Universidade Fernando Pessoa

Gestão de Dispositivos em Cidades Inteligentes. Utilização de Tecnologias Semânticas

Nelson Almeida

Faculdade de Ciências e Tecnologia

Universidade Fernando Pessoa

Dissertação apresentada à Universidade Fernando Pessoa como parte dos requisitos para obtenção do grau de Mestre em Engenharia Informática, ramo Computação Móvel

Orientador: Prof Doutor Feliz Ribeiro Gouveia
Outubro de 2018

Resumo

A IoT é importante para o desenvolvimento das cidades tornando-as "inteligentes", cobrindo interesses como mobilidade, ambiente e gestão de recursos.

Existe a necessidade de gerir semanticamente a informação gerada por milhares de dispositivos para que seja possível a administração de uma cidade inteligentes.

Este trabalho teve por objetivo demonstrar a utilização de tecnologias semânticas para representar e processar informação oriunda dos dispositivos para que sejam gerados alertas de apoio à gestão.

Foi implementado um ambiente simulado de cidade inteligente, utilizando uma plataforma de IoT existente, definida uma ontologia específica e definidas regras de processamento semântico para gerar alertas.

Demonstrou-se a viabilidade da utilização das tecnologias semânticas em ambientes de IoT, recorrendo a conjuntos de dados reais e sintéticos, e a sua importância para a gestão das cidades inteligentes.

Abstract

IoT is important for the development of cities making them "smart", covering interests such as mobility, environment and resource management.

There is a need to semantically manage the information generated by thousands of devices so that it is possible to manage a smart city.

The objective of this work was to demonstrate the use of semantic technologies to represent and process information from the devices to generate management support alerts.

We implemented a simulated smart city environment using an existing IoT platform, defined a specific ontology, and defined semantic processing rules to generate alerts.

The feasibility of using semantic technologies in IoT environments, using real and synthetic datasets, and their importance for the management of intelligent cities has been demonstrated.

Agradecimentos

Ao meu orientador, Professor Doutor Feliz Ribeiro Gouveia, cujas críticas contribuíram para o aperfeiçoamento deste documento. Aos demais professores do curso pela disponibilidade em auxiliar.

Aos meus familiares e namorada pelo suporte incondicionais em todo o meu percurso académico.

Aos meus amigos, presentes ou não, pelas palavras e gestos de apoio e carinho.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

IoT *Internet of Things*

CPS *Cyber-Physical Systems*

M2M *Machine to Machine*

SSN *Semantic Sensor Network*

MQTT *Message Queuing Telemetry Transport*

HTTP *Hypertext Transfer Protocol*

CoAP *Constrained Application Protocol*

TIC *Tecnologias da Informação e da Comunicação*

J2EE *Java Platform, Enterprise Edition*

RDF *Resource Description Framework*

RDFS *RDF Schema*

OWL *Web Ontology Language*

API *Application Programming Interface*

SAREF *Smart Appliances REFerenc*

SEAS *Smart Energy Aware Systems*

SAN *Semantic Actuator Network*

SWRL *Semantic Web Rule Language*

CSV *Comma Separated Values*

JSON *JavaScript Object Notation*

URI *Uniform Resource Identifier*

Capítulo 1

Introdução

Nas últimas décadas a utilização de computação móvel tem vindo a crescer exponencialmente e nos últimos anos tem havido uma grande aposta na utilização de dispositivos nos mais diversos ambientes para identificar e controlar determinados fatores.
A utilização de tais dispositivos veio criar um novo conceito designado Internet of Things (IoT) que se caracteriza por um conjunto de dispositivos e conexões em rede capaz de adquirir e transmitir dados.
O custo destes dispositivos com o passar dos anos tem vindo a descer consideravelmente e nos dias de hoje é relativamente barato adquirir um conjunto de dispositivos para tornar um ambiente inteligente, isto é, com capacidade sensorial e de tomada de decisão.
Devido ao atual baixo custo dos dispositivos a Internet of Things tem crescido em grandes domínios como as casas inteligentes, as indústrias e as cidades onde ainda há muito por explorar através desta tecnologia.
Com o aumento da diversidade de dispositivos possíveis de implementar nas cidades estas têm a possibilidade de estar cada vez mais inteligentes. Esta inteligência possibilita controlo de luminosidade, de tráfego, recolha de dados de meteorologia, poluição, lixo, parques de estacionamento, acesso à internet, informações de pontos de interesse, de transportes públicos e muitas mais aplicações.
Com o crescimento da utilização de dispositivos para as mais diversas situações a quantidade de dados gerados num intervalo de tempo é cada vez maior e a necessidade de tratamento destes dados é um campo ainda pouco explorado.
Tendo esta ideia como ponto de partida para a resolução de um problema que começa a surgir nas cidades que estão a adotar a IoT, a criação de uma plataforma de administração de dispositivos possíveis de implementar nas cidades inteligentes é algo que já é necessário e cada vez irá ser mais.
Idealmente a plataforma não se pode limitar a disponibilizar os dados enviados pelos dispositivos porque para os administradores esta será uma tarefa inviável de cumprir devido ao seu elevado número. Por outro lado, a leitura isolada dos sensores pode não dar informação que seja relevante para identificar um dado problema.

O objetivo de ter acesso a informação de mais alto nível é importante e implica atribuir a todos os dispositivos, além da telemetria que eles enviam, informação semântica e que através desta informação seja possível o cruzamento de informações e tomada de decisões. Deste modo o administrador será capaz de perceber as necessidades, comportamentos e de atuar sobre um determinado grupo de dispositivos.
Nos últimos anos têm sido desenvolvidos padrões de caracterização de dispositivos, padrões estes que ainda não estão implementados de modo a resolver problemas de administração. Dentro destes padrões destaca-se por exemplo o padrão de gestão de energia, o padrão de modo de funcionamento entre outros.
O desenvolvimento de ontologias para a IoT é outro campo que tem estado em constante desenvolvimento e a utilização de tais ontologias para caracterização semântica é algo necessário para que exista uma normalização.
A junção do valor semântico dos dispositivos à informação transmitida pelos próprios dispositivos, e ainda aliando esta a cruzamento de informação e tomada de decisão numa única plataforma irá facilitar bastante a administração dos dispositivos presentes pelas cidades e com isso fazer com que a adoção deles para tornar as cidades inteligentes seja ainda maior.

1.1 Razão da escolha

Tendo um grande interesse na automatização de tarefas simples e que facilitem o dia-a-dia da população a utilização de dispositivos é um campo de grande interesse nos mais diversos ambientes. A utilização destes dispositivos, por exemplo, em casas inteligentes está num elevado crescimento devido ao facto de serem locais relativamente pequenos e que necessitam de um conjunto "pequeno" de dispositivos. Em relação às cidades inteligentes o processo de adoção destes dispositivos e comodidades tem tido uma adesão positiva, mas num processo mais lento muito devido à necessidade de um conjunto muito maior de dispositivos para cobrir todo o território de uma cidade.
A utilização de mais dispositivos implica a uma maior necessidade de tratamento dos dados e essa é a razão do desenvolvimento deste trabalho. Com um leque alargado de dispositivos a informação disponibilizada vai ser maior e a administração destes dispositivos será um problema que necessitará de resolução.

1.2 Problema

A existência de um número elevado de dispositivos numa cidade torna a administração desses individualmente uma tarefa inviável, existindo assim a necessidade de criar regras para automatizar processos e tomadas de decisões para que a informação visualizada pelo

administrador seja de alto nível em vez de ser a simples leitura dos dispositivos.
Este será o principal problema a tentar solucionar nesta dissertação, reaproveitando as funcionalidades básicas já presentes na maioria das plataformas existentes, juntando uma camada de valor semântico e desenvolvendo mecanismos de cruzamento de dados e tomada de decisões.
A existência de inúmeros dispositivos para as mais diversas funcionalidades e estes serem desenvolvidos por diferentes empresas torna igualmente a administração destes um problema devido a cada fabricante produzir os dispositivos conforme considera que seja a melhor implementação dos mesmos.
Deste modo um dispositivo para uma determinada funcionalidade pode ter uma manutenção diferente de um outro dispositivo para a mesma funcionalidade, mas desenvolvido por um fabricante diferente. Tomando como exemplo um dispositivo que inclua uma bateria, um determinado fabricante pode colocar uma bateria de uma capacidade maior que outro fabricante, o dispositivo desenvolvido pelo primeiro fabricante pode incluir um painel solar e tornar assim o dispositivo sem necessidade de se carregar manualmente enquanto o outro fabricante pode desenvolver o dispositivo de modo a este necessitar de corrente elétrica constante.
Este tipo de informações e outras do mesmo género normalmente não são comunicadas através dos protocolos de comunicação presentes nos dispositivos, mas são informações necessárias para quem faz a administração da rede de dispositivos de uma cidade inteligente.
O modo como cada fabricante designa determinada informação também não é uniforme o que torna o cruzamento de dados entre dispositivos de diferentes fabricantes uma tarefa complexa pois para o mesmo tipo de informação existem diferentes designações atribuídas.
Tomando como exemplo um dispositivo de recolha de informação de meteorologia, um dado dispositivo pode ter o valor de temperatura denominada de “temperatura”, enquanto que outro, de um fabricante diferente, pode ter denominado de “temp”.

1.3 Objetivos

O objetivo principal deste trabalho é desenvolver um protótipo de um sistema que analise constantemente todos os dispositivos da cidade, normalize as informações disponibilizadas por estes, cruze essas informações e despolete avisos de alto nível para o utilizador, facilitando assim a administração de um número elevado de dispositivos possíveis de inserir numa cidade.
Uma ferramenta deste género irá ser extremamente necessária nos próximos anos quando as cidades tiverem ainda mais dispositivos do que têm na atualidade.

Tecnicamente para cobrir as funcionalidades básicas já presentes na maioria das plataformas, tais como visualização de dispositivos, visualização de atributos e telemetria individualmente, ambiente gráfico para melhor interação e visualização de painéis informativos, entre outras funcionalidades será utilizada uma já existente.
O adicionar de informação de caracterização dos dispositivos será algo a cobrir devido à necessidade de mais informação do que a disponibilizada através dos protocolos de comunicação dos dispositivos.
Para uma representação normalizada dos dados recebidos dos diferentes dispositivos, estes devem ser convertidos para que respeitem um modelo a desenvolver baseado nos padrões e ontologias já desenvolvidas.
Para tornar a tarefa de cruzamento de informação viável é necessário que os dados sejam armazenados com o maior detalhe possível seguindo sempre uma normalização e para isso deverá ser utilizada uma base de dados de triplos.

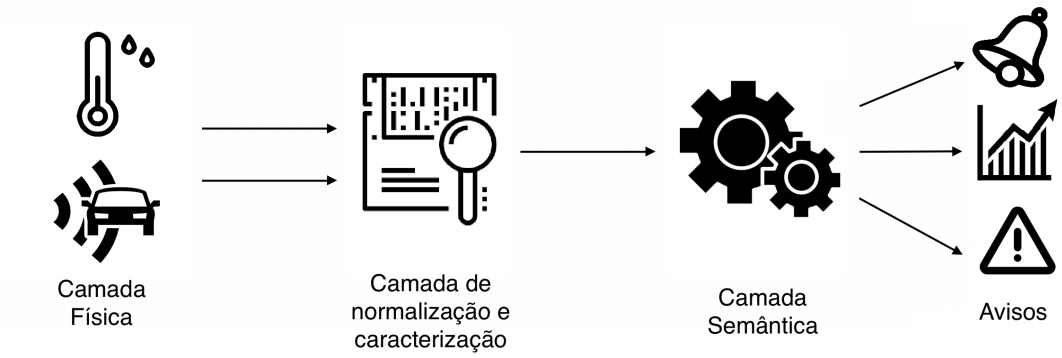

Figura 1.1: *Objetivos do projeto*

Resumidamente foram definidos os seguintes objetivos:

- Utilizar tecnologias da web semântica para representar e analisar as informações recebidas dos dispositivos.
- Definir e testar regras para despoletar alertas de apoio à gestão de cidades inteligentes.

Para atingir esses objetivos, definiram-se os seguintes objetivos técnicos:

- Escolher uma plataforma disponível para assegurar as funcionalidades básicas de gestão dos dispositivos.
- Definir a normalização dos dados recebidos dos dispositivos.
- Enriquecer a caracterização dos dispositivos usando ontologias.

Estes objetivos foram definidos para um cenário de cidades inteligentes, simulado, embora utilizando alguns conjuntos de dados reais.

1.4 Metodologia

O estudo e implementação da solução descrita neste trabalho tem como princípio uma análise bibliográfica sobre IoT e cidades inteligentes.
Através da análise de soluções já desenvolvidas são definidos requisitos para desenvolver uma plataforma que facilite a administração de uma cidade inteligente.
Posteriormente à análise e definição dos requisitos, é apresentada uma proposta de arquitetura com a demonstração dos diferentes módulos a implementar.
De seguida é demonstrada a implementação efetuada e testes para verificar o funcionamento previsto e a utilidade do sistema.
Finalmente são retiradas as devidas conclusões.

1.5 Estrutura do trabalho

Este trabalho é introduzido pelo capítulo 1 que expõe um breve enquadramento à temática, a razão pela qual foi escolhido este tema, qual o problema que visa resolver e quais os objetivos a alcançar.
No capítulo 2 temos diferentes referências do estado da arte, começando por conceitos mais introdutórios de enquadramento como Cyber-Physical Systems e Internet of Things. Temos também referência a diferentes associações, iniciativas e projetos de IoT. Padrões de caracterização de dispositivos da IoT e conceitos genéricos desta temática também são abordados. O capitulo descreve ainda projetos de cidades inteligentes, plataformas de gestão destas mesmas, ontologias de caracterização e para finalizar o capítulo uma abordagem ao raciocínio sobre informações obtidas de dispositivos.
No capítulo 3 definem-se as especificações do projeto, contando com a arquitetura do projeto final, assim como a arquitetura dos módulos de tratamento de dados e de leitura e tomada de decisões. Ainda neste capítulo definimos os requisitos funcionais, não-funcionais e de sistema a que este trabalho se propõe e casos de uso utilizados para testar a sua mais valia e funcionalidade.
No capítulo 4 é descrito o protótipo de implementação começando por uma explicação do tipo de dados utilizado e o tratamento dado aos mesmos para que estes sejam possíveis de comparar entre eles para posteriormente tomar decisões. A base de dados utilizada é também descrita assim como a ontologia desenvolvida neste trabalho para a caracterização da informação dos diferentes dispositivos. Por fim, ainda neste capítulo encontra-se o protótipo de cruzamento de informações e tomada de decisões, assim como as funcionalidades

utilizadas pela plataforma escolhida para representação da informação.
O capítulo 5 é dedicado à demonstração da validação do protótipo, segundo o modo como os casos de uso propostos foram cumpridos bem como a avaliação dos resultados obtidos. Encerra-se o trabalho com o capítulo 6 onde se encontra uma discussão dos resultados obtidos, as limitações ainda existentes no sistema, bem como se apontam modificações possíveis de serem feitas e a proposta de trabalhos a serem desenvolvidos no futuro.

Capítulo 2

Estado da Arte

2.1 Introdução

Neste capítulo apresenta-se uma visão geral do que existe relacionado com IoT e cidades inteligentes. Inicia-se com uma descrição do que são sistemas Ciber-Físicos e Internet of Things, seguido de associações, projetos, padrões e conceitos relacionados com esta último. Projetos relacionados com cidades inteligentes também são abordados, assim como plataformas existentes para administração destas mesmas. Por fim é feita uma abordagem a ontologias disponíveis para IoT e cidades inteligentes, ferramenta de manuseamento com ontologias e mecanismos de raciocínio sobre estas.

As soluções atuais de IoT são adaptadas, principalmente, para aplicações verticais, utilizando o conhecimento apenas de um domínio. Para tirar todo o partido destas precisam de ser substituídas para se adaptar a aplicações horizontais, onde se inclua aquisição de conhecimento e compartilhamento de recursos.

Grandes sistemas integrados de IoT são difíceis de construir devido à heterogeneidade dos protocolos, formatos e esquemas de dados, interfaces de serviços, assim como devido a requisitos de escalabilidade, restrições de recursos e mobilidade de dispositivos (Maarala and Riekki, 2017).

Para os administradores das cidades inteligentes, a atualização das infraestruturas de dados e comunicações e implantação de sensores criam as condições para uma tomada de decisão informada, ou seja, baseada em dados. Dados ricos permitem que os serviços da cidade funcionem de maneira mais eficiente e preditiva, o que resulta em um benefício direto para os cidadãos (Synchronicity, 2018).

Por estes motivos o interesse em desenvolver plataformas e soluções para cidades inteligentes está em crescimento e diversas iniciativas têm surgido.

2.2 Sistemas Ciber-Físicos

Cyber-Physical Systems (CPS) são sistemas que são monitorizados, coordenados, controlados e integrados por núcleos de comunicação e computação. Tal como a internet transformou a maneira como as pessoas interagem entre si, os sistemas ciber-físicos estão a transformar a maneira como se interage com o mundo físico à nossa volta como por exemplo transportes, energia, água, sustentabilidade urbana, meteorologia, qualidade do ar entre muitos outros (Lin, 2011).

Entramos numa era em que tudo é um computador desde telemóveis a computadores industriais. Os sistemas ciber-físicos ficam no meio de três tendências que se designam por colaboração em massa, transformação digital e sustentabilidade (Curley, 2012).

Dentro desta área existe um grande interesse de desenvolvimento e a criação de modelos multi-agente para CPS é um dos pontos fortes. Destes modelos destaco "Modeling Cyber-Physical Systems with Semantic Agents" (Lin et al., 2010) como exemplo de um modelo usado para representar a estrutura estática e o comportamento dinâmico de uma rede inteligente de distribuição de água como um estudo de caso da CPS.

CPS são sistemas complexos que acoplam fortemente objetos físicos do mundo usando interfaces de computação, comunicação e controlo enquanto que IoT é direcionado a pequenos dispositivos (Desai, 2013).

Por este motivo este trabalho não inclui CPS, embora seja de prever que a necessidade de os incluir numa plataforma semelhante venha a existir no futuro.

2.3 Internet of Things

O termo *Internet of Things* (IoT) é cada vez mais conhecido pela população pois a utilização de dispositivos deste género está num crescimento acentuado. Existem algumas definições para o termo mas será adotada a que considera a IoT como "a possibilidade de dotar todos os objetos do cotidiano com a habilidade de se identificarem, comunicar com outros objetos e possivelmente computar".

Automação é um dos principais objetivos da utilização da IoT, tornando tarefas simples do cotidiano mais rápidas e sem interação do homem.

Com a utilização cada vez maior de dispositivos a quantidade de dados gerados tem aumentado exponencialmente gerando um novo conceito denominado de Big Data que consiste num grande número de dados possíveis de serem analisados para retirar informação com valor.

A utilização da designação dispositivos é propositada pois num ambiente de IoT não dispomos somente de sensores, mas também de atuadores e deste modo ficam ambos considerados.

Um sensor é um dispositivo que se limita a recolher e enviar informação, já os atuadores

são dispositivos que têm a capacidade de receber informação e executar as instruções recebidas.

2.3.1 Associações

Existem diversas associações que se preocupam com a temática IoT e nomeadamente a sua integração nas Smart Cities.
De seguida são referenciadas algumas destas, sobre as quais incidiu alguma da pesquisa sobre projetos relacionados com este estudo.

European Research Cluster on the Internet of Things

O IERC (IERC, 2016) tem como principal objetivo abordar o grande potencial das capacidades baseadas em IoT na Europa e coordenar a convergência das atividades em curso. Reúne projetos financiados pela União Europeia com o objetivo de definir uma visão comum e os desafios de pesquisa de tecnologia e desenvolvimento da IoT a nível europeu na visão de desenvolvimento global.

The Alliance for Internet of Things Innovation (AIOTI)

A AIOTI (AIOTI, 2018) foi iniciada pela Comissão Europeia em 2015 com o objetivo de fortalecer o diálogo e a interação entre os desenvolvedores de IoT na Europa e contribuir para a criação de um ecossistema dinâmico de IoT, assim como fomentar a experimentação, replicação e implantação da IoT e apoiar a convergência e a interoperabilidade dos padrões de IoT.

European Telecommunications Standards Institute (ETSI)

O ETSI (ETSI, 2018) sendo um membro da oneM2M (oneM2M, 2018) (entidade que desenvolve padrões para Machine to Machine Communications e Internet of Things) visa fornecer uma interface *Machine to Machine* (M2M) padronizada, permitindo deste modo que vários dispositivos sejam conectados na IoT independentemente da rede subjacente. Dentro da ETSI são abordadas várias aplicações da tecnologia M2M como aparelhos inteligentes, medição inteligente, cidades inteligentes, tabelas inteligentes, sistemas de transporte inteligentes, automação industrial sem fios entre outras.

2.3.2 Projetos de IoT

Numa pesquisa por projetos de IoT encontra-se diversos já desenvolvidos nas diferentes vertentes de aplicação deste conceito. No entanto, foram selecionados para apresentar a seguir alguns que se relacionassem com cidades inteligentes, plataformas de gestão

de dispositivos da IoT e que abordassem técnicas possíveis de serem utilizadas para o desenvolvimento deste projeto.

INTER-IoT

O objetivo do INTER-IoT (InterIoT, 2016) é conceber, implementar e testar uma plataforma que permita a interoperabilidade de diferentes plataformas de IoT, concentrando-se em seis camadas da arquitetura de IoT com módulos cobrindo gestão de dispositivos, qualidade de serviço, integração de serviços, serviços de sistemas externos, armazenamento e virtualização. O projeto considera toda a camada de comunicação de rede e o conjunto completo de gestão de segurança (Gluhak and Vermesan, 2016).
A ideia deste projeto é ter uma abordagem de várias camadas que seja compatível com diferentes dispositivos, redes, plataformas, serviços e aplicações da IoT e alem disso facilitar a reutilização e integração dos sistemas de IoT existentes e futuros, criando deste modo um ecossistema global de plataformas de IoT interoperável.
Como ainda existe uma ausência de normas de IoT, o objetivo deste projeto é permitir a qualquer empresa projetar e desenvolver novos dispositivos ou serviços de IoT fazendo com que o ecossistema existente cresça rapidamente.

FIESTA-IOT

FIESTA-IoT (FIESTA-IoT, 2018) fornece as ferramentas, técnicas, processos e práticas para permitir a interconexão de plataformas de IoT usando tecnologias semânticas.
Visa reunir num só local um conjunto de setores como saúde, cidades inteligentes, energia, transportes, entre outros, para uma mais fácil interação entre eles.
É composto por 14 parceiros e integra projetos em curso na União Europeia que já utilizam tecnologias Web semânticas como o OpenIoT, Citypulse, VITAL, Spitfire, IoT-est, IoT-A, IOT6, iCore e Sensei.
O projeto FIESTA-IoT (Gyrard and Serrano, 2018) aborda problemas de interoperabilidade semântica em sete níveis que são:

- Hardware Level em que fornece middleware baseado em semântica para lidar com dispositivos de hardware heterogêneos;
- Data Level que unifica os dados produzidos por dispositivos anotando dados semanticamente;
- Model Level que alinha as ontologias de IoT existentes para garantir uma melhor interoperabilidade;
- Query Level que consulta as bases de conhecimento unificadas;

- Reasoning Level que unifica o caminho para deduzir informações significativas dos sensores para evitar redundância;
- Service/Application Level que traz a ideia de Experimentation-as-a-Service (Eaas) da Cloud Computing Stack;
- Applicative Domain Level que constrói aplicações verticais para interconectar e reutilizar aplicações atuais ou especificas dos domínios atuais.

M2MLabs Mainspring

M2MLabs Mainspring (M2MLabs, 2015) é uma estrutura de aplicação para construção de aplicações máquina - máquina. Em tais aplicações, normalmente um dispositivo remoto equipado com atuadores e sensores por exemplo de GPS, temperatura ou pressão comunica com uma aplicação de servidor que está a executar o protocolo de comunicação e configuração do dispositivo, o armazenamento de dados enviados pelos dispositivos, bem como a lógica comercial da aplicação e a camada de apresentação.
O Mainspring trata da comunicação do dispositivo, da configuração, bem como do armazenamento e recuperação de dados para que os desenvolvedores da aplicação possam concentrar-se apenas na lógica do negócio. As aplicações M2M podem ser prototipadas mais rapidamente e, finalmente, transferidas para um ambiente de execução, construído em cima de um servidor *Java Platform, Enterprise Edition* (J2EE) e base de dados Apache Cassandra.
As suas principais características são modelagem flexível de dispositivos, suas partes e suas características, configuração do dispositivo, comunicação de dados do dispositivo para a aplicação, envio de comandos do aplicativo para o dispositivo, validação e normalização dos dados, armazenamento de longo prazo de dados enviados por dispositivos, funções de recuperação de dados para aplicativos externos usando REST API e IDE baseado na web para gestão de dispositivos e execução de simulações.

