UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Eduardo Willwock Lussi

Descrição do Projeto Final: Gerenciador de Rede IoT com Detecção de Intrusão

Florianópolis
2020

Eduardo Willwock Lussi

Descrição do Projeto Final: Gerenciador de Rede IoT com Detecção de Intrusão

Trabalho submetido à disciplina INE 5404 – Programação Orientada a Objetos II do curso de Ciência da Computação da Universidade Federal de Santa Catarina.
Orientador: Prof. Alex Sandro Roschildt, Dr.

Florianópolis

2020

1 INTRODUÇÃO

O estudo da segurança em redes de computadores deve sempre acompanhar a evolução das próprias tecnologias das redes para que haja um equilíbrio. Em ambientes IoT, como em casas e condomínios inteligentes, que controlam as mais diversas “coisas”, isso se torna ainda mais fundamental. Existem várias soluções dentro deste contexto que protegem o sistema contra ataques externos, mas há poucas considerando um possível ataque interno da rede, causado pela infecção de algum nó ou até um mal funcionamento. Com esse objetivo, pensou-se em desenvolver um gerenciador de rede IoT com um sistema de detecção de intrusão (IDS) com o objetivo de proteger o dispositivo Fog de possíveis ataques internos, com foco de aplicação em casas e condomínios inteligentes.

Este trabalho apresenta uma descrição completa desse software, que foi desenvolvido e aperfeiçoado durante o decorrer da disciplina. A seção 2 trata da descrição geral da ideia do projeto. A seção 3 mostra a modelagem do sistema, com uma análise de requisitos, diagrama de classes e diagrama entidade-relacionamento (ER). E por último, a seção 4 descreve a implementação do projeto, detalhando os subsistemas, bibliotecas e ferramentas utilizadas.

