

Segurança em IoT - Caso de estudo FIWARE

Rodrigo Lopes Ananias

Mestrado em Segurança Informática

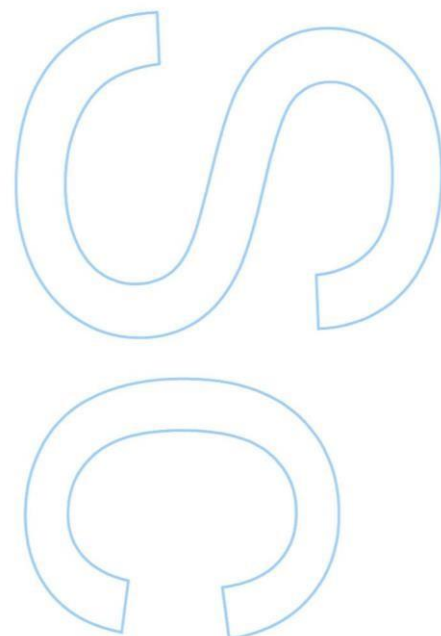
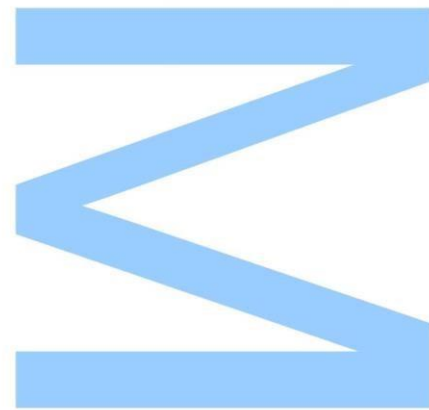
Faculdade de Ciências da Universidade do Porto 2022

Orientador

João Soares, Professor Auxiliar, FCUP

Coorientador

Rolando Martins, Professor Auxiliar, FCUP

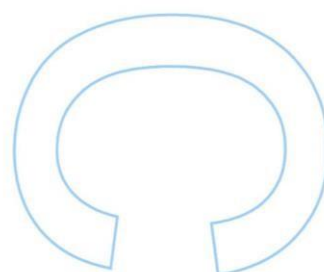
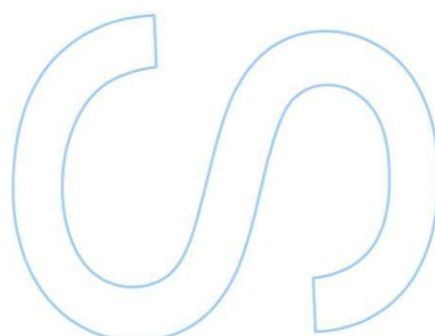
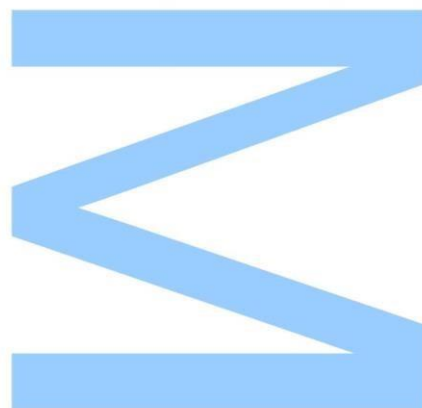




Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ____/____/____



Dedico esta tese aos meus amados pais, irmã, esposa e filho.

Declaração de Honra

Eu, Rodrigo Lopes Ananias, inscrito(a) no Mestrado em Segurança Informática da Faculdade de Ciências da Universidade do Porto declaro, nos termos do disposto na alínea a) do artigo 14.º do Código Ético de Conduta Académica da U.Porto, que o conteúdo da presente dissertação/relatório de estágio/projeto [indicar o aplicável] reflete as perspetivas, o trabalho de investigação e as minhas interpretações no momento da sua entrega.

Ao entregar esta dissertação, declaro, ainda, que a mesma é resultado do meu próprio trabalho de investigação e contém contributos que não foram utilizados previamente noutros trabalhos apresentados a esta ou outra instituição.

Mais declaro que todas as referências a outros autores respeitam escrupulosamente as regras da atribuição, encontrando-se devidamente citadas no corpo do texto e identificadas na secção de referências bibliográficas. Não são divulgados na presente dissertação quaisquer conteúdos cuja reprodução esteja vedada por direitos de autor.

Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Rodrigo Lopes Ananias

Porto, 30 de setembro de 2022.

Agradecimentos

Obrigado a todos que contribuíram para o desenvolvimento desta tese, tanto de forma direta, como indireta. Aos meus pais por sempre me incentivarem a prosseguir com os estudos e realizar os meus sonhos. A minha esposa e filho por viverem este momento comigo. Aos meus amigos com quem partilho a vida desde a adolescência e aos amigos que fiz durante esta jornada académica em terras portuguesas que certamente já fazem parte da minha vida. Aos meus colegas de trabalho no banco Itaú por me ajudar e incentivar antes e durante a realização deste curso. Ao meu orientador Professor João Soares, pela sua dedicação e compreensão nos momentos em que tive de conciliar família trabalho e universidade. Ao Professor Manuel Barbosa, pelo acolhimento na universidade que foi o primeiro passo para chegar até este momento. Muito obrigado.

Abstract

In the world of IoT it is possible to find several innovative solutions that can be incorporated into our daily lives. These solutions, being used in isolation or as part of an ecosystem, need to ensure information security, since their users share information that should not be used for purposes other than those agreed upon, from the perspective of supplier and user. The present work aims to study an IoT platform, Fiware, to understand how information security is treated with a focus on the entities and interactions of the platform, through an analysis of adherence to a set of information security principles (Control access, Identity, Authentication, Authorization, Responsibility, Auditability, Integrity, Authenticity, Confidentiality, Privacy, Non-Repudiation, Availability) among which I highlight "Responsibility and Auditability" as important criteria in choosing an IoT solution since they are directly linked with legal and regulatory aspects, identify possible GAPS, propose recommendations to improve the platform accompanied by a trial of the proposed solution for some of the themes identified throughout this dissertation.

Keywords: Authenticity; Integrity, Auditability, Accountability, IoT.

Resumo

No mundo de IoT é possível encontrar diversas soluções inovadoras que podem ser incorporadas no nosso dia-a-dia. Estas soluções sendo utilizadas de forma isolada ou como parte de um ecossistema, precisam zelar pela segurança da informação, uma vez que os seus utilizadores partilham informações que não devem ser utilizadas para outros fins que não sejam os acordados, na ótica de fornecedor e utilizador. O presente trabalho tem como objetivo o estudo de uma plataforma de IoT, Fiware, perceber como a segurança da informação é tratada com foco nas entidades e interações da plataforma, através de uma análise de aderência a um conjunto de princípios de segurança da informação (Controle de acesso, Identidade, Autenticação, Autorização, Responsabilidade, Auditabilidade, Integridade, Autenticidade, Confidencialidade, Privacidade, Não Repúdio, Disponibilidade) dentre os quais destaco a "Responsabilidade e Auditabilidade" como critérios importantes na escolha de uma solução de IoT uma vez que estão ligados diretamente com aspetos legais e regulatórios, identificar possíveis GAPS, propor recomendações para melhorar a plataforma acompanhada de uma experimentação da solução proposta para alguns dos temas identificados ao longo desta dissertação.

Palavras chaves: Responsabilidade; Auditabilidade; Autenticidade; Integridade; IoT.

Conteúdo

Agradecimentos	i
Abstract	iii
Resumo	v
Conteúdo	x
Lista de Tabelas	xi
Lista de Figuras	xv
Acrónimos	xvii
1 Introdução	1
1.1 Contexto	1
1.2 Metodologia	2
1.3 Objetivos	2
1.4 Organização	2
2 Ecosistema FIWARE	3
2.1 FIWARE	3
2.1.1 IoT Device	4
2.1.2 IoT Agent	4
2.1.3 Context Brokers	5

2.1.4	Fluxos de Comunicação	7
2.1.5	FogFlow	8
2.1.5.1	Entidade	9
2.1.5.2	Operador (tipo)	9
2.1.5.3	Task (Estrutura)	10
2.1.5.4	Task Instance	11
2.1.5.5	Service Template	11
2.1.5.6	Fog Function (Conceito)	11
2.1.5.7	Service Topology (Conceito)	13
3	Estado da Arte	15
4	Segurança da plataforma FIWARE	19
4.1	IDM - Keyrock	19
4.1.1	Funcionamento	23
4.1.1.1	Autenticação	23
4.1.1.2	Autorização	25
4.2	AuthZForce	25
4.2.1	Funcionamento	27
4.3	PEP Proxy - Wilma	29
4.3.1	Funcionamento	30
4.4	Usage Control	33
4.4.1	Funcionamento	35
5	Análise de segurança FIWARE	37
5.1	Escolha da Ferramenta	37
5.1.1	Arquitetura de Segurança	38
5.2	Métricas de Avaliação	39
5.2.1	IDM - Keyrock	41

5.2.2	Wilma - PEP Proxy	43
5.2.3	Authzforce	45
5.2.4	Arquitetura de Segurança FIWARE	47
5.3	Discussão	49
6	Proposta de solução	51
6.1	Propostas	52
6.2	Arquitetura	53
6.3	Protótipo	54
6.3.1	Desenho da Solução e Fluxos	55
6.4	Implementação	58
7	Experimentação e Testes	63
7.1	Ambiente	63
7.2	Testes	64
7.2.1	Resultados	65
8	Conclusão	69
8.1	Trabalho Futuro	70
A	Apêndice-a	71
A.1	Funcionamento Fogflow	71
A.1.1	Componentes	71
A.1.2	Orquestração	73
A.1.3	Operacionalização (Como funciona)	74
A.1.3.1	Função de Fog	75
A.1.3.2	Service Management - Service Topology	76
A.1.3.3	Context Management	76
A.1.3.4	Data Processing - Worker	78

Lista de Tabelas

4.1	Componentes na arquitetura XACML	27
5.1	Matriz de Aderência aos critérios de segurança - IDM Keyrock.	43
5.2	Matriz de Aderência aos critérios de segurança - Wilma PEP Proxy.	45
5.3	Matriz de Aderência aos critérios de segurança - Authzforce PDP.	47
5.4	Matriz de Aderência aos critérios de segurança - Componentes FIWARE	49
A.1	Componentes do FogFlow	72

Lista de Figuras

2.1	Framework FIWARE (fonte:[10])	4
2.2	FIWARE Entidades (fonte:[10])	6
2.3	FIWARE Entidades Chave estrangeiras (fonte:[10])	6
2.4	FIWARE - Northbound (fonte:[11])	7
2.5	FIWARE - Southbound (fonte:[11])	8
2.6	FogFlow benefícios (fonte:[10])	8
2.7	FogFlow visão alto nível (fonte:[10])	10
2.8	FogFlow função de fog (fonte:[10])	12
2.9	FogFlow serviço de topologia (fonte:[10])	13
2.10	FogFlow serviço de topologia - objeto de itent (fonte:[9])	14
4.1	Keyrock Integração eIDAS (fonte:[12])	20
4.2	Keyrock Integração 2 (fonte:[12])	21
4.3	Keyrock fluxo (fonte:[13])	22
4.4	Keyrock Autenticação (fonte:[13])	24
4.5	Keyrock Autenticação organização (fonte:[13])	24
4.6	Keyrock Autorização (fonte:[13])	25
4.7	Keyrock Autorização Organização (fonte:[13])	26
4.8	Arquitetura XACML (fonte:[19])	26
4.9	AuthZForce Pedido (fonte:[19])	28
4.10	AuthZForce Retorno (fonte:[19])	29

4.11	Exemplo (fonte:[1])	29
4.12	PEP Wilma com Nginx (fonte:[14])	31
4.13	PEP Wilma Nivel 1 Autenticação (fonte:[14])	31
4.14	PEP Wilma Nivel 2 (fonte:[14])	32
4.15	PEP Wilma Nivel 3 (fonte:[14])	32
4.16	Modelo Usage Control (fonte:[1])	34
4.17	Modelo Usage Control (fonte:[1])	35
4.18	Keyrock Usage Control (fonte:[1])	36
4.19	Dashboard Usage Control (fonte:[1])	36
5.1	FogFlow Https (fonte:[12])	38
5.2	FogFlow fluxo com ferramentas de segurança (fonte:[12])	39
6.1	Arquitetura de segurança recomenda para o Fogflow - fonte:[15]	54
6.2	Arquitetura proposta	54
6.3	Prototipo - Exemplo de uma Aplicação	56
6.4	Prototipo	56
6.5	Prototipo	57
6.6	Prototipo	57
6.7	Prototipo	58
6.8	PEP PROXY WILMA - config.js	58
6.9	Exemplo Log - ECS	59
6.10	PEP PROXY WILMA - includes app.js	60
6.11	PEP PROXY WILMA - logs app.js	60
6.12	PEP PROXY WILMA - logs ecs app.js	61
6.13	PEP PROXY WILMA - Final	61
7.1	Resultados Testes NOK	66
7.2	Resultado Testes OK	67

7.3	Gráfico Comparativo	67
A.1	FogFlow Arquitetura (fonte:[10])	71
A.2	FogFlow deploy (fonte:[11])	73
A.3	FogFlow Funcionamento (fonte:[8])	74
A.4	FogFlow Função de Fog (fonte:[11])	76
A.5	FogFlow Servico de Topologia (fonte:[11])	77
A.6	FogFlow IOT Discovery (fonte:[11])	77
A.7	FogFlow IOT Worker (fonte:[11])	79

Acrónimos

API	Application Programming Interface	OASIS	Organization for the Advancement of Structured Information Standards
IoT	Internet of Things	PEP	Policy Enforcement Point
NGSI	Next Generation Service Interface	ODRL	Open Digital Rights Language
NGSI10	the context data API of FIWARE NGSI	CEP	Complex Event Process
NGSI9	the context data API of FIWARE NGSI	WSN	wireless sensor network
IDM	Identity management	URN	Uniform Resource Name
PDP	Policy Decision Point	MAC	Message authentication code
PRP	Policy Repository Point	RGPD	Regulamento Geral de Proteção de Dados
PAP	Policy Administration Point	DLT	Distributed Ledger Technology
PIP	Policy Information Point	OECD	Organization for Economic Cooperation and Development
PTP	Policy Translation Point	ECS	Elastic Common Schema
PXP	Policy Execution Point	QoS	Quality of Service
GE	(Generic Enables	AD	Active Directory
NGSI-LD	Next Generation Service Interface - Linked Data	EDPS	European Data Protection Supervisor
eIDAS	Electronic Identification, Authentication and Trust Services	HMAC	Hash-based message Authentication Code
eID	Secure electronic identification	MAC	Message Authentication Code
OTP	One Time Password	PMBOK	Project Management Body of Knowledge
XACML	eXtensible Access Control Markup Language	SO	Sistema Operacional

Capítulo 1

Introdução

1.1 Contexto

Quando falamos de Internet of Things (**IoT**) nos remete a visão do pequenos dispositivos que conhecemos e estão presentes em nosso dia-a-dia exemplo os sensores de presença e luminosidade, câmaras de segurança, relógios inteligentes dentre outros. Estes dispositivos possuem tamanhos e capacidades de processamento variadas, alguns podem ter uma unidade para armazenamento local (de acordo com o seu tamanho e utilização) e em alguns casos se pode realizar o processamento no dispositivo ou apenas o utilizar como interface para transmissão dos dados coletados.

O uso destes dispositivos bem como a sua integração em muitos equipamentos utilizados atualmente tem crescido exponencialmente [5]. Um dispositivo **IoT** pode estar ligados a um sistema, outros dispositivos e também podem fazer parte de um ecossistema maior. Cito um exemplos das cidades inteligentes, onde temos vários dispositivos a prover informações para o sistema local que realiza algum pré-processamento e ou tratamento dos dados e envia a um outro sistema, que ira novamente tratar os dados e gerar as informações e serem aplicadas a um determinado contexto pré definido.

Pode-se contextualizar um sistema de **IoT** como uma enorme rede de dispositivos conectados à internet, e estes dispositivos vão além dos que tradicionalmente conhecemos tais como sensores, câmaras, eletrodomésticos, etc. Uma vez tendo capacidade computacional, podem se conectar a rede e partilhar dados. Segundo o artigo "*Internet of Things (IoT): A vision, architectural elements, and future directions* [5]" existem alguns "padrões" para **IoT** que permite classificar os dispositivos mediante a sua utilização.

Por se tratar de uma tecnologia relativamente nova, ainda existem pontos a serem explorados, identificados e aprimorados. No que tange o armazenamento e segurança tanto para os seus utilizadores como para os dados, identifica-se alguns desafios relacionados aos sistemas centralizados vs descentralizados, auditoria e responsabilidade sobre as ações e dados dos utilizadores e implementação de políticas para o tratamento das informações que necessitam de níveis distintos de proteção [7].

Ao longo desta dissertação serão explorados com mais profundidades os temas referentes a segurança informática no âmbito de um ecossistema de **IoT**.

1.2 Metodologia

Para o desenvolvimento do trabalho utilizou-se metodologias de pesquisa e boas praticas de gestão de projetos, tendo como referencia alguns princípios do Project Management Body of Knowledge (**PMBOK**) para estruturar a organização e recolha de informações. Iniciou-se com as componentes de pesquisa bibliográficas sobre o tema e analise da documentação, seguida das componentes praticas de experimentação, testes e prototipar (com algum desenvolvimento) por fim a componente de analise e apresentação dos resultados obtidos seguida da conclusão e algumas recomendações para trabalhos futuros.

1.3 Objetivos

A presente dissertação tem como objetivo analisar o cumprimento de um conjunto de princípios de segurança da informação na arquitetura de segurança (para Internet of Things (**IoT**)) utilizada no ecossistema da FIWARE, uma plataforma de aplicações para **IoT** que será explicada ao longo deste trabalho. Dentre os contributos deste trabalho cito a analise da aderência da plataforma FIWARE aos princípios de segurança da informação, identificação de requisitos necessários para garantir Accountability e Auditability, prototipar e desenvolver solução para atendimento dos requisitos identificados.

1.4 Organização

Este documento esta organizado em quatro partes. Parte I introdução (**Capítulo 1** e **Capítulo 2**) são apresentados os conceitos básicos, *Iot*, Segurança da informação e a plataforma *FIWARE*, para o entendimento deste trabalho; Parte II Analise (**Capítulo 4** e **Capítulo 5**) onde se apresenta os componentes de segurança e o contributo do autor ao analisar a arquitetura proposta na perspetiva de segurança informática; Parte III Proposta (**Capítulo 3**, **Capítulo 6** e **Capítulo 7**), estado da arte dos requisitos não atendidos, identificados na analise, e a proposta para atender este requisitos na arquitetura proposta pela plataforma FIWARE. Parte IV Conclusão (**Capítulo 7** e **Capítulo 8**), temos o fecho com as considerações finais e endereçamento de temas a serem abordados em trabalhos futuros.

Capítulo 2

Ecosistema FIWARE

2.1 FIWARE

Parafraseando o slogan da plataforma, "FIWARE consiste num conjunto de Application Programming Interface (API)s para mentes abertas"[4]. É uma plataforma de aplicações para Internet of Things (IoT) que possui dois objetivos principais: criar as fundações para uma norma de tecnologia aberta e criar um ecossistema de inovação aberto e orientado para a tecnologia.

Resumidamente consiste em um *framework* com componentes de código aberto que podem ser montados juntos e ou com outros componentes de terceiros para construir soluções inteligentes de forma mais rápida, fácil e com custo reduzido.

Para uma solução ser considerada aderente a FIWARE, “Powered by FIWARE”, ela necessita ter o componente *Context Broker Generic Enabler* (obrigatório) que implementa as funções fundamentais como a necessidade de gerir informações de contexto, permitindo realizar atualizações e o acesso ao contexto. Tem como base uma API simples (FIWARE NGSI), que permite a integração de componentes e funciona como uma base para a interoperabilidade e replicação de soluções inteligentes.

Com a evolução da *framework* foram desenvolvidas camadas complementares, tal como representado na Figura 2.1, de modo a suportar as funcionalidades necessárias para uma plataforma IoT tais como:

- Interface to IoT, Robots and third-party systems, para capturar atualizações nas informações de contexto e traduzir as atuações necessárias.
- Data/API management, publication Monetization, trazendo suporte ao controlo do uso e a oportunidade de publicar e monetizar parte dos dados de contexto geridos.
- Processing, Analysis, Visualization, implementando o comportamento esperado dos aplicativos e / ou auxiliando os usuários finais na tomada de decisões inteligentes.

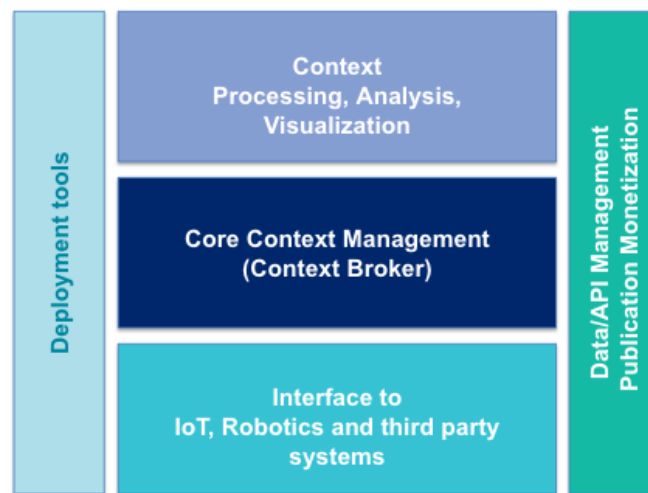


Figura 2.1: Framework FIWARE (fonte:[10])

Antes de compreender o funcionamento da plataforma, é preciso perceber alguns conceitos utilizados, dentre eles posso destacar os conceitos de **IoT Device**, **IoT Agents**, **Context Brokers** e fluxos de comunicação comentados a seguir.

2.1.1 IoT Device

Dentro da plataforma, podemos interpretar como sendo um dispositivo físico, com um identificador único que tem a capacidade de se conectar a uma rede para enviar e receber dados.

Estes dispositivos **IoT** pode ser classificado em três grupos:

Sensor, que apenas reporta os estados para o mundo utilizar estes dados em outros processos e sistemas para tomada de decisões, ex sensor de movimento;

Atuador, que simplesmente reage a comandos recebidos e executa ações pré-definidas, ex door bell;

Híbridos, que recebem e respondem a comando e reportam estado, ex Smart Lamp.

2.1.2 IoT Agent

É um componente que permite a comunicação/gerência dos dispositivos **IoT** a partir de um protocolo específico, no caso da FIWARE NGSIv1 ou NGSIv2, encapsulando questões de compatibilidade e segurança para viabilizar a ponte entre o fornecedor e o consumidor/gerenciador.

Um agente pode se comunicar com um ou mais dispositivos **IoT** trazendo como benefícios para quem o utiliza:

Padrão de comunicação com os dispositivos; Lista de dispositivos e em muitos casos um mapa de contextos e atributos; Tratamento de questões de segurança como autorizações e autenticação.

Usualmente é definida uma porta específica para interação com o consumidor/gerenciador (North Port) que vai receber e enviar dados no formato especificado.

Exemplos de agentes para consumidores/gerenciadores que usam NGSI (FIWARE):

- IoTAgent-JSON - Dispositivo usa nativo HTTP/MQTT (payload JSON)
- IoTAgent-LWM2M - Dispositivo usa nativo o protocolo Lightweight M2M e o NGSI
- IoTAgent-UL – Dispositivo usa nativo o protocolo HTTP/MQTT (payload UltraLight2.0)
- IoTAgent-LoRaWAN – Dispositivo usa nativo LoRaWAN

2.1.3 Context Brokers

Por definição, o contexto é uma informação sobre um determinado assunto, que pode ser considerada uma abstração dos dispositivos e forma de acesso aos dados. Exemplo, o ato de ter a informação da temperatura de um determinado ambiente pode ser um contexto dentro de um aplicativo ou uma situação específica.

Na FIWARE este conceito de contexto é muito utilizado pois o foco são informações de vários dispositivos diferentes ou seja, cria-se um contexto e depois temos as entidades que podem fazer parte deste contexto ou prover informações.

Existe o Consumidor de Contexto e o Provedor de contexto onde basicamente: o Provedor é quem gera a informação e o Consumidor quem utiliza esta informação ou subscreve-se para receber as atualizações de contexto. Exemplo o *backend* para atualizar os dados faz uma subscrição e assim recebe a informação e processa as suas atualizações e rotinas internas.

No âmbito da FIWARE isso pode ser percebido com a criação de entidades, atributos, relacionamentos, e fontes de dados(provedores).

Temos a Entidade como uma representação de algo físico ou um objeto lógico que vai promover o relacionamento entre as entidades. Cada entidade possui atributos que podem ser atualizados de forma estática ou dinâmica(provedor de contexto ou uma API externa) e por sua vez podem estar relacionados com outras entidades, no formato de chaves estrangeiras. As Figuras 2.2 e 2.3 apresentam uma representação do modelo de dados utilizado para definir as entidades e os seus relacionamentos o que ajuda a compreender esta ligação.

No que tange aos provedores de contexto e as atualizações dinâmicas, é preciso ter um provedor de dados (acedido de preferência via API) e registar-lo como provedor de contexto na

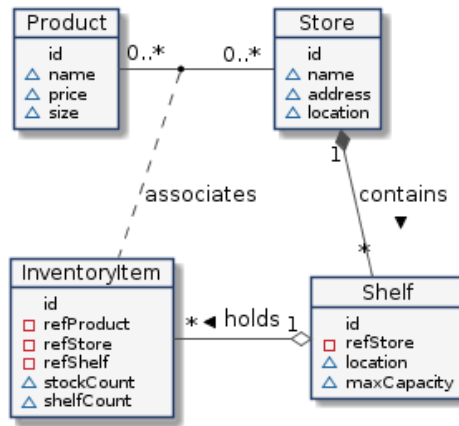


Figura 2.2: FIWARE Entidades (fonte:[10])

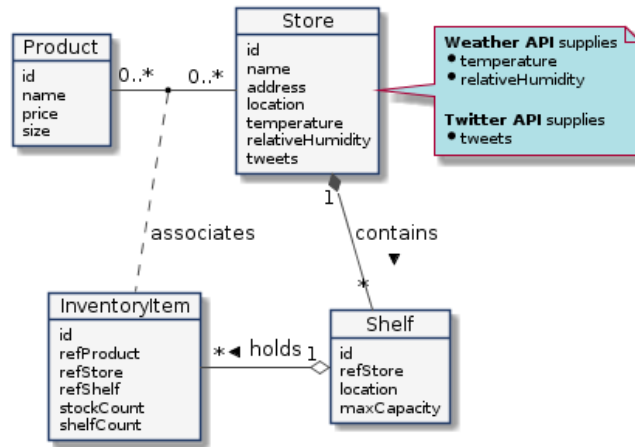


Figura 2.3: FIWARE Entidades Chave estrangeiras (fonte:[10])

plataforma para, quando for solicitada informações sobre a entidade/atributo seja realizado o acesso a este provedor para obter a informação e retornar no contexto da entidade. Muitas vezes quando o atributo tem a referencia de pertencer a uma fonte externa o seu valor não é armazenado no contexto da entidade.