The Schema Editor of OpenIoT for Semantic Sensor Networks

The Schema Editor of OpenIoT for Semantic Sensor Networks (Jayaraman et al., 2015) consiste no desenvolvimento de um editor que fornece uma interface web para definir novos tipos de dispositivos, usando a ontologia *Semantic Sensor Network* (SSN) como modelo central.
Este editor está integrado com a plataforma OpenIoT para recolher informação e gerar descrições de sensores virtuais e automatizar a sua anotação semântica e registo.
OpenIoT é uma plataforma de middleware genérica para aplicações de Internet of Things, que permite unir dispositivos ligados à Internet e serviços da Web semânticos por meio de uma interface amigável.

A interface automatiza a geração de descrições *Resource Description Framework* (RDF) para a informação do dispositivo inserido pelo utilizador.
Este é um exemplo de uma implementação utilizada para inserção de informação semântica dos dispositivos, informação esta que não é transmitida através dos protocolos de comunicação. Com uma ferramenta deste género o administrador é orientado a inserir os dados de uma forma normalizada para caracterizar o dispositivo seja ele de um qualquer tipo ou fabricante.

2.4 Padrões de dispositivos

Na maioria dos casos de uso com sensores e atuadores é exigido que eles suportem características, como ser barato, leve, pequeno, móvel, eficiente em energia ou com energia autônoma. Isso cria restrições quanto a fontes de energia disponíveis e leva a diferentes tipos e modos de operação.
Com base na terminologia existente e em exemplos adicionais, pretendeu-se descrever essas restrições de energia e os modos de operação sob a forma de padrões.
Tendo em conta que as informações mais recorrentes se tornam evidentes, a ideia consiste em agrupar e extrair os princípios básicos para formar padrões mais abstratos.
Neste sentido já foram desenvolvidos alguns padrões (Reinfurt et al., 2016) que serão descritos a seguir:

Padrão de tipo de comunicação

Baseia-se na ideia dos dispositivos não comunicarem todos na mesma linguagem e existir a necessidade de os interligar a uma rede comum para que haja a possibilidade de analisar e cruzar os dados provenientes de diferentes tipos de dispositivos.

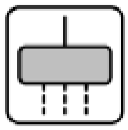

Figura 2.1: *Padrão do tipo de comunicação*

A solução passa por conectar os dispositivos a um device gateway intermediário que traduza a tecnologia de comunicação suportada pelo dispositivo para a tecnologia de comunicação da rede e vice-versa, como por exemplo a passagem de Bluetooth para MQTT.

Padrão de visibilidade

Alguns dispositivos estão apenas intermitentemente *online*, outros estão constantemente *online* e outros só alteram o modo para *online* quando é detetado algum acontecimento.

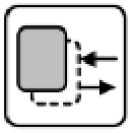

Figura 2.2: *Padrão de visibilidade*

Esta gestão é feita normalmente para que haja uma economia de energia.

Padrão de bloqueio remoto

Consiste na possibilidade de bloquear remotamente o dispositivo e apagar a sua informação.

Figura 2.3: *Padrão bloqueio remoto*

Deste modo caso haja, por exemplo, um roubo é possível eliminar os dados para que pessoas não autorizadas não obtenham nenhuns dados.

Padrão de tipo de fornecimento de energia

Alguns dispositivos necessitam de estar constantemente ligados à corrente, outros utilizam bateria que necessita de ser carregada manualmente ou possuem componentes que tornam o carregamento automatizado, como é o caso de dispositivos com painéis solares. Outros possuem bateria mas têm um baixo consumo energético e esta durará para toda a sua vida útil.

Figura 2.4: *Padrão do tipo de fornecimento de energia*

A escolha da solução mais apropriada para cada dispositivo deve ser feita, por exemplo, tendo em conta o gasto energético do mesmo.

Padrão do tipo de funcionamento

O padrão "Sempre Ligados" requer uma fonte de energia constante ou recolha de energia autónoma, caso contrário a sua manutenção torna-se dispendiosa.
No caso do padrão "Normalmente a Dormir" pode ter uma fonte de energia como uma bateria.

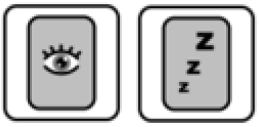

Figura 2.5: *Padrão de tipo de funcionamento*

Dependendo do dispositivo e da quantidade de vezes que este acorda deve-se decidir qual a melhor estratégia.

Através destes padrões é possível ter uma noção do comportamento dos dispositivos e juntamento com uma análise de ontologias para IoT, que irão ser explicadas posteriormente, desenvolver uma caracterização genérica dos dispositivos.

2.5 Conceitos

Esta secção apresenta três conceitos fundamentais em IoT, a interoperabilidade, a segurança e a normalização.

2.5.1 Interoperabilidade em IoT

Para existir comunicação contínua entre aplicações e dispositivos de IoT é necessário abordar e resolver com eficiência os problemas de interoperabilidade em diferentes níveis. Interoperabilidade é definida como a capacidade de dois ou mais sistemas ou componentes trocarem dados e usarem informações, o que proporciona muitos desafios sobre como obter informações, trocar dados e entender e processar as informações (Jennings, 2015).
Existem diversas iniciativas de criar interoperabilidade na IoT e um exemplo disso é o projeto Wise-IoT que é financiado ao abrigo do programa H2020 e que consiste num projeto colaborativo entre a Europa e a Coreia que tem por objetivo melhorar a interoperabilidade dos sistemas de IoT existentes.
Os objetivos gerais do wise-IoT (Wise-IoT, 2015) são:

- Fornecer uma IoT interoperável em todo o mundo que utilize uma grande variedade de diferentes sistemas de IoT e combine-os com informações contextualizadas de várias fontes de dados;
- Provar que esse sistema pode oferecer serviços de IoT dinâmicos, em tempo real e remotos com segurança e confiabilidade, com adaptação automática aos recursos disponíveis e aos dados em qualquer lugar do mundo;
- Fortalecer a normalização internacional da IoT.

As questões de interoperabilidade foram assumidas no trabalho, e não foram contempladas no protótipo desenvolvido.

2.5.2 Segurança em IoT

IoT apresenta novos desafios nas áreas de redes e segurança, desafios estes que podem-se tornar barreiras para a implantação da IoT em larga escala. Quanto maior for a aplicação ou serviço baseado em IoT maior é a vulnerabilidade a ataques e a roubo de informações. Vulnerabilidade é a oportunidade de uma ameaça causar perda e uma ameaça é qualquer perigo potencial para um recurso, originário de qualquer coisa e / ou qualquer pessoa que tenha o potencial de causar uma ameaça (Jennings, 2015).
Atualmente não existem normas de segurança para IoT e deste modo cada fabricante implementa o nível de segurança que lhe interessa no momento do desenvolvimento do dispositivo. Acontece que os dispositivos podem ficar em operação durante anos e grande parte deles sem que sejam atualizados.
Existem diversos estudos sobre segurança em IoT e todos eles abordam as mesmas questões como por exemplo evitar vulnerabilidades conhecidas, não ter padrões inseguros e evitar o rastreio dos sistemas (Schneier, 2017).
De entre estes estudos pode-se destacar, por exemplo, projetos como oneM2M que fornece especificações técnicas (oneM2M, 2016a) e soluções de segurança (oneM2M, 2016b) para a IoT.
Um exemplo de ataque que envolveu dispositivos de IoT foi o ataque DDoS (negação de serviços distribuídos) ao portal KrebsOn Security.com (Krebs On Security, 2018) em 2016 e que teve como consequência a indisponibilidade deste portal por quase quatro dias. O ataque foi realizado por intermédio de uma rede de dispositivos de IoT controlados pelo atacante, como routers de Internet, câmaras de segurança e gravadores de vídeo digital.
As principais falhas que originaram este ataque foi o facto de os fabricantes de dispositivos, para torna-los mais económicos, cometerem falhas a nível de segurança e os utilizadores destes não os protegerem devidamente nem que seja simplesmente pela troca da senha de segurança padrão.

Sendo a segurança um tema muito importante, quando se fala de IoT e informática no geral, não é no entanto o foco deste trabalho. Assumirei deste modo para questões de protótipo que a segurança será tratada num modulo adjacente.

2.5.3 Normalização IoT

A IoT baseia-se na ideia de que qualquer coisa pode ser conectada a qualquer momento, de qualquer lugar a qualquer rede, preservando a segurança e a privacidade.
Devido ao facto de qualquer fabricante poder fornecer dispositivos IoT, em qualquer domínio, existe uma grande diversidade de marcas, modelos e protocolos, todos diferentes e sem qualquer esforço de normalização ou de interoperabilidade. Esta falta de normalização traz custos acrescidos para os clientes, ao implicar esforço de interligação e de normalização de formatos, dados e protocolos de comunicação.
As normas globais são necessárias para obter economia de escala e interoperabilidade.
Deve ser feito um esforço em identificar os requisitos e especificações da indústria e as necessidades dos padrões de IoT em diferentes domínios para evitar a duplicação de esforços.
A complexidade da IoT vem do facto de que esta pretende oferecer suporte a várias aplicações diferentes, abrangendo uma ampla gama de disciplinas que não fazem parte do domínio de *Tecnologias da Informação e da Comunicação* (TIC).
Seria, portanto, benéfico desenvolver uma abordagem mais ampla à normalização e incluir a antecipação da formulação de políticas emergentes ou em andamento nas áreas de aplicação alvo e, assim, estar preparado para seu impacto potencial na normalização relacionada à IoT.
São necessárias normas para interoperabilidade dentro e entre domínios.
Dentro de um domínio, as normas são realizadas de um modo simplista, cobrindo somente as necessidades básicas necessárias. Um domínio pode ser até mesmo uma organização ou empresa específica que implementa uma aplicação de IoT. Entre os domínios, a interoperabilidade assegura a cooperação entre os domínios interligados e é mais orientada para uma implantação adequada de IoT (Jennings, 2015).

2.6 Projetos de Cidades Inteligentes

Cidades por todo o mundo procuram desenvolver estratégias de digitalização amplas e ambiciosas (Porto Digital, 2018), (Smart Santander, 2018).
Como as cidades partilham problemas e ambições, soluções comuns devem ser encontradas para controlar o espaço urbano.

Figura 2.6: *Cidades Inteligentes*

O uso de informação e tecnologias de comunicação permite gerir melhor as cidades. Cidades inteligentes devem usar a informação que produzem para melhorar a vida das pessoas, otimizar processos, reduzir custos e abrir as portas a novos mercados.
Com este intuito diversas iniciativas têm surgido com diferentes propósitos mas todas elas com o objetivo de tornar as cidades inteligentes.
As iniciativas que serão descritas de seguida foram selecionadas através da pesquisa por projetos de cidades inteligentes. Fatores como escala da implementação, atualidade de desenvolvimento, programas envolvidos, como por exemplo Horizonte 2020 e quantidade de informação disponibilizada foram determinantes na escolha.

2.6.1 Synchronity City IoT

O projeto SynchroniCity visa desenvolver um mercado global de Internet of Things, no qual cidades e empresas desenvolvem serviços digitais com grande foco na melhoria da vida dos cidadãos e economias locais. O projecto Synchronity recebeu financiamento do programa de investigação e inovação Horizonte 2020 da União Europeia.
Através da Associação Porto Digital e em parceria com outras sete cidades europeias, a cidade do Porto pretende expandir os serviços de IoT com base na infraestrutura tecnológica existente com um financiamento de 3 milhões de euros.
Um dos focos de trabalho tecnológico esta relacionado com os dados onde o objetivo é criar condições para uma tomada de decisão informada sobre várias organizações ligadas à gestão da cidade e fornecer dados abertos para a comunidade (Hinchliffe, 2018).

Algumas das soluções de IoT para resolver os desafios da cidade são uma Plataforma Urbana que recolhe e analisa dados de mobilidade, meio ambiente, energia, resíduos e proteção civil, aumentando assim a inteligência e a eficiência da gestão e prestação de serviços da cidade. Uma plataforma de dados abertos que fornece conjuntos de dados de várias organizações públicas aumentará a transparência e permitirá que a comunidade agregue valor.
Além disso uma rede de sensores ambientais, composta por sensores que medem parâmetros meteorológicos, poluentes do ar e ruído, permitirá que a cidade atue de forma reativa e / ou preventiva, com o objetivo de minimizar os impactos negativos (Synchronicity, 2018).

Resumo:

- **Origem**: várias cidades (Porto incluído);
- **Tipo de projeto:** plataforma urbana de dados abertos a público;
- **Tipo de sensores:** mobilidade, energia, resíduos, meteorológicos, poluição do ar e ruído entre outros;
- **Quantidade de sensores:** depende das organizações envolvidas;
- **Tipo de tratamento:** tomada de decisão informada, fornecer dados abertos para a comunidade atuar de forma reativa e/ou preventiva.

2.6.2 Sharing Cities

As Smart Cities estão em constante evolução e prova disso são os recorrentes encontros de debates sobre esta temática, Portugal Smart Cities Summit 2018 aconteceu no Centro de Congressos de Lisboa, de 11 a 13 de abril de 2018 e entre outros projetos como "A Sustentabilidade Inteligente Para o Futuro da Água" e a "Summit Da Energia", pode-se destacar Sharing Cities.
Sharing Cities consiste num projeto, do programa H2020, iniciado em Janeiro de 2016, com a duração de 5 anos. O projeto propõe uma nova abordagem para criar cidades inteligentes, promovendo a cooperação internacional entre a indústria e as cidades (SapoTek, 2018), (Labs, 2018).
Cidades como Lisboa, Londres e Milão estão envolvidas neste projeto, com o objetivo de desenvolver uma solução integrada que responda aos desafios energético-ambientais da atualidade como por exemplo menos poluição, mais eficiência energética, maior produção renovável de energia, menos emissões de CO2, melhor mobilidade, mais emprego e mais inclusão. A aproximação dos cidadãos às cidades, com recurso a novas tecnologias, o desenvolvimento de um plano de ação inovador que sirva de modelo para outras cidades

da UE e a angariação de investimento privado e de novas formas de negócio nas cidades, com base na informação gerada pela aplicação das soluções integradas são outros dos objetivos (Lisboa E-Nova, 2018), (Collinge, 2018).

Resumo:

- **Origem**: várias cidades (Lisboa, Londres, Milão);
- **Tipo de projeto:** implantação de tecnologia e desenvolvimento de plataforma de partilha urbana;
- **Tipo de sensores / serviços:** serviços de partilha de carros e bicicletas eletricas, sistemas de gestão energia, iluminação inteligente, parque inteligente, plataforma de partilha urbana através do envolvimento com os cidadãos entre outros;
- **Soluções propostas:** cooperação internacional entre a indústria e as cidades, responder aos desafios energético-ambientais da atualidade (ruido, eficiência energética, energia renovável, emissões de CO2, melhor mobilidade);

2.6.3 Project ESPRESSO

O projeto ESPRESSO baseia-se no desenvolvimento de um painel de informações de cidades inteligentes com base em normas relevantes, tecnologias e modelos de informação. O projeto analisa possíveis lacunas e sobreposições entre as normas e fornece diretrizes sobre como abordá-las efetivamente (ETSI, 2018).
Para proporcionar um futuro sustentável as cidades inteligentes devem integrar sistemas físicos, digitais e humanos. No entanto, devido à complexidade tecnológica, bem como à complexidade dos vários serviços envolvidos numa cidade inteligente, exige uma abordagem sistêmica à normalização, devendo esta abordagem promover a reutilização das normas já existentes para acelerar a implementação (ESPRESSO, 2018).
ESPRESSO concentra-se no desenvolvimento de uma Smart City Information Framework baseada em padrões já existentes, consistindo assim numa plataforma para cidades inteligentes e vários serviços de processamento e fornecimento de dados para integrar dados e processos relevantes.

Resumo:

- **Tipo de projeto:** desenvolvimento de uma plataforma;
- **Tipo de tratamento:** analise de possíveis lacunas e sobreposições entre as normas e fornecimento de diretrizes sobre como abordar;

- **Soluções propostas:** desenvolvimento de uma Smart City Information Framework baseada em padrões já existentes, tecnologias e modelos de informação. Exige uma abordagem sistêmica à padronização, devendo esta abordagem promover a reutilização dos padrões já existentes.

2.6.4 SmartSantander

Smart Santander é uma das maiores cidades inteligentes experimentais, situada na cidade de Santander no norte de Espanha. Esta cidade ficou transformada num laboratório experimental e é agora usada para testes experimentais para a pesquisa e experimentação de arquiteturas, tecnologias, serviços e aplicações para a Internet das Coisas no contexto de cidades inteligentes.
Dentro da plataforma SmartSantander, mais de 15.000 sensores foram instalados em torno de uma área de aproximadamente 22 km quadrados na cidade.
A grande maioria dos sensores está escondida dentro de caixas brancas colocadas em infra-estruturas de rua, tais como lâmpadas de iluminação , edifícios e postes de serviço, enquanto outros são enterrados no pavimento, por exemplo, sensores de estacionamento.
Nem todos os sensores são estáticos sendo alguns colocados na rede de transportes públicos da cidade, incluindo autocarros, táxis e carros da polícia.
Os sensores fornecem informações em tempo real sobre diferentes parâmetros ambientais (luz, temperatura, ruído, CO2), bem como outros parâmetros como a ocupação do estacionamento.
Comunicação entre sensores, repetidores e gateways são feitas através do IEEE Interface 802.15.4, enquanto os gateways usam Wi-Fi, GPRS / UMTS ou interfaces Ethernet para conectar com o SmartSantander (Cheng et al., 2015).

Resumo:

- **Origem:** Santander (Espanha);
- **Tipo de projeto:** criação de cidade inteligente modelo;
- **Tipo de sensores:** luminosidade, poluição, ruído, transportes, estacionamento, localização (polícia);
- **Quantidade de sensores:** mais de 15.000 sensores em 22km quadrados da cidade;
- **Soluções propostas:** desenvolvimento de cidade inteligente experimental;
- **O que usam:** comunicação entre sensores, repetidores e gateways são feitas através do IEEE Interface 802.15.4, enquanto os gateways usam Wi-Fi, GPRS / UMTS ou interfaces Ethernet para conectar com o SmartSantander.

2.6.5 Citibrain

É um consórcio, com sede em Portugal, mais especificamente em Aveiro, que nasceu em 2014. É especializado em criar soluções inteligentes para desafios globais e baseia-se na ideia de disponibilizar uma solução unificada para as áreas urbanas adicionando inteligência em múltiplos domínios da vida na cidade (Citibrain, 2018).
O Citibrain aposta em duas sub-áreas, a mobilidade e o ambiente, com duas soluções para cada uma delas, sendo estacionamento inteligente e monitorização de trânsito inteligente na área da mobilidade e na área do ambiente os resíduos e a qualidade do ar (Sapo24, 2018).

Resumo:

- **Origem:** Aveiro (Portugal);
- **Tipo de projeto:** implantação de dispositivos para tornar cidades inteligentes;
- **Tipo de sensores:** estacionamento, qualidade do ar, trânsito;
- **Quantidade de sensores:** mais de 15.000 sensores em 22km quadrados da cidade;
- **Soluções propostas:** disponibilizar uma solução unificada para as áreas urbanas adicionando inteligência em múltiplos domínios da vida na cidade. Aposta em duas sub-áreas, a mobilidade e o ambiente, com duas soluções para cada uma delas, sendo estacionamento inteligente e monitorização de trânsito inteligente na área da mobilidade e na área do ambiente os resíduos e a qualidade do ar.

2.6.6 Km4City

Km4City (Km4City, 2018) é uma iniciativa de desenvolvimento de uma plataforma urbana para implementar a visão sobre a cidade, monitorização e fornecimento de novos serviços para melhorar a qualidade de vida dos cidadãos.
Esta é uma iniciativa que tem como preocupação a necessidade de desenvolver uma plataforma que não se baseie apenas na recolha, armazenamento e visualização de dados pois considera isso ferramentas insatisfatórias para produzir valor a partir dos dados e permitir a construção de serviços inteligentes que tornem sustentáveis as cidades e prestadores de serviços (Group Km4City, 2018).
Em vez disso, o sistema gere um grande volume de dados proveniente de diversas fontes como trafego, estacionamento, poluição, meteorologia, localização de transportes públicos entre outras. Mapeia esses dados para uma ontologia de cidade inteligente, neste caso chamada Km4City, e armazena em um RDF-Store onde podem ser consultados via SPARQL para fornecer novos serviços aos utilizadores (Bellini et al., 2014).

Km4City conta com uma ferramenta de desenvolvimento e um App que fornece acesso a dados multi-domínio (mobilidade e transporte, cultura, eventos, estacionamento, turismo, saúde, segurança, entre outros), informação sobre problemas e alertas, possibilidade de comentar serviços e acontecimentos entre outras funcionalidades.

Resumo:

- **Origem:** Itália;
- **Tipo de projeto:** desenvolvimento de plataforma;
- **Tipo de sensores:** trafego, estacionamento, poluição, meteorologia, localização (transportes públicos), entre outros;
- **Tipo de tratamento:** gerar um grande volume de dados, mapeiar para uma ontologia e armazenar em um RDF-Store onde podem ser consultados via SPARQL;
- **Soluções propostas:** plataforma para ter uma visão sobre a cidade, monitorização e fornecimento de novos serviços para melhorar a qualidade de vida dos cidadãos;
- **O que usam:** mapeamento para uma ontologia, RDF datastore, consultas via SPARQL.

2.6.7 Análise dos projetos

Como vimos alguns projetos visam a inserção de dispositivos nas cidades e outros baseiam-se na gestão destes mesmos.
A maioria dos projetos de gestão de dispositivos simplesmente recolhem, armazenam e possibilitam a visualização da informação. Nenhum deles tem a capacidade de análise semântica dos dados.
Deste modo torna-se difícil a gestão devido ao elevado número de dispositivos presentes numa cidade.
Será na solução desta lacuna que o desenvolvimento deste trabalho se baseará.

2.7 Plataformas de IoT

As plataformas de IoT são a componente chave para ligar dispositivos, recolher e processar dados e possibilitar a interação através de interfaces de utilizador baseadas na web. Através delas é possível coordenar e gerir um número significativo de dispositivos conectados, garantindo a segurança e a privacidade dos dados trocados, através da integração de protocolos de segurança, e resolvendo problemas de interoperabilidade.

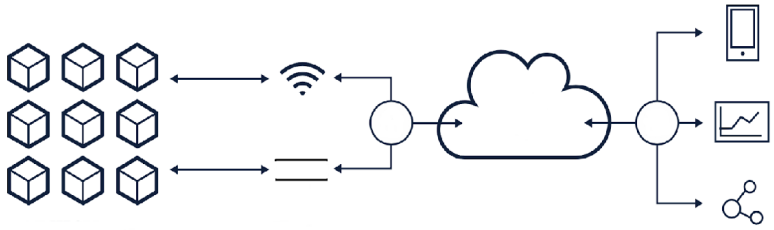

Figura 2.7: *Plataforma de IoT*

Os principais elementos que permitem a existência do ecossistema de plataformas de IoT são segurança e privacidade, processamento de dados e partilha de dados, atividade dos programadores e identificação dos utilizadores finais (Barchetti et al., 2017).
Existem diversas plataformas para recolha e visualização de informação de dispositivos e alguma delas direcionadas a cidades inteligentes. A procura para este trabalho incidiu em plataformas recentes ou com atualizações constantes, documentadas, visualmente apelativas, de código-fonte aberto e possíveis de utilizar para cobrir as funcionalidades básicas necessárias.

2.7.1 Kaa Project

O Kaa project (Kaa, 2018) é uma plataforma de middleware de IoT, de código-fonte aberto, que pode ser usada para criar soluções e produtos para diferentes domínios onde se utiliza a IoT.
Lida com a conexão de dispositivos e suporta uma variedade de plataformas comerciais de hardware de IoT. A abstração da conexão de rede, como Wi-Fi, Ethernet, ZigBee, MQTT, CoAP, XMPP, TCP é possível e com isso possibilita a criação de aplicações que comunicam com dispositivos, mesmo com ligações de dados intermitentes.
Kaa fornece também processamento de dados escalável, um sistema de entrega de mensagens e funcionalidades simples de gestão, permitindo assim que as mensagens possam ser endereçadas e filtradas para um subconjunto de endereços (*endpoints*) ou o seu comportamento de software possa ser modificado.
A *Application Programming Interface* (API) da plataforma é principalmente RESTful, fornecendo um conjunto diversificado de bibliotecas cliente para ligar dispositivos de IoT à Internet. As implementações incluem SDKs de cliente para Java, C, C ++ e Objective C. Para lidar com dispositivos não IP, como Bluetooth ou Zigbee, Kaa assume gateways, ou seja, pontes de ligação de diferentes protocolos de comunicação entre dispositivos e / ou plataforma da IoT.
Para uma fácil adaptação à plataforma esta conta com um ambiente de testes e demonstrações para visualizar as suas capacidades (Gluhak and Vermesan, 2016).

