UNIVERSIDADE DO VALE DO SAPUCAÍ
GUILHERME SANCHES PEREIRA
JÉSSICA ADRIELE DO NASCIMENTO

GERENCIADOR DE CONSUMO DE ENERGIA ELÉTRICA RESIDENCIAL

POUSO ALEGRE, MG
2016

UNIVERSIDADE DO VALE DO SAPUCAÍ
GUILHERME SANCHES PEREIRA
JÉSSICA ADRIELE DO NASCIMENTO

GERENCIADOR DE CONSUMO DE ENERGIA ELÉTRICA RESIDENCIAL

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade do Vale do Sapucaí como requisito parcial para obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof.º Artur Luis Ribas Barbosa

POUSO ALEGRE, MG
2016

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE CÓDIGOS

SUMÁRIO

1 QUADRO METODOLÓGICO

Neste capítulo serão abordados e descritos os procedimentos definidos e utilizados para conduzir a pesquisa. Serão apresentados o tipo de pesquisa, seu contexto, bem como os instrumentos, sua diagramação e os procedimentos para o seu desenvolvimento.

1.1 Tipo de pesquisa

Para Tartuce (2006), método (do grego *methodos*; met'hodos significa, literalmente, "caminho para chegar a um fim") é o caminho em direção a um objetivo; já a metodologia é o estudo do método, ou seja, é o corpo de regras e procedimentos estabelecidos para realizar uma pesquisa.

Gil (2007, p. 17) qualifica pesquisa como:

> [...] procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa desenvolve-se por um processo constituído de várias fases, desde a formulação do problema até a apresentação e discussão dos resultados.

De forma objetiva, a pesquisa é o meio utilizado para buscar respostas aos mais diversos tipos de indagações, ela envolve a abertura de horizontes e a apresentação de diretrizes fundamentais, que podem contribuir para o desenvolvimento do conhecimento. A pesquisa é realizada quando se tem um problema e não se tem informações suficientes para solucioná-lo.

Para o desenvolvimento deste projeto foi escolhida a pesquisa aplicada, que segundo Moresi (2003), procura gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos, envolvendo verdades e interesses locais.

Já de acordo com Marconi e Lakatos (2009, p. 6), "Pesquisa aplicada caracteriza-se por seu interesse prático, isto é, que os resultados sejam aplicados ou utilizados, imediatamente, na solução de problemas que ocorrem na realidade".

Tendo sido empregados estes conceitos, esta pesquisa configurou-se como aplicada, pois agregou maiores vantagens e demonstrou melhores resultados ao desenvolvimento da aplicação *web*, graças à elaboração do levantamento das informações, que possibilitaram o gerenciamento e controle do consumo de energia elétrica residencial.

1.2 Contexto de pesquisa

A energia elétrica é um insumo essencial à sociedade, indispensável ao desenvolvimento socioeconômico das nações. No Brasil, a principal fonte de geração é a hidrelétrica (água corrente dos rios), que responde por 62% da capacidade instalada em operação no país, seguida das termelétricas (gás natural, carvão mineral, combustíveis fósseis, biomassa e nuclear), com 28%. O restante é proveniente de usinas eólicas (energia dos ventos) e importação da energia de outros países. (ANEEL, 2016).

De acordo com os dados apontados pela EPE – Empresa de Pesquisa Energética, o consumo de energia elétrica no Brasil no ano de 2015 foi de 464.544.535 MWh, sendo 51.809.758 MWh do estado de Minas Gerais. Até o primeiro semestre de 2016 os dados registraram, 231.502.008 MWh, destes 12.668.620 MWh são do estado de Minas Gerais. Segundo dados do IBGE - Instituto Brasileiro de Geografia e Estatística, coletados no dia 17 de agosto de 2016, o Brasil tinha uma população de 206.307.710 habitantes, sendo 21.013.919 habitantes do estado de Minas Gerais.

Este número elevado da população fez com que o consumo de energia elétrica aumentasse e as geradoras de energia tiveram maiores gastos que foram repassados aos consumidores pelas concessionárias responsáveis. Outros fatores impactantes no custo da energia elétrica foram: o uso das usinas térmicas (mais caras), que começou em 2013, para compensar a escassez de água nos reservatórios das hidrelétricas, a falta de contratos de longo prazo - que forçou as empresas a buscarem energia no mercado livre - e assinatura de novos contratos de longo prazo já com preços mais altos.

Diante dos constantes aumentos nas tarifas de energia elétrica tornou-se de suma importância que o consumidor tivesse acesso instantâneo do valor de sua respectiva conta ao longo do mês. Munido dessas informações em tempo real, ele não levaria susto ao receber a conta no fim do mês, pois iria prever o valor conforme a evolução do consumo.

Monitorando o consumo diariamente, o consumidor também poderia iniciar um racionamento em qualquer tempo se assim achar conveniente para reduzir o valor da próxima conta, otimizando então o consumo na residência monitorada. Além dessas vantagens, problemas de medição como defeitos no medidor, equipamentos consumindo energia exageradamente e desperdícios poderiam ser detectados.

Pensando nisto, este projeto teve como objetivo o desenvolvimento de uma aplicação *web*, capaz de mostrar ao cliente final informações mais próximas dos valores reais do consumo

de energia elétrico diário e os custos acumulados ao longo do mês através de relatórios gerados em tempo real de forma prática e compreensível e facilmente acessíveis no mercado.

1.3 Instrumentos

Os instrumentos são conjuntos de ferramentas usadas para a coleta de dados. Como afirma Marconi e Lakatos (2009), eles abrangem: questionários, entrevistas e formulários.

Por questionário entende-se um conjunto de questões que são respondidas por escrito pelo pesquisado. Entrevista, por sua vez, pode ser entendida como a técnica que envolve duas pessoas numa situação "face a face" e em que uma delas formula questões e a outra responde. Formulário, por fim, pode ser definido como a técnica de coleta de dados em que o pesquisador formula questões previamente elaboradas e anota as respostas (GIL, 2007).

Outra ferramenta segundo Kioskea (2014), são as reuniões que representam um meio de compartilhamento e partilha, em um determinado grupo de pessoas, com mesmo nível de conhecimento sobre um assunto ou um problema para tomada de decisões de forma coletiva.

Reuniões entre os participantes do projeto foram contínuas, a fim de colher informações sobre o escopo do projeto, análise do progresso do desenvolvimento e validação das atividades concluídas, para se ter gerenciamento e controle dos prazos estabelecidos previamente no cronograma, evitando assim imprevistos ou situações não estabelecidas no mapa de riscos do projeto.

Com intuído de obter análise exploratória e quantitativa do número de pessoas que acreditaram na viabilização deste trabalho, aplicou-se um formulário online com questões diretamente ligadas à aprovação ou não da temática escolhida e informações relevantes sobre o público alvo do produto final.

O formulário iniciou-se com a coleta de informações a respeito da faixa etária do usuário, sua renda e gênero, seguido de questões sobre seu atual consumo de energia, bem como o seu interesse ou não em obtenção de um software para controle e gerenciamento de energia elétrica e quanto pagaria pelo mesmo. O detalhamento das questões contidas neste formulário encontra-se no capítulo de anexo.

Durante o processo de desenvolvimento desta pesquisa foram empregados instrumentos de busca de informações que trouxeram todo embasamento teórico e prático para execução do trabalho proposto. Estas ferramentas foram de enorme relevância no desenvolvimento deste trabalho apresentado, pois coletaram o máximo de informações possíveis no levantamento realizado e nortearam as tomadas de decisões.

1.4 Diagramas

A maioria dos problemas encontrados em sistemas orientados a objetos tem sua origem na construção do modelo, no desenho do sistema. Muitas vezes as empresas e profissionais não dão muita ênfase à essa fase do projeto, e acabam cometendo diversos erros de análise e modelagem. Isso quando há modelagem, pois, profissionais da área sabem que muitas vezes o projeto começa já na fase de codificação (DEVMEDIA, 2016).

