UNIVERSIDADE FEDERAL DE GOIÁS

ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

CURSO DE ENGENHARIA DE COMPUTAÇÃO

RAFAEL RATACHESKI DE SOUSA RAULINO

DESENVOLVIMENTO DE SISTEMA PARA GERENCIAR E AUXILIAR NA TOMADA DE DECISÃO ENVOLVENDO CONSUMO DE ENERGIA ELÉTRICA

TRABALHO DE CONCLUSÃO DE CURSO

GOIÂNIA

2019

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC nº 1204/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei nº 9610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou *download*, a título de divulgação da produção científica brasileira, a partir desta data.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG):

Nome completo do autor: Rafael Ratacheski de Sousa Raulino

Título do trabalho: Desenvolvimento de Sistema para Gerenciar e Auxiliar na Tomada de Decisão Envolvendo Consumo de Energia Elétrica

2. Informações de acesso ao documento:

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

Havendo concordância com a disponibilização eletrônica, torna-se imprescindível o envio do(s) arquivo(s) em formato digital PDF do TCCG.

Assinatura do(a) autor(a)

Ciente e de acordo:

Assinatura do(a) orientador(a)

Data: 09 / 07 / 2019

RAFAEL RATACHESKI DE SOUSA RAULINO

DESENVOLVIMENTO DE SISTEMA PARA GERENCIAR E AUXILIAR NA TOMADA DE DECISÃO ENVOLVENDO CONSUMO DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal de Goiás, como requisito parcial para obtenção do grau de Engenheiro de Computação.
Orientador: Prof. Dr. Marcelo Stehling de Castro

GOIÂNIA

2019

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Raulino, Rafael Ratacheski de Sousa
Desenvolvimento de Sistema para Gerenciar e Auxiliar na Tomada de Decisão Envolvendo Consumo de Energia Elétrica [manuscrito] / Rafael Ratacheski de Sousa Raulino. - 2019.
101 f.: il.

Orientador: Prof. Dr. Marcelo Stehling de Castro.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Engenharia de Computação, Goiânia, 2019.
Anexos. Apêndice.
Inclui siglas, abreviaturas, algoritmos, lista de figuras, lista de tabelas.

1. Consumo energético. 2. Banco de dados. 3. Monitoramento de rede energética. 4. Sistema de monitoramento. 5. Desenvolvimento de sistema. I. Castro, Marcelo Stehling de, orient. II. Título.

CDU 62:004.3/.4

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO
COORDENAÇÃO DE ESTÁGIO E PROJETO FINAL

ATA DE AVALIAÇÃO DE PROJETO FINAL

CURSO

( ) Eng. Elétrica ( ) Eng. Mecânica ( ) Eng. de Computação
( ) Projeto Final 1 (X) Projeto Final II

AVALIAÇÃO DE PROJETO FINAL

Título do projeto:	Desenvolvimento de Sistema para gerenciar e auxiliar na tomada de decisão envolvendo consumo de energia elétrica

BANCA AVALIADORA

Membro 1: Marcelo Stehling de Castro

Membro 2: Gustavo Dias de Oliveira

Membro 3: Sandrerley Ramos Pires

ESTUDANTES

Matrícula	Nome
201108023	Rafael Ratacheski de Sousa Raulino

NOTAS

Matrícula	Membro 1 - Marcelo Stehling de Castro				Membro 2 - Gustavo Dias de Oliveira				Membro 3 - Sandrerley Ramos Pires				Média
	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	NPT	NTE	NAA	NF	
201108023	10	10	10	10	–	10	10	10	–	10	10	10	10

NPT – Nota plano de trabalho; NTE – Nota do trabalho escrito; NAA – Nota de apresentação e arguição

Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA

Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

Goiânia, 05 de julho de 2019.

Marcelo Stehling de Castro

Marcelo Stehling de Castro

Gustavo Dias de Oliveira

Gustavo Dias de Oliveira

Sandrerley Ramos Pires

Sandrerley Ramos Pires

FREQUÊNCIA – a ser preenchido pelo orientador(a)	
Nome do(a) estudante	Frequência (%)
Rafael Ratacheski de Sousa Raulino	100

Marcelo Stehling de Castro
Professor(a) Orientador(a)

ATA DE AVALIAÇÃO DE PROJETO FINAL - Observações

Preencher com modificações solicitadas, caso existam. Em caso de reprovação, informar a justificativa.

Ao meu pai, por sempre acreditar em mim,
por seus ensinamentos que me tornaram quem eu sou.
Sua presença significou segurança e certeza de
que não estou sozinho, e mesmo estando longe de você,
tentarei sempre honrar sua memória em minhas decisões.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado o dom do conhecimento e da criatividade, me possibilitando chegar até esse momento.

À minha família por sempre acreditar no meu sonho e investirem tempo e dedicação na realização do mesmo. À minha mãe Neusa por em vários momentos se privar de oportunidades para me dar o presente de adquirir conhecimento. Ao meu pai Paulo por mesmo não estando mais neste mundo, continuar me dando respostas inteligentes aos meus questionamentos sobre a humanidade. Aos meus irmãos, Bárbara e Renato, por me alegrarem nos momentos difíceis, das noites longe de minha família.

Agradeço também à minha fiel companheira Natalia que esteve comigo durante toda minha caminhada, me auxiliando e dando alento nos momentos difíceis, por ter surgido quando eu mais precisava e fazendo minha vida muito melhor.

Agradeço ao meu orientador Marcelo Stehling de Castro por acreditar no meu trabalho e me apoiar em meu momento de pausa no desenvolvimento deste projeto, e principalmente agradeço ao Gustavo Dias por todo ensinamento que me proporcionou em todo o meu período de estágio e pesquisa deste projeto.

"A imaginação é mais importante que o conhecimento.
Pois, o conhecimento é limitado, enquanto a imaginação
abraça o mundo inteiro, estimulando o progresso,
dando origem à evolução."
(Albert Einstein)

Resumo

Nos últimos anos com o intenso desenvolvimento de dispositivos eletrônicos para facilitar nosso dia a dia, o consumo de energia elétrica só tem aumentado e, aliado a esse aumento, cresceu também o impacto ambiental e econômico causado por esse consumo. Sendo assim a preocupação com a criação de medidas para o entendimento e o controle dos impactos é cada dia mais presente. Propõe-se com este trabalho, desenvolver um sistema, para fazer o monitoramento e auxiliar no controle da rede de energia elétrica da Universidade Federal de Goiás e dos seus ativos, disponibilizando uma interface para visualização e análise dos dados coletados pelos sistemas de monitoramento da rede energética online, que permita fazer o cadastramento em uma base do seu acervo de ativos, tais como transformadores, inversores, usinas geradoras de energia elétrica, subestações, juntamente com a possibilidade de inserção de dados históricos do consumo através da leitura dos arquivos das faturas das unidades consumidoras da Universidade, alimentando o banco de dados do sistema com os dados oficiais da concessionária de energia de faturamento dos meses passados.

Palavras-chaves: consumo energético, banco de dados, monitoramento de rede energética, sistema de monitoramento, desenvolvimento de sistema.

Abstract

In recent years with the intense development of electronic devices to facilitate our day to day, electricity consumption has only increased, and along with this increase has also grown the environmental and economic impact caused by such consumption. Therefore, the concern to create measures for the understanding and control of impacts is increasingly present. It is proposed, with this work, the development of a system to monitor and assist in the control of the energy network of the Federal University of Goiás and its assets, providing an interface for appreciation and analysis of the data collected by the energy grid monitoring online systems, which allows to make the registration of assets, such as transformers, inverters, power plants, substations, along with the possibility of insertion of historical data of the consumption by reading the files of the invoices of the consumer units of the University, feeding the system database with the official data of the billing energy utility of past months.

Key-words: energy consumption, database, energy network monitoring, monitoring system, system development.

Lista de Códigos Fonte

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	*Application Programming Interface* (Interface de Programação de Aplicativos)
ASCII	*American Standard Code for Information Interchange* (Código Padrão Americano para Intercâmbio de Informações)
CRC	*Cyclic Redundancy Check*
CRLF	*Carriage Return and Line Feed*
CSV	*Comma-separated values*
DB	*Database* (Banco de Dados)
DMCR	Demanda Máxima Corrigida
EIA-232	*Electronic Industries Alliance 232*
EIA-485	*Electronic Industries Alliance 485*
EPE	Empresa de Pesquisa Energética
FP	Fator de Potência
GWh	Gigawatt-hora
IP	*Internet Protocol*
Kbps	Kilobits por segundo
Mbps	Megabits por segundo
PDF	*Portable Document Format*
QDC	Quadro de Distribuição de Circuitos
REST	*Representational State Transfer* (Transferência de Estado Representacional)
RS-232	*Recommended Standard 232*
RS-485	*Recommended Standard 485*
RTU	*Remote Terminal Unit*
SGBD	Sistema Gerenciador de Banco de Dados

SIDE	Sistema Interpretador de Dados Energéticos
SOAP	*Simple Object Access Protocol* (Protocolo Simples de Acesso a Objetos)
TCP	*Transmission Control Protocol*
UFER	Unidade de Faturamento de Energia Reativa
UTP	*Unshielded Twisted Pair* (Par Trançado Não Blindado)
UUID	*Universally Unique Identifier* (Identificador Único Universal)
VLAN	*Virtual Local Area Network* (Rede Local Virtual)

Sumário

Parte I

Introdução

1 Introdução ao Problema

Nesta introdução serão apresentados os objetivos do trabalho, a definição de alguns conceitos que se fazem necessários para que este se situe e para que a motivação e desafios do mesmo sejam esclarecidos e, por fim, a estrutura da monografia.

1.1 Objetivos

Os objetivos principais deste trabalho são:

- desenvolver um sistema para comunicação com os medidores de energia elétrica instalados na universidade;
- desenvolver um modelo de dados para armazenamento dos dados coletados por esses medidores, bem como os dados da própria concessionária de energia;
- desenvolver um sistema para leitura, interpretação e armazenamento dos dados contidos nas faturas energéticas geradas pela concessionária;
- implantar o sistema e disponibilizá-lo para o uso da instituição;

1.2 Motivação

A motivação desse trabalho surgiu no desenvolvimento inicial de um projeto durante estágio na instituição, onde o escopo inicial do sistema de interpretação dos dados das faturas energéticas foi definido, e onde posteriormente foi inserido a parte de comunicação com os medidores e a criação de uma interface para apreciação dos dados durante o desenvolvimento do **Projeto de Final de Curso 1** do mesmo autor descrito e disponível no apêndice A.

1.3 Conceitos

A seguir, serão abordados alguns dos conceitos de alto nível utilizados ao longo do trabalho, estes conceitos possibilitarão um entendimento inicial das técnicas escolhidas para a solução do problema inicial, bem como possibilitarão uma aproximação maior do leitor ao tema.

1.3.1 Protocolo Modbus

O protocolo Modbus é uma estrutura de mensagem aberta desenvolvida pela Modicon na década de 70, utilizada para comunicação entre dispositivos mestre-escravo / cliente-servidor. (MODICON, 1996)

Após a compra da Modicon pela Schneider os direitos sobre o protocolo foram liberados pela Organização Modbus. Muitos equipamentos industriais utilizam o Modbus como protocolo de comunicação, e graças às suas características, este protocolo também tem sido utilizado em uma vasta gama de aplicações como:

- Instrumentos e equipamentos de laboratório;
- Automação residencial;
- Automação de navios;

1.3.2 Desenvolvimento Ágil

A partir de 1990 as definições desenvolvimento de software evoluíram como parte de uma reação contra métodos desgastantes de desenvolvimento, caracterizados por uma densa regulamentação.

O processo originou-se da visão de que o modelo em cascata era burocrático e lento, uma forma contrária com a qual engenheiros de software realizavam seus trabalhos. Inicialmente, foram desenvolvidas técnicas para para agilizar o desenvolvimento e a esse conjunto de técnicas foi dado o nome de métodos leves.

Em 2001, membros da comunidade se reuniram em *Snowbird* e publicaram um manifesto para reunir os princípios e práticas desta metodologia de desenvolvimento, dando o nome de Manifesto Ágil (BECK et al., 2001). Posteriormente, formou-se a *Agile Alliance*, uma organização sem fins lucrativos que ajuda na divulgação do desenvolvimento ágil.

1.3.3 Banco de Dados

Korth (1994) define um banco de dados como sendo "uma coleção de dados inter-relacionados, representando informações sobre um domínio específico", ou seja, quando houver a possibilidade de se agrupar dados que se relacionam e fazem parte de um mesmo universo de um assunto, pode-se dizer que tem-se um banco de dados.

Já um sistema de gerenciamento de banco de dados (SGBD) é um *software* que possibilita o usuário de fazer a manipulação e visualização, das informações gravadas no banco de dados. O SGBD adotado para o desenvolvimento deste projeto foi o PostgreSQL.

Fazendo o uso de um SGBD para acessar as informações do banco de dados, tem-se então o conceito de sistema de banco de dados como o conjunto de quatro componentes básicos: dados, hardware, software e usuários. (DATE, 2004) conceituou que "sistema de bancos de dados pode ser considerado como uma sala de arquivos eletrônica".

