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 FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Corrente Alternada
ANSI	*American National Standards Institute*
API	*Application Programming Interface*
ARM	*Advanced RISC Machine*
CSS	*Cascading Style Sheet*
GPL	*General Public License*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IDE	*Integrated Development Environment*
I/O	*Input/Output*
IoT	*Internet of Things*
MIT	*Massachusetts Institute of Technology*
REST	*Representational State Transfer*
SVG	*Scalable Vector Graphics*
TC	Transformador de Corrente
USB	*Universal Serial Bus*
VML	*Vector Markup Language*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*

SUMÁRIO

1QUADRO TEÓRICO

Para desenvolvimento do projeto, serão empregadas diversas tecnologias, ferramentas e conceitos que serão discutidos neste capítulo.

1.1 Internet das Coisas

A Internet das Coisas ou IoT, do inglês, *Internet of Things*, é a revolução de uma rede de objetos tecnológicos, incorporados com eletrônica, *software*, sensores e conectividade com a internet, que permitem coletar e trocar dados e informações entre si e entre outros dispositivos conectados à internet, como computadores, *notebooks*, *smartphones* e *tablets* (SILVEIRA, 2016).

O termo *Internet of Things* (Internet das Coisas) foi dito pela primeira vez, em 1999, por Kevin Ashton, cofundador do Auto-ID Center do *Massachusetts Institute of Technology* (MIT). Em artigo, Ashton (2009) alegou que a ideia original do termo Internet das Coisas previa a conexão entre todos os objetos físicos à internet, com capacidade de absorver informações por meio de identificação por radiofrequência e tecnologias de sensoriamento - as quais os permitiriam observar, identificar e compreender o mundo independentemente das pessoas e suas limitações de tempo, atenção e precisão. Porém, como a internet atualmente está presente em quase todos os lugares, ela se torna um meio mais eficiente para a transmissão e coleta desses dados.

A internet hoje, conta com quase três bilhões de usuários conectados segundo "*Global Internet Report*" (KENDE, 2014). Segundo previsões de Stamford (2013), em 2020 o número de dispositivos conectados e interligados será de 26 bilhões. Algumas previsões mais otimistas, segundo a CISCO, prevê 50 bilhões de objetos conectados no mesmo período, gerando uma movimentação de mercado de US$ 14,4 trilhões até 2022 (EVANS, 2011).

Portanto segundo Lacerda e Lima Marques, (2015, p. 161):

> As inovações que surgem no âmbito da Internet das Coisas ampliam o potencial humano em diversas áreas - tais como planejamento urbano (cidades, edifícios e trânsitos inteligentes), meio ambiente (energia, água), indústria, comércio, turismo, educação, saúde, trabalho, segurança, programas sociais, governo - com soluções capazes de promover desenvolvimento econômico, sustentabilidade e qualidade de vida.

Tais fatos trazem para sociedade atual, uma demanda de pessoas, órgãos e empresas privadas, capazes de desenvolverem e programarem recursos de internet das coisas como forma de gerarem lucro financeiro e se inserirem em um mercado com ampla expansão. Este conceito de uma nova geração de conectividade, alinhado ao uso de novas tecnologias e ferramentas disponíveis no mercado, justificado pelo atual estado de crescimento tecnológico torna-se viável e eficaz o uso de recursos de *IoT* neste projeto.

Neste sentido, foi empregado neste trabalho o conceito de *IoT*, para fazer com que os dispositivos interligados pudessem fazer a medição do consumo de energia elétrica de uma residência e exibam estes dados na aplicação web para os usuários terem um maior controle de seus gastos.

1.2 Microcomputador, sensores e prototipação

O presente trabalho teve uma integração entre *software* e *hardware* resultando em uma solução composta de um microcomputador, uma plataforma de prototipação juntamente com sensores conectados a ela. Adiante está discriminado cada um desses componentes.

1.2.1 Raspberry Pi

"A *Raspberry Pi* é uma máquina completa, com considerável poder de processamento, em uma placa de circuito impresso menor do que um cartão de crédito" (UPTON e HALFACREE, 2013, p. 26).

Foi desenvolvida, em 2006, por Eben Upton, Rob Mullins, Jack Lang e Alan Mycroft, com base na Universidade de Laboratório de Informática de Cambridge e na empresa *Atmel ATmega*, que desenvolve placas com processadores ARM. Eben Upton e sua equipe, em 2009, oficialmente estabeleceram a *Raspberry Pi Foundation*, fundação de caridade educacional, com base no Reino Unido, que tem como meta, permitir que as pessoas de todas as idades possam aprender a programar e entender como funcionam os computadores. (RASPBERRY PI, 2014).

O coração do dispositivo *Raspberry Pi* é o processador multimídia *Broadcom* BCM2835, que tem a maioria dos componentes do sistema montado em um único e é baseado na arquitetura de conjunto de instruções ARM, desenvolvida pela *Acorn Computers*, no final dos anos 80, sendo ele, capaz de operar apenas com alimentação de energia de 1A e 5V, fornecida pela sua porta micro-USB. O baixo consumo do chip é traduzido, diretamente, em pouco desperdício de energia, mesmo durante tarefas complexas de processamento. Outra

diferença importante entre o *Raspberry Pi* e seu desktop, sem falarmos no tamanho e no preço, é o sistema operacional – Linux, que possibilita fazer download de todo o código-fonte referente ao sistema operacional e realizar quaisquer alterações necessárias. (UPTON e HALFACREE, 2013).