1.4.1 Casos de uso

Esse diagrama documenta o que o sistema faz do ponto de vista do usuário. Em outras palavras, ele descreve as principais funcionalidades do sistema e a interação dessas funcionalidades com os usuários do mesmo sistema. Nesse diagrama não há aprofundamentos em detalhes técnicos (DEVMEDIA, 2016).

Ainda segundo Devmedia (2016) este artefato é geralmente derivado da especificação de requisitos, que por sua vez não faz parte da UML. Portanto, pode ser utilizado também para criar o documento de requisitos.

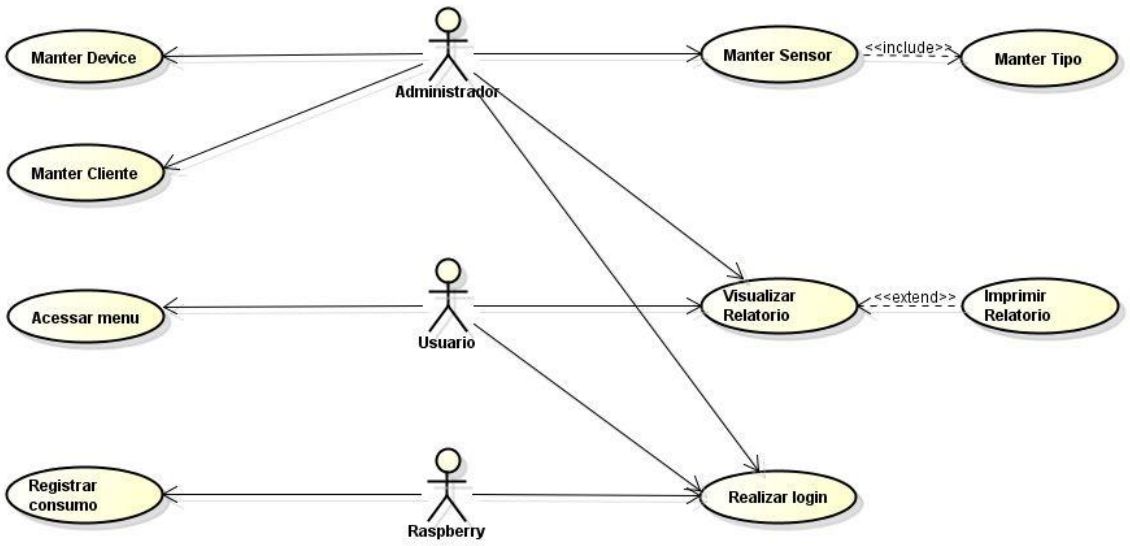
Diagramas de casos de uso são compostos basicamente por quatro partes:

- Cenário: Sequência de eventos que acontecem quando um usuário interage com o sistema.
- Ator: Usuário do sistema, ou melhor, um tipo de usuário.
- Caso de uso: É uma tarefa ou uma funcionalidade realizada pelo ator (usuário)
- Comunicação: é o que liga um ator com um caso de uso

Antes de iniciar o desenvolvimento foi efetuado a diagramação relacionando os 3 tipos de atores existentes (usuário web, Raspberry e administrador) com suas principais atividades desempenhadas no sistema.

A Figura 1 ilustra os casos de uso do projeto juntamente com as relações com cada tipo de usuário.

Figura 1 - Caso de uso



Fonte – Elaborado pelos autores

Como pode ser visto na Figura X, existem três atores responsáveis por atividades, algumas distintas e somente feitas por um deles, e outros comuns entre vários atores. Este fato traz para a fase de desenvolvimento uma técnica de modularizar nossa aplicação web de forma a validar níveis de acesso distintos para cada ator.

1.5 Procedimentos e resultados

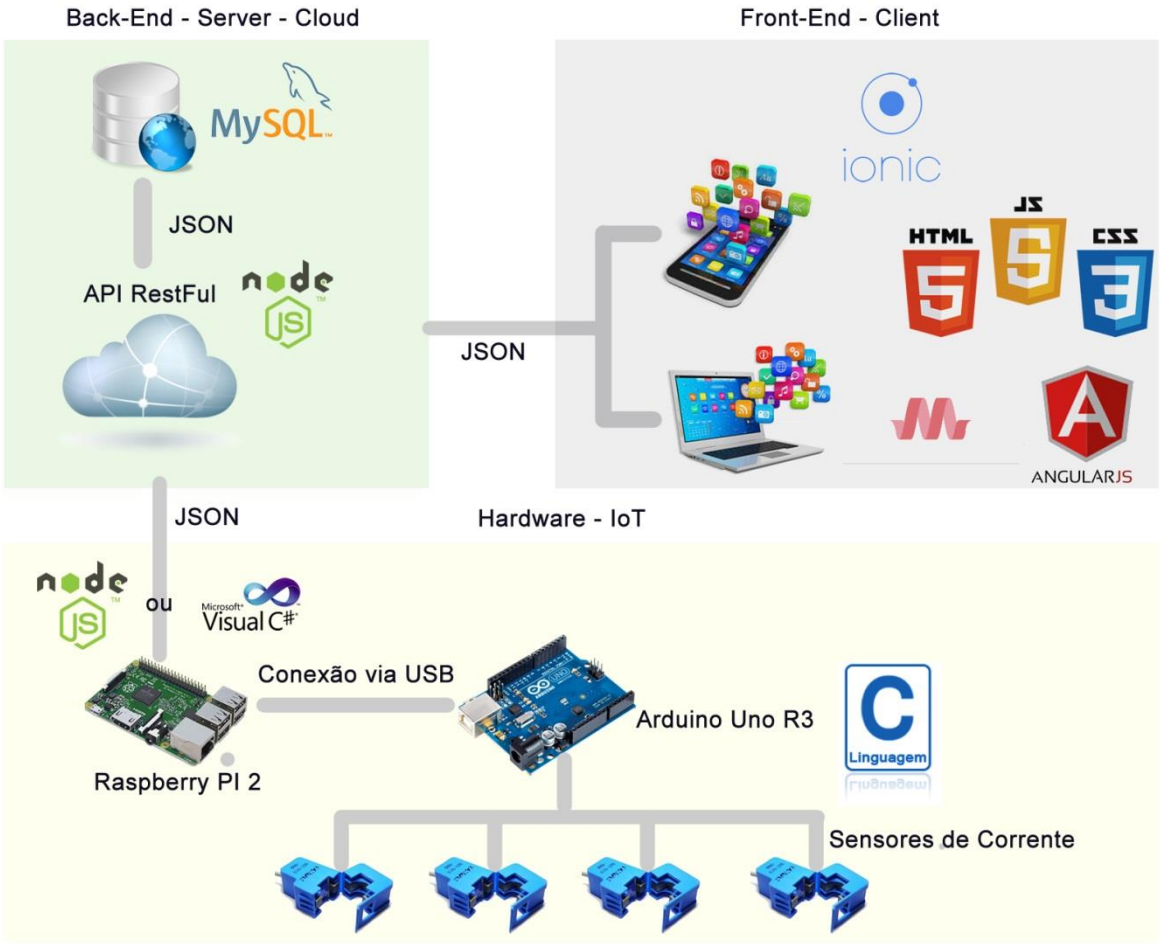
Nesta pesquisa, foi desenvolvida uma aplicação web juntamente com uma arquitetura de hardware para criar uma solução capaz de gerenciar consumo de energia elétrica de uma residência. Para organização no processo de desenvolvimento foi previamente definido um cronograma onde nele continha informações referentes ao tempo estimado para cada atividade. As atividades progrediam conforme era feito o estudo, implementação e análise dos resultados de cada tecnologia empregada no projeto.