1.4 Estrutura do trabalho

Tendo como objetivo uma apresentação do cenário energético e econômico brasileiro atual, este trabalho faz um levantamento de dados de pesquisa e valores de consumos da instituição analisada no capítulo 2.

Logo após, um aprofundamento nos processos gerais de comunicação dos medidores através do protocolo Modbus e no processo de desenvolvimento de um sistema web, é feito nos capítulos 3 e 4.

Tendo a base conceitual das etapas, técnicas e problemas bem definida e contextualizada, a infraestrutura que fornecerá os dados para esse sistema e seus devidos meios de comunicação serão expostos no capítulo 5. A exposição da modelagem inicial do banco que fará o armazenamento dos medidores e suas medições, das unidades consumidoras e suas devidas faturas energéticas, bem como todas as tabelas auxiliares para o funcionamento do sistema estarão declarados no capítulo 6.

Expostos os modelos de como estarão distribuídas e armazenadas as informações serão explanados os subsistemas que irão compor a estrutura do Sistema Interpretador de Dados Energéticos (SIDE), sendo o sistema sincronizador de dados de medição descrito no capítulo 7, o sistema interpretador de faturas descrito no capítulo 8, e o sistema de interface com o usuário que fará a utilização de todos os outros subsistemas apresentado no capítulo 9.

Por fim, os resultados encontrados e confirmados ao longo do projeto são dispostos nos capítulos 10, 11 e 12, as conclusões deste apresentadas no capítulo 13 e uma análise das possibilidades e trabalhos futuros na área no capítulo 14.

Parte II

Referenciais teóricos

2 Contextualizando o Cenário Energético Brasileiro

Como mostrado anteriormente, o projeto a ser desenvolvido está totalmente inserido na abordagem de energia e consumo energético. Por isso é muito importante apresentar de maneira resumida o contexto histórico que estimula cada vez mais o estudo de soluções para possibilitar um consumo mais consciente dos recursos energéticos.

2.1 Aumento do Consumo de Energia

O consumo de energia é um dos principais índices de desenvolvimento econômico e qualidade de vida da sociedade. Através dele podemos analisar tanto o ritmo de atividade do comércio e das indústrias, quanto a disponibilidade da população em adquirir bens e serviços mais avançados tecnologicamente, como eletrônicos, automóveis, eletrodomésticos e serviços que necessitam do consumo energético para seu funcionamento.

Além do desenvolvimento econômico, outro fator ao qual o consumo de energia está diretamente relacionado é o crescimento populacional – indicador obtido pelo cálculo diferença entre as taxas de natalidade e mortalidade associado à medição de fluxos migratórios. No Brasil, no período de 2000 a 2005, o crescimento populacional teve uma tendência de queda relativa, registrando variação média anual de 1,46%, segundo relata o estudo Análise Retrospectiva constante do Plano Nacional de Energia 2030, produzido pela Empresa de Pesquisa Energética.

Ainda assim, a tendência do consumo de energia no período foi de crescimento: 13,93%, mostrando que mesmo que com uma queda do crescimento populacional a influência do crescimento econômico do cenário analisado se mostrou como mais influente sobre o consumo energético populacional. O Produto Interno Bruto do país, no mesmo período, registrou um crescimento acumulado de 14,72%, conforme dados do Ipea.

Analisando um cenário mais recente pode-se observar também aumento no consumo de energia elétrica no país, que totalizou 463.948 gigawatts-hora (GWh) em 2017, correspondendo a um crescimento de 0,8%, segundo levantamento da Empresa de Pesquisa Energética (EPE). Somente em dezembro, o consumo foi de 39.288 GWh, alta de 1,7% em relação ao verificado no mesmo período do ano anterior. (ANEEL, 2008)

O consumo no mercado cativo (atendido pelas distribuidoras) teve queda de 5,6% em 2017 e de 3% em dezembro, influenciada pela migração de consumidores para o mercado livre, que cresceu 18,4% e 13,7%, respectivamente.

Dentre as classes de consumo, destaque para o segmento industrial, que cresceu 1,3% no ano de 2017, alcançando 165.883 GWh, após duas quedas consecutivas nos anos anteriores, reflexo da melhora no cenário econômico, o que reforça a dependência direta entre o consumo e a economia.

2.2 Utilização de Fontes Renováveis

As energias renováveis se tornam cada vez mais atraentes como alternativa de micro geração distribuída uma vez que houve redução do preço de células fotovoltaicas e aero geradores, já que atualmente aliado ao poder de compra do consumidor está o alto valor de conta de energia elétrica pago pelo brasileiro, considerada uma das contas mais elevadas no mundo. (BARBOSA, 2013)

Fazendo uma análise inicial, as fontes renováveis, aparentemente, possuem um custo final mais elevado do que o sistema convencional centralizado de fornecimento energético. Entretanto o processo como um todo de produção da energia promove uma consequente redução em seu valor total quando todos os processos necessários são contabilizados.

Os recursos fósseis necessitam de extração, transporte para as refinarias onde são preparados para a queima e, após a geração de eletricidade, esta deve ser transmitida através de linhas de alta tensão para o consumidor, enquanto que os resíduos devem ser eliminados. A utilização de máquinas rotativas, tais como turbina e gerador, necessitam de uma extensa rotina de manutenção , devido ao desgaste natural das peças móveis, além de gerar poluição sonora durante o seu funcionamento. Por outro lado, a energia solar não necessita de um pesado e caro processo de extração, não demanda refinamento e nem transporte para o local da consumo, devido o mesmo ser o qual é próximo ao local de geração, evitando assim os custos com a transmissão em alta tensão. Utiliza células solares, responsáveis pela geração de energia, e um inversor para transformar a tensão gerada em corrente contínua para os valores nominais dos aparelhos de consumo em corrente alternada. Este processo é mais simples, sem emissão de gases poluentes ou ruídos e com uma pequena dependência de manutenções periódicas.(GALDINO et al., 2000)

Os custos envolvendo todas estas etapas necessárias para a geração de energia devem ser computados no momento em que se compara a energia solar com as outras fontes. Devido à sua simplicidade, esta forma renovável de obter eletricidade possui vantagens econômicas.

3 A Comunicação *Modbus*

3.1 O protocolo *Modbus*

O *Modbus* é um protocolo amplamente utilizado no setor de automação industrial, principalmente por sua simplicidade e facilidade de implementação. Este protocolo permite que seja usada uma mesma linguagem de comunicação em diversos padrões de meio físico, variando a velocidade de comunicação, quantidade de dispositivos suportados na rede bem como o comprimento máximo da mesma, fazendo assim com que cada tipo de estrutura física seja adequada à um cenário de projeto, justificando com isso a sua ampla utilização.

Os padrões de meio físico definem o modo de comunicação através do qual os dados serão transmitidos entre o mestre e seus escravos. O protocolo pode ser utilizado por uma aplicação por um meio serial ou através de conexões TCP (*Ethernet*), tendo por exemplo os padrões RS-232,RS-4222 e RS-485 para a comunicação serial e o modbus TCP para a comunicação através de endereços IP para cada dispositivo (DUTERTRE, 2007).

O padrão RS-232 (Recommended Standard 232) ou EIA-232 (Electronic Industries Alliance 232) é utilizado apenas quando se possui apenas dois dispositivos na rede, formando a chamada comunicação ponto a ponto, onde no protocolo *Modbus* representa o mestre e o escravo. Este padrão alcança velocidades em uma média de 115 Kbps, podendo ultrapassar este valor em um pequeno percentual, dependendo da qualidade estrutural da rede. A distância máxima entre o escravo e o mestre da rede é de aproximadamente 30 metros.

O padrão RS-485 (Recommended Standard 485) ou EIA-485 (Electronic Industries Alliance 485) é um dos padrões mais utilizados industrialmente, dentre os outros com comunicação serial. Uma das principais vantagens em relação ao RS-232 é que a comunicação mestre escravo não fica limitada a apenas dois dispositivos, podendo estabelecer uma comunicação de até 32 dispositivos por barramento na rede. Além disso a sua velocidade de conexão é bem superior aos 115 Kbps descritos no parágrafo anterior, podendo chegar à 50Mbps, dependo do comprimento da rede, sendo que quanto maior for, menor será a velocidade de comunicação sendo que nesse caso o alcance máximo da rede é por volta de 1200 metros.

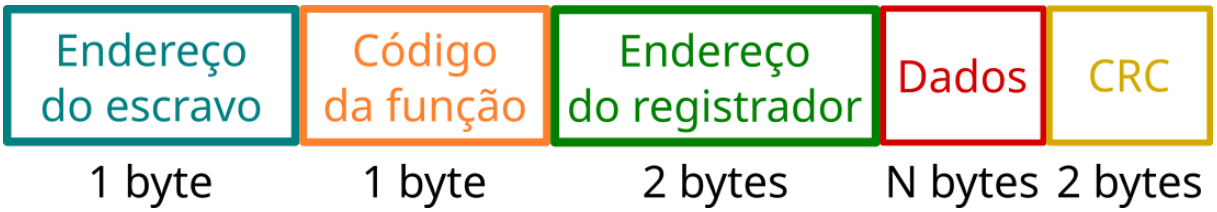
Já o protocolo TCP, como o próprio nome sugere, utiliza TCP como camada de comunicação, porém tentando manter uma compatibilidade com o protocolo serial Modbus. Para possibilitar a diferenciação de cada escravo é atribuído um endereço IP para o dispositivo através do qual o mesmo será identificado na rede, além disso o protocolo *Modbus* TCP utiliza por padrão um número de porta IP específico (502) para a comunicação. O Modbus TCP possui uma vantagem econômica devido à maior facilidade de implantação

por conta da ampla disponibilidade de redes compatíveis com a comunicação TCP, porém esta alta disponibilidade traz consigo uma preocupação maior em relação à segurança dos dados transmitidos na rede.

3.2 Formato de uma mensagem *Modbus*

A estrutura de uma mensagem transmitida com base no protocolo *Modbus* RTU é definida na figura 1, porém podemos utilizar a mesma como base para o modelo ASCII diferenciando-se por um caractere inicial ":"(ASCII 0x3Ah) e um CRLF (*Carriage Return and Line Feed*) representado em ASCII por 0x0Dh + 0x0Ah ao final da mensagem:

Figura 1 – Estrutura da Mensagem *Modbus*



Fonte: Próprio Autor

3.2.1 Endereçamento

No caso da comunicação serial a mensagem conterá inicialmente 1 byte com o endereçamento do dispositivo de destino da mensagem, podem ir de 1 a 256. Já para a comunicação TCP o início da mensagem conterá um cabeçalho que basicamente é composto pelo endereçamento IP do dispositivo de destino da mensagem.

3.2.2 Função

Neste campo de 1 byte o dispositivo mestre indicará qual o tipo de serviço ou função que será solicitada ao escravo. No protocolo *Modbus*, cada função é utilizada para acessar um tipo específico de dado. Algumas dessas funções estão descritas na tabela 1 para efeito de exemplificação.

Tabela 1 – Lista de Algumas Funções do Protocolo Modbus

Código da Função	**Descrição**
0x01	Leitura de bloco de bits do tipo coil (saída discreta)
0x02	Leitura de bloco de bits do tipo entradas discretas
0x03	Leitura de um número variável de registros retentivos (saídas analógicas ou memórias)
0x04	Leitura de um número variável de registros de entrada (entradas analógicas)
0x05	Escrita em um único bit do tipo coil (saída discreta)
0x06	Escrita em um único registrador (altera o estado de uma saída analógica)
0x07	Leitura do conteúdo de 8 estados de exceção (registro de erros)
0x08	Execução de uma série de testes para verificação da comunicação e erros internos

3.2.3 Dados

Este campo possui um tamanho em bytes variável dependendo do conteúdo de retorno da mensagem solicitada pelo mestre através do código da função passada ou da informação que será registrada no escravo, sendo que esta mensagem será inserida no corpo da mensagem pelo mestre.

3.2.4 CRC

Para os protocolos com comunicação serial, ao final da mensagem existirão 2 bytes contendo uma mensagem de verificação de erros na transmissão da mesma, preenchidos por um algoritmo de CRC (*Cyclic Redundancy Check*). Essa mensagem garantirá a consistência do envio completo da mensagem. Já para o protocolo de comunicação TCP, os bytes finais de verificação de erro já não se fazem necessários.

4 O Desenvolvimento de Aplicações *Web*

Neste capítulo, serão apresentadas as etapas envolvidas no desenvolvimento da aplicação deste trabalho, que vaõ desde o levantamento de requisitos, criação de um versionamento e utilização de conceitos de engenharia de *software*.

4.1 Levantamento de Requisitos

O levantamento de requisitos é uma importante etapa para a concepção de um projeto no âmbito da engenharia de requisitos, onde a pessoa responsável pela análise deve fazer o uso de todas informações disponíveis que definirão os requisitos para poder identificar as funcionalidades que o sistema deverá ter (WAZLAWICK, 2013).

(BEZERRA, 2016) define para o levantamento de requisitos os seguintes meios de obtenção

- questionários, tendo por objetivo a descoberta de problemas a serem abordados, identificar os principais procedimentos e definir a expectativa dos entrevistados têm a respeito do produto;
- entrevistas, que consistem em diálogos direcionadas à um fornecedor de requisitos no formato "pergunta-resposta" com um propósito específico tendo os mesmos objetivos dos questionários;
- observação, que consiste em fazer uma análise externa do comportamento e o ambiente dos indivíduos de vários níveis organizacionais, capturando as necessidades, modo de trabalho e as deficiências de cada atividade relacionada ao projeto;
- entre outros.