Resumo:

- Diferentes protocolos de comunicação
- Armazenamento de dados
- Visualização dos dados (visualização de telemetria, estatísticas, geolocalização entre outros em widgets)
- Gestão de dispositivos (possibilidade de dar atributos aos dispositivos e criar filtros com base neles)
- Processamento e análise de dados
- Arquitetura modular
- Capacidade de configurar dispositivos
- Comunicação bidirecional para os dispositivos
- Escalável
- Segurança – Comunicação protegida por TLS e DTLS
- Vários casos de uso desenvolvidos

2.7.2 Eclipse Kura

O Eclipse Kura (Eclipse, 2018) é um projeto do Eclipse IoT que fornece uma plataforma para construir gateways, ou seja, pontes de ligação de diferentes protocolos de comunicação entre dispositivos e / ou plataforma da IoT. É um contentor de aplicações inteligentes que permite a gestão remota desses gateways e fornece uma ampla variedade de APIs para permitir que se implementem aplicações de IoT.
O Kura é executado sobre a Java Virtual Machine (JVM) e aproveita o OSGi, um sistema dinâmico de componentes para Java, para simplificar o processo de criação de blocos de construção de software reutilizáveis. As APIs do Kura oferecem fácil acesso ao hardware subjacente, incluindo portas serie, GPS, watchdog, USB, GPIOs, I2C, entre outros. Além disso também oferece um pacote OSGI para simplificar a gestão de configurações de rede, a comunicação com servidores de IoT e a gestão remota do gateway.
Kura inclui serviços de I/O, serviços de dados, serviços na nuvem, serviço de configuração, gestão remota, redes, serviço de vigilância e administração.

Resumo:

- Serviço de I/O
- Serviço de dados (armazenamento e encaminhamento de dados de telemetria)
- Serviço de Cloud
- Serviço de configuração
- Gestão remota
- API para configurações

2.7.3 OpenHAB

A plataforma OpenHAB (Openhab, 2018), desenvolvida em Java, ao invés de suportar dispositivos específicos, reúne um largo número de protocolos que são utilizados pela maioria da IoT e deste modo consegue comunicar com praticamente todos os dispositivos possíveis de interagir.
O OpenHAB além de todos os protocolos que já suporta permite adicionar novos através de plugins cujas inclusões e remoções não comprometem a execução do programa.
As suas maiores implementações são no ambiente de casas inteligentes pois a aplicação conta com um ambiente de gestão de dispositivos voltada para este propósito.
Como suporta uma grande quantidade de protocolos, aliado à facilidade de configuração e parametrização dos dispositivos bem como a existência de uma grande comunidade ativa e de inúmeros trabalhos realizados usando esta ferramenta é uma possibilidade a explorar, desta vez num contexto diferente, as cidades inteligentes.

Resumo:

- Integrável (suporta diferentes tecnologias e sistemas)
- Automatizável (mecanismo de desenvolver regras, scripts, notificações entre outras tarefas)
- Aplicável em qualquer sistema operativo
- Largamente conhecido e utilizado por diversas empresas
- Modular (pode ser estendido através de “complementos”)
- Ligação a diferentes tecnologias (Bluetooth, zigBee, zwave, kodi, MQTT, ...)
- Possibilita integração de sistemas externos

- Possibilita diferentes ações para diferentes funcionalidades (mensagem, comunicação, transformação de dados)
- Possibilita a persistência de dados (diferentes bases de dados possíveis de utilizar)
- Possibilita a transformação de dados

2.7.4 Thingsboard

É uma plataforma de código-fonte aberto de IoT, disponibilizada sob a licença Apache 2.0, para recolha, processamento, visualização e gestão de informação. Tem um ambiente gráfico para facilitar a interação e permite a conexão de dispositivos através de protocolos para IoT como o MQTT, CoAP e HTTP (Thingsboard, 2018).
O ambiente gráfico é personalizável contando com mais de 30 widgets configuráveis e possibilidade de desenvolver mais utilizando o editor incorporado (Veres, 2017).
Thingsboard tem um gestor de dispositivos que fornece a capacidade de registo e gestão, permitindo analisar os atributos e telemetria deles.
Conta com a possibilidade de personalização e integração através de um sistema de regras e plug-ins para processar dados recebidos dos dispositivos, possibilitando o encaminhamento dos dados para sistemas externos ou o acionar de alarmes usando as regras.
Thingsboard como é uma plataforma de administração, conta com a possibilidade de ter vários administradores e também com um sistema de tolerância a falhas onde estas são detetadas automaticamente podendo os nós afetados serem substituídos sem tempo de inatividade. A nível de segurança suporta criptografia de transporte para os protocolos, autenticação e gestão de credenciais.
Esta plataforma tem disponível uma REST API que pode ser explorada utilizando a interface de utilizador do Swagger para interagir com o Thingsboard e controlar a maioria das suas funcionalidades.
Swagger é uma ferramenta de desenvolvimento de API que permite o desenvolvimento desta desde projeto e documentação até desenvolvimento e testes.
Thingsboard conta com aplicações nas áreas de Energia Inteligentes, Agricultura, Edifícios Inteligentes Cidades Inteligentes e Telecomunicações.

Resumo:

- Gestão de dispositivos
- Processamento de dados
- Ambiente gráfico
- Diversos widgets para visualização da informação e possibilidade de criar e integrar novo

- Conexão com dispositivos através de diferentes tecnologias (*Message Queuing Telemetry Transport* (MQTT), *Constrained Application Protocol* (CoAP), *Hypertext Transfer Protocol* (HTTP))
- Sistema de regras e Plug-ins
- Possibilidade de encaminhar dados para sistemas externos
- Possibilidade de caracterizar dispositivos através de atributos
- REST API disponível
- Possibilidade de subdividir os dispositivos (por exemplo em zonas dentro de uma cidade)
- Gestão de alarmes
- Capacidade de ter diferentes administradores
- Escalável
- Diferentes casos de uso implementados em diferentes áreas

2.7.5 Análise das plataformas

Existem diversas plataformas para recolha e visualização de informação de dispositivos sendo algumas delas direcionadas a cidades inteligentes.
A tabela que se encontra de seguida representa um resumo dos principais pontos comuns nas plataformas analisadas. Podemos verificar que estas aceitam diferentes protocolos de comunicação com os dispositivos, têm a capacidade de processar dados provenientes dos dispositivos, inserir dados de caracterização (atributos) manualmente, possuem um ambiente gráfico para melhor interação e possibilitam a visualização dos dados de atributos e de telemetria textualmente ou em widgets (com a exceção da plataforma Eclipse KURA). Todas as plataformas estão bem documentadas e são extensíveis devido ao seu desenvolvimento modular.
KAA e Eclipse KURA são plataformas desenvolvidas para a IoT genérica, OpenHAB foi desenvolvida para IoT mas com um foco principal na automação de Casas Inteligentes enquanto que Thingsboard foi desenvolvido com um foco em Cidades Inteligentes.

Tabela 2.1: *Características das plataformas analisadas*

	KAA	Eclipse KURA	OpenHAB	Thingsboard
Protocolos de conexão	✓	✓	✓	✓
Processamento de dados	✓	✓	✓	✓
Inserção manual de dados	✓	✓	✓	✓
Ambiente gráfico	✓	✓	✓	✓
Visualização de dados	✓	✓	✓	✓
Visualização em widgets	✓		✓	✓
Documentação	✓	✓	✓	✓
Expansível	✓	✓	✓	✓
Foco de implementação	IoT	IoT	Smart Home	Smart Cities

Podemos comprovar que todas estas plataformas se limitam a recolher informação de sensores e mostrá-la textualmente ou em graficamente. Nenhuma destas demostra capacidade de tratar dados de um grande número de dispositivos visto que é inviável analisar informação de sensor a sensor. É necessário que haja um cruzamento de dados e tomada de decisão para simplificar essa tarefa.

A escolha da plataforma a utilizar para cobrir as funcionalidades presentes na maioria das plataformas já existentes teve como base esta análise e deste modo a plataforma escolhida foi Thingsboard.

Thingsboard conta com um ambiente gráfico apelativo para organizar uma cidade, visto que foi desenvolvida para este propósito e conta com a possibilidade de se utilizar diferentes protocolos de comunicação para cobrir a generalidade dos dispositivos já em utilização nas cidades.

Ao realizar testes experimentais das aplicações, com o intuito de escolher uma, Thingsboard mostrou-se intuitivo e possível de desenvolver um protótipo de implementação do objetivo proposto nesta dissertação.

2.8 Ontologias

2.8.1 Introdução

A normalização dos dados auxilia a interpretação destes por máquinas para tomada de decisão, e para adaptação a diferentes situações e contextos. As tecnologias semânticas facilitam o raciocínio a partir de múltiplas informações heterogéneas, fontes e domínios e promovem a interoperabilidade entre uma variedade de aplicações e sistemas.

O principal benefício da utilização de ontologias é a organização de informações e representação do conhecimento formalmente, permitindo deste modo a partilha e reutilização

do conhecimento representado.
A utilização destas tecnologias semânticas nos sistemas de IoT pode automatizar a recolha de informações e tomada de decisões e, deste modo, facilitar o desenvolvimento de aplicações (Maarala and Riekki, 2017).
Atualmente existem diferentes ontologias para IoT, algumas específicas a alguns domínios como por exemplo a SEAS que se baseia no conceito de energia ou a M3m e oneM2M que se baseiam no conceito de máquina para máquina e outras mais genéricas como FIESTA-IOT que é uma agregação de diferentes ontologias e IoT-O que contém vários módulos como sensores, atuadores, serviços, energia entre outros.
Ontologias fornecem uma camada abstrata sobre os dados em domínios como a internet das coisas fazendo com que os dados dos dispositivos possam ser recolhidos e tratados por pessoas e aplicações (Jayaraman et al., 2015).
Uma ontologia pode ser definida como uma descrição formal de um domínio de conceitos, também denominados de classes, que possuem um conjunto de indivíduos ou instâncias e a representação de relacionamentos entre classes ou entre indivíduos.
A figura seguinte é um exemplo de uma parte da ontologia (SOSA/SSN).

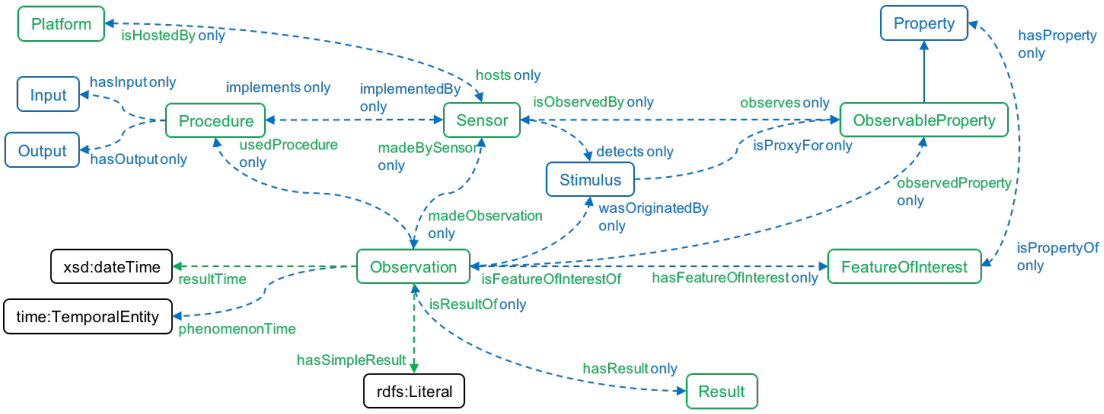

Figura 2.8: *Ontologia SOSA/SSN*

Combinando algumas ontologias será possível não só conseguir interoperabilidade semântica, como também permitir a utilização de ferramentas e técnicas avançadas para uma melhor análise e gestão.
Já existe um interesse em desenvolver uma ontologia genérica para IoT (Choi and Rhee, 2014) com o objetivo de normalizar os desenvolvimentos futuros mas ainda não existe uma implementação com uma aceitação considerável.
Os fabricantes não estão preocupados com o que são as ontologias do nível superior, mas sim preocupados com a identificação fácil do vocabulário que é relevante para eles (Groves et al., 2016). Basicamente cada empresa de dispositivos para IoT cria a sua ontologia baseando-se na informação que é relevante para o seu propósito.

A criação de uma ontologia geral é uma mais valia para facilitar o desenvolvimento, visto que tendo esta, todos os fabricantes iram respeitar essa normalização e deste modo iremos ter uma caracterização uniforme independentemente do fabricante que desenvolver o dispositivo. Quando as informações dos dispositivos respeitarem a normalização, o cruzamento de dados e tomada de decisão será facilitado não sendo necessário um pré processamento dos dados antes de os armazenar.

2.8.2 Ontologias para a web (OWL)

Tecnologias semânticas como *Web Ontology Language* (OWL) são padrões para o tratamento e definição de conceitos e relacionamento em diversos ambientes e devido a essa característica constituem uma solução promissora para ajudar a lidar com esse problema (Jayaraman et al., 2015).
O OWL[1] é uma linguagem da web semântica que foi desenvolvida pelo W3C para representar, de um modo rico e complexo, o conhecimento sobre determinado contexto. É baseada em lógica computacional para que o conhecimento expresso em OWL possa ser explorado computacionalmente. Apesar de OWL ter sido criado originalmente no contexto da web semântica foi adotada por diversas áreas da inteligência artificial para fins de representação do conhecimento.
O OWL é uma extensão da linguagem *RDF Schema* (RDFS), que por sua vez estende a linguagem RDF e todas são parte da pilha tecnológica da Web Semântica.

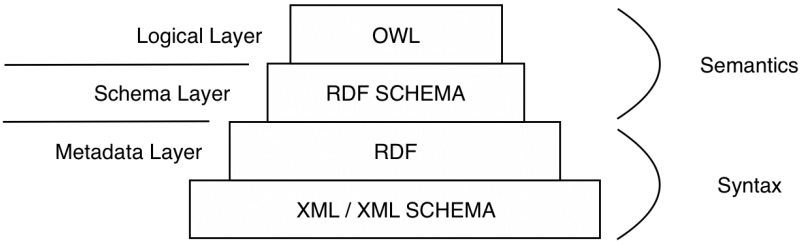

Figura 2.9: *Pilha tecnológica da Web Semântica*

Os dados RDF podem ser considerados como uma tabela com três colunas (coluna do recurso, coluna de propriedade e coluna de valor). O recurso no RDF é análogo a uma entidade numa base de dados relacional podendo estar definido numa tabela própria; a chave identificadora do recurso pode ser usada em várias outras tabelas para o associar às diferentes propriedades.
Em RDF, os campos são representados como linhas de propriedade e valor separadas que partilham o mesmo sujeito, geralmente a mesma chave única com o predicado sendo

[1] `https://www.w3.org/OWL/`

análogo ao nome da coluna e o objeto com os dados reais.
Ao contrário das bases de dados relacionais, a coluna do valor é heterogênea, ou seja, o tipo de dados por célula normalmente é implícito (ou especificado na ontologia) pelo valor da propriedade. Além disso, ao contrário do modelo relacional, o RDF pode ter várias entradas por propriedade.
O RDF limita-se a representar afirmações, não trazendo consigo nenhuma regra semântica, sendo estas atribuições realizadas pelo RDFS ou pelo OWL que são linguagens de modelação de dados, mas que dependem do RDF para serem implementadas.
O RDFS possibilita a descrição de grupos de recursos relacionados (RDF) e relações entre eles. Exemplos dessas propriedades são classes, subclasses e domínios.
Por exemplo um individuo pode ter uma instância RDF de um Animal e um Cão, o RDFS pode especificar que o Animal é uma classe e Cão é uma subclasse de Animal.
Importante mencionar que o RDFS é uma linguagem fraca o suficiente para permitir que afirmações inconsistentes ou incoerentes sejam feitas, por exemplo, afirmar que um recurso é simultaneamente instância e classe, ou mesmo uma instância pertencer a duas classes que deveriam ser disjuntas.
O OWL traz mais construções semânticas e possibilidade de restrições que o RDFS não consegue lidar. Exemplo disto é a possibilidade de declarar que duas classes são disjuntas entre si, ou seja, que, sendo A e B disjuntas, uma instância de A nunca pode ser instância de B.
Na figura seguinte, através da ferramenta Protégé, pode-se verificar que a classe Meteorology é uma subclasse de Telemetry e que Battery é uma subclasse de Attibute. Deste modo uma instância de meteorologia nunca pode ser instância de bateria.

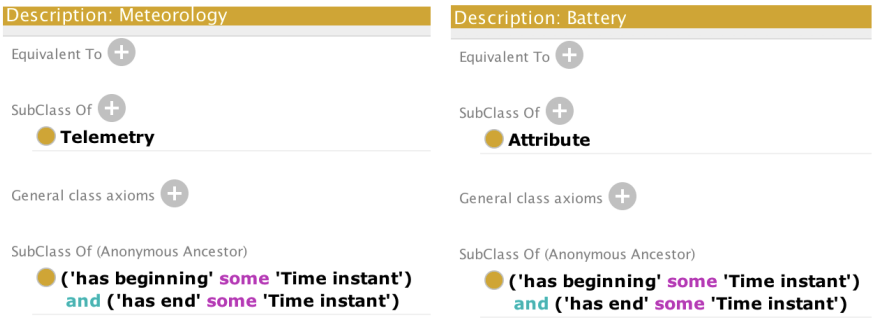

Figura 2.10: *Exemplo de classes e subclasses no Protégé*

O OWL permite também a criação de dois tipos de propriedades denominadas de ObjectProperty e DataProperty onde uma relação do primeiro tipo implica que tanto o sujeito como o objeto são instâncias enquanto que uma relação do segundo tipo implica que o sujeito seja uma instância e o objeto um valor.

Nas duas figuras seguintes pode-se verificar um exemplo de ObjectProperty e outro de DataProperty. No ObjectProperty temos o sameTime onde o sujeito e o objeto são instâncias, enquanto que no caso do DataProperty temos o AirTemperature onde se pode verificar que o sujeito é uma instância e o objeto é um valor, neste caso (xsd:unsignedInt).

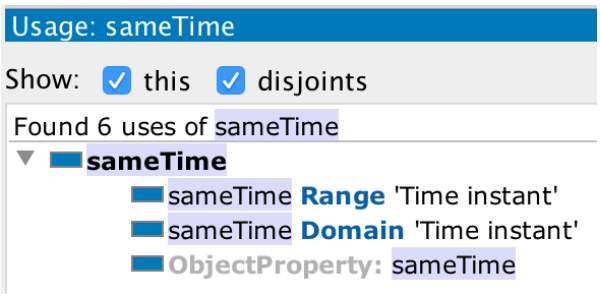

Figura 2.11: *Exemplo de um ObjectProperty*

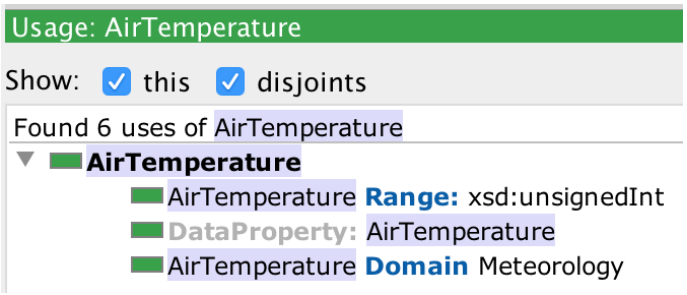

Figura 2.12: *Exemplo de um DataProperty*

Uma classe em OWL é uma classificação de dispositivos que partilham características comuns. Se um dispositivo é de uma determinada classe então enquadra-se na classificação semântica dada pela classe OWL permitindo que mecanismos de raciocínio desenhem um modelo inferido do modelo de ontologia base (Choi and Rhee, 2014).

2.8.3 Ontologias para IoT

O portal de vocabulários ligados (Linked Open Vocabularies, 2018) apresenta um grande resultado para uma pesquisa por etiquetas como IoT e para um conhecimento mais aprofundado, de seguida, fica uma descrição breve de algumas como por exemplo W3C SOSA/SSN, IoT-Light, SAREF, SEAS, M3, M3-lite, FIESTA-IoT, IoT-O, SAN e oneM2M. No portal Love4IoT podemos também encontrar um conjunto de vocabulários e ontologias usadas em IoT (Love4IoT, 2018).

A seleção das ontologias a apresentar para posteriormente utilizar na caracterização dos dispositivos teve como critério de escolha a cobertura das caracterizações que seriam pre-

cisas para o protótipo a desenvolver, assim como a fiabilidade da fonte destas e utilização em projetos relacionados com a temática IoT e cidades inteligentes.

W3C SOSA/SSN

(https://www.w3.org/TR/vocab-ssn/)
Semantic Sensor Network Ontology (SSN) é uma ontologia desenhada para descrever sensores, as suas observações, procedimentos, propriedades de interesse, e atuadores. Para isso recorre a uma outra ontologia denominada de SOSA (Sensor, Observation, Sample, and Actuator) e acrescenta-lhe mais caracterização.
Estas ontologias foram desenhadas e estão alinhadas com as recomendações e normas da W3C para a Web Semântica.

IoT-Light

(http://lov.okfn.org/dataset/lov/vocabs/iot-lite)
IoT Light é uma simplificação de SSN. É uma ontologia pequena e foi desenvolvida para representar recursos, entidades e serviços da Internet das Coisas.
Reduz a complexidade de outros modelos da IoT, descrevendo apenas os principais conceitos do domínio.
É focada em deteção, embora tenha um conceito de alto nível de atuação que permite qualquer extensão futura nessa área.
O facto de ser simples permite a representação e o uso por plataformas de IoT sem consumir tempo excessivo de processamento ao consultar a ontologia.

SAREF

(http://ontology.tno.nl/saref/)
A ontologia *Smart Appliances REFerenc* (SAREF), financiada pela UE e desenvolvida pela ETSI foi publicada como Especificação Técnica em Novembro de 2015 (TS 103 264 - V1.1.1). SAREF baseia-se no conceito de dispositivo. Um dispositivo é um objeto físico que desempenha uma determinada tarefa. Para desempenhar essa tarefa executa determinadas funções. Por exemplo, uma máquina de lavar desempenha uma determinada tarefa (lavar roupa) para a qual executa determinadas funções (iniciar, parar, entre outras). As funções têm comandos associados. SAREF tem muito em comum com a ontologia oneM2M, e foi estendida por proposta do grupo SmartM2M da ETSI.

SEAS

(https://ci.mines-stetienne.fr/seas/)
(http://w3id.org/seas/)

A ontologia *Smart Energy Aware Systems* (SEAS) define sistemas, conexões entre sistemas e pontos de conexão. A ontologia tem várias especializações, por exemplo no domínio da energia elétrica e sistemas de energia.

M3m, M3 Lite

(http://sensormeasurement.appspot.com/)
(http://lov.okfn.org/dataset/lov/vocabs/m3lite)
A ontologia M3 Lite foi proposta pelo projeto H2020 FIESTA-IOT, e é uma simplificação da ontologia Machine to Machine Measurement (M3) desenvolvida pela Eurocom. M3 Lite é compatível com ontologias utilizadas pelos projetos SmartSantander, University of Surrey, KETI (Coreia) e Com4Innov (França).
Esta taxonomia classifica dispositivos, o seu domínio de interesse (Saúde, Domótica, Cozinha Inteligente, Ambiente, entre outros), fenómenos e unidades de medida.

FIESTA-IOT

(http://ontology.fiesta-iot.eu/ontologyDocs/fiesta-iot/doc)
O projeto FIESTA-IOT propôs a sua própria ontologia que é uma agregação de M3 Lite, SSN e IoT-lite, além de outras ontologias fora do domínio de IoT.

IoT-O

(https://www.irit.fr/recherches/MELODI/ontologies/iot-o-sosa.html)
IoT-O é uma ontologia horizontal que pode ser estendida com conhecimento vertical (específica a áreas de aplicação). Contém vários módulos: sensores (baseado em SSN), atuadores (baseado em SAN), serviços (baseado em MSM) ciclo de vida e energia (baseado em powerOnt).

SAN

(http://lov.okfn.org/dataset/lov/vocabs/SAN)
Semantic Actuator Network (SAN) é uma ontologia que descreve atuadores. Um atuador é um dispositivo físico que tem um efeito no mundo. Muitos dos conceitos são importados de SSN, e permitem separar as duas ontologias, uma sobre atuadores e outra sobre sensores. Esta ontologia é também um módulo da ontologia IoT-O.

oneM2M

(http://www.onem2m.org)
O seu objetivo é a definição de especificações técnicas de interoperabilidade entre máquinas.

A ontologia define uma entidade básica a partir da qual as ontologias específicas podem derivar os seus conceitos.
A oneM2M visa permitir que diferentes dispositivos sejam ligados na IoT independentemente da rede subjacente.
As caracterizações presentes em oneM2M cobrem requisitos, arquitetura, especificações da interface de programação de aplicativos (API), soluções de segurança e mapeamento para protocolos comuns da indústria, como CoAP, MQTT e HTTP.
Com base em protocolos que permitem que aplicações em todos os segmentos da industria comuniquem uns com os outros, a oneM2M permite que os fornecedores de serviços combinem diferentes dispositivos, tecnologias e aplicações de IoT, o que é uma característica importante no esforço para fornecer serviços numa variedade de industrias.
oneM2M já foi usado em implantações de fornecedores de serviços na Coreia do Sul, Ásia e Europa para implantações de sistemas inteligentes de cidades e transportes.