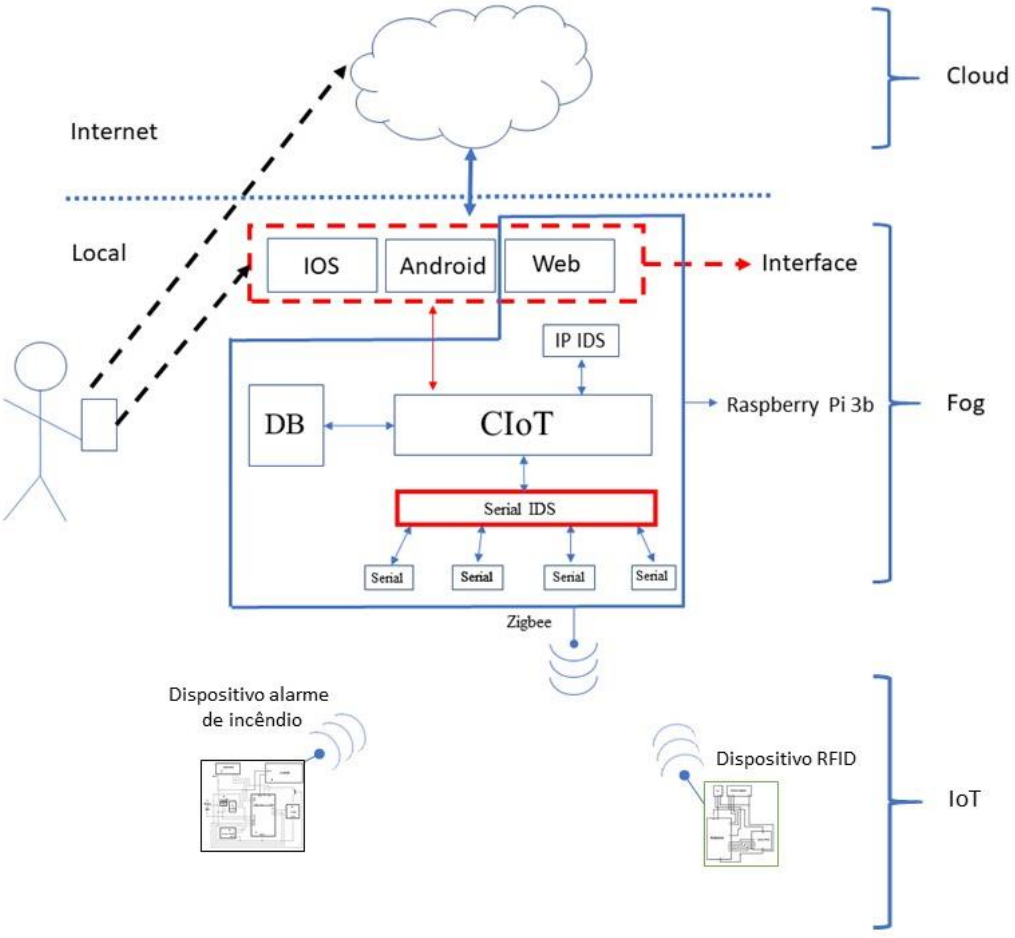
2 DESCRIÇÃO DA IDEIA

O software será uma extensão do sistema de detecção de intrusão proposto por Cardoso et al. (2019) [1], que monitora os pacotes TCP/IP, protegendo o sistema contra ataques externos. Enquanto isso, o projeto aqui descrito é um gerenciador de rede IoT que possui um sistema de detecção de intrusão (IDS) contra ataques de negação de serviço vindos de possíveis dispositivos IoT infectados. Os dispositivos IoT estão em comunicação com o sistema via porta serial. Cada dispositivo IoT possui um protocolo padrão Procotol Data Unit (PDU), que descreve os atributos lidos por cada sistema e tem como principal utilidade filtrar os pacotes, evitando possíveis ataques através da verificação de endereço e PDU e da contagem de pacotes por segundo que chegam à rede, bloqueando o dispositivo caso essa taxa supere o limite estabelecido. O PDU também permite o armazenamento dos dados analisados, para futuras análises e comparações.

2.1 AMBIENTE DE IMPLEMENTAÇÃO

A figura 1 mostra a arquitetura de aplicação do sistema. A camada IoT é composta, até então, por um alarme de incêndio proposto por Sampaio et al. (2019) [2] e um dispositivo de controle de acesso RFID, que estão em comunicação com a camada Fog por meio de uma antena Zigbee. O dispositivo Fog, que atuará como um servidor do logradouro inteligente, é um Rasberry Pi 3b, que possui um banco de dados e um servidor Web. O sistema possui uma Central IoT (CIoT), que gerencia os serviços e as comunicações. O Serial IDS analisa os pacotes vindos das portas seriais e bloqueia o endereço caso encontre alguma anomalia. O nó Fog recebe os dados filtrados pelo Serial IDS, armazena do banco de dados e mostra a informação na aplicação Web.

Figura 1 – Arquitetura do Sistema



3 MODELAGEM E ANÁLISE DE REQUISITOS

Para especificar melhor a ideia do sistema, foi primeiramente construído um breve texto para descrever o projeto, com o objetivo de identificar seus elementos e auxiliar na modelagem e estruturação. O texto construído é o que segue:

> O software será um gerenciador de DISPOSITIVOS IOT que possui um sistema de detecção de intrusão (IDS) contra ataques de negação de serviço vindos de possíveis dispositivos IoT infectados. Os DISPOSITIVOS IOT estão em comunicação com o sistema via porta serial. Cada DISPOSITIVO IOT possui um protocolo padrão Procotol Data Unit (PDU), que descreve os ATRIBUTOS lidos por cada sistema, que serve para filtrar os pacotes, evitando possíveis ataques e permitindo o armazenamento dos DADOS ANALISADOS, para futuras análises e comparações. O IDS também analisa a quantidade de pacotes recebidos por segundo e bloqueia o dispositivo a caso encontre alguma anomalia.