Os provedores de contexto (NGSY PROXY) estão registrados em um *Broker* para recebe e enviar contexto. Quando se cria uma fonte de dados no provedor de contexto é preciso fazer o registo no Broker para ser atribuído um ID único e criar o mapeamento HTTP para aceder a informação desejada. Cada fonte de dados pode ser utilizado em uma ou mais entidades entretanto, ira retornar somente os dados para o *Broker* a qual foi cadastrada. Isso quer dizer que, se o *broker* correspondente ao NGSi PROXY estiver off, o acesso a fonte de dados também fica indisponível.

Quando tratamos de dispositivos *IoT* a serem conectados ao sistema, ao invés de utilizar o provedor de contexto utilizamos o *IoT Agent* que tem funções similares mais permite registrar uma gama maior de dispositivos e também tem a ligação a um *broker* para enviar e receber os contextos. Ele possui a capacidade de transformar os pedidos nos formatos aceito pelos

dispositivos o que permite a integração com uma variedade maior de dispositivos.

É importante ressaltar alguns pontos positivo para este tipo de sistema, com base em contexto, a subscrição e notificação. Na pratica reduz significativamente o volume de dados a serem enviados, fazendo com que seja possível criar uma *trigger* para receber a informação desejada de um determinado dispositivo e enviar uma notificação das mudanças de contexto apenas para os consumidores subscritos.

2.1.4 Fluxos de Comunicação

Pode-se definir que a comunicação entre a plataforma e os dispositivos ocorrem em dois sentidos, conceitos de comunicação **Northbound** e **Southbound**, para o fluxos de dados a serem utilizados na comunicação entre os dispositivos IoT e a plataforma FIWARE.

Northbound, Fluxo de dados do sensor para o *broker*(nó central)

Southbound, Fluxo de dados do *broker*(no central) para o sensor

O Fluxo de comunicação possuem alguns passos que podem ser visualizados nas Figuras 2.4 e 2.5 que exemplificam a comunicação do dispositivo x *broker* no contexto da FIWARE e os passos necessários.

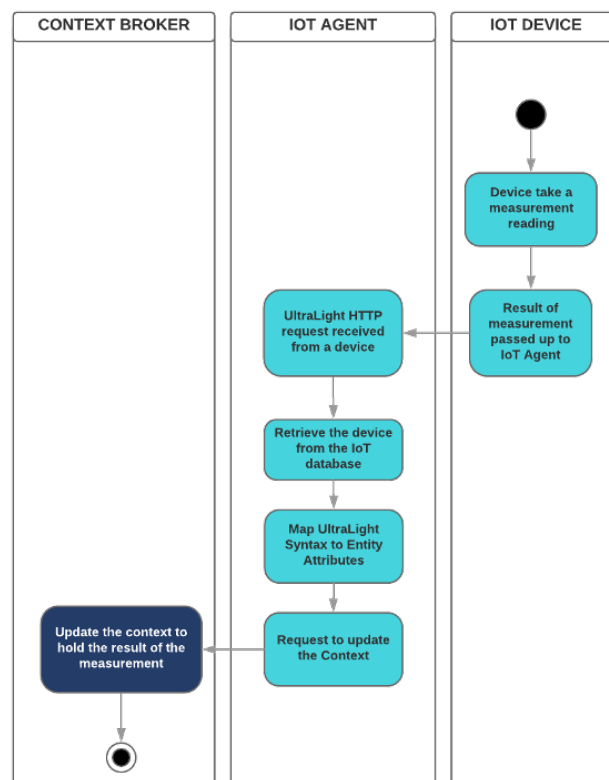


Figura 2.4: FIWARE - Northbound (fonte:[11])

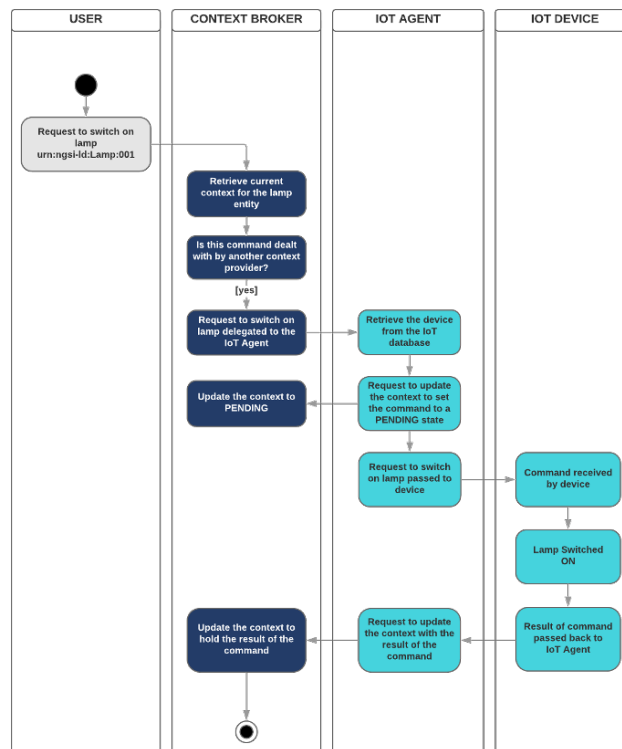


Figura 2.5: FIWARE - Southbound (fonte:[11])

2.1.5 FogFlow

O FogFlow é um componente desenvolvido com base no framework e plataforma FIWARE **orientado ao contexto**, o que significa ser capaz de orquestrar fluxos de processamento de dados dinâmicos. Fornece uma abstração e ou forma de implementar os fluxos que recebe com input, nos dispositivos IoT utilizando os padrões Next Generation Service Interface (NGSI) [10].

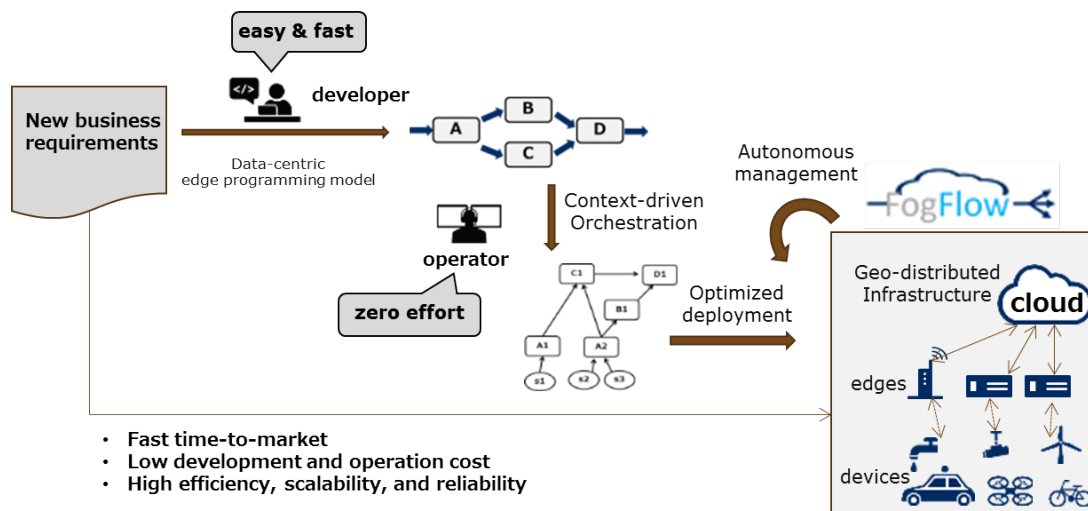


Figura 2.6: FogFlow benefícios (fonte:[10])

Destaca-se como benefícios da ferramenta a sua aplicabilidade em infraestruturas *cloud*, geo-distribuídas e edges; Possuir um formato otimizado para desenvolvimento e implementações das soluções e a sua integração nativa com outros componentes da FIWARE. Conforme ilustrado na Figura 2.6 onde podemos visualizar um modelo de aplicação que utiliza este componente.

Tem como base um modelo de dados padronizado, unificado e interface de comunicação, ou seja, **NGSI**, o FogFlow é capaz de ver o conteúdo de todos os dados gerados por sensores e tarefas de processamento de dados no sistema, tais como tipo de dados, atributos, meta-dados registados, relações e Geo-Localizações. Por exemplo, para consumidores de serviço, eles podem especificar quais tipos de resultados são esperados sob qual tipo de Quality of Service (**QoS**) dentro de qual escopo geográfico; Para provedores de dados, eles podem especificar como seus dados devem ser utilizados e por quem. As decisões de orquestração são feitas para atender aos objetivos definidos pelo utilizador durante o tempo de execução.

Por ser uma ferramenta orientada a contexto, tem em sua base três tipos de contextos principais:

- **Contexto do sistema:** recursos disponíveis que estão mudando ao longo do tempo.
Contexto dos dados: a estrutura e os meta-dados registados dos dados disponíveis, incluindo dados brutos do sensor e dados intermediários.
Contexto de uso: intenções de alto nível definidas por todos os diferentes tipos de utilizadores (desenvolvedor, consumidor de serviço, provedores de dados) para especificar o que desejam alcançar.

Estes contextos estão ilustrados na Figura 2.7 com os componentes a serem utilizados em cada contexto e também o fluxo de ligação.

Nesta sessão irei descrever alguns conceitos que julgo necessários para o entendimento desta ferramenta, todos os conceitos podem ser consultados também no website da ferramenta [9].

2.1.5.1 Entidade

O conceito de Entidade no contexto do FogFlow pode ser um dispositivo **IoT** que possui um ID único, tipo e localização e uma *task*. Pode ter uma ou mais EntityStream e estar conectado via opção de *stream* permitindo fazer mais do que um tratamento para o dado.

2.1.5.2 Operador (tipo)

Representado por uma entidade que recebe e retorna dados utilizando uma porta de comunicação e dois tipos de mensagens no padrão the context data API of FIWARE NGSI (**NGSI10**), *notify* para receber e *updates* para retornar, e tem uma porta de escuta publicada para receber as

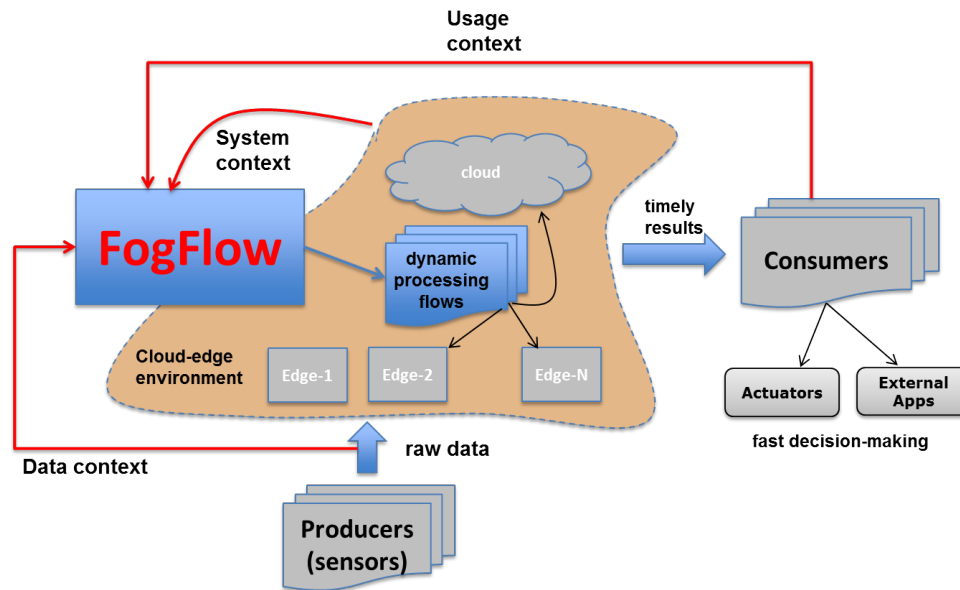


Figura 2.7: FogFlow visão alto nível (fonte:[10])

mensagens. A sua implementação é feita vinculando a entidade a uma imagem *docker*, criada é vinculada a um operador e deve ter o nome da imagem compatível com a imagem, para suportar múltiplos hardware pode estar associado a mais de uma imagem desde que implemente o mesmo processo lógico. Isso é permitido pois pode existir diferenças de hardware S.O onde o processo será executado.

Importante: Quando uma imagem é criada no FogFlow tem um parâmetro "*Prefetched*" que se estiver marcado, significa que todos os *edge node* começarão a buscar esta imagem do *docker* hub ou localmente com antecedência; Caso contrário, a imagem *docker* do operador é buscada sob demanda, apenas quando os *edge node* precisam executar uma tarefa agendada associada a este operador.

2.1.5.3 Task (Estrutura)

Representa uma tarefa dentro de uma topologia de serviço a forma como vão ser tratados/executados/geridos os pedidos e precisa estar obrigatoriamente associada a um operador e é composta por:

- Nome da task;
- Operador associado;
- GroupBy agrupamento usado para determinar quantas instâncias vai ter esta task;
- Input_streams lista de input selecionados e identificado pelo entity type;
- Output_streams lista de output gerados identificados pelo entity type;

- Shuffling (usado no input stream determinar que input stream vai para qual instância, se for Broadcast vai ser replicado para todas as instâncias e se for Unicast vai ser associado a uma única instância somente);
- Scoped determina se vai ser utilizado o factor de geo-scope para selecionar as mensagens de input.

2.1.5.4 Task Instance

É a execução de uma task com parâmetros de entrada e saída em um "*container*" específico que pode estar na cloud ou dispositivo edge de acordo com a implementação dos nós realizada.

2.1.5.5 Service Template

Cada serviço **IoT** pode ser descrito como um serviço de topologia com um conjunto de operadores vinculados ou uma função de FOG com um único operador. Quando se usa por exemplo o serviço de topologia, pode-se ter os parâmetros:

- Nome;
- Descrição;
- Priority (define o nível de prioridade de cada *task* na topologia e pode ser usado para definir qual recurso vai ser disponibilizado para *task*);
- Resource usage (se o recurso é exclusivo ou não para as *task*).

2.1.5.6 Fog Function (Conceito)

É uma função que possibilita o tratamento e ou envio de informações a um determinado dispositivo, pode ser construída na interface do FogFlow ou via APIs. O código da função devem ser escrito em JAVA/Python/Javascript e disponibilizado dentro de um operador, imagem *docker*, registado. É necessário definir o dispositivo, função, tarefa e o controlador pois, quando o dispositivo estiver ativo, com estes parâmetros sabe-se os dados serem tratados na função de FOG.

Pré-requisito para criar uma fog-function no contexto do FogFlow é a existência e ou criação de um operador e a sua associação a pelo menos uma imagem *Docker* que ira conter a função a ser executada. Isso porque o FogFlow no momento de execução da função pode selecionar a imagem mais adequada para o nó selecionado, exemplo ter imagens adaptadas a um Raspberry e a um Arduino ou Linux e uma outra versão de SO suportado pelo nó.

A função de fog, pode ser comparada a uma função normal com parâmetros de input e output definidos que esta contida dentro de um *container* (operador). No FogFlow é definida com o conceito de Topologia e possui:

- **TASK** que permiti especificar o nome, local da função (qual operador/*container* esta a função) e o output e vai estar ligado com o uma ou mais Entity Stream.
- **Entity Stream**, que contem a definição do Tipo de fluxo (dispositivo) que ira acionar a função, os atributos que serão repassados, o agrupamento para saber quantas instâncias serão ativadas para cada tipo de mensagem e um *SCOPED* que defini se os dados tem uma localização especifica ou não. Quando não se tem *SCOPED*, é enviada para todos os fluxo deste tipo.

A Figura 2.7 ilustra a criação de uma função de Fog via editor.

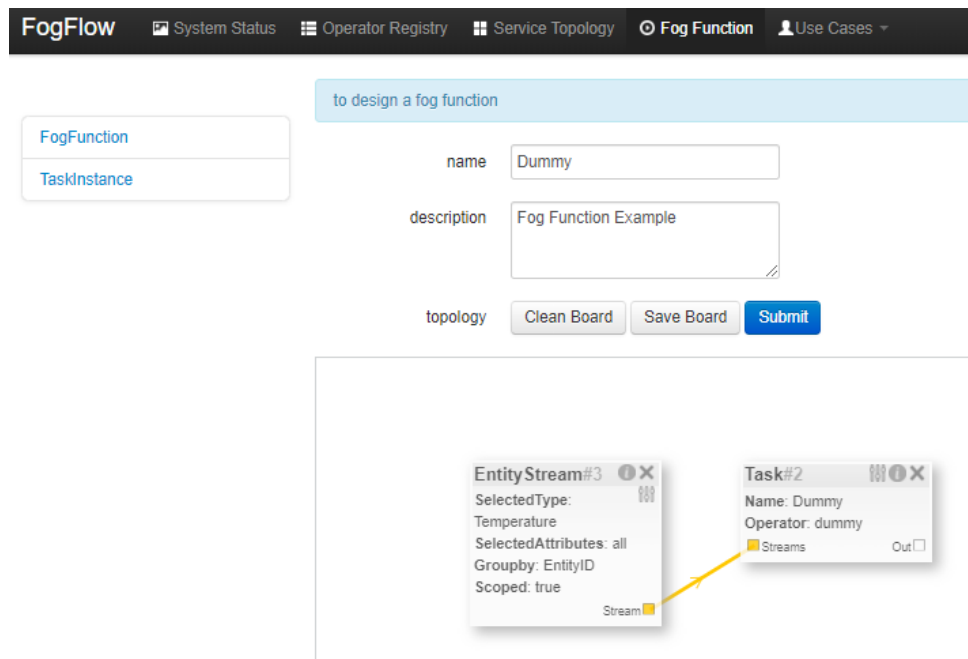


Figura 2.8: FogFlow função de fog (fonte:[10])

O FogFlow permite que o utilizador defina a sua própria função de Fog, dentro de um operador, deste que atenda os requisitos para que se consiga orquestra-la. Os parâmetros de entrada de uma função de fog são predefinidos e fixos, incluindo:

- **ContextEntity**: representando os dados da entidade recebidos;
- **Publisher** : a função de retorno de chamada para publicar o resultado gerado de volta para o sistema;
- **Query** : opcional, isso é usado apenas quando sua própria lógica de função interna precisa consultar alguns dados de entidade externa do sistema de gerenciamento de contexto;
- **subscribe** : opcional, isso é usado apenas quando sua própria lógica de função interna precisa inscrever alguns dados de entidade externa ao sistema de gerenciamento de contexto.

2.1.5.7 Service Topology (Conceito)

Uma topologia de serviço permite ao utilizador implementar um fluxo de execução mais complexo, e poder aciona-lo sob demanda com o uso de um objeto customizado, “intent”. Tal como uma função de fog, necessita de se criar um operador, vincular a uma ou mais imagens *docker*, definir task e entity streams.

Pré-requisito para criar um serviço de topologia no contexto do FogFlow é a existência dos operadores e a sua associação a pelo menos uma imagem *docker* que ira conter a função a ser executada. Isso porque o FogFlow no momento de execução da função pode selecionar a imagem mais adequada para o nó selecionado, exemplo ter imagens adaptadas a um Raspberry e a um Arduino ou Linux e uma outra versão de SO suportado pelo nó.

A Figura 2.9 apresenta um exemplo de serviço de topologia no editor do FogFlow.

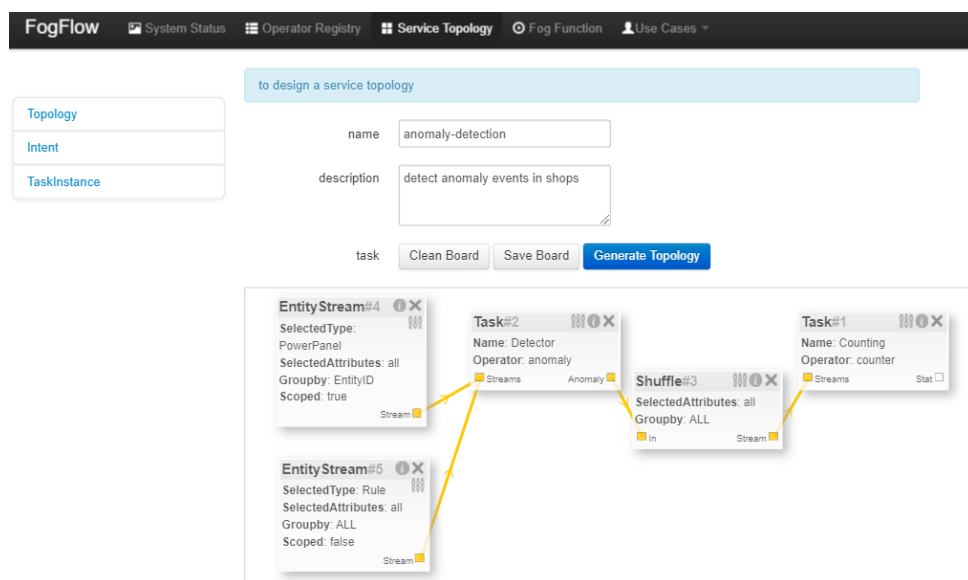


Figura 2.9: FogFlow serviço de topologia (fonte:[10])

introduz como diferencial o objeto *Shuffle*, o seu acionamento via **Intent Object**.

Suffle: Serve de elo de ligação entre as task, possibilita a conexão entre duas tasks utilizando o output de uma como input e redireciona para ser o input da outra.

Intent object: Um conjunto de parâmetros para ser iniciada a *service topology*, tais como:

- Topology: Nome da topologia a ser acionada;
- Priority: define a prioridade de cada *task* e será utilizado para designar recursos para cada *task*;
- Resource Usage: define como a topologia ira utilizar os recursos, partilhado com outras topologia ou exclusivo da topologia;
- Objective: define o throughput, latency e cost da task associada ao worker, este

- parâmetro não está ativo;
- Geoscope: define a geo-localização onde os dados para input devem ser selecionados.

A Figura 2.10 ilustra a criação de um objeto de intenção

The screenshot shows the FogFlow web interface. At the top is a dark navigation bar with the 'FogFlow' logo and menu items: 'System Status', 'Operator Registry', 'Service Topology', 'Fog Function', and 'Use Cases'. On the left is a sidebar with three options: 'Topology', 'Intent' (which is highlighted in blue), and 'TaskInstance'. The main content area has a light blue header that says 'to specify an intent object in order to run your service'. Below this is a form with five dropdown menus, each with a label and a value: 'Topology' is set to 'anomaly-detection', 'Priority' is set to 'low', 'Resource usage' is set to 'inclusive', 'Objective' is set to 'None', and 'Geoscope' is set to 'global'. At the bottom of the form is a blue 'Send' button.

Figura 2.10: FogFlow serviço de topologia - objeto de itent (fonte:[9])

uma visão mais detalhada sobre o funcionamento desta aplicação pode ser encontrada no Apêndice A.

Capítulo 3

Estado da Arte

Em busca do estado da arte para o conceito de segurança em Internet of Things (IoT), encontram-se referências à segurança by design, aspectos legais de cada país, o regulamento da união europeia de proteção de dados (que foi um grande motivador para a mudança dos formatos de responsabilidade dos dispositivos), os consentimentos explícitos para uso das informações e também a sua derivação para as trilhas de auditorias.

O conceito de segurança em IoT pode ser aplicado em diversos contextos de soluções inteligentes que tem por objetivo utilizar os diversos dispositivos, que podem se conectar a uma rede para o envio de dados e troca de informações, e conseguir extrair as suas informações a medida que necessitam ser consumidas pelos diversos tipos de utilizadores. Foram criadas redes específicas para estes dispositivos, cito as wireless sensor network (WSN) compostas por hardware / comunicator stack / middleware e secure data aggregation, que realizam a gestão destes dispositivos atribuindo a eles um Uniform Resource Name (URN) que permite identifica-los e se necessário reutiliza-los em outros processos, descrito no artigo "Internet of Things (IoT): A vision, architectural elements, and future directions [5]".

Desde então estes conceitos vem sofrendo evoluções, muitas oriundas na grande adesão ao uso destes dispositivos, investimento em ecossistemas que tem como base a informação gerada a partir da agregação de dados fornecidos por vários dispositivos e.g as soluções de smartcity e não menos importante a popularização do uso da cloud que representa uma redução no investimento em infraestrutura o que proporciona realizar mais testes e coleta de informações de diversos pontos.

Ao Falar sobre a segurança da informação, temos questões de integridade, confidencialidade, responsabilidade e auditabilidade que podem ser tratadas no nível dos repositórios e ou nos sistemas que geram e tratam estes dados. Em outros ambientes já é utilizado o conceito de se gerar uma Hash a ser utilizada na validação do conteúdo transmitidos. No ecosystema de IoT pode-se replicar o mesmo conceito, conforme é abordado no artigo "Internet of Things (IoT): A vision, architectural elements, and future directions [5]" com a temática de uso do Message Authentication Code (MAC) na comunicação entre os sensores nos diversos níveis de um ecosystema.

Nas literaturas pesquisadas, no âmbito de segurança em **IoT**, encontra-se informações que ajudam a perceber alguns dos critérios destacados na análise, responsabilidade(Accountability) e auditabilidade(Auditability).

No âmbito do gerenciamento de um sistema em **IoT** accountability é um fator crucial para expor a confiança no que se refere as responsabilidade e gestão dos recursos. O artigo, Accountability in the Internet of Things [18], conceitua Accountability, como uma palavra derivada do termo em Latin "computare(calcular)", em linhas gerais como o mecanismo de responsabilização que incorpore elementos de prestação de contas direcionado para a obrigação de uma pessoa (accountable) para outra pessoa (accountee), de forma que o primeiro deve dar conta, explicar e justificar a sua ações ou decisões de forma apropriada sendo este um pré requisito para processos de legitimidade em termos de governo. Deve abranger dois objetivos principais: Promoção da compreensão pública do sistema de negócios e consequentemente manutenção da confiança no sistema; Garantia de um grau adequado de proteção do consumidor.

