FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Janaine Correia de Barcelos

DESENVOLVIMENTO DE UM SISTEMA MONITOR DE CONSUMO DE ENERGIA ELÉTRICA INTITULADO VIOLET ENERGY

Americana, SP

2019

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Janaine Correia de Barcelos

DESENVOLVIMENTO DE UM SISTEMA MONITOR DE CONSUMO DE ENERGIA ELÉTRICA INTITULADO VIOLET ENERGY

Projeto de Conclusão de Curso, apresentado como requisito parcial para obtenção do grau Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Faculdade de Tecnologia de Americana.

Orientador: Prof. Dr. Kleber de Oliveira Andrade

Área de concentração: Engenharia de Software

Americana, SP

2019

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

B218d BARCELOS, Janaine Correia de

Desenvolvimento de um sistema monitor de consumo de energia elétrica intitulado Violet Energy. / Janaine Correia de Barcelos. – Americana, 2019.

80f.

Monografia (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kleber de Oliveira Andrade

1 Internet das coisas 2. Aplicativos WEB I. ANDRADE, Kleber de Oliveira II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: *681.518*

Janaine Correia de Barcelos

DESENVOLVIMENTO DE UM SISTEMA MONITOR DE CONSUMO DE ENERGIA ELÉTRICA INTITULADO VIOLET ENERGY

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pelo CEETEPS/Faculdade de Tecnologia – Fatec/ Americana.
Área de concentração: Engenharia de Software

Americana, 12 de Junho de 2019.

Banca Examinadora:

Kleber de Oliveira Andrade (Presidente)
Doutor
Fatec Americana

José Luís Zem (Membro)
Doutor
Fatec Americana

Rossano Pablo Pinto (Membro)
Mestre
Fatec Americana

RESUMO

Internet of Things (Internet das Coisas) tem como objetivo conectar objetos, gerando dados para extrair informações úteis para conveniência da vida humana. O intuito dessas informações é dar a possibilidade de agir proativamente. Neste trabalho de graduação, usamos o monitoramento do consumo de energia em uma casa como um exemplo da utilidade desse tipo de tecnologia. O processo de monitoramento extrai informações em tempo hábil, de modo a antecipar a decisão de diminuir o consumo de energia de um equipamento antes que a conta de energia chegue. Assim, o objetivo deste projeto é desenvolver um sistema de monitoramento de consumo de energia que ajude o usuário a gerenciar seu consumo de energia de forma mais consciente. O sistema de monitoramento é composto por uma interface web, hospedada em um servidor que fornece dados de consumo e o protótipo de um equipamento de monitoramento de energia elétrica. O protótipo é desenvolvido com um sensor não invasivo, conectado via rede sem fio à Internet, e envia dados de consumo para a nuvem a qual alimenta todo o sistema.

Palavras Chave: IoT; Desenvolvimento Web.

ABSTRACT

Internet of things aims to connect objects and generates data which offers useful information for human life convenience. The purpose of this information is to give users the possibility to act proactively. In this paper we use home energy consumption monitoring as an example of the usefulness of this type of technology. The monitoring process extracts information in a timely manner so as to anticipate the decision to decrease energy consumption of an equipment before the energy bill arrives. So, the purpose of this paper is to develop an energy consumption monitor system which helps user to consume energy more consciously. The monitoring system is composed of a web interface, hosted on a server which provides consumption data, and the prototype of an electric energy monitoring equipment. The prototype was created with a non-invasive sensor, wirelessly connected to the Internet, and it sends consumption data to the cloud and feeds the whole system.

Keywords: IoT; Web Development.

AGRADECIMENTOS

Um projeto de conclusão de curso é sem sombra de dúvidas um divisor de águas na vida de um graduando, o esforço necessário é extraordinário, a ansiedade é um obstáculo e os dias passam mais rápido do que o previsto. A determinação e o foco é sem dúvida essencial para tornar o trabalho possível, mas não apenas isso foi fator decisivo para conclusão deste trabalho de graduação, apesar deste levar apenas o nome de sua autora, muitos foram os que apoiaram e a esses agradeço por toda ajuda.

Em especial agradeço meu orientador, Kleber de Oliveira Andrade, pela paciência, otimismo e por guiar-me por todo esse processo sempre me dando toda atenção que fosse necessário. Ao meu companheiro, Vitor Ghiraldelli Silva, reservo os mais estimados agradecimentos por apoiar-me, motivar e sempre, prontamente, partilhar seus conhecimentos do âmbito profissional para alavancar meu desenvolvimento profissional e acadêmico. Aos meus colegas de trabalho, que por todo o meu período de estágio me deram a oportunidade de desenvolver-me como profissional e conhecer as tecnologias que possibilitaram grande parte deste projeto. E por fim, mas não menos importante, agradeço aos meus pais que forneceram a mim o privilégio, o sustento e as bases para estudar sem nunca me cobrar por isso, apenas com a intenção de me ajudar a conquistar o futuro almejado.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Segundo o vice-presidente de pesquisa da empresa Gartner, Mark Hung, em 2020 é esperado por volta de 20 bilhões de equipamentos, além dos convencionais computadores e celulares, conectados à Internet (HUNG, 2017). Estes bilhões de equipamentos que estão conectados são a *Internet of Things*[1] (IoT) e toda essa conectividade promete mudar a vida das pessoas, como relata Evans (2011, p. 6).

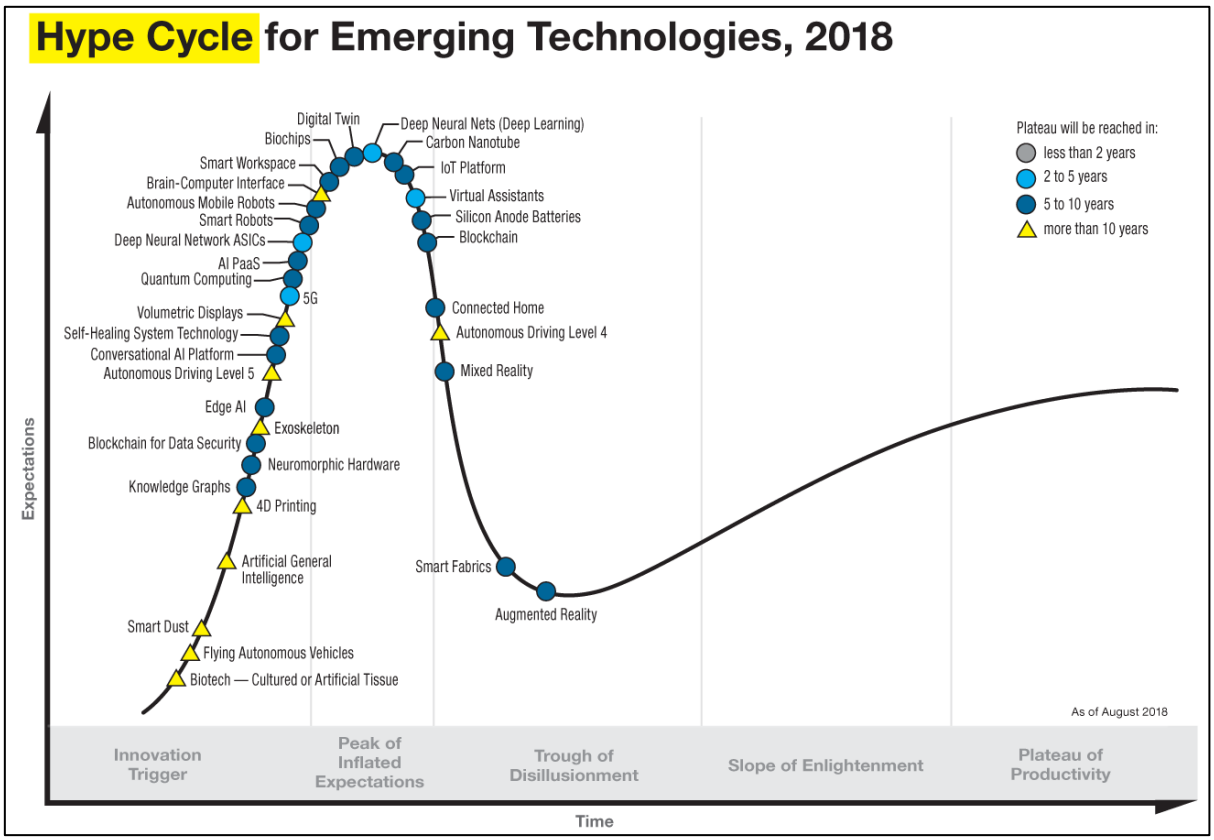
> [...] IoT se torna imensamente mais importante, pois é a primeira evolução real da Internet, um salto que levará a aplicações revolucionárias com potencial para melhorar consideravelmente a forma como as pessoas vivem, aprendem, trabalham e se divertem.

Para Evans (2011) a IoT é a evolução da Internet. Ela vem para tornar a vida humana mais conveniente, trazer maior qualidade de vida, coletando e transformando dados para a humanidade progredir como um todo. Essa revolução tecnológica tem grande potencial, segundo o IDC (*International Data Corporation*), gastos com o mercado de IoT pode chegar a 754 bilhões de dólares. Devido aos resultados positivos observados nos dados gerados pelos equipamentos conectados à Internet, ajudando na toma de decisão e obtenção de resultados rápidos, a indústria vê a IoT com otimismo (IDC, 2019). Já a parcela do mercado de IoT para consumidores finais, com soluções como casas e carros inteligentes, pode chegar a 108 bilhões de dólares (IDC, 2019).

Outro número expressivo é a quantidade de dados gerados na Internet em 2017: São 2.5 quintilhão de bytes por dia, sendo que 90% de todos os dados foram criados nos 2 últimos anos e mais da metade deste tráfego de dados vêm dos dispositivos móveis, como celulares (DOMO, 2017). Segundo Evans (2011), a IoT representa uma parte importante na coleta, análise e distribuição desses dados, pois a transformação desses dados em informação pode contribuir para importantes avanços. Esse enorme potencial levanta grandes expetativas do mercado sobre as aplicações IoT. Como é possível observar na Figura 1, a curva de Gartner exibe em que etapa a Internet das Coisas se encontra nos *Hype Cycle* [2] das tecnologias emergentes.

[1] Em português: Internet das Coisas.
[2] Em português: Ciclo de Expectativas.

Figura 1 – Curva de Gartner

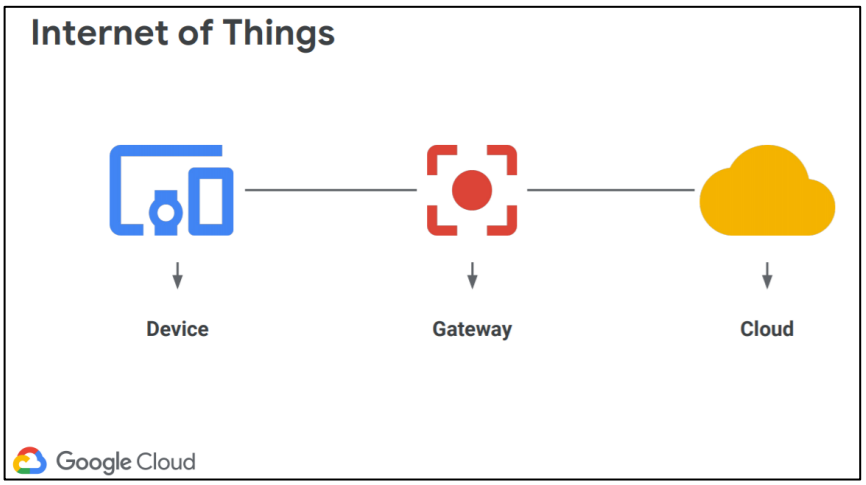


Fonte: Gartner, 2018.

É possível observar na Figura 1 o ciclo das expectativas para tecnologias emergentes; os dois eixos representam o ciclo da inovação até a solidificação da tecnologia no mercado. A expectativa em relação a tecnologia, eixo vertical, e o tempo, eixo horizontal, estima o tempo para a tecnologia atingir o platô de produtividade. Para a IoT estima-se entre 5 a 10 anos para atingir a maturidade de um produto viável para o mercado, descendo o pico da curva aonde as expectativas ainda estão infladas. Há muito entusiasmo e publicidade sobre a Internet das Coisa, neste momento muitas ideias estão surgindo e sendo testadas.

Para entender o funcionamento da IoT basta observar a Figura 2, apresentada no evento Cloud Next ’18 do Google.

Figura 2 – Internet das Coisas

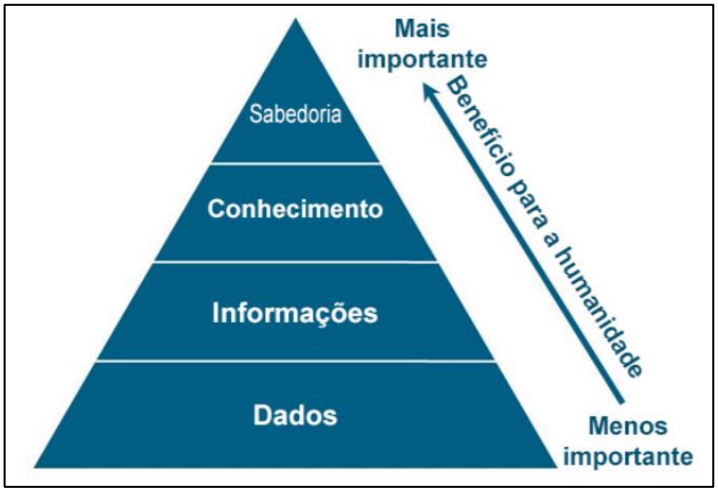


Fonte: Google Cloud Platform, 2018.

A Figura 2 apresenta, de modo simples, o funcionamento da IoT. Um equipamento qualquer conecta-se a um *gateway*, que pode ser uma conexão Wi-Fi, rede 3G, 4G ou outra qualquer, que, por fim, conecta-se à nuvem. Os dados captados são então armazenados e manipulados e deles são extraídas informações úteis para o propósito que melhor convir. Portanto, o fluxo da IoT consiste na captura de dados para transformá-los em informações valiosas, é fundamental compreender esse fluxo, pois esse é o funcionamento e motivação da IoT. Como menciona Evans (2011), essa tecnologia pretende tornar a vida das pessoas mais conveniente e manter a humanidade um passo à frente. Por essa razão as expectativas sobre essa tecnologia são elevadas.

A pirâmide da Figura 3 mostra o nível de importância da transformação de um dado em sabedoria.

Figura 3 – Transformação de dados em Sabedoria



Fonte: Cisco, 2011.

Evans discorre sobre a importância da transformação de dados em sabedoria, pois os dados sozinhos são pedaços de informações sem conexão ou sentido; ao serem transformados em informação eles podem apresentar padrões, tendências as quais somam em conhecimento, que pode ser valioso para compreender um evento. A sabedoria em função do conhecimento adquirido é então o resultado mais benéfico e importante para a humanidade (EVANS, 2011), estando no nível mais alto de importância exibido na pirâmide da Figura 3.

Em suma, a Internet das Coisas pretende captar dados emitidos por objetos, canalizá-los para gerar informações e conhecimentos benéficos. Um exemplo do uso dessa tecnologia é o projeto desenvolvido nesse trabalho de graduação: IoT para eficiência energética, o qual faz uso dessa tecnologia para captar informações como o consumo de energia de aparelhos elétricos e transformar esses dados, o quais foram captados pelo equipamento monitor, em informações úteis para seus usuários, possibilitando um ambiente mais inteligente. Como mencionado pelo IDC, o mercado para casas inteligentes. está dentro de uma fatia grande de investimento em IoT (cerca de 108 milhões de dólares (IDC, 2019)). Portanto, esse tipo de automação gera grandes expectativas no mercado e já há empresas que disponibilizam sistemas monitores de consumo de energia.

A Tabela 1 contém exemplos reais de sistemas de eficiência energética para casas inteligentes, é importante mencionar que alguns dos sistemas presentes na tabela referida não são apenas voltados para consumo de energia elétrica, mas também para outros tipos de automações. Porém, como este projeto de graduação

desenvolve um sistema para o caso de uso de consumo de energia elétrica, a tabela irá se ater apenas às funcionalidades relacionadas à eficiência energética.

Tabela 1 – Tabela de sistemas de monitoramento de consumo de energia elétrica relacionados

Sistema	**Funcionalidades e características do sistema**	**Preço**
Open Energy Monitor	• Equipamento monitor de consumo de energia elétrica. • Equipamento conectado à rede Wi-Fi. • Web site para visualizar dados de consumo de energia. • Consumo em quilowatts-hora(kWh). • Consumo de energia diário. • Indicador do custo de consumo. • Monitoramento em tempo real.	R$901,86, Libra Esterlina a R$5,15, dia 16/05/2019.
Sense	• Equipamento monitor de consumo de energia elétrica. • Aplicação mobile para visualizar dados de consumo de energia. • Equipamento Conectado à rede Wi-Fi. • Consumo em quilowatts-hora(kWh). • Consumo mensal e diário. • Analise dos dados de consumo. • Sugere melhorias de consumo de energia. • Identifica equipamentos consumindo energia. • Monitoramento em tempo real.	Sense R$1.203,36, Dólar a R$4,02, dia 16/05/2019.
Engage	• Equipamento monitor de consumo de energia elétrica. • Equipamento conectado a um hub o qual conecta-se via cabo a um roteador de Internet. • Aplicação mobile, desktop e web para visualizar dados de consumo de energia. • Monitoramento em tempo real. • Consumo mensal, semanal e diário. • Consumo em quilowatts-hora(kWh). • Indicador do custo de consumo.	Kit Online Energy monitor engage R$371.39, Euro a R$4,50, dia 16/05/2019.

Smappee	• Equipamentos monitor de consumo de energia elétrica. • Aplicação mobile e web para visualizar dados de consumo de energia elétrica. • Monitoramento em tempo real. • Consumo mensal, semanal e diário. • Consumo em quilowatts-hora(kWh). • Analise dos dados de consumo. • Indicador do custo de consumo. • Controle dos dispositivos inteligentes através da aplicação. • Personalização e automatização dos dispositivos inteligentes. • Conexão a Amazon Alexa, reconhecimento de comandos por voz.	Equipamento monitor de energia R$ 2.465,89. App gratuito.

Fonte: Elaborado pelo autor.

Considerando as funcionalidades dos sistemas relacionados, apresentados na Tabela 1, será feito um comparativo entre as funcionalidades do sistema Violet Energy, desenvolvido no decorrer deste projeto e os sistemas previamente citados na Tabela 1, visível na Tabela 2. Na Tabela 2 as siglas S1, S2, S3, S4 e S5 representam cada um dos sistemas respectivamente, Violet Energy, Open Energy Monitor (OPENENERGYMONITOR, 2017), Sense (SENSE, 2018), Engage (EFERGY, 2019), Smappee (SMAPPEE, 2019).

Tabela 2 – Comparativo de funcionalidades entre o sistema Violet Energy e os sistemas relacionados

Funcionalidade	**S1**	**S2**	**S3**	**S4**	**S5**
Equipamento monitor de consumo de energia elétrica	X	X	X	X	X
Conexão com a rede de Internet sem fio do equipamento	X	X	X		X
Gráfico com consumo total de todos equipamentos em kWh	X	X	X	X	X
Gráfico com o consumo descriminado por equipamento em kWh	X		X		X
Consumo Diário	X	X	X	X	X
Consumo Semanal		X	X	X	X
Consumo Mensal ou 30 dias corridos.	X	X	X	X	X
Indicador de custo		X	X	X	X

Informação de local e do último registro de consumo de um equipamento.	X		X	X	X
Informação *RealTime* de consumo de equipamentos ligados	X	X	X	X	X
Acesso online a essas informações de qualquer lugar com conexão à Internet	X	X	X	X	X
Conta Privada	X	X	X	X	X
Controle dos dispositivos inteligentes através da aplicação.					X

Fonte: Elaborado pelo autor.

O objetivo deste trabalho de graduação é desenvolver um sistema para o usuário conhecer seu consumo de energia elétrica dos últimos trinta dias em kWh (quilowatts-hora). Esse sistema foca principalmente no desenvolvimento de uma interface web simples e agradável que apresenta os dados de consumo, em kWh, do aparelho monitorado, e um equipamento protótipo monitor com sensor não invasivo, conectado à rede de Internet Wi-Fi, que transmite valores de corrente e potência. A intenção é que o usuário possa antecipar-se, ao saber quais equipamentos estão consumindo mais energia e, ao fazer uso dessas informações, gerenciar conscientemente o seu consumo de energia elétrica. Quanto aos objetivos específicos são os seguintes:

- Uso da metodologia Scrum para o desenvolvimento do sistema.
- Desenvolvimento de um sistema web para fornecer informações de consumo de energia elétrica ao Usuário.
- Desenvolver um protótipo de equipamento monitor de consumo de energia elétrica.
- Disponibilizar o sistema web em produção para o usuário poder acessá-lo de qualquer lugar com conexão à Internet.

Quanto a organização deste trabalho de graduação, apresenta-se da seguinte maneira: O capítulo 2 discorre sobre os requisitos do sistema e as tecnologias usadas. O capítulo 3 trata da modelagem do sistema e etapas importantes como o funcionamento do *Back-end*[3] do sistema web, o banco de dados noSQL e o circuito e software embarcado do equipamento protótipo. O capítulo 4 aborda o

[3] Neste projeto também é referido como *server*, é a parte da aplicação responsável por servir a aplicação *client.*

desenvolvimento e entrega do sistema sob a metodologia Scrum, etapa de *deploy*[4] da aplicação web, plano de testes e telas do sistema, e por fim, no capítulo 5 é abordado as considerações finais sobre o projeto.

[4] *Deploy* é o momento em que o software deste projeto é levado para o ambiente de produção.

