



**UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO**

PATRIC D. VENTURINI

**ANÁLISE E ESTUDO DE CASO EM INTERNET DAS COISAS
BASEADO NO PROJETO KAA**

**CHAPECÓ
2018**

PATRIC D. VENTURINI

**ANÁLISE E ESTUDO DE CASO EM INTERNET DAS COISAS
BASEADO NO PROJETO KAA**

Trabalho de conclusão de curso de graduação
apresentado como requisito para obtenção do
grau de Bacharel em Ciência da Computação da
Universidade Federal da Fronteira Sul.

Orientador: Prof. Dr. Marco Aurélio Spohn

CHAPECÓ

2018

D. Venturini, Patric

Análise e Estudo de Caso em Internet das Coisas baseado no projeto Kaa / por Patric D. Venturini. – 2018.

35 f.: il.; 30 cm.

Orientador: Marco Aurélio Spohn

Monografia (Graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, RS, 2018.

1. Internet das Coisas. 2. IoT. 3. Integração com Kaa. 4. Kaa Platform. 5. Solução de fim a fim. I. Aurélio Spohn, Marco. II. Título.

© 2018

Todos os direitos autorais reservados a Patric D. Venturini. A reprodução de partes ou do todo deste trabalho só poderá ser feita mediante a citação da fonte.

E-mail: patric.venturini@hotmail.com

PATRIC D. VENTURINI

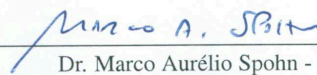
**ANÁLISE E ESTUDO DE CASO EM INTERNET DAS COISAS
BASEADO NO PROJETO KAA**

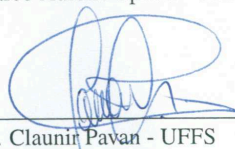
Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.


Orientador: Prof. Dr. Marco Aurélio Spohn

Este trabalho de conclusão de curso foi defendido e aprovado pela banca em: 09/07/2018

BANCA EXAMINADORA:


Dr. Marco Aurélio Spohn - UFFS


Dr. Claunir Pavan - UFFS


Dr. Emílio Wuerges - UFFS

AGRADECIMENTOS

Há muitas pessoas que devem ser devidamente citadas pelo auxílio emocional ou prático que fizeram para esse trabalho ser concluído. Em primeiro lugar, agradeço ao meu orientador e professor para a vida, Marco, por sempre ter questionamentos que reforçassem as minhas certezas e certezas para os meus questionamentos, sem ele eu nunca teria seguido nesse ramo de estudo. Em seguida, meus agradecimentos à Débora por suportar as conversas sobre uma área alienígena à dela e ainda assim ficar do meu lado. Agradeço também ao Trichez, por todas as conversas de madrugada que me ajudaram a consolidar pensamentos e rever alguns pontos de vista. Agradeço de coração ao URSM pelo suporte constante ao meu cuidado físico e pelas cobranças. Por último, mas não menos importante, agradeço a Michel, Otávio, Hollweg, Salko, Gênaro, Carol e Ronai pela motivação e confiança sempre presentes.

RESUMO

A Internet das Coisas é uma das principais áreas de estudo atualmente, que traz novos paradigmas a serem trespassados, novas tecnologias a serem estudadas e novos desafios a serem superados.

Com a proposta de trazer uma solução em Internet das Coisas de fim a fim, o Kaa apresenta um conjunto de *middlewares* para facilitar o seu uso, além de um servidor já configurado e pré-configurações para as “coisas”. Esse trabalho se propôs, então, a avaliar a plataforma através de um Estudo de Caso, que utilizará um medidor de temperatura e umidade ambiente como sensor e um módulo de comunicação wi-fi, com o objetivo de analisar a situação atual da plataforma, pontos onde ela pode ser melhorada e descrever sobre a possibilidade de uso da mesma em um caso real.

Palavras-chave: Internet das Coisas. IoT. Integração com Kaa. Kaa Platform. Solução de fim a fim.

ABSTRACT

The Internet of Things is one of the main areas of study nowadays, bringing new paradigms and technologies to be learned and new challenges to be surpassed.

With the purpose of be an end-to-end IoT solution, the Kaa project introduces a set of middlewares to ease the platform's usability, a ready server to use and some pre-configurations to the "things". So, this present study aims to evaluate the platform through a case study, which will count with a temperature and humidity sensor and a wi-fi communication module, with the objective of analyse the actual plataform's situation, key points where it can be improved and describe the possibility of use in a real case scenario.

Keywords: Internet of Things. IoT. Kaa. Kaa integration. End-to-end solution.

LISTA DE FIGURAS

Figura 2.1 – A arquitetura do Kaa. [9]	14
Figura 2.2 – Exemplo de uso dos <i>nodos</i> por usuários inquilinos. [9]	15
Figura 5.1 – Código usado para testar o DHT	21
Figura 5.2 – Tela inicial do Kaa Sandbox	23
Figura 5.3 – Tela de Administrador para criar nova aplicação	23
Figura 5.4 – Collection usada para representar os dados de temperatura e umidade nos logs	24
Figura 5.5 – Finalizado processo de adicionar um novo esquema	25
Figura 5.6 – Collection usada para representar os dados de temperatura e umidade nas configurações	26
Figura 5.7 – Configuração padrão das portas do Kaa.....	27
Figura 5.8 – Exemplo de configuração do CMakeLists do Kaa	28
Figura 5.9 – Código de exemplo do uso do SDK do Kaa em Linux	29

SUMÁRIO

1 INTRODUÇÃO	10
1.1 Objetivos	11
1.1.1 Objetivo geral	11
1.1.2 Objetivos específicos	11
1.2 Justificativa	11
1.3 Delimitação da Pesquisa	12
1.4 Estrutura do Trabalho	12
2 REFERENCIAL TEÓRICO E PRÁTICO	13
2.1 O Projeto Kaa	13
2.2 A Arquitetura do Kaa	13
2.2.1 O Servidor	14
2.2.2 Extensões do Kaa	16
2.2.3 Os SDK Endpoints	17
3 METODOLOGIA	18
3.1 Metodologia de pesquisa	18
4 TRABALHOS RELACIONADOS	19
5 DESENVOLVIMENTO	20
5.1 Caso de Uso	20
5.2 A Concepção da Coisa	20
5.2.1 DHT 11	20
5.2.2 Módulo Wi-Fi ESP8266-01	21
5.2.3 Integração dos módulos	22
5.2.4 Configurando o Kaa Localmente	22
5.2.5 Configurando o Kaa na Nuvem	26
5.2.6 Sobre a Segurança - Multi-Tenancy	27
5.2.6.1 SDK e Linux	28
5.2.7 Teste de Carga na Aplicação	29
6 RESULTADOS	31
6.1 Integrando de Ponta a Ponta	31
6.2 Melhorias desejáveis	32
6.3 Implantação em sistemas reais	32
7 CONCLUSÃO	33
7.1 Trabalhos futuros	33
REFERÊNCIAS	34

1 INTRODUÇÃO

A internet trouxe mudanças significativas para a forma como nos comunicamos e vemos o mundo. Muitos itens do dia a dia foram adaptados para o digital e influenciaram tanto a economia dos países quanto a vida das pessoas. Os computadores pessoais começaram a virar realidade e pouco tempo depois surgiram os celulares, igualmente conectados à internet graças aos avanços das redes sem-fio. Hoje em dia essas tecnologias evoluíram tanto que dão espaço para uma nova etapa: a Internet das Coisas.

O termo “Internet das Coisas” surgiu recentemente, em meados dos anos 2000, como uma ideia ainda pouquíssimo explorada de conectar coisas na rede. Segundo [19], a Internet das Coisas são objetos do mundo físico equipados com sensores eletrônicos capazes de perceber o mundo à sua volta e se comunicar com outros dispositivos. Objetos comuns mesmo, como uma xícara ou uma caneta, uma geladeira ou um forno, dotados da capacidade de se comunicar com outros aparelhos, dizendo sua temperatura interna, avisando mal funcionamento ou se o suco da geladeira está gelado. Não somente isso, é uma tecnologia que nos permite, por exemplo, saber o estado exato das condições do solo a que cada planta de uma horta está submetida, desde que a mesma possua sensores que consigam medir esses valores.

