

João Pedro Schmitt

**Plataforma de Desenvolvimento de
Sistemas para Internet das Coisas (IoT)**

Monografia apresentada no curso de Graduação do Centro Universitário Católica de Santa Catarina como requisito parcial para obtenção do certificado do curso.

Jaraguá do Sul
2015

João Pedro Schmitt

Plataforma de Desenvolvimento de Sistemas para Internet das Coisas (IoT)

Monografia apresentada no curso de Graduação do Centro Universitário Católica de Santa Catarina como requisito parcial para obtenção do certificado do curso.

Área de Concentração: Ciência da Computação

Orientador: Manfred Heil Júnior
Co-orientador: Maurício Heinning

Jaraguá do Sul
2015

Schmitt, Joao Pedro
Plataforma de Desenvolvimento de Sistemas para Internet das Coisas (IoT).
Jaraguá do Sul, 2015.

Monografia - Centro Universitário Católica de Santa Catarina.

1. Internet das Coisas
2. Computação em Nuvem
3. Plataforma I.Centro Universitário Católica de Santa Catarina. Curso de Bacharaledo em Sistemas de Informação.

Dedico este trabalho a todos meus amigos e familiares que me apoiaram durante todo meu período de estudos.

Agradecimentos

Agradeço aos meus pais Sandra e Paulo por terem me dado suporte durante toda minha vida, me mostrando os caminhos certos e me auxiliando nas minhas escolhas, agradeço a meus amigos de sala de aula Joe Jonas e Fabrício Konell pela parceria em vários projetos realizados durante nossa faculdade, agradeço também a todos meus professores pelo apoio e pelo conhecimento compartilhado em especial o Prof. Manfred, meu orientador que me ajudou muito em meu crescimento profissional e intelectual. De forma geral agradeço a todos os meus amigos e familiares que sempre me apoiaram e indicaram os caminhos certos da vida.

Sumário

Agradecimentos	ii
Sumário	iii
Lista de Figuras	vi
Lista de Tabelas	viii
Lista de Algoritmos	ix
Lista de Símbolos	x
Lista de Abreviações	xi
Resumo	xiv
Abstract	xvi
Capítulo 1	
Introdução	1
1.1 Problema	2
1.2 Justificativa	2
1.3 Objetivos	3
1.3.1 Objetivo Geral	3
1.3.2 Objetivos Específicos	3
1.4 Procedimentos Metodológicos	3
1.5 Estrutura do Trabalho	4
Capítulo 2	
Fundamentação Teórica	5
2.1 Computação Distribuída	5
2.2 Computação em nuvem	6
2.2.1 Compartilhamento de serviços	7
2.3 O que é Internet das Coisas	8
2.3.1 Tecnologias Ativas	9

2.3.1.1	Aplicações	10
2.3.1.2	Composição de serviços	11
2.3.1.3	Gerenciamento de serviços	11
2.3.1.4	Abstração de objeto	11
2.3.2	Semântica	11
2.3.3	OGC Standarts	13
2.3.4	ZigBee	15
2.3.5	WEB 3.0	16
2.3.6	Social WEB das Coisas	17
2.4	Arquiteturas de Plataformas	18
2.4.1	Modelo de Distribuição de Dados	18
2.4.2	Modelo de Gerenciamento de Serviços	19
2.4.3	Modelo do Serviço Nuvem	21
2.5	Conclusões	23

Capítulo 3

Desenvolvimento	24	
3.1	Importância do Sistema	24
3.2	Projeto Relacionados	25
3.3	Limitações	26
3.4	Desenvolvimento do Sistema	26
3.5	Arquitetura de Comunicação	27
3.6	Modelagem do Sistema	28
3.6.1	Modelo de Dados	29
3.6.2	Servidor de Aplicação	31
3.6.2.1	Modelo Persistência	32
3.6.2.2	Segurança da Aplicação	33
3.6.2.3	Comunicação entre Middleware e o Servidor	37
3.6.3	Programa do Middleware	40
3.6.3.1	Modelagem componentes do Middleware	44
3.6.3.2	Modelagem core do Middleware	45
3.7	Conclusão	46

Capítulo 4

Controlando um Abrigo de Cultivo com a Plataforma Wireway	47
4.0.1 Programa de comunicação do Abrigo de Cultivo	47
4.0.2 Rádio XBee	48
4.0.3 Placa coletora	50
4.0.4 Middleware controlador	51
4.1 Conclusão	54
Capítulo 5	
Conclusões	58
Capítulo A	
Apêndice Circuito Coletor	60
Capítulo B	
Apêndice Telas da Aplicação WEB	63
Referências Bibliográficas	66

Lista de Figuras

2.1	Arquitetura SOA para Middleware IoT (ATZORI; MORABITO, 2010)	10
2.2	Exemplo SensorML (CONSORTIUM, 2015)	14
2.3	Possíveis Arquiteturas da WEB das Coisas (KOMAROV; NEMOVA, 2013)	17
2.4	Arquitetura PubSubCoord (AN; GOKHALE, 2014)	19
2.5	Interações da camada de gerenciamento de serviços (NGUYEN, 2014)	21
2.6	Arquitetura Nuvem IoT : Plataforma baseada em Nuvem IoT (ZHOU, 2013) . .	22
3.1	Modelo arquitetural OMG DDS Adaptado	28
3.2	Arquitetura WSN (NIKOLIDAKIS, 2013)	29
3.3	Modelo de dados da aplicação WIREWAY	30
3.4	Modelo da camada de persistência da aplicação WIREWAY	33
3.5	Caso de uso da Autenticação WIREWAY	34
3.6	Diagrama de sequencia para autenticação WIREWAY	35
3.7	Rotas aplicação WIREWAY	37
3.8	Pacote de comunicação do protocolo WIREWAY	40
3.9	Dashboard de Execução da aplicação WIREWAY	41
3.10	Código de controle do Middleware do Dashboard do Abrigo de Cultivo . .	42
3.11	Modelo de componentes do Middleware WIREWAY	44
3.12	Modelo de pacotes do Core do Middleware WIREWAY	46
4.1	Rádio XBee S2 (DIGI, 2015)	49
4.2	Circuito do Coletor na Protoboard.	51
4.3	Raspberry PI 2 MODEL B (UPTON, 2015)	53
A.1	Circuito Conexões Baixo	60
A.2	Circuito Conexões Cima	61
A.3	Placa dos componentes	61

A.4	Pontos de solda e furo	62
B.1	Tela de Login	63
B.2	Tela de Gerenciamento de Dashboards	64
B.3	Tela de escolha de Dashboards	64
B.4	Iniciando Middleware	65

Lista de Tabelas

4.1 Comparação entre desenvolvimento com e sem usar o sistema WIREWAY . 56

Lista de Algoritmos

1	Associação de clientes em salas de controles	38
2	Comunicação entre clientes e middlewares	39
3	Programa simples de controle para abrigos de cultivo	42
4	Programa do coletor	52

Lista de Símbolos

\in - Pertence

\leftarrow - Recebe

$= =$ - Comparação de Igualdade

$< >$ - Comparação de Desigualdade

\geq - Comparação de Maior ou Igual

Σ - Símbolo de somatória

\approx - Comparação por proximidade

Lista de Abreviações

- AES** - Advanced Encryption Standard
- API** - Application Program Interface
- COA** - Cloud-Oriented Architecture
- CPU** - Central Unit Processing
- CSS** - Cascading Style Sheets
- dBm** - decibel-milliwatts
- DDS** - Data Distribution Service
- DSSS** - Direct Sequence Spread Spectrum
- EDA** - Event-Driven Architectures
- FCC** - Federal Communications Commission
- GPIO** - General-purpose input/output
- GPS** - Global Positioning System
- H1** - Header 1
- HD** - Hard Disk
- HDMI** - High-Definition Multimedia Interface
- HTML** - Hyper Text Markup Language
- HTTP** - Hypertext Transfer Protocol
- I/O** - Input/Output
- I2C** - Inter-Integrated Circuit
- IaaS** - Infrastructure as a Service
- IDE** - Integrated Development Environment
- IEEE** - Institute of Electrical and Electronics Engineers
- IGF** - Internet Governance Forum
- IIS** - Internet Information Services
- IoT** - Internet of Things
- ISM** - Institute for Supply Management

ITU - International Telecommunications Union

JS - JavaScript

JSON - JavaScript Object Notation

JVM - Java Virtual Machine

KBps - KBytes Per Second

LAN - Local Area Network

LOD - Linked Open Data

M2M - Machine to Machine

NCAP - Network Capable Application Processor

NFC - Near field communication

OEM - Original Equipment Manufacturer

OGC - Open Geospatial Consortium

O&M - Observations and Measurements

OMG - Object Management Group

OS - Operating System

PaaS - Platform as a Service

PnP - Plug and Play

PSK - Pre-shared key

PubSubCoord - Publishier, Subscribier and Coordinator

PWM - Pulse Width Modulation

QoS - Quality of Service

REST - Representational State Transfer

RFID - Radio Frequency Identification

RISC - Reduced instruction set computing

ROA - Resource-Oriented Architecture

SaaS - Software as a Service

SensorML - Sensor Markup Language

SHA-1 - Secure Hash Algorithm 1

SLA - Service Level Agreement

SOA - Service Oriented Architecture

SOAP - Simple Object Access Protocol

SOS - Sensor Observation Service

SPS - Sensor Planning Service

SQL - Structured Query Language

SSD - Solid State Disk

SSN - Semantic Sensor Network

SWE - Sensor Web Enablement
TED - Transducer Electronic Data Sheet
TKIP - Temporal Key Integrity Protocol
UML - Unified Modeling Language
USB - Universal Serial Bus
URI - Uniform Resource Identifier
XML - Extensible Markup Language
W3C - World Wide Web Consortium
WEP - Wired Equivalent Privacy
WPA - Wi-Fi Protected Access
WSN - Wireless Sensor Network
WTIM - Wireless Transducer Interface Module

Resumo

Cada vez mais sistemas autônomos integrados à Internet estão aparecendo no mercado, as pessoas e as empresas necessitam de sistemas de controle com maior precisão, processamento em tempo real e disponibilidade de acesso ao serviço à qualquer momento e lugar. Esses objetivos estão sendo alcançados graças a Internet das Coisas (IoT), tema muito debatido pelas comunidades de desenvolvedores e pesquisadores. A capacidade de poder controlar, avaliar, compartilhar e programar diversas ações de sistemas de IoT tem deixado o mercado de serviços embarcados muito aberto a inovação, isso vem alavancando o nascimento de muitas empresas pelo mundo com diversas soluções para os mais diferentes cenários. Acredita-se que a popularização da IoT trará consigo um crescimento muito grande em relação a informação produzida pelos mais diversos tipos de sistemas, estima-se que por volta de 2025 a Internet estará presente em todas as coisas de nossa vida, desde um pacote de pão até uma nave espacial. Atualmente muitas questões de performance, segurança e qualidade de informação estão sendo estudadas, questões como: quais as melhores práticas para realizar a integração de dados com diferentes dispositivos em diferentes redes, quais os melhores métodos para realizar o processamento de grandes massas de dados geradas por esses dispositivos em redes homogêneas, quais são as melhores arquiteturas para serem utilizadas por sistemas e dispositivos de IoT. Todas essas questões tem aberto muitas possibilidades de estudo e produtos para as universidades e empresas. Esse trabalho tem por objetivo o desenvolvimento de uma plataforma para integração, criação e controle de sistemas genéricos de IoT, de nome WIREWAY, visando utilizar os padrões aplicados pelas comunidades de desenvolvedores. A plataforma WIREWAY tem por objetivo realizar a comunicação, persistência, controle, segurança e visualização dos sistemas (IoT), através de um protocolo de comunicação baseado no padrão PubSubCoord em Salas de Conversa, utilizando três camadas básicas: servidor de aplicação em nuvem, software de Middleware e um sistema de coleta e atuação.

Palavras-chave: IoT, Java Script, Arduíno, Raspberry, Java, Computação em Nuvem.

Abstract

Increasingly autonomous systems integrated Internet are appearing on the market, people and companies need to control systems with greater precision, real-time processing and availability of access to the service at any time and place. These criteria are being delivered through the Internet of Things (IoT), hotly debated topic by communities of developers and researchers. The ability to monitor, evaluate, share and set various actions of IoT systems have left the market very open embedded services innovation, it has been leveraging the birth of many companies around the world with various solutions for different scenarios. It is believed that the popularization of IoT will bring a very large increase compared to information produced by various types of systems, it is estimated that by 2025 the Internet will be present in all things of our life, from a package of bread to a spaceship. Currently many performance, safety and quality information are being studied, questions like: what are the best practices for the integration of data on different devices on different networks, which are the best methods to accomplish the processing of large amounts of data generated by these devices in homogeneous networks, which are the best architectures for use by IoT systems and devices. All of these issues has opened many possibilities of study and products in universities and companies. This study aims to develop a Platform for integration, creation and control of generic IoT systems. In developing the Platform titled WIREWAY is based on the new standards applied to IoT, basically. The Platform makes communication, persistence, control, security and visualization of devices (IoT) through a communication protocol based on the standard PubSubCoord and Chat Rooms, using three basic layers: cloud application server, middleware software and system collecting and acting.

Keywords: Embedded, IoT, service, control.

Capítulo 1

Introdução

Nos dias atuais, cada vez mais busca-se formas de automatizar atividades rotineiras com intuito de monitorar e controlar informações e comportamentos de diversos ambientes. Com o advento da *Internet das Coisas (Internet of Things - IoT)* esses desejos podem virar realidade. Muitas empresas de tecnologia investem nesse campo oferecendo serviços e produtos que atendem necessidades como: equipamentos de saúde, casas automatizadas, segurança e conforto. A capacidade de poder controlar, avaliar, compartilhar e programar diversas ações de sistemas de IoT tem deixado o mercado de serviços embarcados muito aberto a inovação, isso alavanca o nascimento de muitas empresas pelo mundo com diversas soluções para os mais diferentes problemas.

Além das várias empresas que vem criando seus produtos, e das inúmeras pesquisas feitas por instituições acadêmicas no campo de IoT, um vasto grupo de pessoas, muitas vezes pesquisadores ou entusiastas de tecnologia não atrelados a nenhuma organização também estão adentrado no campo de IoT criando sistemas caseiros. Esse público normalmente tem como objetivos criar suas próprias soluções e saem na Internet em busca de ferramentas para poder publicar suas aplicações de IoT na WEB de forma rápida, simples e segura.

Com o crescimento da IoT, haverá também um crescimento em relação a informação produzida pelos mais diversos tipos de sistemas, a *United States National Intelligence Council* acredita que por volta de 2025 a Internet estará presente em todas as coisas de nossa vida, como por exemplo: comida, apartamentos, documentos, etc. Agora imagine essa massa de equipamentos coletando e compartilhando informações entre si, isso traz muitas dificuldades relacionadas à forma de tornar essas informações úteis (transformar essa informação em sabedoria) (ATZORI; MORABITO, 2010). Muitas organizações como a W3C estão desenvolvendo padrões para que os modelos de comunicação possam ser compartilhadas e usados dentro de vários sistemas diferentes, esses trabalhos tem o

objetivo de organizar a IoT (W3C, 2011). Para as ferramentas que serão produzidas no campo de IoT, faz-se necessário que as mesmas atendam a esses padrões, para oferecer compatibilidade de integração com outros sistemas.

1.1 Problema

Com o avanço dos computadores embarcados e microcontroladores muitas aplicações vem sendo desenvolvidas pelas pessoas, porém é possível fazer pequenas e médias aplicações visíveis e comunicáveis na WEB de forma simples e rápida?

Além disso, com a crescente demanda por sistemas de IoT, é possível realizar a padronização e integração em um único sistema dos dados coletados contendo uma forma de transformar esses dados em conhecimento, devido a grande heterogeneidade que os sistemas podem ter?

1.2 Justificativa

Com o crescimento da IoT, percebe-se que a padronização de dados e unificação em um único sistema é importante devido à grande heterogeneidade das aplicações e da quantidade de sistemas diferentes que não interagem entre si, além disso também existe a necessidade de plataformas que possuam sua arquitetura composta por camadas para que as pessoas possam desenvolver seus sistemas com maior foco na solução tendo mecanismos que gerem sabedoria sobre os dados, ao invés de possuirmos sistemas estruturados sem definições de responsabilidades por camadas.

Vê-se que a complexidade de se publicar sistemas de IoT é alta devido ao conhecimento necessário sobre diversas áreas diferentes como: redes de computadores, desenvolvimento WEB, programação e eletrônica. Isso acaba fazendo com que os desenvolvedores percam o foco da solução ao pesquisar temas desnecessários para suprir uma necessidade de desenvolvimento aplicado, como por exemplo: monitorar algum ambiente com sensores pela WEB.

Facilitar a criação de sistemas e torná-los acessíveis em qualquer lugar através de computadores, smartphones ou tablets, necessário para quem deseja controlar ou monitorar suas aplicações em tempo real, poucos sistemas possibilitam realizar essas atividades de forma clara e objetiva.

1.3 Objetivos

A seguir serão apresentados os principais objetivos deste trabalho, geral e específico.

1.3.1 Objetivo Geral

Desenvolver uma plataforma como serviço de Computação em Nuvem (PaaS - Platform as a Service) de nome WIREWAY, que permita criar sistemas de automação oferecendo maior produtividade e padronização no processo de desenvolvimento e manutenção.

1.3.2 Objetivos Específicos

Para o desenvolvimento deste trabalho faz-se necessário os seguintes objetivos específicos.

- Estudar um modelo para comunicação de dados entre a camada de aplicação e os clientes embarcados, baseados em estudos semânticos.
- Desenvolver uma aplicação de Middleware (Software Gerenciador) para comunicação entre clientes embarcados e servidor.
- Criar um gerador de Dashboards (Painéis Administrativos) para os usuários podem configurar suas ações.
- Publicar a plataforma em rede local para testes.
- Criar um método de configuração de programas para controladores que executem a máquina virtual Java.
- Demonstrar o uso do sistema através de uma aplicação real.

1.4 Procedimentos Metodológicos

Para o desenvolvimento deste trabalho, a aplicação da técnica de pesquisa experimental é aplicada para avaliar o comportamento do software em relação à funcionalidade, flexibilidade e mudanças dos sistemas de IoT. A pesquisa experimental consiste em determinar um objeto de estudo, e selecionar as variáveis que seriam capazes de influenciar o comportamento do sistema, além de definir as formas de controle e fazer as observações

dos efeitos que a variável produz no objeto (GILL, 1991). Os objetos de estudo deste trabalho para a pesquisa de experimentação serão o software de aplicação e o software embarcado, buscará-se avaliar o comportamento da aplicação em relação à Internet oferecida em baixas taxas de transmissão e constante queda de conexão. Avaliando a capacidade de recuperação das aplicações embarcadas com o software do servidor de aplicação. Por fim será avaliado o tempo/complexidade de desenvolvimento de duas aplicações uma utilizando a plataforma WIREWAY proposta e outra não, comparando seus resultados.

Através de pesquisa laboratorial, que em situações controladas será realizada em locais fechados e abertos, possibilitando realizar testes de sensoriamento e comunicação com a plataforma WIREWAY.

1.5 Estrutura do Trabalho

O trabalho se estrutura da seguinte maneira: no Capítulo 2 introduzimos os conceitos de Computação em Nuvem, Internet das Coisas, Web Semântica, Padrões de Comunicação de Dados, Tecnologias Ativas e Modelos de comunicação distribuída. Após isso, o Capítulo 3 apresenta o desenvolvimento do trabalho iniciando pela introdução e forma de desenvolvimento, então a seguir é apresentado o modelo do banco de dados, o modelo da plataforma WEB e do software do Middleware. O Capítulo 4 cria um sistema de monitoramento de um abrigo de cultivo e apresenta os resultados obtidos comparando o desenvolvimento de um sistema dentro da plataforma WIREWAY e um sistema criado totalmente do zero, e por fim as conclusões sobre este trabalho estão no Capítulo 5.

Capítulo 2

Fundamentação Teórica

Através da fundamentação será apresentado os principais tópicos pesquisados para o desenvolvimento deste trabalho, os assuntos serão apresentados serão: o conceito de computação distribuída e em nuvem, o compartilhamento de serviços, o conceito de IoT, as tecnologias ativas usadas para o desenvolvimento de aplicativos de IoT, depois o conceito de semântica em IoT, os padrões de semântica, então será introduzido o novo conceito da WEB 3.0 e por fim as principais arquiteturas de comunicação para IoT.

2.1 Computação Distribuída

Na literatura um sistema distribuído é uma coleção de computadores independentes que fornece um sistema único na visão do usuário. Nesta abordagem temos que ficar atentos a dois aspectos, o primeiro deles refere-se ao hardware que irá suportar o sistema e o segundo refere ao software do sistema, estes dois aspectos em uma estrutura distribuída terão que trabalhar em conjunto entre todos os nós, independente se a estrutura é homogênia ou heterogênia. Assim existem vários tipos de sistemas distribuídos, e um dos melhores exemplos é a própria Internet (TANENBAUM, 2002). Na arquitetura de um sistema distribuído cada nó executa sua própria instância, permitindo de forma transparente ao usuário um sistema único de alto desempenho (DANTAS, 2005).

Nos últimos anos o conceito da computação em nuvem ou em inglês Cloud Computing vem sendo largamente usado, vários produtos na Internet usam este tipo de conceitos como: Amazon Web Services, o Google AppEngine e Microsoft Azure. Quando se fala em computação em nuvem fala-se sobre sistemas distribuídos, onde serviços são fornecidos por demanda como, processamento, armazenamento e outros. Basicamente a computação em nuvem é o uso da Internet compartilhando serviços por demanda. Segundo a visão do Menasce (MENASCE, 2009), a computação em nuvem tem muitos significados dife-

rentes, no entanto, uma definição básica que engloba praticamente todas as definições é a seguinte: a computação em nuvem é uma modalidade de computação distribuída que caracteriza a disponibilidade de recurso por demanda, de uma forma dinâmica e escalável. O termo recurso pode ser usado para representar infraestruturas, plataformas, software, serviços ou armazenamento. Um provedor de nuvem é responsável por disponibilizar os recursos necessários sob demanda para os usuários, e também é responsável por garantir os recursos de forma eficiente para as necessidades dos usuários. Por exemplo, em nuvem a infraestrutura oferece serviços de infraestrutura computacional, normalmente sob uma forma de máquinas virtuais em servidores físicos e assim são cobrados pela quantidade de recursos consumidos.

