DOI: 10.48011/asba.v2i1.1618

Desenvolvimento e aplicação de software de gerenciamento energético para comparação da demanda no período de pandemia do COVID-19

Rafael R. G. de Oliveira. Iago R. M. R. Morais. Ulisses C. Paixão J. Vinicius B. Andrade. Maria E. L. Tostes. Carminda C. M. M. Carvalho.

Centro de Excelência em Eficiência Energética da Amazônia, Universidade Federal do Pará, Belém, Pará

Abstract: An energy management system has by objective monitor, control, and manage electrical variables quantities, seeking the optimize for the given system. In this paper, the electric energy management system (SISGEE), on the collection of energy data for the loads behaviour analysis at the Federal University of Pará (UFPA) is presented. Therefore, a case study on UFPA is presented, showing the COVID-19 pandemic period impact on demand and on energy costs in the current quarantine interval (2020), when compared to the same academic period in 2019.

Resumo: Um sistema de gestão energética tem como objetivo monitorar, controlar e gerir variáveis de grandezas elétricas, buscando otimizar determinado sistema. Neste artigo, é apresentado o desenvolvimento do software e a aplicação do sistema de gestão de energia elétrica (SISGEE) na coleta de dados para análise do comportamento das cargas da Universidade Federal do Pará (UFPA). Por conseguinte, é apresentado um estudo de caso na UFPA, evidenciando o impacto do período de pandemia do COVID-19 na demanda e nos custos de energia no intervalo atual de quarentena (2020), quando comparado ao mesmo período letivo do ano de 2019.

Keywords: Energy Management Software; Computational System; Electricity Costs; Electricity Demand; COVID-19.

Palavras-chaves: Software de Gestão Energética; Sistema Computacional; Custos de Energia Elétrica; Demanda de Energia Elétrica; COVID-19

1. INTRODUÇÃO

Segundo Falcão (2010), diversos sistemas e negócios modelam-se ou sofrem constante evolução de forma natural de acordo com novas tecnologias, o que os torna objetos inteligentes capazes de capturar e fornecer informações de contexto capazes de auxiliar na tomada de decisões, por meio de seu monitoramento em tempo real. O mesmo se aplica aos projetos de tratamento de dados referente ao consumo energético, que em sua maioria, ocorre pela integração de módulos e obtenção de diagnósticos dos eventos ocorridos na rede elétrica.

Dados estatísticos da ONS (2020) apontam que a demanda por energia nos últimos anos vem crescendo, e por isso seu abastecimento deve ser mensurado, com a finalidade de apresentar tais dados eficientemente e de forma confiável, tendo em vista o planejamento da rede, tanto em situação de consumo normal, como em situações adversas. Para maior confiabilidade do estudo, recomenda-se uma base de dados com no mínimo 12 meses de acompanhamento de dados registrados.

O monitoramento e análise das faturas de energia tornam-se ações importantes para a execução de gestão energética em instalações. O resultado da análise permite gerir tanto a manutenção do sistema, quanto o contrato vigente entre a concessionária de distribuição e o proprietário da unidade consumidora, podendo implicar em redução de despesas com a eletricidade.

No trabalho atual, tem-se por objetivo apresentar parte do software SISGEE, desenvolvido para monitorar e mensurar os valores de grandezas elétricas que proporcionam a gestão de energia utilizada na Universidade Federal do Pará (UFPA). Como estudo de caso, o artigo retrata a comparação do comportamento da demanda da Universidade em um período letivo normal (2019) e o período de pandemia com o COVID-19 (2020), bem como a diferença nos custos de energia no período de pandemia.

2. MODELAGEM DO SISTEMA

O SISGEE foi projetado como um sistema apresentado em 3 camadas: a web, concebida de informações processadas e atuando como interface de comunicação com o usuário; o banco de dados, responsável pelo armazenamento de dados; e o core, responsável pela comunicação entre os medidores e o banco de dados. Tal formato visa o ganho em produtividade de

escalabilidade, além da independência no desenvolvimento das demais partes do sistema e facilidade para conduzir testes e análises em diferentes esferas da gestão energética.

O fluxograma apresentado na Fig. 1 exemplifica o ciclo de funcionamento da comunicação do software do sistema de gestão de energia elétrica (SISGEE). O banco de dados requisita dicionários em linguagem Python para armazenamento dos dados advindos dos medidores, então, quando finalizado o tempo limite de cada ciclo, tais medidas passam por critérios de integralização para que os dados sejam salvos no banco de dados, ficando disponíveis para consulta através da página web.

