Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemática Aplicada
Programa de Pós-Graduação em Sistemas e Computação
Mestrado Acadêmico em Sistemas e Computação

Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes

Cícero Alves da Silva

Natal-RN
Agosto/2015

Cícero Alves da Silva

Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte como requisito para obtenção do grau de Mestre em Sistemas e Computação.

Linha de pesquisa:
Engenharia de Software

Orientador

Prof. Dr. Gibeon Soares de Aquino Jr.

PPgSC – Programa de Pós-Graduação em Sistemas e Computação
DIMAp – Departamento de Informática e Matemática Aplicada
CCET – Centro de Ciências Exatas e da Terra
UFRN – Universidade Federal do Rio Grande do Norte

Natal-RN

Agosto/2015

Catalogação da Publicação na Fonte. UFRN / SISBI / Biblioteca Setorial
Centro de Ciências Exatas e da Terra – CCET.

Silva, Cícero Alves da.
Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes / Cícero Alves da Silva. - Natal, 2015.
115 f. : il.

Orientador: Prof. Dr. Gibeon Soares de Aquino Junior.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Norte. Centro de Ciências Exatas e da Terra. Programa de Pós-Graduação em Sistemas e Computação.

1. Arquitetura de software – Dissertação. 2. Cidades inteligentes – Dissertação. 3. Extensibilidade – Dissertação. I. Aquino Junior, Gibeon Soares de. II. Título.

RN/UF/BSE-CCET CDU: 004.273

CÍCERO ALVES DA SILVA

Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes

Esta Dissertação foi julgada adequada para a obtenção do título de mestre em Sistemas e Computação e aprovado em sua forma final pelo Programa de Pós-Graduação em Sistemas e Computação do Departamento de Informática e Matemática Aplicada da Universidade Federal do Rio Grande do Norte.

Dr. Gibeon Soares de Aquino Junior – UFRN
(Presidente)

Dr. Uirá Kulesza – UFRN
(Coordenador do Programa)

Banca Examinadora

Dr. Uirá Kulesza – UFRN
(Examinador)

Dr. Kiev Santos da Gama – UFPE
(Examinador)

Dedico este trabalho aos meus pais, Cosme e Ana Lúcia, que são os principais professores que eu tive em minha vida.

Agradecimentos

A Deus, em quem tanto depositei minha fé nos momentos de problemas e aflições enfrentados durante toda minha vida.

A Renovação Carismática Católica grande responsável pela minha formação espiritual, social e familiar.

Aos meus pais, Cosme Fernandes da Silva e Ana Lúcia Alves da Silva, por todos os ensinamentos e por terem lutado para que eu tivesse uma boa educação e me tornasse uma pessoa de bem.

À minha irmã, Carmem Lúcia Alves da Silva, que tanto me apoiou e torceu para que eu obtivesse êxito em todos os projetos que empreendi em minha vida.

Ao meu orientador, Gibeon Soares de Aquino Jr., em quem procuro me espelhar para no futuro me tornar um bom professor. Além disso, o agradeço por ter acreditado em mim desde o início e por ter direcionado o meu estudo apresentando ideias, críticas e sugestões essenciais ao desenvolvimento da pesquisa. Por fim, por sua amizade e pelos conselhos.

Aos meus companheiros de "time" de mestrado, Itamir, Mário, Jean e Héldon, pelas experiências divididas e pela amizade.

Aos meus melhores amigos, Bruno, Danilo, Fábio Gomes, Fábio Dantas, José Alves, José Geraldo, Luiz, Pedro e Yuri, que sempre estiveram ao meu lado quando em muitas ocasiões de minha vida precisei de uma palavra amiga de conforto, apoio e estímulo.

Ao meu orientador de TCC, Flavius da Luz e Gorgônio, que me incentivou e tanto insistiu para que eu seguisse o caminho acadêmico e ingressasse no mestrado.

Aos ex-colegas e eternos amigos do curso de Sistemas de Informação, Alisson, Aragonense, Fladson, Gutto, Isaac Danilo, João Paulo, Lucas e Pablo, pelo companheirismo durante todo o período em que estudamos juntos.

Aos professores, Dr. Kiev Santos da Gama e Dr. Uira Kulesza, pelas significativas contribuições dadas ao meu trabalho.

A vida é uma peça de teatro que não permite ensaios. Por isso, cante, chore, dance, ria e viva intensamente, antes que a cortina se feche e a peça termine sem aplausos.

Charles Chaplin

Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes

Autor: Cícero Alves da Silva
Orientador(a): Prof. Dr. Gibeon Soares de Aquino Jr.

Resumo

O ambiente urbano possui uma grande quantidade de dispositivos de tecnologias variadas, o que torna difícil a integração dos dados originados nos mesmos devido sua heterogeneidade. Porém, é importante que esses dados sejam gerenciados de maneira integrada possibilitando a troca de informações entre os domínios existentes e auxiliando na tomada de decisão. Ademais, não há como prever a forma que esses dados precisarão ser transformados visto que cada aplicação pode necessitar que os mesmos sejam disponibilizados obedecendo processamentos específicos. Assim, este trabalho descreve o projeto e implementação de uma plataforma cujo objetivo é a integração, transformação e disponibilização de fluxos de dados provenientes de fontes heterogêneas no âmbito de cidades. Além disso, ela define um fluxo de transformação extensível facilitando a criação de novos processamentos para um dado existente e a inclusão de um novo tipo de dado, o que permite a adequação dos passos definidos para que eles sejam implementados conforme as características e restrições de cada tipo integrado. Por fim, foi realizado um estudo de caso que utilizou o cenário de um estacionamento e avaliou alguns aspectos importantes relacionados à implementação da plataforma.

Palavras-chave: Cidades inteligentes, Arquitetura de *software*, Extensibilidade.

An extensible platform for transformation heterogeneous data flow in smart cities

Author: Cícero Alves da Silva
Supervisor: Prof. Dr. Gibeon Soares de Aquino Jr.

Abstract

The urban environment has a lot of devices that use different technologies, which makes the integration of data generated in them a difficult process due to their heterogeneity. However, it is important to manage these data in an integrated way, enabling the exchange of information between existing fields and assisting in the decision-making process. Moreover, there's no way to tell how this data will need to be processed since each application may require it to be available obeying specific processes. Thus, this work describes the design and implementation of a platform that aims to integrate, process and make available data streams from heterogeneous sources in the context of cities. In addition, it defines an extensible data processing flow, which makes the creation of new processes for existing data and the inclusion of new types of data easier, allowing the adequation of the defined steps for them to be implemented according to the characteristics and restrictions of each integrated type. Finally, a case study was conducted, which used a parking lot as scenario and assessed some important aspects related to platform implementation.

Keywords: Smart cities, Software architecture, Extensibility.

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ONU – Organização das Nações Unidas

IBGE – Instituto Brasileiro de Geografia e Estatística

TIC – Tecnologias da Informação e Computação

GPS – *Global Positioning System*

CI – Cidade(s) Inteligente(s)

IBD – *Songdo International Business District*

COR – Centro de Operações Rio

QR Code – *Quick Response Code*

SMARTY – *SMARt Transport for sustainable citY*

CPTW – *Cyber Parallel Traffic World*

IoT – *Internet of Things*

OSGi – *Open Services Gateway Initiative*

SOA – *Service Oriented Architecture*

SMS – *Short Message Service*

MAC – Módulo de Acesso e Comunicação

MPD – Módulo de Persistência de Dados

MGR – Módulo de Gerenciamento de Recursos

MGDD – Módulo de Gerenciamento e Distribuição de Dados

ESB – *Enterprise Service Bus*

M2M – *Machine-to-Machine*

Triple DES – Triple Data Encryption Standard

REST – *Representational State Transfer*

CPU – *Central Processing Unit*

GB – *Gigabyte*

RAM – *Random Access Memory*

MB – *Megabyte*

Lista de listagens

Sumário

1 Introdução

Este capítulo tem a finalidade de situar o tema abordado no trabalho. Assim, descreve-se a motivação para a realização do mesmo na Seção 1.1. Na Seção 1.2 é descrito o problema tratado no estudo. Por sua vez, a Seção 1.3 descreve o objetivo geral da pesquisa e enumera os objetivos específicos dela. Em seguida, na Seção 1.4 é explicada a metodologia seguida no trabalho. Por fim, a Seção 1.5 apresenta a forma como o mesmo está organizado.

1.1 Motivação

Em relatório divulgado pelas Nações Unidas (NATIONS, 2014), é mostrado que em 1950 a população do planeta estava em sua maioria concentrada no campo, naquele ano 70% das pessoas viviam na zona rural. No entanto, esse percentual foi diminuindo ao longo do tempo devido em parte ao fenômeno conhecido como êxodo rural, o qual se caracteriza pelo abandono do campo por parte de seus moradores que migram para as cidades em busca de oportunidades de trabalhos e melhores condições de vida.

Assim, no decorrer dos anos a população da Terra foi se concentrando cada vez mais na zona urbana e em 2008 pela primeira vez a maior parte das pessoas do mundo passa a viver nos centros urbanos (CAMPOS, 2012). No gráfico exibido na Figura 1 pode-se ver essa mudança na configuração da distribuição populacional do planeta a partir de 2008. Além disso, a projeção criada nesse estudo mostra que a população urbana crescerá ainda mais até o ano de 2050.

Outro fator responsável pelo aumento da quantidade de pessoas nas cidades é o crescimento demográfico no planeta. De acordo com o Departamento de Economia e Assuntos Sociais da ONU (NATIONS, 2012), em 1950 a população do mundo era um pouco superior a 2,5 bilhões e em 2015 a quantidade de pessoas na Terra ultrapassa os 7 bilhões. Com esses números pode-se perceber um crescimento de aproximadamente 180% na população mundial em um intervalo de apenas 65 anos, o que comprova que o número de pessoas no

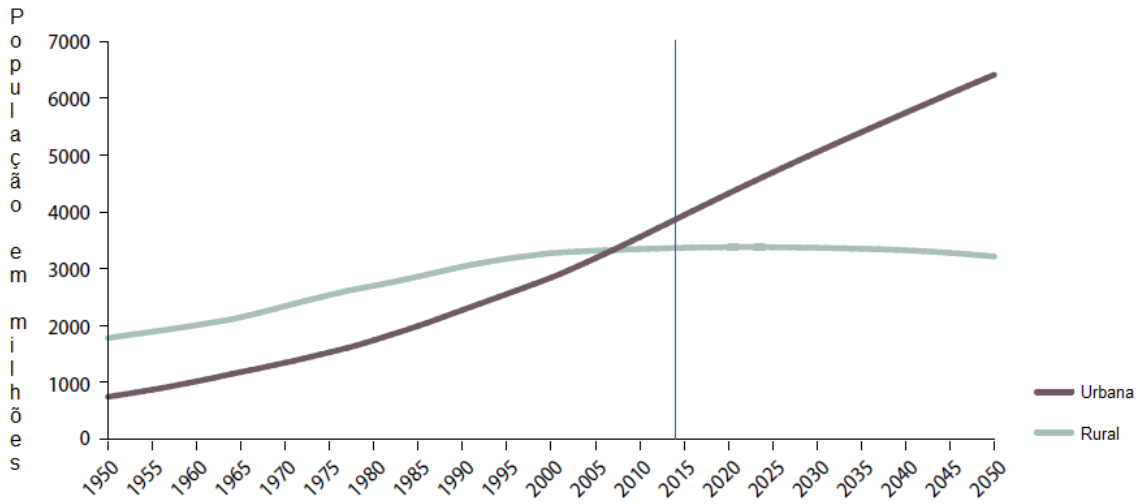


Figura 1: População urbana e rural do mundo de 1950 a 2050, adaptado de(NATIONS, 2014)

mundo cresce em um ritmo bastante acelerado.

O Brasil também acompanhou essas transformações ocorridas na distribuição e crescimento da população. Segundo dados do IBGE, em 1950 o Brasil tinha um pouco mais de 51 milhões de habitantes (IBGE, 2010) e hoje já passa dos 203 milhões (IBGE, 2015), indicando um aumento populacional de quase 300%. Ademais, segundo o mesmo órgão 84,4% dos habitantes do país residiam na área urbana em 2010 (IBGE, 2010) .

Entretanto, mesmo concentrando a maior parte do população da Terra as cidades ocupam apenas 2% da área do planeta (TOPPETA, 2010), o que é um espaço muito pequeno frente a enorme quantidade de habitantes delas. Além disso, a distribuição dessas pessoas nas cidades não ocorre de maneira uniforme, pois poucas delas concentram um número muito grande de cidadãos. Segundo Dobbs et al. (2011), 1,5 bilhão de pessoas viviam em apenas 600 cidades no ano de 2011, representando 22% da população global. E, além do mais, o mesmo estudo citado anteriormente estima que 2 bilhões de pessoas viverão nessas mesmas 600 cidades em 2025, o que representará 25% da população do mundo.

Contudo, as cidades não estavam preparadas para receber esse enorme contingente de pessoas e isso resultou no sobrecarregamento dos serviços oferecidos nesses locais, pois os mesmos não acompanharam a grande demanda ocasionada por essa mudança na configuração da população global.

Em consequência disso, muitos trabalhos envolvendo Tecnologias da Informação e Computação (TIC) vêm sendo desenvolvidos na intenção de melhorar os serviços oferecidos nas cidades e, consequentemente, a condição de vida de seus habitantes. Esses projetos procuram com uso das tecnologias incrementar inteligência aos serviços tornando-os mais

eficazes e eficientes e, assim, torna as cidades mais inteligentes.

1.2 Problema

No cenário atual, as cidades possuem uma grande quantidade de dispositivos inteligentes, os quais usam as mais variadas tecnologias para prover dados sobre os diversos serviços e ambientes urbanos. Assim, os cidadãos podem identificá-los em diversos lugares no seu cotidiano, tais como: ônibus são equipados com dispositivos de GPS permitindo o acompanhamento da posição deles ao longo das rotas percorridas; as ruas são monitoradas através de câmeras, o que permite as autoridades identificar crimes, acidentes e problemas de mobilidade urbana; vários sensores estão espalhados nas cidades coletando informações de temperatura, umidade do ar, vento, entre outras informações ambientais.

Entretanto, esses dados gerados são gerenciados de maneira isolada devido a grande quantidade de fontes de dados existentes em uma cidade e, principalmente, em virtude da incompatibilidade existente entre os dispositivos ocasionada pela diversidade de tecnologias envolvidas em um ambiente urbano. Porém, para se obter um melhor funcionamento dos diversos setores de uma cidade é importante que o gerenciamento desses dados seja realizado de maneira integrada, permitindo a troca de informações entre eles e ajudando no processo de tomada de decisão.

No entanto, essa integração de dados não é uma tarefa trivial em razão da heterogeneidade dos dispositivos envolvidos (SU; LI; FU, 2011; GAMA; TOUSEAU; DONSEZ, 2012; ZANELLA et al., 2014), que são de variadas tecnologias, utilizam diferentes protocolos de comunicação e produzem fluxos de dados em múltiplos formatos e com diferentes características. Além disso, uma plataforma com a finalidade de integração tem que se preocupar com diversos desafios, como: privacidade dos dados, compartilhamento de informações entre órgãos envolvidos na administração pública, extração de conhecimento a partir do grande volume de dados, entre outros.

O tratamento desses dados no interior da plataforma também deve ser uma preocupação no desenvolvimento da mesma, pois eles estarão sendo recebidos a todo momento e, ao longo do tempo, novos tipos dos mesmos irão surgir gerando a necessidade que ela tenha a capacidade de integrá-los. Além do mais, não há como prever a forma como eles precisarão ser transformados, visto que cada aplicação ou serviço pode necessitar que os mesmos sejam disponibilizados obedecendo processamentos específicos. Por outro lado, devido a diversidade de tipos de dados existentes é essencial que uma plataforma tenha a

capacidade de tratá-los conforme suas restrições e características.

Desse modo, para trabalhar com esses dados de acordo com suas especificidades e na intenção de garantir que as novas necessidades de processamentos que venham a surgir sejam atendidas, se faz necessário que também exista a possibilidade de extensibilidade com relação ao processo de transformação dos mesmos.

Assim, para permitir a integração dos dados gerados em uma cidade é importante a definição, projeto e implementação de uma plataforma que esteja preparada para receber dados independente da tecnologia utilizada nas fontes geradoras dos mesmos. Além do mais, é necessário que a plataforma, depois de um processo de transformação desses dados, permita que eles sejam utilizados para gerar novas soluções inteligentes que tenham o intuito de melhorar a qualidade de vida no ambiente urbano. Por fim, também é importante que a mesma possibilite extensibilidade em relação ao tratamento dos dados em virtude do que foi mencionado nos parágrafos anteriores.

1.3 Objetivos

Este trabalho tem como objetivo geral o desenvolvimento de uma plataforma extensível para gerenciar os dados em uma cidade inteligente, a qual seja capaz de trabalhar com as variadas tecnologias e dispositivos envolvidos em alguns contextos urbano. Dessa forma, baseado neste objetivo enumera-se os seguintes objetivos específicos:

- Identificar e descrever soluções de *software* utilizadas para gerenciar dados gerados em cidades inteligentes;
- Identificar e especificar requisitos desejados em uma plataforma extensível para trabalhar com dados provenientes de cidades inteligentes;
- Projetar uma plataforma extensível capaz de receber e transformar os dados gerados em ambientes urbanos;
- Implementar a plataforma projetada;
- Avaliar os aspectos extensibilidade e performance da plataforma implementada, através do uso da mesma em cenários típicos de uma cidade, como por exemplo o de um estacionamento.

1.4 Metodologia

De início foi realizada uma revisão exploratória objetivando familiarizar o pesquisador na área de cidades inteligentes. E, também, para realizar um levantamento de plataformas propostas com o intuito de gerenciar esses ambientes ou outros tipos de ambientes, como por exemplo o de Internet das Coisas, compostos por dispositivos de tecnologias heterogêneas.

Em seguida, partiu-se para a definição de uma proposta de plataforma que fosse capaz de integrar essa variedade de dados heterogêneos existentes e que contivesse um processo de transformação extensível para os mesmos, no qual os passos têm responsabilidades específicas, como exibido na Figura 2, e permitem extensão possibilitando que esses dados sejam tratados de acordo com suas peculiaridades. Dessa forma, a partir dessas características definiu-se um conjunto de requisitos e realizou-se o projeto da arquitetura para que a mesma provesse essas funcionalidades.

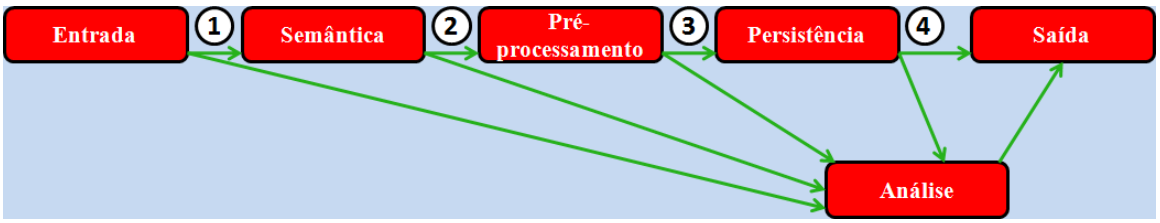


Figura 2: Sequência das atividades do fluxo de dados

Depois, realizou-se a implementação da plataforma e executou-se um estudo de caso no qual se estendeu a mesma para trabalhar com o cenário de um estacionamento inteligente, com o objetivo de avaliar os aspectos extensibilidade e performance relacionados à plataforma implementada. Por fim, descreveu-se o projeto de uma segunda extensão da plataforma para trabalhar com o cenário de controle de multidões.

1.5 Organização do trabalho

O restante deste trabalho encontra-se organizado da seguinte maneira:

- O Capítulo 2 mostra a fundamentação teórica contendo os conceitos necessários ao entendimento deste estudo;
- O Capítulo 3 apresenta a revisão exploratória realizada neste trabalho, a mesma tinha o objetivo de familiarizar o pesquisador no campo de arquiteturas para cidades

inteligentes e levantar um conjunto de plataformas cujo objetivo é gerenciar esse tipo de ambiente ou outros ambientes em que sejam necessária a integração de fontes de dados heterogêneas;

- O Capítulo 4 exibe a plataforma extensível para transformação de fluxo de dados heterogêneos em cidades inteligentes. Além disso, são disponibilizados os requisitos que ela contempla e os padrões arquiteturais que a mesma utiliza;

- O Capítulo 5 apresenta a avaliação da plataforma proposta através de um estudo de caso que examina alguns aspectos relacionados a implementação da mesma e descreve um projeto de utilização dela para trabalhar com o cenário de controle de multidões;

- O Capítulo 6 expõe as considerações finais deste trabalho, destacando as principais conclusões, as contribuições, as limitações e os trabalhos futuros.

2 Fundamentação teórica

Neste capítulo são apresentados os conceitos essenciais ao entendimento desta dissertação. Assim, a Seção 2.1 mostra algumas definições de cidades inteligentes, conceitos relacionados a essa área de pesquisa e destaca alguns exemplos de cidades inteligentes e aplicações relacionados a esse campo de estudo que estão sendo desenvolvidos ao redor do mundo. Na Seção 2.2 é definido o termo Internet das Coisas e mostrada qual a relação do mesmo com a área de cidades inteligentes. Além disso, na Seção 2.3 é descrita a tecnologia OSGi, a qual é de suma importância no entendimento do Capítulo 4.

2.1 Cidades inteligentes

De acordo com Kuper e Kuper (2003), a definição para o termo cidade não é equivalente nas diferentes linguagens, porém esses autores definem essa palavra como "uma concentração razoavelmente grande e permanente de pessoas em um território limitado". No Brasil o significado dessa palavra está ligado a questões político-administrativas, razão pela qual o IBGE a define como "a localidade onde está sediada a Prefeitura Municipal" (IBGE, 2010).

De uma forma geral, as cidades são territórios habitados por um grupo de cidadãos cuja quantidade pode variar de algumas centenas até milhões. E, essas localidades servem de núcleo para as pessoas se socializarem e ajudarem a desenvolver soluções para muitas questões globais através do comércio, tecnologia, arte, cultura, etc (TOPPETA, 2010).

Além disso, nesses ambientes urbanos os cidadãos interagem no seu dia-a-dia com uma grande quantidade de serviços públicos que são "atividades de utilidade ou comodidade para a população, podendo ser executada diretamente pelo poder público ou ser delegada a terceiros" (SIQUEIRA, 2000). No entanto, muito dos problemas enfrentados nas cidades estão ligados a esses serviços, o que dificulta o cotidiano das pessoas e faz com que elas tenham pior qualidade de vida. O aumento populacional mencionado no Capítulo 1, é

um dos fatores que provocou o surgimento de alguns desses problemas, tais como favelamento, congestionamento, violência e poluição, que configuram o chamado caos urbano (SIQUEIRA, 1998).

A Figura 3 ilustra alguns desses problemas dentre os vários enfrentados nos serviços oferecidos nas cidades e como se pode perceber é notável as dificuldades encaradas pelos cidadãos em seu cotidiano. Por esse motivo, vários estudiosos da academia e indústria dedicam esforços para melhorar a qualidade de vida das pessoas através da concepção da ideia de *Smart Cities* (**Cidades Inteligentes**). Entretanto, ainda não existe uma definição padrão para o termo *Smart Cities* (NAM; PARDO, 2011).

Figura 3: Alguns exemplos de problemas enfrentados nas cidades

O termo ***Smart Cities***, apesar de bastante difundido na atualidade, não apresenta uma padronização quanto ao seu significado. Dessa forma, vários autores o interpretam de maneira diferente e criam suas próprias definições para essa expressão. Em sequência, são mostradas algumas definições para **Cidades Inteligentes** encontradas na literatura.

Segundo Angelidou (2014), cidades inteligentes (CI) são vistas como um modelo conceitual de desenvolvimento urbano que utiliza capital humano, coletivo e tecnológico unidos para alcançar a prosperidade no ambiente urbano.

No artigo de Nam e Pardo (2011), é mencionado que o termo *smart city* tem surgido como um modelo para diminuir os atuais problemas das cidades tornando-as locais melhores de se viver. Os autores ainda falam que existem vários conceitos relacionados a esse termo e rotulam os mesmos em três dimensões: **tecnologia** (*digital city*, *intelligent city*, *ubiquitous city*, *wired city*, *hybrid city*, *information city*), **pessoas** (*creative city*, *learning*

city, *humane city*, *knowledge city*) e **integrado** (*smart community*).

Na dissertação de mestrado de Tomas (2014), cidade inteligente é definida como "a combinação de Tecnologia da Informação e Comunicação (TIC) com todos os aspectos que compõem uma cidade, desde de aspectos físicos e governamentais baseados nas necessidades dos cidadãos".

Na visão de Su, Li e Fu (2011), *smart city* significa o uso de tecnologias da informação e comunicação para entender, analisar e integrar informações dos sistemas centrais que compõem as cidades.

De acordo com Toppeta (2010), *smart cities* são cidades que combinam TIC e Web 2.0 com outros esforços organizacionais para criar soluções inovadoras que gerenciem o espaço urbano, objetivando aumentar a qualidade de vida das pessoas e a sustentabilidade. O estudo de Bakici, Almirall e Wareham (2013), também segue essa mesma linha de pensamento definindo *smart city* como cidades que utilizam TIC para melhorar a qualidade de vida dos seus habitantes e prover desenvolvimento sustentável.

O trabalho de Harrison e Donnelly (2011), utiliza o termo *smart city* de um ponto de vista mais tecnológico, adotando essa expressão para aplicações de sistemas de informação complexos responsáveis por integrar a operação de infra-estrutura e serviços, tais como distribuição elétrica, edifícios, rede de transportes, entre outros.

Os pesquisadores Murgante e Borruso (2014), criticam o fato de várias definições encontradas na literatura tratarem o termo cidade inteligente apenas de um ponto de vista tecnológico e esquecerem da essência da cidade com seus desafios e problemas. No artigo de Neirotti et al. (2014), em conformidade com as ideias anteriores, é defendido que *smart city* não está ligado apenas a mudanças tecnológicas, mas também em mudanças nas condições e práticas urbanas e investimentos em capital humano.

Por sua vez, a revista ComputerWorld (2013), diz que a área de *smart city* está preocupada em tornar a gestão urbana mais inteligente e também menciona que ela deve abranger tecnologia, pessoas e instituições. Além disso, cita algumas conclusões sobre cidades inteligentes: não tem tudo a ver com tecnologia; é preciso mais do que tecnologia da informação e comunicação; os cidadãos devem ser envolvidos; é mais barato trabalhar direto com fabricantes; as outras cidades devem servir de referência.

Em Giffinger et al. (2007), é mostrado um conjunto de exemplos onde se é utilizado o termo *smart city*. Quando relacionado a economia e empregos essa expressão refere-se a uma cidade com uma indústria que utiliza tecnologias de última geração no seu processo

produtido. Em se tratando da educação, uma *smart city* é uma cidade cujos habitantes têm um elevado grau de educação. Por fim, também é mencionado que o termo é usado para se referir a relação entre cidadãos e governo, além do uso da tecnologia no ambiente urbano. Os autores citam um conjunto de seis eixos que compõem uma cidade inteligente:

- ***Smart Economy*** – inclui fatores relacionados a competitividade econômica;
- ***Smart People*** – relacionado ao capital social e humano, levando-se em conta qualificação dos cidadãos e as interações sociais dos mesmos;
- ***Smart Governance*** – pertinente ao funcionamento dos serviços públicos, além do processo administrativo de uma cidade e a participação política da população;
- ***Smart Mobility*** – ligado ao processo de acessibilidade a ambientes de uma cidade, bem como o uso de tecnologias da informação e comunicação nos sistemas de transportes modernos e sustentáveis;
- ***Smart Environment*** – associado ao desenvolvimento da cidade sem esquecer de um uso sustentável dos recursos naturais;
- ***Smart Living*** – referente a qualidade de vida dos habitantes de uma cidade.

Dessa forma, uma cidade inteligente deve almejar um melhor desempenho desses seis eixos relacionados ao ambiente urbano (NAM; PARDO, 2011; MURGANTE; BORRUSO, 2014; GRECO; BENCARDINO, 2014).

Neste trabalho considera-se que ***Smart City*** **é uma cidade que utiliza tecnologias da informação e comunicação na tentativa de amenizar e/ou solucionar os problemas enfrentados pelas pessoas, governo e empresas do meio urbano, sem esquecer das questões sociais e ambientais do local**.

2.1.1 Iniciativas de cidades inteligentes

Esta seção traz exemplos de algumas cidades inteligentes que estão sendo construídas ao redor do mundo e mostra também iniciativas nessa área de estudos desenvolvidas em cidades já existentes.

2.1.1.1 *PlanIT Valley*

A cidade de PlanIT Valley, ilustrada na Figura 4, é um iniciativa idealizada e desenvolvida pela empresa privada Living PlanIT cuja ideia central é reduzir o impacto ambiental gerado pelo estilo de vida na zona urbana. Do ponto de vista territorial, está localizada na cidade de Paredes ao norte de Portugal, ocupará 1.700 hectares e terá 225.000 habitantes (PLANIT, 2013).

Figura 4: Ilustração do projeto da cidade de PlanIT Valley(PLANIT, 2010)

PlanIT Valley contará com uma ampla rede de sensores que permitirá a quantificação de todos os tipos de fluxo (trânsito, água, energia, pessoas, etc). E, por fim, serão fornecidas aplicações para telefones e terminais de computadores no entorno da cidade para ajudar os moradores em suas atividades diárias (CAMPOS, 2012).

2.1.1.2 *Songdo International Business District* (IBD)

É uma cidade criada do zero que começou a ser construída no ano de 2.005 e tem previsão para ficar totalmente pronta em 2.015. Ela se localiza na Coreia do Sul e ocupará uma área de 1.570 km^2 onde morarão 40 mil pessoas. A ideia principal desse projeto é explorar as melhores práticas de planejamento urbano atreladas à sustentabilidade com o intuito de fornecer uma ótima qualidade de vida a seus habitantes (INTERNATIONAL, 2014).

IBD tem uma localização estratégica, como mostrado na Figura 5, devido sua proximidade ao *Incheon International Airport*, sendo esse aeroporto uma rota importante no cenário mundial, uma vez que, fica a aproximadamente três horas e meia de voo para 1/3 da população do planeta (Japão, China e Rússia) (GROVER, 2013). E, por esses motivos,

a cidade passou a ser chamada de “Aerotrópoles”.

Figura 5: Detalhe da localização estratégica da cidade de Songdo(IBD, 2014)

Songdo ainda incentivará o descarte correto do lixo identificando através de sensores nas lixeiras as pessoas que o fizeram e bonificando-as. Assim, todo lixo produzido na cidade é levado por meio de canos pressurizados para debaixo da terra, local onde se dará a separação e reciclagem desses materiais (INTERNATIONAL, 2014).