Essa é chamada de “a quarta onda da internet” pois apresenta novos paradigmas que precisam ser ultrapassados e que mudarão outra vez a forma como enxergamos a internet. Entretanto, para darmos esse passo, como explica [14], precisamos progredir ainda mais as nossas redes para suportarem os bilhões de dispositivos conectados, a forma de identificarmos essas “coisas” com RFIDs, os sensores a serem utilizados, a segurança envolvendo sua utilização/conexão com a rede, a privacidade dos usuários, o consumo de energia, etc.

Como é uma área ainda em crescimento e com muito a ser explorado, não existe nenhum sistema que seja um referencial a ser seguido, todas as plataformas estão evoluindo rapidamente para tentar conquistar esse posto e entregar a melhor solução de ponta a ponta. Motivado por estudar essas tecnologias emergentes, compará-las e ajudá-las a progredir, esse trabalho tem por objetivo analisar a plataforma Kaa e como a solução entregue por ela pode contribuir para o futuro desse mercado.

O Kaa, como descrito em [9], é uma plataforma de código aberto que traz uma solução de ponta a ponta para a área de Internet das Coisas. Ela apresenta um conjunto de implementações pré-programadas que cobrem a grande maioria dos casos de uso comuns da área a fim de

acelerar o processo de utilização do sistema, além de contar com um sistema de hierarquia de usuários para controle de privacidade e envio de dados e um serviço de servidor para rodar o *back-end* da aplicação.

Com o objetivo de analisar o Kaa e suas funcionalidades, será aplicado um estudo de caso em cima da plataforma através de um Arduino Uno R3 equipado com um sensor de temperatura e umidade ambiente, o DHT 11, e um módulo de conexão com a internet, ESP8266-01.

1.1 Objetivos

1.1.1 Objetivo geral

Avaliar a plataforma disponibilizada pelo projeto Kaa, testando desde os componentes de ponta, onde fica a “coisa”, até os *middlewares* e o servidor, construindo em cima dele um estudo de caso.

1.1.2 Objetivos específicos

- Estudar e utilizar as funcionalidades já desenvolvidas pelo projeto Kaa, encontrando e compreendendo suas limitações físicas, de rede, de carga e de segurança.
- Aplicar em cima da ferramenta um leitor de temperatura ambiente para estudo de caso.
- Amostrar e descrever os testes e resultados obtidos.

1.2 Justificativa

Considerando os avanços das redes sem fio e o crescimento acelerado da área de Internet das Coisas, explorar as ferramentas que estão sendo desenvolvidas, estudar suas tecnologias e contribuir em sua construção são elementos indispensáveis para que continue havendo evoluções nesse ramo emergente da computação.

Atualmente, apesar das contantes progressões da tecnologia, ainda encontramos desafios que precisam ser ultrapassados a fim de que possamos utilizar a Internet das Coisas em todo seu potencial. Há limitações para o tamanho dos dispositivos, limitações de segurança devido às brechas que as “coisas” podem apresentar e mesmo limitações físicas das redes por causa da quantidade de conexões simultâneas que serão necessárias. A própria velocidade de nossas

conexões sem-fio e 4G precisam progredir para que essa tecnologia venha a ser difundida e utilizada de fato.

Através do projeto Kaa, com um protótipo de leitor de temperatura ambiente usando Arduino, pretende-se amostrar dados e relatórios de desempenho da plataforma, levando em consideração a base e ferramentas oferecidas pela mesma, sua performance de segurança, de gestão e de carga, assim como transcrevendo suas limitações e possibilidades de melhorias futuras.

1.3 Delimitação da Pesquisa

Como delimitação desse trabalho, foi utilizado um estudo de caso que possibilita avaliar como o Kaa e suas funcionalidades se comportam, quais as diferenças entre um servidor hospedado localmente e outro na nuvem, a viabilidade para uma implantação real e levantar as possíveis melhorias ou problemas que precisam ser corrigidos para um melhor aproveitamento da plataforma.

1.4 Estrutura do Trabalho

Esse trabalho está organizado em seis partes, começando pela 1 onde é feita a introdução do mesmo, tendo seu conceito apresentado juntamente com os objetivos, justificativa e uma breve delimitação. A parte 2 explica o referencial teórico e prático, descrevendo detalhadamente os componentes que estarão presentes no trabalho, desde a "coisa", feita em Arduino, até uma explicação aprofundada do Kaa, que faz o papel de *middleware*, e seus servidores, que serão baseados em Ubuntu. Após, na parte 3, é apresentada a metodologia utilizada para a pesquisa. Depois vem a parte de 5, contando com uma descrição aprofundada do passo a passo do andamento do projeto, seguida pela seção de 6, onde estarão presentes os resultados da pesquisa, e, por fim, uma 7 seguida da parte 4, onde são citados os artigos que contribuíram para a idealização do trabalho.

2 REFERENCIAL TEÓRICO E PRÁTICO

Segundo [19], a Internet das Coisas é composta de três elementos principais: As ferramentas, que nesse caso serão os objetos; a rede, que conectará todos as coisas; e por fim os sistemas que usarão os dados dos objetos.

Esmiuçando cada uma dessas partes, temos primeiro a ferramenta. Ela será um objeto inteligente e precisa de alguns requisitos para ser válida, afinal, precisa *pensar*, *sentir* e conseguir se *comunicar* sozinha. Para fazer esse papel, será usado um *Arduino Uno R3* devido à facilidade de acesso à documentação, quantidade de conteúdo relacionado e principalmente por ser uma plataforma de hardware livre, que facilita a obtenção e manutenção do equipamento.

Na segunda parte, fazendo a estrutura de ligação de uma ponta a outra, temos a rede. Nesse trabalho, será utilizado a plataforma de *middlewares* Kaa, que oferece uma solução fim a fim, trabalhando junto com a Arduino e o sistema final para trazer uma solução eficiente.

Por fim, precisamos da aplicação que utilizará os dados do estudo de caso e seus resultados. Como o foco desse trabalho é o estudo aprofundado da plataforma Kaa, será utilizado um caso de leitura de temperatura ambiente, que poderá futuramente ser evoluído para algo mais complexo e que servirá para explorar o que a ferramenta tem a oferecer.

2.1 O Projeto Kaa

O Kaa, segundo [9], é uma plataforma de *middlewares* altamente flexível, multiuso e 100% *open-source* cujo objetivo é servir de base para implementações ponta a ponta de IoT, aplicações conectadas e produtos inteligentes. Ela oferece um conjunto de recursos para IoT de nível comercial que podem ser facilmente conectados e utilizados para implementar a grande maioria dos casos de uso do mercado. Os recursos da plataforma incluem gerenciamento de dispositivos, coleta de dados, gerenciamento de configuração, mensagens, uma parte de gestão com níveis de hierarquia de usuário, etc.

2.2 A Arquitetura do Kaa

A plataforma de IoT Kaa consiste basicamente do servidor do Kaa, das extensões do Kaa e dos SDK endpoints.