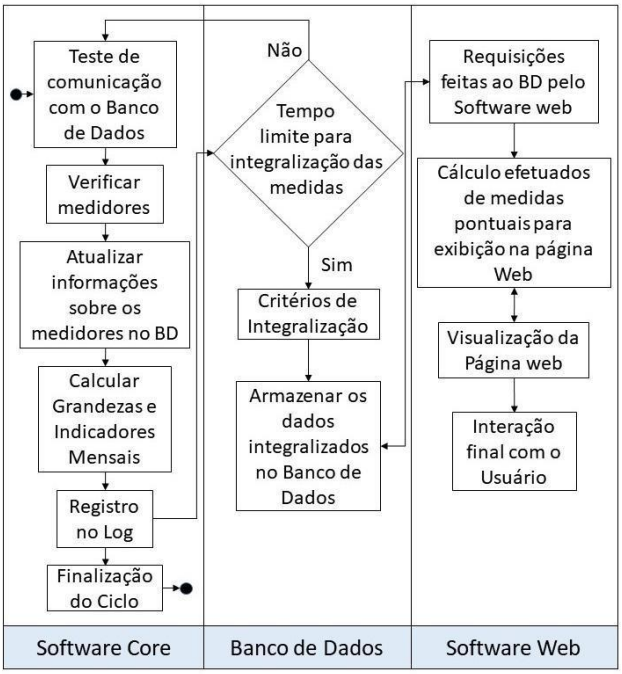

Fig. 1 Fluxograma de funcionamento do software.

Para o desenvolvimento do software foi utilizada a linguagem de programação Python e o banco de dados MongoDB para efetuar o tratamento de dados advindos dos medidores presentes na universidade. A ferramenta de banco de dados mencionada é bem difundida na esfera do descongestionamento de dados em grandes empresas, tal como Google, Amazon, Globo, etc, tendo em vista a grande capacidade de dinamizar o acesso aos dados, bem como facilitar o filtro de valores e estimativas do passado e do futuro.

A arquitetura de modelo, visualização e controle (MVC, do inglês, Model, View and Controller) adotada no sistema, é utilizada para controle, configuração, clareza e organização dos testes na plataforma em que o projeto é desenvolvido, como no caso do sistema implantado no domínio Wordpress e o sistema utilizado no framework Laravel. Além deste, os sistemas desenvolvidos e utilizados no Twitter também utilizam o modo de modelagem de sistemas inteligentes de coleta, armazenamento e visualização de dados (Bogueira, 2016).

2.1 Sistema Core

É a parte responsável pela aquisição e controle dos dados, além de efetuar a comunicação entre o software e os medidores. A comunicação para requisição dos dados dos medidores existentes ocorre por meio do protocolo Modbus (arquitetura do modelo OSI - Open System Interconnection), juntamente com a linguagem de programação Python, que é a responsável pela requisição e tratamento de informações de acordo com as determinações projetadas para o sistema.

2.1.1 Protocolo Modbus

O Modbus é um protocolo de redes que atua na sétima camada do modelo OSI, sintetizando sua independência ao habilitar a comunicação entre dispositivos heterogêneos, nos mais diversos campos. Sua função é influenciar ativamente o software e hardware dos dispositivos envolvidos na comunicação, a partir do momento que define estruturas genéricas e práticas para a construção das metodologias de troca de informações entre esses equipamentos (Fig. 2).

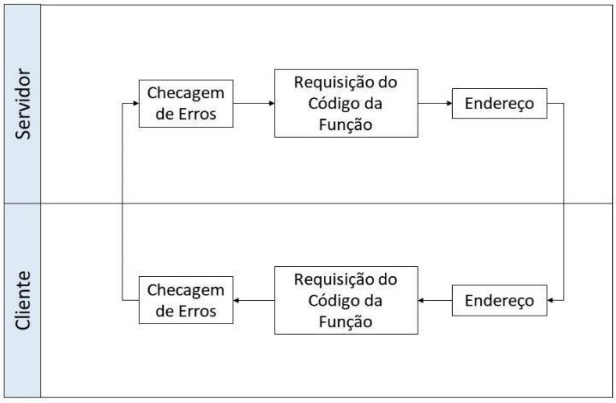

Fig. 2 Definição dos ciclos de consulta do Modbus.

Para comunicação entre os medidores presentes na universidade – medidores modelo TN4000 (Embrasul, 2015), sendo esse modelo diferente do modelo da concessionária, adquirido pela universidade – e o software há a implementação da arquitetura cliente/servidor, sendo estabelecidos 4 estágios para a realização da comunicação. Assim, o dispositivo designado a solicitar as informações e dados é chamado de cliente, e o dispositivo a transmitir a resposta para tal solicitação será o servidor.

Com a comunicação cliente e servidor, ocorre o processo de troca de dados entre o medidor e o software, por meio da rede interna da universidade. Tal troca é composta, primeiramente, da requisição do cliente e envio da mensagem para o servidor com sua solicitação, contendo os códigos das funções requisitados e as demais informações; e após isso, o servidor executa o comando recebido por meio da requisição do cliente; na terceira etapa tem-se o envio de resposta à solicitação do cliente, junto aos códigos das funções; e por fim, o cliente recebe a resposta.