2.2 Computação em nuvem

Os usuários cada vez mais acessam serviços na Internet em dispositivos menores em comparado aos computadores tradicionais. Então, como o processamento não fica nos Desktops torna-se evidente o uso da computação em nuvem (DIKAIAKOS et al., 2009).

Atualmente os conceitos de computação em nuvem incluem virtualização, software orientado à serviço, computação em malha, flexibilidade nas configurações e eficiência de poder computacional. Normalmente os serviços em nuvem são oferecidos para compra da seguinte forma: IaaS (*infrastructure-as-a-service*), PaaS (*platform-as-a-service*) ou SaaS (*software-as-a-service*) e venda de serviços de valor (DIKAIAKOS et al., 2009).

Distingui-se as formas de serviço de computação em nuvem de duas formas, a primeira é computação por demanda com o modelo de como os recursos são alocados conforme a necessidade por serviços de SaaS e PaaS, e a segunda é designada para prover dados em computação para aplicações por capacidade escalar. Assim, os serviços em nuvens são realizados no formato pago por demanda pelo que vai usar como modelo econômico, e os serviços de hardware para computação em nuvem podem variar (DIKAIAKOS et al., 2009). A seguir será apresentado algumas características desejadas para computação em nuvem.

- **ARQUITETURA DE HARDWARE E SOFTWARE**

A constantemente evolução dos serviços de infraestrutura de hardware, faz com que adicionar memória e capacidade de processamento em servidores não seja o suficiente, mas sim a necessidade de adicionar novos núcleos e acrescer o processamento paralelo em tempo real, então assim, as novas aplicações em hardwares e softwares devem atender à esses requisitos de processamento distribuído (DIKAIAKOS et al.,

2009).

- **GERENCIAMENTO DE DADOS**

É o campo de estudos que aborda a forma de tratamento de dados, ou seja, a forma de como os dados armazenados em serviços em nuvem atravessam a Internet para entregar ao cliente a informação, de maneira segura, armazenando em hosts inseguros informações sem serem replicadas de forma maliciosa para outros servidores, e por fim estuda também as formas de retirar conhecimento das informações para as empresas (DIKAIAKOS et al., 2009).

- **INTEROPERABILIDADE**

É a capacidade dos clientes poderem acessar os serviços independente de plataforma de hardware ou sistema através da web, garantindo segurança, velocidade, qualidade e portabilidade das aplicações e dos dados (DIKAIAKOS et al., 2009).

- **SEGURANÇA E PRIVACIDADE**

Os serviços em nuvem, devem garantir a segurança dos dados de cada usuário, porque, como essas informações são acessadas de ambientes externos, isto possibilita que os dados possam ser armazenados de forma insegura, aberta, e infringindo os direitos de cópia do conteúdo (DIKAIAKOS et al., 2009).

- **PROVISIONAMENTO DE SERVIÇOS**

A computação em nuvem oferece seus serviços através de contratos de níveis de serviço SLA (*Service-level agreement*), que basicamente possuem as definições de quantidade de processamento que o usuário terá disponível na CPU (*Central Processing Unit*), memória, HD e outros recursos necessário, assim a incapacidade de oferecer esses recursos de hardware pelo fornecedor pode levar à quebra do contrato SLA sujeito às penalidades, e o oferecimento de recursos a mais, resultando em subestimação do uso do hardware, por isso essas definições são importantes e tornam-se um desafio para os provedores de serviços (DIKAIAKOS et al., 2009).

2.2.1 Compartilhamento de serviços

A Internet feita pelo compartilhamento de serviços traz junto consigo pesquisas, vendedores de tecnologias, provedores de serviço e usuários finais em direção a criar uma revolução de serviços - a Internet dos serviços. As comunidades de serviços na Internet procuram desenvolver especificações técnicas de hardware e software, fazendo com que o

desenvolvimento de serviços traga consigo arquitetos, desenvolvedores e integradores de serviços. Além disso, os usuários passam a ser um tipo especial, porque eles podem se tornar um composto de serviço em nuvem além de simples consumidores (KOMAROV; NEMOVA, 2013).

Bibliotecas de serviços disponíveis serão a composição de todos os serviços que podem ser consumidos e oferecidos por uma empresa de serviços, assim em novos desenvolvimentos, os arquitetos e integradores de serviços podem definir interfaces de serviços, avaliando se já não existem serviços prontos antes de desenvolver um novo (KOMAROV; NEMOVA, 2013).

2.3 O que é Internet das Coisas

A Internet das Coisas (IoT) traz uma ideia básica de presença de várias coisas à nossa volta trocando informações e colaborando em objetivos comuns. O ganho com a IoT será nos campos de domótica, vida assistida e e-health. E para os campos industriais os principais ganhos serão com automação, manufatura, logística, gerenciamento de processos e negócios, transporte inteligente, etc. (ATZORI; MORABITO, 2010).

O principal objetivo em se conectar diversos dispositivos em uma rede de IoT, é poder captar a realidade do mundo externo e converter para o mundo digital, assim possibilitando ao sistema compreender o ambiente e tomar decisões sobre ele (BARNAGHI, 2012).

Basicamente a interconexão das coisas é feita usando materiais dielétricos tais como: cabos, fibra-ótica ou ondas eletromagnéticas, materiais que servem para conectar as coisas ao redor do mundo. Mas os objetos que realizam transmissão via sinais elétricos ou ópticos devem ser entendidos somente como entidades ou "coisas". Então, o real significado de IoT é sobre a informação gerada por "coisas" (HUANG; LI, 2010).

Muitas empresas estão pesquisando no campo de IoT para entender sobre a tecnologia e fornecer conhecimento para a comunidade de desenvolvedores, por exemplo, a Cisco, a Microsoft, a Google, e outras estão fazendo pesquisas sobre as visões da IoT, levantando os maiores benefícios na vida cotidiana das pessoas, com o objetivo de fornecer aos leitores a oportunidade sobre o que pode ser feito, como: algoritmos e protocolos, mostrando fatores desse processo evolucionário e quais são os pontos principais de fraqueza e risco (ATZORI; MORABITO, 2010).

Em 2005, um relatório da International Telecommunications Union (ITU) sugeriu que a "Internet das Coisas" irá conectar o mundo com objetos, com sensores de uma maneira inteligente. Próximo de combinar vários desenvolvimentos tecnológicos, o ITU

tem descrito quatro dimensões em IoT: *item identification* ("coisas tagueadas por RFID por exemplo"), *redes wireless de sensores sensíveis ao ambiente* ("perceber as coisas"), *sistemas embarcados* ("coisas que pensam") e *nano tecnologia* ("encolher as coisas ainda mais") (UNION, 2005).

O termo genérico de IoT, vem sendo visto por organizações também como "Internet Orientada" ou "Coisas Orientadas" devido ao foco que as empresas dão para esse tipo de tecnologia. E não deve-se esquecer nunca que as duas palavras: "Internet" e "Coisas" trabalham junto em um alto nível de inovação. Semanticamente falando "Internet das Coisas" significa: muitos dispositivos conectados à WEB, se comunicando através de protocolos nos mais diversos sistemas heterogêneos (ATZORI; MORABITO, 2010).

Basicamente, o conceito de coisas vem com o objetivo de entender a transformação de ações do mundo real para um entendimento digital. Além disso, entende-se também que os humanos poderão receber e enviar informações para esse mundo digital através das coisas, e essas coisas nada mais são do que pequenos dispositivos no mundo que terão uma identificação única. O consórcio CASAGRAS acredita que IoT significa: coisas serão comunicadas através de sistemas de alto nível que servirão ao benefício da humanidade (ATZORI; MORABITO, 2010).

2.3.1 Tecnologias Ativas

A qualquer momento, lugar e formato é uma frase que ao longo do tempo emprega uma ideia de como as "coisas" serão estudadas dentre as comunidades de pesquisa. Nesse contexto, as tecnologias wireless ficam em evidência, e muitos requisitos como quantidade de dados a ser transmitidos, consumo de energia e tamanho dos dispositivos são avaliados para desenvolver os novos comunicadores (ATZORI; MORABITO, 2010).

Dentro desses desenvolvimentos, muitos softwares são desenvolvidos utilizando Middlewares para gerenciar todos os que se conectam a ele removendo a complexidade de integrar sistemas diferentes e dar suporte a sistemas legados, funcionando como serviços SOA (Service Oriented Architecture). Basicamente, os componentes de uma arquitetura que se destacam são: aplicativos, composição de serviços, gerenciamento de serviços e abstração do objeto.

O modelo de camada para sistemas IoT é demonstrado na Figura 2.1.

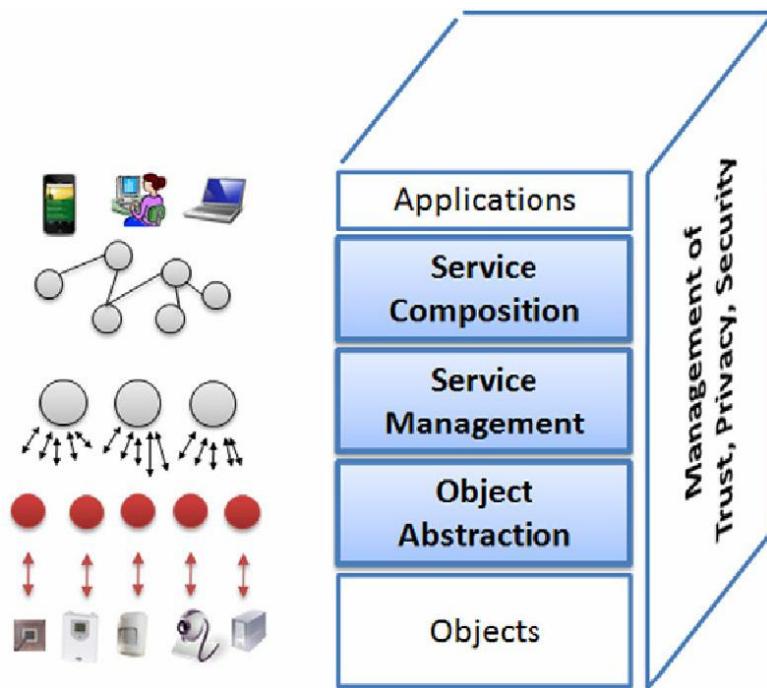


Figura 2.1: Arquitetura SOA para Middleware IoT (ATZORI; MORABITO, 2010).

2.3.1.1 Aplicações

A camada de aplicação é a de mais alto nível, normalmente é entregue aos usuários. As aplicações possuem as funcionalidades do sistema, entendendo que essas mesmas se integram com as funcionalidades de todo o consumo de dados entregue para camada de Middleware (ATZORI; MORABITO, 2010).

IoT tem potencial para impactar a sociedade, o meio ambiente e a economia. Pequenas medições feitas em localidades podem gerar conhecimento para realizar ações que mudem o ambiente, isso já tem sido feito em várias áreas da indústria e certamente tende a crescer ainda mais (COETZEE et al., 2011). Segundo An, existem duas categorias de alto nível para as aplicações de IoT, são elas: Informação e Análise e também Automação e Controle (AN; GOKHALE, 2014).

Em Informação e Análise, os serviços necessitam de recebimento de informações de qualidade, isso faz com que a análise dos dados tenha maior precisão. Essa categoria se aplica para *Rastreamento* (produtos em um cadeia de logística) e *Consciência situacional* (Sensores em infraestruturas ou condições ambientais) que através de eventos em tempo real, podem fornecer um maior feedback das análises dos sensores (COETZEE et al., 2011).

Automação e Controle implica em reagir ao ambiente contra os dados recebidos e processados dos sensores, assim áreas que poderão se beneficiar com isso são: processos

(por exemplo em indústrias químicas aonde a análise e o reajuste de composições de forma autônoma entregam benefícios de segurança, velocidade e custo) e áreas de consumo de recursos, por exemplo, em casas que o sistema pode gerenciar os itens que estão ou não estão sendo usados e desativar ou balancear a carga para áreas de maior necessidade (COETZEE et al., 2011).

2.3.1.2 Composição de serviços

Composição de serviços é a camada no topo da arquitetura SOA-Middleware. Ela provê as funcionalidades para compor vários serviços, oferecendo uma rede de objetos para construir aplicações específicas. Nessa camada, não se conhece quais dispositivos são visíveis para o sistema e sim como os diversos serviços se comunicam definindo *Workflows* (fluxo de trabalho) de processos (ATZORI; MORABITO, 2010).

2.3.1.3 Gerenciamento de serviços

Gerenciamento de serviços é a camada que provê o gerenciamento das funções principais que são esperadas por cada objeto que pode ser gerenciado no contexto de IoT. Um básico conjunto de serviços pode ser encontrado em: descobrimento de objetos dinâmicos, monitoramento de status e configuração de serviços. Basicamente, entra em contexto vários sub conjuntos de gerenciamento de qualidade do serviço (ATZORI; MORABITO, 2010).

2.3.1.4 Abstração de objeto

Define vários objetos podendo se comunicar na Internet, para isso é necessário uma linguagem comum para que todos tenham a capacidade de se compreender harmonicamente. Normalmente definem-se duas subcamadas para se trabalhar nessa estrutura, uma camada de interface e uma de rede (ATZORI; MORABITO, 2010).

Entendido o modelo de como a comunidade de desenvolvedores vem estruturando os sistemas de IoT, na próxima sub-sessão será estabelecido como a semântica é importante para a IoT.

2.3.2 Semântica

O significado de semântica é sobre o que as palavras significam, ou na Internet, como as informações são catalogadas e identificadas. Por exemplo, em HTML a tag H1

identifica o título de um texto, facilitando o deciframento daquela informação e indicando como processá-la. Segundo Barnaghi, a semântica é uma área de estudo em IoT, para que os serviços possam ser desenvolvidos de forma correta, fazendo com que os dados tenham um destino correto. A semântica é ainda mais útil quando se pensa em vários sistemas heterogêneos se comunicando freneticamente. Dentre os estudos de semântica podemos citar os seguintes:

- Semântica para interoperabilidade
- Integração de dados IoT
- Abstração de dados IoT
- Pesquisa e descobrimento de recursos e serviços de IoT
- Raciocínio semântico e interpretação

Precisa-se entender também que a semântica não irá sozinha resolver todos os problemas de descobrimento, gerenciamento de dados e suporte à interações autônomas. A semântica será processada de várias formas em diferentes domínios (BARNAGHI, 2012). Por isso deve-se levar em consideração os seguintes itens:

- Ontologia e semântica só fazem sentido quando a mesma é compartilhada entre diversos sistemas, caso contrário, ela não garante interoperabilidade.
- As anotações semânticas precisam ser processadas e analisadas, para darem sentido aos dados.
- Semânticas não é uma campanha publicitária que ao longo dos anos, muitas tecnologias foram lançadas para trabalharmos com filtros e compartilhamento.

Para ajudar a resolver problemas de interoperabilidade entre sistemas de IoT, causados pela natureza da heterogeneidade das "Coisas (Things)", a comunidade já vem desenvolvendo algumas semânticas para descrever os objetos do mundo real e seus eventos. Dentre os diversos trabalhos de grupos de pesquisa já realizados podemos citar os que tiveram maior relevância (BARNAGHI, 2012).

- O grupo OGC (Open Geospatial Consortium) fez um trabalho de Sensor Web Enablement (SWE), com a definição de "Observation & Measurements O&M", que é baseado em um padrão XML para codificar aplicações de tempo real em suas observações e medidas dos dados dos sensores, entre algumas especificações como SensorML, SOS, SPS e PUCK Protocol que serão detalhadas a seguir.

- W3C Semantic Sensor Networks Incubator Group tem desenvolvido a ontologia SSN (Semantic Sensor Network), que provê um alto nível de abstração sobre os objetivos.

As anotações de semântica nos objetos em IoT descrevem o que eles fazem, porém precisa-se conseguir conectar todos esses dados para gerar conhecimento em relação a essa rede de conexões, para que seja possível fazer a conexão desses dados é necessário que sigamos alguns padrões em URI's (Uniform Resource Identifier), visibilidade, descrição e conexões, como o atual Linked Open Data (LOD). Trabalhando sobre essas definições, é possível gerar sistemas que sejam capazes de criar consultas e verificar as condições do mundo real através dos sensores, como por exemplo, ao realizar a query: "me diga o trânsito na Waldemar Gruba, Jaraguá do Sul, SC, Brasil", através da disponibilidade das conexões o sistema pode retornar todas as medições dos dados que são retirados naquela localidade, e o padrão de cadastro de novos sensores deve seguir a mesma ontologia usada pelos outros, para que ele possa fazer parte dessa rede de conexões (BARNAGHI, 2012).

2.3.3 OGC Standards

As iniciativas do grupo de padronização de sensores WEB da OGC são aplicadas em todo o mundo, possibilitando que desenvolvedores possam fazer todos os tipos de sensores, transdutores e repositórios de dados de sensores, acessíveis e utilizáveis via Internet ou outros tipos de rede. Alguns exemplos de aplicações para os padrões da OGC tem sido: monitoramento de tráfego, satélites de imagem, monitores de ambiente, monitoramento de saúde, processo de monitoramento industrial e outros (CONSORTIUM, 2014).

SensorML (*Sensor Markup Language*), foi o primeiro padrão feito pela OGC inicialmente, e depois o grupo trabalhou com a IEEE para harmonizar o padrão de linguagem de marcação com o padrão IEEE 1452 "Smart Sensor" e com o consórcio Marine Plug-and-Work para harmonizar com o padrão PUCK (CONSORTIUM, 2014).

Atualmente, sensores interconectados estão se proliferando em taxas fenomenais, pequenos telefones incluem giroscópios, acelerômetros, GPS, Wi-Fi, Bluetooth, som, luz, força magnética, timers, NFC, bússulas e câmeras. Futuramente termômetros e gravímetros, assim, para que as aplicações feitas para esses equipamentos possuam um grande efeito na rede, elas dependem de seus padrões abertos (CONSORTIUM, 2014).

Os principais padrões para redes de sensores da OGC são:

- Sensor Model Language (SensorML)
- Observation & Measurements (O&M)

- Sensor Observation Service (SOS)
- Sensor Planning Service (SPS)
- SWE Common Data and Services

OGC SensorML é um padrão desenvolvido pela OGC com foco em prover um padrão semântico entre processos e processamento de componentes associados a medição e pós-medição de observações, este inclui sensores e atuadores. Possibilitando que complexos fluxos de trabalhos entre sensores possam ser compartilhados entre agrupamentos de sensores diferentes sobre a Internet. Atualmente a especificação está na versão 2.0 (CONSORTIUM, 2014).

Basicamente, esse padrão permite que sensores sejam identificados em redes distintas, usando diversas notações XML, os dados podem ser repassados para processamento. O padrão utiliza UML para identificar o que cada componente deve utilizar sobre suas medições e objetivos.

A Figura 2.2 demonstra o formato de dados simplificado da transmissão de um pacote com a localização do componente.

```

<sml:position>
  <swe:Vector definition="http://sensorml.com/ont/swe/property/SensorLocation"
    referenceFrame="http://www.opengis.net/def/crs/EPSG/6.7/4326">
    <swe:coordinate name="Lat">
      <swe:Quantity definition="http://sensorml.com/ont/swe/property/Latitude" axisID="Lat">
        <swe: uom code="deg"/>
        <swe:value>47.8</swe:value>
      </swe:Quantity>
    </swe:coordinate>
    <swe:coordinate name="Lon">
      <swe:Quantity definition="http://sensorml.com/ont/swe/property/Longitude" axisID="Long">
        <swe: uom code="deg"/>
        <swe:value>2.3</swe:value>
      </swe:Quantity>
    </swe:coordinate>
  </swe:Vector>
</sml:position>

```

Figura 2.2: Exemplo SensorML (CONSORTIUM, 2015)

A linguagem de marcação SensorML define modelos para trabalhar com a descrição dos sensores, entradas e saídas, histórico dos sensores, posicionamento, atividades, trajetória, definição física, e assim por diante.

2.3.4 ZigBee

ZigBee é um dos padrões para redes de sensores sem fio que vem se difundindo largamente. Foi desenvolvido para ser um padrão global para entregar baixo-custo, baixo-consumo de energia em comunicações M2M (Machine to Machine). O padrão ZigBee opera sobre a especificação física IEEE 802.15.4, baseado em pacotes de dados de baixo custo. Para dispositivos com baterias o protocolo permite vários dispositivos se comunicarem em uma variedade de topologias de rede e manter-se funcionando durante anos (DIGI, 2015).

O protocolo ZigBee foi criado por membros da companhia ZigBee Alliance. Basicamente, mais de 300 produtores de semicondutores, firmas tecnológicas, OEMs e companhias de serviço acolhem o padrão ZigBee que foi projetado para providenciar um uso fácil em redes sem fio, de forma segura e confiável. O protocolo ZigBee é desenvolvido para transmitir dados RF através de ambientes hostis, que é comum em aplicações industriais e comerciais (DIGI, 2015). Algumas características das redes ZigBee são:

- Suporte a múltiplas topologias de rede, como ponto a ponto, estrela e redes mesh.
- Baixo ciclo de trabalho - provendo uma longa vida da bateria.
- Baixa latência.
- Espectro de propagação de sequencia direta (DSSS)
- Mais de 65,000 nós por rede.
- Criptografia de dados AES de 128-bits.
- Anulação de colisão, re-tentativa e conhecimento (ACK).