WGS84

(https://www.w3.org/2003/01/geo/)
Não é uma ontologia de IoT mas é indispensável para representar localizações e por isso é referida aqui. Os conceitos base incluem SpatialThing (prédio, pessoa, viatura,...) e TemporalThing (intervalo, instante,...).

2.8.4 Protégé

Protégé [1] (Musen, 2016) é uma ferramenta, de código-fonte aberto, baseada em Java, que foi desenvolvida pela Universidade de Stanford com o propósito de ser um editor de ontologias, auxiliando assim a sua modelização.
Tem um forte apoio da sua comunidade de utilizadores desde académicos, governamentais e corporativos e é utilizado para construir soluções, baseadas no conhecimento, para as mais diversas áreas de aplicação (Horridge, 2011).
Consiste numa aplicação com interface gráfica com uma arquitetura de plug-in que pode ser adaptada para desenvolver aplicações baseadas em ontologias simples e complexas.
O resultado obtido do Protégé pode ser integrado com sistemas de regras para desenvolver um sistema inteligente.
Suporta a utilização de motores de inferência, bem como facilita a criação de regras *Semantic Web Rule Language* (SWRL) e consultas SPARQL.

[1] `http://protege.stanford.edu/`

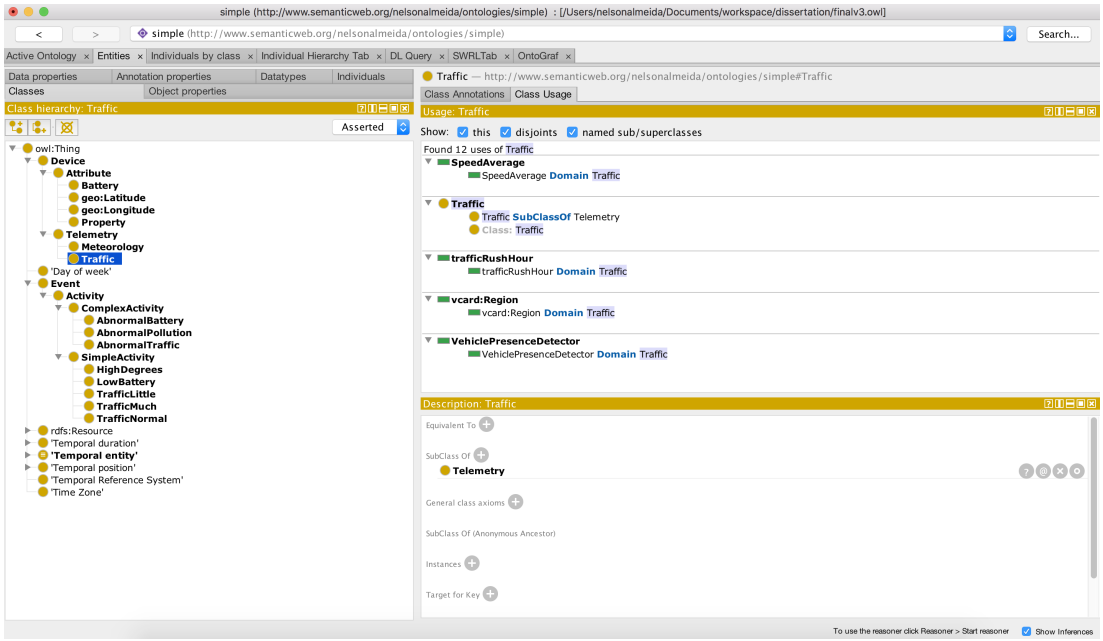

Figura 2.13: *Protégé*

Deste modo a sobrecarga de trabalhar diretamente com sintaxe XML/RDF fica aliviada tornando esta ferramenta útil.

2.9 Processamento semântico

Raciocinar consiste em tirar conclusões baseadas em fatos novos que não existem no conhecimento base. Raciocinar com regras é tipicamente baseado em logica de primeira ordem de predicado ou lógica descritiva para tirar conclusões de uma sequência de declarações derivadas por regras predefinidas.
Um mecanismo de raciocínio é uma ferramenta de software que realiza raciocínio com regras, podendo lidar com um largo conjunto de RDF e OWL. Um mecanismo destes conclui fatos de dados semânticos e ontologias baseado em regas pré-definidas (Maarala and Riekki, 2017).

2.9.1 SWRL

SWRL é uma linguagem proposta para a Web Semântica que pode ser usada para expressar regras e lógicas. As regras em SWRL são compostas por duas partes, um antecedente (corpo) e um consequente (cabeça).
Um exemplo típico da sintaxe do SWRL é:

$$hasParent(?x1, ?x2) \wedge hasBrother(?x2, ?x3) \implies hasUncle(?x1, ?x3)$$

Tal lê-se da seguinte forma: caso uma instância x1 possua como objeto da propriedade hasParent, uma instância x2, e este possua como objeto da propriedade hasBrother a instância x3, então x1 possui como objeto da propriedade hasUncle, a instância x3. De um modo mais simples, se x1 tem um pai (x2) e se x2 tem um irmão (x3), então x3 é tio de x1.
Assim sendo, se a base de conhecimeto possuir dois factos, "Rui" hasParent "Ana" e "Ana" hasBrother "João", um motor de inferência com a regra acima consegue inferir o facto "Rui" hasUncle "João".

2.10 Conclusão

Tendo como base o estado da arte analisado neste capítulo podemos comprovar que existe um grande interesse no desenvolvimento relacionado com a IoT, existindo diversas associações projetos e desenvolvimentos relacionados com esta temática.
No caso das cidades inteligentes estão a surgir cada vez mais iniciativas de implementação de dispositivos e desenvolvimento de plataformas de gestão destes mesmos para facilitar o crescimento da IoT nas cidades e torna-las assim inteligentes. No entanto, como ainda nos encontramos numa fase precoce desta abordagem ainda muito trabalho está por realizar.
No que a plataformas de gestão de dispositivos diz respeito, podemos encontrar diversas iniciativas, que na sua maioria cobrem a mesma necessidade: leitura, armazenamento e visualização dos dados disponibilizados pelos dispositivos. Este tipo de soluções provem da ideia de plataforma de gestão de casas inteligentes, campo que já se encontra mais desenvolvido devido ao muito menor número de dispositivos necessários de analisar.
Com uma análise sobre as ontologias existentes para IoT e cidades inteligentes verifica-se que ainda não dispomos de uma ontologia genérica de caracterização dos dispositivos, algo que dificulta a integração dos tão diversos dispositivos, desenvolvidos por inúmeras empresas distintas, cada um com caracterizações diferentes. Esta é uma necessidade que deve ser tratada o mais rapidamente possível para que exista uma norma que todas as empresas possam seguir e assim normalizar a caracterização dos dispositivos.
No caso das cidades, devido ao elevado número de dispositivos, não basta ler e mostrar a informação dos dispositivos pois torna-se inviável a análise desta informação por um administrador.
Para contornar esta situação é necessário tornar as plataformas "inteligentes", utilizando processamento semântico, de maneira que esta sejam capazes de ler, armazenar, interpretar, cruzar informação e posteriormente dar informações mais relevantes tendo em conta esta análise previa.
É com este objetivo que o desenvolvimento desta dissertação e respetivo protótipo de implementação se vai basear.

Capítulo 3

Especificação

3.1 Introdução

O objetivo do trabalho é definir a arquitetura e um protótipo de um sistema que possua automação para inferir situações simples para possibilitar o cruzamento dessas mesmas e permitir tomar decisões complexas num grau de abstração mais elevado.
O sistema proposto no presente trabalho tem como objetivos:

- Obter dados de dispositivos;
- Normalizar os dados obtidos e armazená-los;
- Caracterizar dispositivos através de atributos;
- Analisar os dados por intervalos de tempo e tipo de dispositivo;
- Utilizar regras para determinar e reagir a situações previamente definidas, assim como inferir atividades complexas;

Para fins deste trabalho, distingue-se atividades simples de atividades complexas. Atividades simples são acontecimentos como excesso de trânsito ou temperatura elevada, ou seja, tomando como exemplo sensores de trânsito um excesso de veículos num determinado intervalo de tempo será considerado como "Muito Trânsito" e no caso dos sensores de meteorologia, um registo de temperatura a cima de determinado valor é considerado "Temperatura Elevada".
Uma atividade complexa resulta do cruzamento de atividades simples, ou seja, caso aconteça a atividade simples "Muito Trânsito" e no mesmo intervalo de tempo a atividade "Temperatura Elevada" é assumida a atividade complexa "Possível Poluição".
Tendo como base solucionar problemas de mobilidade, ambiente e manutenção de dispositivos, para um cenário de teste foram simulados sensores na cidade do Porto, agrupados por diferentes zonas. Os sensores usados para este cenário de teste foram sensores de

trânsito e meteorologia, mas qualquer tipo de dispositivo pode ser adicionado. Nesse caso é necessário programar a aplicação para ao ler os dados provenientes destes os converta tendo em conta a normalização estipulada para armazenamento e posterior análise.
Este capítulo descreve a metodologia utilizada, a arquitetura do sistema, assim como a arquitetura dos diferentes módulos que o compõe e o seu contexto de utilização.
Em seguida elenca os requisitos funcionais, não-funcionais e de sistema necessários para a implementação.
Por fim, define os casos de uso que serão utilizados para demonstração das funcionalidades do sistema, na fase de testes e avaliação.

3.2 Metodologia utilizada

Para o desenvolvimento do protótipo foi utilizada uma metodologia de engenharia de software denominada de "The Waterfall Model" que consiste num desenvolvimento de software sequencial no qual o processo é visto como um fluir constante para a frente (como uma cascata) através das fases de comunicação, planeamento, implementação e testes.

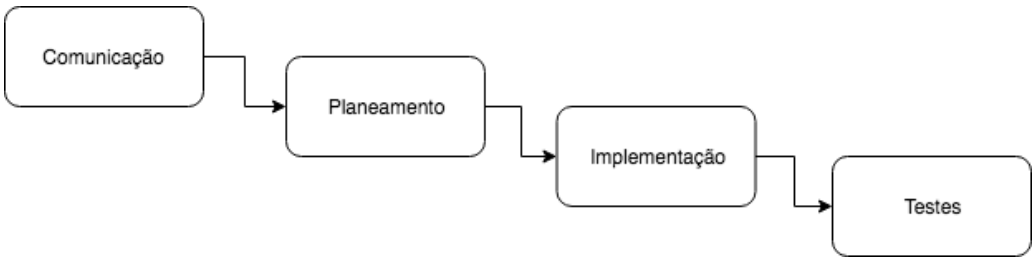

Figura 3.1: *The Waterfall Model*

Tendo como base este modelo, o projeto de implementação foi planeado de modo a atingir resultados práticos dentro do tempo definido para desenvolvimento do modelo demonstrativo do conceito desta dissertação.
Na fase de planeamento foi desenvolvida uma arquitetura do sistema, assim como dos diferentes módulos, nomeadamente o modulo de tratamento de dados e o modulo de leitura de dados e tomada de decisão. Foram definidos requisitos funcionais, não funcionais e de sistema baseados em projetos analisados ao longo do estudo para o desenvolvimento deste projeto. O contexto do utilizador e os casos de uso a desenvolver para posteriores testes também foram definidos nesta fase.
Na fase de desenvolvimento foi implementado o sistema pensado para demonstrar a viabilidade de uma plataforma do género descrito nesta dissertação.
Finalmente na fase de testes foi testada a plataforma tendo em conta os casos de uso definidos previamente e tiradas conclusões.

3.3 Arquitetura do sistema

O protótipo contará com um número elevado de informação de telemetria disponibilizada pelos dispositivos, que neste cenário de teste serão virtuais e que comunicarão com a plataforma através dos protocolos de comunicação usados pela maioria dos dispositivos de IoT em cidades inteligentes.

Para esta comunicação irá ser desenvolvida uma aplicação que leia dados de ficheiro *Comma Separated Values* (CSV) e os converta para *JavaScript Object Notation* (JSON) para enviar para a plataforma Thingsboard e converta também em RDF para enviar para a base de dados externa.

Esta camada adicional de tratamento de dados também irá contar com a possibilidade de normalizar a informação vinda dos dispositivos para a posterior tarefa de cruzamento de dados ser possível.

O modulo de análise de dados e tomada de decisões recorrerá a informação de telemetria armazenada em RDF na base de dados e à informação de atributos armazenada na plataforma Thingsboard para proceder à análise e envio dos resultados para painéis informativos a organizar na plataforma.

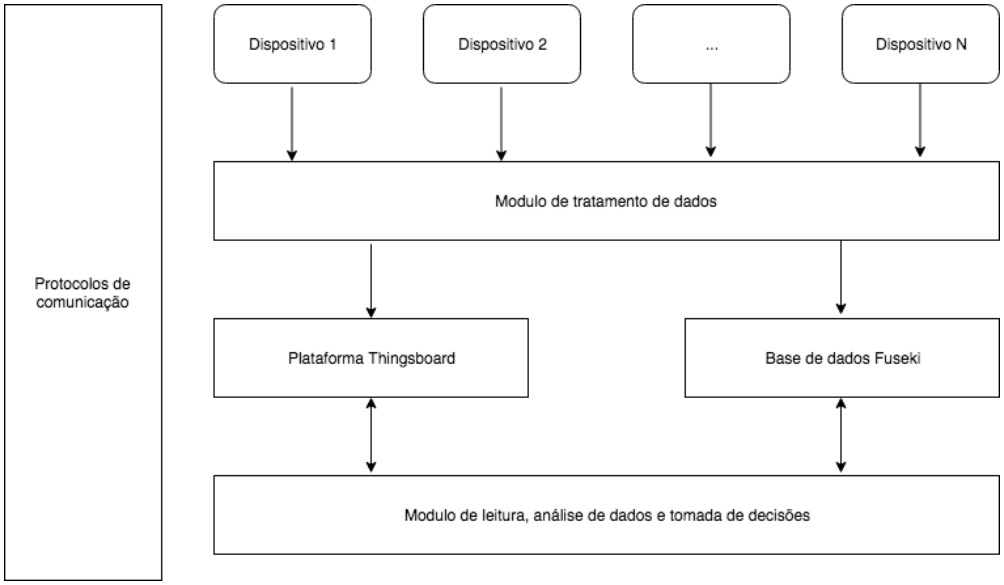

Figura 3.2: *Arquitetura do sistema*

Deste modo o protótipo utiliza dispositivos simulados, um modulo de tratamento de dados, uma base de dados externa (Fuseki), uma plataforma de IoT disponível (Thingsboard) e um modulo de leitura, análise de dados e tomada de decisões.

3.4 Arquitetura dos diferentes móludos

De seguida é apresentado o módulo de tratamento de dados e o modulo de leitura de dados e tomada de decisão. Em ambos é possível perceber a sua funcionalidade e o encaminhamento da informação entre os dispositivos, base de dados, módulos e plataforma.

3.4.1 Modulo de tratamento de dados

No diagrama seguinte encontra-se o encadeamento do carregamento de informações para a base de dados de triplos (Fuseki) e para a plataforma externa (Thingsboard).

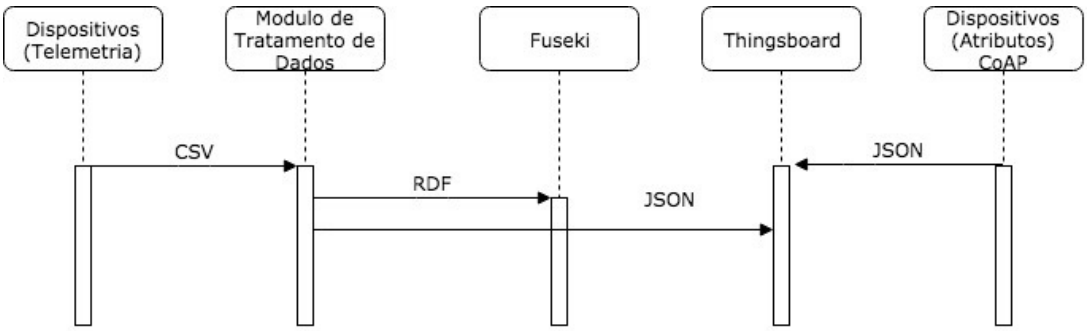

Figura 3.3: *Diagrama de carregamento de informação*

Podemos verificar que os dados de telemetria dos dispositivos simulados são enviados em formato CSV para o modulo de tratamento de dados. Neste modulo os dados recebidos são tratados para respeitarem a normalização desenvolvida e posteriormente convertidos para formato RDF para armazenamento na base de dados de triplos. Os dados são também convertidos para o formato JSON para envio para a plataforma Thingsboard.
Em relação aos dados de atributos dos dispositivos, como raramente são alterados posteriormente à sua inserção, são carregados diretamente, em formato JSON, para a plataforma Thingsboard através do protocolo CoAP.

3.4.2 Modulo de leitura de dados e tomada de decisão

No diagrama seguinte encontra-se o encadeamento da obtenção dos dados de telemetria e de atributos para posterior inferência e tomada de decisão de possíveis situações previstas.

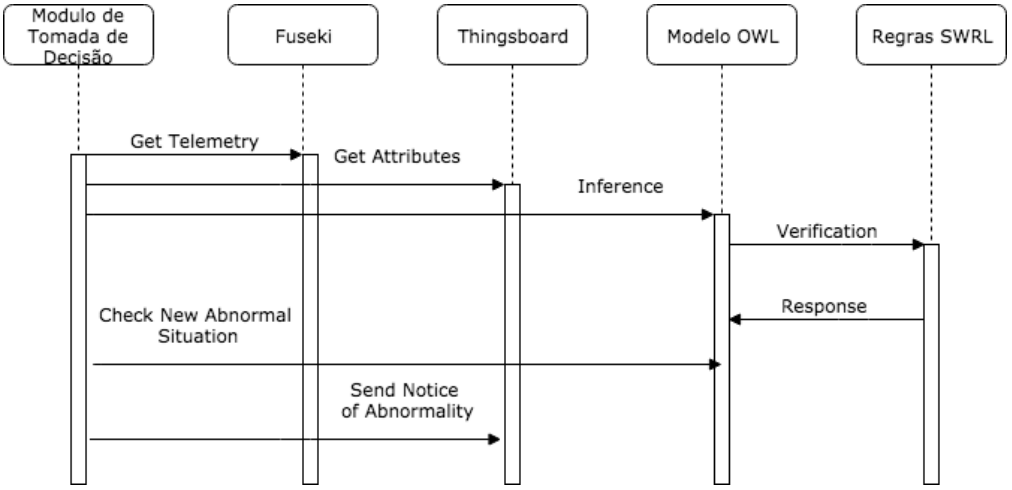

Figura 3.4: *Diagrama do leitura de dados e tomada de decisão*

Como vemos, existe um modulo de tomada de decisão que obtém os dados de telemetria. Esses dados são analisados em função de alguns parâmetros dependendo do tipo de dispositivos, tipo este que é obtido através da leitura dos atributos.
Um exemplo de um parâmetro analisado é, no caso da deteção de uma situação de trânsito, a verificação de hora de ponta tendo em conta a hora da leitura analisada.
Dependendo do resultado desta análise caso haja a necessidade de verificação para tomada de decisão, estes dados são inferidos no modelo OWL e posteriormente verificada a ocorrência ou não de possíveis situações através das regras SWRL definidas.

3.5 Contexto de utilizador

A implementação prevê a existência de um administrador que tenha acesso aos avisos despoletados pelo sistema, avisos estes que são gerados tendo em conta o cruzamento de informações de diferentes dispositivos para uma mais fácil análise das ocorrências que realmente tem interesse.

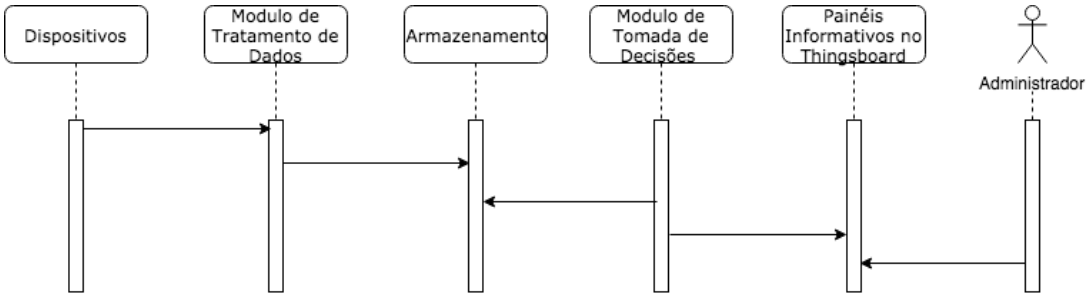

Figura 3.5: *Contexto de Utilizador*

As tomadas de decisão são relativamente simples, a nível de funcionalidade, mas foram pensadas somente para o cenário de teste. Qualquer outro cruzamento de informação para posterior tomada de decisão é possível de implementar.
Ao fim de um período previamente estipulado é analisada a possibilidade de novas ocorrências provenientes do cruzamento dos dados e caso haja alguma situação de realce essa informação é gerada e enviada para um painel de informações presente no Thingsboard.

3.6 Requisitos

Tendo como base a análise dos projetos de cidades inteligentes e plataformas de gestão destas mesmas os requisitos foram selecionados tendo em conta as lacunas que foram encontradas e necessidades que seria relevante implementar.
A necessidade de um grande número de dispositivos para tornar uma cidade inteligente e a necessidade de um mecanismo que analise dados destes para obter informação de alto nível são o principal foco deste projeto para que, deste modo, a tarefa dos administradores deste tipo de sistemas seja facilitada.

3.6.1 Requisitos funcionais

ID: RF 1
Título: Leitura e conversão de dados
Descrição: Deverá ser possível recolher e tratar dos dados provenientes de dispositivos simulados de modo a poder padroniza-los.

ID: RF 2
Título: Melhoramento da informação obtida
Descrição: Deverá ser possível acrescentar informação de caracterização aos dados recebidos dos dispositivos.

ID: RF 3
Título: Ontologia
Descrição: Deverá ser desenvolvida uma ontologia uniforme a todos os dispositivos.

ID: RF 4
Título: Envio de dados para platafoma existente
Descrição: Deverá ser possível o envio da informação dos dispositivos simulados para uma plataforma já existe no formato aceite pela mesma.

ID: RF 5
Título: Envio de dados para base de dados externa à plataforma

Descrição: Deverá ser possível o envio da informação em formato RDF com respetiva caracterização dos dados para base de dados de Triplos.

ID: RF 6
Título: Análise de dados e tomada de decisões
Descrição: Deverá ser possível a recolha e análise dos dados armazenados na base de dados externa com o propósito de tomar decisões através do cruzamento de informações de diferentes dispositivos.

ID: RF 7
Título: Inserção de dispositivos
Descrição: Deverá ser possível inserir dispositivos através da plataforma utilizada.

ID: RF 8
Título: Visualização de informação dos dispositivos
Descrição: Deverá ser possível visualizar informação de atributos e de telemetria dos dispositivos adicionados à plataforma.

ID: RF 9
Título: Visualização de avisos
Descrição: Deverá ser possível visualizar avisos provenientes da tomada de decisão

3.6.2 Requisitos não funcionais

ID: RNF 1
Título: Baixo custo
Descrição: Todas as aplicações e frameworks utilizadas devem ser gratuitas e os equipamentos caso necessários deverão ser de baixo custo.

ID: RNF 2
Título: Escalabilidade
Descrição: O sistema deverá permitir a inclusão de novos dispositivos e funcionalidades de forma fácil e segura.

3.6.3 Requisitos de sistema (Software e Hardware)

ID: RS 1
Título: Linguagem de programação de alto nível
Descrição: Idealmente o programa será desenvolvido em Java para poder tomar partido da sua grande popularidade na comunidade e pela consequente oferta de soluções gratuitas.

ID: RS 2
Título: Aplicações / Tecnologias de comunicação
Descrição: O sistema devará possibilitar a comunicação via HTTP, MQTT e CoAP.

ID: RS 3
Título: Base de dados
Descrição: O sistema deverá utilizar uma base de dados de triplos.

3.7 Casos de uso

Tendo como base solucionar problemas de mobilidade, ambiente e manutenção de dispositivos foi definido um caso de uso para cada um dos problemas para demostrar a viabilidade de uma plataforma deste género.
A escolha destes casos de uso teve também como base projetos analisados e dados de telemetria disponíveis para consulta e exploração.

ID: CU 1
Título: Controlo de trânsito anormal
Descrição: O sistema alerta o utilizador quando uma situação de trânsito anormal acontece. Uma situação de trânsito anormal baseia-se na ocorrência de trânsito lento fora das horas de ponta.

ID: CU 2
Título: Controle de possível ambiente poluído
Descrição: O sistema alerta o utilizador quando uma situação de possível poluição do ambiente fora do normal ocorre. Esta situação ocorre quando existe uma situação de trânsito elevado e a temperatura ambiente está elevada.

ID: CU 3
Título: Controlo de possível necessidade de carregar o dispositivo
Descrição: O sistema alerta o utilizador quando existe a necessidade de carregar um dispositivo manualmente. Caso o dispositivo seja alimentado por uma bateria, mas tenha um sistema de "auto carregamento" como por exemplo um painel solar, este não entrará nos avisos ao utilizador.

