SOA, fica mais fácil testar e fazer debugging aos aplicativos do que a grandes blocos de código.
- **Independência de localização**: Os serviços são localizados por meio do registo de serviço e podem ser acessados por meio do Uniform Resource Locator (URL), portanto, eles podem alterar sua localização ao longo do tempo sem interromper a experiência do consumidor no sistema.
- **Escalabilidade**: Como SOA permite que os serviços sejam executados em várias plataformas, linguagens de programação e serviços, a compatibilidade com diferentes tipos de aplicações aumenta, permitindo que o sistema posso sofrer um maior crescimento.
- **Independência de plataforma**: A Arquitetura Orientada a Serviços permite o desenvolvimento de uma aplicação integrando diferentes serviços escolhidos em diferentes fontes que a tornam independente da plataforma.

[2]Imagem retirada de https://commons.wikimedia.org/wiki/File:SOA_Elements.png

- **Fracamente acoplado**: SOA incentiva fortemente o desenvolvimento de serviços independentes para aumentar a eficiência do aplicativo de software.
- **Reutilização**: Um aplicativo baseado em SOA é desenvolvido acumulando serviços de funcionalidade pequenos e independentes. Ele permite a reutilização dos serviços em vários aplicativos de forma independente, sem interagir com outros serviços.
- **Agilidade**: A capacidade de reunir aplicativos de componentes ou serviços reutilizáveis, em vez de reescrever e reintegrar cada novo projeto de desenvolvimento, ajuda os desenvolvedores a projetar um aplicativo rapidamente em resposta aos novos requisitos de negócios, o que, por sua vez, aumenta a agilidade de SOA.
- **Facilidade de manutenção**: Como a arquitetura orientada a serviços é uma unidade ou entidade independente, a manutenção ou atualizações do aplicativo tornam-se muito mais fáceis sem a necessidade de se preocupar com outros serviços.
- **Disponibilidade**: As instalações SOA estão facilmente disponíveis para qualquer pessoa, mediante solicitação.

Numa Arquitetura Orientada a Serviços os serviços são interconectados por uma conexão ponto-a-ponto. A complexidade surge quando existe a necessidade de interconectar um grande número de serviços dentro de uma organização[13].

Para além disso, nos dias de hoje existe uma grande demanda por integração de serviços de diferentes domínios de aplicações e partilha de ativos de TI dentro e fora de uma organização[3].

Essa demanda gera novas necessidades como:

- A expansão dos negócios de uma organização exige um ambiente livre de restrições para os seus negócios;
- A mudança na demando do mercado precisa de uma arquitetura de sistema flexivel, escalável e compatível;
- A utilização completa dos ativos de TI da organização;
- A utilização eficiente dos serviços do sistema e a sua manutenção, reconfiguração e reutilização;

É comum que uma empresa execute dezenas ou centenas de aplicações, que podem ser construidas de forma personalizada ou adquiridas de terceiros. Então cada serviço requer interfaces

de comunicação com outros serviços. Por vezes é muito complexo e demorado para estabelecer tais conexões (múltiplas conexões ponto-a-ponto) assim como configurá-las[13].

Outro problema é a **interoperabilidade** entre diversos serviços heterogéneos. A integração de serviços significa a fusão de vários serviços numa plataforma comum onde pode trocar dados e informações por meio de protocolos[4]. É necessária a **conversão de protocolo** entre serviços de comunicação que suportam diferentes protocolos subjacentes para que se possa efetuar essa troca de dados. Uma Arquitetura Orientada a Serviços não dispõe dessa capacidade por base.

Tais lacunas podem ser preenchidas com a ajuda de um ESB com sua variedade de conectores que formam uma interface bem definida para comunicação com serviços de diferentes domínios de aplicações[4].

Outros pontos negativos na adoção deste paradigma incluem [10]:

- **Overhead elevado**: uma validação dos parâmetros de entrada dos serviços é feita sempre que os serviços interagem, o que diminui o desempenho à medida que aumenta a carga.
- **Alto investimento**: Um grande investimento inicial é necessário para SOA.
- **Gestão de serviço complexa**: quando os serviços interagem, eles trocam mensagens. o número de mensagens pode chegar a milhões. É uma tarefa complicada lidar com um grande número de mensagens.

O uso de SOA é motivado pela grande quantidade de beneficios apresentados em relação aos pontos pontos negativos relativos à sua utilização.

2.2 Enterprise Service Bus

2.2.1 Conceito

Atualmente, os serviços devem ser robustos, ágeis, acessíveis e disponíveis para seus clientes. As empresas procurar integrar serviços de diferentes domínios de aplicação e também a interoperabilidade entre vários sistemas heterogéneos dentro e fora de uma organização [3].

Para entrega segura e garantida de serviços, cada grande organização está mudando seu modelo de entrega de serviço para Enterprise Service Bus (ESB).

Um Enterprise Service Bus (ESB) é uma plataforma integrada de middleware que fornece serviços fundamentais de interação e comunicação para aplicações de software complexas por meio de um mecanismo de mensagens baseado em padrões e orientado a eventos numa Arquitetura Orientada a Serviços. A sua elasticidade permite, que seja compatível com uma grande variedade de aplicações. Fornece uma abstração de camadas na implementação de um sistema empresarial de mensagens, que permite a integração, reutilização e compatibilidade de serviços. ESBs devem ser baseados em padrões flexíveis, suportando vários meios de transporte [2, 14].

As aplicações estão indiretamente ligadas através do ESB em vez de serem ligadas diretamente umas às outras. O ESB é responsável por toda a lógica embutida necessária para fazer as estruturas interagirem.

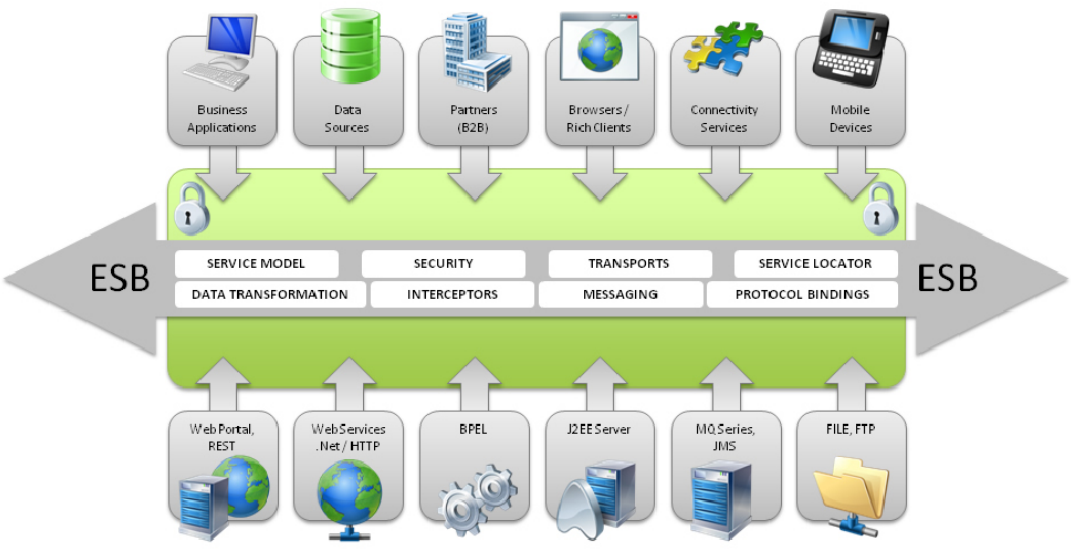

Figure 2.4: Arquitetura de um ESB.[3]

[3] Imagem retirada de https://www.hcltech.com/blogs/everything-you-need-know-about-enterprise-service-bus-esb

Devido ao comportamento distribuído do ESB, ele não tem limites de região, ou seja, os serviços podem ser acessados de qualquer lugar. ESB também suporta vários modelos de dados e informações de transformação para aumentar a agilidade e acessibilidade dos sistemas SOA [3].

Como agente mediador oferece suporte à interconectividade para vários serviços. Essa comunicação é geralmente baseada na troca de mensagens entre as duas partes. Isto permite que o ESB receba mensagens de clientes, processe-as de acordo e encaminhe-as para o ponto de serviço designado. O ESB fornece várias interfaces para diferentes canais de suporte aos serviços [3].

A relação entre consumidores de serviços, fornecedores e ESB é ilustrada abaixo na Fig. 2.5.

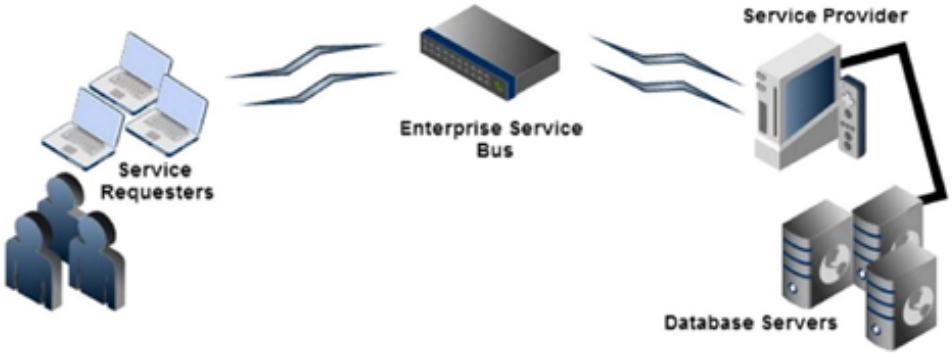

Figure 2.5: Relação entre consumidores, fornecedores e ESB.[4]

O ESB assume a responsabilidade de garantir que os dados enviados pelos consumidores de serviço correspondam aos requisitos de formato dos prestadores de serviço [15].

Um ESB é uma estrutura ideal para a implementação de Arquitetura Orientada a Serviços (SOA) porque oferece um mecanismo padrão que integra todas as instalações necessárias às soluções de informação organizacional como [2]:

- Gestão da cadeia de abastecimento (SCM);
- Contabilidade e Finanças (AF);
- Gestão da relação com o cliente (CRM);
- Marketing e Vendas (SM);
- Gestão de recursos humanos (HRM).

[4]Imagem retirada de https://www.semanticscholar.org/paper/The-Performance-Metric-for-Enterprise-Service-Bus-Bhadoria-Chaudhari/b7c7a84c841f48304e0d148a97a2e271d9e87d77/figure/0

2.2.2 Características

As principais funcionalidades oferecidas por um Enterprise Service Bus incluem: impulsionamento de mensagens, transformação de messagem, orquestração de serviço, gestão de transações, roteamento e segurança [2].

- **Aprimoramento de mensagens**: melhora, transforma ou remove informação contida numa mensagem para faze-la compativel com o fornecedor do serviço.
- **Transformação de mensagens**: Para transformar a estrutura e a carga da demanda do utilizador num método conveniente para o fornecedor do serviço. Um ESB permite transformar uma mensagem recebida em várias formas e estruturas.
- **Integração de aplicações heterogéneas**: Permite que aplicações construidas a partir de diferentes Linguagens de Programação e em diferentes plataformas e que correm em Sistemas Operativos distintos possam ser integrados no mesmo sistema e que comuniquem de forma eficaz entre si.
- **Orquestração de serviço**: O ESB age como uma camada de middleware (broker) que inspeciona os serviços e sincroniza o desempenho de numerosas aplicações.
- **Gestão de transações**: Lida com a demanda de serviços de negócio como uma unidade de trabalho distinta.
- **Roteamento**: Um ESB tem a capacidade de transmitir um pedido de um utilizador para um fornecedor de mensagem especifico estabelecido em padrões de roteamento flexíveis.
- **Segurança**: Oferece a oportunidade de proteger serviços de login inautorizado, através de validação, autorização, avaliação e controlo.

O ESB funciona como um corretor para todas as mensagens trocada entre serviços, ele pode receber mensagens de um serviço, empacotá-las e encaminhá-las para outro serviço, e/ou retornar uma resposta para o remetente[13]. Durante a comunicação, os solicitantes do serviço não precisam saber a identidade do cliente; é responsabilidade do ESB gerir essa identidade para cada solicitante de serviço[16].

O ESB é designado para lidar com vários protocolos e partilhar dados em diferentes formatos, implementando desta forma um sistema de comunicação entre softwares heterogéneos que interagem entre si numa Arquitetura Orientada a Serviços, removendo as dependências entre aplicações[1].

Ele oferece uma variedade de conectores que oferecem fluxos de mensagens para conectar e interagir com outros serviços, como por exemplo, Salesforce, Twitter, Ebay, Gmail. O cliente também pode criar e personalizar os conectores de acordo com suas necessidades[4].

Para além de ser o principal ponto de mediação de mensagens, o ESB também providencia segurança ao mais alto nível e também tem a habilidade de registar eventos. Permite também a fácil integração de vários serviços sem a necessidade do utilizador escrever/alterar o código fonte do sistema[13].

Providencia suporte a vários aspectos de SOA, incluindo compartilhamento de recursos, manuseio de mensagens e controlo de versão de serviço. Com o ESB, é possível projetar, desenvolver, implantar e monitorar serviços em tempo de execução. Isso melhora o conceito de **reutilização** no ESB [3].

Os serviços de negócios devem ser controlados e monitorados para um melhor desempenho e podem ser entregues com o auxílio de uma pilha de serviços bem definida. A pilha de serviços é um conjunto de serviços na arquitetura do Sistema SOA que estende os recursos da mesma [3]. A pilha de serviços encontra-se representada na figura 2.6.

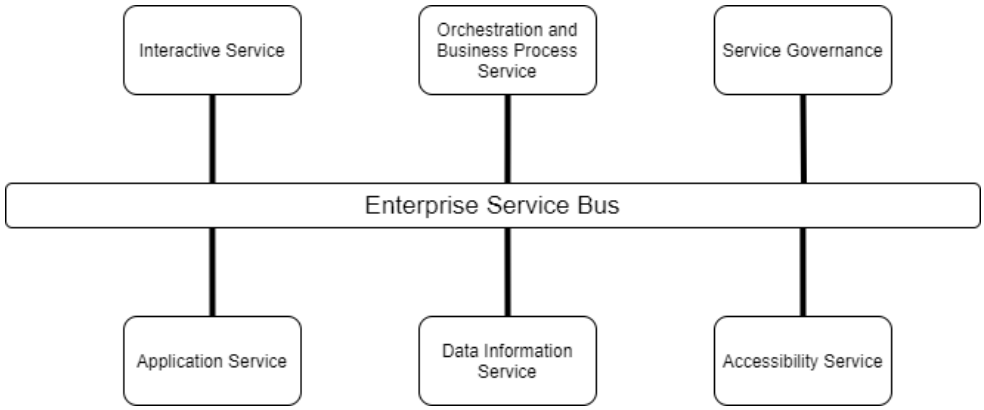

Figure 2.6: Pilha de serviços de ESB.

Esses serviços dentro da estrutura ESB, são nomeadamente [3]:

- **Serviços interativos**: tais serviços fornecem recursos essenciais para o fornecimento de funcionalidade e dados de TI.
- **Orquestração e Serviço de Processo de Negócios**: a orquestração de serviço é definida como coordenação entre vários serviços e descrita como uma única unidade agregada para facilitar a interação entre eles.
- **Serviço de Informação de Dados**: Esses serviços federam, replicam e transformam informações de diferentes fontes de dados, essenciais para a transformação de mensagens.
- **Serviço de aplicativo**: permite a implementação da lógica funcional principal do negócio. Esses serviços são realmente executados e chamados por solicitantes de serviço.
- **Serviço de Acessibilidade**: Oferece suporte na disponibilização de serviços aos solicitantes. O ESB atua como um mediador ao permitir tais serviços que são entregues via SOA. Também ajuda a incorporar serviço com seus dados associados.
- **Governança de serviço**: fornece monitoramento do conteúdo associado ao serviço. Também ajuda no cálculo de custos, planejamento e configuração de serviços de aluguel sob demanda. Garante a acessibilidade para determinado serviço.

2.2.3 Vantagens

Muitas da vantagens associadas à Arquitetura Orientada a Serviço que foram referidas na secção 2.1.3, estão associadas ao uso de um ESB no sistema de comunicação sendo essas vantagens adquiridas por transição.

Algumas das vantagens subjacentes a um Enterprise Service Bus passam por [2, 3]:

- Uma **integração** que promove a **flexibilidade** com os sistemas existentes e que pode ser realizada de maneira simples.
- **Fiabilidade**, pois graças ao controlo de transações e roteamente inteligente, as mensagens chegam garantidamente ao destino.
- A **segurança**, pois como foi visto, os ESB geralmente possuem carateristicas como validação, autorização e controlo.
- A **escalabilidade**, pois ao fornecer uma maneira padronizada e automatizada para as aplicações trocarem informações, o ESB torna a expansão dos serviços existentes mais fácil.
- O suporte de vários modelos de dados e informações de transformação aumenta a **acessibilidade** e **agilidade** dos sistemas.
- A **reutilização** de recursos, que evita que os desenvolvedores de processos de negócios tenham que recodificar manualmente os serviços sempre que uma integração ocorrer.
- A **interoperabilidade** que permite a troca de informação entre aplicações heterogéneas dentro do sistema.
- Melhor capacidade de gestão por meio de um **único ponto de controlo**.
- Melhor **desempenho** e redes otimizadas.

Usar um ESB numa Arquitetura Orientada a Serviços significa que as aplicações que a formam se comunicarão entre si através do barramento. Isso tem a vantagem de reduzir o número de conexões ponto a ponto necessárias para que as aplicações se comuniquem entre si. A metodologia de conexão ponto a ponto é extremamente tediosa e deve ser substituída por uma solução integrada de orquestração de serviços numa plataforma comum e compartilhada, o ESB. Isso torna a análise de alterações e manutenção de software mais simples e clara[1].

Com o uso do ESB existe agora uma maneira de incorporar Web Services numa arquitetura para integração de aplicações e serviços que abrange uma empresa estendida em grande escala.

Dessa forma, todos os Web Services e aplicações podem enviar ou receber dados num formato de mensagem suportado e cada nó (onde as aplicações são implantadas) será conectado através de uma interface para esse ponto de acesso exclusivo, o ESB[13].

A criação e utilização de conectores facilita a mediação entre vários serviços, o que resolve a questão da interoperabilidade[4].

Devido a todas as vantagens apresentadas, as organizações decidem adotar esta ferramenta de modo a atingir os seguintes objetivos[3]:

- Diminuir o custo da integração de aplicações
- Acelerar as implementações de TI
- Reduzir a complexidade de TI

2.2.4 ESB Comercial vs ESB Open-Source

Os Enterprise Service Buses podem ser divididos em 2 categorias:

- ESBs Comerciais
- ESBs Open-Source

O software de código aberto (Open-Source) e o software proprietário (Comercial) não são uma revolução na indústria, já cá estão há algum tempo. Ambas as soluções oferecem características semelhantes, o que torna a a escolha entre uma e outra uma decisão crítica.

A escolha do ESB, deve cumprir os requisitos do sistema, de modo a ajudar na integração das aplicações da organização. Portanto devem ser seguidos alguns critérios na escolha de qual ferramenta uma organização deve optar. De seguida serão apresentados individualmente os principais critérios que geralmente são tomados em consideração na escolha entre um ESB Comercial e um Open-Source[17].

2.2.4.1 Usabilidade

A usabilidade diz respeito à facilidade com que se pode dominar a ferramenta, o quão complicada é a sua instalação, quantas ferramentas adicionais são necessárias para trabalhar com ela e se o ambiente de desenvolvimento é intuitivo ou não.

Em geral a instalação de ESBs comerciais é bastante complexa e é necessário o suporte de consultores especializados para que seja possível a instalação e uso correto da ferramenta, para além de serem necessárias algumas ferramentas adicionais para que o seu funcionamento seja pleno.

No caso de ESBs de código aberto, a instalação costuma ser mais simples e intuitiva e alguns minutos após a instalação já será possível utilizar a ferramenta. Este tipo de ESBs costumam ser ferramentas unificadas, o que significa que contêm intrinsecamente todas as funcionalidades necessárias para o seu funcionamento correto.

2.2.4.2 Administração e Monitoramento

Estas caraterísticas definem de que forma o ESB pode ser administrado, se existe uma Interface Gráfica que possibilita a fácil gestão das funcionalidades, se a administração é feita por uma consola ou por outro mecanismo.

No caso geral, os ESBs comerciais possuem ferramentas de administração e monitoramento mais poderosas do que os ESBs Open-Source e essa administração é feita por Interface gráfica na maioria dos casos o que facilita a vida ao utilizador.