2.1.2 Linguagem Python

No sistema Core, a linguagem de programação Python foi definida como padrão e escolhida devido o sistema obter códigos abertos, feitos para atender inúmeras demandas contempladas por recursos, próprios da linguagem Python (Python Documentation, 2000). Tal linguagem, além de ser uma linguagem implementada em alto nível, é aplicada ordinariamente em sistemas feitos em arquitetura de multiplataforma (Carvalho, 2014), sendo passível de testes nas mais variadas plataformas e nos mais diversos sistemas operacionais.

Conceitos específicos são aplicados a linguagem Python como um todo. Em sua documentação como linguagem de programação, consta a orientação a objetos, que remete ao tratamento de vários dados por referências aos mesmos. Além de sua estrutura, a linguagem fornece a tendência e facilidade na criação de scripts simples e com alto desempenho computacional. Devido ao sistema Core do SISGEE possuir seu funcionamento na forma de *script*, tem-se que as atuais, bem como as futuras, são resultados de testes automatizados e rotinas recorrentes de eventos do próprio sistema.

2.1.3 Aquisição de dados

O sistema core do SISGEE é composto também de diversas bibliotecas pertencentes à linguagem, cuja função é lapidar o arranjo do sistema, tornando-o mais direto e objetivo. Um exemplo é citar a função *datetime*(), que tem a capacidade de computar o horário de início e final em que as medidas são aferidas nas edificações ativas no sistema, além de registrar o momento quando ações não planejadas acontecem, sendo os erros registrados em arquivo de texto, via função pertencente à classe *logging()*, e demais eventos sendo registrados em suas respectivas categorias (Fig. 3).

```
# Comunicacao com o banco de dados
database = Database()
try:  # Tenta buscar informacoes do servidor
    database.client.server_info()  # Faz a requisicao das informacoes
                                   # ao servidor
    if not NOME_DATABASE in database.client.database_names():
        database.createIndex()  # Cria os indices e chaves unicas para
                                 # as colecoes, caso elas nao existam
except ServerSelectionTimeoutError:  # Caso o banco nao responda
                                     # indica erro no log e fecha o
                                     # sistema
    print("Erro no acesso ao Banco de Dados!")
    log.ErroBD(tempo=Data().time.strftime("%Y-%m-%d %H:%M:%S"))
    print("Salvo no Arquivo de Log!")
    print("")
    print("Finalizando Sistema...")
    raise SystemExit
```

Fig. 3 Trecho de código com função mostrando a comunicação entre o servidor e o banco de dados, fazendo uso das funções citadas anteriormente.

Demais funções da linguagem existem para manter a persistência do software na coleta de medidas das edificações presentes na universidade, como citado anteriormente. A classe que faz uso da função *logging*(), projeta métodos indicando cada erro e evento que possa ocorrer no sistema, como salvamento das medidas adquiridas via medidores, erro de comunicação com o banco de dados, falhas de comunicação entre medidores e o software, registro de novos medidores na rede, erros nas *strings* de dados enviadas pelos medidores para o tratamento no software, entre outros.

Como o sistema está organizado a funcionar de forma direta, mas cíclica, a comunicação entre o software e os medidores instalados nas edificações ao longo da universidade é o primeiro teste realizado pelo software. Em seguida, é coletada a lista dos medidores ativos, por meio de teste de conexão. Então, dá-se início as medições cíclicas, tendo em vista as grandezas e indicadores mensais sendo calculados pontualmente (Fig. 4).

```
# Testes referentes a comunicacao entre o banco de dados e os medidores
erro_comunicacao = False
for medidor in medidores:...

if erro_comunicacao:
    continue
# Medidores encontrados sendo adicionados a 'medidores' apos teste
medidores = list(filter(lambda medidor: medidor is not None, medidores))

# Dicionario recebendo lista de requisicao de medidas
medidas_dict = {}

while True:
    # print("Inicializando Medidas")
    actual_time = Data().time
    #actual_time recebe o tempo atual do sistema
    InicializarMedidas.Grandezas(medidores, medidas_dict, medidores_defeito)
    # inicio_medidas_ciclicas = Data().time
    print("Iniciando Medidas Cíclicas...")
    print("")
    # Inicia as medicoes ciclicas em sequencia
```

Fig. 4 Trecho de código mostrando fim dos testes de comunicação e início das medições cíclicas.

2.1.4 Tratamento de dados

A linguagem Python é utilizada para implementar o acesso do software ao medidor, por meio do protocolo Modbus, requisitando as amostras aferidas em tempo real pelo protocolo. Conforme apresentado na Fig. 5, após a obtenção dos dados, ocorrem cálculos, seguidos da integralização das medidas pela média e a conversão das séries de amostras em tempo real integralizadas, em um período de 10 minutos. Anteriormente, o Critério de Chauvenet era utilizado como método de correção para medidas discrepantes em relação à média, removendo tais medidas (Muniz, 2019). Tal método foi utilizado em momento anterior à correção dos erros em valores advindos de alguns medidores.