A cidade pretende ser livre de congestionamentos e para isso utiliza uma rede de sensores no asfalto que detecta a velocidade dos veículos fornecendo dados para o cálculo do tempo de abertura dos semáforos (INTERNATIONAL, 2014).

Para concluir, a cidade conta ainda com muitas áreas verdes, escolas públicas, unidades de saúde, atividades culturais, instalações esportivas, lojas, mercados, entre outros ambientes que utilizarão tecnologia de ponta para fornecer uma melhor qualidade de vida aos cidadãos (HODGKINSON, 2011).

2.1.1.3 *Masdar City*

Masdar City está localizada na Ásia mais especificamente no deserto de Abu Dhabi nos Emirados Árabes. Ela ocupará uma área de 6 km^2 e contará com uma população de 50.000 pessoas. O projeto está centrado em prover alta qualidade de vida com pouco impacto ambiental (HAUBENSAK, 2011).

A cidade busca utilizar fontes de energia limpa, considerando isso o projeto prever 87.000 painéis para produzir energia solar e vários cataventos para gerarem energia eólica. Ainda pensando em evitar a emissão de carbono na atmosfera, o sistema de transporte será todo elétrico e contará com ônibus, carros e linhas de metrô, todos movidos a esse tipo de energia (MASDAR, 2011).

Ela utilizará vários sistemas e tecnologias para reduzir o consumo de água, um exemplo

são os contadores inteligentes que poderão identificar vazamentos permitindo a diminuição do desperdício de água. Além disso, os esgotos serão tratados para que a água possa ser reutilizada o que reduzirá 60% do consumo da mesma e a água do mar será dessalinizada para que possa ser consumida pelos habitantes de Masdar City (ALNASERA; ALNASERB, 2011).

2.1.1.4 Rio de Janeiro

O Rio de Janeiro é a cidade brasileira que tem destaque na área de cidades inteligentes, a mesma conta com um Centro de Operações (COR), exibido na Figura 6, tido como o mais avançado do mundo. O COR é um ambiente de tomada de decisão que funciona 24 horas por dia e sete dias por semana. Além disso, nele mais de 400 operadores se revezam no monitoramento da cidade que é realizada por mais de 900 câmeras pertencentes à prefeitura e a órgãos parceiros (GELLI, 2015).

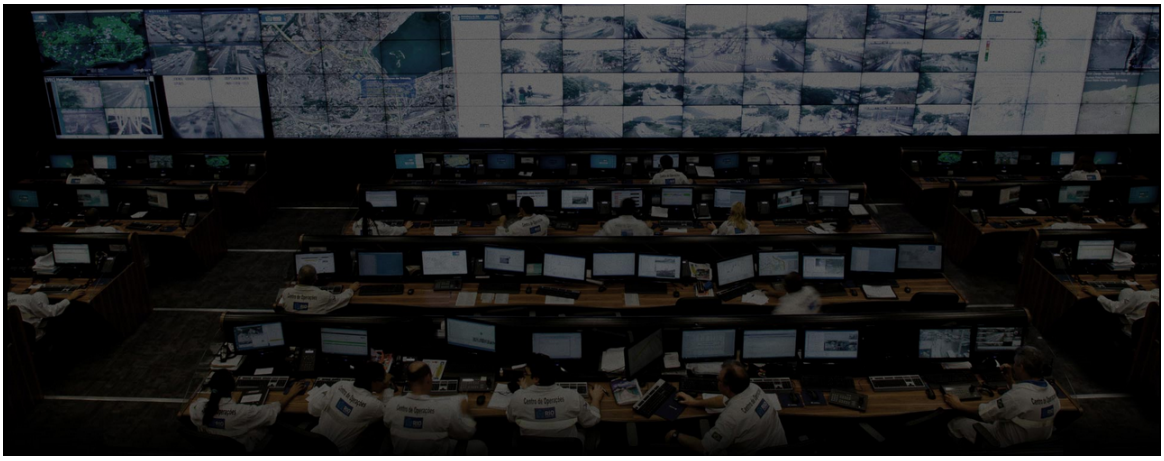


Figura 6: Sala de controle do Centro de Operações Rio(JANEIRO, 2015)

Além do mais, o COR engloba mais de 30 órgãos relacionados a administração da cidade que trabalham em conjunto com o intuito de tomar melhores decisões no que diz respeito aos problemas que acontecem no dia-a-dia dela, tais como: alagamentos, engarrafamentos, acidentes, etc. (GELLI, 2015).

Outra iniciativa da cidade do Rio de Janeiro é a central única de teleatendimento (Disque-Rio), ela funciona através do número telefônico 1746 e de um aplicativo para *smartphones* e *tablets*. A finalidade do Disque-Rio é unificar o atendimento aos cidadãos em um único canal, acabando com a necessidade do munícipe de ter que memorizar diferentes números das várias secretarias que compõem a administração da cidade (CUNHA, 2011).

Por fim, a prefeitura do Rio de Janeiro também espalhou adesivos *QR Code* em 5.000 pontos de ônibus da cidade. Esses adesivos podem ser lidos através de um aplicativo fornecido nas tecnologias *Android*, *iOS* e *Windows Phone* e eles armazenam dados sobre pontos turísticos, linhas e trajetos de ônibus (DAVID, 2014).

2.1.1.5 SmartSantander

A cidade de Santander está localizada na Espanha e seu projeto de cidade inteligente tem quatro objetivos: o primeiro deles é prover um modelo que sirva de referência para sistemas de **internet das coisas** existentes no mundo real; o segundo é fornecer um facilitador de experimentos baseados em escalabilidade, heterogeneidade e confiabilidade; disponibilizar um conjunto de casos de uso implementados; e gerar um vasto conjunto de experimentos e resultados sobre **internet do futuro** (SANCHEZ et al., 2011; SMARTSANTANDER, 2015).

SmartSantander possui 12.000 dispositivos de diferentes tecnologias espalhados em seu ambiente. Além disso, ela tem pesquisadores, usuários finais e provedores de serviço como alvos e envolve uma grande variedade de cenários urbanos, tais como: monitoramento ambiental, gerenciamento de área de estacionamento, irrigação de parques e jardins, entre outros (SMARTSANTANDER, 2015).

2.1.1.6 Barcelona

A ideia de cidade inteligente em Barcelona está centrada no uso de tecnologia de ponta para torná-la avançada conectando pessoas, informações e elementos da cidade. Além disso, utiliza-se novas tecnologias para criar uma cidade mais sustentável, verde, aumentar a competitividade do comércio, melhorar a qualidade de vida e simplificar a administração (BAKICI; ALMIRALL; WAREHAM, 2013).

A cidade inclui energias alternativas, gerenciamento de transporte e políticas de construções verdes. Além disso, ela é interconectada através de uma cobertura de banda larga e fornece pontos de acesso grátis para os cidadãos. Por fim, em Barcelona existem vários sensores espalhados no seu território captando diversas informações do meio urbano e a cidade possui a preocupação da abertura das informações do governo para o acesso público (ZYGIARIS, 2012).

2.1.2 Algumas aplicações de *software* em cidades inteligentes

Esta seção busca exemplificar algumas aplicações criadas na área de cidades inteligentes e pensados para melhorar a qualidade de vida dos cidadãos que residem nas localidades urbanas.

2.1.2.1 *Active Citizenship*

É um sistema de informação criado objetivando melhorar e agilizar a comunicação entre cidadãos e os órgãos responsáveis pela administração da cidade de Jeddah na Arábia Saudita. Assim, criou-se um aplicativo para facilitar o compartilhamento dos problemas das cidades, o mesmo utiliza a câmera e o GPS do *smartphone* para gerar uma imagem e a localização do problema percebido pelo usuário que, em seguida, o compartilha através da *internet* (FARDOUN; ALTALHI; LÓPEZ, 2012).

Depois, o órgão responsável recebe a notificação do problema, posteriormente, realiza as ações necessárias para solucioná-lo e, finalmente, responde ao cidadão agradecendo-o por reportá-lo e indicando os prazos previstos para a resolução do mesmo (FARDOUN; ALTALHI; LÓPEZ, 2012).

2.1.2.2 SMARTY

É um projeto fundado na região da Toscana (Itália) que objetiva o desenvolvimento de serviços de transportes inovadores em cidades inteligentes e coleta informações sobre o fluxo do tráfego, boletins meteorológicos, poluição, atrasos de serviços de transporte, disponibilidade de estacionamentos, entre outros (ANASTASI et al., 2013).

Os dados do *SMARTY* são coletados através de sensores ambientais e sociais, passam por um processo de pré-processamento e, em seguida, são analisados através de técnicas de mineração de dados a fim de gerar conhecimento que possam ser utilizados na melhoria da mobilidade urbana da região (ANASTASI et al., 2013).

2.1.2.3 MOBISEC

Esse projeto foi criado objetivando aumentar a segurança dos ciclistas nas estradas. Assim, ele recolhe informações sobre as rotas dos ciclistas e as interações deles com outros usuários das rodovias através de um aplicativo para *smartphones* e *tablets* que envia os dados para um servidor (MELENDRERAS-RUIZ; GARCIA-COLLADO, 2013).

Ao iniciar sua atividade física o ciclista inicia o aplicativo que registra os lugares por onde ele passa e quando chegar ao término de seu exercício o mesmo deve pará-lo. Dessa forma, pretende-se obter informações sobre as rotas regulares dos ciclistas de uma cidade para utilizá-las como base na realização de iniciativas que melhorem a segurança dos mesmos nas rodovias (MELENDRERAS-RUIZ; GARCIA-COLLADO, 2013).

2.1.2.4 *Cyber Parallel Traffic World* (CPTW)

O CPTW também é um projeto direcionado ao campo da melhoria na mobilidade urbana. Nele pedestres, veículos e obstáculos (semáforos, placas, etc) enviam dados de suas localizações para um servidor que as utiliza para representar a situação das rodovias por meio de um mapa. Assim, essas informações podem ser utilizadas com diversas finalidades, indo desde a prevenção de acidentes até o treinamento de novos motoristas através de simulações com dados reais. Por fim, o envio desses dados pode ser feito por dispositivos de vários tipos como carros, *smartphones*, *tablets*, semáforos, etc (MURATA; SAITO, 2014).

2.2 Internet das coisas

A **Internet das Coisas** (*Internet of Things* ou IoT) é um paradigma no qual os objetos da vida cotidiana são equipados e habilitados para se comunicar com outros objetos e usuários o que torna esses objetos parte da *internet* (ZANELLA et al., 2014). Nesse paradigma os variados objetos que hoje nos cercam são conectados para proverem informações sobre o ambiente no qual os mesmos estão inseridos. Assim, essas informações podem ser utilizadas na criação de novas soluções e serviços com o intuito de melhorar a qualidade de vida das pessoas nesses locais.

Dessa maneira, tais objetos também são capazes de trabalhar em diversos ambientes do contexto das cidades fornecendo dados dos mesmos e possibilitando o uso dessas informações por parte da administração pública, cidadãos e empresas. Hoje existem vários cenários de aplicação para a Internet das Coisas no contexto urbano, como por exemplo o uso de dispositivos que forneçam informações sobre os meios de transporte, objetos que gerem dados relacionados a saúde das pessoas, sensores que coletem dados ambientais, entre outros (ATZORI; IERA; MORABITO, 2010).

Desse modo, as tecnologias de Internet das Coisas são importantes como ferramentas para monitorar o meio urbano no intuito de resolver alguns problemas existentes no ambiente das cidades e na tentativa de proporcionar melhores condições de vida aos ci-

dadãos (TOMAS, 2014). Assim, é evidente a correlação existente entre as áreas de cidades inteligentes e internet das coisas sendo que as mesmas podem ser utilizadas em conjunto para resolver ou amenizar alguns problemas existentes na zona urbana.

2.3 *Open Services Gateway Initiative* (OSGi)

OSGi é um conjunto de especificações que define um sistema dinâmico para o ambiente Java, facilitando o desenvolvimento de *softwares* de maneira modular nessa plataforma. Essas especificações são mantidas por um consórcio de empresas que foi fundado em março de 1999, chamado OSGi Alliance[1].

A modularidade proporcionada pelo OSGi traz diversos benefícios aos desenvolvedores, o código torna-se menos complexo devido a divisão do problema em partes menores, o *deploy* é mais gerenciável visto que é permitido administrar cada módulo individualmente e o sistema de *build* torna-se melhor. Além disso, o uso do OSGi facilita a manutenção do sistema e reduz os custos das empresas nesse processo.

2.3.1 *Bundles*

Um *bundle* é a unidade de modularização na especificação OSGi. Eles são arquivos *.jar* compostos por arquivos Java e outros recursos que juntos fornecem funcionalidades aos usuários (ALLIANCE, 2014). Além do mais, *bundles* podem compartilhar pacotes entre si e um *software* baseado em OSGi é composto por um conjunto de *bundles* que interagem entre eles.

Cada *bundle* possui um arquivo *MANIFEST.MF* localizado no diretório *META-INF*, esse arquivo é responsável por descrever o *bundle* fornecendo as informações necessárias para que um ambiente OSGi possa gerenciá-lo. Na Listagem 2.1 é mostrado um exemplo do arquivo *MANIFEST.MF* e se pode perceber que ele é composto por um conjunto de atributos definidos pela especificação OSGi.

```
1 Manifest-Version: 1.0
2 Bundle-ManifestVersion: 2
3 Bundle-Name: AloMundo
4 Bundle-SymbolicName: AloMundo
5 Bundle-Version: 1.0.0
6 Bundle-Activator: alomundo.Activator
```

[1] http://www.osgi.org/

```
7 Bundle-Vendor: CICERO
8 Bundle-RequiredExecutionEnvironment: JavaSE-1.7
9 Import-Package: org.osgi.framework;version="1.3.0"
```

Listagem 2.1: *MANIFEST.MF*

Os atributos mais utilizados no *MANIFEST.MF* podem ser vistos na Tabela 1, na qual também é possível visualizar uma breve descrição de cada atributo citado. Vale ressaltar que um ambiente OSGi aceita mais de um *bundle* com os mesmos valores para a propriedade *Bundle-SymbolicName* desde que esses *bundles* tenham valores diferentes para a propriedade *Bundle-Version*, isso é importante caso *bundles* em execução no ambiente utilizem versões diferentes de um mesmo *bundle*.

Tabela 1: Tabela atributos do *MANIFEST.MF* (ALLIANCE, 2012).

Atributo	**Descrição**
Bundle-ManifestVersion	Define que o *bundle* segue as regras da especificação **X** do OSGi
Bundle-Name	Define um nome legível para o *bundle*
Bundle-SymbolicName	Define um nome que identifica o *bundle* no ambiente OSGi
Bundle-Version	Especifica a versão do *bundle*
Bundle-Activator	Especifica a classe responsável por iniciar e parar o *bundle*
Bundle-Vendor	Especifica o vendedor/criador do *bundle*
Bundle-RequiredExecutionEnvironment	Define o ambiente de execução onde o *bundle* deve ser executado
Import-Package	Especifica uma lista de pacotes importados pelo *bundle*
Export-Package	Especifica uma lista de pacotes exportados pelo *bundle*
Bundle-ActivationPolicy	Especifica como o *framework* deve ativar o *bundle* após iniciado

Como exibido na Figura 7, um *bundle* pode assumir um conjunto de estados dentro do *framework* OSGi devido ao seu ciclo de vida. Ao ser instalado o *bundle* assume o estado *INSTALLED*. Em seguida, caso o OSGi consiga resolver todas as dependências dele, o mesmo passa ao estado *RESOLVED* e ao ser parado um *bundle* também assume esse mesmo estado. Posteriormente, quando está em execução o *bundle* passa ao estado *ACTIVE* e ao ser desinstalado o *bundle* adota o estado *UNINSTALLED*.

Um *bundle* é ativado através de uma classe que implementa a interface *BundleActivator* identificada no atributo *Bundle-Activator* do arquivo *MANIFEST.MF*. Ao im-

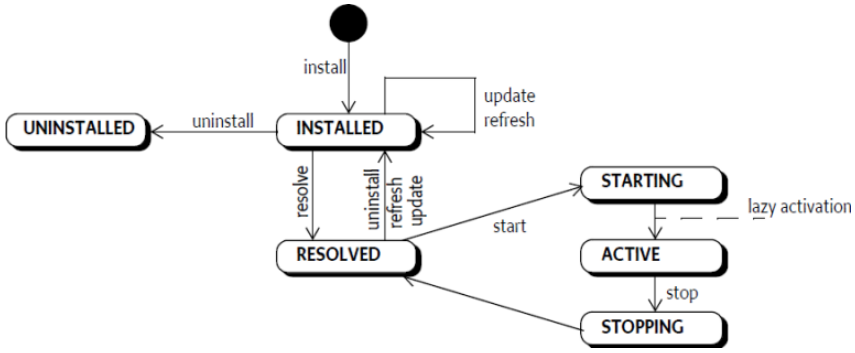


Figura 7: Diagrama de estados dos *bundles* (ALLIANCE, 2011)

plementar essa interface a classe deve sobrescrever os métodos *start* e *stop* definidos no *BundleActivator*. Na Listagem 2.2 tem-se um exemplo da implementação de uma classe que implementa a interface *BundleActivator*, no método *start* o desenvolvedor deve implementar todas as operações que o *bundle* requisita para funcionar e no método *stop* o programador deve finalizar as operações executadas pelo *bundle*.

```
1  public class Activator implements BundleActivator {
2          private static BundleContext context;
3          static BundleContext getContext() {
4                  return context;
5          }
6          public void start(BundleContext bundleContext) throws Exception {
7                  System.out.println("Bundle iniciado");
8          }
9          public void stop(BundleContext bundleContext) throws Exception {
10                 System.out.println("Bundle parado");
11         }
12 }
```

Listagem 2.2: Exemplo de implementação da interface *BundleActivator*

Os métodos *start* e *stop* do *BundleActivator* recebem por parâmetro um objeto do tipo *BundleContext*. A partir desse objeto um *bundle* tem acesso ao *framework* OSGi no qual está instalado. E, dessa forma, ele pode interagir com outros *bundles*, além de ter acesso ao gerenciamento do *deployment* e do ciclo de vida de outros *bundles* (HALL et al., 2011) .

2.3.2 *Services*

Um *service* é um objeto Java registrado como uma ou mais interfaces através do serviço de registro (ALLIANCE, 2014). Ao utilizar eles muitos benefícios são obtidos, como: menor acoplamento entre provedores e consumidores do serviço, facilidade de reuso de componentes, maior ênfase em interfaces ao invés de classes, descrição clara de dependências e suporte para múltiplas implementações do mesmo serviço (HALL et al., 2011).

Na Listagem 2.3 pode-se perceber como é simples o processo de registro de *Services*. O objeto *BundleContext* fornece o método *registerService* cuja funcionalidade é publicar serviços no *framework*. Esse método recebe três parâmetros sendo o terceiro parâmetro opcional. O primeiro deles é o nome de registro do serviço, o segundo recebe o objeto que implementa o serviço e o último é uma lista de propriedades do serviço.

```
1 public void start(BundleContext bundleContext) throws Exception {
2         ExemploServico servico = new ExemploServicoImp();
3         bundleContext.registerService(ExemploServico.class.getName(),
              servico, null);
4 }
```

Listagem 2.3: Exemplo de registro de um *Service*

A Listagem 2.4 exibe um exemplo de como recuperar um serviço na plataforma OSGi. Primeiramente, obtém-se uma referência para o serviço utilizando o método *getServiceReference* que recebe como parâmetro o nome do serviço. Em seguida, recupera-se o serviço usando o método *getService* que recebe por parâmetro o objeto retornado no *getServiceReference*.

```
1 public void start(BundleContext bundleContext) throws Exception{
2         ServiceReference ref = bundleContext.getServiceReference(
              ExemploServico.class.getName()) ;
3         ExemploServico service = (ExemploServico) bundleContext.getService(
              ref);
4         service.funcionalidadeDoServico();
5 }
```

Listagem 2.4: Exemplo de recuperação de um *Service*

2.3.3 *Events*

Os *Events* (Eventos) são gerenciados pelo serviço *Event Admin* que é um modelo *publish/subscribe* para a troca de mensagens entre os *bundles* em execução na plataforma

OSGi. O *Event Admin* permite que os publicadores e assinantes dos eventos se comuniquem sem que eles estejam acoplados, pois para o envio e recebimento dessas mensagens não é necessário que esses *bundles* criem serviços para fazer essa comunicação (BAKKER; ERTMAN, 2013).

Um *Event* possui dois atributos *Topic* e *Properties*. O *Topic* representa o tipo do *Event* e *Properties* são um conjunto de informações sobre o *Event*. Eles podem ser enviados tanto de maneira síncrona quanto assíncrona (BAKKER; ERTMAN, 2013).

A Listagem 2.5 traz um exemplo de envio de um evento de maneira síncrona. Para enviar um *Event*, primeiramente, é necessário recuperar uma referência para o serviço *Event Admin* como feito na linha 8 e, posteriormente, acessar o serviço *Event Admin* propriamente dito conforme realizado na linha 10. Em sequência, necessita-se apenas configurar os atributos *Topic* e *Properties* no novo *Event*, como realizado nas linhas 20, 21 e 22, e chamar o método *sendEvent* passando-o como parâmetro, passo esse equivalente a linha 23. No que diz respeito ao envio de *Events* assíncronos, o processo é o mesmo do envio síncrono porém é necessário substituir o método *sendEvent* por *postEvent*.

```
1  public class Activator implements BundleActivator{
2          private static BundleContext context;
3          ServiceReference sr = null;
4          EventAdmin ea = null;
5          Dictionary properties = null;
6          public void start(BundleContext bundleContext) {
7                  this.context = bundleContext;
8                  sr = bundleContext.getServiceReference(EventAdmin.class.
                       getName());
9                  if (sr != null) {
10                         ea = (EventAdmin) bundleContext.getService(sr);
11                         if (ea != null) {
12                                 enviarEvento();
13                         }
14                 }
15         }
16         public void stop(BundleContext arg0) throws Exception {
17                 this.context.ungetService(sr);
18         }
19         public void enviarEvento() {
20                 properties = new Hashtable();
21                 properties.put(EventConstants.BUNDLE_SYMBOLICNAME, "br.org.
                       alomundo");
22                 Event event = new Event("br/org/alomundo", properties);
```

```
23                  ea.sendEvent(event);
24          }
25  }
```

Listagem 2.5: Exemplo de envio de *Event* síncrono

Para receber *events* é necessário realizar a implementação da interface *EventHandler*. O próximo passo é registrar a classe como um serviço *EventHandler* passando-se como parâmetro as propriedades que identificam o evento que se quer receber, como exibido na linha 11. Por fim, os eventos serão recebidos no método *handleEvent* mostrado na linha 16, no qual o desenvolvedor deve programar as ações que se quer realizar com o *event*.

```
1  public class Activator implements BundleActivator, EventHandler{
2          private static BundleContext context;
3          final static String[] topic = { "br/org/alomundo" };
4          String filter = "(bundle.symbolicName=br.org.alomundo)";
5          ServiceRegistration register;
6          public void start(BundleContext bundleContext) {
7                  this.context = bundleContext;
8                  Dictionary dict = new Hashtable();
9                  dict.put(EventConstants.EVENT_TOPIC, topic);
10                 dict.put(EventConstants.EVENT_FILTER, filter);
11                 register = bundleContext.registerService(EventHandler.class
                       .getName(), this, dict);
12         }
13         public void stop(BundleContext arg0) throws Exception {
14                 register.unregister();
15         }
16         public void handleEvent(Event event) {
17                 //Fazer algo com o event recebido
18         }
19 }
```

Listagem 2.6: Exemplo de um assinante de *Event*

2.3.4 Camadas do OSGi

De acordo com o relatório técnico Alliance (2014) e como se pode ver na Figura 8, as funcionalidades da arquitetura OSGi são divididas em quatro camadas: camada de segurança, camada modular, camada de ciclo de vida e camada de serviço.

A **camada de segurança** (*Security layer*) é baseada em Java 2, porém adiciona um

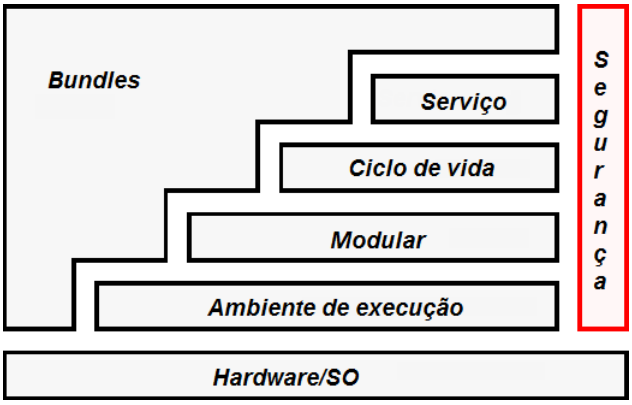


Figura 8: Camadas OSGi, adaptado de (ALLIANCE, 2014)

conjunto de restrições e preenche algumas lacunas deixadas em aberto pelo padrão Java. Com essa camada os desenvolvedores têm suporte a um ambiente controlado no qual permissões definem as regras de implantação e gerenciamento.

Na **Camada modular** (*Modular layer*) é definido o modelo de modularização para OSGi, os chamados *bundles*, e estabelecido as regras para o compartilhamento e ocultação de pacotes entre *bundles*.

A **Camada de ciclo de vida** (*Life Cycle Layer*) fornece uma API de ciclo de vida para gerenciar os *bundles*, definindo como eles são instalados, iniciados, parados, alterados e desinstalados no ambiente OSGi.

Na **Camada de serviço** (*Service Layer*) é fornecido um modelo de programação dinâmico simplificando o desenvolvimento de *bundles* de serviço por desacoplar as definições dos serviços (interfaces) das implementações (classes). Nela um *bundle* registra um serviço no *framework* para que outros *bundles* possam acessá-lo e utilizá-lo, como mostrado na Figura 9. Com essa camada, os desenvolvedores podem gerar diferentes implementações de um serviço e selecionar qualquer uma delas em tempo de execução.

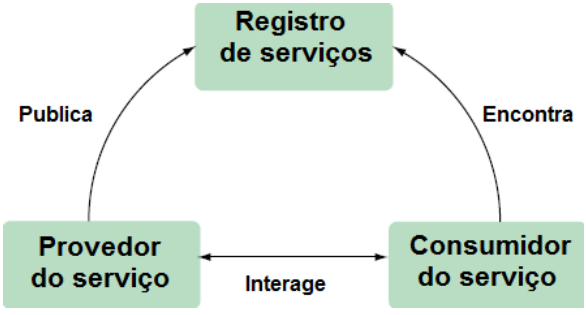


Figura 9: Funcionamento da camada de serviço, adaptado de (HALL et al., 2011)

3 Revisão exploratória

A revisão exploratória é uma metodologia de pesquisa utilizada para recolher informações preliminares sobre um tema, sendo conduzida para explorar um tópico de estudo objetivando situar o pesquisador sobre o mesmo (KOTLER; ARMSTRONG, 2012; BABBIE, 2007). Esses tipos de trabalhos não possuem um processo definido em um guia e são mais flexíveis permitindo a inclusão de estudos reportados em vários tipos de mecanismos, como em relatórios técnicos e páginas *Web*. Por outro lado, uma revisão exploratória é mais subjetiva do que outras estratégias de pesquisa o que dificulta ou impossibilita uma tentativa de reprodução dela.

Assim, neste trabalho optou-se por realizar um estudo exploratório com o intuito de familiarizar o pesquisador no campo de arquiteturas para cidades inteligentes e para levantar um conjunto de plataformas cujo objetivo é gerenciar esse tipo de ambiente ou outros ambientes em que seja necessária a integração de fontes de dados heterogêneas.

Dessa forma, primeiramente foi selecionada a dissertação de Tomas (2014), que serviu de base para este trabalho em razão da mesma definir uma arquitetura para cidades inteligentes. Em seguida, selecionou-se o artigo de Borja e Gama (2014), pelo fato do mesmo descrever um *middleware* que trabalha com dados heterogêneos. Por conseguinte, foram selecionados os trabalhos referenciados nos dois estudos anteriores que descreviam bem suas propostas de plataforma. Em sequência, analisou-se as plataformas levantadas no *survey* de Silva et al. (2015), e foram selecionadas aquelas que além de trabalhar com o cenário de Internet das Coisas abordam a área de cidades inteligentes.

Desse modo, foram analisados trabalhos que encontram-se publicados nos principais repositórios da área tecnológica, soluções disponibilizadas em páginas *web*, além de uma dissertação de mestrado e artigos reproduzidos em outros veículos de publicação.

Por fim, como mostrado na Tabela 2 cada um dos estudos explorados recebeu um código para facilitar as referências no processo de análise dos mesmos.

Tabela 2: Códigos atribuídos aos trabalhos

Trabalho	Código
(ANTHOPOULOS; FITSILIS, 2010)	**P1**
(FILIPPONI et al., 2010)	**P2**
(HERNANDEZ; LARIOS, 2014)	**P3**
(GAMA; TOUSEAU; DONSEZ, 2012)	**P4**
(VALENTE; MARTINS, 2011)	**P5**
(BLACKSTOCK et al., 2010)	**P6**
(TOMAS, 2014)	**P7**
(BORJA; GAMA, 2014)	**P8**
(ANDREINI et al., 2011)	**P9**
(FIWARE, 2011)	**P10**
(OCTOBLU, 2014)	**P11**
(CARRIOTS, 2011)	**P12**
(DIGI, 2013)	**P13**
(CYBERVISION, 2014)	**P14**
(PRISMTECH, 2014)	**P15**
(SEECONTROL, 2014)	**P16**
(INDRA, 2014)	**P17**

3.1 Resultados

Esta seção mostra os trabalhos examinados nesta revisão exploratória, trazendo detalhes das características de cada um deles e as tecnologias utilizadas nos mesmos. Assim, esses estudos foram divididos em três subseções de acordo com o tipo de ambiente de dados heterogêneos em que eles trabalham. Na Subseção 3.1.1 são destacados os estudos que lidam com o ambiente de cidades inteligentes. Por sua vez, a Subseção 3.1.2 aborda os trabalhos centrados no ambiente de internet das coisas. Por último, na Subseção 3.1.3 são mostrados os estudos que combinam internet das coisas e cidades inteligentes em suas soluções.

3.1.1 Trabalhos que focam em cidades inteligentes

Nesta subseção são abordados os trabalhos que propõem soluções para o ambiente de dados heterogêneos de uma cidade inteligente.

3.1.1.1 Plataforma P1

No estudo de Anthopoulos e Fitsilis (2010), os autores realizam o levantamento de algumas experiências de cidades digitais para elaborar uma arquitetura comum de gerência

de serviços de espaços urbanos. A arquitetura proposta segue os princípios de SOA e combina experiências de cidades ubíquas.

