UNIVERSIDADE FEDERAL DO PARANÁ
ENGENHARIA ELÉTRICA

CAIO AUGUSTO ALBANO PASQUAL

DESENVOLVIMENTO SOLUÇÃO DE BALANÇA IOT COM ARMAZENAMENTO NA NUVEM E ACOMPANHAMENTO POR APLICAÇÃO WEB PWA

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2018

CAIO AUGUSTO ALBANO PASQUAL

DESENVOLVIMENTO SOLUÇÃO DE BALANÇA IOT COM ARMAZENAMENTO NA NUVEM E ACOMPANHAMENTO POR APLICAÇÃO WEB PWA

Trabalho de conclusão de curso de graduação apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Paraná como requisito à obtenção do grau de Engenheiro.

Orientador: Prof. Dr. Eduardo Parente Ribeiro

CURITIBA
2018

AGRADECIMENTOS

Ao orientador Prof. Dr. Eduardo Parente Ribeiro, pelo incentivo e suporte no desenvolvimento deste projeto.

Aos meus colegas de sala e grandes amigos que me acompanharam em toda a minha jornada acadêmica.

A todos os professores que sempre acreditaram no meu potencial e me motivaram a sempre conquistar meus objetivos.

À Secretaria do Curso, por todo o apoio.

RESUMO

Esse trabalho aborda o projeto de uma solução IOT de balança para ser utilizada por pessoas que desejam acompanhar suas medidas de peso. O produto foi idealizado para auxiliar pessoas a terem um acompanhamento mais preciso da variação da sua massa corporal, facilitando a verificação de progressos ou retrocessos de acordo com objetivos de ganho ou perda de peso. O acompanhamento das medidas será feito através de uma aplicação *web* progressiva (PWA), que poderá ser acessada a qualquer momento, com ou sem internet. Este documento começa determinando os requisitos do aparelho e as premissas da aplicação. Assim, é desenvolvido um protótipo funcional, documentando-se o processo de desenvolvimento de cada parte da aplicação. A aplicação é constituída de *back-end*, *front-end*, *firmware* e *hardware*, que utiliza um módulo ESP8266.

Palavras-chave: IOT, Balança, Web, PWA, Cloud

ABSTRACT

This paper discusses the design of an IOT scale solution to be used by people who desire to follow their weight measurements. The product was designed to assist people on having a more precise monitoring of the variation of their body mass, facilitating the verification of progress or setbacks according to goals of weight gain or loss. The measures will be monitored through a progressive web application (PWA), which can be accessed at any time, with or without internet. This document begins by determining the device requirements and the application assumptions. Thus, a functional prototype is developed, documenting the development process of each part of the application. The application consists of back-end, front-end, firmware and hardware, which utilizes an ESP8266 module.

Keywords: IOT, Scale, Web, PWA, Cloud

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ADC	_	*Analog-Digital Converter*
API	_	*Application Program Interface*
BNDES	_	Banco Nacional do Desenvolvimento
CPF	_	Cadastro de Pessoa Física
CSS	_	*Cascading Style Sheet*
DBMS	_	*Database Management System*
GPIO	_	*General-Purpose Input/Output*
HTML	_	*Hypertext Markup Language*
IOT	_	*Internet of Things*
JSON	_	*JavaScript Object Notation*
LED	_	*Light Emitting Diode*
PWA	_	*Progressive Web Application*
REST	_	*Representational State Transfer*
SOAP	_	*Simple Object Access Protocol*
SQL	_	*Structured Query Language*
SSID	_	*Service Set Identifier*
SSL	_	*Secure Sockets Layer*
TLS	_	*Transport Layer Security*
UFPR	_	Universidade Federal do Paraná
URI	_	*Uniform Resource Identifier*
URL	_	*Uniform Resource Locator*
URN	_	*Uniform Resource Name*
XML	_	*Extensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

Diversos projetos de soluções inovadoras buscam melhorar a qualidade de vida das pessoas utilizando tecnologias recentes, principalmente incorporando-as no dia a dia, por exemplo através de dispositivos de internet das coisas (IOT). Esta classe de dispositivos está gerando o que é chamado de "Revolução IOT", que traz a capacidade de integração da internet no contexto local dos usuários, com dispositivos enxutos, portáteis e de baixo consumo, permitindo assim a análise, coleta e acessibilidade de dados, tanto no meio industrial quanto residencial.