3.1 REQUISITOS FUNCIONAIS

Após a descrição com um texto sucinto, foram identificados e especificados os requisitos funcionais do sistema, subdivididos em três categorias, necessários, desejáveis e opcionais:

- Necessários:
    - Mostrar ATRIBUTOS dos DISPOSITIVOS IOT
    - Detectar automaticamente os DISPOSITIVOS IOT conectados
    - Filtrar dados e prevenir ataques através do sistema de detecção de intrusão
- Desejáveis:
    - Permitir análise dos DADOS ANALISADOS
    - Consultar histórico dos DADOS ANALISADOS
    - Enviar dados e controlar os DISPOSITIVOS IOT
- OPCIONAIS
    - Monitorar tempo restante de bateria dos DISPOSITIVOS IOT com bateria

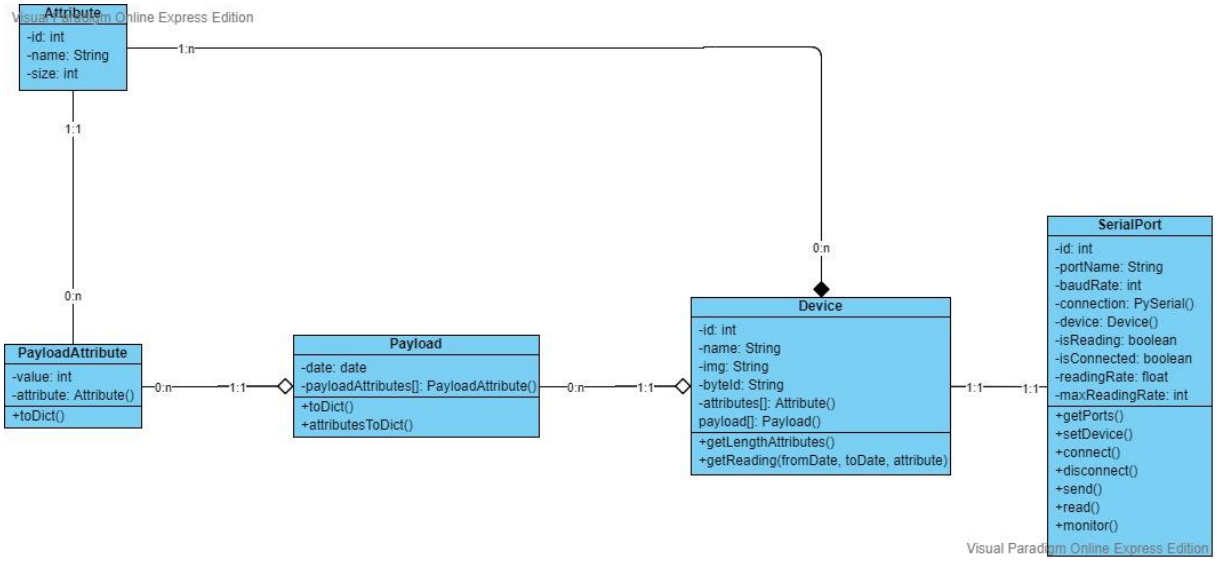
- Monitorar gasto de energia dos DISPOSITIVOS IOT conectados diretamente a uma fonte de energia

Dentre os requisitos apresentados, foram implementados todos menos os opcionais, pois se tratam de questões mais complexas que requerem mais tempo de pesquisa e planejamento.

3.2 DIAGRAMA DE CLASSES

Tendo a ideia do projeto bem estruturada, é possível seguir para a construção do diagrama de classes. A partir da observação dos substantivos presentes no texto que descreve o projeto, foi possível identificar as classes bases da implementação do software, destacadas em letras maiúsculas e após muito tempo de análise e adaptações, o diagrama de classes ficou conforme a figura 2.

Figura 2 – Diagrama de Classes



O diagrama é composto por 5 classes. A classe SerialPort é a principal classe do software, ela representa as portas seriais que estão conectadas aos dispositivos IoT, por esse motivo, é composta por métodos para leitura, escrita, monitoramento e identificação das portas seriais. Dentre seus atributos, há um Device, que representa o sistema IoT correspondente, identificado através do primeiro byte do seu PDU, representado pelo atributo byteId e definido através do método setDevice. O método getPorts é um método estático que faz a leitura de todas as portas seriais e retorna uma lista com as portas ativas.