Porém, o trabalho trata apenas da descrição da arquitetura. Assim, ele não aborda detalhes de implementação da plataforma proposta e tampouco discute qualquer forma de validação referente a ela. Desse modo, não é possível identificar os módulos que devem ser implementados para que as camadas propostas no mesmo funcionem e, além disso, não se pode atestar se elas são eficazes para gerenciar o ambiente urbano.

3.1.1.2 Plataforma P2

O estudo de Filipponi et al. (2010), apresenta uma arquitetura baseada em eventos que permite o gerenciamento e cooperação de sensores com tecnologias heterogêneas para monitoramento de espaços públicos. Dessa forma, cada evento representa uma mudança detectada por tecnologias de informação e comunicação. Os principais elementos da arquitetura são: *Processador de Conhecimento* que é responsável por inserir e consumir notificações de eventos e *Corretores de Informação* responsáveis por armazenar eventos, analisar duplicações e notificar a ocorrência de eventos. Todavia, essa arquitetura tem um foco de utilização muito restrito e não engloba vários requisitos de uma plataforma para gerenciamento do ambiente de uma cidade inteligente, tais como: privacidade de dados, monetização, inclusão de outros tipos de fontes de dados diferentes de redes de sensores, etc.

3.1.1.3 Plataforma P3

O trabalho de Hernandez e Larios (2014), propõe uma arquitetura em *cloud computing* para implementar os serviços digitais de uma cidade inteligente. A plataforma é composta por quatro módulos: 1) **segurança** que implementa as permissões de acesso aos serviços oferecidos; 2) **serviço**, módulo responsável por gerenciar a demanda de processamento na *cloud*; 3) **carga de trabalho** cuja atribuição é realizar a otimização de recursos (consumo de energia, memória, espaço em disco rígido e processos de *software*) e 4) **compactação de dados** que tem a função de realizar a compressão dos dados resultantes após a finalização de uma tarefa da arquitetura. Como pode ser visto, a plataforma é centrada na otimização de recursos, mas não inclui outros requisitos de uma arquitetura para cidades inteligentes.

3.1.2 Trabalhos que focam em internet das coisas

Esta subseção mostra os estudos cujas soluções propostas trabalham com os dados heterogêneos do ambiente de internet das coisas.

3.1.2.1 Plataforma P4

No artigo de Gama, Touseau e Donsez (2012), é proposto um *middleware* para **Internet das Coisas** que permite a combinação de serviços com tecnologias heterogêneas. Ele foi implementado utilizando OSGi e baseia-se nos princípios de computação orientada a serviços, permitindo baixo acoplamento, flexibilidade para as aplicações e fácil integração de dispositivos heterogêneos. Contudo, o estudo não fornece grandes detalhes de seu processo de armazenamento de dados. Além disso, não inclui realização de extração de conhecimento a partir dos dados integrados. Ademais, não possui estratégias de privacidade das informações trafegadas nela e não define um fluxo de transformação de dados em seu interior.

3.1.2.2 Plataforma P5

O trabalho de Valente e Martins (2011), também apresenta um *middleware* de **Internet das Coisas** voltado para integração de objetos heterogêneos. A implementação dele foi realizada na linguagem de programação Java e utilizou-se o barramento de serviços *Fuse Enterprise Service Bus*, o qual implementa algumas funcionalidades do OSGi. Porém, mesmo realizando a integração de fontes de dados heterogêneas, esse estudo não é centrado na área de cidades inteligentes, o que faz com que muitos requisitos dessa área de estudo não sejam atendidos.

3.1.2.3 Plataforma P6

O artigo de Blackstock et al. (2010), também é centrado na interoperabilidade de objetos e propõe a plataforma de **Internet das Coisas** *MAGIC Broker 2*. Ela foi implementada utilizando a tecnologia OSGi e seus módulos são divididos em seis categorias: *Presentations* que é a porta de entrada e saída de dados na arquitetura; *Subscribers* que implemeta as várias modalidades de comunicação (SMS, REST, etc) dos assinantes de dados; *Publisher* responsável pelos fornecedores de dados; *Core* que possui os módulos que realizam o controle das demais partes da arquitetura; *Cache* que fornece armazenamento em memória dos dados acessados mais frequentemente; *DataStore* responsável pela

persistência de dados.

3.1.3 Trabalhos que combinam internet das coisas e cidades inteligentes

Nesta subseção são descritos os estudos cujas soluções combinam internet das coisas e cidades inteligentes.

3.1.3.1 Plataforma P7

Na dissertação de mestrado de Tomas (2014), é apresentada a especificação, projeto e implementação de uma arquitetura para o ambiente de cidades inteligentes que possibilita o desenvolvimento de soluções baseadas em **Internet das Coisas**. Ela foi desenvolvida utilizando a tecnologia OSGi na sua implementação Equinox e, de um ponto de vista lógico, encontra-se dividida em quatro elementos, que são: Módulo de Acesso e Comunicação (MAC), Módulo de Persistência de Dados (MPD), Módulo de Gerenciamento de Recursos (MGR) e Módulo de Gerenciamento e Distribuição de Dados (MGDD) .

Apesar dessa arquitetura proposta em Tomas (2014), incorporar uma grande quantidade de requisitos relacionados a cidades inteligentes, ela não aborda alguns pontos importantes. Não existe qualquer mecanismo para a realização da segurança dos dados, pelo contrário, no MGDD um consumidor é capaz de se registrar em qualquer canal de dados. Além disso, a arquitetura não compreende o requisito de monetização e, por fim, ela não apresenta um processo de fluxo de transformação extensível.

3.1.3.2 Plataforma P8

No artigo de Borja e Gama (2014), é exposto um modelo de arquitetura baseada no conceito de barramento de serviços objetivando facilitar a integração de fontes de dados heterogêneas existentes em um ambiente urbano. Assim, essa plataforma almeja simplificar o desenvolvimento de novas aplicações utilizando os dados desse conjunto de fontes.

Dessa forma, esse *middleware* foi implementado através de um ESB para se utilizar das capacidades de orquestração oferecidas por esse tipo de arquitetura. Além disso, ele baseia-se na ideia de componentes *Adapters* que são responsáveis por integrar os recursos provedores de dados à plataforma. As fontes de dados que podem ser agregadas por meio

dos *Adapters* vão além de dispositivos físicos, dando a possibilidade de inclusão de recursos como Twitter e Facebook.

Entretanto, a arquitetura do trabalho de Borja e Gama (2014), não possui um processo de fluxo de transformação de dados bem definido. Porém, é mencionado que o módulo *Pooler* realiza algumas transformações, mas não se aborda os detalhes de como ela é realizada e os passos que são seguidos nesse processo. Além disso, a plataforma definida utiliza apenas o banco de dados NoSQL impedindo que desenvolvedores usem outro tipo de local de armazenamento de dados. Ademais, o *middleware* não possui uma política de segurança para os dados trafegados em seu interior o que pode ocasionar no uso de dados sigilosos por aplicações que não deveriam ter a capacidade de acessá-los.

3.1.3.3 Plataforma P9

No estudo de Andreini et al. (2011), é proposta uma arquitetura focada na geolocalização dos dispositivos para obter um mecanismo eficiente de recuperação de serviços. Nela cada objeto é identificado através de um nome que é usado para acessá-lo e consultar serviços. Além disso, ela é inspirada nos princípios de orientação a serviços e usa DHT para dar suporte a geolocalização. Contudo, o trabalho é limitado apenas a questões de localização geográfica dos objetos inteligentes.

3.1.3.4 Outras plataformas

Fiware (**P10**) é uma plataforma que possibilita a criação de um ecossistema sustentável e aproveita-se das oportunidades de integração de novas tecnologias a *Internet*. Ela é baseada em *cloud*, busca facilitar o desenvolvimento de aplicações inteligentes e realiza análise de *Big Data* (FIWARE, 2011).

Octoblu (**P11**) é uma plataforma que se propõe a conectar qualquer coisa e tem segurança como seu princípio, o que a leva a utilizar encriptação e um modelo de permissões para acesso aos dados. Ela também é baseada em *cloud* e permite o processamento de fluxo de dados, porém não especifica como tal é realizado (OCTOBLU, 2014).

Carriots (**P12**) é um plataforma que trabalha baseada em serviços, projetada para trabalhar no ambiente de Internet das Coisas e possibilitar a interação *Machine-to-Machine* (M2M). Ela se propõe a coletar, combinar e armazenar qualquer tipo de dado originado dos variados dispositivos espalhados no ambiente e disponibilizá-los para que novas aplicações sejam criadas a partir deles. Assim, a plataforma pode ser usada para

coletar dados dos sensores implantados nas cidades que servirão de base para a criação de aplicações que gerenciem os serviços urbanos, como por exemplo: gerenciamento de resíduos, estacionamentos, tráfego, iluminação, etc. (CARRIOTS, 2011).

A plataforma **Digi** (**P13**) também utiliza uma abordagem em *cloud* e a mesma está focada em integrar qualquer tipo de dispositivo e dado. Ela é escalável, possibilita uma integração simples e se preocupa com a segurança das informações trafegadas. No âmbito de cidades inteligentes, a mesma pode ser aplicada para realizar a comunicação dos veículos responsáveis pela segurança pública, na gestão de água e dos resíduos, no processo de iluminação inteligente das ruas, melhoria da segurança, gerenciamento de estacionamentos e do tráfego urbano (DIGI, 2013).

Kaaproject (**P14**) é uma plataforma altamente flexível e escalável criada para permitir o desenvolvimento, gerenciamento e integração de *software* relacionado ao espaço de Internet das Coisas. Além disso, ela é projetada para ser robusta, fácil de usar e garantir a integridade dos dados trafegados em seu interior. Essa plataforma também fornece suporte a coleta e armazenamento dos dados recebidos e tem um sistema de entrega de notificações que são enviadas às aplicações inscritas para recebê-las (CYBERVISION, 2014).

A **Prismtech** (**P15**) é uma plataforma inteligente que combina compartilhamento, entrega e análise de dados oriundos dos dispositivos heterogêneos espalhados no ambiente. Essa plataforma é eficiente, segura e permite o compartilhamento de dados em tempo real. Além disso, a mesma também permite a interoperabilidade independente das características dos sensores, sistemas embarcados, dispositivos móveis, servidores empresariais, computadores pessoais e qualquer outro tipo de dispositivo computacional que possa compartilhar dados (PRISMTECH, 2014).

Seecontrol (**P10**) permite conectar, analisar, controlar e gerenciar facilmente dispositivos, objetos e aplicações. Além do mais, ela é a baseada em *cloud*, provê escalabilidade, garante a segurança dos dados integrados e suporta os principais protocolos usados nos dispositivos e padrões de Internet das Coisas (SEECONTROL, 2014).

Sofia2 (**P17**), surgiu a partir da plataforma descrita na Seção 3.1.1.2, ela é um *middleware* que permite a interoperabilidade dos diversos dispositivos e sistemas existentes. Além disso, possibilita a coleta de informações do mundo real e disponibilização das mesmas para a criação de aplicações inteligentes. Ela também incorpora requisitos como segurança, análise de dados, armazenamento *Big Data*, independência de protocolo de comunicação, modularidade, extensibilidade e escalabilidade (INDRA, 2014).

3.2 Conclusões da revisão exploratória

A Tabela 3 traz um resumo das características gerais dos estudos analisados nesta revisão exploratória. Além disso, a Tabela 4 mostra uma síntese dos requisitos atendidos por cada uma das plataformas propostas nos trabalhos considerados neste estudo.

Tabela 3: Tabela características gerais dos estudos analisados.

Trabalho	**Ambiente**	**Aborda *cloud***	**Outras observações**
P1	CI	Não	Sem detalhes
P2	CI	Não	Sem detalhes
P3	CI	Sim	Sem detalhes
P4	IoT	Não	Utiliza OSGi
P5	IoT	Não	Utiliza ESB
P6	IoT	Não	Utiliza OSGi
P7	CI e IoT	Sim	Utiliza OSGi
P8	CI e IoT	Não	Utiliza ESB
P9	CI e IoT	Não	Utiliza DHT
P10	CI e IoT	Sim	Utiliza Java
P11	CI e IoT	Sim	Sem detalhes
P12	CI e IoT	Sim	Sem detalhes
P13	CI e IoT	Sim	Sem detalhes
P14	CI e IoT	Não	Utiliza Java
P15	CI e IoT	Sim	Sem detalhes
P16	CI e IoT	Sim	Sem detalhes
P17	CI e IoT	Sim	Utiliza Java

Tabela 4: Requisitos atendidos pelos estudos analisados.

Requisito	**Trabalho(s)**
Obter dados de fontes heterogêneas	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16 e P17
Criação de novos serviços	P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16 e P17
Suportar agregação de dados	P1, P2, P4, P5, P7, P8, P10, P11, P12 e P17
Processo de transformação de dados bem definido e extensível	-
Permitir extração de conhecimento	P8, P10, P11, P13, P16 e P17
Abordagem modular	P2, P3, P4, P5, P6, P7, P8, P10, P12 e P17
Abordagem plugável	P4, P6, P7, P8, P10 e P17
Manter a privacidade dos dados	P1, P3, P10, P11, P12, P13, P14, P15, P16 e P17
Monetização	P8 e P17

Por meio da leitura dos trabalhos anteriores, pode-se inferir algumas conclusões no

que diz respeito à construção de arquiteturas que permitem a inclusão e gerenciamento de fontes heterogêneas de dados. A utilização de OSGi é uma prática recorrente nas soluções que apresentam detalhes de sua implementação. Dessa forma, os trabalhos **P4**, **P6** e **P7**, fazem uso de OSGi para a construção das plataformas de maneira modular e com o intuito de usufruir das facilidades de manutenibilidade e gerenciabilidade proporcionadas por ele.

Outra estratégia que também apareceu nas soluções foi a utilização do *Enterprise Service Bus*, em virtude do mesmo proporcionar alta disponibilidade, distribuição e orquestração de serviços, assim, os estudos **P5** e **P8** fazem uso dessa abordagem em suas plataformas.

Da mesma forma, o uso de uma abordagem em *Cloud* para fornecer escalabilidade ao grande volume de dados e quantidade de dispositivos é muito utilizada nas plataformas analisadas, sendo que nove delas (**P3**, **P7**, **P10**, **P11**, **P12**, **P13**, **P15**, **P16** e **P17**) empregam essa estratégia.

O estudo **P1** mesmo sendo centrado em espaços urbanos traz apenas uma descrição da proposta de arquitetura, impossibilitando a identificação dos módulos que a compõem e a atestação de seu funcionamento. Por outro lado, as plataformas **P2**, **P3** e **P9** apesar de especificarem bem a solução são muito restritivas quanto aos requisitos abordados, o primeiro é focado principalmente na geolocalização dos objetos inteligentes, o segundo é centrado na otimização de recursos através do uso de *cloud computing* e o último engloba apenas o monitoramento de espaços públicos com o uso de sensores.

Os trabalhos **P4**, **P5** e **P6** abordam a interoperabilidade no ambiente de **Internet das Coisas**. Dessa forma, os mesmos não têm enfoque na área de cidades inteligentes e não abordam alguns requisitos importantes, como por exemplo, inclusão de dados da Web. Porém, eles foram considerados nesta revisão exploratória por também se proporem a trabalhar com dados heterogêneos.

Dentre o conjunto de trabalhos analisados, dez têm o objetivo de interoperabilidade em cidades inteligentes. O primeiro deles é a dissertação **P7**, porém esse estudo não incorpora segurança e monetização e o outro é o artigo **P8** que também não trata a questão da privacidade dos dados e seu banco de dados é restrito ao NoSQL. As informações das demais plataformas (**P10**, **P11**, **P12**, **P13**, **P14**, **P15**, **P16** e **P17**) encontram-se disponibilizadas através de páginas *Web*, as quais não detalham bem as mesmas. Ainda assim foi possível identificar que ambas estão preocupadas em proporcionar escalabilidade e manter a segurança dos dados integrados.

Por fim, nenhum dos trabalhos analisados nesta revisão exploratória definem um fluxo de transformação bem definido e extensível para o tratamento dos dados em suas propostas de arquitetura, requisito esse que é o foco principal deste trabalho de mestrado. Nesse fluxo determina-se os passos de processamento necessários para o tratamento do conjunto de dados integrados à plataforma, a fim de entregá-los da melhor maneira possível para que eles sejam utilizados na criação de novos sistemas. Além disso, a extensibilidade das etapas definidas para o fluxo possibilita a adequação do processamento desses passos para que eles possam ser realizados conforme as características e restrições de cada tipo de dado existente nos "ecossistemas" de uma cidade inteligente.

4 Uma plataforma extensível para a transformação de fluxo de dados heterogêneos em cidades inteligentes

É importante que o gerenciamento dos dados gerados nos dispositivos provedores de dados em uma cidade inteligente seja realizado em um mesmo local para se obter um melhor funcionamento dos diversos setores da cidade, permitindo a troca de informações entre eles e ajudando no processo de tomada de decisão. Porém, essa integração de dados não é uma tarefa trivial em razão da heterogeneidade dos dispositivos (SU; LI; FU, 2011; WU et al., 2011; GAMA; TOUSEAU; DONSEZ, 2012; ZANELLA et al., 2014; TOMAS, 2014) que utilizam diferentes protocolos de comunicação, produzem fluxos de dados em múltiplos formatos e possuem diferentes características. Para resolver isso, este trabalho propõe uma plataforma que permite a unificação do gerenciamento dos dados desses diversos dispositivos. Além disso, o trabalho define um fluxo de transformação desses dados na plataforma, o que provê aos desenvolvedores um melhor entendimento de como tratá-los e da responsabilidade de cada passo do processamento. O fluxo ainda possui a característica de extensibilidade permitindo que implementações diferentes dos passos de transformação sejam geradas e "plugadas" na plataforma.

4.1 Requisitos da plataforma

Partindo da ideia da criação de uma plataforma que permitisse a integração, transformação, armazenamento, agregação e disponibilização dos diferentes tipos de dados existentes no ambiente urbano. E, principalmente, que a mesma fosse extensível do ponto de vista do processo de transformação de dados em seu interior, foi pensado um conjunto de requisitos para que tal plataforma atendesse a essas necessidades. Estes requisitos são:

- R1: obtenção de dados de diferentes tipos de recursos;
- R2: criação de novos serviços a partir dos dados gerados pela plataforma;
- R3: agregação de dados;
- R4: seguir uma abordagem modular;
- R5: ser plugável;
- R6: ter um processo de transformação de dados bem definido e extensível;
- R7: permitir extração de conhecimento a partir dos dados;
- R8: manter a privacidade dos dados trafegados;
- R9: monetização.

4.1.1 R1: obtenção de dados de diferentes tipos de recursos

É imprescindível que uma plataforma de *Smart City* considere a possibilidade de obtenção de dados de diferentes recursos. Uma cidade inteligente tem um ambiente bastante diversificado, no qual os dispositivos envolvidos são os mais variados e possuem as mais diversas tecnologias. Portanto, é essencial que uma plataforma que tem o intuito de gerenciar os dados de uma cidade inteligente esteja preparada para receber os mesmos independentemente dos dispositivos provedores deles.

4.1.2 R2: criação de novos serviços a partir dos dados gerados pela plataforma

Também é importante que os desenvolvedores tenham suporte para criar novos serviços a partir dos dados oriundos da plataforma. Assim, é essencial que a mesma dê suporte para que qualquer programador possa se utilizar dos dados integrados a ela para realizar o desenvolvimento de novas aplicações que melhorem os serviços de uma cidade, beneficiando dessa forma os usuários de tais serviços e melhorando a qualidade deles.

4.1.3 R3: agregação de dados

O sistema urbano é bastante complexo devido a grande variedade de domínios (transporte, elétrico, hídrico, saneamento, segurança, saúde, etc.) envolvidos na vida dos cidadãos. Desse modo, é importante que a plataforma possibilite o agrupamento de dados

de diferentes domínios ou de um mesmo domínio para que se possa realizar agregação de dados, uma vez que, esses domínios muitas vezes devem trabalhar de maneira integrada para obter um melhor funcionamento. Assim, a plataforma deve permitir que os desenvolvedores possam utilizar em conjunto os dados de diferentes fontes.

4.1.4 R4: seguir uma abordagem modular

Com essa abordagem é possível estabelecer responsabilidades específicas para cada parte da plataforma. Dessa forma, cada módulo possuirá seu conjunto de funções básicas, aumentando a organização e facilitando o entendimento dos módulos implementados.

4.1.5 R5: ser plugável

Com o passar do tempo pode fazer-se necessário a inclusão de novas funcionalidades a uma plataforma de cidades inteligentes. Assim, é importante que um *software* desse tipo seja "plugável" para facilitar a inserção dessas novas características a mesma. Além disso, com a abordagem "plugável" também se torna simples o processo de remoção de módulos da mesma, o que facilita caso se queira descontinuar um módulo que esteja em execução.

4.1.6 R6: ter um processo de transformação de dados bem definido e extensível

Os fluxos de dados originados nas fontes que se encontram distribuídas pelo ambiente das cidades devem ser continuamente processados na tentativa de fornecer respostas adequadas para eventos acontecidos no ambiente e para prover informações que ajudem na criação de novas aplicações. Assim, os sistemas que se propõem a processar fluxos de informação as filtram, combinam e agregam para produzir novos dados como saída ajudando o entendimento do problema através das informações processadas (CUGOLA; MARGARA, 2012).

Dessa forma, é necessário que exista um conjunto de módulos responsáveis por transformar os dados de um recurso ao longo do seu fluxo na plataforma. Assim, é importante que os mesmos tenham responsabilidades bem definidas dentro da mesma e que se estabeleça um conjunto de passos para preparação dos dados, terminando com a disponibilização deles para o uso fora da plataforma.

Além disso, conforme mostrado na Figura 10 os passos definidos nesse processo de

transformação devem ser extensíveis permitindo que o processamento possa ser adequado para trabalhar de acordo com as especificidades e restrições de cada tipo existente no ecossistema de dados.

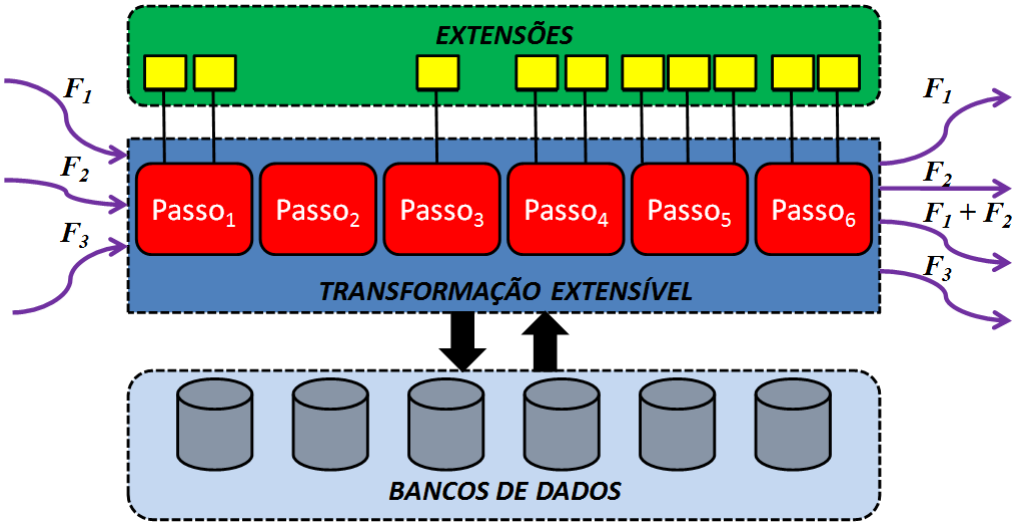


Figura 10: Processo de transformação extensível

Com isso, também se tem a possibilidade de existirem diferentes tratamentos para um mesmo fluxo de dados, como exemplificado na Figura 11. Nela existe um fluxo de dados ($\boldsymbol{F_1}$) que em um primeiro momento (**A**) possui apenas um conjunto de tratamentos, representados pelas extensões $\boldsymbol{T_1}$, os quais geram como saída $\boldsymbol{F_1'}$. No entanto, em um segundo momento (**B**) se faz necessário que o fluxo $\boldsymbol{F_1}$ seja tratado de maneira diferente. Para que isso seja possível, as extensões responsáveis pelo novo tratamento $\boldsymbol{T_2}$ são desenvolvidas e "plugadas" nos seus respectivos passos de transformação e, assim, passa a existir um novo tipo de saída ($\boldsymbol{F_1''}$).

Outro fator que torna necessária a existência da extensibilidade é que com o passar do tempo novos fluxos de dados irão surgir e precisarão ser adicionados e transformados pela plataforma, como ilustrado na Figura 12. Conforme mostrado na Figura 12A, a plataforma trabalhava com o fluxo $\boldsymbol{F_1}$, o qual era transformado pelas extensões $\boldsymbol{TF_1}$ e gerava como saída $\boldsymbol{F_1'}$. Todavia, em determinado instante (**B**) aparece um novo fluxo ($\boldsymbol{F_2}$) que precisa ser suportado pela plataforma, porém os dados do mesmo têm características diferentes dos já incorporados. Desse modo, é imprescindível a realização de um processamento que esteja de acordo com as peculiaridades (formato, protocolos de comunicação, etc.) desses dados, o que é atendido através das extensões $\boldsymbol{TF_2}$, as quais concebem a saída $\boldsymbol{F_2'}$.

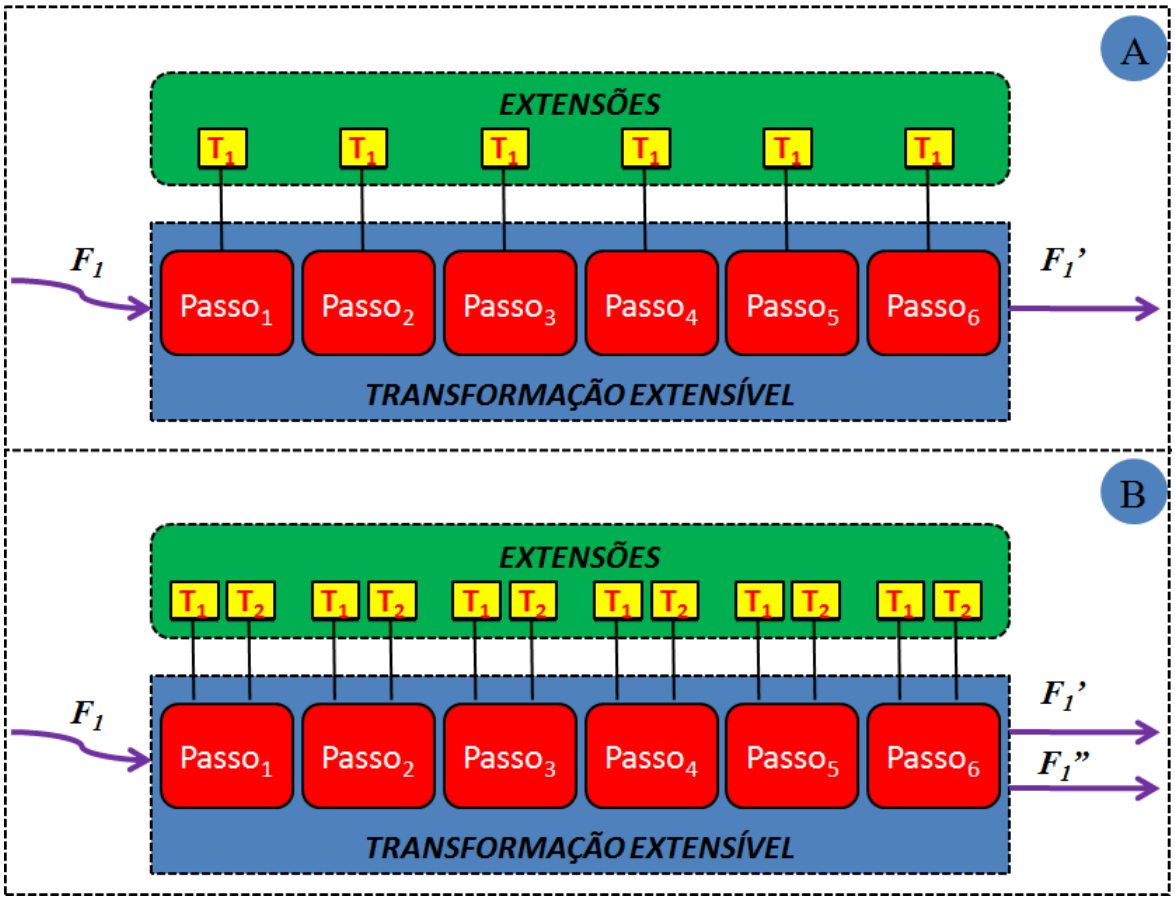


Figura 11: Criação de um novo tratamento para um fluxo de dado existente

4.1.7 R7: permitir extração de conhecimento a partir dos dados

Um requisito importante que deve ser compreendido pela plataforma é a possibilidade de realizar extração de conhecimento a partir dos dados originados nas variadas fontes que se encontram espalhadas pelo ambiente urbano. Desse modo, é imprescindível que um módulo do fluxo permita aos desenvolvedores a possibilidade de criação de suas próprias estratégias de extração de conhecimento para processar os dados que trafegam na mesma.

4.1.8 R8: manter a privacidade dos dados trafegados

Muitos dados de um ambiente urbano são de natureza crítica o que torna necessário o estabelecimento de políticas de privacidade para o acesso dos mesmos. Um exemplo disso são os dados de posicionamento geográfico das viaturas ao longo de uma cidade, os quais não podem ser acessados por qualquer indivíduo, pois o conhecimento deles facilitaria as ações de criminosos. Por isso, é essencial que seja criado na plataforma um sistema de permissões, no qual seja preciso da permissão ao tipo de dados para se ter acesso a ele.