No meio industrial, dispositivos IOT podem ser utilizados para coletar dados como temperatura e umidade de grandes áreas, com melhor compatibilidade eletromagnética e sem um extenso e complexo cabeamento, permitindo também a mobilidade e rearranjo dos sistemas como necessário. Já no âmbito residencial, este tipo de dispositivo auxilia nas mais diversas tarefas diárias, como por exemplo controle de termostato, ar condicionado, dispositivos de segurança e integração de sistemas de "casas inteligentes".

Tendo em vista o contexto apresentado, demonstra-se grande interesse no desenvolvimento e implementação destes dispositivos. O ecossistema de produtos IOT está em constante crescimento, e projetos inovadores de grande impacto são sempre apreciados pelo mercado. A crescente importância de mercado desses dispositivos atraiu o interesse do BNDES (Banco Nacional de Desenvolvimento Econômico e Social), que realizou pesquisa sobre o mercado IOT no Brasil, no período entre o começo de 2017 até aproximadamente Março de 2018. A partir dos dados coletados a instituição, que atua como principal instrumento do Governo Federal para o financiamento de longo prazo e investimento em todos os segmentos da economia brasileira, definiu um plano de ação para incentivar os investimentos no mercado IOT no Brasil.

A conclusão destes estudos foi de que os dispositivos IOT têm uma grande capacidade de impacto na economia do país, sendo essenciais para melhorar o "custo Brasil" – custos adicionais de burocracia, infra-estrutura e impostos que penalizam empreendimentos brasileiros frente à realidade mundial. Além disso o mercado global de IOT recebe um investimento anual de aproximadamente 3.3 bilhões de dólares. Valores estes que têm crescido exponencialmente nos últimos anos. E com uma expansão tão intensa de receita

mundial o mercado de IOT para o ano de 2025 é previsto na casa dos quatro trilhões de dólares.

Quanto às implicações práticas, importa mencionar que dentre os setores mais impactados por estes dispositivos estão a indústria de manufatura e a área da saúde. No que tange à área da saúde, os dispositivos IOT prestam-se à realização de monitoramentos efetivos aptos a proporcionar melhorias diretas na qualidade de vida de pessoas que, padecendo de alguma enfermidade, precisem controlar constantemente determinadas condições médicas. Neste âmbito, a utilização de tais dispositivos vai desde aplicações mais complexas, como marca-passos, que regulam os batimentos cardíacos, até as mais simples, tais como a que se propõe neste projeto: uma balança de uso cotidiano que permite acompanhar mais precisamente as variações de peso do usuário.

Outro fator importante que se considera é a disponibilidade do serviço e facilidade de acesso. Usuários de smartphones se mostram cada vez mais resistentes à instalação de novos aplicativos. Por este motivo uma aplicação *web*, que não precisa ser instalada, mas pode ser acessada como uma aplicação nativa é muito mais bem aceita pelo mercado. Estas aplicações são denominadas aplicações *web* progressivas, ou PWA. Outro ganho é a alta disponibilidade, mesmo quando o usuário não possui acesso à internet.

Com efeito, objetiva-se, aqui, desenvolver uma solução inovadora em face do grave quadro de saúde da população brasileira, que tem aproximadamente metade dos adultos acima do peso, com cerca de 15% de indivíduos obesos, em todas as classes sociais. Portanto, a situação crítica da obesidade no país, evidencia a necessidade de soluções inovadoras para atacar este problema.

1.1. OBJETIVO GERAL

Projetar, construir e desenvolver uma solução completa de IOT de baixo custo para acompanhamento das medidas de peso do usuário com armazenamento na nuvem. As medidas poderão ser acompanhadas por qualquer dispositivo com acesso à internet que possua um navegador *web*, e ser acessíveis mesmo sem conexão à rede. Os dados gravados deverão conter o peso, data e hora da pesagem e ser únicos para cada usuário.

