UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

THARYCK SOUTO VASCONCELOS

SISTEMA ELETRÔNICO PARA GERENCIAMENTO INDIVIDUALIZADO DE CONSUMO DE ÁGUA

CAMPINA GRANDE - PB

2019

THARYCK SOUTO VASCONCELOS

SISTEMA ELETRÔNICO PARA GERENCIAMENTO INDIVIDUALIZADO DE CONSUMO DE ÁGUA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Professora Dra. Joseana Macêdo Fechine Régis de Araújo.

CAMPINA GRANDE - PB

2020

V331s Vasconcelos, Tharyck Souto.
Sistema eletrônico para gerenciamento individualizado de consumo de água. / Tharyck Souto Vasconcelos. – 2019.

9 f.

Orientador: Prof. Dr. Joseana Macêdo Fechine Régis de Araújo.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Consumo de água - medição. 2. Medição individualizada de consumo hídrico. 3. Sistema de gerenciamento de consumo de água. 4. Sensor de vazão. I. Araújo, Joseana Macêdo Fechine Régis de. II. Título.

CDU:004.6(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

THARYCK SOUTO VASCONCELOS

SISTEMA ELETRÔNICO PARA GERENCIAMENTO INDIVIDUALIZADO DE CONSUMO DE ÁGUA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Professora Dra. Joseana Macêdo Fechine Régis de Araújo
Orientadora – UASC/CEEI/UFCG

Professor Dr. Elmar Uwe Kurt Melcher
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de novembro de 2019.

CAMPINA GRANDE - PB

Sistema Eletrônico para Gerenciamento Individualizado de Consumo de Água

Trabalho de Conclusão de Curso

Tharyck Souto Vasconcelos
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
tharyck.vasconcelos@ccc.ufcg.edu.br

1. Resumo

A Lei nº 13.312, de 12 de julho de 2016, altera a Lei nº 11.445, de 5 de janeiro de 2007, que estabelece diretrizes nacionais para o saneamento básico, tornando obrigatória a medição individualizada do consumo hídrico nas novas edificações condominiais. Porém, tal lei não contempla as edificações construídas antes dessa data. Neste Contexto, este trabalho apresenta a proposta de um sistema de gerenciamento de consumo individualizado eletrônico (SGCA), levando em conta o consumo real de cada imóvel, bem como facilitando a leitura do volume gasto em cada ponto de medição.

2. Introdução

No estado da Paraíba, estima-se que em grande parte dos prédios antigos o consumo de água não possui medição individualizada. Desta forma, condôminos que possuem baixo consumo terminam por custear o gasto daqueles que possuem uma família mais numerosa, ou que não fazem um bom uso deste bem. Além disto, problemas como vazamentos nas tubulações são mais difíceis de serem identificados.

Por ser um recurso finito, com alto custo atrelado, e que o seu mau uso gera consequências para toda a sociedade, torna-se relevante que ações sejam tomadas, visando à redução de custos para o condomínio, redução de fraudes e tornando justa a divisão entre os condôminos. Porém, tais medidas esbarram geralmente no alto custo de implantação, dificuldades de engenharia, adequações a novas normas técnicas, além de todos os incômodos associados a uma reforma e adaptação de tamanha complexidade.

Diante deste cenário, surge a ideia de usar a tecnologia para tornar possível tal medida, visando facilitar a individualização, controle de consumo, e identificação de possíveis vazamentos. Para tanto, a proposta consiste em utilizar um sensor de vazão na tubulação de água na entrada de cada residência e em pontos estratégicos do condomínio, bem como utilizar um sistema de informação que gerencie e torne possível a leitura, o controle e a consulta para os usuários do sistema.

3. Solução

Nesta seção, serão descritos os elementos que compõem a solução proposta .

3.1 Descrição

O Sistema de gerenciamento de consumo de água (SGCA) é um sistema de informação que realiza a leitura de dados de sensores capazes de contabilizar o consumo de água por residência, conectados na mesma rede, de modo a tornar justo a divisão de valor entre os proprietários de um mesmo condomínio. A ferramenta viabiliza a contabilidade deste consumo de maneira automática, realizando requisições para os endereços dos sensores cadastrados, dispensando o uso de mão de obra, bem como equipamentos especializados para a medição. Um usuário que tenha privilégios de administrador pode cadastrar novos moradores, residências, medidores digitais e outros novos usuários. Cada morador consegue visualizar o consumo de sua residência, as contas geradas e a média do consumo dos últimos meses.