Podemos abstrair accountability como uma framework que possibilita responsabilizar os utilizadores sobre as suas ações e interações no ecossistema de **IoT**, tendo como base três elementos, Padrões Informações e Sanções seguindo o principio de um acordo entre as partes contendo deveres, obrigações e sanções para casos de incumprimentos. No contexto legal esta relacionados com os temas de garantir aos utilizadores os seus direitos sobre os conteúdos e também ter ferramentas para poder contestar ações realizadas pelo sistema em quaisquer situações.

No âmbito dos dados, pode-se também incluir os princípios de proteção de dados que esta presente no regulamento geral de proteção de dados Regulamento Geral de Proteção de Dados (**RGPD**) que faz referencia a accountability como uma forma de se atribuir e demonstrar responsabilidade.

O artigo, Demonstrably doing accountability in the Internet of Things [17], comenta a relação do principio accountability com o **RGPD**, sendo um mecanismo de governança utilizado para implementação das praticas e garantias de proteção de dados nos ambientes e isso foi enfatizado no European Data Protection Supervisor (**EDPS**), onde citou que "accountability não é uma medida meramente preocupada com a validação, mas é sobre liderança pro-ativa para promover uma ampla cultura de responsabilidade". O artigo defende que desde de o design da aplicação deve-se pensar nas formas de garantir e demonstrar para o seus utilizadores o que esta a ser feito e como estão a ser tratadas as informações utilizadas pelo sistemas de forma a tornar o sistema em conformidade com os requisitos legais e regulamentações vigentes. Segundo o artigo, demonstrar accountability é semelhante a contar a um história e pode ser visto operando em três níveis sequenciais. Em sua forma mais simples, onde apenas obriga uma organização a relatar suas ações; No próximo nível requer mecanismos para que essa história seja questionada inclusive pelos próprios titulares dos dados. O terceiro nível estabelece sanções para quando uma ação não acontecer em conformidade. Tendo como base o **RGPD** pode-se extrair alguns requisitos para ter accountability em **IoT**, o artigo apresenta uma compilação com sete requisitos que são bases para ter um processo de accountability:

1. Limitando os dados iniciais: Limitação da finalidade Artigo 5(1b); minimização de dados Artigo 5.º, n.º 1-C); limitação de armazenamento Artigo 5(1e)
2. Restrições à transferência internacional de dados: Dados enviados para fora da Europa com base na decisão de adequação Artigos 44 e 45; regras societárias vinculantes Artigo 47; salvaguardas apropriadas Artigo 46
3. Respondendo ao direito do espectro de controle: Direito de acesso Artigo 15; à retificação do artigo 16; para objeto Artigo 21; restringir o artigo 18; à portabilidade Artigo 20; apagar o artigo 17; fornecimento de informações cadeia (transmitindo pedidos de retificação, apagamento, restrição) Artigo 19.
4. Garantindo direitos de transparência: Transparência das informações Artigo 12; direitos ao fornecimento de informações Artigos 13 e 14; algorítmico perfil do artigo 22; manutenção de registros Artigo 30
5. Garantir a legalidade de em processamento: Legalidade com base em motivos específicos (artigo 5.º, n.º 1, e Artigo 6.º, por exemplo, cumprimento do interesse legítimo do contrato); requisitos de consentimento Artigo 4 (11), Artigo 7, Artigo 8º e Artigo 9º
6. Protegendo o armazenamento de dados e segurança: Exatidão dos dados Artigo 5.º, n.º 1, alínea d); integridade e confidencialidade Artigo 5.º, n.º 1, alínea f); notificação de violação às autoridades artigo 33.º e ao titular dos dados, artigo 34.º; segurança de processamento Artigo 32
7. Articular e respondendo às responsabilidades de processamento: Articulando responsabilidades, Impacto na proteção de dados Avaliações Artigo 35; certificações incluindo selos, marcas e organismos de certificação Artigos 42 e 43; novo códigos de conduta artigos 40 e 41 Respondendo às responsabilidades: DPO Artigos 37 e 39; Artigo 25 do DPbD

Mesmo com estes requisitos, no âmbito de dados, temos um ponto de preocupação que é não comprometer a performance na coleta e tratamento dos dados de forma que muitas soluções não podem ser utilizadas AS-IS e recomenda-se a recorrer a soluções de terceiros para visualização e demonstração dos dados, tais como dashboards, consentimentos e geração de alertas.

O princípio da transparência das informações bem como ter o fluxo de dados definido são essenciais para ter rastreabilidade uma vez que ajudam na escolha das informações a serem utilizadas. Uma visão interessante sobre o tema de Auditability é comentada no artigo "Accountability in the IoT: Systems, Law, and Ways Forward"[16] que aborda este tema e apresenta a visão, do ponto de vista da tecnologia, do que pode ser implementado e utilizado para facilmente ter os controlos necessários e formas de auditar os processos em conformidade com os contextos legais, considerando os aspectos de Governança, Responsabilidades, Privacidade, Vigilância, Segurança e Proteção. Ressalta que Accountability é um aspeto que deve receber uma atenção para garantir que as tecnologias implementadas sejam aceites, adotadas, confiáveis e adequadas ao propósito. Uma das opções que deve-se ter em consideração é que a implementação destes

conceitos deve-se maioritariamente ser feita em tempo de desenvolvimento pois temos os requisitos, solicitante e pode-se revisar e ter o software pronto já com estas garantias, entretanto se isso pode-se complicar se pensarmos no cenário de um ecossistema onde se tem o uso e integração com de componentes de diversos fornecedores. O artigo foca-se mais neste contexto onde se tem uma governança diversificada e interações dinâmicas com os dispositivos o que torna o processo de accountability um desafio complexo pois não se consegue ser tão evasivo ao nível dos componentes. O Utilizador precisa ter confiança para utilizar a ferramenta e como contrapartida a mesma deve alertá-lo como se deve utilizar e os possíveis riscos associados, este é um dos requisitos de proteção que esta no contexto de accountability e pode ser assegurados com os mecanismos disponíveis tais como aspetos legais, framework de governança e standards para IoT. Ressalta o uso de Controlos para determinar o que esta a acontecer no sistema e auditoria para tornar visível o que acontece ou aconteceu no sistema em tempo real possibilitando tomar ações e ou responsabilizar os envolvidos e cita como exemplo "Mecanismos de controle técnico (governança e responsabilidade, privacidade e vigilância e segurança e proteção) permitem ter os passos necessários para: cumprir suas responsabilidades, lidar com a segurança e questões de responder sobre uso de dados pessoais, aumentar a transparência sobre sistemas e dados que permite tomada de decisão, fornecer evidências para auxiliar nas investigações, descobrir riscos e ajudar a responsabilizar as entidades."Dentre as opções citadas para accountability destaco a Federação de dados(Arquitetura Federada), Gerência do fluxo de dados(métodos para acompanhar o fluxo do início ao fim) e Trusted Computing (mesmo não madura pode auxiliar na verificação de partes de código executadas). Destaco o uso de Blockchain ou Distributed Ledger Technology (DLT) como formas de guardar as evidências sobre as operações, fluxos e interações, Smart Contracts (que não gerenciam apenas contratos e podem ter os respectivos logs e registros para reconstruir e auditar).

Segundo o artigo Accountability in the Internet of Things (IoT): Systems, law ways forward [18] "Accountability pode se entender como governança e responsabilidade, privacidade e monitoria, segurança e proteção". Pode-se interpretar o conceito de Auditability como a importância de ter logs das falhas para apreender com eles. Estes conceitos e critérios estão presentes em muitos cenários como, os meios de pagamentos, uso de blockchain, smart contracts entre outros que podem ser perfeitamente utilizados para no âmbito de IoT.

À data de hoje, tenho como referencia para o tema de IoT a FIWARE que possui um grande acervo de ferramentas, documentações e fóruns para soluções de Smartcity que estão a serem utilizadas como base para muitos estudos sobre o tema. Tem focado em projetar à escala global a divulgação da proposta e soluções que já estão implementadas em algumas cidades europeias. O artigo "FIWARE: uma plataforma de desenvolvimento de soluções inteligentes"[2] apresenta uma visão geral bem como as, missões e pilares desta organização que tem auxiliado no desenvolvimento das soluções IoT.

Capítulo 4

Segurança da plataforma FIWARE

Neste capítulo serão apresentados, alguns conceitos para uma plataforma de Internet of Things (IoT) ter os requisitos mínimos de segurança no âmbito dos dados e interação dos seus utilizadores. Estes conceitos estão incorporados em ferramentas que cobrem alguns requisitos de segurança permitindo tornar protegida e segura a comunicação e utilização face muitas das ameaças conhecidas até a presente data.

Para este trabalho optou-se por utilizar as ferramentas disponibilizadas e recomendadas pela FIWARE que utilizados em conjunto com as suas aplicações, será utilizado o FogFlow, consegue prover mais segurança e robustez a aplicação.

4.1 IDM - Keyrock

A plataforma FIWARE disponibiliza o Keyrock para gerir autenticação e autorização, disponibilizando funções de OAuth2. Implementa os conceitos principais de um Identity management (IDM), faz a segregação de acessos por utilizador, aplicações e organização e permite definir papéis e permissões aos utilizadores e aplicações.

Possibilita realizar o processo de autenticação utilizando tabelas externas e outros autenticadores como o LDAP. Neste cenário o keyrock apenas duplica o utilizador, a palavra passe é sempre validada contra o autenticador externo.

Dentre as integrações disponíveis posso destacar a integração com o Electronic Identification, Authentication and Trust Services (eIDAS) que possibilita aceitar a utilização de um Secure electronic identification (eID) e assim garantir a identificação única do utilizador. É necessário que a aplicação além de ser registrada no Keyrock seja registrada no eIDAS para que o sistema possa fornecer esta forma de login e redireccionar para um *gateway* apropriado onde vai fazer o login e retornar a aplicação para proceder a validação.

A Figura 4.1 apresenta o fluxo de uma aplicação que possui integração com o eID.

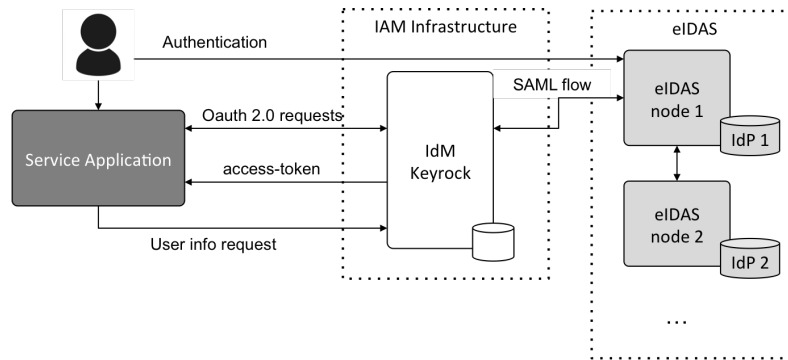


Figura 4.1: Keyrock Integração eIDAS (fonte:[12])

As configurações do Keyrock são realizadas em sua maioria por um ficheiro que contem os parâmetros a serem utilizados na ferramenta, destaco algumas tais como OAUTH2, token e utilizador:

OAUTH2.0, é possível definir o tempo de expiração do token e refresh token e ainda podemos dizer se vamos ter o conceito de url única ou não na aplicação.

TOKEN, permite definir o tipo de algoritmo a ser utilizado na geração do token para autenticação, funções do OAuth2, aplicação suporta HS256,HS384,HS512,RS256.

UTILIZADOR, quando se opta por identificar/utilizar o email, é possível implementar filtros de endereços , lista de email confiáveis e não confiáveis que pode-se utilizar na autenticação.

Possui a funcionalidade do segundo fator de autenticação o que proporciona uma segurança maior, utilizando aplicações de terceiros para geração de um One Time Password (OTP) aderente a RFC4226 e 6238. É possível selecionar quais atributos do utilizador serão partilhados com a aplicação e pode-se escolher um para se utilizar na validação do segundo fator de autenticação. O segundo fator consiste basicamente em três opções e uma delas pode se escolher para se utilizar como validação, pergunta secreta:

- Forneça uma pergunta e sua resposta (mantenha em segredo)
- Digitalize o QR fornecido com o aplicativo ou insira o segredo manualmente
- Insira o código gerado pelo aplicativo no Keyrock

A sua utilização e integração pode ser feita Application Programming Interface (API)s, IDM API, é compatível com os formatos:

- RESTful web services.
- HTTP/1.1.

- JSON and/or XML data serialization formats.

A Figura 4.2 apresenta um trecho extraído do ficheiro de configuração.

```
config.oauth2 = {  
  allowEmptyState: false,  
  authorization_code_lifetime: 5 * 60,  
  access_token_lifetime: 60 * 60,  
  ask_authorization: true,  
  refresh_token_lifetime: 60 * 60 * 24 * 14,  
  unique_url: false  
};
```

Figura 4.2: Keyrock Integração 2 (fonte:[12])

É importante perceber os principais conceitos utilizados na ferramenta Keyrock:

- Utilizador
 - Quem possui uma conta registada no Keyrock;
 - Pode gerir organizações e registar aplicativos.
- Organizações:
 - São grupos de utilizadores que partilham recursos de uma aplicação (funções e permissões).
 - Os utilizadores podem ser membros ou proprietários (gerir a organização).
- Aplicações:
 - Tem a função de cliente na arquitetura OAuth 2.0 e solicitará dados protegidos do utilizador.
 - São capazes de autenticar utilizadores usando suas credenciais OAuth (ID e secret) que identificam inequivocamente o aplicativo.
 - Defina funções e permissões para gerir a autorização de usuários e organizações.
 - Pode registar Agentes Internet of Things (IoT).
- Regras (roles):
 - São as regras, um conjunto de instruções e ou permissões que podem ser associados a um utilizador e ou organização do aplicativo.
 - Os utilizadores e ou organizações que associados, tem acesso a executar estas regras e também permissões para aceder os dispositivos envolvidos.
- Permissões:

- Acessos e ou autorizações para executar determinada operação/funcionalidade no sistema.
- São concedidas para que seus utilizadores possam validar o acesso a determinada função.

A Figura 4.3 ilustra as entidades e sua interação na ferramenta.

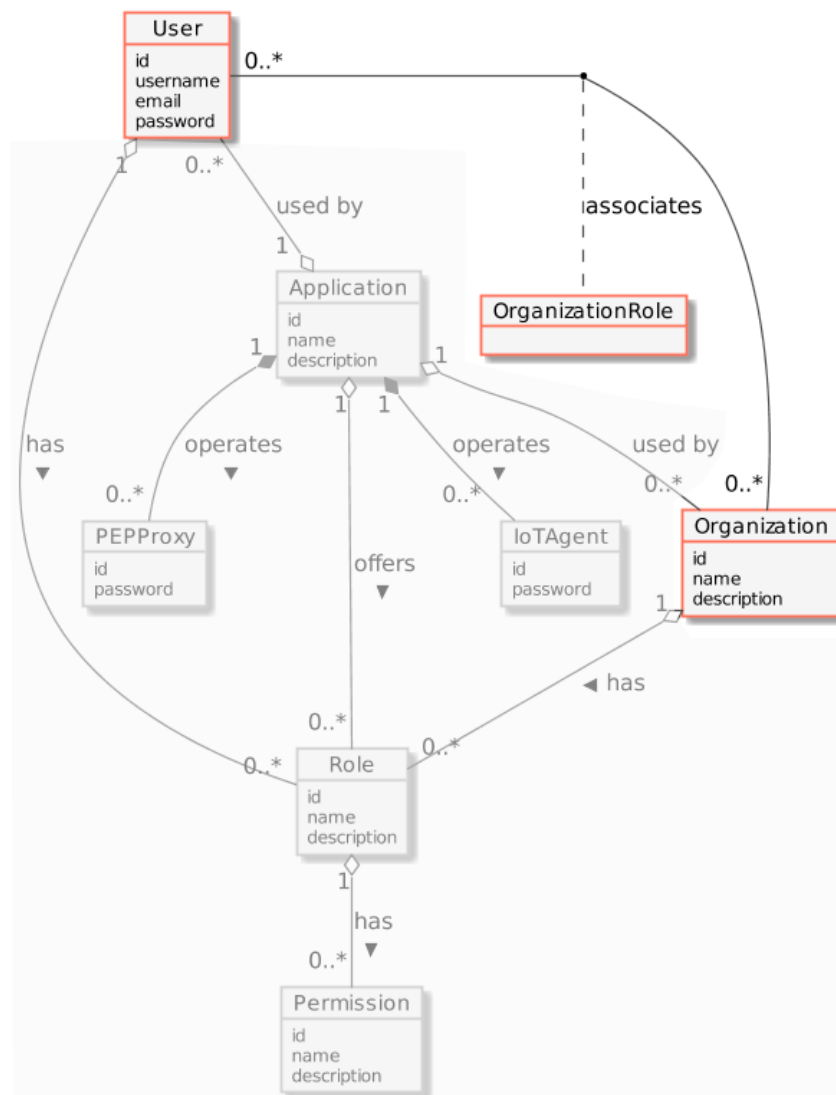


Figura 4.3: Keyrock fluxo (fonte:[13])

O Keyrock consegue gerir algumas validações básicas como acesso ao recurso, HTML Verb + path e permite especificar políticas no padrão XACML no âmbito de autorização. Se quisermos algo mais robusto é necessário utilizar outra ferramenta, o AuthZForce, que trata regras como Http verb + recurso e XACML advanced em conjunto com o PEP Proxy necessário para viabilizar a integração, sendo necessário ativar o Keyrock Policy Decision Point (PDP).

Pode-se considerar a existência de três níveis **PDP** controle de acesso:

- Nível 1: Acesso de autenticação - permite todas as ações para cada utilizador conectado e nenhuma ação para um usuário anónimo.
- Nível 2: Autorização básica - Verifique a quais recursos e verbos o utilizador conectado no momento deve ter acesso
- Nível 3: Autorização Avançada - Controle refinado por meio de XACML.

Pode ser usado para oferecer um *PDP* simples de nível 1 e 2 por conta própria e pode oferecer o nível 3 combinado com ferramentas adicionais.

4.1.1 Funcionamento

Para se utilizar a ferramenta, em linhas gerais, é necessário que as aplicações sejam cadastradas na plataforma bem como os seus componentes(sensores/atuadores). Cada aplicação tem um cliente Id e Secret, deve possuir ao menos um utilizador e ou organização para que seja possível realizar uma gestão de acessos a aplicação. Caso a organização onde será utilizada a aplicação já possua um catálogo de utilizadores/grupos, como exemplo um LDAP/AD/etc., é possível fazer uma integração do mesmo para autenticação e evitar a réplica de catálogos.

Podemos citar como exemplo, no FogFlow, o Keyrock é responsável por gerar e validar os tokens que serão utilizados pelas aplicações, dispositivos e utilizadores para se comunicar interna e externamente. No caso dos utilizadores ela gerência as permissões/políticas de acesso.

4.1.1.1 Autenticação

Ao se criar uma aplicação, é atribuída uma credencial, Oauth2 Client ID e Client Secret, a ser a sua identificação no sistema. Para cada aplicação é possível associar um proxy e os sensores **IoT** que podem aceder a esta aplicação. A seguir deve-se associar os utilizadores e ou organizações que podem aceder à mesma e também dizer quais papéis devem possuir relacionados a esta aplicação.

A Figura 4.4 ilustra a interface da ferramenta com as informações.

A autenticação, controlo de acesso, é realizado com o Oauth2 sendo solicitado ao utilizador informar User/Password e/ou ClientId e de acordo com o tipo de autenticação definido na aplicação, é retornado um access token contendo as informações necessárias para a aplicação decidir se concede ou não o acesso ao recurso solicitado. Já os dispositivos, associados a uma aplicação, possuem um id e password e no momento da autenticação é feita a gestão e troca dos tokens.

O tipo de token padrão é o Bearer Token mais suporta o JWT. Os tokens podem ser gerados como temporário ou permanentes.

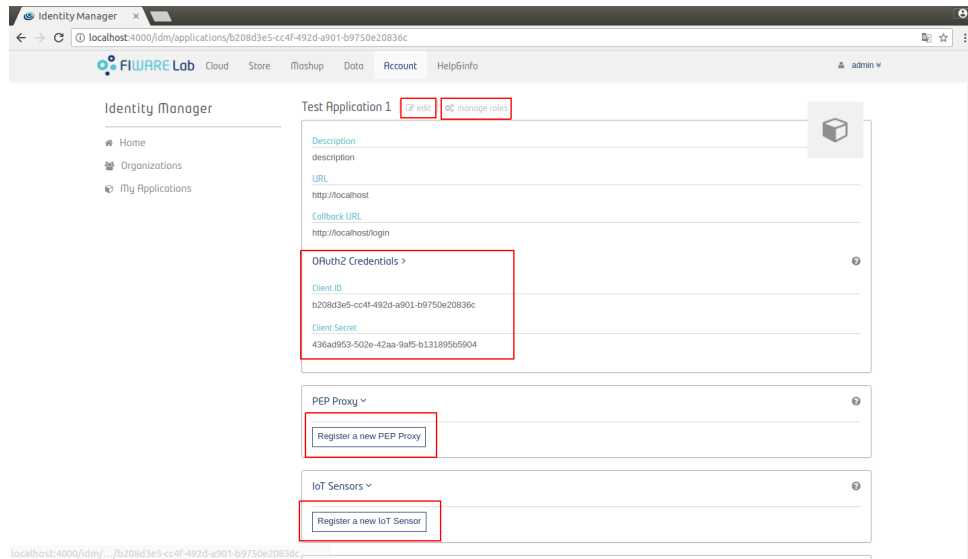


Figura 4.4: Keyrock Autenticação (fonte:[13])

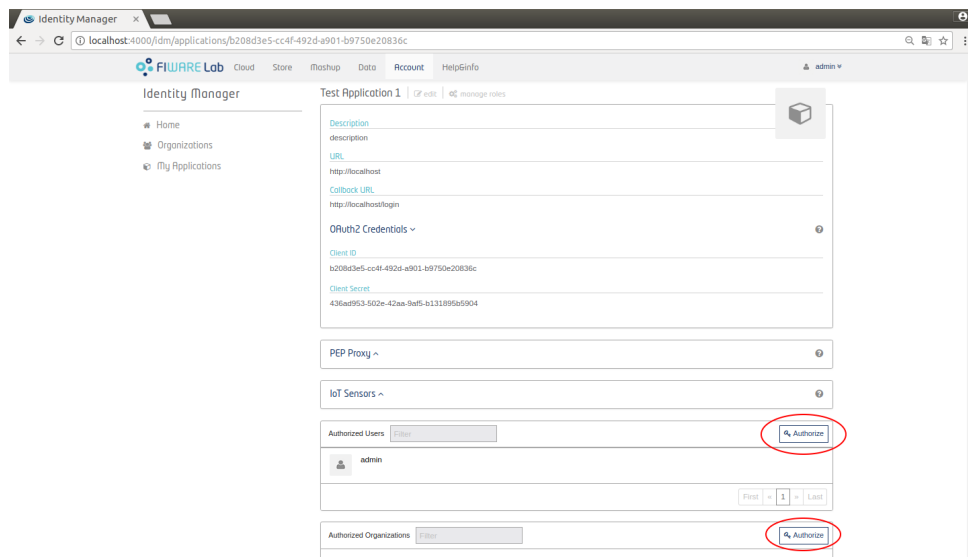


Figura 4.5: Keyrock Autenticação organização (fonte:[13])

O OAuth2, descrito na RFC6749, oferece suporte aos *grant types*:

- **Authorization Code Grant**, utiliza o clientid criado para a aplicação é válida se o utilizador está autorizado retornando um código válido a ser utilizado para gerar o token de acesso a aplicação.
- **Implicit Grant**, este modo otimiza o processo ao invés de retornar um código para ser enviado e retorna o token, já retorna o token directamente ao solicitante reduzindo uma chamada ao serviço por parte do utilizador.
- **Resource Owner Credentials Grant**, este modo permite ao invés de se informar o clientId utilizar o user e password do utilizador e receber directamente o accesstoken para

aceder as aplicações.

- **Client Credentials**, neste modo apenas se informar o clientID e consegues obter um token válido para aceder a aplicação.
- **Refresh Token**, este modo permite solicitar um novo token quando o atual ainda está válido e ou próximo da sua expiração.

4.1.1.2 Autorização

Para a gestão das políticas de uso dos dados, o Keyrock possui os papéis de Policy Administration Point (**PAP**) e Policy Information Point (**PIP**), estas opções estão disponíveis na configuração da role. É possível definir as permissões por recursos e também o uso ou não de expressões regulares.

O processo de configuração pode ser observado nas Figuras 4.6 e 4.7 , ondem podemos visualizar as permissões e os grupos bem como a informações necessárias para criar uma nova permissão na interface do Keyrock.

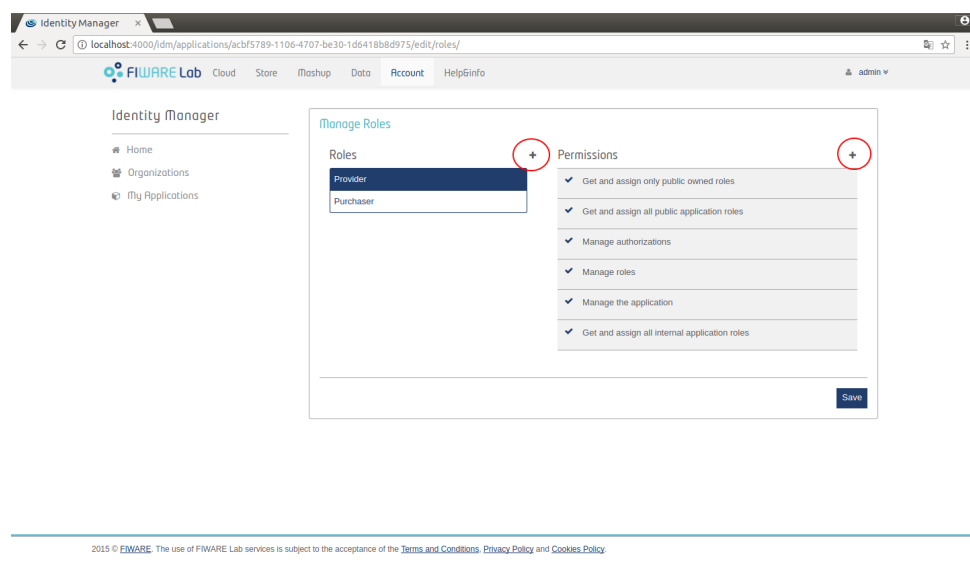


Figura 4.6: Keyrock Autorização (fonte:[13])

4.2 AuthZForce

É uma ferramenta que realiza a gestão das autorizações, Policy Decision Point (**PDP**), e a sua comunicação é realizada com o uso de REST Application Programming Interface (**API**) compatíveis com os formatos eXtensible Access Control Markup Language (**XACML**), um standard da Organization for the Advancement of Structured Information Standards (**OASIS**) de autorização e lógicas de avaliação. O **XACML** foi desenhado para ser uma linguagem de controle de acesso e formatos de solicitação-resposta, utilizada na maioria dos casos de uso de controle de acesso.