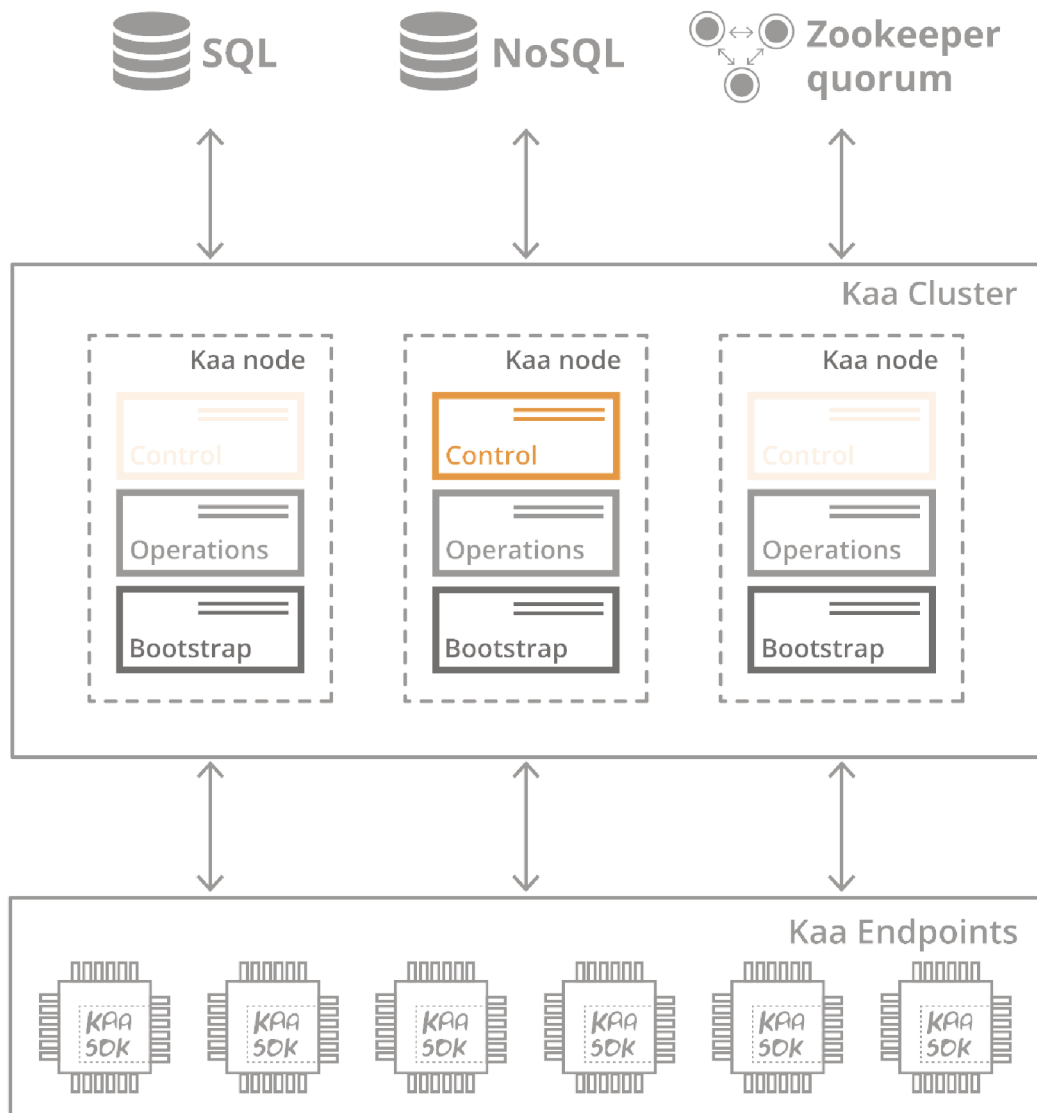


Figura 2.1: A arquitetura do Kaa. [9]

2.2.1 O Servidor

Começando pelo servidor do Kaa, chamado de *Kaa node* na figura 2.1, temos o *back-end* da aplicação. Nesses nodos ficam as regras de negócio, o gerenciamento dos inquilinos (*tenants*), aplicações, usuários e dispositivos, assim como a interface integrada e as funções administrativas. Um nodo ou um conjunto desses nodos interconectados através do *Apache ZooKeeper* é chamado de *Kaa Cluster*. Mesmo estando interconectados, cada um deles roda uma instância própria do Kaa independente das outras. Como oferecido pelo Kaa, serão utilizados tanto um servidor local baseado em Ubuntu 14.04 64-bits quanto um servidor externo, hospedado na Amazon, para servirem de *clusters*.

Nesses *nodos*, fazendo o papel de módulo de configuração, temos o *Kaa Sandbox*. O *Sandbox* é um ambiente virtual pré-configurado pensado para quem quer usar uma instância privada da plataforma Kaa para propósitos de estudo, desenvolvimento ou prova de conceito. Através dele, conseguimos analisar, salvar e utilizar os dados adquiridos pelas “coisas”.

Também são neles que o gerenciamento dos papéis é feito, distribuídos em 4 níveis de hierarquia: Administrador Kaa, Inquilino Administrador, Inquilino Desenvolvedor e o Inquilino. Considerando que será usado uma hierarquia de multi-inquilinos, segundo [11], temos que os inquilinos são um grupo de usuários que compartilham um acesso comum com privilégios específicos. Começando então pelo nível mais alto da hierarquia, temos o Administrador Kaa, com ele é possível criar, editar e deletar Inquilinos Administradores. Como o Kaa dá suporte a uma arquitetura de multi-inquilinos, o Administrador Kaa pode criar e gerenciar escopos separados para cada instância de um inquilino. Abaixo dele, os Inquilinos Administradores gerenciam as aplicações, usuários e famílias de classes de eventos. Com um usuário Inquilino Administrador também é possível criar novas aplicações. Após temos os Inquilinos Desenvolvedores, que podem criar novos SDKs baseados nas necessidades dos clientes, podendo manter a estrutura de dados dessas aplicações, criar grupos de “coisas”, controlar os processos de notificações, etc. Inquilinos Desenvolvedores só podem trabalhar com aplicações criadas pelos respectivos Inquilinos Administradores. Por fim, temos os inquilinos propriamente ditos, que apenas usam a aplicação, como representado na figura 2.2.

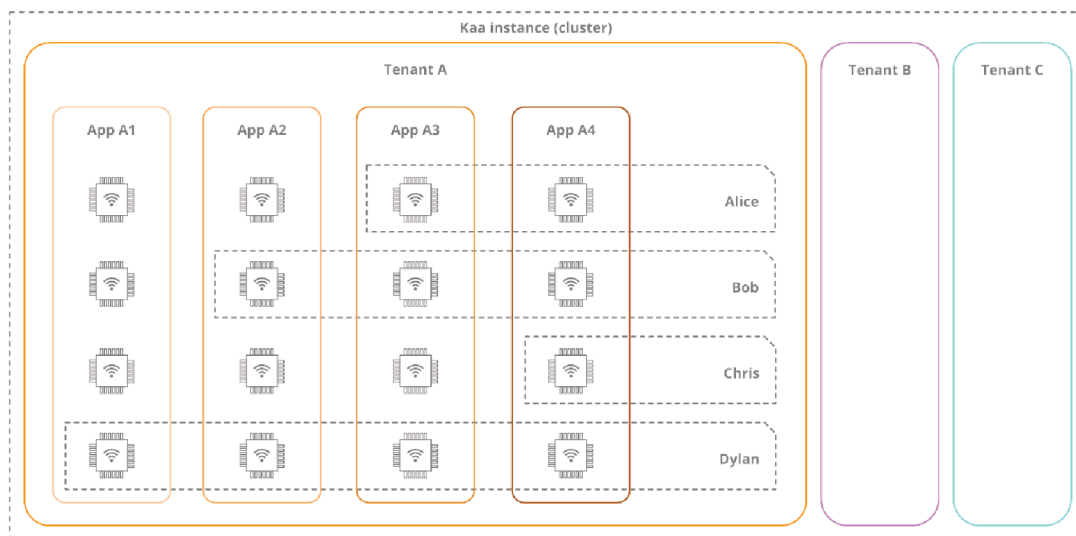


Figura 2.2: Exemplo de uso dos *nodos* por usuários inquilinos. [9]

Explorando ainda mais a fundo os nodos, temos que eles são compostos por três serviços

fundamentais: Controle, Operações e *Bootstrap*.