Segundo Upton e Halfacree (2013), existem dois modelos da placa *Raspberry Pi*: Modelo A e Modelo B. A diferença entre eles é que o modelo B possui uma placa Ethernet e duas portas USBs, enquanto a modelo A, contém apenas uma porta USB e nenhuma Ethernet.

Nesse projeto foi utilizado o modelo *Raspberry Pi 2 model B+*, devido sua melhor configuração de *hardware* e aplicabilidade do projeto.

A Figura 1 apresenta o microprocessador *Raspberry Pi 2 Model B+*:

Figura 1 - *Raspberry Pi* 2

Fonte: https://www.raspberrypi.org/wp-content/uploads/2015/01/Pi2ModB1GB_-comp.jpeg

A *Raspberry Pi* recebe os dados coletados pela Arduino, apresentada na subseção 1.2.2 por meio dos sensores eletrônicos de corrente TC, que serão apresentados na subseção 1.2.3 e envia as informações para o banco de dados *online*.

1.2.2 Arduino

Arduino é uma plataforma de desenvolvimento e prototipagem *open-source*[1] que, segundo Evans, Noble e Hochenbaum (2013, p.25), "a Arduino teve início no *Interaction Design Institute* de Ivrea, na Itália, em 2005".

[1] OPEN-SOURCE – Software de código aberto que qualquer um pode inspecionar, modificar e melhorar.

Esta plataforma teve sua criação destinada a ajudar estudantes de uma universidade italiana que tinha como premissa que: “[...] o preço almejado não poderia ser mais do que um estudante gastaria se saísse para comer uma pizza” (EVANS, et. al., 2013, p.25).

Após sua criação, foram vendidas rapidamente as unidades fabricadas e a mesma começou a ser amplamente divulgada nas universidades da Itália e mais tarde pelo mundo todo.

Esta plataforma trabalha no conceito de entrada de dados, através de portas eletrônicas e/ou digitais, realiza-se um processamento dos dados e gera-se uma saída, sendo este procedimento executado em *loops* predefinidos pela linguagem de programação.

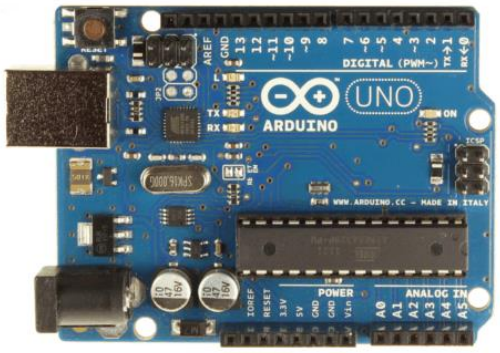
A linguagem utilizada nesta plataforma é a linguagem C, que se tornou uma linguagem adaptável a qualquer dispositivo que siga o padrão ANSI definido para a linguagem.

Segundo o site oficial da plataforma, no que diz respeito à evolução de sua criação:

> Ao longo dos anos, Arduino tem sido o cérebro de milhares de projetos, desde objetos do cotidiano para instrumentos científicos complexos. A comunidade mundial de tomadores - estudantes, amadores, artistas, programadores e profissionais - reuniu em torno desta plataforma *open-source*, suas contribuições acrescentaram-se a uma quantidade incrível de conhecimento acessível que pode ser de grande ajuda para novatos (ARDUINO, 2016, p.1).

O *hardware* que compõe a plataforma Arduino é variado conforme os diversos modelos que foram lançados desde seu lançamento em 2005. Para este projeto iremos analisar o modelo UNO REV 3:

Figura 2 - Modelo plataforma Arduino UNO REV 3.



Fonte: http://www.embarcados.com.br/wp-content/uploads/2013/11/imagem_01.png.

Como visto na Figura 2, a Arduino é composta por:

- 14 pinos de entrada e saída digital (pinos 0-13):

Esses pinos podem ser utilizados como entradas ou saídas digitais de acordo com a necessidade do projeto.

- 6 pinos de entradas analógicas (pinos A0 - A5):

Esses pinos são dedicados a receber valores analógicos, por exemplo, a tensão de um sensor. O valor a ser lido deve estar na faixa de 0 a 5 V que serão convertidos para valores entre 0 e 1023.

- 6 pinos de saídas analógicas (pinos 3, 5, 6, 9, 10 e 11):

São pinos digitais que podem ser programados para serem utilizados como saídas analógicas.

A alimentação da placa pode ser feita a partir da porta USB do computador ou através de um adaptador AC. Para o adaptador AC recomenda-se uma tensão de 9 a 12 volts. Juntamente na placa existe um processador da família Atmel que muda em cada modelo de Arduino.