No caso dos ESBs Open-Source, existem muitos casos onde essa administração é feita através de consolas em vez de Interfaces Gráficas, o que requer que o utilizador tenha de ter conhecimento de como trabalhar com a linha de comandos. Para além disso muitos ESBs desta categoria não trazem um sistema de administração e monitoramento de base, sendo necessário instalar software adicional para esse propósito.

2.2.4.3 Comunidade

A comunidade é importante para fornecer ajuda no uso da ferramenta assim como na resolução de problemas que possam aparecer ao longo do processo.

Os ESBs comerciais não costumam ter uma comunidade dedicada ao suporte, mas dispõem de um serviço de suporte pago. Já no caso dos ESBs open-source, regra geral, têm uma comunidade que ajuda na resolução dos problemas e que até contribui para o desenvolvimento de novas funcionalidades para o ESB.

2.2.4.4 Suporte

O suporte em ambas as categorias de ESB é pago, mas no caso do comercial é mais disponível (geralmente 24/7), mais especializado e com mais garantias de resolução dos problemas. Muitos ESBs Open-Source não possuem esse suporte especializado.

2.2.4.5 Funcionalidade

Os ESBs comerciais, geralmente apresentam mais funcionalidades de integração do que os Open-Source, assim como funcionalidades extra fora do contexto da integração.

2.2.4.6 Flexibilidade e Expansibilidade

A flexibilidade diz respeito à capacidade de alterar a ferramenta ESB para servir às necessidades e expansibilidade é a capacidade de adicionar novas funcionalidades. No caso dos ESBs comerciais essas mudanças têm de ser pedidas à empresa que vende os serviços do ESB e geralmente têm um custo elevado para serem atendidas ou o processo para a alteração é demorado.

No caso Open-Source, o código da ferramenta encontra-se disponível publicamente sendo possível qualquer pessoa fazer as alterações necessárias à Ferramenta e podendo até publicar a nova versão dependendo da licença sobre a qual o ESB esteja abrangido.

2.2.4.7 Conectores

Os conectores são responsáveis por criar uma interface entre o ESB e serviços de terceiros. Neste critério ambas as categorias de ESB disponibilizam um número semelhante de conectores e permitem também a criação de conectores personalizados.

2.2.4.8 Custo

O custo pode abranger o uso do produto, a manutenção, os produtos auxiliares e os conectores. Alguns destes itens são gratuitos em alguns ESBs Open-Source e no geral o custo dos itens todos é muito mais barato nesta categoria do que nos ESBs comercial, onde podem ser atingidos valores consideráveis.

2.2.4.9 Licenciamento

Os ESBs comerciais apresentam uma lista de preços mais complexa e uma licença mais restrita onde é permitido exclusivamente o uso da ferramenta.

Os Open-Source apresentam na generalidade uma licença mais permissiva, com um modelo de subscrição mais simples onde é possível serem efetuados upgrades e downgrades no serviço, para além da licença poder permitir a alteração e publicação de alterações ao software base.

Critério	**Open-Source**	**Comercial**
Usabilidade	x	
Administração e Monitoramento		x
Comunidade	x	
Suporte		x
Funcionalidade		x
Flexibilidade e Expansibilidade	x	
Conectores	x	
Custo	x	
Licenciamento	x	

Table 2.1: Comparação ESB Open-Source vs Comercial.

A tabela 2.1 apresenta a categoria que se destaca em cada critério. Com base na informação apresentada na tabela, concluimos que os ESBs comerciais em geral fornecem mais carateristicas e um suporte/manutenção mais poderosos, mas tem maior complexidade e custo. Por outro lado, ESBs open source oferecem grande flexibilidade, facilidade de uso, manutenção mais fácil e custo mais baixo [2]. As ferramentas open source oferecem um maior número de vantagens em relação às comerciais.

Com este projeto é pretendido fazer a integração de uma ferramenta simples de utilizar, flexivel e com baixa complexidade de manutenção. O custo no entanto representa o fator de maior importancia na escolha da categoria a utilizar, sendo portanto um ESB Open-Source a escolha acertada.

2.3 ESBs Open-Source

Esta secção será dedicada a apresentar alguns Enterprise Service Buses Open-Source que se encontram disponiveis no Mercado. Um destes poderá ser a ferramenta escolhida para integrar o iPortalDoc.

2.3.1 WSO2 ESB

WSO2 é um ESB Open-Source leve baseado em Java que providencia melhor compatibilidade e alto desempenho em termos de acessibilidade de serviço. Ele suporta quase todos os recursos de um ESB como governança, monitoramento e gerenciamento de serviços, roteamento de mensagens, transformação e transporte de dados, mediação de mensagens, orquestração de serviço, bem como hospedagem de APIs e serviços [18].

Figure 2.7: Funcionalidades do WSO2 ESB.[5]

Permite conectar e reutilizar ativos e sistemas de TI existentes implementados usando tecnologias heterogêneas, incluindo Web Services, Micro Services, HTTP, JMS, JDBC, entre outros [18].

O ESB pode ser implantado em qualquer lugar: no local, em qualquer infraestrutura de cloud, em clouds privadas e até mesmo usando sistemas de container. Contém recursos de monitoramento e análise para fornecer análises em tempo real.

[5]Imagem retirada de https://wso2.com/library/articles/2017/07/what-is-wso2-esb/

O ESB vem com ferramentas abrangentes de desenvolvimento e debugging para ajudar no rápido desenvolvimento de aplicações corporativas integrando e expondo serviços.

Contém documentação abrangente, tutoriais, material de treino gratuito para garantir a entrega confiável e bem-sucedida do desenvolvimento de soluções no WSO2 ESB.

2.3.2 Mule ESB

Mule, é um Enterprise Service Bus (ESB) Open-Source leve baseado em Java que permite aos desenvolvedores conectar aplicativos de forma rápida e fácil, permitindo que eles troquem dados. Ele permite a fácil integração de sistemas existentes, independentemente das diferentes tecnologias que os aplicativos usam, incluindo JMS, Web Services, JDBC, HTTP entre outros. O ESB pode ser implantado em qualquer lugar, integrar e orquestrar eventos em tempo real e conta com conectividade universal [19].

O Mule tem funcionalidades que incluem: Criação e hospedagem de serviços, Mediação de serviços, Roteamento de mensagens e Transformação de dados.

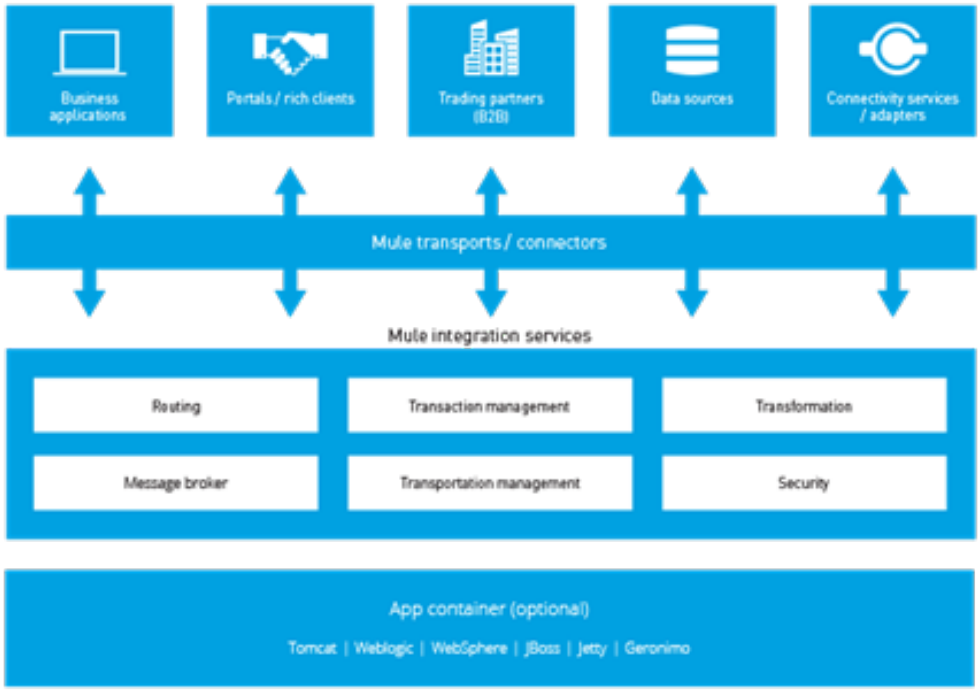

Figure 2.8: Arquitetura do Mule ESB.[6]

[6]Imagem retirada de https://www.mulesoft.com/pt/resources/esb/what-mule-esb

O ESB gere todas as interações entre aplicativos e componentes de forma transparente, independentemente de eles existirem na mesma máquina virtual ou na Internet, e independentemente do protocolo de transporte subjacente usado [19].

O Mule ESB permite uma reutilização significativa dos componentes. Os componentes não exigem código específico para serem executados no Mule, e nenhuma API programática é necessária [19].

As mensagens podem estar em qualquer formato, desde SOAP até arquivos de imagem binários. O Mule não impõe restrições, como mensagens XML ou contratos de serviço WSDL [19].

2.3.3 Apache Synapse ESB

Apache Synapse é um Enterprise Service Bus (ESB) Open-Source leve e de alto desempenho. Equipado com um mecanismo de mediação rápido e assíncrono, o Apache Synapse oferece suporte para XML, Web Services e REST. Além de XML e SOAP, o Apache Synapse oferece suporte a vários outros formatos de transformação de conteúdo, como texto simples, binário e JSON. A ampla variedade de adaptadores de transporte disponíveis para o Synapse permite que ele se comunique por meio de muitos protocolos de camada de aplicação e transporte [20].

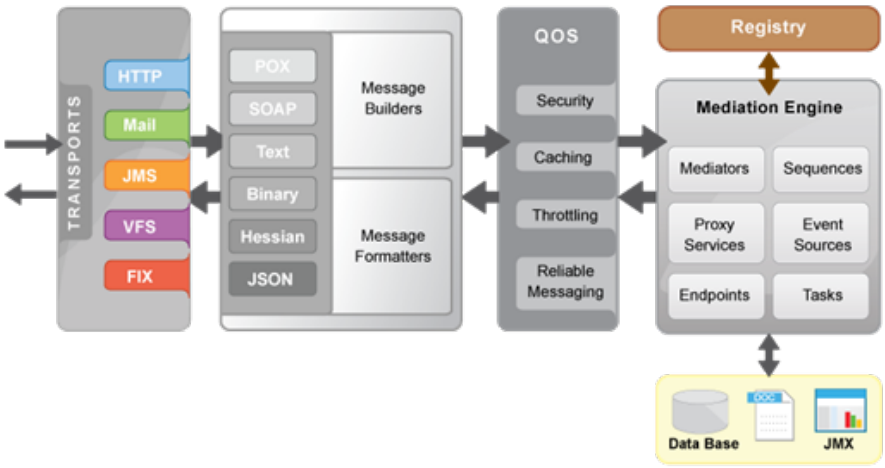

Figure 2.9: Arquitetura do Apache Synapse ESB.[7]

[7]Imagem retirada de https://synapse.apache.org/

O transporte HTTP sem bloqueio, o mecanismo de mediação multithread e o infoset XML de streaming combinam-se para garantir que o Synapse possa mediar volumes muito altos de mensagens por meio do barramento de serviço com o mínimo de atraso e uso de recursos [20].

O Synapse também vem com recursos abrangentes de registo, recolha de estatísticas e suporte de monitoramento JMX, que são cruciais em implementações de produção. Synapse é configurado usando uma linguagem de configuração baseada em XML [20].

2.3.4 Apache ServiceMix

O Apache ServiceMix é um contentor de integração flexível e open source que unifica e encapsula os recursos e as funcionalidades do Apache ActiveMQ, Camel, CXF e Karaf numa poderosa plataforma que pode ser usada para criar soluções de integração personalizadas. Ele fornece um ESB completo e pronto para empresas, desenvolvido exclusivamente com OSGi, que é uma estrutura modular de desenvolvimento e implantação de aplicações modulares e de bibliotecas em Java[21].

As suas principais características são[21] [22]:

- Mensagens confiáveis com **Apache ActiveMQ**, outro projeto Apache, é o intermediário de mensagens que é usado para trocar mensagens entre componentes. Além disso, o ActiveMQ também pode ser usado para criar um ESB totalmente distribuído;
- Mensagens, roteamento e padrões de integração empresarial (EIP) com o **Apache Camel**;
- Web Services Soap e REST com **Apache CXF**;
- O Kernel do ServiceMix é baseado em **Apache Karaf** (um runtime baseado em OSGi) e lida com os principais recursos fornecidos pelo ServiceMix, como implantação de aplicações, provisionamento de bibliotecas ou aplicativos, acesso remoto usando ssh, gerenciamento JMX, sendo portanto o contentor para as aplicações dentro do Apache ServiceMix.
- Possui uma **Consola Web**, que pode ser usado para iniciar o ServiceMix incorporado num aplicativo da web.
- Além da integração com serviços da Web por meio do CXF, o ServiceMix fornece muitos **componentes** que você pode usar para integrar com vários outros padrões e tecnologias.

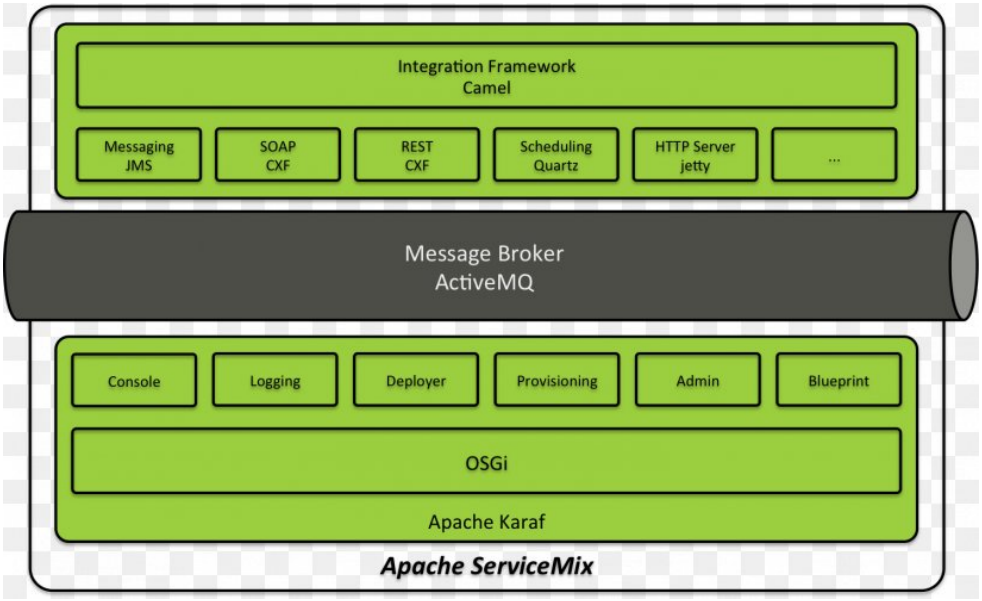

Figure 2.10: Arquitetura do Apache ServiceMix.[8]

Por meio de recursos adicionais instaláveis, o ServiceMix também oferece suporte a:

- Motor BPMN via Activiti
- Suporte completo a JPA via Apache OpenJPA
- Gestão de transações XA via JTA via Apache Aries

Os aplicativos para ServiceMix podem ser criados usando:

- Blueprints OSGi
- Spring Dynamic Modules
- Pacotes (Bundles) OSGi

Blueprints OSGi e Spring são scripts XML com a configuração da aplicação (chamada de bundle), mas utilizando uma estrutura diferente. Os Bundles OSGi são construidos em Java e implantados manualmente no contentor do ServiceMix.

[8]Imagem retirada de https://www.pngwing.com/en/free-png-iftfo

2.4 Escolha do Enterprise Service Bus

O primeira fase do desenvolvimento do projeto consiste em escolher uma ferramenta ESB que apresente funcionalidades capazes de atingir os objetivos pretendidos, o primeiro passo será portanto definir um conjuntos de requisitos para os quais terão de ser cumpridos pela ferramenta escolhida.

2.4.1 Requisitos do ESB

O primeiro requisito consiste na ferramenta ser Open Source, pois essa caracteristica fornece uma flexibilidade à ferramenta no que toca à integração com outros sistemas como o iPortalDoc, entre outras vantagens que foram apresentadas mais ao detalhe na secção 2.2.4, para não falar que uma das premissas do iPortalDoc e da IPBRICK é a utilização de software Open Source nos seus serviços.

O segundo requisito diz respeito às funcionalidades apresentadas pelo ESB, ou seja que tipo de serviços permite criar e se esses serviços apresentam relevância para os resultados que se pretendem obter com o projeto.

Outro requisito necessário é uma forma de integrar a ferramenta com o iPortalDoc, uma forma de comunicação entre a plataforma e o ESB que permita o acesso ao mesmo através da plataforma, a criação dos serviços de comunicação e a invocação desses serviços a partir do mesmo. Este requisito é essencial pois o projeto em si depende dele para a obtenção da solução pretendida.

Por fim outro requisito que irá simplificar o acesso e utilização do ESB que é a existência de uma interface web intuitiva onde é possível fazer a administração da ferramenta e dos seus serviços.

Para além dos requisitos obrigatórios pode também ser definido um requisito adicional, de caráter facultativo mas que pode ser decisivo na escolha do ESB, que é a quantidade de conectores de serviços de terceiros que é possível adicionar ao ESB para usufruir desses serviços.

2.4.2 Avaliação dos requisitos

2.4.2.1 Requisito 1

De modo a cumprir com o primeiro requisito foram escolhidas 3 ferramentas ESB Open Source. Essa escolha baseou-se na popularidade do uso das ferramentas por desenvolvedores, assim como

Requisito Obrigatório 1	O ESB tem que ser Open Source.
Requisito Obrigatório 2	O ESB tem de permitir a criação de serviços de comunicação diferentes.
Requisito Obrigatório 3	O ESB tem de ser capaz de se comunicar com o iPortalDoc.
Requisito Obrigatório 4	O ESB tem que possuir uma interface web de administração intuitiva.
Requisito Facultativo	O ESB deve possuir diversos conectores para serviços de terceiros.

Table 2.2: Tabela de requisitos do ESB.

a quantidade de menções em artigos cientificos e a quantidade de informação/material de suporte disponivel sobre a ferramenta.

As ferramentas escolhidas foram:

- **WSO2 ESB** (Mais detalhes sobre a ferramenta na secção 2.3.1)
- **Mule ESB** (Mais detalhes sobre a ferramenta na secção 2.3.2)
- **Apache ServiceMix** (Mais detalhes sobre a ferramenta na secção 2.3.4)

2.4.2.2 Requisito 2

WSO2 ESB

O WSO2 ESB oferece a possibilidade de criar e invocar Data Services que são serviços que permitem realizar operações sobre a informação armazenada numa fonte de informação como uma base de dados através de Queries de SQL. Esses Data Services podem ser invocados como Web Services Soap ou REST consoante o que for definido na configuração do Data Service. O WSO2 ESB possui compatibilidade com todos os Sistemas de Gestão de Base de Dados mais populares. No caso deste projeto possui compatibilidade com PostgreSQL.

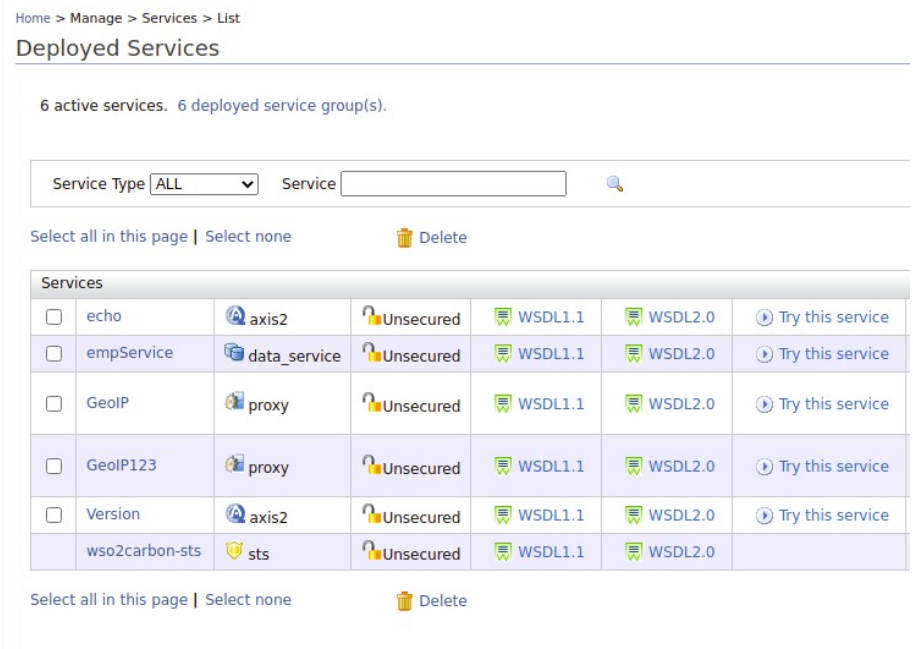

Figure 2.11: Lista de serviços definidos no WSO2.