1.5.1Modelagem da arquitetura

O primeiro passo tomado para a criação do sistema desta pesquisa, após o levantamento das tecnologias necessárias, foi à modelagem da arquitetura que seria utilizada.

O modelo proposto pode ser observado na Figura 2. Como pode ser visto, os sensores de corrente não invasivo são conectados ao micro controlador Arduíno responsável por receber os dados coletados e enviar para o microcomputador Raspberry.

Figura 2 - Arquitetura final da solução



Fonte – Elaborado pelos autores

A arquitetura desenvolvida contou com total desacoplamento entre as camadas e refletiu em resultados positivos em relação à manutenibilidade do sistema bem como sua facilidade em desenvolvimento de tarefas para a equipe, já que não havia acoplamento que gerasse conflito de atividades ou impedimentos até que um ponto fosse desenvolvido.

Outro ponto referente à arquitetura, está na utilização de uma API para centralizar e disponibilizar serviços tanto para cliente final quanto para o *hardware*. Tal fator evitou replicação de códigos desnecessários e trouxe uma arquitetura mais robusta à solução final.

1.5.2 Configuração de ambiente da aplicação web e API RESTful

Para desenvolvimento desta pesquisa foi necessário configurar um ambiente de desenvolvimento local para que fosse possível elaborar, criar protótipos, desenvolver e testar o produto final. O sistema operacional dos computadores para desenvolvimento é Windows nas versões 8 e 10.

Alguns pré-requisitos foram necessários para que houvesse sucesso no início no desenvolvimento da solução: A primeira ação tomada foi instalação das seguintes ferramentas exibidas no Quadro 1.

Quadro 1 - Listagem de ferramentas

Ferramenta	Versão	Disponível em	Característica
NodeJS	4.5.0	https://nodejs.org/en/download/	Linguagem de servidor
NPM	2.15.8	https://nodejs.org/en/download/	Gerenciador de dependências do NodeJS
GIT for Windows	2.9.3	https://git-scm.com/download/win	Controlador de versão
VSCode	1.4	https://code.visualstudio.com/download	Editor de texto
Arduino IDE	1.6.11	https://www.arduino.cc/en/Main/Software	IDE para upload do código para a Arduíno

Fonte – Elaborado pelos autores

Todas estas ferramentas podem ser encontradas pelos links de download, e após seu download, a sua instalação segue os padrões normais, sem nenhuma alteração, ou seja, basta clicar duas vezes no instalador e seguir os passos padrões sugeridos pelo programa.

Ao término das instalações das ferramentas o ambiente estava pronto para iniciar o desenvolvimento. Não se obteve qualquer erro nas etapas de instalação do ambiente.

1.5.3 Configuração de ambiente da Raspberry

Além da plataforma de desenvolvimento Arduíno, a pesquisa traz a integração com o microcomputador *Raspberry Pi* para coleta, armazenamento temporário e envio dos dados coletados pela Arduíno para a API RESTful onde é tratado e persistido para consulta dos clientes através da interface *web*.

Para que fosse possível iniciar qualquer desenvolvimento neste microcomputador, foi necessário primeiramente realizar a instalação de seu sistema operacional e para este procedimento necessitou-se dos seguintes itens:

- *Raspberry* Pi
- Fonte de alimentação - 5V 650ma
- Cartão SD (mínimo 4GB)
- TV com entrada HDMI ou vídeo composto
- Teclado
- Mouse
- Computador (notebook ou Desktop) (para preparar o cartão SD)

Além dos itens acima listados, necessitou-se dos seguintes softwares listados no Quadro 2.

Quadro 2 - Listagem de ferramentas para Raspberry

Ferramenta	Versão	Disponível em	Característica
SD Formatter	4.0.0	https://www.sdcard.org/downloads/formatter_4/	Formatação de cartões SD
Noobs	1.9.2	http://downloads.raspberrypi.org/noobs	ISO's de sistemas operacionais para Raspberry

Fonte – Elaborado pelos autores

O Raspbian é uma versão de Debian para *Raspberry* que é a mais difundida e a recomendada para todos que estão começando a utiliza-lo. Entretanto há diversos outros

sistemas operacionais compatíveis com este microcomputador (Distribuições do Debian, RaspbianOS, Pidora e até mesmo Windows 10 para *IoT*).

Para realizar sua instalação na *Raspberry*, seguiu-se as seguintes etapas.

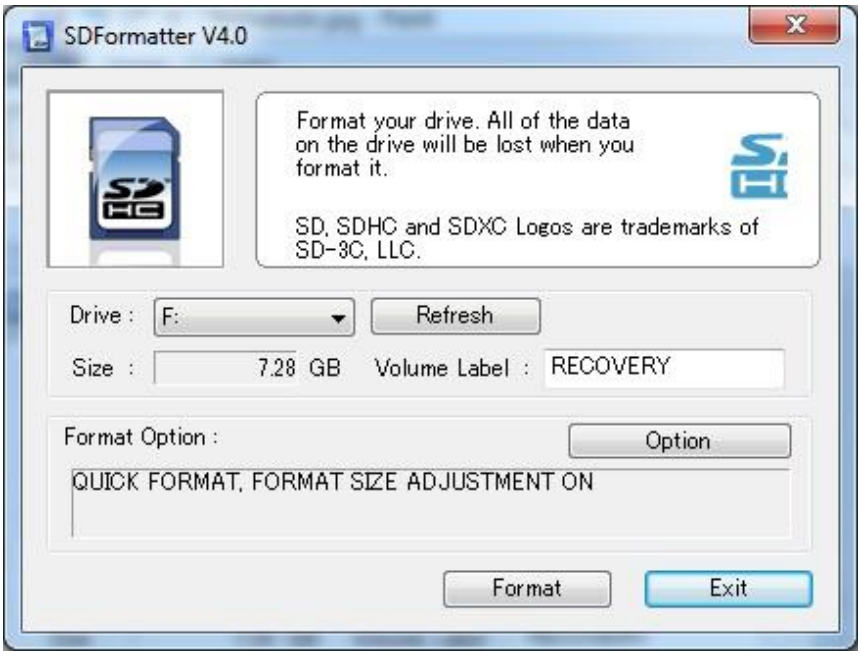
1º - Baixar os arquivos e verificar a integridade deles.

Baixou-se todas as ferramentas citadas acima.

2º - Preparar o cartão SD.

Colocou-se o cartão SD (4GB ou +) no computador e formatou-o utilizando a ferramenta SDFormatter4 (Windows ou MAC). Este procedimento é necessário para que o cartão SD seja capaz de aceitar a ISO do sistema operacional e sirva para a *Raspberry* assim como um HD (*hard disk*) serve para um computador pessoal ou *notebook*. Na Figura 3 mostra a tela do software responsável por este procedimento.

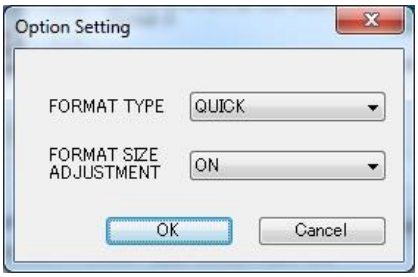
Figura 3 - SDFormatter 4



Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

Instalou-se o programa, abriu-o e foi selecionada a unidade onde estava o cartão SD. Após, clicado em '*Option'* e alterado para '*ON*' a opção '*FORMAT SIZE ADJUSTMENT*' conforme mostra a Figura 4.