2 PROJETO DO VIOLET ENERGY SYSTEM

O conteúdo deste capítulo apresenta um resumo do que será o projeto, os requisitos funcionais, requisitos não funcionais e todas as ferramentas envolvidas no desenvolvimento. O projeto intitulado Violet Energy System é composto de uma interface web que possibilita o monitoramento do consumo de energia elétrica e o equipamento protótipo que coleta corrente e potência e transmite os dados.

O software web, que estará hospedado na nuvem, pode ser acessado de qualquer lugar, por um usuário autenticado, que tem seus equipamentos monitorados. Nesse software é exibido, através de uma interface gráfica, os dados de monitoramento dos aparelhos do usuário.

O monitor de consumo de energia é um equipamento protótipo desenvolvido para captar corrente e potência de equipamentos de até 20A, através de um sensor não invasivo que enviará esses dados para a nuvem através de sua conexão à rede Wi-Fi.

Este sistema segue o conceito de Internet das Coisas, portanto ele pretende conectar equipamentos à Internet; no caso deste projeto, são os equipamentos que estejam consumindo eletricidade, os quais serão monitorados. Os dados gerados, desse monitoramento, serão usados para tornar mais cômodo o gerenciamento de energia elétrica pelo usuário e para isso o software web irá trazer todos esses dados formatados e apresentados de modo simples e agradável através de seu design gráfico.

2.1 Levantamento de requisitos

Um cliente, ao descrever um sistema que atenderá suas necessidades, está na verdade descrevendo os requisitos desse sistema e o processo de coletar, analisar e documentar estes requisitos é chamada de Engenharia de Requisitos, ou RE - *Requirements Engineering* (SOMMERVILLE, 2007). Os requisitos de um sistema podem ser divididos em duas categorias: requisitos funcionais e requisitos não funcionais.

2.1.1 Requisitos funcionais

Os requisitos funcionais de um sistema detalham entradas, saídas e exceções do mesmo, descrevendo, portanto, as funções reais do sistema, ou seja, o que ele faz de fato. (SOMMERVILLE, 2007). A Tabela 3 apresenta os requisitos funcionais deste projeto.

Tabela 3 – Requisitos funcionais do projeto

Identificação	Requisito Funcional	Prioridade
RF001	O *hardware* deve monitorar o consumo de energia elétrica.	Essencial
RF002	O sistema deve calcular o consumo de energia elétrica dos equipamentos monitorados dos últimos 30 dias em KWh.	Essencial
RF003	O sistema deve calcular o consumo médio em KWh de um equipamento por dia.	Essencial
RF004	O sistema deve permitir que o usuário acesse seu consumo de energia com login e senha.	Essencial
RF005	O sistema deve permitir visualizar o consumo de cada equipamento.	Essencial
RF006	O sistema deve permitir visualizar o consumo geral dos equipamentos.	Essencial
RF007	O sistema deve permitir visualizar o consumo dos equipamentos ligados.	Essencial
RF008	Quando clicado no nome de um equipamento mais informações sobre ele devem ser exibidas.	Essencial
RF009	Informações de consumo devem ser exibidas assim que o equipamento monitorado for ligado.	Essencial
RF010	Uma janela pop-up para logar deve surgir quando o usuário acessar a página web.	Desejável
RF011	Um gráfico de linha com o consumo dos últimos 30 dias do equipamento clicado deve ser exibido.	Essencial
RF012	A página web somente será acessível após o login.	Essencial

Fonte: Elaborado pelo autor.

2.1.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles que não estão diretamente associados as funções do sistema. Geralmente detalham características como segurança, *design*, armazenamento, entre outros, os quais não descrevem diretamente o que o sistema deve fazer ou entregar (SOMMERVILLE, 2007). A Tabela 4 apresenta os requisitos não funcionais deste projeto.

Tabela 4 – Requisitos não funcionais do projeto

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Um(a) usuário(a) deve ser capaz de compreender os gráficos.	*Design*	Essencial
RNF002	Um gráfico em formato de rosca deve apresentar o consumo em KWh dos equipamentos monitorados nos últimos 30 dias.	*Design*	Essencial
RNF003	O *hardware* deve transmitir os dados por Internet sem fio.		Essencial
RNF004	A página web não será acessível sem conexão à Internet.		Essencial
RNF005	A interface deve ser simples e apresentar todos os gráficos dinamicamente em apenas uma página.	*Design*	Essencial
RNF006	Os elementos na tela devem se adaptar aos limites da janela.	*Design*	Desejável
RNF007	O login do usuário deve ser feito em duas etapas: Autenticação e Autorização.	Segurança	Desejável

Fonte: Elaborado pelo autor.

2.2 Recursos e ferramentas

Esta seção contempla as ferramentas de programação e os conceitos necessários para o desenvolvimento e compreensão do sistema monitor de consumo de energia elétrica Violet Energy.

- **NodeMCU ESP8266**: Plataforma de desenvolvimento, com conversor USB-serial e regulador de tensão, essa plataforma permite fácil integração e prototipagem, já equipada para conectar-se à rede Wi-Fi (OLIVEIRA, 2017). Esse módulo será utilizado no projeto para ler os valores vindos do sensor,

executar o algoritmo embarcado para chegar aos valores de corrente e potência do equipamento monitorado e por fim transmitir para o banco de dados um objeto JSON, contendo valor de corrente e potência do equipamento.

- **Arduino IDE**: Esse *Integrated Development Environment* (IDE), em português, Ambiente de Desenvolvimento Integrado, permite codificar e instalar programas em placas e microcontroladores do Arduino. O Arduino IDE é oferecido para uso em duas modalidades: online, sendo necessário acessar a página oficial do Arduino para codificar, e a segunda para download, disponibilizado também na página oficial de acordo com o sistema operacional e após o download pode-se codificar na aplicação (ARDUINO, 2019). Neste projeto é feito uso da versão para download, específica para o sistema operacional Windows 10.
- **ESP8266Wifi**: Baseada e seguindo os padrões da biblioteca Arduino Wi-Fi *library*[5], esta biblioteca possibilita a conexão de uma placa ESP8266 a rede sem fio padrão *IEEE 802.11*[6] (ESP8266 ARDUINO CORE'S DOCUMENTATION, 2017). Para este trabalho de graduação esta biblioteca possibilita a conexão à Internet do equipamento monitor de consumo de energia elétrica.
- **EmonLiteESP**: Baseada na biblioteca EmonLib do projeto *Open Energy Monitor*, permite medição de corrente elétrica para placas ESP8266 (PÉREZ, 2017). Neste projeto esta biblioteca possibilita o equipamento medir corrente elétrica dos equipamentos monitorados pelo mesmo.
- **FirebaseArduino**: Esta biblioteca é capaz de conectar placas e microcontroladores Arduino ao banco noSQL do Firebase. Capaz de lidar com tipos puro da linguagem C e do Arduino, permitindo chamadas em linguagem C++ a API REST do Firebase e cuidando da interpretação de todo *JSON*[7] (FIREBASE-ARDUINO, 2015). Neste projeto esta biblioteca possibilita a conexão ao banco de dados do Firebase e envio de dados para o mesmo.
- **C++**: Esta linguagem de programação fortemente tipada, pode seguir diversos paradigmas de programação como, estruturado, orientado a objeto, entre outros, sendo portável para os mais distintos *hardwares*. Esta linguagem é

[5] Em português: Biblioteca.
[6] Família de protocolos definidos pelo grupo IEEE (*Institute of Electrical and Eletronics Engineers*), o padrão 802.11 é o definido para rede sem fio mais conhecida como Wi-Fi (TELECO, 2012).
[7] JSON acrônimo para JavaScript Object Notation é um pacote de dado simples e leve, formatado para ser inteligível por humanos (JSON, 2007).

compilada diretamente para a linguagem de máquina, portanto sua execução tem alta performance (ALBATROSS, 2012). Neste projeto esta linguagem de programação será usada para desenvolver o sistema embarcado do microcontrolador.

- **Firebase *Realtime* Database**: Este banco de dados é noSQL e hospedado na nuvem, portanto, permanece disponível mesmo quando a aplicação *client* está offline. Sincronizável, *Realtime* (tempo real), para todas aplicações nele conectadas. Este banco conversa com as mais diferentes plataformas e seus dados são armazenados como *JSON* através da API do Firebase, a qual permite leitura, escrita e estruturação destes dados de acordo com as regras de segurança definidas pelo usuário do banco. Com uma licença gratuita de até 1G na modalidade *RealTime* entre outros, ele permite o uso desde que não ultrapasse os limites definidos na licença gratuita (FIREBASE, 2016). Este banco noSQL, será usado neste projeto para armazenar os dados colhidos do sistema monitor de consumo de energia elétrica e alimentar a página web, dentro dos limites gratuitos da licença.
- **Visual Studio Code**: Este é editor de código da Microsoft, é leve e desenvolvido para criação de projetos web e aplicações cloud, suportando linguagem como: JavaScript, C, entre outras; com formatação de sintaxe, este editor facilita o processo de codificação. Nele também está incorporado um controlador GIT para facilitar a conexão a repositórios (VISUAL STUDIO CODE, 2019). Este Editor será usado neste projeto para o desenvolvimento da página web, que irá expor os dados de monitoramento do sistema monitor de consumo.
- **Node.JS**: Node.js é um *run-time*[8] para JavaScript no *server*. A linguagem assíncrona JavaScript da qual seria dependente de uma página web para ser executada. Devido ao *run-time* do Node.js ter integrado um *event-loop*[9], o assíncrono no JavaScript é levado para o *Back-end,* agregando performance (NODE.JS, 2016). Neste projeto de conclusão de curso o *Back-end* será

[8] Em português: Ambiente de Execução.
[9] Node.js é *single thread* (thread única), o *loop* de eventos é o que permite o Node.js operar rotinas sem bloquear a *thread* principal (NODE.JS, 2016).

codificado utilizando Node.JS. O *Back-end,* quando for requisitado em sua API, prepara os dados para que sejam entregues ao *Front-end*[10].

- **NPM:** É um Gerenciador de pacotes e dependências, que nasceu com o intuito de possibilitar o compartilhamento de bibliotecas, *frameworks*[11], entre outros, para toda comunidade JavaScript (NPM, 2019). Neste projeto, o NPM atuará no gerenciamento das dependências do projeto, possibilitando a instalação de qualquer biblioteca ou *framework* que for necessária para o funcionamento do software web de monitoramento de consumo de energia elétrica.
- **Express**: É um *framework* que implementa a maioria das soluções para trabalhar com web, como: Roteamento de requisições, *middlewares*, configurações de definição de portas, entre outras implementações. Desenvolvido justamente para o ambiente Node.JS, o Express é um dos *frameworks* mais populares para o desenvolvimento web com Node.JS (MDN WEB DOCS, 2019). Neste projeto, o Express será usado para as codificações da API do *Back-end*, *middleware* e define a porta de comunicação com o *server*.
- **React**: É uma biblioteca criada pelo Facebook para construir interfaces visuais de maneira simples e baseada em componentes. O React facilita o uso de bibliotecas de componentes, já preparados para serem exibidos e personalizados, e a sua sintaxe torna o código mais agradável para o desenvolvimento do *Front-end.* (FACEBOOK, 2016). Neste projeto, o *Front-end* será desenvolvido utilizando React, o que permitirá ao usuário visualizar os dados de consumo de energia elétrica através de sua interface.
- **Axios**: Um *client* http para navegadores e Node.js, que permite fazer requisições utilizando métodos http de modo simples, entre outras funcionalidades (AXIOS, 2019). Neste projeto, o *client* é usado com a finalidade de possibilitar a comunicação do *Front-end* com *Back-end*.
- **React *Router***: É um *framework* para criar rotas de uma página web, antes da renderização de componentes, como parte da inicialização da aplicação (REACT TRAINING, 2017). Neste projeto este *framework* será usado para a criação de rotas no *client*, possibilitando renderização dos componentes adequados de acordo com a rota.

[10] Também referido como *client* neste projeto, é a interface de interação com o usuário.
[11] Uma *Framework* une necessidades em comum e oferece funcionalidades genéricas para serem usadas (SUAVÉ, 2002).

- **Material-UI**: Uma biblioteca *open-source* de componentes React que segue o guia de design do Google (MATERIAL-UI, 2014). Estes componentes serão usados para exibir elementos visuais como o *header* e textos no *Front-end*.
- **React Charts.JS 2**: Biblioteca de componentes React para exibir gráficos (AYERST, 2019). Neste projeto esta biblioteca será responsável por possibilitar a criação de componentes gráficos, onde será exibido os dados de consumo do sistema de monitoramento.
- **Firebase Authentication**: O Firebase fornece serviços de autenticação para seus usuários, seguindo o padrão *OAuth2*[12]. O Firebase oferece um SDK para fazer o fluxo de login com o serviço do Google Gmail, Facebook, entre outros, de modo simples, deixando que o provedor de serviço de login cuide da encriptação e segurança da senha do usuário (FIREBASE, 2016). Neste projeto esse SDK do Firebase será usado para permitir que o usuário faça login na aplicação com sua conta Gmail, agindo como a camada de autenticação da aplicação.
- **Git**: Sistema para controle de versionamento de software, o qual permite acompanhar o desenvolvimento de uma aplicação, conectando a repositórios como o GitHub. O Git permite identificar e acompanhar as modificações feitas no repositório local e enviá-las para um repositório online (GIT, 2019). Neste projeto essa ferramenta será usada para acompanhar a evolução do desenvolvimento de todo o sistema, bem como criar uma redundância do código no repositório GitHub.
- **GitHub**: Uma plataforma online gratuita, para hospedar código e garantir o versionamento do mesmo, também possibilita a colaboração no desenvolvimento através de suas funcionalidades que permite criar múltiplas instâncias de um código, para que uma ou várias alterações possam ser feitas sem afetar o código principal (GITHUB, 2016). Esta plataforma será usada para hospedar o código fonte deste projeto, nela poderá ser visualizada as modificações feitas no código e garantir a redundância do mesmo.

[12] Padrão de autenticação o qual permite delegar a outros serviços a manipulação de login e senha (OAUTH, 2007).

- **Google Cloud Platform (GCP):** Uma plataforma de computação em nuvem, armazenamento, *Big Data*[13] e *Machine Learning*[14], com o GCP é possível fazer *Build*, teste e *Deploy* na nuvem com segurança e escalabilidade (GOOGLE, 2019). Neste projeto o GCP disponibilizará o sistema web.
- **Systemd:** É um gerenciador de sistema e serviços do Linux, esse sistema oferece capacidade de paralelização de serviços, iniciação de serviços, rastreamento de processos, entre outros tipos de serviços, oferecidos pelo Systemd (SYSTEMD, 2018). Esse gerenciador será responsável por manter o sistema web ativo, reiniciá-lo caso necessário, para garantir sua disponibilidade sem precisar subir a página web manualmente pela máquina virtual.
- **Apache2:** Servidor HTTP Apache é um servidor web que auxilia na comunicação e disponibilização de serviço web (APACHE, 2019). Neste projeto o servidor apache2 será responsável por servir os arquivos estáticos e possibilitar a comunicação entre *client* e o *server,* através de um proxy reverso que faz o roteamento da requisição, também é responsável por sempre apontar para o arquivo principal quando houver uma mudança de página.
- **API REST**: *Application Programming Interface* (API), em português, Interface de Programação de Aplicação, é uma interface que possibilita um software acessar um serviço ou sistema (FOLDOC, 1995). *Representational State Transfer* (REST), em português, Transferência de Estado Representacional, são conjunto de critérios de design para aplicações em rede, orientadas a recursos, como propõem Richardson; Ruby (2007). Portanto API REST usa os critérios REST para permitir acesso simples aos recursos endereçáveis de uma aplicação. Neste trabalho de graduação será feito uso de uma API REST para permitir a comunicação entre o *client* e o *server* do software web.
- **Armazenamento em nuvem**: O armazenamento em nuvem é a capacidade de armazenar dados na nuvem, a qual representa servidores remotos gerenciados e preparados para receber dados de um fornecedor, esse tipo de armazenamento pode ter custos adaptados apenas ao espaço usado, portanto, a nuvem permite que não seja necessário ter um *hardware* próprio, conhecimento sobre gerenciamento desse tipo de equipamento ou custos de

[13] *Big Data* é uma grande volumetria de dados variados e crescente (GARTNER, 2016).
[14] Aprendizado de máquina, do inglês *Machine Learning*, é um ramo da inteligência artificial que envolve o desenvolvimento e treinamento de um algoritmo para aprender padrões com informação (GARTNER, 2016).

manutenção (AMAZON, 2017). Neste projeto os dados de consumo de energia elétrica ficam armazenados na nuvem do Firebase.

- ***Middleware***: É um software intermediário que conecta dois softwares, ele canaliza o fluxo de dados entre ambos como se fosse uma tubulação (REDHAT, 2018). O *middleware* deste projeto é responsável por filtrar todas as solicitações que chegam e canalizá-las para as saídas corretas.
- **NoSQL**: Termo utilizado para banco de dados não relacionais, apresentam modelos de dados diferentes como, chave e valor, gráficos, entre outros. São conhecidos pela simplicidade na implementação, alta performance e baixa latência (AMAZON, 2018). Nesse projeto o banco utilizado para armazenar os dados do consumo de energia elétrica é noSQL, é essencial devido a sua simplicidade para receber dados e a capacidade de retorná-los dados com velocidade.

3 MODELAGEM

Neste capítulo será feita a documentação do projeto, que segue os padrões da linguagem de modelagem *Unified Modeling Language* (UML), para modelar os casos de uso e os diagramas de sequência. Segundo Pressman (2011), a UML é uma linguagem para documentar e padronizar um projeto de software.

As documentações feitas neste capítulo têm o intuito de padronizar e facilitar a compreensão do projeto através dos diagramas.

3.1 Casos de uso

Em um caso de uso estão os cenários de uso do sistema, exibindo do ponto de vista do usuário, as funcionalidades do sistema, portanto nele estão presentes os requisitos funcionais do sistema (PRESSMAN, 2011), esses casos de usos podem ser representadas de várias maneiras, dentre elas, está o diagrama de caso do uso que demonstra visualmente o ator do sistema, como um boneco, as funcionalidades como retângulos e balões com descrições dentro, conectados por traços os quais representam as relações (SOMMERVILE, 2007).

Os atores que interagem com o sistema são: O Usuário (Utilizador do sistema), API do *Back-end*, API do banco de dados *RealTime*. A API do *server* se trata do sistema REST[15] o qual acessa o banco de dados.

- **Usuário** é o ator que representa os utilizadores do sistema. Um usuário pode visualizar todos os dados disponibilizados pelo sistema, já que este usuário(s) tem seus equipamentos monitorados. Somente ele terá acesso a ferramenta, devido a sensibilidade das informações ali exibidas.
- **API de Autenticação** permite que o usuário através da API de autenticação do Firebase, que se conecta a API do Google, faça o login na aplicação através de sua conta de e-mail do Gmail.
- **API do *Back-end*** representa a API que permite obter e calcular dados de consumo de energia elétricas contidos no banco de dados do Firebase.
- **API do Banco de dados *RealTime*** representa a API que oferece o serviço de banco de dados noSQL, na qual está condido os dados de consumo de

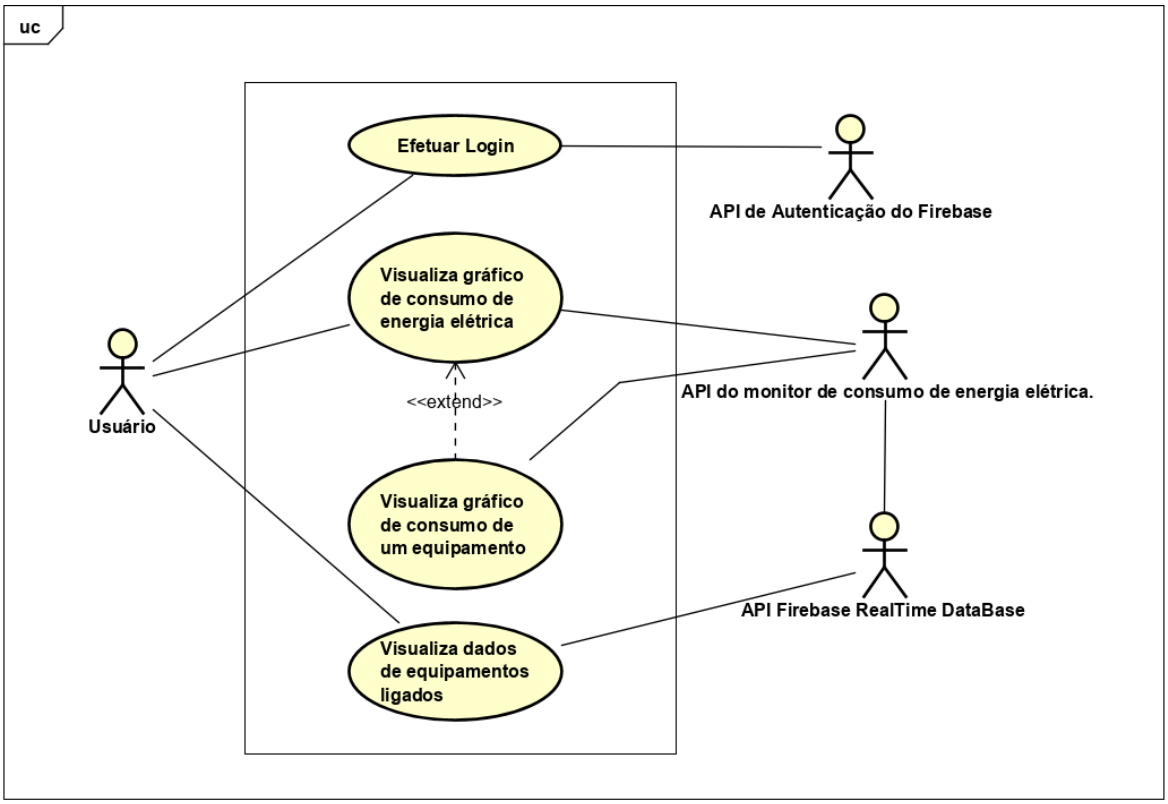
[15] REST (*Representational State Transfer*) define um conjunto de princípios arquiteturais os quais podem ser utilizados para projetar *web services* focados em recursos do sistema

energia elétrica. Este banco de dados é alimentado pelo equipamento monitor de consumo de energia, que envia dados no formato *JSON* via chamadas *REST* ao banco.