Uma componente chave do ZigBee é a capacidade de criar redes mesh se comunicando com um coordenador. Nesse tipo de rede, os nós roteadores e dispositivos finais se comunicam através de caminhos que são armazenados nas tabelas de roteamento dos nós ZigBee, fazendo-os recuperáveis há quedas. Cada nó em uma rede mesh é capaz de auto descobrir a rede e entrar ou sair da mesma (DIGI, 2015).

Além disso, projetos utilizando ZigBee para camada de comunicação com aplicações de Middleware em que nós de rádio são configurados como *EndDevices* ou Roteadores e possuem capacidades auto descriptivas para ingressar em redes de sensores sob a posse de um *gateway* que redireciona as informações para um servidor central onde estão sendo criados. Muitos padrões foram desenvolvidos para simplificar a forma de auto descrição

dos sensores, entre muitos projetos um que tem bastante destaque é um *Framework* iPA-GAT para redes WSN do padrão IEEE 1451 junto com o padrão ZigBee (FERNANDES SAMUEL G MATOS, 2013).

2.3.5 WEB 3.0

A WEB 3.0 demonstra a nova era da internet feita pela IoT, ela é um conceito que tem diferentes significados, que não se limita a usar somente informações das máquinas como na WEB 1.0, versão que foi capaz de prover informações estáticas em um modelo *Broadcast* para todos e como a atual a WEB 2.0, que possibilitou usuários poderem ser também produtores de informação. WEB 3.0 significa que as coisas (objetos), poderão ter o poder de contribuir, aprender e decidir (KOMAROV; NEMOVA, 2013). A seguir algumas definições sobre a WEB 3.0 segundo alguns autores:

- Segundo Marcelo Dias de Amorim, que no projeto FP6 incluiu a descrição para WEB 3.0 como da Internet das Coisas, em que a comunicação também é feita de máquina para máquina (M2M) (CORDIS, 2014).
- Segundo Tim Berners-Lee (diretor da W3C) determina a WEB 3.0 como Internet que você poderá ler, escrever e executar - em termos de execução ele explica a semântica da WEB e dos serviços WEB (BERNERS-LEE, 2009).
- Segundo Srmana Mitra, o conceito está entre tecnologia e filosofia e para ela WEB 3.0 é resultado da combinação de conteúdo, comércio, comunidade e contextos, com personalizações e pesquisas verticalizadas. Para Srmana contexto é referente ao motivo pelo que você navega na WEB (MITRA, 2009).
- E o conceito mais filosófico para WEB 3.0 é de Ted Hoy (vice-presidente da Digital River Company), como parte da computação em nuvem, aplicações de usuários estão se tornando ricas dentro da WEB (HOY, 2007).

Na WEB 3.0, a usabilidade para os usuários será diferente, não se limitará somente a usuários utilizando Desktops únicos, mas sim pulando de coisas em coisas pela WEB. Grande parte dos serviços ficarão hospedados em nuvem, ilimitando a necessidade de dispositivos únicos para as pessoas (KOMAROV; NEMOVA, 2013).

Arquitetura Orientada a Nuvem (*COA - Cloud-Oriented Architecture*) é um modelo conceitual que segundo Bloomberg, agregará muitos objetos em ambientes de nuvem, o desenvolvimento de uma arquitetura de nuvem global será essencial para a IoT, em que qualquer dispositivo poderá ser identificado. A COA está relacionada a arquitetura

service-oriented architecture SOA e também com event-driven architectures (EDA), com a combinação dessas duas arquiteturas, SOA e EDA defini-se uma arquitetura nova conhecida por resource-oriented architecture (ROA). Em ROA, qualquer coisa pode ser um recurso, desde sensores, celulares, computadores, servidores, etc (KOMAROV; NEMOVA, 2013).

Como apresentado até agora, a WEB 3.0 deverá se moldar entre a composição de diversas arquiteturas, formando no final um modelo genérico baseado em serviços em nuvem e arquiteturas de recursos, a Figura 2.3 a seguir demonstra a divisão e a comunicação entre as camadas de usuário, serviço, WEB e coisas.

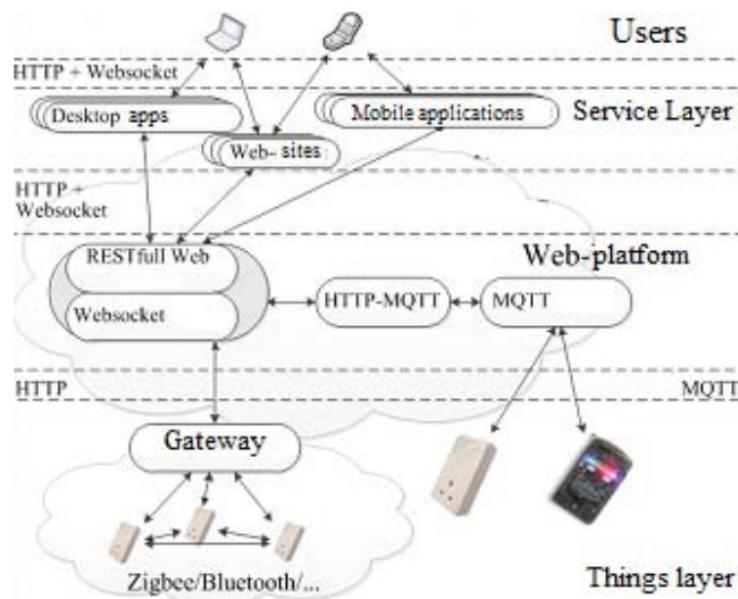


Figura 2.3: Possíveis Arquiteturas da WEB das Coisas (KOMAROV; NEMOVA, 2013)

2.3.6 Social WEB das Coisas

As novas tendências de IoT possibilitará uma nova era em que as *coisas* serão conectadas e centralizadas em uma WEB social, para que possam fazer carga de dados, consumir dados e trocar informações. Dessa forma, os serviços poderão ser desenvolvidos em uma plataforma aberta para dispositivos de baixo nível entregando produtos de alto nível, assim mudando a maneira que os usuários usam os dados. Para que a segurança na troca de dados entre serviços como seja confiável, os sistemas deverão seguir regulamentos e políticas da *United Nations* para proteção de dados pessoais e direitos humanos. Este, que é um *Internet Governance Forum (IGF)* definido pela *United Nations* foca em

políticas da Internet e algumas regras sobre proteção de dados (KOMAROV; NEMOVA, 2013).

2.4 Arquiteturas de Plataformas

Para o desenvolvimento de projetos voltados à IoT, muitos tipos de arquiteturas podem ser aplicadas. Algumas arquiteturas tem sido mais bem aceitas por atender a maior parte dos requisitos. A seguir será apresentado alguns modelos de arquiteturas estudadas para o desenvolvimento deste trabalho.

2.4.1 Modelo de Distribuição de Dados

Aplicações que compartilham informações necessitam de um modelo conceitual de comunicação. O OMG (Object Management Group) Data Distribution Service (DDS) é uma especificação padrão que define uma forma para que objetos possam compartilhar dados e assuntos comuns através de conexões dinâmicas, permitindo a centralização dos dados e o gerenciamento por publicadores e consumidores de conteúdo. Esse modelo conceitual traz muitas vantagens pelo seu baixo acoplamento com as tecnologias e também seu suporte a muitos aspectos de QoS (Quality of Service) como confiança e velocidade na transmissão de dados em ambientes dinâmicos. Porém, o modelo OMG DDS não define uma forma de como os objetos são descobertos ou ingressados em uma rede (AN; GOKHALE, 2014).

Estudos feitos na área de OMG DDS, com arquiteturas que possibilitam que representantes possam se comunicar em redes locais sobre um assunto de interesse, e também compartilhar assuntos com representantes de outras redes, nesse contexto os assuntos entre os representantes são áreas de interesse. Um exemplo de arquitetura de OMG DDS é a arquitetura PubSubCoord (Publisher / Subscribier / Coordinator), que define um modelo baseado em representantes de borda e representantes de roteamento. Na arquitetura PubSubCoord, os representantes de borda são responsáveis por se conectar diretamente aos representantes de roteamento e servir como ponte entre diferentes LANs (Local Area Network - Rede Local) transmitindo os dados de seus agentes. Os representantes de roteamento são responsáveis por compartilhar os assuntos e inscrever novos representantes no mesmo assunto. A Figura 2.4 demonstra o modelo arquitetural (AN; GOKHALE, 2014).

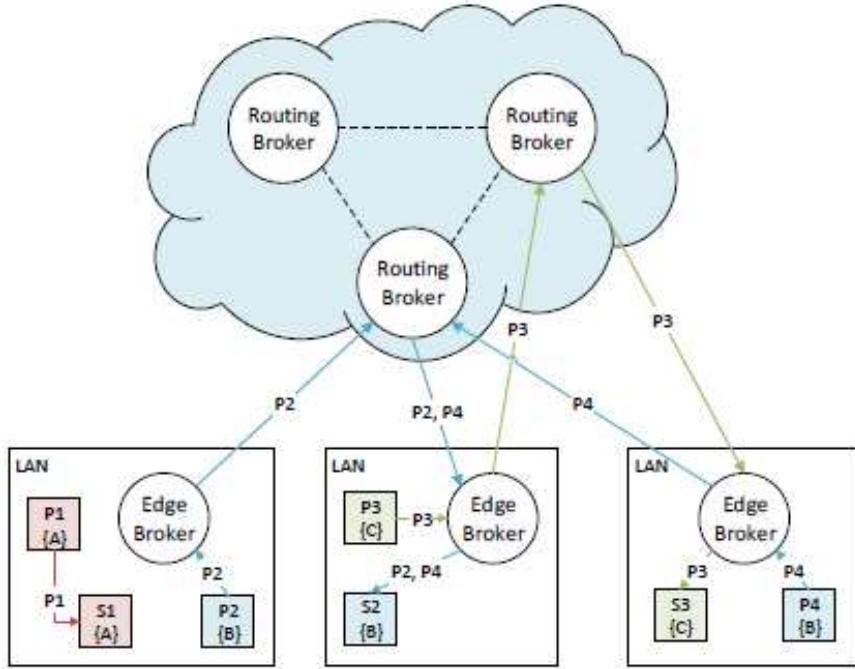


Figura 2.4: Arquitetura PubSubCoord (AN; GOKHALE, 2014)

Na Figura 2.4 percebe-se como funciona o modelo PubSubCoord, nesta $Px\{y\}$ é um publicador de conteúdo identificado pela letra x , e interessado no tópico y . Como podemos ver no tópico A temos P1 e S1 interessados no mesmo tópico dentro da rede LAN se comunicando via *multicast* por UDP. P2, P4 e S2 estão interessados no tópico B e estão localizados em redes diferentes, então os representantes de borda servem como pontes para que os representantes de roteamento possam encaminhar os pacotes entre as redes. De forma bem abstrata, o algoritmo responsável por gerenciar a rede de assuntos segue a ideia do padrão de projetos *observer* em que o assunto é usado como agrupador para os representantes (AN; GOKHALE, 2014).

2.4.2 Modelo de Gerenciamento de Serviços

Aplicações desenvolvidas para gerenciar serviços de IoT, normalmente são divididas em camadas. Estudos de desenvolvimento de software desse tipo tem sido feitos sobre arquiteturas SOA. Como, por exemplo, o Hecate um modelo de arquitetura baseado em cinco camadas utilizado para publicação de serviços. As camadas do Hecate são (NGUYEN, 2014):

- Aplicação que é responsável pelo registro e descobrimento de serviços, gerenciamento de mensagens e gerenciamento de serviços
- controle de acesso, pode ser público ou privado

- gerenciamento de serviços com a criação de serviços, mapeamento de serviços, monitoramento de serviços, gerenciamento de QoS (Quality of Service) e gerenciamento de balanceamento de carga
- Virtualização de dispositivos com o descobrimento de dispositivos e gerenciador de virtualização
- Dispositivos com Cloud, RFID, Sensores, Smart Phones e Computadores.

Essa arquitetura permite a aplicação contribuir para a rede de serviços com serviços públicos ou privados de armazenamento (storage), publicadores/consumidores de dados (messaging) ou processamento computacional distribuído (computing) (NGUYEN, 2014). Embora essa arquitetura seja interessante para uma rede totalmente aberta de serviços de IoT, busca-se especializar o gerenciamento de serviços para este trabalho. Mas antes, necessita-se um breve detalhamento de como funciona a camada de Gerenciamento de Serviços utilizados pelo Hecate.

Serviços precisam ser gerenciados e monitorados para manter a aplicação saudável e eficiente. Depois que um serviço é criado e publicado por um dispositivo no Hecate, são passadas várias fases para determinar seu escopo e parametrizar suas configurações de nome, identificação, tipo e descrição para propósitos de busca. Dependendo do tipo de serviço, eles podem possuir três tipos diferentes de QoS. A Figura 2.5 demonstra a iteração entre os serviços da camada de Gerenciamento de Serviços. Por exemplo, um serviço é mapeado para um objeto virtualizado pelo Service Mapper (Mapeador de Serviço), o QoS Manager (Gerenciador de Qualidade) irá receber as informações das condições do serviço e sua qualidade através do Service Monitor (Monitor de Serviço), antes de passar pelo balanceamento de carga para determinar como o serviço deverá ser tratado na virtualização (NGUYEN, 2014).

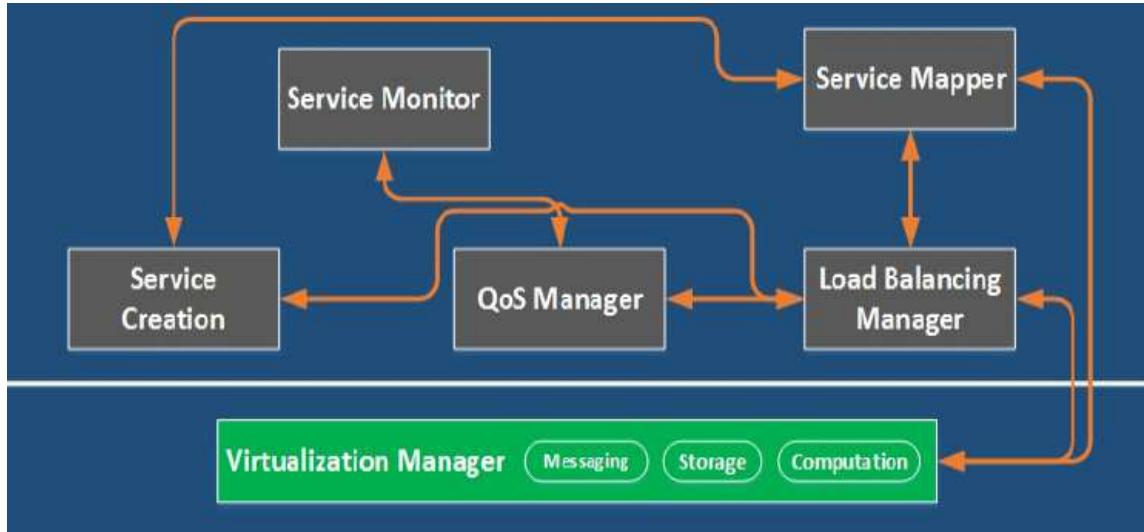


Figura 2.5: Interações da camada de gerenciamento de serviços (NGUYEN, 2014)

A arquitetura utilizada para realização desse trabalho é similar à utilizada no Gerenciamento de Serviços do Hecate quando um cliente se conecta à aplicação, alguns parâmetros são definidos, como: a quantidade de dados que será transmitida, os tipos de dados e a prioridade de tempo real da conta solicitante, são avaliados pelo serviços de QoS assim, a parametrização para determinar qual servidor deverá ser encaminhada as requisições para processamento. Após isso, a nova requisição é re-encaminhada para o solicitante passando as credenciais que podem ser usadas para acessar o serviço aconselhado.

2.4.3 Modelo do Serviço Nuvem

O que faz um serviço nas nuvens para IoT diferente da simples e convencional Internet das Coisas, é a habilidade de desenvolver, publicar, rodar e gerenciar as "Coisas" da aplicação Online. A Figura 2.6 demonstra os principais recursos de uma plataforma baseada na nuvem e suas interações com o modelo de computação em Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). A Figura 2.6 especifica tecnicamente uma solução de rede, iteração e integração com a nuvem (ZHOU, 2013).

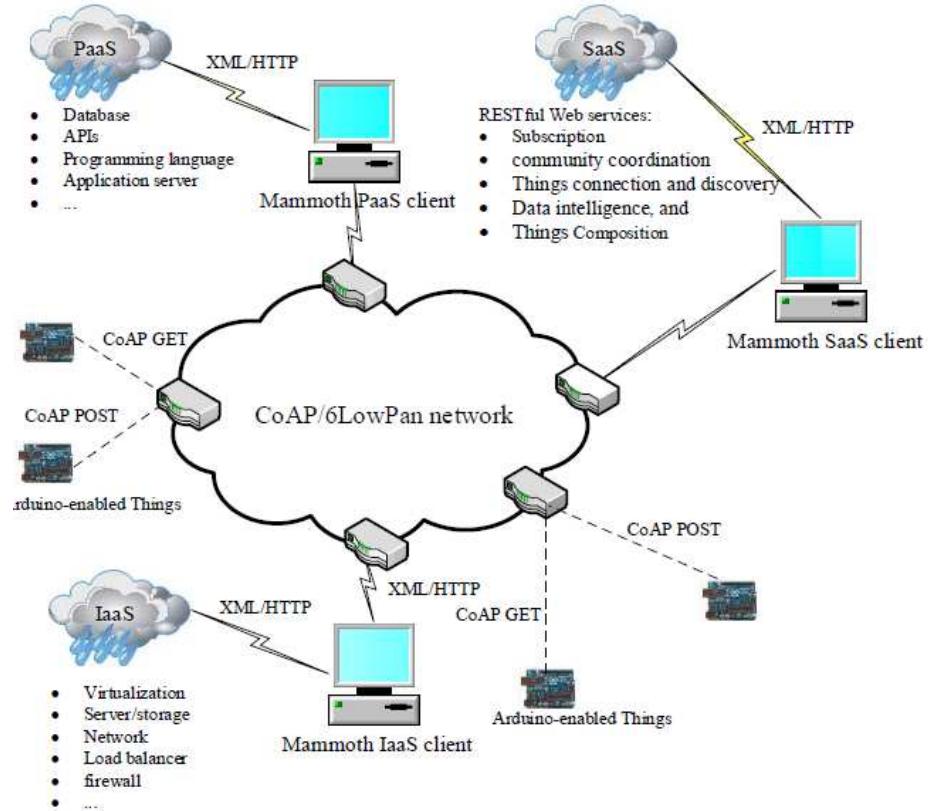


Figura 2.6: Arquitetura Nuvem IoT : Plataforma baseada em Nuvem IoT (ZHOU, 2013).

Um serviço nas nuvens é uma plataforma online que permite sistemas se integrarem e proverem soluções de problemas para alavancar uma estrutura completa de desenvolvimento, publicação, operação e composição de serviços para que uma arquitetura dessas seja atendida ela deve ser composta dos seguintes três módulos (ZHOU, 2013):

- IaaS (Infrastructure as a Service) permite usuários rodar qualquer aplicação no hardware em nuvem de qualquer lugar. Simplificando o desenvolvimento deixando o usuário livre sem se preocupar com configuração de servidores, atualização de sistemas, entre outros, reduzindo o tempo em que uma aplicação fica funcional. O mesmo provê armazenamento para a coleta dos dados das "Coisas" (ZHOU, 2013).
- PaaS (Platform as a Service) disponibiliza ao usuário um ambiente para desenvolver as aplicações e disponibiliza-las (ZHOU, 2013).
- SaaS (Software as a Service) são os portais de operação do software, onde as aplicações públicas pelos usuários vão de fato ser utilizadas (ZHOU, 2013).

A plataforma deste trabalho será disponibilizada em nuvem como PaaS para que o usuário possa escolher as configurações do ambiente, como: tamanho do banco de

dados para armazenar os dados de sensores sob valores contratuais, uma IDE (Integrated Development Environment) para programação WEB, onde poderão ser configuradas todas as funções que os clientes executarão quando conectados a plataforma e serviço para visualização dos Dashboards de aplicação onde os sistemas de fato serão controlados e executados por clientes.

2.5 Conclusões

Pode-se perceber que a Internet vem há algum tempo já tomando um novo caminho, assim os novos serviços em nuvem para Internet das Coisas estão cada vez mais integrados e inteligentes, conforme visto na fundamentação muitos conceitos de arquiteturas orientadas a serviços já foram criados tal qual modelos de comunicação distribuídos. Dessa forma, o desenvolvimento de aplicações para IoT que visem facilitar a criação e publicação de novos serviços é de grande ajuda para contribuir com as comunidades de pesquisa. O desenvolvimento deste trabalho se baseia nos padrões de computação em nuvem, utilizando uma arquitetura PaaS para disponibilizar uma plataforma que permite integrar e criar novos sistemas para Internet das Coisas. O próximo capítulo descreve a forma de desenvolvimento desta plataforma utilizando-se dos conceitos até então apresentados como, modelo de comunicação, modelo de escalonamento por usuários por tipo de serviço, modelo de integração de dados, arquitetura separada entre: cliente, aplicação, middleware e dispositivos.