Figura 4 - Tela opções SDFormatter



Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

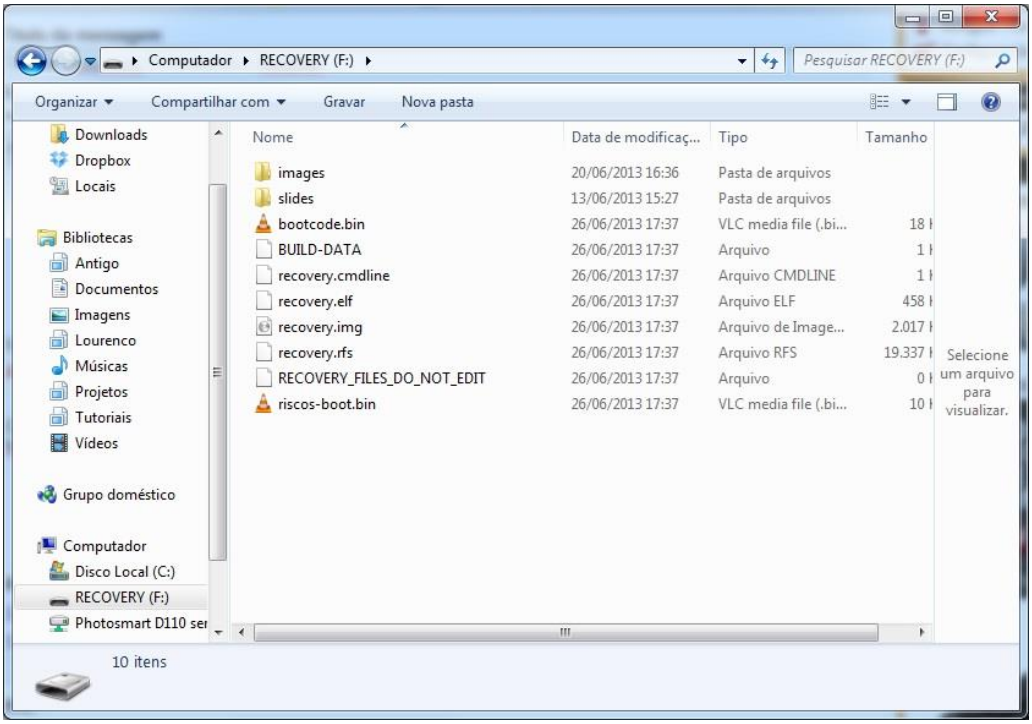
Verifique se selecionou a unidade certa e clique em *Format*.

“ ATENÇÃO: Selecionando a unidade errada você pode apagar os arquivos que não queira. Tenha muito cuidado ao executar esta ação ”.

Com o cartão formatado, utilize um descompactador de arquivos para extrair na raiz do cartão SD, o arquivo NOOBS_v1_9_2.zip.

A raiz do cartão deve ficar seguindo a estrutura da Figura 5:

Figura 5 - Diretório raiz



Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

3º - Ligando o *Raspberry* Pi

Conecte mouse, teclado, cabo HDMI ou vídeo composto do seu monitor e o cartão SD no *Raspberry* e só depois ligue a fonte de alimentação.

Com ela ligada, se você não ver nenhuma imagem no monitor você pode utilizar as teclas 1, 2, 3 e 4 do teclado, para alternar entre diferentes modos de vídeo.

Abaixo segue lista com o que cada tecla faz:

1- Modo Padrão HDMI

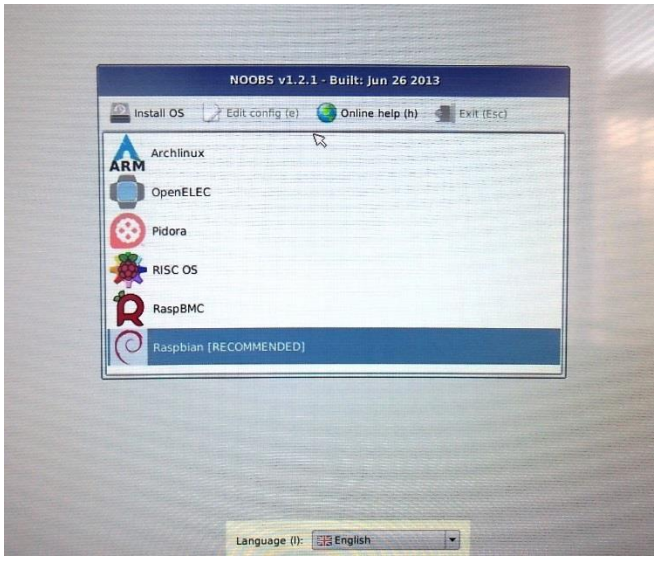
2- HDMI *Safe Mode* - Use este modo se não ver nenhuma imagem no modo 1

3- Modo vídeo composto PAL (saída de vídeo composto)

4- Modo vídeo composto NTSC (saída de vídeo composto)

Após as etapas acima, a Figura 6 exibe a tela que foi exibida.

Figura 6 - Tela de seleção do Sistema Operacional



Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

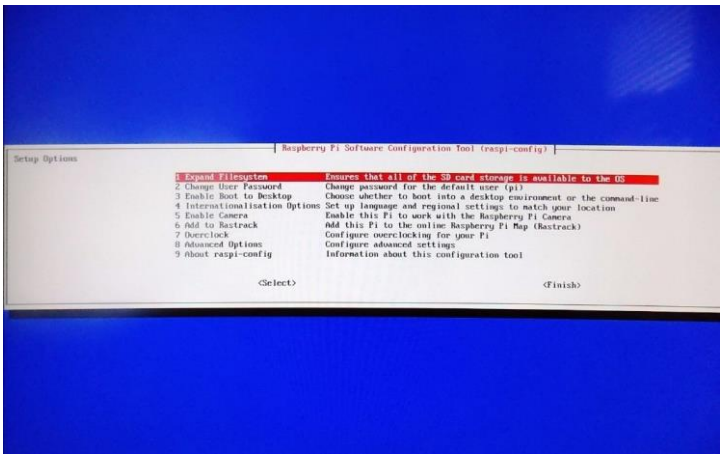
Nesta tela listam vários sistemas operacionais disponíveis para *Raspberry Pi*, neste projeto, foi instalado o Raspbian que está marcado como recomendado.

Clique nele e confirme a instalação. Nesta hora, ele irá particionar e copiar a imagem do Linux o cartão SD.

Cerca de 10 minutos depois, você verá uma tela confirmando que a imagem foi transferida com sucesso.

Ele irá reiniciar com um menu em um fundo azul, como pode ser visto na Figura 7.

Figura 7 - Tela de configuração da Raspberry



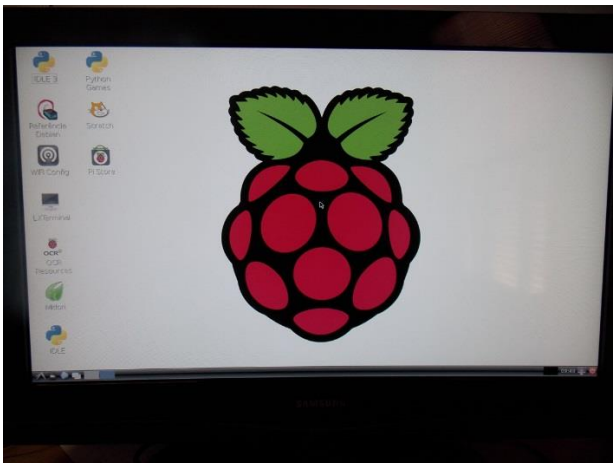
Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

Este é um auxiliar de configuração de sistema e nele existem várias opções para personalizar a instalação. Nesta tela pode-se configurar as opções de linguagem, a matriz do teclado, usuário e senha e diversas outras funções referentes ao sistema. Pode-se navegar utilizando as teclas: Direção, Enter, Esc e Backspace do teclado (nada de mouse nesta etapa).

Ao finalizar as configurações, confirme em Finish (seta para esquerda para chegar até ela).

Ele irá reiniciar e a tela da com a interface gráfica do Raspbian será mostrada, como pode ser visto na Figura 8.