3.2 Documentação dos casos de uso

A Figura 4 apresenta o caso de uso da página Web.

Figura 4 – Diagrama de caso de uso do sistema Web



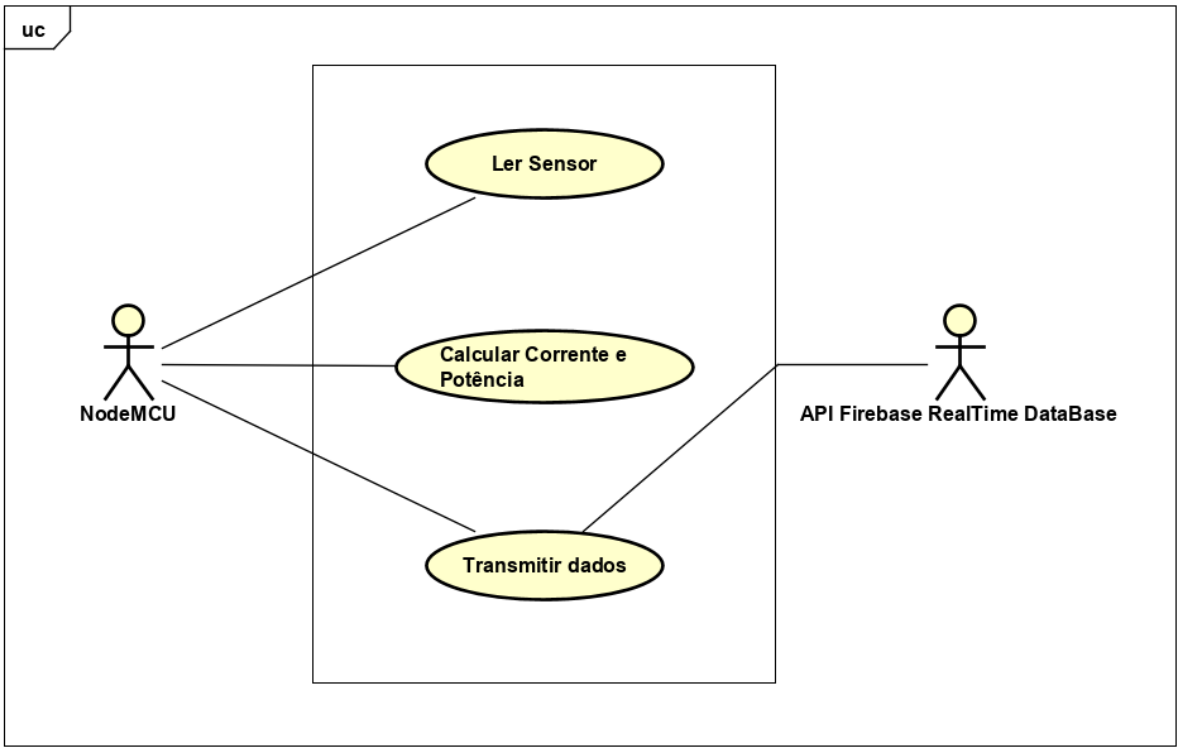
Fonte: Elaborado pelo autor.

A Figura 5 apresenta o caso de uso do equipamento monitor de consumo de energia elétrica.

3.2.1 Documentação dos Casos de Uso

Cada funcionalidade dos diagramas de casos de uso será descrita da Tabela 5 à Tabela 9.

Figura 5 – Diagrama de caso de uso do equipamento monitor de consumo de energia elétrica



Fonte: Elaborado pelo autor.

Tabela 5 – Caso de uso "Efetuar login"

Nome do caso de uso	Efetuar login
Atores envolvidos	Usuário, API do Firebase, *Front-end* do sistema
Objetivo	Este caso de uso descreve os passos do login para o usuário do sistema
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O acessa a página web	
	2. Uma tela pop-up aparece solicitando que o usuário efetue login, com sua conta do Google
3. O usuário efetua o login	
	4. O Google autentica o usuário
	5. O *Front-end* do Sistema autoriza

Validações	Para que o login seja efetuado, o usuário passa por duas etapas, na primeira ele deve ser autenticado pelo Google o qual retorna se este usuário realmente tem login e senha válidos. Na segunda etapa o *Front-end* verifica se o usuário está autorizado a acessar o sistema verificando o login que a API do Google retorna

Fonte: Elaborado pelo autor.

Tabela 6 – Caso de uso "Visualizar gráfico de consumo de energia elétrica"

Nome do caso de uso	Visualizar gráfico de consumo de energia elétrica
Atores envolvidos	*Front-end*, API do Sistema consumidor de energia elétrica, API do banco de dados *RealTime* do sistema
Objetivo	Este caso de uso descreve os passos para exibição do gráfico em forma de rosca no *Front-end*
Pré-condição	Ter efetuado o Login
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O *Front-end* faz uma requisição a API do *Back-end* no caminho "/piechartdata"	
	2. O *middleware* do *Back-end* verifica a origem da requisição e a autoriza a seguir em frente
	3. O *Back-end* realiza uma chamada a API do bando de dados para trazer todos os dados de consumo dos últimos 30 dias
	4. O *Back-end* calcula qual o consumo de energia elétrica em quilowatts-hora e retorna esses dados ao *Front-end*
5. O *Front-end* exibe esses dados em um gráfico em formato de rosca	
Validações	Toda chamada feita ao *Back-end* passa pelo *middleware* que, antes de encaminha a chamada para sua respectiva rota, verifica no *header* da requisição se a origem da chamada é permitida, caso não seja ela é então barrada

Fonte: Elaborado pelo autor.

Tabela 7 – Caso de uso "Visualizar gráfico de consumo de um equipamento"

Nome do caso de uso	Visualizar gráfico de consumo de um equipamento
Atores envolvidos	Usuário, *Front-end*, API do Sistema consumidor de energia elétrica, API d DB *RealTime* do sistema.
Objetivo	Este caso de uso descreve os passos para exibição do gráfico de linha no *Front-end*
Pré-condição	Ter efetuado o Login.
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O usuário do sistema clica sobre o nome de um equipamento, no gráfico de rosca	
	2. O *Front-end* faz uma requisição à API do *Back-end* na rota "/linechartdata"
	3. O *middleware* do *Back-end* verifica a origem da requisição e a autoriza a seguir em frente
	4. O *Back-end* realiza uma chamada a API do banco de dados para trazer todos os dados de consumo dos últimos 30 dias
	6. O *Back-end* calcula qual o consumo de energia elétrica em quilowatts-hora por dia, do equipamento, e retorna esses dados ao *Front-end*
	7. O *Front-end* exibe esses dados em um gráfico de linhas
Validações	Toda chamada feita ao *Back-end* passa pelo *middleware* que, antes de encaminha a chamada para sua respectiva rota, verifica no *header* da requisição se a origem da chamada é permitida, caso não seja ela é então barrada

Fonte: Elaborado pelo autor.

Tabela 8 – Caso de uso "Visualiza dados de equipamentos ligados"

Nome do caso de uso	Visualiza dados de equipamentos ligados
Atores envolvidos	*Front-end* do sistema, API do banco de dados *RealTime*

Objetivo	Este caso de uso descreve os passos para ser exibido os dados monitorados de um equipamento que esteja ligado, no *Front-end*
Pré-condição	Ter efetuado o Login
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O *Front-end* faz uma requisição a API do banco para retornar os dados do horário atual, que estejam sendo inseridos no banco de dados	
	2. O banco de dados do Firebase retorna todo dado que esteja sendo inseridos, a partir do horário predefinido pelo *client*
3. O *Front-end* do projeto exibe os que estão sendo inseridos no banco de dados, ordenados em uma lista lateral	
Validações	A API do banco de dados noSQL *RealTime* do Firebase faz uma validação, verificando se o domínio que realiza a chamada é permitido

Fonte: Elaborado pelo autor.

Tabela 9 – Caso de uso "Monitor de Consumo de energia elétrica"

Nome do caso de uso	Diagrama de caso de uso do equipamento monitor de consumo de energia elétrica.
Atores envolvidos	NodeMCU, Software embarcado, API *RealTime* Firebase.
Objetivo	Monitorar o consumo de energia elétrica de um equipamento através de leituras de um sensor, calcular corrente e potência, e transmitir esses dados para o banco de dados.
Prioridade de desenvolvimento	Essencial
Ações do ator	**Ações do Sistema**
1. O NodeMCU capta informações vindas do sensor	
	2. Calcula corrente e potência, e converte os dados em um *JSON*.

	3. Faz uma inserção no banco de dados encaminhando o *JSON* criado.
Validações	O software verifica se a leitura é igual a zero, se for, não envia para o banco.

Fonte: Elaborado pelo autor.

3.3 Diagrama de sequência

O diagrama é responsável por mostrar a sequência de eventos desencadeadas por uma ação no sistema (PRESSMAN, 2011). São essenciais para compreender um fluxo de ações, os quais podem não ficar muito claros no diagrama de caso de uso (PRESSMAN, 2011). Objetivando expor como a comunicação do serviço web e do equipamento monitor se dá, um diagrama de sequência demonstrará o desencadear das ações dos atores no sistema.

3.3.1 Documentação dos diagramas de sequência

Na Figura 6 está o diagrama do fluxo de eventos ocorridos quando o equipamento monitor (ator nodeMCU) está em funcionamento.

Segue a descrição de cada um dos eventos desencadeados, enumerados de 1 a 1.2.:

1. **sensorRead()**: O equipamento NodeMCU lê os dados vindos do sensor;

1.1. **calculatePower()**: O Sistema realiza cálculo de corrente e potência com os dados vindos do sensor;

1.2. **sendData()**: O sistema envia um *JSON* contendo as informações de corrente e potência para a API Firebase *RealTime* DataBase.

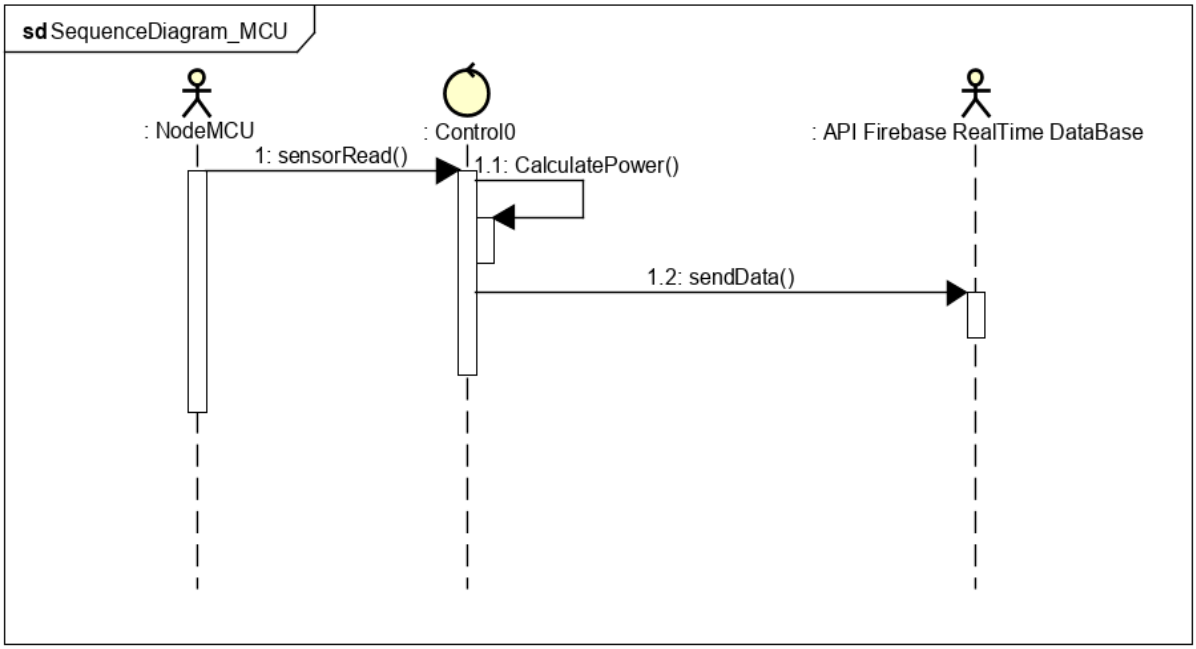
Na Figura 7 está o diagrama de sequência que demonstra todo o fluxo de eventos da página web, quando acessada pelo usuário. Anteriormente ao diagrama na Figura 7, o fluxo de eventos enumerado é brevemente explicado para compreensão de cada passo realizado no sistema.

Segue a descrição de cada um dos eventos desencadeados, enumerados de 1 a 4:

1. **login()**: O usuário realiza o login no sistema com uma conta Gmail válida.

1.1. **authentication()**: O usuário é autenticado pela API de autenticação Firebase.

Figura 6 – Diagrama de sequência do equipamento monitor de consumo de energia elétrica



Fonte: Elaborado pelo autor.

1.1.1. **userData()**: API de autenticação retorna para o *Front-end* os dados do usuário.

1.1.1.1. **GET pieChartData()**: Uma requisição para o *Back-end* é feita para obter os dados de consumo de todos os equipamentos, se o usuário for autorizado.

***Back-end Middleware*()**: O *middleware* do *Back-end* intercepta todas as requisições que chegam e verifica se a origem da requisição é autorizada.

1.1.1.1.1. **onChildAdded()**: As requisição de dados do gráfico de rosca, autorizada pelo *middleware,* faz uma chamada à API do banco, pedindo pelo dado de consumo dos últimos 30 dias.

1.1.1.1.1.1. **JSONArray()**: A API do Firebase responde com um objeto do tipo *JSON,* com todos os dados dos últimos 30 dias.

1.1.1.1.1.1.1. **pieConsumptionData()**: A API do *Back-end* responde um dicionário de dados, com o nome dos equipamentos e o consumo total desses nos últimos 30 dias.

1.1.1.1.2. **code403()**: O *Back-end* responde com 403 para qualquer requisição de uma origem não autorizada.

1.1.1.2. **errorPage()**: A página de erro é exibida para o usuário que não é autorizado a acessar o sistema.

2. **hardwareLabelClick()**: O usuário do sistema clica em um *hardware*;

2.1. **GET lineChartData()**: Uma requisição para o *Back-end* é feita para obter os dados de consumo, nos últimos 30 dias, do equipamento clicado.

2.1.1. **onChildAdded()**: A requisição de dados sobre um equipamento, autorizada pelo middware do *Back-end,* faz uma chamada na API do banco pedindo pelo dado de consumo dos últimos 30 dias.

2.1.1.1. **JSONArray()**: A API do Firebase responde com um objeto do tipo JSON, com todos os dados dos últimos 30 dias.

2.1.1.1.1. **hardwareConsumptionData()**: A API *Back-end* responde um dicionário de dados, com nome do equipamento, dados do consumo e informações deste, nos últimos 30 dias.

2.1.2. **code403()**: O *Back-end* responde com 403 a uma requisição de uma origem não autorizada.

3. ***RealTime*Data()**: O *Front-end* faz uma requisição direta a API do banco para estabelecer uma conexão, em tempo real, solicitando dados que estejam sendo inseridos daquele momento em diante.

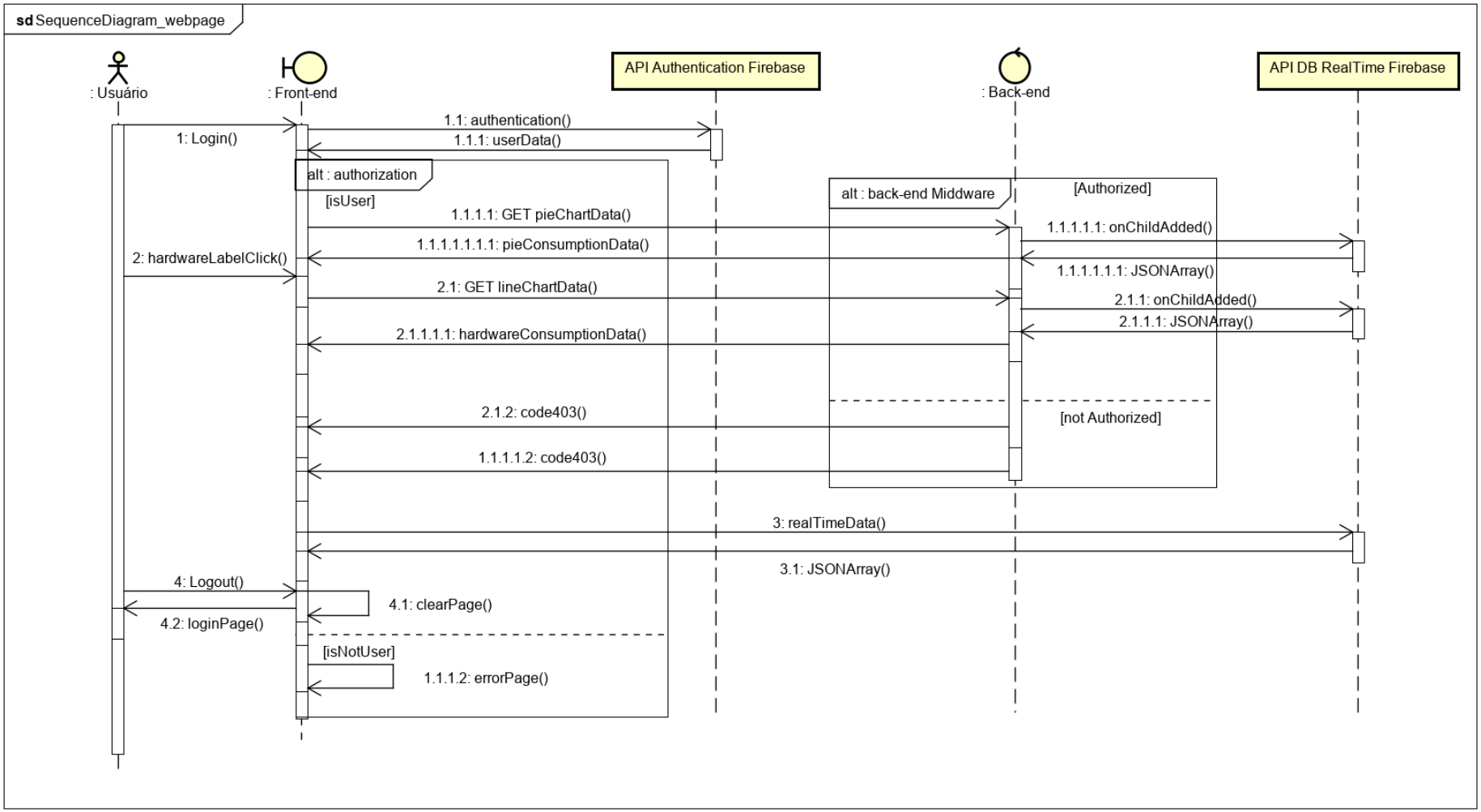
3.1. **JSONArray()**: O banco passa a enviar ao *Front-end* objetos do tipo *JSON,* com os dados que estiverem sendo inseridos.

4. **logout()**: O usuário desloga da página web.

4.1. **clearPage()**: O *Front-end* desabilita todos os componentes gráficos da tela.

4.2. **loginPage()**: O *Front-end* redireciona o usuário para página de login.

Figura 7 – Diagrama de sequência da página web



Fonte: Elaborado pelo autor.

3.4 *API* do *back-end*

A estrutura do *Back-end* do serviço de web segue critérios de design RESTful, recebendo requisições através de sua API e roteando estas para os respectivos terminais. Por sua vez, cada *endpoint* é responsável por estabelecer uma conexão com o banco de dados, manipular dados, e os devolver a quem o solicitou. É importante mencionar que o *server* deste projeto tem uma camada de proteção devido ao *middleware,* o qual intercepta toda requisição para verificar se sua origem é autorizada.

A Tabela 10 apresenta as rotas disponíveis na API do *Back-end* e uma breve descrição sobre elas.

Tabela 10 – Recursos da API do *Back-end*

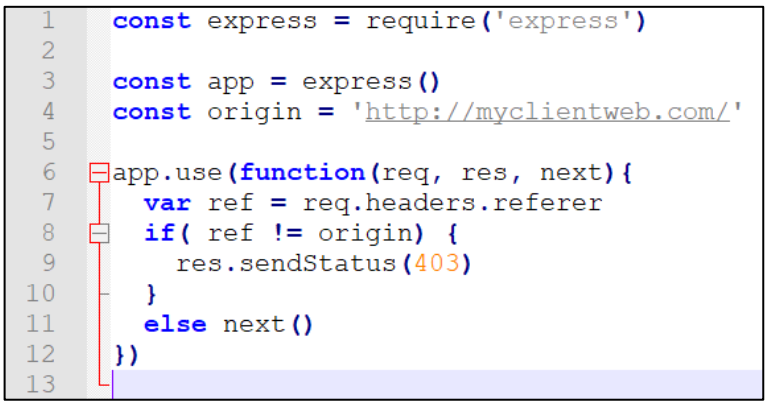
Recursos disponibilizados pela *API* de usuários			
Endereço	**Método**	**Autenticação**	**Descrição**
/api/piechartdata	GET	Sim	Obtém dados de consumo geral de todos os equipamentos nos últimos 30 dias e kWh.
/api/linechartadata	GET	Sim	Obtém dados de consumo dos últimos 30 dias de um equipamento em kWh.

Fonte: Elaborado pelo autor.