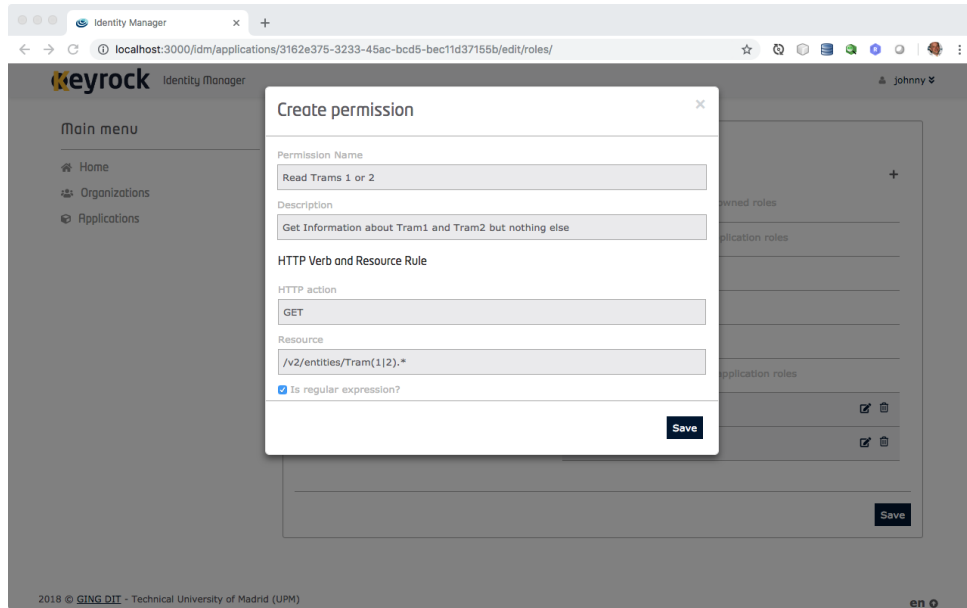


Figura 4.7: Keyrock Autorização Organização (fonte:[13])

Para o seu coreto funcionamento e estar em conformidade com a arquitetura de referencia XACML, é necessária outra ferramenta, um Policy Enforcement Point (PEP) que ira intercetar as requisições a serem autorizadas por este PDP.

A Figura 4.8 ilustra esta arquitetura.

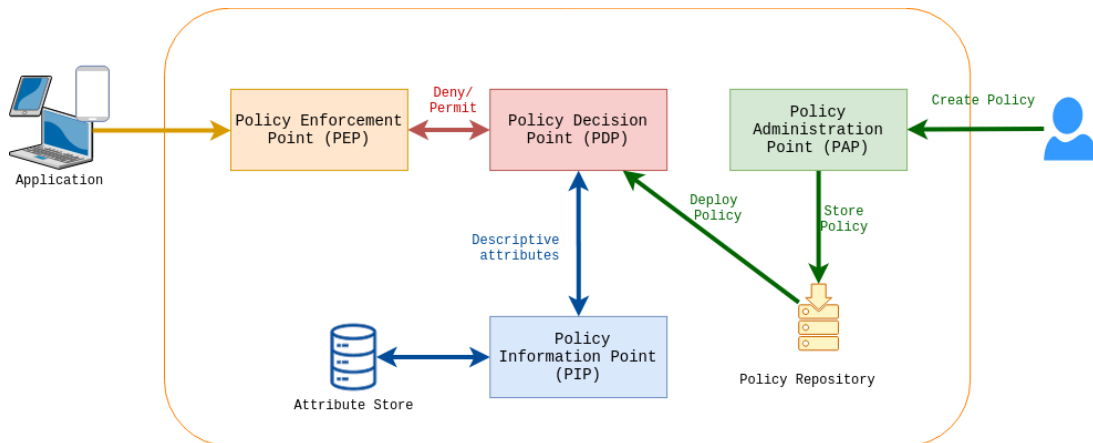


Figura 4.8: Arquitetura XACML (fonte:[19])

A Tabela 4.1 apresenta as funcionalidades de cada componente na arquitetura XCAML.

O AuthZForce suporta a linguagem XACML 3.0 que o permite aplicar políticas de controle de acesso genéricas e complexas. Possui nas suas novas versões suporte para o formato Fast infoset, um standard da ITU-T/ISI para representação XML no formato binário e utiliza um servidor, ex TOMCAT, para configuração e exposição dos seus recursos¹.

¹O AuthZForce para Policy Repository Point (PRP) não utiliza um Banco de dados e sim sistema de ficheiros.

Tabela 4.1: Componentes na arquitetura XACML

Componentes	Funcionalidade
PAP	Gerenciar as políticas de autorização de acesso.
PDP	Avalia as solicitações de acesso em relação às políticas de autorização antes de emitir decisões de acesso.
PEP	Intercepta a solicitação de acesso do usuário a um recurso, faz uma solicitação de decisão ao PDP para obter a decisão de acesso e atuar sobre ela.
PRP	É o repositório onde as políticas de autorização de acesso XACML são armazenadas.
PIP	Actuar como uma fonte de valores de atributos.

Possui o conceito de múltiplos domínios (ou tenant) o que possibilita criar um isolamento entre políticas, segregação de políticas por ambientes e projetos de forma a garantir mais privacidade para as pessoas e organizações. Pode-se definir um conjunto de padrões que vai ser aplicado sobre todos os domínios de forma a normaliza-los. Para implementar este conceito é importante perceber no que vai consistir cada domínio:

- Vários meta-dados sobre o domínio: ID atribuído pela API AuthZForce, ID externo (atribuído pelo cliente de aprovisionamento), descrição;
- Um repositório de políticas;
- Configuração dos provedores de atributos: os provedores de atributos fornecem atributos que o PEP NÃO fornece diretamente no XACML <Request>. Por exemplo, um provedor de atributos pode obter valores de atributos de um banco de dados externo.

O sistema tem como base ficheiros de configurações, armazenamento de políticas e um sistema de cache para otimizar o processamento e faz-se necessário a utilização de aplicativos externos para no caso de se utilizar múltiplos domínios, garantir a sincronização e atualização do sistema.

AuthZForce fornece as seguintes APIs, na terminologia XACML:

- PDP: API para obter decisões de autorização, por um mecanismo de controlo de acesso compatível com o XACML;
- PAP: API para gerir políticas XACML a serem tratadas pelo Authorization Service PDP.

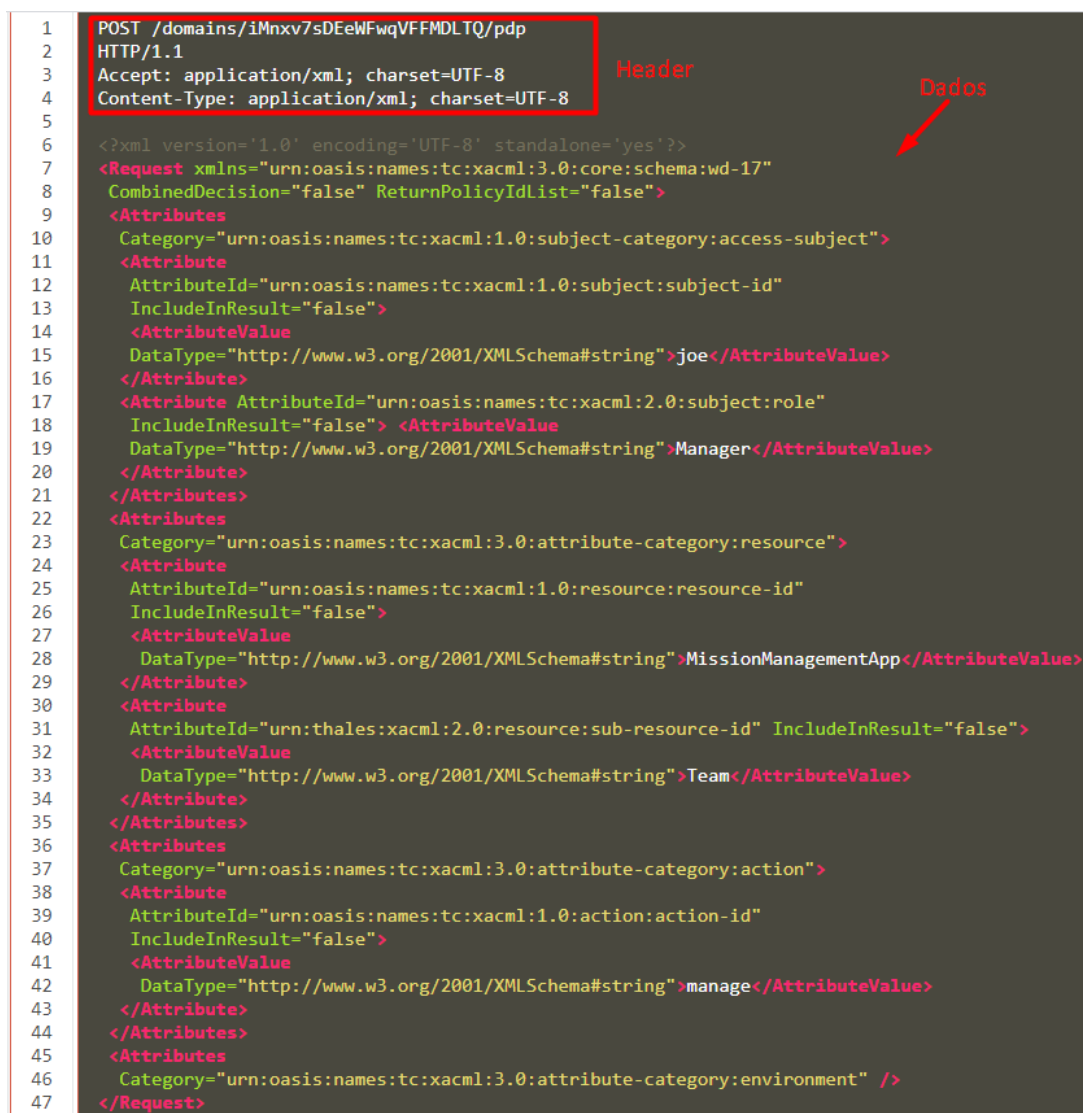
A lista completa de APIs RESTful esta descrita disponível no repositório do projeto em um ficheiro XML/WADL [3].

4.2.1 Funcionamento

O funcionamento tem como base a chamada das Rest API, tanto para a sua administração e configuração como na sua utilização.

A sua utilização massiva é feita pela API Rest de **PDP** que permite enviar as informações nos pedidos, utilizar o método POST. O retorno desta chamada, quando a operação for bem sucedida é um objeto contendo as tag de resultado da decisão e os valores que podem ser *Permit* ou *Deny*.

As Figuras 4.9 e 4.10 , apresentam um exemplo de uma chamada a API de *Policy Decision* ao método POST com as informações necessárias para validação, XACML definido no formato XACML 3.0 presente no *body* do request e o retorno da chamada a **API** contendo a decisão a ser repassada.



```

1 POST /domains/iMnxv7sDEeWFwqVFFMDLTQ/pdp
2 HTTP/1.1
3 Accept: application/xml; charset=UTF-8
4 Content-Type: application/xml; charset=UTF-8
5
6 <?xml version='1.0' encoding='UTF-8' standalone='yes'?>
7 <Request xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
8   CombinedDecision="false" ReturnPolicyIdList="false">
9   <Attributes
10     Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
11     <Attribute
12       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
13       IncludeInResult="false">
14       <AttributeValue
15         DataType="http://www.w3.org/2001/XMLSchema#string">joe</AttributeValue>
16       </AttributeValue>
17     <Attribute AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
18       IncludeInResult="false"> <AttributeValue
19         DataType="http://www.w3.org/2001/XMLSchema#string">Manager</AttributeValue>
20       </AttributeValue>
21     </Attributes>
22   <Attributes
23     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
24     <Attribute
25       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
26       IncludeInResult="false">
27       <AttributeValue
28         DataType="http://www.w3.org/2001/XMLSchema#string">MissionManagementApp</AttributeValue>
29       </AttributeValue>
30     <Attribute
31       AttributeId="urn:thales:xacml:2.0:resource:sub-resource-id" IncludeInResult="false">
32       <AttributeValue
33         DataType="http://www.w3.org/2001/XMLSchema#string">Team</AttributeValue>
34       </AttributeValue>
35     </Attributes>
36   <Attributes
37     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
38     <Attribute
39       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
40       IncludeInResult="false">
41       <AttributeValue
42         DataType="http://www.w3.org/2001/XMLSchema#string">manage</AttributeValue>
43       </AttributeValue>
44     </Attributes>
45   <Attributes
46     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:environment" />
47 </Request>
  
```

Figura 4.9: AuthZForce Pedido (fonte:[19])

Podemos observar melhor o funcionamento e interligação entre os componentes no exemplo apresentado na Figura 4.11. Representa a o processo de validar a autorização para aceder a um documento os passo a serem realizados são:

1. Um utilizador envia uma solicitação que é intercetada pelo **PEP**;

```

1 HTTP/1.1 200 OK
2 Content-Type: application/xml; charset=UTF-8
3
4 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
5 <Response xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
6   <Result>
7     <Decision>Permit</Decision>
8   </Result>
9 </Response>

```

Figura 4.10: AuthZForce Retorno (fonte:[19])

2. O **PEP** converte a solicitação em uma solicitação de autorização **XACML**;
3. O **PEP** encaminha a solicitação de autorização para o **PDP**;
4. O **PDP** avalia o pedido de autorização em relação às políticas com as quais está configurado. As políticas são adquiridas através do **PRP** e geridas pelo **PAP**. Se necessário, ele também recupera valores de atributo de **PIP** Policy Information Points (PIP) subjacentes;
5. Ao **PDP** chega a uma decisão (Permitir / Negar / Não Aplicável / Indeterminado) e a devolve ao **PEP**.

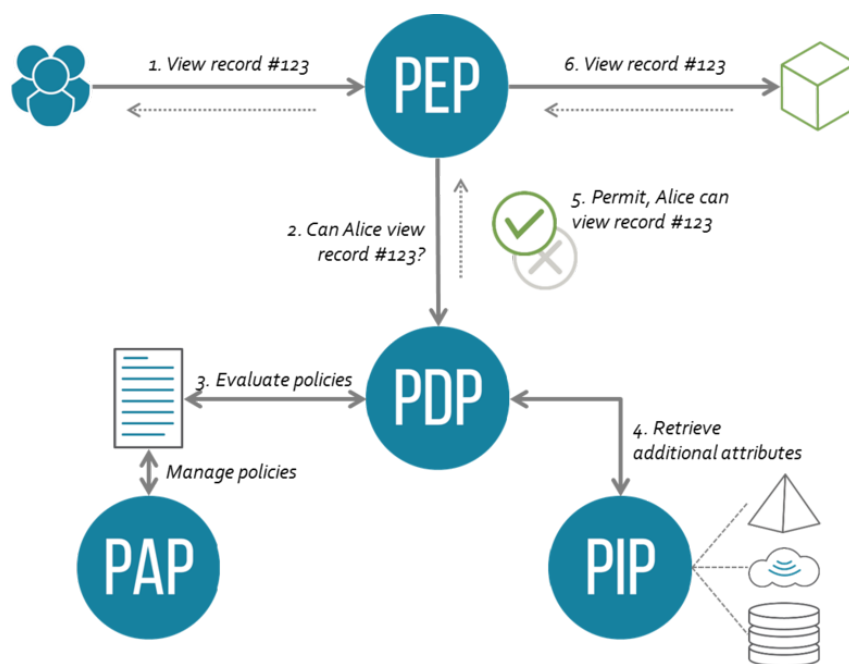


Figura 4.11: Exemplo (fonte:[1])

4.3 PEP Proxy - Wilma

Policy Enforcement Point (**PEP**) componente com a finalidade de escutar todos os pedidos de uma aplicação, como se fosse um filtro, e permitir realizar ações e validações sobre o pedido antes

de ser recebido pelo destinatário. Permitindo realizar o processo de autenticação e autorização em conjunto com o AuthZForce. O Wilma tem por defeito a opção de guardar os dados em uma cache por um período de tempo pré-definido e assim os pedidos são validados primeiro na cache e caso não encontre é então acionado o fluxo definido para a aplicação, ex: validação de dados e tokens no Identity management (**IDM**).

Pode-se realizar a integração com as ferramentas de **IDM** (Keyrock) e de Policy Decision Point (**PDP**) (AuthZForce) onde, será atribuída uma chave de acesso e *endpoints* específicos para a ferramenta interagir e realizar os processo de autenticação e autorização para aceder à aplicação. Nos cenários de aplicações que já possuem um proxy reverso exemplo Nginx, pode-se utilizar o Wilma PEP Proxy via Application Programming Interface (**API**). Neste caso em concreto se utilizar o modulo `auth_request` do Nginx, que ira acionar o *proxy* para reduz a complexidade e mitigar a questão de múltiplos *proxys* em diversas camadas.

A sua configuração é definida em um ficheiro da aplicação, disponível para consulta na documentação da ferramenta [14], cito algumas itens que julgo relevantes:

- **App**, é preciso definir uma aplicação a ser protegida, se houve mais do que uma, deve-se subir uma instância para cada aplicação;
- **PEP**, deve-se indicar na aplicação que vai ser utilizado o proxy e as credenciais geradas tais como no caso dos sensores devem estar presentes neste ficheiro de configurações;
- **CACHE_TIME**, o tempo é predefinido e deve ser menor ou igual ao tempo utilizado no **IDM** caso contrário pode-se estar a utilizar um token invalido.
- **Authorization**, é possível definir os níveis de autorização e se vai ser utilizado um componente externo como o AuthZForce para validar.
- **public_paths**, permite definir uma lista de recurso(path) que não serão validados para autenticação e ou autorização.

A implementação pode ser visualizada na Figura 4.12 que ilustra o fluxo do processo de autenticação de uma aplicação utilizando o Wilma.

4.3.1 Funcionamento

O PEP Proxy Wilma após ser instalado e configurado, funciona recebendo as chamadas as **API** da aplicação (backend) sendo obrigatória a informação do header X-Auth-Token: ou Authorization: que contém o token oauth obtido no **IDM** a ser validado.

O Wilma se autentica no **IDM** ou **PDP** e inicia o processo de validação de acordo com um dos três níveis de segurança configurados para a aplicação. De acordo com o retorno obtido, ele então repassa a chamada para ser recebida pela aplicação de backend ou retorna o erro ao chamador.

Os níveis de segurança disponíveis são:

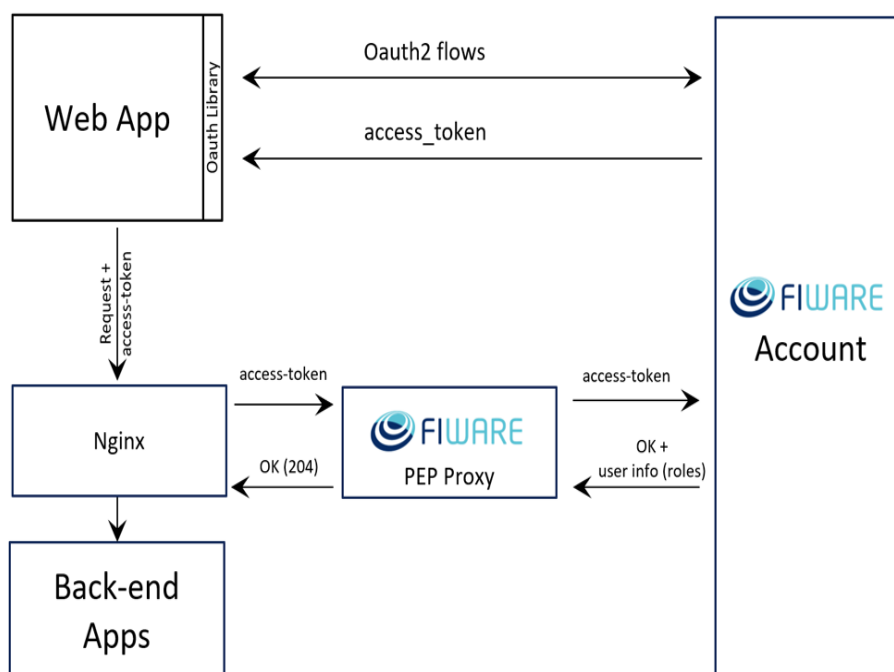


Figura 4.12: PEP Wilma com Nginx (fonte:[14])

- Nível 1: Autenticação, O PEP Proxy verifica se o token incluído na solicitação corresponde a um usuário autenticado na ferramenta de **IDM** FIWARE. Como pode-se verificar na Figura 4.13 que ilustra o fluxo de autenticação nível 1.

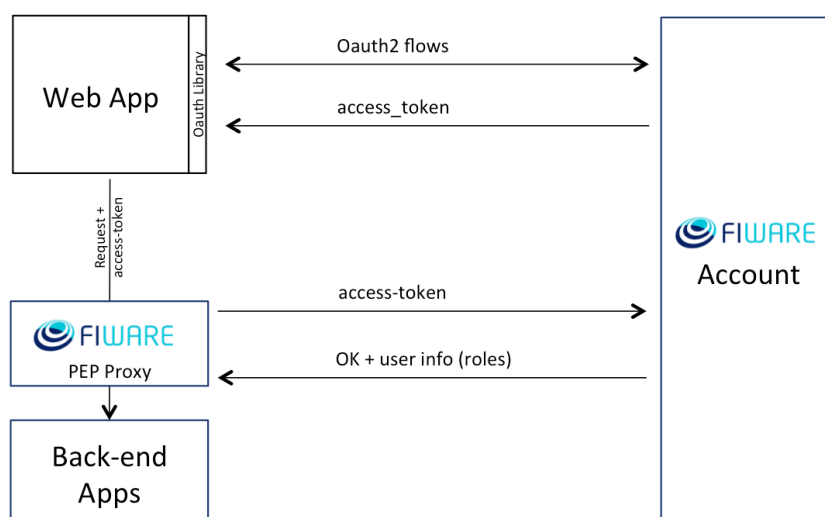


Figura 4.13: PEP Wilma Nível 1 Autenticação (fonte:[14])

- Nível 2: autorização básica, o PEP Proxy verifica se o token incluído na solicitação

corresponde a um usuário autenticado na ferramenta de **IDM** FIWARE, mas também se as funções que o usuário possui permitem que ele utilizar o recurso especificado na solicitação. Isso se baseia no verbo HTTP e no caminho. Como pode-se verificar na Figura 4.14 que ilustra o fluxo de autorização nível 2.

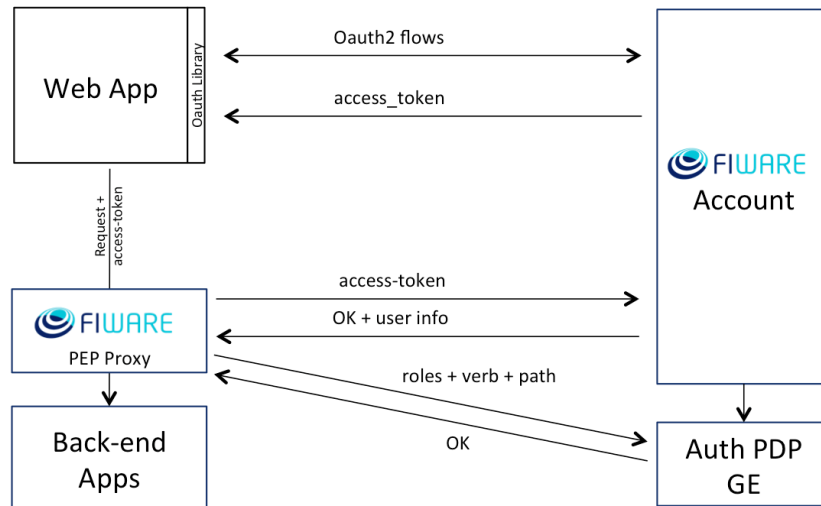


Figura 4.14: PEP Wilma Nível 2 (fonte:[14])

- **Nível 3: Autorização avançada**, o PEP Proxy verifica se o token incluído na solicitação corresponde a um usuário autenticado na ferramenta de **IDM** FIWARE, mas também a outros parâmetros avançados, como o corpo ou os cabeçalhos da solicitação. Como pode-se verificar na Figura 4.15 que ilustra o fluxo de autorização avançada nível 3.

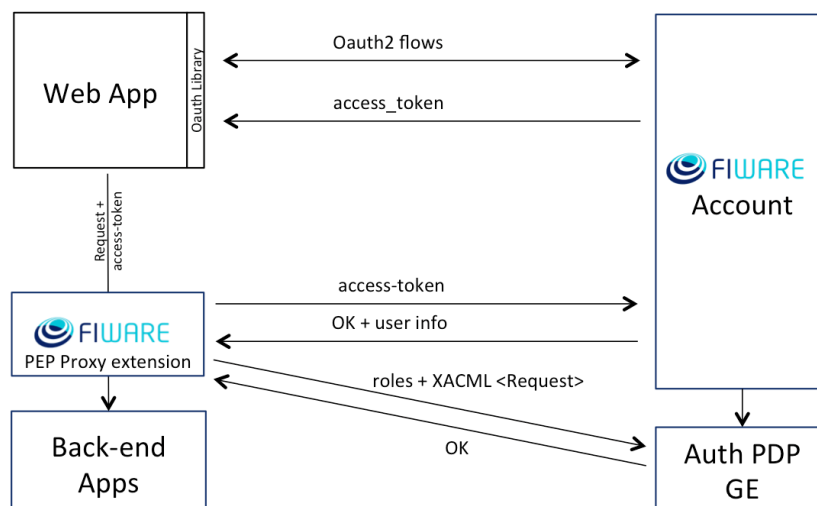


Figura 4.15: PEP Wilma Nível 3 (fonte:[14])

4.4 Usage Control

Para a gestão das políticas de uso dos dados, foi adicionada a ferramenta **Usage Control** que permite definir políticas para um controlo de uso de recursos. A sua utilização está diretamente ligada com o uso de um Identity management (**IDM**), no contexto da FIWARE o Keyrock, que possui os papéis de Policy Administration Point (**PAP**) e Policy Information Point (**PIP**) em uma arquitetura desenhada para suportar Usage Control.