Figura 8 - Tela inicial Raspberry



Fonte – http://labdegaragem.com/profiles/blogs/tutorial-raspberry-pi-instalando-o-raspbian-raspberry-pi-debian

Após a instalação do sistema operacional foi necessário instalar também o NodeJS no microcomputador para desenvolvimento de um servidor que está na *Raspberry* realizando o papel de central de recepção e distribuição de dados. Para este procedimento necessitou-se executar os seguintes comandos.

Quadro 3 - Atualizar dependências

```
$ sudo apt-get update
```

Fonte – Elaborado pelos autores

O Comando do Quadro 3 atualiza todos os pacotes do sistema operacional. Após término precisou-se recuperar a versão do NodeJS mais atualizada do repositório remoto.

Quadro 4 - *Download* Nodejs para *Linux*

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
```

Fonte – Elaborado pelos autores

Após término da atualização do repositório de versões do NodeJS precisou-se executar o comando mostrado no Quadro 4.

Quadro 5 - Instalação Nodejs

```
$ sudo apt-get install -y nodejs
```

Fonte – Elaborado pelos autores

O comando do Quadro 5 executa a instalação do NodeJS na Raspberry.

Para saber se tudo ocorreu como esperado foi executado o comando do Quadro 6 para verificar se realmente foi instalado com sucesso.

Quadro 6 - Verificar versão instalado

```
$ node -v
```

Fonte – Elaborado pelos autores

1.5.4 Instalação do sensor de corrente com Arduíno

O projeto integra *hardware* com *software*, e para este procedimento obter o resultado esperado foi necessário realizar ampla pesquisa sobre os equipamentos utilizados e suas limitações com relação a integração.

A ligação do sensor TC com a plataforma Arduíno seguiu os padrões de ligação entre os componentes e esta ligação deve seguir alguns critérios e cuidados,

Segundo informações do *datasheet*, o sensor de corrente SCT-013-020 (20A) tem na saída uma variação de tensão, e o SCT-013-000 (100A e utilizado no projeto), tem na saída uma variação de corrente.

Assim, no micro controlador Arduíno conseguimos ler quase que diretamente a variação de tensão, porém no de 100A precisamos de um componente adicional: o "*burden resistor*" ("resistor de carga"), para gerar a variação de tensão que precisamos para efetuar a leitura na Arduíno.

Para calcular o resistor de carga, vamos seguir alguns passos, segundo Thomsen (2015):

1 – Determinar a corrente máxima que vamos medir

No nosso caso, é um sensor de 100A, logo vamos determinar esse valor como corrente máxima

2 – Converter a corrente máxima RMS para corrente de pico, multiplicando-a por $\sqrt{2}$

corrente-pico-primaria = corrente-RMS $\times \sqrt{2}$ = 100 A $\times$ 1.414 = 141.4A

3 – Dividir a corrente de pico pelo número de voltas do CT (2000) para determinar a corrente de pico na bobina secundária:

corrente-pico-secundaria = corrente-pico-primaria / numero-voltas = 141.4 A / 2000 = 0.0707A

4 – Para melhorar a resolução da medição, a voltagem através do resistor de carga no pico de corrente deve ser igual a metade da tensão de referência do Arduíno (AREF/2). Como a tensão de referência no Arduíno é de 5V, teremos:

resistor-carga-ideal = (AREF/2) / corrente-pico-secundaria = 2.5 V / 0.0707 A = 35.4 Ω

Resumindo o cálculo anterior:

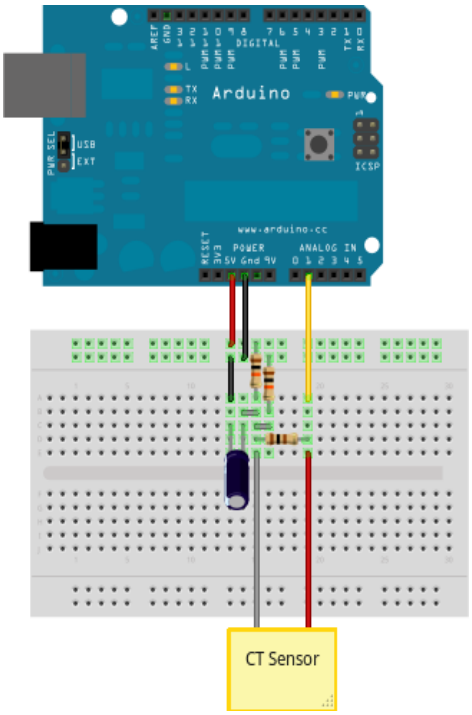
Resistor de carga (ohms) = (AREF * CT-VOLTAS) / (2√2 * corrente-máxima-pico), onde CT-VOLTAS refere-se ao número de voltas da bobina interna do sensor (este dado se encontra no *datasheet* do mesmo).

Após obter os dados necessários para montagem do circuito, foi realizada a prototipação através de uma *protoboard*. Os componentes necessários, foram:

- Sensor de Corrente SCT-013-000
- 2 Resistores de 10K
- 1 Resistor de 33 Ω (para o resistor de carga)
- 1 Capacitor 10 µF

A Figura 9 ilustra o circuito construído:

Figura 9 - Integração sensor TC com Arduíno



Fonte – http://blog.filipeflop.com/arduino/medidor-de-energia-eletrica-com-arduino.html

O próximo passo foi a abertura do programa ArduinoIDE já instalado na preparação do ambiente e a criação do seguinte código exibido no Código 1:

Código 1 - Configuração e leitura de corrente com Arduíno