3.4.1 *Middleware*

Neste projeto o *middleware* atua como um filtro pelo qual todas as requisições feitas à API devem passar antes de seguirem para o roteador, verificando o *header* de toda requisição em busca do endereço de onde partiu a mesma. Caso esteja dentro das especificações exigidas pelo *middleware* a requisição pode seguir em frente. No caso deste projeto o *middleware* atua como uma proteção para que a API não esteja exposta, portanto apenas o *Front-end* deste projeto tem permissão de fazer chamadas para o *server*, qualquer requisição de uma origem distinta da permitida é barrada. A Figura 8 exibe a codificação do *middleware* deste projeto.

Figura 8 – *middleware* do *Back-end*



```
1  const express = require('express')
2
3  const app = express()
4  const origin = 'http://myclientweb.com/'
5
6  app.use(function(req, res, next){
7    var ref = req.headers.referer
8    if( ref != origin) {
9      res.sendStatus(403)
10   }
11   else next()
12 })
13
```

Fonte: Elaborado pelo autor.

Como é possível observar na Figura 8 o *framework* Express oferece um *middleware* em poucas linhas de código, posicionando o “app.use” (linha 6) no início do documento principal da API do *server*, toda requisição feita a ele deve obrigatoriamente passar pelo *middleware*, então ele verifica se o endereço da requisição “req.headers.referer” é diferente da origem “http://myclientweb.com/”, caso esta condição seja verdadeira a requisição é barrada e o código 403(Proibido), é devolvido ao *client*, caso a condição seja falsa a requisição é autorizada a seguir em frente pela função “next()”.

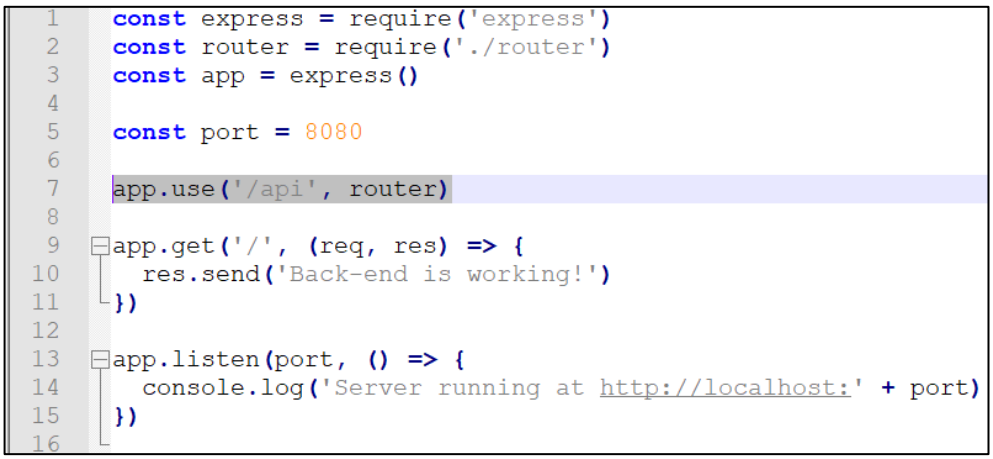
3.4.2 *Router* do *server*

O roteador nada mais é que a estruturação de caminhos para um ou mais terminais do sistema, os quais irão responder a solicitações feitas a ele de um *client* qualquer. Estas solicitações são compostas de um caminho, uma *URI*[16], e um método *HTTP*, o qual vai dizer o tipo de operação que será realizada naquele caminho.

O roteador deste projeto utiliza a *framework* Express para rotear suas chamadas, para melhor compreender seu funcionamento e codificação as figuras de 9 a 11 demonstram como foi elaborado.

[16] URI (Uniform Resource Identifier): É o caminho ou identificação para um recurso (INTERNET SOCIETY, 2005).

Figura 9 – API do *Back-end*

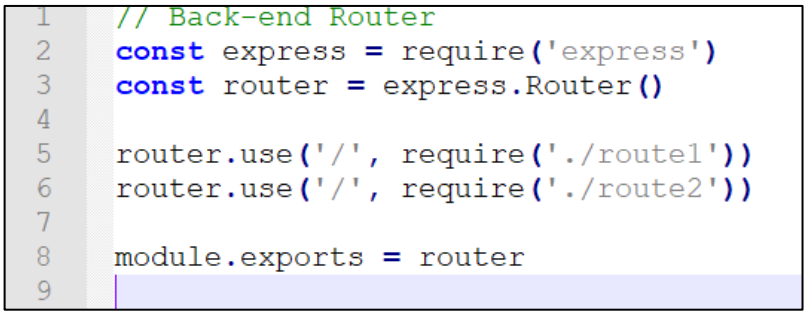


```
1  const express = require('express')
2  const router = require('./router')
3  const app = express()
4
5  const port = 8080
6
7  app.use('/api', router)
8
9  app.get('/', (req, res) => {
10   res.send('Back-end is working!')
11 })
12
13 app.listen(port, () => {
14   console.log('Server running at http://localhost:' + port)
15 })
16
```

Fonte: Elaborado pelo autor.

Na Figura 9 é possível ver toda a *API* do *Back-end*, especificadamente na linha 7, se encontra o *middleware* do roteador Express, portanto toda requisição "/api" navegará para o *Router* que encontra no diretório "./router".

Figura 10 – Roteador do *Back-end*



```
1  // Back-end Router
2  const express = require('express')
3  const router = express.Router()
4
5  router.use('/', require('./route1'))
6  router.use('/', require('./route2'))
7
8  module.exports = router
9
```

Fonte: Elaborado pelo autor.

Na Figura 10, é possível ver a codificação do "express.Route()", um mini aplicativo para roteamento modular da *framework* Express. A chamada para a *API* encontra o *Router* presente na Figura 10, o qual irá rotear a chamada mais a dentro, para algum dos caminhos especificado em "router.use()". Supondo que um *client* qualquer tenha feito uma chamada do tipo GET em "/api/route1", o *Router* encaminha a requisição para o *endpoint* "route1".

Figura 11 – Terminal do Roteador

```
1  // EndPoint n1
2  const express = require('express')
3  const router = express.Router()
4
5  router.get('/route1', (req, res)=>{
6      res.send("Hello World from route n1!")
7  })
8
9  module.exports = router
10
```

Fonte: Elaborado pelo autor.

A Figura 11 mostra um exemplo de *endpoint*[17] de uma rota, o qual responderá um texto à requisição do tipo GET feita em seu endereço. A requisição feita no endereço “/api/route1”, é roteada para “/route1” e obtém, como resposta, o conteúdo da função “res.send()”.

3.4.3 Terminais do *server*

Como, brevemente demonstrado na Tabela 10 de Recursos da API, cada *endpoint* do *Back-end* retorna ao *client* dados estruturados e formatados. Esses dados são preparados para que o *Front-end* não necessite realizar cálculos ou muitas manipulações, deixando para que o *server*, que tem maior capacidade de processamento, se encarregar de executar essas rotinas de preparação dos dados.

3.4.3.1 PieChartData

Este *endpoint* ao receber uma requisição se encarrega de estabelecer uma conexão com a API do banco de dados do Firebase, onde estão hospedados os dados de consumo, essa conexão, que também é uma requisição, pede pelos dados dos últimos 30 dias e o banco retorna uma *array* de objetos *JSON*. Ao receber essa *array* o *Back-end* realiza um cálculo, por equipamento, para chegar ao consumo de energia elétrica em KwH(quilowatts-hora). A Equação (1) é usada para chegar ao valor de quilowatts-hora:

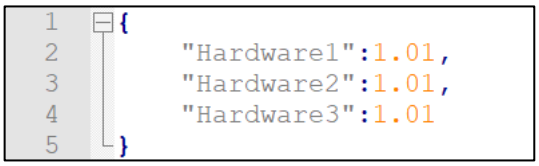
Consumo = (potência em watt x número de horas ligado) / 1000 = total em KwH. (1)

[17] Terminal de um sistema.

Entretanto uma mudança na equação (1) fez-se necessária, para possibilitar o cálculo com os dados vindos do banco. Devido os dados do sensor serem registrados no banco por minuto, ao invés de fazer a divisão por 1000(mil), para horas que o equipamento passou ligado, ela foi feita por 60000(sessenta mil), para os minutos que o equipamento passou ligado.

Ao final da rotina de cálculo do consumo, este *endpoint* retorna para o *client* um dicionário, uma estrutura de dados de chave e valor, sendo as chaves os nomes dos equipamentos e o valor o consumo mensal desses, a Figura 12 apresenta o modelo dos dados retornados pela chamada em “/api/piechartdata”.

Figura 12 – Dicionário PieChartData

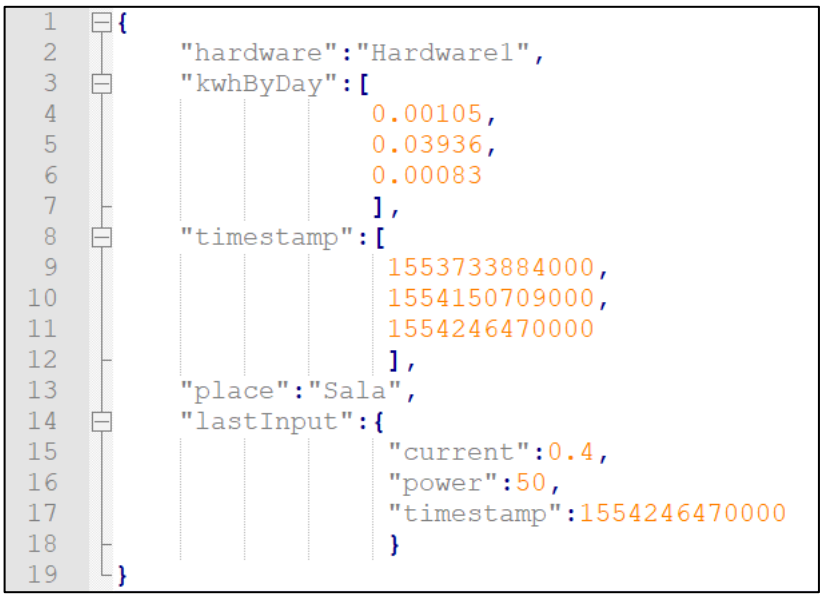


Fonte: Elaborado pelo autor.

3.4.3.2 LineChartData

LineChartData é um *endpoint*, que assim como o PieChartData, realiza uma chamada ao banco para recuperar os dados dos últimos 30 dias e efetuar o cálculo de consumo de energia, chegando a um valor em quilowatts-hora, porém este *endpoint* irá retornar dados somente sobre um equipamento, o que significa que ele espera que o *client* informe o nome do equipamento, para isso a URI de requisição para LineChartData recebe um parâmetro adicional, o nome do equipamento a ser retornado, ficando “/api/linechartdata/hardware”. Tendo o nome do equipamento o *server* realiza o cálculo de consumo por dia e devolve um dicionário com todas as informações necessárias sobre o *hardware* especificado para o *Front-end*. A Figura 13 exibe o modelo de dados retornado pelo *endpoint* “/linechartdata”.

Figura 13 – Dicionário LineChartData



Fonte: Elaborado pelo autor.

Os dados retornados atendem as necessidades do *client* para popular sua interface visual, sobre detalhes do equipamento. Como é possível observar na Figura 13, *hardware* (equipamento), *place* (local), *lastInput* (última medição) e listas contendo *timestamp*[18] e *kwhByDay* (consumo), são as informações retornadas para que o cliente preencha texto com mais detalhes sobre o equipamento e um gráfico de linhas sobre o consumo diário do mesmo.

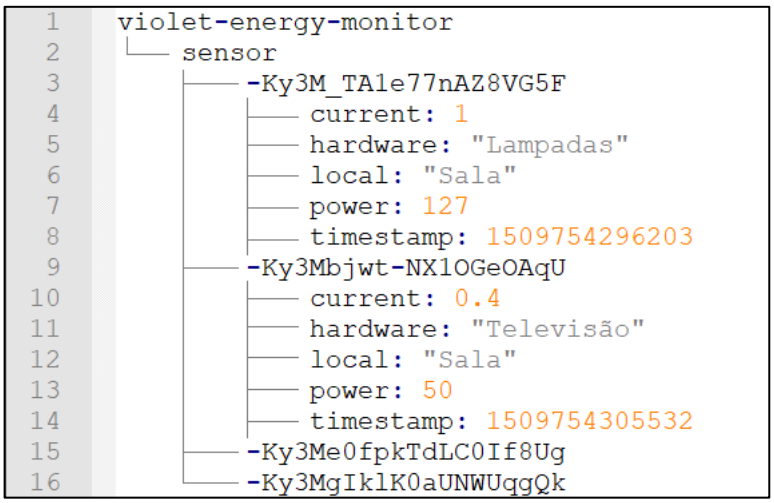
3.5 Banco de dados

O banco de dados deste projeto de graduação foi pensado para atender as necessidades de um projeto IoT e Web, portanto ao decorrer desta seção será desenvolvido sua arquitetura, requisitos e tecnologias para melhor entender o seu funcionamento. As tecnologias envolvidas nessa seção é o espaço gratuito, até 1 Gigabyte, disponibilizados pela nuvem do Firebase *RealTime* e sua API para manipular os dados ali contidos, apresentados no formato JSON. Este banco noSQL permite uma simples inserção e rápida recuperação de dados na estrutura de chave e valor (FIREBASE, 2019).

A Figura 14, exibe como os dados deste projeto estão armazenados no banco de dados noSQL do Firebase.

[18] Um registro de tempo.

Figura 14 – Exemplo de dados inseridos no Banco



Fonte: Elaborado pelo autor.

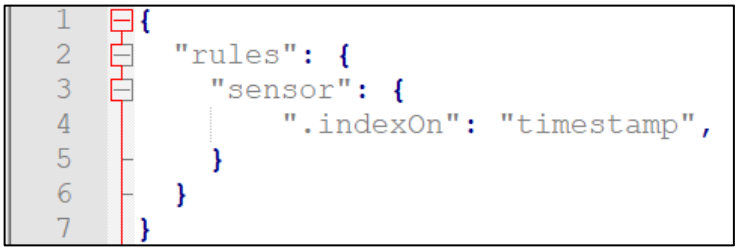
Como é possível observar na Figura 14 a hierarquia dos dados segue uma simples estrutura: **BancoDeDados { Sensor { ChaveUnica { Dados de consumo } } }**, O nome do banco, neste caso “violet-energy-monitor”, representa o próprio banco criado na *cloud*. Já a chave “sensor”, é considerada a raiz, o nível mais alto de hierarquia, onde todos os dados de todos os equipamentos são postados. Cada equipamento inserido no banco recebe uma chave única e dentro de cada uma dessas chaves está um valor de medição de um equipamento, ou seja, os dados são inseridos por redundância.

A redundância de dados foi necessária para que não houvesse sobrescrita de dados, pois qualquer dado que for inserido sob o mesmo nome de chave sobrescreve o anterior, portanto para cada medição, uma chave única e aleatória é criada pelo próprio Firebase, e os dados *hardware* e local são redundantemente inserido em todas as novas medições, variando apenas os valores *timestamp*, corrente e potência. Porém devido a natureza do banco e de como esses dados foram inseridos, pode dificultar a manipulação dos mesmos, pois uma das regras de negócio é que os dados sejam recuperados de 30 dias passados, para que o cliente tenha uma ideia de quanto é o consumo mensal dos equipamentos monitorados, no entanto conforme o banco de dados cresce mais tempo leva para o *Back-end* fazer essa classificação, por estas razões o banco de dados foi indexado por *timestamp*.

3.5.1. Indexação dos dados

O Firebase permite adicionar as suas regras a indexação de dados, para melhorar o desempenho das consultas ao banco de dados (FIREBASE, 2019). A Figura 15, mostra como estão as regras do banco com indexação de dados atendendo as necessidades deste projeto.

Figura 15 – Indexação do banco de dados



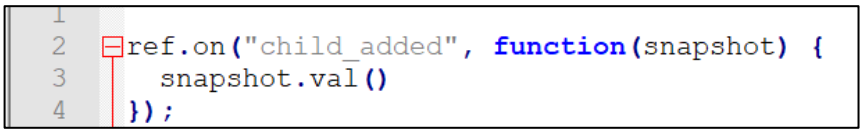
Fonte: Elaborado pelo autor.

Após indexar o banco por *timestamp* o Firebase permite fazer consultas a uma coleção de dados usando a chave filha "*timestamp*", portanto o banco passa a ser classificado por *timestamp,* facilitando recuperar dados a partir da data dos que foram inseridos há 30 dias atrás. Deixando para que o banco realize tal classificação, o código do *Back-end* ganha performance ao recuperar os dados.

3.5.2. Retornando dados do banco *realtime*

Para retornar dados do banco *RealTime* para o *Back-end* de uma aplicação Node.JS é necessário fazer uma chamada para o banco. A Figura 16 demonstra a codificação dessa função de chamada.

Figura 16 – Requisição de dados *RealTime* para o banco de dados



Fonte: Elaborado pelo autor.

Esta função, apresentada na Figura 16, é assíncrona, a qual retornará os nós filhos do banco referenciado e com isso também irá retornar os nós filhos que forem adicionados posteriormente, o que significa que o banco se encarrega de notificar o *Back-end* e encaminhar a ele todo filho que está sendo adicionado no banco do qual

foi solicitado. O Firebase retorna a requisição com dados no formato JSON. A Figura 17 apresenta uma amostra de uma lista de JSON retornada pela API do banco.

Figura 17 – Array de JSON


```
1 [{"current":0.5,"hardware":"Lampadas","place":"Sala","power":63,"timestamp":1553113193000},
2 {"current":0.5,"hardware":"Lampadas","place":"Sala","power":63,"timestamp":1553113193000},
3 {"current":0.5,"hardware":"Lampadas","place":"Sala","power":63,"timestamp":1553113193000}]
```


Fonte: Elaborado pelo autor.

Essa é uma das funções (Figura 16) disponibilizadas pela API do Firebase *RealTime*, a qual é feita para capturar os dados e fazer a visualização de dados *RealTime* que estejam sendo inseridos no banco, sem ter de fazer o *server* consultar o banco constantemente em busca destas atualizações.

3.6 Circuito do equipamento monitor de consumo

O equipamento monitor de consumo de energia elétrica exerce uma função vital neste sistema, realizando a função de captar e enviar dados de corrente e potência para a *cloud*. É importante mencionar que este equipamento foi inicialmente desenvolvido tendo como base o projeto de medidor de energia elétrica disponibilizado pelo portal FelipeFlop (2015), porém diversas mudanças foram feitas no circuito e codificação do sistema embarcado. Para compreender melhor esse equipamento, no decorrer deste capítulo, será abordado aspectos importantes na prototipação do mesmo.

A Tabela 11, apresenta os componentes utilizados, juntamente aos preços.

Tabela 11 – Componentes utilizados na data 29/04/2019 dólar valendo R$3,95

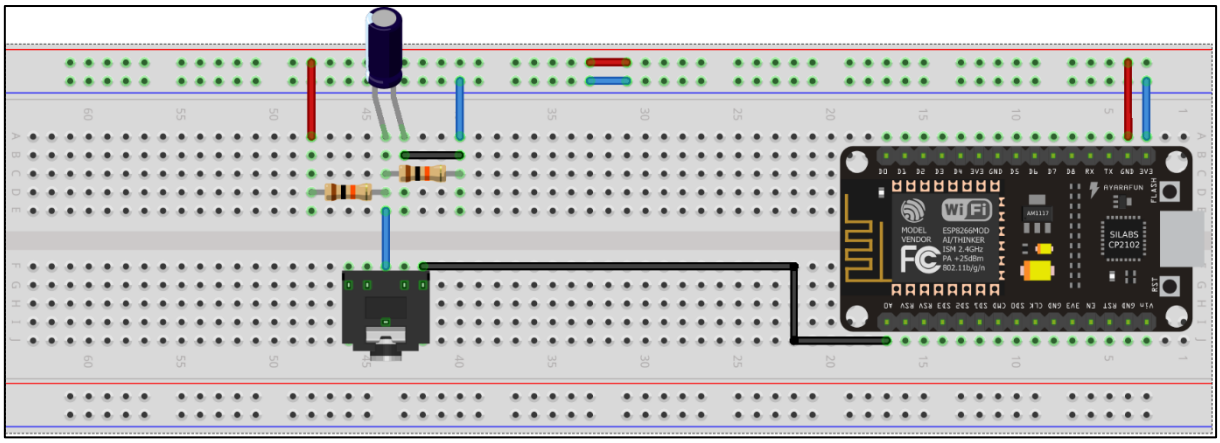
Nome do componente	Preço (R$)
Esp8266 NodeMCU	R$ 9,67
Conector de Áudio Jack J2 (10 unidades)	R$ 29,50
Resistor 10k (10 unidades)	R$ 1,50
Protoboard 830 Pontos	R$ 16,90
Jumpers Macho (65 Unidades)	R$ 10,90
Capacitor 100uF / 16V unidade	R$ 0,15
Sensor Corrente Não Invasivo 20A /1V SCT-013	R$ 56,90

Fonte: Elaborado pelo autor.

A Figura 18 apresenta o circuito utilizado. O microcontrolador alimenta a protoboard eletricamente, que por sua vez possibilita que o conector de áudio, Jack J2, receba os dados do sensor não invasivo, que estiver conectado a ele, e os

transmite ao ESP8266. O microcontrolador, ao receber os dados, calcula valores de corrente e potência e os envia por WI-FI para a nuvem do Firebase.