O processo de levantar os requisitos de um projeto é um tanto quanto complexa, pois um escopo mal definido ou um mau entendimento dos usuários quanto às capacidades e limitações do sistema computacional podem se tornar um grande problema para o projeto como um todo, além da própria volatilidade dos requisitos, que mudam ao longo da execução do mesmo (PRESSMAN; MAXIM, 2016).

Em casos onde são utilizados diagramas de atividades para modelar os principais processos aos quais a aplicação deve abranger, o levantamento de requisitos deve identificar quais as funções são necessárias para realizar as atividades destes processos corretamente (WAZLAWICK, 2013). Por este motivo, uma boa prática é que a primeira atividade

do levantamento de requisitos seja a modelagem do negócio,através da construção de diagramas de atividades que descrevam detalhadamente os processos do negócio.

4.2 Versionamento

Versionamento ou controle de versão é o uso de ferramentas para realizar o gerenciamento de alterações ou diferentes versões no desenvolvimento de um determinado documento, projeto ou aplicação.

Os sistemas de controle de versão, comumente utilizados no desenvolvimento de *software*, são ferramentas com o objetivo de gerenciar as inúmeras alterações nos documentos produzidos ao longo do processo de desenvolvimento.

De acordo com (MURTA; WERNER, 2006), do ponto de vista do desenvolvimento, o gerenciamento de software é dividido em três sistemas principais, que são: controle de modificações, controle de versões e gerenciamento de construção.

Segundo (MASON, 2006), um sistema de controle de versão é basicamente um local para armazenamento de artefatos gerados durante o desenvolvimento de um sistema. Atuando como uma espécie de controle do tempo para a equipe de desenvolvimento, as ferramentas de controle de versão permitem a navegação por múltiplos arquivos de múltiplos autores a qualquer versão anterior.

Segundo Araujo (2011) as vantagens encontradas nos sistemas que utilizam mecanismos de controle de versão são:

- Controle de histórico: Possibilita o usuário fazer a análise e comparação de versões anteriores do projeto podendo exportá-las e executá-las, podendo inclusive desfazer alterações voltando o documento ao estado em que se identificava nas versões anteriores.
- Suporte a colaboração: Possibilita a identificação de alteração de determinado documento,permitindo ao usuário interessado em alterar o mesmo documento, optar por aguardar a liberação do usuário que está trabalhando atualmente no documento, ou trabalhar de forma paralela, e posteriormente os usuários podem efetuar o *merge* (mesclar informações de dois ou mais documentos a fim de gerar um só).
- Suporte a marcação e resgate de versões estáveis: Alguns sistemas de controle de versão possuem propriedades que identificam versões estáveis, possibilitando através do histórico, selecionar uma versão estável para exportar e utilizar.
- Ramificação de projeto: Possibilita o trabalho por equipes diferentes em um mesmo produto, onde esses ramos posteriormente podem ser unidos ao projeto principal através de uma solicitação ao gerente do sistema de versionamento.

Nos documentos de especificação de software o emprego desses mecanismos de gerenciamento de versões é essencial. Os requisitos mudam constantemente devido a variados motivos, podendo eles ser desde ao fato de os problemas aos quais se refere um requisito não foram inteiramente definidos, desde uma evolução no entendimento dos desenvolvedores acerca do problema, passando por melhorias em sistemas antigos ou automatização de algum processo manual. Além disso, regras e ambiente técnico também são fatores para uma atualização nos requisitos de software. Dessa forma, requisitos podem ser atualizados, incluídos ou removidos.

Certo de que estes documentos representam determinada funcionalidade ou propriedade de um sistema, e impõe grandes responsabilidades de quem for alterá-los, fica necessário o rastreamento das mudanças, por conta disso, opções de acesso a leitura de versões anteriores, para o usuário poder comparar versões, são peças fundamentais nos gerenciadores de requisitos de software.

4.3 Engenharia de Software

De acordo com (PRESSMAN; MAXIM, 2016), a engenharia de software pode ser definida como "aplicação de uma abordagem sistemática, disciplinada e quantificável, para desenvolvimento, operação e manutenção do software, isto é a aplicação da engenharia ao software".

A evolução da complexidade no processo de produção de um software, devido à novos dispositivos, tecnologias e a melhoria nas comunicações e redes trouxe novos problemas para engenheiros e desenvolvedores. Devido estes e outros fatores, houve a necessidade de representar o processo em modelos que tentam se adaptar a essa nova realidade.

4.3.1 Modelos de Processo

Um modelo de processo de software, de acordo com (SOMMERVILLE, 2011), é uma representação abstrata de um processo de software. Cada modelo de processo representa um processo a partir de uma perspectiva particular, de maneira que proporciona apenas informações parciais sobre o processo. O modelo de processo abordado neste sistema será um modelo baseado em métodos ágeis com adaptações devido a ausência de equipe, oferecendo agilidade e uma metodologia indispensáveis aos projetos atuais, de acordo com (BECK et al., 2001).

4.3.2 Métodos Ágeis

Embora os métodos ágeis estejam sendo aplicados a muito tempo, recentemente vem se tornando cada vez mais popular no Brasil devido ao fato de ser uma abordagem simplificada. Entretanto essa simplicidade e aceleração requerem disciplina e organização.

Em 2001, um grupo de 17 autores e representantes de diversas técnicas e metodologias se reuniram com o objetivo de estabelecer um padrão de desenvolvimento de projeto com as técnicas e metodologias existentes na época. O resultado final dessa reunião foi o Manifesto para o Desenvolvimento Ágil de Software (BECK et al., 2001), que definiu um *framework* comum para processos ágeis, trazendo novas ideias e sugestões para a melhoria de processos, técnicas e métodos de desenvolvimento e gestão de projetos de forma ágil.

De acordo com (PRESSMAN; MAXIM, 2016), a utilização de métodos ágeis pode trazer benefícios como:

- maior satisfação dos clientes;
- melhoria na comunicação e aumento na colaboração entre envolvidos nos projetos;
- melhoria na qualidade do produto;
- menores custos de produção;

4.3.3 Uso de Framework para Desenvolvimento de Aplicações

Várias definições sobre framework são descritas na literatura, mas segundo (GAMMA et al., 2004) "um framework é um conjunto de classes que cooperam entre si provendo assim um projeto reutiliável para um domínio específico de classes de sistema"

Um *framework* é uma estrutura de suporte definida através da qual outro projeto de software pode ser estruturado e desenvolvido. Um *framework* pode incluir programas de suporte, bibliotecas, modelagens de dados, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um novo projeto.

Utilizando *frameworks* tem-se como principal vantagem a redução de custos, devido ao fato de já existir uma estrutura definida, podendo assim a etapa de desenvolvimento concentrar-se apenas em implementar as regras específicas do negócio em que o sistema deve atuar.

Um *framework* ainda proporciona a reutilização de códigos e a fatoração de problemas em aspectos comuns a várias aplicações, permitindo a criação de sistemas com códigos menos frágeis e menos suscetíveis à defeitos.

Parte III

Desenvolvimento dos Sistemas

5 Infraestrutura da rede de medidores

A infraestrutura da rede de medidores da Universidade Federal de Goiás foi montada com o intuito de permitir realizar o monitoramento do consumo energético em vários níveis, desde uma medição geral, de maneira semelhante ao que é feito pela concessionária de energia, passando por uma medição em cada QDC de cada prédio, até a medição de um equipamento em específico, por exemplo o sistema de ar condicionado de um centro de aulas. Além disso essa infraestrutura deve contemplar o monitoramento da Potência Gerada pela planta fotovoltaica dos *campus*.

Cada medidor em cada um desses pontos é um escravo da rede *Modbus* de medição e possui um identificador próprio que o separa dos demais, portanto deve-se também ter um mestre para gerenciar essa estrutura, permitindo requisitar os dados de medição, cadastrar novos medidores, alterar endereçamento IP, identificar o status atual de cada medidor.

Abaixo, será descrito cada ponto de medição e após isso, será exposto um visual esquemático geral dessa rede onde cada ponto terá uma representatividade, demonstrando como cada comunicação em particular forma a infraestrutura completa da rede de medidores. Todos os dispositivos utilizados para compor a rede de medição são da marca CCK, incluindo o *software* de monitoramento que cria o mestre da rede e permite o gerenciamento dos escravos.

5.1 Medição de Fronteira

Na medição de fronteira, é instalado o dispositivo CCK 6700E que é responsável por fazer a medição das fases, onde através de uma tomada de isolação ótica CCK 50, ele se conecta ao medidor da concessionária fazendo assim a leitura dos dados e armazenando em uma memória de massa que suporta um período de 35 dias, sobrescrevendo o primeiro dia após a inserção do trigésimo sexto. A figura 2 mostra uma representação desse sistema de medição.

Figura 2 – Esquema Conexões Medição de Fronteira

Fonte: Ltda. (2019)

Feita essa comunicação com medidor, a comunicação do CCK 6700E com o mestre pode ser feito por:

- Cabo UTP - É feita a conexão direta entre o dispositivo e o *Switch* com a porta configurada para a VLAN da rede *Modbus*
- Fibra Ótica - Através de um conversor de mídia o medidor é conectado à rede por fibra ótica onde na outra extremidade é usado outro conversor para receber a fibra e converter para UTP novamente
- Comunicação por Rádio - Um dispositivo recebe os sinais dos medidores através do protocolo RS485, enviando os dados por rádio frequência até outro dispositivo que recebe esse sinal e envia para um dispositivo CCK 7010W que faz a conversão de comunicação serial para Ethernet.

5.2 Medição Interna

Cada medidor de fronteira das unidades consumidoras fornece energia para um ou mais prédios de determinada região, sendo assim para um monitoramento mais preciso, faz-se necessário a implantação de subsistemas de medição específicos para cada edificação ou até mesmo em cenários onde se quer ter um controle sobre um consumo mais específico, pode-se ser instalado um medidor em um circuito mais fechado.

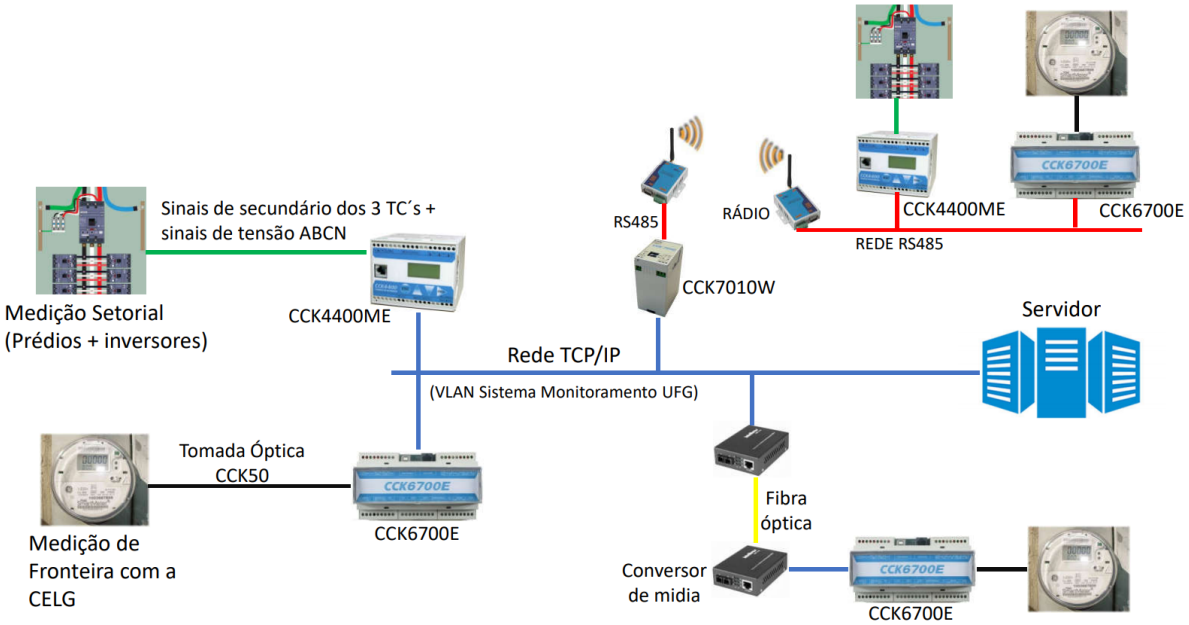
O dispositivo responsável por essa medição é o CCK 4400ME, fazendo a leitura e enviando através dos três modos citados na seção 5.1. Esse aparelho também possui memória de massa para armazenamento de até 35 dias de leitura.

Além da medição de prédios e equipamentos específicos, o CCK 4400ME é responsável por fazer a leitura de geração dos sistemas de placas fotovoltaicas dos blocos da Universidade. Essa medição é feita através de conexão com a saída do inversor do sistema gerador, enviando os dados para o mestre da rede através dos 3 tipos de comunicação disponíveis.