1.2. OBJETIVOS ESPECÍFICOS

a) Utilizar soluções Firebase
b) Utilizar módulo ESP8266
c) Definir arquitetura do banco de dados
d) Desenvolver sistema de acesso ao banco
e) Desenvolver *front-end* da aplicação
f) Integrar sistemas

1.3. JUSTIFICATIVA

Unindo as soluções de IOT que estão entrando no mercado e as novas tecnologias de desenvolvimento *web* é possível criar produtos específicos para melhorar a qualidade de vida das pessoas. Através de um produto que auxilie tanto médicos quanto nutricionistas a acompanhar as variações do peso de um paciente obeso, pode-se aumentar consideravelmente a efetividade de um tratamento.

Por sua vez, pessoas que não estão em risco podem ser beneficiadas com o acompanhamento de sua massa corporal no dia a dia, como medida de prevenção à obesidade e controle de tais medidas em médio e longo prazo. Assim, o escopo deste projeto é demonstrar a possibilidade de criar um produto de baixo custo com preço acessível, utilizando novas tecnologias disponíveis no mercado, trazendo às pessoas uma solução completa de IOT que as auxilie a ganhar, perder ou acompanhar seu peso de maneira simples e eficiente.

1.4. PROJETOS SIMILARES

No mercado, existem alguns produtos que contemplam as funcionalidades principais do projeto desenvolvido, porém com um alto custo. A maioria destes aparelhos não mede apenas o peso, mas também a bioimpedância do corpo – um valor de resistividade relativa proporcional aos índices de massa corporal, muscular e óssea que pode ser utilizado na medida destes parâmetros. A seguir se encontra um quadro comparativo entre os principais produtos similares presentes no mercado.

QUADRO 1 - PRODUTOS SIMILARES

Produto	Bioimpedância	Aplicação	Painel	Preço (R$)
Fitbit Aria 2	Sim	Sim	Sim	~900,00
Nokia Body Cardio	Sim	Sim	Sim	~1500,00
Yunmai Mini	Sim	Sim	Não	~290,00
Garmin Index	Sim	Sim	Sim	~1300.00
Eufy BodySense	Sim	Sim	Não	~680,00

FONTE: O Autor (2018)

Assim, percebe-se a falta de soluções de baixo custo para acompanhamento de medidas de peso, inexistindo produtos na faixa de valores inferior a R$ 150,00.

2. FUNDAMENTAÇÃO TEÓRICA

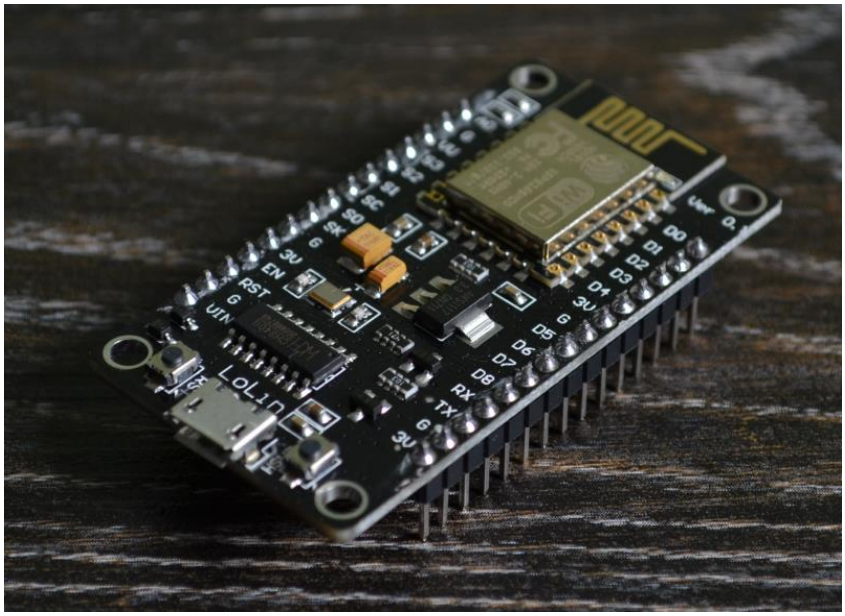
A seguir estão descritos os recursos de *hardware*, *firmware* e *software* utilizados neste projeto. A escolha das tecnologias utilizadas se baseou principalmente em seu custo, facilidade e uso e disponibilidade.