3.8 Conclusão

Este protótipo servirá para comprovar a necessidade de uma camada de mais alto nível para as plataformas existentes.

Quando estamos perante um elevado número de dispositivos, torna-se inviável a análise individual de cada um deles. Deste modo necessitamos que os dados de dispositivos de diferentes fabricantes, mas com o mesmo objetivo, sejam caracterizados da mesma maneira para que seja possível o processamento semântico de maneira a deduzir factos através de um grande número de informação, sem a interação do ser humano, através de regras definidas.

Para demonstrar a funcionalidade de um sistema deste género serão testados os casos de uso especificados em cima, que apesar de serem simples, servem para demonstrar que qualquer regra mais elaborada pode ser definida caso os dados estejam armazenados respeitando uma norma de caracterização.

Capítulo 4

Implementação

4.1 Introdução

Este capítulo descreve a implementação do projeto e o teste das tarefas requeridas pelos casos de uso descritos no capítulo anterior.
Na primeira secção descreve-se o tipo de dados recolhidos para a simulação dos dispositivos utilizados neste protótipo de implementação. Alguns destes dados foram recolhidos de amostras presentes em outros desenvolvimentos ou estudos de modo a tentar que os resultados fossem o mais realistas possíveis.
A segunda secção apresenta o modulo de tratamento de dados onde é realizada uma comunicação MQTT para simular o envio de informação de diferentes dispositivos para a plataforma. Resumidamente esta secção explica a recolha de dados de ficheiros CSV, envio através do protocolo MQTT, tratamento dos dados, dependendo do tipo de dispositivo que os envia, com o objetivo de os tornar em conformidade com o padrão estipulado e reencaminhamento dos dados já tratados para o Thingsboad e para uma base de dados de triplos para uma posterior análise.
A terceira secção apresenta a base de dados utilizada para o armazenamento dos dados depois de tratados pelo modulo anterior e que servirá de base de consulta para a posterior análise e cruzamento de dados para tomada de decisões.
Na quarta secção é apresentada a ontologia do protótipo, seguido do modelo de dados utilizado e sua explicação.
Na quinta secção temos a explicação da manipulação efetuada aos dados antes de estes serem inferidos para análise perante as regras SWRL definidas.
Na sexta secção encontra-se a explicação da manipulação do modelo OWL.
Na sétima secção encontra-se o modulo de cruzamento de dados para tomada de decisão onde se pode encontrar uma explicação sobre as regras SWRL. A inclusão de instâncias e realização de inferências também é detalhada assim como todo o processo de tomada de decisão deste protótipo.

Por último na oitava secção é apresentado o Thingsboard. Thingsboard é a plataforma usada para fornecer as funcionalidades que já se encontra na maioria das aplicações existentes para lidar com IoT em cidades inteligentes, como inserção de dispositivos, visualização da informação disponibilizada por estes, organização de painéis informativos e widgets entre outras. Nesta secção são descritas as principais funcionalidades do Thingsboard, quais as usadas neste protótipo, algumas configurações realizadas e demonstrações da sua utilização.

4.2 Conjunto de dados utilizados

Para o desenvolvimento da simulação foi realizada uma pesquisa por conjuntos de dados reais de dispositivos disponíveis em cidades inteligentes.
Existiu uma grande dificuldade em encontrar dados de diferentes tipos de dispositivos presentes numa mesma cidade e num mesmo intervalo de tempo.
A dificuldade aumenta quando se quer encontrar dados de dispositivos que dê para aplicar nos casos de uso definidos no capítulo anterior, ou seja, dados de trânsito e de temperatura de uma mesma região.
A pesquisa por este tipo de dados incidiu em projetos já realizados e que disponibilizaram os dados provenientes dos seus testes. Alguns desses projetos podem ser encontrados no seguinte endereço que reúne vinte e cinco conjuntos de dados para Deep Learning em IoT (https://hub.packtpub.com/25-datasets-deep-learning-iot/).
Um dos projetos que está listado entre os 25 presentes no endereço anterior é o CityPulse EU FP7 Project (http://iot.ee.surrey.ac.uk:8080/datasets.html#traffic) e neste foi encontrado um conjunto de dados de trânsito em formato CSV, datado de 2014, possível de utilizar no protótipo planeado devido à quantidade extensa de dados, simplicidade dos mesmo e sua fácil compreensão. Neste mesmo projeto encontram-se dados de meteorologia, mas estes são fornecidos em formato de texto e somente contém a informação do instante da recolha e a informação recolhida o que se torna muito simplista para a análise.
O projeto Smart Dublin (https://data.smartdublin.ie/dataset) também disponibiliza conjuntos de dados mas depois de uma análise aos poucos que ainda se encontravam disponíveis e que serviam para utilizar nos casos de uso planeados, ou seja, trânsito e meteorologia, não se mostraram intuitivos para uma integração no projeto.
Projetos como SynchroniCity (https://opendata.synchronicity-iot.eu/) e SmartSantander (https://data.lab.fiware.org/organization/santander) foram outros dos explorados para encontrar dados possíveis de utilizar mas novamente não foram encontrados grandes conjuntos de dados de meteorologia e de trânsito simples de integrar no protótipo desenvolvido.
A partir da pesquisa efetuada foram escolhidos os conjuntos de dados a utilizar neste projeto e estes serão apresentados a seguir.

4.2.1 Dados de trânsito

Para simular sensores de trânsito foi utilizado um conjunto de dados de contagem de veículos por intervalo de tempo, proveniente do CityPulse EU FP7 (CityPluse, 2018) e que se encontram no formato CSV.
É um conjunto de dados de 2014 e foi escolhido devido à grande quantidade de leituras e simplicidade da informação recolhida dos dispositivos.

```
status,avgSpeed,TIMESTAMP,vehicleCount,_id,REPORT_ID
OK,50,2014-08-01T07:50:00,5,20746220,traffic_zone1_01
OK,50,2014-08-01T07:55:00,6,20746392,traffic_zone1_01
OK,60,2014-08-01T08:00:00,4,20746723,traffic_zone1_01
OK,60,2014-08-01T08:05:00,1,20747172,traffic_zone1_01
OK,58,2014-08-01T08:10:00,3,20747545,traffic_zone1_01
OK,52,2014-08-01T08:15:00,6,20747994,traffic_zone1_01
OK,55,2014-08-01T08:20:00,6,20748443,traffic_zone1_01
OK,59,2014-08-01T08:25:00,6,20748892,traffic_zone1_01
OK,55,2014-08-01T08:30:00,13,20749341,traffic_zone1_01
OK,59,2014-08-01T08:35:00,10,20749790,traffic_zone1_01
OK,59,2014-08-01T08:40:00,4,20750239,traffic_zone1_01
OK,57,2014-08-01T08:45:00,6,20750688,traffic_zone1_01
OK,56,2014-08-01T08:50:00,9,20751137,traffic_zone1_01
OK,61,2014-08-01T08:55:00,10,20751586,traffic_zone1_01
OK,61,2014-08-01T09:00:00,10,20752035,traffic_zone1_01
...
```

No código anterior podemos ver que a informação recebida das leituras resume-se ao estado do dispositivo, média de velocidade dos carros que por ele passaram no intervalo de tempo da leitura, a data e hora da leitura, a contagem de veículos que passaram pelo dispositivo dentro do intervalo de tempo, a identificação da leitura e por fim a identificação do dispositivo que a efetuou.
Em relação a esta última, ou seja, a identificação do dispositivo que efetuou a leitura, foi a única informação alterada nos dados de trânsito para uma perceção mais intuitiva na integração na plataforma.

4.2.2 Dados de meteorologia

Como na pesquisa efetuada não foram encontrados dados de trânsito e de meteorologia de uma mesma região possíveis de utilizar neste protótipo, os dados de meteorologia foram simulados respeitando o formato dos dados de trânsito.
Deste modo os dados resumem-se ao estado do dispositivo, temperatura e humidade registadas, capacidade de bateria disponível, data e hora da leitura, a identificação da leitura e por fim a identificação do dispositivo que a efetuou.

Em relação à capacidade da bateria, esta informação foi adicionada para possibilitar a realização do caso de uso da verificação da necessidade de carregamento do dispositivo.

```
status,temperature,humidity,available_battery,TIMESTAMP,_id,REPORT_ID
OK,14,53,54,2014-08-01T08:00:00,20746980,meteorology_zone1_01
OK,13,60,53,2014-08-01T08:05:00,20747429,meteorology_zone1_01
OK,14,64,52,2014-08-01T08:10:00,20747802,meteorology_zone1_01
OK,15,62,51,2014-08-01T08:15:00,20748251,meteorology_zone1_01
OK,14,62,50,2014-08-01T08:20:00,20748700,meteorology_zone1_01
OK,15,67,49,2014-08-01T08:25:00,20749149,meteorology_zone1_01
OK,16,57,48,2014-08-01T08:30:00,20749598,meteorology_zone1_01
OK,16,61,47,2014-08-01T08:35:00,20750047,meteorology_zone1_01
OK,17,61,46,2014-08-01T08:40:00,20750496,meteorology_zone1_01
OK,18,60,45,2014-08-01T08:45:00,20750945,meteorology_zone1_01
OK,17,65,44,2014-08-01T08:50:00,20751394,meteorology_zone1_01
OK,18,60,43,2014-08-01T08:55:00,20751843,meteorology_zone1_01
OK,18,59,42,2014-08-01T09:00:00,20752292,meteorology_zone1_01
...
```

Para tornar o protótipo mais realista foi implementado um mecanismo de comunicação MQTT para que os dados sejam lidos e enviados através deste protocolo tal como poderia acontecer num cenário real.
MQTT foi escolhido como protocolo a utilizar devido a ser um protocolo leve de conexão máquina a máquina (M2M) e utilizado na Internet das Coisas. Através deste protocolo não existirá problema em ligar diversos dispositivos para simular uma cidade inteligente.
Este mecanismo conta com um "subscriber" que fica constantemente à espera de comunicações vindas dos dispositivos e cada dispositivo conta com um "publisher" que fica responsável por enviar a informação do dispositivo que neste caso é uma linha do CSV que corresponde a uma leitura do dispositivo.

```
MqttClient client=new MqttClient("tcp://localhost:1883", MqttClient.generateClientId());
client.setCallback( new SimpleMqttCallBack() );
client.connect();

client.subscribe("iot_data");
```

Figura 4.1: *MQTT - Subscriber*

No código anterior podemos ver a utilização do protocolo MQTT para iniciar um "subscriber" que neste caso ficou com a identificação de "iot_data".
Este mantem-se à escuta de comunicações de dispositivos direcionadas a ele.

```
System.out.println("== START PUBLISHER ==");

MqttClient client = new MqttClient("tcp://localhost:1883", MqttClient.generateClientId());
client.connect();
MqttMessage message = new MqttMessage();
message.setPayload(messageString.getBytes());
client.publish("iot_data", message);

client.disconnect();

System.out.println("== END PUBLISHER ==");
```

Figura 4.2: *MQTT - Publisher*

Em relação ao "publisher", existe um para cada dispositivo. No código anterior pode-se verificar que foi iniciado um cliente MQTT e as publicações originadas neste dispositivo direcionadas para "iot_data" que é o subscriber definido anteriormente para receber as leituras de todos os dispositivos presentes na implementação.
Como atualmente não existe uma normalização da informação disponibilizada pelos dispositivos de IoT, foi desenvolvido no modulo de tratamento de dado uma solução de normalização dos dados recebidos através do protocolo MQTT.

4.3 Tratamento dos dados

O modulo de tratamento de dados é dividido em duas partes:

- Tratamento dos dados dos dispositivos para posterior envio para a plataforma Thingsboard.
- Tratamento dos dados dos dispositivos para posterior envio para a base de dados externa.

O tratamento dos dados dos dispositivos para posterior envio para a plataforma Thingsboard tem de ser efetuado porque encontram-se em formato CSV e a plataforma somente aceita JSON.
Para que esta passagem seja autónoma e rápida foi desenvolvida em Java uma classe para fazer a passagem do formato CSV para JSON e enviar para o respetivo dispositivo no Thingsboard a informação. Cada dispositivo no Thingsboard corresponde a um dispositivo real ou no caso deste protótipo um dispositivo simulado.
Os dados recebidos são lidos linha a linha, pois cada linha significa um envio de informação do dispositivo. Para tornar mais realista foi estipulado um tempo de pausa para que os dados não sejam enviados sequencialmente, mas sim com um intervalo de tempo.
Esta classe ao receber os dados, através da leitura da identificação do dispositivo, consulta o Thingsboard à procura de algum dispositivo com a mesma identificação. Encontrando,

verifica nos seus atributos qual o tipo dele (neste caso, TRAFFIC ou METEOROLOGY) e dependendo desta informação traduz as designações dos dados recebidos para corresponder com a normalização previamente estipulada.
Deste modo e usando diferentes ficheiros CSV para simular diferentes dispositivos contamos com um mecanismo que simula a existência de dispositivos pela cidade a enviar informação constantemente para a plataforma.
A figura seguinte mostra a parte do código Java que implementa o envio da informação para o Thingsboard.

```
public static void tbUpload(String msgReceived, String deviceToken) {

    String Authorization = "Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiIyODAwOEB1ZnAuZWR1LnB0Iiwic2NvcGVzIjpbIlRFTkFOVF9

    String urlDevice = "http://localhost:8080/api/v1/" + deviceToken + "/telemetry";

    String[] command = { "/usr/bin/curl", "-X", "POST", "--header", "Content-Type: application/json", "--header",
            "Accept: */*", "--header", "X-Authorization:", Authorization, "-d", msgReceived, urlDevice };

    ProcessBuilder process = new ProcessBuilder(command);
    Process p;
    try {
        p = process.start();
        BufferedReader reader = new BufferedReader(new InputStreamReader(p.getInputStream()));
        StringBuilder builder = new StringBuilder();
        String line = null;
        while ((line = reader.readLine()) != null) {
            builder.append(line);
            builder.append(System.getProperty("line.separator"));
        }
        String result = builder.toString();

    } catch (IOException e) {
        System.out.print("error");
        e.printStackTrace();
    }
}
```

Figura 4.3: *Envio de informação para Thingsboard*

Como podemos ver na figura anterior, o método recebe uma mensagem e uma identificação do dispositivo que enviou essa mensagem.
Existe uma chave de autorização para interação com a API e a identificação do dispositivo com o qual é pretendido comunicar é inserida num URL.
Tanto a chave de autorização, como o URL que possui a identificação do dispositivo, como a respetiva mensagem a enviar, são inseridas num comando que posteriormente é iniciado.
Do lado do Thingsboard podemos visualizar a informação a chegar ao respetivo dispositivo como mostrado na figura seguinte. É mostrado o identificador do dispositivo, a marca temporal do envio, a telemetria recebida (neste caso a velocidade média e a contagem de veículos), a zona de localização do dispositivo, e o estado do dispositivo.

Latest telemetry

Last update time	Key	Value
2018-07-27 15:40:07	_id	40768198
2018-07-27 15:40:07	avgSpeed	53
2018-07-27 15:40:07	REPORT_ID	traffic_zone3_01
2018-07-27 15:40:07	status	OK
2018-07-27 15:40:07	TIMESTAMP	2014-08-01T19:00:00
2018-07-27 15:40:07	vehicleCount	1

Figura 4.4: *Telemetria de um dispositivo no Thingsboard*

A segunda parte refere-se ao tratamento dos dados dos dispositivos para posterior envio para a base de dados externa, ou seja, o FUSEKI.
O objetivo desta parte é permitir a representação de mais alto nível e com maior valor semântico a partir de dados recebidos de diferentes dispositivos simulados. O tratamento dos dados recebidos, antes de enviar para a base de dados é feito de modo a estes respeitarem a ontologia desenvolvida para caracterizar, de um modo uniforme, os dados prevenientes de dispositivos com a mesma funcionalidade, mas, por exemplo, de diferentes fabricantes e consequentemente com diferentes designações para o mesmo tipo de informação.
Como, num cenário real, estamos perante um largo número de diferentes tipos de dispositivos, estes têm de ser categorizados nos seus atributos com um tipo, tipo este que influenciará o padrão de telemetria a armazenar na base de dados.
O tipo do dispositivo, assim como outras informações de atributos estão armazenadas na plataforma Thingsboard.
Na imagem seguinte podemos visualizar uma parte do código Java que tem a função de perceber de que tipo de dispositivo é a informação que está a chegar, e de seguida normalizar os dados dependendo do tipo do dispositivo.

```
public void fusekiUpload(Map<String, Object> map, String deviceType) {

    if (deviceType.equals("traffic")) {

        String status = (String) map.get("status");
        String avgSpeed = (String) map.get("avgSpeed");
        String TIMESTAMP = (String) map.get("TIMESTAMP");
        String vehicleCount = (String) map.get("vehicleCount");
        String _id = (String) map.get("_id");
        String REPORT_ID = (String) map.get("REPORT_ID");

        final String UPDATE_TEMPLATE = "prefix ontology: <http://www.semanticweb.org/nelsonalmeida/ontologies/2018/2/smartcity#>"
                + "prefix sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#>"
                + "prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>"
                + "prefix xsd: <http://www.w3.org/2001/XMLSchema#>"
                + "prefix iot-lite: <http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#>"
                + "prefix ssn: <http://www.w3.org/ns/ssn/systems/>"
                + "INSERT DATA"
                + "{ <http://localhost:1212/smartcities/deviceinformation/%s>   sosa:State                  \""
                + status + "\" ;"
                + "                                                             m3-lite:SpeedAverage            \""
                + avgSpeed + "\" ;"
                + "                                                             xsd:TIMESTAMP               \""
                + TIMESTAMP + "\" ;"
                + "                                                             m3-lite:VehiclePresenceDetector         \""
                + vehicleCount + "\" ;"
                + "                                                             iot-lite:id                 \""
                + _id + "\" ;"
                + "                                                             sosa:Device             \""
                + REPORT_ID + "\" ." + "}   ";

        System.out.println(String.format("Adding in FUSEKI: %s", _id));
        UpdateProcessor upp = UpdateExecutionFactory.createRemote(
                UpdateFactory.create(String.format(UPDATE_TEMPLATE, _id)), "http://localhost:3030/ds/update");
        upp.execute();

    } else if (deviceType.equals("meteorology")) {
```

Figura 4.5: *Carregamento para Fuseki*

A função recebe a informação de telemetria e o tipo do dispositivo que a enviou. Seguidamente dependendo da informação do tipo de dispositivo, realiza a normalização dos dados pretendida fazendo um mapeamento das designações originais para as designações que estão de acordo com a ontologia definida neste protótipo.

De seguida podemos ver que a informação fica armazenada no Fuseki com a normalização estipulada, que neste trata-se da normalização de um dispositivo de trânsito.

```
<http://localhost:1212/smartcities/deviceinformation/20758770>
        <http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#id>
                "20758770" ;
        <http://purl.org/iot/vocab/m3-lite#SpeedAverage>
                "57" ;
        <http://purl.org/iot/vocab/m3-lite#VehiclePresenceDetector>
                "3" ;
        <http://www.w3.org/2001/XMLSchema#TIMESTAMP>
                "2014-08-01T10:15:00" ;
        <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#Device>
                "traffic_zone1_01" ;
        <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#State>
                "OK" .
```

Figura 4.6: *Dados no FUSEKI*

O armazenamento em formato RDF contém uma caracterização mais completa dos dados onde consta, neste caso:

- Recurso: um URI que identifica a informação
- Propriedade: um URI que identifica a propriedade do valor respeitando a ontologia
- Valor: o valor correspondente à propriedade

Na secção seguinte é descrita a tecnologia de bases de dados adequada para representar este tipo de informação.

4.4 Base de dados

Com o objetivo de caracterizar a informação da telemetria vinda dos dispositivos para posterior análise e consequente tomada de decisão foi utilizada uma base de dados de triplos para que deste modo a informação armazenada em formato RDF esteja em conformidade com a ontologia definida.
O RDF é um modelo de dados baseado em afirmações segundo o formato recurso - propriedade - valor, também conhecido como um triplo. Um triplo também pode ser representado como um grafo dirigido, no qual o recurso e o valor são nós e a propriedade é a aresta que os liga. A propriedade é uma relação entre dois recursos ou entre um recurso e um valor. Do ponto de vista do RDF, recursos e propriedades são identificados por um *Uniform Resource Identifier* (URI).
SPARQL é uma linguagem de consulta RDF, ou seja, uma linguagem de consulta semântica para bases de dados, capaz de recuperar e manipular dados armazenados em formato RDF.
O Apache Jena é uma Framework Java, de código fonte aberto, originalmente desenvolvida por investigadores do HP Labs, para desenvolver aplicações da web semântica (W3C, 2015). Fornece uma extensa biblioteca para facilitar o desenvolvimento de código que manipule RDF, RDFS, OWL e SPARQL de acordo com as recomendações publicadas do W3C. Jena inclui também um mecanismo de inferência baseado em regras para executar o raciocínio baseado em ontologias OWL e RDFS, e uma variedade de estratégias de armazenamento para armazenar triplos RDF.
Tem integração com bases de dados SDB (em formato SQL) ou TDB (em formato de triplos) e um servidor HTTP (Fuseki) para servir de ligação para outras aplicações.
Fuseki, sendo um sub projeto do Jena, é um servidor SPARQL com uma interface HTTP para dados RDF. Fornece protocolos SPARQL para consulta e atualização de dados.
A figura seguinte mostra o ambiente gráfico do Apache Jena Fuseki onde pode ser feita a gestão da base de dados, a visualização de consultas, entre outras funcionalidades.

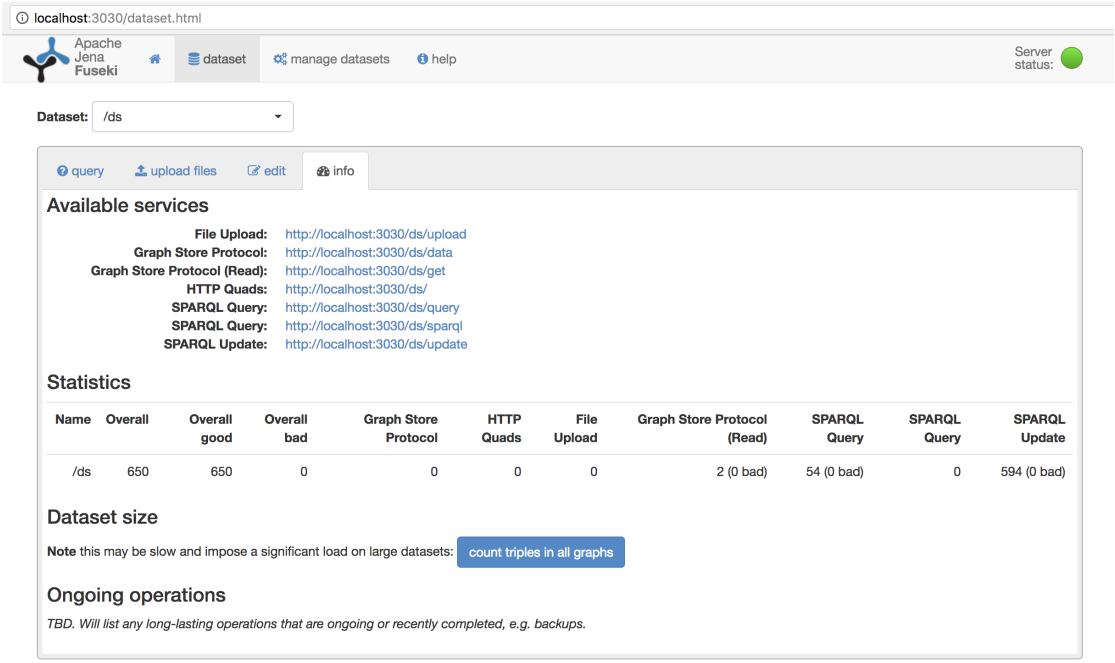

Figura 4.7: *Interface HTTP do FUSEKI*

O Jena foi utilizado para permitir a manipulação do modelo criado, ou seja, a criação das instâncias das classes, bem como a realização de inferências com base nas regras SWRL. Um exemplo de uma consulta à base de dados RDF utilizada no protótipo implementado é a recolha de dados de temperatura entre dois intervalos de tempo como podemos ver no código da imagem seguinte.

```
String serviceURI = "http://localhost:3030/ds/query";

String query = "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX m3-lite: <http://purl.org/iot/vocab/m3-lite#> "
        + "PREFIX sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#> "
        + "SELECT ?temp ?ts ?reportID WHERE { ?subject m3-lite:AirTemperature ?temp . ?subject xsd:TIMESTAMP ?ts . "
        + "?subject sosa:Device ?reportID . FILTER (xsd:dateTime(?ts) > xsd:dateTime(\""
        + beginningTime + "\") && xsd:dateTime(?ts) < xsd:dateTime(\"" + endTime + "\"))}";

QueryExecution qq = QueryExecutionFactory.sparqlService(serviceURI, query);
```

Figura 4.8: *Consulta a base de dados RDF*

Na consulta vemos os diferentes prefixos (xsd, m3-lite, sosa) que são utilizados na consulta, cada um com a respetiva designação. Depois dos prefixos temos a seleção (select) onde podemos ver que neste caso é pretendido obter os valores de ?temp (Temperatura), ?ts (Timestamp) e ?reportID (Token do dispositivo) e de seguida a condição onde é especificada a atribuição do prefixo ao campo que queremos obter e o filtro com o intervalo de tempo pretendido.