O intervalo de 10 minutos entre a integralização das amostras visa atender ao módulo 8 dos Procedimentos de Distribuição (PRODIST) da Agência Nacional de Energia Elétrica (ANEEL) que exige a obtenção de no mínimo 1008 medições no intervalo de uma semana, para analisar a qualidade de energia elétrica.

```
if actual_time.minute >=0 and actual_time.minute <= 9:
    auxiliarminutos = auxiliarminutos.replace(minute=9)

elif actual_time.minute >= 10 and actual_time.minute <= 19:
    auxiliarminutos = auxiliarminutos.replace(minute=19)

elif actual_time.minute >=20 and actual_time.minute <= 29:
    auxiliarminutos = auxiliarminutos.replace(minute=29)

elif actual_time.minute >=30 and actual_time.minute <= 39:
    auxiliarminutos = auxiliarminutos.replace(minute=39)

elif actual_time.minute >=40 and actual_time.minute <= 49:
    auxiliarminutos = auxiliarminutos.replace(minute=49)

elif actual_time.minute >=50 and actual_time.minute <= 59:
    auxiliarminutos = auxiliarminutos.replace(minute=59)

while actual_time.minute <= auxiliarminutos.minute: # Laco para cada 10 minutos
    # Recebimento do horario atual para o teste de verificacao dos 10 minutos
    n_medida += 1
    print("Rodada de Medição" + " " + str(n_medida))
```

Fig. 5 Trecho de código demonstrando os testes realizados para continuidade na medição pontual.

A verificação Cíclica de Redundância (CRC), tem o objetivo de identificar mudanças na cadeia de dados durante a transmissão da informação obtida pelo sistema (Warren, 2012). No software SISGEE, a verificação dos dados trazidos pelo medidor ao software, ocorre ao final de cada ciclo de medições e, de acordo com as condições de recebimento, os dados são integralizados. Além deste, também ocorre a verificação periódica para determinar *timeout* do sistema ou em medidores isolados tendo em vista falhas de comunicação em medidores, que pode ocorrer em razão das quedas de energia na universidade (Fig. 6).

```
log = Log()

def ExceptCRCUrms(medidas_dict, medidor):
    """
    Funcao para tratamento da excecao
    de CRC da grandeza Urms.
    :param medidas_dict: dicionario que armazena os valores das colecoes
    :param medidor: variavel local que explora os valores contidos nas medicoes.
    """
    medidas_dict[medidor.nome]["urms"]["valor"]. \
        append({"Fase A": 0.0, "Fase B": 0.0, "Fase C": 0.0})
    medidas_dict[medidor.nome]["urms"]["erros"]. \
        append(CRC_ERROR)
    # print("Tensão RMS: Erro de CRC!")
    log.MedidaErroCRC(id=str(medidor.id),
                      nome=medidor.nome,
                      ip=medidor.ip,
                      medida="urms",
                      tempo=Data().time.strftime
                      ("%Y-%m-%d %H:%M:%S"))
    # print("Salvo no Arquivo de Log!")

    return medidas_dict[medidor.nome]["urms"]["valor"], \
           medidas_dict[medidor.nome]["urms"]["erros"]
```

Fig. 6 Trecho de código exemplificando método de checagem do CRC na grandeza tensão (Urms) durante a execução do ciclo de medições no software.

Além do Critério de Chauvenet, utilizado na integralização dos dados, a cada hora completa são realizados testes cujo objetivo é verificar a permanência das informações de energia aferidas pelos medidores no banco de dados (Fig. 7), visando mensurar dados e informações inerentes a cada medidor, como data, hora e o número de série para o mesmo, garantindo a leitura dos dados de forma correta.

```
n_integralizacao += 1

actual_time = Data().time   # Recebimento do horario atual
                            # para o teste de verificacao horária
                            # E igual ao tempo de duracao da medicao na hora atual

if actual_time.minute < 1 or n_integralizacao >= 6:
    print("O sistema irá reiniciar automaticamente...")
    # print("")
    break
        # Se o horario atual tiver passado de 1h,
        # ou as integralizacoes tenham atingido 6
        # e o tempo atual seja inferior a 1 minuto,
        # interrompe as medidas ciclicas para
        # poder integralizar e salvar no banco de dados
```

Fig. 7 Trecho de código exemplificando a verificação periódica após a integralização das medidas.

2.2 Software web

Por via da revolução tecnológica, muitos itens utilizados no dia a dia foram adicionados a uma imensa rede de computadores, com aparelhos versáteis e conectados a outros (Lóscio, 2016). O software web do SISGEE, atua como interface de comunicação com o usuário final. Sendo a camada mais próxima do usuário, o software exibe as informações obtidas e processadas pelo banco de dados e sistema core, respectivamente.