Para desenvolver nesta plataforma, é necessário o uso de um *software* que faça *upload* do código programado em linguagem C para dentro do micro controlador embutido na placa e que o mesmo seja executado quando a Arduino for conectada à rede elétrica. Segundo o *site* da equipe Embarcados (2016) para se programar na Arduino é necessário a utilização de uma IDE que permite a criação de um código para a placa e após a escrita do código deve-se clicar sobre o botão de *upload* da IDE que então irá traduzir este código feito em linguagem C para uma outra linguagem que possa ser compreendida pelo micro controlador.

Neste projeto a Arduino é responsável pela conectividade com os sensores que realizam a leitura da energia elétrica e disponibilizam estes dados para a central *Raspberry* conectada à nuvem. Sua facilidade de aprendizado e compra por um baixo custo foram fatores relevantes na escolha desta plataforma.

1.2.3 Sensor de corrente TC

Os sensores de correntes TC são dispositivos eletrônicos desenvolvidos para serem aplicados em diversos circuitos elétricos, através de variadas plataformas existentes de prototipagem (Arduino PIC, *Raspberry*) para mensurarem a corrente elétrica de algum dispositivo. Ele é conectado diretamente à Arduino e faz a mensuração dos valores de corrente de cada cômodo/disjuntor enviando os dados para a plataforma de desenvolvimento.

Existem diversos modelos no mercado com diversas potências máximas de mensuração. Para o presente projeto estão sendo utilizados modelos de 100 amperes.

Figura 1 - Modelo de Sensor de Corrente TC SCT 013-000

Fonte: http://s3.amazonaws.com/img.iluria.com/product/18B9DA/3A3A8A/450xN.jpg

Os sensores da família TC SCT são denominados não-invasivos, ou seja, como pode ser visto na Figura 3, ele detém de um encaixe lateral que se abre e fecha para a colocação do fio que precisa ser medido a corrente, este fato faz com que não necessite qualquer manuseio da fiação, oferecendo maior segurança na sua instalação.

1.3 Linguagens de programação

O desenvolvimento de um *software* é baseado em uma escrita de códigos em uma linguagem de programação específica. Dentre as mais variadas linguagens presentes e disponíveis na comunidade tecnológica foram utilizadas as linguagens descritas abaixo.

1.3.1 NodeJS

Node.js é uma plataforma construída sobre o motor *JavaScript* V8[2] desenvolvido pela *Google* para facilitar o desenvolvimento de aplicações de redes rápidas e escaláveis. *Node.js* usa um modelo de I/O direcionada a evento não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos (MOREIRA, 2013).

Na JSConf Européia de 2009, um programador jovem chamado Ryan Dahl mais 14 colaboradores apresentaram um projeto para solucionar um problema de bloqueio de execução em sistemas com muitas requisições simultâneas, que fazia com que um processo travava os outros até que se terminasse a execução do mesmo e assim os demais viessem a ser executados

[2] JAVASCRIPT V8 - interpretador ultrarrápido, escrito em C++ que faz download do mecanismo e o redireciona para uso no servidor.

em fila. Este projeto era uma plataforma que combinava a máquina virtual *JavaScript* V8 e um laço de eventos para aproveitar o poder e a simplicidade do *Javascript*, tornando aplicações assíncronas tarefas fáceis de escrever (MOREIRA, 2013).

Já Pereira menciona que:

> Esta tecnologia possui um modelo inovador, sua arquitetura é totalmente *non-blocking thread* (não-bloqueante), apresentando uma boa performance com consumo de memória e utilizando ao máximo e de forma eficiente o poder de processamento dos servidores, principalmente em sistemas que produzem uma alta carga de processamento (PEREIRA, 2015, p.2).

Usuários de sistemas *Node* estão livres de aguardarem por muito tempo o resultado de seus processos, e principalmente não sofrerão de *dead-locks* [3]no sistema, porque nada bloqueia em sua plataforma e desenvolver sistemas nesse paradigma é simples e prático (Pereira, 2013).

Esta tecnologia foi aplicada no desenvolvimento do serviço *RESTful*, sendo implementada no *back-end* [4]do projeto, pois não requer um grande volume de lógica, já que simplesmente procura valores em um banco de dados e monta uma resposta que gera um pequeno volume de tráfego utilizando somente a memória necessária para a conexão.

1.3.2 JavaScript

JavaScript é uma linguagem de programação criada pela Netscape em parceria com a Sun Microsystems. Sua primeira versão, definida como *JavaScript* 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador *Netscape Navigator* 2.0 (SILVA, 2010).

Conforme *Mozilla Developer Network* (2016), *JavaScript* é baseado na linguagem de programação ECMAScript, o qual é padronizado pela Ecma International na especificação ECMA-262 e ECMA-402.

JavaScript está sendo utilizado no *core* aplicação *web* que contém a interface que o cliente/usuário acessa para verificação de todos seus dados lidos pelos sensores. A utilização de um *framework JavaScript* também foi utilizado e detalhado logo abaixo, nas próximas seções.

---

[3] DEAD-LOOK – É quando duas ou mais tarefas bloqueiam uma à outra permanentemente, sendo que cada uma tem o bloqueio de um recurso que a outra tarefa está tentando bloquear.