3.2 Arquitetura

Por se tratar de uma aplicação web com persistência de dados e um protótipo de hardware para o medidor de consumo digital, o projeto foi dividido em três módulos, sendo estes: *front-end*, *back-end* e protótipo de hardware. O *back-end* é a parte do servidor, responsável por gerir os dados: manipular, calcular, salvar, recuperar, servir, etc. O *front-end* é a parte da aplicação que roda no *browser*, sendo a interface que permite ao usuário interagir com o sistema. O protótipo de hardware é a parte física responsável por contabilizar o consumo efetivo por residência e que disponibiliza tal informação para o sistema em si.

3.2.1 Tecnologias do *Back-end*

A tecnologia escolhida para o lado servidor foi o Node.js, com o *framework* Express. Ambos possuem uma documentação bastante detalhada e amplo suporte da comunidade.

Para persistência dos dados, foi utilizado o MongoDB, que é um banco de dados NoSQL orientado a documentos, cujos dados são representados no formato JSON (*JavaScript Object Notation*) e, assim como o Node.js, de simples de instalação e utilização.

3.2.2 Estrutura do *Back-end*

O lado servidor da aplicação provê uma API (Application Programming Interface) REST (*Representational State Transfer*), que é consumida pelo lado do cliente. Como é possível observar na Figura 1, o código foi organizado em cinco seções principais, a saber: config, controllers, middlewares, models e routers.

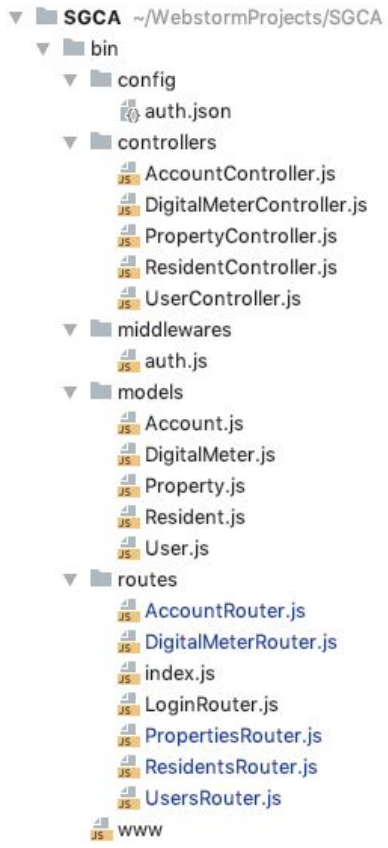

Figura 1 - Estrutura de diretórios do servidor.

O seção *config* é responsável por algumas das configurações da aplicação, na qual ficam contidos arquivos como o 'auth.json', que guarda uma palavra chave criptografada, que compõe o token gerado para cada usuário.

A seção *controllers* é responsável pela parte lógica do servidor e processa as chamadas HTTP (*Hypertext Transfer Protocol*). Estes possuem funções que são executadas de acordo com a rota do servidor que foi requisitada.

No arquivo UserController são processadas as chamadas de login e CRUD (acrônimo do inglês *Create*, *Read*, *Update and Delete*) de usuários. DigitalMeterController lida com o CRUD dos medidores digitais, assim como realiza requisições e atualização dos respectivos consumos. PropertyController lida com o CRUD das propriedades, bem como faz associação dos medidores digitais e dos moradores com esta. AccountController lida com o CRUD das contas processadas de cada propriedade e ResidentController cadastra um morador, associando a sua conta de usuário e sua residência. Os *middlewares* são responsáveis pela parte lógica do servidor.