Também permite a criação de sequências de mediação que podem ser usadas para filtrar, ou realizar outras operações sobre a informação que é enviada ou recebida para outros serviços externos como Web Services Soap e REST que sejam externos ao ESB. É possível criar REST APIs no WSO2 de modo a permitir o uso dessas sequências.

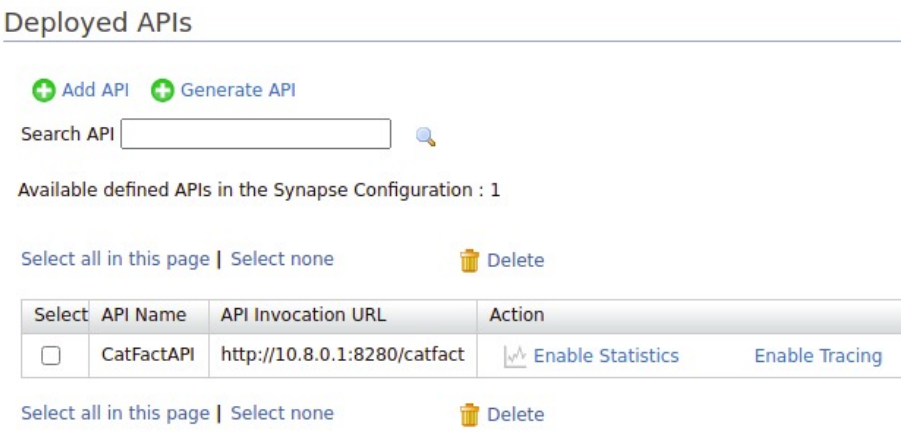

Figure 2.12: Lista de APIs definidas no WSO2.

Para além disso o WSO2 permite que sejam implantados 2 conectores, o conector File (figura

2.13) que permite a transferência de ficheiros através do protocolo FTP e o conector “Email Connector” (figura 2.14) que permite enviar e receber e-mails através dos protocolos IMAP, POP3 e SMTP.

Figure 2.13: Conector "File".

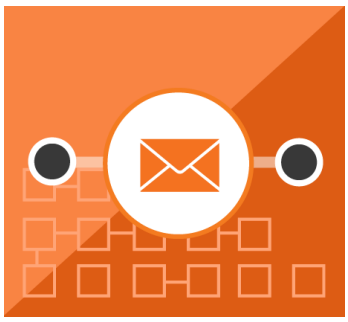

Figure 2.14: Conector "Email Connector".

Mule ESB

As funcionalidades do Mule ESB funcionam todas à base de conectores, O Mule possui um “Database Connector” que permite estabelecer comunicação entre o ESB e uma base de dados. Este conector é compatível com quase todas as bases de dados JDBC e permite que sejam efetuadas operações sobre a informação a partir de Queries de SQL.

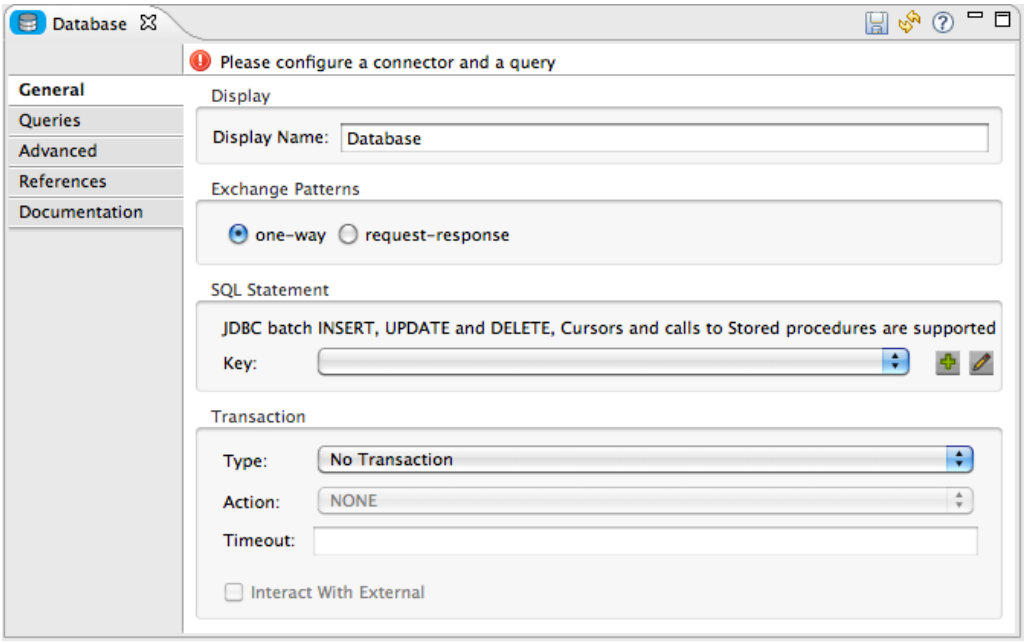

Figure 2.15: Configuração do "Database Connector" no Mule ESB.

Também possui conectores para Email como o “Email connector” que trabalha com SMTP e

"IMAP Connector" e "POP3 Connector" para trabalhar com esses mesmos protocolos. Permite o uso do protocolo FTP a partir dos conectores "File Connector" e "FTP Connector".

A partir de todos esses conectores é possível criar serviços que podem ser acedidos a partir de Web Services Soap ou REST.

Apache ServiceMix

O Apache ServiceMix permite através do Apache Camel utilizar componentes que são módulos que fornecem determinadas funcionalidades e que podem ser implantadas no ambiente do ESB. Existem centenas de componentes disponiveis e entre eles existe o componente FTP para transferência de ficheiros entre servidores e o componente Mail que faz uso dos protocolos IMAP, POP3 e SMTP para enviar e receber e-mails.

```
karaf@root>
karaf@root> features:list
State         Version          Name                                Repository                Description
[uninstalled] [5.4.2         ] activemq                            cxf-2.6.9
[installed  ] [2.6.9         ] cxf-specs                           cxf-2.6.9
[installed  ] [2.6.9         ] cxf-jaxb                            cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-abdera                          cxf-2.6.9
[installed  ] [1.6.11        ] wss4j                               cxf-2.6.9
[installed  ] [2.6.9         ] cxf-core                            cxf-2.6.9
[installed  ] [2.6.9         ] cxf-ws-policy                       cxf-2.6.9
[installed  ] [2.6.9         ] cxf-ws-addr                         cxf-2.6.9
[installed  ] [2.6.9         ] cxf-ws-rm                           cxf-2.6.9
[installed  ] [2.6.9         ] cxf-ws-mex                          cxf-2.6.9
[installed  ] [2.6.9         ] cxf-ws-security                     cxf-2.6.9
[installed  ] [2.6.9         ] cxf-http                            cxf-2.6.9
[installed  ] [2.6.9         ] cxf-http-jetty                      cxf-2.6.9
[installed  ] [2.6.9         ] cxf-bindings-soap                   cxf-2.6.9
[installed  ] [2.6.9         ] cxf-jaxws                           cxf-2.6.9
[installed  ] [2.6.9         ] cxf-jaxrs                           cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-rs-security-xml                 cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-rs-security-cors                cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-rs-security-oauth               cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-rs-security-oauth2              cxf-2.6.9
[installed  ] [2.6.9         ] cxf-databinding-aegis               cxf-2.6.9
[uninstalled] [2.6.9         ] cxf-databinding-jibx                cxf-2.6.9
[installed  ] [2.6.9         ] cxf-databinding-jaxb                cxf-2.6.9
```

Figure 2.16: Lista de funcionalidades instaladas do ServiceMix

Para além desses componentes é possível associar o componente CXF para expor o serviço como um Web Service Soap ou CXF-RS para expor o serviço como um Web Service REST.

Resumo do Requisito 2

No geral as 3 ferramentas apresentam capacidades semelhantes relativamente aos serviços que dispõem e que são relevantes para o desenvolvimento do projeto.

No entanto a criação dos serviços nas 2 primeiras ferramentas é mais simples e intuitiva, passando pelo preenchimento de parâmetros ou a escolha de componentes numa sequência, algo que geralmente é acompanhado de uma interface gráfica que ajuda a visualizar o desenvolvimento

do serviço. Já no Apache ServiceMix a criação de serviços está dependente exclusivamente de scripts em XML, apresentando uma complexidade maior de desenvolvimento.

2.4.2.3 Requisito 3

Este requisito avalia se os ESBs possuem métodos para comunicar com o iPortalDoc e a eficiência desses métodos.

Como já referido os 3 ESBs possuem a capacidade de criar serviços que dão a possibilidade de serem invocados através de Web Services Soap e REST, mas no que toca à comunicação entre o ESB e o iPortalDoc é necessário ter algo a mais que possibilite a realização de mais operações sobre o ESB a partir do iPortalDoc. Entre essas operações, pretende-se que seja possivel criar, gerir e remover utilizadores do ESB a partir do iPortalDoc.

Das 3 ferramentas apresentadas apenas o WSO2 apresenta uma solução que cumpre com este requisito, pois ele dispõe aos utilizadores um conjunto de Web Services de Administração, que permitem que o utilizador a partir de um cliente Soap (que pode ser facilmente criado em PHP) realizar operações sobre a ferramenta que vão desde a gestão de utilizadores, à criação de serviços, APIs, endpoints, datasources, sequências de mediação e obtenção de informação acerca dos recursos mencionados.

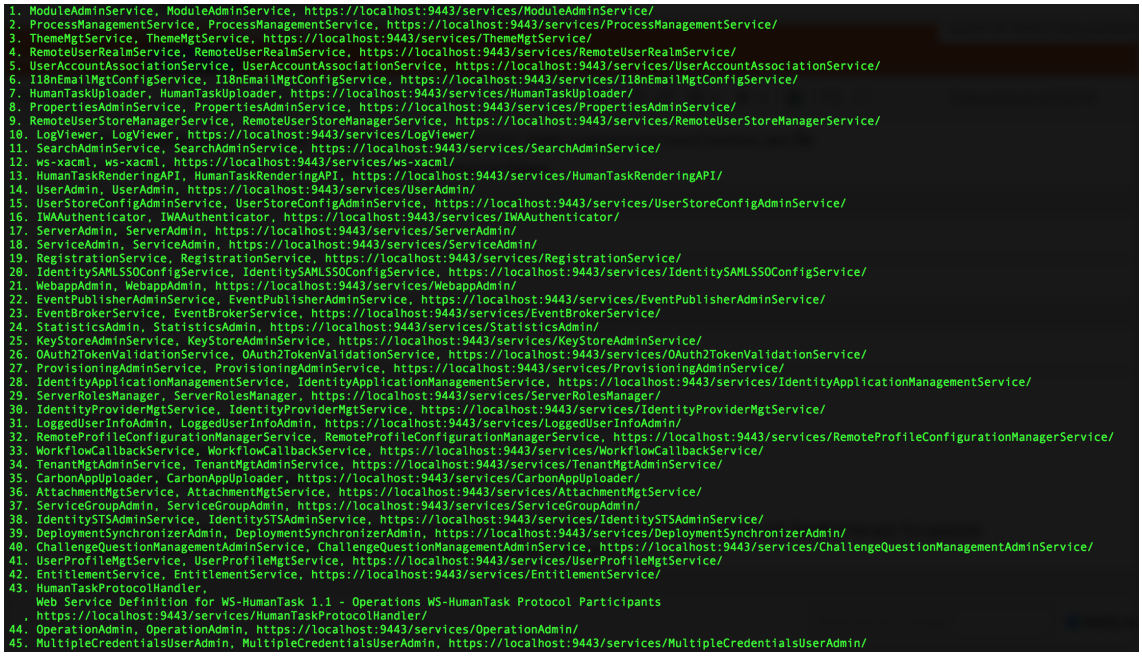

Figure 2.17: Lista de Web Services de Administração do WSO2.

O Mule ESB e o Apache ServiceMix não possuem qualquer funcionalidade que permite controlar o ESB de forma remota pelo iPortalDoc e portanto não cumprem com este requisito.

2.4.2.4 Requisito 4

WSO2 ESB

O WSO2 ESB traz de base uma consola de gestão (WSO2 Management Console[23]) que é uma interface web para a criação e gestão dos recursos fornecidos pela ferramenta assim como a gestão dos utilizadores do ESB.

A interface é bastante intuitiva e permite:

- A criação, modificação ou remoção de serviços ou APIs;
- A inserção de aplicações criadas externamente ao ESB e que podem conter diversos serviços ou APIs;
- A inserção e uso de conectores para os mais variados serviços de terceiros;
- Criação de utilizadores e a atribuição de diferentes permissões aos mesmos;
- Criação de endpoints para servir de interface com serviços externos;
- Definição de datasources para a realização de operações sobre a informação presente em bases de dados;
- Acesso a um sistema de monitoramento que fornece logs e estatisticas dos serviços em execução, permitindo o controlo de variáveis e a deteção de erros;
- Segurança de autenticação através de LDAP;
- Segurança nas comunicações através do uso de HTTPS.

Mule ESB

O Mule Management Console[24] (MMC) é uma interface web que centraliza as funções de gestão e monitoramento para todas as implementações do Mule ESB, para a sua versão paga, a versão Enterprise. A consola não permite criar instâncias (serviços do Mule ESB) a partir da mesma, sendo necessário recorrer a um IDE especifico o Anypoint Studio para desenvolver as instâncias.

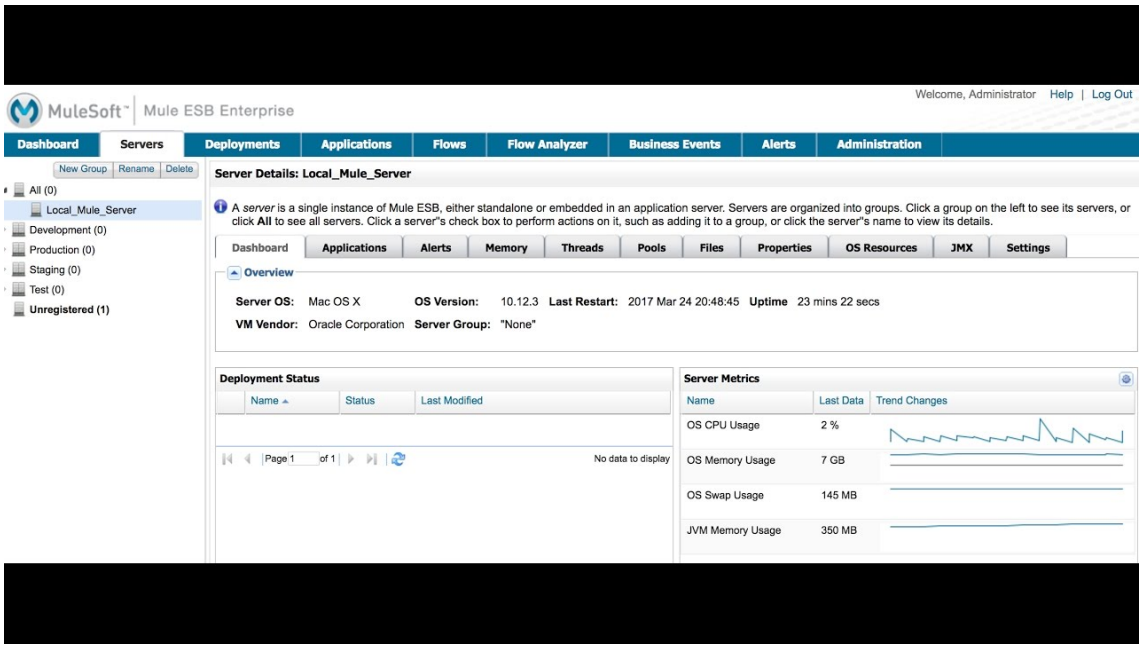

Figure 2.18: Mule Management Console.

Esta ferramenta permite:

- A análise das métricas das instâncias em tempo real;
- A deteção de erros no ambiente de desenvolvimento, teste e produção;
- O acesso a estatisticas que revelam a performance e o consumo dos recursos do ESB;
- Controlo da execução dos recursos, permitindo iniciar, reiniciar ou parar os mesmos;
- Fornece um sistema de alertas e notificações para detetar falhas na execução das instâncias;
- Segurança de autenticação através de LDAP;
- Segurança nas comunicações através do uso de HTTPS.

Apache ServiceMix

No caso do Apache ServiceMix, a ferramenta possui uma consola Web que é fornecida pelo módulo Apache Karaf[25] e que não vem de base com o ESB mas pode ser instalada no contentor.

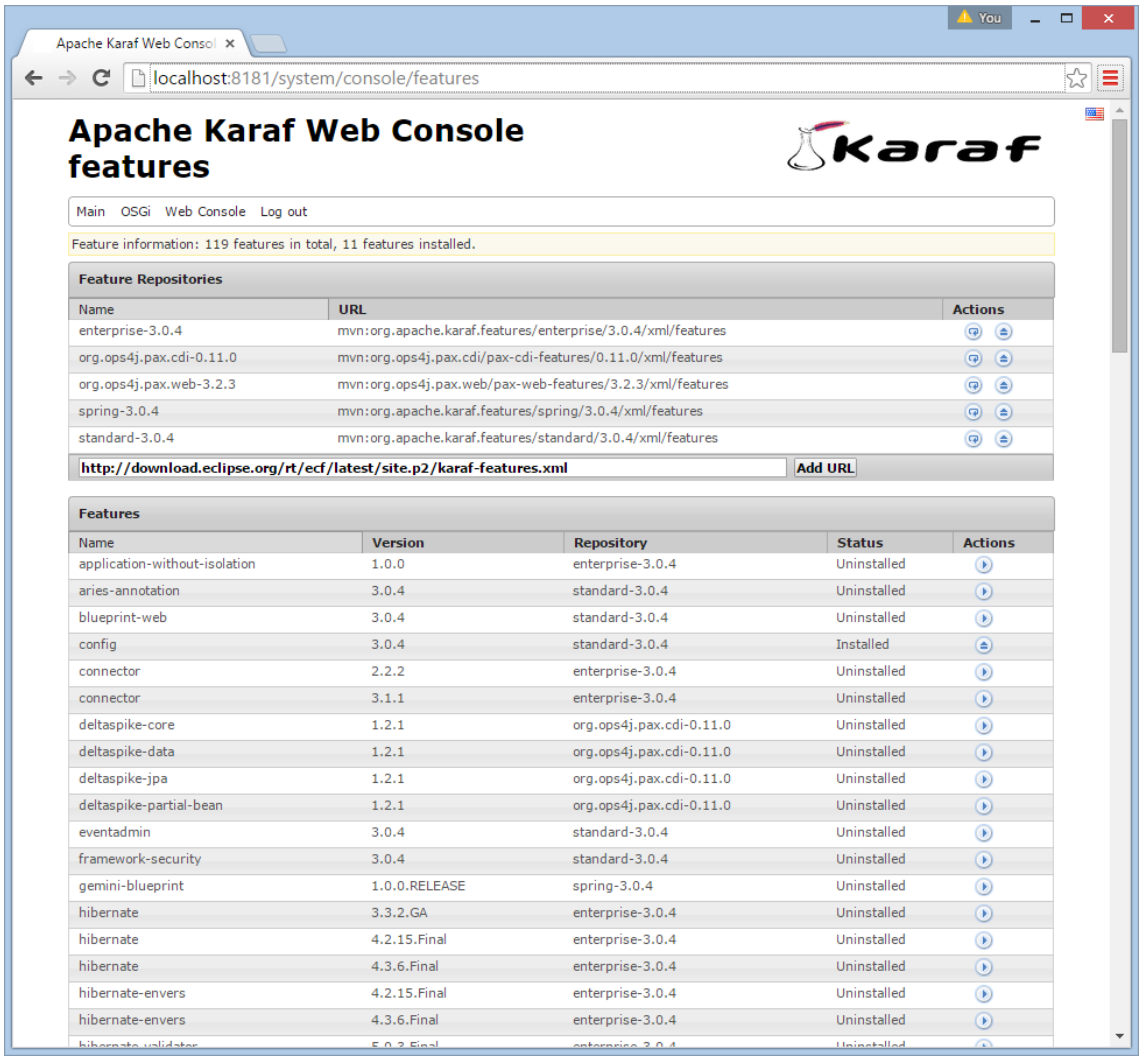

Figure 2.19: Apache Karaf Web Console.