[4] BACK-END - Sistema responsável pela regra de negócios, webservices e APIs de uma aplicação.

1.3.3 Linguagem C

As linguagens de programação servem para escrever programas que permitem a comunicação entre usuário e máquina. Programas especiais chamados tradutores (compiladores ou interpretadores) convertem as instruções escritas em linguagens de programação em instruções escritas em linguagens de máquina (0 e 1 bits) que a máquina pode entender. (AGUILAR, 2011).

Foi criada em 1972 em um laboratório chamado *Bell Telephone Laboratories*, por Dennis Ritchie, para que fosse permitido a escrita de um sistema operacional, o UNIX, que utilizasse os benefícios de uma linguagem de alto nível, deixando de lado o uso de linguagens de baixo nível que detinham de uma maior complexidade para seu uso (DAMAS, 2007).

Inicialmente, a linguagem servia para desenvolvimento de programas de sistemas que são, segundo Schildt (1986, p.2), "[...] uma classe de programa que, ou são parte, ou operam em conjunto com o sistema operacional do computador [...]".

Atualmente, após mais de quatro décadas de sua criação, a linguagem se apresenta ao mercado também em dispositivos eletrônicos de pequeno porte como micro controladores pelo fato de ser uma linguagem de fácil portabilidade. Tal expansão da linguagem pode ser observada em Klotz (2010, p.3) que diz que "uma das melhores características da linguagem C é que não está ligada a qualquer equipamento ou sistema em particular. Isto torna mais fácil para o usuário escrever programas que serão executados sem quaisquer alterações em praticamente todas as máquinas"[5].

Devido a todos estes conceitos teóricos e históricos no que tange a abrangência da linguagem C em dispositivos embarcados, tornou-se viável o uso da mesma no projeto, atuando na plataforma de desenvolvimento embarcado, o Arduíno, fazendo toda a leitura da corrente elétrica e enviando os dados para uma central receptora, *Raspberry* que será detalhada nas próximas seções.

1.4 Frameworks e bibliotecas

Para obter o resultado esperado com maior rapidez e qualidade, optou-se por utilizar algumas bibliotecas e frameworks. As bibliotecas são empregadas para, na maioria das

[5] Original "One of the best features of C is that it is not tied to any particular hardware or system. This makes it easy for a user to write programs that will run without any changes on practically all machines". Traduzido pelos autores.

ocasiões, prover uma simplificação de criação de código nas etapas do desenvolvimento, já que as mesmas fornecem uma gama de funções e métodos pré-definidos, poupando assim, o esforço do desenvolvedor em criar todo código a partir do zero. Diferentemente das bibliotecas, em que adequamos seu uso dentro do código, em *frameworks* temos a adequação do código dentro do *framework*, e nesse contexto temos então diversos recursos disponíveis ao optar por utilizar um *framework*. Confira adiante as especificações das bibliotecas e *frameworks* utilizados no trabalho.

1.4.1 AngularJS

AngularJS é um *framework open-source* mantido pela Google. Seu objetivo é estruturar o desenvolvimento *front-end*[6] utilizando o modelo de arquitetura *model-view-controller* (MVC). O *framework* associa os elementos do documento HTML com objetos JavaScript, facilitando a manipulação dos mesmos.

É considerado um *framework* estrutural para desenvolvimento de aplicativos *Web* dinâmicos. Ele permite a utilização do *HTML* como um *template*, possibilitando que a *sintaxe* HTML seja estendida para representar de forma mais clara e sucinta os componentes da aplicação. O *AngularJS* é um arquivo *JavaScript*, não necessitando procedimentos de instalação. (ANGULARJS, 2016).

O objetivo do *AngularJS* é trazer grande parte das ferramentas e capacidades que são codificadas no lado servidor para o lado cliente. Com isso, pretende-se facilitar o desenvolvimento, os testes e a manutenção dos sistemas. O Angular permite também a extensão do *HTML* expressando as funcionalidades através de elementos personalizados, atributos, classes e comentários (FREEMAN, 2014, p. 45).

O *framework* faz uso dos conceitos de *Data Binding* e de injeção de dependência, que são importantes para o entendimento da ferramenta. O *Data Binding* é uma técnica que se constitui em modificar algo no modelo (dados dos objetos) da aplicação e refletir no DOM as novas informações ou vice-versa. Bibliotecas como *jQuery* proveem maneiras de modificar parte do DOM separadamente, sem a necessidade de recarregar a página completamente. Neste caso, são adicionados somente dados no *template*, que são atualizados com a ajuda da função *innerHtml*. A execução desta função alcança o objetivo principal que é a atualização do *template* de maneira assíncrona, sem necessidade de novo carregamento da página, porém

[6] FRONT-END - Interface de interação com o usuário.

demanda cuidados não triviais para o controle tanto dos dados no HTML quanto dos dados nos objetos *JavaScript*. Com o uso do *AngularJS*, esse trabalho de verificação é geralmente automático, e a preocupação com essa vinculação é reduzida (GREEN e SESHADRI, 2013).