O serviço de Controle do Kaa gerencia todas as informações do sistema, processa as demandas das chamadas das APIs tanto da web quanto dos sistemas integrados externos e notifica o sistema de Operações. Também é papel dele manter uma lista atualizada dos serviços de Operações disponíveis através das informações contínuas recebidas do ZooKeeper. Adicionalmente, o serviço de Controle roda os componentes administrativos, trabalhando com todas as hierarquias acima citadas. A fins de disponibilidade do serviço, o Kaa recomenda que seus *clusters* tenham pelo menos dois nodos com serviços de Controle habilitados, para que quando um entre em *standby* o outro assuma no lugar.

Já o papel principal do serviço de Operações é se comunicar com múltiplas “coisas” paralelamente. Sua função é processar as requisições feitas pelas “coisas” e mandar os dados de volta para elas. Para fins de escalonamento horizontal, o Kaa recomenda que cada nodo de um *cluster* esteja com o serviço de Operações habilitado. Nesse cenário, todos os serviços de Operações estarão funcionando ao mesmo tempo, podendo fazer um balanço de carga entre os nodos e melhorando a qualidade da comunicação de uma ponta a outra da aplicação.

Por fim, temos o serviço de *Bootstrap*. Ele é quem envia as informações para a “coisa” sobre os parâmetros de conexão com os serviços de Operações. Dependendo da pilha de protocolos configurados, os parâmetros de conexão podem incluir endereço IP, porta TCP, credenciais de segurança, etc. Os SDKs do Kaa contêm uma lista pré-gerada dos serviços Bootstrap disponíveis no *cluster* do Kaa, então eles consultam os serviços Bootstrap desta lista para receber os parâmetros de conexão para os serviços de Operações atualmente disponíveis. Os serviços do Bootstrap mantêm suas listas do serviço de Operações disponíveis através do ZooKeeper.

2.2.2 Extensões do Kaa

As ditas extensões do Kaa são módulos de softwares independentes que ajudam a melhorar o desempenho da plataforma. Como mostrado no figura 2.1, são eles os bancos de dados SQL e NoSQL e o serviço Apache Zookeeper.

Começando pelo Apache Zookeeper, a função do mesmo é coordenar os nodos dos *clusters*. Cada nodo manda informações contínuas sobre seus parâmetros de conexão, serviços habilitados e a carga atual dos serviços. Outros nodos do Kaa usam essas informações para conhecer e se conectar com os nodos irmãos. O serviço de Controle ativo usa essas informações sobre os serviços *Bootstraps* disponíveis e seus parâmetros de conexão durante a geração dos

SDKs.

O banco de dados SQL é usado para armazenar os inquilinos, aplicações, grupos de “coisas” e outros metadados que não crescem ao mesmo passo que o número de “coisas” conectadas. Uma maior disponibilidade dos *clusters* do Kaa é alcançada quando é implantado o SQL em modo HA (High Availability). No momento, o Kaa dá suporte para MariaDB e PostgreSQL.

O banco de dados NoSQL é usado para armazenar os dados das “coisas” que crescem linearmente conforme o número delas aumenta. Os nodos do banco de dados NoSQL podem estar co-locados com os nodos do Kaa na mesma máquina física ou virtual e deve ser implantada em modo HA para que esteja sempre disponível para o sistema. Atualmente, o Kaa dá suporte para o Apache Cassandra e MongoDB como banco de dados NoSQL.

2.2.3 Os SDK Endpoints

No *SDK Endpoint* fica localizada a “Coisa”. O Kaa já oferece um conjunto de pré-configurações prontas para diversos modelos de aplicações e em diversas linguagens. Para o estudo de caso deste trabalho, o escolhido foi o Arduino.

O Arduino é uma plataforma de prototipagem eletrônica de hardware livre cujo objetivo é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de se usar tanto por artistas quanto amadores.

A mesma foi escolhida para fazer parte do projeto graças à vasta documentação encontrada na internet sobre praticamente qualquer exemplo de aplicação, à facilidade de aquisição e troca das peças e também por já ser uma das plataformas suportadas pelo Kaa para servir de “coisa”.

Como base, para fazer o leitor de temperatura ambiente, será utilizado o Arduino Uno R3 juntamente com o módulo ESP8266-01 Serial Wi-Fi Wireless Module + Adapter for Arduino para a conexão com a internet e um sensor de umidade e temperatura DHT11, que é o mais comum do mercado e suficiente para a prova de conceito, cobrindo temperaturas de 0 a 50 graus e medindo umidade com uma taxa de erro aceitável.

3 METODOLOGIA

Nesse capítulo serão descritas as metodologias utilizadas para alcançar os objetivos propostos e os resultados esperados.

3.1 Metodologia de pesquisa

A pesquisa consiste na criação de um Estudo de Caso passível de evoluções futuras, onde será possível avaliar o desempenho, a arquitetura e os componentes já presentes no Kaa (conforme ilustrados na figura 2.1).

Para isso, as seguintes etapas foram seguidas:

1. Desenvolvimento do Estudo de Caso. Nessa etapa, utilizando a estrutura fornecida pelo Kaa, foi configurado e adaptado um leitor de temperatura ambiente para que o mesmo mantenha o servidor sempre a par das variações de temperatura no local.
2. Durante o processo de configuração e instalação do dispositivo, foi identificado os problemas existentes na plataforma e suas possíveis melhorias, juntamente com uma análise sobre a facilidade de instalação, aprendizado e obtenção de informações referentes ao uso do Kaa, seja por documentação da própria plataforma ou fóruns na Internet.
3. A partir da obtenção dos dados através da “coisa”, será analisada a tolerância a carga de conexões do Kaa, o mínimo necessário para o funcionamento, o tempo de processamento das informações, as diferenças de desempenho entre um servidor hospedado localmente e outro na nuvem e o consumo de banda das mensagens.

4 TRABALHOS RELACIONADOS

Por estar utilizando uma plataforma ainda em versão beta, não há muitos trabalhos em que se basear quanto a implementação e desenvolvimento no Kaa em si. Porém, as ideias quanto as métricas e os desafios da área vieram de muitos *surveys*, da leitura do [8] e da leitura da documentação da própria plataforma.

Dentre todos, o artigo mais importante foi o [8], nesse artigo os autores citam como as plataformas de Internet das Coisas têm se desenvolvido e como elas têm buscado conquistar o mercado, algumas se vendendo como PaaS, *Platform-as-a-Service*, outras como SaaS, *Software-as-a-Service*, e quais são seus pontos fortes e suas diferenças. É importante citar que eles reforçam a questão da segurança e privacidade dadas pelas plataformas, suas tecnologias e como elas se posicionam no mercado.

O segundo trabalho que mais influenciou no desenvolvimento foi o [19]. Esse, apesar de ser um *survey*, é bastante recente e, além de construir todo o embasamento sobre o que é a área de Internet das Coisas, também traz uma visão atual da área e quais são seus prós e contras, onde pode ser aplicada e quais são seus desafios.

5 DESENVOLVIMENTO

Nesse capítulo serão abordados os pontos principais do desenvolvimento do Projeto. Começando pelo desenvolvimento do estudo de caso, seguido pela integração com o Kaa e finalizando com os testes de segurança e desempenho da plataforma.

5.1 Caso de Uso

O caso de uso, desenvolvido com Arduino Uno R3, conta basicamente com o Arduino, o módulo de medição de temperatura DHT11 e um módulo Wi-Fi ESP8266-01, que é suportado pela plataforma Kaa, consistindo no papel de "coisa".

O restante do sistema será construído utilizando as ferramentas já disponibilizadas pelo Kaa para desenvolvimento. O Kaa, por ter como proposta englobar todo o projeto de ponta a ponta, entrega desde uma parte para a coisa até a comunicação e o *Backend* da aplicação. Serão nesses módulos de comunicação onde os testes serão aplicados.