4.5 Ontologia do protótipo

4.5.1 Introdução

Tendo em conta a grande quantidade de ontologias já existentes, desenvolver uma nova não era necessário, mas sim reaproveitar as já existentes e conjuga-las de modo a obter o melhor de cada uma delas para com uma ontologia genérica facilitar a tarefa de gestão de uma grande quantidade de dispositivos.
Através da união destas ontologias foi definida uma simples ontologia de caracterização de dispositivos que inclui somente a caracterização da informação necessária para testar os casos de uso em estudo neste protótipo. Um estudo mais aprofundado sobre as necessidades de informação que as cidades têm daria para desenvolver uma ontologia genérica para a utilização pelos diferentes desenvolvedores de dispositivos de IoT.
Protégé foi utilizado para reunir as diferentes ontologias já existentes para as áreas de Cidades Inteligentes e IoT como SSN, SOSA, SAREF, SEAS, IOT-LITE, M3-LITE, VCARD, XSD e WGS84 POS. Nestas ontologias foram encontradas caracterizações para os dados que pretendia caracterizar tais como a temperatura, contagem de veículos, nível de bateria, entre outras.
Protégé foi utilizado também para facilitar o desenvolvimento de regras SWRL mas esta característica será abordada numa secção seguinte.

4.5.2 Modelo de dados da ontologia

Os dispositivos utilizados na demostração deste protótipo foram caracterizados com as seguintes informações de atributos e de telemetria:

Atributos gerais

Todos os dispositivos foram caracterizados com a respetiva localização através de coordenadas de “Latitude” e “Longitude”, com a zona a que estão associados através da característica “Region” (zona corresponde a uma divisão da cidade por partes), com o tipo de dispositivo (traffic, meteorology,...) através da característica “Property”, com o tipo de fonte de alimentação (battery, electric,...) através da característica “Electric Power”, com o tipo de carregamento (auto, manual) através da característica “is Powered By” e com o respetivo “Token” que identifica o dispositivo.

Caracteristica: Latitude
Prefixo: wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#lat>

Caracteristica: Longitude

Prefixo: wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#long>

Caracteristica: Region
Prefixo: vcard: <http://www.w3.org/2006/vcard/ns#Region>

Caracteristica: Property
Prefixo: saref: <https://w3id.org/saref/saref_Property>

Caracteristica: Electric Power
Prefixo: seas: <https://w3id.org/seas/seas_electricPower>

Caracteristica: is Powered By
Prefixo: seas: <https://w3id.org/seas/seas_isPoweredBy>

Caracteristica: Token
Prefixo: sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#Device>

Telemetria de tráfego

A telemetria de trafego foi caracterizada com um "Status" (OK, NO) que identifica o estado do dispositivo, com a média de velocidade dos veículos que passaram no intervalo de tempo da leitura através da característica "Avg Speed", com a contagem de veículos que passaram no intervalo de tempo da leitura através da característica "Vehicle Count", com a marca temporal da leitura "Timestamp", com o identificador da leitura "ID" e com o identificador do dispositivo que efetuou a leitura "Report ID".

Caracteristica: Status
Prefixo: sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#State>

Caracteristica: Avg Speed
Prefixo: m3-lite: <http://purl.org/iot/vocab/m3-lite#SpeedAverage>

Caracteristica: Vehicle Count
Prefixo: m3-lite: <http://purl.org/iot/vocab/m3-lite#VehiclePresenceDetector>

Caracteristica: Timestamp
Prefixo: xsd: <http://www.w3.org/2001/XMLSchema#TIMESTAMP>

Caracteristica: ID

Prefixo: iot-lite: <http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#id>

Caracteristica: Report ID
Prefixo: sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#Device>

Telemetria de meteorologia

Em relação à telemetria de meteorologia, esta foi igualmente caracterizada com um "Status" (OK, NO) que identifica o estado do dispositivo, com a temperatura e humidade registada caracterizada por "Temperature" e "Humidity"respetivamente, com a quantidade de bateria disponível do dispositivo caracterizada como "Available Battery", com a marca temporal da leitura "Timestamp", com o identificador da leitura "ID" e com o identificador do dispositivo que efetuou a leitura "Report ID".

Caracteristica: Status
Prefixo: sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#State>

Caracteristica: Temperature
Prefixo: m3-lite: <http://purl.org/iot/vocab/m3-lite#AirTemperature>

Caracteristica: Humidity
Prefixo: m3-lite: <http://purl.org/iot/vocab/m3-lite#Humidity>

Caracteristica: Available Battery
Prefixo: ssn: <http://www.w3.org/ns/ssn/systems/BatteryLifetime>

Caracteristica: Timestamp
Prefixo: xsd: <http://www.w3.org/2001/XMLSchema#TIMESTAMP>

Caracteristica: ID
Prefixo: iot-lite: <http://purl.oclc.org/NET/UNIS/iot-lite/iot-lite#id>

Caracteristica: Report ID
Prefixo: sosa: <https://www.irit.fr/recherches/MELODI/ontologies/IoT-O/sosa#Device>

4.5.3 Representação da ontologia

Usando a ferramenta Protégé e a sua capacidade de representar a ontologia em modo gráfico podemos ver nas imagens seguintes, de uma forma mais clara, a estrutura do modelo desenvolvido para este protótipo. O modelo possui uma classe Device que pode representar um qualquer dispositivo ao nível de atributos (Attribute) e telemetria (Telemetry).
No caso dos atributos são comuns a todos os tipos de dispositivos. Em relação à telemetria, esta conta com características comuns como:

- Identificação da leitura (id)
- Identificação do dispositivo que efetuou a leitura (Device)
- Quantidade de bateria disponível (BatteryLifeTime)
- Data e hora da leitura (xsd:TIMESTAMP)
- Estado do dispositivo (State)

Além destas características comuns conta também com características especificas do tipo de dispositivo, que neste caso são dois tipos (Traffic e Meteorology).
Na imagem seguinte é possível ver a caracterização descrita anteriormente.

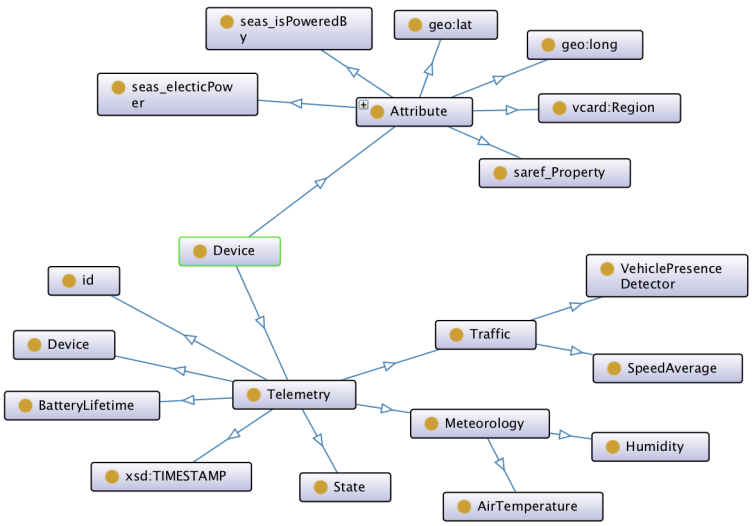

Figura 4.9: *Gráfico da ontologia utilizada nos dispositivos*

O modelo possui também uma classe chamada Event cujo propósito é representar um evento caracterizado por um instante inicial (propriedade hasBeginning) e um instante final (propriedade hasEnd). Ambas as propriedades pertencem à Ontologia OWL-Time.

Um evento ocorre quando uma atividade simples ou complexa é detetada. Na imagem seguinte é possível ver a caracterização das atividades simples e complexas.

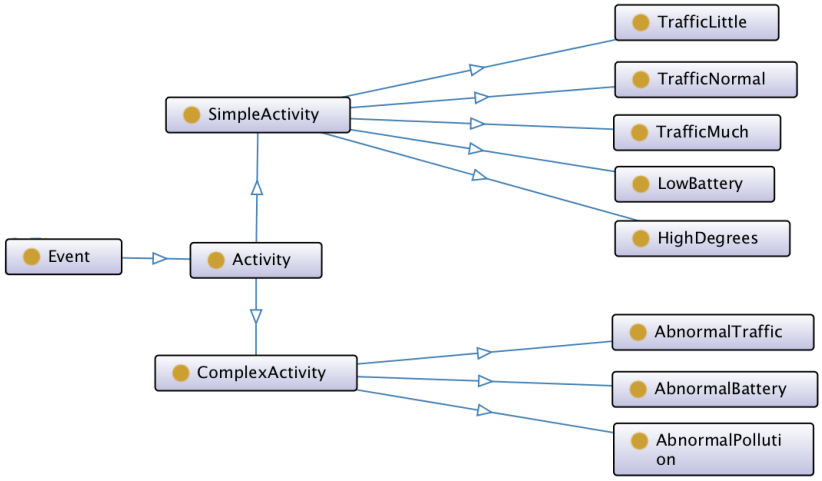

Figura 4.10: *Gráfico da ontologia utilizada nas actividades*

O objetivo neste trabalho é que cada dispositivo seja caracterizado, num cenário ideal, com a maior quantidade de atributos possível de inserir sobre o dispositivo e que seja comum a todos os diferentes tipos. Desta forma a informação será normalizada e posteriormente será possível tomar decisões sobre ela.
Para este senário de teste apenas alguns atributos foram adicionados, tais como, latitude e longitude (geo:lat e geo:long), zona onde está inserido o dispositivo (vcard:Region), tipo do dispositivo (saref:Property), tipo de fonte de alimentação (seas:electricPower) e tipo de carregamento (seas:isPoweredBy).
Estes foram os atributos necessários para realizar os casos de uso pretendidos, mas num cenário ideal uma boa e completa caracterização dos dispositivos daria origem a um vasto leque de oportunidades de tomadas de decisões em função das características de cada dispositivo e com isso automatizar pequenas tarefas ou otimizar serviços.
A subclasse Telemetry, alem das caracteristicas comuns, é subdividida em função dos tipos de dispositivos, ou seja, se tivermos dispositivos de trânsito e meteorologia, como neste protótipo, temos duas subclasses.
Fazendo esta divisão a caracterização de dispositivos de um mesmo tipo, sejam eles do mesmo fabricante ou não, é normalizada bastando para isso que seja feito um pré-processamento dos dados respeitando a caracterização estipulada na ontologia e somente depois armazenar essa informação na base de dados.
Para este cenário de teste, à semelhança dos atributos, também somente alguma caracterização foi adicionada. No caso dos dispositivos de trânsito temos a contagem de veículos

realizada pelo dispositivo (VehiclePresenceDetector) e média de velocidade dos veículos que passaram pelo sensor (SpeedAverage). Quanto aos sensores de meteorologia são caracterizados com temperatura (AirTemperature) e humidade (Humidity). A telemetria também é caracterizada por informação comum a qualquer tipo de dispositivo, informação como o estado do dispositivo (State), data e hora da telemetria (xsd:Timestamp), percentagem de bateria quando existente (BatteryLifetime), id da telemetria (id) e id do dispositivo que a emitiu (Device).
Quando determinados eventos ocorrem em simultâneo, ou quando eles ocorrem durante outros, segundo a álgebra de Allen (Allen, 1983), considera-se a existência de uma atividade complexa.
Deste modo a classe Activity subdivide-se em duas que são as SimpleActivity e as ComplexActivity.
Nas SimpleActivity, neste cenário de teste, podemos encontrar ocorrências simples como TrafficLittle, TrafficNormal, TrafficMuch, LowBattery e HighDegrees que são acontecimentos detetados através da informação disponibilizada pelos dispositivos e analisada tendo em conta regras SWRL definidas e que serão mais a frente explicadas.
Nas ComplexActivity podemos encontrar ocorrências mais complexas, tais como AbnormalTraffic, AbnormalBattery e AbnormalPollution, e que são detetadas através do cruzamento de informações simples obtidas igualmente através de regras SWRL.
Cada uma destas ocorrências vai ser explicada na secção seguinte.

4.6 Manipulação dos dados

Tendo em conta a possibilidade de um grande número de dispositivos instalados numa cidade, torna-se inviável tomar decisões analisando a informação uma a uma. Para resolver tal situação, neste protótipo a cidade foi dividida em zonas e a inferência de informação foi realizada, não de informação a informação, mas sim, calculando a média dentro de uma zona e de um intervalo de tempo de um tipo de dispositivos.
Tomando como exemplo os dispositivos de meteorologia, a informação disponibilizada por estes e armazenada na base de dados de triplos dentro de um intervalo de tempo que, para questões de teste, é de 15 minutos, é reunida, calculada a sua média e somente depois inferido um só valor de temperatura para a determinada zona naquele instante de tempo.
No caso dos dados disponibilizados pelos sensores de presença de carros o procedimento é o mesmo. Cada sensor envia o número de carros detetados num intervalo de tempo definido no próprio dispositivo. Essa informação é normalizada e enviada para a base de dados. Posteriormente no momento da consulta, esta é realizada com um intervalo de tempo igualmente de 15 minutos e calculada a média para de seguida ser inferido um só valor de número de veículos para aquela determinada zona naquele intervalo de tempo.

```
START AVERAGE DEVICE TEMPERATURE!!
averageDeviceTemperature - Temperature - x = 15
averageDeviceTemperature - Temperature - x = 15
averageDeviceTemperature - Temperature - x = 16
averageDeviceTemperature - Temperature - x = 14
MÉDIA METEOROLOGIA NO INTERVALOR E NA ZONA 1: 15
INDIVIDUAL meteorology in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#meteorology-test8
END AVERAGE DEVICE TEMPERATURE!!
START AVERAGE DEVICE VEHICLE COUNT!!
compareRushHourWithTimestamp - HORA DE PONTA
compareRushHourWithTimestamp - HORA DE PONTA
beginningTime e endTime DENTRO do horario de ponta!!!
averageDeviceVehicleCount - VehicleCount - x = 13
averageDeviceVehicleCount - VehicleCount - x = 6
averageDeviceVehicleCount - VehicleCount - x = 6
averageDeviceVehicleCount - VehicleCount - x = 6
MÉDIA TRAFEGO NO INTERVALOR E NA ZONA 1: 7
INDIVIDUAL traffic in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#traffic-test16
END AVERAGE DEVICE VEHICLE COUNT!!
```

Figura 4.11: *Calculo da média e inferência de dados de meteorologia e trânsito*

Em relação aos dados de bateria estes são inferidos um a um caso estejam a baixo de 20 % da bateria e estejam dentro do intervalo de tempo da consulta efetuada à base de dados que num cenário real seria o último instante de 15 minutos.
A restante análise somente será realizada nas regras SWRL distinguindo se o carregamento destes dispositivos com baixa percentagem de bateria é AUTO ou MANUAL.

4.7 Manipulação do modelo OWL

O modelo OWL representado pelas figuras 4.9 e 4.10 será o ponto de partida para a criação de instâncias de acontecimentos com o Jena. O objetivo do módulo é possibilitar a representação de instâncias de eventos relevantes para a caracterização de uma atividade complexa.
O Jena é formado por vários componentes, onde se destaca o Ontology API e o Reasoner API. O primeiro é composto por diversas classes que tratam da manipulação de modelos OWL.
Como se viu anteriormente, um modelo possui classes, estas possuem instâncias e as relações entre classes são representadas por propriedades. Há propriedades que representam o relacionamento entre uma classe e um tipo de dados.
A primeira tarefa do Jena é criar uma instância da classe OntModel, que representa o modelo OWL. Isto é feito pelo método createOntologyModel da classe ModelFactory e pode-se passar como argumento o motor de inferência a ser utilizado. Por fim, faz-se a instância da classe OntModel ler o InputStream que tem o conteúdo do ficheiro com o modelo criado pelo Protégé.
A figura seguinte representa as tarefas acima descritas.

```
String file="final.owl";
OntModel ontModel=ModelFactory.createOntologyModel(PelletReasonerFactory.THE_SPEC);
InputStream in= Utils.getInputStream(file);
ontModel.read(in,"RDF/XML");
```

Figura 4.12: *Exemplo de como carregar um modelo usando Jena*

A figura que se segue representa as operações básicas de um modelo OWL. Uma ontologia OWL pode importar outras ontologias para utilizar o modelo por elas representado. Assim, cada ontologia possui um espaço de nomes próprio capaz de identificá-la. Da mesma forma, o Jena identifica as entidades de um modelo com base no espaço de nomes da ontologia da qual estas façam parte.
A classe OntModel possui métodos para obter classes (getOntClass()), relacionamento entre instâncias (getObjectProperty()), relacionamento entre uma instância e um tipo de dados (getDatatypeProperty()), assim como criar instâncias (createIndividual()) e um Literal com um valor num certo tipo de dado (createTypedLiteral()). Uma instância pode adicionar relacionamentos como sendo do tipo ObjectProperty ou DatatypeProperty, passando, respetivamente, um Individual ou um Literal.
Mostrando uma parte do código da criação de um Individual para uma amostra de trânsito podemos ver a criação e adição do DataTypeProperty e respetivo Literal de cada propriedade.

```
@Override
public Individual createIndividual(OntModel ontModel) {
    Individual newTraffic=super.createIndividual(ontModel);

    DatatypeProperty trafficFlow = ontModel.getDatatypeProperty("http://purl.org/iot/vocab/m3-lite#VehiclePresenceDetector");
    Literal flow = Utils.createXSDUnsignedInt(this.flow);
    newTraffic.addProperty(trafficFlow, flow);

    DatatypeProperty trafficRushHour = ontModel.getDatatypeProperty(Utils.getNs() + "trafficRushHour");
    Literal rushHour = Utils.createXSDUnsignedInt(this.resultRushHour);
    newTraffic.addProperty(trafficRushHour, rushHour);

    DatatypeProperty trafficZone = ontModel.getDatatypeProperty("http://www.w3.org/2006/vcard/ns#Region");
    Literal zone = Utils.createXSDUnsignedInt(this.zone);
    newTraffic.addProperty(trafficZone, zone);

    return newTraffic;
}
```

Figura 4.13: *Exemplo de como criar DataTypeProperty e Literal*

A tarefa de inferir novos factos é feita automaticamente pelo motor de inferência passado para o modelo, não sendo necessária qualquer intervenção do utilizador. Para consultar a lista de instâncias de uma classe inferida, basta obter a classe através do OntModel e utilizar o método listInstances(), que retornar um iterador. A figura seguinte demonstra isso mesmo.

```
public static List<Individual> listIndividuals(final OntClass cls){
    List<Individual> individuals=new ArrayList<>();
    ExtendedIterator<Individual> it= (ExtendedIterator<Individual>) cls.listInstances();
    while(it.hasNext()){
        individuals.add(it.next());
    }
    return individuals;
}
```

Figura 4.14: *Obter e iterar uma classe inferida*

O Jena ainda possibilita a criação de Listeners para permitir que uma ação seja executada quando o modelo for alterado. Para tal, é necessário que a classe implemente a interface ModelChangedListener e inclua o código a ser executado num dos métodos constantes do contrato desta interface.
A estratégia do desenvolvimento da aplicação que implementa a gestão do modelo OWL é a criação de uma classe Java para cada classe OWL. Cada uma destas classes terá um método de criar o Individual para a respetiva instância, de modo a incluir a nova instância no modelo OWL. Um exemplo deste método pode ser visto na figura seguinte.

```
public Individual createIndividual(OntModel ontModel){
    OntClass eventClass = Utils.getOntClass(className);

    OntProperty hasBeginning=null;
    OntProperty hasEnd=null;
    Individual hasBeginningInd=null;
    Individual hasEndInd=null;
    if(validateInstant(this.getHasBeginning())) {
        hasBeginning=Utils.getTimeObjectProperty("hasBeginning");
        hasBeginningInd=this.getHasBeginning().createIndividual(ontModel);
    }
    if(validateInstant(this.getHasEnd())){
        hasEnd=Utils.getTimeObjectProperty("hasEnd");
        hasEndInd=this.getHasEnd().createIndividual(ontModel);
    }
    Individual newEvent=ontModel.createIndividual(Utils.getNs()+this.getEventURI(),eventClass);

    if(validateInstant(this.getHasBeginning())) {
        newEvent.addProperty(hasBeginning,hasBeginningInd);
    }

    if(validateInstant(this.getHasEnd())){
        newEvent.addProperty(hasEnd,hasEndInd);
    }
    return newEvent;
}
```

Figura 4.15: *Método que cria um Individual a partir de uma instância Java*

Este parte do código é comum a todos os Individuals e é nesta parte onde é atribuído o tempo de início e o tempo de fim do acontecimento.

4.8 Cruzamento de dados para tomada de decisão

Com a utilização de ontologia OWL é possível criar regras SWRL, com o propósito de inferir atividades complexas a partir da realização de diversas atividades simples em simultâneo.

4.8.1 Regras SWRL

A tomada de decisão implica a criação de várias regras cujas condições permitam a realização de inferências com o intuito de identificar atividades complexas para posteriormente serem tratadas de modo a simplificar a análise dos acontecimentos detetados pelos dispositivos nas cidades inteligentes.
Começando pelas regras que originam as SimpleActivity podemos ver através da seguinte que obtemos um TrafficMuch se recebermos informação de um sensor de Traffic (informação que identifiquei com a variável ?x) e se nessa informação estiver presente a informação de mlite:VehiclePresenceDetector, informação esta que fica atribuída à variável ?y e finalmente se, utilizando as condições presentes no swrlb, o valor da variável ?y for maior ou igual a 5. Para a situação de TrafficLittle e TrafficNormal somente as condições de swrlb tiveram de ser alteradas para o desejado.
Swrlb significa swrl *built-ins* (integrado) e trata-se de um sistema de recursos internos para ajudar na interoperabilidade do SWRL, fornecendo assim uma infraestrutura extensível e modular.

```
simple:Traffic(?x) ^ mlite:VehiclePresenceDetector(?x, ?y) ^
swrlb:greaterThanOrEqual(?y, 5) -> simple:TrafficMuch(?x)
```

Em relação à receção de informação de Meteorology (informação identificada com a variável ?x) se esta contiver informação de mlite:AirTemperature fica atribuído o seu valor na variável ?y e posteriormente usando a regra swrlb:greaterThanOrEqual caso o seu valor seja maior ou igual a 20 temos um HighDegrees.

```
simple:Meteorology(?x) ^ mlite:AirTemperature(?x, ?y) ^
swrlb:greaterThanOrEqual(?y, 20) -> simple:HighDegrees(?x)
```

Quando a regra SWRL implica a informação vinda de mais do que um Event é necessário saber se estes eventos estão dentro do mesmo intervalo de tempo para que não seja possível comparar dados de intervalos de tempo diferentes.
Na condição exposta de seguida pode-se ver que temos dois eventos, um atribuído à variável ?x e outro atribuído à variável ?y e de seguida coloca-se a informação de time:hasBeginning e time:hasEnd de cada um deles em variáveis ?startX, ?endX, ?startY e ?endY. Posteriormente especifica-se que as variáveis *start* e *end* são do tipo inXSDDateTime e por fim

verifica-se se os *starts* são iguais assim como os *ends*. Caso se comprove obtém-se um sameTime dos enventos ?x e ?y.

```
simple:Event(?x) ^ simple:Event(?y) ^
time:hasBeginning(?x, ?startX) ^ time:hasBeginning(?y, ?startY) ^
time:hasEnd(?x, ?endX) ^ time:hasEnd(?y, ?endY) ^
simple:inXSDDateTime(?startX, ?timeStartX) ^
simple:inXSDDateTime(?endX, ?timeEndX) ^
simple:inXSDDateTime(?startY, ?timeStartY) ^
simple:inXSDDateTime(?endY, ?timeEndY) ^
swrlb:equal(?timeStartX, ?timeStartY) ^
swrlb:equal(?timeEndX, ?timeEndY) -> simple:sameTime(?x, ?y)
```

A regra anterior é utilizada para determinar se dois eventos, por exemplo um de trânsito e outro de meteorologia, acontecem ao mesmo tempo para poder determinar se há probabilidade de ambiente poluente na zona.
Na regra seguinte podemos ver as condições necessárias para ocorrer AbnormalPollution na zona 1.
Como vemos tem de existir um Event Traffic que fica atribuído à variável ?t e este tem de conter um mlite:VehiclePresenceDetector que fica atribuído à variável ?tFlow e para a regra acontecer tem de ser maior ou igual a 5. Além disto o Traffic tem de conter um vcard:Region que fica atribuído à variável ?tZone e tem de ser igual a 1 para pertencer à zona 1. Tem de existir um segundo Event, desta vez de Meteorology que contenha um mlite:AirTemperature que fica atribuído à variável ?mDegrees e tem de ser maior ou igual a 20. O Meteorology tem de conter um vcard:Region que fica atribuído à variável ?mZone e tem de ser igual a 1 para pertencer à zona 1 tal como o evento de Traffic. Por fim ambos tem de respeitar a condição de sameTime explicada anteriormente. Respeitando todas estas condições estamos perante um AbnormalPollution.

```
simple:Traffic(?t) ^ mlite:VehiclePresenceDetector(?t, ?tFlow) ^
swrlb:greaterThanOrEqual(?tFlow, 5) ^
vcard:Region(?t, ?tZone) ^ swrlb:equal(?tZone, 1) ^
simple:Meteorology(?m) ^ mlite:AirTemperature(?m, ?mDegrees) ^
swrlb:greaterThanOrEqual(?mDegrees, 20) ^
vcard:Region(?m, ?mZone) ^ swrlb:equal(?mZone, 1) ^
simple:sameTime(?t, ?m) -> simple:AbnormalPollution(?t)
```

Em relação à regra que determina o AbnormalTraffic, Podemos ver a seguir que é necessário a ocorrência de um Event Traffic que fica atribuído à variável ?x e este tem de conter um mlite:VehiclePresenceDetector que fica atribuído à variável ?f e para a condição acontecer tem de ser maior ou igual a 5. Além disto o Traffic tem de conter um vcard:Region que fica atribuído à variável ?z e tem de ser igual a 1 para pertencer à zona 1. Por fim

Traffic tem de conter um trafficRushHour que fica atribuído à variável ?r e vai identificar se estamos perante uma hora de ponta ou não. Caso esta variável seja 1 estamos em hora de ponta, caso seja menor do que 1, ou seja 0, é porque não estamos em hora de ponta. A regra de AbnormalTraffic somente acontece quando existe um excesso de tráfego fora das horas de ponta.

```
simple:Traffic(?x) ^ mlite:VehiclePresenceDetector(?x, ?f) ^
swrlb:greaterThanOrEqual(?f, 5) ^ vcard:Region(?x, ?z) ^
swrlb:equal(?z, 1) ^ simple:trafficRushHour(?x, ?r) ^
swrlb:lessThan(?r, 1) -> simple:AbnormalTraffic(?x)
```

Por fim em relação à regra de AbnormalBattery, esta acontece quando estamos perante uma ocorrência de um evento Battery, que contem um BatteryLifetime que fica atribuído à variável ?y e que seja menor do que 20. Além disso a regra só é executada se existir um seas_isPoweredBy que fica atribuído à variável ?z e caso este seja "MANUAL". Battery também tem um Region que identifica a zona onde está o dispositivo e por exemplo caso este esteja na zona 1 este atributo tem de ser igualmente 1.

```
simple:Battery(?x) ^ ssn:BatteryLifetime(?x, ?y) ^
swrlb:lessThan(?y, 20) ^ seas:seas_isPoweredBy(?x, ?z) ^
swrlb:stringEqualIgnoreCase(?z, "MANUAL") ^ vcard:Region(?x, ?zoneB) ^
swrlb:equal(?zoneB, 1) -> simple:AbnormalBattery(?x)
```

Na secção seguinte será abordada a utilização da plataforma Thingsboard que além das funcionalidades de visualização de dispositivos, telemetria e atributos servirá para visualizar, em painéis informativos, a informação originada das tomadas de decisão realizadas em função das regras anteriormente demonstradas.