Capítulo 3

Desenvolvimento

A Plataforma apresentada neste trabalho intitulada WIREWAY, tem como foco entregar ao cliente uma aplicação para o desenvolvimento de sistemas de IoT em nuvem, possibilitando criar telas de monitoramento e atuação em componentes como sensores e atuadores, além de configurar regras de interação entre os componentes. As telas da aplicação são Dashboards (Painéis Administrativos), onde todos os componentes de um sistema são visíveis pelo usuário e podem ser manipulados. Por exemplo: em um sistema de controle de trancas de um criadouro composto por vários rebanhos de animais, deseja-se controlar o acesso dos animais aos campos. Assim um Dashboard administrativo pode conter uma tela com vários botões catalogados pelo nome de cada porteira de acesso entre os campos, e o acionamento do botão interagir diretamente com o Middleware que interage com o atuador para executar a ação de travamento ou destravamento das porteiros, reduzindo os custos e o tempo de alguém ter de executar essa tarefa manualmente. Além disso, a grande vantagem do sistema WIREWAY é a possibilidade de configurar a forma como o Dashboard vai agir em relação a cada botão. Assim em vez de se criar um botão para cada porteira que da acesso a um pasto, pode-se criar um botão para cada tipo de pasto por vegetação ou ambiente que o rebanho deveria ter acesso naquele dia. Pode-se então transformar os botões em ações de negócio, assim em vez de destrancar as porteira 1,2,5 e 8, uma de cada vez para dar acesso para criação ao pasto de vegetação mais densa para engorda, crie-se um único botão que abre automaticamente esse caminho por completo, por exemplo.

3.1 Importância do Sistema

A importância do sistema WIREWAY se dá ao avanço na área de automação de sistemas que se divide em duas partes: servidor WEB que é responsável pelas confi-

gurações das aplicações de IoT, por fazer a interface para comunicação com os programas embarcados e a interface com o usuário no navegador, armazenando no banco de dados as informações coletadas pelo sistema embarcado (Middleware) e atuando sobre o mesmo. Já a segunda parte do sistema é o Middleware, uma aplicação baixada pelo cliente por meio do site. Esta aplicação irá executar as regras e comandos programadas pelo usuário fazendo a comunicação com o hardware da aplicação e o servidor de aplicação. Basicamente o Middleware irá autenticar no servidor baixar as regras de negócio que serão executadas, e enviar para o servidor dados das leituras dos sensores e ficar a espera de comandos do servidor por meio do navegador do cliente ou por meio do código de programa (script).

A grande vantagem desse modelo é a flexibilidade de configuração disponibilizada e o acesso de qualquer lugar do mundo através da Internet, assim o usuário de uma aplicação poderá intervir no fluxo de negócio da aplicação online.

3.2 Projeto Relacionados

Muitos projetos vêm sendo desenvolvidos pelas comunidades, apresentam-se uma pequena amostra de tais projetos e qual o foco da suas funcionalidades. Por exemplo, um desses projeto é um Framework para monitoramento de campos de viticultura e agricultura usando redes de sensores pelo padrão IEEE 1451 que visa disponibilizar uma solução para acomodar diversos tipos de sistemas, o padrão IEEE 1451 é composto por diversos outros padrões. E padrão IEEE 1451 provê a padronização estrutural da rede e do processo de aquisição de dados. O projeto adotou o padrão IEEE 802.15.4/ZigBee para se comunicar com o padrão IEEE 1451, assim permitindo dispositivos PnP (Plug-and-Play) ingressar na rede. Foi aplicado o modelo NCAP (*Network Capable Application Processor*) e também o modelo WTIM para os transdutores, cada transdutor que possui um TED (transducer electronic datasheet), fazendo com que pudessem ser configuráveis e auto descritivos na rede. Como camada de Gateway foi utilizado o padrão iPAGAT que encapsula o NCAP e os serviços de comunicação de dados com a central de coleta e gerenciamento de dados (FERNANDES SAMUEL G MATOS, 2013).

Outro projeto desenvolvido foi o INCOME (LI SHAOLIANG PENG, 2012), um modelo para distribuição de sensores em terras de cultivos que busca taxas de luz e sombra. Esse projeto aplica um modelo matemático para distribuição dos sensores em agrupamentos amostrais nas plantações, onde são avaliados o posicionamento dos sensores que serão implantados para poder tirar as taxas de sombra e luz nos campos de cultivos causados pela própria planta, essas taxas influenciam na fotossíntese das plantas negativamente.

Basicamente, os sensores são implantados em ciclos incrementais, a cada ciclo, os cálculos são refeitos e os novos sensores são reposicionados estrategicamente.

Outro projeto realizado automatizou o sistema de irrigação de cultivos utilizando redes de sensores. Assim através desse sistema levantou-se a quantidade de água que um cultivo recebe pela da chuva, umidade e pelo terreno, e a quantidade de água o cultivo perde para o ambiente através da evaporação, com essas medidas o sistema calcula a taxa de água ideal necessária baseada no histórico das medições anteriores (NIKOLIDAKIS, 2013).

O projeto será disponibilizado através desta dissertação como guia de consulta para construção de aplicativos, como modelo de arquitetura de serviços de aplicação em nuvem para IoT.

3.3 Limitações

As limitações estão predominantemente relacionadas à infra-estrutura do local de instalação, assim para a configuração do sistema WIREWAY é obrigatório o acesso a Internet, após o equipamento ter sido configurado o mesmo poderá executar sem Internet onde o dono da aplicação não poderá acessá-lo de alguma ambiente externo (offline) para o acesso externo (online) é necessário que acesso a Internet no local.

Existem alguns sistemas no mercado que executam tarefas similares à plataforma WIREWAY, como o Bluemix da IBM, que surgiu no mercado em meados de 2014. O Bluemix possui diversos serviços de computação em nuvem à disposição, entre todos os serviços disponíveis os que geram contraste com a plataforma WIREWAY são: *Internet of Things Foundation* e *IoT Real-Time Insights* que oferece diversas integrações com dispositivos de IoT com ferramentas analíticas (IBM, 2015).

3.4 Desenvolvimento do Sistema

O sistema foi desenvolvido utilizando as seguintes tecnologias: MySQL, JavaScript, HTML, CSS, Java, Linux, NodeJS, XBee, ATMeI e Raspberry. De forma simplista a parte de servidor roda sob o servidor NodeJS e utilizando MySQL para persistência dos dados. Nos capítulos seguintes serão apresentadas as APIs e Frameworks utilizados, já a parte o servidor Middleware irá rodar sob a plataforma Java devido as capacidades multiplataformas da JVM (Java Virtual Machine) que são necessárias para o funcionamento do programa. Na próxima seção começaremos a explicar a forma detalhada de desenvol-

vimento da aplicação WIREWAY tanto o Servidor como o Middleware. A comunicação entre o Middleware e o Servidor WEB, e entre o Servidor WEB com o Navegador do Cliente será feita através de WebSocket, uma evolução do HTTP permitindo comunicação bidirecional entre cliente-servidor.

3.5 Arquitetura de Comunicação

A plataforma escolhida para este trabalho, irá utilizar uma arquitetura derivada da PubSubCoord, devido ao modelo de comunicação ser muito similar porém com alguns aspectos de validação e controle que serão necessários para atender aos requisitos da plataforma. Os principais requisitos de comunicação da plataforma desenvolvida neste trabalho conforme Figura 3.1 são:

- Os clientes possuem prioridades de serviços diferentes, alguns precisam de processamento crítico em tempo real e outros não;
- Os clientes embarcados (Middlewares) podem se conectar ao servidor de aplicação para serem controlados;
- Os clientes de navegadores de Internet podem se conectar ao servidor de aplicação para controlar os clientes embarcados (Middlewares);
- Cada cliente possui acesso à vários painéis de controles, e cada painel de controle funciona como uma sala privada de compartilhamento (assunto em comum);
- Qualquer cliente, navegador de Internet ou embarcado (Middleware), pode se conectar a um controle, contanto que tenha as credenciais válidas;

Assim o modelo redefine alguns papéis do modelo PubSubCoord:

- Os representantes coordenadores são os servidores de aplicação, que gerenciam as salas privadas.
- Os representantes de borda embarcados se conectam diretamente as salas privadas para conversar.
- Os representantes de borda embarcados são Middlewares de aplicação que abstraem a comunicação com o servidor e normalizam os dados antes do envio.
- Os representantes de borda navegador de Internet são interfaces usadas por pessoas com acesso às salas privadas para controlar os clientes embarcados daquelas salas.

- As salas privadas ou painéis de controles são definidas como Dashboards e são identificadas por um identificador único.

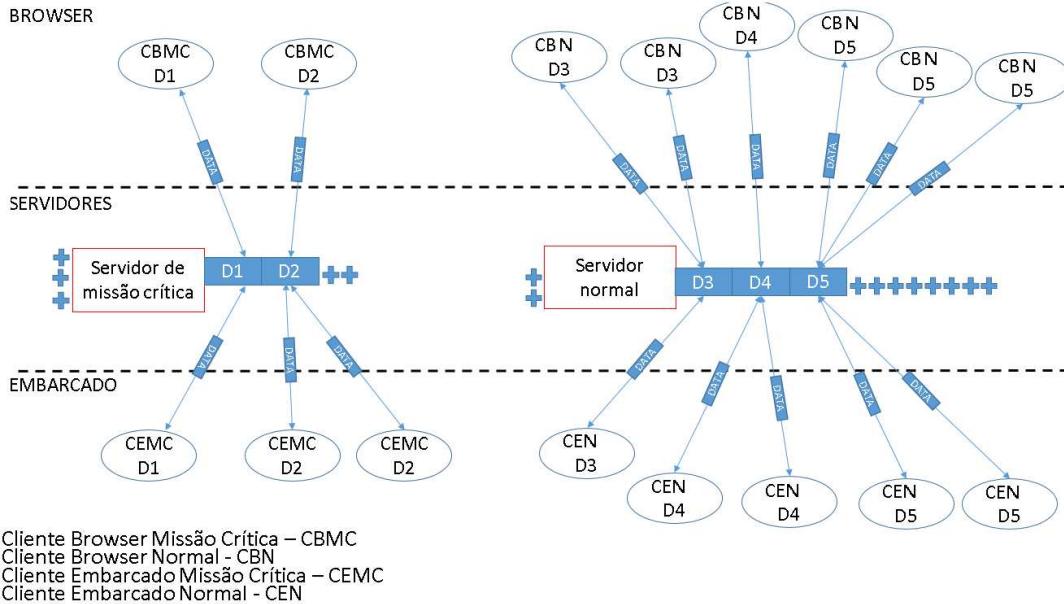


Figura 3.1: Modelo arquitetural OMG DDS Adaptado

Devido aos custos de aluguel de um servidor com alta escala de processamento para execução de missão crítica, fora realizado o teste somente em máquinas locais simulando servidores de uso normal. Os computadores utilizados possuíam processador Intel Core i5 com 6GB de memória RAM.

3.6 Modelagem do Sistema

O detalhamento da forma de desenvolvimento do sistema será iniciada pela abordagem do modelo de persistência, depois pelo servidor. Então será apresentado a camada core de comunicação entre Middleware e Servidor baseado no modelo PubSubCoord apresentado na seção Modelo de Distribuição de Dados. Depois será apresentado a forma de gerenciamento das aplicações do usuário, e a forma arquitetural do Middleware e sua configuração. Por fim o desenvolvimento de um programa prático para análise de um abrigo de cultivo de flores. Uma macro visão da aplicação final está apresentada na Figura 3.2:

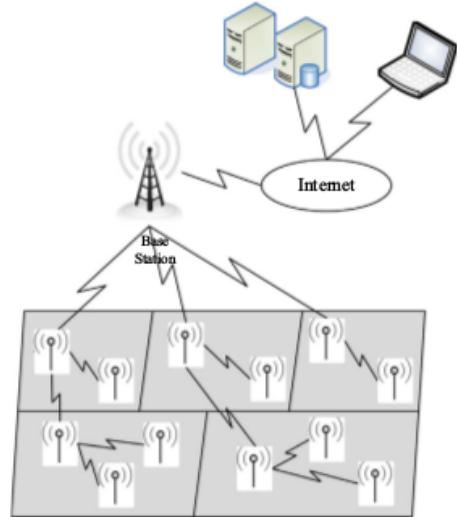
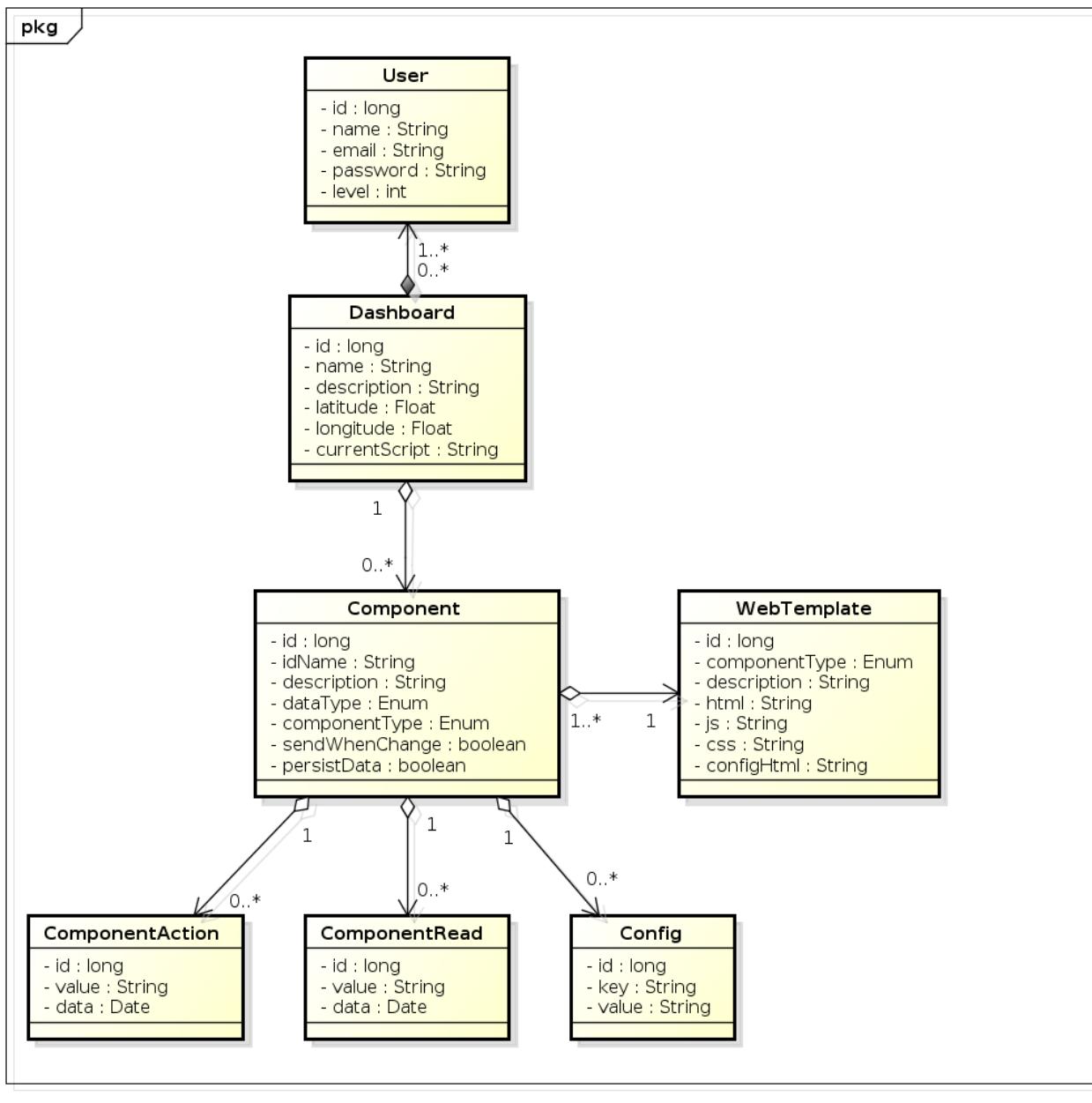


Figura 3.2: Arquitetura WSN (NIKOLIDAKIS, 2013).

3.6.1 Modelo de Dados

Para o desenvolvimento da aplicação WIREWAY, adotou-se o uso do banco de dados relacional MySQL usado para persistir os dados de gerenciamento das tabelas da aplicação, como Dashboards, Scripts, Usuários, entre outros. A Figura 3.3 ilustra o modelo objeto entidade relacional da aplicação.



powered by Astah

Figura 3.3: Modelo de dados da aplicação WIREWAY

Com o modelo segundo à Figura 3.3, é possível atender todo o funcionamento conceitual necessário para por a aplicação em funcionamento. Cada componente desempenha um papel extremamente importante para a convergência do sistema.

A tabela ”User” é responsável por armazenar as informações necessárias sobre o usuário, o núcleo do sistema necessita do usuário para realizar a autenticação para o funcionamento do Middleware. Além de associar os Dashboards pertencentes à determinados usuários, esses usuários podem controlar a aplicação IoT via WEB como será demonstrado nas próximas seções. O Middleware será o responsável por atuar sobre os atuadores

e ler os sensores rodando nas camadas mais baixas.

A tabela ”Dashboard” irá armazenar as informações de configuração do painel de controle das aplicações, nessa tabela haverá a identificação por aplicação, por exemplo, usuários podem ter diferentes Dashboards para controlar abrigos de cultivo, por exemplo: Dashboards para um abrigo de cultivo de orquídea, Dashboards para abrigo de cultivo de eucalipto, etc. Além disso um Dashboard tem um script de execução associado no HD, esse arquivo é baixado pelo Middleware e usado para iniciar a aplicação embarcada.

A tabela ”WebTemplate” armazena informações relacionadas a configuração visual do componente na página WEB dos Dashboards de usuário, por exemplo, ele carrega os scripts de estilo (CSS) e execução de componentes (JS), como: botões digitais, *spinners* de sensores analógicos, controles PWM (Pulse Width Modulation) (*button*) e o layout dos Dashboards. Além disso esses componentes são auto descritivos quanto à seu formulário de configuração pela propriedade ”configHtml”.

A tabela ”Component”, ”ComponentAction”, ”ComponentRead” e ”Config” armazena as informações relacionadas aos componentes (sensores e atuadores), a tabela ”Component” é usada em duas visões, pela visão da aplicação WEB (navegador), e pela visão da aplicação de Middleware, basicamente essa tabela corresponde a descrição e armazenamento de estados de um componente de um Dashboard, por exemplo, um componente de Rele Digital. Devido a heterogeneidade dos sensores a tabela ”Config” chave valor armazena o restante das propriedades que cada componente possui pela sua especialidade. A tabela ”ComponentAction” armazena as iterações feitas pelo navegador na tela WEB com o Middleware, e a tabela ”ComponentRead” armazena as informações do Middleware enviadas para o servidor.

O modelo será gerido pelo servidor de aplicação, que será o responsável pela parte de persistência, na próxima seção será melhor detalhado a arquitetura utilizada no servidor de aplicação.

3.6.2 Servidor de Aplicação

O servidor de aplicação escolhido para o WIREWAY foi o NodeJS, essa escolha foi feita baseado em valores de mercado que provam seu crescimento em relação a outras tecnologias e também pela padronização da linguagem de desenvolvimento que é o JavaScript. O NodeJS é um servidor de aplicação escalável com o conceito de ”Single Process” que suporta milhares de usuários com pouco uso de hardware em relação a servidores de aplicação como Jboss, Tomcat, IIS, entre outros. O Node JS já oferece uma biblioteca de comunicação WebSocket - Socket.IO sobre um protocolo baseado em eventos, além

disso para a persistência do modelo apresentado na seção anterior será utilizado o Sequelize, uma biblioteca responsável por mapear os objetos JavaScript para o modelo entidade relacional. A parte de autenticação da aplicação será feita pelo Passport, um Framework designado para gerenciar as credenciais dos usuários. As telas serão feitas usando Angular JS e Twitter Bootstrap. Para a camada de processamento REST (Representational State Transfer) será usado a biblioteca do Express, que oferece um alto nível de abstração para o processamento das chamadas HTTP, nas próximas seções serão mostrados os modelos de classe aplicados.

3.6.2.1 Modelo Persistência

A camada de persistência do servidor foi escrita manualmente, devido a flexibilidade de troca de banco e de Framework de persistência caso venha a ser necessário. Usar somente o Sequelize sem a abstração do mesmo, geraria muito código a ser reescrito caso surgisse a necessidade de trocar o Framework ou sobre-escrever algum método mais específico. Além disso esse modelo de persistência nos permite avaliar a possibilidade de utilizar dois bancos para persistência, um banco de dados MySQL para persistir os dados que requerem transações e maior segurança em caso de *Rollback*, e outro banco NoSQL como Mongo para persistir os dados das tabelas "ComponentRead" e "ComponentAction". Esse modelo consome o modelo de dados apresentado na seção anterior, conforme a Figura 3.4.