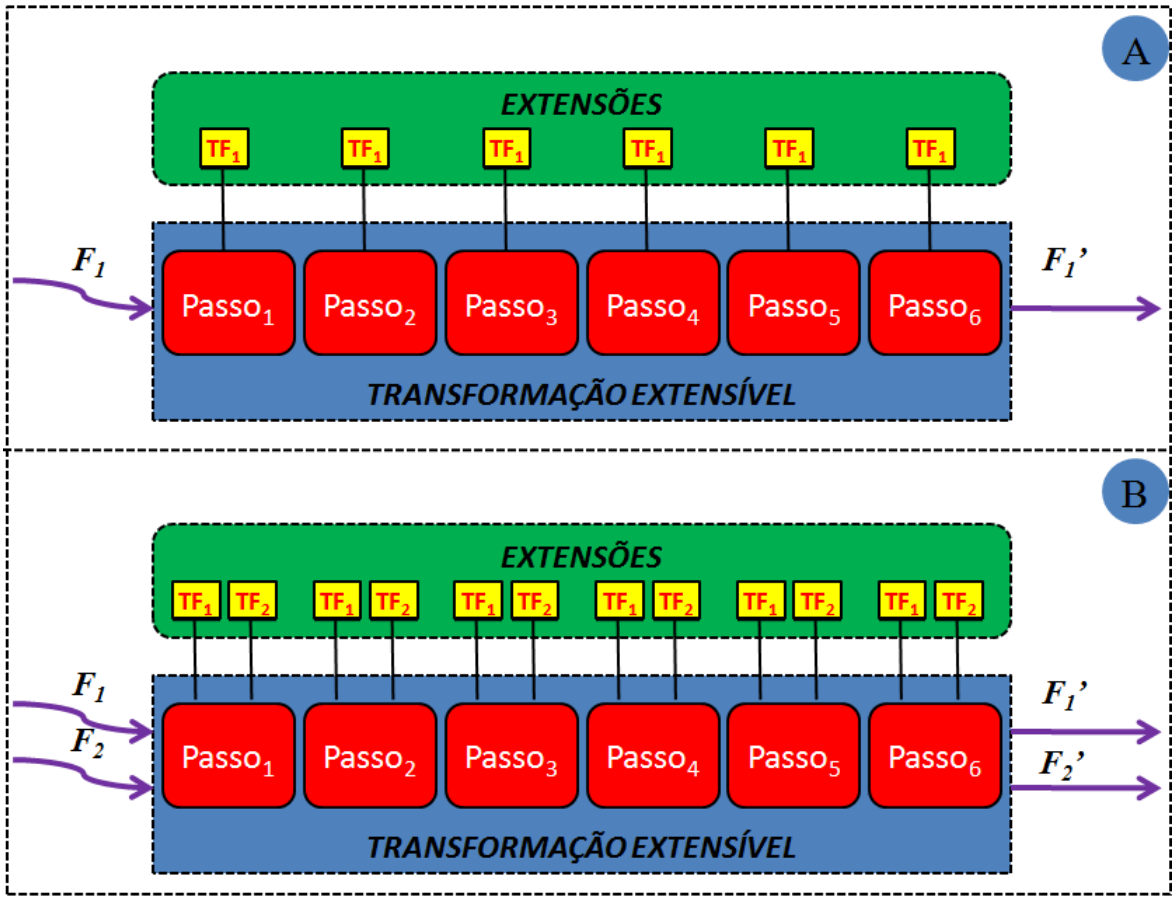


Figura 12: Novo fluxo de dados adicionado à plataforma

4.1.9 R9: monetização

É primordial em uma plataforma de cidades inteligentes que os desenvolvedores tenham suporte para comercializar os dados criados nos sistemas desenvolvidos pelos mesmos. Desse modo, uma plataforma desse tipo deve contabilizar a quantidade de mensagens e os consumidores das mesmas possibilitando uma posterior cobrança para esses serviços.

4.2 Arquitetura proposta

Na Figura 13 é exibida a plataforma proposta neste estudo. Assim, pode-se perceber que a mesma foi pensada para suportar o recebimento de dados de diferentes fontes existentes nas cidades. Além disso, essa ilustração mostra que a plataforma possui um conjunto de **módulos padrões** que são os módulos responsáveis por definir os passos do fluxo de transformação de dados na mesma. Ademais, a solução proposta também possui um conjunto de **módulos auxiliares**, os quais incrementam funcionalidades importantes a ela.

Ainda na Figura 13 nota-se que após seu tratamento na plataforma os dados são disponibilizados para a criação de novas aplicações, o que permite que novas soluções de *software* que tenham o intuito de melhorar os serviços públicos das cidades sejam desenvolvidas e liberadas à população. Por fim, é possível reparar que a plataforma é flexível quanto ao banco de dados a ser usado e suporta a utilização de diferentes tipos desses.

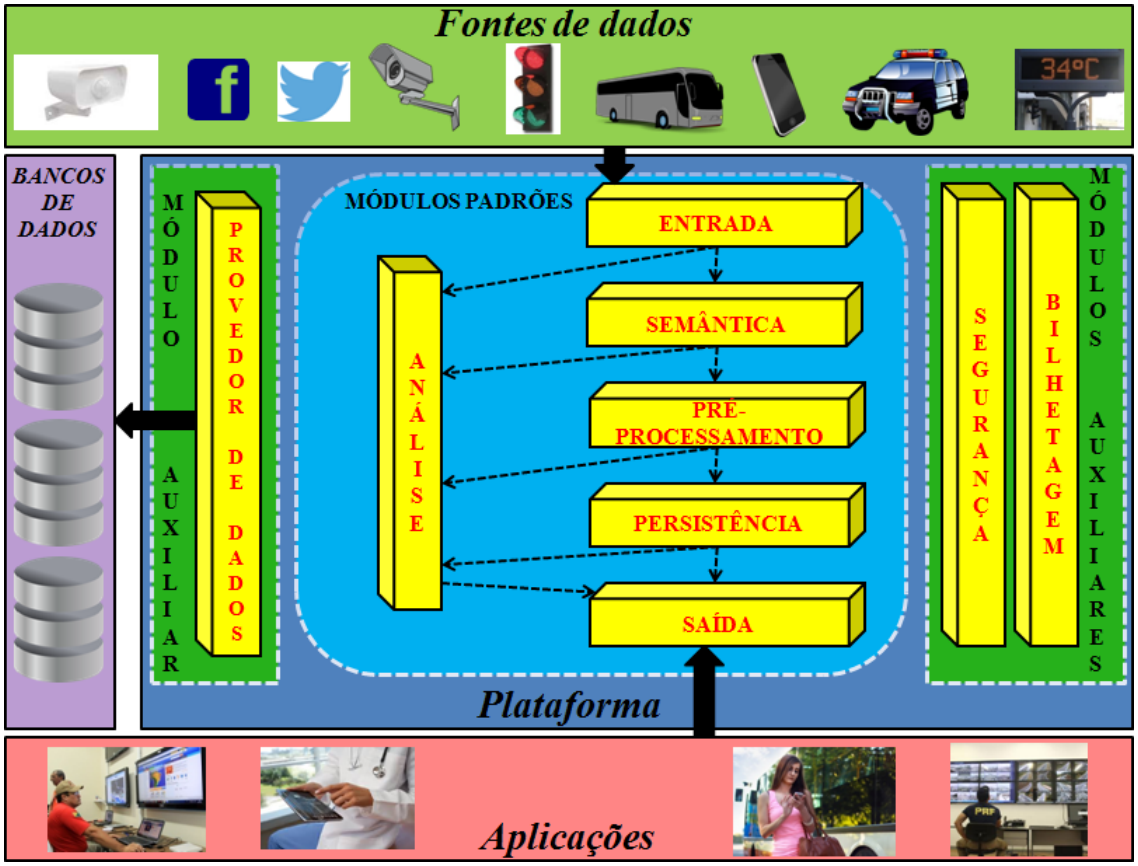


Figura 13: Plataforma proposta

4.2.1 Visão dos módulos

Como mostrado na Figura 14, para atender o R5 os **módulos padrões** da arquitetura definem pontos de extensão permitindo que implementações diferentes deles sejam geradas e "plugadas" nesta solução. Assim, para adicionar um novo recurso na arquitetura é necessário que seja feita a extensão dos **módulos padrões** gerando **módulos específicos** que sejam capazes de trabalhar com o(s) tipo(s) de dados do novo recurso que se queira inserir e os quais representam etapas de processamento de dados dentro da arquitetura.

Além disso, é possível reparar na Figura 14 que existe um módulo responsável pela troca de mensagens entre os **módulos padrões** da arquitetura. O chamado *Event Ad-*

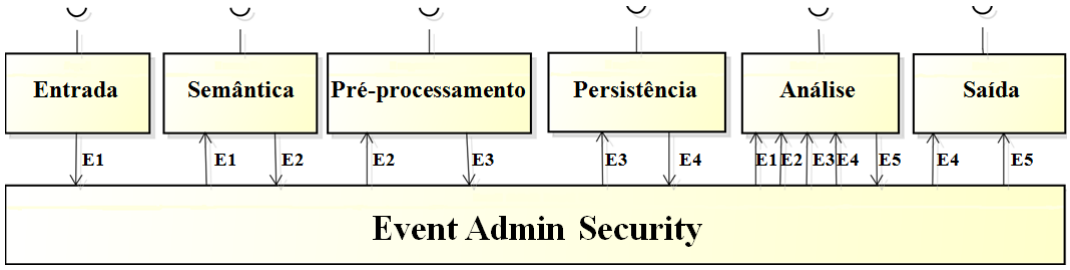


Figura 14: Arquitetura proposta

min Security que implementa um requisito adicional de segurança relacionado ao acesso das mensagens. Essa capacidade garante que somente os **módulos padrões** recebam mensagens diretamente do *Event Admin Security* e impede que **módulos específicos** alternem o fluxo padrão das mensagens, além de viabilizar a adição de requisitos, tais como privacidade e integridade dos dados.

Todos os **módulos padrões** da plataforma executam um conjunto básico de atividades. Primeiro, eles recebem um conjunto de dados. Em seguida, verificam os **módulos específicos** interessados no tipo de dado recebido e repassa-os somente aos que têm a permissão de acesso a eles. Assim, esses **módulos específicos** realizam o processamento e retornam os dados para o **módulo padrão** que, finalmente, os publica por meio do *Event Admin Security*. Nesse processo de publicação de dados, o **módulo padrão** envia ao *Event Admin Security* os dados gerados a partir dos processamentos realizados nos **módulos específicos** e algumas características dos mesmos, tais como: o tipo de dado que está sendo enviado, se ele é público ou pago, etc. Por sua vez, o módulo *Event Admin Security* entrega essas informações ao próximo passo do fluxo que as utiliza no seu conjunto de atividades.

Esta arquitetura é composta por seis **módulos padrões** que juntos atendem o R6, os quais são:

- *Entrada* – é por meio desse módulo que o R1 é atendido, pois ele tem por responsabilidade integrar diferentes recursos de cidades inteligentes à arquitetura proposta. Assim, o mesmo funciona como canal de entrada para os dados provenientes dos mais variados dispositivos que compõem o ambiente de uma *smart city*. Por fim, **Entrada** gera o evento 1 **(E1)** e publica os dados recebidos das suas respectivas fontes;

- *Semântica* – esse módulo recebe os dados do **E1** e tem a tarefa de representá-los da maneira mais oportuna para que eles possam ser tratados dentro da arquitetura.

Além disso, nesse módulo é aconselhável que os dados recebidos passem a ser representados no paradigma orientado a objetos, facilitando a manipulação deles devido ao elevado nível de abstração proporcionado pelo uso desse paradigma. No entanto, o desenvolvedor pode escolher outra forma de representação caso ache-a mais adequada. Após realizado o processo de representação, esse módulo cria o evento 2 **(E2)** e publica seus dados;

- *Pré-processamento* – a atribuição desse módulo é receber o **E2** gerado pelo módulo **Semântica**. Além disso, ele implementa um processo de preparação realizando uma limpeza dos dados, na qual é feito o tratamento de dados incompletos e ruidosos, eliminada a duplicação deles e selecionado os dados centrais ao problema. Por fim, é criado o evento 3 **(E3)** e os dados são publicados;

- *Persistência* – esse módulo tem a função de receber o **E3** e a principal responsabilidade dos módulos "plugados" a ele é armazenar os dados fazendo uso da estratégia de armazenamento que desejar. Vale ressaltar que cada **módulo específico** de **Persistência** implementa sua política de armazenamento, com a decisão de armazenar os dados em banco de dados ou memória, a escolha do tipo de banco de dados, entre outros. Finalmente, **Persistência** cria o evento 4 **(E4)** e o publica;

- *Análise* – esse módulo implementa o R7, assim, ele tem o papel de receber todos os eventos gerados pelos módulos mencionados anteriormente. Dessa forma, seus **módulos específicos** processam esses dados e quando seus algoritmos conseguem inferir algum conhecimento relevante o evento 5 **(E5)** é publicado;

- *Saída* – nesse módulo é atendido o R2, dessa forma, ele é encarregado de receber o **E4** criado no módulo **Persistência** e o **E5** criado no módulo **Análise**. Finalmente, cada módulo **Saída** específico provê acesso aos dados da arquitetura da *smart city* permitindo, assim, que novas aplicações que necessitem utilizar tais dados possam ser criadas.

4.2.2 Visão de etapas

Como definido na Figura 15, fazendo uso desses seis módulos e utilizando um cenário simplificado onde existe apenas um **módulo específico** "plugado" a cada **módulo padrão**, o fluxo extensível básico se dá através de um conjunto de 14 etapas, as quais são:

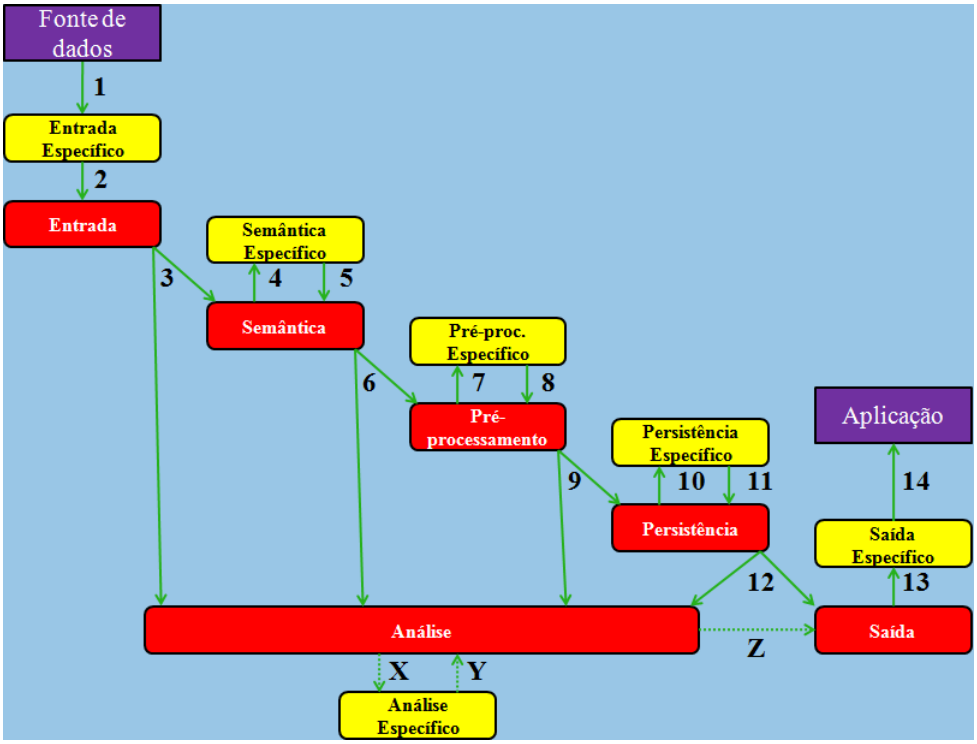


Figura 15: Etapas do fluxo de transformação de dados

- Etapa 1: Os dados são enviados da fonte para o módulo **Entrada Específico**;
- Etapa 2: **Entrada Específico** encaminha os dados recebidos das fontes para o módulo **Entrada** padrão;
- Etapa 3: Os dados são repassados de **Entrada** para o próximo passo de fluxo (**Semântica**);
- Etapa 4: **Semântica** encaminha os dados ao **Semântica Específico**, até esse momento os mesmos ainda permanecem no formato recebido da fonte;
- Etapa 5: Nos **módulos específicos** de **Semântica** ocorrem as primeiras transformações de dados, pois é nesse instante que eles passam a ser representados no formato escolhido pelo desenvolvedor do sistema específico. Após isso, os mesmos são retornados para **Semântica**;
- Etapa 6: **Semântica** publica os dados que já estão utilizando a representação escolhida pelo desenvolvedor;
- Etapa 7: **Pré-processamento** entrega os dados ao **Pré-processamento Específico**;

- Etapa 8: Acontece um processo de filtragem, onde o conjunto de dados é submetido a uma limpeza e uma seleção. Logo, os dados filtrados retornam ao módulo **Pré-processamento**;
- Etapa 9: **Pré-processamento** envia os dados filtrados;
- Etapa 10: Os dados são repassados de **Persistência** para **Persistência Específico**;
- Etapa 11: O **Persistência Específico** realiza o processo de armazenamento em memória ou banco de dados. Em seguida, o último estado dos dados alvos são retornados para **Persistência**;
- Etapa 12: **Persistência** encaminha os dados ao próximo passo do fluxo;
- Etapa 13: **Saída** repassa os dados para o **Saída Específico**;
- Etapa 14: **Saída Específico** torna os dados acessíveis às aplicações. Nessa etapa o programador também tem a liberdade de escolher a estratégia de disponibilização dos dados.

Além disso, a Figura 15 também mostra um fluxo de descoberta de conhecimento. Nele o módulo **Análise** recebe todos os dados entregues nas etapas 3, 6, 9 e 12 do fluxo básico. A cada momento que um conjunto de dados é recebido, **Análise** repassa-os ao **Análise Específico** (etapa X) que a todo instante realiza processamentos na tentativa de identificar algum conhecimento relevante ao problema alvo. Logo, realizada a identificação de algo significativo, **Análise Específico** retorna as informações para **Análise**. Por sua vez, esse último repassa-as ao módulo **Saída**. Por fim, para esse fluxo foram utilizadas as letras X, Y e Z porque não se tem como prever em que momento cada etapa dessa será executada no fluxo de transformação.

4.2.3 Outros módulos

A arquitetura proposta possui dois **módulos auxiliares** cujas funcionalidades são utilizadas por todos os **módulos padrões** do fluxo de transformação de dados. A Figura 16 exibe esses dois módulos e a forma como um **módulo padrão** se relaciona com eles. Além disso, existe um terceiro **módulo auxiliar** (Provedor de Dados) que pode ser usado por qualquer um dos **módulos específicos** "plugados" na plataforma. Esses módulos e suas funcionalidades são:

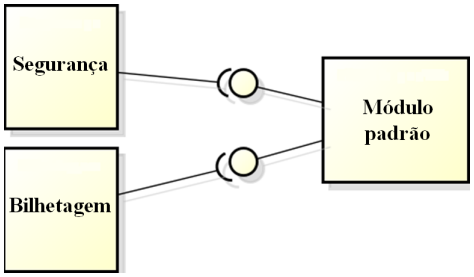


Figura 16: Relacionamento entre Segurança, Bilhetagem e módulos padrões da arquitetura

- *Segurança* – nesse módulo é atendido o R8. Desse modo, ele é responsável por gerenciar o acesso aos dados trafegados no *Event Admin Security*. Esse módulo analisa cada **módulo específico** verificando se eles possuem permissão para acessar os dados recebidos nos **módulos padrões**. Por exemplo, ao receber os dados de um evento, o **módulo padrão** solicita que o módulo de **Segurança** verifique cada **módulo específico** plugado a ele, identificando quais deles têm permissão para receber o tipo de dado recebido nesse evento. Por fim, o **módulo padrão** só repassa os dados para os **módulos específicos** cujo acesso tenha sido liberado pelo módulo de **Segurança**;

- *Bilhetagem* – este módulo atende o R9. Assim, ele contabiliza as mensagens acessadas pelos **módulos específicos** para possibilitar uma posterior cobrança relacionada ao acesso dos dados.

- *Provedor de dados* – tem a responsabilidade de realizar o armazenamento e gerenciamento dos dados tornando-os acessíveis aos **módulos específicos**. Esse módulo define uma interface padrão que pode ser estendida por um programador para que ele gere seu **Provedor de dados** específico e o mesmo é utilizado para buscar, adicionar, alterar e remover dados.

Também é possível gerar **módulos adicionais** caso eles sejam necessários para facilitar a implementação dos **módulos específicos** de algum recurso. Um exemplo comum que demonstra essa capacidade é quando o programador cria um módulo que modele os dados recebidos em classes para facilitar a manipulação deles em seu fluxo dentro da arquitetura. Dessa forma, os **módulos específicos** do recurso importariam esse módulo para utilizar as classes definidas nele com o intuito de facilitar a manipulação dos dados recebidos por eles e para que não ocorra duplicação de definição de classes, uma vez que, certamente o desenvolvedor teria que definir tais classes em mais de um dos **módulos específicos**.

Além disso, não é exigida para um recurso a implementação de todos os **módulos específicos** para todos os **módulos padrões** definidos na arquitetura. Isso foi pensado para o caso que o desenvolvedor não necessite de todos os módulos do fluxo de dados. Por exemplo, se um recurso já fornece os dados para o módulo **Entrada** da maneira que se quer usar na arquitetura então não faz sentido para o programador gerar um módulo de **Pré-processamento** que não terá utilidade. Assim, a arquitetura está preparada para repassar os dados para o próximo módulo do fluxo de dados quando não for encontrado em um **módulo padrão** implementações específicas responsáveis para o tipo de dado recebido.

Por fim, o R4 é atendido através da implementação da solução proposta por meio de módulos que fornecem as funcionalidades da plataforma, os quais são de quatro tipos: **módulos padrões**, **módulos específicos**, **módulos auxiliares** e **módulos adicionais**.

4.2.4 Agregação de dados

Como definido no R3, a agregação de dados busca combinar dados oriundos de diferentes domínios do meio urbano ou de diferentes fontes existentes em um mesmo domínio para fornecer um melhor entendimento e funcionamento do ambiente no qual as mesmas estão inseridas.

Assim, para um módulo realizar a agregação de dados basta que ele tenha o conjunto de permissões para acessar os tipos de dados que juntos formarão o novo dado composto. Dessa forma, tendo acesso ao conjunto de dados, o módulo será responsável por gerar um novo tipo de dados e disponibilizar os mesmos na plataforma. Portanto, os módulos que queiram acessar os dados agregados necessitam ter a permissão para acessar o tipo de dado gerado por ele.

Na Listagem 4.1 é exibido um exemplo de uma classe que realiza agregação de dados. Nas linhas 9 e 10 ela adiciona as permissões para os dados que juntos conceberão um novo tipo de dado composto. E, por fim, a classe define, na linha 15, o tipo de dado que é criado por ela. Assim, sempre que as classes *Entrada_A* e *Entrada_B* enviarem dados na plataforma os mesmos serão recebidos pela classe *Semântica_AB*, a qual os agrega para criar um novo tipo de dado, como mostrado na Figura 17.

```
1 public class Semantica_AB
2         extends Semantica {
3
4         //atributos e metodos omitidos
```

```
5
6           public Map<String, Boolean> getPermissoes() {
7                   Map<String, Boolean> mapPermissions =
8                                               new HashMap<String, Boolean>();
9                   mapPermissions.put("Entrada_A", true);
10                  mapPermissions.put("Entrada_B", true);
11                  return mapPermissions;
12          }
13
14          public String getTipoDeDado() {
15                  return "Semantica_AB";
16          }
17  }
```

Listagem 4.1: Classe realizando agregação de dados

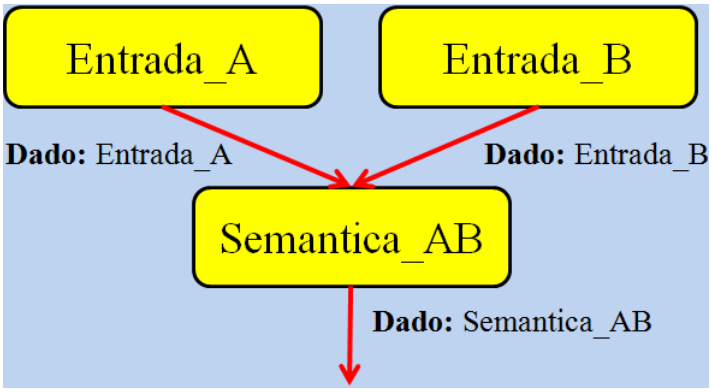


Figura 17: Exemplo de agregação de dados na plataforma

4.2.5 Padrões arquiteturais adotados

Esta seção mostra os padrões arquiteturais utilizados na plataforma proposta.

4.2.5.1 *Pipe and Filter*

A forma como os módulos funcionam se assemelha ao conceito do padrão arquitetural *Pipe and Filter*, porém eles não seguem puramente as suas definições. O *Pipe and filter* é um padrão que fornece uma estrutura para sistemas que processam fluxo de dados, nele o processamento é dividido em um conjunto de passos arranjados de forma que a saída de um elemento é a entrada para o próximo (BUSCHMANN et al., 1996; DOOLEY, 2011; HOHPE; WOOLF, 2004).

Devido a plataforma desenvolvida neste trabalho ter a função de prover uma estrutura para processamento de dados dos recursos de uma cidade, ela tem muitas semelhanças em relação ao padrão *Pipe and Filter*. No padrão é sugerido que se faça a divisão das tarefas de processamento e cada passo é encapsulado nos *filters* como ocorre nos **módulos padrões**. Além de que, na plataforma a comunicação dos **módulos padrões** se dá por meio do *Event Admin Security*, ideia semelhante ao dos *pipes*. Na plataforma também existe o conceito de fonte de dados que no caso deste trabalho são os recursos das cidades inteligentes e do disseminador de dados que representa os novos serviços gerados a partir dos dados da plataforma.

Entretanto, a plataforma se diferencia em alguns aspectos da definição do padrão *Pipe and Filter*. No padrão os *filters* são os elementos responsáveis diretamente pela realização do processamento dos dados, já na plataforma os **módulos padrões** delegam a função de processamento para os **módulos específicos** a eles plugados. Além de que, no *Pipe and Filter* os dados são passados apenas para *filters* adjacentes, porém na plataforma o módulo **Análise** recebe dados de todos os quatro módulos anteriores a ele e **Saída** recebe dados tanto de **Persistência** quanto de **Análise**. E, por fim, no padrão cada ponte de comunicação é implementada por um *pipe* diferente, no que diz respeito à plataforma, a comunicação entre os **módulos padrões** é realizada unicamente através do *Event Admin Security* e o recebimento dos dados da fonte e disseminação é realizada através da *internet*.

Por fim, por usar conceitos parecidos com o do *Pipe and Filter* a plataforma também herda alguns pontos positivos e negativos desse padrão. Os componentes são mais fáceis de serem reusados em razão da utilização de pequenos passos de processamento, a troca de um módulo é mais fácil em decorrência do baixo acoplamento e o sistema pode ser estendido e modificado mais facilmente. Por outro lado, esse padrão aumenta a sobrecarga no processo de transformação de dados deixando-o mais lento e pode acarretar em um difícil entendimento dos módulos devido sua quantidade.

4.2.5.2 *Publish/subscribe*

O padrão publish/subscribe é usado para tratar cenários onde muitos componentes dependem dos dados que constantemente mudam em um determinado lugar. Assim, faz-se necessário sincronizar os estados desses componentes notificando-os sobre as mudanças acontecidas (BUSCHMANN et al., 1996; HOHPE; WOOLF, 2004).

Na plataforma esse padrão é utilizado na tarefa de troca de mensagens realizada entre os **módulos padrões** e implementada pelo *Event Admin Security*. Dessa forma, o

módulo padrão (*subscribe*) inscreve-se para receber dados de outro **módulo padrão** (*publish*) sempre que eles forem publicados no *Event Admin Security*.

Além disso, o *publish/subscribe* também é usado no repassamento de dados entre **módulos padrões** e **módulos específicos** da plataforma. No momento em que um **módulo padrão** recebe um novo conjunto de dados, o mesmo solicita que **Segurança** verifique as permissões dos **módulos específicos** "plugados" a ele e repassa esses dados apenas aqueles que tenham se inscrito para recebê-los.

A principal vantagem do uso desse padrão é o desacoplamento, pois um assinante (*subscribe*) não precisa conhecer o publicador (*publish*) das informações e o publicador (*publish*) não precisa conhecer o assinante (*subscribe*).

4.3 Implementação

Nesta seção são mostrados detalhes da implementação da plataforma proposta, a qual foi desenvolvida utilizando o *framework* OSGi. Essa tecnologia tem por característica facilitar o desenvolvimento de *software* modulares na linguagem de programação Java, devido o fato da mesma proporcionar vários benefícios relacionados a gerenciabilidade e manutenibilidade (HALL et al., 2011; BAKKER; ERTMAN, 2013), o que é essencial para esta solução, pois o processo de transformação extensível pensado para a mesma busca usar as abordagens modular e "plugável" permitindo que as extensões sejam inseridas e removidas facilmente.

Logo, esta seção pretende fornecer um detalhamento dos módulos implementados, os pacotes e as classes que os compõem e como eles se relacionam. Dessa forma, a Seção 4.3.1 descreve os **módulos auxiliares**, a Seção 4.3.2 mostra o **Event Admin Security**, a Seção 4.3.3 aborda o **módulo utilitário**, a Seção 4.3.4 traz detalhes dos **módulos padrões**, a Seção 4.3.5 aborda os **módulos específicos**, a Seção 4.3.6 especifica os **módulos adicionais** e a Seção 4.3.7 detalha algumas **operações** realizadas na plataforma.

4.3.1 Módulos auxiliares

Os **módulos auxiliares** são assim chamados porque eles não realizam nenhuma das etapas do fluxo de transformação de dados definidas. No entanto, eles fornecem algumas funcionalidades essenciais ao funcionamento dos **módulos padrões** ou **módulos específicos** e atendem alguns requisitos da plataforma proposta. Dessa forma, neste trabalho

foram implementados três módulos desse tipo: **Segurança**, **Bilhetagem** e **Provedor de dados**.

Como pode ser visto na Figura 18, o módulo **Segurança** é composto por dois pacotes: *br.org.smartcity.seguranca.seguranca* e *br.org.smartcity.seguranca*. No primeiro deles é disponibilizada a classe *Seguranca*, a qual implementa a política de acesso aos dados trafegados na plataforma através de um processo de controle de permissões.

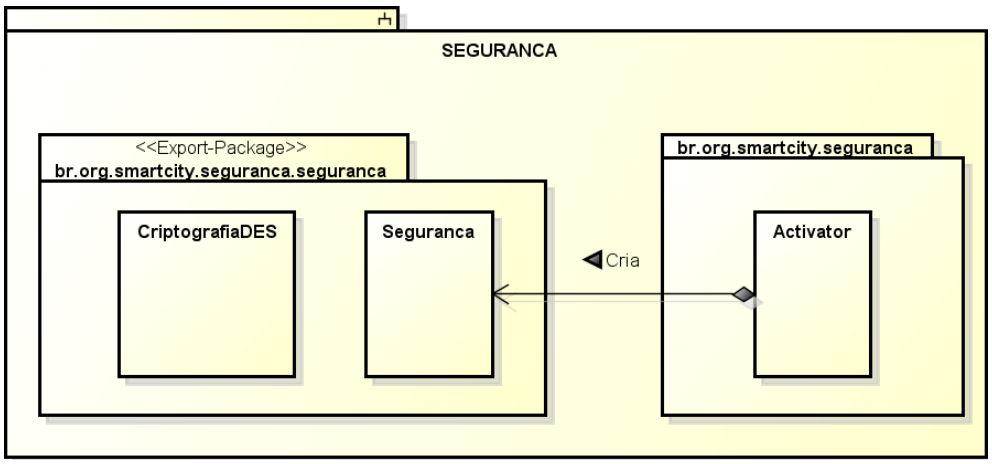


Figura 18: Diagrama de classes do módulo **Segurança**

Ainda nesse pacote, pode-se encontrar a classe *CriptografiaDES* que provê o algoritmo de criptografia de dados *Triple DES* utilizado pelos **módulos padrões** e pelo **Event Admin Security**. Além disso, esse pacote é exportado pelo módulo **Segurança** permitindo que outros módulos possam fazer uso das classes que o compõe. O segundo pacote do módulo **Segurança** possui apenas a classe *Activator*, responsável por instanciar a classe *Seguranca* e registrá-la como um serviço na camada de serviços do OSGi.