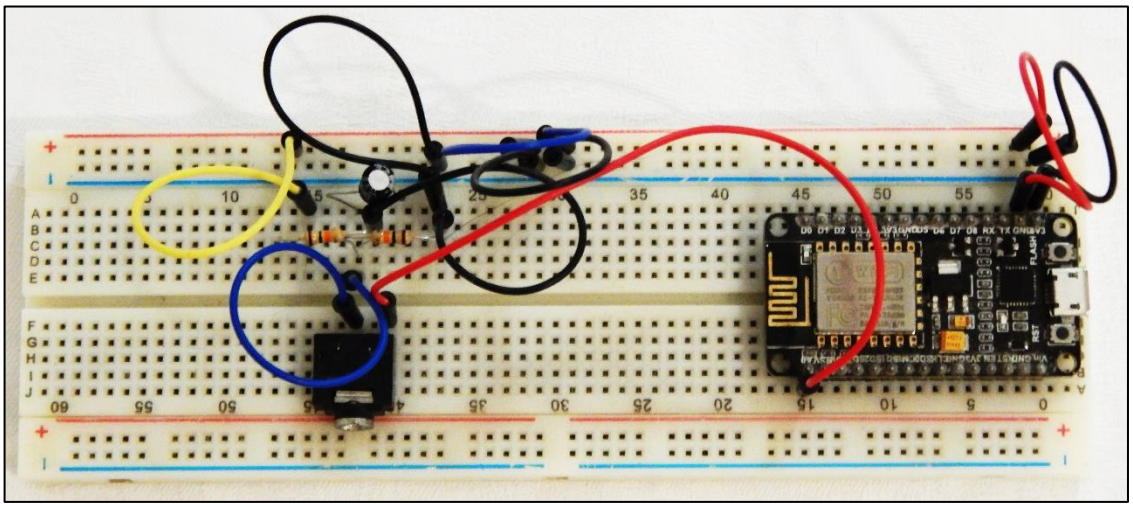
Figura 18 – Circuito



Fonte: Elaborado pelo autor.

A Figura 19, exibe uma foto do protótipo do equipamento monitor de consumo de energia elétrica finalizado.

Figura 19 – Equipamento protótipo



Fonte: Elaborado pelo autor.

No equipamento exibido na Figura 19 fica conectado, ao conector Jack J2 de áudio, o sensor de corrente não invasivo 20A SCT-013, que pode ser visto na Figura 20.

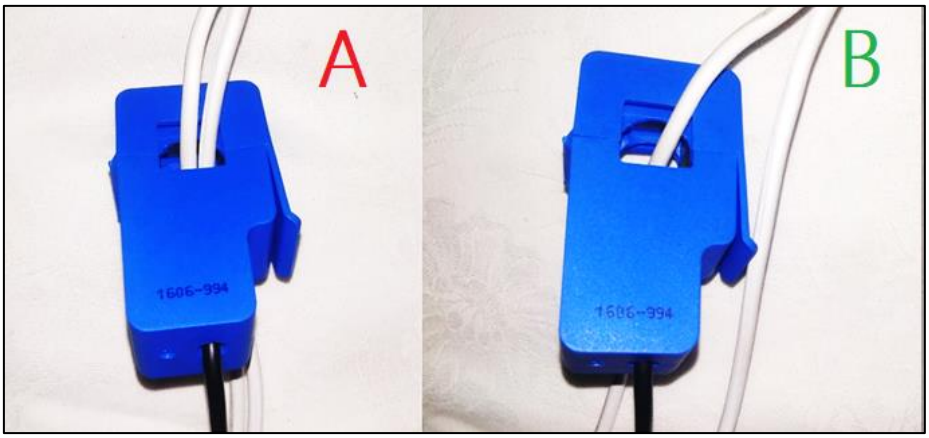
A Figura 20, exibe um sensor de corrente AC (Corrente Alternada), que suporta corrente até 20A. É importante mencionar que o sensor deve ser colocado da maneira

correta no fio que conecta um equipamento à eletricidade, para que possa fazer a medição, veja a Figura 21.

Figura 20 – Sensor de corrente não invasivo 20A SCT-013

Fonte: FelipeFlop, 2015.

Figura 21 – Posicionamento do sensor de corrente



Fonte: Elaborado pelo autor.

A Figura 21, apresenta dois modos de posicionar o sensor, entretanto apenas o lado “B” corresponde à maneira correta de fazê-lo, segurando apenas 1 dos fios, podendo ser qualquer um. Caso o sensor esteja posicionado contendo ambos os fios, lado “A”, o sensor não é capaz de medir a corrente, pois em um fio a corrente entra e no outro sai, resultando em correstes opostas as quais se repelem, segundo a Lei de Ampère (BRESCANSIN, 2013), o resultante dessa medição é um valor nulo impossibilitando o sensor de fazer medições.

A Figura 22 apresenta as definições de variáveis necessárias para que a biblioteca EmonLiteESP realize a medição de corrente do equipamento monitorado, pelo sensor não invasivo.

Figura 22 – Definição das variáveis para medição de consumo

```
/*------ESP8266 DIFINES FOR MEASUREMENT-----------*/
// Precision of the ADC measure in bits. Arduinos and ESP8266 use 10bits ADCs, but the
// ADS1115 is a 16bits ADC
#define ADC_BITS                10
// If using a node ESP8266 board it will be 1.0V, if using a NodeMCU there
// is a voltage divider in place, so use 3.3V instead.
#define REFERENCE_VOLTAGE       3.3
// This is basically the volts per amper ratio of your current measurement sensor.
// If your sensor has a voltage output it will be written in the sensor enclosure,
// something like "30V 1A", otherwise it will depend on the burden resistor you are
// using.
#define CURRENT_RATIO           21
// This version of the library only calculate aparent power, so it asumes a fixe
// main voltage
#define MAIN_VOLTAGE            127
// Analog GPIO(General Purpose Input/Output data) on the ESP8266
#define CURRENT_PIN             0
// Number of decimal positions for the current output
#define CURRENT_PRECISION       3
// Number of samples each time you measure
#define NUMBER_OF_SAMPLE        1024

// instantiate an object for library EmonLiteESP
EmonLiteESP power;
```

Fonte: Elaborado pelo autor.

Após as variáveis terem sido definidas, veja a Figura 22, o microcontrolador NodeMCU recebe os dados do sensor não invasivo, os quais são interpretados pela biblioteca EmonLiteESP, que entrega os valores de corrente de acordo com o valor de tensão, pré-definido em “MAIN_VOLTAGE”. Para obter o valor de potência, basta multiplicar corrente por “MAIN_VOLTAGE”. A Figura 23, mostra a codificação para retornar os valores de corrente e potência.

Figura 23 – Calculo de Corrente e Potência

```
// Callback function that the object will use to retrieve the ADC value
unsigned int currentCallback() {
// If using the ADC GPIO in the ESP8266
  return analogRead(CURRENT_PIN);
}

void setup(){
// Object Configuration
  power.initCurrent(currentCallback, ADC_BITS, REFERENCE_VOLTAGE, CURRENT_RATIO);
}

void loop(){
// To get the current reading (in amps) call the getCurrent method passing the number of samples to take
  float current = power.getCurrent(NUMBER_OF_SAMPLE);
// To get power value multiply current by MAIN_VOLTAGE
  int power = int (current * MAIN_VOLTAGE);
}
```

Fonte: Elaborado pelo autor.

Após obter os dados de corrente e potência é necessário enviar esses dados para a nuvem do Firebase, esses dados serão transformados em um objeto JSON, o qual pode ser observado na Figura 24.

Figura 24 – Objeto JSON

```
DynamicJsonBuffer jsonBuffer;
JsonObject& timeObject = jsonBuffer.createObject();
timeObject["place"] = place;
timeObject["hardware"] = hardware;
timeObject["current"] = current;
timeObject["power"] = power;
JsonObject& tempTime = timeObject.createNestedObject("timestamp");
tempTime[".sv"] = "timestamp";
```

Fonte: Elaborado pelo autor.

Este objeto JSON criado na Figura 24, recebe valores como nome do equipamento “*hardware*”, local onde está o equipamento “*place*”, corrente “*current*”, potência “*power*” e o registro do momento em que foi criado “*timestamp*”. Estando pronto, o objeto é então transmitido via Wi-Fi para a nuvem do Firebase. Veja na Figura 25 o uso dessas bibliotecas.

Figura 25 – Transmitindo dados para a Nuvem

```
//Your Wi-Fi SSID and Password
#define WIFI_SSID "wifi_name"
#define WIFI_PASSWORD "password"
// Firebase config
#define FIREBASE_HOST "your_host"
#define FIREBASE_AUTH "auth"

void setup(){
// connect to Wi-Fi
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
// Start Firebase
 Firebase.begin(FIREBASE_HOST,FIREBASE_AUTH);
}

void loop(){
// push JSON object created to be "sensor/" child, genereting a random key to each object push.
  Firebase.push("sensor/", timeObject);
}
```

Fonte: Elaborado pelo autor.

Na Figura 25 o JSON criado na Figura 24 é enviado para o banco, usando as bibliotecas “ESP8266WiFi.h”, a qual conecta o equipamento ao Wi-fi, e a biblioteca “FirebaseArduino.h” é responsável por estabelecer conexão e enviar o JSON para o banco de dados Firebase.

4 DESENVOLVIMENTO

A etapa de desenvolvimento deste projeto foi feita sob uma das metodologias ágeis, o Scrum. Scrum é um *framework* para o desenvolvimento de projetos complexos, com entregas dinâmicas e sempre buscando melhorar continuamente (SCRUM, 2017).

Um time fazendo uso desse framework é dividido em papéis: *Product Owner*, *Scrum Master* e o Time de Desenvolvimento. O *Product Owner,* ou dono do produto, é o responsável por ampliar o valor do produto entregue pelo time de desenvolvimento. Ele responde pelo produto, gerencia e garante que o time compreenda o *backlog*, saiba quais os próximos passos e trabalhe de modo a agregar mais valor ao produto. O *Product Owner* toma as decisões e decide como o time deve trabalhar de modo que o *backlog* seja priorizado (SCRUM, 2017). O *Scrum Master* age como um facilitador entre o *Product Owner* e o time de desenvolvimento, garante que os eventos Scrum, reuniões de *Planning*, D*aily* e Retrospectiva, ocorram e a comunicação entre o time de desenvolvimento e o *Product Owner* aconteça. Ajuda a maximizar o valor do produto e do time, auxiliando o time de desenvolvimento a atender as necessidades do projeto. Finalmente, o Time de Desenvolvimento é o responsável por entregar o produto desenvolvido ao longo de várias *sprints*, ele deve atender aos requisitos e agregar o máximo valor possível nas funcionalidades do produto. O time de desenvolvimento é auto gerenciável e tem todas as habilidades necessárias para entregar as funcionalidades do produto dentro das *sprints* definidas. O time deve ser ágil, pequeno e capacitado o suficiente para o produto ser entregue (SCRUM, 2017).

Dentro do Scrum o evento principal são as *Sprints,* espaço de tempo que tem por objetivo fazer a entrega de funcionalidades de um produto. Dentro dessas *Sprints* há importantes tipos de reuniões, a *Planning*, ou Planejamento: onde tudo que deve ocorrer e ser entregue dentro do tempo delimitado, é exposto ao time. Após a *Planning* vem a *Daily*, reuniões diárias e rápidas, até o final da *Sprint*, para que a evolução do entregável seja acompanhada e os bloqueios do time sejam expostos para serem tratados, essa reunião é essencial para que o *Scrum Master* possa agir como facilitador e o time de desenvolvimento garantir que fará a entrega do programado. Ao fim da *sprint* a Retrospectiva é feita com o intuito de repassar tudo que ocorreu durante a *sprint*, as melhorias que podem ser feitas e o que ficou para *backlog* da próxima *sprint*. Todos esses eventos Scrum são formas de acompanhar a evolução, otimização

e garantir a entrega do produto. Dessas reuniões, pode ser feita a extração do gráfico *Burndown*, que está associado ao que foi programado e entregue de funcionalidade na *Sprint,* facilitando o acompanhamento da performance do time de desenvolvimento (SCRUM, 2017).

Portanto seguindo esta metodologia o projeto desse trabalho de graduação objetiva entregar um produto, delimitando o tempo por *sprints*, de modo ágil e otimizado, buscando sempre fazer melhorias.

4.1 Etapas de desenvolvimento

As entregas, separadas por *sprints* de 21 dias, foram planejadas levando em consideração o desenvolvimento e documentação do sistema para este trabalho de conclusão de curso, portanto este capítulo irá discorrer sobre cada entrega feita durante as *sprints* desse projeto. Um resumo sobre o objetivo de cada uma das entregas, de 1 a 5, é feito a seguir:

- **Entrega 1:** Início do projeto de conclusão de curso, documentações iniciais e equipamento monitor de consumo.
- **Entrega 2:** Refinando dados captados pelo equipamento, início do desenvolvimento da aplicação.
- **Entrega 3:** Requisitos para a aplicação mudaram, a aplicação passa ser web cloud, documentações revisadas e desenvolvimento do *Back-end*.
- **Entrega 4:** Documentação e desenvolvimento do *Front-end*.
- **Entrega 5:** *Deploy* da aplicação no GCP e finalização das documentações.

4.1.1 Entrega 1

No dia 01/01/2019 foram realizados o planejamento da *sprint* e dos objetivos a serem cumpridos durante os próximos 21 dias. Para cada objetivo foi atribuído a quantidade de dias que poderia levar para ser realizado e uma pontuação relativa a seu nível de dificuldade, sendo 1 - Fácil, 2 - Fácil/Médio, 3 - Médio, 4 - Médio/Difícil e 5 – Difícil. A Tabela 12, apresenta detalhadamente as atividades, seu tempo estimado de realização em dias e sua pontuação.

Tabela 12 – Planejamento realizado da primeira entrega

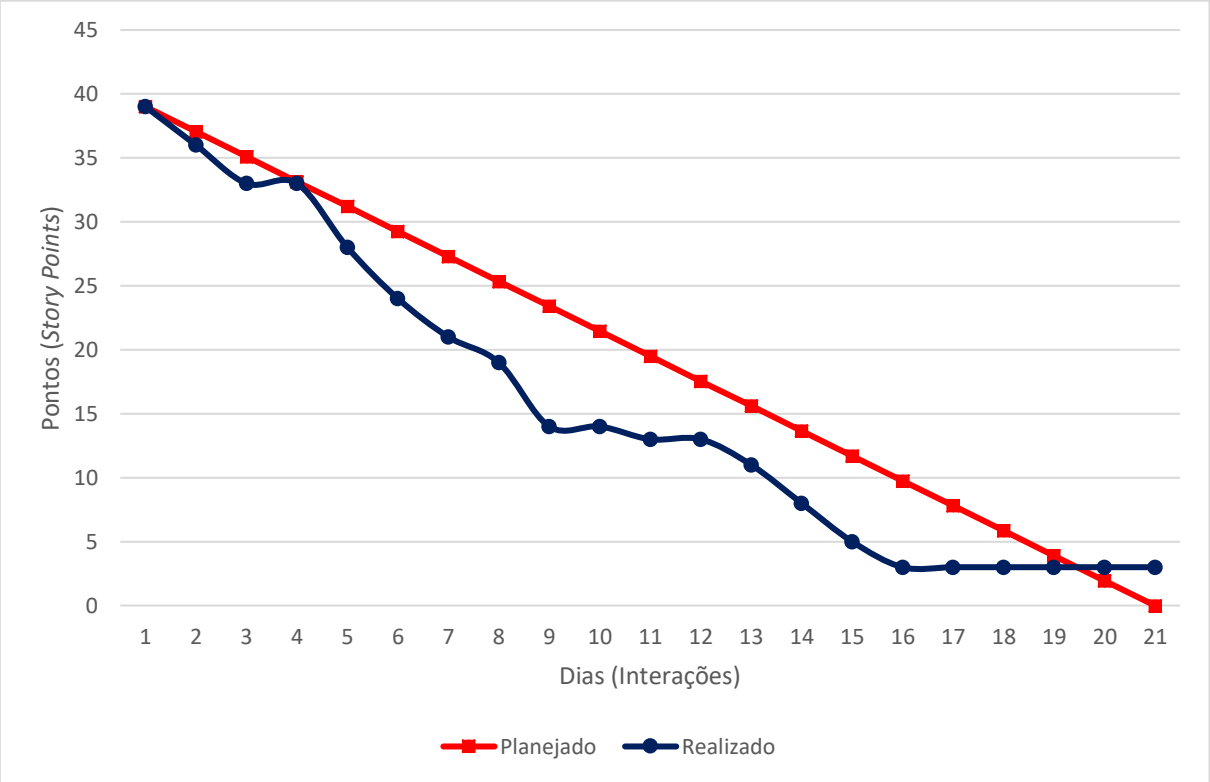
Atividade	Tempo	Pontos
Introdução do Trabalho de Conclusão de Curso	4	5

Objetivo do trabalho de Graduação	1	3
Lista de Tecnologias usadas no trabalho de Graduação	1	3
Requisitos Funcionais e Não Funcionais	2	4
Diagramas do Trabalho de Conclusão de Curso	2	3
Compra dos itens para montar o equipamento monitor de consumo de energia elétrica	1	2
Desenvolver equipamento monitor de consumo de energia elétrica	2	5
visualizar valores de corrente e potência vindos do sensor	1	1
Conectar equipamento à rede Wi-Fi	1	2
Conectar com API do banco de dados Firebase	1	3
Transmitir um JSON com dados de consumo de energia elétrica	1	3
Montar equipamento com lâmpadas incandescentes compradas para serem monitoradas	1	2
Aperfeiçoar medição dos dados	1	3
Total	**19**	**39**

Fonte: Elaborado pelo autor.

A Figura 26 apresenta o gráfico de *Burndown* da entrega 1, o qual exibe o progresso na realização das atividades planejadas para a *sprint,* no decorrer de 21 dias.

Figura 26 – Gráfico de *Burndown* da entrega 1



Fonte: Elaborado pelo autor.

No dia 23/01/2019 foi feito a revisão do entregável. Esta etapa consiste na autoavaliação das atividades feitas, os pontos a serem melhorados e atividades

deixadas para a próxima *sprint*. A visão geral da revisão com os pontos mais importantes foram:

- O que deu certo: Maior parte do que foi planejado foi cumprido.
- O que deu errado: Arduino uno causa muita interferência nas medições do sensor, levando a dados muito distantes da realidade, portanto o aperfeiçoamento das medições de consumo ficou para a próximo *sprint*. Também foi possível notar que os dados JSON transmitidos ao banco sobrescreviam um ao outro, caso a chave tivesse o mesmo nome.
- Ações de melhorias: Trocar o Arduino uno pelo NodeMCU ESP8266, o qual tem uma voltagem menor e já vem equipado para se conectar à rede Wi-Fi e mudar a hierarquia dos dados enviados para o banco.

4.1.2 Entrega 2

É importante mencionar que o espaço de tempo entre a *sprint* 1 e esta *sprint* foi devido à alta demanda proveniente do estágio, o qual exigiu atenção exclusiva da autora deste projeto, resultando no início da segunda entrega na data 01/02/2019.
No dia 01/02/2019 foram realizados o planejamento da *sprint* e dos objetivos a serem cumprido durante os próximos 21 dias. Para cada objetivo foi atribuído a quantidade de dias que poderia levar para ser realizado e uma pontuação relativa a seu nível de dificuldade. Neste planejamento a revisão de documentações foi necessárias já que o equipamento foi modificado. A Tabela 13 apresenta detalhadamente as atividades, seu tempo estimado de realização em dias e sua pontuação.

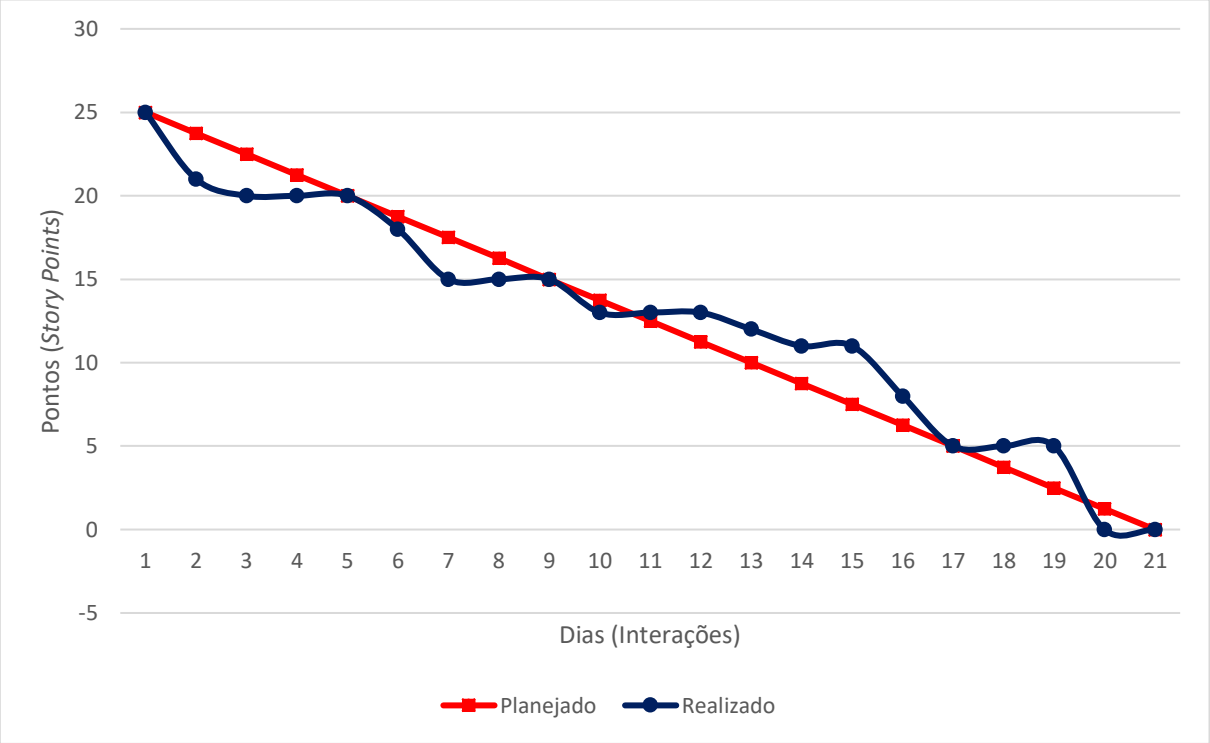
Tabela 13 – Planejamento realizado da segunda entrega

Atividade	**Tempo**	**Pontos**
Revisão da introdução do Trabalho de conclusão de curso	2	4
Substituir o microcontrolador Arduino por ESP8266	1	1
Revisão do código embarcado para se adequar ao equipamento ESP8266	1	2
Atualizar diagramas e lista de tecnologias	2	3
Aperfeiçoamento das leituras de consumo de energia elétrica	1	2
Transmitir um JSON por minuto para o Banco de Dados	1	1
Mandar dados de consumo por redundância sem hierarquia	1	1
Android APP: estabelecer conexão com o banco *RealTime*	1	3
Android APP: Exibir lista *RealTime* dos dados vindo do banco	2	5
Capítulo circuito do equipamento	1	3
Total	**13**	**25**

Fonte: Elaborado pelo autor.