A textura adotada no design do sistema web tem como objetivo espelhar as características do mundo físico, tornando o usuário consentâneo ao cotidiano, sobretudo aos efeitos incluídos no sistema, principalmente os de contraste e de sombreamento. A tipografia adotada tem por funcionalidade trazer a experiência de imersão ao usuário. Os efeitos de transição presentes são compostos de continuidade e efeitos de coerência são inerentes para as dinâmicas encontradas no mundo físico.

Para o software SISGEE, as telas presentes no sistema web possuem capacidade adaptativa aos mais diversos dispositivos, tendo em vista os conteúdos presentes nas telas de medidores, gráficos comparativos e quadro de tarifas, ajustáveis ao tipo de navegação realizada, tanto na navegação Desktop como na Mobile (Fig. 8 e 9). Assim, o software adapta o tamanho das fontes e a quantidade de espaço de tela ocupado por determinadas funcionalidades (Carvalho, 2019).

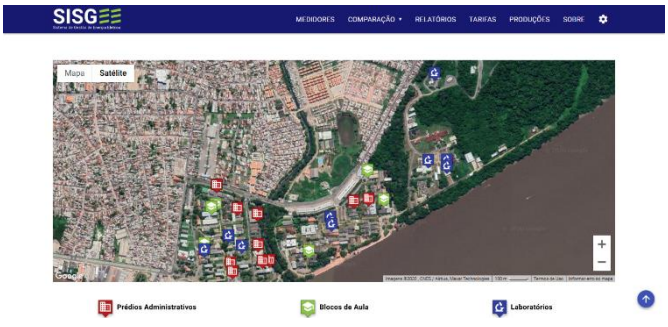

Fig. 8 Página web inicial referente ao software SISGEE.

Fig. 9 Página web inicial referente a versão mobile do SISGEE.

O acesso para a visualização dos dados aferidos, se faz por meio do software na parte web e a divisão das ferramentas (abas) estão explicadas a seguir.

● HOME – tela inicial para o site, com a API do Google oferecendo a representação da localização física para cada medidor, além da categoria de sua UC;

● MEDIDORES – mostra o status de cada medidor, dividido por área física na Universidade, além de mostrar o id, suas principais grandezas, período de medição e a localização dele no campus;

● COMPARAÇÃO – comparação de medidas como consumo, entre os medidores, de acordo com o período definido pelo usuário;

● RELATÓRIOS – área para escolha de medidores com o objetivo de gerar relatórios em planilhas, seja este de falhas, comparações ou análises;

● TARIFAS – comparação tarifária entre os medidores;

● SOBRE – resumo do projeto SISGEE e exemplificação de fórmulas para cálculo das grandezas.

Dentre estas, a aba MEDIDORES (Fig. 10) apresenta maior interação com o usuário, por proporcionar a escolha das grandezas que se deseja analisar, facilitando a visualização das medidas por meio de gráficos.

Fig. 10 Página web representando a aba de MEDIDORES presentes no software.

2.3 Módulo Database

Os bancos de dados, são relevantes como suprimento para demandas de organização física e lógica de dados, sendo utilizado no trabalho atual como ferramenta de extrema importância para garantir a permanência dos dados, bem como ajudar nas pesquisas e cálculos de tarifas energéticas.

A Fig. 11 apresenta o módulo *Database* no SISGEE, responsável pelo armazenamento e manutenção de dados e informações presentes no sistema, além de servir para eventuais pesquisas e acesso dos dados, tanto para testes como para atualização de informações oriundas de cálculos.

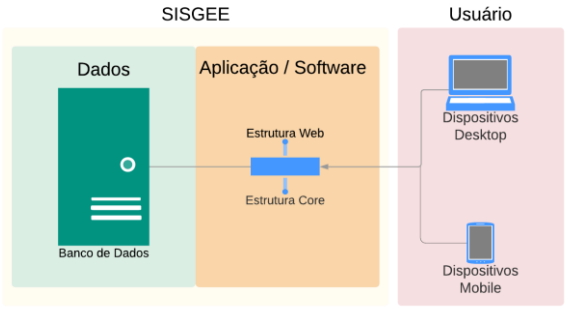

Fig. 11 Banco de dados no projeto estrutural do SISGEE.

3. ESTUDO DE CASO UTILIZANDO O SISGEE

A partir de 19 de março de 2020 (UFPA, 2020), a UFPA decretou a suspensão das atividades acadêmicas e administrativas de seus campi, em atendimento a lei nº 13.979, de fevereiro de 2020, aprovada pela Câmara dos deputados (2020), que instaurou a possibilidade de isolamento social e quarentena por meio das autoridades, devido a doença Coronavirus-2019 (COVID-19). As medidas de isolamento social causaram a redução no consumo de energia elétrica em diferentes classes consumo. O estudo de caso deste artigo, será efetuada com base nas informações do campus UFPA Guamá.