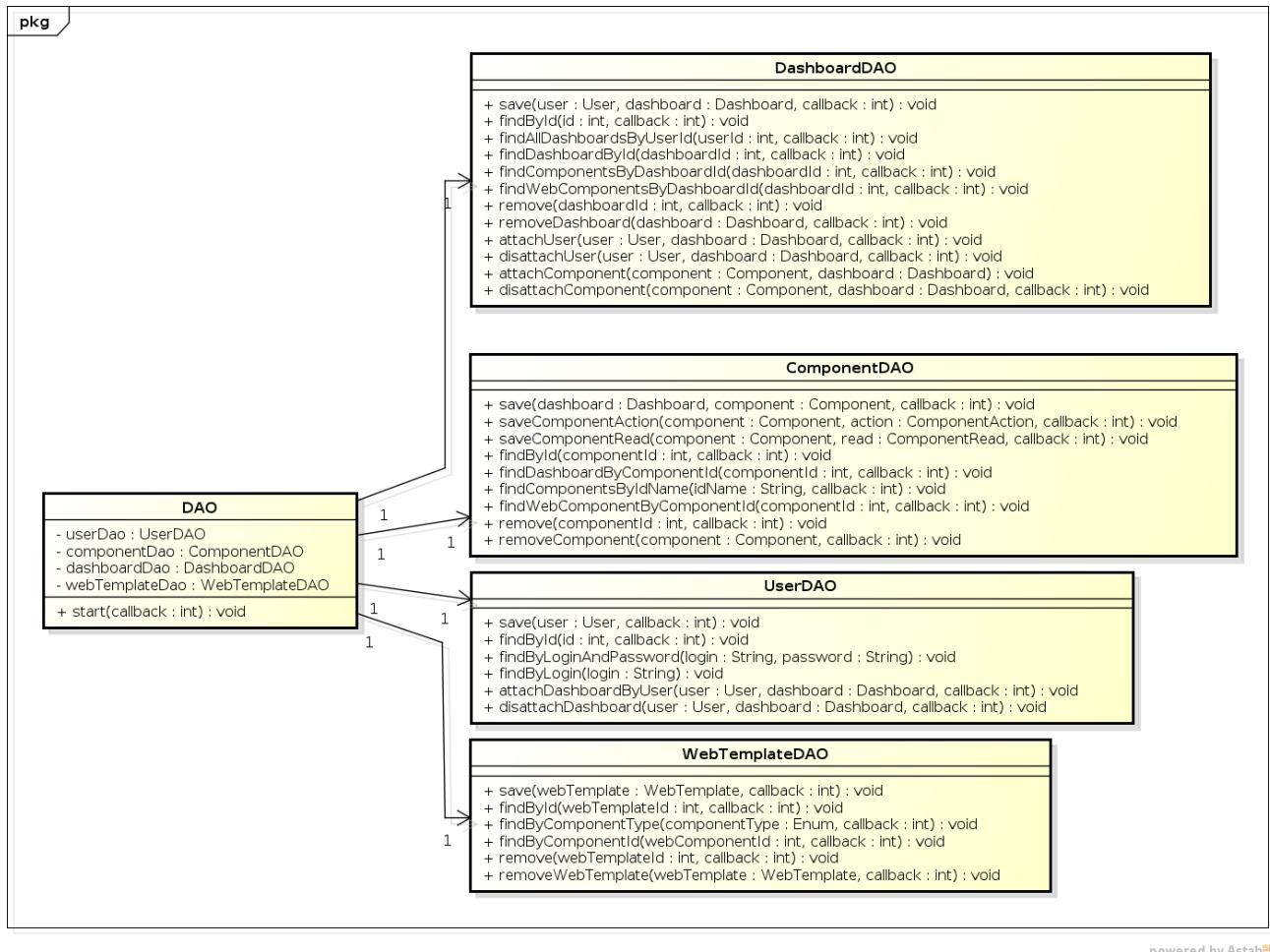


Figura 3.4: Modelo da camada de persistência da aplicação WIREWAY

3.6.2.2 Segurança da Aplicação

Na plataforma WIREWAY um usuário precisa ter uma conta de acesso para poder utilizar os recursos da aplicação conforme modelo adaptado PubSubCoord, assim o mesmo deve criar uma conta caso ele já não tenha, seguindo o caso de uso da Figura 3.5. No momento de criação da conta, no caso de uso ”Cria registro” a senha é gerada em resumo criptográfico SHA-1, e após esse processo o mesmo já pode logar no sistema para fazer uso. Esse modelo de caso de uso apresenta dois casos fundamentais para o funcionamento da aplicação: ”Entrar em sala de controle (Dashboard)” e ”Entrar em sala de controle (Middleware)”, basicamente esses dois modelos vão funcionar através da tecnologia WebSocket que será apresentado posteriormente através de um diagrama de sequência. Resumidamente esse dois casos permitem que um usuário possa controlar qualquer sistema de IoT associado ao seu perfil. Quando o usuário inicia através da aplicação de Middleware no sistema WIREWAY o mesmo começa a executar sobre as configurações estabelecidas pelo

seu perfil, e quando o usuário deseja controlar a aplicação que ele iniciou pelo Middleware o mesmo loga no seu Dashboard WEB através do navegador para poder manipular o Middleware. A Figura 3.5 mostra o caso de uso de autenticação.

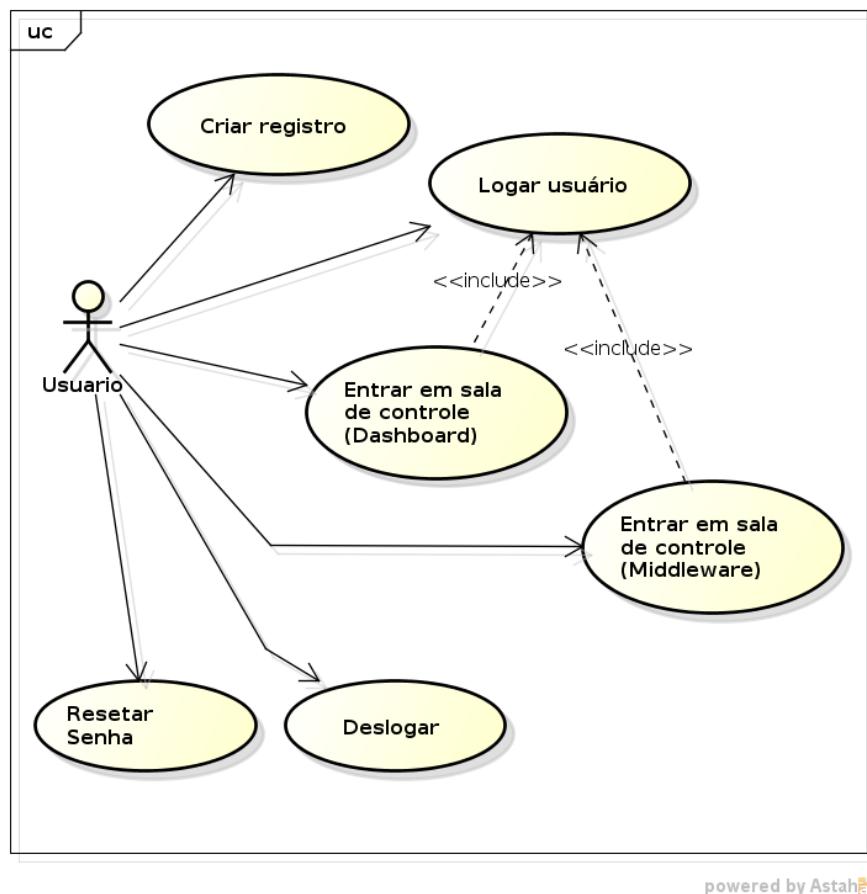


Figura 3.5: Caso de uso da Autenticação WIREWAY

O fluxo de autenticação e inicialização do sistema Middleware é demonstrado na imagem 3.6. O diagrama funciona da seguinte forma, o usuário possui um perfil de acesso, mas ele necessita logar na aplicação de Middleware para que seja carregado as configurações de atuação do mesmo. Assim o Middleware será o responsável pelas leituras e atuações de transdutores. Para realizar as atividades o Middleware precisa de uma sessão válida no servidor para poder entrar em uma sala de controle. Isso é feito pelo usuário através do seguinte fluxo: o usuário loga na aplicação como Midleware através de um script de autorização local que se comunica com o servidor, dessa forma ele pode selecionar um dos Dashboards que estão sob seu domínio, então será buscado na base todas as configurações de componentes válidos e também o programa que dará ação ao Middleware. Assim, essa camada consegue fazer a leitura e atuação dos seus sensores e atuadores e mandar os dados para o servidor. Agora com o Middleware funcionando o

usuário pode logar pela Internet no site de aplicação WIREWAY através do navegador de Internet e acompanhar através do Dashboard do Middleware a sua execução, ver como os sensores estão se comportando e influenciar no seu comportamento através da interação com os componentes, conforme Figura 3.6.

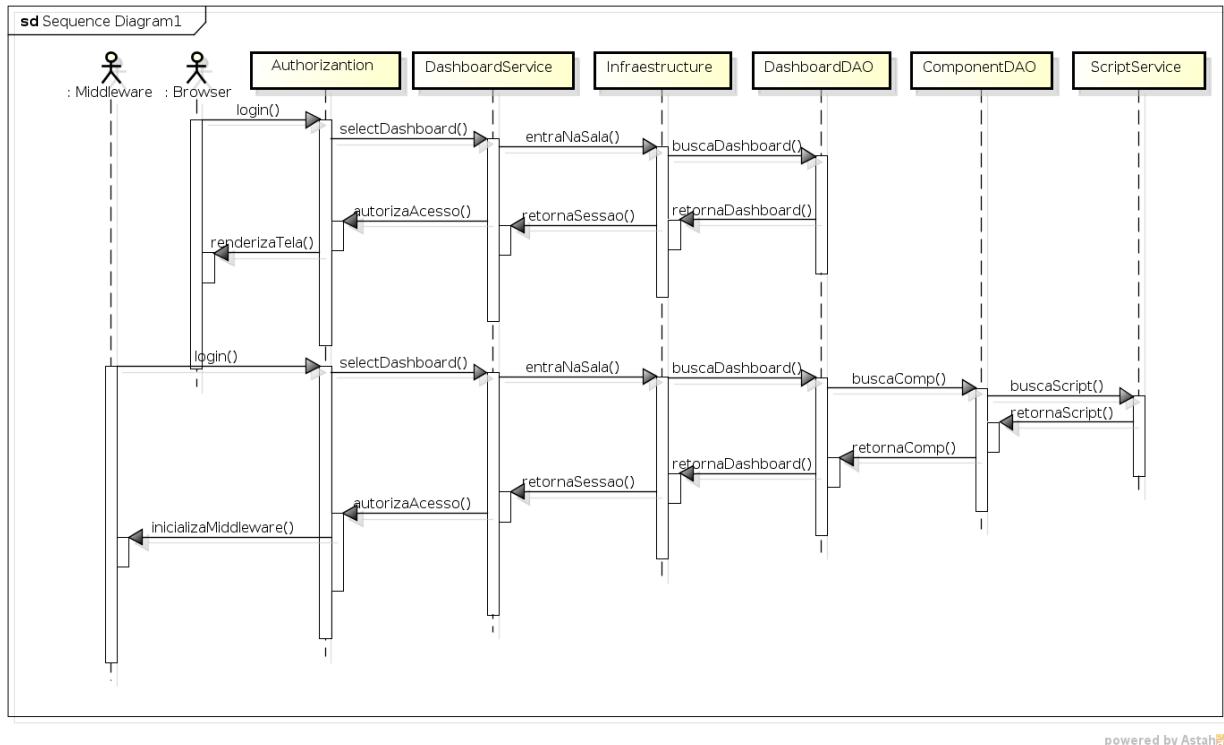


Figura 3.6: Diagrama de sequencia para autenticação WIREWAY

O modelo de rotas da Figura 3.7 contém todas as chamadas *REST* utilizadas no servidor. Esclarecendo a organização da Figura, todas as rotas são concebidas por verbos *GET*, *POST*, *DELETE* ou *PUT*. As que possuem o sinal de negativo são rotas privadas que requerem autenticação e privilégios da aplicação para poderem ser acessadas, além disso quando um rota tem retorno **.html* quer dizer que ela retorna uma página WEB caso contrário é retornado um objeto literal do tipo JSON (*Java Script Object Notation*) (fora desenvolvido um modelo de comunicação com JSON por ser mais enxuto em relação ao XML usado pelos padrões OGC e SensorML). Grande parte das rotas é acessível pelo usuário. A rota */admin/** é responsável por definir quais páginas podem e quais não podem ser acessadas pelos usuários. A rota */dashboard/** contém todos os serviços necessários para se criar e gerenciar os Dashboards do usuário. Os componentes que são renderizados na tela para controle do Middleware mais especificamente vem das rotas */dashboard/DashboardView* e */dashboard/All*. A rota */home/** retorna a página principal e os Dashboards que o usuário pode inicializar enquanto a rota */public/** disponibiliza

acesso a todos os arquivos que são acessíveis sem credencial, como a página de login da aplicação e a página de cadastro. A rota `/script/*` gerencia a criação de programas executáveis que são interpretados pelo Middleware, ela permite manipular os arquivos organizados por seus respectivos Dashboards. Através dela se tem acesso a somente os próprios arquivos do usuário. A rota `/template/*` disponibiliza o carregamento de todos os componentes usados no processo de carregamento e renderização de um Dashboard do navegador, essa rota também permite o administrador da aplicação criar novos componentes que serão utilizados em Dashboards. Por fim o *Framework Passport* gerencia o credenciamento dos usuários, disponibilizando as rotas de *login* e *logout*.

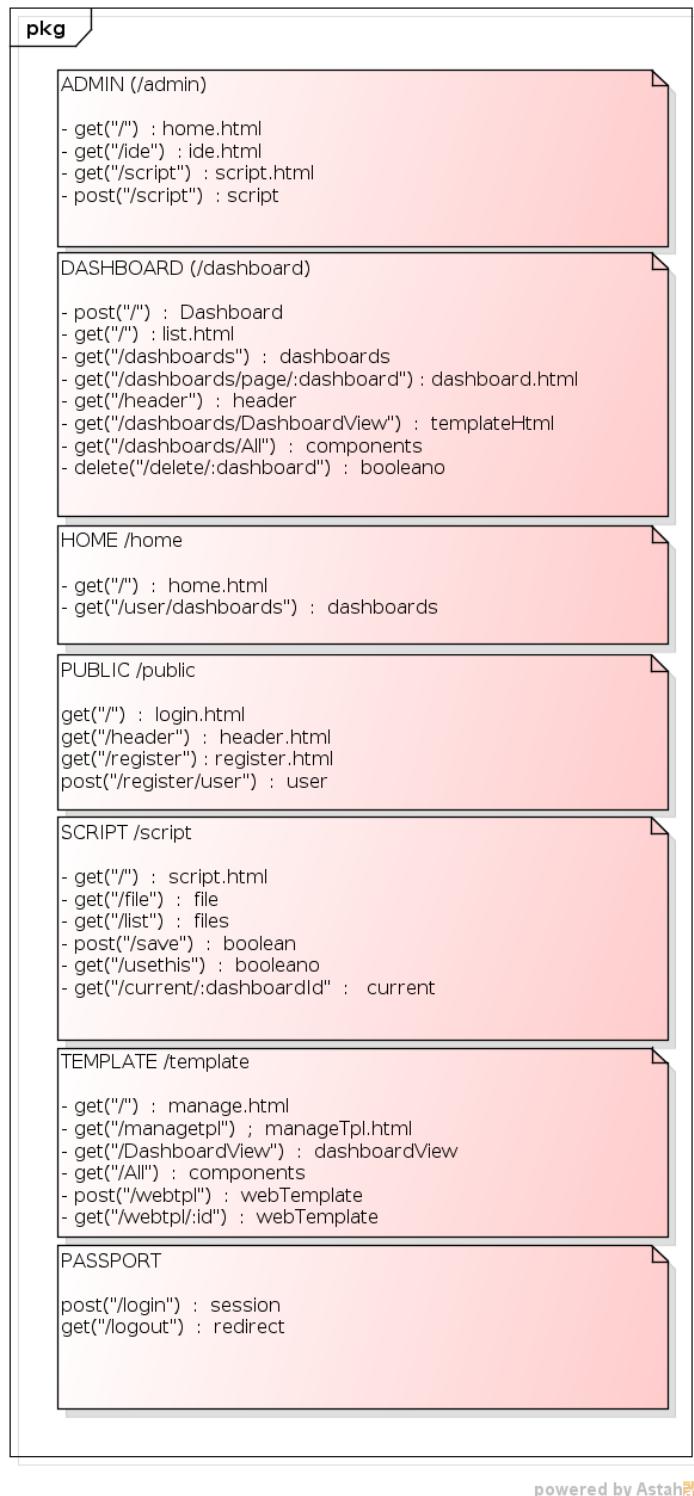


Figura 3.7: Rotas aplicação WIREWAY

3.6.2.3 Comunicação entre Middleware e o Servidor

Conforme apresentado na seção anterior sobre a base do funcionamento do servidor de aplicação, para a camada de comunicação entre servidor e clientes é necessário um

modelo de comunicação. Basicamente essa camada se divide em duas partes seguindo o modelo PubSubCoord adaptado, fazendo uma analogia aos grupos do Whatsapp (aplicativo de mensageiria) aonde as pessoas que fazem parte de um mesmo grupo e podem trocar informações. O servidor trata as conexões de forma parecida com o funcionamento da conexão através do Dashboard como assunto em comum entre um Middleware e um cliente via navegador de Internet. O Algoritmo 1 representa a lógica de conexão do cliente.

Algorithm 1 Associação de clientes em salas de controles

```

1: procedure WEBSOCKETCLIENTCONNECTION(User)
2:   if User ∈ Dashboard AND User ∈ ServerSession then
3:     if ADDTOROOM(Dashboard.Id, User) == Success then
4:       if User is Middleware then
5:         packet.script ← Dashboard.Script
6:         packet.components ← Dashboard.Components
7:         NOTIFYSUCCESS(packet)
8:       else
9:         NOTIFYSUCCESS
10:      end if
11:    else
12:      NOTIFYERROR
13:    end if
14:  end if
15: end procedure
  
```

Quando um cliente se conecta a aplicação WIREWAY no servidor via WebSocket através do navegador de Internet iniciado pelo Dashboard escolhido ele já deve ter uma sessão válida gerada por uma autenticação HTTP. No caso essa autenticação é estabelecida no momento que o usuário loga no servidor pela tela de login do Browser (Navegador de Internet) ou pela aplicação de Middleware quando escolhe o Dashboard que será utilizado, quando o Dashboard é selecionado ambos solicitam uma requisição para entrar em uma sala de controle, que é representado pela chamada de método *WebSocketClientConnection*. Assim é feita uma validação de segurança para validar a qual Dashboard o cliente pertence, se ele é um cliente de Browser ou um cliente de Middleware e se realmente ele tem um sessão válida no servidor, caso positivo, o mesmo é adicionado a sala de controle, no caso de um cliente de Browser a notificação de sucesso é imediata, já para o Middleware são carregados os parâmetros de inicialização de que a aplicação vai precisar para executar, então são baixados o programa e os Componentes do Dashboard, para então o mesmo ser notificado com o pacote de configuração. Caso o cliente não seja adicionado o mesmo recebe uma mensagem de erro.

A partir do momento que um Middleware está conectado a um Dashboard e um

cliente está visualizando através de um navegador de Internet o Dashboard, os clientes podem fazer a comunicação através de eventos, as funções apresentadas no Algoritmo 2 detalham o funcionamento desse método, a questão é que navegadores de Internet enviam pacotes para clientData e Middlewares enviam pacotes para middlewareData:

Algorithm 2 Comunicação entre clientes e middlewares

```

1: procedure MIDDLEWAREDATA(Packet)
2:   if Packet ∈ DashboardRoom AND Packet ∈ ServerSession then
3:     PROCESSMIDDLEWAREPACKET(packet)
4:     NOTIFYCLIENTS(packet)
5:   end if
6: end procedure
7: procedure CLIENTDATA(Packet)
8:   if Packet ∈ DashboardRoom AND Packet ∈ ServerSession then
9:     PROCESSCLIENTPACKET(packet)
10:    NOTIFYMIDDLEWARES(packet)
11:   end if
12: end procedure
  
```

A vantagem desse modelo de comunicação é o custo computacional devido ao pouco processamento para trocar mensagens entre clientes em navegadores e Middlewares de uma mesma sala. O funcionamento é muito simples quando um cliente no navegador e um Middleware pertencem a mesma sala relacionados a um mesmo Dashboard ambos se comunicam por ela, assim a diferença é que o Middleware se comunica como papel de Middleware chamando a função middlewareData que envia a mensagem tratada para todos os clientes de navegadores, enquanto os clientes de navegadores com papel de cliente enviam as mensagens para todos o Middlewares através da função clientData. Além disso, os dois métodos passam por avaliação e execuções antes de repassar a mensagem, como, por exemplo, salvar o pacote no banco de dados, adicionar valores, validações, etc.

A Figura a 3.8 descreve o pacote que trafega entre as três camadas: navegador, servidor e Middleware.

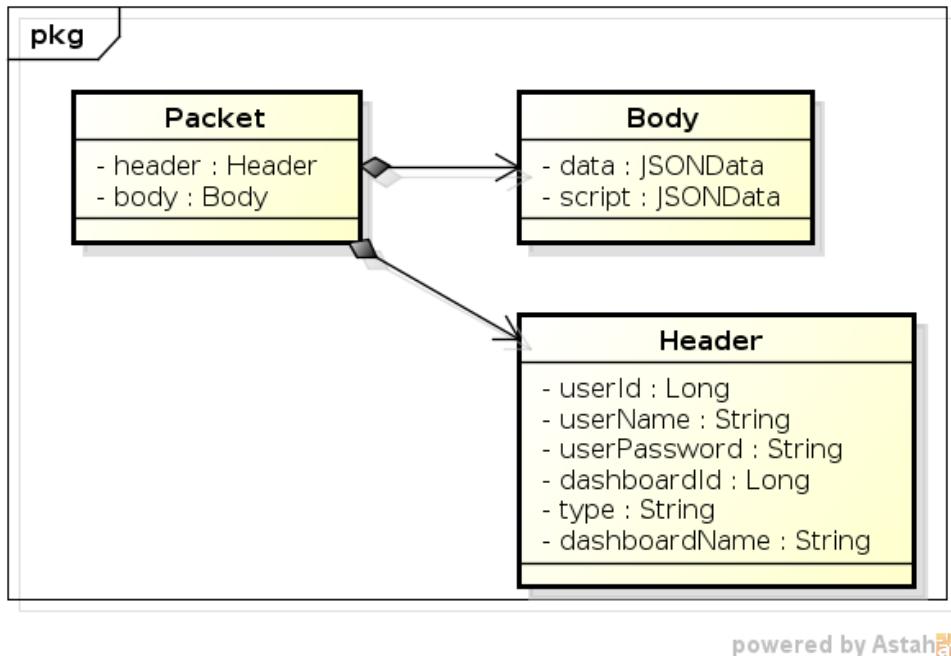


Figura 3.8: Pacote de comunicação do protocolo WIREWAY

Esse pacote é o principal modelo do protocolo de comunicação cliente-servidor *WebSocket* transmitido entre o navegador, servidor e Middleware. A classe *Header* determina as informações sobre o estado da comunicação em andamento em uma mesma sala no servidor para o navegador e o Middleware, basicamente ela carrega informações sobre o perfil do usuário que está transmitindo a mensagem e sobre o respectivo Dashboard em execução, enquanto a classe *Body* carrega as ações que o cliente e o Middleware estão executando. Porém o Middleware possui um acréscimo do objeto *script* que carrega o fonte inicial que o Middleware irá utilizar no *setup* do programa. Após iniciado o *setup*, os pacotes passam a ter somente os dados dos objetos no corpo de *Body*. Os dados que normalmente são transmitidos entre navegador, servidor e Middleware, são informações sobre transdutores como: sensores ou atuadores, que são visto na forma de componentes conforme modelo do banco de dados. Os componentes irão carregar consigo valores de leitura de sensores e estados para que os atuadores deverão receber, além disso irão também trazer metadados auto descritivos. O protocolo foi moldado dessa forma para permitir uma comunicação mais efetiva economizando a quantidade de parâmetros necessários.