No que tange o controlo de dados é possível definir o controlo de acesso e uso dos dados.

Controle de acesso a dados (Data Access control):

- Especifica quem tem permissões para aceder a um recurso;
- Quais são os direitos de acesso (ações) que tem acesso;

Controle de uso de dados (Data Usage Control):

- Fornece mecanismo para definir a propriedade dos dados;
- Definir o que pode acontecer com os dados (uso futuro);
- Visualizar o relacionamento, ingestão e processamento de dados;
- Contexto de proteção de propriedade intelectual, proteção de privacidade, conformidade com regulamentos e gerenciamento de direitos digitais.

No âmbito da FIWARE, foi idealizada uma arquitetura que tem como base em três principais componentes: *Provedor de Dados*, *Consumidor de Dados* e *Controlador de Dados* sendo que em alguns cenários o provedor e o controlador podem estar juntos sendo um só componente. Este esquema é derivado de um modelo híbrido baseado na Directiva de Privacidade de Dados 95/46 / EC e na arquitetura de referência IDS. A Figura 4.16 ilustra este modelo.

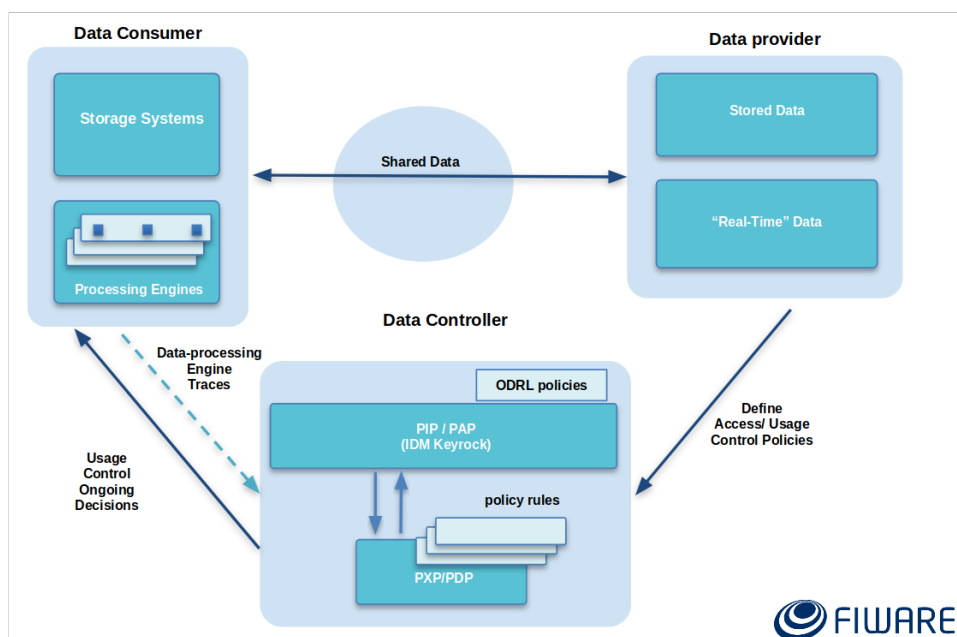


Figura 4.16: Modelo Usage Control (fonte:[1])

A implementação destes componentes, tecnicamente, exige o uso de mais aplicativos para realizar o processo de gestão destas permissões. Estão descritas as ferramentas necessárias para a implementação das funcionalidades descritas neste módulo.

Consumidor de dados:

- **Cluster Apache Flink** : mecanismo de processamento de Big Data no qual as tarefas do cliente são executadas. O consumidor de dados pode escrever jobs de processamento de dados em tempo real usando Flink for Scala e o FIWARE Cosmos Orion Flink Connector para ter uma ingestão directa de dados do Orion no mecanismo de processamento.

Provedor / controlador de dados:

- **Orion Context Broker** : Componente que permite gerir todo o ciclo de vida das informações de contexto;
- **IdM Keyrock** : Componente para definir as políticas de Controle de Acesso e Uso;
- **Policy Enforcement Point (PEP) Proxy** : Componente para aplicar políticas de controle de acesso;
- **Policy Translation Point (PTP)** : Componente para traduzir as Políticas FI-ODRL em um programa que verifica a conformidade em tempo real;
- **Policy Execution Point (PXP) / Policy Decision Point (PDP)** : Componente com recursos de processamento de eventos complexos (Complex Event

Process (CEP)) para análise dos logs e verificar o cumprimento das obrigações definidas no IDM e se necessário, fazer cumprir as punições.

A Figura 4.17 apresenta uma visão mais técnica dos componentes utilizados para suportar esta funcionalidade.

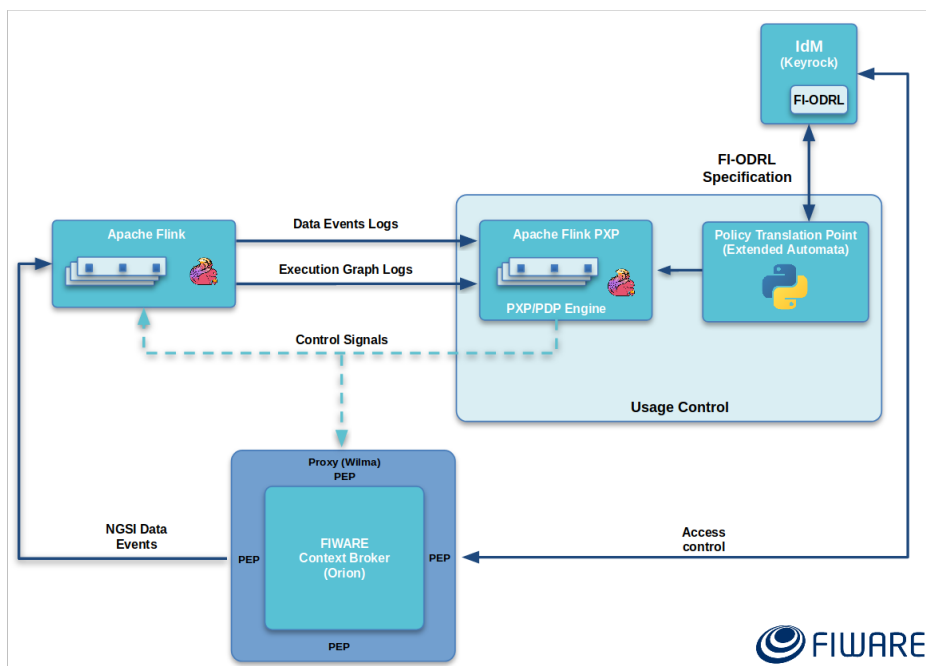


Figura 4.17: Modelo Usage Control (fonte:[1])

4.4.1 Funcionamento

A utilização desta funcionalidade consiste em definir as políticas e aplicações que vamos aplicar as devidas permissões e punições.

O Keyrock permite definir os tipos de política, descritos abaixo, na configuração das regras para uma aplicação:

- Agregação
- Contar
- Personalizado.

Nota: O tipo Personalizado permite que seja incluída as regra de política seguindo a especificação Open Digital Rights Language (ODRL).

A Figura 4.18 ilustra este modelo.

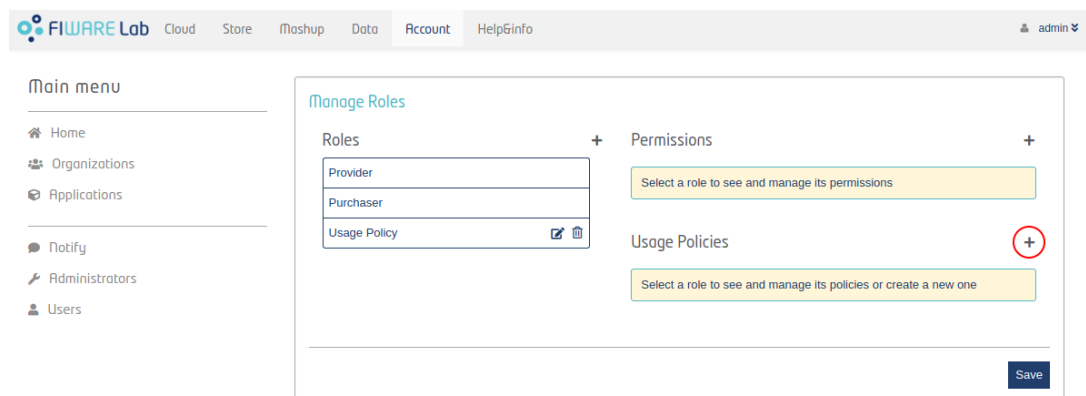


Figura 4.18: Keyrock Usage Control (fonte:[1])

A visualização e acompanhamento, é possível produzir um dashboard para ter um controlo da utilização e estado das políticas em real-time. A Figura 4.19 apresenta um exemplo de dashboard.

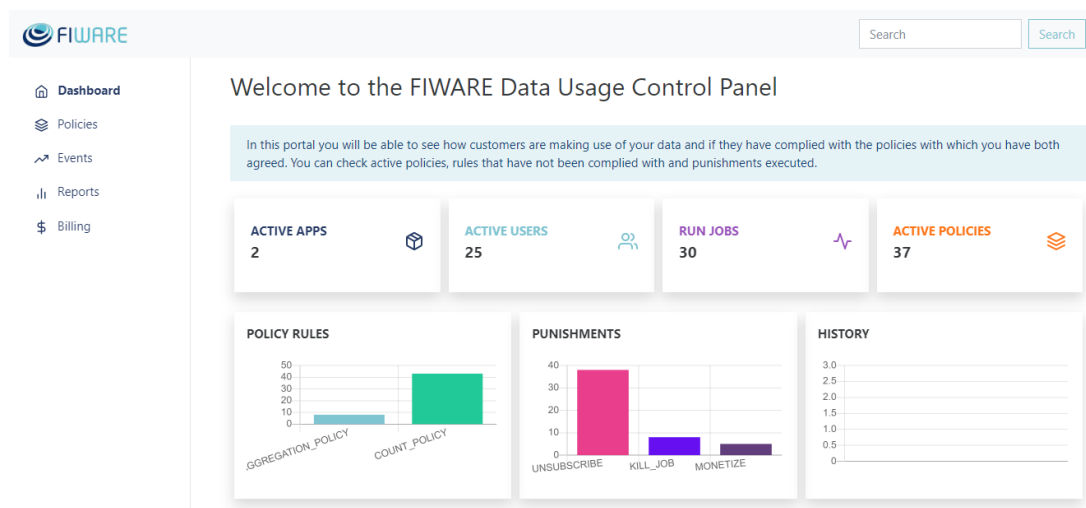


Figura 4.19: Dashboard Usage Control (fonte:[1])

Capítulo 5

Análise de segurança FIWARE

Neste capítulo, será apresentada análise dos componentes de segurança disponíveis na plataforma FIWARE. Esta análise teve como objetivo identificar como a segurança/privacidade é tratada na plataforma e aplicada em cada aplicação.

A plataforma FIWARE oferece algumas ferramentas no âmbito de segurança tais como, Keyrock (Identity management (**IDM**)), Wilma PEP Proxy (Proxy), Authzforce (Policy Administration Point (**PAP**)/Policy Decision Point (**PDP**)), que ajudam neste percurso para tornar a utilização das aplicações e a própria plataforma mais robusta e com isso, ter mecanismos de segurança para as informações por ela tratadas.

A análise foi realizada no âmbito de segurança da informação, tendo como base o estudo isolado dos componentes nos contextos dos dados e entidades representadas. O resultado será apresentado com o auxílio de uma matriz que irá ilustrar a aderência aos critérios de segurança da informação avaliados.

O material base para esta análise foi a documentação disponibilizada pelo fornecedor, FIWARE, as ferramentas para realização de teste pratico, POC - Prova de conceito, e validar alguns pontos que podem não ser de fácil compreensão na documentação.

Foi escolhida uma ferramenta para experimentação e suporte da análise, neste capítulo será apresentada uma breve introdução do seu funcionamento para auxiliar na compreensão da análise dos componentes descrito como recomendados para uso em conjunto com o FogFlow, a ferramenta escolhida para o âmbito desta análise.

5.1 Escolha da Ferramenta

Para suportar e materializar os conceitos de Internet of Things (**IoT**) e armazenamento de dados, foi utilizada a FIWARE que possui um framework e um conjunto ferramentas disponíveis para implementar uma plataforma e soluções de **IoT**. Em particular foi observado o FogFlow, uma ferramenta que utiliza contexto e permite realizar um processamento mais próximo dos

dados/dispositivos com isso temos uma redução significativa no tempo de resposta e quantidade de informações transferidas na rede.

5.1.1 Arquitectura de Segurança

A documentação da aplicação FogFlow recomenda a instalação com o uso protocolo https para ter a comunicação entre os Node de Cloud e Edge protegidas [12]. Isso já acrescenta uma garantia de que a comunicação não ira ser feita em claro e que os dados que serão transmitidos entre os *brokers* estão minimamente protegidos. Para isso é necessária a utilização de um serviço de DNS e certificados digitais para cada um dos nos a serem validados durante a comunicação. A Figura 5.1 ilustra esta comunicação.

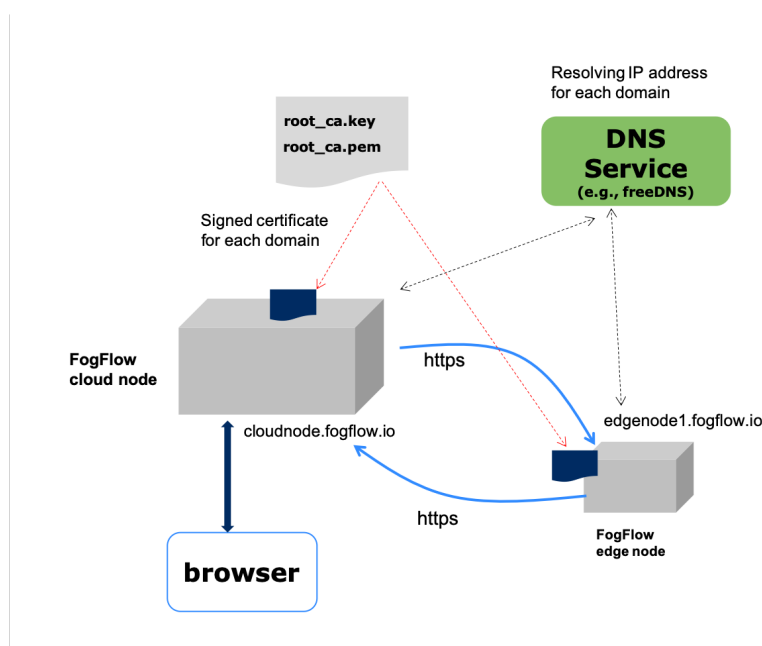


Figura 5.1: FogFlow Https (fonte:[12])

É recomendado utilizar um processo de gestão de acessos, Identity Management (IDM), que permite ter controlos para autenticação e autorizações de utilizadores, dispositivos e aplicações. A implementação pode ser realizada com a utilização de um proxy, PEP Proxy Wilma, que possibilita a intercetação dos pedidos para realizar as validações antes de chegar ao seu destino.

A plataforma FIWARE oferece as ferramentas: ¹

Identity management (IDM), Keyrock, para habilitar o processo de identificação e autenticação dos utilizadores e utiliza OAuth2 para gerar os tokens e esta comunicação ocorrer de forma segura;

¹Foi incluído um novo componente, não contemplado neste trabalho, que permite fazer uma autorização no âmbito dos dados a serem utilizados na plataforma se habilitar o *usage control* que é composto por algumas ferramentas no âmbito de monitorizar o acesso a determinados recursos do sistema.

Policy Enforcement Point (PEP), PEP Proxy Wilma, que permite filtrar os acesso e direcionar para o autenticador e ou autorizador e mediante o retorno seguir ou redirecionar o pedido;

AUTHZFORCE, autorizações quando é necessário tratar critérios complexos e personalizados e.g. Utilizar X aceder ao recurso Y no horário Z do local ABC, os basicos como recurso e verbo o próprio **IDM** consegue resolver.

A Figura 5.2 representa o fluxo sugerido na documentação para o FogFlow com os componentes da FIWARE para prover os serviços e camada de segurança.

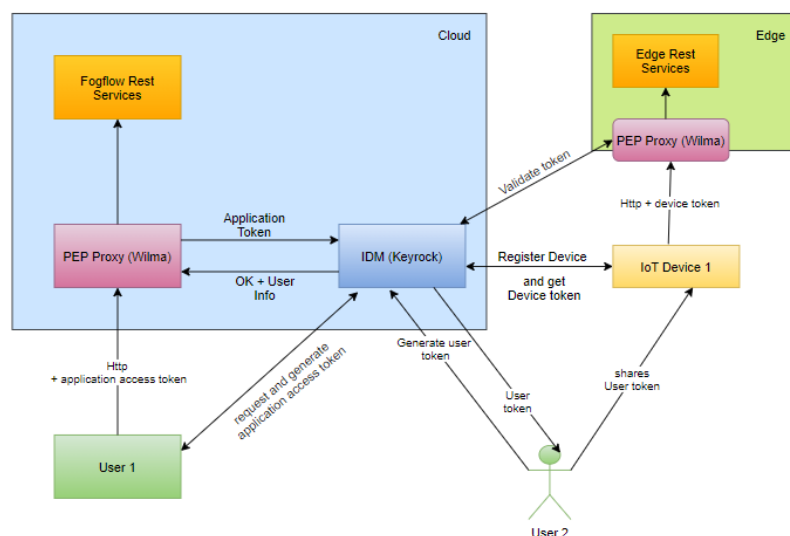


Figura 5.2: FogFlow fluxo com ferramentas de segurança (fonte:[12])

5.2 Métricas de Avaliação

Para a avaliação de segurança dos componentes utilizou-se como critérios alguns dos princípios de segurança da informação, descritos nesta secção, identificou-se funcionalidades para satisfazer cada princípio, no âmbito conceptual, que fossem comum a todos os componentes para ser possível a sua comparação. A validação da funcionalidade no componente foi realizada no contexto de entidades, de forma a perceber se a mesma existe e atender o critério totalmente, parcialmente ou não se aplica.

- **Controle de acesso / Access Control (AC)** - fornece mecanismos para monitorizar e gerir privilégios de acesso a entidades dentro do sistema e que evita que entidades não autorizadas acessem recursos e informações do sistema; É esperado que o componente possua um modelo de autenticação de utilizadores, registo e identificação de aplicações e recursos. Método de controlo de autorizações complementar as autenticações. Ex OAuth e LDAP

- **Identidade / Identity (Id)** - Fornece mecanismos para identificar inequivocamente cada entidade no sistema; É esperado que o componente possua um registo com níveis de granularidade definido para todas as entidades que tem interação com o sistema, sejam os utilizadores ou aplicações e recursos das aplicações.
- **Autenticação / Authentication (Auth)** - Fornece mecanismos para entidades do sistema comprovarem a sua identificação e permitirem o acesso ao sistema; É esperado que o componente possua um mecanismo para validar que a informação é autêntica por uma entidade interna ou externa.
- **Autorização / Authorization (Authz)** - Fornece mecanismos para gerir (conceder e revogar) o acesso aos recursos do sistema para cada entidade, com base em suas políticas de identidade e autorização; É esperado que o componente possua uma funcionalidade para gerir a concessão/revogação de acesso a recursos dentro das aplicações e deve ser nominal ou para um grupo de utilizadores/entidades.
- **Responsabilidade / Accountability (Acc)** - Fornece mecanismos para monitorizar as ações da entidade dentro do sistema, identificando assim as responsabilidades pelas ações tomadas por cada entidade (ou seja, responsável); É esperado que o componente possua uma funcionalidade onde se pode consultar as atividades/acessos realizadas no sistema bem como ter a informação de quem as executou (entidade/utilizador).
- **Auditabilidade / Auditability (Aud)** - Fornece mecanismos para monitorizar e “registrar” as interações da entidade com o sistema, para permitir a auditoria do sistema (também conhecida como rastreabilidade); É esperado que o componente possua uma funcionalidade que gera logs das atividades realizadas com informações de trilha de auditoria (data/hora, utilizador, origem e destino) de forma a ser consultada.
- **Integridade / Integrity (Int)** - Fornece mecanismos de validação da integridade das interações e dados das entidades do sistema; É esperado que o componente possua uma funcionalidade que permite validar se as informações das entidades e dados estão íntegros(sem alterações por pessoas não autorizadas) desde a sua criação e ou última alteração.
- **Autenticidade / Authenticity (Aut)** - Fornece mecanismos para comprovar a autenticidade das entidades, interações e dados do sistema; É esperado que o componente possua uma funcionalidade que permite verificar que as entidades foram registadas no sistema por alguém autorizado e se houver quaisquer alterações as mesmas, devem ser registadas e tenha uma visibilidade destes atos no sistema. Os mesmos podem estar protegidos ou em claro tendo por exemplo um hmac para validar a autenticidade.
- **Confidencialidade / Confidentiality (Conf)** - Fornece mecanismos que evitam a divulgação não autorizada de informações confidenciais; É esperado que o componente possua uma funcionalidade que trata as informações pré definidas como confidenciais (ex:tokens, secrets, password) de modo que só podem ser acedidas pelos seus proprietários ou adm do sistema.

- **Privacidade / Privacy (Priv)** - Fornece mecanismos que evitam que as identidades de cidadãos ou entidades externas ao sistema sejam determinadas com base nas informações mantidas pelo sistema; É esperado que o componente possua uma funcionalidade que permita não retornar/exibir dados de identificação das entidades sem associar estes ao código de utilizador durante os processos de autorização e autenticação.
- **Não Repúdio / Non-Repudiation (NR)** - Fornece mecanismos que evitam que as informações de uma entidade sejam utilizadas por outras entidades como suas; É esperado que o componente possua uma funcionalidade que garanta que as operações foram executadas ao seu solicitante(ex: envio de email, comprovativos, segundo fator e outros) podem ser enviadas ou armazenadas.
- **Disponibilidade / Availability (Avail)** - Fornece mecanismos para garantir que os usuários autorizados tenham acesso oportuno e ininterrupto ao sistema e / ou as informações no sistema. É esperado que o componente possua uma funcionalidade de armazenamento e recuperação de dados de fontes externas ex BD/Cache e tenha o seu processamento escalável de modo a não depender da infra totalmente para disponibilidade.

A análise teve como foco o contexto das entidades na ferramenta por considerar a sua relevância para o estudo e a identificação de que os dados das aplicações não são o foco da ferramenta e podem ser tratados com a utilização de outras ferramentas da plataforma FIWARE.

5.2.1 IDM - Keyrock

Keyrock, uma ferramenta de Identity management (**IDM**) da plataforma FIWARE, possibilita aos seus utilizadores implementar processos de autorização e autenticação; Permite a integração com sistemas de gestão de utilizadores que utilizam o protocolo LDAP e.g. Active Directory (**AD**) e também disponibiliza funções para controlo de permissões de acesso para as aplicações.

Os requisitos de segurança e privacidade são atendidos pela ferramenta de forma que cada aplicação somente está visível para os seus membros e organizações cadastradas, não sendo possível aos demais utilizadores visualizar todas as aplicações existentes, exceto os Administradores do sistema. Com relação as permissões, pode-se implementar ao nível dos membros e ou organizações ligadas à aplicação.

Permite controlo de autenticação dos dispositivos e serviços ,exemplo *proxy*, vinculados a uma aplicação. Entretanto no que tange aos dispositivos, se o mesmo necessitar ser utilizado por outra aplicação não é possível realizar esta associação na interface, tornando o uso de sensores exclusivo por uma aplicação.

Papéis e permissões (roles), as mesmas não pode ser partilhadas com outras aplicações o que obriga fazer a gestão por aplicação e ter replicas de roles e permissões.

Ressalto que as permissões(roles), não foi possível atribui-las para os sensores o que deixar

uma lacuna para validar as autorizações e pode gerar um acesso irrestrito a recursos da aplicação e também representa uma vulnerabilidade a ser explorada.

Critérios atendidos

Controle de acesso / Access Control (AC), é atendido com a funcionalidade para gerir utilizadores, aplicações e seus componentes (sensores e proxy) e implementa tokens com standard oauth.

Identidade / Identity (Id), permite que cada entidade (utilizadores e aplicações) receba um código interno único que permite identifica-la no sistema para todos os processo e.g. autenticação com a geração de tokens.

Autenticação / Authentication (Auth), atendido com a funcionalidade de geração dos tokens que permite comprovar e validar a entidade que se autentica no sistema.

Autorização / Authorization (Authz), atendido com a funcionalidade para gerir autorização a nível dos recursos da aplicação que permite na validação dos tokens e verificar os acessos autorizados para o utilizador na entidade solicitada.

Critérios não atendidos ou parcialmente atendidos

Responsabilidade / Accountability (Acc), não foi encontrada na interface uma funcionalidade para consulta das ações que foram realizadas. Os logs disponíveis não tem informações de quem pediu e sim os dados de acesso. Para isso, julgo, ser necessário incluir os utilizadores e mais dados no log e ter funcionalidade na interface para consulta.

Auditabilidade / Auditability (Aud), a ferramenta gera os logs entretanto a sua consulta não esta disponível na interface, se necessário deve-se consultar as logos via console da estação/servidor e ou utilizar ferramentas de terceiros para interpreta-lo.

Integridade / Integrity (Int), mesmo tendo uma base de dados que ofereça esta funcionalidade, não tem a validação na aplicação para mitigar a inconsistência de alguns dados.

Autenticidade / Authenticity (Aut), este critério na interface (visão do utilizador) qualquer um que tenha acesso a aplicação, pode os dados e não tem uma visibilidade de alterações, incluindo a visão de ADM, não há uma consulta onde consiga ver quem fez a alteração da entidade/componente.

Confidencialidade / Confidentiality (Conf), as informações confidenciais da ferramenta tem este tratamento, as informações das entidades(tokens e secrets e recursos) não tem uma parametrização e estão visíveis a todos quem tem acesso no nível utilizadores e administradores.

Privacidade / Privacy (Priv), a ferramenta tem uma funcionalidade que permite a um utilizador autenticado apenas com o ID da aplicação receber os dados de identificação da mesma assim como alguns dados do utilizador basta ter um token de acesso valido e não é necessário ser administrador.