Na Figura 19 é mostrada a estrutura do módulo **Bilhetagem**. No pacote *br.org.smartcity.bilhetagem.modelo* tem-se a classe *Bilhetagem* cuja atribuição é realizar a contabilização das mensagens acessadas e que é registrada como um serviço por meio da classe *Activator* do outro pacote do módulo.

O terceiro módulo auxiliar é o **Provedor de dados** que tem a responsabilidade de realizar o armazenamento e gerenciamento dos dados tornando-os acessíveis aos **módulos específicos**. Na Figura 20 tem-se a ilustração do diagrama de classe desse módulo. A classe *ProvedorDeDados* define um conjunto de métodos básicos, sendo justamente ela que deve ser estendida para a criação de novas implementações de **Provedores de dados** e é por meio dela que os **módulos específicos** terão acesso aos dados armazenados.

Além disso, o módulo possui o *ProvedorDeAcesso* que é a classe responsável por libe-

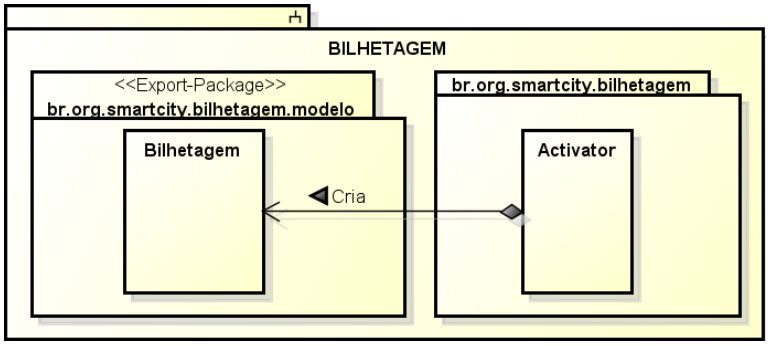


Figura 19: Diagrama de classes do módulo **Bilhetagem**

rar ou bloquear o acesso dos objetos *ProvedorDeDados* aos **módulos específicos**. Esse processo é realizado através de uma tarefa de verificação de permissões realizado pelo módulo **Segurança**. Por fim, a última classe desse módulo é a *Activator* cuja atribuição é instanciar e registrar o objeto *ProvedorDeAcesso* na camada de serviço para que ele possa ser recuperado pelos **módulos específicos**.

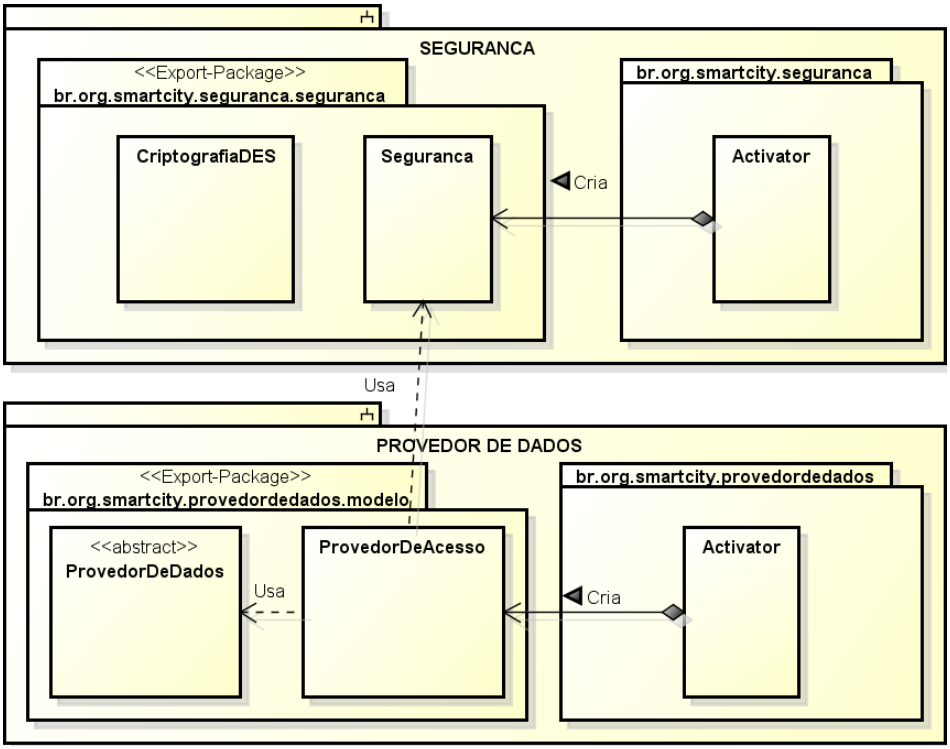


Figura 20: Diagrama de classes do módulo **Provedor de dados**

4.3.2 Módulo Event Admin Security

Esse módulo é uma extensão do *Event Admin* disponibilizado no OSGi e mencionado na Seção 2.3.3, sendo sua responsabilidade gerenciar a troca de mensagens entre os **módulos padrões** da plataforma por meio de um modelo *publish/subscribe*. E, além disso,

a extensão teve como objetivo adicionar um requisito adicional de segurança relacionado ao acesso das mensagens trafegadas.

A Figura 21 exibe o diagrama de classes do módulo **Event Admin Security** e a forma como ele se relaciona com outros módulos. No que diz respeito a extensão realizada, a principal modificação feita foi na classe *EventHandlerTrackerSecurity*, a mesma herda da classe *EventHandlerTracker* do Equinox que tem entre suas responsabilidades adicionar os serviços aptos a receber mensagens, ou seja, os *EventHandler*'s que já foram abordados na Seção 2.3.3.

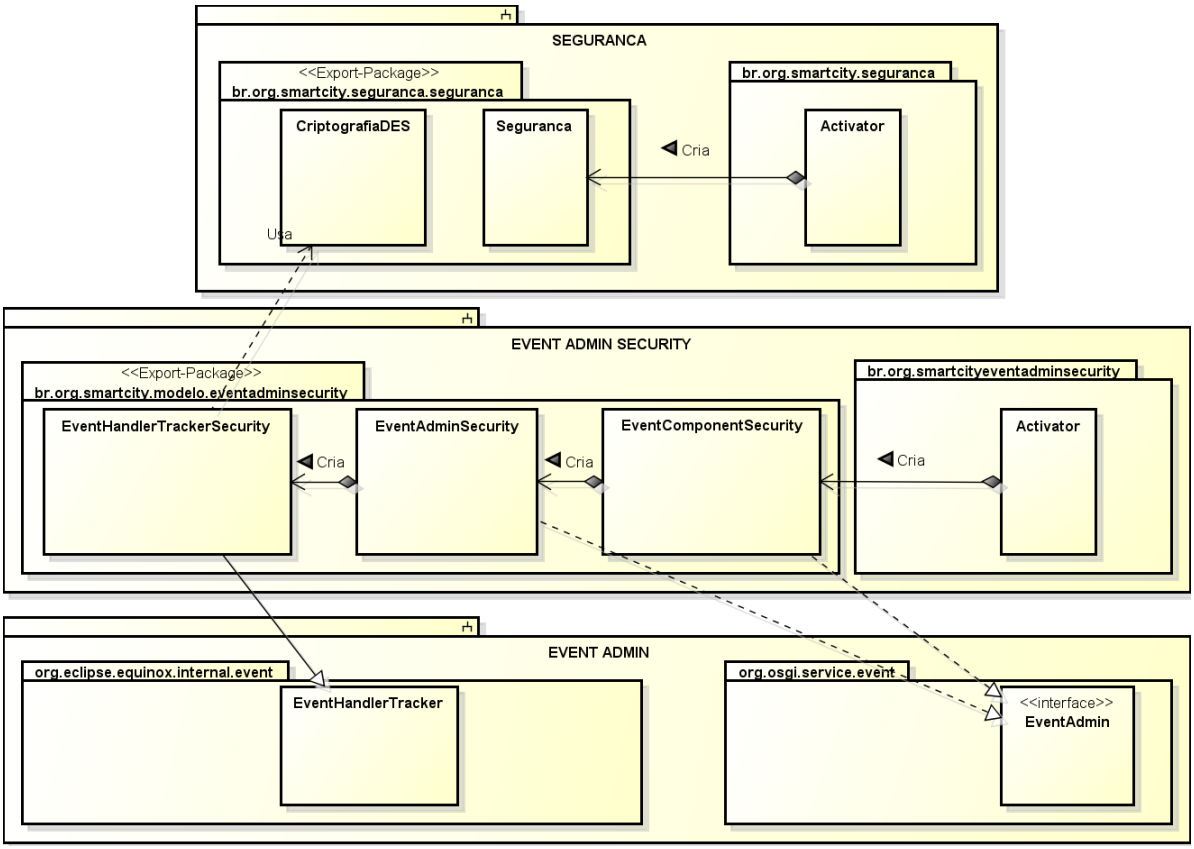


Figura 21: Diagrama de classes do módulo **Event Admin Security**

Assim, a classe *EventHandlerTrackerSecurity* sobrescreve o método *addingService* fazendo com que ele só adicione os *EventHandler*'s que tiverem a permissão especial de **módulo padrão** da plataforma. Essa permissão é adicionada no momento em que o **módulo padrão** registra-se na camada de serviços como sendo um *EventHandler*. Dessa forma, garante-se que apenas os **módulos padrões** sejam capazes de receber as mensagens trafegadas no **Event Admin Security**. Além disso, pode-se ver que essa classe tem uma dependência para a classe *CriptografiaDES*. Isso se deve ao fato da permissão especial ser enviada encriptada com esse algoritmo para garantir maior confidencialidade. Desse modo, o *EventHandlerTrackerSecurity* a utiliza para decriptá-la antes de conferi-la.

Além dessa classe, foi necessária a implementação de mais outras duas que fazem parte do módulo *Event Admin*. A classe *EventAdminImpl* que implementa todo o processo de entrega de mensagens e também tem a responsabilidade de instanciar *EventHandlerTracker* no Equinox. Assim, era necessária uma adaptação para que fosse realizada a instanciação da extensão desenvolvida neste trabalho. No entanto, *EventAdminImpl* só oferece construtores privados e não possui um método *set* para o *EventHandlerTracker*. Desse modo, implementou-se *EventAdminSecurity* que tem a mesma estrutura de *EventAdminImpl* e instancia *EventHandlerTrackerSecurity* ao invés de *EventHandlerTracker*.

A outra classe é a *EventComponent*, a qual representa o serviço de mensagens a ser cadastrado na camada de serviços do OSGi e que instancia *EventAdminImpl*. Dessa forma, era necessário que ela passasse a realizar a instanciação da nova implementação criada para o *EventAdminImpl*. Entretanto, a classe também só possui construtores privados e não disponibiliza nenhum método *set*. Em razão disso, foi desenvolvida a classe *EventComponentSecurity* com o mesmo comportamento de *EventComponent*, porém instanciando *EventAdminSecurity*. Por fim, a classe *Activator* do **Event Admin Security** realiza a instanciação de *EventComponentSecurity* e cadastra ele como um serviço da camada de serviços.

4.3.3 Módulo Utilitário

De acordo com o que foi mostrado na Seção 4.2, os **módulos padrões** tem um modo de funcionamento muito parecido, diferenciando apenas nas suas atribuições no processo de transformação de dados dentro do fluxo. Assim, com o intuito de eliminar replicações de código, desenvolveu-se o módulo **Utilitário** para reunir os comportamentos que eram comuns aos **módulos padrões**.

Na Figura 22 é mostrado o diagrama de classes desse módulo e os vínculos com outros módulos necessários para seu funcionamento. A interface *ServicoSmartCity* foi criada para definir todos os métodos que são essenciais ao funcionamento dos serviços registrados pelos módulos da plataforma, como mostrado na Figura 23.

A classe *ActivatorSmartCity* possui a implementação do conjunto de métodos comuns as classes *Activator*'s dos **módulos padrões**. Ainda na Figura 22, pode-se notar que essa classe possui dependência para os **módulos auxiliares** definidos na Seção 4.3.1. Desse modo, é ela que faz a chamada requisitando que *Seguranca* verifique quais **módulos específicos** têm permissão para acessar os dados recebidos no **módulo padrão** e também solicita que *Bilhetagem* contabilize os dados acessados.

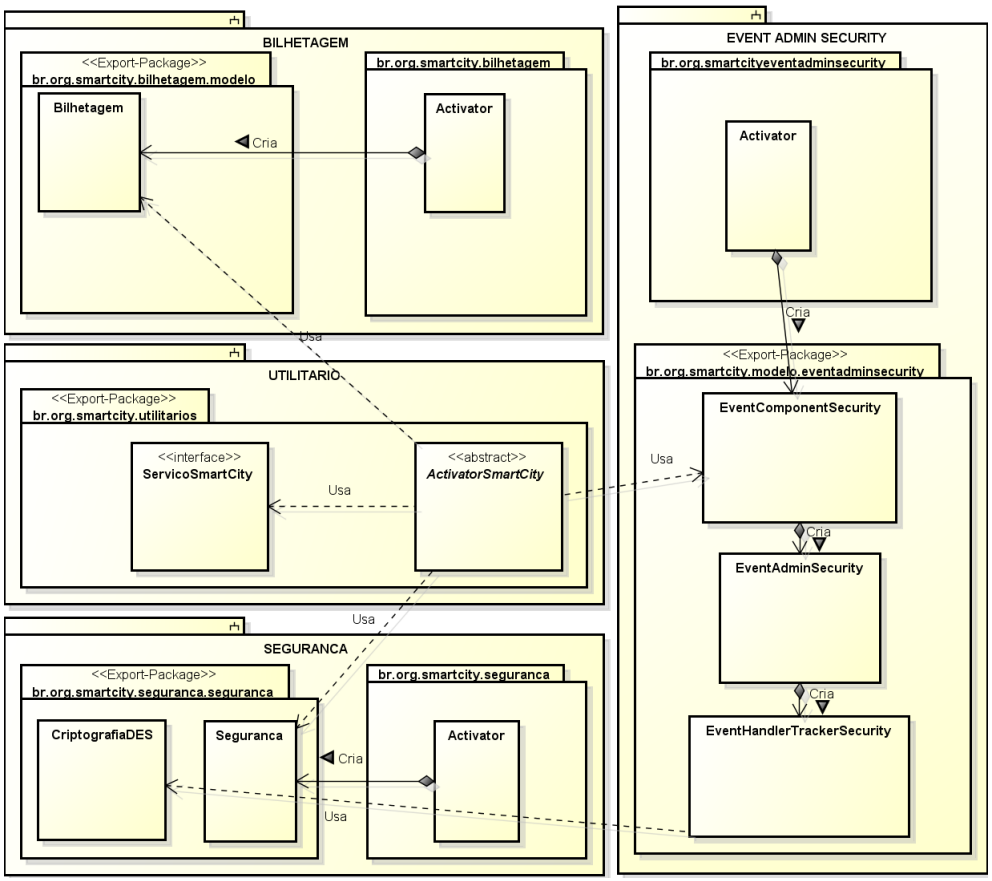


Figura 22: Diagrama de classes do módulo **Utilitário**

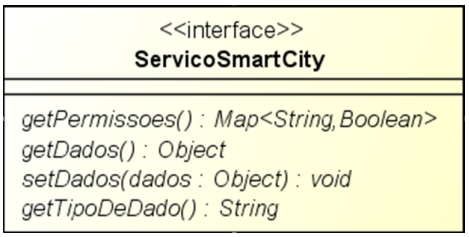


Figura 23: Interface *ServicoSmartCity*

Na Listagem 4.2 é mostrado o método de *ActivatorSmartCity* que aplica as operações de **Segurança** e **Bilhetagem**. Esse método recebe por parâmetro um objeto *BundleContext* que proporciona interação com o OSGi, um *ServicoSmartCity* que representa um dos serviços específicos plugados no **módulo padrão**, o *Bundle* que cadastrou o serviço, os dados (*Object*) e o evento (*Event*) recebido pelo **módulo padrão**. Por fim, é liberado ou bloqueado o acesso dos dados ao serviço de acordo com as permissões do mesmo.

```
1 private boolean aplicarSegurancaBilhetagem(BundleContext context,
     ServicoSmartCity servico, Bundle bundle, Object dados, Event e) {
2   ServiceReference srSeguranca = context.getServiceReference(Seguranca.
       class.getName());
3   if (srSeguranca != null) {
```

```
4           seguranca = (Seguranca) context.getService(srSeguranca);
5     }
6     if (seguranca.acesso(tipoDeDadoDoEvento, servico.getPermissoes())) {
7           if (dados != null) {
8             if (conteudoPagoEvento) {
9                   ServiceReference srBilhetagem = context.getServiceReference
                        (Bilhetagem.class.getName());
10                  if (srBilhetagem != null) {
11                    Bilhetagem bilhetagem = (Bilhetagem) context.getService(
                          srBilhetagem);
12                    if (bilhetagem != null) {
13                          if (bilhetagem.transacao(context.getBundle(
                                idBundleFornecedorDosDados), bundle, precoEvento
                                )) {
14                            operacoesEspecificasDoBundle(servico, dados,
                                bundle, e);
15                          }
16                    }
17                  }
18            } else {
19                  operacoesEspecificasDoBundle(servico, dados, bundle, e);
20            }
21          }
22          return true;
23    } else {
24          return false;
25    }
26  }
```

Listagem 4.2: Método de *ActivatorSmartCity* que aplica **Seguranca** e **Bilhetagem**

Nas linhas de 2 à 4 o método recupera o serviço *Segurança* cadastrado no OSGi e na linha 6 é solicitado que ele verifique se o serviço alvo tem a permissão para acessar o tipo de dado recebido. Caso o acesso não seja liberado encerra-se a execução do método. No entanto, se o serviço tinha permissão para o tipo de dado, é verificado na linha 8 se os dados são pagos. Sendo eles gratuito, o método chama as operações específicas do módulo e retorna. Porém, atendida a condição recupera-se *Bilhetagem* nas linhas de 9 à 11 e na linha 13 aplica-se a transação. Por fim, se a ela foi bem sucedida é realizada a chamada para as operações específicas do módulo.

Além do mais, *ActivatorSmartCity* ainda contém uma dependência para a classe *EventComponentSecurity* do módulo **Event Admin Security**. Isso ocorre porque essa

classe fornece a implementação do método que prepara e publica eventos através do serviço de envio de mensagens do *EventComponentSecurity*, o qual é necessário a quase todos os módulos do fluxo de transformação de dados da plataforma.

A Listagem 4.3 exibe o método de *ActivatorSmartCity* que publica eventos. Nas linhas de 2 à 4 é recuperado o serviço *EventComponentSecurity*. Além disso, cria-se um objeto que carrega as propriedades do evento na linha 6. Ademais, nas linhas de 7 à 14 são configuradas as propriedades e na linha 15 cria-se o evento. Por fim, a linha 16 realiza o envio do evento criado e configurado no método.

```
1  private void enviar(BundleContext context, ServicoSmartCity servico, Bundle
       bundle) {
2    ServiceReference sr = context.getServiceReference(EventComponentSecurity.
       class.getName());
3    if (sr != null) {
4          EventAdmin ea = (EventAdmin) context.getService(sr);
5          if (ea != null) {
6            Dictionary properties = new Hashtable();
7            properties.put(EventConstants.BUNDLE_SYMBOLICNAME,
                getNomeEventoEnviar().replaceAll("/", "."));
8            if (servico.getDados() != null) {
9                  properties.put("DADOS", servico.getDados());
10           }
11           properties.put("TIPO_DE_DADO", servico.getTipoDeDado());
12           properties.put("ID_BUNDLE_FORNECEDOR_DOS_DADOS", bundle.
                getBundleId());
13           properties.put("PAGO", servico.isPago());
14           properties.put("PRECO", servico.preco());
15           Event event = new Event(getNomeEventoEnviar(), properties);
16           ea.postEvent(event);
17         }
18   }
19 }
```

Listagem 4.3: Método de *ActivatorSmartCity* que publica um evento através de **Event Admin Security**

4.3.4 Módulos padrões

Os **módulos padrões** são assim chamados pois são aqueles que fazem parte do fluxo de transformação de dados na plataforma. Porém, eles não realizam o processamento

propriamente dito em razão dessa atribuição ser dos **módulos específicos** plugados a cada um deles. Dessa forma, eles recebem os dados, repassam para que os **módulos específicos** interessados neles realizem esse processo e entregam para o próximo passo do fluxo de transformação ou disponibilizam para o disseminador de dados.

A Figura 24 traz uma representação genérica de como é o diagrama de classes desses **módulos padrões** e como eles se relacionam com os outros módulos. De uma forma geral, módulos desse tipo definem um *Servico* abstrato que definem as operações de processamento a serem realizadas pelos **módulos específicos** que venham a ser plugados a eles.

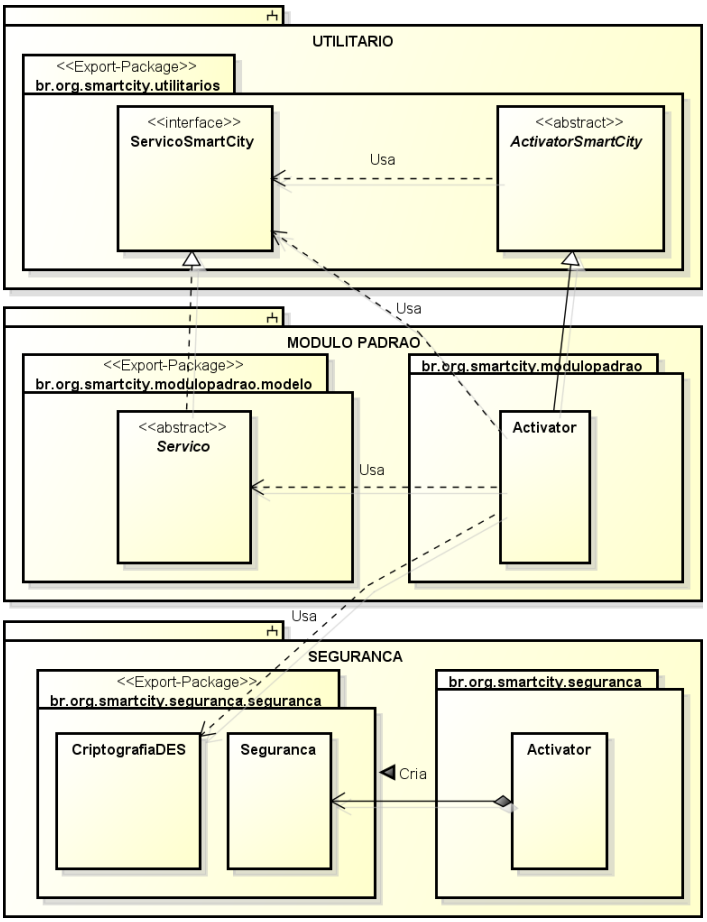


Figura 24: Diagrama de classes representando um **Módulo Padrão**

Um **módulo padrão** possui uma classe *Activator* que herda de *ActivatorSmartCity* e implementa seus métodos abstratos. Além disso, uma *Activator* de módulos desse tipo tem o comportamento diferente das descritas até o momento nesta seção. Nesse caso, a mesma não registra serviços no OSGi e sim utiliza o *Servico* definido no módulo para recuperar todos os serviços desse tipo registrados na camada de serviço, de maneira análoga a Listagem 2.4 mostrada na Seção 2.3.2. Em posse desse conjunto de serviços, o módulo

tem a capacidade de identificar quais deles podem acessar os dados recebidos em cada evento e, assim, delegar a responsabilidade de processamento para os que são capazes de trabalhar com determinado tipo de dados.

No entanto, existem algumas particularidades para alguns **módulos padrões** do fluxo. O **Entrada** não recebe eventos do **Event Admin Security**, pois é o primeiro do processo de transformação e seus **módulos específicos** se conectam diretamente as fontes de dados. E, **Saída** não publica eventos por ser o último módulo do processo e seus **módulos específicos** fornecerem dados aos disseminadores.

Ainda relacionado a Figura 24, percebe-se que existe uma dependência da *Activator* com a classe *CriptografiaDES* do módulo **Segurança**, o que se deve ao fato dela também ser um *EventHandler* para receber eventos do **Event Admin Security**. Dessa forma, *CriptografiaDES* é utilizado para encriptar a permissão especial de acesso às mensagens trafegadas na plataforma no momento em que a *Activator* se registra como *EventHandler*. No entanto, a *Activator* de **Entrada** não possui essa dependência por não receber mensagens devidos aos motivos mencionados no parágrafo anterior.

A Listagem 4.4 mostra uma *Activator* de um **módulo padrão** se registrando como um *EventHandler* em seu método *start*. A linha 2 cria o objeto que armazena as propriedades com as quais o *EventHandler* é registrado. No passo da linha 4 instancia-se um objeto do tipo *CriptografiaDES*. Além disso, na linha 5 configura-se a propriedade *PERMISSAO* com o resultado da encriptação da concatenação do *ID* do módulo com o nome da permissão e usando uma chave. Em seguida, registra-se a classe como um *EventHandler* na linha 6.

```
1 public void start(BundleContext bundleContext) {
2         Dictionary propriedades = new Hashtable();
3         //variaveis omitidas
4         CriptografiaDES des = new CriptografiaDES();
5         propriedades.put("PERMISSAO", des.encriptar(Activator.context.
              getBundle().getBundleId() + " " + "NOME_PERMISSAO", "CHAVE"));
6         context.registerService(EventHandler.class.getName(), this,
              propriedades);
7 }
```

Listagem 4.4: *Activator* de **módulo padrão** se registrando como *EventHandler*

4.3.5 Módulos específicos

Esses módulos recebem esse nome porque são aqueles que se "plugam" aos **módulos padrões** para fornecer uma implementação própria de um passo do processo de transformação de dados definido na plataforma. Na Figura 25 vê-se o diagrama de classe de um **módulo específico** e a maneira como ele se relaciona com um **módulo padrão**.

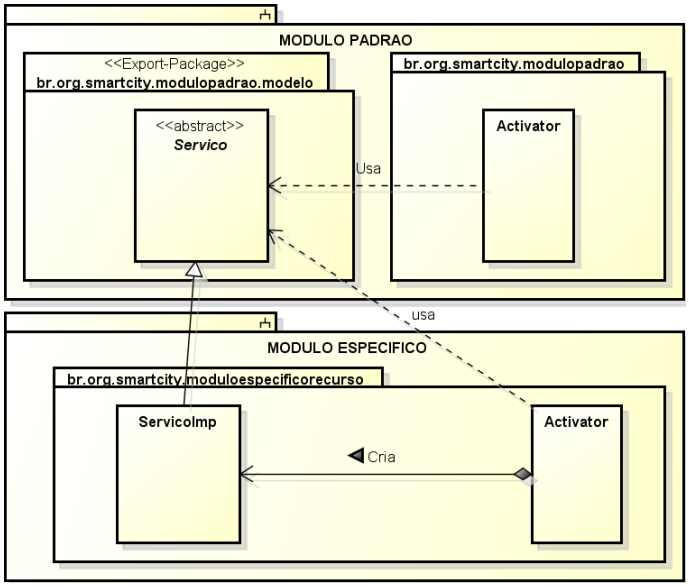


Figura 25: Diagrama de classes representando os **Módulo Específico**

De maneira geral, um **módulo específico** tem uma classe que implementa *Servico* definido no **módulo padrão**. No caso exemplificado, a mesma é *ServicoImp* cuja responsabilidade é estender o serviço definido e prover a implementação de seus métodos abstratos, os quais são usados no processo de transformação de dados da plataforma.

Além disso, ele tem uma classe *Activator* responsável por instanciar o *ServicoImp* e registrá-lo como um serviço do tipo *Servico* na camada de serviços do OSGi. Assim, ocorre o processo de "plugagem" do **módulo específico** no padrão. Dessa forma, um **módulo padrão** passa a ter a capacidade de recuperar uma referência do *ServicoImp* definido no **módulo específico** para que ele possa ser utilizado no passo de processamento de dados.

Na Figura 26 pode-se ver o processo necessário para plugar um **módulo específico** em um **módulo padrão**. Esses últimos definem serviços básicos que são classes abstratas responsáveis por definir os métodos necessários ao comportamento desses módulos na plataforma, como exemplificado na 26A. Assim, um **módulo específico** estende o serviço definido e implementa os métodos abstratos definidos no serviço básico, conforme mostrado na Figura 26B. Por fim, a classe *Activator* do **módulo específico** se encarrega de

criar um objeto do tipo *ServicoImp* e registrá-lo como um *Servico* no sistema de registro de serviços do OSGi, passo ilustrado na 26C.

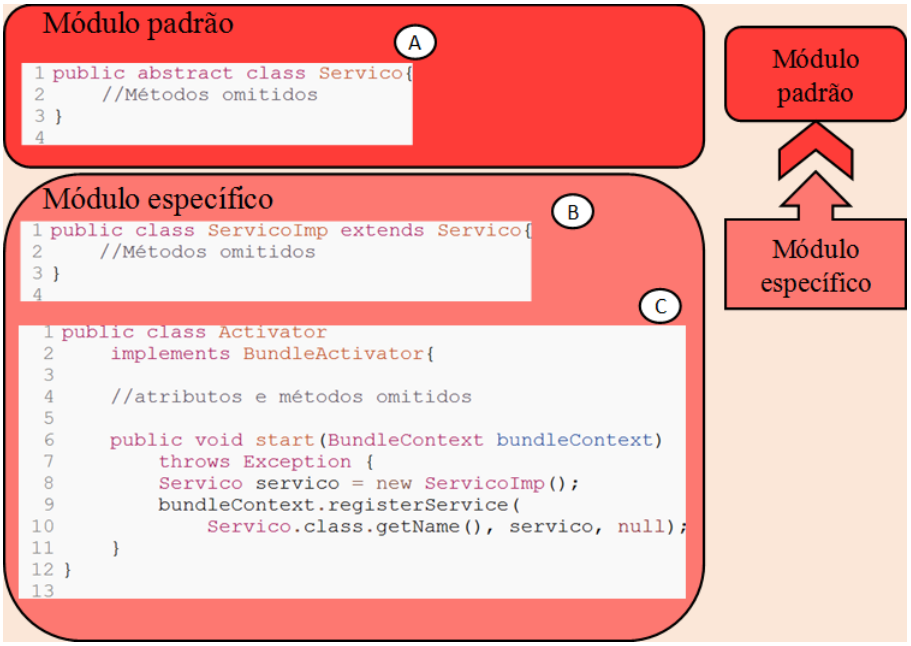


Figura 26: **Módulo Específico** se plugando ao **Módulo Padrão**

4.3.6 Módulos adicionais

Além de todos os tipos de módulos mencionados, ainda é possível que desenvolvedores criem **módulos adicionais** para ajudar no desenvolvimento dos **módulos específicos**. A Figura 27 traz um exemplo comum da utilização desses tipos de módulos. Nela o **módulo adicional** define todas as classes de domínio para um recurso e as exporta para que outros módulos a utilizem. Com isso, todos os **módulos específicos** do mesmo recurso importam o pacote que as contém e fazem uso dela nas suas implementações. Assim, o programador não necessita definir as mesmas classes em vários módulos e, além do mais, a manutenção das classes de domínio só precisa ser feita em um único módulo.