3.6.3 Programa do Middleware

O Middleware é responsável por abstrair a comunicação de baixo nível entre o hardware embarcado e o software de aplicação em nuvem, porém ele executa essa tarefa

de forma dinâmica e programável permitindo que o mesmo possa ser programado em *JavaScript*. Por exemplo: um cliente cria uma tela para monitoramento das condições ambientais de um abrigo de cultivo de flores, ele faz a tela com dois botões e cinco sensores, as respectivas ações desses botões são: ligar o ventilador e ligar o fogger (névoa de água), os sensores possuem as respectivas atribuições: leitura de temperatura, umidade, pressão atmosférica, umidade do solo e chuva. As iterações que o usuário faz com os botões na tela refletem no Middleware. Toda vez que um comando é acionado um pacote conforme descrito na seção anterior é criado e enviado para o Middleware realizar o processamento de atuação. O mesmo acontece quando o Middleware realiza a leitura de algum dos sensores do abrigo de cultivo, será gerado um pacote com os dados dos sensores e esse pacote será entregue para a tela do navegador atualizar os valores dos labels. Um exemplo da tela descrita está na Figura 3.9 com valores de sensores e com os atuadores em desligado:

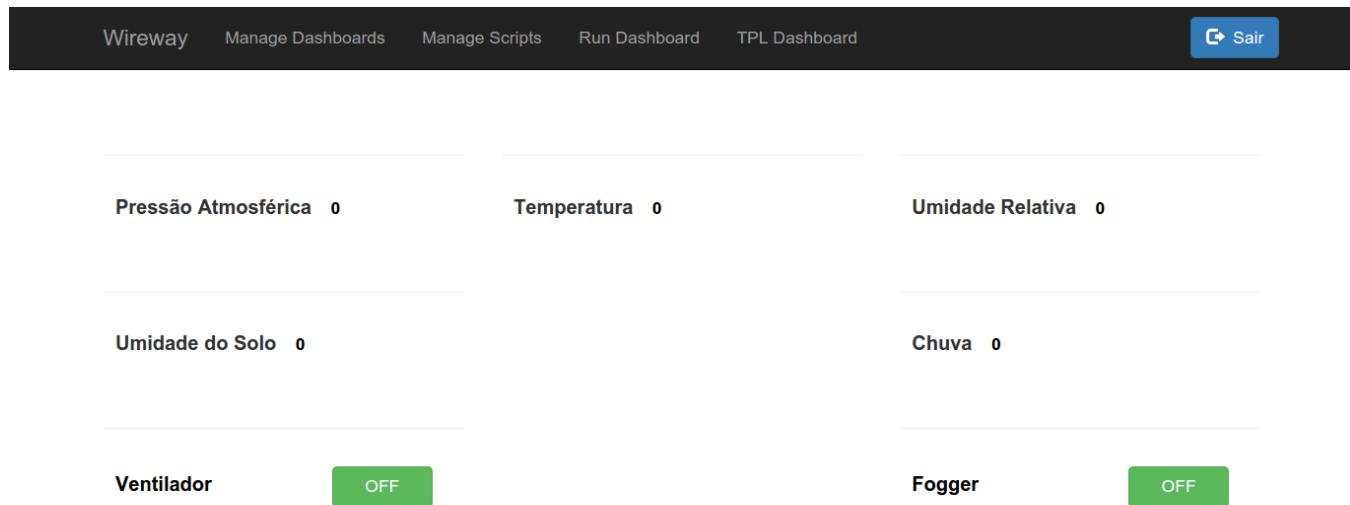


Figura 3.9: Dashboard de Execução da aplicação WIREWAY

O funcionamento do Middleware depende de algumas ações chave, como a programação da ação que o Middleware deve executar. Continuando com o exemplo do abrigo de cultivo a mesma deve ter um código simples para definir como os dados serão tratados pelo processador no cliente embarcado, por exemplo: a estrutura do código ficaria conforme Algoritmo 3.

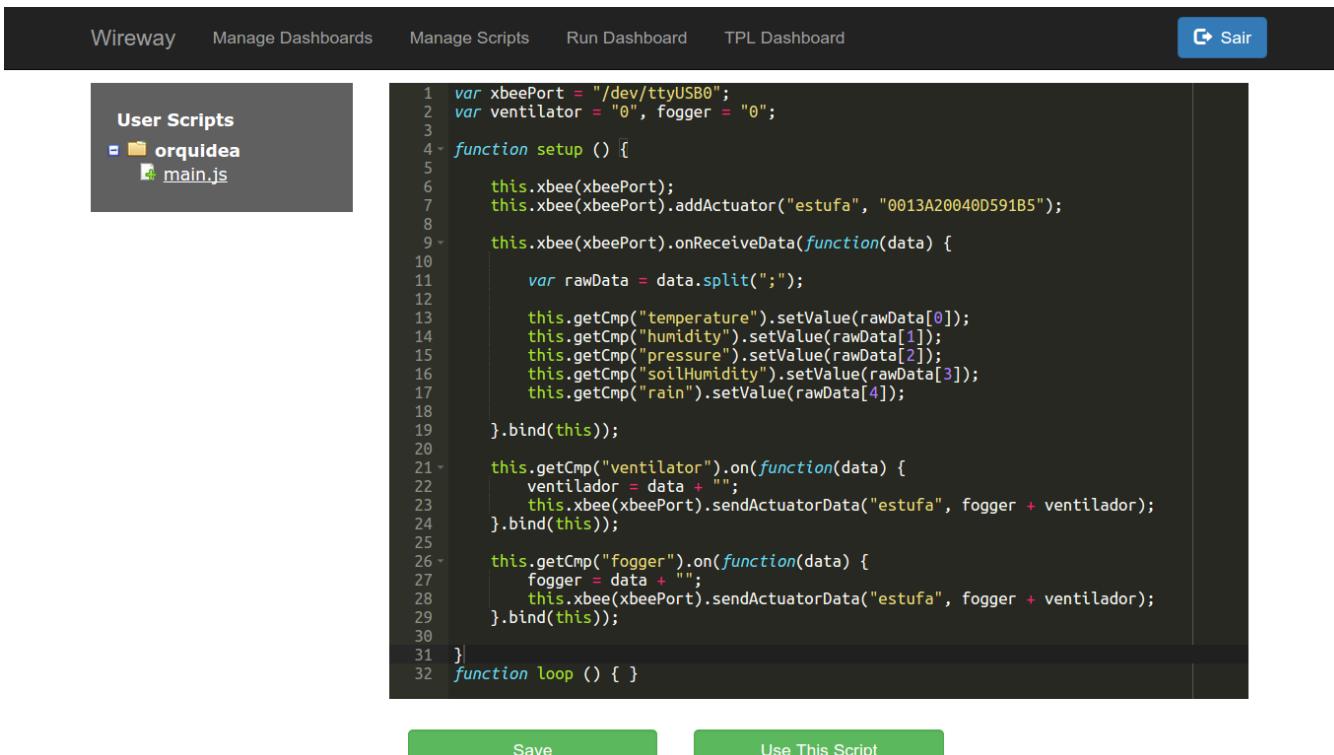
Neste algoritmo o rádio é um composto muito útil para facilitar a implantação

Algorithm 3 Programa simples de controle para abrigos de cultivo

```

1: RADIO.INITRADIO
2: RADIO.INITACTUATORS
3: if RADIO.RECEIVEPACKET then
4:   PROCESSPACKET
5:   UPDATEVIEWVALUES
6: end if
7: if VIEW.RECEIVEVENTILATORACTUATOR then
8:   NOTIFYVENTILATORACTUATOR
9: end if
10: if VIEW.RECEIVEFOGGERACTUATOR then
11:   NOTIFYFOGGERACTUATOR
12: end if
  
```

de sistemas heterogêneos, dessa forma o Middleware utiliza funções de acesso ao módulo *Serial* para se comunicar com o nó de sensoriamento e atuação. A Figura 3.10 demonstra o código funcional interpretado pela aplicação de Middleware, essa interpretação é um recurso do Java 8, o código é executado pelo *Nashorn* um motor de avaliação de códigos *JavaScript* pela JVM (Java Virtual Machine). Esse código é baixado dentro da propriedade *script* da entidade *body* na hora do pacote de comunicação, quando o Middleware inicializa sua configuração.



```

var xbeePort = "/dev/ttyUSB0";
var ventilator = "0", fogger = "0";
function setup () {
  this.xbee(xbeePort);
  this.xbee(xbeePort).addActuator("estufa", "0013A20040D591B5");
  this.xbee(xbeePort).onReceiveData(function(data) {
    var rawData = data.split(",");
    this.getCmp("temperature").setValue(rawData[0]);
    this.getCmp("humidity").setValue(rawData[1]);
    this.getCmp("pressure").setValue(rawData[2]);
    this.getCmp("soilHumidity").setValue(rawData[3]);
    this.getCmp("rain").setValue(rawData[4]);
  }.bind(this));
  this.getCmp("ventilator").on(function(data) {
    ventilador = data + "";
    this.xbee(xbeePort).sendActuatorData("estufa", fogger + ventilador);
  }.bind(this));
  this.getCmp("fogger").on(function(data) {
    fogger = data + "";
    this.xbee(xbeePort).sendActuatorData("estufa", fogger + ventilador);
  }.bind(this));
}
function loop () { }
  
```

[Save](#)
[Use This Script](#)

Figura 3.10: Código de controle do Middleware do Dashboard do Abrigo de Cultivo

O código que é interpretado no Middleware roda em cima da JVM e tem como pré-requisito uma versão superior ou igual a 1.8, pois depende de recursos da mesma, como o Nashorn (motor de interpretação JavaScript). A configuração do programa que roda no Middleware é muito simples, todo trabalho necessário é baixar o arquivo compactado do site principal, descompactar ele e executar o .bat (Windows) ou .sh (Linux). Quando a aplicação iniciar ela irá pedir suas credenciais de acesso, que são as mesmas do site para poder listar quais Dashboards estão disponíveis para execução, conforme discutido no diagrama de sequencia, Figura: 3.6.

Além dos recursos de comunicação *serial* pelo padrão ZigBee com rádios XBee, o Middleware disponibiliza funcionalidades para uso em aplicações embarcadas, dentre as tais são:

- Comunicação serial com rádios Xbee pelo padrão ZigBee.
- Comunicação serial limpa, sem nenhum protocolo, útil para comunicação com microcontroladores.
- Controle dos pinos GPIO nas versões para Raspberry.
- Servidor de aplicação REST com Jersey.
- Atualização na tela em tempo real.

Essas possibilidades fazem com que a camada de Middleware se torne flexível. O programa do Middleware disponibiliza acesso a esses módulos dentro do contexto das funções *setup* e *loop*, utilizando a palavra reservada *this* é possível alcançar todos os métodos de utilização dos recursos disponíveis. A principal característica dos códigos interpretados das funções principais *setup* e *loop* tem *setup* como a primeira função a ser chamada pela aplicação e a mesma só executa uma vez. Após isso o programa entra em uma repetição eterna na função *loop* realizando um processo programável repetitivo. Os métodos *getCmp* no código da Figura 3.10 retornam os componentes que são as representações da tela do Dashboard, o identificador definido para o componente na hora de criação do Dashboard é o mesmo identificador usado dentro do código no método *getCmp*. Essa é a chave que amarra o Middleware a tela do navegador, é a forma que ambos tem de conhecer sobre qual componente estão conversando, ou seja, de forma simples, o código programado na página WEB é executado no Middleware em Java, através de um contêiner de execução dinâmico.

3.6.3.1 Modelagem componentes do Middleware

O Middlware é desenvolvido em Java e planejado para executar em vários tipos de sistemas operacionais com suporte a Windows, Linux e MAC OS, as necessidades de bibliotecas de comunicação como a *Serial* são providas pela própria aplicação. O core da API Middleware está dividida em camadas, basicamente os pacotes estão organizadas conforme Figura 3.11, as funções de cada componente são:

- CORE é responsável por agrupar e disponibilizar em um contexto de execução os recursos disponíveis, ele serve com FAÇADE (padrão de projetos estrutural) para o código da aplicação do Dashboard;
- SOCKET realiza a comunicação bidirecional com o servidor através de pacotes;
- INTERPRETER executa o programa escrito no servidor usando a API Nashorn;
- EXCEPTION gerencia todos os tipos de erros que ocorrem no servidor;
- SERIAL provê funcionalidades para comunicação com as interfaces seriais;
- GPIO gerencia as portas de I/O somente disponível para Middlewares Raspberry;
- HTTP sobe um servidor de aplicação local;

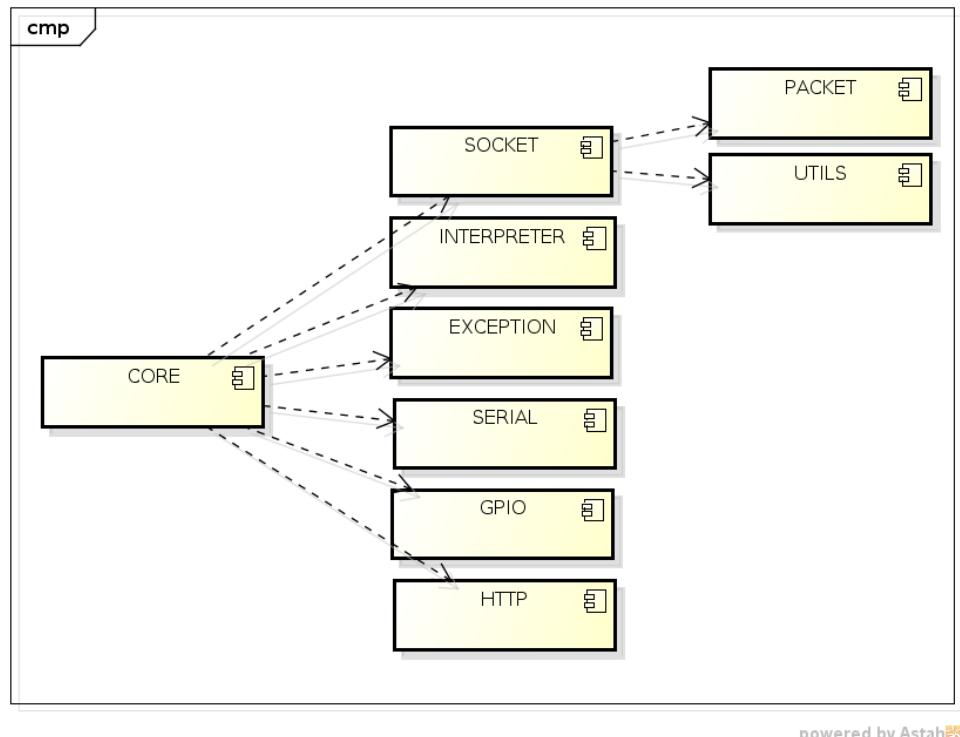


Figura 3.11: Modelo de componentes do Middleware WIREWAY

3.6.3.2 Modelagem core do Middleware

Além do modelo de alto nível da organização de pacotes da aplicação, o principal pacote como já apresentado é o CORE, ele requer uma descrição mais detalhada sobre seu modelo de classes apresentado na Figura 3.12. A funcionalidade de cada classe é descrita a seguir:

- *Scope* provê um modelo genérico que será usado como contexto para a execução do *script*, de forma prática toda vez que você usar *"this"* dentro dos métodos *setup* ou *loop* estará acessando o objeto Scope.
- *Context* estende a classe abstrata *Scope* e provê acesso aos módulos e funcionalidades que poderão ser usadas no Middleware, quando o *Context* é iniciado ele recebe as características do Dashboard selecionado com seus componentes.
- *Dashboard* é a classe que encapsula as características do Dashboard baixado do servidor no momento de inicialização do *Context*.
- *Component* São os componentes associados aos Dashboards escolhidos na inicialização do Middleware, os *Components* são representações de atuadores ou sensores que irão atualizar a tela ou executar ações conforme a comunicação com o servidor descrever, representando componentes como: Reles, Sensores de Umidade, Motores, etc, toda vez que um valor de Component é modificado no Middleware ele também é modificado automaticamente na tela.
- *ComponentEvent* são os eventos que um componente pode ter, toda vez que uma atuação é enviada da tela de Dashboard para o Middleware o *Component* que tem um evento atrelado irá chamar as rotinas cadastradas para aquele evento.
- *GPIO* provê a funcionalidade de alterar ou ler estados das portas de propósito geral do Middleware quando o mesmo é um Raspberry.
- *Run* É a classe que inicializa a aplicação, ele solicita as credencias do site e configurações de rede e então carrega o arquivo de configuração da aplicação, *Run* solicita o Dashboard que será executado após validar as credenciais e carrega todo o contexto com suas dependências, também carrega o programa de execução e chama o INTERPRETER.

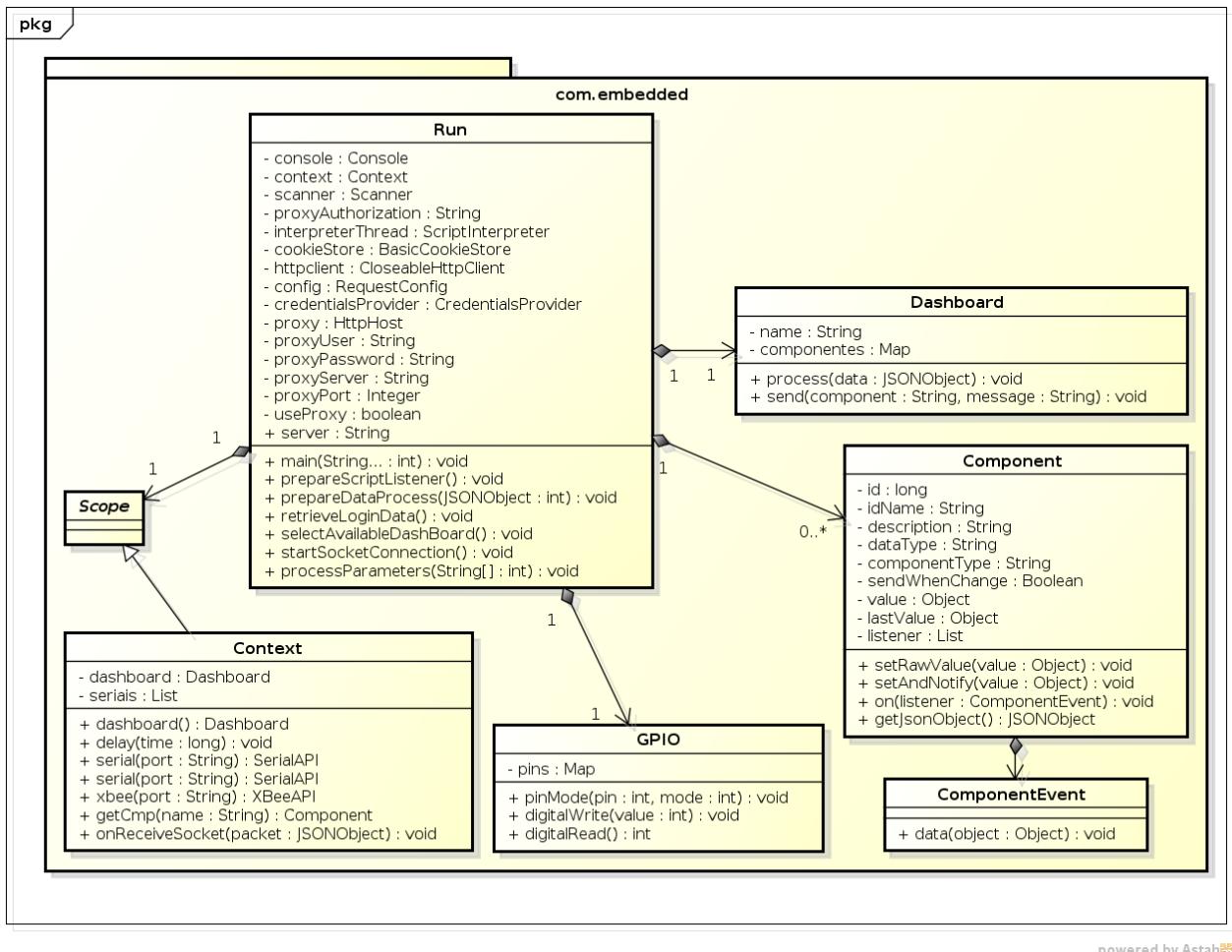


Figura 3.12: Modelo de pacotes do Core do Middleware WIREWAY

3.7 Conclusão

Este capítulo apresentou os processos, tecnologias e arquiteturas usadas para desenvolver a plataforma WIREWAY, a plataforma WIREWAY apresentada no desenvolvimento é capaz de criar e publicar aplicativos de IoT através do site e do programa de Middleware.

Capítulo 4

Controlando um Abrigo de Cultivo com a Plataforma Wireway

Como já citado anteriormente para testar a plataforma WIREWAY desenvolvida neste trabalho, criou-se uma aplicação modelo utilizando XBee e o controlador ATMEL 328P. Nesta seção será apresentada toda a etapa de criação de uma aplicação embarcada para IoT, desde o processo de configuração do hardware até o uso da plataforma WIREWAY para realizar a telemetria e controle de um abrigo de cultivo de flores.