2.1. ESP8266 NODEMCU WI-FI DEVKIT

O módulo ESP8266 NodeMCU Wi-Fi Devkit (FIGURA 1) é uma plataforma de desenvolvimento para o ESP-12E, e funciona de maneira similar às *launchpads* da Texas Instruments.

Ele apresenta uma interface micro-USB para programação do módulo e comunicação serial com o controlador. O ecossistema que acompanha o módulo permite que o microcontrolador seja programado em Lua, Micro Python ou utilizando o ambiente de desenvolvimento do Arduino.

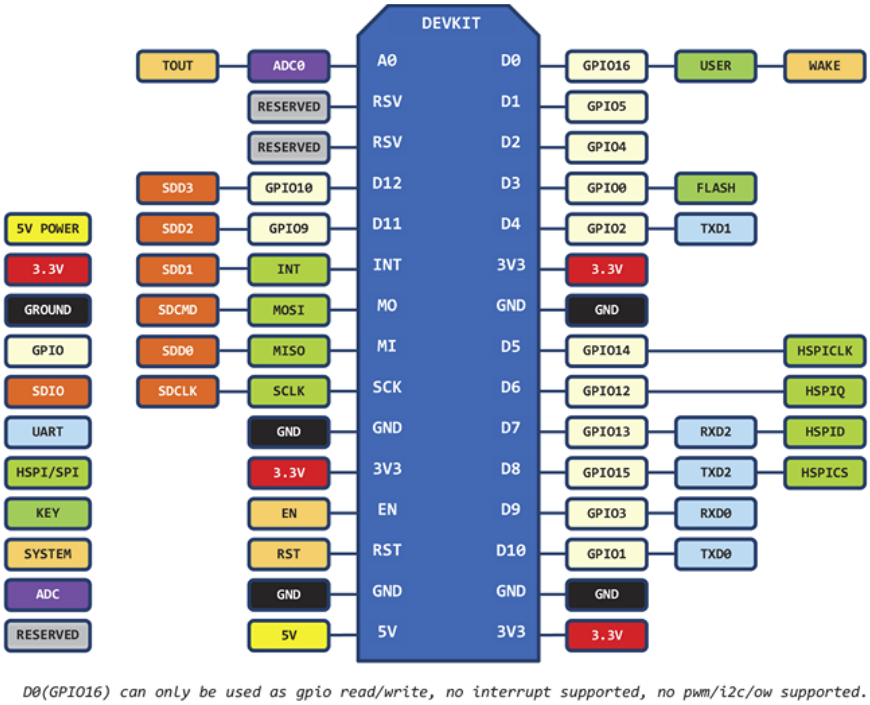
FIGURA 1 - MÓDULO ESP8266 NODEMCU WI-FI DEVKIT



FONTE: O Autor (2018)

A placa contém um módulo de programação e *debugging*,um regulador 3.3V, com saídas 5V ou 3.3V, um botão de *reset* e outro botão de *flash* (FIGURA 2).

FIGURA 2 - PINOUT ESP8266 NODEMCU WI-FI DEVKIT



FONTE: http://www.handsontec.com/pdf_learn/esp8266-V10.pdf

2.2. ESPRESSIF ESP8266

O ESP8266 (FIGURA 3) é um módulo microcontrolado de baixo custo e baixo consumo com capacidade de comunicação Wi-Fi desenvolvido pela Espressif. O módulo pode ser utilizado de maneira independente, mas normalmente é associado a uma plataforma contendo módulos de fonte e microcontrolador para rápida prototipagem.

O módulo conta com comunicação Wi-Fi 802.11 b/g/n com antena embutida. A capacidade de memória é de 32 kB de memória RAM e 4MB de memória flash.

