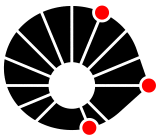

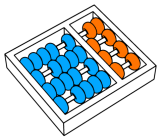

Sistema para Monitoramento Remoto da Refrigeração do Data Center do Instituto de Computação

L. C. Tonon *M. M. Yonue* *J. F. Borin* *R. Ferrari*

Relatório Técnico - IC-PFG-20-12
Projeto Final de Graduação
2020 - Agosto

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Sistema para Monitoramento Remoto da Refrigeração do Data Center do Instituto de Computação

Lucas Chinaglia Tonon* Juliana Freitag Borin† Mariane Massago Yonue‡
Rafael Ferrari§

Resumo

Este projeto tem por objetivo auxiliar os mantenedores dos ares condicionados do Data Center do Instituto de Computação (IC) da Universidade Estadual de Campinas a identificar problemas no funcionamento dos mesmos de forma rápida e visual através de gráficos e alarmes gerados por um sistema de monitoramento baseado em IoT (Internet of Things).

A motivação para o desenvolvimento deste trabalho veio de uma falha - não detectada rapidamente por falta de um sistema de notificação apropriado - ocorrida nos ares condicionados do IC. Considerando que um defeito nestes aparelhos afetaria a capacidade de manter a sala resfriada, isto poderia causar danos aos equipamentos do Data Center com o aquecimento progressivo das máquinas, podendo queimar processadores dos servidores caso não seja corrigido rapidamente [1], o que acarretaria custos à universidade para os substituir.

A solução propõe adicionar sensores de temperatura e umidade, fornecendo auxílio na detecção de comportamentos anômalos no resfriamento do ambiente monitorado, de forma a enviar alarmes via e-mail em caso de medições fora de um intervalo previamente definido. Também é esperado que, com o tempo e uma base de dados de medições cada vez mais completa - adquiridas em situações, épocas e climas diferentes - seja possível desenvolver projetos capazes de prever falhas no sistema e até mesmo diagnosticar suas causas.

1 Introdução

Data Centers são responsáveis por armazenar e lidar com grandes quantidades de dados de forma a serem usados no desenvolvimento de soluções computacionais para resolver

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
†Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.
‡Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP
§Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, 13083-852 Campinas, SP

problemas do dia a dia. Para isto, existem critérios relacionados a infraestrutura do ambiente em que estão instalados que são necessários para garantir o funcionamento em tempo integral dos aparelhos neles inseridos.

Fatores como alimentação de energia, refrigeração adequada com sistema de backup, sistemas de notificação e supressão de incêndios e outras adaptações de infraestrutura são requisitos básicos de um Data Center para garantir que falhas eventuais não afetem os serviços que dependem destes dados e servidores [2]. Tratando-se do sistema de refrigeração, existem opções no mercado já focadas no contexto de ares condicionados, trazendo mais de um aparelho com um modelo de backup entre eles de modo que exista um reserva e, caso um ar condicionado pare de funcionar, um dos reservas toma seu lugar de forma automática sem necessitar da intervenção de um dos indivíduos responsáveis pelo funcionamento do local.

No caso do Instituto de Computação da Unicamp segue-se um sistema similar, em que o resfriamento do Data Center possui no total três ares condicionados. A configuração determina que duas máquinas sempre devem estar ativas e a terceira permanece em 'estado de espera' para que sirva como substituta em caso de falha de uma das primeiras. Ainda que isto já consiga prover uma boa cobertura de problemas que podem vir a ocorrer no controle de temperatura da sala, outros erros comuns em ares condicionados ou até mesmo neste controle de backup podem causar falhas inesperadas que poderiam ser solucionadas provendo um modo eficaz de monitoramento das condições de temperatura e umidade dentro do ambiente monitorado, de modo a ajudar os mantenedores a terem uma noção da existência de um problema recorrente através da análise de dados exibidos de forma gráfica.

2 Justificativa

Recentemente houve uma falha no funcionamento dos ares condicionados dentro do Data Center do Instituto de Computação, em que um deles ficou inativo e o sistema de backup não foi acionado, fazendo com que apenas um dos três aparelhos estivesse funcional. Por falta de um sistema de notificação e pelo fato do erro ter acontecido durante um final de semana, esta falha só foi detectada depois de um dia, o que resultou em altas temperaturas dentro do Data Center e no desligamento automático de vários servidores. Este fato evidenciou a necessidade de se ter um sistema mais robusto para detecção e predição de falhas deste tipo no Data Center.

3 Objetivos

Este projeto tem como objetivo o projeto e implementação de um sistema de baixo custo para monitoramento e detecção de falhas do sistema de refrigeração do Data Center do Instituto de Computação da Unicamp. Mais especificamente, espera-se:

- desenvolver um protótipo de dispositivo composto por sensores para coleta de dados de temperatura, umidade e presença no Data Center e capaz de enviar os dados coletados para uma plataforma em nuvem;

– desenvolver um sistema de notificação de falhas baseado nos dados coletados no Data Center;

– desenvolver uma aplicação web para monitoramento em tempo real dos dados coletados no Data Center;

4 Desenvolvimento

O primeiro passo do projeto consistiu em discutir e averiguar com os mantenedores do Data Center os problemas e necessidades em relação ao monitoramento dos ares condicionados. A conclusão foi de que seria interessante encontrar uma forma de, dadas as informações - como temperatura de certos componentes, umidade total da sala, vibração e ruído do aparelho e afins - aferidas por diferentes sensores dentro de um ambiente refrigerado, prever e diagnosticar erros no funcionamento dos ares condicionados. Contudo, através do estudo de trabalhos disponíveis na literatura [3] [4] ficou claro que, embora existam métodos de detecção e diagnóstico utilizando parâmetros como a temperatura em vários pontos de um equipamento, que apresentam seus resultados dentro de uma margem bastante confiável, a complexidade para tais previsões fugiria ao escopo do projeto pois, além do desenvolvimento deste ter se dado a distância[1], a ausência de um conjunto de dados referentes às medições dos aparelhos em diferentes estágios de uso - entende-se por 'estágio de uso' o funcionamento do aparelho de ar condicionado ao longo do tempo, quando novo, após algumas semanas de uso, momentos antes de ocorrer uma falha, após manutenção - torna-se fator determinante para a adaptação da ideia original.

Para o desenvolvimento do projeto foi idealizada uma arquitetura com a qual fosse possível incorporar múltiplos sensores, possibilitando a obtenção de dados através de requisições HTTP para uma API - *Application Programming Interface* - e, em posse destas informações, permitir a exibição de gráficos e outras informações em uma página web.