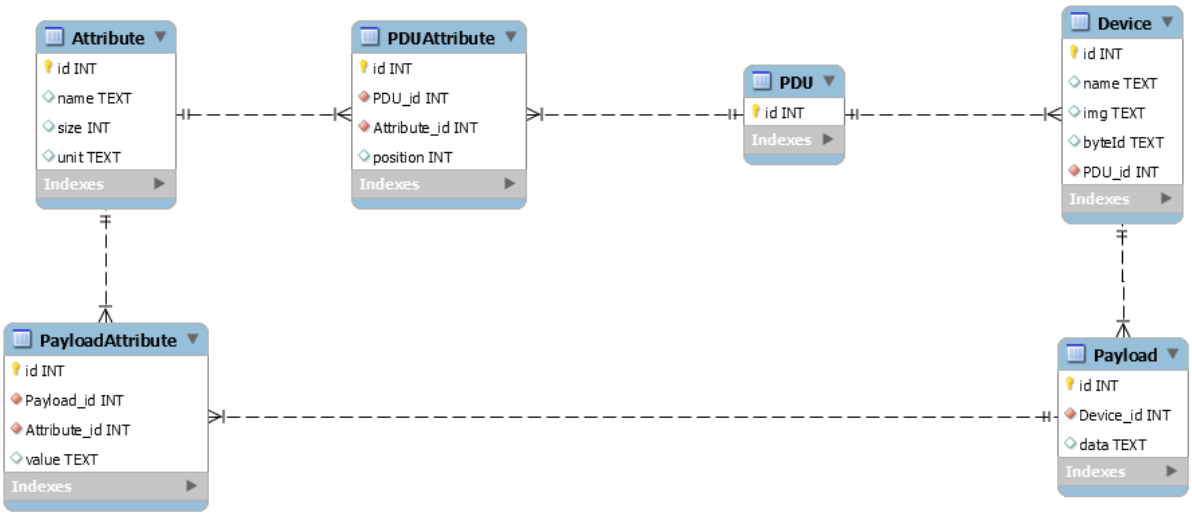
A classe Device é composta por vários atributos e nenhum ou vários payloads. Ela representa os dispositivos IoT e serve de base para identificar e moldar o dispositivo definido. Seus atributos são da classe Attribute, que possui somente um id, nome e tamanho. Já a classe payload representa os dados lidos durante o monitoramento, ela possui uma data e nenhum ou vários PayloadAttribute, que representam a leitura de um determinado atributo.

O sistema desenvolvido até então pressupõe que todos os dispositivos IoT possuem o mesmo tipo de padrão e ele ainda não está adaptado para funcionar de forma mais dinâmica, com diferentes dispositivos que possuem diferentes tipos de atributos. A implementação dos padrões de projeto Factory Method e Façade para criar e gerenciar uma classe para cada tipo de dispositivo, permitindo uma melhor personalização de cada sistema IoT.

3.3 PERSISTÊNCIA DOS DADOS

Após a especificação de requisitos e modelagem do diagrama de classes, é possível prosseguir para a modelagem do banco de dados, que neste caso, está sendo utilizado o banco relacional SQLite. O diagrama entidade-relacionamento (ER) está representado na figura 3.

Figura 3 – Diagrama ER



O diagrama ER além de possuir o dispositivo, payload, payloadAttribute e o atributo, possui tabelas para o PDU. Essas tabelas armazenam como será o padrão de pacote lido, pois durante a leitura, o dispositivo IoT envia uma quantidade definida de bytes e o software precisa saber o que cada byte significa. Por convenção, o primeiro byte é o identificador do

sistema IoT, os próximos dois bytes são o endereço de origem do pacote e o restante são os valores lidos de cada atributo em hexadecimal conforme a tabela ASCII.