4.3.7 Operações

Esta seção aborda as principais operações realizadas pelos módulos que compõem a plataforma proposta. A Figura 28 exibe os passos necessários para que um **módulo padrão** publique um evento por meio do **Event Admin Security**. Primeiramente, a *Activator* do **módulo padrão** chama o método *enviar* da classe *ActivatorSmartCity*. Em sequência, essa última recupera uma referência do *EventComponentSecurity*. Depois,

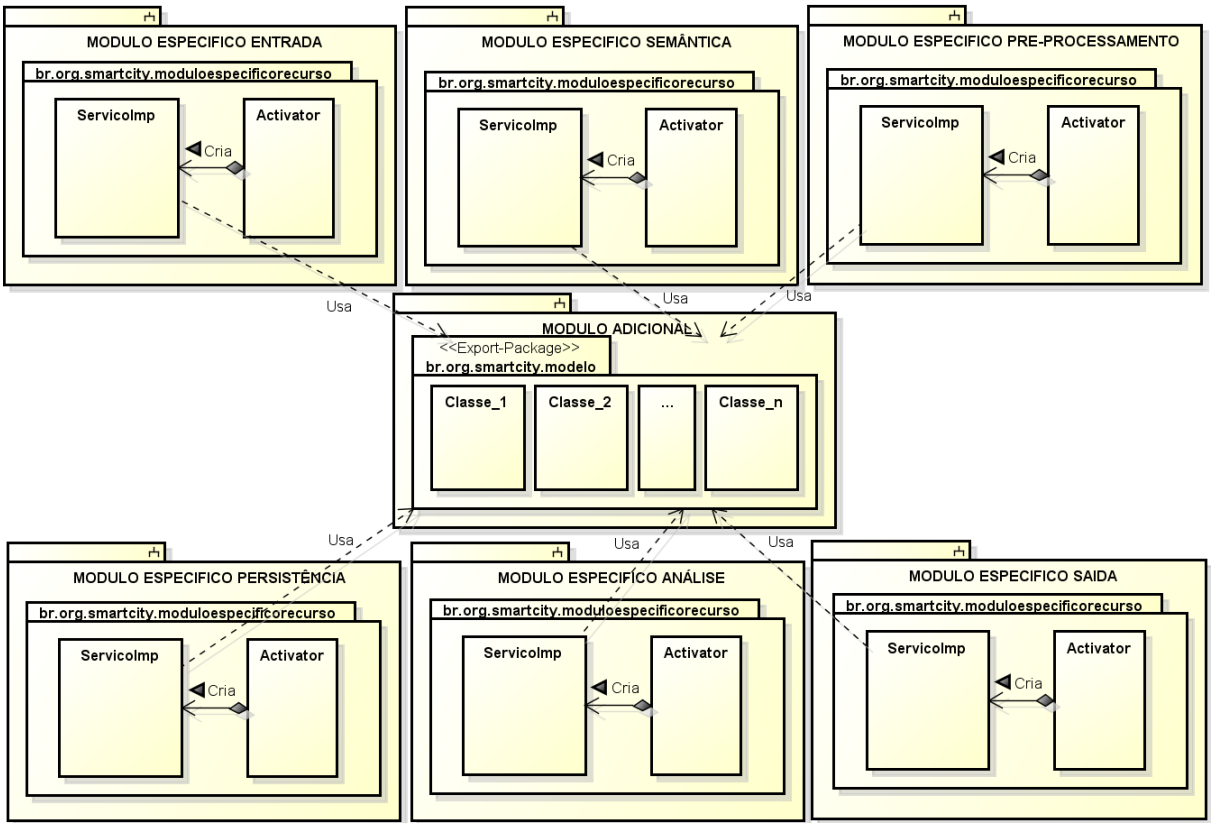


Figura 27: Diagrama de classes exemplificando um **Módulo Adicional**

ela configura as propriedades do evento e o publica através do método *postEvent* de *EventComponentSecurity*.

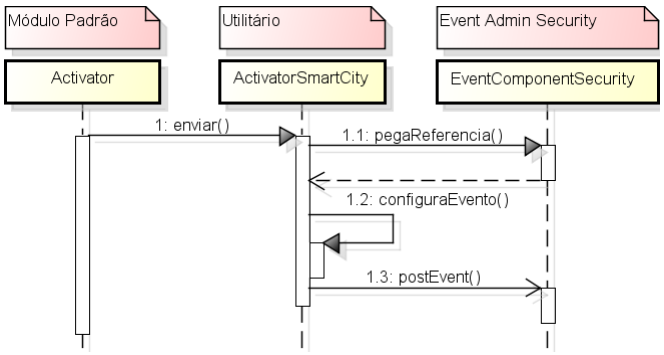


Figura 28: Diagrama de sequência dos passos para publicação de um evento

Ao receber o evento, como mostrado na Figura 29, o *EventComponentSecurity* o repassa para a classe *EventAdminSecurity*. Ela verifica dentre os **módulos padrões** qual(is) está(ão) inscrito(s) para receber o tipo de evento. E, em seguida, o entrega ao conjunto de inscritos identificados.

Com relação a Figura 30, ela ilustra os passos necessários para que um **módulo padrão** repasse aos **módulos específicos** os dados recebidos em um evento. A priori, o

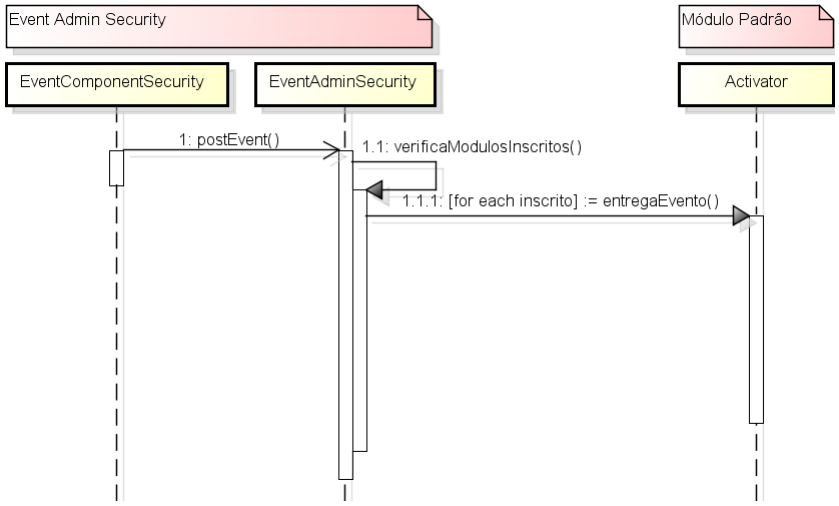


Figura 29: Diagrama de sequência dos passos realizados pelo **Event Admin Security** para entrega de eventos

módulo padrão recupera todos os módulos plugados a ele. Em seguida, ele solicita que *ActivatorSmartCity* aplique as operações de **Segurança** e **Bilhetagem** para cada um deles.

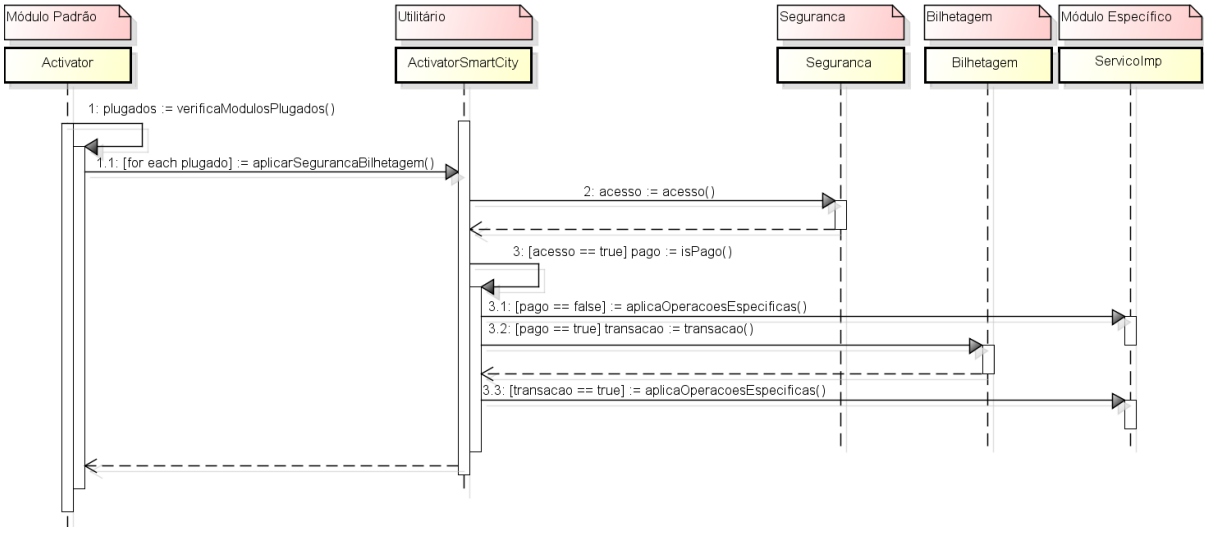


Figura 30: Diagrama de sequência dos passos para entrega de dados ao **módulo específico**

Dessa forma, é pedido que a classe *Seguranca* verifique se o módulo plugado tem permissão para acessar os dados. Caso não tenha, a execução é retornada para o **módulo padrão**. No entanto, se ele tinha permissão de acesso aos dados, é verificado se o conteúdo a ser repassado é pago. Assim, sendo os dados gratuitos, *ActivatorSmartCity* chama as operações específicas do serviço registrado no **módulo específico** e a execução retorna para o **módulo padrão**. Caso contrário, são aplicadas as operações do módulo **Bilhetagem**. Dessa modo, se as mesmas forem bem sucedidas, solicita-se que as operações

específicas de *ServicoImp* sejam executadas e, após isso, retorna-se ao **módulo padrão**.

4.4 Comparativo com trabalhos relacionados

Apesar do estudo de Anthopoulos e Fitsilis (2010), atender vários requisitos de arquiteturas para cidades inteligentes, a plataforma proposta no mesmo não possibilita extração de conhecimento dos dados integrados, não segue uma abordagem modular, não é plugável e não permite monetização dos dados. Por outro lado, a plataforma proposta nesta dissertação atende todos esses requisitos.

Além do mais, a plataforma implementada neste trabalho se diferencia dos estudos de Filipponi et al. (2010), Valente e Martins (2011), em razão dos mesmos não incorporarem as abordagens modular e plugável, não manterem a privacidade dos dados trafegados, não permitirem extração de conhecimento a partir do grande volume de dados e não possibilitarem a monetização dos mesmos.

Ademais, este trabalho se distingue do estudo de Hernandez e Larios (2014), pois a plataforma proposta nesse último não permite agregação de dados, não segue as abordagens modular e plugável, não permite extração de conhecimento e não incorpora monetização de dados.

O presente trabalho, diferentemente das plataformas de Gama, Touseau e Donsez (2012), Tomas (2014), Blackstock et al. (2010), mantém a privacidade dos dados trafegados, possibilita extração de conhecimento e aborda monetização dos dados. Além do mais, em Blackstock et al. (2010) não se aborda agregação de dados.

O trabalho de Andreini et al. (2011), dentre os requisitos atendidos por este trabalho, incorpora apenas os requisitos de obtenção de dados de fontes heterogêneas e criação de novos serviços a partir dos dados integrados. Com relação a plataforma Octoblu (2014), a mesma não permite a monetização dos dados trafegados e não segue as abordagens modular e plugável.

Diferentemente da plataforma proposta neste trabalho, o estudo de Borja e Gama (2014), não se preocupa com a questão da privacidade dos dados e o estudo Fiware (2011), não possibilita monetização dos dados. No que diz respeito a plataforma descrita em Carriots (2011), a mesma não é plugável, não permite extração de conhecimento a partir dos dados integrados e não possibilita a monetização dos dados.

Os trabalhos Digi (2013), Cybervision (2014), Prismtech (2014), Seecontrol (2014),

se distinguem deste estudo por não permitirem agregação de dados, não possibilitarem a monetização dos mesmos e não seguirem as abordagens modular e plugável. Além disso, os estudos Cybervision (2014), Prismtech (2014), não permitem extração de conhecimento dos dados integrados.

Por fim, nenhum dos trabalhos relacionados definem um fluxo de transformação bem definido e extensível para o tratamento dos dados em suas propostas de arquitetura. Nesse fluxo determina-se os passos de processamento necessários para o tratamento do conjunto de dados integrados à plataforma a fim de entregá-los da melhor maneira possível para que eles sejam utilizados na criação de novos sistemas. Além disso, a extensibilidade das etapas definidas para o fluxo possibilita a adequação do processamento desses passos para que eles possam ser realizados conforme as características e restrições de cada tipo de dado existente nos "ecossistemas" de uma cidade inteligente.

5 Avaliação

Neste capítulo aborda-se o processo de avaliação da plataforma proposta no Capítulo 4. Dessa forma, o mesmo apresenta um estudo de caso na Seção 5.1, o qual mostra a avaliação de alguns aspectos importantes da plataforma implementada. Por fim, a Seção 5.2 descreve um projeto de utilização da plataforma para trabalhar com o cenário de controle de multidões.

5.1 Estudo de caso 1: estacionamento inteligente

Esta seção descreve um estudo de caso no qual os dados de um estacionamento foram integrados à plataforma implementada no intuito de verificar alguns aspectos comportamentais da mesma. Dessa forma, o objetivo principal do mesmo é avaliar a tarefa de criação de **módulos específicos** (**Extensibilidade**) e a capacidade de processamento de dados (**Performance**). Por fim, o planejamento e descrição do mesmo seguiram as diretrizes estabelecidas por Yin (2003), Kitchenham, Pickard e Pfleeger (1995) e Runeson e Höst (2009).

5.1.1 Planejamento

O planejamento deste estudo de caso envolve definir as questões de pesquisa investigadas, os sujeitos envolvidos, o objeto utilizado, as unidades de análise e os procedimentos para coleta de dados. Cada uma dessas informações será definida nas próximas seções.

5.1.1.1 Questões de pesquisa

Com o intuito de alcançar o objetivo estabelecido, este estudo de caso deve responder as seguintes questões de pesquisa:

1. O processo de extensão da plataforma através do desenvolvimento de **módulos**

específicos é uma atividade trabalhosa?

2. O desempenho do processo de tratamento do fluxo de dados é prejudicado em função da existência de um conjunto de passos de transformação das informações?

3. O desempenho do processo de tratamento do fluxo de dados é prejudicado quando são utilizados **módulos específicos** "plugados" nos **módulos padrões**?

No propósito de responder essas questões de pesquisa realizou-se a extensão da plataforma para que a mesma fosse capaz de trabalhar com dados provenientes de um estacionamento inteligente, os quais fornecem informações sobre a ocupação de vagas nesse ambiente.

5.1.1.2 Sujeitos

A plataforma proposta e implementada neste trabalho foi utilizada pelo próprio pesquisador, o qual possui experiência no desenvolvimento de aplicações Java e OSGi. Portanto, o sujeito envolvido neste estudo de caso é o respectivo criador da plataforma.

5.1.1.3 Objeto

O objeto utilizado foi a extensão da plataforma proposta para integrar os dados de um estacionamento, o qual é supervisionado por um conjunto de câmeras responsáveis por monitorar suas respectivas vagas. Dessa forma, essas câmeras proveem um conjunto de imagens para um sistema que as processa e, posteriormente, disponibiliza as informações do conjunto de vagas monitorados por cada uma delas através de um servidor implementado na arquitetura REST. Assim, pretendia-se acessar os dados desse servidor, transformá-los por meio do fluxo extensível da plataforma definida no Capítulo 4 e, em seguida, torná-los disponíveis para que aplicações que necessitassem utilizar esses dados pudessem ser desenvolvidas, como mostrado na Figura 31.

O ambiente de um estacionamento foi escolhido por ser um cenário bastante comum nas cidades e, em razão, de alguns deles utilizarem vários dispositivos em sua administração, os quais podem ser usados como fontes de dados para as cidades inteligentes. Dessa forma, com os dados gerados em um ambiente desse tipo pode-se criar aplicativos que forneçam novas funcionalidades aos usuários desses locais proporcionando maiores informações para os mesmos e, consequentemente, uma melhoria no funcionamento desse serviço.

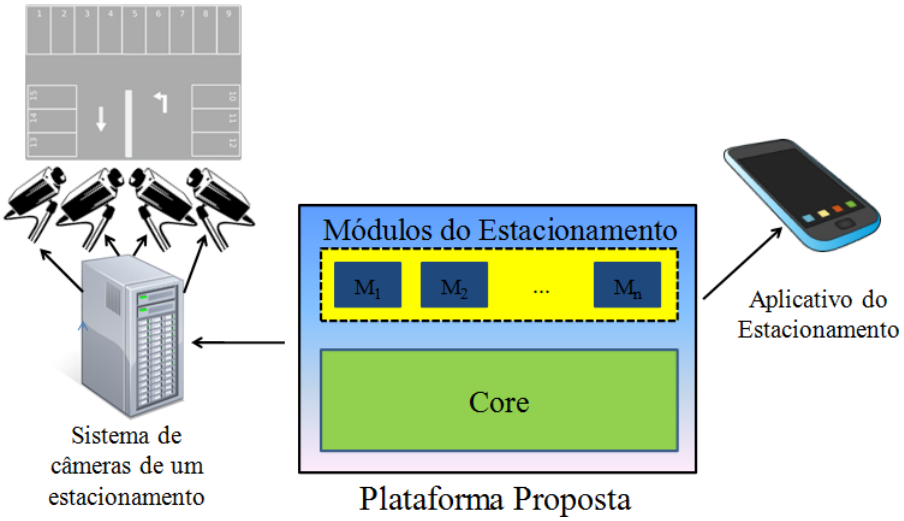


Figura 31: Inclusão do recurso estacionamento na plataforma proposta

Assim, como pode ser visto na Figura 32, os dados disponibilizados por esse recurso são um *array* de n **câmeras** sendo que cada **câmera** possui um **id** e um *array* que representa as **vagas** monitoradas por ela. Por fim, cada **vaga** possui o seu **id** e um atributo **ocupada** que identifica se a **vaga** está ocupada ou livre.


```
array [4] - Câmeras
▼ 0 {2}
    id : 1
  ▼ vagas [5]
    ▼ 0 {2}
        id : 1
        ocupada : true
    ▶ 1 {2}
    ▶ 2 {2}
    ▶ 3 {2}
    ▶ 4 {2}
▶ 1 {2}
▶ 2 {2}
▶ 3 {2}
```


Figura 32: Representação dos dados gerados no estacionamento

5.1.1.4 Unidades de análise

Este estudo de caso possui duas unidades de análise, as quais são: a plataforma implementada e a extensão da mesma para trabalhar com os dados do estacionamento. Assim, os **módulos padrões** da plataforma foram avaliados com relação ao desempenho do processo de tratamento do fluxo de dados nos mesmos. Já a extensão para lidar com o

recurso estacionamento foi explorada na análise da extensibilidade e também na avaliação de performance do fluxo de dados quando inseridos **módulos específicos** "plugados" aos **módulos padrões**.

5.1.1.5 Coleta de dados

Os seguintes dados foram coletados neste estudo de caso:

1. A quantidade de linhas de códigos que foram necessárias para realizar a extensão da plataforma por meio dos **módulos específicos**, os quais trabalham com os dados do recurso estacionamento;
2. Os tempos de tráfego dos fluxos de dados quando utilizado apenas os **módulos padrões** da plataforma;
3. Os tempos de tráfego dos fluxos de dados quando **módulos específicos** são "plugados" à plataforma.

5.1.2 Execução

Para incluir os dados do estacionamento na plataforma proposta foi gerada a implementação dos **módulos específicos** para trabalhar com esse recurso, como pode ser visto na Figura 33. Além disso, nela também nota-se que os **módulos específicos** ao realizarem a extensão dos padrões herdam suas interfaces que já disponibilizam os comportamentos básicos de cada passo de processamento de dados e define os comportamentos que devem ser implementados nos módulos a serem "plugados".

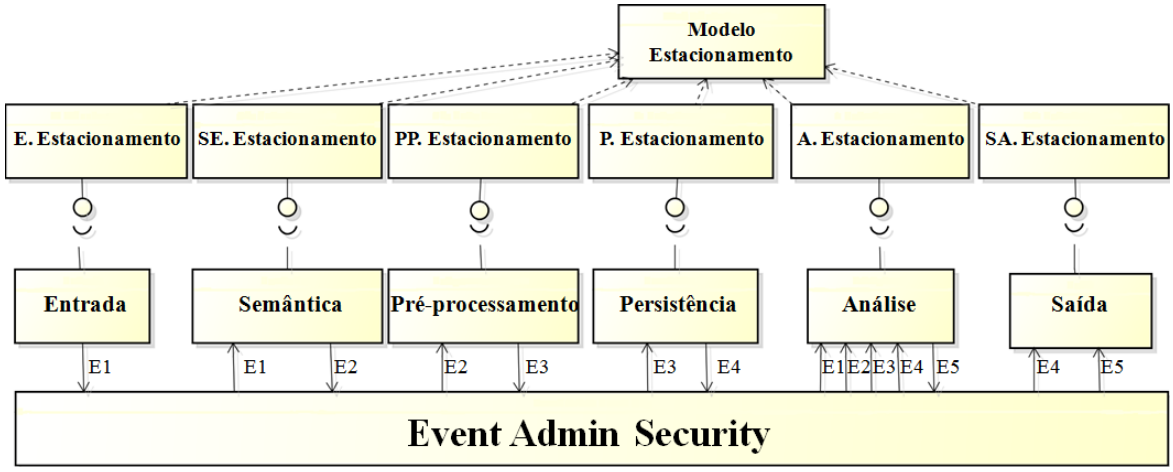


Figura 33: Implementação da arquitetura para o recurso estacionamento

A priori, foi desenvolvido o módulo **Modelo Estacionamento** responsável pelas classes que modelam os dados do recurso **estacionamento** dentro da arquitetura, nesse caso as classes criadas foram: **Camera** e **Vaga**, conforme mostrado na Figura 34. Assim, todos os **módulos específicos** desse recurso importam o **Modelo Estacionamento** para simplificar a manipulação desses dados.

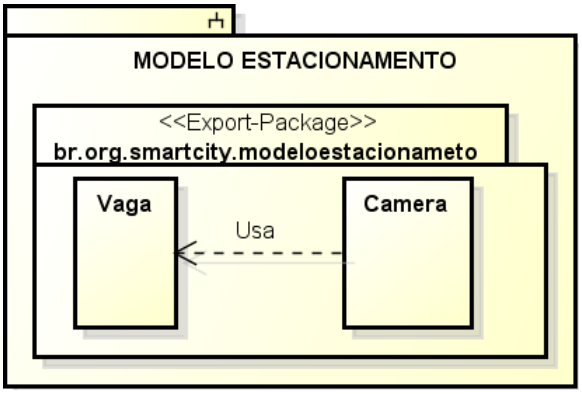


Figura 34: Diagrama de classes de **Modelo Estacionamento**

Em seguida, implementou-se o módulo **E Estacionamento** (Entrada Estacionamento) que na arquitetura serve como canal de entrada para os dados gerados no estacionamento. Desse modo, para manter os dados desse recurso atualizados tal módulo acessa periodicamente o servidor de câmeras. Posteriormente, foi gerado o **SE Estacionamento** (Semântica Estacionamento) responsável por representar tais dados em objetos, os quais são instanciados a partir das classes definidas em **Modelo Estacionamento**.

Por conseguinte, foi desenvolvido o módulo **PP Estacionamento** (Pré-processamento Estacionamento) cujas atribuições são eliminar a duplicação de dados decorrente da existência de interseção entre os conjuntos de vagas monitorados pelas câmeras. E, selecionar apenas os dados centrais ao problema do recurso estacionamento, ou seja, o conjunto de todas as vagas com as informações das mesmas. Em sequência, foi implementado **P Estacionamento** (Persistência Estacionamento) que apenas é responsável por realizar o passo de armazenamento dos dados recebidos.

Neste estudo também desenvolveu-se o módulo **A Estacionamento** (Análise Estacionamento) que apenas armazena em um arquivo o *log* de todos os eventos 1, 2, 3 e 4 enviados no *Event Admin Security*. Isso foi importante para confirmar a sequência de eventos enviados na plataforma. No entanto, em cenários mais complexos esse módulo deveria processar constantemente os dados transportados nesses eventos para extrair informações relevantes aos problemas tratados.

O módulo **SA Estacionamento** (Saída Estacionamento) é um módulo REST que

funciona como porta de saída para os dados do recurso estacionamento transformados na arquitetura. Assim, caso um desenvolvedor queira se utilizar dos dados desse recurso para gerar um novo serviço isso será feito através desse módulo. A Figura 35 ilustra um aplicativo criado na tecnologia Android que apenas lista o conjunto de vagas do estacionamento e o estado (Ocupada/Disponível) das mesmas. Essa aplicação foi implementada apenas para comprovar o processo de disseminação dos dados transformados na plataforma, função a qual ela também se propõe a realizar.

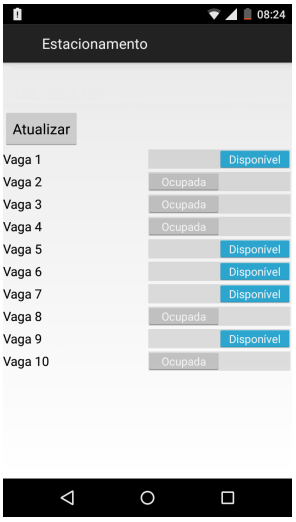


Figura 35: Tela do aplicativo que consume os dados do estacionamento transformados na plataforma

Dessa forma, com esses módulos implementados e "plugados" na plataforma, o fluxo de dados do cenário estacionamento ocorre como apresentado na Figura 36. A priori, os dados no formato *json* do sistema de câmeras são acessados e inseridos na plataforma através de **E Estacionamento**, o qual solicita ao módulo **Entrada** que os repasse na arquitetura. Vale ressaltar que o servidor de câmeras está sendo simulado e que os dados relacionados à ocupação das vagas, disponibilizados pelo mesmo, são gerados aleatoriamente.

Em seguida, **Semântica** recebe esses dados e faz a solicitação para que **SE Estacionamento** represente-os no formato desejado. Assim, como mostrado na etapa 5 os dados no formato *json* passam a ser representados no paradigma orientado a objetos e depois são retornados ao **módulo padrão**, o qual os publica.

Pré-processamento obtém os dados e requisita que **PP Estacionamento** processe os mesmos. Conforme exibido na etapa 9 é realizado um processo de limpeza e filtragem, onde respectivamente são eliminadas as duplicações e selecionado os dados centrais ao problema. Após isso, **Pré-processamento** publica os dados pré-processados.

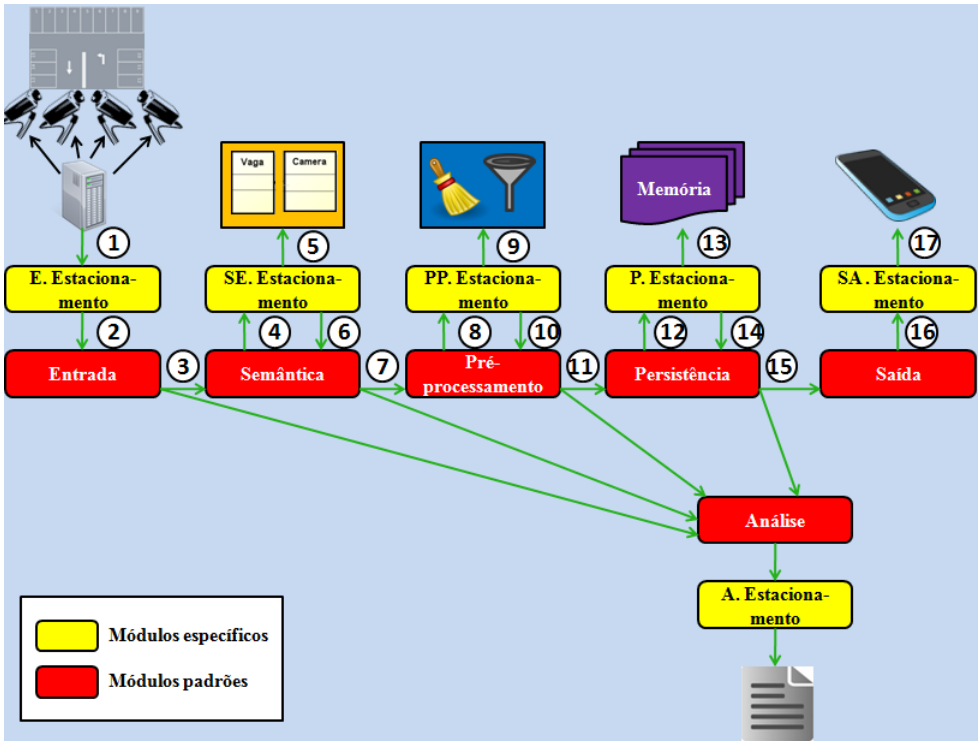


Figura 36: Fluxo de dados do cenário estacionamento

Posteriormente, **Persistência** os recebe e solicita que **P Estacionamento** realize o processo de armazenamento, o qual nesse caso é feito em memória (Etapa 13). Logo, o último estado das vagas armazenadas são publicadas. Em sequência, **Saída** acessa tais dados e os repassa para que **SA Estacionamento** realize a disponibilização dos mesmos. Desse modo, o aplicativo criado pode acessá-los e exibi-los aos usuários.

Em paralelo a realização dessas tarefas, o módulo **Análise** recebe todos os dados enviados nas etapas 3, 7, 11 e 15 e armazena em um arquivo de texto o *log* da sequência de eventos transmitidos na plataforma.

Após a implementação de todos esses módulos e execução do fluxo, partiu-se para a etapa de avaliação dos aspectos **extensibilidade** e **performance** da plataforma. Essa avaliação foi realizada em uma máquina com sistema operacional Windows 8.1 Single Langue 64 bits, processador Intel (R) Core (TM) i3-3227U CPU @ 1.90GHz e 3,87 GB de memória RAM .