A plataforma em nuvem escolhida para armazenar e tratar os dados foi a Konker [5] devido ao fato de se tratar de um serviço confiável e conhecido, visto que já existem projetos baseados em Internet das Coisas dentro do contexto da iniciativa Smart Campus [6] na UNICAMP que a utilizam. Através de seu uso, os dispositivos eletrônicos tornam-se aptos a receber e enviar dados para um servidor, e então uma aplicação web poderá realizar requisições - via REST API - para a plataforma, retornando os dados de forma gráfica - considerada, do ponto de vista de *User Interface* e *User Experience*, mais intuitiva.

4.1 Arquitetura do Sistema

O projeto foi pensado de forma a ser modular, considerando a necessidade de implementação de circuitos diferentes para contextos distintos e a implementação de uma interface visual que se comunica com um serviço de back-end. Portanto, a arquitetura básica projetada inicialmente consistia de um conjunto de sensores e microcontroladores, responsáveis por todas as medidas e envio de dados para um back-end, que deveria então prover essas

[1] Devido a pandemia de COVID-19, não foi possível acessar a sala do Data Center na Unicamp

informações para um front-end capaz de exibí-las de maneira clara e intuitiva, auxiliando no monitoramento. Esta arquitetura é apresentada na Figura 1.

Para realizar o monitoramento de temperatura e umidade optou-se por associar ao microcontrolador o sensor DHT11 [7], que permite medição simultânea de ambas as variáveis. Levando em consideração o tamanho do ambiente e a existência de mais de um ponto de interesse a ser monitorado, ficou evidente que seriam necessários múltiplos microcontroladores, e, por consequência, um código capaz de lidar com as diversas medições provindas de diferentes sensores DHT11. Pensando nisso, o código foi estruturado de forma a ser permeável à variação do número de dispositivos - aqui entende-se por dispositivos microcontroladores associados a sensores -, tornando simples o processo de configuração. Para tanto, o método escolhido foi através do uso de "variáveis de ambiente" que, no caso deste projeto, consistem em constantes definidas em um arquivo *env*.h e importadas no código do microcontrolador; desse modo, se for desejado reconfigurar o comportamento do circuito, estas mudanças serão concentradas, essencialmente, na alteração do valor destas constantes. Por exemplo, para adicionar um novo dispositivo capaz de se comunicar com a plataforma Konker será necessário alterar, no arquivo *env*.h, os valores correspondentes às credenciais do dispositivo para conexão MQTT e em seguida realizar a compilação do código.

Além do microcontrolador e do sensor DHT11 também foi incorporado um sensor de presença PIR - HC-SR501 [8]. De menor relevância no contexto deste trabalho, e considerado complementar, apenas um sensor foi pensado a ser posicionado próximo à entrada da sala para indicação de fluxo de pessoas adentrando o recinto - nota-se aqui que o objetivo deste sensor não é trabalhar com um quantitativo de pessoas presentes na sala, mas sim, indicar se houve, ou não, movimentação em um certo momento do dia.

Definido o modo a serem coletadas as medições, começou-se a pensar onde estes dados seriam salvos para que posteriormente requisições de software pudessem acessá-los. A ideia original pressupunha a criação de front-end e back-end, no entanto, por motivos que serão justificados mais detalhadamente na Seção 4, apenas a implementação do front-end foi necessária para tornar possível a obtenção dos dados dos sensores a partir de seu local de armazenamento. Em grande parte essa simplificação se deve à escolha da arquitetura ao fazer uso da plataforma Konker para o armazenamento de dados. A Figura 2 ilustra a arquitetura final.

4.2 Plataforma em nuvem

A escolha de utilização da plataforma em nuvem da Konker se deu embasada no fato de outros projetos no contexto de Smart Campus já fazerem, ou terem feito, uso deste serviço para lidar com os dados provindos de dispositivos eletrônicos dentro da universidade. A vantagem imediata de utilizar esta plataforma está na diminuição no esforço de implementação, não sendo necessário projetar a conexão entre dispositivos e um back-end especializado, tão pouco um banco de dados para armazenar as medições. Dessa forma, expande-se a possibilidade de melhor investimento de tempo de desenvolvimento no aprimoramento do modo como os dados são apresentados ao usuário e na estruturação do código, para tornar simples a configuração de novos sensores, assim permitindo que o sistema seja mantido inteiramente pelos funcionários do Instituto de Computação, e facilitando em caso de desejo de expansão