Antes de uma requisição chegar em um *controller*, essa pode passar por um *middleware*, que verifica se o usuário está logado no sistema, por exemplo, como no caso do arquivo auth.js. Os *models* são responsáveis por definir o esquema de cada modelo que será usado no banco de dados e alguns comportamentos específicos do modelo. O modelo de usuário possui os atributos name, email, privileges, password, enabled, createdAt e updatedAt, e seus respectivos tipos são String, String, Boolean, String, Boolean, Date e Date, como apresentado na Figura 2.

```
const UserSchema = new mongoose.Schema({
    name: {
        type: String,
        require: true,
    },

    email: {
        type: String,
        require: true,
        unique: true,
        lowercase: true,
    },

    privileges: {
        type: Boolean,
        require: true,
        default: false,
    },

    password: {
        type: String,
        require: true,
    },

    enabled: {
        type: Boolean,
        default: false,
        require: true,
    },

    createdAt: {
        type: Date,
        default: Date.now(),
    },

    updatedAt: {
        type: Date,
        default: Date.now(),
    },

    residents: {
        type: String,
        ref: 'Resident'
    } ,

});
```

Figura 2 -Modelo do usuário.

Para facilitar a comunicação do servidor com o MongoDB, foi utilizada a biblioteca mongoose. Nos *routers* são definidos os *endpoints* do servidor e cada rota aponta para uma função de um *controller*.

3.2.3 Autenticação de Usuários

Como a API REST é *stateless*, ou seja, não guarda estado e não possui sessões, a autenticação de usuários é feita através de tokens JWT (*JSON Web Token*). Ao realizar o login, o cliente recebe um token de autenticação gerado do lado servidor e o guarda. Toda chamada que for feita do lado do cliente, deve enviar o token no cabeçalho Authorization, para que esse seja validado e seja possível identificar qual usuário está logado e, prosseguir com o processamento da chamada. Cada token transporta informações criptografadas, como o identificador do usuário e, se este possui privilégios e o seu tempo de validade. Para facilitar essa implementação, as bibliotecas JSON Web Token e bcript foram utilizadas.

3.2.4 Tecnologia do *Front-end*

A tecnologia escolhida para ser utilizada do lado do cliente foi o ReactJs, que é uma biblioteca JavaScript de código aberto, que além de ter uma documentação bastante detalhada, possui uma grande quantidade de componentes já desenvolvidos, o que agiliza consideravelmente a implementação de uma solução. Uma das suas maiores vantagens é a utilização de funções JavaScript dentro do código HTML, facilitando as ações do desenvolvedor, pois este não precisa utilizar diferentes linguagens e tecnologias para uma mesma aplicação.

3.2.5 Estrutura do *Front-end*

O código do lado cliente da aplicação foi organizado em *components*, *pages* e *services*. Nesses, ficam os HTML, CSS, Arquivos JavaScript e a lógica de todas as páginas do sistema.

Em *pages*, ficam as respectivas páginas e lógicas, que permitem ao desenvolvedor associar determinados comportamentos a elementos HTML. No *front-end* todas as páginas são protegidas e só podem ser acessadas se o usuário estiver logado. Os *services* são códigos que interceptam todas as chamadas HTTP feitas e acrescentam o token de autenticação no cabeçalho, caso o usuário esteja logado, além de verificar erros nas respostas do servidor. Os *pages* definem as telas da aplicação e possuem os mesmos atributos dos modelos do servidor. No arquivo app.js todas as dependências da aplicação são injetadas e no routes.js as rotas são definidas e cada uma delas aponta para uma página do sistema. Os *components* são relativos à estrutura principal da aplicação, considerando que o ReactJs é um *framework* que adota princípios SPA (*Single Page application*), fazendo com que apenas parte da página seja recarregada, conforme a interação do usuário, o que torna a experiência mais agradável e fluida.