Atualmente, existem duas versões do *framework*, a primeira e mais amplamente conhecida é a que foi utilizada neste projeto, enquanto a outra lançada no ano de 2016 contempla a utilização de outra forma de desenvolvimento que utiliza o *TypeScript*, que nada mais é que uma abstração da linguagem *JavaScript* para facilitar o desenvolvimento e se assemelhar à outras linguagens de programação como Java por exemplo, através da criação de componentes, existência de herança e variáveis com tipos definidos.

1.4.2 Materialize

Um *framework front-end* com *layout* moderno e responsivo que segue os padrões de desenvolvimento baseado em *Material Designer* da Google (MATERIALIZE, 2016).

Criado e projetado pela Google, o *Material Designer* é a combinação de princípios clássicos de projetos bem-sucedidos que contemplam inovação e tecnologia, seguindo o objetivo de unificar a experiência do usuário em todos os seus produtos independentemente de qual plataforma está sendo utilizada (MATERIALIZE, 2016).

Neste projeto foi implementado em toda aplicação que possui acesso direto dos usuários. Sua utilização favorece a usabilidade do cliente que contará com uma interface totalmente responsiva e inovadora.

1.4.3 JQuery

JQuery é uma biblioteca baseada em *JavaScript* de *software* livre e aberto criado por John Resig e seu uso está sob as licenças do MIT e GPL, ou seja, é permitido que qualquer pessoa ou empresa use de seus recursos para fins pessoais como também profissionais (SILVA, 2014).

Em 2005, o autor começou a elaborar suas primeiras intenções sobre como resolver problemas de robustez de código para solução de simples problemas (SILVA, 2014). Em 2006, foi lançado o primeiro *plugin* [7]para a biblioteca e disponibilizado de forma pública.

[7] PLUGIN - Todo programa, ferramenta ou extensão que se encaixa a outro programa principal para adicionar mais funções e recursos a ele.

Ainda segundo Silva (2014), a função dessa biblioteca é adicionar interatividade e dinamismo às páginas *web* que consequentemente irão proporcionar ao desenvolvedor, mecanismos necessários à criação de *scripts* [8]fácil e usável, sem obstrução e proporcionando ao usuário final uma melhor experiência de usabilidade na *web*.

Esta biblioteca segue os padrões de desenvolvimento *web* e é totalmente compatível com os novos recursos do atual CSS3. Porém, deve-se atentar para que esteja nas conformidades das normas do W3C.

Seu uso no projeto tem como premissa a disponibilização de maior rapidez e usabilidade no desenvolvimento do código *front-end* que é destinada à aplicação *web* proposta ao usuário final.

1.4.4 Highcharts

Highcharts é uma biblioteca de gráficos escrito em *JavaScript* puro, lançado em 2009, oferecendo uma maneira fácil de adicionar gráficos interativos para seu *site* ou aplicação *web*. Atualmente suporta linha, spline área, areaspline, coluna, barra, torta, dispersão, medidas de ângulo, arearange, areasplinerange, columnrange, bolha, gráfico de caixa, barras de erro, funil, cachoeira e tipos de gráficos polares. (HIGHCHARTS, 2016).

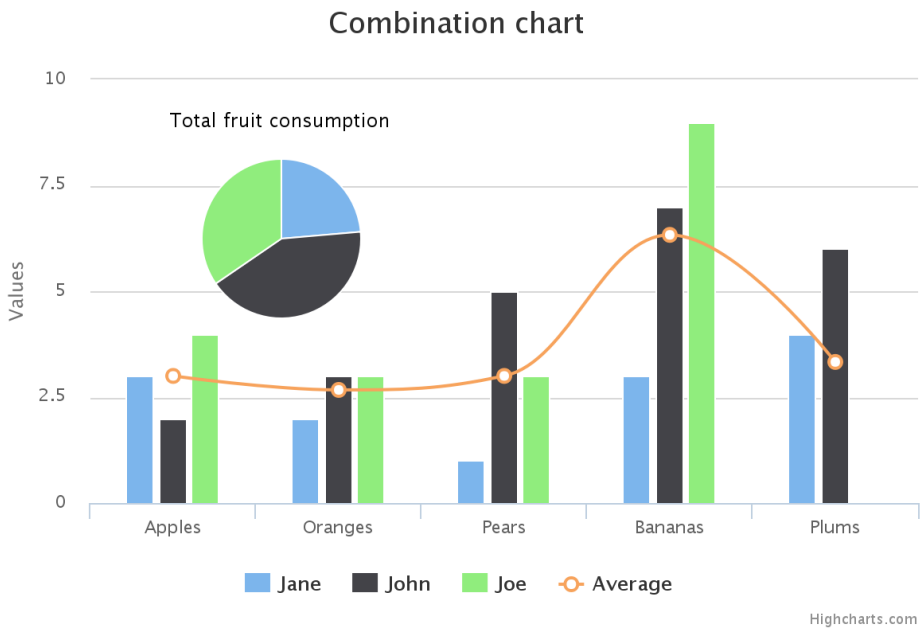
É um software de criação de gráficos *web* popular que produz animados gráficos impressionantes e suaves *JavaScript* e HTML5 SVG. Ele está entre o líder de *software* no mercado e tem sido utilizado em diversos setores para *sites* sociais. Embora seja construído em cima do *jQuery*, é tão simples de construir que você precisa pouca habilidade de programação para criar um gráfico *web* simples. (HIGHCHARTS, 2016).