5.2 A Concepção da Coisa

Para um melhor entendimento da concepção, as etapas serão apresentadas de forma modular, começando com o leitor de temperatura, depois o módulo WiFi, seguindo para a integração dos módulos, integração com o Kaa e algumas falhas apresentadas pelo Kaa no processo.

5.2.1 DHT 11

O DHT 11 é o componente mais simples da família DHT, ele é um componente barato, que não tem uma amplitude muito grande de medição de temperatura e umidade, mas ainda assim suficiente. Ele consegue medir de 0.0 °C a 50.0 °C, com uma imprecisão de 2.0 °C, e medir umidade de 20 a 90%, com imprecisão de 5,0%.

Como nesse caso será apenas para testes fechados, essa imprecisão do módulo não gera efeitos colaterais negativos, mas caso seja necessário o uso para um fim mais sensível, é ideal o uso do DHT 33 ou DHT 44.

Sobre a implementação, o módulo DHT 11 conta com 4 pinos de conexão, sendo eles, da esquerda para a direita, o VCC, o de dados, um neutro e o GND (*ground*). O VCC, que é o de alimentação do módulo, é ligado em 5V no Arduino, o de dados é ligado á qualquer

entrada analógica, no nosso caso a A1, e o GND é conectado ao GND da placa do Arduino. Para suprimir um pouco mais da imprecisão, foi colocado um resistor entre o VCC e os dados.

Como o DHT 11 precisa ser calibrado antes de usar, foi usado uma biblioteca que já trata os dados, distribuído pela Adafruit Inc. Não é necessária muita configuração a mais para usar o módulo, só precisa ser respeitado os 9600 *bauds* do módulo e o *delay* de leitura de em torno de 1 segundo por leitura.

```
#include <DHT.h>
#include <DHT_U.h>

#define DEBUG true
#define DHTPIN A1 // pino que estamos conectado
#define DHTTYPE DHT11 // DHT 11

// Conecte pino 1 do sensor (esquerda) ao +5V
// Conecte pino 2 do sensor ao pino de dados definido em seu Arduino
// Conecte pino 4 do sensor ao GND
// Conecte o resistor de 10K entre pin 2 (dados)
// e ao pino 1 (VCC) do sensor
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  // A leitura da temperatura e umidade pode levar 250ms!
  // O atraso do sensor pode chegar a 2 segundos.
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  // testa se retorno é valido, caso contrário algo está errado.
  if (isnan(t) || isnan(h)) {
    Serial.println("Failed to read from DHT");
  } else {
    Serial.print("Umidade: ");
    Serial.print(h);
    Serial.print(" %t");
    Serial.print("Temperatura: ");
    Serial.print(t);
    Serial.println(" *C");
  }
}
```

Figura 5.1: Código usado para testar o DHT

5.2.2 Módulo Wi-Fi ESP8266-01

O módulo Wi-Fi é um caso diferente do DHT 11 e com muito mais dependências. Ele foi escolhido por ser um dos mais baratos da categoria e um dos dois modelos suportados pela plataforma.

O ESP8266-01 possui oito pinos de conexão e, apesar de existir um módulo especial para fazer a ligação dele na protoboard, foi usado uma ligação direta com a peça, que também

é possível.

Um detalhe importante sobre a placa é que se faz necessário o uso de uma fonte de alimentação externa ao Arduino para que ela funcione perfeitamente. Nesse caso foi usado o regulador de tensão LM117, que é o mais indicado para usar com a placa.

Sobre os pinos, dispostos em dois trilhos, dois são para comunicação serial dos dados com o Arduino, RX e TX, que são respectivamente conectados ao TX e RX do Arduino, um do lado oposto ao outro no módulo. Temos mais o de alimentação, de 3.3V e o GND, também um do lado oposto ao do outro. Dos quatro pinos centrais, os dois mais internos são GPIO (*General Purpose Input/Output*), que não serão usados no Estudo de Caso, e sobram os dois mais externos, um de RST (*reset*), que também não será utilizado, e o CH_PD (*Chip Enable*) que também precisa receber 3.3V de alimentação. É indicado também colocar um capacitor antes dos cabos de alimentação, para estabilizar a corrente e diminuir ruídos de comunicação.

Para testar se a placa estava funcionando, foi mandado o comando "AT" por comunicação Serial do Arduino, que responderá "OK" se a placa estiver funcionando corretamente.

5.2.3 Integração dos módulos

Com o terreno preparado, faz-se necessário finalizar a integração entre o Sensor DHT11 e o Módulo Wi-Fi ESP8266-01, encerrando a configuração do *hardware* do "Objeto" presente no Estudo de Caso.

É necessário entender, então, a forma de comunicação entre o sensor e o módulo Wi-Fi. O sensor trabalha na mesma taxa de *bauds* que o iniciado pelo Arduino e o módulo Wi-Fi, por padrão, trabalha em 115200 *bauds*. Para equalizar os dois, basta inicializar o Serial do Arduino em 115200. Outro ponto importante será a forma de comunicação do Arduino em si com o módulo Wi-Fi, que no caso pode ser por protocolos UART/USART ou Serial TX/DX.

5.2.4 Configurando o Kaa Localmente

O primeiro passo para começar a trabalhar com o Kaa é fazer o cadastro na plataforma e baixar a imagem disponibilizada em seu site para uso local. Para conseguir avaliar a facilidade de instalação e sua documentação, foi seguido a indicação dos próprios criadores e instalado a imagem em uma máquina virtual no *Oracle VM Virtual Box*.

Mantendo a máquina com todos as configurações recomendadas de instalação, a mesma iniciou sem maiores problemas. Com a máquina virtual configurada, a plataforma ficou local-

mente hospedada no endereço: "127.0.0.1:9080/sandbox".



Figura 5.2: Tela inicial do Kaa Sandbox

Então, entrando na sessão de administrador, "*Administrator UI*", primeiro foi utilizada a conta de administrador padrão, Usuário: "admin", Senha: "admin123". Em uma aplicação real, considerando o fator humano envolvido na segurança[13], é necessário alterar essa conta, tanto usuário quanto senha, pois é ela que controla todas as aplicações do sistema. Com acesso à essa conta, é possível deletar qualquer aplicação do sistema, modificar os usuários, etc; mas mesmo com acesso a ela, não se consegue ver as aplicações com mais detalhes, seguindo a hierarquia de *Tenants* adotadas pelo Kaa.

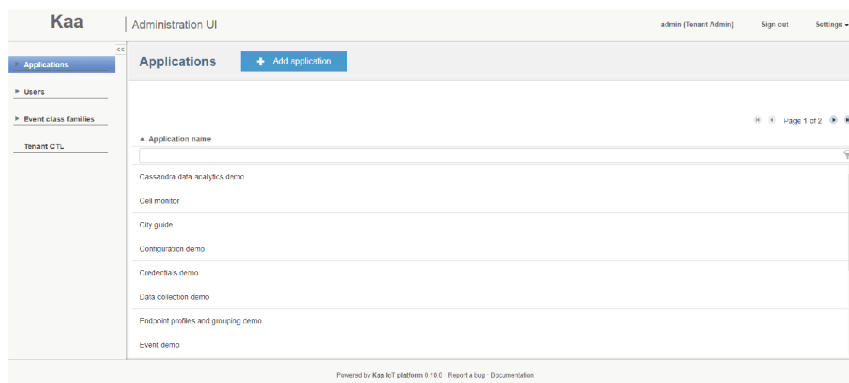


Figura 5.3: Tela de Administrador para criar nova aplicação

Com todos esses detalhes em mente, foi criada uma aplicação, configurada como *trustful*, de nome *Temperature and Humidity Sensor*, encerrando o uso do usuário administrador. Para poder usar o servidor do Kaa ainda faltam dois passos: a criação das *collections* a serem utilizadas e a geração do SDK da aplicação.