Esta interface permite:

- A instalação de novas funcionalidades através dos componentes do Apache Camel;
- O controlo dos bundles (os serviços criados através de configurações XML), permitindo instalá-los e iniciar ou parar a sua execução;
- Monitorização dos bundles através de logs.

Resumo do Requisito 4

O objetivo pretendido com a Interface Web neste projeto é dar aos utilizadores do iPortalDoc a possibilidade de interagirem com o ESB e poderem criar e gerir serviços a partir desse meio.

Apesar dos 3 ESBs apresentarem interfaces Web para interação com o utilizador, fica claro que o WSO2 e o Mule apresentam os 2 interfaces bastante mais intuitivas e com mais funcionalidade do que o Apache ServiceMix que apresenta uma interface muito simples, obsuleta e que não permite realizar grande variedade de operações.

No entanto o WSO2 apresenta uma clara vantagem em relação ao Mule (e também em relação ao Apache ServiceMix), pois dá a possibilidade ao utilizador de criar os serviços pretendidos a partir da interface, não sendo portanto necessário recorrer a um IDE externo para desenvolver esses serviços, mas não retirando a possibilidade do uso de um IDE para esse efeito.

Outra caracteristica que favorece o WSO2 neste requisito é que a Interface encontra-se disponivel para uso com todas as suas funcionalidade na versão gratuita para uso do WSO2, enquanto que a Interface do Mule está restrita à versão Enterprise desse ESB, que requer a subscrição de um plano pago.

2.4.2.5 Análise de Requisitos

Nesta secção cada ESB vai receber uma pontuação por requisito que vai de 0 a 3, sendo a avaliação da seguinte realizada da seguinte maneira:

- **0** - Se o requisito não for cumprido;
- **1 a 3** - Se o requisito for cumprido, sendo a pontuação atribuida de forma crescente conforme o número de vantagens apresentadas para o cumprimento do requisito.

A análise é a apresentada na seguinte tabela:

Requisito 1 (Ser Open Source)	WSO2 ESB	Mule ESB	Apache ServiceMix.
Requisito 2 (Capacidade de criar serviços de comunicação necessários)	3	3	3
Requisito 3 (Capacidade de comunicação com o iPortalDoc)	3	0	0
Requisito 4 (Interface Web para interação com utilizadores)	3	2	1
Total	9	5	4

Table 2.3: Análise de requisitos do ESB.

Com base na pontuação da tabela relativamente aos requisitos discutidos neste capitulo, podemos concluir que o WSO2 ESB é a ferramenta mais apropriada para integração a solução que se pretende.

Assim sendo o uso do requisito facultativo definido na tabela 2.2 não se aplica neste caso. No entanto pode-se referir que o Apache ServiceMix apresenta um número bastante elevado de conectores (centenas deles) bem superior ao apresentado pelo WSO2 e Mule, que se encontram bem equilibrados neste requisito.

Chapter 3

A plataforma iPortalDoc

Este capítulo contextualiza e pormenoriza a plataforma sobre a qual vai incidir a dissertação, nomeadamente a plataforma iPortalDoc da IPBRICK, e serão apresentados os conceitos e funcionalidades pertencentes a essa plataforma que terão relevancia para o desenvolvimento deste projeto. Permitindo atribuir um contexto a esses conceitos no desenvolvimento da solução e um melhor entendimento de cada etapa que o constitui.

3.1 iPortalDoc

O iPortalDoc é um Sistema de Gestão de Documentação e Workflow que permite aos utilizadores gerirem o fluxo dos documentos de uma empresa ou instituição, assim como os processos adjacentes a essas organizações, permitindo a uniformização e automatização dos processos de trabalho. Garante que os dados, informações ou tarefas passam de pessoa em pessoa, de acordo com a hierarquia do fluxo pré-definido. Esse fluxo é definido com recurso a Workflows. O iPortalDoc contribui para o aumento da produtividade nas empresas e da qualidade dos serviços prestados, facilita a pesquisa e evita perdas de tempo e de informação[26].

Permite o registo e o tratamento de todas as comunicações como e-mails, conversas de chat, chamadas telefónicas e que fiquem associadas aos documentos/processos a que dizem respeito, podendo ser consultados com segurança a qualquer momento mediante a atribuição de permissões. A aplicação permite obter por pasta da hierarquia documental, o tipo de perfil, utilizador, grupos, as permissões, os tipos de documento, workflows e macros que os utilizadores têm atribuídos, permitindo efetuar a análise e gestão desta informação[27].

Para o iPortalDoc funcionar corretamente é necessário que estejam a correr no Sistema Operativo IPBRICK OS. É a partir desse Sistema Operativo que é possível introduzir e gerir utilizadores no iPortalDoc, assim como gerir os grupos aos quais eles pertencem[28].

3.2 IPBRICK OS

O IPBRICK OS é um sistema operativo baseado em Linux Debian que atua como plataforma de Comunicações para Empresas. Para além de ter integrado no seu ambiente o iPortalDoc, apresenta outros 3 módulos essenciais para a comunicação dentro de organizações, sendo eles:

- **IPBRICK.MAIL**: Que assegura um acesso seguro e uma gestão eficaz da conta de e-mail, podendo ser acedido por serviços como o Outlook ou o Mozilla Thunderbird e permitindo também um acesso a calendários, contactos, notas e tarefas[29].

- **IPBRICK.CAFE**: Que é uma rede social corporativa que permite que os utilizadores possam ser divididos em grupos consoante o departamento a que pertencem, e dá acesso a um chat e a possibilidade de reuniões por voz ou videoconferência. Também dispõe de um feed onde podem ser partilhadas noticias e informações importantes pelos membros da organização[30].

- **IPBRICK.UcoIP**: Que é uma central de Comunicações Unificadas sobre IP, que funciona em Cloud Privada. Está integrada com o iPortalDoc e permite que as Chamadas, Emails e Conversas de Chat sejam gravadas e fiquem associadas aos documentos e processos a que dizem respeito, podendo posteriormente ser consultados[31].

O IPBRICK OS permite a gestão dos utilizadores do iPortalDoc apenas ao perfil designado como administrador do sistema. Inicialmente será lhe pedido para se autenticar utilizando as suas credenciais e depois será lhe mostrada a página com as definições do sistema (figura 3.1) [28].

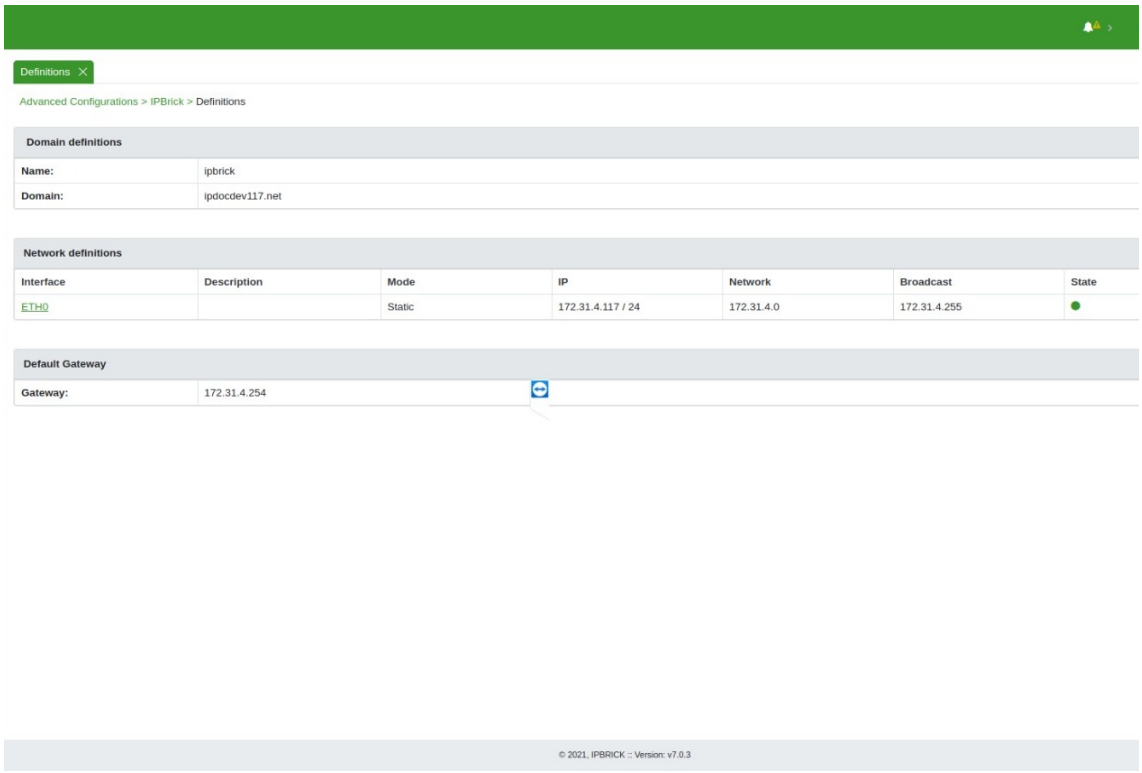

Figure 3.1: Página inicial do IPBRICK OS.

No menu é possível através das opções Directory→Users Management → Users List visualizar a lista de utilizadores e adicionar novos com os parâmetros apresentados na figura 3.2. A estes utilizadores será dado posteriormente o acesso ao iPortalDoc.

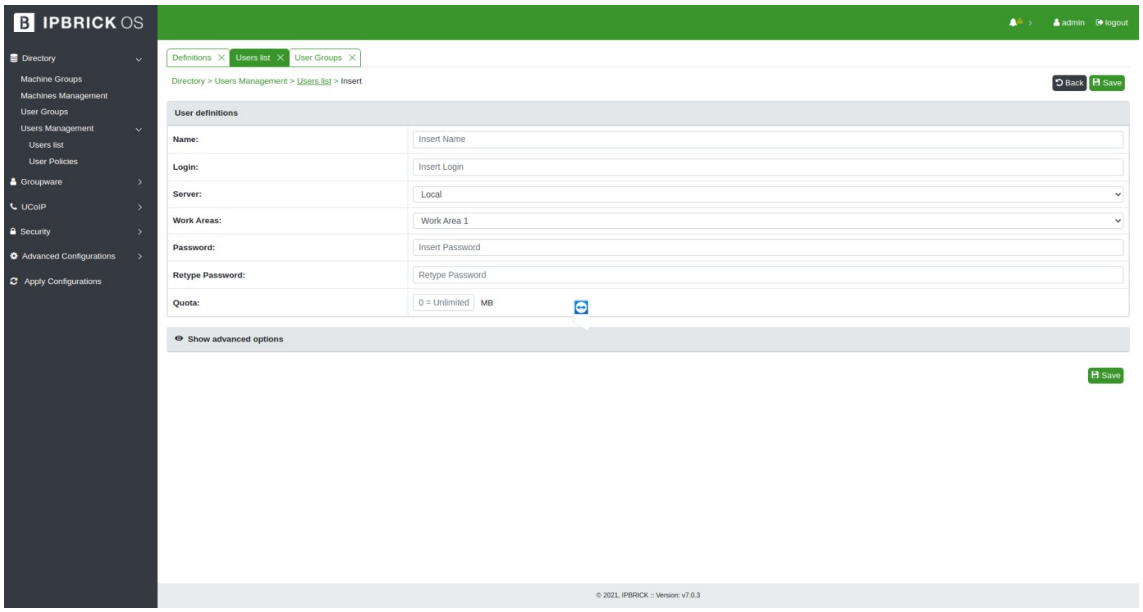

Figure 3.2: Adicionar um novo utilizador ao IPBRICK OS.

É também possível associar os utilizadores definidos a grupos com diferentes permissões através das opções Directory→ User Groups (figura 3.3).

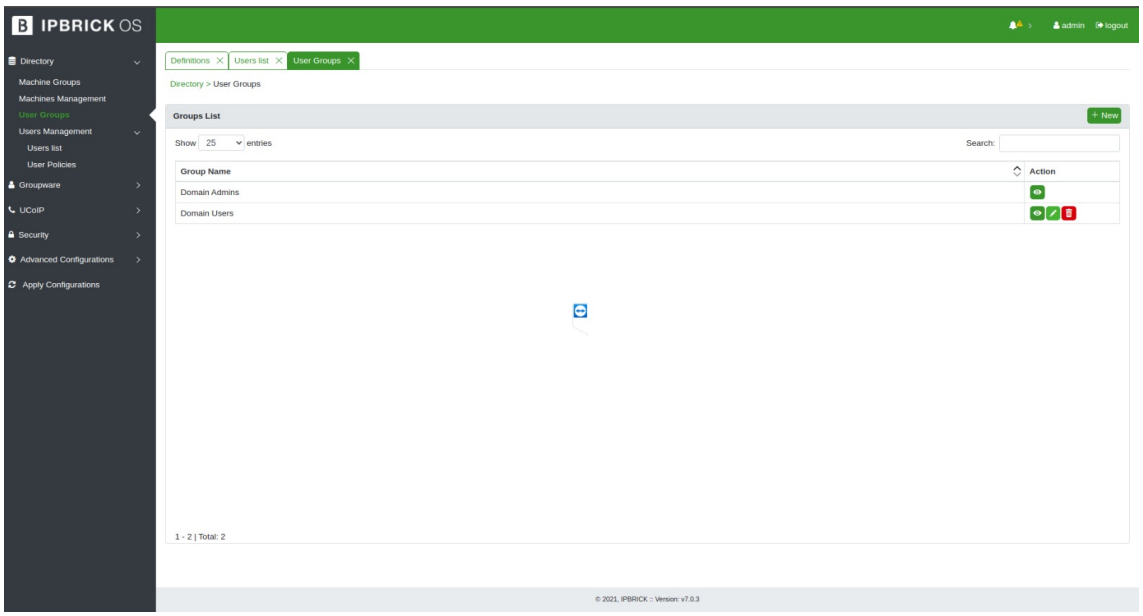

Figure 3.3: Grupos aos quais os utilizadores podem ser associados.

Após salvar o progresso é possível ao administrador através do iPortalDoc dar acesso a essa plataforma aos utilizadores definidos previamente. Esse acesso é dado transferindo os utilizadores

da coluna “Utilizadores de Sistema” para a coluna “Utilizadores iPortalDoc ativos” (figura 3.4).

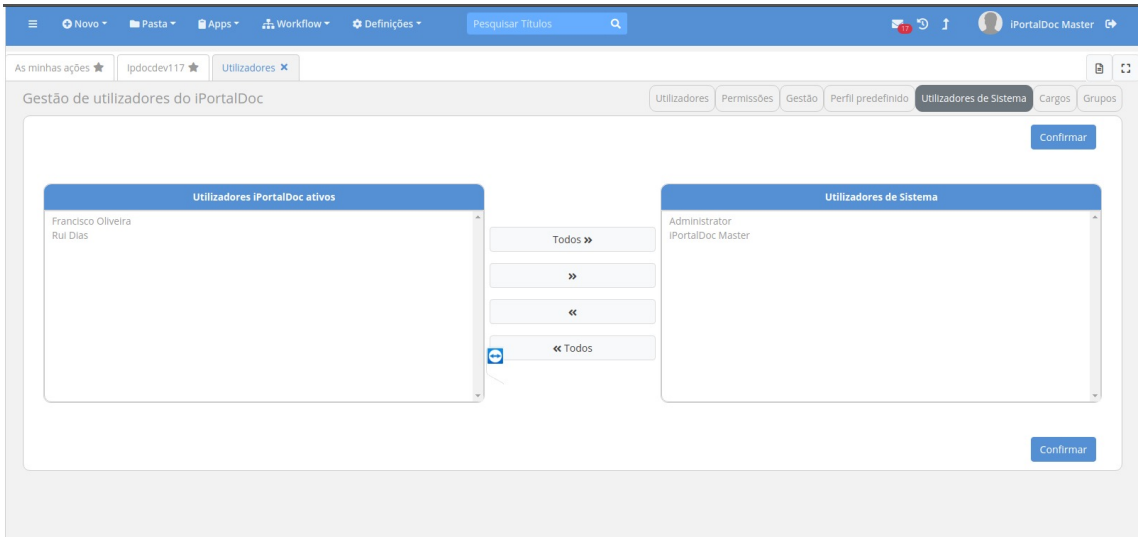

Figure 3.4: Acesso dos utilizadores ao iPortalDoc.

3.3 Workflows

É através dos workflows (fluxos de trabalho) que é possível fazer a gestão de documentos e processos no iPortalDoc . Qualquer documento que se encontre no iPortalDoc tem um workflow próprio. No mınimo possui uma etapa de armazenamento, ou poderá ter predefinido um circuito no qual vários utilizadores podem participar, realizando tarefas sobre o documento até que ele atinja o estado final [28].

O iPortalDoc apresenta um menu chamado Workflow onde os utilizadores com cargo de “Super User ” ou o Administrador podem criar, instanciar e configurar novos Workflows.

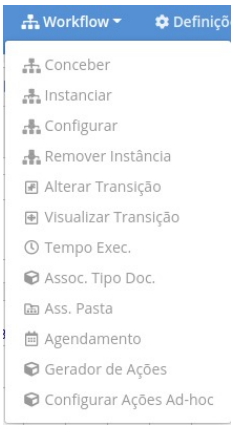

Figure 3.5: Menu Workflow do iPortalDoc.

Podem ser criados Workflows do zero ou alterados templates já disponiveis de origem com o iPortalDoc. O esboço do Workflow é criado no motor de workflow próprio do iPortalDoc que está apresentado na figura 3.6. Para criar esse esboço estão disponiveis no menu da esquerda simbolos que podem ser inseridos na zona quadriculado por Drag and Drop.

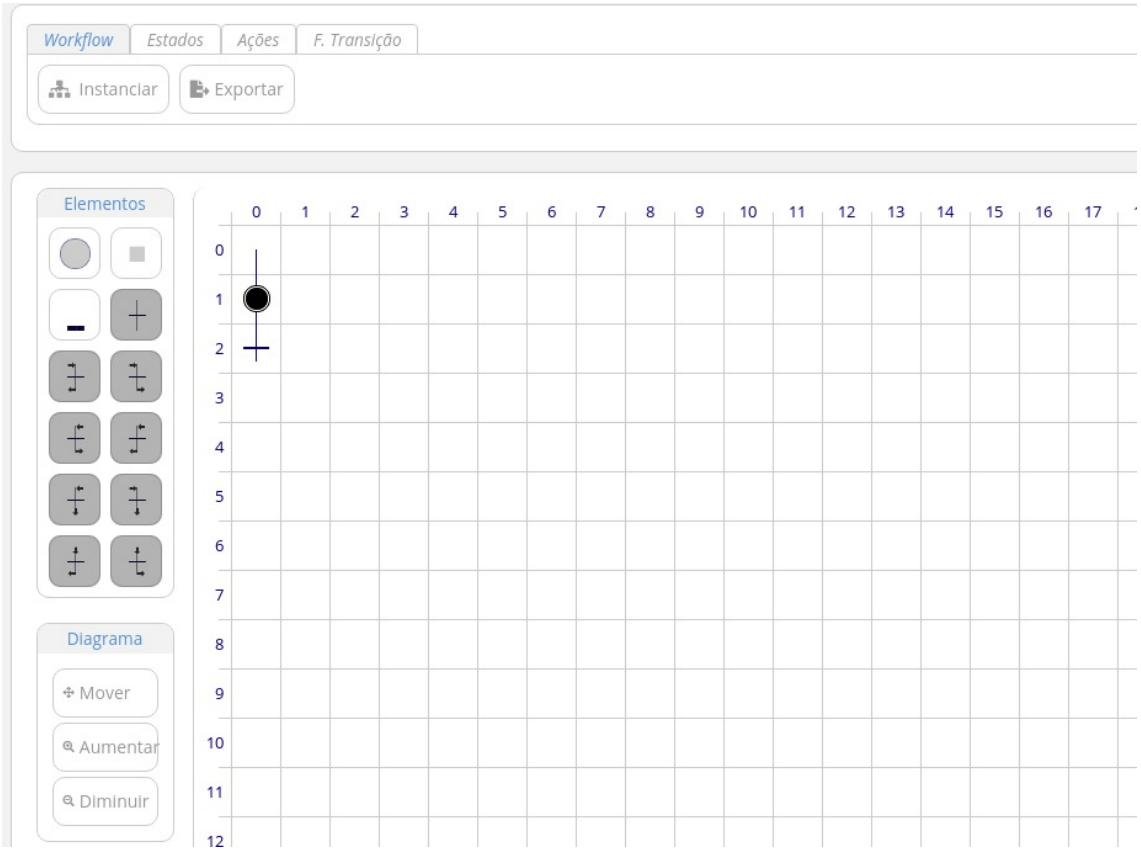

Figure 3.6: Motor de Workflow do iPortalDoc.