Segundo a Resolução Aneel n. 4, os grupos tarifários em vigor são grupo A e grupo B. A Universidade Federal do Pará enquadra-se no subgrupo A4 com demanda contratada de 5.900kW na tarifa verde (Equatorial Energia, 2020), por ser atendida em média tensão (13,8 kV), sujeita a tarifa binômia.

Os dados da demanda (potência ativa) presentes no estudo foram extraídos no ponto de acoplamento comum entre a UFPA e a concessionária local, por meio do software SISGEE. Visando efetuar a comparação entre os períodos com e sem os impactos do COVID-19, utilizou-se uma semana do ano letivo de 2019, de 30/03/2019 a 05/04/2019, conforme apresentado na Fig. 12; e uma semana do ano letivo de 2020, 28/03/2019 a 03/04/2020, conforme apresentado na Fig. 13. A diferença de dias foi necessária para considerar o mesmo intervalo de dias da semana, considerando de sábado a sexta feira.

Ambos os gráficos apresentados nas Fig. 12 e 13 possuem as potências ativas por fase e a potência total. Sendo assim, para comparação entre os períodos utilizou-se apenas as potências ativas totais (demanda), conforme apresentado na Fig. 14, geradas por meio do Excel.

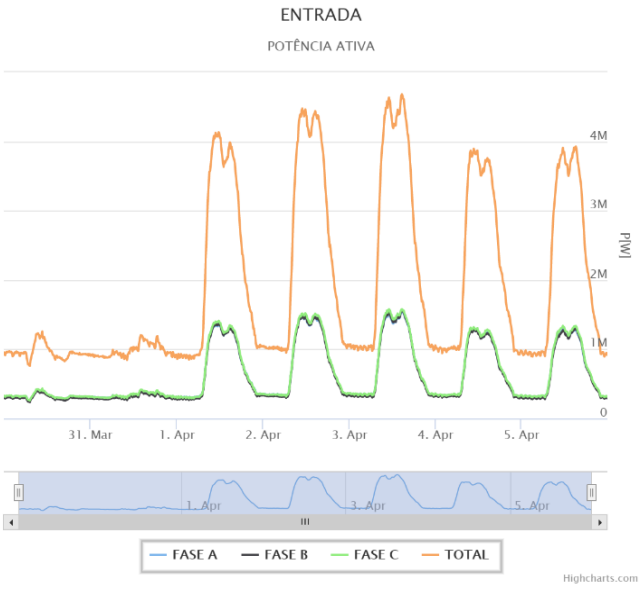

Fig. 12 Potência ativa da UFPA no período de 30/03/2019 a 05/04/2019.

Fig. 13 Potência ativa da UFPA no período de 28/03/2020 a 03/04/2020.

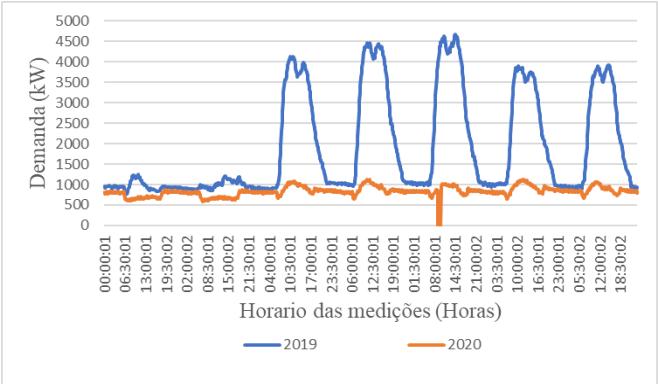

Fig. 14 Comparação entre as demandas de 2019 e 2020.

Analisando a Fig. 14, tem-se que o perfil de demanda de energia elétrica semanal de 2020, período de pandemia, é semelhante a demanda da universidade durante os finais de semana para o ano de 2019, período letivo normal.

Em 2020, os valores de demanda raramente ultrapassaram os valores de 1.000 kW, atingindo o pico máximo de 1.129,6 kW para o ano de 2020. A curva de demanda no ano de 2019 teve seu comportamento previsível, de modo que atinge seu pico por volta das 12:00 horas, com uma demanda máxima de 4.680,24 kW.

Considerando que a demanda contratada atual (5.900 kW) e a demanda medida no período de 2019 são maiores do que a demanda medida em 2020, foi utilizada a Eq. 1 para calcular o valor de demanda adequada para o período de pandemia. Sendo assim, tem-se que a demanda a ser contratada neste período deve ser de 1.076 kW, aproximadamente.

$$DC = \frac{MDR}{1.05} \qquad \text{(Eq. 1)}$$

Onde:

DC é a demanda contratada;

MDR é a máxima demanda registrada.