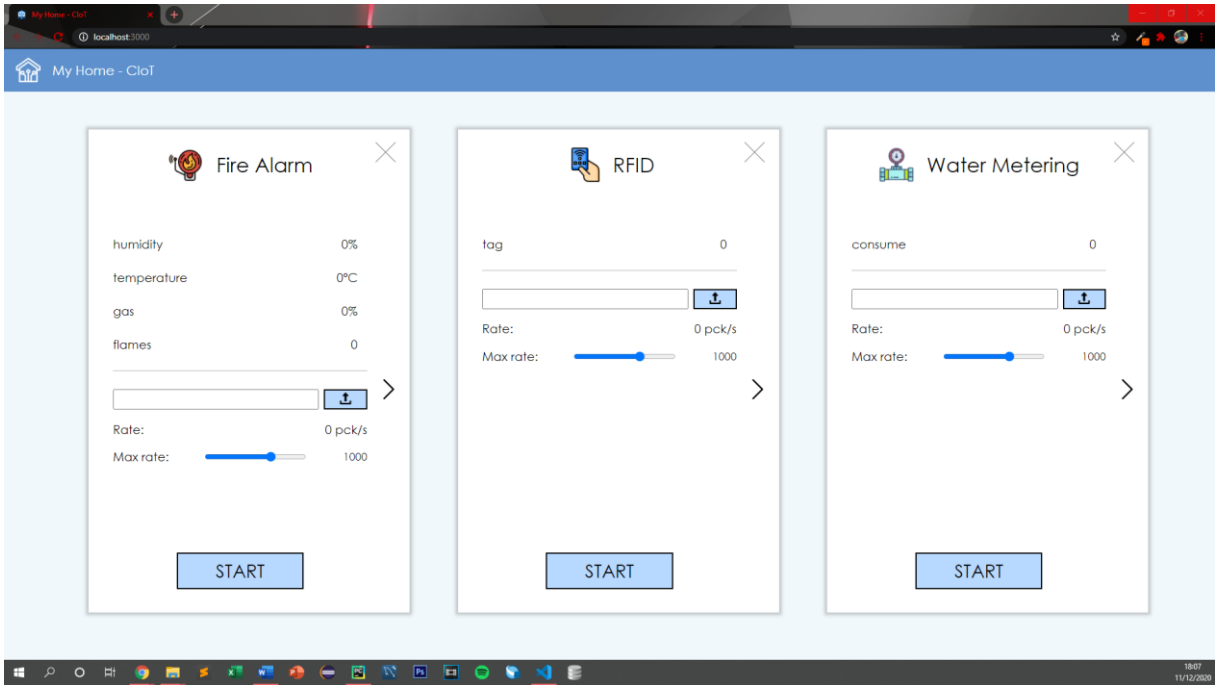
4 IMPLEMENTAÇÃO

O sistema construído é composto por duas partes principais, um servidor python construído através do framework bottle na qual é realizada toda a gerencia da rede através da implementação do diagrama de classes, e um site com interface gráfica desenvolvido com o auxílio da biblioteca React.