O workflows são baseados em máquinas de estados e todos partem de um estado inicial de armazenamento (representado pelo circulo preto na figuras 3.6 e 3.7) que aquando da inserção do documento no iPortalDoc transita para o primeiro estado definido.

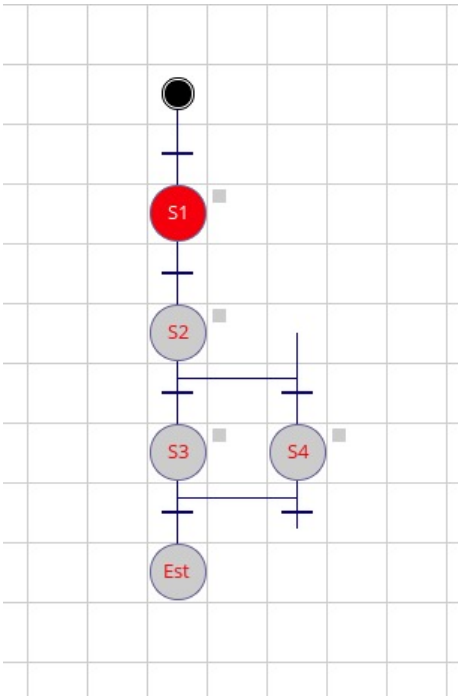

Figure 3.7: Exemplo de Workflow.

Os estados são representados pelos circulos no circuito e cada estado pode ter associadas uma ou mais ações, que são representadas pelos quadrados de menor dimensão que se encontram representados à direita dos estados. As ações são processos adjacentes ao estado e que produzem resultados, que irão disparar uma transição de estado especifica que irá transitar o documento para um outro estado do workflow.

Por exemplo, a ação do estado S2 que é "Classificar Documento" pode produzir 2 resultados: "Classificado" o que faz o Workflow transitar para o estado S3 ou "Não Classificado" que faz o Workflow transitar para o estado S4. As transições de estado são representadas pelo simbolo da figura 3.9.

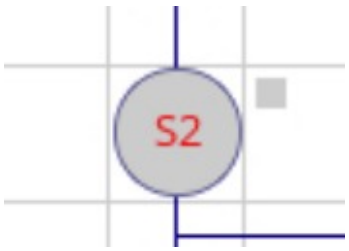

Figure 3.8: Estado e ação correspondente.

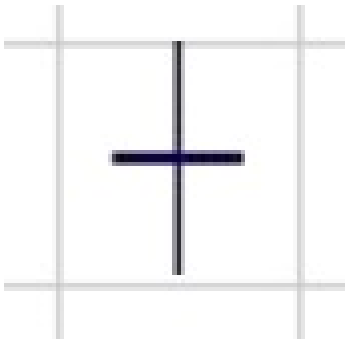

Figure 3.9: Transição de estado.

O último estado do workflow tem que obrigatóriamente se chamar “Estado Final” e como o primeiro é um estado de armazenamento.

Após a criação do workflow é possível instanciá-lo atribuindo um nome à intância.

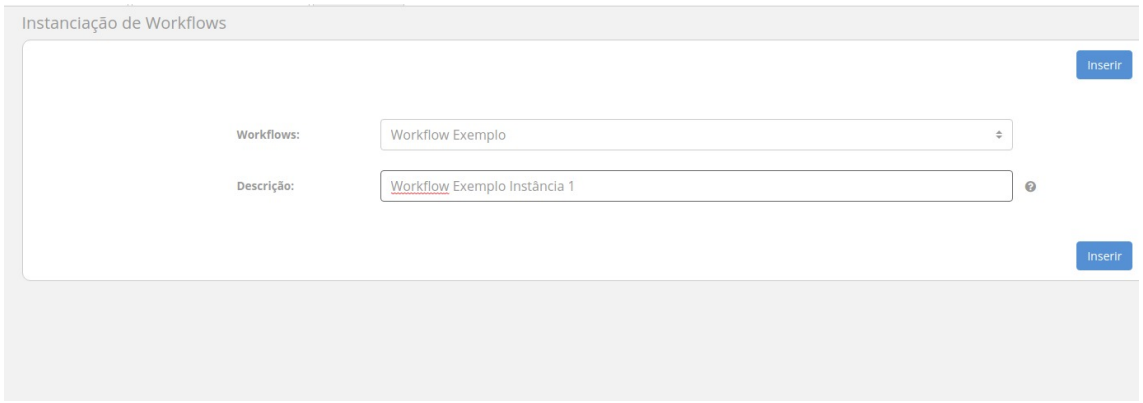

Figure 3.10: Criação da instância de um Workflow.

Para utilizar o Workflow criado é necessário configurá-lo, isto significa atribuir a cada estado o utilizador responsável por realizar a ação ou ações que a ele estão adjacentes e selecionar a opção “Ativo” para ativar o Workflow. Assim o Workflow fica disponivel para ser usado aquando da inserção de um documento no iPortalDoc como demonstrado na figura 3.12.

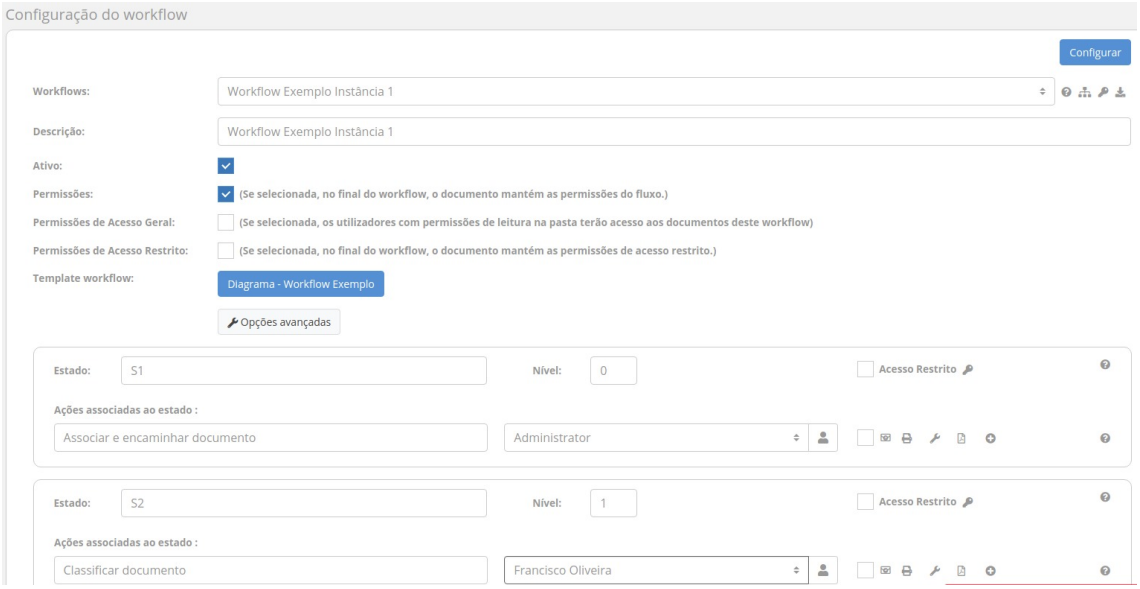

Figure 3.11: Configuração de um Workflow.

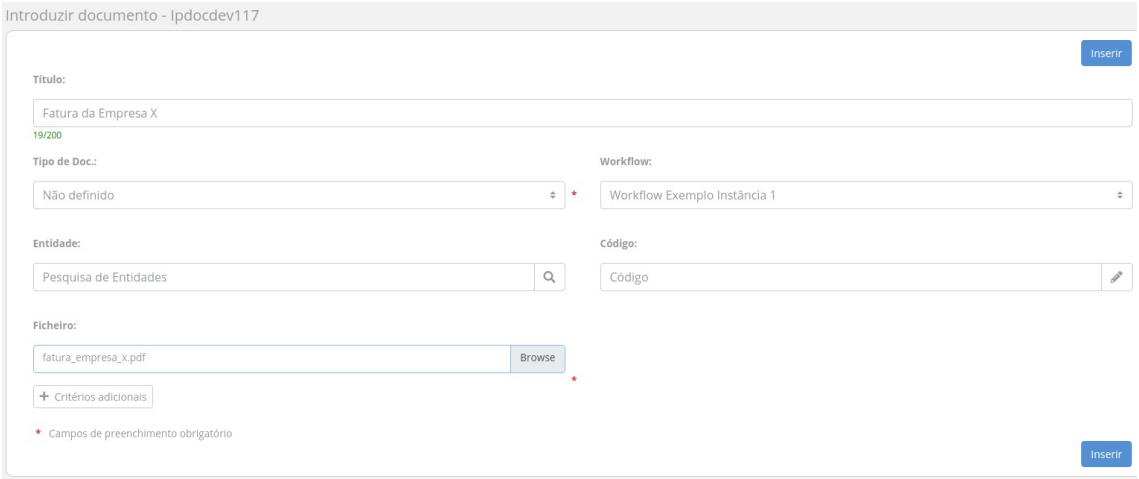

Figure 3.12: Inserção de documento para seguir determinado Workflow.

3.4 Ações

As ações referem-se a processos que podem ser executados na transição para ou durante um estado de um Workflow. O iPortalDoc fornece de base um conjunto de ações que representam processos que podem ser habitualmente aplicados a documentos, como classificação, assinatura digital ou encaminhamento, mas também é possível criar ações personalizadas.

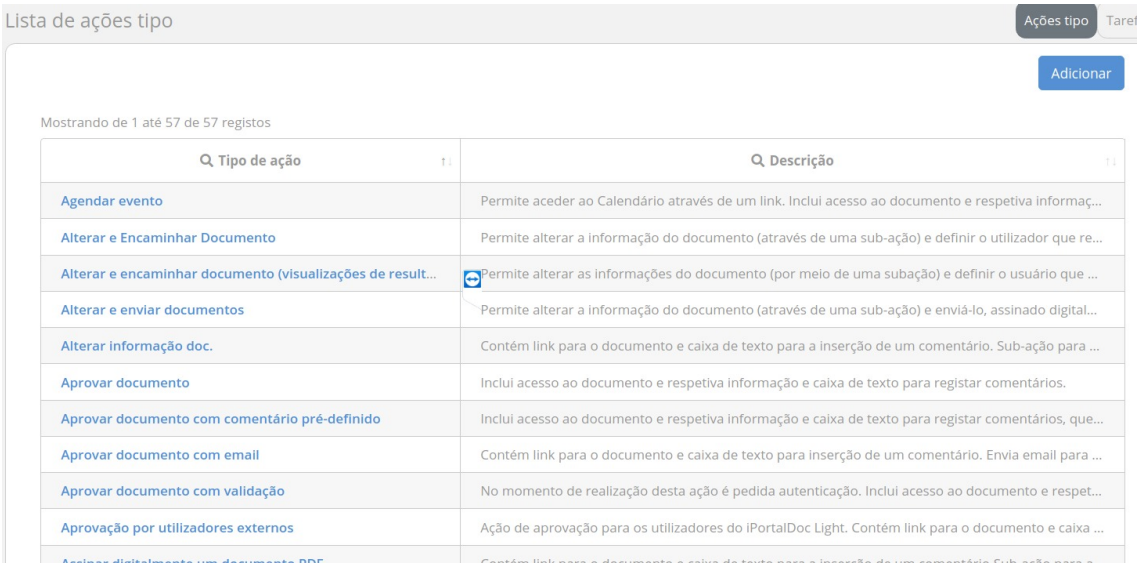

Figure 3.13: Lista de ações do iPortalDoc

A criação das ações personalizadas é permitida a utilizadores com cargo de "Super User" ou ao administrador a partir da opção "Gerador de Ações" do menu Workflow (figura 3.5), onde é possível criar ações do zero ou adaptar as ações que vêm de base.

As ações nada mais são do que um conjuntos de tarefas que são selecionadas e configuradas pelo criador da ação e que vão ser executadas quando a ação for executada num estado do Workflow. Para além das tarefas é possível adicionar outras configurações adicionais como sub-ações que permitem realizar outra ações dentro da primeira, ou tornar a execução da ação automática. As tarefas serão abordadas mais ao detalhe na secção 3.5.

Figure 3.14: Tarefas disponiveis para adicionar numa ação.

3.5 Tarefas

As tarefas são funções que podem ser selecionadas para serem executadas numa ação. Cada tarefa apresenta parâmetros de configuração que vão ser definidos durante a criação da ação e que vão definir o comportamento da mesma durante o sua execução no Workflow.

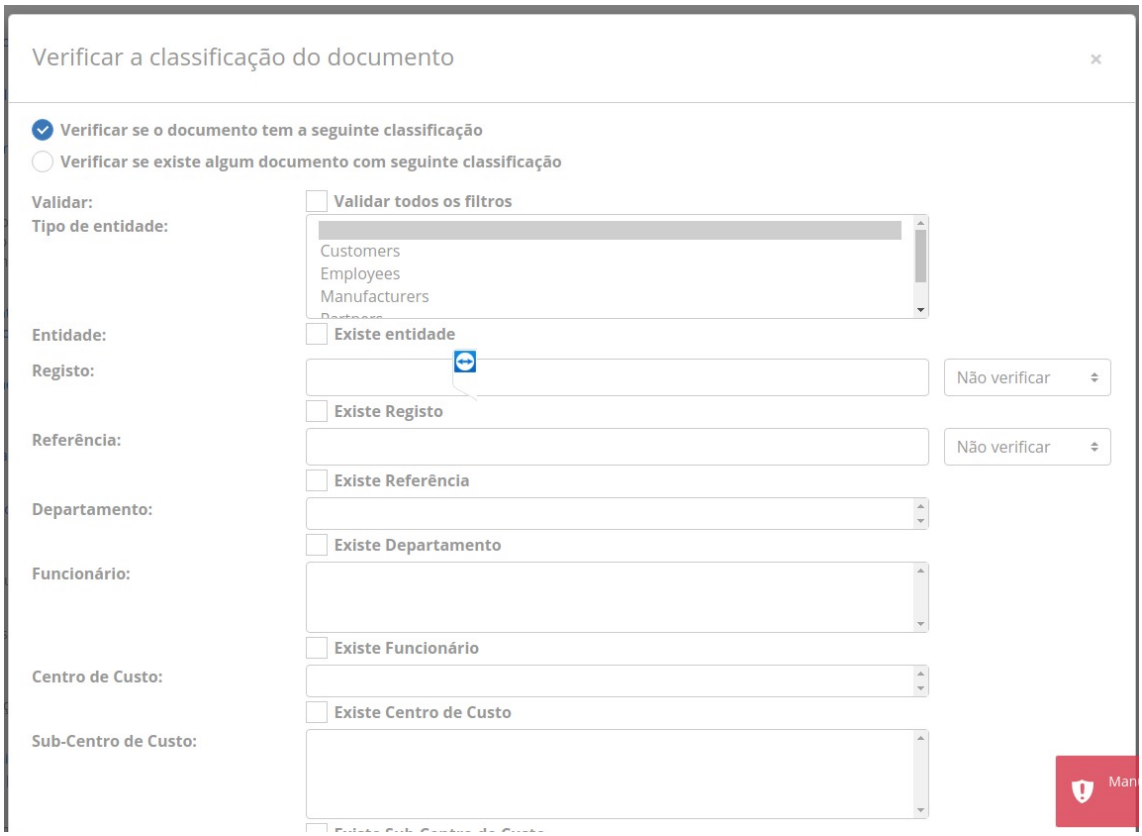

Figure 3.15: Configuração da tarefa "Verificar a classificação do documento".

Ao contrário das ações, as tarefas não podem ser criadas de maneira personalizada, ou seja apenas é possível ao utilizador associar à ação as tarefas que vêm previamente com o iPortalDoc, podendo no máximo atribuir uma configuração diferente à mesma tarefa.

Para adicionar novas tarefas ao iPortalDoc seria necessário trabalhar no código fonte e criar a função que define a nova tarefa no backend, assim como criar a interface de configuração com todos os parâmetros da tarefa no frontend.

Chapter 4

Integração do ESB no iPortalDoc

Este capitulo é dedicado à apresentação das fases de integração do WSO2 ESB no ambiente do iPortalDoc.

4.1 Fases da integração

A integração propriamente dita do ESB no iPortalDoc comporta as seguintes fases:

- A gestão de contas de acesso ao ESB pelo iPortaldoc, ou seja, deverá ser criado um mecanismo dentro do iPortalDoc que interaja com o ESB e crie, configure ou remova contas de acesso ao ESB de acordo com o desejado;
- A criação de uma interface no iPortalDoc para a gestão dos utilizadores do ESB, que permite ao administrador gerir de forma simples e intuitiva o acesso dos utilizadores do iPortalDoc ao ESB. Essa interface deve fazer uso do mecanismo criado na fase anterior;
- Adicionar o ESB ao menu de aplicações do iPortalDoc, apresentando-o apenas aos utilizadores com acesso à ferramenta. Essa opção do menu deverá abrir a interface web do WSO2 numa outra aba do navegador e já ter o utilizador autenticado de forma automática na sua conta do ESB.

4.2 Tecnologias

Nesta secção serão apresentadas as tecnologias que serão usadas no desenvolvimento da projeto.

4.2.1 PHP e Javascript

O iPortalDoc trata-se de uma aplicação Web que tem o backend construido em PHP e o frontend construido em Javascript. Será necessário criar as funções para interação com o ESB em PHP e usar Javascript para criar as funções que interagem com a Interface do iPortaldoc.

4.2.2 Web Services REST e Soap

Serão criados e usados Web Services REST e Soap para integrar a ferramenta ESB no iPortalDoc e para acessar aos serviços definidos no ESB.

4.2.3 JSON e XML

Os dados usados nos pedidos e respostas dos Web Services estarão descritos nas linguagens neutras de JSON e XML, o que permite que possam ser interpretados e tratados por qualquer linguagem de programação, neste caso por PHP.

4.2.4 PostgreSQL

PostgreSQL é o sistema de gestão da base de dados do iPortalDoc e onde serão armazenadas os dados de acesso ao ESB pelos utilizadores do iPortalDoc e dados acerca dos Serviços do ESB que podem ser acessados pelo iPortalDoc.

4.3 Ferramentas de trabalho

Nesta secção serão identificadas as ferramentas de trabalho a utilizar ao longo do projeto.

4.3.1 Máquina de testes

Uma máquina virtual providenciada pela IPBRICK com uma cópia do código do iPortalDoc de modo a permitir que sejam feitas alterações para a integração do ESB e para posteriormente serem realizados testes às alterações.

4.3.2 Enterprise Service Bus

A ferramenta Enterprise Service Bus que for escolhida será integrada no ambiente do iPortalDoc e onde serão criados serviços para permitir a comunicação entre diversas aplicações.

4.3.3 Overleaf

A plataforma Overleaft irá permitir formatar todos os documentos relacionados à dissertação através da linguagem de formatação Latex.

4.3.4 SoapUI e Postman

O Postman e o SoapUI permitem testar Web Services que permitiram a integração do ESB no iPortalDoc assim como permitirão ter acesso a serviços definidos através dessa ferramenta. Ambas

as ferramentas permitem testar Web Services REST e Soap, que terão o papel mais importante neste projeto.

4.3.5 Apache Netbeans

O Apache Netbeans será o IDE onde serão feitas as alterações ao código do iPortalDoc e onde será feito o debbug desse mesmo código para que sejam detetados possiveis erros.

4.4 Mecanismo de gestão de contas de acesso ao ESB

De maneira a criar um mecanismo de gestão do acesso ao ESB, temos de perceber em primeiro lugar como são criadas e configuradas as contas de utilizadores no WSO2.

A forma mais simples e direta de criar e configurar uma conta no WSO2 ESB é através da Interface Web, onde autenticando com as credenciais do administrador é possivel aceder à aba 'Configure' e na opção 'Users and Roles', temos a opção de criar novos utilizadores. Ao aceder a essa opção é nos mostrado o painel da figura 4.2.

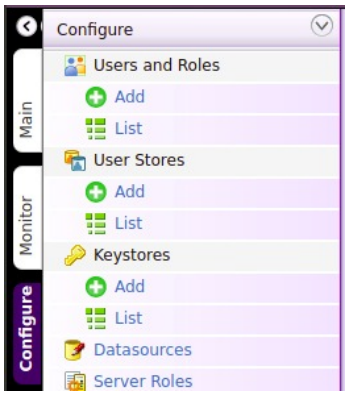

Figure 4.1: Aba 'Configure' com as opção 'User and Roles'.