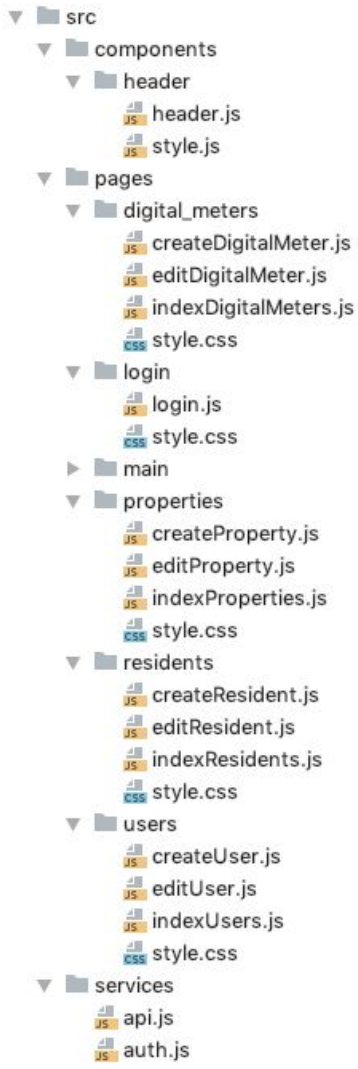

Figura 3 - Estrutura de diretórios do *front-end*.

3.2.6 Protótipo de Hardware

Para a construção do protótipo de hardware foi utilizado um dispositivo ESP8266 (Figura 4), associado a um sensor de vazão (Figura 5). Este dispositivo (chip) foi configurado como um servidor HTTP, que provê em um determinado IP e porta pré-configurados uma saída em formato JSON, e foi programado para ler, em uma de suas entradas de sinal, o consumo medido pelo sensor. O protótipo (Figura 6) foi alimentado, inicialmente, por uma conexão USB de 5V, tendo sua tensão posteriormente convertida para 3,3V, necessários para o funcionamento do chip.

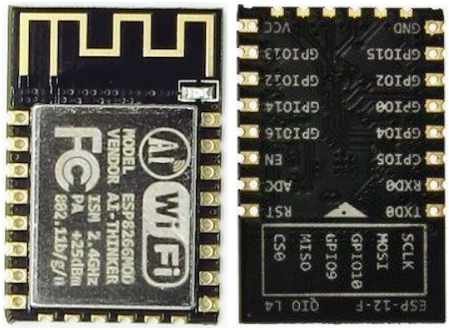

Figura 4 - Dispositivo ESP8266 (modelo ESP-12-F).

Figura 5 - Sensor de vazão.

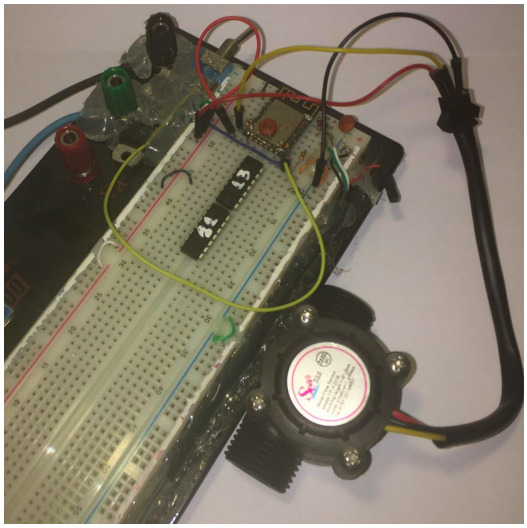

Figura 6 - Montagem do protótipo de hardware.

3.2.7 Lógica da Aplicação

O sistema exige que o usuário esteja cadastrado. Para tanto, um usuário com privilégios de administrador deve informar seu nome, um e-mail, e uma senha. Além disto, o usuário que possuir privilégios de administrador, pode gerar as contas mensais de todos os imóveis cadastrados. Para cadastrar um medidor digital, o usuário com privilégios de administrador preenche um pequeno formulário, indicando um número, um endereço IP, uma porta de serviço, o MAC Adress do chip e se este está habilitado. Esses dados ficam salvos em um objeto do tipo *DigitalMeter*, que é um modelo, e seus atributos podem ser vistos na Figura 7. O objeto *properties* é um modelo referente à residência em que o medidor estará instalado, o qual armazena o número do imóvel, bloco, quadra e se está habilitado.

Cada residência, por sua vez, possui moradores associados, que terão acesso ao consumo registrado pelo medidor e suas respectivas contas mensais.

Figura 7 - Tela de medidores cadastrados.