```
1   //Programa : Medidor de corrente com Arduino e SCT-013 100A
2   //Autor : Guilherme Sanches
3
4   //Baseado no programa exemplo da biblioteca EmonLib
5
6   //Carrega as bibliotecas
7   #include "EmonLib.h"
8
9   EnergyMonitor emon1;
10
11  //Tensao da rede eletrica
12  int rede = 127;
13
14  //Pino do sensor SCT
15  int pino_sct = A1;
16
17  void setup()
18  {
19    Serial.begin(9600);
20    //Pino, calibracao - Cur Const= Ratio/BurdenR. 2000/33 = 60
21    emon1.current(pino_sct, 60);
22  }
23
24  void loop()
25  {
26    //Calcula a corrente
27    double Irms = emon1.calcIrms(1480);
28    //Mostra o valor da corrente no serial monitor
29    Serial.print("Corrente : ");
30    Serial.print(Irms); // Irms
31
32    delay(1000);
33  }
```

Fonte – Elaborado pelos autores

O Código 1 foi comentado para melhor entendimento por outros usuários. Neste exemplo a Arduíno realiza a cada 1 segundo a leitura de corrente do fio inserido por dentro do sensor de corrente. O resultado da leitura é exibido no Serial Monitor da própria IDE da Arduíno. Ao realizar a integração com a Raspberry, a mesma recebe estes dados via conexão

USB. Para realizar o upload deste código para a Arduíno bastou selecionar a opção de *UPLOAD* disponível na IDE.

1.5.5 Prototipação, wireframes e mockups

Boas práticas de programação estão presentem ao redor dos desenvolvedores como uma forma de otimizar o desenvolvimento e facilitar o entendimento de qualquer desenvolvedor ao ler um código de outro devido ao fato de considerar padrões de programação, arquitetura e desenvolvimento seguindo práticas que são consideráveis as melhores para desenvolvimento *web*.

Uma dessas boas práticas é a construção e elaboração de imagens prévias que representam como sistema final deve estar. Estas imagens são chamadas de *wireframes* ou *mockups*, caso elas representam um *designer* de baixa fidelidade do produto final é considerado *wireframes*, caso representam o produto final com média e alta fidelidade, são consideradas *mockups*.

Nesta pesquisa foi elaborado alguns *wireframes* para prototipação de uma aplicação de testes para se analisar os impedimentos e facilidades que poderia haver no desenvolvimento.

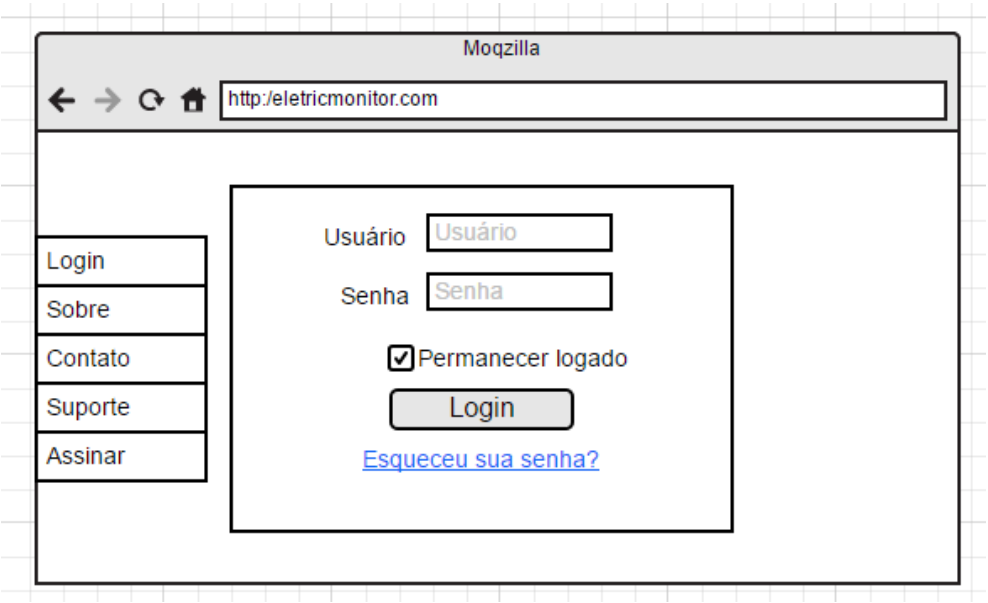
Ao término da prototipação do sistema, foi colhido alguns *feedbacks* positivos como rapidez no desenvolvimento seguindo o *wireframes* e outros negativos como dificuldade em posicionar alguns componentes presentes no *wireframes* devido ao utilizarmos o padrão de *Material Designer* como *framework* de *layout web*.

Abaixo segue 2 *wireframes* criados para criação da aplicação *web*.

Os mesmos foram gerados através de uma aplicação *online* com versão paga e gratuita com recursos restritos (http://mockuos.com).

O primeiro wireframes refere-se à Figura 10 e exibe a tela de login com algumas funcionalidades básicas de acesso necessárias aos usuários. Além destas funcionalidades, percebe-se o layout do menu e da caixa de login de forma centralizada ao centro e no menu centralizado ao centro verticalmente. Estas previsões de posicionamento foram fundamentais na construção e desenvolvimento da tela final da aplicação.

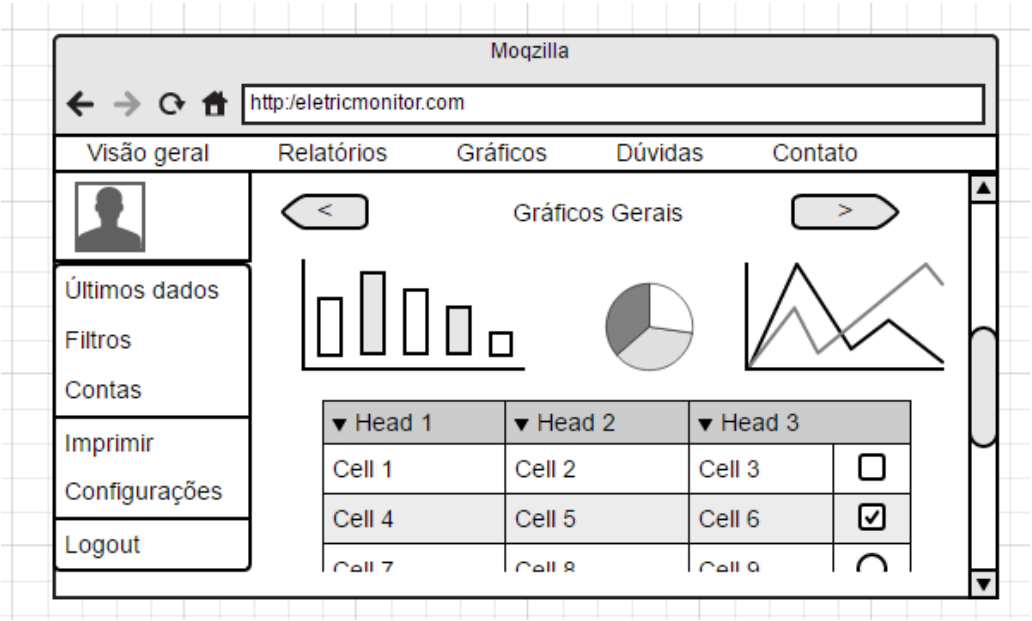
Figura 10 - Wireframe tela de login



Fonte – Elaborado pelos autores

Abaixo , na Figura 11, segue *wireframe* do painel principal do usuário

Figura 11 - Wireframe painel principal da aplicação



Fonte – Elaborado pelos autores

1.5.6 Modelagem do banco de dados

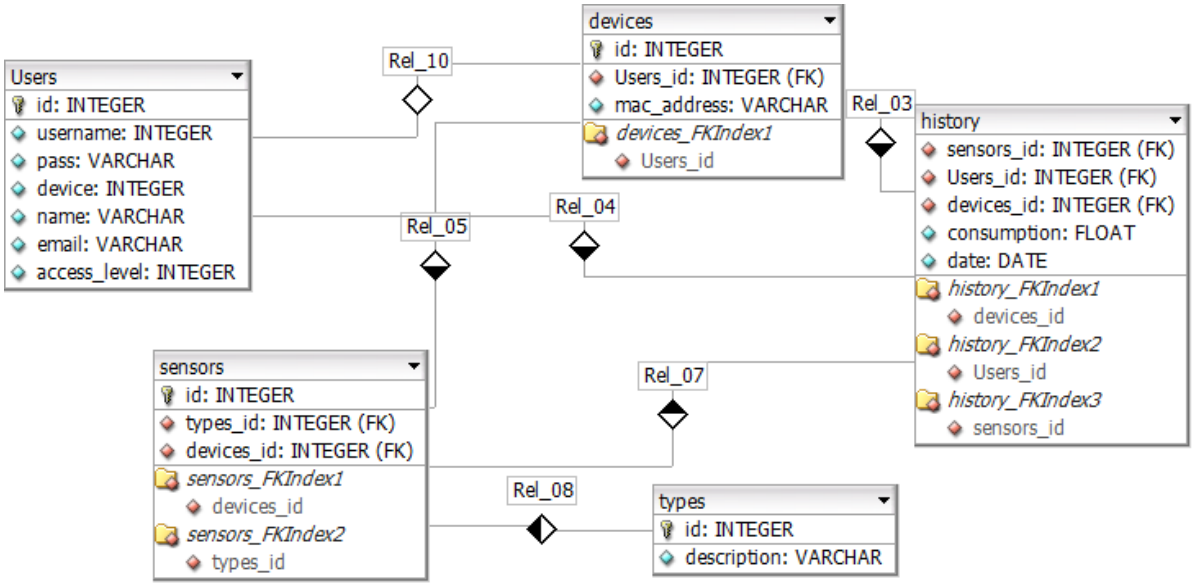
Qualquer aplicação que precise de armazenamento de dados para futuras consultas necessita de uma forma de armazenamento de dados, e quando falamos de aplicação *web* precisamos de um banco de dados para realizar esta persistência dos dados.

Neste cenário utilizamos o MySQL para esta tarefa e precisamos antes de iniciar qualquer desenvolvimento de código, criar um modelo de banco de dados.

O banco de dados da aplicação precisou ser esboçado primeiramente na mão e depois modelado utilizando uma ferramenta adequada para este recurso. Optou-se por utilizar o DBDseigner 4 disponível para download gratuitamente em http://fabforce.net/downloadfile.php.

Segue, na Figura 12, o modelo utilizado no desenvolvimento da aplicação.

Figura 12 - Modelamento do banco de dados



Fonte – Elaborado pelos autores

Este modelo foi seguido e implementado, porém algumas futuras mudanças serão necessárias quando novos recursos e funcionalidades forem elaboradas e liberadas na solução final.

Para fase de desenvolvimento a base de dados foi mantida em computadores locais e ao final exportada para uma instancia *online.*

1.5.7 Desenvolvimento da API RESTful

Com o ambiente local instalado, o ambiente da Raspberry, os *wireframes* e modelo de dados foi iniciado o desenvolvimento da API RESTful que realiza a integração entre *hardware*, aplicação *web* e banco de dados.

Inicialmente foi feita a instalação e configuração do *framework* ExpressJS através da linha de comando exibida no Quadro 7.

Quadro 7 - Instalação ExpressJS