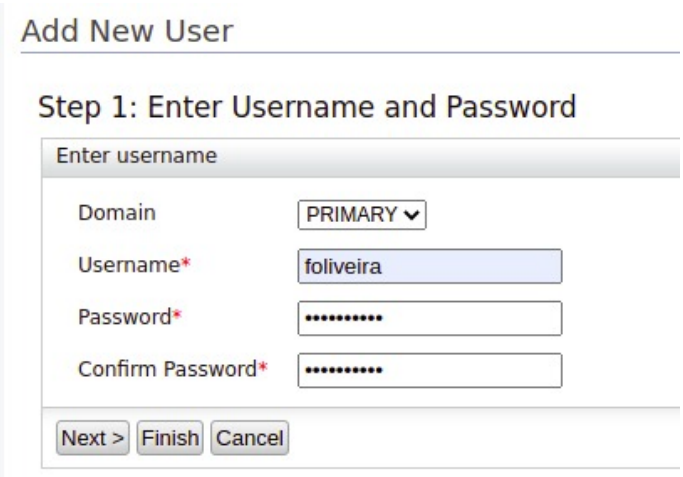

Figure 4.2: Formulário de criação de utilizador.

Através desse formulário podemos simplesmente definir um utilizador pelo nome e palavra-passe e de seguido procedemos à atribuição de permissões a esse utilizador (figura).

Figure 4.3: Formulário de atribuição de permissões.

Concluindo a configuração podemos verificar que o utilizador se encontra listado juntamente com a conta do administrador.

Figure 4.4: Lista de utilizadores.

Apesar desta forma de criação e configuração de utilizadores ser bastante simples e intuitiva, não possibilita a criação dos mesmos a partir do iPortalDoc, seria necessário o administrador do iPortalDoc entrar no ESB externamente para criar os utilizadores e não é isso que se pretende.

No entanto é também possivel fazer esta gestão de utilizadores a partir do Web Services de Administração que o WSO2 fornece. Neste caso especifico isso é possivel através do serviço **UserAdmin** que permite a invocação de alguns métodos que permitem criar, configurar e remover utilizadores.

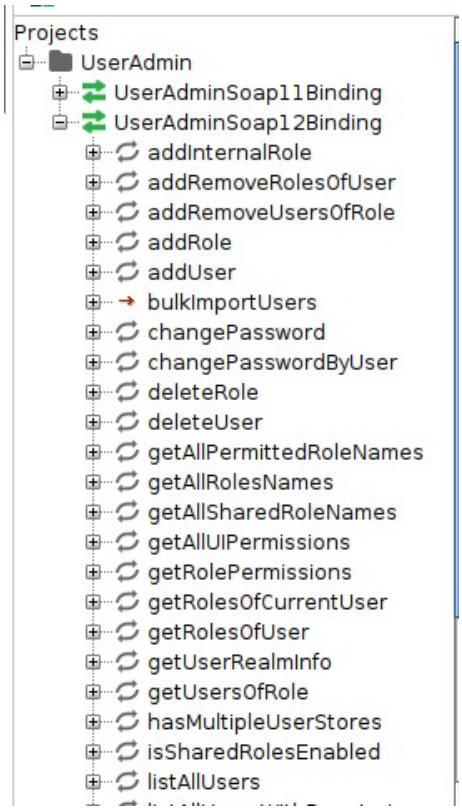

Figure 4.5: Métodos do serviço 'UserAdmin' no SoapUI.

Dos métodos apresentados, aqueles que são de interesse para as funcionalidades pretendidas são:

- **addUser**: Para a criação e configuração do utilizador;
- **deleteUser**: Para a remoção de um utilizador;
- **listAllUsers**: Para evitar utilizadores com o mesmo Username.

Os parâmetros necessários para cada um destes métodos são apresentados nas seguintes figuras:

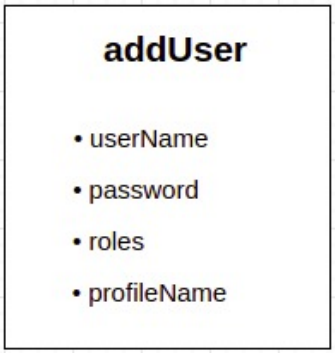

Figure 4.6: Parâmetros do método 'addUser'.

Figure 4.7: Parâmetros do método 'deleteUser'.

O método **listAllUsers** não apresenta argumentos de input, apenas retorna uma resposta com a lista de utilizados definidos no ESB.

Sabendo que estes métodos estarão no centro do mecanismo de gestão de utilizadores do ESB no iPortalDoc, foi criada uma classe em PHP no backend, chamada **libESB_WS** que agrupa as funções que irão chamar estes métodos.

Todas as funções incluem um cliente Soap em PHP que é gerado partindo do seguinte modelo:

```php
ini_set('soap.wsdl_cache',0);
ini_set('soap.wsdl_cache_enabled',0);


$wsdl_url="https://localhost:9443/services/UserAdmin?wsdl";

$context = stream_context_create(array(
    'ssl' => array(
    'verify_peer' => false,
    'verify_peer_name' => false,
    'allow_self_signed' => true
    )
));


$client=new SOAPClient($wsdl_url, array('stream_context' => $context,
 'location' => $wsdl_url,
'uri' => 'http://org.apache.axis2/xsd',
'login'    => 'admin',
'password' => 'admin'));

$request = ['userName' => $login];

$client->deleteUser($request);
```

Figure 4.8: Template de cliente Soap em PHP.

As primeiras 2 linhas desativam a funcionalidade de WSDL Caching, de seguida definimos o url do WSDL correspondente ao serviço que vamos usar, neste caso o **UserAdmin**, definimos todas as configurações do Cliente Soap e instânciamos um objeto da classe **SOAPClient** com essas configurações e com as credenciais do administrador do ESB.

De seguida definimos os parâmetros num array associativo e utilizamos esses parâmetros para fazer uma chamada ao método que pretendemos usar, caso da figura 4.8, o método **deleteUser**.

4.4.1 Estrutura da Classe libESB_WS

A classe libESB_WS possui na sua estrutura 3 funções:

- function **addUser** ($login, $password, $name)

A função cria um novo utilizador com os parâmetros que recebe e atribui-lhe permissões de administrador através da chamada do método **addUser** utilizando o modelo de cliente Soap especifico na figura 4.8.

- function **deleteUser** ($login)

A função remove o utilizador especificado pela parâmetro $login através da chamada do método **deleteUser** utilizando o modelo de cliente Soap especifico na figura 4.8.

- function **userExists** ($login)

A função invoca o método **listAllUsers** utilizando o modelo de cliente Soap especifico na figura 4.8 e recebe a lista de utilizadores do ESB, compara com o parâmetro $login e retorna o valor boleano **true** se existir e **false** no caso contrário.

4.5 Interface para gestão dos utilizadores do ESB

Nesta fase do projeto pretende-se desenvolver uma interface no iPortalDoc que permita ao administrador controlar o acesso dos utilizadores do iPortalDoc ao ESB. Para esse efeito foi usada como base a interface de gestão de utilizadores do iPortalDoc já apresentada na figura 3.4.

A interface deve apresentar 2 colunas, uma com os utilizadores do iPortalDoc com acesso ao ESB e outra com os que não têm acesso e no meio botões de interação que permitam fazer alterar as permissões deles. O resultado final da interface é o da figura 4.16.

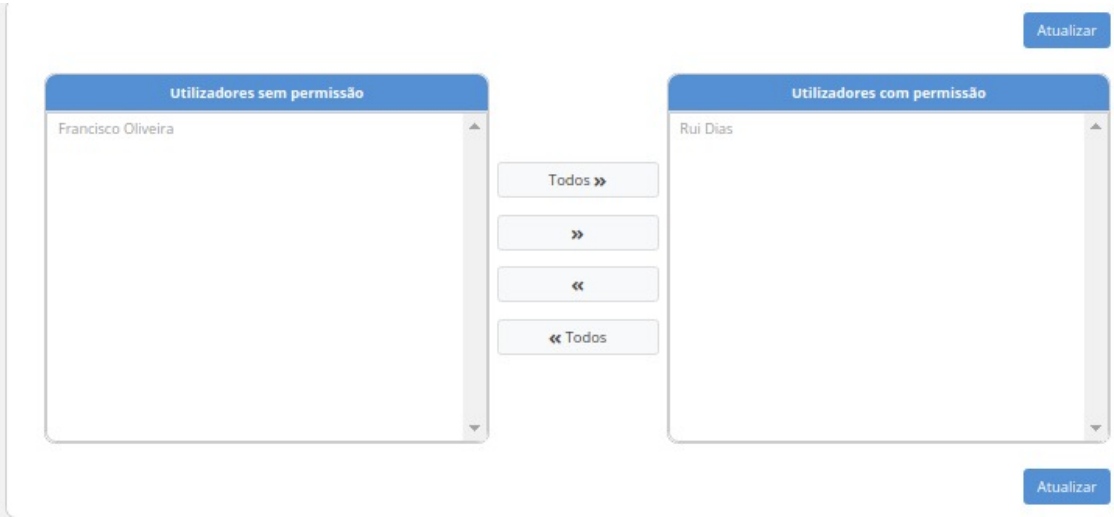

Figure 4.9: Interface de gestão de utilizadores do ESB.

De modo a conseguir obter as informações em relação aos utilizadores que têm acesso ao iPortalDoc e que irão aparecer na interface da figura 4.16 é necessário entender como funciona a gestão dos mesmos no iPortalDoc.

Os utilizadores ao serem adicionados através da IPBRICK como descrito na secção 3.2 são geridos a partir do protocolo LDAP que armazena as suas informações numa tabela da base de dados do iPortalDoc, a tabela **mailutilizador** apresentada na figura **??**.

utilizador	mail	nome	activo	login	imp_var
10027	rdias@ipdocdev117.net	Rui Dias	f	rdias	09KEplEaqdvvwkEEySBwSg==
1	ipdocmaster@ipdocdev117.net	iPortalDoc Master	f	ipdocmaster	Q8uZMUvW8TUmQDYmznJSnw==
10000	administrator@ipdocdev117.net	Administrator	f	administrator	X4u7b4MQZfAj0XcHFNyUNw==
10001	foliveira@ipdocdev117.net	Francisco Oliveira	f	foliveira	Q8uZMUvW8TUmQDYmznJSnw==

(4 rows)

Figure 4.10: Tabela mailutilizador.

Após serem adicionados nessa tabela pela IPBRICK os utilizadores necessitam de permissão para acesso ao iPortalDoc que é dado pelo administrador na Interface de gestão como demonstrado na figura 3.4. Após receberem esse acesso ao iPortalDoc, o campo 'activo' da tabela **mailutilizador** passa para o valor **true**.

utilizador	mail	nome	activo	login	imp_var
10027	rdias@ipdocdev117.net	Rui Dias	f	rdias	09KEplEaqdvvwkEEySBwSg==
10001	foliveira@ipdocdev117.net	Francisco Oliveira	t	foliveira	Q8uZMUvW8TUmQDYmznJSnw==
1	ipdocmaster@ipdocdev117.net	iPortalDoc Master	f	ipdocmaster	Q8uZMUvW8TUmQDYmznJSnw==
10000	administrator@ipdocdev117.net	Administrator	f	administrator	X4u7b4MQZfAj0XcHFNyUNw==

(4 rows)

Figure 4.11: Utilizador "foliveira" com acesso ao iPortalDoc.

Foi definido que os utilizadores que podem ter acesso ao ESB, são os que já possuem acesso ao iPortalDoc, portanto foi criado uma nova tabela na base de dados do iPortalDoc chamada **utilizador_esb_access** e que através da mudança do valor do campo 'activo' da tabela **mailutilizador** para **true**, armazena as informações desses utilizadores na tabela nova com um campo adicional 'esb_access' que é um boleano e será sempre iniciado a **false** aquando a inserção de um novo utilizador.

utilizador	mail	nome	activo	login	esb_access
10001	foliveira@ipdocdev117.net	Francisco Oliveira	t	foliveira	f

(1 row)

Figure 4.12: Tabela utilizador_esb_access.

Tendo isto em mente podemos descrever esta fase de integração com base no seguinte esquema:

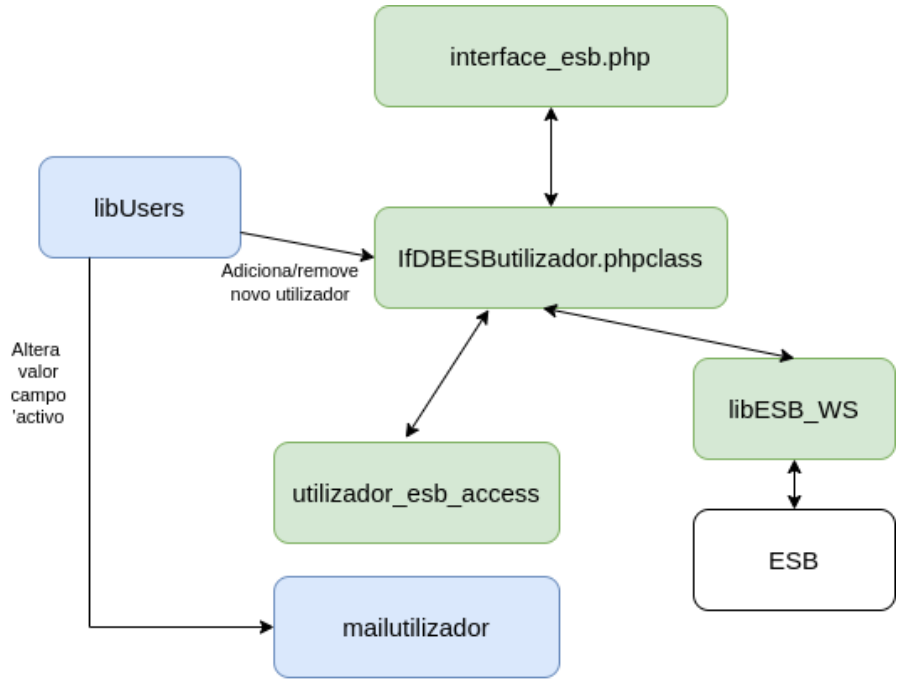

Figure 4.13: Esquema de gestão de utilizadores

A azul encontram-se representados os ficheiros que já se encontravam no iPortalDoc e a verde os novos ficheiros que foram criados.

É a partir do ficheiro **libUsers.php** que é alterado no backend a valor do campo 'activo' na tabela **mailutilizador**, a partir dessa alteração são adicionados ou removidos da tabela **utilizador_esb_access** os utilizadores do iPortalDoc.

Essa gestão é efetuada pelo ficheiro **IfDBESButilizador** que é uma classe que é responsável pela atualização da tabela **utilizador_esb_access** através de queries SQL e que ao mesmo tempo utiliza as funções da classe **libESB_WS** para gerir as contas no ESB através da invocação dos serviços de administação do WSO2.

O ficheiro **interface_esb** é o ficheiro que contém o frontend da Interface representada na figura 4.16 e é através do ficheiro **IfDBESButilizador** que o campo **esb_access** da tabela **utilizador_esb_access** é atualizada consoante as permissões definidas na interface, **true** se o utilizador tiver acesso ao esb e **false** em caso contrário.

4.6 Adição do ESB ao menu do iPortalDoc

Para ser possivel aceder ao ESB a partir do iPortalDoc é necessário adicioná-lo ao menu de aplicações do mesmo.

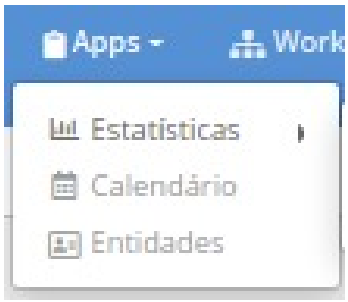

Figure 4.14: Menu Apps do iPortalDoc.

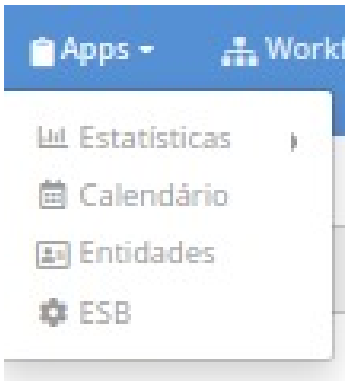

Figure 4.15: Menu Apps com a adição do ESB

Para isso é necessário adicioná-lo ao ficheiro libMenu que faz a gestão dos menus do iPortalDoc. Mas a opção para aceder ao ESB deverá apenas aparecer quando o utilizador em questão tiver permissão para aceder à ferramenta.

Isso é possivel através do seguinte esquema:

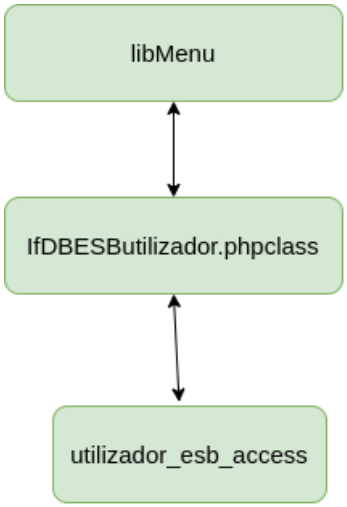

Figure 4.16: Esquema para apresentação do ESB no menu Apps.

O ficheiro libMenu deverá instanciar um objecto da classe **IfDBESButilizador** e verificar se o

utilizador tem acesso ao ESB através da tabela **utilizador_esb_access**. O nome do utilizador em questão está guardado numa variável global do iPortalDoc "**$perfil->utilizador**" e através desse valor é possivel verificar o acesso ao ESB.

Se o utilizador tiver acesso, a opção "ESB" estará disponivel no menu Apps e deverão ser retornados os credenciais da conta do ESB, que são exatamente os mesmo do iPortalDoc.

4.7 Login Automático no ESB

O sistema responsável pelo login automático pode ser definido pelo seguinte diagrama de blocos.

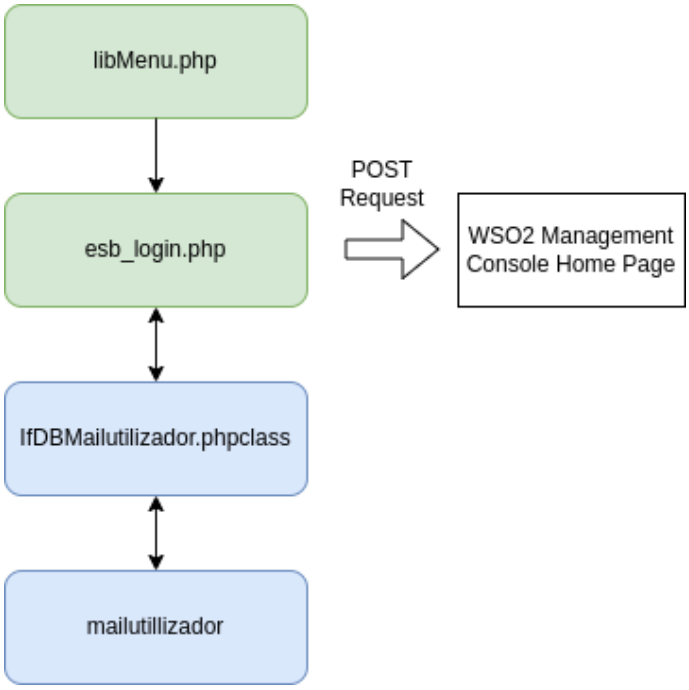

Figure 4.17: Diagrama do sistema de Login Automático.

Como citado na secção 4.6, se for verificado que o utilizador do iPortalDoc tem acesso ao ESB, as suas credenciais serão retornadas a partir da tabela **mailutilizador** e o login pode ser feito através de um pedido POST definido no ficheiro **esb_login.php** que abre a página inicial da Interface Web do WSO2 numa nova aba do navegador já autenticado na devida conta.