Não Repúdio / Non-Repudiation (NR), tem apenas a funcionalidade de logs que é fraca. Não permiti fornecer garantias para ser utilizado como prova de realização da operação.

Disponibilidade / Availability (Avail), na configuração só identificou-se a separação da base de dados, entretanto isso não é suficiente para garantir a disponibilidade da ferramenta. Não há utilização de cache e outros métodos para atender os pedidos, e.g. mais de um endpoint para resposta/acesso.

Tabela 5.1: Matriz de Aderência aos critérios de segurança - IDM Keyrock.

Componente	Contexto	Critérios											
		AC	Id	Auth	AutZ	Acc	Aud	Int	Aut	Conf	Priv	NR	Avail
KEYROCK	Entities	Y	Y	Y	Y	N	P	N	N	P	N	N	N

- **Critérios:** Controle de acesso (AC) | Identidade (Id) | Autenticação (Auth) | Autorização (Authz) | Responsabilidade (Acc) | Auditabilidade (Aud) | Integridade (Int) | Autenticidade (Aut) | Confidencialidade (Conf) | Privacidade (Priv) | Não Repúdio (NR) | Disponibilidade (Avail).
- **Legenda:** (Y) – Atende, (N) – Não atende, (NA) Não se Aplica, (P) Não atende de forma nativa entretanto, fornece dados para ser utilizado em outras ferramentas.

5.2.2 Wilma - PEP Proxy

PEP Proxy Wilma, uma ferramenta de proxy que possui integração nativa com outras ferramentas da FIWARE, **IDM**, Policy Decision Point (**PDP**) o que mitiga o uso de ferramentas de terceiros na orquestração e otimiza a comunicação. Tem como ponto importante o facto de possuir uma cache para armazenar dados e com isso minimizar as chamadas ao **IDM**. No contexto da FIWARE/FogFlow, é o componente utilizado para interceptar as chamadas e promover uma camada para validar autorização e autenticação definidas nos demais componentes de segurança.

Como esta ferramenta vai a frente do backend, ela consegue fazer a intercetação dos dados e solicitar a outras ferramentas que sejam realizados os processos de autenticação e autorização.

Por ter uma instalação e funcionamento no server sem recorrer a um nó central, a ferramenta está em conformidade para ser utilizada numa abordagem descentralizada. Considerando que cada aplicação deve possuir a sua instância de *proxy* e a ferramenta só suporta uma instância por instalação.

Os requisitos de segurança são atendidos de forma parcial e muito se deve ao facto de possuir integração com outros componentes e até outro proxy, exemplo nginx utilizado pela FIWARE, para dar respostas aos pedidos efetuados garantindo assim o acesso as aplicações.

Na perspetiva de segurança da informação, o uso do cache pode ser um ponto de atenção, pois se não estiver bem configurado pode permitir um acesso invalido, uma vez que guarda as

informações do ultimo token/acesso realizado com sucesso e isso pode permitir reutilizar tokens expirados em algumas situações bem específicas. Deixo a recomendação de se configurar com atenção para mitigar possíveis vulnerabilidades.

Um item que ressalto é que o ficheiro de configurações necessita estar armazenado em um local com os acesso restritos pois em caso de falhas e ou perda, pode trazer prejuízos no acesso ao back-end que vai deixar de funcionar.

Critérios atendidos

Identidade / Identity (Id), atendido com a funcionalidade de gerar uma identificação para as aplicações e entidades que podem ter acessos e assim consegue identificar os destinos, back-ends, a serem enviados os pedidos.

Autenticação (Auth), atendido com a funcionalidade de exigir um token para autenticar e assim realizar a validação de acesso as aplicações.

Critérios não atendidos ou parcialmente atendidos

Controle de acesso / Access Control (AC), é atendido de forma parcial pois necessita de uma integração com uma ferramenta de **IDM** que possa validar os acessos deste utilizador e a partir deste ponto já pode o fazer por um período utilizando as informações recebidas.

Autorização/Authorization (Authz), atendido de forma parcial com a integração com sistemas de autorizações que retornam as informações para este repassar ou negar o acesso a entidades e recursos.

Responsabilidade / Accountability (Acc), não atendido, e não foi encontrada na ferramenta uma funcionalidade para consulta das ações que foram realizadas. Não tem evidencias nas logs que por sua vez não tem uma funcionalidade para extrai-las ou se conectar uma ferramenta para armazena-las.

Auditabilidade / Auditability (Aud), não é atendido mesmo a ferramenta gerando logs com poucas informações que ficam disponíveis na console.

Integridade / Integrity (Int), atendido de forma parcial com a introdução da cache permite a ferramenta realizar algumas validações incluindo a integridade dos dados fornecidos pelo **IDM**

Autenticidade / Authenticity (Aut), atendido de forma parcial com a exigência e validação dos certificados no uso do protocolo http. A ferramenta não consegue realizar a validação dos recursos exposto pelo backend apenas repassa os dados após uma validação de acesso e autorizações.

Confidencialidade / Confidentiality (Conf), atende de forma parcial com a integração a ferramentas de **IDM** que já tenham estas funcionalidade. Analisando a ferramenta de forma isolada, ela tem um tratamento de alguns dados com tidos com confidenciais.

Privacidade / Privacy (Priv), atende de forma parcial e com integração com ferramentas que permitem definir a privacidade de entidades. A analise da ferramenta de forma isolada, não tem

parâmetros para se configurar este critério.

Não Repúdio / Non-Repudiation (NR), não atendido, as configurações da ferramenta não conseguem fornecer esta funcionalidade dado que trata a requisição que pode ser recebida inúmeras vezes e se estiver autorizada, será tratada pelo back-end já fora do âmbito desta ferramenta.

Disponibilidade / Availability (Avail), atendido de forma parcial se considerar que a ferramenta tem um ficheiro de configurações que pode ser replicado e com isso ter varias instâncias para suportar os pedidos. Isso esta totalmente relacionado com a infraestrutura, se analisar a ferramenta de forma isolada da infraestrutura, pode-se concluir que não atende ao critério.

Tabela 5.2: Matriz de Aderência aos critérios de segurança - Wilma PEP Proxy.

Componente	Contexto	Critérios											
		AC	Id	Auth	AutZ	Acc	Aud	Int	Aut	Conf	Priv	NR	Avail
PEP PROXY WILMA	Entities	P	Y	Y	P	N	N	P	P	P	P	N	P

- **Critérios:** Controle de acesso (AC) | Identidade (Id) | Autenticação (Auth) | Autorização (Authz) | Responsabilidade (Acc) | Auditabilidade (Aud) | Integridade (Int) | Autenticidade (Aut) | Confidencialidade (Conf) | Privacidade (Priv) | Não Repúdio (NR) | Disponibilidade (Avail).
- **Legenda:** (Y) – Atende, (N) – Não atende, (NA) Não se Aplica, (P) Não atende de forma nativa entretanto, fornece dados para ser utilizado em outras ferramentas.

5.2.3 Authzforce

Esta ferramenta permite a implementação de políticas, simples e complexas, para autorizações de acesso e uso dos recursos das aplicações. As validações de autenticação são tratadas pelo **IDM** e Policy Enforcement Point (**PEP**) no contexto da FIWARE.

As políticas com todo o seu conteúdo podem ser divididos em domínios que conseguem ter uma atuação quase independente; O contexto pode ser implementado apenas no domínio correspondente para permitir a partilha com hierarquia de políticas por ambientes e definir uma política global valida para todos os domínios. Os acessos para receber e enviar os dados são realizados via API.

Na versão observada, 10.0.0, a ferramenta tem o suporte para múltiplos domínios (tenant), o que auxilia na redução da latência.

No âmbito da segurança foi possível observar que ainda existem muitas fragilidades, principalmente no que tange os acessos e proteção das informações. Cito os ficheiros com as permissões de acessos que não estão protegidos e tem o seu nível de acesso a ser definido pelo sistema operacional e o tema de permitir a execução de ficheiros JARs, como não há etapas de validação, isso pode comprometer o sistema. Para monitorizar deve ser acoplada uma ferramenta para leitura dos logs e ou extrair as informações geradas.

Critérios atendidos

Identidade / Identity (Id), atendido com a funcionalidade de se criar e identificar os domínios que são únicos e possuem códigos de acesso distintos.

Não Repúdio / Non-Repudiation (NR), atendido com a funcionalidade de se ter os conteúdos em um nó central o que ajuda a mitigar duplicidades de informações a serem fornecidas para as entidades e ter garantidas para uma validação do que foi executado na política.

Critérios não atendidos ou parcialmente atendidos

Controle de acesso / Access Control (AC), não atendido se focarmos no contexto da ferramenta estas funcionalidades estão fora do seu âmbito entretanto como pode sempre ter uma integração com as ferramentas de **IDM** e **PEP** da a possibilidade de validar os acessos dos utilizador com estas integrações e a partir deste ponto já pode atender de forma parcial.

Autenticação / Authentication (Auth), atendido parcialmente se considerarmos que a ferramenta pode utilizar as integrações nativa com outras ferramentas de FIWARE, **IDM**, para realizar esta validação. Se considerar a ferramenta isolada, não tem funcionalidade para atender a este critério.

Autorização/Authorization (Authz), a ferramenta sendo para controlo de autorizações, não tem funcionalidade para controlo dos acessos a mesma e na sua documentação deixa explícito que deve ser tratado no nível de quem pode ter acesso as APIS por um **IDM** para mitigar os acesso indevidos.

Responsabilidade / Accountability (Acc), não atendido, mesmo fornecendo as logs de execução, não guarda informações de quem executou as ações e alterações no sistema o que inviabiliza o atendimento ao critério.

Auditabilidade / Auditability (Aud), atendido de forma parcial se considerar que podem ser utilizados os logs de interações na ferramenta resultantes das chamadas as Application Programming Interface (**API**)s via tomcat e logs dos servidores que podem ser trabalhos e consumidos por outras ferramentas.

Integridade / Integrity (Int), atendido de forma parcial com a funcionalidade de validação dos domínios e validação das políticas implementadas na ferramenta. Como podemos ter múltiplos domínios, não é possível garantir em detalhe as politicas isso fica a cargo do seu utilizador.

Autenticidade / Authenticity (Aut), não atende pelo facto de não ter registos de alterações e as informações serem definidas e guardadas em ficheiros sem um check de consistência. Isso pode ser tratado a nível do Sistema Operacional (**SO**) entretanto na visão isolada da ferramenta não tem esta funcionalidade.

Confidencialidade / Confidentiality (Conf), atende de forma parcial se considerar que fornece a funcionalidade para os seus utilizados gerir as suas aplicações. A ferramenta não tem para os seus dados este tratamento.

Privacidade / Privacy (Priv), não atendido uma vez que não tem funcionalidade para tratar as informações da própria ferramenta. Se considerar o contexto das entidades que a utilizam, é possível fornecer esta funcionalidade através das políticas incluídas na ferramenta.

Disponibilidade / Availability (Avail), atendido de forma parcial mesmo tendo a sua instalação de forma nativa um modo onde se permite configurar múltiplos domínios para responder localmente, exige que se tenha um webAPI para receber e delegar os pedidos e ter os apontamentos de forma a garantir o correto funcionamento do sistema.

Tabela 5.3: Matriz de Aderência aos critérios de segurança - Authzforce PDP.

Componente	Contexto	Critérios											
		AC	Id	Auth	Autz	Acc	Aud	Int	Aut	Conf	Priv	NR	Avail
<i>autzforce</i>	Entities	N	Y	P	N	N	P	P	N	P	N	Y	P

- **Critérios:** Controle de acesso (AC) | Identidade (Id) | Autenticação (Auth) | Autorização (Authz) | Responsabilidade (Acc) | Auditabilidade (Aud) | Integridade (Int) | Autenticidade (Aut) | Confidencialidade (Conf) | Privacidade (Priv) | Não Repúdio (NR) | Disponibilidade (Avail).
- **Legenda:** (Y) – Atende, (N) – Não atende, (NA) Não se Aplica, (P) Não atende de forma nativa entretanto, fornece dados para ser utilizado em outras ferramentas.

5.2.4 Arquitetura de Segurança FIWARE

Com o uso das ferramentas recomendadas pela FIWARE, na arquitetura de segurança proposta para o Foflow, (**IDM**, **PEP**, **PDP**), consegue-se prover um conjunto de requisitos de segurança para as aplicações bem como nova funcionalidades para tornar a sua utilização robusta.

No âmbito uma de arquitetura descentralizada, constatou-se que será necessária uma adequação das ferramentas para responder aos novos requisitos. Tendo em consideração o pressuposto das ferramentas de **IDM** e **PDP** necessitarem de um servidor central para realizar a suas funcionalidades.

Na análise constatou-se que após passar a etapa de autenticação, os critérios de segurança analisados não estão presentes no FogFlow e isso representa um campo aberto para vulnerabilidades. É perceptível a preocupação com a performance e agilidade no processamento entretanto, isso fez com que algumas "portas" permanecessem abertas cito o baixo nível de rastreabilidade e integridade das informações, deixado a cargo do fornecedor ou ferramentas de terceiro o controlo de acesso para interpretar e mostrar estas informações.

Depois de verificar individualmente e em conjunto as ferramentas, foi possível gerar uma visão global da arquitetura de segurança oferecida pela plataforma FIWARE, recomendada para ser aplicada ao FogFlow, que tem como objetivo envolver as aplicações e segregar o que é core de cada componente. O uso deste componentes podem ser determinados pela necessidade e critérios

de segurança que pretende satisfazer na aplicação. Algumas combinações das ferramentas podem atender um conjunto de requisitos diferentes e devem ser vistas de acordo com a necessidade da aplicação. É sempre recomendado se utilizar o Wilma PEP proxy, pois dentre outras funcionalidades, ele permite criar uma camada intermediária entre o pedido do utilizador e a aplicação onde pode-se implementar os demais componentes como o **IDM** Keyrock e **PDP** Authzforce.

Apresento algumas configurações tidas como recomendadas para ter um grau tido como satisfatório de completude dos critérios de segurança.

Configurações

- **Wilma + Keyrock** - Permite ter um controlo de utilizadores e acessos a entidades.
- **Wilma + Keyrock + authzforce** - Permite ter um controlo de utilizados, Políticas de acesso a entidades.
- **Wilma + Keyrock + Usage control** - Permite além do controlo dos utilizadores, ter política de acesso a dados.
- **Wilma + Keyrock + Usage Control + authzforce** - Permite a gestão de utilizados, Políticas de acesso a entidades e dados.

A título de conhecimento, existem uma ferramenta, ainda não homologada, Usage control, que se compromete a fazer uma gestão das políticas com relação ao acesso a dados. Durante a análise não foi possível validar a sua integração com o FogFlow pois os níveis pré-definidos novamente só atendem as subscrições.

Por defeito o FogFlow e outros componentes da FIWARE não tem, em suas configurações, integrações nativas com as ferramentas de **IDM** e **PDP** sendo a recomendação utiliza-las em uma camada superior ou em alguns casos entre o backend e frontend para realizar estas interações. A abordagem de ter os controlos delegados a ferramentas especializadas, mitiga o trabalho e complexidade em desenvolver a solução e por sua vez pode trazer problemas e ou vulnerabilidades consoante a forma de integração.

Para se montar um panorama desta arquitetura proposta, levou-se em consideração a análise das ferramentas para perceber se as combinações geram um grau de segurança que atenda os critérios ou não e a ponderação dos critérios atendidos de forma parcial e não atendidos de modo que fosse possível compara-los entre as ferramentas e definir que: Atende, quando satisfeito totalmente ou parcial por todas as ferramentas; Atende Parcialmente, quando satisfeito parcialmente e não satisfeito em uma ou mais ferramentas; Não atender, quando não satisfeito por todas as ferramentas em alguns casos mesmo que tenha um parcial. Isso está representado na matriz de aderência aos critérios de segurança.

- **Crítérios:** Controle de acesso (AC) | Identidade (Id) | Autenticação (Auth) | Autorização (Authz) | Responsabilidade (Acc) | Auditabilidade (Aud) | Integridade (Int) | Autenticidade

Tabela 5.4: Matriz de Aderência aos critérios de segurança - Componentes FIWARE

Componente	Critérios											
	AC	Id	Auth	Authz	Acc	Aud	Int	Aut	Conf	Priv	NR	Avail
<i>Visão Geral</i>	Y	Y	Y	Y	N	P	P	N	P	N	N	P
<i>Keyrock</i>	Y	Y	Y	Y	N	P	N	N	P	N	N	N
<i>Wilma PeP Proxy</i>	P	Y	Y	P	N	N	P	P	P	P	N	P
<i>Autzforce</i>	N	Y	P	N	N	P	P	N	P	N	Y	P

(Aut) | Confidencialidade (Conf) | Privacidade (Priv) | Não Repúdio (NR) | Disponibilidade (Avail).

- **Legenda:** (Y) – Atende, (N) – Não atende, (P) Não atende de forma nativa entretanto, fornece dados para ser utilizado em outras ferramentas e.g. Logs.

5.3 Discussão

A análise dos componentes de forma isolada e em conjunto, permitiu ter profundidade na identificação de comportamentos que são comuns as ferramentas e por sua vez também foi possível identificar as formas de integração com as aplicações disponíveis pela FIWARE para prover uma camada de segurança.

Esta camada de segurança prove uma serie de funcionalidades que não necessariamente precisam estar integradas com as aplicações entretanto, se as aplicações não possuem formas de interagir e ou partilhar os seus dados, não se consegue atingir um nível maior de segurança, o que foi identificado na análise para o não cumprimento de alguns critérios.

Abaixo partilho uma lista dos pontos relevantes obtidos durante na análise dos componentes e ou ferramentas:

- Possuir uma solução completa para autenticação e autorização;
- No que tange o tratamento de dados, as autorizações de acesso são verificadas somente para o consumidor dos dados, subscritores, e não dos atores envolvidos no processamento. Isso deixa uma abertura para o consumo de informações dentro de processos sem autorização e.g não tenho autorização para consumir dados do sensor "A"entretanto posso incluir no meu processamento a busca de dados do sensor "A"pois o "utilizador técnico"tem esta permissão;
- Falta de informações para compor trilhas de auditorias e logs;
- Comunicação entre os backends e sensores protegida;
- Não permite validar autorizações/autenticações dentro da aplicação exemplo no FogFlow nas funções de construção dos processos, criação de topologia, funções de fog e subscrições não tem exposição de APIS logo não se pode mapear;

- Nível baixo de segurança com relação ao dados mesmo utilizando as ferramentas apresentadas. Para cobrir este requisito vejo a necessidade de se criar políticas ao nível dos fornecedores de dados.

Pode-se dizer, em linhas gerais, que este modelo proposto pela FIWARE atende parcialmente os requisitos de segurança identificados para uma plataforma de Internet of Things (**IoT**) com uma interdependência alta de componentes de terceiros. Dentre os critérios identificados na análise como não atendidos e parcialmente atendidos, identificou-se que os temas de Responsabilidade (ACC) e Auditabilidade (AUD) serão passíveis de serem atendidos na arquitetura de segurança existente uma vez que, já se tem os dados e informações relevantes para que estes possam ser atendidos e proporcionar uma melhoria significativa para a ferramenta. Os demais critérios na visão extraída desta análise, exigem uma reformulação da arquitetura proposta para serem implementados e assim tornar a camada de segurança robusta ao nível de atender estes critérios.

Alguns critérios para ter um tratamento efetivo devem estar contidos nas aplicações que tem o conhecimento das entidades e podem gerir estes processo como por exemplo o tratamento para disponibilidade e integridade das aplicações. Diante neste trabalho, será dada continuidade com uma profundidade maior na análise dos critérios de Responsabilidade(Accountability) e Auditabilidade(Auditability) no âmbito de identificar e propor uma solução para atender estes critérios sem muitas alterações na arquitetura de segurança proposta atualmente pela FIWARE.

Capítulo 6

Proposta de solução

Na análise realizada, identificou-se que as ferramentas recomendadas na arquitetura de segurança proposta pela FIWARE, não possuem funcionalidades que satisfaçam na sua totalidade os critérios utilizados na análise de segurança para plataformas de Internet of Things (IoT), apresentados no Capítulo 4.3. Neste trabalho selecionaram-se dois critérios, Accountability (Responsabilidade) e Auditability (Auditabilidade), que atualmente não são atendidos na sua totalidade e tem relação direta na adoção de uma plataforma de IoT no âmbito desta estar em conformidade com os aspetos legais, tais como Regulamento Geral de Proteção de Dados (RGPD), Directive 95/46/EC e Organization for Economic Cooperation and Development (OECD).

Conceptualmente os critérios serão representados em forma de requisitos macros para expressar o que éra desejável encontrar nas ferramentas recomendadas.

• Responsabilidade / Accountability (Acc) - Requisitos Macros

- Fornecer meios para monitorizar as ações das entidade dentro do sistema, com a possibilidade de detetar e alertar em casos de mau funcionamento (erros e situações não previstas);
- Definir responsabilidades e sanções paras as entidades (o que pode fazer e o que acontece se fizer diferente do expectável);
- Formas de demonstrar para o utilizador o fluxo das informações e entidades do sistema (ter transparência nas informações);
- Atribuir responsabilidades sobre as ações tomadas por cada entidade (ou seja, definir responsáveis);
- Fornecer mecanismos para inferir de de forma pro-ativa nos processos e aplicar as sanções quando necessário (monitoria e alertas);
- Disponibilizar consulta ao registo das operações em tempo real ou near real time(logs).

• Auditabilidade / Auditability (Aud) - Requisitos Macros

- Fornece mecanismos para "registar" as interações da entidade e utilizadores da plataforma;
- Ter mecanismos para permitir o rastreio das atividades na plataforma (auditoria);
- Ter mecanismos para refazer as ações e interações de forma a fornecer evidências do que foi realizado para uma entidade externa;
- Fornecer garantias de que as informações não foram adulteradas e o sistema é fiável.

6.1 Propostas

Para se atender aos requisitos apresentados na sessão anterior, foi proposta a alterações nas ferramentas Keyrock (Identity management (**IDM**)) e Wilma (Policy Enforcement Point (**PEP**)), responsáveis pelos controlos e realização de acesso as entidades e funcionalidades, para implementar as funcionalidades de log e rastreio de pedidos de forma a atender os requisitos.

O rácio para escolha destas ferramentas tem como base a arquitetura de segurança existente para a plataforma FIWARE, onde identificou-se três atores que podem interagir com a plataforma (utilizador, Internet of Things (**IoT**) Device, **PEP**) sem gerar registo destas interações. O Utilizador tem as credenciais para realiza os pedidos e ter acesso ao sistema, o **PEP** e o **IoT** Device são atores que fazem parte de uma aplicação e também possuem credenciais para interagir com o sistema de autenticação.

Descrevo as alterações propostas para as ferramentas de forma a atender o problema apresentado no inicio deste Capítulo.

- **Keyrock (IDM)**, possui o conceito de uma aplicação ter recursos, utilizadores e ou organizações, **IoT** Device e PEPs Proxy. Para um nível mais robusto de segurança, recomenda-se:
 - Quando se cria um utilizador, é expectável expor como serão tratadas e armazenadas as suas informações, quais são as suas permissões e nível de acesso(mostrar ou ter formas de consultar);
 - Ao se incluir/remover um utilizadores/organização em uma aplicação é necessário ter o registo deste pedido e também formas de consultas as permissões que o utilizador passa a ter com esta ação;
 - Incluir a gestão de consentimentos do utilizador no ato da recolha das informações e ter de forma clara quais informações serão armazenadas(IP/Tokens/URI/etc..), os fins e o período de retenção;
 - Armazenar as alterações realizadas nas aplicações de forma a possibilitar uma reconstrução de eventos ex: ID de dispositivos, formatos dos tokens e outros atributos relevantes para o efeito;

- Na etapa de autenticação do PEP Proxy, validar a url da aplicação (requerer esta informação do proxy para validar em tempo de autenticação).
- **Wilma PEP** é a ferramenta que permite o utilizador utilizar as aplicações, com a aplicação de algumas regras e validações, recomenda-se:
 - Ter o registo (Log) das chamadas realizadas pelo utilizador incluindo as informações necessárias para identificar: o utilizador, origem e destino do pedido, msg de retorno (o body pode ser encriptado);
 - Alterar o formato de logs atual para um formato standard e.g. Elastic Common Schema (ECS) tornando compatível com as ferramentas existentes de gestão de logs.
 - Permitir a configuração de onde serão armazenadas as Logs(local ou um repositório);
 - Gerar na sua inicialização uma chave para assinar as logs de forma a garantir que os mesmo são legítimos e a partir deste momento não podem ser alterados;
 - Enviar a url que esta no ficheiro de configuração para o IDM no ato de autenticar(validar se corresponde a aplicação solicitada);
 - Gerar um novo header que ira conter um identificador único do utilizador a ser incluídos nas chamadas as outras ferramentas para conseguir ter uma visibilidade do fluxo de chamadas realizados pelo utilizador durante o uso da plataforma/token
- **Wilma e Keyrock** recomenda-se que passem a utilizar um novo header (TraceID) com um ID a ser incluído nas logs para ter a rastreabilidade do fluxo end to end.

6.2 Arquitetura

A arquitetura de segurança proposta tem como base a recomenda para o fogflow. Foi incluído um componente no nó de cloud (centralizado) para receber e armazenar os dados de comunicação e logs, mitigando problemas de armazenamento e acesso aos componentes para consultas de dados. Pensou-se em uma contingência onde as informações podem ser temporariamente mantidas no dispositivo e depois transmitidas. O Grande objetivo é assegurar que toda a comunicação dos utilizadores com os aplicativos da plataforma sejam realizadas através do Policy Enforcement Point (PEP) e que as suas informações e seus consentimentos sejam registados e disponibilizados para consultas futuras.

Para ter melhor compreensão, pode-se observar na Figura 6.1 a arquitetura de segurança recomenda para o FogFlow, onde pode-se perceber o cenário atual da aplicação instanciada com os componentes de segurança implementados.

A seguir pode-se observar na figura 6.2 a arquitetura proposta e desenhada para atender a aplicação com base na solução proposta na secção 6.1.

Abaixo descrevo as alterações necessárias para cada ferramenta, que estão ilustradas com os fluxos e desenhos de solução inseridos na sessão 6.3.1.