Com relação a extensibilidade, coletou-se a quantidade de linhas de código implementadas em cada um dos **módulos específicos** do sistema de estacionamento, conforme mostrado na Tabela 5. Nessa contagem todas as linhas dos arquivos de código fonte foram contabilizadas, o que inclui as importações, declarações, etc. Além disso, também foram contabilizadas as linhas de código que estão relacionadas diretamente as tarefas

necessárias a "plugagem" desses módulos na plataforma, mostrado na Tabela 6, as quais são definidas nos **módulos padrões** e implementadas nos **módulos específicos**.

Tabela 5: Linhas de código do sistema estacionamento

Número de linhas de código	
E Estacionamento	78
SE Estacionamento	66
PP Estacionamento	79
P Estacionamento	68
A Estacionamento	78
SA Estacionamento	61
Modelo Estacionamento	56
Total	**486**

Tabela 6: Quantidade de linhas de código dos módulos do sistema de estacionamento (ignorando declarações e código geral)

Número de linhas de código	
E Estacionamento	7
SE Estacionamento	9
PP Estacionamento	8
P Estacionamento	10
A Estacionamento	9
SA Estacionamento	12
Total	**55**

No que diz respeito a performance, esse requisito foi avaliado usando a medida da taxa de transferência de fluxo de dados. Particularmente, esta medida foi calculada com base no tempo necessário para uma mensagem trafegar a partir do ponto de entrada até o fim do fluxo de transformação. Para este efeito, foi calculado o tempo médio que uma certa quantidade de pacotes enviados de uma vez leva para percorrer todas as etapas de processamento da plataforma. Além disso, para cada quantidade de pacotes enviados coletou-se dez amostras e, em seguida, gerou-se uma média geral dessas amostras obedecendo a equação 5.1.

$$m = \frac{\sum_{j=1}^{j=10} \frac{\sum_{i=1}^{i=qp} ti}{qp}}{10} \tag{5.1}$$

Onde:

- $\boldsymbol{m}$ – representa a média geral;

- ***qp*** – representa a quantidade de pacotes recebidos;
- ***ti*** – representa o tempo de tráfego do pacote ***i***.

Desse modo, na Tabela 7 tem-se as primeiras amostras coletadas, as mesmas refletem os tempos (em segundos) de tráfego quando usado apenas os **módulos padrões** da plataforma, neste caso limitou-se a memória em 512 MB. Nas colunas são mostrados os tempos referentes a cada quantidade de pacotes (**Q.P**), a qual varia de 1 até $5,53x10^5$ pacotes. Por sua vez, as linhas indicam a amostra (**Amost.**) em que tais tempos foram medidos.

Tabela 7: Tempos de tráfego no fluxo usando-se apenas **módulos padrões** (limite de 512 MB)

Amost. / Q. P.	**1**	**10**	**100**	**1000**	$\mathbf{1x10^4}$	$\mathbf{1x10^5}$	$\mathbf{5x10^5}$	$\mathbf{5,53x10^5}$
1	0,003	0,001	0,038	0,046	0,245	4,001	122,693	354,319
2	0,001	0,001	0,048	0,153	0,242	4,435	128,171	337,556
3	0,002	0,001	0,042	0,051	0,228	3,865	131,748	367,169
4	0,002	0,001	0,037	0,036	0,284	3,944	124,395	350,268
5	0,003	0,001	0,044	0,032	0,258	3,569	123,170	363,569
6	0,002	0,000	0,043	0,030	0,236	3,525	118,208	345,420
7	0,000	0,001	0,042	0,030	0,233	3,887	126,845	356,196
8	0,004	0,001	0,029	0,023	0,228	3,431	124,550	358,718
9	0,002	0,001	0,035	0,020	0,229	3,587	121,525	348,555
10	0,005	0,002	0,027	0,035	0,252	3,447	125,146	357,003
Média	0,002	0,001	0,039	0,046	0,244	3,769	124,645	353,877

Com relação aos dados exibidos na Tabela 8, os mesmos são de um experimento análogo ao da Tabela 7, porém nele o limite de memória para execução da plataforma foi dobrado. As colunas dessa tabela mostram os tempos (em segundos) coletados para cada quantidade de pacotes (**Q.P**), a qual variou de 1 até $1,106x10^6$ pacotes e as linhas indicam a amostra (**Amost.**) em que tais tempos foram medidos.

Por fim, a Tabela 9 mostra o tempo de tráfego (em segundos) quando utilizados os **módulos específicos** do sistema de estacionamento "plugados" na plataforma. As linhas indicam a amostra (**Amost.**) em que cada tempo foi coletado e as colunas mostram os tempos referentes a cada quantidade de pacotes (**Q.P**), a qual variou de 1 até 100.000 pacotes.

Tabela 8: Tempos de tráfego no fluxo usando-se apenas **módulos padrões** (limite de 1.024 MB)

Amost. / Q. P.	**1**	**10**	**100**	**1000**	$\mathbf{1x10^4}$	$\mathbf{1x10^5}$	$\mathbf{5x10^5}$	$\mathbf{1x10^6}$	$\mathbf{1,106 x10^6}$
1	0,001	0,001	0,040	0,104	0,259	3,407	24,834	277,317	588,032
2	0,002	0,001	0,051	0,140	0,270	3,664	28,726	271,934	581,146
3	0,002	0,001	0,039	0,055	0,290	3,745	25,934	258,339	580,306
4	0,002	0,001	0,049	0,039	0,249	3,558	22,750	277,446	595,412
5	0,002	0,001	0,025	0,041	0,266	3,047	24,310	272,817	588,159
6	0,002	0,001	0,027	0,023	0,256	3,311	22,112	259,500	574,412
7	0,002	0,001	0,039	0,045	0,254	3,318	23,281	262,176	559,866
8	0,001	0,001	0,014	0,035	0,246	3,147	22,613	271,118	580,699
9	0,002	0,001	0,040	0,032	0,253	3,106	23,392	264,620	583,546
10	0,002	0,001	0,029	0,037	0,239	3,183	23,854	261,356	581,050
Média	0,002	0,001	0,035	0,055	0,258	3,349	24,781	267,662	581,263

Tabela 9: Tempos de tráfego no fluxo quando usado os **módulos específicos** de estacionamento (limite de 512 MB)

Amost./ Q. P.	**1**	**10**	**100**	**1000**	**10000**	**100000**
1	0,016	0,050	0,404	3,175	23,418	225,472
2	0,000	0,047	0,402	2,513	21,745	224,427
3	0,000	0,068	0,333	2,248	20,835	239,182
4	0,015	0,067	0,320	2,048	20,474	235,292
5	0,015	0,052	0,373	2,091	22,190	233,126
6	0,015	0,062	0,288	2,044	21,712	220,499
7	0,015	0,075	0,287	1,985	20,750	232,160
8	0,000	0,058	0,302	1,996	21,449	219,868
9	0,016	0,053	0,314	1,968	20,775	226,203
10	0,015	0,062	0,270	1,973	21,325	223,685
Média	0,011	0,059	0,329	2,204	21,467	227,991

5.1.3 Ameaças à validade

Para o estudo de caso realizado foram avaliados os quatro tipos de validade:

1. *Validade de construção*: a captura dos dados da execução deste estudo de caso foi realizada por meio de levantamentos quantitativos relacionados aos fatores analisados para a plataforma implementada. Além disso, esse processo ocorreu em uma única máquina evitando que alterações em configurações computacionais comprometessem os valores coletados no estudo.

2. *Validade interna*: o estudo de caso foi realizado pelo próprio criador da plataforma, o

que diminuiu o risco de que fatores relacionados a inexperiência dos programadores no desenvolvimento de aplicações baseadas em OSGi e Java saíssem do controle.

3. *Validade externa*: a implementação dos **módulos específicos** utilizados neste estudo de caso foram realizadas pelo próprio desenvolvedor da plataforma, o qual possui experiência no desenvolvimento de *software* utilizando a linguagem Java e o *framework* OSGi. Dessa forma, programadores iniciantes nessas tecnologias podem gerar soluções com uma maior quantidade de linhas de código. Além disso, os mesmos levarão mais tempo para o entendimento dos comportamentos dos **módulos específicos** e implementações de suas funcionalidades. Por fim, os tempos coletados representam a performance da plataforma na máquina cujas configurações foram especificadas na Seção 5.1.2. Assim, a utilização de outras configurações de máquina influenciarão no tempo de tráfego das mensagens transformadas por ela.

4. *Validade de conclusão*: neste estudo foram utilizados dados quantitativos que colaboraram para o processo de avaliação da plataforma. No que diz respeito aos dados de performance, os mesmos foram coletados em várias amostragens para a realização do cálculo de uma média, impedindo que desvios refletindo apenas um momento específico influenciassem o resultado do estudo.

5.1.4 Respostas às questões de pesquisa

Como mencionado na Seção 5.1.2, o sujeito deste estudo de caso implementou os **módulos específicos** que realizam a transformação dos dados relacionados ao recurso estacionamento e, em seguida, coletou-se os dados no intuito de responder as questões de pesquisa.

5.1.4.1 Questão de pesquisa 1: O processo de extensão da plataforma através do desenvolvimento de módulos específicos é uma atividade trabalhosa?

A extensão de módulos responsáveis pelo o processamento de fluxo de dados não é uma tarefa trabalhosa, já que é necessário a implementação de apenas uma pequena parte do código a fim de ligá-los na plataforma. Como mostrado na Tabela 5, para executar a aplicação específica dos seis módulos do sistema de estacionamento foi necessária a implementação de 486 linhas de código. É importante ressaltar que esta medida considerou todas as linhas de código, incluindo as importações, declarações, etc.

No entanto, para ter uma ideia mais precisa sobre o trabalho do programador para realizar a extensão da plataforma foram contadas apenas as linhas de código que estão diretamente relacionadas as operações para a "plugagem" dos **módulos específicos** implementados. Assim, observando a Tabela 6 é percebido que menos de 1/8 das linhas contadas na Tabela 5 são diretamente responsáveis por fornecer o processo de extensão definido nos **módulos padrões**.

Por fim, dada a quantidade de linhas de código apresentadas na Tabela 6 pode-se concluir que a aplicação das tarefas de processamento nos **módulos específicos** de estacionamento não foi um processo trabalhoso e para programar todos os passos de seu fluxo de dados foram necessárias apenas 55 linhas de código.

5.1.4.2 Questão de pesquisa 2: O desempenho do processo de tratamento do fluxo de dados é prejudicado em função da existência de um conjunto de passos de transformação das informações?

A Figura 37 mostra o gráfico com o tempo médio para o tráfego de mensagens em função da quantidade de pacotes enviados. Neste caso a quantidade de memória RAM para a execução da plataforma foi limitada em 512 MB e foram utilizados apenas os **módulos padrões** definidos para o fluxo de transformação.

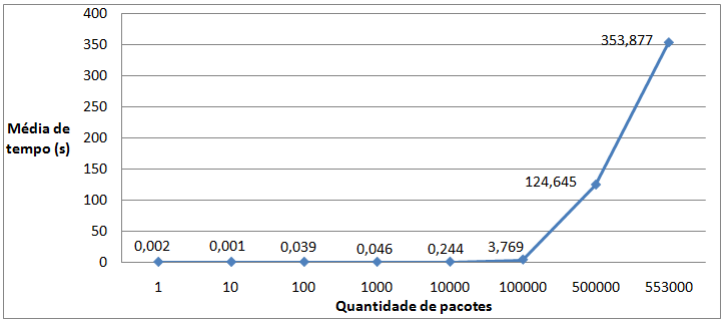


Figura 37: Média de tempo para transporte das mensagens dependendo da quantidade de pacotes (limite de 512 MB)

Assim, nota-se que o tempo médio de encaminhamento de todas as mensagens para a quantidade de 10.000 pacotes é inferior a 0,3 segundos. No entanto, quando são enviados 100.000 pacotes o tempo médio aumentou para quase 4 segundos e ao enviar-se a quantidade de 500.000 pacotes a média cresce exponencialmente para 124 segundos.

Finalmente, com o limite de memória imposto, a plataforma foi capaz de processar e entregar com 0% de perda a quantidade máxima de 553.000 pacotes recebidos ao mesmo tempo e para esse número de mensagens o tempo médio ao final das dez amostras foi

353,877 segundos.

A Figura 38 mostra os resultados de um teste semelhante ao anterior, mas neste caso o limite de memória para execução da plataforma foi estendido para 1.024 MB de RAM, a fim de avaliar como isto afeta a performance do tráfego de mensagens e a quantidade máxima de pacotes suportada por ela.

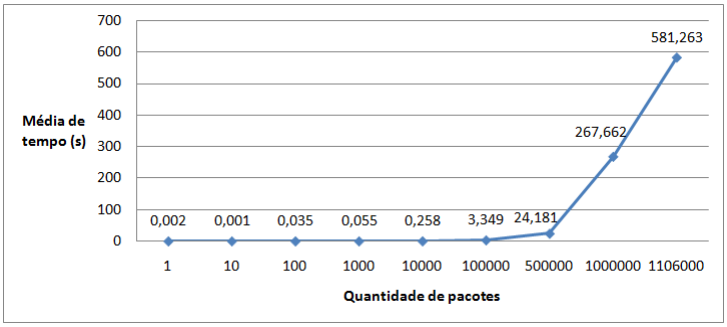
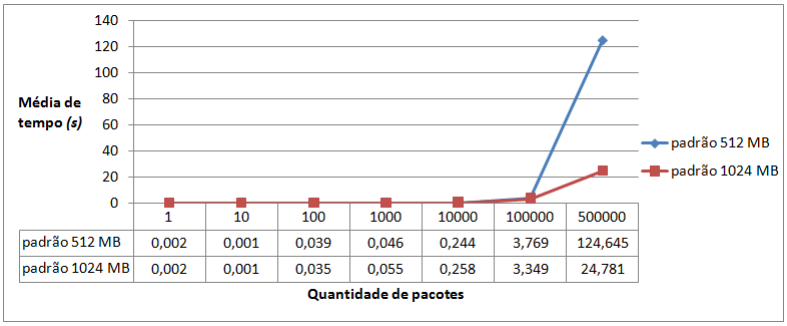


Figura 38: Média de tempo para transporte das mensagens dependendo da quantidade de pacotes (limite de 1.024 MB)

Assim, no gráfico comparativo exibido na Figura 39 é possível identificar que até a quantidade de 100.000 pacotes a velocidade média do tráfego de ambos os experimentos mantém-se praticamente a mesma. No entanto, quando se aumenta a quantidade de pacotes para 500.000 pode-se observar uma grande redução no tempo médio do experimento com limite de 1024 MB em relação ao experimento com limite de 512 MB.



	1	10	100	1000	10000	100000	500000
padrão 512 MB	0,002	0,001	0,039	0,046	0,244	3,769	124,645
padrão 1024 MB	0,002	0,001	0,035	0,055	0,258	3,349	24,781

Figura 39: Gráfico comparativo dos experimentos com limite de 512 MB e 1024 MB

Além disso, disponibilizar o dobro da memória RAM para a plataforma também duplicou o número máximo de pacotes suportados, o que mostra que a quantidade de mensagens suportado pelo *Event Admin Security* depende da quantidade de memória disponível. Assim, trabalhando com essa configuração a plataforma foi capaz de processar e distribuir uma quantidade máxima de 1.106.000 pacotes, tendo para esse valor uma média de tempo de tráfego de 581,263 segundos e uma taxa de entrega de 100%.

Portanto, com os dados de tráfego para o fluxo percebe-se que os passos de transformação de dados não prejudicam a performance da plataforma. No entanto, essa característica depende das configurações da máquina em que a mesma seja colocada em execução, pois com os resultados mostrados na Figura 39 nota-se que o tempo médio de tráfego até a quantidade de 100.000 pacotes recebidos em simultâneo mantém-se estável, porém quando esse número de mensagens é aumentado o tempo de tráfego cresce bastante devido as restrições do ambiente em que o experimento foi executado. Outro fator que pode comprovar esse fato é o momento em que se duplicou o limite de memória. Com isso, o tempo de tráfego para a quantidade de 500.000 pacotes caiu 80,11% comparado ao experimento com limite de 512 MB de memória RAM.

5.1.4.3 Questão de pesquisa 3: O desempenho do processo de tratamento do fluxo de dados é prejudicado quando são utilizados módulos específicos "plugados" nos módulos padrões?

Para verificar esta questão, os **módulos específicos** de estacionamento foram "plugados" na plataforma com o objetivo de avaliar o tempo para tráfego de pacotes quando utilizado esse tipo de módulos. Assim, foi realizado um experimento análogo aos anteriores, porém neste caso as mensagens enviadas foram os da fonte de dados estacionamento e para este teste limitou-se a memória de execução da plataforma em 512 MB. Desse modo, os resultados das médias obtidos para cada quantidade de pacotes são mostrados na Figura 40.

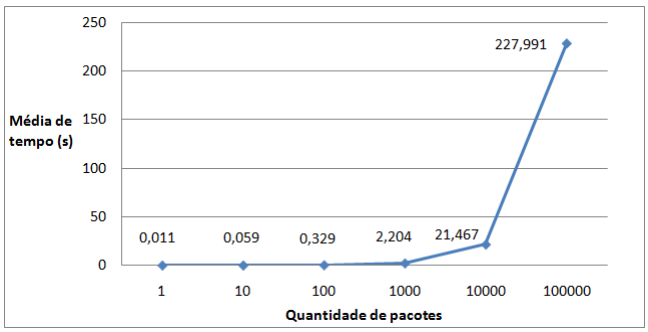
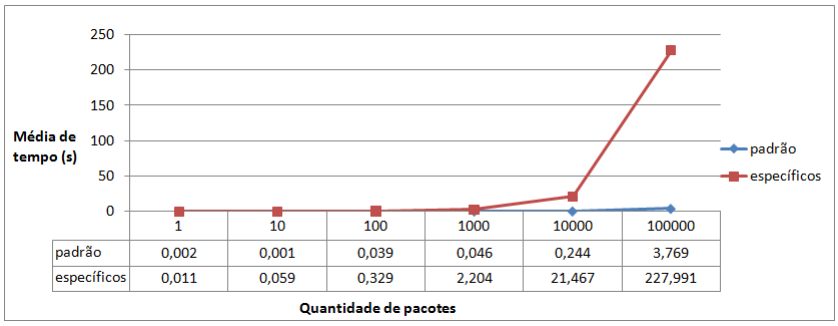


Figura 40: Média de tempo para transporte das mensagens dependendo da quantidade de pacotes utilizando os **módulos específicos** de estacionamento (limite de 512 MB)

Através do gráfico exibido na Figura 41 percebe-se que com o uso dos **módulos específicos** para realizar o processamento das mensagens do tipo de dados do estacionamento ocorre uma perda de desempenho na entrega dos pacotes quando comparado com o experimento que utiliza apenas os **módulos padrões**.

Isso já era esperado e ocorre devido ao fato dos **módulos padrões** precisarem re-



	1	10	100	1000	10000	100000
padrão	0,002	0,001	0,039	0,046	0,244	3,769
específicos	0,011	0,059	0,329	2,204	21,467	227,991

Figura 41: Gráfico comparativo entre os experimentos utilizando apenas **módulos padrões** e o que utiliza os **módulos específicos**

passar os dados aos **módulos específicos**, os quais realizam o processamento dos dados para que o **módulo padrão** publique-os em seguida. Assim, até a quantidade de 100 pacotes a diferença de tempo de tráfego comparando o primeiro experimento com o último permanece inferior a 1 segundo. Porém, para o valor de 1.000 pacotes a diferença é de 2,158 segundos. E, ao aumentar-se o número de pacotes para 10.000 essa diferença cresce para aproximadamente 21 segundos.

No entanto, o valor do tempo médio só atinge valores muito altos quando a quantidade de mensagens recebidas em simultâneo do recurso estacionamento são bastante elevadas, como se pode comprovar na Figura 40 onde até a quantidade de 1.000 mensagens o tempo médio de tráfego é de aproximadamente 2 segundos.

5.1.5 Considerações finais sobre o estudo de caso

Através da extensão realizada para trabalhar com o cenário do estacionamento pode-se incluir na plataforma dados de um primeiro tipo de fonte. Além disso, com a criação do aplicativo comprovou-se a capacidade de disponibilização do dados transformados na plataforma.

Com relação aos **módulos específicos** desenvolvidos, os mesmos seguiram as abordagens modular e "plugável" e com eles foi possível testar o fluxo de transformação de dados extensível. Assim, os mesmos realizam o processamento dos dados conforme as características e peculiaridades dos dados do recurso estacionamento para que esses sejam adequados as necessidades do aplicativo criado. Além disso, foi possível atestar o funcionamento dos **módulos padrões** e dos **módulos auxiliares** definidos neste trabalho.

Por fim, com o estudo de caso foi possível avaliar dois aspectos importantes relacionados a implementação da plataforma proposta. Estender a plataforma mostrou-se um

processo não trabalhoso de ser realizado, devido ao fato dos **módulos padrões** disponibilizarem as interfaces que definem os comportamentos associados aos mesmos, o que facilita no processo de extensão e "plugagem" dos **módulos específicos**. Além do mais, com a avaliação da performance tem-se a comprovação de que os passos definidos não oneram de maneira significativa o tráfego de mensagens na plataforma.

5.2 Estudo de caso 2: projeto do sistema de monitoramento de multidões em eventos

Controle de multidões tem a finalidade de realizar a gerência de uma grande quantidade de pessoas localizadas em um mesmo local (festa, manifestação, evento esportivo, metrô, entre outros) objetivando manter a ordem e a segurança dos presentes (ALMEIDA, 2013). Assim, o ambiente selecionado para o projeto de utilização da plataforma é o de um local onde várias pessoas estão reunidas em um evento.

Esse ambiente foi escolhido pelo fato de várias fontes de dados estarem envolvidas em um cenário desse tipo, o que possibilita a integração de diversos dados heterogêneos à plataforma. Além do mais, também se tem a oportunidade de criação de novas aplicações que forneçam funcionalidades para diferentes atores desse ambiente.

Nesse tipo de problema várias fontes de dados podem ser identificadas e utilizadas, de modo que suas informações sejam empregadas no fornecimento de novos serviços aos cidadãos ou aos órgãos responsáveis pelo gerenciamento do lugar. Assim, pode-se pensar em câmeras fornecendo imagens de diferentes locais do ambiente, *smartphones* enviando a localização dos indivíduos presentes no lugar, redes sociais fornecendo os dados de *check-in* de seus usuários no evento, conteúdo de postagens publicadas em mídias sociais, entre outras.

Como mostrado na Figura 42, nesta seção pretende-se descrever o projeto para inclusão desse conjunto de fontes de dados e o desenvolvimento dos **módulos específicos** para realizarem o gerenciamento das mesmas e para inferirem informações a partir dos dados fornecidos. Além disso, serão descritos as aplicações que proverão novos serviços às pessoas e aos órgãos gerenciadores do evento através dos dados disponibilizados pela plataforma.

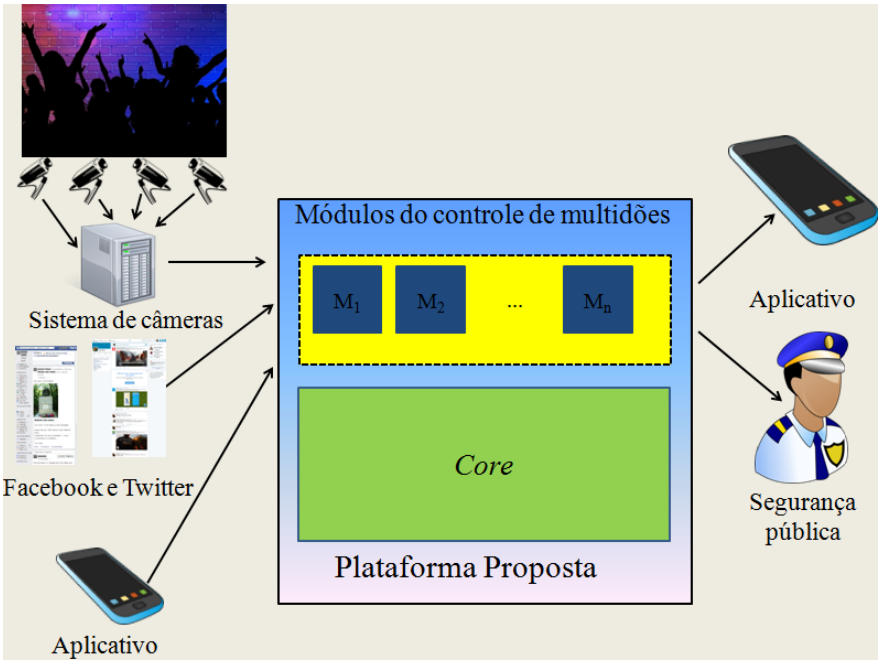


Figura 42: Plataforma trabalhando com o cenário de controle de multidões

5.2.1 Requisitos da solução

As fontes fornecedoras de dados e as informações disponibilizadas pelas mesmas para serem utilizadas na plataforma nessa descrição de solução são:

1. Um serviço que disponibiliza dados sobre a ocupação do ambiente no qual o evento está sendo realizado. Esses dados são gerados a partir de processamentos realizados sobre as imagens do circuito interno de câmeras do local. A priori, o mesmo fornece para a plataforma dados sobre o conjunto de câmeras que estão espalhadas no ambiente, identificando unicamente cada uma delas, informando o local que a mesma monitora, a área do mesmo e a capacidade máxima de pessoas suportadas pelo lugar. Além disso, baseado nas imagens coletadas pelas câmeras, esse serviço constantemente envia informações atualizadas sobre a densidade de pessoas nos ambientes do local;

2. Um aplicativo móvel que fornece ao usuário informações sobre o evento, tais como: promoções, programação, preços das bebidas, etc. Além disso, essa aplicação coleta o *e-mail* do usuário para através dessa informação identificar o perfil do mesmo no Twitter e no Facebook e colhe informações sobre a localização do mesmo no ambiente do evento. Por fim, a aplicação disponibiliza esses dados para serem tratados pela plataforma;

3. Mídias sociais que fornecerão informações dos perfis dos participantes do evento para a plataforma. Assim, o Facebook será utilizado para fornecer informações das características disponibilizadas no perfil do participante, lista de amigos do mesmo e as postagens dele. Além disso, o Twitter também será usado para fornecer informações sobre o perfil do participante e as postagens do mesmo.

As possíveis aplicações a serem criadas a partir dos dados integrados, transformados e disponibilizados pela plataforma nesta descrição de solução e suas funcionalidades são:

1. Aplicativo para paquera que baseado nas informações do perfil do Facebook de uma pessoa busca participantes do evento que possuam características parecidas com as dela. Além disso, esse aplicativo pode indicar em quais locais as pessoas listadas se encontram, ajudando o usuário para que ele saiba para onde deve se deslocar caso queira encontrá-las;

2. Aplicativo para recomendação de locais ao usuário, o qual baseado nos dados de quantidade de pessoas nos ambientes do local recomenda lugares com menor ocupação para que o usuário fique mais confortável no evento;

3. Aplicativo para exibir a avaliação do evento baseado em postagens dos participantes publicadas através do Twitter e Facebook. Com isso, os usuários podem ter uma visão geral da satisfação das pessoas para com o evento;

4. Aplicação de segurança pública que forneça os dados de ocupação dos ambiente do local onde o evento está sendo realizado. Além disso, essa aplicação lançará alertas informando caso uma área ultrapasse a capacidade máxima de pessoas para que os responsáveis pela segurança do local tomem medidas no intuito de evitar problemas devido a superlotação. Também serão gerados alertas para informar quando locais estão se a aproximando de atingir sua capacidade máxima para que os profissionais de segurança possam atentar para os mesmos e prevenir uma possível superlotação.

5.2.2 Projeto da solução

Para a solução de controle de multidões ser executada é necessário que sejam gerados os **módulos específicos** que serão responsáveis por transformar esses dados na plataforma, conforme mostrado na Figura 43. Vale destacar que essa solução pode variar de acordo com o projetista do sistema e que esse conjunto de módulos foram projetados por decisão do pesquisador deste estudo.

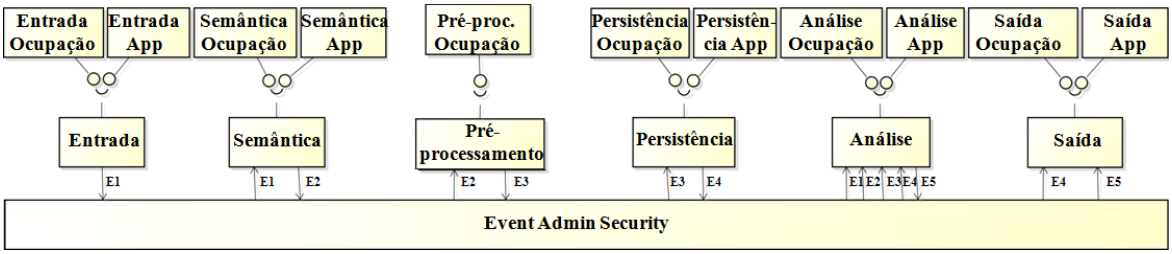


Figura 43: Implementação da arquitetura para o cenário controle de multidões

Logo, os módulos pensados e suas responsabilidades são:

- **Entrada ocupação**: esse módulo tem a função de receber os dados do serviço de câmeras, as quais realizam o monitoramento do ambiente no qual o evento está acontecendo. Dessa forma, esse serviço fornece a identificação de cada câmera, o ambiente supervisionado pelas mesmas, a área do local monitorado e a capacidade máxima de pessoas comportado por eles. Além disso, esse módulo recebe constantemente informações atualizadas da quantidade de pessoas em cada ambiente do evento;

- **Entrada App**: responsável por receber os dados do aplicativo utilizado pelos participantes da festa e as informações das mídias sociais Twitter e Facebook;

- **Semântica Ocupação**: recebe os dados integrados à plataforma por meio do módulo **Entrada Ocupação** e é encarregado de representar os mesmos em classes para facilitar a manipulação dessas informações na mesma;

- **Semântica App**: também na intenção de simplificar a manipulação das informações, o mesmo tem a incumbência de representar em objetos os dados do aplicativo de participantes do evento recebidos através do módulo **Entrada App**;

- **Pré-processamento Ocupação**: realiza uma filtragem nos dados que utilizam a representação escolhida em **Semântica Ocupação**. Assim, esse módulo elimina as duplicações ocasionadas pela interseção dos ambientes monitorados pelas câmeras e seleciona os dados centrais ao problema, ou seja, a lista de ambientes, a área deles, a capacidade máxima dos mesmos e a quantidade de pessoas que os ocupam;

- **Persistência Ocupação**: é encarregado de efetuar o armazenamento das informações pré-processadas no módulo **Pré-processamento Ocupação**;

- **Persistência App**: tem a função de armazenar os dados representados no módulo **Semântica App**. A criação de um módulo para pré-processar esses dados não

se faz necessário em razão das informações geradas no aplicativo de participantes do evento não precisar passar por uma etapa de limpeza e em virtude do módulo **Entrada App** buscar no Twitter e Facebook apenas as informações essenciais a serem utilizadas nesta solução;

- **Análise Ocupação**: esse módulo utiliza os dados repassados tanto por **Persistência Ocupação** quanto por **Persistência App**. Assim, ele constantemente processa as informações da ocupação dos ambientes provenientes do servidor de câmeras e os dados do posicionamento dos participantes da festa oriundos do aplicativo em execução nos seus dispositivos móveis. E, além disso, os agrega para identificar quando um ambiente do local no qual o evento está acontecendo se encontra sobrecarregado de pessoas ou quando se aproxima da superlotação. Desse modo, ao identificar algum ambiente nessas situações, **Análise Ocupação** solicita que o módulo **Análise** publique essa informação.