Funciona em todos os navegadores móveis e desktop modernos, incluindo o *iPhone* / *iPad* e Internet Explorer a partir da versão 6. Nos navegadores padrões deve usar SVG para a renderização de gráficos. Em legado do Internet Explorer gráficos são desenhados usando VML. É baseado exclusivamente em tecnologias de navegador nativo e não requer *plugins* colaterais como *Flash* ou *Java*. Além disso, não precisa instalar nada no seu servidor. (HIGHCHARTS, 2016).

A Figura 4 apresenta um modelo do gráfico *Highchart*:

[8] SCRIPT - são “roteiros” seguidos por sistemas computacionais e trazem informações que são processadas e transformadas em ações efetuadas por um programa principal.

Figura 4 - Modelo gráfico *highcharts - Column, line , and pie*



Fonte: http://www.highcharts.com/demo/combo.

Highcharts foi implementado no desenvolvimento deste projeto, pois tornou mais fácil a criação de gráficos interativos na geração de relatórios nas páginas da aplicação *web* e pela disponibilização dos códigos fontes gratuitamente, que nos permitiu modificações com grande flexibilidade.

1.5 Armazenamento e gerenciamento de dados

Para obter a persistência de dados dentro de um *software* é necessário armazená-los dentro de algum banco de dados, seja ele relacional, orientado a documentos, orientado a grafos ou outros. Este armazenamento é algo fundamental para qualquer solução *web* e neste trabalho foi utilizado dois tipos de banco de dados, cada um para armazenar dados específicos ao qual a informação é necessária ser persistida. Adiante temos suas especificações e referencial teórico.

1.5.1 MySQL

MySQL é o banco de dados de código aberto mais popular do mundo e possibilita a entrega econômica de aplicativos de banco de dados confiáveis, de alto desempenho e redimensionáveis, com base na *web* e incorporados (ORACLE, 2016).

Além da facilidade de uso, do alto desempenho e da confiabilidade do *mySQL*, pode-se beneficiar dos recursos avançados, das ferramentas de gerenciamento e do suporte técnico para desenvolver, implementar e gerenciar seus aplicativos.

É amplamente utilizado no projeto para armazenar os dados lidos e todas as demais configurações e informações que necessitam serem persistidas de forma *online*, como contas de usuário, históricos de acessos e outros.

1.5.2 MongoDB

MongoDB é um banco de dados orientado a documentos de código aberto que fornece alto desempenho, alta disponibilidade e escala automática. A persistência dos dados é feita através de objetos JSON com esquema dinâmico. Isso significa que você pode armazenar seus registros sem se preocupar com a estrutura de dados, com o número de campos ou tipos de campos para armazenar valores.

É um banco de dados multi-plataforma NoSQL, mantido pela empresa 10gen e foi escrito em linguagem C++. Utiliza javascript como interface para manipulação de dados. Pode ser executado em windows, linux, mac e solares. Suporta as linguagens de programação mais populares, como: C, C++, C#, Java, PHP, Javascript, NodeJs, Ruby e Python.

Guarda os dados em documentos ao invés de tabelas. Você pode alterar a estrutura de registros (que é chamado como documentos no MongoDB) simplesmente adicionando novos campos ou excluindo os existentes. Esta capacidade do MongoDB é útil para representar relações hierárquicas, para armazenar matrizes, e outras estruturas mais complexas de uma maneira mais simples (SOARES, 2016).

Nele trabalhamos com o conceito schema-less, ou seja, não existem relacionamentos de tabelas, nem chaves primárias ou estrangeiras e sim documents que possuem embedded documents e tudo mantido dentro de uma collection. Outra vantagem do schema-less é que os atributos são inseridos ou removidos em runtime, sem a necessidade de travar uma collection, tornando este banco de dados flexível a grandes mudanças. Isso diminui o número de consultas complexas no banco de dados e principalmente evita criar joins para carregar diversas informações de uma vez (PEREIRA, 2013).

MongoDB introduz novos mecanismos de armazenamento que ampliam as capacidades do banco de dados, permitindo-lhe escolher as tecnologias ideais para as diferentes cargas de trabalho em sua organização. Executar vários mecanismos de armazenamento em uma implantação e alavancar a mesma linguagem de consulta, escalando métodos e ferramentas

operacionais em todos eles para reduzir significativamente o desenvolvimento e complexidade (MONGODB, 2016).

A grande vantagem de trabalhar com esse modelo de banco de dados é a grande compatibilidade e suporte mantido pela comunidade própria Node.js.

1.6 Métodos de integração e suas tecnologias

Para atender aos requisitos do projeto necessitou-se o uso de uma forma de desacoplar as camadas envolvidas na solução, a fim de facilitar futuras alterações e manutenção do *software*. Estas tecnologias e conceitos são descritas abaixo.

1.6.1 RESTful

Em 2000, Roy Fielding, em sua tese de doutorado apresentou uma nova forma de integrar sistemas hipermídias distribuído chamado REST (*Representational State Transfer*) (SANTOS, 2009).