3.2.8 Ambiente de Produção

O servidor da aplicação é hospedado em um servidor local, que gerencia toda a parte da infraestrutura automaticamente. Essa plataforma foi escolhida principalmente pelo fato de as requisições realizadas ao protótipo serem direcionadas a uma rede local, o que facilita ao usuário final configurar e cadastrar novos dispositivos. Para hospedar o banco de dados, foi escolhido o MongoDB Atlas, pois é o serviço oficial do MongoDB e possui um plano grátis. Apesar de possuir um limite de 512 MB de armazenamento, este é suficiente para o projeto. Além disto, também possui gerenciamento automático da infraestrutura e provê réplicas do banco, aumentando a disponibilidade e a tolerância a falhas.

4. Avaliação

O SGCA é voltado para condomínios que ainda não possuem a medição individualizada de água. Porém, devido ao tempo para realização deste trabalho e ao custo de implantação, não foi possível que tal ferramenta fosse testada e validada pelo público alvo.

Foi elaborado, então, um formulário com uma série de perguntas que levam em consideração, principalmente, a usabilidade e experiência do usuário, no papel de gestor.
O formulário foi aplicado a 7 (sete) pessoas após a utilização do SGCA,o qual contém as seguintes perguntas:

1. Sua residência tem medição individualizada de água?
2. Você considera justo que pessoas que fazem um uso desigual de um bem, devem pagar o mesmo valor?
3. Você pagaria para ter um sistema de medição individualizado de consumo de água?
4. Foi fácil aprender a utilizar o aplicativo?
5. Foi fácil cadastrar um novo medidor digital?
6. Você conseguiu navegar bem pelo aplicativo?

As três primeiras perguntas deveriam ser respondidas com **sim** ou **não**. Para a primeira, 60% das respostas foram sim. Para a segunda, todas as respostas foram não e, para a terceira, todos responderam sim.
Para as três últimas perguntas foi utilizada uma escala de 0 a 10 para metrificar as respostas, em que 0 corresponde a "Discordo totalmente" e 10 a "Concordo totalmente". Estas foram bem avaliadas, resultando em uma média 8,75.
A avaliação se mostrou relevante para indicar possíveis melhorias no sistema. A melhoria do cadastro de um medidor digital e a usabilidade em uma plataforma mobile foram os pontos mais citados pelos voluntários. Porém, destaca-se, que a aplicação foi considerada útil para estas pessoas, quanto ao propósito para o qual foi desenvolvida.

5. Experiência

A seguir, será descrita a experiência adquirida ao longo do desenvolvimento do projeto.

5.1 Processo de Desenvolvimento

Para o desenvolvimento do Software foram inicialmente levantadas quais tecnologias poderiam ser utilizadas no projeto. As escolhas foram baseadas na facilidade de desenvolvimento, na comunidade existente para sanar eventuais dúvidas, e, principalmente, no desafio de aprender uma nova linguagem que está se tornando cada vez mais popular. Logo após, foi realizada uma análise de requisitos, a partir da qual foram definidas as principais funcionalidades do sistema e para cada funcionalidade, foi realizado um estudo para determinar a viabilidade técnica, considerando as tecnologias escolhidas.
Para o desenvolvimento do hardware, foram pesquisados chips que tivessem baixo consumo, combinado à fácil programação, mas que ao mesmo tempo fosse possível associá-lo a um sensor de vazão genérico. Assim, foi prototipado um hardware, com o uso de um ESP-12-F (Figura 4), bem como um esboço inicial do sistema, com o objetivo de atestar a exequibilidade do projeto.

5.2 Desafios

Associar um sensor de vazão a um ESP-12-F, programá-lo, analisar seus respectivos pontos frágeis (falta de alimentação, possíveis lacunas, etc) e tentar tornar sua configuração o mais simples possível, foi extremamente desafiador. Aliado a isto, a criação de uma interface de usuário simples, porém, visualmente agradável, fácil de ser utilizada e intuitiva, foi uma das partes mais desafiadoras do projeto (como visto na Figura 8).
Por se tratar de um produto cujos potenciais usuários não possuem habilidades para programar e configurar um chip, o SGCA precisa ser prático e o mais simples possível, para atrair usuários e tornar a tarefa a qual se propõe algo rápido e prático.