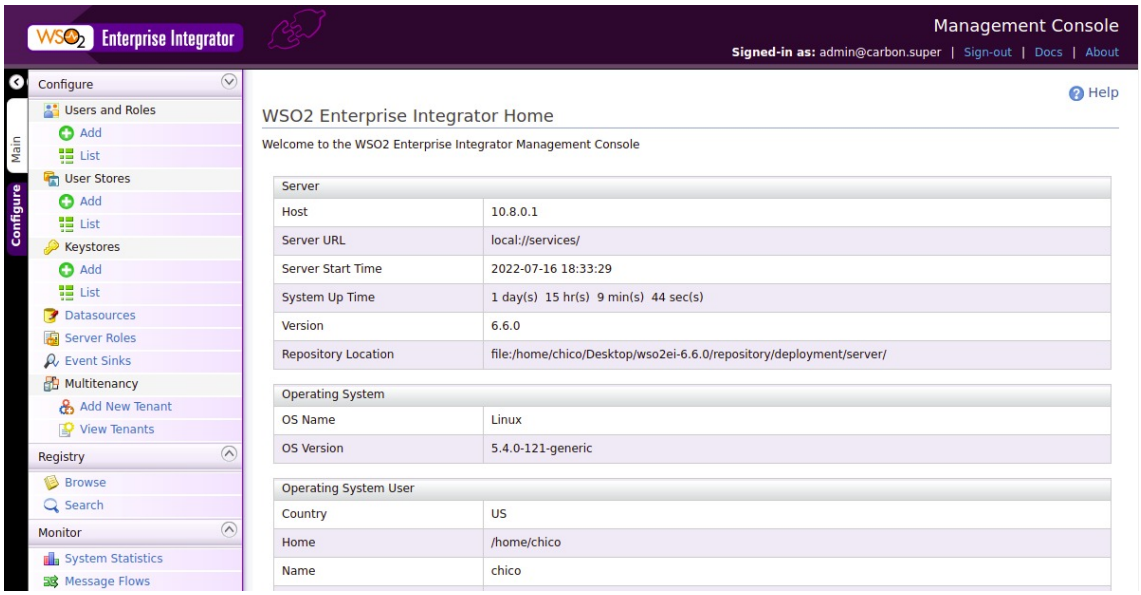

Figure 4.18: Página inicial do WSO2 Management Console.

Chapter 5

Caso de estudo

De modo a demonstrar a funcionalidade da solução implementado pelo projeto, vai ser realizado um caso de estudo em que será apresentada a forma prática de utilizar o serviços do ESB através do iPortalDoc e ao mesmo tempo validando a solução.

O caso em especifico fará uso de Data Services que serão invocados através das ações de um Workflow que será criado no iPortalDoc e que terá como função criar ou atualizar uma entidade que estará associada a um documento inserido no sistema.

5.1 Criação de entidades

As entidades podem ser adicionadas manualmente ao iPortalDoc a partir da aplicação Contacts, o utilizador pode aceder a partir do menu Apps e será redirecionado para a aplicação, onde poderá se autenticar e adicionar manualmente a entidades e as suas informações associadas.

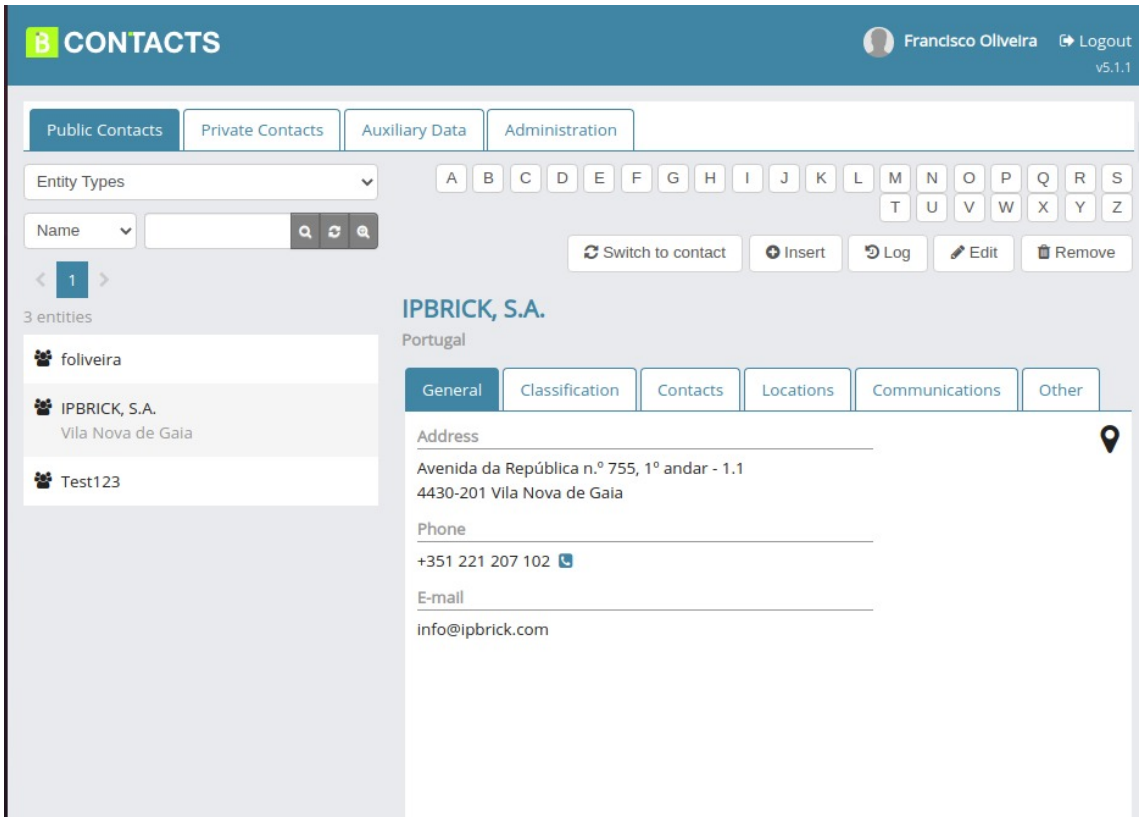

Figure 5.1: Aplicação Contacts e entidades definidas.

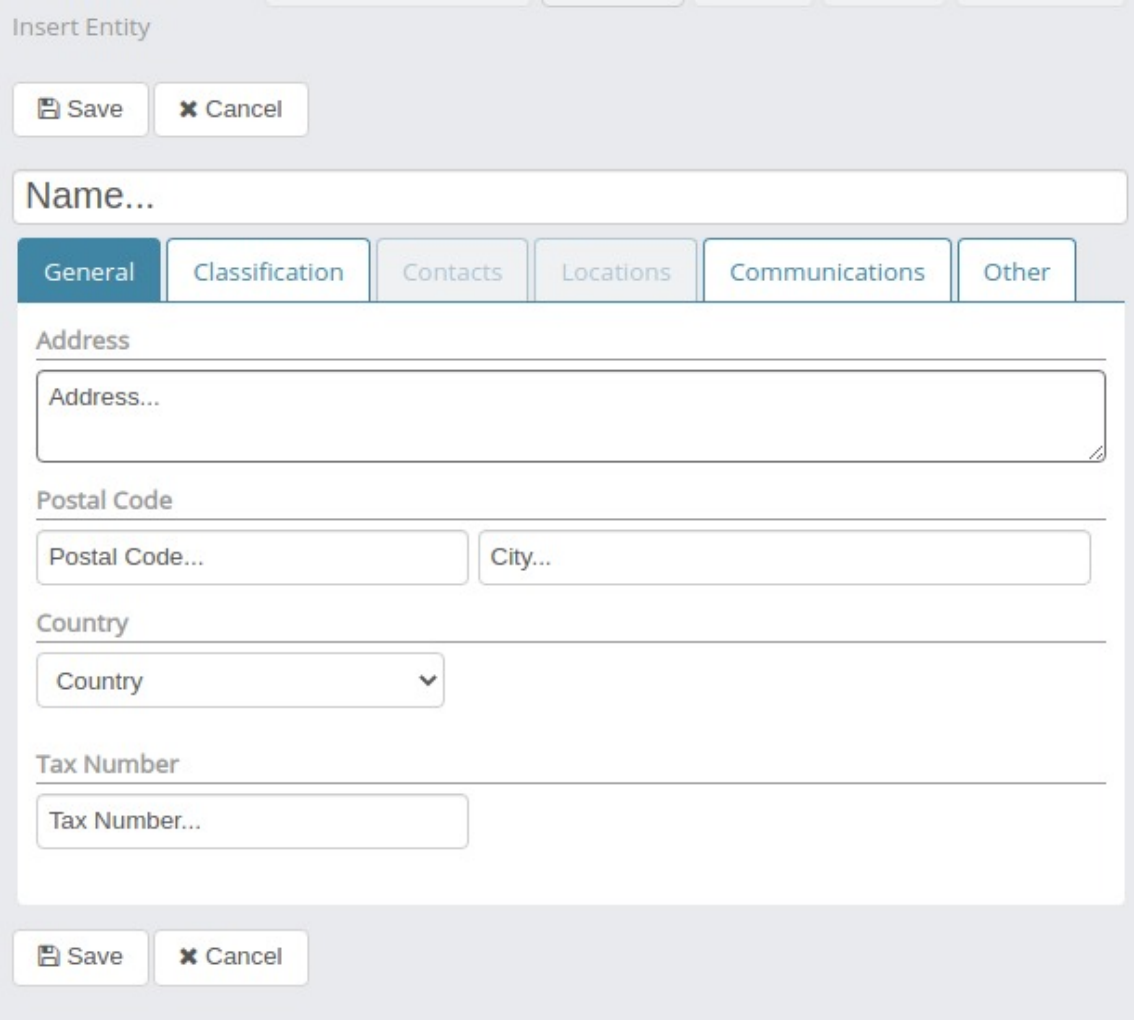

Figure 5.2: Configuração da entidade no Contacts.

No entanto, proceder à criação ou atualização da informação das entidades de maneira manual é um processo pouco eficiente e que acaba por ser trabalhoso quando se trata de uma grande quantidade de documentos a serem inseridos no iPortalDoc. Dado isso, uma solução automatizada através de workflows parece ser a solução ideal para efetuar essas operações.

5.2 Solução com workflow

Inicialmente será necessário adicionar ao formulário de inserção de documentos no iPortalDoc, campos de informação novos, contendo o nome da Entidade associada aquele documento, como informação associada à mesma. Neste caso foi apenas selecionado o NIF como informação da Entidade para além do nome da mesma.

Figure 5.3: Campos adicionais no formulário de inserção do documento.

Agora ao inserir o documento, o utilizador também poderá preencher esses campos e selecionar o workflow devido, que será especificado daqui para a frente.

O workflow deve ser capaz de ler os campos "Nome da Entidade" e "NIF", deve verificar se pelo menos o primeiro campo se encontra preenchido e em caso positivo deve verificar se a Entidade já existe no iPortalDoc. No caso da entidade existir, o valor do NIF deve ser atualizado e no caso de não existir a entidade deve ser criada. Por fim deve ser retornado o ID da Entidade.

O caso descrito pode ser representado pelo seguinte Workflow:

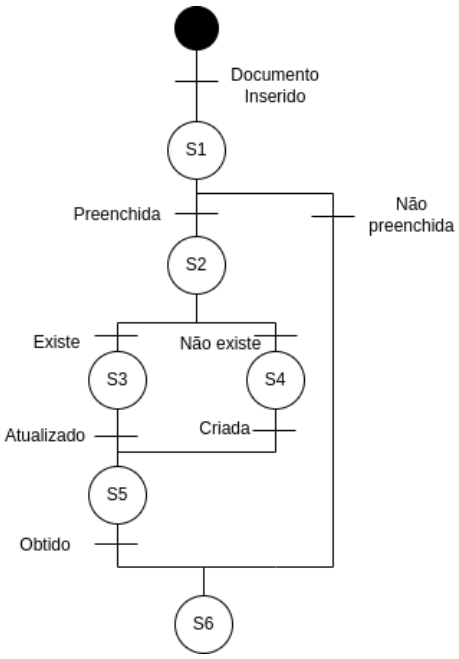

Figure 5.4: Estrutura do Workflow.

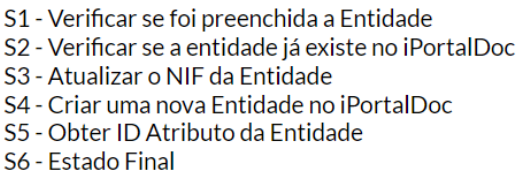

Figure 5.5: Definição dos estados do Workflow.

Dos estados definidos na figura 5.5, com a exceção do estado S6, cada um dos outros, terá associado uma ação que será executada com base num Data Service definido no ESB.

5.3 Definição dos Data Services

De modo a obter a funcionalidade esperada em cada estado do workflow da figura 5.4, terão de ser definidos Data Services no ESB que irá trabalhar com informação armazenada na Base de Dados do iPortalDoc.

O primeiro passo após o acesso à interface do WSO2 é a definição de um Datasource, ou seja de uma interface com a base de dados do iPortalDoc que irá servir como fonte de dados para todos os Data Services que serão definidos para este Workflow.

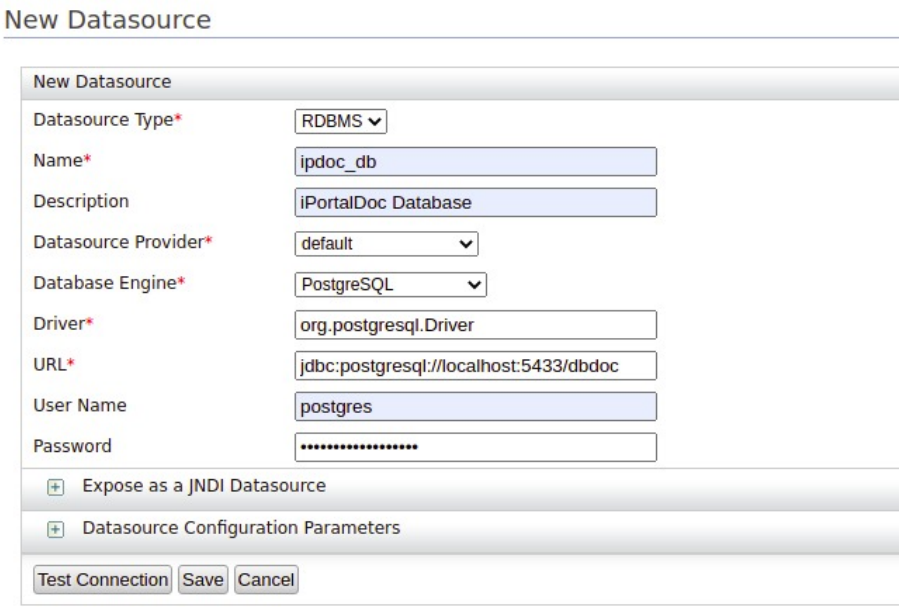

Figure 5.6: Definição da Base de Dados do iPortalDoc num Datasource.

Após a definição do Datasource é possivel prosseguir para a criação dos Data Services, começando pela definição do nome do Data Service e uma descrição que permita a compreenção do objetivo da Data Service.

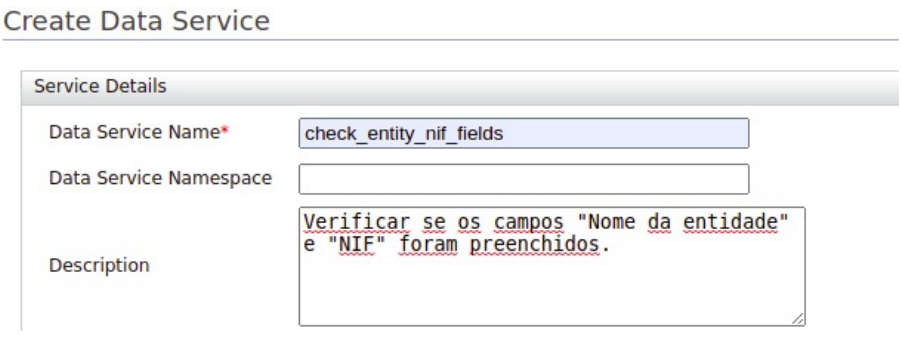

Figure 5.7: Definição das informações do Data Service.

É necessário agora associar uma ou mais Datasources ao Data Service, de maneira a que ele possua fontes de informação para poder realizar operações sobre a mesma.

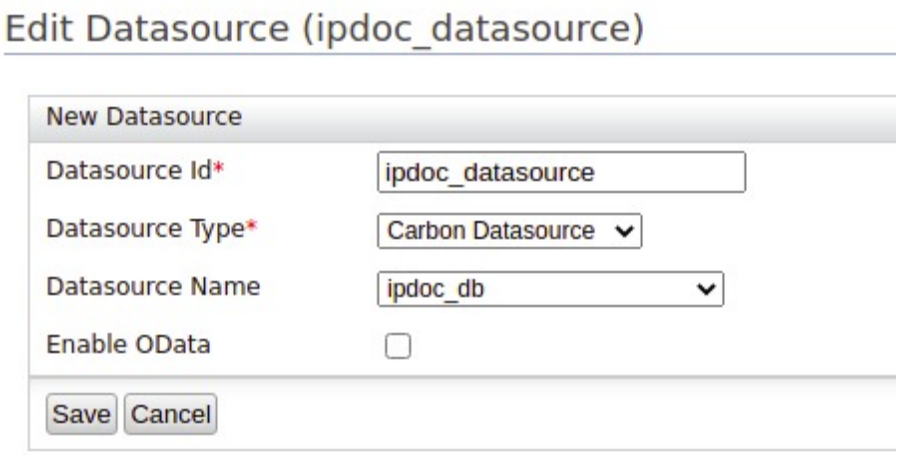

Figure 5.8: Associação da Datasource ao Data Service.

O passo seguinte é definir a Query em SQL, que define a operação que se pretende executar sobre a informação de uma Datasource. Para além disso deve haver um mapeamento dos inputs e um mapeamento da resposta que se pretende obter com a invocação deste serviço.

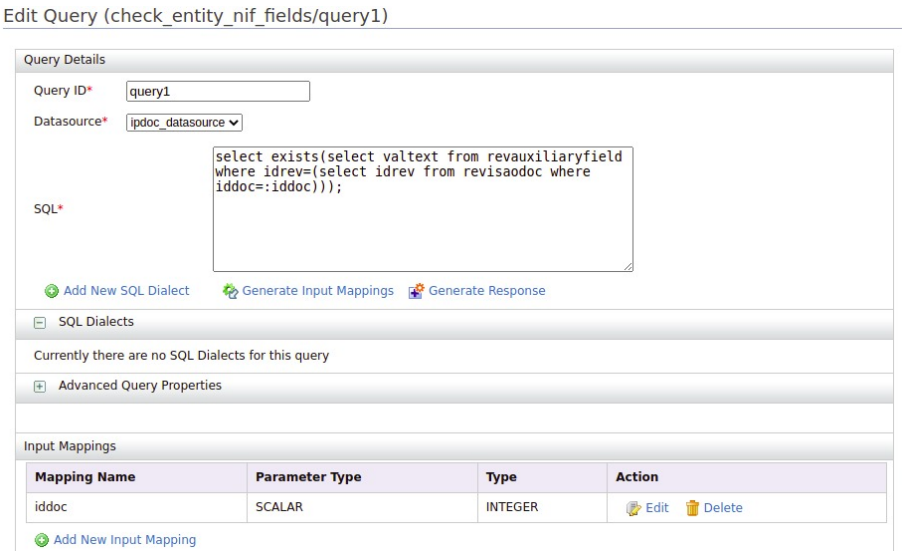

Figure 5.9: Definição da Query SQL e mapeamento do input.

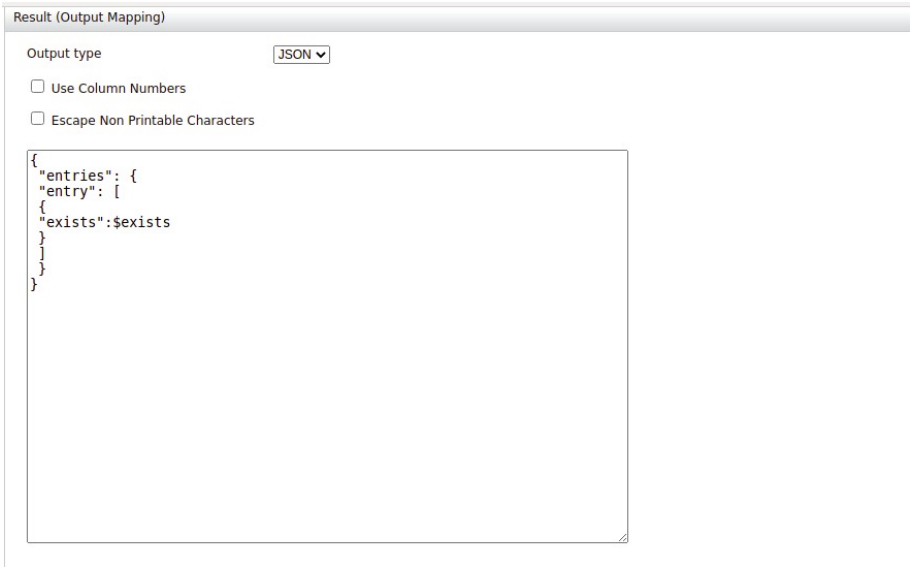

Figure 5.10: Mapeamento do output.