```
$ npm install –g express
```

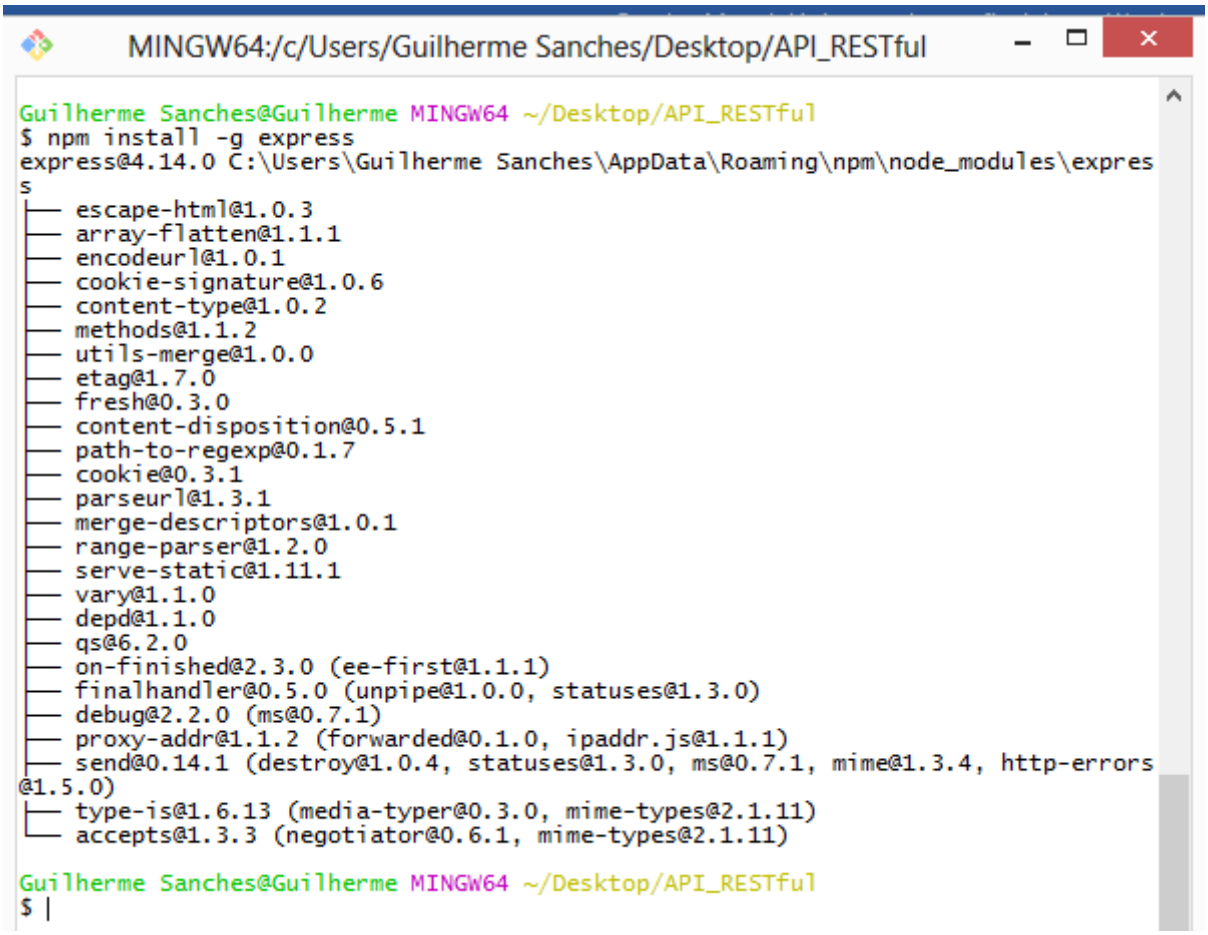
Fonte – Elaborado pelos autores

O ExpressJS foi instalado como um módulo global através do parâmetro "-g", conforme mostrado na Figura 13.

O Express fornece uma camada leve de recursos fundamentais para aplicativos da *web*, sem atrapalhar os recursos do Node.js já existentes.

Através deste *framework* se pôde escrever códigos modularizados, de forma rápida, flexível, utilizando *middlewares* fornecidos pela ferramenta. Sua forma de trabalhar com rotas REST foi de principal importância para gerar resultados satisfatórios ao final do desenvolvimento. E sua documentação repleta de recursos serviu de consulta diversas vezes durante a fase de pesquisa, prototipação e desenvolvimento.

Figura 13 - Resultado instalação ExpressJS



Fonte – Elaborado pelos autores

Depois do *framework* Express, foi necessário iniciar o projeto criando um arquivo de configuração para aplicações NodeJS, o “package.json”. Para criar este arquivo executou-se o comando “npm init” pelo terminal, como exibido no Quadro 8.

Quadro 8 - Criação do arquivo package.json

```
$ npm init
```

Fonte – Elaborado pelos autores

Informou-se os dados que foram pedidos como pode ser visto na Figura 14. Após preenchimento dos dados precisou-se digitar a palavra “*yes*” para confirmar os dados. Este procedimento criou o arquivo “package.json” do projeto com todas as informações referentes a

versionamento, palavras chaves, informações sobre licença de conteúdo, informações sobre os autores, bem como dependências e suas respectivas versões.

Figura 14 - Resultado criação arquivo package.json

Fonte – Elaborado pelos autores

Com projeto iniciado precisou-se começar o desenvolvimento do primeiro arquivo principal do projeto, o “main.js”.

Este arquivo tem a função de iniciar o servidor da aplicação RESTful. Seu conteúdo pode ser visto no Código 2.

Código 2 - Servidor básico ExpressJS