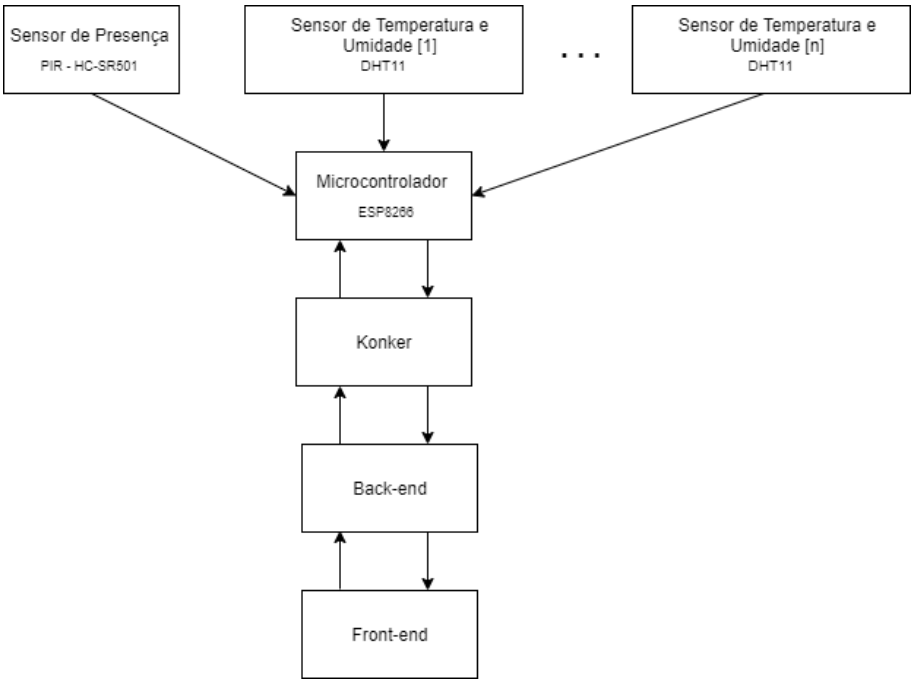

Figura 1: Arquitetura inicial

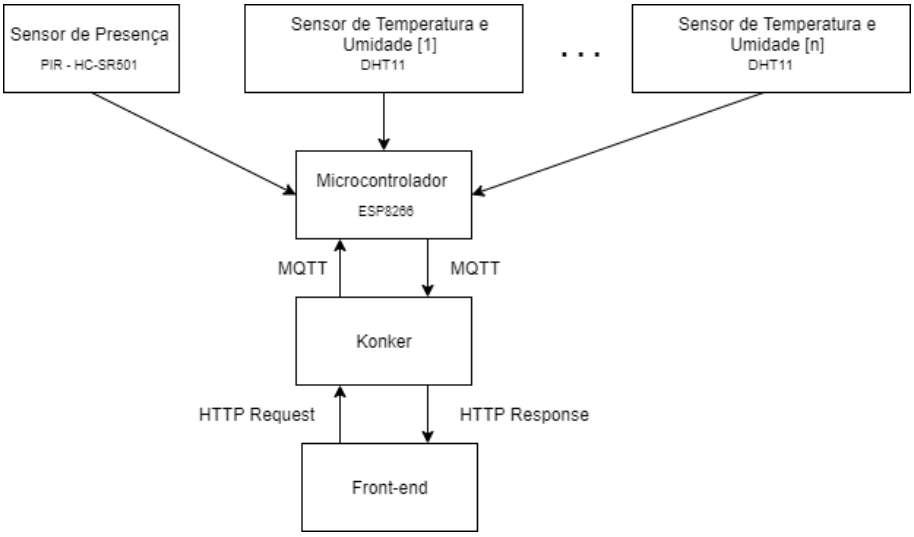

Figura 2: Arquitetura final sem back-end

futura do projeto.

Além de permitir salvar os dados das medições dos sensores DHT11 e PIR - HC-SR501, a Konker possui uma API REST [9] capaz de acessar os dados através de requisições HTTP diretamente a partir do front-end. O serviço também permite o gerenciamento de diversos

dispositivos IoT, armazenamento de dados, conexão e envio/recebimento de mensagens via HTTP ou MQTT, entre outras funcionalidades.

4.3 Hardware

Para montagem do sistema físico foram utilizados kits disponibilizados pelos orientadores. Dentre os componentes disponíveis haviam jumpers, placas, sensores de presença, umidade, temperatura, e outros não utilizados neste projeto.

A construção dos circuitos foi feita utilizando um microcontrolador NodeMCU ESP8266 [10] - responsável pela comunicação dos sensores com a internet e, a partir desta, conexão com o serviço da Konker -, sensores DHT11 - para medição de temperatura e umidade - e sensor de presença PIR - HC-SR501. Além disso, para assegurar uma boa modularidade dos componentes foram criadas diversas variáveis de ambiente - responsáveis pelo armazenamento de valores como login e senha da internet, informações de conexão do dispositivo com a Konker, valores usados para definir limites do alarme e informações para envio dos e-mails - de modo a permitir que o comportamento lógico possa ser alterado sem dificuldades.

Pensando no objetivo final, foi natural - recapitulando o exposto na seção de arquitetura - pensar na separação dos componentes em duas disposições de sistemas eletrônicos, uma que consiga medir temperatura e umidade e outra que possa detectar presença dado algum movimento, posto que deverão existir múltiplos sensores de umidade e temperatura distribuídos pela sala - visando melhorar a precisão dos dados -, e apenas um sensor de presença alocado na entrada para identificar caso alguém passe pela porta.

A abordagem inicial, pesquisas, definição de tecnologias e arquitetura do projeto foram realizadas integralmente em conjunto. A partir de então as tarefas foram nominalmente alocadas - embora, como esperado de um trabalho em conjunto, interações (*pair programming*) e discussões para troca de informações, ajustes e melhorias tenham ocorrido. A seguir, são descritas as atividades de cada um dos alunos do grupo nessa parte do projeto.

4.3.1 Atividades de Mariane Massago Yonue

A aluna ficou responsável pela montagem dos circuitos de detecção de presença e mensuração de umidade e temperatura, seguida pela codificação do comportamento base destes. Tendo como guia o mapeamento dos pinos contido nos datasheets dos componentes:

- DHT11
  1. Fonte VDD 3.5V 5V
  2. Serial DATA
  3. NC, Not Connected
  4. GND, fio terra
- PIR - HC-SR501
  1. GND, fio terra
  2. Serial DATA
  3. Fonte VDD

foi feita a conexão entre sensores e microcontroladores.

No ESP8266, para a entrada de dados foram escolhidos os pinos D3 e D4, sendo que a utilização do D3 ocorre somente em caso de acoplamento de múltiplos sensores ao dispositivo. Existem três configurações distintas criadas para este projeto. Duas delas objetivam

medidas de temperatura e umidade - *single-dht11* (Figura 3) e *double-dht11* (Figura 4) -, e a última visa a detecção de presença - circuito nomeado de *single-presence* (Figura 5).

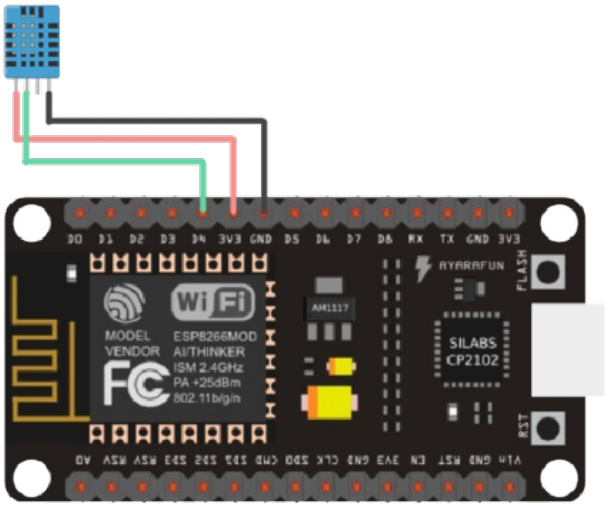

Figura 3: Montagem single-dht11, representando um único sensor DHT11 associado ao ESP8266

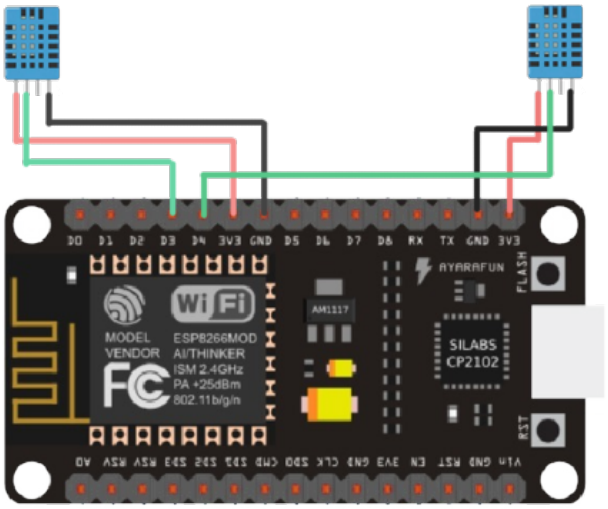

Figura 4: Montagem double-dht11, representando dois sensores DHT11 associados ao ESP8266