Keyrock

- Alterar o modelo de dados:
 - Novas tabelas para contemplar o processos de gestão de consentimentos;
 - Alterar tabelas para contemplar o controlo de versões para os dispositivos, proxy e aplicações;
 - Incluir os campos para contemplar o conceitos equipamentos ativos e desativados;
- Alterar a aplicação para:
 - Criar ecrãs para solicitar e armazenar o consentimento do utilizador;
 - Alterar ecrãs de criação e gestão de aplicações para suportar a funcionalidade de desativar uma aplicação, dispositivos ou proxy.
 - Ajustar as APIS para receber no endpoint de autenticação do PEP Proxy (POST /v3/auth/tokens) a url da aplicação e validar a mesma contra a informação existente no cadastro da aplicação.

Wilma

- Alteração na Aplicação:
 - Incluir a geração de Logs e autenticação para todos os pedidos recebidos, contendo : Headers, Id do host, Id do utilizador, URL e Metodo, Retorno do Pedido, Status Http code, Dados do agente solicitante, Body do pedido(se houver) e Assinatura do gerador.
 - No fluxo de autenticação com o IDM, enviar a URL da aplicação (informação existente no ficheiro e variáveis de configuração)
 - No ficheiro de configuração, permitir selecionar onde serão armazenadas as Logs (pasta local ou informar o endereço de um repositório).
 - No ficheiro de configuração assim como para o uso de HTTPS, permitir informar um certificado ou chave para utilizar na autenticação do Log.

6.3.1 Desenho da Solução e Fluxos

Nesta sessão podes visualizar os novos fluxos bem como o desenho de solução para ilustrar a proposta de solução apresentada.

Na Figura 6.3, pode-se visualizar o fluxo a ser executado por uma aplicação que seja instanciada com os as melhorias implementadas na solução proposta.

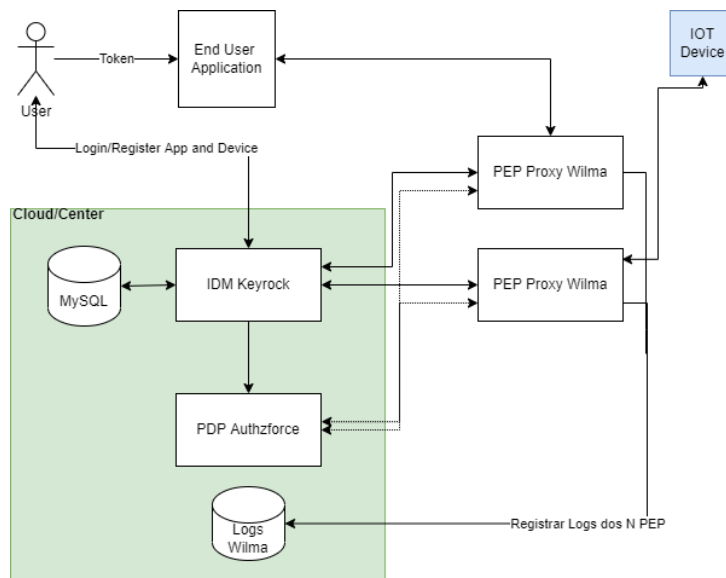


Figura 6.3: Protótipo - Exemplo de uma Aplicação

O Log a ser gerado pela solução proposta, Figura 6.4, será compatível com as ferramentas de gestão de logs e será disponibilizado no formato JSON e vai ter uma HASH para garantir a autenticidade.

Logs do Wilma

```

{
  "@timestamp":"2022-06-13T13:44:38.419Z",
  "log_level":"info",
  "message":"ffff:192.168.1.146 -- [13/Jun/2022:13:44:38+0000] \"GET / HTTP/1.1\" 401 141 1 \"-\" \"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36\"",
  "http":{
    "version":"1.1",
    "request":{
      "method":"GET",
      "headers":{
        "host":"192.168.1.127:5555",
        "connection":"keep-alive",
        "cache-control":"max-age=0",
        "upgrade-insecure-requests":"1",
        "user-agent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36",
        "accept":"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9",
        "accept-encoding":"gzip, deflate",
        "accept-language":"pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7"
      }
    },
    "response":{
      "status_code":401,
      "headers":{
        "access-control-allow-origin":"","",
        "access-control-allow-credentials":"true",
        "www-authenticate":"IDM uri = undefined",
        "content-type":"application/json; charset=utf-8",
        "content-length":"141",
        "etag":"W/\"5d-llsdj79MbnV7jpulNPWmpxY3QVQI\""
      },
      "body":{
        "bytes":141,
        "data":"XPTO XPTO"
      }
    }
  },
  "uri":{
    "path":"/",
    "domain":"192.168.1.127",
    "full":"http://192.168.1.127:5555/"
  },
  "client":{
    "address":"ffff:192.168.1.146",
    "ip":"ffff:192.168.1.146",
    "port":31056
  },
  "user_agent":{
    "original":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/102.0.5005.63 Safari/537.36",
    "sign":"0xb613679a0814d9ec772f95d776c35fc5ff697c493715653dc712144292c5ad"
  }
}

```

Figura 6.4: Protótipo

O Fluxo de autenticação para os utilizadores não deve ter impactos para o utilizador e o que será implementado, conforme a Figura 6.5, será a geração de Logs desta autenticação e validação do seu consentimento.

O Fluxo de autenticação dos dispositivos será implementado, conforme a Figura 6.6, com a

Fluxo de Autenticação - User

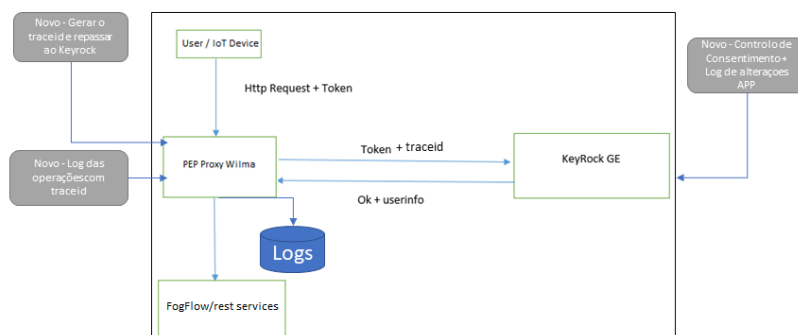


Figura 6.5: Prototipo

geração de Logs da autenticação.

Fluxo de Autenticação - Dispositivo

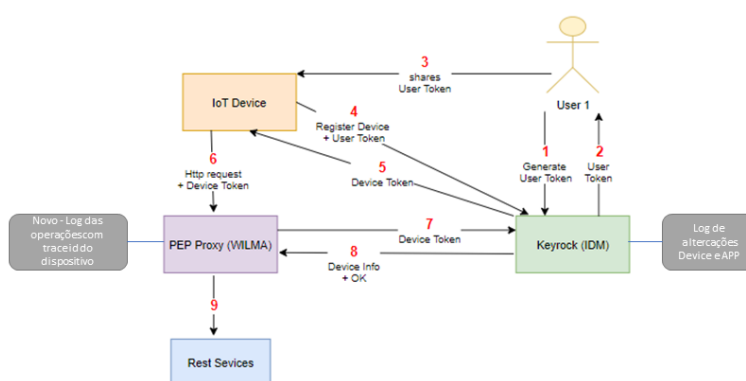


Figura 6.6: Prototipo

O Modelo de dados necessita de novas tabelas e campos para suportar nas novas funcionalidades, na Figura 6.7 podemos visualizar o modelo de dados atual do Keyrock e a indicação das novas tabelas.

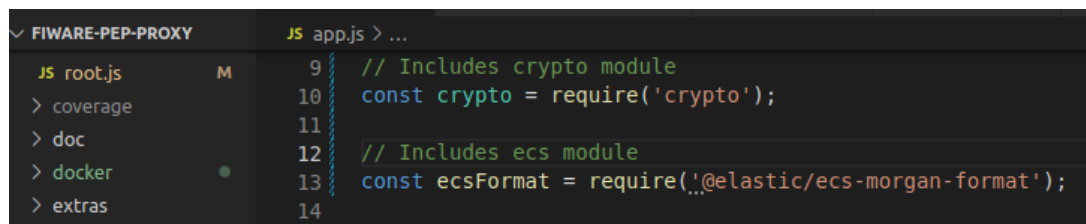
das informações. A Figura 6.9 apresenta um exemplo de uma resposta formatada.

```
{
  "@timestamp": "2021-01-16T00:03:23.279Z",
  "log.level": "info",
  "message": ":::1 - - [16/Jan/2021:00:03:23 +0000] \"GET / HTTP/1.1\" 200",
  "ecs": {
    "version": "1.6.0"
  },
  "http": {
    "version": "1.1",
    "request": {
      "method": "GET",
      "headers": {
        "host": "localhost:3000",
        "accept": "*/*"
      }
    },
    "response": {
      "status_code": 200,
      "headers": {
        "x-powered-by": "Express",
        "content-type": "text/html; charset=utf-8",
        "etag": "W/\"d-HwnTDHB9U/PRbFMN1z1wps51lqk\""
      },
      "body": {
        "bytes": 13
      }
    }
  },
  "url": {
    "path": "/",
    "domain": "localhost",
    "full": "http://localhost:3000/"
  },
  "user_agent": {
    "original": "curl/7.64.1"
  }
}
```

Figura 6.9: Exemplo Log - ECS

O código fonte foi alterado para exibir mais informações no log das chamadas que atualmente utiliza o package Morgan e esta configurado por defeito para o formato **"dev"**, que retorna método, Url, status, tempo de resposta e tamanho da resposta. Foi então criada uma nova sessão para gerar estas logs com o formato **"combined"**, que retorna o endereço de origem, user, data, método, url e versão do protocolo Http, status, tamanho da resposta, header de referencia (se houver) e informações do agente que realizou a chamada. Seguindo com as alterações, foi instanciado um componente criptográfico para gerar um código para garantir a integridade da mensagem e utilizou-se o **HMAC** uma forma de autenticar a mensagem com uma chave que apenas quem originou conhece e mitiga a possibilidade de um adversário recalculer o Message Authentication Code (**MAC**) do registro de log e também foi incluído o pacote Elastic Common Schema (**ECS**) Logging with Morgan para deixar o registro de log alinhado ao um padrão. A

Figura 6.10 apresenta a alteração realizada no código.



```

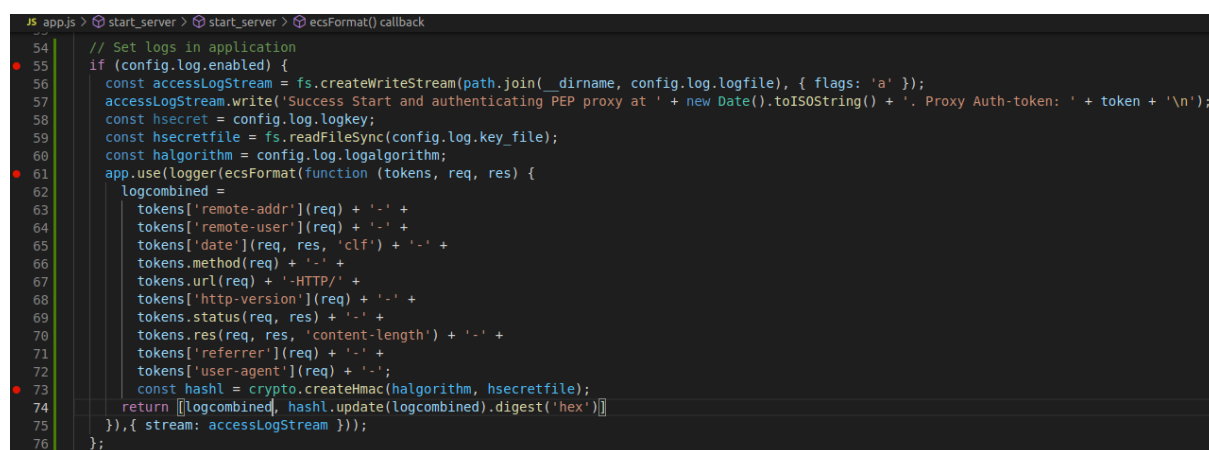
JS root.js M
> coverage
> doc
> docker
> extras

JS app.js > ...
9 // Includes crypto module
10 const crypto = require('crypto');
11
12 // Includes ecs module
13 const ecsFormat = require('@elastic/ecs-morgan-format');
14

```

Figura 6.10: PEP PROXY WILMA - includes app.js

Foi possível gerar o log em ficheiros e incluir a hash com a ajuda da funcionalidade de formatar o log e utilizar funções pre-definidas que o pacote Morgan chama de tokens [6] e assim possibilitou que fossem geradas as hash sobre as mensagens de logs. A Figura 6.11 apresenta a alteração realizada no código.



```

JS app.js > start_server > start_server > ecsFormat() callback
54 // Set logs in application
55 if (config.log.enabled) {
56   const accessLogStream = fs.createWriteStream(path.join(__dirname, config.log.logfile), { flags: 'a' });
57   accessLogStream.write('Success Start and authenticating PEP proxy at ' + new Date().toISOString() + '. Proxy Auth-token: ' + token + '\n');
58   const hsecret = config.log.logkey;
59   const hsecretfile = fs.readFileSync(config.log.key_file);
60   const halgorithm = config.log.logalgorithm;
61   app.use(logger(ecsFormat(function (tokens, req, res) {
62     logcombined =
63       tokens['remote-addr'](req) + '-' +
64       tokens['remote-user'](req) + '-' +
65       tokens['date'](req, res, 'clf') + '-' +
66       tokens.method(req) + '-' +
67       tokens.url(req) + '-' +
68       tokens['http-version'](req) + '-' +
69       tokens.status(req, res) + '-' +
70       tokens.res(req, res, 'content-length') + '-' +
71       tokens['referrer'](req) + '-' +
72       tokens['user-agent'](req) + '-';
73     const hashl = crypto.createHmac(halgorithm, hsecretfile);
74     return [[logcombined, hashl.update(logcombined).digest('hex')]];
75   })), { stream: accessLogStream }));
76 }

```

Figura 6.11: PEP PROXY WILMA - logs app.js

De forma a garantir uma compatibilidade e melhorar a visualização da Log, foi implementada a chamada ao pacote ECS Logging with Morgan que também preciso ser alterado para contemplar a Hash de forma a ter o registro formatado com as informações necessárias. A Figura 6.12 apresenta a alteração realizada no código.

Apos as alterações foi realizado um teste unitário para validar o resultado esperado e foi otido um registo de Logs com as informações esperadas, conforme pode ser visualizado na Figura 6.13

Optamos por não implementar as alterações no Keyrock neste trabalho, isso será ser abordado em trabalhos futuros. Deixo como recomendação a implementação de um processo para gestão de chaves e a sua utilização em tempo de execução, cito o exemplo o HashiCorp Vault, para mitigar temas como extravio de chaves guardadas em ficheiros.

```

node_modules > @elastic > ecs-morgan-format > JS index.js > ecsFormat > formatter
102
103   return function formatter (token, req, res) {
104     const ecsFields = {
105       'timestamp': new Date().toISOString(),
106       'log.level': res.statusCode < 500 ? 'info' : 'error',
107       message: fmt(token, req, res),
108       ecs: { version }
109     }
110
111     if (serviceField) {
112       ecsFields.service = serviceField
113     }
114     if (eventField) {
115       ecsFields.event = eventField
116     }
117     //Insert hash
118     ecsFields.loghash = ecsFields.message[1];
119     ecsFields.message[1] = null;

```

Figura 6.12: PEP PROXY WILMA - logs ecs app.js

```

1 {
2   "@timestamp": "2022-09-29T00:11:42.057Z",
3   "log.level": "info",
4   "message": "::ffff:192.168.1.128-undefined-29/Sep/2022:00:11:42 +0000-GET-/-HTTP/1.1-401-141-undefined-Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0-,"
5   "ecs": {
6     "version": "1.6.0"
7   },
8   "http": {
9     "version": "1.1",
10    "request": {
11      "method": "GET",
12      "headers": {
13        "host": "192.168.1.128:5555",
14        "user-agent": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0",
15        "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8",
16        "accept-language": "en-US,en;q=0.5",
17        "accept-encoding": "gzip, deflate",
18        "connection": "keep-alive",
19        "cookie": "_csrf=9JHisz0sby8L46cw4k2iXuW8",
20        "upgrade-insecure-requests": "1",
21        "x-auth-token": "4780c9dfa6956355120f001f5329afd2a100defc"
22      }
23    },
24    "response": {
25      "status_code": 401,
26      "headers": {
27        "access-control-allow-origin": "*",
28        "access-control-allow-credentials": "true",
29        "content-type": "application/json; charset=utf-8",
30        "content-length": "141",
31        "etag": "W/\"8d-F8alhAxvhvnmFE+JUhQFXmjgAyhs\""
32      },
33      "body": {
34        "bytes": 141
35      }
36    }
37  },
38  "url": {
39    "path": "/",
40    "domain": "192.168.1.128",
41    "full": "http://192.168.1.128:5555/"
42  },
43  "client": {
44    "address": "::ffff:192.168.1.128",
45    "ip": "::ffff:192.168.1.128",
46    "port": 55692
47  },
48  "user_agent": {
49    "original": "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:104.0) Gecko/20100101 Firefox/104.0"
50  },
51  "loghash": "5848e8afe0257c2e64b6754d78091836ae2e7bb09d9578dca6d6b7491214d74b"
52 }

```

Figura 6.13: PEP PROXY WILMA - Final

Capítulo 7

Experimentação e Testes

Neste capítulo será demonstrada a implementação de algumas alterações nas ferramentas da FIWARE a título de validar e testar partes da solução proposta no âmbito acadêmicos.

Serão abordados temas como a preparação do ambiente, instanciar a aplicação, realizar a recolha dos dados e testes de performance com e sem o mecanismos propostos para atender os critérios destacados na aviação realizada e descrita no capítulo 5.

7.1 Ambiente

O Ambiente utilizado foi construído sobre o S.O Linux, distribuição Ubuntu v20.04.2, composto das ferramentas:

- Identity management (**IDM**) Keyrock v8.0.0
- MySQL 5.7.21-1.1.3
- Policy Enforcement Point (**PEP**) Wilma PeP proxy v8.1.0,
- Policy Decision Point (**PDP**) Authzforce release-9.0.1,
- Fogflow (aplicação FIWARE).
- VS CODE (IDE) v1.66.2,
- Node.JS v10.19.0
- NPM v 6.14.4n
- ECS Logging Node.js package @elastic/ecs-morgan-format@1.1.0
- Docker v20.10.6 e Docker compose v1.27.4
- Postman v9.0.9,
- Firefox v104.0
- Modify Header Value (HTTP Headers) v0.1.8 - extensão para manipulação de headers.

- Apache JMeter v5.5

Para a configuração do ambiente, foi utilizada a instalação do **IDM** Keyrock utilizando docker containers e uma versão local do **PEP** Proxy Wilma com as alterações realizadas.

O fluxo sugerido é iniciar o Keyroc (utilizar o docker-compose) e aceder ao aplicação, que por defeito utiliza a porta 3000, para criar um novo utilizador e uma nova aplicação para testes.

Após a inicialização do Keyrock, é necessário configurar o Wilma, uma vez que a aplicação já existe no keyrock, inicia-se a etapa de configuração do Wilma PEP proxy com a inclusão do ID e Senha do proxy no ficheiro de configuração e start da aplicação a correr por defeito na porta 5555.

Para teste e validação do ambiente, realizou-se chamadas a url do Wilma via browser. O resultado esperado e obtido foi receber uma pagina de erro no browser com a mensagem a informar que não estamos a utilizar o header X-AUTH-TOKEN necessário para autenticar na aplicação.

7.2 Testes

Para este trabalho elaborou-se alguns casos de testes, listados abaixo, a titulo de validar o prototipo e ter uma comparação da performance pós alteração.

Casos para testar os novos parâmetros do ficheiro de configuração:

- **Habilitar e desabilitar os Logs de execução:** Com este caso de teste será possível validar se a log somente é gerada quando habilitadas. O resultado esperado é que quando não esta habilitada, não gera impacto relevante para o funcionamento da aplicação.
- **Definir a origem a ser armazenado o ficheiro:** Com este caso será possível validar o compoartamento da aplicação quando tem a indicação para gerar o ficheiro e o que acontece quando a mesma não é informada. O resultado esperado é que no ficheiro de configuração quando não foi informada a localização e o nome do ficheiro, a aplicação se comporte da forma esperada, ou seja, as logs devem ser geradas na pasta Logs dentro do diretório da aplicação em um ficheiro com um nome gerado por defeito seguindo o formato por defeito LogDDMMAAAA.log
- **Definir a chave hmac:** Com este caso pretende-se validar as formas incluídas para definir a chave de cifra utilizada. A aplicação esta preparada para receber a chave de duas formas sendo informada no ficheiro de configuração de forma explicita, no parâmetro logkey, ou sendo informado um ficheiro que ira conter a chave a ser utilizada. O resultado que se pretende é perceber se a aplicação utilizara por defeito a chave informada no ficheiro externo referenciado no ficheiro de configuração e quando este não existir, busca a chave

informada no ficheiro de configuração e caso não encontre gera um erro e não carrega a aplicação por ser um parâmetro obrigatório quando o log está ativado.

- **habilitar as logs da Aplicação pós alteração:** Com este caso de teste, pretende-se verificar a execução da nova versão com os parâmetros específicos para log habilitados e configurados no ficheiro de configuração, descritos na Figura 6.8. O resultado esperado e obtido ao habilitar esta secção é verificar que a aplicação gerou um ficheiro com o log da conexão com o Identity management (IDM) e também as chamadas Http no formado Elastic Common Schema (ECS), ilustrado na Figura 6.4, que apresenta um log em formato JSON amigável e manteve as informações exibidas na versão original como o Logs de chamadas Http que são apenas exibidas na consola da aplicação com as informações "method, url, status, response-time em ms e res[content-length]".

Casos para medir os tempos de processamento:

- **Chamadas sem envio do Header X-Auth-Token:** Com este caso de teste será possível obter o tempo de resposta da aplicação a uma chamada que não será validada nos autenticadores. O resultado esperado é que a alteração não gere um aumento significativo no tempo de execução.
- **Chamadas com o Header X-Auth-Token, e um token invalido:** Com este caso de teste será possível obter o tempo de resposta da aplicação a uma chamada que será validada nos autenticadores. O resultado esperado é que a alteração não gere um aumento significativo no tempo de execução.
- **Chamadas com o Header e um Token Valido:** Com este caso de teste será possível obter o tempo de resposta da aplicação a uma chamada que será validada nos autenticadores e vai ter o acesso a aplicação realizado. O resultado esperado é que a alteração não gere um aumento significativo no tempo de execução e impactos para a aplicação que será executada.

Todos os casos de teste foram executados com a aplicação Wilma PEP Proxy versão 8.1.0 instalada localmente na VM. Os testes para medir o tempo de processamento foram realizados com as versões:

- 8.1.0 Original(sem alterações);
- 8.1.0 Alterada (sem ativação dos Logs)
- 8.1.0 Alterada (com Logs ativados)

7.2.1 Resultados

Os casos de teste para os novo paraâmetros, foram execução e os resultados obtidos foram os esperados, ou seja, a alteração não impactou o funcionamento atual da aplicação e a ativação e

desativação dos parâmetros mostrou-se uma opção viável para os cenários de utilização atual da aplicação.

Os casos de teste para mensurar a performance e impactos das alterações, foram executados e os resultados obtidos podem ser interpretados como dentro dos limites. O teste foram executados com o auxílio do JMeter (aplicação para teste funcionais e medir performance) e apresentaram uma subida no tempo de resposta da aplicação principalmente nos casos em que se tem todas as novas funcionalidades implementadas (logs e cifras). Para o melhor entendimento apresento um breve resumo dos testes e alguns gráficos a título de ilustrar os resultados.

Os casos em que foram testadas as chamadas com erro, token invalido, foi possível observar que mesmo tendo as novas funcionalidades o comportamento da aplicação teve uma subida ligeira no seu tempo médio que podemos justificar sendo o tempo necessário para executar as novas funções. Na Figura 7.1, pode-se visualizar os tempos de execução para o cenário de 50 chamadas em simultâneo a aplicação.

Testes – Comparação – Casos de Erro (Token Invalido)

Sem alterações										
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	12	9	19	2.39	100.00%	50.3/sec	20.29	8.69	413.0
TOTAL	50	12	9	19	2.39	100.00%	50.3/sec	20.29	8.69	413.0

Logs inativos										
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	12	9	27	3.44	100.00%	50.5/sec	20.37	8.48	413.0
TOTAL	50	12	9	27	3.44	100.00%	50.5/sec	20.37	8.48	413.0

Logs Ativos										
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	50	14	8	65	10.50	100.00%	50.4/sec	20.33	8.12	413.0
TOTAL	50	14	8	65	10.50	100.00%	50.4/sec	20.33	8.12	413.0

Figura 7.1: Resultados Testes NOK

Os casos em que foram testadas as chamadas com sucesso, foi possível observar que com as novas funcionalidades o comportamento da aplicação teve uma subida expressiva no seu tempo médio que podemos justificar sendo o tempo necessário para executar as novas funções em especial posso dizer que a geração de hash tem um peso neste tempo. Na Figura 7.2, pode-se visualizar os tempos de execução para o cenário de 50 chamadas em simultâneo a aplicação.

Quando se compara os resultados, de todos os testes executado e separa-se por casos de erros e sucessos, podemos ter uma visão melhor sobre a alteração e os tempos de resposta que acredito ser um ponto de atenção com relação as novas funcionalidades implementadas pois mesmo sabendo dos benefícios não podemos ignorar os temas de experiência do utilizador e também o desempenho da aplicação. Destaco que algumas configuração podem tem um impacto direto neste requisito cito a a escolha do algoritmo de cifra influencia diretamente uma vez que este processo tem um tempo de processamento ligeiramente alto e exige um pouco mais da maquina para ser concluído. Na Figura 7.3, pode-se visualizar o gráfico com a comparação dos tempos de execução, em milissegundos, onde-se nitidamente notamos que a versão com sucesso

Testes – Comparação – Casos de Sucesso

Sem alterações											
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes	
HTTP Request	50	7	4	56	8.41	0.00%	50.7/sec	151.83	8.77	3066.0	
TOTAL	50	7	4	56	8.41	0.00%	50.7/sec	151.83	8.77	3066.0	

Logs inativos											
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes	
HTTP Request	50	15	4	103	23.09	0.00%	50.9/sec	152.30	8.79	3066.0	
TOTAL	50	15	4	103	23.09	0.00%	50.9/sec	152.30	8.79	3066.0	

Logs Ativos											
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes	
HTTP Request	50	35	4	236	59.37	0.00%	50.7/sec	151.83	8.77	3066.0	
TOTAL	50	35	4	236	59.37	0.00%	50.7/sec	151.83	8.77	3066.0	

Figura 7.2: Resultado Testes OK

tem uma subida no tempo total de processamento mesmo ao utilizar o popular SHA256 para gerar a hash ainda assim temos quase o dobro de tempo gasto no seu ciclo de processamento.