```
1  var express = require('express')
2  var app = express()
3
4  app.get('/', function (req, res) {
5      res.send('Servidor RESTful iniciado na porta 3000 com sucesso')
6  })
7
8  app.listen(3000)
```

Fonte – Elaborado pelos autores

Linha 1: Importa o modulo express.

Linha 2: Cria variável app chamando a execução do servidor express.

Linha 4: Criação de uma rota padrão para testar se o servidor foi iniciado corretamente.

Linha 5: Envia uma mensagem para o console quando a rota “/” for chamada pelo *browser*.

Linha 8: Inicia o servidor na porta 3000.

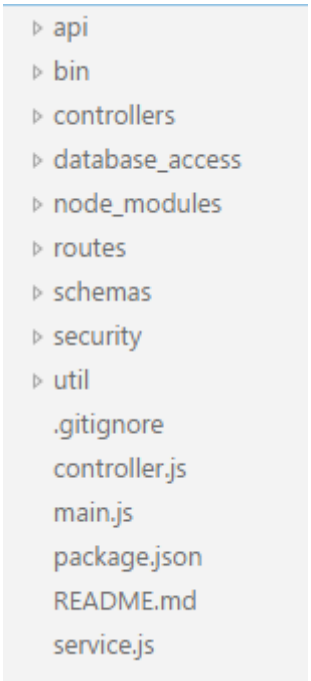
Após todas estas etapas, o ambiente básico para o desenvolvimento da aplicação estava criado. Porém, devido à necessidade da pesquisa, foram adicionados ao sistema os seguintes diretórios:

- ***controllers*:** Diretório responsável pelos arquivos controladores das *Views*;
- ***schemas*:** Diretório responsável pelos arquivos de modelo de dados do banco de dados bem como as funções de validação;
- **util:** Diretório contendo funções úteis em várias partes do código, como conversão de datas por exemplo.
- **security:** Diretório criado para configuração de funções e rotinas de validação, autenticação e segurança da API RESTful.
- **routes:** Diretório criado para centralizar todas as rotas de acesso da API RESTful.

- **database_access:** Contém as funções e configurações necessários para acesso ao banco de dados.
- **api:** Diretório que contém as *features* do sistema, bem como sua lógica de negócios.
- **service.js:** Contém serviços genéricos para várias rotas da API.
- **controler.js:** Contém funções de validação genérica para várias rotas da API.

A estrutura de diretórios é demonstrada na Figura 15.

Figura 15 - Estrutura de diretórios aplicação web



Fonte – Elaborado pelos autores

Na árvore de diretórios, encontram-se também as pastas *bin* e *node_modules*, as quais contém respectivamente o arquivo de inicialização do servidor *web* e os módulos NodeJS necessários para a execução do sistema.

O Arquivo README.md contém informações sobre o projeto para ser visualizado na página do GitHub como auxílio aos usuários interessados em conhecer o projeto e suas características.

Todas as dependências utilizadas no projeto estão sendo exibida na Figura 16.

Figura 16 - Lista de dependências

Fonte – Elaborado pelos autores

1.5.8 Desenvolvimento da aplicação web

A API RESTful possui função de fornecer uma integração entre o hardware e o painel de visualização de dados acessado pelos usuários de forma *online*, portanto foi necessário o desenvolvimento de uma *interface web* para que os usuários das residências possam verificar os dados colhidos pela Arduíno, sensor e Raspberry e gerar dados analíticos a partir dos mesmos.

O primeiro passo nessa etapa de projeto foi a criação do diretório raiz e instalação de um projeto inicial usando o *framework* AngularJS na versão estável, 1.5.8.

Para esta instalação necessitou-se previamente da instalação do gerenciador de dependências NPM que é instalado juntamente com o NodeJS, logo, a sessão de preparação do ambiente da API já havia contemplado este requisito.

Dentro de um terminal de comando aberto dentro do diretório raiz, foi digitado o comando exibido no Quadro 9.

Quadro 9 - Instalação do AngularJS

```
$ npm install angular
```

Fonte – Elaborado pelos autores

Este comando criou uma pasta chamada node_modules contendo todo o código *core* para que pudéssemos inclui-lo em nossa aplicação. E para este procedimento bastou dentro de uma *tag script* no arquivo index.html, incluir o seguinte código.

```
<script src="/node_modules/angular/angular.js"></script>
```

Além do angular houve a necessidade de instalação de um módulo específico para controle de rotas em nossa aplicação, este módulo chama-se angular-route e para sua instalação executamos o seguinte comando exibido no Quadro 11.

Quadro 10 - Instalação angular-route

```
$ npm install angular-route
```

Fonte – Elaborado pelos autores

Este modulo possui função específica de criar e gerenciar nossas rotas e transições entre elas de forma ágil e prática dentro de uma aplicação baseada em AngularJS.

Outro ponto que merece destaque é a instalação da biblioteca jQuery para prover uma agilidade em lidar com recursos do HTML, e ela foi instalada pelo comando listado no Quadro 11.

Quadro 11 - Instalação jQuery

```
$ npm install jquery
```

Fonte – Elaborado pelos autores

E para gerar uma *build* e testes durante desenvolvimento, necessitou-se a instalação de um servidor HTTP local para executar o projeto. Dentro os diversos existentes, deu-se preferência ao servidor http-server, e o mesmo foi instalado pelo comando como segue abaixo no Quadro 12.

Quadro 12 - Instalação servidor HTTP

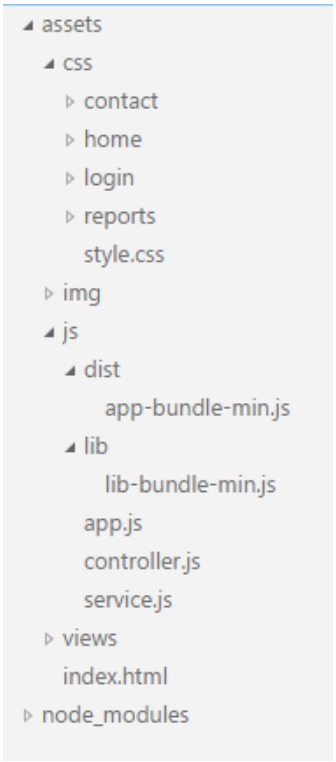
```
$ npm install –g http-server
```

Fonte – Elaborado pelos autores

Como pode ser visto, o modulo foi instalado de forma global, para que o mesmo não seja alocado dentro de nossa pasta node_modules e possa ser usado em qualquer outro projeto que necessite de um servidor http.

Ao final a estrutura do projeto ficou da seguinte forma, como exibido na Figura 17.

Figura 17 - Estrutura de pastas da aplicação



Fonte – Elaborado pelos autores

Após estruturação de pastas, módulos e configuração o desenvolvimento precisou adotar uma prática de layout responsivo seguindo os padrões do Google Material Designer, este padrão é adotado para que haja uma melhor usabilidade do usuário final com o sistema e o mesmo possa ser manutenível de forma padrão e simples, por utilizar conceito de classes de css.

Este recurso foi disponível através da instalação do angular-material que é um modulo que nos traz todos componentes do material designer já para AngularJS.

O comando listado no Quadro 13 fez a instalação do angular-material no projeto:

Quadro 13 - Instalação Angular Material

```
$ npm install angular-material
```

Fonte – Elaborado pelos autores

Para utilizar os componentes, foi preciso acessar o site do angular-material, disponível em: https://material.angularjs.org/. Selecionar quais componentes eram necessários pelo menu lateral esquerdo, como mostra a Figura 18.

Figura 18 - Menu de componentes Angular Material

Fonte – Elaborado pelos autores

Após selecionar o componente necessário, o código do mesmo aparece na tela para ser copiado e incluído ao projeto. Estes recursos trouxeram ganho de tempo durante desenvolvimento por já serem componentes adaptáveis automaticamente à vários tamanhos de telas e também por serem estilizados de forma padrão usando conceitos de Material Designer.

REFERÊNCIAS

ANEEL. ANEEL. **Agência Nacional de Energia Elétrica**, 2016. Disponivel em: <http://www.aneel.gov.br/>. Acesso em: 10 Agosto 2016.

DEVMEDIA. O que é UML e diagramas de casos de uso? **DEVMEDIA**, 2016. Disponivel em: <http://www.devmedia.com.br/o-que-e-uml-e-diagramas-de-caso-de-uso-introducao-pratica-a-uml/23408>. Acesso em: 10 Agosto 2016.

GIL, A. C. **Como elaborar projetos de pesquisa**. 4. ed. São Paulo: Atlas, 2007.

KIOSKEA. Condução de reunião. **CCM**, 2014. Disponivel em: <http://br.ccm.net/contents/gestao-de-projeto-1343593641#579>. Acesso em: 02 Maio 2016.

TARTUCE, T. J. A. **Métodos de pesquisa**. Fortaleza: UNICE -Ensino Superior, 2006.