4.1 INTERFACE GRÁFICA: GUI

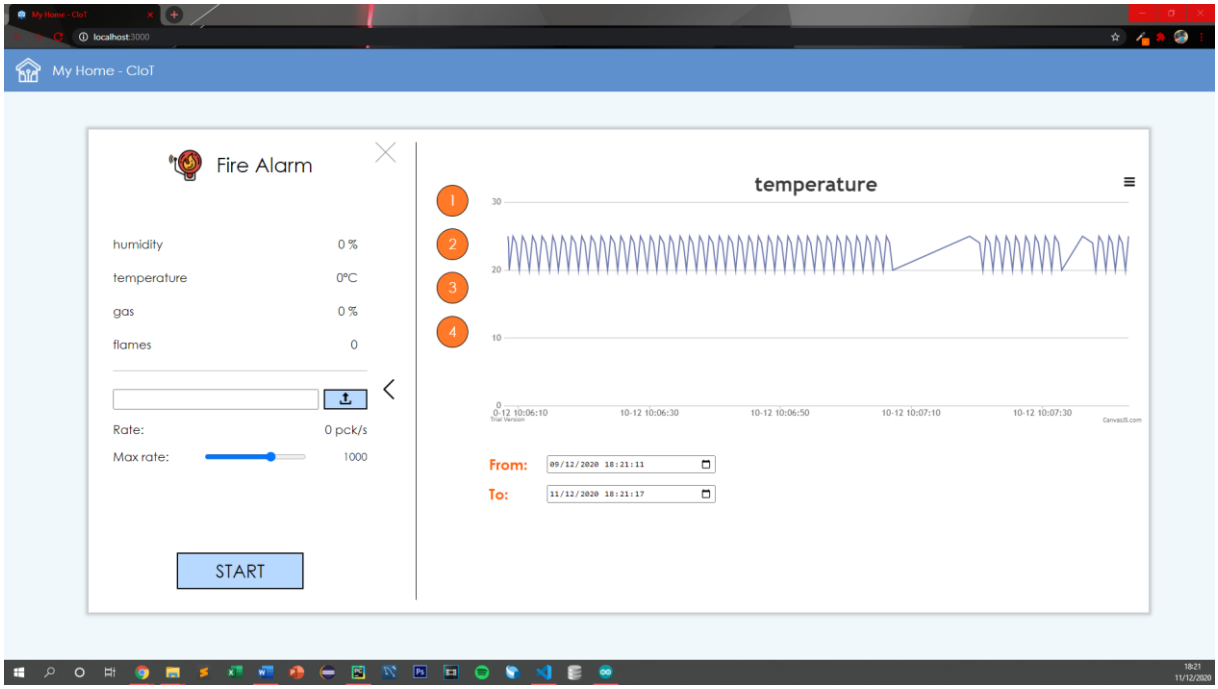
A interface gráfica (GUI) do sistema foi desenvolvida utilizando a biblioteca Javascript React, que, por ser baseada em componentes, auxilia muito para o desenvolvimento de aplicações mais dinâmicas. Essa interface está apresentada na figura 4, nela, é possível perceber três sistemas IoT diferentes listados. Ao entrar na tela inicial, os dispositivos não estão sendo lidos e monitorados. Para iniciar a captura e monitoramento, é necessário selecionar a taxa máxima de pacotes por segundo e clicar no botão start, que enviara uma requisição para o servidor back-end iniciar o monitoramento. Com a captura em andamento, os dados de cada atributo são mostrados em tempo real, a entrada de texto está disponível para enviar dados para o sistema e a taxa de pacotes por segundo, atualizada a cada 5 segundos, através da contagem dos pacotes, é mostrada no campo Rate.

Figura 4 – Interface Gráfica (GUI)



Ao clicar para expandir, o item é expandido mostrando uma interface que permite consultar e analisar os dados lidos ao longo do tempo através de um gráfico desenvolvido com o auxílio da biblioteca de gráficos CanvasJS. Após selecionar a data e hora de início e fim, escolhemos o atributo e os resultados são mostrados no gráfico através do retorno de uma requisição para o back-end que realiza uma consulta no banco. A figura 5 mostra essa funcionalidade do sistema.

Figura 5 – Interface Gráfica (GUI): Gráfico



4.2 BACK-END

Como já dito anteriormente, o servidor backend é completamente desenvolvido em python, com o auxílio da biblioteca bottle. Esse bloco do sistema é orientado a objetos e foi utilizado o padrão de projeto MVC para separar as classes de modelagem, Beans, das classes de comunicação com o banco, DAOs. Além dos pacotes Beans e DAOs, esse subsistema possui um pacote chamada Utils, que contém a classe de comunicação com o banco, realizada através de uma classe de conexão utilizando a biblioteca sqlite3, construída com o padrão de projeto singleton, para evitar que haja várias conexões com o banco de dados, e um arquivo principal server.py que contém o próprio servidor para a gerência das requisições.

Dentre as classes DAOs, existem apenas duas, uma para o dispositivo, com métodos para obter quais são seus atributos e qual o dispositivo com determinado identificador, e outra para o payloadAttribute, com métodos para inserir um Payload e obter os valores lidos entre determinado intervalo de tempo.

A conexão com os dispositivos IoT é realizada através da biblioteca pyserial e as funções de monitoramento e modificações do banco são executadas em threads com a biblioteca nativa do python, threading. A transmissão de dados com o front-end é realizada através do formato JSON.

REFERÊNCIAS

[1] CARDOSO, J. V. et al. DoS attack detection and prevention in fog-based inteligente environments. Brazilian Journal of Development, v. 5, n. 11, p. 23934–23956, 2019.

[2] SAMPAIO, H. V. et al. Autonomic IoT Battery Management with Fog Computing. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v. 11484 LNCS, n. Cc, p. 89–103, 2019.