A criação das disposições *single* e *double-dht11* foram pensadas de forma a cobrir uma maior quantidade de cenários possíveis, bem como de mitigar os efeitos da incerteza de configuração da sala do Data Center. O microcontrolador associado a um único sensor (*single-dht11*) pode ser utilizado em casos de medições espacialmente muito distantes, já a configuração *double* foi pensada para medidas em pontos próximos, com sensores voltados a

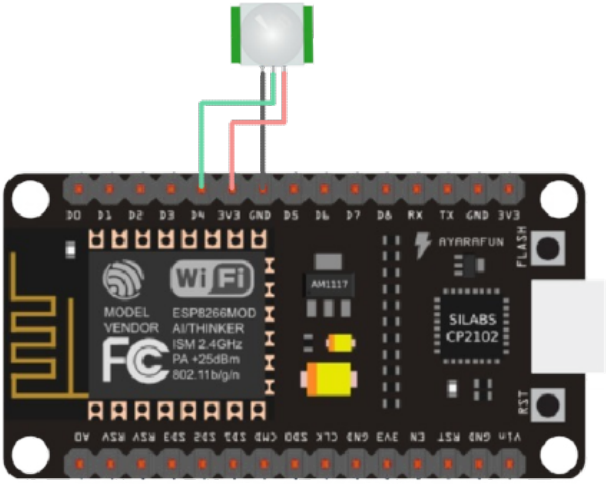

Figura 5: Montagem single-presence, representando um único sensor PIR - HC-SR501 associado ao ESP8266

seus respectivos locais de interesse mas aproveitando o mesmo microcontrolador - reduzindo os gastos com dispositivos ESP8266.

No que tange a utilização do sensor de presença, esta configuração, muito mais simples, necessita de apenas um exemplar localizado em um ponto que acuse movimentação no caso de algum indivíduo adentrar a sala. De imediato o ponto escolhido foi a porta, garantindo que toda presença seja captada.

Quanto à lógica de funcionamento, para o circuito dos sensores DHT11, o desenvolvimento assumiu, inicialmente, um único sensor. Fazendo uso da biblioteca "DHT Sensor Library". As medições são realizadas em intervalos de um minuto, e então são enviadas para a Konker, com impressão dos valores na saída serial a cada nova medida. Validado o sistema com um único sensor, foi feita a refatoração do código para aceitação de dois sensores. Já o código do circuito responsável pela detecção de presença, apesar desse ter um uso mais direto e simples, apresentou maior complexidade lógica. O principal problema consistia na dificuldade em avaliar se uma pessoa estava ou não na sala caso permanecesse sem se mover por muito tempo. Uma vez que o PIR - HC-SR501 leva em conta o movimento para acusar presença, assim que este cessa, ele deixa de sinalizar que existe alguém na sala. Para atenuar falsos negativos, foi criado um timer. Dessa forma, se houver ocorrência de movimento, o timer é disparado e considera-se que existe alguém na sala até que o tempo definido - aqui como cinco minutos, considerado adequado à movimentação esperada no Data Center - se esgote. Caso seja novamente detectado um novo movimento, a contagem do timer é reiniciada.

4.3.2 Atividades de Lucas Chinaglia Tonon

O aluno, nesta parte do projeto, focou principalmente na implementação da comunicação com a Konker e no envio de e-mails a partir dos módulos ESP8266.

Para definir a comunicação com os serviços da Konker usou-se como referência um material provido pela própria empresa em formato de vídeo educativo [11] que ensina como usar e conectar dispositivos com a aplicação. Na documentação, denota-se que o envio de mensagens dos dispositivos poderia ser feito tanto via HTTP quanto via MQTT, e considerando que o protocolo MQTT [12] foi especificamente desenvolvido para lidar com IoT, provendo um modo leve de realizar subscrição e publicação de mensagens entre componentes eletrônicos, este foi o escolhido para realizar a implementação do projeto.

A conexão é feita através do uso das bibliotecas **PubSubClient.h** e **ESP8266WiFi.h**, que são responsáveis por criar um cliente de publicação e subscrição de mensagens e pela conexão a internet, respectivamente. Para definir o cliente, é necessário ter o endereço do servidor MQTT provido pela Konker, bem como a porta a ser utilizada, ambos obtidos facilmente no próprio site da aplicação. Um exemplo do que foi feito encontra-se a seguir:

```
WiFiClient espClient;

PubSubClient client("mqtt.prod.konkerlabs.net", 8883, callback, espClient);
```

Figura 6: Objetos de conexão com a internet e canal MQTT

Com estas duas variáveis definidas, é possível conectar ao servidor MQTT da Konker usando o login e a senha definidos para o dispositivo, com cada microcontrolador tendo uma credencial própria. É possível também criar os canais de publicação e subscrição de mensagens, que são um modo de definir o contexto de uma mensagem recebida e enviada dentro da Konker para facilitar a filtragem de informações de um tipo específico diretamente pela API da plataforma. Como existem três tipos de informação distintas para serem monitoradas, foram criados os canais *temperature*, *humidity* e *presence*, sendo que cada um passa um JSON com dados que fazem sentido no contexto - valor e unidade para temperatura e umidade, e um binário para presença.

```
{
   value :  "26.03",
   unit :  "celsius",
   type :  "temperature"
}
```

Figura 7: Exemplo de um payload de temperatura

Para a implementação do envio de e-mails de notificação foram pesquisadas alternativas já existentes que permitissem mandar as informações com segurança e de modo simples para os usuários. A biblioteca mais promissora encontrada foi a EMailSender [13], cujo foco principal é o envio de e-mails usando o servidor SMTP - Simple Mail Transfer Protocol - do Gmail, mas que permite usar um construtor para definir outros servidores SMTP. Esse foi um fator determinante na escolha visto que para enviar os e-mails foi solicitada a criação

Figura 8: Exemplo de um payload de umidade

Figura 9: Exemplo de um payload de presença no caso sem ninguém na sala

de um usuário aos funcionários do Instituto de Computação para os dispositivos de alarme para que estes possam se conectar ao servidor do IC e enviar notificações para as pessoas escolhidas a partir de uma conta pré-determinada.

A definição de quem vai receber estes e-mails de notificação também foi feita pensando na necessidade de se alterar a lista no futuro, para isto existe um e-mail alias para os funcionários do Instituto de Computação que encaminha os e-mails recebidos nesta conta para os responsáveis pelo Data Center.