O passo final da definição do Data Service passa por definir o recurso de acesso ao Serviço para posteriormente ser possivel invocar o mesmo através de um Web Service REST.

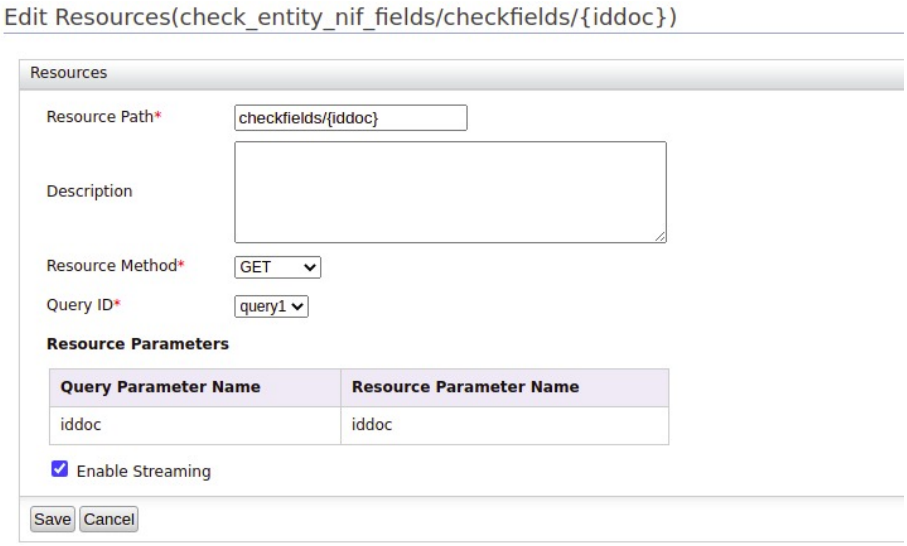

Figure 5.11: Definição do Recurso REST para invocação do serviço.

O parâmetro de acesso a todos os Data Services é o iddoc (identificador do documento), que é uma variável gerada pelo iPortalDoc no momento em que o documento é inserido no sistema.

Após a definição do Data Service é possivel aceder ao URL do endpoint que permite a invocação do mesmo.

Figure 5.12: Endpoint do Data Service definido.

5.4 Queries dos Data Services

Cada Data Service vai ter uma estrutura similar, apenas diferenciando-se dos outros na Query SQL. Partiremos de um exemplo em que os inputs dos campos "Nome da Entidade" e "NIF" são os apresentados na figura **??**.

Figure 5.13: Inputs durante a inserção do documento.

5.4.1 Estado S1

A ação deste estado tem o objetivo de "Verificar se o campo **Nome da Entidade** foi preenchido". O Data Service terá de aceder a duas tabelas: **revisaodoc** para obter o valor "idrev" através do "iddoc" do documento e **revauxiliaryfield** para verificar através do valor de "idrev", se os inputs foram preenchidos.

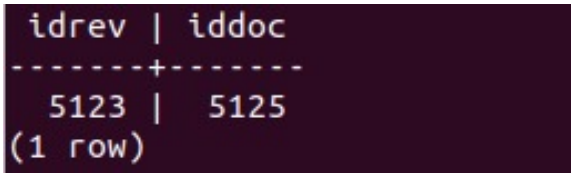

Figure 5.14: Tabela **revisaodoc**.

```
 idrev |  auxiliaryfield  |  valtext  |  valdate   | valnumeric
-------+------------------+-----------+------------+------------
  5123 | Nome da Entidade | empresa_x | 2022-07-21 |
  5123 | NIF              | 12345678  | 2022-07-21 |
(2 rows)
```

Figure 5.15: Tabela **revauxiliaryfield**.

```
select exists(select valtext from revauxiliaryfield
where idrev=(select idrev from revisaodoc where
iddoc=:iddoc)));
```

Figure 5.16: Query do estado S1.

5.4.2 Estado S2

A ação deste estado tem o objetivo de "Verificar se a entidade já se encontra registada no iPortal-Doc". O Data Service retorna o valor do Nome da Entidade da tabela **revauxiliaryfield** e verifica se ela está registada na tabela **atributo**.

```
 idatributo |     nome      |          morada          |   codpostal    |     telmovel     | numcontrib
------------+---------------+--------------------------+----------------+------------------+------------
          2 | IPBRICK, S.A. | Rua Passos Manuel, 66/76 | 4000-381*Porto | +351 221 207 102 |
          3 | foliveira     |                          |                |                  |
(2 rows)
```

Figure 5.17: Tabela **atributo**.

```
select exists(select * from atributo where nome=
(select valtext from revauxiliaryfield where idrev=
(select idrev from revisaodoc where iddoc=:iddoc)
AND auxiliaryfield='Nome da Entidade'));
```

Figure 5.18: Query do estado S2.

5.4.3 Estado S3

A ação deste estado tem o objetivo de "Atualizar o NIF da Entidade". O Data Service retorna o valor do NIF da tabela **revauxiliaryfield** e atualiza o valor na tabela **atributo**. Este caso apenas se aplica quando a Entidade já se encontra registada.

```
update atributo set numcontrib=(select valtext from
revauxiliaryfield where idrev=(select idrev from
revisaodoc where iddoc=:iddoc) AND
auxiliaryfield='NIF') where nome=(select valtext
from revauxiliaryfield where idrev=(select idrev
from revisaodoc where iddoc=:iddoc) AND
auxiliaryfield='Nome da Entidade');
```

Figure 5.19: Query do estado S3.

5.4.4 Estado S4

A ação deste estado tem o objetivo de "Registar a Entidade nova no iPortalDoc". O Data Service irá inserir as informações da Entidade na tabela **atributo**.

```
insert into atributo values ((select
max(idatributo) from idatributo)+1, (select valtext
from revauxiliaryfield where idrev=(select idrev
from revisaodoc where iddoc=:iddoc) AND
auxiliaryfield='Nome da Entidade'), null, null,
null, (select valtext from revauxiliaryfield where
idrev=(select idrev from revisaodoc where
iddoc=:iddoc) AND auxiliaryfield='NIF');
```

Figure 5.20: Query do estado S4.

5.4.5 Estado S5

A ação deste estado tem o objetivo de "Obter o ID Atributo da Entidade registada". O Data Service obtém o "idatributo" da tabela **atributo**.

```
select idatributo from atributo where nome=(select
valtext from revauxiliaryfield where idrev=(select
idrev from revisaodoc where iddoc=:iddoc) AND
auxiliaryfield='Nome da Entidade');
```

Figure 5.21: Query do estado S5.

5.5 Acesso aos Data Services pelo iPortalDoc

De modo a que os Data Services definidos possam ser acedidos pelo Workflow do iPortalDoc é necessário defini-los como ações do iPortalDoc que serão associadas ao respetivo estado do Workflow da figura 5.4.

É possivel invocar cada serviço, como um Web Service REST e o iPortalDoc tem uma tarefa que permite a invocação de Web Services REST, portanto tudo o que é necessário é fazer a configuração correta da tarefa para cada ação. A figura **??** exemplifica a configuração da tarefa para o estado S1.

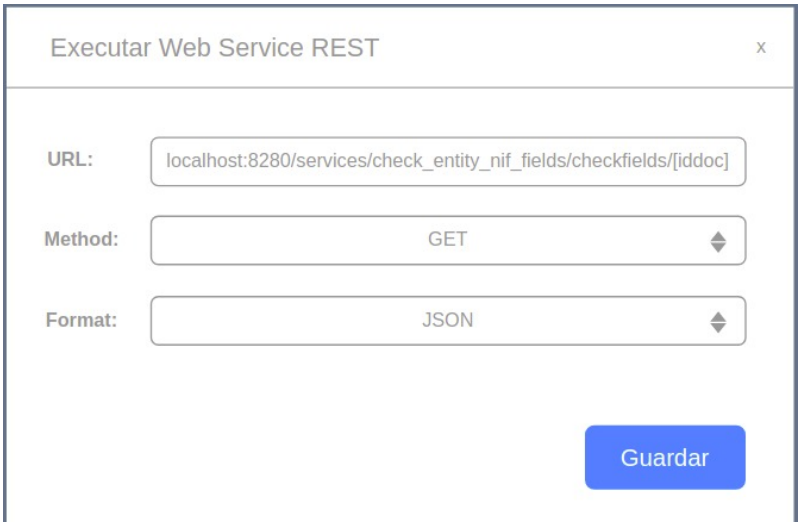

Figure 5.22: Configuração do Web Service REST para o estado S1.

5.5.1 Resultado

Após a execução deste Worflow, a entidade nova é adicionada ao Contacts.

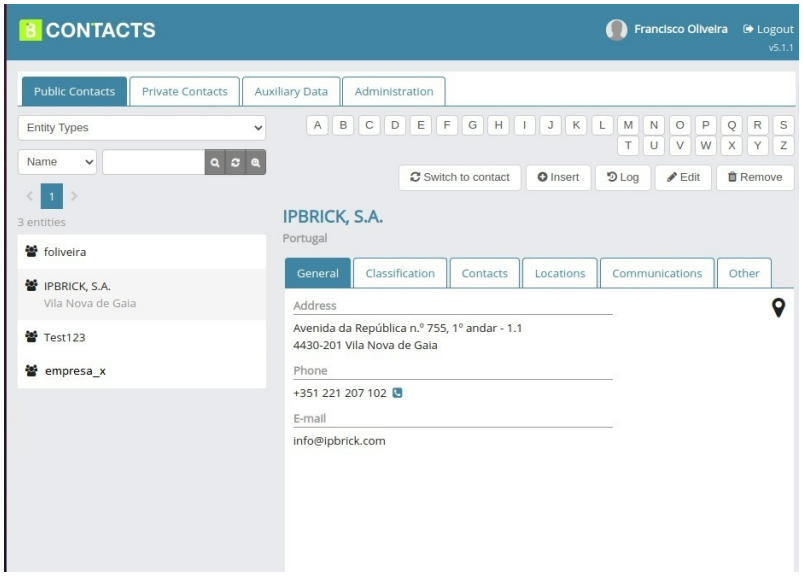

Figure 5.23: Entidade 'empresa_x' adicionada ao Contacts.

5.6 Diagrama de Use Case

O processo de acesso ao ESB por um utilizador e criação deste caso de estudo pode ser representado pelo seguinte diagrama de Use Case.

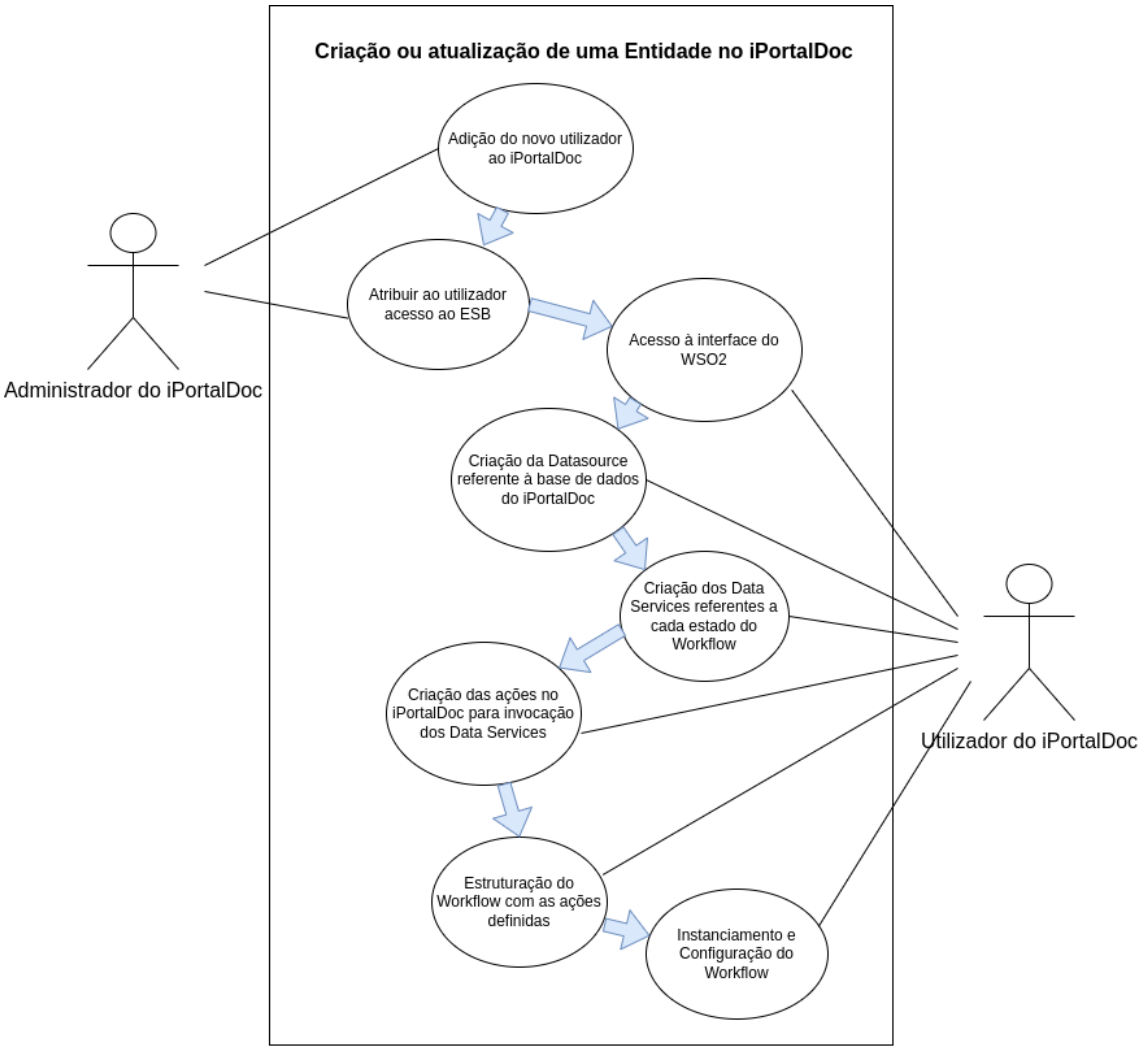

Figure 5.24: Diagrama de Use Case.

A alternativa para esta implementação seria criar tarefas no iPortalDoc no código fonte, onde seria necessário criar um módulo para estabelecer a conexão à base de dados e outros módulos para ler as Queries, validá-las e executá-las. Todo este processo seria extremamente trabalhoso e implicaria que essa implementação de código fosse apenas possivel de executar pelos desenvolvedores do iPortalDoc. Desta maneira os utilizadores do iPortalDoc não poderiam realizar a sua implementação personalizada.

Chapter 6

Conclusões e Trabalho Futuro

6.1 Satisfação dos Objetivos

Este projeto, visa o uso de um ESB como motor de configuração de serviços a serem utilizados na plataforma iPortalDoc. Os objetivos propostos foram concluidos com sucesso. Foi criado um mecanismo que permite a gestão dos utilizadores do ESB a partir do iPortalDoc. Foi também criada uma interface gráfica no iPortalDoc para a fácil gestão pelo administrador, assim como o software que permite a interação entre a interface e o backend correspondente. Para além disso, o ESB foi integrado com sucesso no Menu de Aplicações do iPortalDoc e pode ser acedido diretamente, devido ao mecanismo de login automático implementado.

O trabalho foi integrado na plataforma iPortalDoc da IPBRICK e vai permitir assim, que os utilizadores do iPortalDoc possuam uma ferramenta que lhes permite uma configuração mais simples de novos métodos de comunicação que podem ser usados nos seus fluxos de trabalho. Também permite uma certa independencia no que toca ao uso da plataforma pois com o uso do ESB conseguem como que adicionar novas funcionalidades ao iPortalDoc sem ter acesso ao código fonte que apenas se encontra disponivel para os desenvolvedores.

6.2 Trabalho Futuro

O trabalho futuro no que diz respeito a este projeto passa por criar novos casos de estudo que envolvam protocolos diferentes e outros métodos de comunicação que não foram explorados no decorrer do projeto, como a aplicação de FTP e protocolos relacionados com email ou com filas de mensagens.

References

[1] Nunzio D'Agostino. *Enterprise Service Bus: a business case study*. PhD thesis, Politecnico di Torino, 2018.

[2] Omer Aziz, Muhammad Shoaib Farooq, Adnan Abid, Rubab Saher, and Naeem Aslam. Research trends in enterprise service bus (esb) applications: a systematic mapping study. *IEEE Access*, 8:31180–31197, 2020.

[3] Robin Singh Bhadoria, Narendra S Chaudhari, and Geetam Singh Tomar. The performance metric for enterprise service bus (esb) in soa system: Theoretical underpinnings and empirical illustrations for information processing. *Information Systems*, 65:158–171, 2017.

[4] Robin Singh Bhadoria, Narendra S Chaudhari, and VG Tharinda Nishantha Vidanagama. Analyzing the role of interfaces in enterprise service bus: a middleware epitome for service-oriented systems. *Computer Standards & Interfaces*, 55:146–155, 2018.

[5] Service-oriented architecture, Aug 2020. URL: https://pt.wikipedia.org/wiki/Service-oriented_architecture.

[6] Kathryn B Laskey and Kenneth Laskey. Service oriented architecture. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):101–105, 2009.

[7] what is service-oriented architecture (soa)? URL: https://www.visual-paradigm.com/guide/development/what-is-service-oriented-architecture/.

[8] Service-oriented architecture (soa) definition. URL: https://www.service-architecture.com/articles/web-services/service-oriented-architecture-soa-definition.html.

[9] Naghmeh Niknejad, Waidah Ismail, Imran Ghani, Behzad Nazari, Mahadi Bahari, et al. Understanding service-oriented architecture (soa): A systematic literature review and directions for further investigation. *Information Systems*, 91:101491, 2020.

[10] Service-oriented architecture, Dec 2020. URL: https://www.geeksforgeeks.org/service-oriented-architecture/.

[11] Service-oriented architecture (soa). URL: https://www.w3schools.in/service-oriented-architecture/.

[12] Key benefits of service oriented architecture. URL: https://www.decipherzone.com/blog-detail/service-oriented-architecture.

[13] Stefan Pitulić, Slaviša Ilić, Siniša Ilić, and Dragana Radosavljević. Data exchange using wso2 enterprise service bus.

[14] Enterprise service bus, Jun 2019. URL: https://pt.wikipedia.org/wiki/Enterprise_Service_Bus.

[15] Sanjay P Ahuja and Amit Patel. Enterprise service bus: A performance evaluation. *Commun. Netw.*, 3(3):133–140, 2011.

[16] Jose Vicente Berna-Martinez, Claudia Ivette Castro Zamora, Francisco Maciá Pérez, Carlos Ramón López Paz, et al. Method for the integration of applications based on enterprise service bus technologies. 2018.

[17] Choosing the right esb for your integration needs. URL: https://www.infoq.com/articles/ESB-Integration/.

[18] What is wso2 esb? URL: https://wso2.com/library/articles/2017/07/what-is-wso2-esb/.

[19] O que é o mule esb? - português. URL: https://www.mulesoft.com/pt/resources/esb/what-mule-esb.

[20] apache synapse - the lightweight esb. URL: https://synapse.apache.org/.

[21] Apache servicemix. URL: https://servicemix.apache.org/.

[22] The apache open source esb. URL: https://dzone.com/refcardz/servicemix.

[23] Getting started with the management console. URL: https://docs.wso2.com/display/IS530/Getting+Started+with+the+Management+Console.

[24] Mule management console. URL: https://docs.mulesoft.com/mule-management-console/3.8/.

[25] Apache servicemix web console. URL: https://servicemix.apache.org/docs/6.x/users-guide/web-console.html.

[26] iportaldoc. URL: https://www.iportaldoc.pt/pt-pt/.

[27] Funcionalidades do iportaldoc. URL: https://www.iportaldoc.pt/pt-pt/funcionalidades/.

[28] IPBRICK International. *iPortalDoc - Gestor Documental - Manual de Utilizador}. 2012.*

[29] Ipbrick.mail. URL: https://www.ipbrick.com/pt-pt/ipbrick-mail-email-ferramentas-colaborativas.

[30] Ipbrick.cafe. URL: https://www.ipbrick.com/pt-pt/ipbrick-cafe-digital-workplace.

[31] Ipbrick.ucoip. URL: https://www.ipbrick.com/pt-pt/ipbrick-ucoip-comunicacoes-unificadas.