4.9 Thingsboard

A utilização de uma plataforma para a realização de tarefas como inserção, armazenamento e visualização de dispositivos e informação foi essencial para o foco do trabalho ser o desenvolvimento de um modulo de normalização e tomada de decisões que completasse esta falha que existe em todas as plataformas até à data exploradas.
Thingsboard foi a plataforma escolhida devido ao seu ambiente gráfico completo, vasto leque de protocolos de comunicação, documentação existente, casos de uso realizados e uma API REST que possibilita interagir com o Thignsboad para controlar a maioria das suas funcionalidades como configurações de administração, utilizadores, autenticação, alarmes, regiões, dispositivos, Plug-in, regras entre outras.

4.9.1 Instalação através do Docker

Thingsboard pode ser instalado em diferentes sistemas como Windows, Linux, Raspberry Pi 3 e Docker, sendo esta ultima a escolhida para a instalação e utilização da plataforma. Doker é uma plataforma de código fonte aberto, escrita em Go, que visa facilitar a criação e administração de ambientes isolados, permitindo assim a independência entre aplicativos e infraestruturas e os programadores e as operações (Docker, 2018).
Docker não é um sistema de virtualização tradicional. Enquanto num ambiente tradicional temos um S.O. completo e isolado, dentro do Docker temos recursos isolados que utilizam bibliotecas de Kernel em comum (entre computadores e contentores) (Diedrich, 2015).
Com o Docker é possível empacotar uma aplicação ou ambiente inteiro dentro de um contentor e a partir desse momento o ambiente inteiro torna-se portável para qualquer outro computador que contenha o Docker instalado. Deste modo reduz-se o tempo de implementação do sistema em infraestruturas ou até mesmo aplicações, pois deixa de haver a necessidade de ajustes de ambiente para o correto funcionamento do serviço. Sendo o ambiente sempre o mesmo, basta configura-lo uma vez e replicar quantas vezes for necessário.

4.9.2 Configuração

Depois de finalizada a instalação do Thingsboard ficamos perante um ambiente de administração web com diferentes menus como Plugins e Rules onde é possível criar regras e comportamentos mediante a informação atual dos dipositivos. Temos também um menu de Customers onde é possível definir diferentes tipos de clientes e suas caracterizações. Existe também o menu de Assets onde é possível subdividir por exemplo países em cidades e cidades em zonas e o menu de Devices onde pode ser visualizada toda a informação de atributos e telemetria de cada um dos dipositivos que sejam inseridos na plataforma assim como criar alarmes mediante valores obtidos na telemetria. Estes três últimos menus (Customers, Assets e Devices) são fundamentais para a organização da plataforma pois diferentes Customers podem ser responsáveis por diferentes Assets e cada Device fica atribuído a um Asset para uma maior organização. De seguida temos o menu de Widgets Library que contém um leque alargado de widgets tais como alarmes, mostradores analógicos e digitais, gráficos, tabelas, mapas e widgets de controlo além de possibilitar também o desenvolvimento de novos. Por fim contamos com o menu Dasboard onde podemos organizar vários paineis com diferentes disposições e widgets para diferentes Customers e Assets.

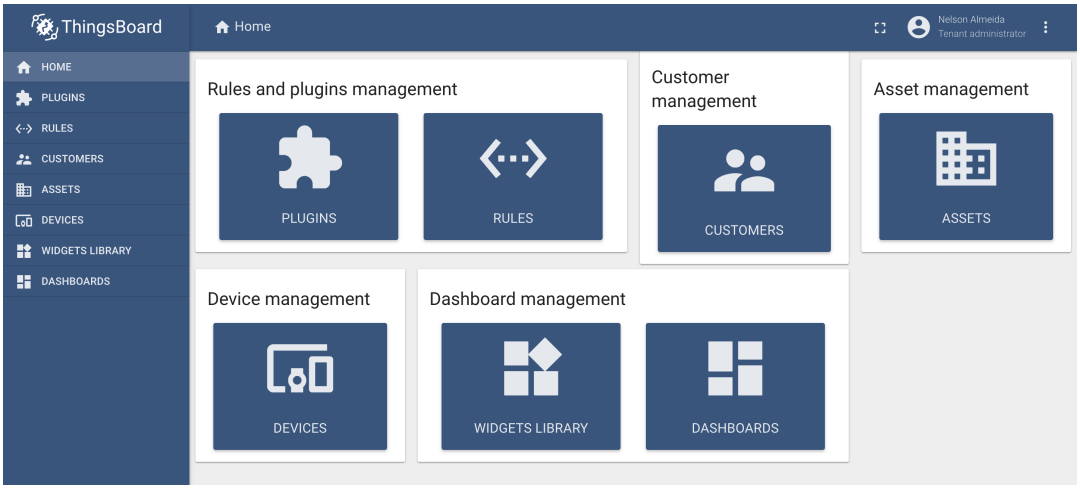

Figura 4.16: *Menu principal do Thingsboard*

Para a caracterização de dispositivos simulados foi utilizado o protocolo CoAP e a API REST para carregar as informações de atributos do dispositivo.
O protocolo CoAP foi utilizado por ser uma das opções disponibilizadas pelo Thingsbord e além disso, por ser um protocolo desenvolvido para aplicações máquina a máquina (M2M) com a possibilidade de ser utilizado na Internet das Coisas.
Em seguida é possivel visuailizar um dos comandos CoAP para carregamento de informação para dispositivos.

```
cat attributes-data.json | coap post
coap://127.0.0.1/api/v1/meteorology\_zone1\_02/attributes
```

A seguir temos um exemplo de atributos carregados para um dispositivo.

```
{"wgs84\_pos\#lat":"41.151886",
"wgs84\_pos\#long":"-8.610172",
"vcard\#Region":"1",
"saref\#saref\_Property":"meteorology",
"seas\#seas\_electicPower":"battery",
"seas\#seas\_isPoweredBy":"AUTO",
"sosa\#Device":"meteorology\_zone1\_02"\}
```

Em relação aos atributos este foi o método utilizado neste protótipo pois a alteração dos seus valores é pouca e para um cenário de teste não seria um requisito fundamental para comprovar a sua utilidade.
Em relação a telemetria este não foi o método utilizado. Em vez de estar constantemente a carregar dados através deste protocolo, esta tarefa foi automatizada como é explicado no capítulo anterior de Tratamento dos Dados.

A funcionalidade de painel do Thingsboard foi usada para mostrar o mapeamento das zonas, assim como a informação obtida através das tomadas de decisão realizadas fora do Thingsboard. Para este protótipo foram apenas programadas três zonas distintas dentro da cidade do Porto, uma centrada na zona dos Aliados, outra centrada na Ponte do Freixo e outra centrada no nó da VCI com a A3. Nas imagens seguintes pode-se ver a disposição das zonas no mapa, a lista destas mesmas zonas e três painéis de alerta de anomalias, um para alertas de trânsito, outro para alertas de possível poluição e outro para alertas de dispositivos com bateria reduzida. Os dois primeiros painéis foram programados para ter a informação do tipo de alerta, mensagem de alerta, zona do acontecimento e data e hora do mesmo.

Em relação ao painel de alerta de bateria reduzida, este disponibiliza a informação do tipo de anomalia, mensagem que identifica qual o dispositivo que necessita de ser carregado manualmente, a data e hora que essa informação foi obtida e a zona onde o dispositivo se encontra.

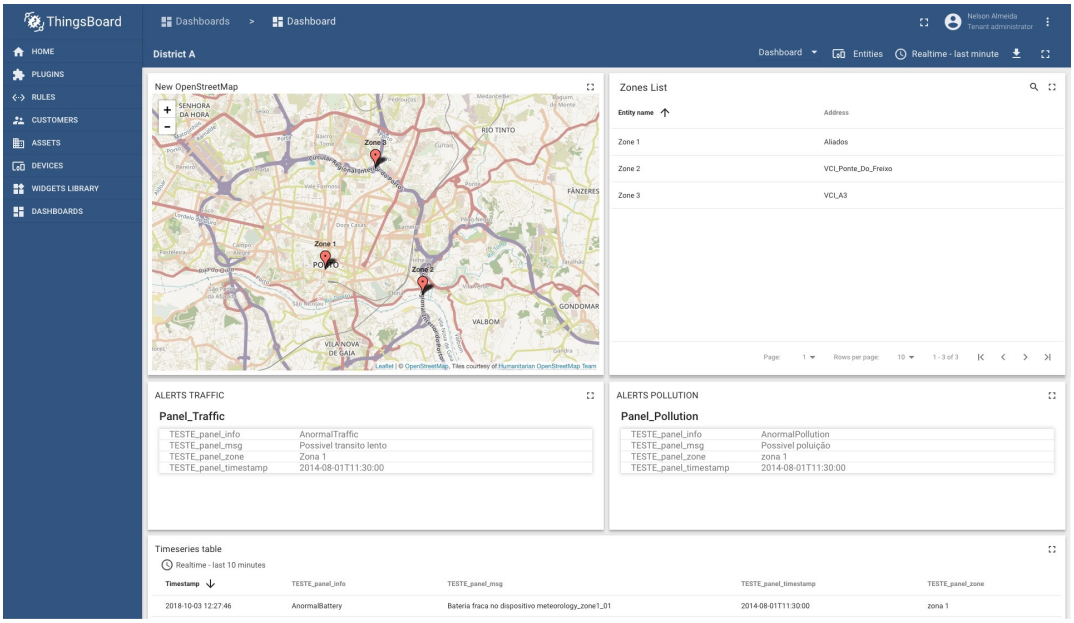

Figura 4.17: *Painel principal do Thingsboard*

Selecionando qualquer uma das zonas temos acesso a outro painel secundário com informação mais detalhada dos dispositivos dessa zona onde por exemplo podemos encontrar a disposição de sensor a sensor, a telemetria instantânea dos dispositivos entre outras informações que podem ser adicionadas.

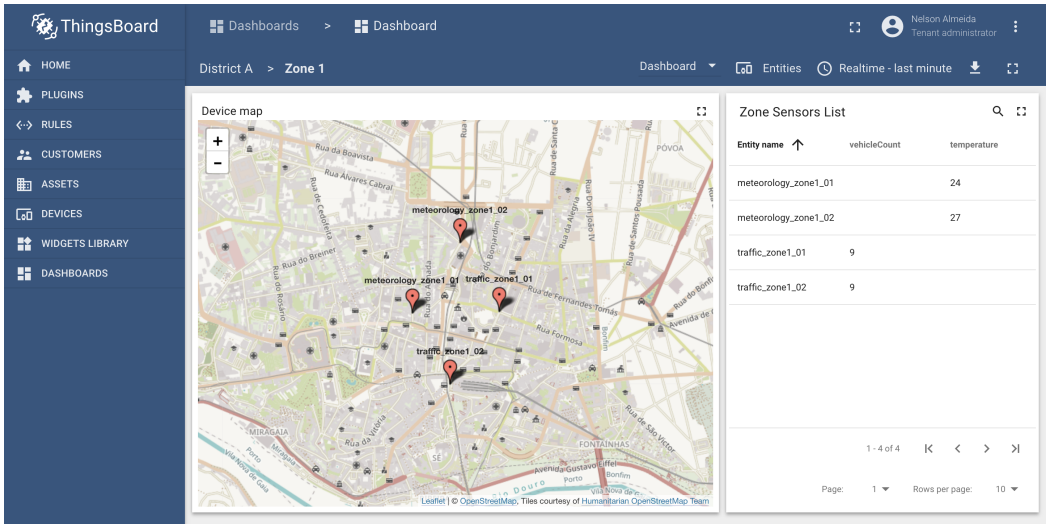

Figura 4.18: *Painel por zona do Thingsboard*

Na figura anterior é possível visualizar a localização dos dispositivos referentes à zona 1 através de um mapa e a respetiva informação de telemetria mais relevante disponibilizada por eles.

4.10 Conclusão

O capítulo descreveu o modo como os diferentes módulos do sistema foram implementados e quais os dados, tecnologias e ferramentas utilizadas.

Em relação aos dados utilizados, os de trânsito foram recolhidos de situações reais no âmbito de um outro projeto já realizado. Quanto aos de meteorologia foram simulados respeitando o formato dos de trânsito. Ambos tem uma caracterização simples que se traduziu numa mais valia para a perceção do correto funcionamento do protótipo.

Como ainda não existe uma normalização da caracterização da informação proveniente dos dispositivos, esta tem de ser tratada para ser coerente entre dispositivos de diferentes fabricantes. Na secção de tratamento de dados foi explicado isso mesmo e detalhada a normalização efetuada antes de enviar para a plataforma e para a base de dados.

Os dados de telemetria normalizados foram convertidos para o formato RDF para um armazenamento mais detalhado e para isso foi utilizada uma base de dados de triplos chamada Fuseki.

Em relação à ontologia, esta foi desenvolvida reaproveitando caracterizações presentes em ontologias relacionadas com IoT e Cidades Inteligentes pois apesar de não existir uma ontologia genérica, existem diferentes sobre o mesmo tema e como tal, o desenvolvimento de uma nova não se mostrou necessário.

Uma explicação detalhada da manipulação dos dados antes de realizar a inferência foi

realizada onde se pode verificar que os dispositivos foram divididos por zonas dentro da cidade e para inferência foram consideradas as médias, dentro de um intervalo de tempo, destes dispositivos por zona.
Em relação à manipulação do modelo OWL é descrito todo o processo de implementação programática deste modelo e sua utilização para posteriormente ser possível a tomada de decisões em função do cruzamento destas informações.
As regras desenvolvidas para que a tomada de decisão autónoma fosse possível foram também descritas. Pode-se comprovar que as regras são relativamente simples pois o intuito foi perceber e demonstrar que é possível e necessária a utilização destas para facilitar a tarefa de um administrador de uma cidade inteligente. O encadeamento de regras para tomada de decisões mais complexas também foi demonstrado e concluído que havendo mais informação de diferentes dispositivos, a criação de novas regras e sua inclusão é relativamente fácil e deste modo tomadas de decisão mais complexas podem ser implementadas.
Por fim foi demonstrada a utilização da plataforma escolhida (Thingsboard) para cobrir as funcionalidades de visualização de informação individual dos dispositivos, assim como da distribuição destes pela cidade e também da visualização de painéis informativos das decisões tomadas através dos módulos anteriores.

Capítulo 5

Testes e Avaliação

5.1 Introdução

Este capítulo destina-se a avaliar o sistema implementado, tendo em vista a eficácia do mesmo em satisfazer os casos de usos referidos. Cada caso de uso utiliza dados CSV como fonte, sendo que os dados de trânsito são reais, enquanto que os de meteorologia e de bateria são simulados respeitando o mesmo modelo de dados. Esta abordagem mostra também a utilização da plataforma Thingsboard como forma de criar um ambiente gráfico para as funcionalidades mais comuns já desenvolvidas na maioria das plataformas.
Todos os testes foram realizados num único equipamento que executava as implementações dos diferentes módulos.

Tabela 5.1: *Especificações da máquina de teste*

Parâmetro	Valor
Memória RAM	16 Gb
Disco	256 Gb SSD
Processador	2,9 GHz Intel Core i5
Sistema Operativo	macOS High Sierra

Na tabela anterior é possível ver as especificações do equipamento que serviu de base para a elaboração e testes do protótipo descrito.
Em seguida serão detalhados os resultados aos casos de uso estipulados previamente.

5.2 Caso de uso 1: Controlo de trânsito anómalo

O sistema alerta o utilizador quando uma situação de trânsito anómalo acontece. Para este caso de uso somente são utilizados dados de trânsito e uma distinção entre as horas de ponta e as restantes.

O objetivo é ter um sistema que armazene a informação, analise o último intervalo de tempo definido (neste caso os últimos 15 minutos de recolha de dados) e detete se a situação de trânsito nas respetivas zonas está normal ou fora do normal.
Os intervalos de hora de ponta estipulados foram entre as 07:59 e as 09:00 e entre as 17:59 e as 20:00. Durante este período qualquer situação de trânsito será considerada como normal e não disparará avisos ao utilizador. Caso se detete uma anomalia no trânsito e estejamos fora deste intervalo de tempo um aviso será mostrado ao utilizador através do painel desenvolvido na plataforma Thingsboard.
Como independentemente do fabricante dos dispositivos de trânsito toda a informação enviada deles está armazenada na base de dados de triplos respeitando a normalização desenvolvido, torna-se fácil a recolha e análise de todos os dispositivos de determinada zona. Para este caso de uso realiza-se uma recolha dos dados de presença de veículos, por zona, no último intervalo de tempo (15 minutos) e calcula-se a média, igualmente por zona, destes mesmo. O valor obtido é inferido e posteriormente testado sobre as regras SWRL desenvolvidas.
A regra SWRL desenvolvida para este caso de uso implica o acontecimento de uma média de presença de veículos no intervalo de tempo igual ou superior a 5 e o horário tem de estar fora da hora de ponta.
Como vemos na figura seguinte, é feita uma análise dos dados entre as 08:15 e as 08:30, ou seja, ambos os horários estão dentro do intervalo de tempo de hora de ponta como indica nas leituras da consola. São lidos e posteriormente calculada a média dos valores obtidos das leituras de presença de veículos, onde neste caso o valor final foi 7. Estes dados são inferidos para análise perante as regras SWRL desenvolvidas.
Em relação ao número de veículos estamos perante uma situação de trânsito anómala mas como estamos perante um intervalo de tempo dentro da hora de ponta essa situação é descartada e a informação que obtemos é “No Traffic News!”.
Neste caso, nada é enviado para o painel do Thingsboard.

```
START NEW beginningTime and endTime!!
ACTUAL beginning: 2014-08-01T08:15:00
ACTUAL end: 2014-08-01T08:30:00
START AVERAGE DEVICE VEHICLE COUNT!!
compareRushHourWithTimestamp - HORA DE PONTA
compareRushHourWithTimestamp - HORA DE PONTA
beginningTime e endTime DENTRO do horario de ponta!!!
averageDeviceVehicleCount - VehicleCount - x = 13
averageDeviceVehicleCount - VehicleCount - x = 6
averageDeviceVehicleCount - VehicleCount - x = 6
averageDeviceVehicleCount - VehicleCount - x = 6
MÉDIA TRAFEGO NO INTERVALOR E NA ZONA 1: 7
INDIVIDUAL traffic in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#traffic-test16
END AVERAGE DEVICE VEHICLE COUNT!!
START DETECT PROBLEMS!!
DetectProblems.verific() - No Traffic News!
DetectProblems.verific() - No Pollution News!
DetectProblems.verific() - No Battery News!
END DETECT PROBLEMS!!
```

Figura 5.1: *Consola de situação de trânsito normal*

Na figura seguinte podemos ver o caso em que acontece uma situação de trânsito anómala, ou seja, um intervalo de tempo fora da hora de ponta e uma média de veículos igual ou superior a 5.

```
START NEW beginningTime and endTime!!
ACTUAL beginning: 2014-08-01T09:00:00
ACTUAL end: 2014-08-01T09:15:00
START AVERAGE DEVICE VEHICLE COUNT!!
FORA do horario de ponta!!!
averageDeviceVehicleCount - VehicleCount - x = 8
averageDeviceVehicleCount - VehicleCount - x = 15
averageDeviceVehicleCount - VehicleCount - x = 4
averageDeviceVehicleCount - VehicleCount - x = 6
MÉDIA TRAFEGO NO INTERVALOR E NA ZONA 1: 8
INDIVIDUAL traffic in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#traffic-test19
END AVERAGE DEVICE VEHICLE COUNT!!
START DETECT PROBLEMS!!
checkNewAbnormalTraffic - ALERT ZONE - 1
checkNewAbnormalTraffic - ALERT endTime - 2014-08-01T09:15:00
New Inference - actualActivity: AbnormalTraffic
DetectProblems.verific() - New Abnormal Traffic!
DetectProblems.verific() - No Pollution News!
DetectProblems.verific() - No Battery News!
END DETECT PROBLEMS!!
```

Figura 5.2: *Consola de situação de trânsito fora do normal*

Neste caso recebemos um aviso de "New Abnormal Traffic!" e a informação é enviada para o painel do Thingsboard como podemos ver na figura seguinte.
No painel identifica-se o evento, o instante em que foi detetado, e a zona da cidade. Esta informação poderia ser desagregada permitindo ao administrador navegar para a zona em causa e inspeccionar os valores individuais dos sensores para identificar mais precisamente o ou os locais com trânsito lento. Poderia ser possível, por exemplo, o operador ativar câmaras que pudessem auxiliar mais rapidamente na identificação das causas do evento detetado.

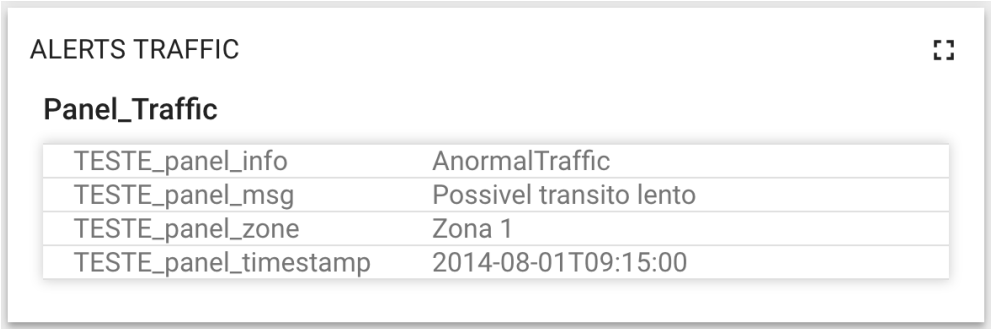

Figura 5.3: *trânsito anormal na dashboard do Thingsboard*

Na figura seguinte são apresentados os dados originais de trânsito utilizados nesta amostra.

traffic_zone1_01

status	avgSpeed	TIMESTAMP	vehicleCount	_id	REPORT_ID
OK	61	2014-08-01T09:05:00	8	20752484	traffic_zone1_01
OK	58	2014-08-01T09:10:00	4	20752933	traffic_zone1_01

traffic_zone1_02

status	avgSpeed	TIMESTAMP	vehicleCount	_id	REPORT_ID
OK	84	2014-08-01T09:05:00	15	20752506	traffic_zone1_02
OK	87	2014-08-01T09:10:00	6	20752955	traffic_zone1_02

Figura 5.4: *Dados de trânsito*

Como se pode comprovar, realizando a média destas quatro leituras de presença de veículos (8,4,15,6) obtemos o valor de 8 carros e o horário destas leituras está fora do horário de ponta estipulada. Deste modo comprova-se que o protótipo comportou-se como previsto, avisando o administrador da existência de uma situação de trânsito fora do normal.