- **Análise App**: esse módulo utiliza os dados armazenados por **Persistência App** para identificar pessoas na festa que têm características parecidas. Além disso, ele calcula a localização dos participantes no local do evento e com base nesse cálculo infere os locais menos ocupados do ambiente. Esse módulo também processa as postagens encontradas no Twitter e Facebook que falam sobre o evento para identificar se o mesmo está sendo bem ou mal avaliado;

- **Saída Ocupação**: tem a função de disponibilizar os dados da ocupação dos ambientes do local do evento repassados por **Persistência Ocupação** e os alertas de sobrecarregamento de um ambiente gerados por **Análise Ocupação**;

- **Saída App**: disponibiliza os dados gerados nos processamentos realizados por **Análise App**.

Na Figura 44 mostra-se as etapas percorridas pelos dados desse cenário na plataforma, através dessa ilustração percebe-se que **Análise Ocupação** recebe dados de **Persistência Ocupação** e de **Persistência App**. Isso acontece porque esse módulo utiliza as informações de ambas as fontes para identificar superlotação nos ambientes do local em que o evento está acontecendo para em seguida notificar **Saída Ocupação** que uma ocorrência desse tipo foi detectada.

Como descrito antes na Seção 4.2.4, o processo de agregação de dados está ligado diretamente ao sistema de checagem de permissões. Dessa forma, para um módulo realizar

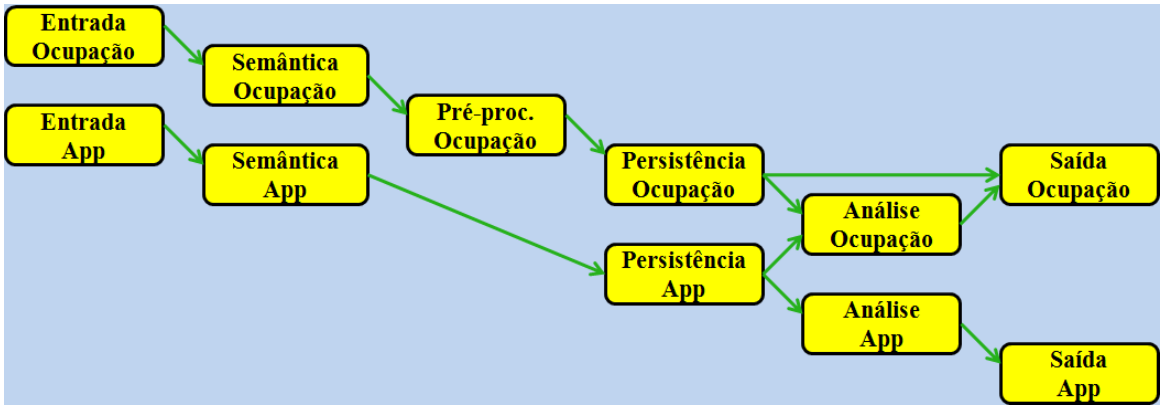


Figura 44: Etapas percorridas pelos dados do cenário de controle de multidões

a fusão de diferentes tipos de dados, ele precisa possuir o conjunto de permissões para receber esses dados que juntos formarão um novo tipo de dado.

A Figura 45 mostra um exemplo no qual são destacadas as permissões necessárias para o módulo **Análise Ocupação** receber os dois tipos de dados usados por ele para identificar a superlotação de ambientes. De início, **Persistência** publica no ***Event Admin Security*** os dados persistidos pelos módulos **Persistência Ocupação** (PeO) e **Persistência App** (PeA). Em seguida, esses dados são entregues a **Análise**, o qual após solicitar a checagem de permissões nos seus respectivos **módulos específicos** identificará que **Análise Ocupação** está autorizado a receber ambos. Assim, esse módulo constantemente recebe e processa esse conjunto de dados e quando identifica a superlotação de algum ambiente ou quando um deles se aproxima dessa situação, ele cria um novo dado (AnO) e solicita que seu **módulo padrão** o publique. Por fim, esse dado chega a **Saída** que o repassa ao **módulo específico** autorizado a recebê-lo, o qual o disponibiliza para utilização dessa informação nas aplicações.

Assim, pode-se criar uma aplicação que mostre a concentração de pessoas em diferentes locais do ambiente gerenciado. Desse modo, a mesma pode ser usada pela polícia, bombeiros e/ou responsáveis pelo evento para servirem de base na tomada de decisões relacionada a segurança do lugar.

5.2.3 Considerações finais sobre o estudo de caso

Embora este estudo de caso não tenha sido implementado, ele já demonstra bem as características de agregação de dados e extração de conhecimento da plataforma proposta. Assim, através do mesmo é proporcionada uma visão das tarefas que devem ser realizadas para que tais capacidades possam ser usadas em outros cenários.

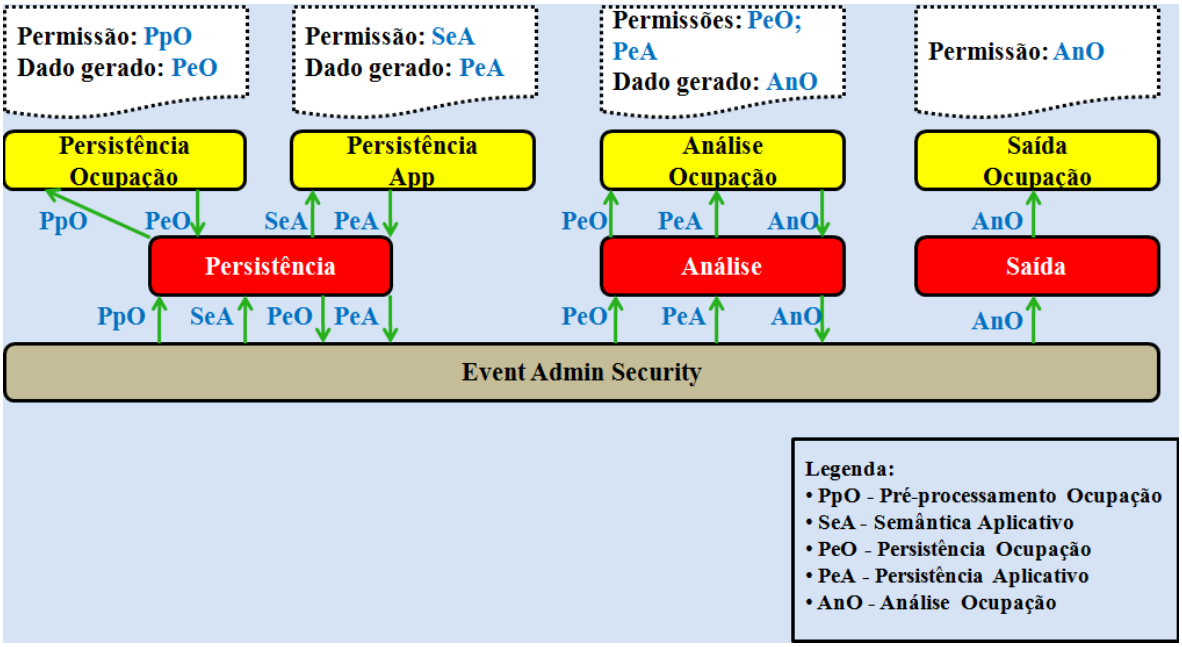


Figura 45: Permissões para agregação de dados em Análise Ocupação

Além disso, caso venha a surgir uma nova fonte de dados para ser utilizada na solução de controle de multidões existem duas possibilidades para que a mesma seja incorporada à plataforma. A primeira delas é a criação de novos **módulos específicos** para trabalharem com os dados da nova fonte. A segunda é a modificação dos módulos de **Ocupação** ou **App** para que eles também passem a suportar os novos tipos de dados.

Por fim, existem duas possibilidades para que uma nova aplicação pensada para esse cenário possa se utilizar dos dados do mesmo. O primeiro caso é o da plataforma já disponibilizar todos os dados necessários para a criação da aplicação, assim, a mesma precisará apenas consumi-los. A segunda possibilidade é o caso dos dados necessários ainda precisarem passar por algum processamento na plataforma, o que pode ser feito tanto através da criação de novos **módulos específicos** como pela alteração dos existentes.

6 Considerações finais

Este capítulo apresenta as consideração finais deste estudo e encontra-se organizado da seguinte forma: na Seção 6.1 é relembrado de forma sucinta cada capítulo do trabalho, a Seção 6.2 lista algumas conclusões desta pesquisa, na Seção 6.3 são enumeradas as contribuições do estudo, a Seção 6.4 aborda as suas limitações e na Seção 6.5 discorre-se sobre os trabalhos futuros.

6.1 Retrospectiva do trabalho

Conforme mostrado no Capítulo 1, o grande aumento populacional ocorrido nas cidades ao longo dos anos ocasionou um sobrecarregamento dos serviços oferecidos nessas localidades e, consequentemente, o sucateamento dos mesmos. Além disso, é visto que a heterogeneidade das fontes de dados urbanos acarreta em problemas de interoperabilidade.

De acordo com o que foi apresentado no Capítulo 2, percebe-se que vários esforços têm sido desenvolvidos no campo de cidades inteligentes almejando proporcionar melhoria na qualidade de vida na zona urbana. Ademais, conclui-se que o termo *Smart City* não apresenta uma padronização com relação ao seu significado apesar de ser muito utilizado na atualidade. Ainda nesse capítulo, é possível ver que a modularidade proporcionada pelo *framework* OSGi gera benefícios como: menor complexidade devido a divisão do problema em partes menores, simplificação do *deploy* visto que se pode gerenciar cada módulo individualmente, facilitação da manutenção do sistema, etc.

No Capítulo 3, nota-se que nenhum dos trabalhos analisados na revisão exploratória propõe em suas soluções um fluxo de transformação de dados com a capacidade de extensibilidade, característica que permite a adequação do processamento dos dados para que ele seja realizado conforme as características e restrições de cada tipo existente nos "ecossistemas" de uma cidade inteligente.

O Capítulo 4, mostra que a extensibilidade do fluxo de transformação é importante

porque não se pode prever a maneira como os diferentes tipos de dados precisam ser processados para ser entregues às aplicações e em virtude de que com o passar do tempo novos dados irão surgir e também necessitarão ser "plugados" na plataforma. Além disso, a plataforma foi implementada na tecnologia OSGi em razão das facilidades de gerenciabilidade e manutenibilidade oferecidas pela mesma, as quais são essenciais para a utilização das abordagens modular e "plugável" atendidas pela arquitetura. Por fim, devido as semelhanças com o padrão *Pipe-and-filter* os componentes da plataforma são mais fáceis de serem reusados em razão da utilização de pequenos passos de processamento, a troca de um módulo é mais fácil em decorrência do baixo acoplamento e o sistema pode ser estendido e modificado mais facilmente. Por outro lado, aumenta-se a sobrecarga no processo de transformação de dados deixando-o mais lento.

Com o estudo de caso executado no Capítulo 5, a plataforma pôde trabalhar com um primeiro tipo de fonte e através do aplicativo criado foi possível comprovar a capacidade de disponibilização de dados da mesma. Além disso, avaliou-se dois aspectos importantes relacionados a implementação da plataforma proposta: extensibilidade e performance. A primeira mostrou-se um processo não trabalhoso de ser realizado, em razão da pouca quantidade de linhas de código necessárias à implementação da "plugagem" dos **módulos específicos**. Com relação a performance, comprovou-se que a existência dos passos de transformação de dados não oneram de forma significativa o tráfego de mensagens na plataforma. Por fim, a descrição do projeto de utilização da plataforma no cenário de controle de multidões oferece uma visão das atividades necessárias para a realização de agregação de dados e extração de conhecimento na mesma.

6.2 Conclusões

A definição de um processo de transformação extensível para os dados foi importante para proporcionar maior flexibilidade ao tratamento dos mesmos. Assim, eles podem ser tratados conforme suas peculiaridades, tem-se a possibilidade de existirem diferentes tratamentos para um mesmo tipo de dado e facilita-se a criação de tratamentos para um novo tipo de dado que se queira inserir na plataforma.

Apesar do trabalho ter partido do contexto de cidades inteligentes, a flexibilidade proporcionada pela extensibilidade das transformações de dados definida e implementada na plataforma permite que a mesma seja usada em outros ambientes de Internet das Coisas onde também se tenha o problema de integração de dados heterogêneos.

6.3 Contribuições

Neste trabalho destacam-se as seguintes contribuições:

1. Definição da plataforma extensível capaz de receber dados de fontes heterogêneas, transformá-los e disponibilizá-los possibilitando a criação de novas aplicações. O fluxo de transformação extensível definido na mesma permite que o processamento dos dados possa ser adequado para trabalhar de acordo com as características de cada tipo existente no ecossistema, o que é importante devido ao fato de não haver como prever a forma como eles precisarão ser transformados visto que cada aplicação pode necessitar que os mesmos sejam disponibilizados obedecendo processamentos específicos. Além disso, novos tipos de dados também surgirão com o passar do tempo e precisarão ser transformados de acordo com as restrições deles, o que é possibilitado através desse requisito de extensibilidade;

2. Implementação da proposta de plataforma usando a linguagem de programação Java e o *framework* OSGi, o que possibilita a utilização da mesma de forma prática e propicia seu uso para integrar dados de cenários de uma cidade inteligente;

3. Implementação do estudo de caso do cenário estacionamento com o qual foi possível adicionar à plataforma dados de uma primeira fonte, atestando-se o funcionamento dos diferentes tipos de módulos projetados e do fluxo de transformação extensível da mesma;

4. Avaliação da plataforma através da realização de um estudo experimental cujo objetivo principal foi avaliar dois aspectos comportamentais relacionados a implementação da mesma: a característica de criação de **módulos específicos** (**Extensibilidade**) e a capacidade de processamento de dados (**Performance**);

5. Descrição do projeto de utilização da plataforma para a implementação da solução de controle de multidões, através do qual explica-se as tarefas necessárias para a realização de agregação de dados e extração de conhecimento possibilitadas pela mesma.

Além das contribuições listadas acima, este trabalho gerou alguns resultados que estão reportados na literatura, os quais são:

- SILVA, C. A.; JÚNIOR, G. S. A. An extensible platform for the transformation of heterogeneous data in smart cities. In: *XXI Brazilian Symposium on Multimedia and the Web (WebMedia'15)*. Manaus: ACM, 2015.

- SILVA, C. A.; JÚNIOR, G. S. A. An extensible platform for the treatment of heterogeneous data in smart cities. In: *The Tenth International Conference on Software Engineering Advances (ICSEA)*. Barcelona: IARIA, 2015.

6.4 Limitações

O levantamento dos trabalhos analisados no Capítulo 3 não seguiu uma abordagem sistemática devido essa tarefa ter tido como objetivo situar os pesquisadores no campo de arquiteturas para cidades inteligentes permitindo a identificação de características importantes a serem exploradas nessa área. Consequentemente, a reprodução dessa atividade torna-se difícil ou até mesmo impossível de ser realizada em virtude da subjetividade dessa metodologia. No entanto, a revisão exploratória por ser um procedimento mais flexível permitiu a inclusão de trabalhos reportados em páginas *web* oportunizando a análise de plataformas descritas fora dos meios acadêmicos de publicação.

A plataforma descrita no Capítulo 4, possui uma estratégia de permissões apenas para o consumo dos dados. Porém, é importante que seja definida uma outra estratégia relacionada a publicação dos mesmos, impedindo que módulos específicos publiquem informações falsas para um determinado tipo de dado.

Como mostrado no Capítulo 5, a plataforma foi utilizada para trabalhar apenas com os dados do cenário de um estacionamento inteligente. Isso ocorreu em decorrência da complexidade de se utilizar um cenário real, o qual exige um grande esforço de várias pessoas para desenvolver uma solução completa e, por isso, foge do escopo de uma pesquisa de mestrado. Porém, apesar do estudo de caso ser limitado, ele mostra bons indícios das características da plataforma bem como possibilita a visualização das limitações da mesma.

Outra limitação relacionada ao estudo de caso, é o fato do mesmo ter avaliado apenas os aspectos extensibilidade e performance da plataforma. Essas características foram escolhidas para análise em função delas estarem diretamente relacionadas com a capacidade de extensão do processo de transformação de dados da mesma. Assim, o estudo dessas capacidades já são um passo importante no processo de avaliação da plataforma, pois através das mesmas foi possível identificar que o processo de extensão dela não é uma

tarefa trabalhosa e a performance não se torna significativamente onerosa em função de existência dos passos de transformação de dados.

6.5 Trabalhos futuros

Como trabalho futuro pretende-se tornar a plataforma escalável de forma que a mesma permita a expansão de sua capacidade de atendimento, suportando o crescimento dos usuários e da quantidade de informações trafegadas. Assim, serão avaliados os conceitos de *cloud computing* e outras técnicas que permitam atender esse requisito para, em seguida, realizar a escolha da abordagem a ser implantada.

Além disso, tem-se a intenção de incrementar à plataforma técnicas de *big data*, possibilitando a extração de conhecimento do grande volume de dados gerado nos dispositivos espalhados no meio urbano. E, assim, ajudando no processo de tomada de decisão nos diversos setores envolvidos no ambiente das cidades.

O requisito de monetização pode ser mais aprofundado de forma a explorar as regras de negócio relacionadas a cobrança para o acesso dos dados. Assim, almeja-se desenvolver um sistema de cadastro dos usuários que por meio de contas possa associá-los aos dados consumidos por eles. Além disso, tem-se a intenção de implementar um mecanismo que possa gerar as faturas relacionadas a essas contas e desenvolver um sistema responsável pelas recargas das mesmas. Também se faz necessário o desenvolvimento de um *software* que realize todo o gerenciamento monetário da plataforma possibilitando, por exemplo, pagamentos utilizando cartões de crédito, entre outros requisitos.

Por fim, é essencial a criação de um catálogo de dados que documente todas as informações geradas nos módulos específicos "plugados" à plataforma. Desse modo, pretende-se criar um dicionário de dados que possa informar quais os tipos criados em cada módulo da plataforma e os formatos nos quais os mesmos são disponibilizados, de forma que tais informações possam servir de base para os desenvolvedores que pretendam fazer uso de algum dado integrado para criar novos módulos e aplicações.

Referências

ALLIANCE, O. *OSGi Service Platform*. [S.l.], 2011.

ALLIANCE, O. *OSGi Core Release 5*. [S.l.], 2012.

ALLIANCE, O. *OSGi Core Release 6*. [S.l.], 2014.

ALMEIDA, J. L. P. *Gestão de eventos desportivos: o controlo de multidões e os seus intervenientes na segurança dos estádios*. Dissertação (Mestrado) — Universidade Técnica de Lisboa, 2013.

ALNASERA, W. E.; ALNASERB, N. W. The status of renewable energy in the gcc countries. *Renewable and Sustainable Energy Reviews*, 2011.

ANASTASI, G. et al. Urban and social sensing for sustainable mobility in smart cities. In: *Sustainable Internet and ICT for Sustainability (SustainIT)*. Palermo: IEEE, 2013.

ANDREINI, F. et al. A scalable architecture for geo-localized service access in smart cities. In: *Future Network and Mobile Summit (FutureNetw)*. Warsaw: IEEE, 2011.

ANGELIDOU, M. Smart city policies: A spatial approach. *Cities*, 2014.

ANTHOPOULOS, L.; FITSILIS, P. From digital to ubiquitous cities: Defining a common architecture for urban development. In: *Sixth International Conference on Intelligent Environments (IE)*. Kuala Lumpur: IEEE, 2010.

ATZORI, L.; IERA, A.; MORABITO, G. The internet of things: A survey. *Computer Networks*, 2010.

BABBIE, F. *The practice of social research*. Estados Unidos: Thomson Learning, Inc, 2007.

BAKICI, T.; ALMIRALL, E.; WAREHAM, J. A smart city initiative: the case of barcelona. *Journal of the Knowledge Economy*, 2013.

BAKKER, P.; ERTMAN, B. *Building Modular Cloud Apps with OSGi*. EUA: O Reilly, 2013.

BLACKSTOCK, M. et al. Magic broker 2: An open and extensible platform for the internet of things. In: *Internet of Things (IOT)*. Tóquio: IEEE, 2010.

BORJA, R.; GAMA, K. Middleware para cidades inteligentes baseado em um barramento de serviços. In: *X Simpósio Brasileiro de Sistemas de Informação*. Londrina: Sociedade Brasileira de Computação, 2014.

BUSCHMANN, F. et al. *Pattern-oriented software architecture: a system of patterns*. Inglaterra: Wiley, 1996.

CAMPOS, J. B. N. *Planejamento Urbano: a cristalização dos processos sociais nas linhas das cidades inteligentes*. Dissertação (Mestrado) — Universidade do Porto, 2012.

CARRIOTS. *Carriots*. 2011. Disponível em: <https://www.carriots.com/>. Acesso em: Junho 4, 2015.

COMPUTERWORLD. *SmartCities*. 2013.

CUGOLA, G.; MARGARA, A. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 2012.

CUNHA, A. B. *Prefeito lança o 1746: Central Única de Teleatendimento*. 2011. Disponível em: <http://www.rio.rj.gov.br/web/guest/exibeconteudo?article-id=1646117>. Acesso em: Setembro 09, 2014.

CYBERVISION. *Kaa*. 2014. Disponível em: <http://www.kaaproject.org/>. Acesso em: Junho 4, 2015.

DAVID, F. *Prefeitura lança "Rio Smart City" e instala adesivos inteligentes nos pontos de ônibus*. 2014. Disponível em: <http://www.rio.rj.gov.br/web/guest/exibeconteudo?id=4772022>. Acesso em: Setembro 06, 2014.

DIGI. *Digi*. 2013. Disponível em: <http://www.digi.com/cloud-overview>. Acesso em: Junho 4, 2015.

DOBBS, R. et al. *Urban world: Mapping the economic power of cities*. [S.l.]: McKinsey Global Institute, 2011.

DOOLEY, J. *Software Development and Professional Practice*. Estados Unidos: Apress, 2011.

FARDOUN, H. M.; ALTALHI, A. H.; LÓPEZ, S. R. Active citizenship: A system to inform about problems to the local entities. In: *Proceedings of the 13th International Conference on Interacción Persona-Ordenador*. Elche: ACM, 2012.

FILIPPONI, L. et al. Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In: *Fourth International Conference on Sensor Technologies and Applications*. Veneza: IEEE, 2010.

FIWARE. *Fiware open apis for open minds*. 2011. Disponível em: <http://www.fiware.org/>. Acesso em: Junho 4, 2015.

GAMA, K.; TOUSEAU, L.; DONSEZ, D. Combining heterogeneous service technologies for building an internet of things middleware. *Computer Communications*, 2012.

GELLI, T. *Rio é eleita a cidade mais inteligente e conectada do país*. 2015. Disponível em: <http://www.rio.rj.gov.br/web/guest/exibeconteudo?id=5501443>. Acesso em: Agosto 12, 2015.

GIFFINGER, R. et al. *Smart cities: Ranking of European medium-sized cities*. Viena, 2007.

GRECO, I.; BENCARDINO, M. The paradigm of the modern city: Smart and senseable cities for smart, inclusive and sustainable growth. In: *Computational Science and Its Applications (ICCSA)*. Guimarães: Springer International Publishing, 2014.

GROVER, A. Airport business districts - an indispensable reality. *Creative Space*, 2013.

HALL, R. S. et al. *OSGi in Action: Creating modular applications in Java*. Estados Unidos: Manning, 2011.

HARRISON, C.; DONNELLY, I. A. A theory of smart cities. In: *Proceedings of the 55th Annual Meeting of the ISSS*. Hull: [s.n.], 2011.

HAUBENSAK, O. Smart cities and internet of things. In: *Business Aspects of the Internet of Things, Seminar of Advanced Topics*. Zurique: ETH, 2011.

HERNANDEZ, J. F.; LARIOS, V. M. Cloud computing architecture for digital services into smart cities. *IEEE Smart Cities*, 2014.

HODGKINSON, S. Is your city smart enough? *Digitally enabled cities and societies will enhance economic, social, and environmental sustainability in the urban century*, 2011.

HOHPE, G.; WOOLF, B. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. [S.l.]: Addison-Wesley Professional, 2004.

IBD, S. *A global business hub*. 2014. Disponível em: <http://www.songdo.com/songdo-international-business-district/why-songdo/global-business-hub.aspx>. Acesso em: Setembro 08, 2014.

IBGE. *Censo Demográfico 2010*. Rio de Janeiro, 2010.

IBGE. *Projeção da população do Brasil e das Unidades da Federação*. 2015. Disponível em: <http://www.ibge.gov.br/apps/populacao/projecao/index.html>. Acesso em: Janeiro 03, 2015.

INDRA. *Sofia2 inCloud*. 2014. Disponível em: <http://sofia2.com/sofia2incloud_en.html>. Acesso em: Junho 4, 2015.

INTERNATIONAL, G. *Songdo IBD*. 2014. Disponível em: <http://www.songdo.com/>. Acesso em: Setembro 8, 2014.

JANEIRO, P. R. *Centro de Operações Prefeitura do Rio*. 2015. Disponível em: <http://www.centrodeoperacoes.rio.gov.br/>. Acesso em: Fevereiro 24, 2015.

KITCHENHAM, B.; PICKARD, L.; PFLEEGER, S. L. Case studies for method and tool evaluation. *IEEE software*, 1995.

KOTLER, P.; ARMSTRONG, G. *Principles of Marketing*. Estados Unidos: Pearson Prentice Hall, 2012.

KUPER, A.; KUPER, J. *The Social Science Encyclopedia*. Londres e Nova Iorque: Routledge, 2003.

MASDAR. *Masdar City*. 2011. Disponível em: <http://masdarcity.ae/en/>. Acesso em: Setembro 8, 2014.

MELENDRERAS-RUIZ, R.; GARCIA-COLLADO, A. Mobisec: An european experience directed towards improving cities through citizen participation. In: *International Conference on New Concepts in Smart Cities: Fostering Public and Private Alliances (SmartMILE)*. Gijon: IEEE, 2013.

MURATA, Y.; SAITO, S. Proposal of "cyber parallel traffic world" cloud service. In: *Proceedings of the 2014 ITU Kaleidoscope Academic Conference: Living in a converged world - Impossible without standards?* St. Petersburg: IEEE, 2014.

MURGANTE, B.; BORRUSO, G. Smart city or smurfs city. In: *Computational Science and Its Applications (ICCSA)*. Guimarães: Springer International Publishing, 2014.

NAM, T.; PARDO, T. A. Conceptualizing smart city with dimensions of technology, people, and institutions. In: *12th Annual International Conference on Digital Government Research*. Nova Iorque: ACM, 2011.

NATIONS, U. *World Population Prospects: The 2012 Revision*. 2012. Disponível em: <http://esa.un.org/unpd/wpp/unpp/panel_population.htm>. Acesso em: Dezembro 26, 2014.

NATIONS, U. *World Urbanization Prospects*. Nova Iorque, 2014.

NEIROTTI, P. et al. Current trends in smart city initiatives: Some stylised facts. *Cities*, 2014.

OCTOBLU. *Octoblu: The Integration of Everything*. 2014. Disponível em: <https://www.octoblu.com/>. Acesso em: Junho 4, 2015.

PLANIT, L. *PlanIT Valley*. 2010. Disponível em: <http://www.living-planit.com/images/bank/planIT_valley/screen/Image\20Identification\%20PlanIT\%20Valley\%202011\%20006.jpg>. Acesso em: Agosto 24, 2014.

PLANIT, L. *Living PlanIT: Improving Quality of Life Through Technology*. 2013. Disponível em: <http://www.living-planit.com/>. Acesso em: Abril 29, 2015.

PRISMTECH. *Kaa*. 2014. Disponível em: <http://www.prismtech.com/vortex>. Acesso em: Junho 4, 2015.

RUNESON, P.; HÖST, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 2009.

SANCHEZ, L. et al. Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In: *Future Network e Mobile Summit (FutureNetw)*. Warsaw: IEEE, 2011.

SEECONTROL. *SeeControl*. 2014. Disponível em: <http://www.seecontrol.com/>. Acesso em: Junho 4, 2015.

SILVA, E. C. G. F. et al. Um survey sobre plataformas de mediação de dados para internet das coisas. In: *Seminário Integrado de Software e Hardware (SEMISH)*. Recife: Congresso da Sociedade Brasileira de Computação (CSBC), 2015.

SIQUEIRA, M. M. Transetoriedade dos serviços urbanos: preâmbulos de discussão. *Revista de Administração Pública*, 1998.

SIQUEIRA, M. M. Redes sociais na gestão de serviços urbanos. *Revista de Administração Pública*, 2000.

SMARTSANTANDER. *Santander on fire future internet research e experimentation*. 2015. Disponível em: <http://www.smartsantander.eu/?template=light>. Acesso em: Abril 15, 2015.

SU, K.; LI, J.; FU, H. Smart city and the applications. In: *International Conference on Electronics, Communications and Control (ICECC)*. Zhejiang: IEEE, 2011.

TOMAS, G. H. R. P. *Uma arquitetura para Cidades Inteligentes baseada na Internet das Coisas*. Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2014.

TOPPETA, D. The smart city vision: How innovation and ict can build smart, "liveable", sustainable cities. *The Innovation Knowledge Foundation*, 2010.

VALENTE, B.; MARTINS, F. A middleware framework for the internet of things. In: *The Third International Conference on Advances in Future Internet*. French Riviera, Nice/Saint Laurent du Var, França: IARIA, 2011.

WU, H. et al. A framework for integrating heterogeneous spatial information and applications adaptively based on multi-agent and web service. In: *Third International Conference on Multimedia Information Networking and Security (MINES)*. Shanghai: IEEE, 2011.

YIN, R. K. *Case Study Research: Design and Methods*. [S.l.]: SAGE Publications, 2003.

ZANELLA, A. et al. Internet of things for smart cities. *Internet of Things Journal*, 2014.

ZYGIARIS, S. Smart city reference model: Assisting planners to conceptualize the building of smart city innovation ecosystems. *Journal of the Knowledge Economy*, 2012.