Partindo da análise de requisitos do sistema de telemetria do abrigo cultivo usando WIREWAY, deseja-se analisar os dados de temperatura, umidade, pressão, umidade do solo e chuva e apresentá-los para o usuário através de um Dashboard com botões para que o mesmo possa ligar o ventilador ou o fogger do abrigo de cultivo. Com isso a criação do Dashboard é feita na aba *Manage Dashboards* conforme Figura B.2. Nesta tela o usuário seleciona o modelo de tela que ele vai usar para organizar os componentes e tela, neste caso fora selecionado um modelo de três colunas e adicionado sete componentes de onde cinco são os sensores e dois são os botões para ligar os atuadores conforme Figura 3.9.

4.0.1 Programa de comunicação do Abrigo de Cultivo

O programa que será executado no Middleware é criado pela aplicação na aba *Manage Scripts*, o código utilizado na aplicação do abrigo de cultivo já fora demonstrado na Figura 3.10. E a execução do programa é muito simples, neste caso não há necessidade de nenhuma instrução cíclica no método *loop*, ele vai funcionar como um intermediário entre o rádio e a tela. O algoritmo de forma sequencial executa as seguintes ações:

- Define uma variável que irá carregar a porta aonde o rádio XBee está conectado e depois inicializa duas variáveis que mantém o estado do atuador em memória para atuação, já que o atuador espera receber uma sequência binária com os seguintes

estados: 00 (desliga tudo), 01 (liga o ventilador), 10 (liga o fogger) e 11 (liga fogger e ventilador).

- Após isso o algoritmo entra no método *setup* e então abre a comunicação com o rádio XBee na porta definida anteriormente, para então depois adicionar o endereço do rádio que contém os atuadores, é o mesmo rádio que está no coletor e faz as coletas de dados dos sensores.
- É adicionado um observador para o rádio, assim todo pacote de dados de sensores recebido pelo rádio é transmitido para a função *callback*. Nesse momento os dados vêm no formato separado por ; que são quebrados para um array de cinco posições, cada posição corresponde a medição de um sensor, esse valor é setado para cada componente na tela do navegador.
- É adicionado um observador para o botão da tela para o ventilador e para o fogger, assim a qualquer momento que um atuador desses for clicado na tela no navegador de Internet, o observador vai receber o estado lógico da atuação, montar o pacote de dados no formato esperado pelo atuador e enviar via rádio para o mesmo.

O protocolo de comunicação que o Middleware usa para comunicar com a estação de coleta/atuação é formatado da seguinte forma, quando a estação de coleta faz as leituras dos sensores os dados são arranjados separados por ; para reduzir a quantidade de bytes que será transmitida, fazendo com que a comunicação seja mais eficiente, os dados foram arranjados no formato: *temperatura;umidade;pressao;umidadeDoSolo;chuva;* assim não cria-se pacotes com mais de 40 bytes. O processamento fica na parte de codificação e decodificação, porém é um processamento muito insignificante, avaliando a complexidade do algoritmo. O mesmo fica com complexidade constante de notação $O(n)$, já os dados que são enviados do Middleware para a estação de coleta seguem o formato binário já exemplificado na lista anterior, fizera-se dessa forma por termos dois relês. Os dados coletados vem na seguintes unidades de medidas: graus Celsius para temperatura, porcentagem para umidade relativa, kilos pascal para pressão atmosférica, porcentagem para umidade do solo e porcentagem para chuva (até ser feita calibração do sensor).

4.0.2 Rádio XBee

Para o controle e monitoramento do abrigo de cultivo foram utilizados os rádios XBee Series 2 de 2 miliwatts de potência da DIGI (Figura 4.1), esses rádios são fáceis de achar no mercado e o preço gira em torno dos 150 reais. A escolha desse rádio foi pelas

seguintes características: consumo de 40 miliampere em 3,3 Volts, taxa de dados máxima de 250 kbps, potência de saída de 2 miliwatts (+3dBm), raio de cobertura 120 metros, certificado FCC, criptografia de 128 bits, entre outros. Dois rádios foram utilizados um em modo *ZigBee Coordinator API* com o Firmware versão XB24-ZB (21A7) conectado ao Middleware através de um *sniffer* pela porta USB (/dev/ttyUSB0) e outro rádio em modo *ZigBee Router API* com o Firmware versão XV24-ZB (23A7) conectado ao coletor.



Figura 4.1: Rádio XBee S2 (Digi, 2015).

As configurações para os dois rádios seguem da seguinte forma:

- ZigBee Coordinator API
 - Pan Id (1234)
 - Destination Address High (FFFF)
 - Node Identifier (COORDINATOR)
 - API Enable (2)

- ZigBee Router API
 - Pan Id (1234)
 - Destination Address Low (FFFF)
 - Node Identifier (ROUTER)
 - DIO7 Configuration (DISABLE)
 - API Enable (2)

A escolha do modo API foi feito pelo fato do protocolo ZigBee acrescer a confiabilidade e capacidade de recuperação da rede. Cada pacote que é enviado tanto pelo coordenador como pelo roteador é enviado aguardando uma resposta de confirmação. Além disso cada pacote no modo API possui uma definição, por exemplo, para transmissão de dados é necessário enviar uma *Transmit Request* e para uma transmissão de

status uma *Modem Status* e assim por diante, além da cada pacote poder ser criptografado eles também contém um *checksum* que válida se a mensagem foi recebida corretamente. Os pacotes podem também ser direcionados para uma rede completa via *broadcast* tanto como *unicast*. Com tudo isso a estação coletora envia de tempos em tempos um pacote de *Transmit Request* no modo API para o coordenador da rede com as mensagens no formato com os valores separados por ; conforme já explicado anteriormente, enquanto o coordenador envia para o atuador o valor do Rele quando uma atuação acontece no botão da tela no navegador do Dashboard do abrigo de cultivo.

4.0.3 Placa coletora

O sistema de telemetria e controle do abrigo de cultivo requer um pequena placa com sensores e reles para ser controlada pela plataforma Wireway, o microcontrolador utilizado nessa placa foi o ATMEGA 328PU com Bootloader, a escolha desse modelo foi feita devido as seguintes características: Flash de 32k, CPU de 8 bits AVR, Instruções RISC, Padrão Industrial, 32 x 8 GPIOS, Seis canais PWM, Comunicação Serial e I2C. De fato atendendo os requisitos necessários para coleta, os demais componentes estão listados abaixo:

- 1 - ATMEGA328-PU
- 1 - Fonte regulador com saída de 5 Volts e 3.3 Volts
- 1 - Rádio Xbee Series 2
- 1 - Base para rádio XBee
- 1 - Sensor de temperatura e umidade DHT22 (AM2302)
- 1 - Sensor de pressão atmosférica BMP180
- 1 - Sensor de umidade do solo com comparador LM393
- 1 - Sensor de presença de chuva com comparador LM393
- 2 - Shields reles para 2 reles de 10 Amperes
- 5 - Resistores de 300 Ohms
- 4 - Leds LX21
- 1 - Sniffer XBee

- 2 - Capacitor de 22pf
- 1 - Cristal Oscilador de 16Mhz

O circuito foi modelado utilizando o software Fritzing, e modelado a versão PCB e funcional, maiores informações podem ser encontradas no apêndice Circuito Coletor, uma visão funcional da prototipagem do circuito está na Figura 4.2.

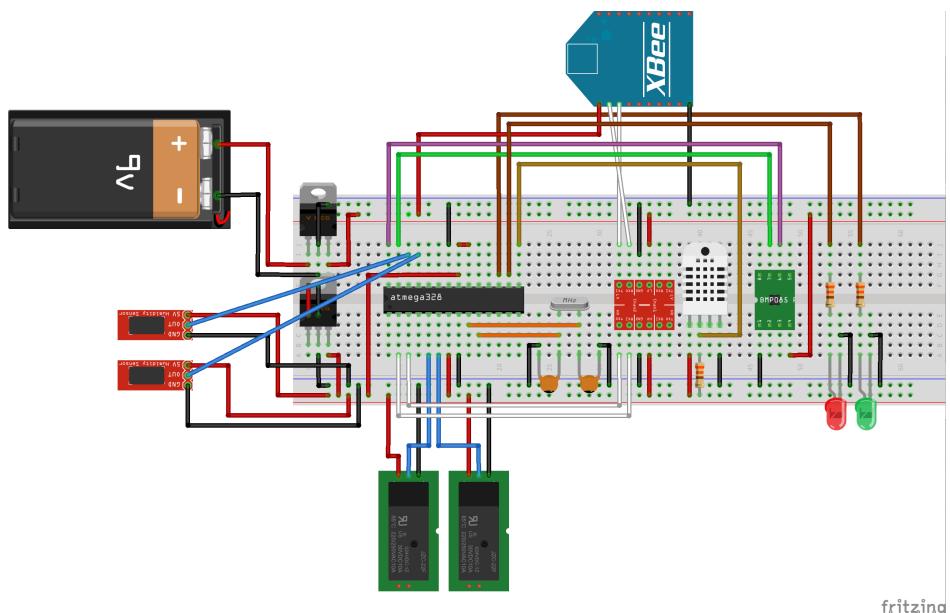


Figura 4.2: Circuito do Coletor na Protoboard.

O programa da placa foi escrito em C++ utilizando a IDE do Arduíno que usa o compilador AVR-GCC e para fazer a gravação do código usa o AVR-DUDE. Para esse programa foram usados as seguintes bibliotecas Open Source: Wire, DHT, AdafruitBMP085, Arduino e XBee. O fluxo do código segue o Algoritmo 4.

4.0.4 Middleware controlador

Para o sistema de controle do abrigo de cultivo foi utilizado o Raspberry PI 2 model B (conforme Figura 4.3) para atuar, como Middleware. Seu custo varia em torno de 270 reais e com ele é possível acessar a Internet via rede cabeadas ou sem fio, além de rodar muitos recursos de sistemas operacionais, pois o mesmo usa uma distribuição derivada do Debian chamada Raspbian.

Algorithm 4 Programa do coletor

```

1: LOADIMPORTS
2: DEFINEVARIABLES
3: procedure SETUP
4:   STARTSENSORS
5: end procedure
6: procedure LOOP
7:   READXBEE
8:   if timeCounter  $\geq$  30s then
9:     SENDCOLLECT
10:    end if
11: end procedure
12: procedure READXBEE
13:   packet  $\leftarrow$  xbee.packet
14:   if packet  $\neq$  null then
15:     if packet.type == response then
16:       STARTRELAYS(packet.message)
17:     end if
18:   end if
19: end procedure
20: procedure SENDCOLLECT
21:   xbee.send(COLLECTDATA)
22:   if xbee.packet  $\neq$  null then
23:     if xbee.type == response then
24:       if xbee.status == SUCCESS then
25:         NOTIFY(SUCCESS)
26:       else
27:         NOTIFY(ERROR)
28:       end if
29:     end if
30:   end if
31:   timeCounter  $\leftarrow$  0
32: end procedure
33: procedure COLLECTDATA
34:   READSENSORS
35:   MOUNTMESSAGE
36:   return message
37: end procedure
  
```

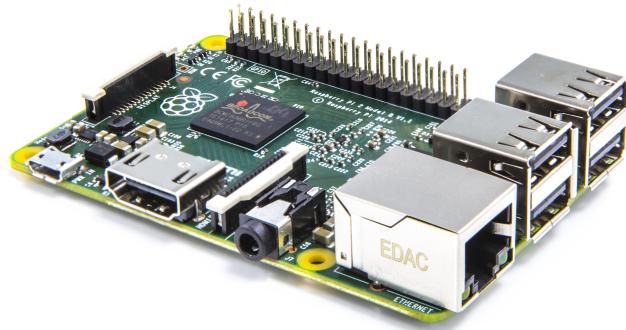


Figura 4.3: Raspberry PI 2 MODEL B (UPTON, 2015)

De forma resumida ele é um computador de menor capacidade de processamento com as seguintes características:

- Processador de 900MHz quad-core ARM Cortex-A7
- 1GB de RAM
- 4 portas USB
- 40 GPIOs
- Saída de vídeo HDMI
- Porta Ethernet
- Slot para cartão de memória (HD da placa)
- Núcleo gráfico VideoCore IV 3D

A versão do Raspbian usada neste trabalho foi a Raspbian Wheezy liberada em 5 de maio de 2015. Essa versão já possui a versão do Java 8 instalado não necessitando fazer a instalação, a única dependência foi instalar o pacote *rxtx-serial java* pelo comando *sudo apt-get install librxtx-java*. Para comunicar o Raspberry com a Internet foi usado um adaptador WiFi Realtek com a seguinte configuração:

- Padrão: IEEE 802.11n, IEEE 802.11g e IEEE 802.11b
- Faixa de frequência: 2.4GHz ISM
- Taxa de transferência de: 150 Mbps
- Interface USB 2.0

- Segurança: WEP/WPA/WPA2/WPA-PSK/WPA2-PSK (TKIP / AES)
- Sistema operacional: Linux / Mac / Windows XP/ Vista / 7 / 8

No Raspberry precisa-se baixar o programa do Middleware que é composto pelo *jar* de execução, um executável .bat ou .sh, e uma pasta com as bibliotecas do sistema operacional para acesso a *interface serial*, antes de iniciar a estação coletora conecta-se ao Raspberry na porta USB usando um Sniffer com o XBee em modo Coordenador API e então o programa pode ser iniciado. Quando o programa do Middleware inicia ele pede as credenciais do usuário e se autentica no servidor, então o servidor mostra ao Raspberry quais são os Dashboards que o usuário autenticado tem permissão para executar. O usuário escolhe um dos Dashboards e então o programa baixa a lista de componentes que serão usados no contexto do Middleware e o código de execução também. Assim o programa inicia a interpretação do código no Middleware e qualquer componente no código quando recebe uma leitura de sensor é refletido no navegador do cliente e qualquer componente do Dashboard do navegador do cliente que recebe um valor é refletido no Middleware durante todo tempo de vida do Middleware.

A partir deste momento a aplicação está operacional, o usuário pode logar no sistema em qualquer lugar para verificar os sensores do abrigo de cultivo e mandar comandos de atuação para o mesmo. O fluxo é bem simples, para atuação, o usuário clica em um botão da tela no navegador que está programado pelo Middleware para escutar o evento de clique, esse valor é recebido no Middleware e a atuação usa o sniffer conectado à porta USB para enviar os dados a estação coletora através dos rádios XBee, quando a estação coletora recebe a pacote a mesma interage com o atuador do abrigo de cultivo.

As leituras dos sensores ocorrem da seguinte forma, quando a estação coletora no abrigo de cultivo faz a leitura de um sensor, a mesma empacota as informações e envia pelo rádio os dados para o Middleware, onde tem um função atrelada pelo programa do usuário que faz a quebra do pacote dos dados dos sensores e define os valores nos componentes, quando recebem uma leitura do coletor enviam o dados para o servidor WEB parar atualizar todos os navegador de clientes logados naquele Dashboard do abrigo de cultivo, com isso o sistema tornou-se funcional para o cliente.

4.1 Conclusão

O sistema proposto neste trabalho visou facilitar a publicação e configuração de sistemas para Internet das Coisas. O desenvolvimento da plataforma WIREWAY, fora feito para facilitar a criação e gerenciamento de Dashboards administrativos de aplicações

embarcadas que seguem a arquitetura navegador, servidor, Middleware, além de disponibilizar a programação de regras dinâmicas para funcionamento dos sistemas embarcados.

Através da aplicação WIREWAY abstraiu-se toda a complexidade dos pacotes de transporte de dados, dessa forma todas aplicações criadas na plataforma WIREWAY seguem a convenção estabelecida na Figura 3.8. Avaliou-se a complexidade da criação de uma aplicação WEB para visualização e controle dos coletores e atuadores comparando os seguintes requisitos sobre o uso da plataforma WIREWAY e sobre o não uso:

- Conhecimento em arquitetura cliente servidor: Necessidade de configuração do servidor de aplicação WEB, entendimento do funcionamento do protocolo HTTP ou WebSocket para aplicações mais atuais, conhecimento em padrões de transferência de mensagens como REST, SOAP, JSON, XML, etc.
- Conhecimento em desenvolvimento WEB: Necessidade de desenvolvimento em JavaScript, CSS, HTML. Conhecimento em responsividade ou adaptatividade para aplicações mais evoluídas e nos padrões de usabilidade atuais. Conhecimento em servidores de aplicação como NodeJS, JBoss, Apache ou IIS e linguagens como Java, C# ou JavaScript.
- Conhecimento em desenvolvimento local: Conhecimento em bibliotecas e linguagens de programação local que fazem o papel de Middleware, nesse caso várias linguagens podem ser usadas como C, C++, Java, Python, JavaScript, C#, Delphi, etc.
- Conhecimento em eletrônica: Conhecimento em eletrônica para desenvolvimento de placas embarcadas ou montagem de circuitos utilizando placas prontas (*shields*), sensores e atuadores.
- Conhecimento em sistemas operacionais: Conhecimento das especificidades da plataforma que está sendo utilizada, caso a aplicação seja desenvolvida para ser multiplataforma deve ter suporte principalmente aos sistemas Windows, Linux e IOS.
- Conhecimento em banco de dados: A persistência dos dados para análises posteriores requer o conhecimento em modelagem de banco de dados, arquitetura de banco de dados e SQL.
- Conhecimento em programação para embarcados: Desenvolvimento para sistemas embarcados, além do conhecimento em sua eletrônica como já citado, é necessário o conhecimento em compiladores, IDEs, padrões e características de microcontroladores.

- Conhecimento em comunicação Serial: Conhecimento no padrão serial para comunicação entre o Middleware com coletores e atuadores, conhecimento obtenção de dados e gerenciamento dos equipamentos via cabo ou rádio frequência.
- Conhecimento em JavaScript: conhecimento em desenvolvimento JavaScript, é um conhecimento necessário para qualquer aplicação que será disponibilizada WEB, a linguagem atualmente pode ser usada no desenvolvimento de páginas WEB, servidores de aplicação, Middlewares e embarcados.

Para a análise da complexidade do conhecimento e tempo de desenvolvimento de aplicações com uso da plataforma WIREWAY em relação ao desenvolvimento de uma aplicação do zero foram levantadas as quantidades de conhecimentos necessários para a produção do produto completo, catalogados da seguinte forma: Conhecimento Desnecessário (Peso 0), Conhecimento Parcial (Peso 1), Conhecimento Completo (Peso - 2) e também considerou-se mais uma variável, a de tempo aonde cada unidade representa uma hora de desenvolvimento, a comparação está representada na tabela 4.1 aonde os levantamentos estão agrupados por com WIREWAY (usando WIREWAY) e sem WIREWAY (desenvolvendo o aplicativo completo sem usar o WIREWAY). A comparação demonstrada na Tabela 4.1 se fez comparando o desenvolvimento de um aplicativo inteiramente do zero (Coletor, Middleware e Servidor) para telemetria de tratores usando as mesmas tecnologias (XBee, ATMEGA328P, Java, NodeJS) que não fora usada nenhuma plataforma contra o desenvolvimento do abrigo de cultivo demonstrado neste trabalho no Capítulo 4.

Tabela 4.1: Comparação entre desenvolvimento com e sem usar o sistema WIREWAY

CRITÉRIO	C/ WIREWAY		S/ WIREWAY	
	Conhecimento	Tempo	Conhecimento	Tempo
Arquitetura cliente e servidor	0	0	2	1
Desenvolvimento WEB	0	0	2	20
Desenvolvimento local (Desktop)	0	0	2	20
Eletrônica	2	20	2	20
Sistemas operacionais	1	1	2	1
Banco de dados	0	0	2	1
Programação para embarcados	2	5	2	5
Comunicação Serial	1	1	2	5
JavaScript	1	5	2	5

A partir da Tabela 4.1 o desempenho em porcentagem do uso do aplicação WIREWAY em questão do tempo e conhecimento necessário para o desenvolvimento de uma aplicação embarcada para WEB em relação ao não uso do WIREWAY é expresso pela Equação 4.1.

$$100 - \left(\frac{\sum_{j=1}^n CW_j * CP_j}{\sum_{j=1}^n SW_j * SP_j} * 100 \right) \quad (4.1)$$

Aonde n é a quantidade de critérios usados na Tabela 4.1, e CW é o conhecimento em peso COM WIREWAY, CP é as horas de desenvolvimento COM WIREWAY, SW é o conhecimento em peso SEM WIREWAY e SP é as horas de desenvolvimento SEM WIREWAY, o resultado fica em torno de $\approx 63\%$. Analisando esse valor podemos dizer que o uso da aplicação WIREWAY desenvolvido nesse trabalho reduz o custo com equipamentos, tempo, estudo e pesquisa em torno de $\frac{5}{8}$ em relação ao desenvolvimento de uma aplicação do zero. As causas disso estão explícitas na Tabela 4.1, pois a plataforma WIREWAY proposta neste trabalho oferece uma arquitetura dinâmica, completa e flexível para desenvolvimento, esse fato elimina a necessidade de pessoas especialistas de outras áreas.

Capítulo 5

Conclusões

Concluiu-se que os aplicativos desenvolvidos em nuvem que seguem um dos modelos de PaaS, SaaS ou IaaS como o software desenvolvido neste trabalho WIREWAY, de fato podem oferecer muitos ganhos em relação ao desenvolvimento convencional como já demonstrado. A padronização e centralização da comunicação das aplicações reduz a complexidade de manter sistemas heterogêneos que até então não compartilhavam de uma mesma semântica e arquitetura. Isso possibilita que os dados possam ser usados com maior facilidade para gerar conhecimento aplicado, em uma aplicação de *data-mining*.