5.3 Diagrama Esquemático Infraestrutura da Rede

Descrito os modos de medição e comunicação dos diferentes pontos da rede, pode-se resumir todos os modos de conexão utilizados para os vários cenários da rede de medições através do diagrama esquemático da figura 3.

Figura 3 – Diagrama Esquemático Rede Modbus de Monitoramento da Universidade Federal de Goiás

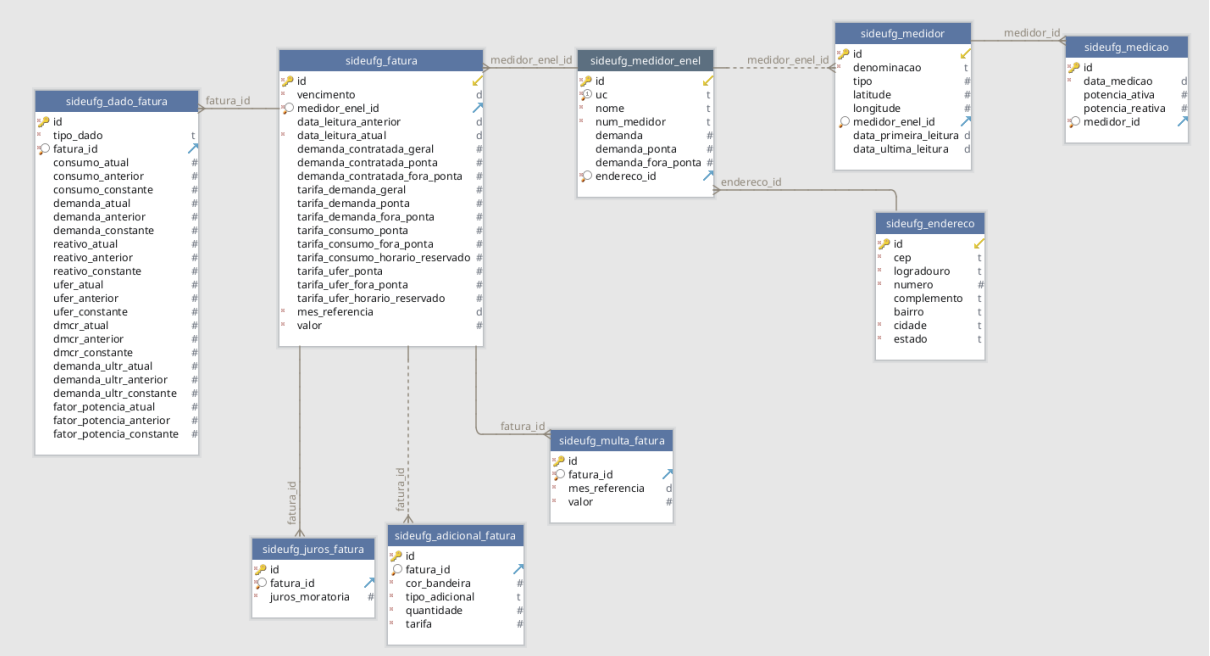


6 Estrutura de Banco de Dados

Após da descrição do Sistema de infraestrutura da rede feita no capítulo anterior, é necessário que os dados dos medidores sejam armazenados para possibilitar consultas futuras. Para isso foi desenvolvida uma estrutura de dados onde pode-se armazenar os dados dos medidores e de suas respectivas medições.Além disso, outro requisito do sistema era que os dados referente às unidades consumidoras pudessem ser armazenados, bem como os dados das suas faturas geradas pela concessionária de energia, com os valores oficiais de medição e cobrança.

Na figura 4 pode-se observar o layout do modelo de dados com as entidades principais do projeto e seus atributos, bem como suas relações.

Figura 4 – Diagrama do Modelo de Dados SIDE



Fonte: Próprio Autor

A modelagem da figura 4, pode ser visualizada através de um modelo online criado no *software DbSchema*, acessível através do link disponível no apêndice A.

Devido ao uso de um *framework* para desenvolvimento ágil não se fez necessário preocupar-se com as estruturas de dados referentes a acessos dos usuários, cadastro, definição de papéis, preferências de acesso, bem como criação de versionamento para a criação de um histórico de alteração dos dados do banco. Na figura acima foram omitidas todas as tabelas criadas pelo *framework* Cuba, com a finalidade de focar o modelo no

problema específico da aplicação. Toda a documentação referente ao *framework* pode ser encontrada em seu *site* oficial.

6.1 Tabela de Medidores da Concessionária

A tabela onde é feito o cadastro dos Medidores da Concessionária é a tabela **sideufg_medidor_enel** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **uc** - Campo obrigatório do tipo *varchar(20)* para armazenar a número da Unidade Consumidora;
- **nome** - Campo obrigatório do tipo *varchar(100)* para armazenar uma denominação para o Medidor;
- **num_medidor** - Campo obrigatório do tipo *varchar(20)* para armazenar o número do medidor daquele ponto de medição;
- **demanda** - Campo do tipo *integer* para armazenar a demanda contratada para aquela Unidade Consumidora quando a mesma não possui contrato específico por período de medição;
- **demanda_ponta** - Campo do tipo *integer* para armazenar a demanda contratada para aquela Unidade Consumidora no período de ponta;
- **demanda_fora_ponta** - Campo do tipo *integer* para armazenar a demanda contratada para aquela Unidade Consumidora no período fora de ponta;
- **endereco_id** - Campo do tipo UUID com uma chave estrangeira para armazenar a chave primária do endereço relacionado àquele medidor;

6.2 Tabela de Medidores da UFG

A tabela onde é feito o cadastro dos Medidores da CCK é a tabela **sideufg_medidor** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **denominacao** - Campo obrigatório do tipo varchar(255) para armazenar uma denominação para o Medidor;
- **tipo** - Campo do tipo *integer* para armazenar o tipo do medidor podendo ser:

  10 - CCK 4400ME;

  20 - CCK 6700E;
- **latitude** - Campo do tipo *float8* para armazenar a latitude do medidor para georreferenciamento;
- **longitude** - Campo do tipo *float8* para armazenar a longitude do medidor para georreferenciamento;
- **medidor_enel_id** - Campo do tipo UUID com uma chave estrangeira para armazenar a chave primária da unidade consumidora relacionada àquele medidor;
- **data_primeira_leitura** - Campo do tipo *timestamp* para armazenar o momento a partir do qual o medidor começou a enviar dados para a base;
- **data_ultima_leitura** - Campo do tipo *timestamp* para armazenar o momento da última leitura que o *SIDE Synchronizer* obteve do medidor;

6.3 Tabela de Medições

A tabela onde é feito o cadastro das medições feitas pelos dispositivos da CCK é a tabela **sideufg_medicao** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **data_medicao** - Campo do tipo *timestamp* para armazenar o momento no qual o medidor fez o registro da medição em sua memória de massa;
- **potencia_ativa** - Campo do tipo *float8* para armazenar o valor de potencia ativa medido;
- **potencia_reativa** - Campo do tipo *float8* para armazenar o valor de potencia reativa medido;
- **medidor_id** - Campo do tipo UUID com uma chave estrangeira para armazenar a chave primária do medidor CCK relacionado àquela medição;

6.4 Tabela de Endereços

A tabela onde é feito o cadastro dos Endereços das Unidades Consumidoras da UFG é a tabela **sideufg_endereco** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **cep** - Campo obrigatório do tipo *varchar(8)* para armazenar o CEP do endereço;
- **logradouro** - Campo obrigatório do tipo *varchar(100)* para armazenar o logradouro do endereço;
- **numero** - Campo obrigatório do tipo *integer* para armazenar o número do endereço;
- **complemento** - Campo do tipo *varchar(100)* para armazenar o complemento do endereço;
- **bairro** - Campo do tipo *varchar(100)* para armazenar o bairro do endereço;
- **cidade** - Campo obrigatório do tipo *varchar(100)* para armazenar a cidade do endereço;
- **estado** - Campo obrigatório do tipo *varchar(50)* para armazenar a sigla do estado do endereço, onde a busca pela sigla ocorre através de uma seleção no cadastro através de um *enumeration*;

6.5 Estrutura de Dados para Armazenamento das Faturas

As faturas energéticas das unidades consumidoras possuem um grande volume de dados que são importantes para as análises econômicas, quantitativas e qualitativas do consumo das edificações da Universidade. Desde os dados financeiros como valor, tarifas individuais, adicionais tarifários, contratos de demanda, até dados referentes ao consumo propriamente dito com as leituras nos horários específicos de ponta, fora de ponta e horário reservado.

Para a modelagem dessa estrutura de dados foi pensado em uma divisão em cinco tabelas, sendo uma tabela principal contendo os principais dados da fatura, e quatro auxiliares contendo desde dados esporádicos como multa e juros até dados repetitivos como as várias leituras de demanda, consumo, UFER, entre outros na fatura.

As tabelas criadas referente ao armazenamento das faturas estão listadas abaixo e serão detalhadas nas sub-sessões 6.5.1, 6.5.2, 6.5.3, 6.5.4 e 6.5.5:

- sideufg_fatura;
- sideufg_dado_fatura;
- sideufg_adicional_fatura;
- sideufg_multa_fatura;
- sideufg_juros_fatura;

6.5.1 Tabela de Faturas

A tabela onde é feito o cadastro das faturas de cada Unidade Consumidora é a tabela **sideufg_fatura** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **vencimento** - Campo obrigatório do tipo *date* para armazenar a data de vencimento da fatura;
- **medidor_enel_id** - Campo obrigatório do tipo UUID com uma chave estrangeira para armazenar a chave primária da unidade consumidora relacionada àquela fatura;
- **data_leitura_anterior** - Campo do tipo *date* para armazenar a data da leitura da fatura anterior;
- **data_leitura_atual** - Campo obrigatório do tipo *date* para armazenar a data da leitura atual da fatura;
- **demanda_contratada_geral** - Campo do tipo *integer* para armazenar a demanda contratada naquele mês para aquela Unidade Consumidora quando a mesma não possui contrato específico por período de medição;
- **demanda_contratada_ponta** - Campo do tipo *integer* para armazenar a demanda contratada naquele mês para aquela Unidade Consumidora no período de ponta;
- **demanda_contratada_fora_ponta** - Campo do tipo *integer* para armazenar a demanda contratada naquele mês para aquela Unidade Consumidora no período fora de ponta;

- **tarifa_demanda_geral** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da demanda geral naquele mês;

- **tarifa_demanda_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da demanda no horário de ponta naquele mês;

- **tarifa_demanda_fora_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da demanda no horário fora de ponta naquele mês;

- **tarifa_consumo_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa do consumo no horário de ponta naquele mês;

- **tarifa_consumo_fora_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa do consumo no horário fora de ponta naquele mês;

- **tarifa_consumo_horario_reservado** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa do consumo no horário reservado naquele mês;

- **tarifa_ufer_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da UFER no horário de ponta naquele mês;

- **tarifa_ufer_fora_ponta** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da UFER no horário fora de ponta naquele mês;

- **tarifa_ufer_horario_reservado** - Campo do tipo *big decimal (19,6)* para armazenar a tarifa da UFER no horário reservado naquele mês;

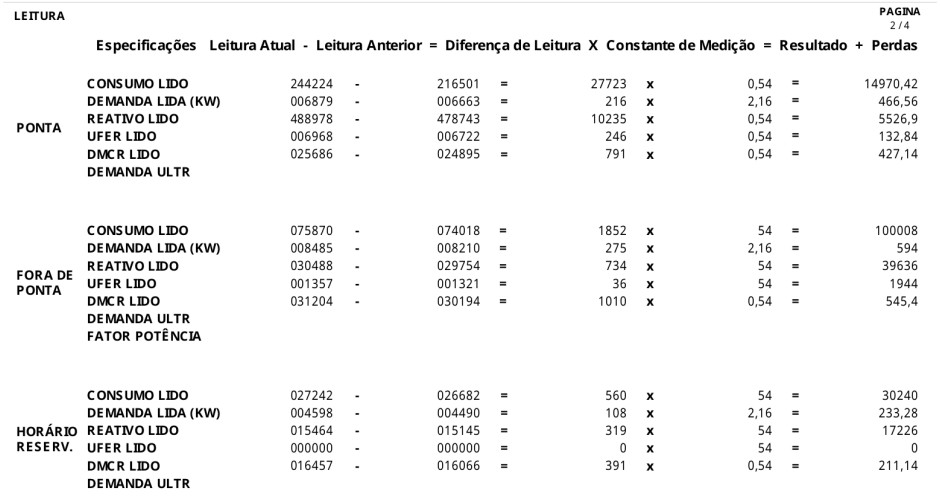
- **mes_referencia** - Campo obrigatório do tipo *date* para armazenar o mês de referência da fatura;

- **valor** - Campo obrigatório do tipo *big decimal (19,2)* para armazenar o valor final da fatura;

6.5.2 Tabela dos Dados da Fatura

A tabela onde é feito o cadastro dos dados específicos da fatura adquiridos pela interpretação da tabela da segunda página das faturas como a do exemplo da figura 5 é a tabela **sideufg_dado_fatura** onde os registros são identificados através do campo **id** do tipo UUID.