Assim, o primeiro passo foi criar uma *collection* seguindo os modelos disponibilizados pelo Kaa e que abrangesse os três campos necessários para a aplicação: Temperatura, umidade

e horário. Nenhum dos campos será opcional, para que os dados coletados tenham validade. Considerando que o DHT 11 também tem uma margem de erro consideravelmente grande, mandar apenas os inteiros dos valores já é suficiente para uma aplicação teste. Caso a aplicação necessite de mais precisão, além de ajustar os campos para receberem valores do tipo *float*, é necessário comprar um componente mais preciso em suas medições, como o DHT 22, DHT 33 ou DHT 44.

```
{
  "type" : "record",
  "name" : "TemperatureAndHumidityRecord",
  "namespace" : "org.kaaproject.kaa.schema.sensor",
  "version" : 1,
  "dependencies" : [ ],
  "displayName" : "Temperature And Humidity Record",
  "description" : "API to record sensor's temperature and humidity",
  "fields" : [
    {
      "name" : "temperature",
      "type" : "int"
    },
    {
      "name" : "humidity",
      "type" : "int"
    },
    {
      "name" : "timeStamp",
      "type" : "long"
    }
  ]
}
```

Figura 5.4: Collection usada para representar os dados de temperatura e umidade nos logs

Com a *collection* criada, é necessário entrar na área administrativa como *Tenant Developer*, que tem por papel cuidar dos detalhes relativos ao desenvolvimento das aplicações em Kaa. É o *Tenant Developer* que vai cuidar das *collections* e dos SDK's da aplicação. Por padrão, a conta de acesso tem o usuário "devuser" e a senha "devuser123". Da mesma forma que com a conta de administrador, é preciso ter atenção com essa conta caso ela seja usada em produção. Mesmo que ela possua menos poderes, ela ainda pode causar um impacto muito grande porque todos os dados de uma aplicação podem ser comprometidos, tanto em integridade quanto em vazamento de informações, além de que essa é a conta mais importante tratando do desenvolvimento em si.

Isto posto, foi dada sequência à configuração da aplicação. Indo na opção *Temperature and Humidity Sensor* na Aba lateral e expandindo, é necessário selecionar as opções *Schemas > Log*, a fim de configurar o novo esquema de dados da plataforma, que é referente à *collection*

criada previamente. É acessado então a sessão de "Add Schema", onde será colocado um título qualquer para o esquema, e dando sequência com *Create New Type > Upload from file* e então clicar em "Upload". O arquivo da *collection* deve respeitar o padrão Avro. Se tudo estiver correto, clique em "Add" para finalizar o processo.

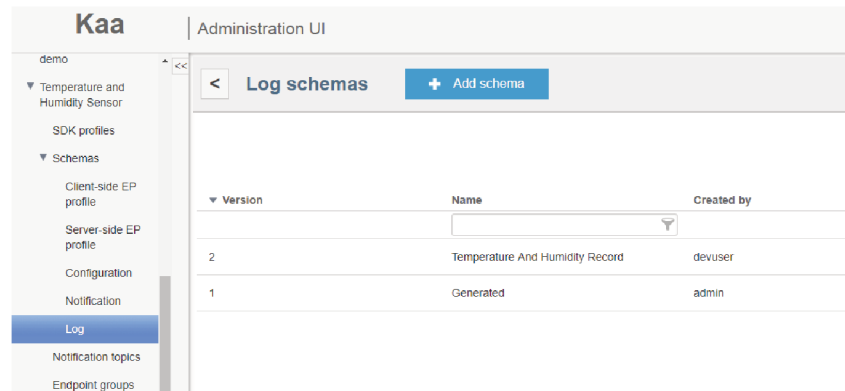


Figura 5.5: Finalizado processo de adicionar um novo esquema

Com o esquema de dados do *Log* criado, um esquema parecido de Configuração tem de ser criado. Então, seguindo nas opções expandidas do *Temperature and Humidity Sensor* na Aba lateral, é necessário selecionar as opções *Schemas > Configuration*, também referente à *collection* criada previamente, porém com alguns dados a mais. É então acessado novamente a sessão de "Add Schema" e repetido o processo de criação do esquema de *Log*.

Um último ponto para que tudo funcione, então, de forma devida, é configurar o *Log Appender* para a aplicação. Para isso, basta clicar na aba *Log Appender* dentro do menu da aplicação *Temperature and Humidity Sensor*. Por enquanto a aplicação não possui nenhum *Log Appender*. Para criar um, basta clicar em "Add Log Appender" no topo da página.

Nessa página, duas coisas são importantes de ressaltar depois de escolher o nome. O primeiro é que todas as opções de "Log Metadata" têm de ser selecionadas, para que o banco seja criado corretamente. A segunda é que é necessário selecionar um banco de dados para trabalhar na opção "Type", sendo sugerido pela plataforma o uso do MongoDB e reforçadamente sugerido para aplicações com foco em escrita de dados pelas características de bancos orientados a documentos quanto a seu desempenho de gravação, principalmente quando se fala em grande volume de dados [15]. O restante das configurações, apesar de poderem ser escolhidos os padrões sugeridas pelo Kaa, é importante atentar às credenciais de autenticação em um Sistema Real.

Para finalizar a configuração do Kaa, falta a geração do SDK, que será integrado à coisa para usufruir da plataforma. Para isso, ainda usando o *Tenant Developer*, é necessário acessar

```

{
  "type" : "record",
  "name" : "TemperatureAndHumidityConfiguration",
  "namespace" : "org.kaaproject.kaa.schema.sensor",
  "version" : 1,
  "dependencies" : [ ],
  "displayName" : "Temperature And Humidity Configuration",
  "description" : "Base configuration sensor's temperature and humidity record",
  "fields" : [
    {
      "name" : "temperature",
      "type" : "int",
      "by_default" : 0
    },
    {
      "name" : "humidity",
      "type" : "int",
      "by_default" : 0
    },
    {
      "name" : "timeStamp",
      "type" : "long",
      "by_default" : 0
    }
  ]
}

```

Figura 5.6: Collection usada para representar os dados de temperatura e umidade nas configurações

a aplicação criada previamente, *Temperature and Humidity Sensor*, e clicar em *Generate SDK* e em seguida em *Add SDK profile*. Nessa página serão escolhidas opções importantes para o funcionamento da aplicação, o que inclui a nova versão do log, criada anteriormente com a *collection* apropriada, representada nesse caso pela versão 2.

Com o perfil do SDK criado, basta clicar na opção abaixo de *Generate SDK* e escolher como alvo o "C", que será usado para configurar no Arduino.

5.2.5 Configurando o Kaa na Nuvem

Para trabalhar com o Kaa na Nuvem o primeiro passo foi criar uma conta na AWS, seguindo o recomendado pela própria plataforma. Foi usado a opção de *deploy* da versão hospedada na AWS *Community*, mas, diferente do recomendado foi selecionado a versão mais leve de servidor "t2.micro". A opção indicada pelo Kaa, de usar um servidor "m3.large" ou maior é para que a plataforma consiga suportar uma grande quantidade de acessos ao servidor, com base em casos reais ou de muito acesso. Não há um motivo real para pegar um servidor tão potente para o caso de testes fechados.

Por fim, antes de acabar a configuração, só foi acertada todas as portas para as padrões do Kaa e feito a publicação.

Protocol	Port	RangeSource
TCP	22	0.0.0.0/0
TCP	8080	0.0.0.0/0
TCP	9999	0.0.0.0/0
TCP	9998	0.0.0.0/0
TCP	9997	0.0.0.0/0
TCP	9889	0.0.0.0/0
TCP	9888	0.0.0.0/0
TCP	9887	0.0.0.0/0
TCP	9080	0.0.0.0/0

Figura 5.7: Configuração padrão das portas do Kaa

No caso das configurações, foram seguidos os mesmos passos da configuração local.