Com o modo de envio, e-mails e biblioteca a serem utilizados bem definidos, a lógica por trás do alarme teve que ser desenvolvida. Inicialmente pensava-se em medir os valores um a um e caso algum deles passasse dos valores limitantes definidos pelas variáveis de ambiente, um alarme seria enviado para notificar os mantenedores dos equipamentos. No entanto, discutindo com os orientadores chegou-se à conclusão de que isto não seria o ideal visto que os sensores poderiam subitamente ter algum erro de medição por alguma falha e interpretar isto como um problema com a sala de fato. Para evitar este problema foram criados vetores para temperatura e umidade que guardam as medidas obtidas de cinco em cinco segundos, dentro de um intervalo de cinco minutos, isto é, um vetor de sessenta elementos. Além destes vetores de medições, foram criadas variáveis para verificar e atualizar o índice atual da última medida obtida, assim, pode-se obter a média de todas as medidas e, ao chegar no fim do vetor e recebendo uma nova medida a ser armazenada, zera-se os índices para iniciar a escrita dos novos valores no começo dos vetores, sobrescrevendo as informações mais antigas.

Este método de medição garante que a média tomada consiga identificar problemas de forma rápida, levando menos de cinco minutos para detectar oscilações na temperatura e notificar os mantenedores do data center. Outro cuidado que foi necessário considerou a questão do spam de e-mails, isto é, caso de fato estivesse ocorrendo algum problema nos aparelhos de ar condicionado e com as medições sendo tomadas a cada cinco segundos e avaliadas para checar a necessidade dos alarmes, se a condição não fosse corrigida seriam enviadas notificações constantemente para os funcionários a cada nova medida fora do limite.

A solução para este problema foi definir um intervalo de trinta minutos para cada tipo de notificação, garantindo que caso um alarme seja enviado o dispositivo irá esperar no mínimo este tempo para enviar outro do mesmo tipo.

4.4 Back-end

A implementação do back-end do sistema, sob responsabilidade do aluno Lucas Chinaglia Tonon, visava permitir que, além de acessar a API da Konker usando um proxy - isto é, quando o front-end fizesse uma requisição para o back-end para obter informações dos sensores, ele estaria na verdade se comunicando com a API da Konker com uma camada extra de abstração entre os dois -, fosse possível realizar outras ações como alertar os usuários através de e-mails e realizar atualizações de valores dos alarmes e da lista de pessoas a serem notificadas dinamicamente.

Um protótipo deste componente da arquitetura foi feito utilizando NodeJS e Express para programação dos endpoints, e o módulo *http-proxy-middleware* [14] para realizar o proxy entre o back-end criado e os endpoints definidos pela plataforma da Konker. Esta prototipagem foi bastante útil para auxiliar o grupo a descobrir como realizar na prática as requisições, e também para entender como autorizar uma operação HTTP para a API pré-definida a partir de uma requisição antes mesmo de existir uma página web, agilizando assim o processo do front-end quando em seu desenvolvimento. Para a autorização das requisições para a Konker foi usado um token enviado via header HTTP de **Authorization** do tipo bearer. Nesta fase do desenvolvimento, o token ficava salvo como variável de ambiente no projeto NodeJS e servia exclusivamente para permitir o acesso ao serviço externo.

A partir de conversas com ambos os orientadores, e com o decorrer da implementação do front-end e do código usado nos microcontroladores, foi definido que a melhor alternativa para o envio de e-mail e lógica por trás dos alarmes deveria ser dada dentro dos próprios componentes eletrônicos para evitar que falhas em algum ponto do processo de comunicação entre os dispositivos e o back-end criado pudesse influenciar na notificação de medições fora do esperado. Assim, os microcontroladores ficaram encarregados de armazenar uma certa quantidade de informações e enviar e-mails para os usuários, dependendo dos limites definidos dentro deles em variáveis de ambiente. Graças a esta alteração, o serviço de back-end criado foi reduzido a apenas um proxy da Konker e perdeu sua relevância dentro do projeto, fazendo com que pudesse ser removido, mantendo na arquitetura apenas um front-end, com o intuito de exibir informações dos sensores em forma de gráficos. Os microcontroladores ficaram responsáveis tanto pelo envio destes dados para o banco de dados quanto pela notificação de valores inesperados.

4.5 Front-end

Para implementação da UI do website foram usadas tecnologias com as quais os alunos já haviam tido algum contato anteriormente - fosse por experiências de estágio ou por projetos pessoais. Para criação dos componentes fez-se uso da biblioteca JavaScript React [15], que permite grande flexibilidade na implementação de páginas web e também o uso de diversos módulos, como Styled Components [16] e Material-UI [17] - que trazem, respectivamente a

definição de estilos CSS usando JavaScript e componentes pré-definidos seguindo o material design. As atividades de cada membro do grupo na construção do front-end são descritas a seguir.

4.5.1 Atividades de Mariane Massago Yonue

A aluna trabalhou na implementação dos componentes de front-end, usando React e CSS - este último feito com a ajuda do módulo Styled Components -, na renderização dos dados em gráficos e também lidou com as escolhas, visuais e de comportamentos, relacionadas à experiência dos usuários do website.

A implementação do design da página passou por dois estágios de prototipação até chegar ao produto final em termos de UI e UX. A primeira versão, feita para testar a integração entre o front-end e as APIs utilizadas, bem como checar os modos de formatar os valores a serem exibidos, mostrava apenas algumas poucas informações - médias, valores máximos e mínimos -, sem ainda dispor de gráficos ou apresentar quaisquer outras informações.

Feito o primeiro protótipo, o próximo passo direcionou-se ao modo de como apresentar, dentro do website, a grande quantidade de dados provinda dos sensores. Pensando no volume dessas medidas, decidiu-se que a maneira mais apropriada de exibi-las seria por meio de gráficos. Para tal, foram pesquisadas opções de módulos que mostrassem boa compatibilidade com um serviço feito em React. Restritas as opções de componentes React específicos para exibição de gráficos, foram realizados testes práticos para checagem da documentação provida por cada módulo, pois seria necessário certo nível de compreensão de seu funcionamento para atingir o nível de personalização desejado. Além disso, critérios como simplicidade de implementação, personalização visual e design intuitivo foram considerados. Seguindo estas restrições de projeto, optou-se pela escolha do módulo *@nivo* [18].