Figura 5 – Modelo de Tabela de Dados Específicos da Fatura



LEITURA — PAGINA 2 / 4

	Especificações	Leitura Atual	-	Leitura Anterior	=	Diferença de Leitura	X	Constante de Medição	=	Resultado	+	Perdas
PONTA	CONSUMO LIDO	244224	-	216501	=	27723	x	0,54	=			14970,42
	DEMANDA LIDA (KW)	006879	-	006663	=	216	x	2,16	=			466,56
	REATIVO LIDO	488978	-	478743	=	10235	x	0,54	=			5526,9
	UFER LIDO	006968	-	006722	=	246	x	0,54	=			132,84
	DMCR LIDO	025686	-	024895	=	791	x	0,54	=			427,14
	DEMANDA ULTR											
FORA DE PONTA	CONSUMO LIDO	075870	-	074018	=	1852	x	54	=			100008
	DEMANDA LIDA (KW)	008485	-	008210	=	275	x	2,16	=			594
	REATIVO LIDO	030488	-	029754	=	734	x	54	=			39636
	UFER LIDO	001357	-	001321	=	36	x	54	=			1944
	DMCR LIDO	031204	-	030194	=	1010	x	0,54	=			545,4
	DEMANDA ULTR											
	FATOR POTÊNCIA											
HORÁRIO RESERV.	CONSUMO LIDO	027242	-	026682	=	560	x	54	=			30240
	DEMANDA LIDA (KW)	004598	-	004490	=	108	x	2,16	=			233,28
	REATIVO LIDO	015464	-	015145	=	319	x	54	=			17226
	UFER LIDO	000000	-	000000	=	0	x	54	=			0
	DMCR LIDO	016457	-	016066	=	391	x	0,54	=			211,14
	DEMANDA ULTR											

Fonte: Próprio Autor

Nesta tabela são armazenados os seguintes campos:

- **tipo_dado** - Campo obrigatório do tipo *varchar(50)* para armazenar o tipo da leitura, podendo ser:

  P - Ponta;

  F - Fora de Ponta;

  H - Horário Reservado;

- **fatura_id** - Campo obrigatório do tipo UUID com uma chave estrangeira para armazenar a chave primária da fatura relacionada àquele dado;

- **consumo_atual** - Campo do tipo *integer* para armazenar a leitura atual de consumo do dado da fatura;

- **consumo_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de consumo do dado da fatura;

- **consumo_constante** - Campo do tipo *float8* para armazenar a constante de medição do consumo do dado da fatura;

- **demanda_atual** - Campo do tipo *integer* para armazenar a leitura atual de demanda do dado da fatura;

- **demanda_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de demanda do dado da fatura;

- **demanda_constante** - Campo do tipo *float8* para armazenar a constante de medição da demanda do dado da fatura;
- **reativo_atual** - Campo do tipo *integer* para armazenar a leitura atual de reativo do dado da fatura;
- **reativo_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de reativo do dado da fatura;
- **reativo_constante** - Campo do tipo *float8* para armazenar a constante de medição do reativo do dado da fatura;
- **ufer_atual** - Campo do tipo *integer* para armazenar a leitura atual de consumo de energia reativa do dado da fatura;
- **ufer_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de consumo de energia reativa do dado da fatura;
- **ufer_constante** - Campo do tipo *float8* para armazenar a constante de medição do consumo de energia reativa do dado da fatura;
- **dmcr_atual** - Campo do tipo *integer* para armazenar a leitura atual de demanda máxima corrigida do dado da fatura;
- **dmcr_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de demanda máxima corrigida do dado da fatura;
- **dmcr_constante** - Campo do tipo *float8* para armazenar a constante de medição da demanda máxima corrigida do dado da fatura;
- **demanda_ultr_atual** - Campo do tipo *integer* para armazenar a leitura atual de demanda ultrapassada do dado da fatura;
- **demanda_ultr_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de demanda ultrapassada do dado da fatura;
- **demanda_ultr_constante** - Campo do tipo *float8* para armazenar a constante de medição da demanda ultrapassada do dado da fatura;
- **fator_potencia_atual** - Campo do tipo *integer* para armazenar a leitura atual de fator de potência do dado da fatura;
- **fator_potencia_anterior** - Campo do tipo *integer* para armazenar a leitura anterior de fator de potência do dado da fatura;
- **fator_potencia_constante** - Campo do tipo *float8* para armazenar a constante de fator de potência do dado da fatura;

6.5.3 Tabela dos Adicionais Tarifários da Fatura

A tabela onde é feito o cadastro dos adicionais tarifários de bandeira de cada fatura é a tabela **sideufg_adicional_fatura** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **fatura_id** - Campo obrigatório do tipo UUID com uma chave estrangeira para armazenar a chave primária da fatura relacionada àquele adicional;
- **cor_bandeira** - Campo obrigatório do tipo *integer* para armazenar a cor da bandeira do adicional daquele mês para aquela fatura, podendo ser:

  1 - Verde;

  2 - Amarela;

  3 - Vermelha;

- **tipo_adicional** - Campo obrigatório do tipo *varchar(50)* para armazenar o tipo do adicional tarifário, podendo ser:

  P - Ponta;

  F - Fora de Ponta;

  H - Horário Reservado;

- **quantidade** - Campo obrigatório do tipo *big decimal (13,2)* para armazenar a quantidade do adicional a ser cobrado naquele mês naquela fatura;
- **tarifa** - Campo obrigatório do tipo *big decimal (19,6)* para armazenar a tarifa do adicional a ser cobrado naquele mês naquela fatura;

6.5.4 Tabela das Multas da Fatura

A tabela onde é feito o cadastro das multas aplicadas à cada fatura é a tabela **sideufg_multa_fatura** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **fatura_id** - Campo obrigatório do tipo UUID com uma chave estrangeira para armazenar a chave primária da fatura relacionada àquela multa;
- **mes_referencia** - Campo obrigatório do tipo *date* para armazenar o mês de referência da multa da fatura;

- **valor** - Campo obrigatório do tipo *big decimal (13,2)* para armazenar o valor da multa da fatura;

6.5.5 Tabela dos Juros da Fatura

A tabela onde é feito o cadastro dos juros monetários aplicados à cada fatura é a tabela **sideufg_juros_fatura** onde os registros são identificados através do campo **id** do tipo UUID.

Nesta tabela são armazenados os seguintes campos:

- **fatura_id** - Campo obrigatório do tipo UUID com uma chave estrangeira para armazenar a chave primária da fatura relacionada àquele juros;
- **juros_moratoria** - Campo obrigatório do tipo *big decimal (13,2)* para armazenar o valor dos juros da fatura;

7 Sistema Sincronizador de Dados

Descrita a estrutura de banco de dados do projeto, bem como a infraestrutura da rede *Modbus* de medição da UFG, pode-se entender agora o modo de funcionamento do Sistema Responsável por alimentar o banco com as informações capturadas dos medidores. Para isso foi desenvolvido uma aplicação na linguagem *node js*, que é acionada pelo *crontabs* de um servidor Linux com o acionamento a cada minuto.

O código do *SIDE Synchronizer* é composto basicamente de dois momentos, como mostrado no algoritmo 1:

1. O sistema acessa o Mestre da rede Modbus e requisita a lista de escravos cadastrados, atualizando o banco de dados com as alterações do mesmo.

2. O sistema utiliza essa lista de medidores para requisitar as últimas medições feitas em cada medidor a partir da data da última leitura de cada um.

Essas duas etapas serão melhor descritas nas sessões 7.1 e 7.2, porém, por hora, pode-se analisar o código abaixo da classe principal do *SIDE Synchronizer*.

sidesynchronizer.js

```
1  const medidores = require('./Medidores/medidores.js')
2  const medicoes = require('./Medidores/medicoes.js')
3
4  var medidoresRetornados = [];
5  async function start () {
6     medidoresRetornados = await medidores.sincronizaMedidoresCCK()
7     for (var medidor of medidoresRetornados){
8        await medicoes.sincronizaMedicoesCCK(medidor)
9     }
10    process.exit();
11 }
12
13
14 start();
```

Algoritmo 1 – Código Principal *SIDE Synchronizer*

Inicialmente, na linha 4 inicializa-se uma lista de medidores que receberá a listagem do método medidores.sincronizaMedidoresCCK() na linha 6. Após o retorno do método e a população do *array*, faz-se a iteração por ele onde a cada iteração é executado o método medicoes.sincronizaMedicoesCCK(medidor) passando o medidor como parâmetro.

Após a iteração, na linha 10 se encerra o processo, que em média leva 1,2 segundos para executar na atual situação da planta de medidores da UFG. após 1 minuto o cron do *Linux* chama esse processo novamente, mantendo sempre assim a base atualizada com as medições da rede *Modbus*.

A seguir serão detalhados os dois métodos que fazem a sincronização dos medidores e suas respectivas medições.

7.1 Sincronização de Medidores

Descrito o funcionamento Geral da aplicação, pode-se analisar o código resumido abaixo, da classe responsável pela sincronização inicial dos medidores, como descrito no algoritmo 2.

medidores.js

```
26 exports.sincronizaMedidoresCCK = async function () {
27     console.log('listando medidores CCK')
28     medidoresCCK = await listaMedidoresCCK();
29     medidoresCCK = await processaDatasMedidores(medidoresCCK)
30     await listaMedidoresBanco(medidoresCCK);
31     console.log('Atualizando medidores do Banco')
32     await insereMedidoresNoBanco(medidoresCCK);
33     return medidoresCCK
34 }
35
36 async function listaMedidoresCCK() {
37     return new Promise((resolve, reject) => {
38         var retornoConsulta = [];
39
40         fetch(urlListagem).then(function (u) {
41             return u.text().then(function (val) {
42                 xml2js.parseString(val, { trim: true }, function (err, result) {
43                     medidoresRetornoCCK = JSON.parse(JSON.stringify(result))
44                     medidoresRetornoCCK.medidores.medidor.forEach(function (medidor) {
45                         var medidorToAdd =
46                         new Medidor(medidor.$.id_medidor,medidor.$.nm_medidor);
47                         retornoConsulta.push(medidorToAdd);
48                     });
49                 });
50                 resolve(retornoConsulta);
51             }).catch(function (err) {
52                 // handle error here
53             });
54         });
55
56     });
57
58 }
```

Algoritmo 2 – Código Resumido da Classe de Medidores

O método sincronizaMedidoresCCK faz inicialmente uma listagem de todos os medidores cadastrados na rede e os armazena na variável medidoresCCK, onde na linha 38 inicializa-se uma lista de retornos da consulta que será populada com os medidores cadastrados na rede *Modbus*. Na linha 40 é feita uma requisição http padrão na Após o retorno do método e a população do *array*, faz-se a iteração por ele onde a cada iteração é executado o método medicoes.sincronizaMedicoesCCK(medidor) passando o medidor como parâmetro.

Após a listagem, é feita no servidor CCK uma outra validação, onde ma linha 29 após passar os medidores se obtém para cada um a data da última e da primeira leitura que o dispositivo fez, atualizando assim mais uma vez a lista medidoresCCK.

Após o preenchimento é buscado no banco de dados os medidores já registrados e feito o cadastro caso não existam na base, e atualização da denominação, e datas de primeira e ultima leitura caso já existam na base. Ao final é retornado a lista de medidores atualizada.

7.2 Sincronização de Medições

Após o retorno da sincronização dos medidores fornecer a lista atualizada de medidores, pode-se iterar essa lista chamando-se, para cada medidor, a sincronização de medições exibida no algoritmo 3.

medicoes.js

```
19 exports.sincronizaMedicoesCCK = async function (medidor) {
20     console.log('listando medições Medidor ' + medidor.denominacao)
21     var ultimaLeitura = await buscaUltimaLeituraMedidorNoBanco(medidor);
22     if (!!ultimaLeitura.dataMedicao) {
23         medicoesCCK = await listaUltimasLeituras(ultimaLeitura);
24     } else {
25         medicoesCCK = await listaTodasLeituras(medidor);
26     }
27     if (medicoesCCK.length > 0) {
28         console.log('Inserindo ' + medicoesCCK.length
29             + ' medições do Medidor ' + medidor.denominacao);
30         await insereMedicoesNoBanco(medicoesCCK);
31     }
32 }
```

Algoritmo 3 – Código Resumido da Classe de Medições

Inicialmente, na linha 21 é feita uma busca no banco pela última medição do medidor, e posteriormente é feita a seguinte validação: se a data de medição da última leitura não for nula é feita uma busca das últimas leituras existentes no servidor a partir daquela data, caso contrário, entende-se que não existem leituras na tabela de medições para aquele medidor e é feita uma busca por todas as leituras daquele escravo no servidor.