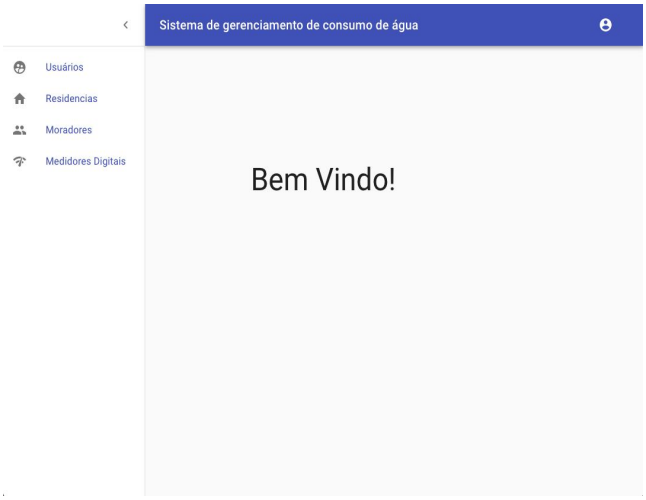

Figura 8 - Tela de início da aplicação.

5.3 Sugestões para Trabalhos Futuros

Trabalhos futuros podem estender o SGCA para uma aplicação ainda mais robusta, a partir da implementação de mais algumas funcionalidades, descritas a seguir.

- Geração de contas de consumo individual de forma automática.
- Melhorias estéticas na tela de usuário.
- Desenvolvimento de uma versão Mobile.
- Melhorias no protótipo do medidor digital, sendo essas:
  - Redundância na alimentação do medidor digital;
  - Tornar a configuração de novos medidores algo fácil e intuitivo;
  - Melhorar a segurança na requisição de consumo ao medidor.

6. Agradecimentos

Quero agradecer primeiramente a Deus, e aos meus pais que me apoiaram durante os anos de graduação, sempre guiando meus passos ao longo do caminho. À professora Joseana Macêdo Fechine Régis de Araújo, que me orientou durante este trabalho. Ao professor Tiago Massoni que me ajudou não só nesse trabalho, mas durante toda a graduação. A Eduarda Maria, que foi minha companheira fiel em todos os momentos difíceis durante o curso, e sempre conseguia me acalmar nas horas de tormenta. A todos meu amigos que direta ou indiretamente contribuíram no meu crescimento profissional. E, em especial, ao meu irmão Thulyo Savyo, que ao lado de Deus, sei que está orando por mim.

7. Referências

[1] Autenticação em Node.js com Passport – Parte 2, Disponivel em: **https://www.luiztools.com.br/post/autenticacao-em-nodejs-com-passport-2/** - Último acesso em 18/09/2019.

[2] Monitoramento de água com IoT - Parte 2, Disponivel em: **https://www.embarcados.com.br/monitoramento-de-agua-com-iot-parte-2/** - Último acesso em 19/09/2019.

[3] Planta IoT com ESP8266 NodeMCU – Parte 4, Disponivel em: **https://www.filipeflop.com/blog/planta-iot-com-esp8266-nodemcu-parte-4/** - Último acesso em 19/09/2019.

[4] Integração do Material UI com ReactJS, Disponivel em: https://blog.rocketseat.com.br/react-material-ui/ - Último acesso em 21/11/2019.

[5] Como usar o ESP-12 em casa, Disponivel em: **http://labarc.ufcg.edu.br/loac/index.php?n=OAC.ESP12** - Último acesso em 08/11/2019.

[6] NodeMCU – Como criar um Web Server e conectar a uma rede WIFI, Disponivel em: **https://blogmasterwalkershop.com.br/embarcados/nodemcu/nodemcu-como-criar-um-web-server-e-conectar-a-uma-rede-wifi/** - Último acesso em 21/11/2019.

[7] Localizador ISS: uso de uma API Web com o ESP8266, Disponivel em: **https://www.filipeflop.com/blog/localizador-iss-api-web-com-o-esp8266/** - Último acesso em 25/10/2019.

Sobre o autor:

Tharyck Souto Vasconcelos é graduando em Ciência da Computação pela Universidade Federal de Campina Grande, onde desenvolveu as habilidades de desenvolvimento e teste de software. Atualmente, trabalha como Analista de software na Accenture Brasil.