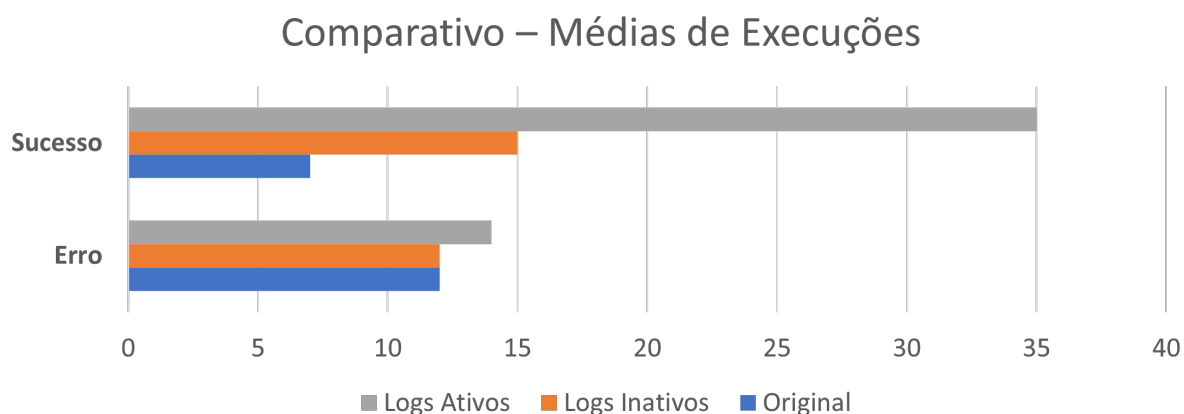


Figura 7.3: Gráfico Comparativo

Com estes resultados podemos acordar que esta alteração funciona e precisa ser revisado alguns pontos para que seja possível otimizar a performance sem perder os requisitos de segurança implementados com a adição destas novas funcionalidades.

Capítulo 8

Conclusão

O ecossistema de Internet of Things (IoT) ainda é uma tecnologia em evolução, que tem muitas oportunidades para propor novas soluções e validar a sua aplicabilidade nos cenários e casos de usos existentes.

Ainda vejo como um grandes desafios a construção de uma arquitetura de segurança descentralizada, onde os componentes sejam autónomos e seja quebrada a barreira de ser um nó central passando a a ter um conjunto de nós trabalhando entre si, tal como é feito nas Distributed Ledger Technology (DLT) eg Blockchain com os conceitos de eleição, e ter as garantias de segurança no armazenamento de dados e uso das informações que, além dos repositórios de dados, se estende aos dispositivos (sensores e atuadores) e os consumidores que possuem necessidades distintas de uso das informações tratadas e armazenadas seja em nós repositórios de dados, memória temporária ou permanente. Sempre considerando os requisitos de garantir a performance da aplicação.

Após os estudos realizados para elaboração desta dissertação, ficou claro que, esta tecnologia, ainda possui requisitos essenciais que não são atendidos na sua totalidade. No que tange a segurança da informação, as ferramentas existente são recomendadas para serem utilizadas como uma "capa" para permitir a integração entre os diversas aplicações, cito o exemplo do fogflow na plataforma FIWARE. Isso em algumas situações permitem que os requisitos que são atendidos na visão *stand-alone* da ferramenta, possam não serem suportados na integração com a aplicação, conforme a arquitetura proposta, impossibilitando utilizar todos o benefícios oferecidos.

E ficou evidente que para se construir uma arquitetura de segurança que seja robusta e suporte as aplicações de IoT o conceito de incluir os requisitos de segurança no desenvolvimento da aplicação ainda continua a ser a melhor opção para ter o alinhamento das necessidades da aplicação com os requisitos necessários para atender os reguladores e dar garantias aos utilizadores de que esta aplicação é fiável.

Concluo este trabalho com o entendimento de que o tema de segurança da informação em IoT está em constante evolução e durante o período de pesquisas e testes constatei que as aplicações estão focadas na performance oferecem garantias básicas se assim posso dizer no âmbito da

segurança. Esta visão está a sofrer alterações face os avanços de ciberataques e expansão do uso de dispositivos **IoT** sendo o tema da segurança priorizado na escolha de plataformas e ferramentas para utilização e manipulação de dispositivos **IoT**. Aqui deixo mais um incentivo para incluir desde o design as preocupações com a segurança e como contributo a análise realizada em uma plataforma que no momento é tida como referencia.

8.1 Trabalho Futuro

Deixo como fonte de inspiração as tabelas com as análises de segurança individuais e em conjunto, dos componentes recomendados pela FIWARE para garantir e montar uma arquitetura de segurança para as aplicações desenvolvidas na plataforma, que podem ser utilizadas como base e um ponto de início na proposta de novos requisitos aplicacionais que possam dar respostas a estes critérios não atendido e parcialmente atendidos.

Outro contributo é a experimentação feita dos componentes na arquitetura proposta para o fogflow, na perspectiva de se oferecer segurança em ambientes híbridos onde temos, novos requisitos que justificam uma análise das ferramentas nesta perspectiva.

Por fim, deixo o resultado das análises com os requisitos não atendimentos que não foram tratados neste trabalho, já endereçados para serem tratados em trabalhos futuros, pode-se explorar soluções para que as aplicações estejam em conformidade com requisitos ou aspetos legais e funcionais de cada aplicação com o selo FIWARE.

Apêndice A

Apêndice-a

A.1 Funcionamento Fogflow

A.1.1 Componentes

O FogFlow esta desenhado logicamente em camadas de serviços, cada uma com componentes distintos que podem ter o seu deploy na cloud (nó central) ou nos Edge (nós próximos ao dispositivo).

A Figura A.1 apresenta a arquitetura da ferramenta com as suas camadas, principais componentes e um breve fluxo.

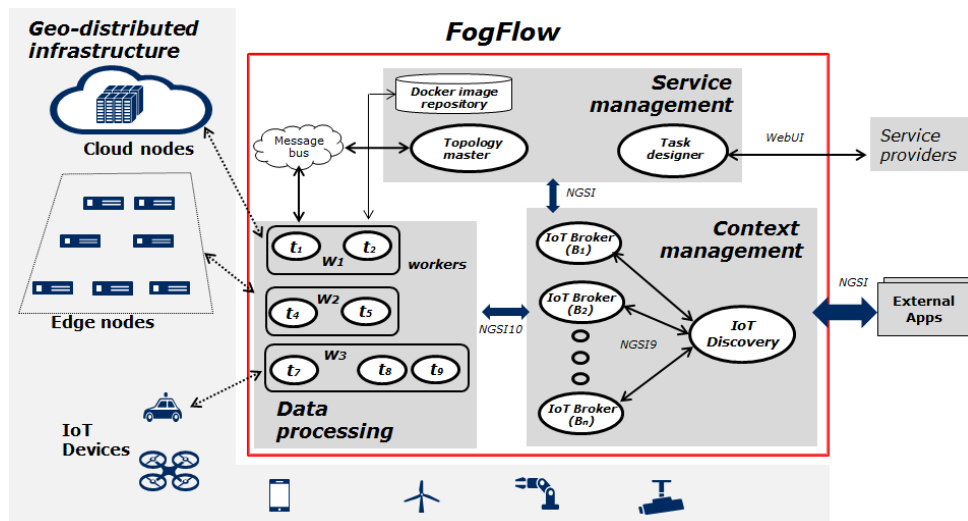


Figura A.1: FogFlow Arquitetura (fonte:[10])

As camadas de serviço, Service management, Context Management e Data processing estão descritas abaixo:

Service management, converte os pedidos, que pode ser uma topologia ou uma

tarefa, em planos de execução e os implementa nos nós (cloud ou edge).

Context management, faz a gerência de informações de contexto, dados de dispositivos cadastrados e nós existentes, e as deixa disponível para serem visualizadas via query ou enviada as interfaces subscritas.

Data processing, faz a execução das tarefas e controla o fluxo de dados entre elas e os seus consumidores.

A ferramenta na sua implementação standard, possuir os componentes descritos abaixo:

Task Designer, interface web para criar, modificar, excluir e registrar task e serviços de topologia.

Topology Master, serviço que faz a criação das instâncias, configura e decide onde deve ser implementada.

IoT Discovery, gerência as informações de contexto disponíveis como ID, tipos de entidades, lista de atributos e meta-dados e disponibiliza isso via the context data API of FIWARE NGSI ([NGSI9](#)).

Worker, responsável por executar as instâncias das tasks, em contêiner Docker, que estão agendadas para este nó (um nó pode ter mais de um worker). Configura os inputs e outputs e gerência as tarefas locais com base na prioridade.

IoT Broker, é um gerenciador de contexto das entidades que são publicadas pelos dispositivos Internet of Things ([IoT](#)) próximos a ele e também fornece uma visão das entidades para os dispositivos utilizarem.

Dock Registry, gerenciador de imagens dockers utilizadas.

RabbitMQ, faz a comunicação entre o topology master e workers.

PostgreSQL, database para salvar as informações de contexto.

A Tabela [A.1](#) apresenta um resumo dos componente por camadas e localização.

Tabela A.1: Componentes do FogFlow

Componentes	Camada	Deploy
Task Designer, Topology Master	Service Management	Cloud (nó central)
IoT Discovery	Context Management	Cloud
IoT Brokerr	Context Managemen	Cloud / Edge
Worker	Data Processing	Cloud / Edge
Dock Registry	Data Processing	Cloud
RabbitMQ	Data Processing e Service Management	Cloud/Edge
PostgreSQL		

A.1.2 Orquestração

Na sua essência, o FogFlow, é um ambiente composto por um Nó Central, na documentação é chamado de Cloud Node, e pode ter um ou mais Node próximos ao dispositivo, na documentação é chamado de Node Edge.

Para criar os Node Edge, é preciso fazer o deploy de alguns componentes (um Worker, IoT broker e um monitor agente metricbeat) no actuador. Entretanto, quanto o disposto não tem muita capacidade de processamento o Broker pode ser opcional, e faz-se a ligação a outro broker do sistema.

A sua integração com outros (Generic Enables (GE), Brokers e ou outras dispositivo que fornecem ou consomem contexto, utiliza uma Application Programming Interface (API) Next Generation Service Interface - Linked Data (NGSI-LD) que faz a conversão de NGSI9 e NGSI10 para NGSI-LD ou pode-se dizer NGSIv1 para NGSIv2, possibilitando a integração com os Brokers de contexto da FIWARE Scorpio, Orion context broker.

A Figura A.2 abaixo ilustra um deploy do sistema.

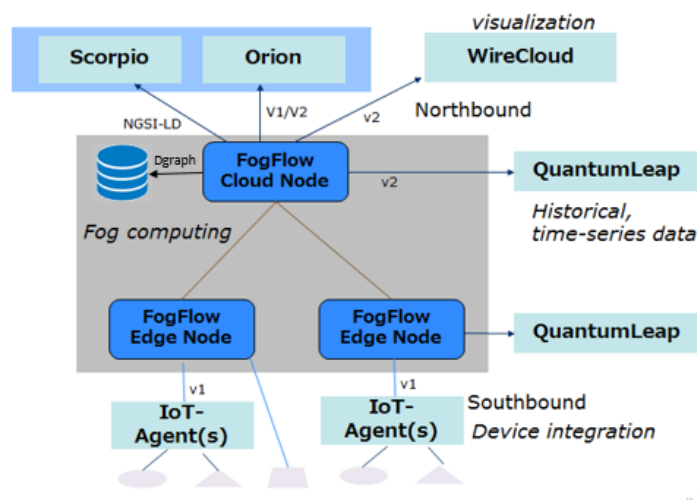


Figura A.2: FogFlow deploy (fonte:[11])

A integração com os dispositivos IoT é feita com base nos protocolos suportados e podemos classifica-las como:

NGSI SUPPORTED, uma integração nativa do FogFlow com o Dispositivo IoT e.g Atuador, que suporta o padrão Next Generation Service Interface (NGSI). Necessário fazer o registo, que pode ser realizado via interface ou um pedido http a API, e as informações são enviadas diretamente do dispositivo para o aplicativo e vice-versa.

NO-NGSI SUPPORTED, para dispositivo que não suportam o padrão NGSI, a plataforma FIWARE tem um IoT Agent que funciona como um intermediário ou tradutor entre o Broker FogFlow (No Cloud) e o dispositivo IoT neste caso o atuador.

Actualmente existe um **IoT Agent** específico para alguns protocolos como **IoT Agent JSON for MQTT based devices**, **IoT Agent UL for Ultralight Devices**, e Outros. Nota: Para este tipo de integração o dispositivo precisa ser registado no **IoT agent** que dispara o registo no **Broker**.

Já a comunicação utiliza os conceitos **Northbound e Southbound**, para a definição dos fluxos de dados a serem utilizados na comunicação entre os dispositivos e os Nodes.

Northbound, Fluxo de dados do sensor para o broker(no central)

Southbound, Fluxo de dados do broker(no central) para o sensor

A.1.3 Operacionalização (Como funciona)

De modo geral o processo de desenvolvimento de um serviço no FogFlow passa pela criação de um operador, definição da função a ser executada, criar uma topologia ou apenas as funções de fog e por fim enviar os resultados aos seus destinatários(subscritores/task/brokers). A Figura A.3 ilustra os passos necessários para criar um processo.

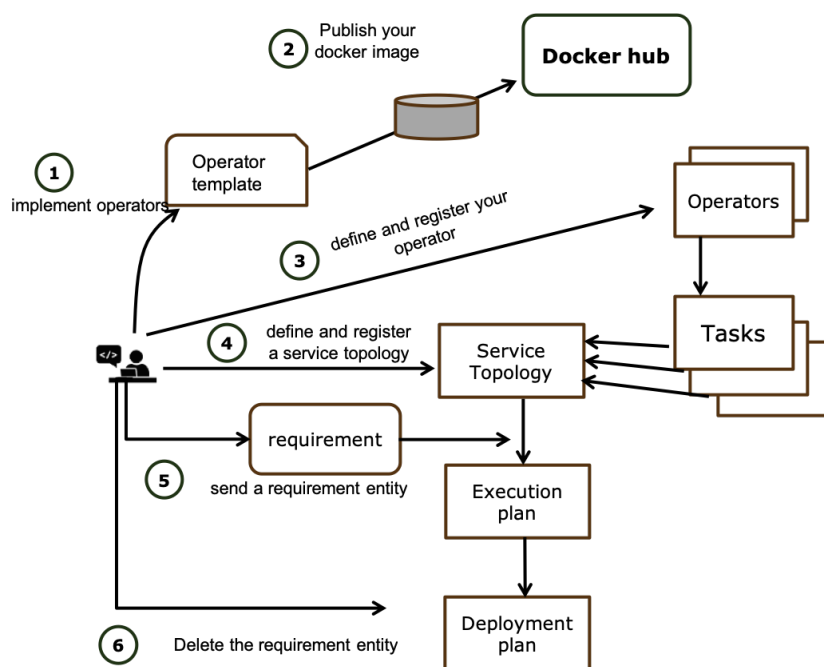


Figura A.3: FogFlow Funcionamento (fonte:[8])

Os dispositivos **IoT** para serem acedidos, necessitam ser previamente registados na ferramenta para ser atribuído um endereço a ser utilizado para as comunicações enviadas/recebidas.

Ao realizar o registo um dispositivo, é criada uma entidade(o dispositivo) com os atributos:

- tipo,

- id
- ispattern
- Lista de comando/operações aceito no device que são:
 - um nome (o comando),
 - type(descreve o tipo do atributo no exemplo command)
- A URI de acesso ao device e porta e por fim tem um parm de duração da comunicação.

Quando se utiliza um *IOT Agent*, é ele que envia os dados para o Broker que ele esta registrado funcionando como um intermediário para comunicação com os dispositivos.

Uma vez que se tenha o dispositivo registado no FogFlow e um fluxo que precise deste dado é detetado, o processamento é iniciado. Procura-se o atuador mais próximo do dispositivo, que possua o contexto solicitado, para ser mais rápido o processamento e não precisar ir buscar o contexto ao central.

A.1.3.1 Função de Fog

A utilização função de fog permite que se construa pequenos fluxos para o tratamento de dados. O FogFlow permite que mais de uma função utilize o mesmo dado e pode ser facilmente modificada, alterando os parâmetros da função e ou a própria função.

Esta funcionalidade permite que seja criada uma função especifica para um determinado tipo de informação, e o seu acionamento é feito automaticamente, sem a intervenção do utilizador.

Cada função possui uma topologia composta por uma Task, e uma ou mais EntityStream (parâmetros de input). Existem alguns parâmetros importantes na definição da função como: A GroupBy, possibilidade de se definir a quantidade de instâncias a serem executadas, informado na Task, o que permite lançar uma instância para tratar cada grupo de informações otimizando o tempo de execução. Por exemplo, ee tenho vários sensores de temperatura posso no limite ter uma instância por cada sensor se agrupar por ID. O Scoped, possibilidade de ser utilizada a localização para execução ou não, permitindo por exemplo executar uma função em broadcast para todos os dispositivos cadastrados com este tipo ou somente os que atendem a área especificada.

São verificadas as funções que podem utilizar os recursos e em seguida são acionadas as TASKs e as TaskInstance são executadas. Para controle de cada instanciá é acionado um worker que vai escutar o dispositivo e coletar os dados para executar a função.

Após ser submetida uma função de fog a ferramenta realiza:

- O acionamento da função de fog enviada, quando seus dados de entrada estiverem disponíveis;

- A verificação de quantas instâncias(task instance) serão criadas de acordo com sua granularidade definida;
- A escolha de onde implantar as instâncias criadas (em qual Node se vai na cloud ou no edge).

Internamente a execução e a orquestração é realizada tendo como base alguns parâmetros informados na criação da função. A Figura A.4 ilustra os passos de uma função de Fog.

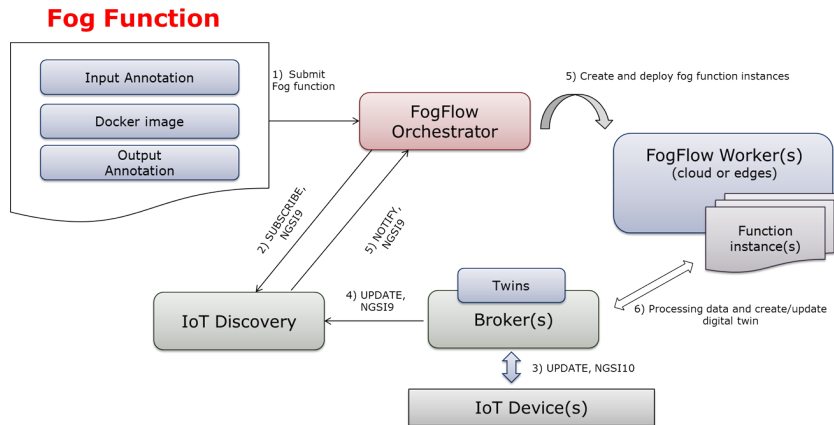


Figura A.4: FogFlow Função de Fog (fonte:[11])

No fogflow é possível verificar os dispositivos cadastrados e os seus parâmetros, quando uma função de Fog é iniciada, é gerada uma TaskInstance com as informações de execução e o seu resultado pode ser verificado na Stream onde vai conter o resultado da instância em execução.

A.1.3.2 Service Management - Service Topology

Esta funcionalidade permite ao utilizador implementar um fluxo de execução complexo e tem com o diferencial de que a sua execução é determinada pelo utilizador(aplicativo e ou consumidor). É necessária a criação de um objeto de intenção, gatilho, para definir os dados de inicialização e geo-localização da topologia.

Utiliza o gerenciador, topology master, aguarda que o sistema receba os dados de input para as tasks e validar se os mesmo estão em concordância com âmbito do intent object e assim executar as tarefas e faz o *deploy* das instâncias nos works correspondentes.

A Figura A.5 ilustra o processo de execução de uma topologia.

A.1.3.3 Context Management

Nesta camada ocorre a gestão de contextos e é realizada por dois componentes do FogFlow o IoT Discovery e IoT Broker.

possui “registro”. Este modo de funcionamento, exclui a necessidade de armazenar muitos dados e status das entidades para conseguir responder aos processos que são solicitados via pedidos no formato NGSI10(update/subscribe/query).

Os dados que cada broker gerência, também estão replicados no cloud Node (Central) e se houver alterações as mesmas são atualizadas no broker correspondente. Na Figura A.6 é possível visualizar um cenário com mais de um broker.

A.1.3.4 Data Processing - Worker

Este componente, worker, é um agente que tem a função de Iniciar/Terminar uma Instância e Adicionar/Remover um input stream para uma instância em execução.

Esta ligado a um Docker-engine local que o permite realizar as funções de criação do container para executar a instância e enviar as informações para parametrização da instância. O seu funcionamento pode ser descrito em sete passos:

1. o worker pede ao seu Docker engine local para criar um novo contêiner para executar a nova instância com um número de porta de escuta especificado;
2. A instância começa a ser executada e escuta no número de porta fornecido para receber fluxos de dados de entrada;
3. O worker envia um objeto de configuração para a instância em execução por meio de sua porta de escuta;
4. O worker emite a subscrição the context data API of FIWARE NGSI (NGSI10) em nome da instância em execução;
5. Os fluxos de dados de entrada necessários fluem para a instância em execução para processamento adicional por meio de sua porta de escuta;
6. A instância da tarefa em execução processa os dados de fluxo de entrada recebidos e, em seguida, gera seu resultado;
7. A instância da tarefa em execução publica o resultado gerado como sua saída, enviando atualizações NGSI10.

A Figura A.7 para ilustrar o funcionamento de um *worker* na execução de uma instância.

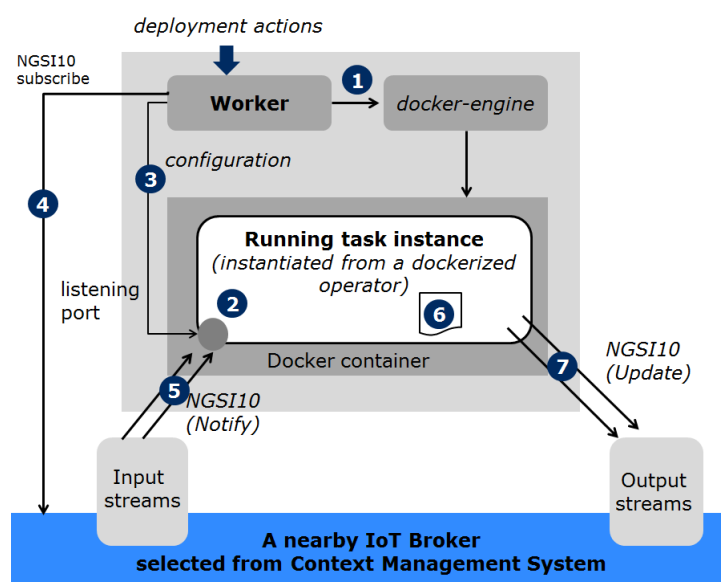


Figura A.7: FogFlow IOT Worker (fonte:[1])

Bibliografia

- [1] Axiomatics. [Xacml architecture and flow](#). Online, July 2016. Acedido em 20/08.
- [2] Amaro Correia and Luis Borges Gouveia. [Fiware: uma plataforma de desenvolvimento de soluções inteligentes](#). *TRS Tecnologia, Redes e Sociedade*, jun-2018.
- [3] FIWARE. [authzforce/rest-api-model](#). Online, July 2021. Acedido em 20/08.
- [4] e.V FIWARE Foundation. [Fiware: The open source platform for our smart digital future](#). Online, July 2021. Acedido em 20/08.
- [5] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. [Internet of things \(iot\): A vision, architectural elements, and future directions](#). *Future Generation Computer Systems*, 29(7):1645–1660, 2013. ISSN: 0167-739X. doi:<https://doi.org/10.1016/j.future.2013.01.010>.
- [6] MIT license. [Morgan](#). Online, September 2022. Acedido em 20/09.
- [7] Hsiao-Ying Lin and Wen-Guey Tzeng. [A secure decentralized erasure code for distributed networked storage](#). *IEEE Transactions on Parallel and Distributed Systems*, 21(11):1586–1594, 2010. doi:<https://doi.org/10.1109/TPDS.2010.27>.
- [8] NEC Revision. [Fiware fogflow 4. api walkthrough](#). Online, July 2021. Acedido em 20/08.
- [9] NEC Revision. [Fiware fogflow 1. core concepts](#). Online, July 2021. Acedido em 20/08.
- [10] NEC Revision. [Fiware fogflow introduction](#). Online, July 2021. Acedido em 20/08.
- [11] NEC Revision. [Fiware fogflow 3. integrations](#). Online, July 2021. Acedido em 20/08.
- [12] NEC Revision. [Fiware fogflow 11. security](#). Online, July 2021. Acedido em 20/08.
- [13] NEC Revision. [Fiware-idm](#). Online, July 2021. Acedido em 20/08.
- [14] NEC Revision. [Fiware-pep-proxy](#). Online, July 2021. Acedido em 20/08.
- [15] NEC Revision. [4. security - fogflow v3.2.8 documentation](#). Online, August 2022. Acedido em 11/08.

-
- [16] J. Singh, C. Millard, C. Reed, J. Cobbe, and J. Crowcroft. [Accountability in the iot: Systems, law, and ways forward](#). *Computer*, 51(07):54–65, jul 2018. ISSN: 1558-0814. doi:10.1109/MC.2018.3011052.
- [17] Lachlan Urquhart, Tom Lodge, and Andy Crabtree. [Demonstrably doing accountability in the Internet of Things](#). *International Journal of Law and Information Technology*, 27(1): 1–27, 12 2018. ISSN: 0967-0769. doi:10.1093/ijlit/eay015.
- [18] Rolf H. Weber. [Accountability in the internet of things](#). *Computer Law and Security Review*, 27(2):133–138, 2011. ISSN: 0267-3649. doi:https://doi.org/10.1016/j.clsr.2011.01.005.
- [19] Pamoda Wimalasiri. [A beginner’s guide to xacml](#). Online, July 2021. Acedido em 20/08.