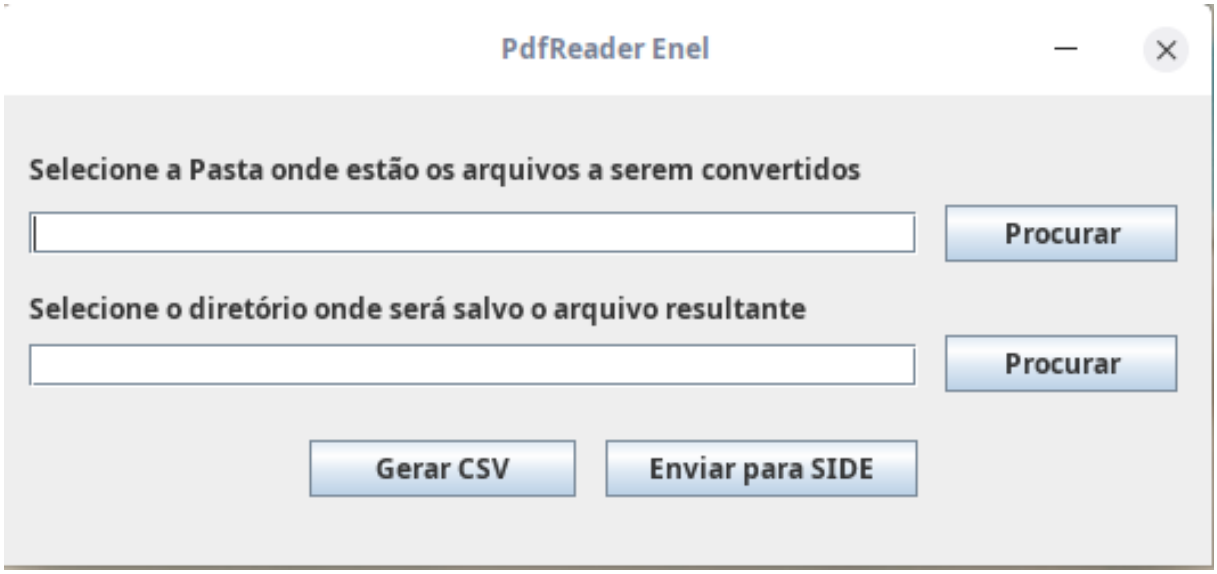
Após a busca das medições caso ela apresente retorno, as medições são inseridas no banco para este medidor. Este processo é iterado para todos os medidores existentes e após isso a execução do sincronizador é finalizada.

8 Sistema Interpretador de Faturas

Descrita a parte de obtenção dos dados dos medidores, uma outra importante fonte de dados, principalmente econômicos, são as faturas de energia, porém devido ao grande número de unidades consumidoras da UFG essa análise e obtenção dos dados apenas de maneira manual seria algo com um alto custo, por isso, foi desenvolvido um sistema capaz de fazer uma carga em massa das faturas, que fosse capaz de interpretar os dados e gerar um resultado para análise tanto por softwares de *business intelligence* tanto como poder alimentar o banco de dados do SIDE.

Este sistema pode ser executado independente do sistema operacional por se tratar de uma aplicação java, que é multiplataforma, e de maneira independente do banco de dados, sendo capaz de gerar um arquivo CSV com os dados extraídos das faturas contidas em um diretório selecionado e seus sub-diretórios, além de ser possível a alimentação do banco quando o dispositivo no qual for executado, tiver acesso a rede e os dados de acesso ao banco de dados do SIDE. A figura 6 segue uma captura da tela do sistema *PdfReader*:

Figura 6 – Tela Principal do Sistema PdfReader



Fonte: Próprio Autor

8.1 Processo de Interpretação da Fatura

Após selecionado o diretório de origem e o de destino do arquivo resultante, ao clicar em um dos dois botões de ação é executado o código do algoritmo 4:

App.java

```
26      public static void app(File path, File dest)
27                      throws IOException, DocumentException {
28              File file = new File(DEST);
29              file.getParentFile().mkdirs();
30              Collection<File> sources = listaDiretorios(path);
31              Object[] paths = sources.toArray();
32              int control = 0;
33              String source;
34              while (control < sources.size()) {
35                      source = paths[control].toString();
36                      /*******************************************************/
37                      // Debug de Paginas Do PDF //
38                      // 1 - Arquivo output/parsed.txt contém o parcial da //
39                      // conversão da última página lida pelo pdfReader //
40                      // //
41                      /*******************************************************/
42
43                      if (source.endsWith(".pdf")) {
44                              new App().parse(source, 1);
45                              Parser.readPage1(source, dest);
46                              new App().parse(source, 2);
47                              Parser.readPage2(source, dest);
48                      }
49                      control++;
50              }
51              return;
52      }
```

Algoritmo 4 – Código Principal *PdfReader*

Inicialmente é criado um arquivo temporário para armazenamento das conversões de PDF para texto puro das faturas, após é inicializado uma *Collection* com todos os arquivos dentro do diretório passado como parâmetro através da variável *path*, e esses arquivos são iterados fazendo inicialmente em cada arquivo a validação se o mesmo é um arquivo com a extensão PDF após isso é feita interpretação da primeira página da fatura onde, dependendo da opção selecionada será carregado o CSV ou o banco com esses dados coletados.

Após a leitura da primeira página o mesmo processo ocorre para a segunda página, e assim sucessivamente até todos os arquivos terem sido lidos, retornando para a tela principal com a mensagem de leitura efetivada com sucesso.r

9 Interface de Usuário

Todo o sistema para alimentação dos dados de consumo, bem como a estrutura dos dados em si está definida, porém falta um sistema capaz se possibilitar uma visualização real pelo usuário dos dados coletas, bem como uma interface que permita que sejam cadastrados dados de gerenciamento dos dispositivos de medição.

Para isso foi desenvolvida a interface *web* do SIDE, como sendo a interface principal do sistema, sendo ela a responsável por cadastro de usuários para a utilização do sistema, visualização dos dados de medição dos medidores CCK, cadastro das unidades consumidoras, visualização georreferenciada dos dispositivos instalados, entre outras.

9.1 Uso do *CUBA Framework*

Para o desenvolvimento da Interface do SIDE, foi escolhida a linguagem de programação Java com o uso de um *framework* que possibilitasse o desenvolvimento ágil facilitando na criação das entidades e seus respectivos relacionamentos e desse base para a criação de uma boa visualização para o usuário.

A plataforma escolhida para desenvolvimento foi o *CUBA Framework*, utilizado em sua versão 6.10 cuja a documentação pode ser acessada através do anexo A.

A interface de usuário disponibilizada pelo CUBA pode ser tanto uma interface padrão utilizando *Spring* quando uma interface mais personalizável utilizando *Polymer 2*, que é um framework JavaScript da empresa Google para desenvolvimento web baseado na utilização de *Web Components*, que foi a escolhida para o desenvolvimento deste projeto.

9.2 Uso da biblioteca *Polymer 2*

Escolhida a Interface de uso Polymer precisa-se entender melhor os padrões de desenvolvimento que a mesma utiliza.

O Polymer é uma biblioteca JavaScript usada para a criação de aplicações web usando o conceito de *Web Components*. Esses *Web Components* podem ser descritos como elementos reutilizáveis que podem ser inseridos em páginas ou aplicações *web*, podendo inclusive ser mesclado com outras bibliotecas JavaScript. O Polymer é como um bloco de montar, que pode ser encaixado de diversas maneiras para construir diferentes tipos de aplicações.

Nesta versão do CUBA é utilizada a versão 2 do Polymer, cuja a documentação pode ser acessada através do anexo B.

9.3 Telas do SIDE

Descrito os frameworks utilizados, pode-se agora definir os modos de funcionamento das telas desenvolvidas para o sistema.

9.3.1 Login

A tela inicial do sistema é a tela de login, mostrada na figura 7, sendo responsável pela autenticação do usuário cadastrado no sistema, gerando uma chave de sessão válida para o mesmo, liberando assim o seu acesso ao sistema.

Esta tela permite também a troca da senha atual do usuário, enviando uma nova senha temporária para o endereço de e-mail cadastrado para o mesmo. Abaixo temos uma imagem da tela de login.

Figura 7 – Tela de Login do SIDE

Fonte: Próprio Autor

Além disso caso não possua acesso ainda, pode ser feito o cadastro do usuário, fornecendo seu nome, e-mail, cpf, e a senha desejada. Este cadastro irá gerar um acesso básico ao sistema que poderá ser controlado pelo administrador do sistema. Na figura 8 temos uma imagem da tela de cadastro.

Figura 8 – Tela de Cadastro do SIDE

Fonte: Próprio Autor

9.3.2 Tela Principal

Após autenticado no sistema o usuário tem acesso ao uso do SIDE, onde inicialmente ele verá um mapa com todos os pontos de medição cadastrado no sistema que possuam suas coordenadas geográficas cadastradas. Esses pontos podem ser clicados, fornecendo informações mais detalhadas como consumo atual através da última leitura, denominação, data da última leitura, unidade consumidora vinculada à ele, e o tipo de medidor.

O mapa foi construído em cima de um *Web Component* do Polymer fornecido pela Google para acesso a api do Google Maps. Na figura 9 temos uma imagem da *home* do sistema.

Figura 9 – Tela Principal do SIDE

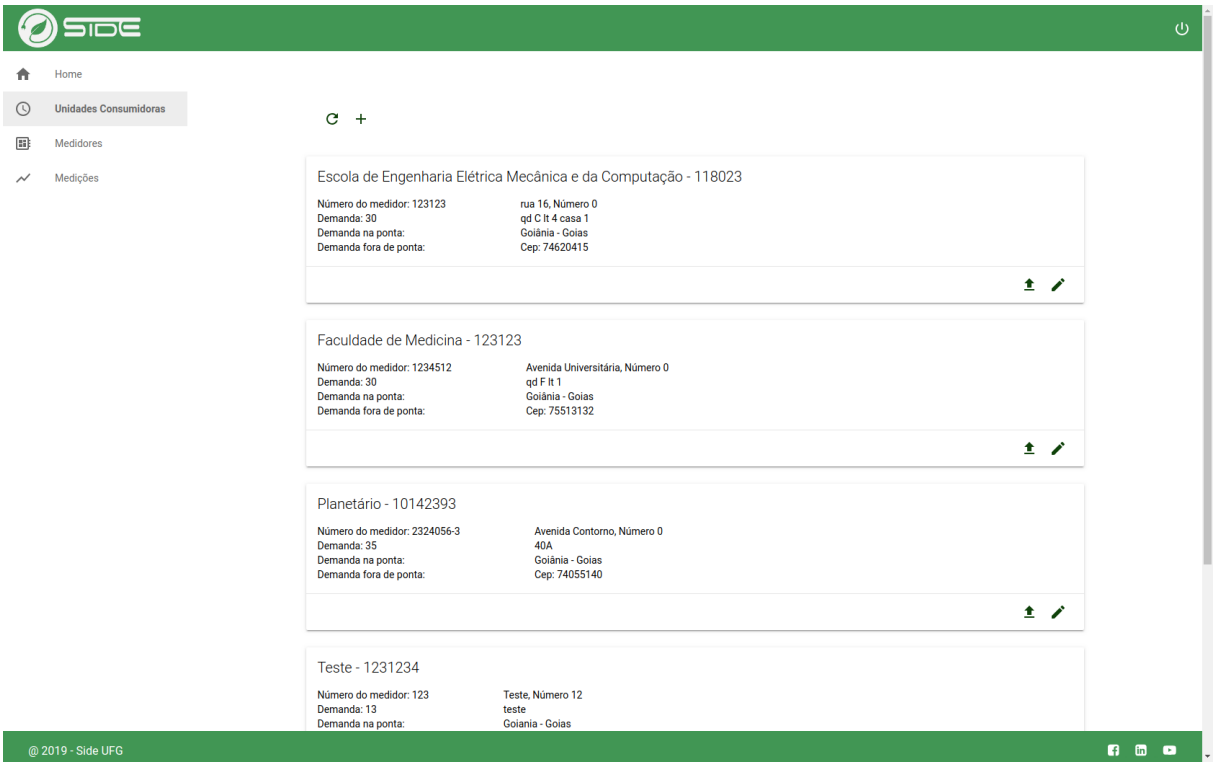
Fonte: Próprio Autor

9.3.3 Cadastro de Unidades Consumidoras

No menu de Unidades Consumidoras podemos visualizar os medidores da concessionária cadastrados na base, descriminados pela junção de seu nome com a número da unidade consumidora do mesmo.

Na listagem de cada unidade, pode-se ver o número do medidor e suas demandas contratadas na primeira coluna, e o endereço completo na segunda coluna. Na figura 10 temos as ações disponíveis para aquele item, sendo a primeira o *upload* de faturas daquela unidade consumidora, onde esta relação será validada no momento do *upload*, e o segundo ícone, mostra a edição do item em si.