5.2.6 Sobre a Segurança - Multi-Tenancy

O Kaa, assim como parte dos sistemas modernos que funcionam como SaaS(Software as a Service) ou mesmo outras plataformas de IoT [3], adotou o sistema de Multi-Tenancy como forma de estabelecer suas comunicações com o servidor e também blindar seu sistema.

O conceito é que, juntando papéis parecidos em uma hierarquia de inquilinos, problemas quanto a segurança, disposição de funcionalidades e encapsulamento de diferentes papéis, fiquem limitados aos níveis de hierarquia de usuários. E, apesar de ser um conceito recente e até bastante explorado, ainda não se tem definições claras de como as aplicações Multi-Inquilinos(MTA) devem se comportar[11].

No caso do Kaa, são 4 os papéis: Usuários Kaa, Administradores Kaa, Inquilinos Administradores e Inquilinos Desenvolvedores.

Mesmo com essa separação de papéis, a plataforma não sugere boas práticas, delimitações de responsabilidades ou uma representação dessa hierarquia aplicada como exemplo. Além disso, os Inquilinos Desenvolvedores, que normalmente ficam responsáveis por gerar os SDKs que serão utilizados, tem bastante poder dentro do sistema, onde pode ser explorado uma falha para comprometer o funcionamento saudável da aplicação. Outro ponto que pode ser explorado de falha são as contas padrões. Deve-se sempre ter cuidado para que as mesmas não sejam esquecidas no sistema, tanto no quesito de alterar as senhas padrões quanto em existência, pois

conhecer um usuário de acesso pode dar espaço para explorar *Social Hacking* ou mesmo força bruta.

Como um passo além de proteção nos SDKs, o Kaa ainda faz automaticamente a geração de chaves públicas e privadas para criptografar a comunicação utilizando RSA[5].

O sistema de Multi-Tenancy também permite que a aplicação rode em vários núcleos e tenha uma consistência dos papéis, cujas desvantagens - e também desafios - são seus custo de manutenção, devido aos dados distribuídos, e a ainda não possuir uma forma eficiente de manter todo o sistema alinhado. Sistemas distribuídos de Multi-Tenancy fazem todo sentido em aplicações IoT, pois abrandam os gargalos que um servidor unificado sofre com uma carga gigantesca de acessos.

5.2.6.1 SDK e Linux

Para testar de forma mais eficiente o SDK que foi configurado, foi implementado ele em um sistema operacional aprovado pela plataforma, no caso o Ubuntu 16.04 LTS. Para que a aplicação funcione, há apenas as dependências: `sudo apt-get install cmake build-essential`.

Isto posto, foi criada uma pasta para o projeto, *application-test*, e dentro dela, em uma subpasta de nome *kaa*, foi descompactado o SDK. Também foram criados outros dois diretórios: *build*, para o código compilado do Kaa, e *src*, para o código de teste. Ainda no diretório base foi adicionado o arquivo padrão do CMake do Kaa com as configurações básicas.

```
cmake_minimum_required(VERSION 2.8.12)
project(kaa-application C)

set(CMAKE_C_FLAGS "${CMAKE_C_FLAGS} -std=gnu99 -g -Wall -Wextra")

add_subdirectory(kaa)

add_executable(kaa-app src/main.c)

target_link_libraries(kaa-app kaac)
```

Figura 5.8: Exemplo de configuração do CMakeLists do Kaa

Dentro do diretório *src*, por conseguinte, foi criado o arquivo "*main.c*" baseado no projeto exemplo de SDK de Linux, que cria uma instancia do *client* do Kaa e logo depois encerra o mesmo.

Para finalizar o teste do SDK, basta entrar na pasta em "*build*", dentro do repositório base, e rodar uma série de comandos no terminal que preparam o Kaa. O primeiro deles, *cmake*

```

#include <stdio.h>
#include <stdlib.h>
#include <kaa/kaa.h>
#include <kaa/platform/kaa_client.h>
#include <kaa/kaa_error.h>

static void sample_function(void *context) {
    printf("Hello, I am a Kaa Application!\n");
    kaa_client_stop(context);
}

int main(void) {
    kaa_client_t *kaa_client = NULL;
    kaa_error_t error = kaa_client_create(&kaa_client, NULL);
    if (error) {
        return EXIT_FAILURE;
    }
    error = kaa_client_start(kaa_client, sample_function, (void *)kaa_client, 0);
    kaa_client_destroy(kaa_client);
    if (error) {
        return EXIT_FAILURE;
    }
    return EXIT_SUCCESS;
}

```

Figura 5.9: Código de exemplo do uso do SDK do Kaa em Linux

.., vai reunir os dados colocados no arquivo CMakeLists e criar uma *build* do Kaa. O segundo é gerar uma aplicação do Kaa, simplesmente usando *make*. Se ocorrer algum problema com o código, é nesse momento que ele avisará. Caso a aplicações exiba muitos *warnings*, ou mesmo os próprios *warnings* do Kaa, a plataforma recomenda descer o nível de *log*. Quando usado em uma aplicação real, é ideal que tanto *warnings* quanto erros não sejam exibidos, pois eles podem oferecer alguma brecha de informações internas do sistema.

Com os passos anteriores validados, o arquivo compilado "kaa-app" será gerado e basta executá-lo para ter um *Client* baseado no SDK gerado pelo servidor do Kaa instanciado e destruído logo em seguida.

5.2.7 Teste de Carga na Aplicação

Quando o projeto foi primeiramente debatido, alguns testes sobre a aplicação como um todo foram levantados como objetivo, entre eles os testes de carga. Os testes de carga, em poucas palavras, são uma rotina que simula o envio de múltiplas requisições para um servidor, de modo que possa ser mensurado uma necessidade de revisão em algum ponto da plataforma

caso o sistema esteja demorando para responder ou o impacto em alguma alteração sensível e, até mesmo, para fazer a medição do escalonamento do tempo de resposta das requisições.

No caso do Kaa em específico, por ele encapsular em seus SDKs a comunicação com o servidor, há grandes chances dos testes saírem com ruídos ou imprecisões, mas ainda assim esse não foi o motivo deles não terem sido feitos. Quando se fala em testes de carga em IoT, os mesmos precisam de muita acuidade, considerando que a quantidade de requisições de uma plataforma dessas tende a crescer exponencialmente. Assim, ter também medições de outros fatores se torna importantíssimo, como, por exemplo, ter um controle maior das aplicações que estão rodando, com dados apurados sobre as mesmas, ou um controle do *status* das ferramentas utilizadas pela plataforma.

6 RESULTADOS

Levando em conta o passo a passo desenvolvido ao longo do capítulo de 5 e os resultados esperados que foram levantados durante a apresentação de aprovação do projeto, os resultados serão qualificados em três categorias: Integrando de Ponta a Ponta, Melhorias desejáveis e Implantação em sistemas reais. A outra categoria levantada: "Documentação do Estudo de Caso", é completamente abrangida no capítulo 5.

6.1 Integrando de Ponta a Ponta

O maior problema enfrentado durante a produção do trabalho encontrou-se na integração com o módulo Wi-Fi ESP8266, devido a como o Kaa oferece suporte para o módulo. Pensando somente no módulo, o mesmo é integrável ao Arduino e a combinação dos dois é amplamente usada para testes de conceito. Inclusive, o próprio Kaa costumava oferecer suporte ao Arduino nos seus primórdios, mas durante a execução do trabalho o Kaa removeu algumas páginas de suporte tanto ao Arduino quanto ao módulo ESP.

Atualmente o suporte do módulo ESP é utilizado junto do Ubuntu e com uma comunicação através de USB que não serviria para uma aplicação para o mundo real. Outro ponto é que o módulo ESP8266 precisa de uma amperagem alta para funcionar corretamente e testar ele direto em uma porta USB pode gerar ruídos que atrapalhem no desenvolvimento, dependendo da forma como a máquina em questão faz a distribuição energética entre as portas.