Durante a incorporação dos gráficos ao projeto, vale citar a dificuldade relacionada à implementação de uma abcissa sobre a qual fosse possível representar, através de data e hora, o instante em que a medida foi tomada. O empecilho residia na extrema lentidão causada pela alta demanda de elementos visuais vetorizados a serem exibidos, em tentativa de renderizar todos os dados coletados. Cabe aqui a explicação para a alta quantia de medidas: inicialmente era possível definir o intervalo de início e de fim das capturas, assim, assumir-se-ia possível exibir os dados de uma semana completa - ou até intervalos de tempo maiores -, entretanto, a página não conseguiria lidar com a renderização simultânea de tantos pontos - com medidas tomadas a cada cinco segundos, um único dia teria mais de dezessete mil medidas, acumulando na semana uma quantia superior a cento e vinte mil medições. De modo a solucionar este problema, foi realizada alteração no seletor de tempo. Ao invés de possibilitar a seleção de um intervalo livre para exibição das medidas, o seletor passa a permitir o apontamento de uma data, ou seja, os dados são apresentados em intervalos fixos de vinte e quatro horas. Ademais, foi alterado o intervalo de envio entre medidas, passando de cinco segundos para cinco minutos - apesar de parecer um espaçamento demasiado grande, pensando nas situações em que possam ocorrer as variações de temperatura e umidade, a resposta ao acontecimento permanece suficientemente rápida para a dinâmica das grandezas físicas envolvidas.

Por fim, com o segundo protótipo cumprindo bem - sem falhas - suas funcionalidades, foram feitas iterações de design, relacionadas às cores, disposição de elementos e identidade visual do site, de modo a definir o layout final da página. As Figuras 10 e 11 apresentam o layout da página desenvolvida.

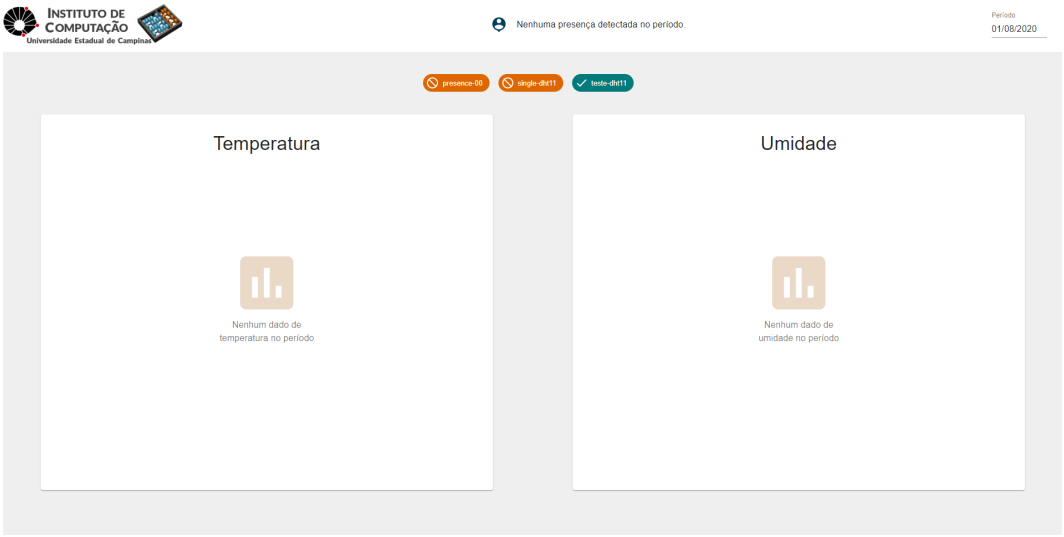

Figura 10: Website sem dados medidos e apenas um dispositivo ativo, em verde

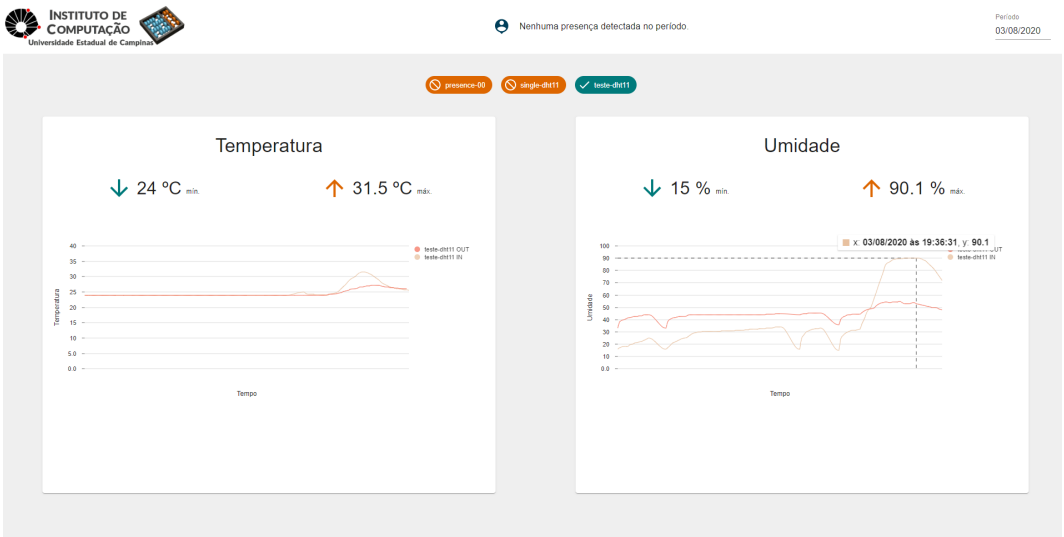

Figura 11: Website com dados no gráfico, máxima e mínima do dia e foco na medida mais alta

4.5.2 Atividades de Lucas Chinaglia Tonon

O aluno ficou responsável pela comunicação entre o front-end com a API REST da Konker, bem como a implementação de um sistema responsivo para o website que permitisse adaptabilidade dos componentes para diferentes configurações de largura e altura.

A comunicação entre o que era mostrado e o back-end que armazena as informações captadas pelos sensores foi feita usando o módulo *@axios* [19], que permite realizar requisições HTTP de forma simples utilizando Promises JavaScript. As rotas usadas no projeto e sua utilidade dentro deste foram:

1. **GET /{application}/devices:** Responsável por retornar a lista de dispositivos de hardware definidos por uma conta dentro da Konker. Isto foi utilizado para que se obtenha os identificadores (ids) de dispositivos dentro da plataforma - chamados nesta de *deviceGuid* - para serem usados na requisição de *health status* abaixo.

2. **GET /{application}/devices/deviceGuid/health:** Responsável por trazer informações quanto à "saúde" de um dispositivo, isto é, se está ativo ou não, o que é útil para exibir informações sobre algum microcontrolador que não esteja mandando informações de seus sensores e detectar problemas com o funcionamento destes de forma simples.

3. **GET /{application}/incomingEvents:** Esta é a mais importante das rotas dentro do sistema por permitir o acesso às informações medidas pelos sensores dentro da sala monitorada. A informação mais relevante sobre este endpoint está na necessidade de query strings - parâmetros enviados na requisição para filtrar o resultado conforme o que for desejado - para obtenção dos dados de somente um período pré definido e organizados em ordem de recebimento da mensagem, de forma que as mensagens mais novas estejam no topo da lista de resultados e a quantidade destas seja limitada pelo período de tempo definido em um formulário no front-end.