Figura 10 – Tela de Listagem das Unidades Consumidoras

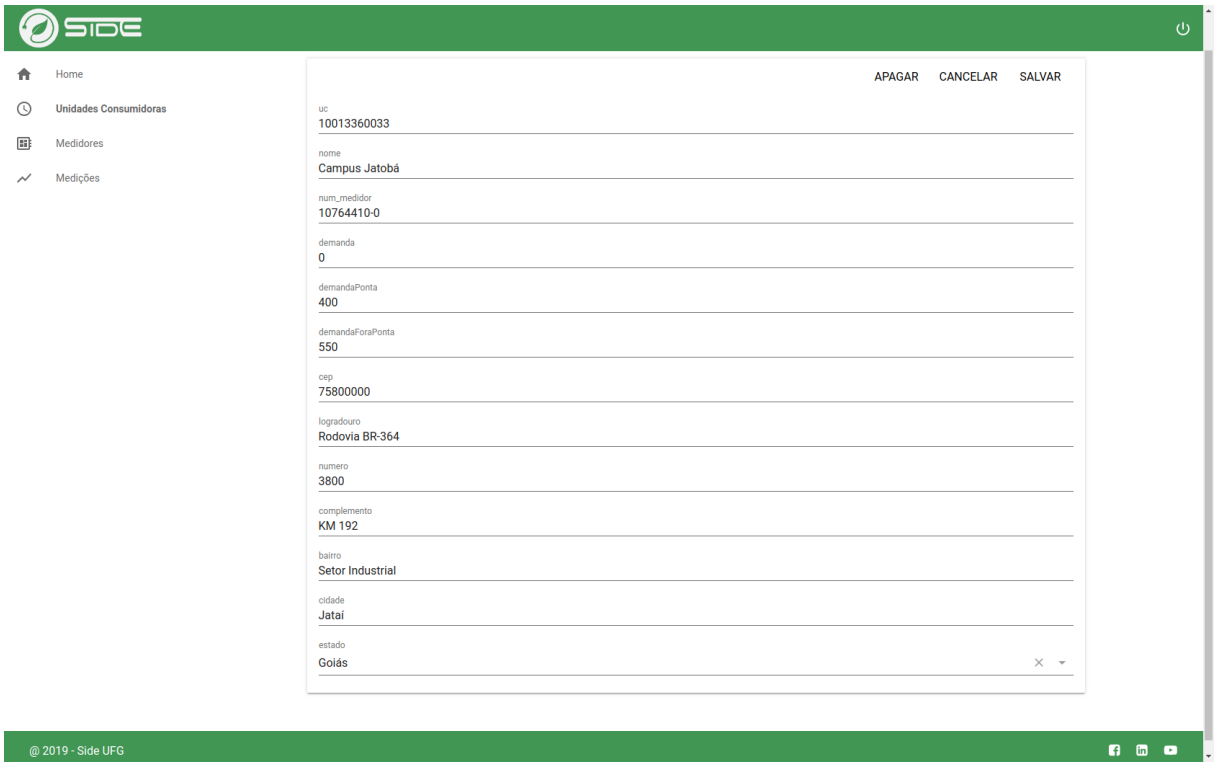


Fonte: Próprio Autor

Ao criar uma Unidade Consumidora, através do ícone superior da tela ou editar uma através do ícone de edição do item, teremos acesso a tela de edição, onde deverão ser preenchidos os dados e após o preenchimento poderão ser salvos. Além disso é possível na tela de edição de uma unidade consumidora, apagar aquela unidade, onde a nível de banco serão desassociados todos os medidores CCK vinculados àquela unidade.

Na figura 11 temos uma imagem da tela de edição de unidade consumidora:

Figura 11 – Tela de Edição das Unidades Consumidoras



Fonte: Próprio Autor

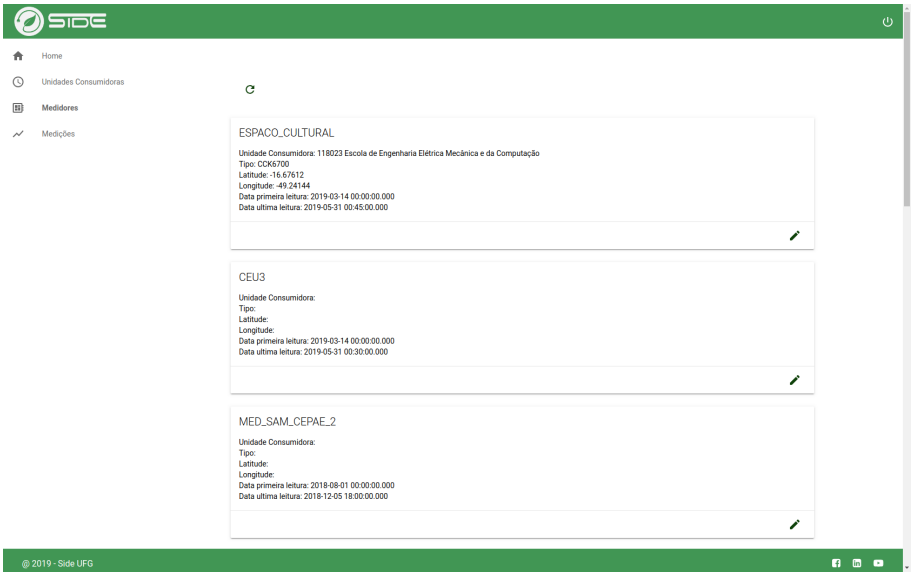
9.3.4 Cadastro de Medidores

No menu de Medidores podemos visualizar os medidores da Universidade cadastrados na base através do *Side Synchronizer*, descriminados pela denominação do mesmo.

Na listagem de cada medidor, pode-se ver a unidade consumidora associada à ele, o tipo de Medidor, suas coordenadas geográficas, bem como suas datas de início e fim de leitura. Abaixo temos disponível a ação de edição para aquele item. Para a tela de medidores não é permitido cadastras ou remover itens, pois o mesmo só pode ser feito pelo dispositivo mestre da rede *Modbus*.

Na figura 12 temos uma imagem da tela de listagem de medidores do sistema.

Figura 12 – Tela de Listagem dos Medidores CCK

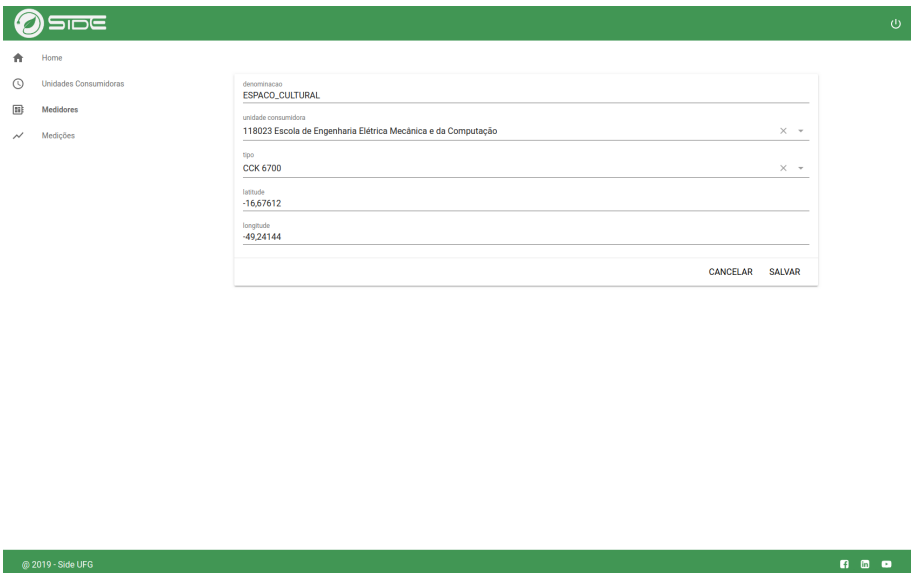


Fonte: Próprio Autor

Ao editar um medidor através do ícone de edição do item, teremos acesso a tela de edição, onde deverão ser preenchidos os dados e após o preenchimento poderão ser salvos. A denominação pode ser alterada pois não interfere na comunicação do *Side Synchronizer* com o mestre da rede *Modbus*, pois a mesma é feita utilizando o id do medidor. Por outro lado não é possível apagar aquele medidor ou criar um novo, essa ação deve ser feita pelo Gerenciador da CCK.

Na figura 13 temos uma imagem da tela de edição de medidor:

Figura 13 – Tela de Edição dos Medidores



Fonte: Próprio Autor

9.3.5 Visualização de Medições

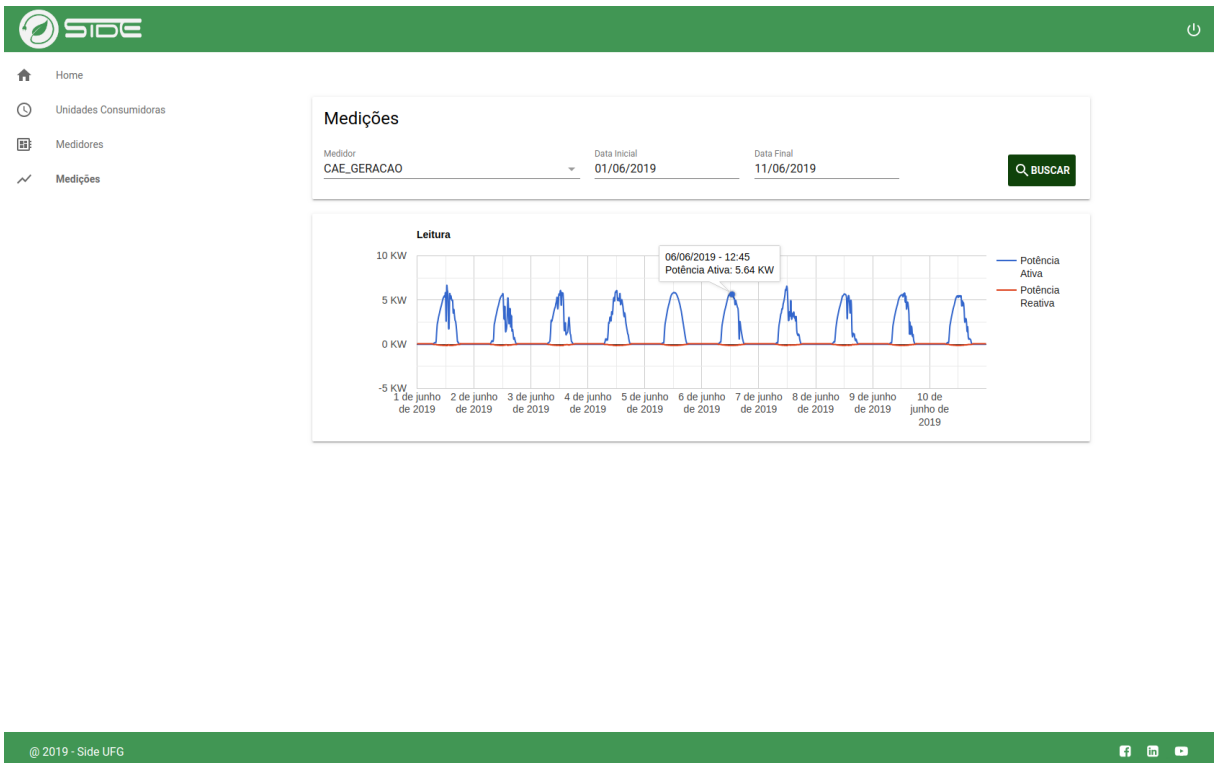
No menu de Medições podemos visualizar o gráfico das medições de um medidor da Universidade cadastrados na base através do *Side Synchronizer*. Além disso as medições podem ser filtradas por data inicial e final.

Ao preencher o filtro da tela selecionando um medidor e suas datas o sistema irá buscar na tabela de medições os resultados limitados pelos parâmetros de busca. Após isso será gerado um gráfico com a curva de potência ativa e reativa distribuídos pelo tempo em um intervalo de 15 minutos.

O gráfico é interativo e permite que o usuário expanda ou comprima os eixos para detalhar melhor as leituras, além disso ao posicionar o ponteiro do mouse sobre um ponto as informações detalhadas daquele ponto serão exibidas.

Na figura 14 temos uma imagem da tela de medições com o gráfico gerado através da seleção de um medidor e um intervalo de data.

Figura 14 – Tela de Gráfico de Medições



Fonte: Próprio Autor

Parte IV

Resultados

10 Obtenção dos Dados dos Sensores

10.1 Comunicação

Devido ao sistema dos sensores ser um sistema proprietário e de código fechado, a comunicação direta com os dispositivos foi inviabilizada, sendo assim a requisição de dados teve que partir para o mestre da rede através de uma API SOAP da própria empresa.

Porém essa API trouxe dados mais limitados, possibilitando apenas consultar as leituras de potências ativas e reativas feitas pelos medidores. Essa informação entretanto já foi de grande valia por se tratar do principal indicar de consumo. As informações de qualidade energética, como tensão por exemplo, não puderam ser obtidas.

O sincronizador desenvolvido para a comunicação com a API da CCK, possibilitou com que os dados que ficavam restritos ao sistema de visualização da empresa pudessem ser salvos na base de dados do SIDE, de maneira ágil e constante.

10.2 Armazenamento

O desenvolvimento do *SIDE Synchronizer* possibilitou um armazenamento de um grande volume de dados. Desde a sua implantação para obtenção das memórias de massa dos sensores, foram gravados mais de 1.200.000 (um milhão de duzentos mil) registros de medição na base de dados dos 70 medidores cadastrados atualmente na rede.

Esses dados serão importantes para as análises não só através do próprio SIDE, mas por agora estarem armazenados em uma estrutura de dados, poderão ser acessados por qualquer sistema de análise de dados como o Kibana[1] ou o PowerBI[2] por exemplo.

[1] Site do Sistema Kibana: <https://www.elastic.co/pt/products/kibana>
[2] Site da Aplicação Microsoft PowerBI: <https://powerbi.microsoft.com/pt-br/>

11 Obtenção dos Dados das Faturas

Assim como a obtenção dos dados dos sensores, os dados históricos das faturas energéticas são dados importantes para análises financeiras, e o desenvolvimento de um sistema que pudesse extrair esses dados e disponibilizá-los para consulta, tornou o acesso às informações mais fácil.