O valor do kW atual no estado do Pará é de R$26,36/kW (Equatorial Energia, 2020), levando em consideração que não ocorra transgressões nos valores de demanda contratada. Então, para precificar os valores financeiros entre a demanda contratada em 2020 e a demanda no período do COVID-19, utilizou-se a Eq. 2:

$$VD = DPA * TD \quad \text{(Eq. 2)}$$

Onde:

VD é o Valor da demanda;

DPA é a demanda de potência ativa;

TD é a tarifa da demanda.

As Fig. 15 e 16 apresentam os custos financeiros calculados para as demandas contratada e proposta para o período do COVID-19, com e sem imposto, respetivamente.

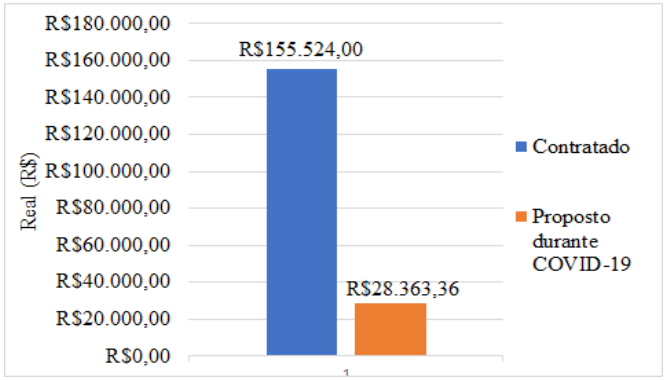

Fig. 15 Valor em R$ das demandas contratada e medidas no período de pandemia, sem imposto.

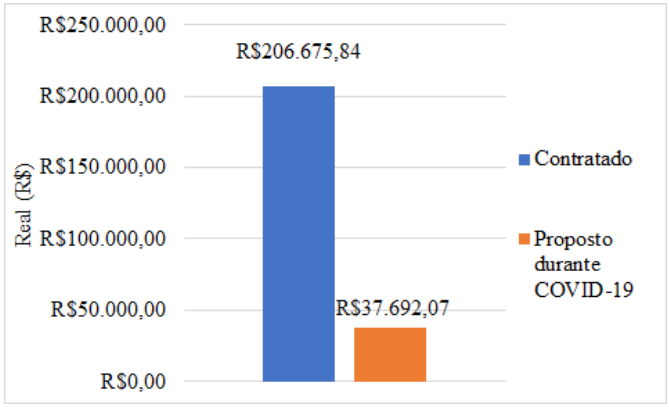

Fig. 16 Valor em R$ das demandas contratada e medidas no período de pandemia, com imposto.

Considerando os dados apresentados nas Fig. 15 e 16, a Fig. 17 apresenta a diferença mensal no custo da demanda representa R$127.160,64, valores sem impostos, e R$168.983,77, valores com impostos, dado o fato da demanda contratada estar ociosa.

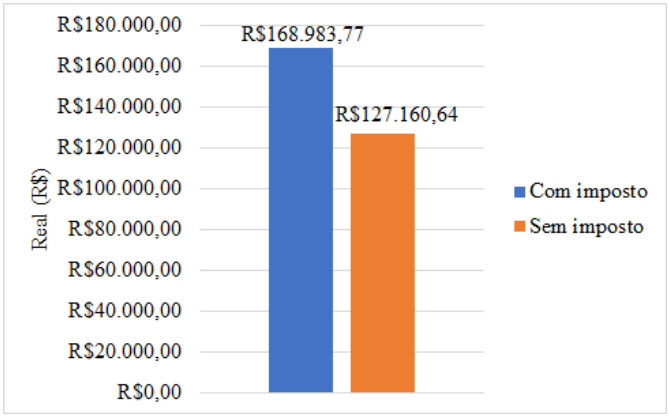

Fig. 17 Diferença entre as demandas contratada e medidas no período de pandemia, com e sem imposto.

Apesar do impacto da redução da demanda de energia devido ausência de atividades no campus ser significativa, o mesmo não representou diferenças financeiras para os cofres da Universidade, de modo que durante situações de calamidade pública na qual ocorra alterações na demanda de energia elétrica, é necessário definir medidas para mitigar o ônus a grandes consumidores. Ressalta-se que para um estudo completo são considerados dados do consumo de energia em ambos períodos, pois estes também influenciarão na diferença do custo de energia da instituição.

4. CONCLUSÃO

O sistema de gestão de energia elétrica é resultado de um conjunto de ferramentas que utiliza a linguagem Python para efetuar a comunicação entre o banco de dados MongoDB e a parte web, integrando os dados extraídos de medidores instalados em edificações da universidade às necessidades dos usuários.

Os códigos de programação apresentados na modelagem do SISGEE são compostos de medições cíclicas, para proporcionar segurança no armazenamento de dados e maior confiabilidade ao sistema, além de apresentar dinamicidade do site e clareza na amostragem dos dados obtidos.