Como dito anteriormente, para poder acessar os endpoints declarados pela Konker é necessário adicionar um cabeçalho de autenticação específico na requisição de forma que toda requisição direcionada para lá tenha um Authorization do tipo bearer com o token obtido dentro da plataforma, o que pode ser feito de forma simples com o módulo escolhido, visto que fica tudo configurado na instanciação do objeto a ser usado para realizar as operações HTTP. Ainda que isto possa trazer dúvidas quanto a segurança do sistema implementado, pois por questão de praticidade o token usado está fixo no código do sistema, após discussão com os funcionários do Data Center notou-se que não era necessária uma segurança muito rígida dentro da aplicação desenvolvida - como autenticação e autorização robustas - pelo fato de que no processo de deploy do projeto este será bloqueado para um IP específico dentro dos servidores do Instituto de Computação.

Quanto à questão do design responsivo do web site, isto foi feito no fim do ciclo de desenvolvimento após notar-se que, ao mudar o tamanho da janela do browser, o site respondia de maneira inadequada escondendo informações e quebrando os estilos. Para corrigir este comportamento foram utilizados padrões de CSS de forma que grande parte dos estilos dos

componentes foi definido usando porcentagens e tamanhos relativos ao viewport da página. Também foram adicionadas condições aos estilos por meio do atributo @media nos CSS de forma que dependendo da largura da página um estilo específico seja utilizado. Os testes para estas alterações foram realizados utilizando o próprio browser, mudando o tamanho da janela, e também usando a funcionalidade que mostra o conteúdo da forma como seria mostrado em um celular específico, como exemplificado na Figura 12.

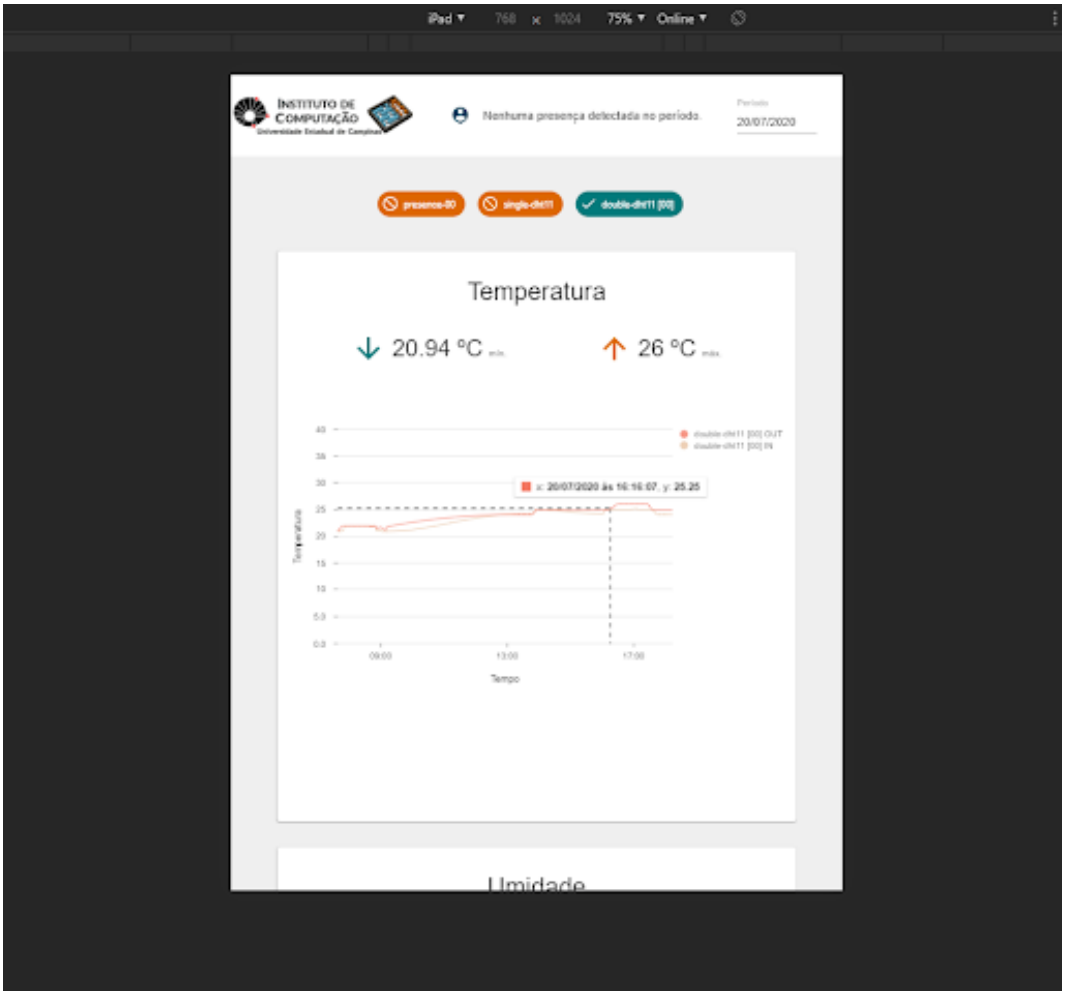

Figura 12: Site exibido através da resolução em um iPad usando função do browser

5 Resultados e Discussão

Seguindo o que foi exposto na Seção 4, obteve-se como produto final um sistema que pode ser melhor explicado em duas partes. Primeiramente, levando em conta as necessidades dos mantenedores de arescondicionados do Instituto de Computação da Unicamp, foi desenvolvida uma arquitetura de alarmes responsável por notificar os funcionários via e-mail

no caso de reconhecimento de anomalia detectada, no Data Center, por microcontroladores associados a sensores de umidade, temperatura e presença. Em segundo lugar, de forma a complementar e auxiliar o serviço primo, foi criado um website que, fazendo uso dos dados coletados pelos sensores e armazenados na Konker - utilizando a plataforma como banco de dados e back-end -, exibe cards com informações gráficas que podem ser acompanhadas em tempo real.

Devido à condições adversas impostas pela pandemia, o desenvolvimento foi realizado em sua totalidade em local diferente ao que será realmente monitorado. Tendo isso em mente, foram tomados os devidos cuidados para que a implementação do trabalho fosse construída de forma a ser flexível, no sentido de ser facilmente ajustada a mudança de ambientes. Para tal, foram utilizadas variáveis de ambiente dentro dos microcontroladores, pois podem ser rapidamente alteradas, garantindo assim o comportamento desejado.