Ainda de acordo com Santos (2009), REST é um estilo de arquitetura de *software* para sistemas hipermídia distribuídos, no qual utilizamos um navegador *web* para acessar recursos, mediante a digitação de uma URL.

Requisições web possui três tipos de componentes importantes: substantivos, verbos e tipos de conteúdo. Uma aplicação *RESTful* deve dar mais importância para os substantivos do que os verbos, além de conter uma variedade de tipos de conteúdo determinados. A principal vantagem de seguir isso é que conseguimos aproveitar toda a estrutura que o protocolo HTTP proporciona (CAELUM, 2016).

a) ***Substantivos***: Substantivos são os nomes dos recursos do seu sistema. Quando fazemos requisições *Web*, precisamos falar o caminho da mesma, ou seja, o identificador único de um recurso. Então ao criar substantivos do nosso sistema devemos levar em consideração que eles devem representar recursos, e não ações. Em um sistema REST, que nosso recurso chama, por exemplo, ‘/produtos/incluir não estará seguindo os padrões RESTful, pois contém um verbo e não está identificando um recurso, mas sim uma operação. Portanto para representar a inclusão de um produto podemos usar apenas ‘/produtos’ juntamente com o método HTTP *POST*, que quer dizer que estamos adicionando alguma informação no sistema referente a ao recurso produtos (CAELUM, 2016).

b) ***Verbos***: Segundo Caelum (2016), dentro de um sistema *RESTFul* devemos contemplar um número pequeno de operações que devem servir uma interface uniforme ao cliente/servidor. O protocolo HTTP possui sete operações: os métodos *GET, POST, PUT, DELETE, HEAD, OPTIONS* e *TRACE*, dentro os quais quatro deles representam o que chamamos de CRUD (*create, retrieve, update e delete*) que são:

GET - recupera informações sobre o recurso identificado. Ex: listar produtos, visualizar o produto 45. Uma requisição GET não deve modificar nenhum recurso você apenas recupera informações do sistema.

POST - adiciona informações. Ex: adicionar um produto. Pode adicionar informações a um recurso ou criar um novo recurso.

PUT - adiciona (ou modifica) um recurso na URI passada. Ex: atualizar um produto.

DELETE - remove o recurso representado pela URI passada. Ex: remover um produto.

c) ***Tipos de Conteúdo:*** Quando utilizamos uma aplicação *web* não trafegamos um recurso pela rede, simplesmente uma representação dele. Por exemplo, quando queremos adicionar um produto ao sistema, passamos para o servidor uma representação do produto: dados do formulário. E quando pedimos uma lista de produtos para o servidor ele responde com uma representação HTML desses produtos. Mas não precisa ser restrito a isso: poderíamos adicionar um produto ao sistema via uma representação em XML, e o servidor poderia nos devolver uma lista de produtos em formato JSON (CAELUM, 2016).

Resumindo: nossas chamadas devem representar recursos, as operações no recurso devem ser indicadas pelos verbos HTTP e podemos ainda falar qual é o formato (JSON, XML) em que iremos comunicar com o servidor com o *Content Type*.

Esta arquitetura e conceito foram aplicados no core da API[9] para padronizar e desacoplar nossas camadas de desenvolvimento que são: *hardware,* serviço e aplicação *web*. Esta arquitetura nos traz manutenibilidade do sistema e benefícios para futuras mudanças na arquitetura de *hardware*.

1.6.2 JavaScript Object Notation (JSON)

JSON é um formato de dados que pode ser trocado, permutado e que se assemelha nos padrões de *scripts*, porém, não é considerado um subtipo da sintaxe do *JavaScript* (MOZILLA DEVELOPER NETWORK, 2014).

[9] API - *Application Programming Interface* (Interface de Programação de Aplicativos). É a interface, normalmente documentada que uma biblioteca ou *framework* disponibiliza para que o programador possa utilizá-la.

“JSON é capaz de representar números, *booleanos*, textos, listas (sequência ordenada de valores) e objetos (mapeamento de valores de texto) compostos por estes valores (ou por outras listas e objetos) ” (MOZILLA DEVELOPER NETWORK, 2014, p. 1).

Um JSON está constituído em duas estruturas:

- Uma coleção de pares nome/valor.
- Uma lista ordenada de valores.

A aplicação deste formato na solução é realizada para desacoplarmos nossas entidades de *hardware*, serviço e aplicação *web*, pois este formato é entendido facilmente por seres humanos e traduzidos com a mesma facilidade nos ambientes de máquinas, ou seja, podemos trocar totalmente qualquer entidade citada anteriormente por outra, até mesmo com outros equipamentos ou outras linguagens, bastando mantermos o formato JSON de entrada e saída de dados.

REFERÊNCIAS

AGUILAR, L. J. **Fundamentos de Programação - Algoritimos, estruturas de dados e objetos**. 3ª Edição. ed. São Paulo: AMGH Editora, 2008.

ANGULARJS. O que é AngularJS. **ANGULARJS**, 2016. Disponivel em: <https://docs.angularjs.org/guide/introduction>. Acesso em: 11 Agosto 2016.