A Figura 27, apresenta o gráfico de *Burndown* da entrega 2, o qual exibe o progresso na realização das atividades planejadas para a *sprint* no decorrer de 21 dias.

Figura 27 – Gráfico de *Burndown* da entrega 2



Fonte: Elaborado pelo autor.

No dia 23/02/2019 foi feito a revisão do entregável. Esta etapa consiste na autoavaliação das atividades feitas, os pontos a serem melhorados e atividades deixadas para a próxima *sprint*. A visão geral da revisão com os pontos mais importantes foram:

- O que deu certo: O equipamento monitor de consumo de energia elétrica ficou operante, conexão entre o app Android e o banco de dados foi feita e dados foram exibidos em uma lista.
- Ações de melhorias: A aplicação poderia ser em nuvem, e deixar totalmente web, deixando para o servidor processar dados de IoT, acrescentando maior performance.

4.1.3 Entrega 3

No dia 24/02/2019 foram realizados o planejamento da *sprint* e dos objetivos a serem cumprido durante os próximos 21 dias. Para cada objetivo foi atribuído a

quantidade de dias que poderia levar para ser realizado e uma pontuação relativa a seu nível de dificuldade. Neste planejamento alguns requisitos mudaram, o sistema passaria a ser web, portanto toda documentação relativa a aplicação teria de ser atualizada. A Tabela 14 apresenta detalhadamente as atividades, seu tempo estimado de realização em dias e sua pontuação.

Tabela 14 – Planejamento realizado da terceira entrega

Atividade	**Tempo**	**Pontos**
Diagrama de caso de uso página web	1	3
Diagrama de sequência	1	5
Revisão de requisitos funcionais e não funcionais	2	5
Tabela de rotas do *Back-end*	1	1
Disponibilizando o *Back-end* na porta 8080	1	1
Desenvolvimento do roteador do *Back-end*	1	3
Desenvolvimento do *middleware* do *Back-end*	1	3
Conexão com a *API* do banco de dados	1	2
Desenvolvimento do algoritmo de cálculo de consumo de energia elétrica em /piechartdata	2	5
Desenvolvimento do algoritmo de cálculo de consumo de energia elétrica em /linechartdata	2	5
Capítulo sobre a *API* do *Back-end*	2	3
Revisão e correção de bugs do *Back-end*	2	3
Total	**17**	**39**

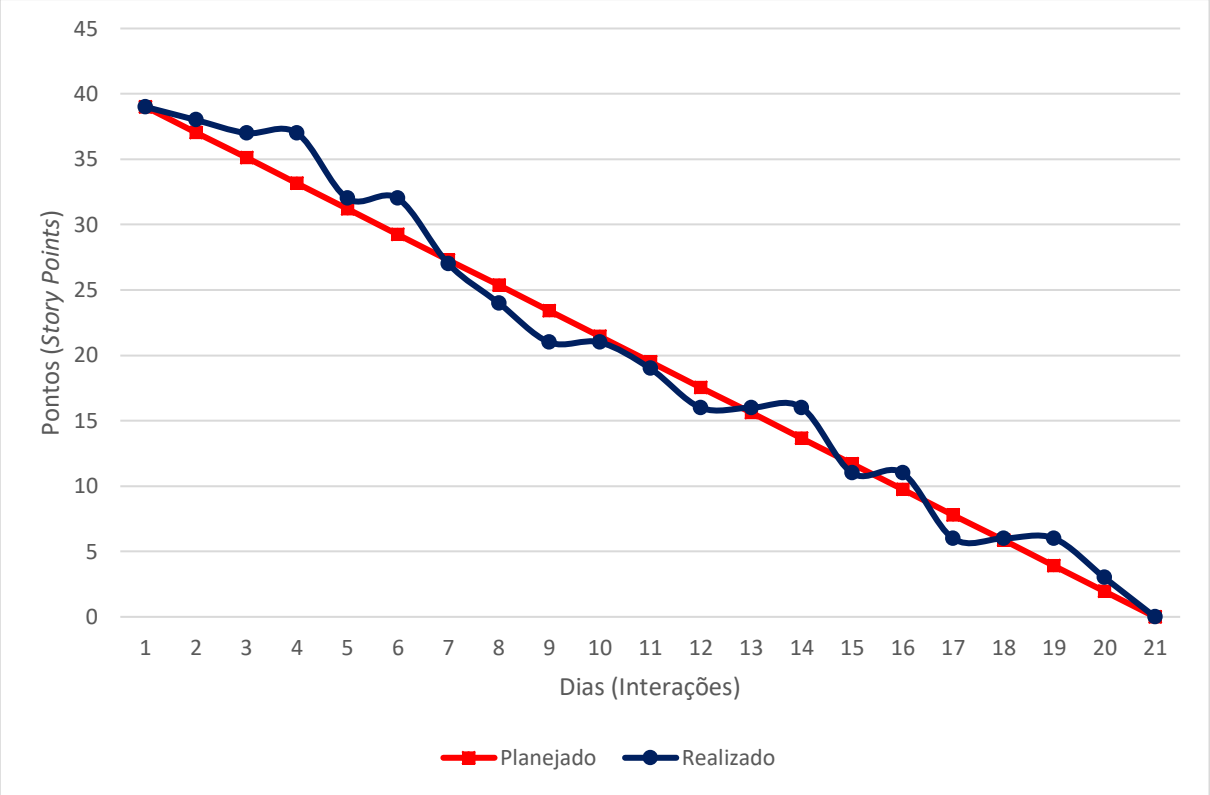
Fonte: Elaborado pelo autor.

A Figura 28, apresenta o gráfico de *Burndown* da entrega 3, o qual exibe o progresso na realização das atividades planejadas para a *sprint,* no decorrer de 21 dias. Percebe-se que a *sprint* correu sem grandes problemas, atendendo todas as atividades dentro do esperado.

No dia 18/03/2019 foi feito a revisão do entregável. Esta etapa consiste na autoavaliação das atividades feitas, os pontos a serem melhorados e atividades deixadas para a próxima *sprint*. A visão geral da revisão com os pontos mais importantes foram:

- O que deu certo: todo o planejado da *sprint*.
- O que deu errado: nada.

Figura 28 – Gráfico de *Burndown* da entrega 3



Fonte: Elaborado pelo autor.

4.1.4 Entrega 4

No dia 19/03/2019 foram realizados o planejamento da *sprint* e dos objetivos a serem cumprido durante os próximos 21 dias. Para cada objetivo foi atribuído a quantidade de dias que poderia levar para ser realizado e uma pontuação relativa a seu nível de dificuldade. A Tabela 15, apresenta detalhadamente as atividades, seu tempo estimado de realização em dias e sua pontuação.

Tabela 15 – Planejamento realizado da quarta entrega

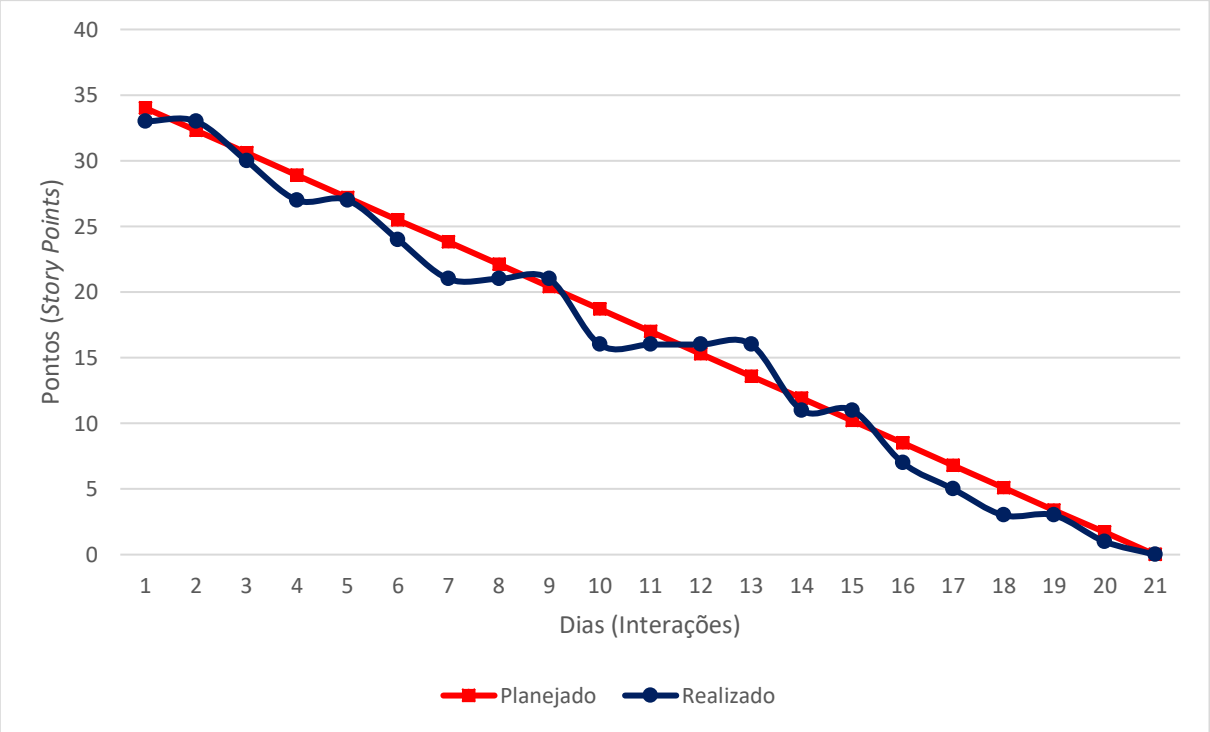
Atividade	Tempo	Pontos
Desenvolvimento da rota do *Front-end*	1	1
Desenvolvimento da API de autenticação do *Front-end*	2	4
Criação do gráfico de rosca	1	3
posicionamento do gráfico de rosca	1	3
Desenvolvimento da lista lateral *RealTime*	1	3
Responsividade do *piechart* ao *linechart*	2	5
Criação do *linechart* data	1	3
Posicionamento do *linechart*	2	5
desenvolvimento da tela de erro	1	2

Botão de login	1	1
Posicionamento do botão de login	1	1
Posicionamento do botão de sair	1	1
Chamada da *API* do *Back-end* para obter dados *piechart*	1	1
chamada da *API* do *Back-end* para obter dados *line chart*	1	1
Total	**17**	**34**

Fonte: Elaborado pelo autor.

A Figura 29 apresenta o gráfico de *Burndown* da entrega 4, o qual exibe o progresso na realização das atividades planejadas para a *sprint,* no decorrer de 21 dias. Apesar de posicionar os gráficos ter sido uma tarefa mais complicada do que o previsto no planejamento, a *sprint* correu sem grandes problemas, atendendo todas as atividades dentro do esperado.

Figura 29 – Gráfico de *Burndown* da entrega 4



Fonte: Elaborado pelo autor.

No dia 10/04/2019 foi feito a revisão do entregável. Esta etapa consiste na autoavaliação das atividades feitas, os pontos a serem melhorados e atividades deixadas para a próxima *sprint*. A visão geral da revisão com os pontos mais importantes foram:

- O que deu certo: todo o planejado da *sprint*.
- O que deu errado: nada.

4.1.5 Entrega 5

No dia 11/04/2019 foram realizados o planejamento da *sprint* e dos objetivos a serem cumprido durante os próximos 21 dias. Para cada objetivo foi atribuído a quantidade de dias que poderia levar para ser realizado e uma pontuação relativa a seu nível de dificuldade. A Tabela 16, apresenta detalhadamente as atividades, seu tempo estimado de realização em dias e sua pontuação.

Tabela 16 – Planejamento realizado da quinta entrega

Atividade	Tempo	Pontos
Adicionar *logger* ao projeto	2	2
Fazer *build* do *Front-end*	1	1
Comprimir os arquivos do *Back-end*	1	1
Comprimir os arquivos estáticos do *fron-end*	1	1
Criar uma máquina virtual na nuvem do GCP	1	2
Liberar tráfego na porta 8080 da máquina virtual	1	2
Levar o arquivo comprimido do *Back-end* para a nuvem	1	1
Levar o arquivo comprimido do *Front-end* para a nuvem	1	1
Instalar Apache2, NPM e Node.JS na máquina virtual	1	1
Instalar as dependências o *Back-end* na máquina virtual	1	1
Criar arquivo de configuração para manter a aplicação sempre disponível na máquina virtual.	1	3
Criar arquivo de configuração de proxy do apache	1	3
Plano de teste	2	3
Capítulo *Deploy* do sistema	2	4
Capítulo Telas do sistema	1	2
Total	**17**	**39**

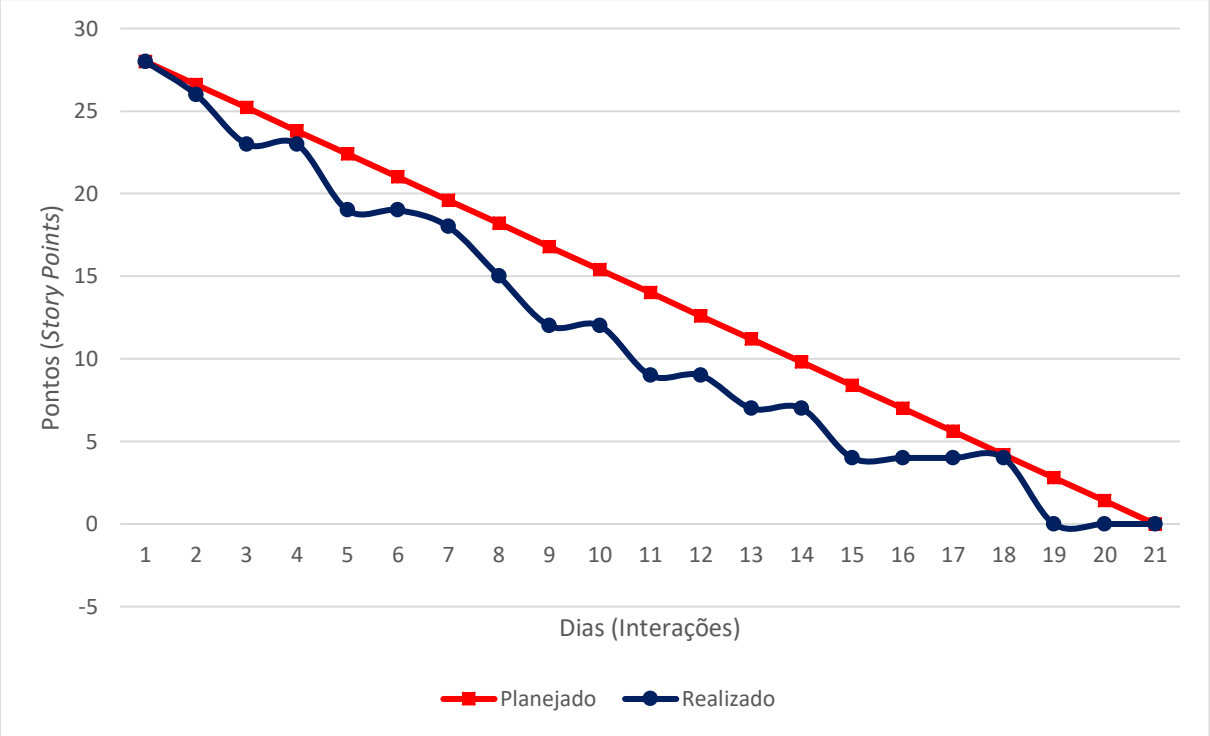
Fonte: Elaborado pelo autor.

A Figura 30 apresenta o gráfico de *Burndown* da entrega 5, o qual exibe o progresso na realização das atividades planejadas para a *sprint* no decorrer de 21 dias. Fase de *deploy* da aplicação concluiu-se antes do esperado, o que auxiliou adiantar o plano de teste, resultando em uma *sprint* sem grandes problemas, atendendo todas as atividades dentro do esperado.

No dia 03/05/2019 foi feita a revisão do entregável. Esta etapa consiste na autoavaliação das atividades feitas, os pontos a serem melhorados e atividades deixadas para a próxima *sprint*. A visão geral da revisão com os pontos mais importantes foram:

- O que deu certo: todo o planejado da *sprint*.
- O que deu errado: nada.

Figura 30 – Gráfico de *Burndown* da entrega 3



Fonte: Elaborado pelo autor.

4.2 Deploy do projeto

Nesta fase o sistema web é levado para a nuvem do Google e disponibilizado para acesso do usuário. Com o objetivo de mostrar como esse processo é feito, esse capítulo apresenta as principais configurações, para o sistema web ficar disponível para o usuário final.

4.2.1 Preparando o projeto para o ambiente de produção

A etapa de preparação do projeto, que anteriormente era apenas executado localmente, para o ambiente de produção do servidor é fundamental para que o site seja disponibilizado para o cliente final, portanto algumas mudanças são necessárias antes de levar o *Back-end* e o *Front-end* para a nuvem do Google.

Anteriormente o projeto estava configurado para um ambiente de desenvolvimento, no qual não existe uma real preocupação sobre como está sendo servido o *Front-end* e a saída de *logs*[19], porém, esses pontos tornam-se problemáticos

[19] Logs ou log de dados: se trata do um registro de eventos do sistema geralmente arquivado (FOLDOC, 2009).

em um ambiente de produção, exigindo modificações para tratar logs e servir os arquivos do *Front-end*.

4.2.2 Logando em produção

Funções como “Console.log” e “Console.err” são síncronas, o que pode resultar na ocupação indevida da thread principal, até que uma resposta seja obtida, tornando o sistema mais lento (NODE.JS, 2013), esses tipos de funções são problemáticas para o ambiente de produção. Uma boa prática em produção é fazer uso de funções assíncronas (Express, 2015), portanto neste trabalho de graduação foi feito uso do Winston, um logger simples para registro de atividades do sistema, o qual oferece várias opções de transporte (WINSTON, 2019). A Figura 31, mostra um exemplo simples de criação de um logger usando Winston.

Figura 31 – Exemplo de uso do Logger Winston

```
const logger = winston.createLogger({
  level: 'info',
  format: winston.format.json(),
  defaultMeta: { service: 'user-service' },
  transports: [
    //
    // - Write to all logs with level `info` and below to `combined.log`
    // - Write all logs error (and below) to `error.log`.
    //
    new winston.transports.File({ filename: 'error.log', level: 'error' }),
    new winston.transports.File({ filename: 'combined.log' })
  ]
});

//
// If we're not in production then log to the `console` with the format:
// `${info.level}: ${info.message} JSON.stringify({ ...rest }) `
//
if (process.env.NODE_ENV !== 'production') {
  logger.add(new winston.transports.Console({
    format: winston.format.simple()
  }));
}
```

Fonte: Winston, 2019.

4.2.3 *Front-end build*

Neste projeto uma versão build do *client* é criada usando o comando “npm run build”, que no *Front-end* React cria uma versão otimizada do projeto para produção em um novo diretório (FACEBOOK, 2019). Essa etapa é essencial, pois simplesmente levar todos os arquivos do *Front-end* para o ambiente de um servidor pode dificultar sua disponibilidade para o usuário. Portanto, ao fazer uso do comando para gerar o arquivo build do projeto React, os arquivos são compactados e otimizados, para serem disponibilizados para o usuário.