O trabalho de uma extração de um arquivo PDF, se tornaria algo muito custoso para o estudo caso tivesse que ser feito manualmente, e com o desenvolvimento da aplicação *PdfReader*, esse custo pode ser transferido para se dedicar a análise dos dados em si.

A possibilidade de extrair esses dados para um arquivo externo do tipo CSV cria o cenário onde se pode entregar esses dados para análise sem dar acesso à todo o restante das informações contidas na base de dados do sistema, focando a análise apenas no histórico de consumo da Universidade. Porém caso seja necessário, devido ao fato de as mesmas informações estarem na base do SIDE, pode-se também fazer uma análise juntamente com os dados das medições, tanto para efeito de auditoria, quanto para efeito de projeções.

Caso futuramente, a concessionária de energia forneça uma API para consulta aos dados das faturas, o sistema pode ser adequado para fazer essa extração diretamente das requisições feitas ao serviço da fornecedora. Além disso esse sistema pode ser adequado à outras cobranças, como fornecimento de água por exemplo.

12 Resultado da Interface do Usuário

A Aplicação Web SIDE foi implantada em ambiente de produção, hospedada em um servidor conforme figura 15 na Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás (EMC - UFG)[1], possibilitando o cadastro de novos usuários que poderão fazer uso do sistema, acompanhando em tempo real as leituras de consumo e geração de todos os prédios da Universidade.

Figura 15 – Servidor de Aplicação e unidades de armazenamento instalados na EMC.

Fonte: Próprio Autor

O desempenho do sistema permite acesso simultâneo por vários usuários mas caso seja necessário a aplicação pode ser replicada tanto o *frontend* quanto o *backend* de maneira separadas, facilitando a escalabilidade do sistema, através de contêineres Docker[2] por exemplo.

[1] URL de Acesso ao SIDE: <http://200.137.220.157:8080/sideufg-front>
[2] Site da Plataforma Docker: <https://www.docker.com/>

13 Conclusões

O desenvolvimento do presente projeto possibilitou uma análise de como um sistema feito a partir de um estudo, pode melhorar o estudo do cenário energético da Universidade Federal de Goiás. Além disso também permitiu que fosse analisada a infraestrutura de medição entregue pela Enel à UFG, observando as suas limitações e os seus benefícios para estudos como troca de contrato de demanda, troca de lâmpadas, políticas de redução de consumo, entre outras. Alguns pontos importantes puderam ser obtidos do desenvolvimento desse trabalho:

- Multidisciplinaridade: o envolvimento de diversas áreas do conhecimento, como engenharia elétrica, economia, ciências sociais, automação, engenharia de software, entre outras, é de extrema importância para um bom desenvolvimento de um projeto com uma área de atuação tão ampla como este.
- Código Livre: o desenvolvimento de um projeto que irá contemplar vários tipos de análises em estudos futuros deve ter o seu código aberto para visualização e contribuição de outros pesquisadores, permitindo que o seu funcionamento não dependa de uma empresa em específico, como é o caso do sistema proprietário da CCK.
- Melhoramento da estrutura: um modelo melhor de estrutura física da rede de medição pode ser estudado, de forma a viabilizar uma comunicação mais aberta entre os sensores e os inúmeros meios de interface que poderão existir caso esses dados sejam mais livres. Para o desenvolvimento deste projeto foi necessário inicialmente criar uma maneira de tornar os dados de medição que eram restritos acessíveis livremente, através de um banco de dados, mas se os sensores não fossem de arquitetura fechada, o mesmo poderia ocorrer sem a necessidade de criação desse sistema.

Os objetivos iniciais de desenvolver três sistemas responsáveis por obter os dados das medições, obter os dados históricos da concessionária através das faturas, armazenar esses dados, criar uma interface para visualização dos dados de medição, e a implantação de todo esse sistema, disponibilizando-o para uso da instituição, foram atingidos.

14 Trabalhos Futuros

Durante o desenvolvimento do sistema proposto inicialmente, surgiram algumas possibilidades de melhorias futuras para o projeto, de maneira a abranger maiores necessidades na pesquisa por soluções econômicas em relação ao consumo energético da Universidade Federal de Goiás.

14.1 Melhoramento da Coleta de Dados

Uma melhor maneira de acesso aos dados dos medidores irá ocasionar na coleta de maiores variáveis de medição, possibilitando assim melhores análises desses dados. Atualmente a API do sistema da CCK disponibiliza apenas a coleta dos dados de potência, enquanto que os dados de tensão, corrente, fator de potência, entre outros, são visíveis apenas no sistema proprietário da empresa. Com uma abertura desse sistema, ou até mesmo o desenvolvimento de outro modo de obtenção dos dados dos medidores, essa limitação seria vencida.

Outra alternativa seria o uso de outro dispositivo, ou até mesmo o desenvolvimento de um dispositivo próprio da Universidade. Porém esta é uma alternativa que ocasiona um custo mais elevado em comparação com uma nova solução de software.

14.2 Melhoramento da Estrutura de Dados

O desenvolvimento da estrutura de dados foi menor devido ao escopo do projeto, porém esta estrutura pode abranger outras entidades, como por exemplo inventário de equipamentos, bem como as entidades existentes podem ser expandidas com outras informações, por exemplo número de patrimônio endereço IP, entre outras.

Essas modificações, ao serem aplicadas no banco podem ser acessadas por qualquer sistema que faça uso do mesmo, uma vez que o banco de dados será a estrutura central de armazenamento das informações da rede de medição da Universidade.

14.3 Desenvolvimento de Aplicação Mobile para acesso aos dados

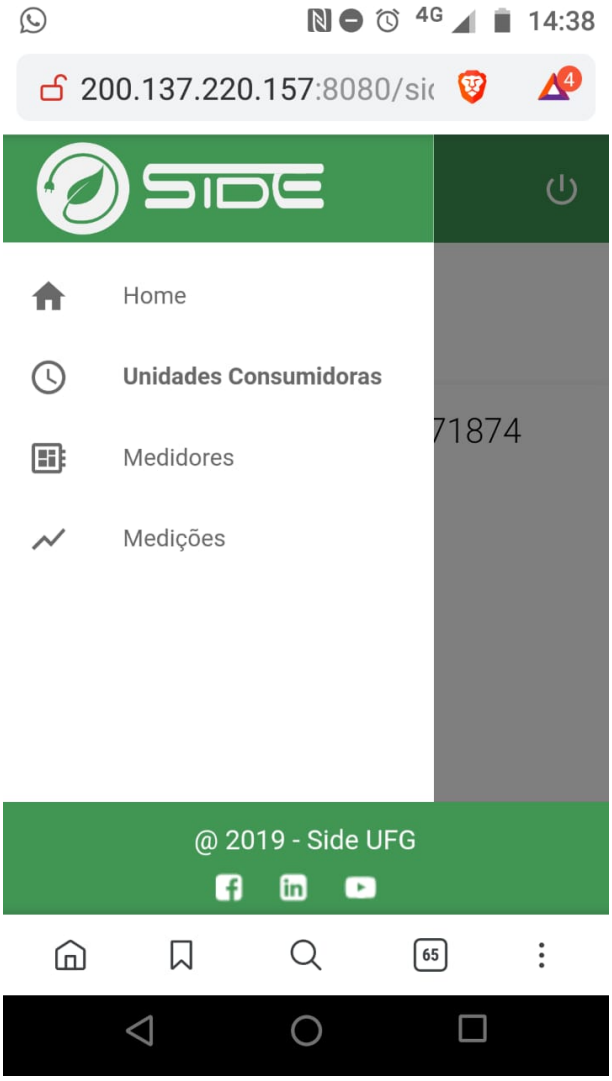
Apesar do SIDE ser uma aplicação web responsiva, o desenvolvimento de uma aplicação *mobile* tornaria a experiência de consumo dos dados muito mais adequada à realidade atual, uma vez que o uso de dispositivos móveis é basicamente uma unanimidade visto que o Brasil é hoje o quinto país no uso de dispositivos móveis[1].

Figura 16 – Exemplo Tela Responsiva SIDE

Fonte: Próprio Autor

[1] *The State of Mobile in 2019*: <https://www.appannie.com/en/insights/market-data/the-state-of-mobile-2019/>

Figura 17 – Menu Mobile SIDE



Fonte: Próprio Autor

Devido ao uso de um *framework* de desenvolvimento, a parte de modelo de negócios do sistema ficou separa da parte visual, fazendo com que assim o desenvolvimento de uma aplicação *mobile* possa fazer uso do mesmo *backend* através de requisições REST.

Referências

ANEEL, A. N. de E. E. Atlas de energia elétrica do brasil - 3ª ediÇÃo. 2008. Disponível em: <http://www2.aneel.gov.br/arquivos/pdf/atlas_par1_cap1.pdf>. Citado na página 35.

ARAUJO, I. Vantagens do controle de versão no desenvolvimento ágil. 2011. Disponível em: <http://blog.myscrumhalf.com/2011/10/vantagens-do-controle-de-versao-no-desenvolvimento-agil/>. Citado na página 42.

BARBOSA, L. F. F. M. Geração de energia renovável em residências: aplicação de tecnologias existentes. Universidade Estadual Paulista (UNESP), 2013. Citado na página 36.

BECK, K. et al. Manifesto for agile software development. 2001. Citado 3 vezes nas páginas 30, 43 e 44.

BEZERRA, E. *Princípios de Análise e Projeto de Sistema com UML*. [S.l.]: Elsevier Brasil, 2016. v. 3. Citado na página 41.

DATE, C. J. *Introdução a sistemas de bancos de dados*. [S.l.]: Elsevier Brasil, 2004. Citado na página 31.

DUTERTRE, B. Formal modeling and analysis of the modbus protocol. In: SPRINGER. *International Conference on Critical Infrastructure Protection*. [S.l.], 2007. p. 189–204. Citado na página 37.

GALDINO, M. A. et al. O contexto das energias renováveis no brasil. *Revista da DIRENG*, p. 17–25, 2000. Citado na página 36.

GAMMA, E. et al. *Padrões de Projeto–Soluções Reutilizáveis de Software Orientado a Objetos*. [S.l.]: Bookman–Porto Alegre, 2004. Citado na página 44.

KORTH, H. F. *Sistema de banco de dados*. [S.l.]: Makron Books/McGraw-Hill, 1994. Citado na página 31.

LTDA., C. A. Cck6700e. 2019. Acessado em 01/06/2019. Disponível em: <http://www.cck.com.br/produtos/produto.php?nmprod=CCK6700E>. Citado na página 48.

MASON, M. *Pragmatic Version Control Using Subversion*. [S.l.]: The Pragmatic Programmers LLC, 2006. Citado na página 42.

MODICON, I. Modicon modbus protocol reference guide. *North Andover, Massachusetts*, p. 28–29, 1996. Citado na página 30.

MURTA, L. G. P.; WERNER, C. M. L. *Gerência de Configuração no Desenvolvimento baseado em Componentes*. [S.l.]: UFRJ, 2006. Citado na página 42.

PRESSMAN, R.; MAXIM, B. *Engenharia de Software-8ª Edição*. [S.l.]: McGraw Hill Brasil, 2016. Citado 3 vezes nas páginas 41, 43 e 44.

SOMMERVILLE, I. Software engineering 9th edition. *ISBN-10*, v. 137035152, 2011. Citado na página 43.

WAZLAWICK, R. *Engenharia de software: conceitos e práticas*. [S.l.]: Elsevier Brasil, 2013. v. 1. Citado na página 41.

Apêndices

APÊNDICE A – Desenvolvimento de Sistema para gerenciar e auxiliar na tomada de decisão envolvendo consumo de energia elétrica

Trabalho apresentado pelo aluno Rafael Ratacheski de Sousa Raulino na disciplina de PROJETO DE FINAL DE CURSO 1 do Curso de Engenharia de Computação, Goiânia, Julho, 2018.

Link para a apresentação no Slideshare: <https://www.slideshare.net/ratacheski/apresentao-sideufg>.

APÊNDICE B – Modelagem de Dados do SIDE

Modelo desenvolvido pelo aluno Rafael Ratacheski de Sousa Raulino através do *software DbSchema*, e disponibilizado no servidor Debian de aplicação e banco de dados da Universidade Federal de Goiás, disponível para acesso através da url abaixo.

Link para acesso ao modelo de dados:

<http://200.137.220.157:8080/sideufg-datamodel/>.

Anexos

ANEXO A – Documentações CUBA Framework v6.10

- Documentação com a descrição de uso do *framework*, apresentando suas classes e métodos de maneira detalhada.

  Link para a documentação:<https://doc.cuba-platform.com/manual-6.10/>.

- Documentação com a descrição de uso da interface de usuário em polymer 2 do *framework*, apresentando os modos de uso das entidades e dos serviços do *backend* pelo *frontend*.

  Link para a documentação:<https://doc.cuba-platform.com/polymer-6.10/>.

ANEXO B – Documentações Biblioteca Google Polymer v2.0

Documentação com a descrição de uso da biblioteca, apresentando suas classes e métodos de maneira detalhada.

Link para a documentação: <https://polymer-library.polymer-project.org/2.0/docs/devguide/feature-overview>.