Pela falta de tempo hábil para testar apropriadamente em outras plataformas suportadas pelo Kaa, devido a descontinuação do Arduino, foram tentadas algumas abordagens para validar o uso para um caso real ainda com o Estudo de Caso estipulado. A primeira foi usando o Arduino como canal de comunicação direta para a placa ESP8266 através dos protocolos UART da placa, entretanto o compilador não funcionava apropriadamente, não reconhecendo o módulo Wi-Fi de forma correta. Outra abordagem foi tentar isolar os componentes de comunicação do Kaa para usar diretamente através do Arduino, porém não houve tempo suficiente para estudar todas as dependências e conseguir produzir um resultado.

Inclusive, uma reclamação recente e pertinente quanto à plataforma foi sobre a falta de suporte da mesma a sistemas embarcados mais simples, dificultando o uso em aplicações reais *standalone* que ficassem fora de um ambiente controlado.

6.2 Melhorias desejáveis

Um dos maiores problemas em trabalhar com o Kaa localmente foi a máquina virtual local. Com ela não foi possível criar outros *Tenants* para teste, além de ela apresentar uma certa lentidão após alguns minutos de execução. O mesmo não aconteceu com o servidor na nuvem, que após a inicialização manteve um uso constante de CPU e nenhuma lentidão aparente.

Outro ponto é que, comparando com outros sistemas que entregam soluções em IoT, a versão 0.10.0 do Kaa também é pouquíssimo intuitiva e documentada para os desenvolvedores. Os manuais são de certa forma grandes mas não explicam o suficiente, fazendo-se necessário o estudo em sempre mais de um lugar para descobrir informações relevantes. Nesse ponto, sistemas como Losant ou ThingsSpeak são muito mais intuitivos e com uma facilidade de integração maior, usando como segurança um sistema de Chaves Públicas e Privadas(RSA) e APIs de comunicação direta.

Por fim, há uma lacuna quanto aos dados apresentados pelo Kaa de forma fácil para o usuário. Mostrar os status dos serviços da aplicação, mostrar dados atualizados das requisições, fazer um trabalho em cima de gráficos, implementar ou integrar com algum sistema de testes(Valid8, IoTify), etc... são futuras implementações que agregariam poder de comunicação da plataforma com os gestores do sistema.

6.3 Implantação em sistemas reais

A plataforma Kaa não está em uma versão final e ainda tem muito a evoluir. Por se tratar somente da versão 0.10.0, é esperado que problemas ou desafios ainda não resolvidos estejam no caminho para que a aplicação funcione 100%, o que pode acabar sendo crítico para um sistema. Entretanto, o Kaa demonstra maturidade em muitos pontos e algumas vantagens quanto aos concorrentes.

Uma das principais vantagens, inclusive, é o sistema de *Multi-Tenancy*, que, aplicado em um modelo descentralizado e de forma consciente[3], cria um sistema blindado e distribuído. Outra vantagem é que a plataforma fica completamente nas mãos de quem a utiliza, não sendo mantida por terceiros, o que é um ponto chave para aplicações proprietárias e de código fechado.

Quanto à desvantagens, a curva de aprendizado da plataforma é lenta e necessita de um leque de conhecimento bem amplo por parte do utilizador. A plataforma também falha em guias para os desenvolvedores que quiserem sair mais dos braços do Kaa.

7 CONCLUSÃO

A área de Internet das Coisas tem crescido de forma acelerada nos últimos anos, mas ainda conta com uma série de limitações que a impedem de se consolidar no momento. Assim, estudos que a ajudem a formar a base de conhecimento e também os que validem os avanços da área se fazem muito necessários. Da mesma forma que nem sempre os resultados serão positivos trabalhando nesse campo do conhecimento no atual momento, o que não necessariamente implica em ser algo negativo. Todas as falhas documentadas, todas as limitações, todos os desafios que forem levantados e solucionados ajudarão a trilhar o caminho que guiará a Internet das Coisas para um novo patamar.

Assim, mesmo as dificuldades de desenvolvimento e os problemas enfrentados durante o processo não parecem tão grandes, pois há muito mais a ser trilhado e essa é só uma das diversas contribuições que podem ser feitas.

7.1 Trabalhos futuros

- Seria interessante, para fins de comparação e mesmo evolução, pegar algumas outras plataformas e compará-las com o Kaa em diversos aspectos (Desempenho, integração, segurança, etc) para posicionar o Kaa ante seus concorrentes.
- Um ponto que hoje não é uma preocupação do Kaa mas pode acabar se tornando algo relevante é o estudo da implementação do Kaa de forma enxuta em um sistema mais leve, para cobrir essas demandas de aplicações cujo objetivo é ficar isoladas de um ambiente controlado.

REFERÊNCIAS

- [1] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
- [2] M. Brachmann, S. L. Keoh, O. G. Morchon, and S. S. Kumar. End-to-end transport security in the ip-based internet of things. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on*, pages 1–5. IEEE, 2012.
- [3] e. a. Cherrier, Sylvain. Multi-tenancy in decentralised iot. *World Forum Internet of Things (WFIoT 2015)*, 2015.
- [4] C. Doukas and I. Maglogiannis. Bringing iot and cloud computing towards pervasive healthcare. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 922–926. IEEE, 2012.
- [5] A. S. Dr. Prerna Mahajan. A study of encryption algorithms aes, des and rsa for security. *Global Journal of Computer Science and Technology*, 2013.
- [6] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.
- [7] P. Harpe, K. A. Makinwa, and A. Baschirotto. Hybrid adcs, smart sensors for the iot, and sub-1v & advanced node analog circuit design.
- [8] e. a. Julien, Minerauda. A gap analysis of internet-of-things platforms. *Computer Communications 000*, page 1–12, 2016.
- [9] Kaa. Getting started, 2017. Disponível em: <<https://kaaproject.github.io/kaa/docs/v0.10.0/Getting-started/>>. Acesso em: 29 de Junho de 2017.
- [10] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay. Towards the implementation of iot for environmental condition monitoring in homes. *IEEE Sensors Journal*, 13(10):3846–3853, 2013.

- [11] R. Krebs, C. Momm, and S. Kouney. Architectural concerns in multi-tenant saas applications. *Proceedings of the 2nd International Conference on Cloud Computing and Services Science*, 2012.
- [12] D. Kyriazis, T. Varvarigou, D. White, A. Rossi, and J. Cooper. Sustainable smart city iot applications: Heat and electricity management & eco-conscious cruise control for public transportation. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–5. IEEE, 2013.
- [13] E. Metalidou, C. Marinagi, P. Trivellas, N. Eberhagen, C. Skourlas, and G. Giannakopoulos. The human factor of information security: Unintentional damage perspective. 147, 08 2014.
- [14] e. a. Miorandi, Daniele. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.
- [15] e. a. Nayak, Ameya. Type of nosql databases and its comparison with relational databases. 5, 03 2013.
- [16] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Sensing as a service model for smart cities supported by internet of things. *Transactions on Emerging Telecommunications Technologies*, 25(1):81–93, 2014.
- [17] K. J. Singh and D. S. Kapoor. Create your own internet of things: A survey of iot platforms. *IEEE Consumer Electronics Magazine*, 6(2):57–68, 2017.
- [18] F. Tao, L. Zhang, V. Venkatesh, Y. Luo, and Y. Cheng. Cloud manufacturing: a computing and service-oriented manufacturing model. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 225(10):1969–1976, 2011.
- [19] e. a. TorĜul, BelkĪz. Internet of things: A survey. *International Journal of Applied Mathematics, Electronics and Computers*, page 104–110, 2016.
- [20] A. Zaslavsky, C. Perera, and D. Georgakopoulos. Sensing as a service and big data. *arXiv preprint arXiv:1301.0159*, 2013.