4.2.4 GCP

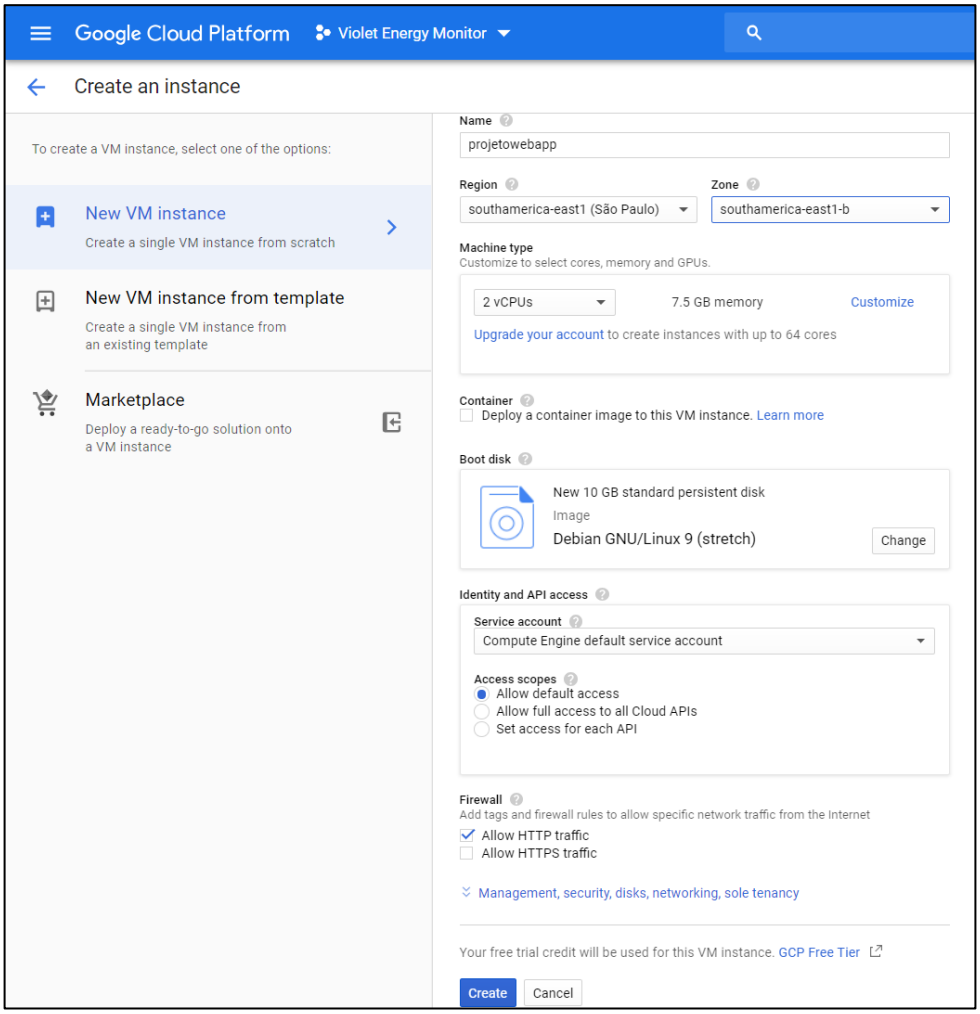
A nuvem escolhida para disponibilizar o sistema web do monitor de consumo de energia elétrica foi a Plataforma Cloud do Google (GCP), pois esta plataforma oferece vasta customização das máquinas virtuais, atendendo as necessidades deste projeto e também oferece crédito com validade de 1 ano para usar seus serviços.

4.2.5 Criação da máquina virtual

O serviço do GCP de computação em máquinas virtuais é o Google Compute Engine, nele será criado a máquina virtual a qual hospedará o sistema web. Ao criar a máquina virtual do projeto, as configurações escolhidas para a instância podem ser visualizadas na Figura 32.

Criada a instância da máquina virtual é importante criar uma regra de firewall, pois como este projeto é desenvolvido fazendo uso de uma arquitetura desacoplada de *server* e *client,* se faz necessário liberar o tráfego de dados na porta 8080, onde a API do *Back-end* está operando. Para criar a regra é necessário ir em “VPC network > Firewall rules > Create Firewall Rule”. A Figura 33, exibe as configurações necessárias para liberar chamadas na porta 8080.

Figura 32 – GCP: Configuração da máquina virtual



Fonte: GCP, 2019.

4.2.6 Configurações da máquina virtual

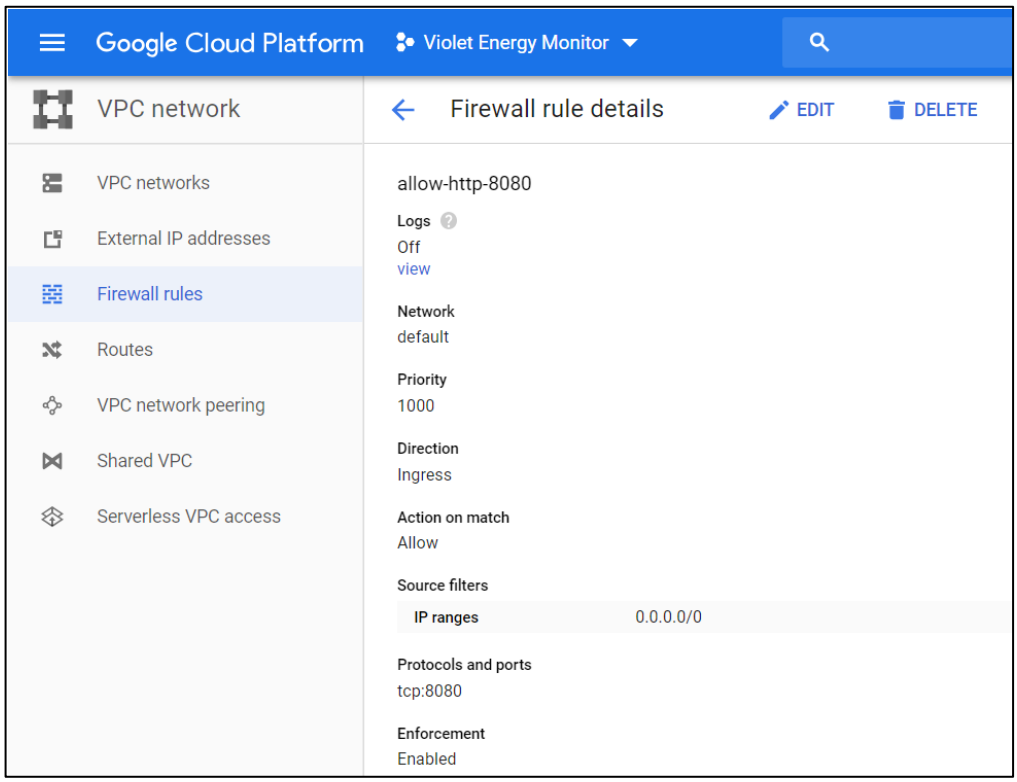
Nesta etapa as configurações essenciais, para que a máquina hospede o *Back-end* e o *Front-end* deste projeto, são feitas. Portanto esse capítulo será subdivido em configurações do *server* e configurações do c*lient*, para atender a cada uma de suas necessidades.

4.2.6.1 Configurações da máquina virtual - *back-end*

Antes de levar o *Back-end* para a máquina virtual, é essencial instalar o gerenciador de dependências NPM na VM (Máquina Virtual, do inglês Virtual Machine), com ele é possível fazer o download do *runtime* Node.JS versão 6.4.1, a qual é a exata versão usada para o desenvolvimento do *server* do projeto. É

fundamental manter as versões utilizadas no ambiente de desenvolvimento, pois assim garante a consistência do projeto na nuvem.

Figura 33 – GCP: Configuração de Regra do Firewall



Fonte: GCP, 2019.

Após ter o *runtime* instalado na VM, o projeto já pode ser levado para a nuvem e suas dependências podem ser instaladas utilizando o gerenciador de pacotes NPM. A próxima etapa essencial é garantir que o projeto inicie e reinicie automaticamente, pois servidores podem cair, resultando na necessidade de reiniciar a aplicação manualmente, o que é problemático, o sistema web pode ficar indisponível ao usuário por muito tempo até seja disponibilizado novamente, por esse motivo será feito uso do serviço Linux Systemd.

Para fazer gerenciamento de uma aplicação Node.JS através do Systemd, um arquivo *unit* é criado dentro do diretório “/etc/systemd/system”, com as configurações expostas na Figura 34.

A Figura 34, exibe um script padrão indicado pelo guia de boas práticas em produção do Express (2015), os pontos mais importantes são:

ExecStart responsável por executar o comando que inicia o *server*;

Environment iguala a variável de ambiente “NODE_ENV” a “production”, para que o Node.JS trabalhe mais eficientemente.

Restart passa a ser “on-failure”, portanto sempre que a aplicação falhar, quebrar, ficar indisponível, ela é reiniciada.

Tendo concluído essas configurações é possível garantir que a aplicação se mantenha disponível ao usuário, independentemente de sessão, ou de inicialização manual no servidor.

Figura 34 – GCP: Configuração do Unit File

```
  GNU nano 2.7.4

[Unit]
Description=Express server

[Service]
Type=simple
ExecStart=/usr/bin/node /home/barcelosjanaine/server/index.js

# Environment variables:
Environment=NODE_ENV=production


# Allow many incoming connections
LimitNOFILE=infinity

# Allow core dumps for debugging
LimitCORE=infinity

StandardInput=null
StandardOutput=syslog
StandardError=syslog
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Fonte: Elaborado pelo autor.

O código fonte do Back-end, desenvolvido neste projeto de conclusão de curso, pode ser encontrado no repositório da autora, acessível em: <https://github.com/JanaineB/MonitorEnergy_PROD/tree/master/server>.

4.2.6.2 Configurações da máquina virtual - *front-end*

Nesta etapa de configuração os arquivos build do *Front-end* são levados para a VM, no caminho de diretórios “/var/www/html”, este é o local padrão onde o servidor web faz a leitura dos arquivos estáticos e os serve para o cliente.

Após levar os arquivos estáticos para VM, é necessário instalar o servidor web Apache e configurar o arquivo “000-default.conf” para habilitar o proxy do *Front-end* e servir uma aplicação *single page* como o React, apontando toda URI para o arquivo index do *Front-end*. A Figura 35 exibe as configurações do arquivo “000-default.conf” encontrado em “/etc/apache2/sites-enabled/”.

Figura 35 – GCP: Configuração do arquivo 000-default.conf do Apache

```
  GNU nano 2.7.4                                        File: /etc/apache2/site

<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com
        ProxyPreserveHost On
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html
        <Directory "/var/www/html">
    RewriteEngine on
    # Don't rewrite files or directories
    RewriteCond %{REQUEST_FILENAME} -f [OR]
    RewriteCond %{REQUEST_FILENAME} -d
    RewriteRule ^ - [L]
    # Rewrite everything else to index.html to allow html5 state links
    RewriteRule ^ index.html [L]
        </Directory>
        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
        ProxyPass /api http://127.0.0.1:8080/api
        ProxyPassReverse /api http://127.0.0.1:8080/api

        ServerName localhost
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Fonte: Elaborado pelo autor.

O código fonte do Front-end, desenvolvido nesse projeto de conclusão de curso, pode ser encontrado no repositório da autora, acessível em: <https://github.com/JanaineB/MonitorEnergy_PROD/tree/master/client>.

4.3 Plano de teste

Neste capítulo será desenvolvido os casos de teste do sistema web deste projeto, esse plano tem como objetivo testar os casos de uso, apresentados no capítulo 3.2.1, para garantir que sejam atendidos.

Tabela 17 – Caso de teste "Efetuar login com sucesso"

Caso de teste	Efetuar login
Descrição	Este caso de teste simula o usuário válido efetuando login no sistema.
Pré-condições	Ter uma conta Google válida.
Entrada	E-mail e senha.
Ações do ator	**Resultados esperados**
O acessa a página web.	Uma tela pop-up aparece solicitando que o usuário efetue login com sua conta do Google.
O usuário efetua o login.	Usuário é redirecionado para tela inicial do sistema.

Fonte: Elaborado pelo autor.

Tabela 18 – Caso de teste "Efetuar login sem sucesso"

Caso de teste	Efetuar login
Descrição	Este caso de teste simula o usuário inválido efetuando login no sistema.
Pré-condições	Ter uma conta Google válida.
Entrada	E-mail e senha.
Ações do ator	**Resultados esperados**
O acessa a página web.	Uma tela pop-up aparece solicitando que o usuário efetue login com sua conta do Google.
O usuário efetua o login.	Usuário é redirecionado para tela de erro do sistema.

Fonte: Elaborado pelo autor.

Tabela 19 – Caso de teste "Visualizar gráfico de consumo de energia elétrica"

Caso de teste	Visualizar gráfico de consumo de energia elétrica
Descrição	Este caso de teste simula a exibição do gráfico em forma de rosca no *Front-end.*
Pré-condições	Ter efetuado o Login, ter dados disponíveis no banco de dados.
Ações do ator	**Resultados esperados**
Acessar a tela inicial do Sistema.	O Sistema exibe um gráfico contendo com dados de consumo de todos os equipamentos.

Fonte: Elaborado pelo autor.

Tabela 20 – Caso de teste "Visualizar gráfico de consumo de um equipamento"

Caso de teste	Visualizar gráfico de consumo de um equipamento
Descrição	Este caso de teste simula a exibição do gráfico de linha no *Front-end.*
Pré-condições	Ter efetuado login, ter dados disponíveis do equipamento no banco de dados.
Ações do ator	**Resultados esperados**
O usuário clica no nome de um equipamento que esteja sendo exibido no gráfico de rosca.	O sistema exibe um gráfico de linha contendo informações de consumo do equipamento clicado.

Fonte: Elaborado pelo autor.

Tabela 21 – Caso de teste "Visualiza dados de equipamentos ligados"

Caso de teste	Visualiza dados de equipamentos ligados
Descrição	Este caso de teste simula a exibição dos dados de um equipamento monitorado que esteja ligado.
Pré-condições	Ter efetuado o Login, ter dados de equipamento sendo inseridos no banco.
Ações do ator	**Resultados esperados**
Acessar a tela inicial do Sistema.	O Sistema exibe uma lista lateral a esquerda da página contendo as informações de consumo de qualquer equipamento monitorado o qual estiver ligado.

Fonte: Elaborado pelo autor.

Tabela 22 – Caso de teste "Monitor de Consumo de energia elétrica"

Caso de teste	Monitorar consumo de energia elétrica.
Descrição	Este caso de teste simula a leitura do consumo de energia elétrica pelo equipamento monitor.
Pré-condições	Ter um equipamento em funcionamento para ser monitorado
Ações do ator	**Resultados esperados**
O NodeMCU capta informações vindas do sensor	O software calcula corrente e potência.
	O Software transmite os dados para o banco de dados.

Fonte: Elaborado pelo autor.

4.4 Telas do sistema

Nesse capítulo são apresentadas as telas do sistema, a interface visual com a qual o usuário final interage.

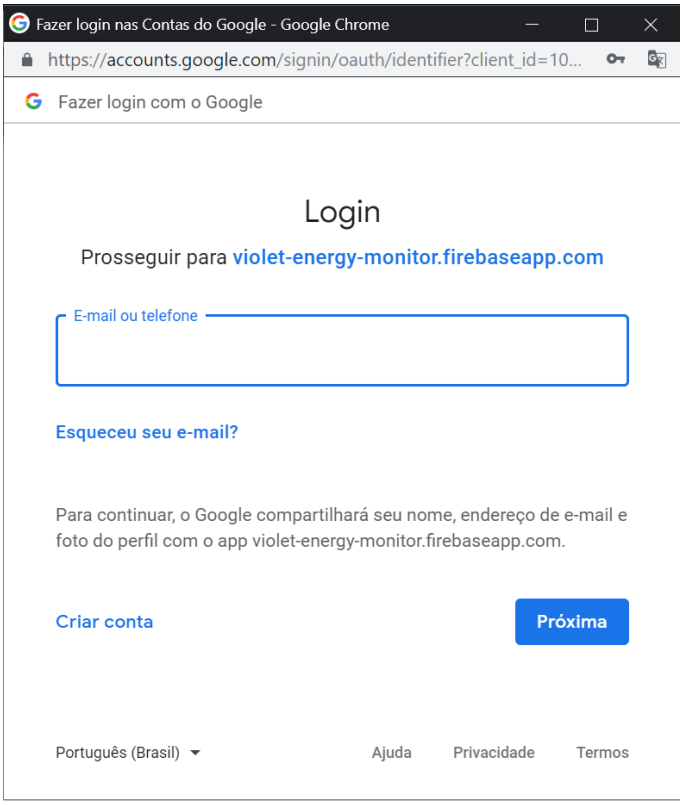
4.9.1 Login

Ao acessar a página web do sistema, uma janela pop-up será exibida para o usuário fazer autenticação com uma conta Google, nessa são solicitados e-mail e senha. A Figura 36, apresenta a tela de login em formato *pop-up*.

Na tela, apresentada na Figura 37, é possível ver um gráfico de consumo de energia elétrica, o qual exibe os dados de consumo total dos últimos 30 dias, dos equipamentos monitorados, em quilowatts-hora. Ao clicar sobre o nome de algum equipamento mais informações sobre ele é exibida, como mostra a Figura 38.

A Figura 38, mostra mais detalhes sobre um equipamento, nela é possível ver um cartão contendo nome do equipamento clicado, local do equipamento e último registro de consumo do equipamento, contendo valores de corrente, potência, data e hora desse registro. Também pode ser observado um gráfico de linha, exibindo o consumo daquele equipamento, em quilowatts-hora, por dia. Caso algum equipamento seja ligado, enquanto o usuário acessa a página web, uma lista lateral irá aparecer exibindo os dados de consumo desse equipamento, que são captados pelo sensor e inseridos no banco. A Figura 39, exibe a lista lateral.

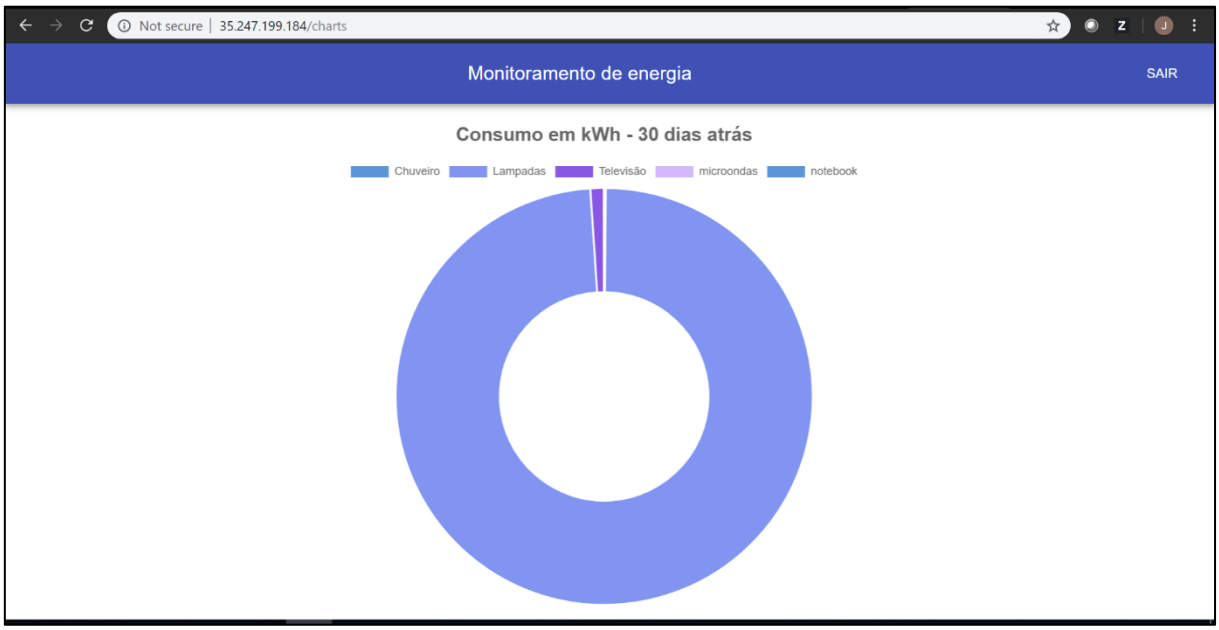
Figura 36 – *Login* do sistema *desktop*



Fonte: Elaborado pelo autor.

Após a autenticação o usuário será direcionado para a tela inicial do sistema. Exibida na Figura 37.

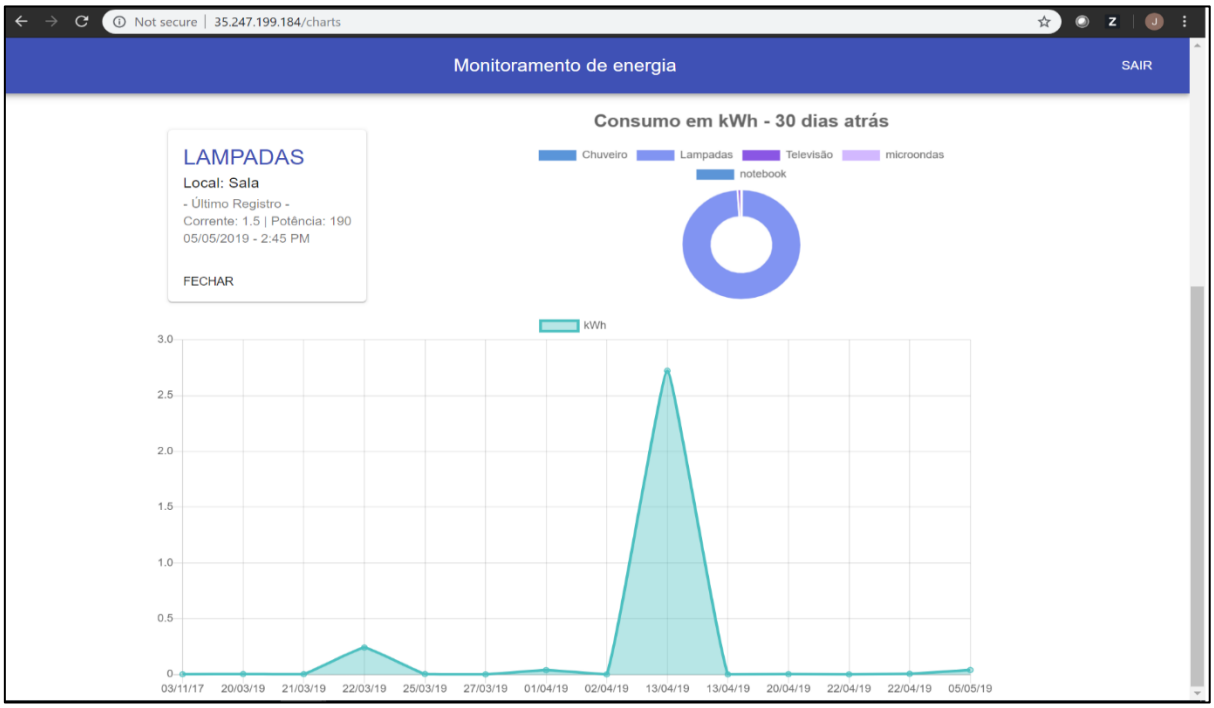
Figura 37 – Tela inicial da página Web do Sistema monitor de Consumo



Fonte: Elaborado pelo autor.