Dentre as possíveis grandezas monitoradas pelo SISGEE, a geração de um banco de dados e a utilização da potência ativa permitiram efetuar um comparativo entre diferentes períodos e possibilitar a tomada de ação de gestores dentro da Universidade, junto a concessionária local. O estudo de caso abordado nesse artigo tratou de uma situação extraordinária, de pandemia, que custará aos cofres públicos em torno de R$160.000,00 por mês, devido a subutilização da demanda contratada.

AGRADECIMENTOS

Os Autores agradecem o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001; e do CNPq para a realização do presente trabalho, no âmbito do projeto “Sistema inteligente para determinação dos impactos harmônicos de múltiplos consumidores nas redes de distribuição de energia

elétrica", aprovado na Chamada Universal - MCTI/CNPq/CT-ENERG Nº 14/2016.

REFERÊNCIAS BIBLIOGRÁFICAS

ANEEL. Resolução Normativa nº 794/2017. Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional, v. 8, p. 88, 2018. Disponível em: <http://www.aneel.gov.br/documents/656827/14866914/M{ó}dulo{_}8-Revis{~{a}}o{_}10/2f7cb862-e9d7-3295>. Acessado em 15 de maio de 2020.

Bogueira, G., Batista, F., Carvalho, J. P. (2015). Sistema Inteligente de Recolha, Armazenamento e Visualização de Informação proveniente do Twitter. Acessado em 30 de abril de 2020.

Câmara dos deputados. Lei Nº 13,979, de 6 de fevereiro de 2020. *Centro de documentação e informação.* Disponível em <https://www2.camara.leg.br/legin/fed/lei/2020/lei-13979-6-fevereiro-2020-789744-norma-pl.html>. Acessado em 18 de maio de 2020.

Carvalho, H, M., Aprendizado de Máquina voltado para Mineração de Dados: Árvores de Decisão. Universidade de Brasília, Brasília, DF. 2014. Acessado em 30 de abril de 2020.

Carvalho, I. S., Júnior, U.C.P., Monteiro, F., Tostes, M.L., Muniz, J.R.S. e Souza, A.C.D.B. (2019). Modelagem e validação de um sistema multiplataforma para Gerenciamento de Energia Elétrica – SISGEE. Anais do 14º Simpósio Brasileiro de Automação Inteligente. Acessado em 23 de abril de 2020.

Equatorial energia. (2020). Valor de tarifas e serviços. Disponível em < https://pa.equatorialenergia.com.br/informacoes-gerais/valor-de-tarifas-e-servicos/#tarifas-grupo-a> acessado em 18 de maio de 2020.

Embrasul. Multimedidor de Grandezas Elétricas TN4000. Disponível em: < http://www.embrasul.com.br/solucoes?catg=6&title=Multimedidores>. Accesso em 30 de abril de 2020.

Falcão, D. M. "Integração de tecnologias para viabilização da Smart Grid". III Simpósio Brasileiro de Sistemas Elétricos, pp. 1–5 (2010). Acessado em 30 de abril de 2020.

Lóscio, B. F., Oliveira, H. R., Pontes, J. C. S. (2016). NoSQL no desenvolvimento de aplicações Web colaborativas. Acessado em 18 de maio de 2020.

Muniz, J. R. S. (2019), PROJETO COMPUTACIONAL DO SISTEMA DE GERENCIAMENTO DE ENERGIA ELÉTRICA SISGEE NA CIDADE UNIVERSITÁRIA JOSÉ DA SILVEIRA NETTO, Universidade Federal do Pará. Acessado em 23 de abril de 2020.

ONS. Histórico de operação. Disponível em < http://www.ons.org.br/Paginas/resultados-da-operacao/historico-da-operacao/carga_energia.aspx>. Acessado em 30 de abril de 2020.

Prodist. QUALIDADE DE ENERGIA, Módulo 8. Acessado em 15 de maio de 2020.

Python Documentation (2000). Disponível em <https://docs.python. org/pt-br/>. Acessado em: 18 de maio de 2020.

Universidade Federal do Pará. (2020). Disponivel em <https://portal.ufpa.br/index.php/ultimas-noticias2/11452-ufpa-emite-nota-sobre-suspensao-de-atividdes-academicas-e-administrativas-presenciais> Acessado em 18 de maio de 2020.

Valor de tarifas e serviços prestados, de acordo com a demanda contratada. Disponível no site da Equatorial Pará: < https://pa.equatorialenergia.com.br/ >. Acesso em: 4 de maio de 2020.

Warren, H. S. Hacker's Delight. Pearson Education, 2012. ISBN 0321842685,9780321842688. Disponível em: https://books.google.com.br/books?id=VicPJYM0I5QC. Acessado em 18 de maio de 2020.