5.3 Caso de uso 2: Controlo de possível ambiente poluído

O sistema alerta o utilizador quando uma situação de possível poluição acontece. Para este caso de uso são utilizados dados de trânsito e de meteorologia.
O objetivo é ter um sistema que armazene a informação e analise o último intervalo de tempo definido (neste caso os últimos 15 minutos de recolha de dados) e perceba se existe uma situação de trânsito elevado e no mesmo intervalo de tempo uma situação de elevada temperatura dentro da mesma zona.
Como independentemente do fabricante dos dispositivos de trânsito e de meteorologia toda a informação enviada deles está armazenada na base de dados de triplos respeitando a normalização desenvolvido, torna-se fácil a recolha e análise de todos os dispositivos de determinada zona. Para este caso de uso realiza-se uma recolha dos dados de presença de veículos, por zona, no último intervalo de tempo (15 minutos) e calcula-se a média, igualmente por zona. Recolhe-se também os dados dos dispositivos de meteorologia da mesma zona e efetua-se igualmente a sua média. Os valores obtidos são inferidos e posteriormente testados sobre as regras SWRL desenvolvidas.
A regra SWRL desenvolvida para este caso de uso implica o acontecimento de uma média de presença de veículos no intervalo de tempo igual ou superior a 5 e uma média de temperatura igual ou superior a 20 graus, todo isto dentro da mesma zona.
Como vemos na figura seguinte, é feita uma análise dos dados entre as 09:45 e as 10:00 onde são obtidos os valores de temperatura e de contagem de veículos dentro desse intervalo e zona, calculada a sua média e inferida para análise pelas regras SWRL.
Como estamos perante um valor médio de temperatura superior a 20 e um valor médio

de contagem de veículos superior a 5, na mesma zona, acontece uma situação de “New Abnormal Pollution!”.

```
START NEW beginningTime and endTime!!
ACTUAL beginning: 2014-08-01T09:45:00
ACTUAL end: 2014-08-01T10:00:00
START AVERAGE DEVICE TEMPERATURE!!
averageDeviceTemperature - Temperature - x = 22
averageDeviceTemperature - Temperature - x = 20
averageDeviceTemperature - Temperature - x = 20
averageDeviceTemperature - Temperature - x = 23
MÉDIA METEOROLOGIA NO INTERVALOR E NA ZONA 1: 21
INDIVIDUAL meteorology in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#meteorology-test14
END AVERAGE DEVICE TEMPERATURE!!
START AVERAGE DEVICE VEHICLE COUNT!!
FORA do horario de ponta!!!
averageDeviceVehicleCount - VehicleCount - x = 9
averageDeviceVehicleCount - VehicleCount - x = 8
averageDeviceVehicleCount - VehicleCount - x = 4
averageDeviceVehicleCount - VehicleCount - x = 6
MÉDIA TRAFEGO NO INTERVALOR E NA ZONA 1: 6
INDIVIDUAL traffic in zone 1 = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#traffic-test22
END AVERAGE DEVICE VEHICLE COUNT!!
START DETECT PROBLEMS!!
checkNewAbnormalTraffic - ALERT ZONE - 1
checkNewAbnormalTraffic - ALERT endTime - 2014-08-01T10:00:00
New Inference - actualActivity: AbnormalTraffic
DetectProblems.verific() - New Abnormal Traffic!
checkNewAbnormalPollution - ALERT ZONE - 1
checkNewAbnormalPollution - ALERT endTime - 2014-08-01T10:00:00
New Inference - actualActivity: AbnormalPollution
DetectProblems.verific() - New Abnormal Pollution!
DetectProblems.verific() - No Battery News!
END DETECT PROBLEMS!!
```

Figura 5.5: *Consola de possivel situação de poluição fora do normal*

Tal como no alerta de trânsito lento, este alerta de poluição deveria levar o administrador a procurar as causas, quer acedendo a informação mais individualizada sobre os sensores quer recorrendo a outra telemetria. Eventualmente o operador poderia considerar o alerta como inevitável, por alguma combinação de fatores fora de controlo, e ignorar o alerta. Seria possível também aceder ao histórico de alertas e tentar contextualizar a informação, detetando eventualmente situações recorrentes a necessitar de alguma intervenção.

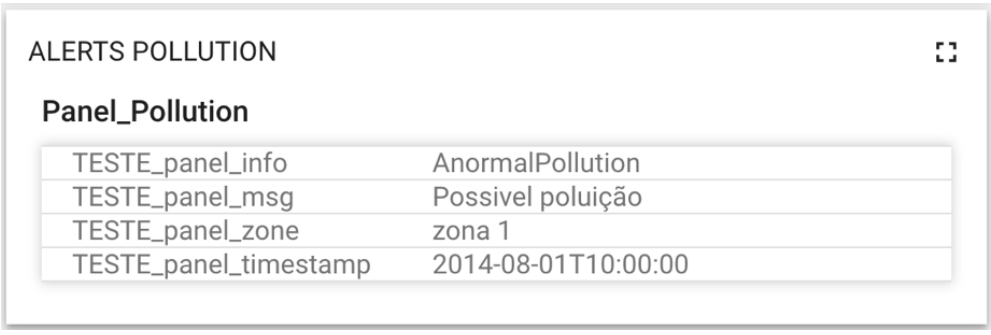

Figura 5.6: *Poluição anómala no painel do Thingsboard*

Nas duas figuras seguintes são apresentados os dados originais de trânsito e meteorologia utilizados para detetar estas situações de possível poluição.

traffic_zone1_01

status	avgSpeed	TIMESTAMP	vehicleCount	_id	REPORT_ID
OK	59	2014-08-01T09:50:00	8	20756525	traffic_zone1_01
OK	57	2014-08-01T09:55:00	6	20756974	traffic_zone1_01

traffic_zone1_02

status	avgSpeed	TIMESTAMP	vehicleCount	_id	REPORT_ID
OK	79	2014-08-01T09:50:00	9	20756547	traffic_zone1_02
OK	79	2014-08-01T09:55:00	4	20756996	traffic_zone1_02

Figura 5.7: *Dados de trânsito*

Como se pode comprovar, realizando a média destas quatro leituras de presença de veículos (8,6,9,4) obtemos o valor de 6 carros, o que origina uma situação de trânsito elevado.

meteorology_zone1_01

status	temperature	humidity	available_battery	TIMESTAMP	_id	REPORT_ID
OK	20	61	33	2014-08-01T09:50:00	20756782	meteorology_zone1_01
OK	20	61	32	2014-08-01T09:55:00	20757231	meteorology_zone1_01

meteorology_zone1_02

status	temperature	humidity	available_battery	TIMESTAMP	_id	REPORT_ID
OK	22	50	10	2014-08-01T09:50:00	20756700	meteorology_zone1_02
OK	23	51	11	2014-08-01T09:55:00	20757149	meteorology_zone1_02

Figura 5.8: *Dados de meteorologia*

Aliado ao resultado de trânsito, como no mesmo intervalo de tempo acontece uma situação de temperatura superior a 20, obtida através da média das leituras dos valores de temperatura (20, 20 ,22 ,23), comprova-se o correto funcionamento da plataforma ao avisar o administrador de uma possível situação de poluição na zona.

5.4 Caso de uso 3: Controlo de possível necessidade de carregamento de dispositivo

O sistema alerta o utilizador quando existe a necessidade de carregar um dispositivo manualmente. Para este caso de uso são utilizados dados de meteorologia, pois como estes foram os dados manipulados, foi acrescentado o valor de percentagem de bateria disponível na telemetria enviados pelo dispositivo.
Tal como os casos de uso anteriores a análise dos dados é feita por intervalo de tempo definido de 15 em 15 minutos. Neste intervalo de tempo é analisada a informação de

percentagem de todos os dispositivos e caso algum esteja a baixo de 20% é inferido para que seja possível analisar pelas regras SWRL.
A regra SWRL desenvolvida para este caso de uso implica o acontecimento de uma percentagem inferior a 20% e que o dispositivo seja carregado de modo "MANUAL" tal como explicado no capítulo anterior.
Como vemos na figura seguinte, é feita uma análise dos dados entre as 11:15 e as 11:30 onde são obtidos os valores de percentagem dos dispositivos que enviaram informação dentro desse intervalo.
No caso demonstrado o dispositivo tem uma percentagem de bateria de 15% e o modo de carregamento (seas_isPoweredBy) é "MANUAL" logo depois de ser inferido estes dados, é detetada uma situação anómala de bateria o que indica que deve ser carregado o dispositivo.

```
START NEW beginningTime and endTime!!
ACTUAL beginning: 2014-08-01T11:15:00
ACTUAL end: 2014-08-01T11:30:00
START DETECT BATTERY PROBLEM!!
detectBatteryProblem - battery - x = 15
BATTERY INFO: % battery: 15
BATTERY INFO: seas_electicPower: battery
BATTERY INFO: seas_isPoweredBy: MANUAL
BATTERY INFO: vcard_Region: 1
baterrvInd ZONA 1: battervInd = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#batterv-22
END DETECT BATTERY PROBLEM!!
START DETECT PROBLEMS!!
DetectProblems.verific() - No Traffic News!
DetectProblems.verific() - No Pollution News!
checkNewAbnormalBattery - ALERT ZONE - 1
checkNewAbnormalBattery - ALERT endTime - 2014-08-01T11:30:00
New Inference - actualActivity: AbnormalBattery
DetectProblems.verific() - New Abnormal Battery!
END DETECT PROBLEMS!!
```

Figura 5.9: *Consola de situação de bateria fora do normal*

Ocorrendo esta situação a informação é enviada para o painel do Thingsboard como podemos ver na figura seguinte.

Figura 5.10: *Bateria baixa no painel do Thingsboard*

Na figura seguinte vemos o caso em que a bateria esta inferior a 20% mas o modo de carregamento é "AUTO" e neste caso não é necessário avisar o utilizador.

```
detectBatteryProblem - battery - x = 18
BATTERY INFO: % battery: 18
BATTERY INFO: seas_electicPower: battery
BATTERY INFO: seas_isPoweredBy: AUTO
BATTERY INFO: vcard_Region: 1
baterryInd ZONA 1: batteryInd = http://www.semanticweb.org/nelsonalmeida/ontologies/simple#battery-5
```

Figura 5.11: *Consola de situação de bateria normal*

Realizados os três casos de uso previstos, e analisando os seus resultados, algumas conclusões foram tiradas e são descritas a seguir.

5.5 Conclusão

Com estes três casos de uso pretende-se comprovar que a necessidade de uma camada de alto nível é grande para que um utilizador não necessite de analisar individualmente os dispositivos presentes numa cidade inteligente.
Os casos de uso demonstrados não têm uma grande complexidade, pois o objetivo dos mesmos era simplesmente demonstrar a ideia de que é possível e viável a recolha de dados, tratamento e tomada de decisão autónoma.
Com uma análise mais aprofundada das necessidades de tomada de decisão numa cidade, casos de uso mais complexos podem ser implementados e assim comprovar mais intensivamente a funcionalidade deste tipo de plataforma.
Em relação às regras, estas não foram muito complexas, devido ao conjunto de dados utilizado não ser complexo e para que os resultados fossem fáceis de analisar para verificar se a plataforma tinha o comportamento desejado.
Visto que a plataforma corresponde ao esperado, novas regras SWRL podem ser adicionadas, bastando para isso colocá-las junto das restantes. As atuais também podem facilmente ser alteradas sem que sejam necessárias alterações profundas no código da plataforma.
A utilização de mais dispositivos de diferentes tipos abre o leque de possibilidades de criação de regras possibilitando assim o desenvolvimento de regras mais complexas.
Através de um sistema deste género é possível automatizar uma cidade inteligente de modo a gerar avisos de alto nível depois de cruzar dados e comparar com regras definidas.
A criação de regras mais complexas e a adição de um maior número de tipos de dispositivos tornará uma aplicação deste género uma ferramenta muito útil na administração de IoTs em larga escala numa cidade inteligente.

Capítulo 6

Conclusão

6.1 Introdução

Os objetivos definidos para este trabalho foram:

- Utilizar tecnologias da web semântica para representar e analisar as informações recebidas dos dispositivos.
- Definir e testar regras para despoletar alertas de apoio à gestão de cidades inteligentes.

Relativamente ao primeiro objetivo foram adicionados, através de uma plataforma existente, dispositivos a um protótipo de cidade inteligente. Estes dispositivos foram devidamente caracterizados seguindo uma ontologia definida. A informação de telemetria disponibilizada pelos dispositivos foi normalizada seguindo igualmente a ontologia definida e foi armazenada numa base de dados de triplos.
Deste modo tornou-se possível o cruzamento de informação de diferentes dispositivos para uma análise semântica.
Em relação ao segundo objetivo foram desenvolvidas regras e através do cruzamento do resultado destas foi possível a inferência de acontecimentos de alto nível.
Ao inferir acontecimentos de alto nível foi possível disponibilizar essa informação em painéis, através da plataforma utilizada, para que um administrador da cidade inteligente tenha informação mais útil para a gestão da mesma.

Em relação aos objetivos técnicos definidos, ou seja:

- Escolher uma plataforma disponível para assegurar as funcionalidades básicas de gestão dos dispositivos.
- Definir a normalização dos dados recebidos dos dispositivos.
- Enriquecer a caracterização dos dispositivos usando ontologias.

Relativamente ao primeiro objetivo a plataforma utilizada foi a Thingsboard e serviu para cobrir as funcionalidades presentes na maioria das plataformas já desenvolvidas, tais como inserção e caracterização de dispositivos, visualização da telemetria e atributos dos mesmos, assim como localização destes num mapa e visualização, em painéis, de avisos de alto nível provenientes da análise semântica dos dados dos dispositivos.
Relativamente ao segundo objetivo foi desenvolvida uma normalização para caracterização de dispositivos de meteorologia e de trânsito. Esta normalização foi simples cobrindo basicamente os dados disponibilizados pelos dispositivos e as informações relevantes para a demonstração deste projeto.
Relativamente ao terceiro objetivo os dispositivos foram caracterizados com informação relevante para uma análise semântica mais completa. Informações de localização, tipo de dispositivo, tipo de alimentação e carregamento são exemplo de atributos que caracterizam todos os dispositivos deste protótipo.

No geral, demonstrou-se que as plataformas direcionadas à gestão de cidades inteligentes podem beneficiar de uma camada de análise semântica, permitindo facilitar e melhorar a sua gestão através de avisos de alto nível.

6.2 Trabalho futuro

Como a IoT está em crescimento nas cidades com o objetivo de as tornar inteligentes diversos desafios vão surgindo no desenrolar dos projetos com este propósito.
Para que seja possível uma análise e tomada de decisão autónoma ficou comprovado que existe uma necessidade de normalizar a informação disponibilizada pelos diferentes tipos de dispositivos dos diferentes fabricantes existentes.
Quanto a isto, foi realizada uma recolha de padrões e ontologias de IoT para desenvolver uma normalização modelo para os casos de uso estipulados. No entanto, é necessário desenvolver uma ontologia genérica para caracterização de atributos e telemetria dos dispositivos. Um estudo mais aprofundado sobre as necessidades de informação que as cidades têm daria para desenvolver uma ontologia deste género para ser utilizada pelos diferentes programadores de dispositivos IoT.
Tendo uma normalização da caracterização dos dispositivos por tipo, uma interface de inserção personalizada de modo a orientar os administradores a inserir a informação necessária seria uma mais valia. Deste modo, um administrador, no momento de inserir novos dispositivos seria orientado, através de campos de preenchimento sobre a informação que teria de inserir tendo em conta o tipo de dispositivo.
Para as tomadas de decisão foram desenvolvidas regras simples para comprovar o correto funcionamento do protótipo, no entanto, com a utilização de mais tipos de dispositivos será possível desenvolver mais regras e fazer o cruzamento destas com as já existentes

com o objetivo de tomar decisões mais complexas e que sejam benéficas para facilitar o trabalho de um administrador de um conjunto de dispositivos tão extenso como o de uma cidade inteligente.

Através de inteligência artificial é possível desenvolver um modulo de aprendizagem para que os parâmetros pré-definidos, como por exemplo os valores limites de temperatura e número de viaturas, sejam calculados mediante determinadas condições. Este mesmo modulo pode ser desenvolvido também com o objetivo de em vez de termos horas de ponta como parâmetro pré-definidos, este seja capaz de determinar locais com congestionamento de trânsito normal em determinadas horas do dia.

Através de um modulo de aprendizagem como o descrito anteriormente, as regras SWRL podem ser melhoradas para receberem os valores das decisões tomadas.

Referências

AIOTI. The Alliance for Internet of Things Innovation, 2018. URL https://aioti.eu/. Last accessed on 2018-09-11.

James Frederick Allen. Maintaining knowledge about temporal intervals. communications of ACM, Vol. 26, No. 11, 832–843. 26(11):832–843. 1983.

V. Barchetti, M. Senigalliesi, O. Vermesan, R. Bahr, T. Macchia, A. Gluhak, S. Hubert, S. Vallet Chevillard, and F. Clari. Analysis on IoT Platforms Adoption Activities. Technical report, European Platforms Initiative, 2017.

Pierfrancesco Bellini, Monica Benigni, Riccardo Billero, Paolo Nesi, and Nadia Rauch. Km4City ontology building vs data harvesting and cleaning for smart-city services. pages 827–839, 2014. URL http://dx.doi.org/10.1016/j.jvlc.2014.10.023.

Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer, and Ernoe Kovacs. Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander. pages 1–8, 2015.

Hoan Suk Choi and Woo Seop Rhee. Iot-based user-driven service modeling environment for a smart space management system. 2014.

Citibrain. Smart solutions for global challenges, 2018. URL http://www.citibrain.com/pt/. Last accessed on 2018-09-11.

CityPluse. CityPulse - Dataset Collection, 2018. URL http://iot.ee.surrey.ac.uk:8080/datasets.html. Last accessed on 2018-09-12.

Andrew Collinge. What is Sharing Cities? 2018.

Martin Curley. Exploitation of Cyber Physical Systems and Ambient Intelligence: An Intel Business Perspective. *AcaTech Integrated Research Agenda Cyber Physical Systems, Berlin*, pages 1–29, 2012.

Pratikkumar Desai. A Semantic Situation Awareness Framework for Indoor Cyber-Physical Systems. page 196, 2013.

Cristiano Diedrich. Docker, 2015. URL https://www.mundodocker.com.br/o-que-e-docker/. Last accessed on 2018-09-12.

Docker. Docker, 2018. URL https://www.docker.com/. Last accessed on 2018-09-12.

Eclipse. Eclipse Kura, 2018. URL https://www.eclipse.org/kura/. Last accessed on 2018-09-11.

ESPRESSO. The ESPRESSO Project, 2018. URL http://espresso.espresso-project.eu/project-2/. Last accessed on 2018-09-11.

ETSI. Internet of things, 2018. URL https://www.etsi.org/. Last accessed on 2018-09-11.

FIESTA-IoT. Federated Interoperable Semantic IoT Testbeds and Applications, 2018. URL http://fiesta-iot.eu/. Last accessed on 2018-09-11.

Alex Gluhak and Alex Vermesan. D03.01 Report on IoT platforms activities. page 103, 2016.

Group Km4City. KM4CIty From data to services for sentient cities. 2018.

Christian Groves, Lui Yan, and Yang Weiwei. Overview of IoT semantics landscape. pages 1–42, 2016.

Amelie Gyrard and Martin Serrano. FIESTA-IoT : Federated Interoperable Semantic Internet of Things (IoT) Testbeds and Applications. pages 1–2, 2018.

Tim Hinchliffe. Porto is Holding an Open Call for EU's SynchroniCity IoT Project, 3M in Financing, 2018. URL https://portugalstartups.com/2018/06/porto-synchronicity-iot/. Last accessed on 2018-09-11.

Matthew Horridge. A Practical Guide To Building OWL Ontologies Using Protégè 4 and CO-ODE Tools Edition 1.3. pages 0–107, 2011.

IERC. European Research Cluster on the Internet of Things, 2016. URL http://www.internet-of-things-research.eu/. Last accessed on 2018-09-11.

InterIoT. InterIoT Project, 2016. URL http://www.inter-iot-project.eu/. Last accessed on 2018-09-11.

Prem Prakash Jayaraman, Jean Paul Calbimonte, and Hoan Nguyen Mau Quoc. The schema editor of OpenIoT for semantic sensor networks. pages 1–6, 2015.

Corinne Jennings. Internet of Things Applications. pages 1–42, 2015.

Kaa. Kaa Project, 2018. URL https://www.kaaproject.org/. Last accessed on 2018-09-11.

Km4City. Open urban platform for a sentient smart city, 2018. URL http://km4city.org/. Last accessed on 2018-09-11.

Krebs On Security. Study: Attack on KrebsOnSecurity Cost IoT Device Owners 323K, 2018. URL https://krebsonsecurity.com/2018/05/study-attack-on-krebsonsecurity-cost-iot-device-owners-323k/. Last accessed on 2018-09-25.

Altice Labs. Sharing Cities. 2018.

Jing Lin. Agent-based analysis and mitigation of failure for cyber-physical systems. 3510897:145, 2011.

Jing Lin, Sahra Sedigh, and Ann Miller. Modeling Cyber-Physical Systems With Semantic Agents. pages 1–6, 2010.

Linked Open Vocabularies. Linked Open Vocabularies, 2018. URL https://lov.linkeddata.es/dataset/lov. Last accessed on 2018-10-09.

Lisboa E-Nova. Sharing Cities, 2018. URL http://lisboaenova.org/wp/sharing-cities/. Last accessed on 2018-09-11.

Love4IoT. Linked Open Vocabularies for Internet of Things, 2018. URL http://lov4iot.appspot.com/. Last accessed on 2018-09-12.

M2MLabs. The M2MLabs Mainspring Project, 2015. URL http://www.m2mlabs.com/. Last accessed on 2018-09-11.

Altti Ilari Maarala and Jukka Riekki. Semantic Reasoning for Context-Aware Internet of Things Applications. *IEEE Internet of Things Journal*, 2017.

Mark A. Musen. The Protégé Project: A Look Back and a Look Forward, 2016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4883684/. Last accessed on 2018-09-12.

oneM2M. ONEM2M TECHNICAL REPORT. pages 0–42, 2016a.

oneM2M. ONEM2M TECHNICAL SPECIFICATION. pages 0–190, 2016b.

oneM2M. oneM2M, 2018. URL http://www.onem2m.org/. Last accessed on 2018-09-20.

Community Openhab. Openhab, 2018. URL https://www.openhab.org/. Last accessed on 2018-09-11.

Porto Digital. Open Call SynchroniCity, 2018. URL https://www.portodigital.pt/. Last accessed on 2018-09-25.

Lukas Reinfurt, Uwe Breitenbücher, Michael Falkenthal, Frank Leymann, and Andreas Riegg. Internet of things patterns. pages 1–10, 2016. URL http://dl.acm.org/citation.cfm?doid=3011784.3011789.

Sapo24. Citibrain, 2018. URL https://24.sapo.pt/tecnologia/artigos/este-projeto-de-aveiro-quer-ser-o-cerebro-da-cidade-sensor-a-sensor. Last accessed on 2018-09-11.

SapoTek. Portugal Smart Cities Summit 2018 , 2018. URL https://tek.sapo.pt/expert/artigos/futuro-das-smart-cities-volta-a-ser-discutido-na-antiga-fil. Last accessed on 2018-09-11.

Bruce Schneier. Schneier on Security, 2017. URL https://www.schneier.com/blog/archives/2017/02/security_and_pr.html. Last accessed on 2018-10-11.

Smart Santander. Santander On Fire, 2018. URL http://www.smartsantander.eu/. Last accessed on 2018-09-25.

Synchronicity. Synchronicity, 2018. URL https://synchronicity-iot.eu/project/porto/. Last accessed on 2018-09-11.

Thingsboard. Thingsboard, 2018. URL https://thingsboard.io/. Last accessed on 2018-09-11.

Peter Veres. Proceedings of facility management days. In SAFM, editor, *IoT Solutions for (smart) buildings*, pages 1–30, 2017.

W3C. Semantic Web, 2015. URL https://www.w3.org/standards/semanticweb/. Last accessed on 2018-09-28.

Consortium Wise-IoT. Worldwide Interoperability for Semantics IoT, 2015. URL http://wise-iot.eu/en/home/. Last accessed on 2018-09-11.