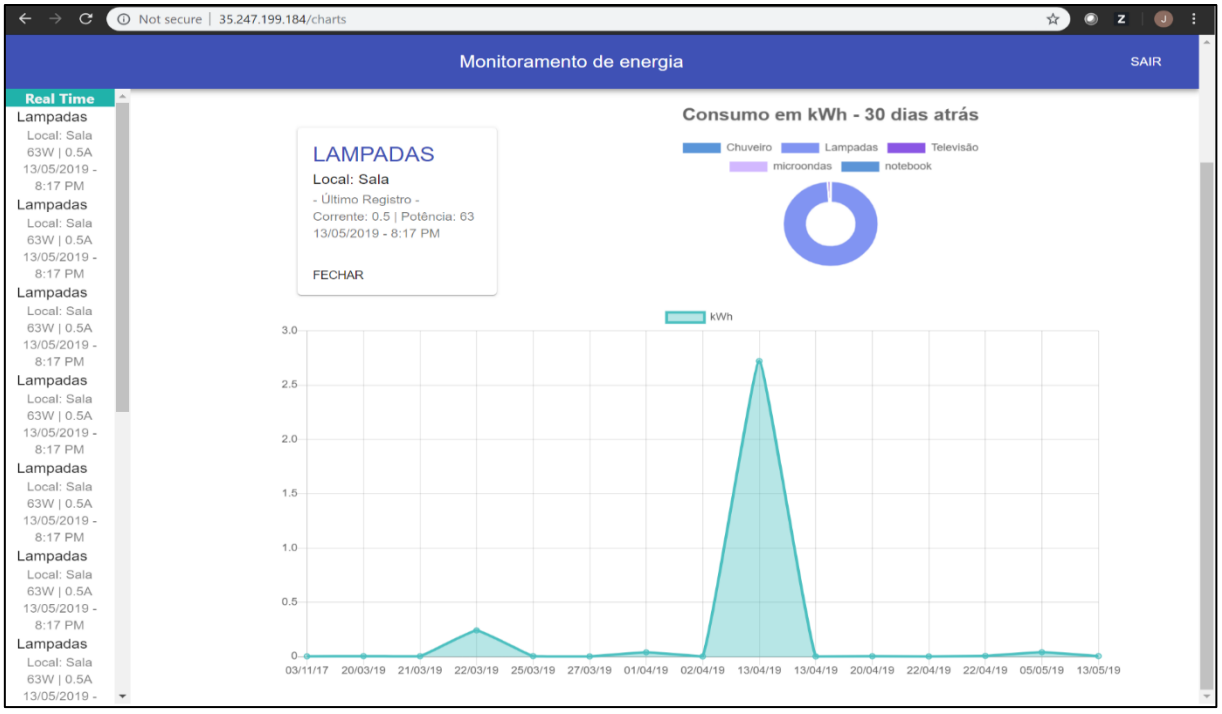
Caso o usuário clique no botão “SAIR”, localizado no canto superior direito da Figura 39, ele realiza o *logoff* do serviço web, então é devolvido a ele uma tela contendo apenas o cabeçalho da página web e um botão para realizar o login escrito “GOOGLE ENTRAR”, como pode ser observado na Figura 40.

Figura 38 – Mais detalhes sobre o consumo de um equipamento



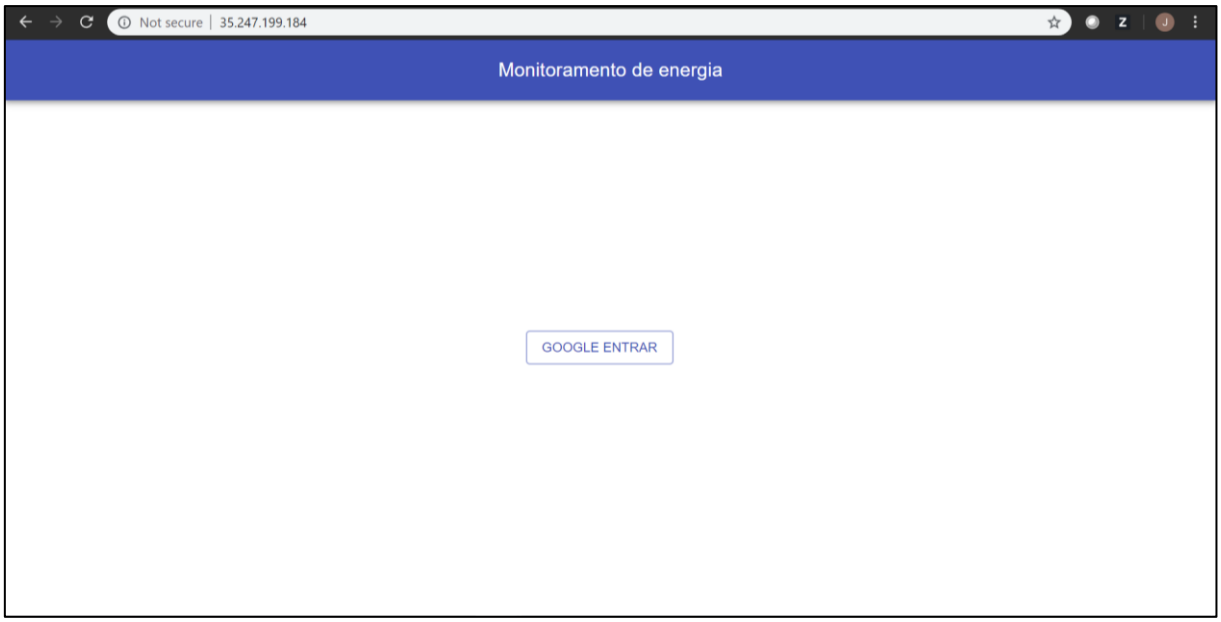
Fonte: Elaborado pelo autor.

Figura 39 – Dados *RealTime* de consumo



Fonte: Elaborado pelo autor.

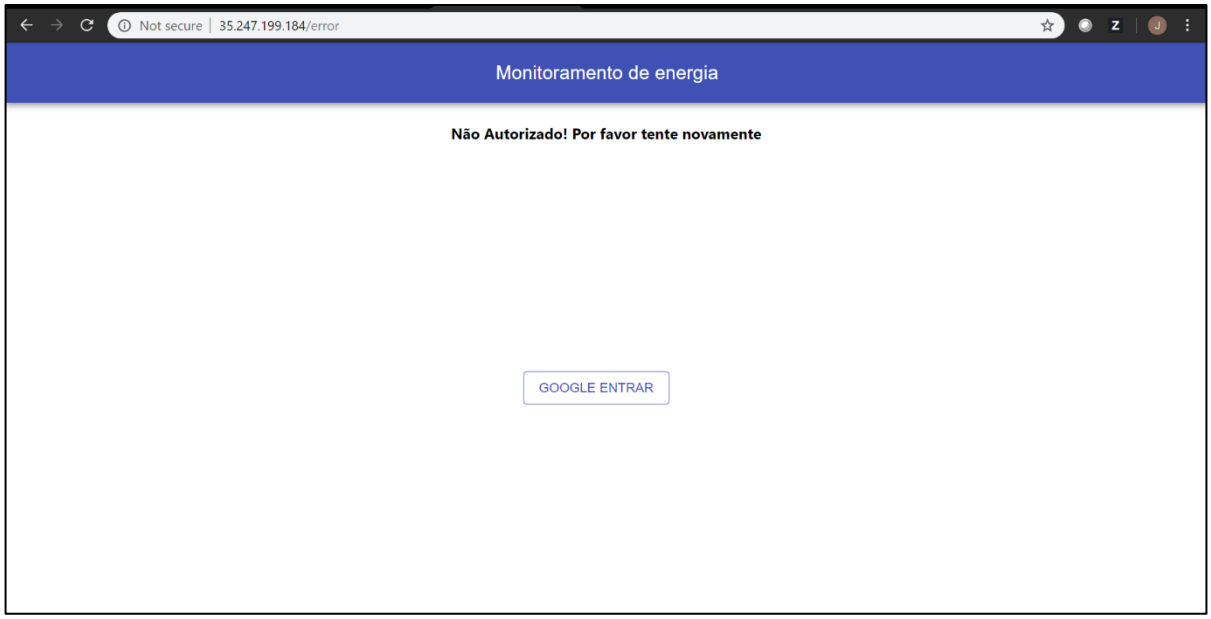
Figura 40 – Tela exiba quando feito o *logoff* do sistema web



Fonte: Elaborado pelo autor.

Caso o usuário realize o *login* na aplicação e por alguma razão não seja capaz de se autenticar no sistema, o mesmo será direcionado a tela exibida na Figura 41.

Figura 41 – Mensagem de confirmação de adição de instituição



Fonte: Elaborado pelo autor.

Como mostra a Figura 41, o usuário que por alguma razão não foi autenticado visualiza uma mensagem de que não autorizado e um botão para tentar realizar o login novamente.

5 CONSIDERAÇÕES FINAIS

Este trabalho de graduação teve como objetivo desenvolver um pequeno sistema monitor de consumo de energia elétrica, que possui um protótipo de um equipamento monitor e um site hospedado em nuvem, contribuindo para que o usuário final tenha acesso aos seus dados de consumo, provenientes dos equipamentos que estejam sendo monitorados. Deixando para um serviço na nuvem o processamento da massa de dados gerada pelo equipamento IoT.

Inicialmente houve muitas mudanças nos requisitos deste trabalho e nas tecnologias envolvidas. O protótipo do equipamento IoT que transmitia os dados para nuvem fazia uso da placa Arduino Uno, porém uma melhor opção foi ofertada pelo Professor e Orientador Kleber Andrade, o NodeMCU ESP8266, o qual já é capacitado para conectar-se à rede Wi-Fi e com maior espaço de memória. Ao ser testado o microcontrolador provou auxiliar na medição dos dados, sem causar grandes interferências, resultando na substituição definitiva da placa Arduino Uno pelo NodeMCU ESP8266. Esse equipamento então transmite os dados de corrente e potência para o banco, os quais seriam expostos em um aplicativo Android.

O processo de desenvolvimento da aplicação Android coincidiu com o momento em que novos conhecimentos, sobre desenvolvimento web e tecnologias Cloud, foram adquiridos ao adentrar o mercado de trabalho. Trabalhando como estagiária, observando a crescente demanda web para o mercado de trabalho e a oportunidade de deixar o processamento de dados IoT na nuvem, os requisitos de uma aplicação para celular foram substituídos pelos requisitos de uma interface web, portanto o sistema que iria expor os dados do cliente passou a ser desenvolvido fazendo uso dessas tecnologias.

Quando o planejamento do site começou, muitas capacitações foram necessárias para desenvolver uma arquitetura web e fazer uso de ferramentas que fossem acessíveis para esse projeto de graduação, por esse motivo, o estágio foi essencial para adquirir o capital intelectual em desenvolvimento web, pois durante esse período foi necessário fazer uso das tecnologias usadas no desenvolvimento do web site desse projeto.

Entretanto melhorias podem ser feitas em relação ao sistema apresentado nessa documentação:

Distribuir o serviço web para outras plataformas, para o usuário ter acesso aos dados de consumo do dispositivo móvel, onde desejar. Com relação ao *design* da interface web, seria interessante deixar o usuário escolher o mês, semana ou dia no qual deseja ver os dados expostos, extraindo o máximo possível de informações dos dados colhidos. Informações de custos, na moeda nacional, respeitando o quanto custa o kWh na região em que o usuário vive, é uma melhoria para maior conveniência e comodidade ao usuário.

Uma proposta de trabalhos futuros, que agregaria grande valor ao sistema deste projeto de conclusão de curso, é um algoritmo para análise dos dados coletados, observando tendências e padrões de consumo, para ofertar ao usuário o máximo de conhecimento sobre seu consumo de energia elétrica, assim como sugestões para poupar energia, esta proposta deixaria o sistema mais inteligente e proativo.

Com relação ao equipamento, trocar o protocolo HTTP pelo MQTT pode ser benéfico, pois este protocolo é mais leve e adequado para equipamento com pouco espaço disponível para manipular, deixando mais viável para tecnologias IoT. Por fim, complementar o sistema com equipamentos capaz de interromper o consumo de energia dos equipamentos monitorados, através da aplicação, é uma proposta de melhoria para dar mais controle para o usuário sobre seu consumo.

REFERÊNCIAS BIBLIOGRÁFICAS

ALBATROSS. ***A Brief Description***, 2012. Disponível em: <http://www.cplusplus.com/info/description>. Acesso em 28 Jan. 2019.

AMAZON. **Armazenamento na nuvem**, 2017. Disponível em: <https://aws.amazon.com/pt/what-is-cloud-storage>. Acesso em 28 Jan. 2019.

AMAZON. **O que é NoSQL?**, 2018. Disponível em: <https://aws.amazon.com/pt/nosql/>. Acesso em 28 Jan. 2019.

API. In.: FOLDOC Free On-Line Dictionary Of Computing, 1995. Disponível em: <http://foldoc.org/API>. Acesso em 20 mar. 2019

ARDUINO. ***Getting Started with Arduino and Genuino products***, 20 Out. 2017. Disponível em: <https://www.arduino.cc/en/Guide/HomePage>. Acesso em 20 Fev. 2019.

AXIOS. **Axios**, 2019. Disponível em: <https://github.com/axios/axios>. Acesso em 28 Jan. 2019.

AYERST, Jeremy. **React Chart.js 2**, 2019. Disponível em: <https://github.com/jerairrest/react-chartjs-2>. Acesso em 28 Jan. 2019.

BRESCANSIN, Luiz Marco. **Física Geral lll – Aula 9 – Campos Magnéticos Produzidos por Correntes – Parte 2**, 19 Jun. 2013. (47m15s). Disponível em: <https://www.youtube.com/watch?v=qtxUAEoYGiM&t=158s>. Acesso em 06 Abr. 2019.

DOMO. **Data Never Sleeps 5.0**: How much data is generated every minute?, 2017 Disponível em: <https://www.domo.com/learn/data-never-sleeps-5?aid=ogsm072517_1&sf100871281=1>. Acesso em 10 Jan. 2019.

ESP8266 ARDUINO CORE. **ESP8266WiFi *library***, 2017. Disponível em: <https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html>. Acesso em 10 Fev. 2019.

EVANS, Dave. **A Internet das Coisas: Como a próxima evolução da Internet está mudando tudo**, 2011. 13p. Disponível em: <https://www.cisco.com/c/dam/global/pt_br/assets/executives/pdf/internet_of_things_iot_ibsg_0411final.pdf>. Acesso em 10 Jan. 2019.

EXPRESS. ***Production best practices: Performance and reliability***, 24 Dez. 2015. Disponível em: <https://expressjs.com/en/advanced/best-practice-performance.html>. Acesso em 10 Mar. 2019.

FACEBOOK. Create React App. ***Creating a Production Build***, 2019. Disponível em: <https://facebook.github.io/create-react-app/docs/production-build>. Acesso em 12 Mar. 2019.

FACEBOOK. **React: *A* JavaScript *library for building user interfaces***, 13 Jun. 2016. 19p. Disponível em: <https://reactjs.org/>. Acesso em 28 Jan. 2019.

FELIPEFLOP. **Como fazer um medidor de energia elétrica com Arduino**, 11 Fev. 2015. Disponível em: <https://www.filipeflop.com/blog/medidor-de-energia-eletrica-com-arduino>. Acesso em 28 Abr. 2019.

FIREBASE. **Firebase *Authentication***, 18 Mai. 2016. Disponível em: <https://firebase.google.com/docs/auth/?hl=pt-br>. Acesso em 28 Jan. 2019.

FIREBASE. **Firebase *RealTime Database***, 2016. Disponível em: <https://firebase.google.com/docs/database>. Acesso em 28 Jan. 2019.

FIREBASE-ARDUINO. ***Class Documentation***, 2015. Disponível em: <https://firebase-arduino.readthedocs.io/en/latest/#>. Acesso em 20 Fev. 2019.

GARTNER. ***Big data***, 21 Dez. 2016. Disponível em: <https://www.gartner.com/it-glossary/big-data/>. Acesso em 10 Mar. 2019.

GARTNER. ***Hype Cycle for Emerging Technologies***, Ago. 2018. Disponível em: <https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018>. Acesso em 10 Jan. 2019.

GARTNER. ***Machine Learning***. 16 Ago. 2016. Disponível em: <https://www.gartner.com/it-glossary/machine-learning/>. Acesso em 10 Mar. 2019.

GIT. ***About***, 04 mai. 2012. Disponível em: <https://git-scm.com/about>. Acesso em 28 Jan. 2019.

GITHUB. GitHubGuides. ***Hello World***, 07 Abr. 2016. Disponível em: <https://guides.github.com/activities/hello-world/>. Acesso em 28 Jan. 2019.

GOOGLE CLOUD PLATFORM. **IoT *at the Edge: Bringing intelligence to the edge using Cloud* IoT (*Cloud Next '18*)**, 2018. (40m23s). Disponível em: <https://www.youtube.com/watch?v=-T9MNR-Bl8I>. Acesso em: 19 jan. 2019.

GOOGLE. ***Google Cloud Platform***, 2019. Disponível em: <https://edu.google.com/intl/pt-BR/products/google-cloud-platform/?modal_active=none>. Acesso em 28 Jan. 2019.

HUNG, Mark. ***Leading the* IoT: Gartner *Insights on How to Lead in a Connected World***, 2017. 29p. Disponível em: <https://www.gartner.com/imagesrv/books/iot/iotEbook_digital.pdf>. Acesso em 10 Jan. 2019.

IDC. IDC Media Center. ***IDC Forecasts Worldwide Spending on the Internet of Things to Reach $745 Billion in 2019, Led by the Manufacturing, Consumer, Transportation, and Utilities Sector***, Framingham/MA-EUA, 03 jan. 2019. Disponível em: <https://www.idc.com/getdoc.jsp?containerId=prUS44596319>. Acesso em 10 Jan. 2019.

JSON. **Introdução ao JSON**. 03 Jul 2007. Disponível em: <https://www.json.org/json-pt.html>. Acesso em 03 Mar. 2007

LOG. In.: FOLDOC Free On-Line Dictionary Of Computing, 2009. Disponível em: <http://foldoc.org/log>. Acesso em 20 mar. 2019

MATERIAL-UI. **MATERIAL-UI**, 15 Nov. 2014. Disponível em: <https://material-ui.com/>. Acesso em 28 Jan. 2019.

MDN WEB DOCS. **Introdução Express/Node**, 18 Mar. 2019. 19p. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Express_Nodejs/Introdução>. Acesso em 20 Mar. 2019.

MICROSOFT. **Visual Studio Code**, 2019. Disponível em: <https://code.visualstudio.com>. Acesso em 28 Jan. 2019.

NODE.JS. ***About*** **Node.js**, 2016. Disponível em: <https://nodejs.org/en/about>. Acesso em 28 Jan. 2019.

NODE.JS. Node.js v12.2.0 *Documentation*. **Console**, 19 Jul. 2013. Disponível em: <https://nodejs.org/api/console.html#console_console_1>. Acesso em 10 Mar. 2019.

NPM. ***About*** **npm**, 2014. Disponível em: <https://www.npmjs.com/about>. Acesso em 28 Jan. 2019.

OAUTH. **Introduction**, 07 Set. 2007. Disponível em: < https://oauth.net/about/introduction/>. Acesso em 10 Mar. 2019.

OLIVEIRA, Sergio. **Internet das Coisas com ESP8266, ARDUINO, E RASPBERRY PI**. 1ª Edição, São Paulo, Editora: Novatec Editora Ltda., 2017. 235p. (15 capítulos)

PÉREZ, Xose. **Emonliteesp:** *Current measure library for* ESP8266 *based board*, 30 Out. 2017. Disponível em: <https://bitbucket.org/xoseperez/emonliteesp/src/master>. Acesso em 15 fev. 2019.

PRESSMAN, R.S. **Engenharia de Software:** Uma abordagem Profissional. 7ª Edição, São Paulo, Editora: Makron Books, 2011. 889p. (32 capítulos)

REACT TRAINING. REACT ROUTER. ***Philosophy***, 14 Jun. 2017. Disponível em: <https://reacttraining.com/react-router/core/guides/philosophy>. Acesso em 28 Jan. 2019.

REDHAT. MIDDLEWARE. **O que é *middleware?***, 2018. Disponível em: <https://www.redhat.com/pt-br/topics/middleware/what-is-middleware>. Acesso em 28 Jan. 2019.

RICHARDSON, Leonard; RUBY, Sam. ***RESTful Web Services***, 1ª Edição, Sebastopol/CA, Editora: O'Reilly Media, 2007. 411p. (12 capítulos)

SCRUM. ***The Scrum Guide*****:** The Definitive Guide to Scrum**:** The Rules of the Game, 2017. 19p. Disponível em: <https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf>. Acesso em 20 Jan. 2019.

SOMMERVILLE, I. **Engenharia de Software**. 8ª Edição. Editora: Pearson Addison-Wesley. São Paulo, 2007. 552p. (32 capítulos)

SUAVÉ, J. Philippe. **O que é um *framework*?**, 1 Fe. 2001. Disponível em: < http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>. Acesse em 23 Mar. 2019.

SYSTEMD. **Systemd *System and Service Manager***, 2018. Disponível em: <https://www.freedesktop.org/wiki/Software/systemd/>. Acesso em 28 Jan. 2019.

TELECO. **LAN / MAN Wireless I:** Redes sem Fio, 06 Ago. 2012. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialrwlanman1/pagina_2.asp>. Acesso em 17 Mar. 2019.

WINSTONJS. **Winston:** A logger for just about everything, 29 jan. 2019. Disponível em: <https://github.com/winstonjs/winston>. Acesso em 15 Fev. 2019.