Os testes práticos do projeto - realizados através de simulações de diferentes condições de ambiente dentro da própria residência dos alunos - foram executados da seguinte forma: utilizando uma vasta gama de medições, obtidas inicialmente com os aparelhos em repouso - isto é, fazendo com que os sensores medissem temperatura e umidade sem interferências ativas - e, posteriormente simulando comportamentos adversos - por meio do vapor decorrente de água fervente, ventiladores, luz solar, gelo, entre outros - foi observado se os disparos de alarmes ocorriam nos momentos esperados e também analisado se o comportamento refletido nos cards gráficos apresentava-se coerente ao decorrer do tempo.

Com relação às limitações do trabalho, é preciso citar o fato de que caso faça-se necessária alteração dos valores limite para definição de disparo dos alarmes, os microcontroladores precisam ter seu uso suspendido pelo intervalo de tempo em que receberão um novo código fonte com as variáveis de ambiente ajustadas para os novos valores desejados de limite superior e inferior de acionamento de notificação, não sendo possível fazê-lo dinâmica e remotamente.

Pensando em futuros desenvolvimentos, pode-se visar a implementação de um sistema de arquivos, dentro dos microcontroladores, associado a um receptor de mensagens que permita a alteração dos valores que controlam o envio de alarmes pelo simples acesso do website, inserindo através dele os novos valores e submetendo uma requisição para que os NodeMCU atualizem a informação armazenada dentro de sua memória e permitam sua customização dinâmica.

6 Conclusões

Este trabalho final de graduação teve como propósito central desenvolver um sistema capaz de contribuir com o monitoramento do Data Center do Instituto de Computação através do acompanhamento das medições de temperatura e umidade da sala, detectando falhas de funcionamento nos ares condicionados rapidamente, esperando assim evitar situações que causem prejuízos à universidade. O objetivo foi cumprido por meio da implementação de uma página web que resume os dados obtidos de forma limpa e clara e de um conjunto de sensores conectados a um servidor responsável por armazenar suas informações e também capazes de enviar e-mail para notificar situações de risco.

7 Agradecimentos

Aos orientadores, Rafael Ferrari e Juliana Freitag Borin, agradecemos a disponibilidade e acompanhamento no trabalho, sempre nos auxiliando de forma solícita contribuindo para a conclusão deste projeto.

Aos funcionários do Data Center, William Lima Reiznautt e Wellington de Oliveira Henrique, agradecemos a contribuição para definição de requisitos do projeto e pelas sugestões de implementação no decorrer de seu desenvolvimento.

Referências

[1] NEXT DATA CENTER. *Why Data Centers Need Air Conditioners and How to Choose It*. Disponível em: <https://www.nexdatacenter.com/why-data-centers-need-air-conditioners-and-how-to-choose-it/>. Acesso em: 20, Agosto de 2020.

[2] M. A. Bel. *Use Best Practices to Design Data Center Facilities*, 2005. Disponível em: <https://www.it.northwestern.edu/bin/docs/DesignBestPractices_127434.pdf/>. Acesso em: 22, Agosto de 2020.

[3] M. S. Breuker, J. E. Braun, *Evaluating the Performance of a Fault Detection and Diagnostic System for Vapor Compression Equipment, HVAC&R Research*, 4:4, 401-425, 1998.

[4] H. Li, J. E. Braun, *An Improved Method for Fault Detection and Diagnosis Applied to Packaged Air Conditioners*. ASHRAE Transactions. 109. 683-692, 2003.

[5] KONKER. *Konker - Construa a sua solução de IoT em dias e não em meses*, c2020. Página Inicial. Disponível em: <http://www.konkerlabs.com/>. Acesso em: 26, Abril de 2020.

[6] UNICAMP. *Smart Campus - Unicamp, Internet das Coisas*, 2020. Página inicial. Disponível em: <https://smartcampus.prefeitura.unicamp.br/>. Acesso em: 18, Fevereiro de 2020.

[7] AOSONG. *Temperature and Humidity Module, DHT11 Product Manual*. Datasheet para o dispositivo eletrônico DHT11. Disponível em: <https://components101.com/sites/default/files/components_datasheet/DHT11-Temperature-Sensor.pdf>. Acesso em: 20, Março de 2020.

[8] COMPONENTS101. *HC-SR501 PIR MOTION DETECTOR*, 2017. Datasheet para o dispositivo eletrônico PIR HC-SR501. Disponível em: <https://components101.com/sites/default/files/component_datasheet/HC%20SR501%20PIR%20Sensor%20Datasheet.pdf>. Acesso em: 20, Março de 2020.

[9] KONKER. *Konker Platform API*, c2020. Página com informações da API. Disponível em: <https://api.demo.konkerlabs.net/v1/swagger-ui.html>. Acesso em: 30, Abril de 2020.

[10] ESPRESSIF. *ESP8266EX Datasheet*, c2015. Datasheet para o dispositivo eletrônico ESP8266. Disponível em: <https://components101.com/sites/default/files/component_datasheet/ESP8266-NodeMCU-Datasheet.pdf>. Acesso em: 20, Março de 2020.

[11] YOUTUBE. *Como conectar um dispositivo IoT com a Plataforma Konker*, 2020. Disponível em: <https://www.youtube.com/watch?v=lL6Fsi1kDSw>. Acesso em: 10, Maio de 2020.

[12] MQTT ORG. *MQTT: The Standard for IoT Messaging*, c2020. Página Inicial. Disponível em: <http://mqtt.org/>. Acesso em: 18, Maio de 2020.

[13] GITHUB: xreef. *Library to send EMail with attachments.* Repositório do código fonte para biblioteca EMailSender. Disponível em: <https://github.com/xreef/EMailSender>. Acesso em: 9, Julho de 2020.

[14] NPM. *http-proxy-middleware*. Página do módulo no repositório de pacotes npm, c2015-2020. Disponível em: <https://www.npmjs.com/package/http-proxymiddleware>. Acesso em: 10, Maio de 2020.

[15] FACEBOOK INC. *React - Uma biblioteca JavaScript para criar interfaces de usuário*, c2020. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 10, Maio de 2020.

[16] STYLED COMPONENTS. *Visual primitives for the component age*, 2020. Disponível em: <https://styled-components.com/>. Acesso em: 10, Maio de 2020.

[17] MATERIAL UI. *Material-ui: Biblioteca de componentes React para um desenvolvimento ágil e fácil. Construa seu próprio design, ou comece com Material Design.* Disponível em: <https://material-ui.com/pt/>. Acesso em: 10, Maio de 2020.

[18] NIVO. *Nivo*, c2020. Disponível em: <https://nivo.rocks/>. Acesso em: 28, Junho de 2020.

[19] NPM. *axios: Promise based HTTP client for the browser and node.js*, 2020. Disponível em: <https://www.npmjs.com/package/axios>. Acesso em: 20, Maio de 2020.