ARDUINO Introduction. **Arduino**, 2016. Disponivel em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 10 Março 2016.

ASHTON, K. That "Internet of Things" Thing. **RFID Journal**, 2009. Disponivel em: <http://www.rfidjournal.com/articles/view?4986>. Acesso em: 20 Março 2016.

CAELUM. Rest - Web Ágil com VRaptor, Hibernate e Ajax. **CAELUM**, 2016. Disponivel em: <https://www.caelum.com.br/apostila-vraptor-hibernate/rest/#11-3-o-triangulo-do-rest>. Acesso em: 02 Agosto 2016.

DAMAS, L. **Linguagem C**. 10. ed. Rio de Janeiro: LTC, 2007.

EVANS, D. **The internet of things:** how the next evolution of the internet is changing everything. [S.l.]: CISCO white paper, 2011. 11 p.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. **Arduino em ação**. São Paulo: Novatec Editora, 2013.

FREMAN, A. **Pro AngularJS**. Nova Iorque: Apress Media LLC, 2014.

GREEN, B.; SESHADRI, S. **AngularJS**. Sebastopol: O'Reilly Media, 2013.

HIGHCHARTS. Highcharts Documentation. **Highcharts**, 2016. Disponivel em: <http://www.highcharts.com/docs>. Acesso em: 26 Julho 2016.

KENDE, M. Global Internet report. **Geneva:** Internet Society, 2014. Disponivel em: <http://www.internetsociety.org/doc/global-internet-report>. Acesso em: 21 Março 2016.

KLOTZ, D. C for Embedded Systems Programming. **NXP**, 11 Novembro 2010. Disponivel em: <http://www.nxp.com/files/training/doc/dwf/AMF_ENT_T0001.pdf>. Acesso em: 10 Março 2016.

LACERDA, F.; LIMA-MARQUES, M. Da necessidade de princípios de Arquitetura da Informação para a Internet das Coisas. **Perspectivas em Ciência da Informação**, Belo Horizonte, 20, Junho 2015. 158-171.

MATERIALIZE. Documentação - Materialize. **Materialcss**, 2016. Disponivel em: <http://materializecss.com/>. Acesso em: 21 Março 2016.

MONGODB. MongoDB for GIANTI Ideas. **MongoDB**, 2016. Disponivel em: <https://www.mongodb.com/>. Acesso em: 01 Agosto 2016.

MOREIRA, R. H. O que é Nodejs? **NodeBr**, 2013. Disponivel em: <http://nodebr.com/o-que-e-node-js/>. Acesso em: 20 Julho 2016.

MOZILLA DEVELOPER NETWORK. **JSON**, 2014. Disponivel em: <https://developer.mozilla.org/pt-BR/docs/JSON>. Acesso em: 20 Março 2016.

MOZILLA DEVELOPER NETWORK. **JavaScript**, 2016. Disponivel em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 22 Abril 2016.

ORACLE. MySQL | O Banco de Dados de Código Aberto Mais Popular. **ORACLE**, 2016. Disponivel em: <http://www.oracle.com/br/products/mysql/overview/index.html>. Acesso em: 20 Fevereiro 2016.

PEREIRA, C. R. **Node.js:** Aplicações web real-time com Node.js. São Paulo: Casa do Código, 2015.

RASPBERRY PI, 2014. Disponivel em: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>. Acesso em: 20 Março 2016.

SANTOS, W. R. D. **Univale**, 2009. Disponivel em: <http://www.univale.com.br/unisite/mundo-j/artigos/35RESTful.pdf>. Acesso em: 23 Fevereiro 2016.

SCHILDT, H. **Linguagem C:** Guia do Usuário. São Paulo: MC Graw Hill, 1986.

SILVA, M. S. **JavaScript:** Guia do Programador. São Paulo: Novatec Editora, 2010.

SILVA, M. S. **JQuery:** A Biblioteca do Programador JavaScript. 3. ed. São Paulo: Novatec Editora, 2014.

SILVEIRA, G. Internet das Coisas - O que é IoT? **Bluelux**, 2016. Disponivel em: <http://www.bluelux.com.br/internet-das-coisas-iot/>. Acesso em: 31 Julho 2016.

SOARES, J. O que é MongoDB e poque usá-lo? **Código SImples**, 2016. Disponivel em: <http://codigosimples.net/2016/03/01/o-que-e-mongodb-e-porque-usa-lo/>. Acesso em: 10 Agosto 2016.

SOUZA, F. Arduino UNO - Conheça os detalhes de seu hardware. **Embarcados**, 2016. Disponivel em: <http://www.embarcados.com.br/arduino-uno/>. Acesso em: 20 Fevereiro 2016.

STAMFORD, C. Gartner Says the Internet of Things installed base will grow to 26 billion units by 2020. **Gartner**, 2013. Disponivel em: <http://www.gartner.com/newsroom/id/2636073>. Acesso em: 21 Março 2016.

UPTON, E.; , G. H. **Raspberry pi:** Manual do usuário. Tradução de Celso Roberto Paschoa. São Paulo: Novatec Editora, 2013.