As principais dificuldades durante o desenvolvimento deste trabalho foram em relação a parte eletrônica que consumiu mais tempo do que o previsto em comparação com o servidor de aplicação e o software do Middleware devido as tecnologias que seriam usadas (ATMEGA328P e Raspberry) pelo não conhecimento do autor. Além dos estudos fora do escopo deste trabalho que tiveram de ser realizados para fabricação do circuito. Alguns dos principais problemas enfrentados foram referente a fabricação da placa e teste dos componentes eletrônicos.

Pontos positivos deste projetos foram: a utilização do NodeJS como servidor de aplicação que possibilitou a criação do servidor em tempo muito rápido e tornou a aplicação muito estável (em rede local, tendo zero perdas de pacotes) para troca de mensagens pela rede através do padrão Socket.IO (WebSocket) que fora aplicado entre o Middleware e o Servidor de Aplicação fazendo com que a comunicação bidirecional entre os componentes fosse muito rápida (em rede local, o ping ficou abaixo de 100 milissegundos) sem necessidade de utilizar métodos de pooling no cliente para iteração com o servidor. Por fim outro ponto positivo foi a utilização do padrão ZigBee nos rádios do modelo XBee entre a comunicação do hardware coletor, que adicionou muito estabilidade e confiabilidade ao sistema de coleta através do modo API conforme descrito nos itens de configuração da Seção 4.0.2.

Ainda para o futuro visa-se evoluir o trabalho e criar interfaces de serviço que sigam os padrões de semântica para consulta e notificação estabelecidos pelo mercado como SensorML do grupo OGC, aonde a criação de micro interfaces para o sistema WIREWAY serão criadas para oferecer integração com outros softwares, como aplicações Mobile, Empresariais e Redes Sociais.

Capítulo A

Círcuito Coletor

Círcuito do coletor parte de baixo Figura A.1 e circuito do coletor parte de cima A.2, está sendo usado dois reguladores de tensão nesta modelagem, porém o protótipo fora feito com uma fonte que já entrega os dois níveis de tensão, assim retira-se os reguladores e utiliza-se a fonte, essa placa poderia ser industrializada, assim na Figura A.3 temos a forma de disposição dos componentes na placa de 137,5 mm por 71,8 mm.

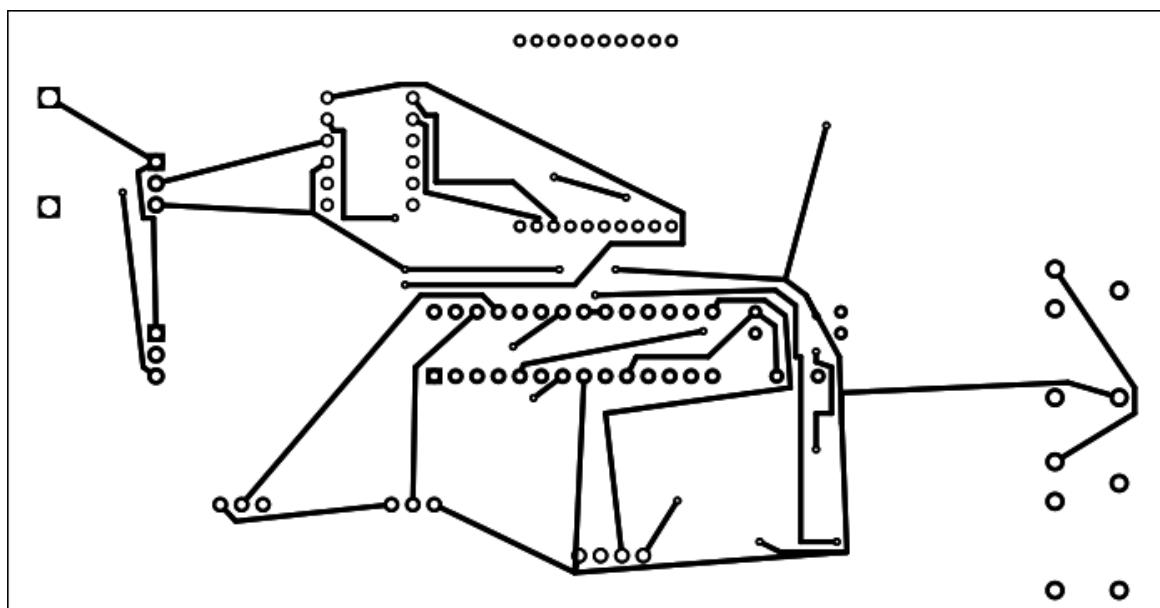


Figura A.1: Circuito Conexões Baixo

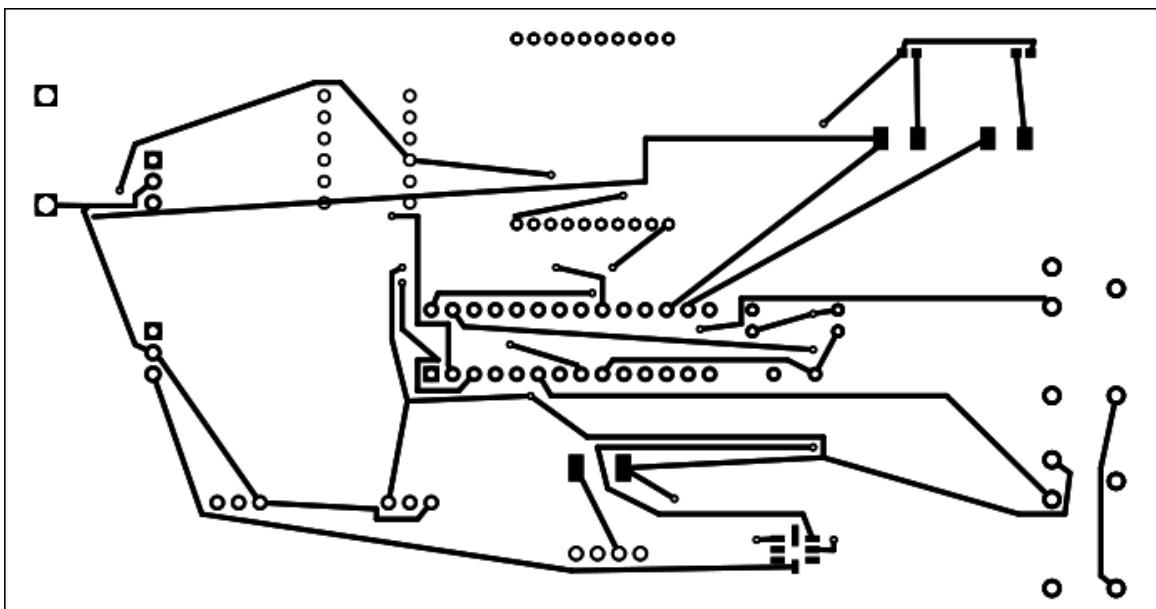


Figura A.2: Circuito Conexões Cima

Outras considerações sobre a placa é o uso dos relês, os relês estão representados por shields, que não está representado nesse circuito o diodo para dissipar a energia da bobina, o conversor lógico também é composto por um shield, e não está representado os componentes do mesmo.

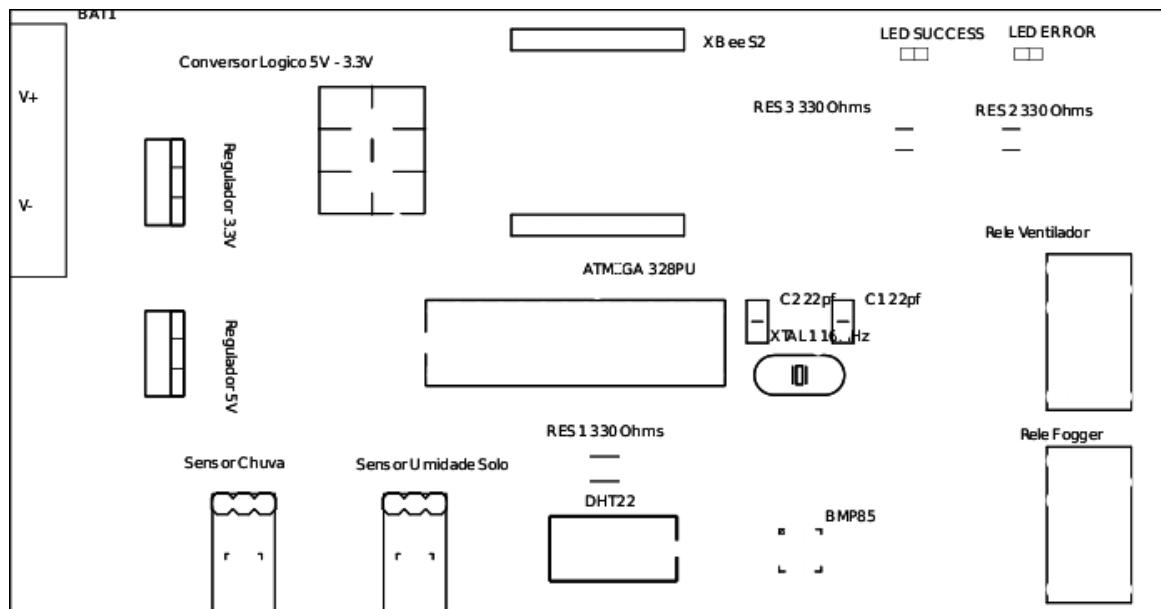


Figura A.3: Placa dos componentes

Os pontos de solda e furo da placa estão apresentados na Figura A.4.

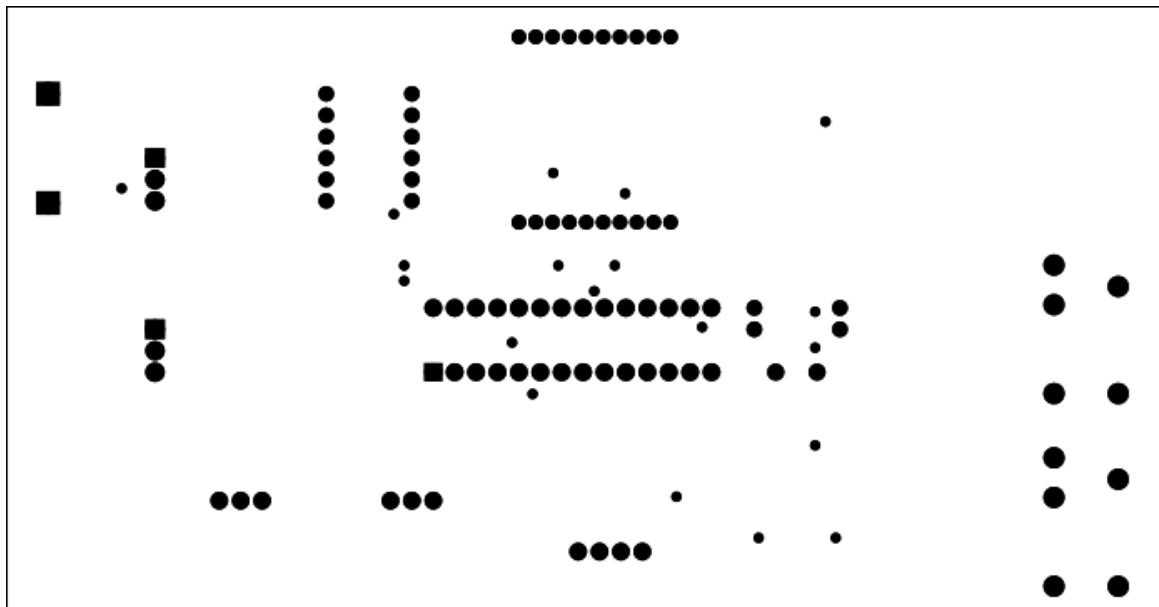


Figura A.4: Pontos de solda e furo

Capítulo B

Telas da Aplicação WEB

Tela de login da aplicação WEB, Figura B.1.

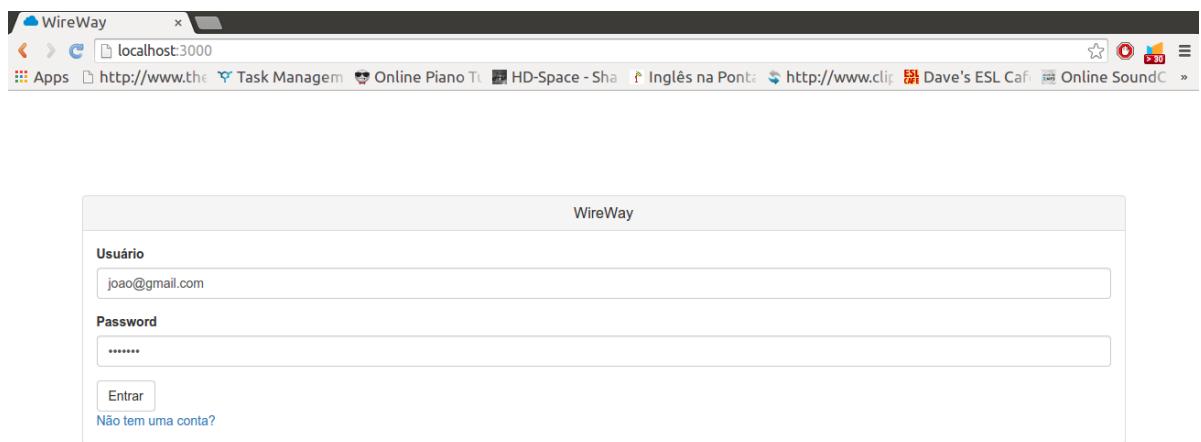


Figura B.1: Tela de Login

A tela usada para criar, deletar e modificar Dashboards administrativos segue na Figura B.2, a tela de seleção de Dashboard é a da Figura B.3.

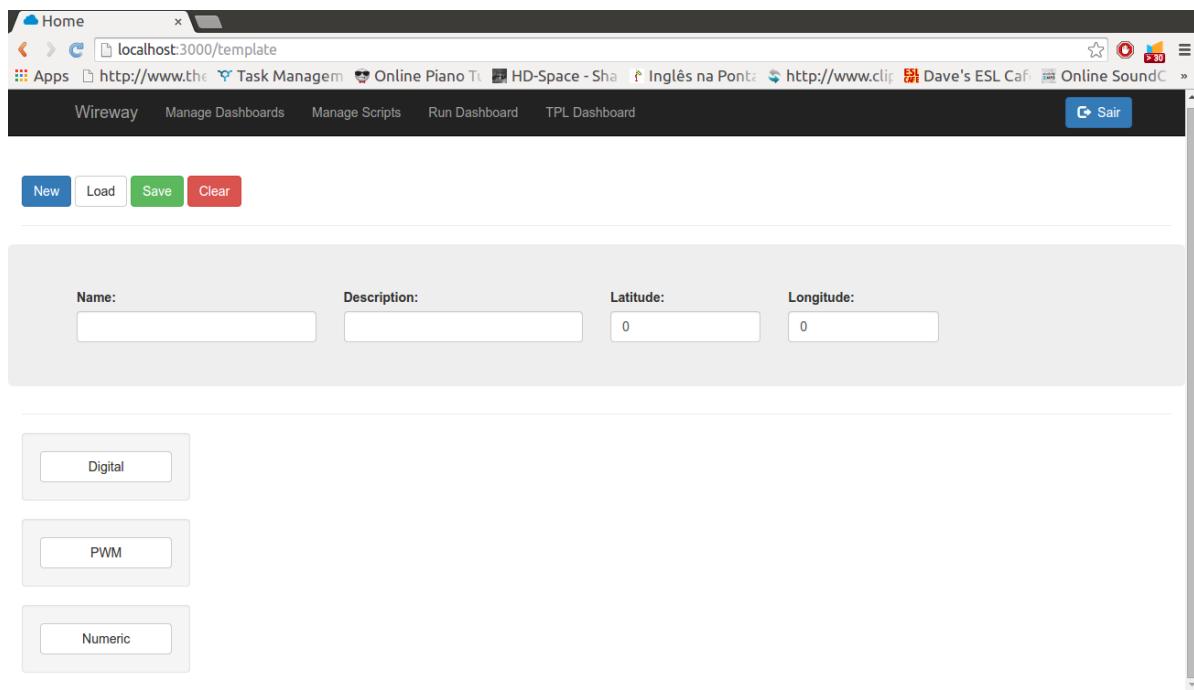


Figura B.2: Tela de Gerenciamento de Dashboards

#	Name	Description	Latitude	Longitude
#1 Select	Pesque Pague Dori	Lago Massaranduba Dori	-26.5669	-49.5565
#3 Select	Estufa Orquídeas	Estufa Orquídeas Garagem	-26.2345	-52.1234

Figura B.3: Tela de escolha de Dashboards

A inicialização do Middleware ocorre conforme Figura B.4, aonde ao executar o comando *ls -l* são exibidos os arquivos baixados do site, e então com o comando *java -jar wireway.jar* a aplicação é iniciada, então é requerido usuário e senha e disponibilizado um Dashboard para execução.

```
root@joao-Inspiron-5423:/home/joao/Área de Trabalho/wireway# ls -l
total 2152
drwxrwxrwx 2 joao joao    4096 Mai  2  2015 native
drwxrwxrwx 6 joao joao    4096 Ago  8 14:44 SO LIBS
-rw-rw-r-- 1 joao joao 2183678 Set  7 11:12 wireway.jar
root@joao-Inspiron-5423:/home/joao/Área de Trabalho/wireway# java -jar wireway.jar
Login: joao@gmail.com
Password:
Availables:
0 = ( Pesque Pague Dori )
1 = ( Estufa Orquídeas )
Select one: 1
```

Figura B.4: Iniciando Middleware

Referências Bibliográficas

- AN, K.; GOKHALE, A. A cloud-enabled coordination service for internet-scale omg dds applications. *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*, ACM New York, p. 310–313, 2014.
- ATZORI, A. I. L.; MORABITO, G. The internet of things: A survey. *Computer Networks*, v. 54, n. 0, p. 2787–2805, 2010.
- BARNAGHI, e. a. P. Semantics for the internet of things: Early progress and back to the future. *International Journal on Semantic Web e Information Systems*, v. 8, n. 1, p. 1–21, 2012.
- BERNERS-LEE, T. *WEB 3.0 and Linked Data*. 2009. Disponível em: <[>](http://www.w3.org/2009/Talks/0427-web30-tbl/\#(1)).
- COETZEE, J. E. L. et al. The internet of things - promise for the future? an introduction. *IST-Africa Conference Proceedings*, IEEE, p. 1–9, 2011.
- CONSORTIUM, O. G. *OGC Sensor ML: Model and XML Encoding Standart*. [S.l.], 2014.
- CONSORTIUM, O. G. *Sensor Model Language (SensorML)*. 2015. Disponível em: <[>](http://www.opengeospatial.org/standards/sensorml).
- CORDIS. 2014. Disponível em: <[>](http://cordis.europa.eu/ictresults/index.cfm?section=news&tpl=article&BrowsingType=Features&ID=89453&highlights=web+3;0).
- DANTAS, M. *Computação Distribuída de Alto Desempenho. Redes Clusters e Grids Computacionais*. [S.l.: s.n.], 2005.
- DIGI. *ZigBee*. 2015. Disponível em: <[>](http://www.digi.com/technology/rf-articles/wireless-zigbee).
- DIKAIAKOS, G. P. M. et al. Cloud computing - distributed internet computing for it and scientific research. *Internet Computing*, IEEE, v. 13, p. 10–13, 2009.

- FERNANDES SAMUEL G MATOS, e. a. M. A. A framework for wireless sensor networks management for precision viticulture and agriculture based on ieee 1452 standart. *Computer and Electronics in Agriculture*, Elsevier, p. 19–30, 2013.
- GILL, A. C. *Como elaborar um projeto de pesquisa*. [S.l.]: FESBH, 1991.
- HOY, T. *Basic Definitions: Web 1.0, Web 2.0, Web 3.0*. 2007. Disponível em: <<http://www.practicalecommerce.com/articles/464-Basic-Definitions-Web-1-0-Web-2-0-Web-3-0>>.
- HUANG, Y.; LI, G. Descriptive models for internet of things. *Intelligent Control and Information Processing (ICICIP), 2010 International Conference*, IEEE, p. 483–486, 2010.
- IBM. *Internet of Things on Bluemix*. 2015. Disponível em: <<https://www.ibm.com/cloud-computing/bluemix/solutions/iot/>>.
- KOMAROV, M. M.; NEMOVA, M. D. Emerging of new service-oriented approach based on the internet of services and internet of things. *e-Business Engineering (ICEBE), 2013 IEEE 10th International Conference*, IEEE, p. 429–434, 2013.
- LI SHAOLIANG PENG, e. a. S. Income: Pratical land monitoring in precision agriculture with sensor networks. *Computer Communications*, Elsevier, p. 459–467, 2012.
- MENASCE, P. N. A. D. Understanding cloud computing: Experimentation and capacity planning. *Computer Measurement Group Conference*, 2009.
- MITRA, S. *Retailers: Embrace Web 3.0*. 2009. Disponível em: <<http://www.forbes.com/2009/12/03/web3-blue-nile-intelligent-technology-retail.html>>.
- NGUYEN, A. G. V. A vision of a future iot architecture supporting messaging, storage, and computation. *International Journal of Future Computer and Communication, IJFCC*, v. 3, p. 405–410, 2014.
- NIKOLIDAKIS, e. a. S. A. Energy efficient automated control of irrigation in agriculture by using sensor networks. *Computers and Electronics in Agriculture*, Elsevier, p. 154–163, 2013.
- TANENBAUM, A. S. *Distributed Systems: Principles and Paradigms*. [S.l.: s.n.], 2002.
- UNION, I. T. *The Internet of Things*. [S.l.], 2005.

UPTON, E. *RASPBERRY PI 2 ON SALE NOW AT 35.* 2015. Disponível em: <<https://www.raspberrypi.org/blog/raspberry-pi-2-on-sale/>>.

W3C. *Semantic Sensor Network XG Final Report.* 2011. Disponível em: <<http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/>>.

ZHOU, e. a. J. Cloudthings: A common architecture for integrating the internet of things with cloud computing. *Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE International Conference*, IEEE, p. 651–657, 2013.