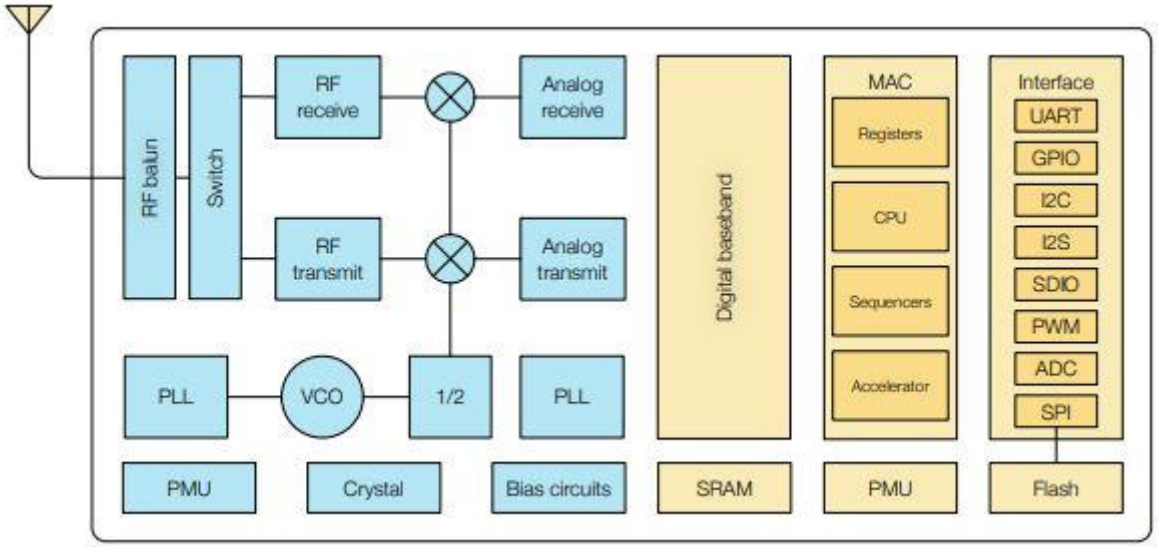
São disponibilizadas 10 portas GPIO e 1 ADC de 10 bits. Nesta versão, possui apenas um núcleo de processamento 32 bits, e dois modos de *clock*, 80 MHz ou 160 MHz.

FIGURA 3 - ESPRESSIF ESP8266

FONTE: https://tech.scargill.net/wp-content/uploads/2015/06/ESP12-E.jpg

Um diagrama de blocos de alto nível ilustrando a estrutura interna do microcontrolador, separando a parte digital e analógica pode ser observado na FIGURA 4

FIGURA 4- DIAGRAMA DE BLOCOS ESP8266



FONTE: Espressif ESP8266EX DataSheet (2018 p.06)

2.3. BANCO DE DADOS

Banco de dados consiste em uma coleção de dados que os relaciona e classifica para que sejam acessados eletronicamente. A estrutura e modelagem de um banco de dados define a relação entre os dados, auxiliando na sua contextualização e permitindo seu acesso de maneira eficiente.

Este tipo de estrutura se difere de estruturas de arquivos tradicionais, como arquivos de texto pois apresenta diversas características e funcionalidades que auxiliam na indexação e manutenção dos dados armazenados.

Para ambientes on-line, bancos de dados se mostram muito mais efetivos em prover ao usuário um acesso eficiente aos seus dados, com maior flexibilidade e confiabilidade.

Para administrar estes dados são utilizados sistemas de gestão de banco de dados, chamados de *database management system*, ou DBMS. O DBMS é responsável por gerenciar o acesso aos dados, permitindo a definição, recuperação e atualização de dados do banco.

Sistemas modernos de armazenamento em nuvem utilizam esquemas complexos de replicação de dados e sincronismo, para permitir que usuários de qualquer lugar do mundo possam acessar seus dados dinamicamente de forma eficaz.

Os bancos de dados mais amplamente utilizados se classificam primariamente em dois tipos, os relacionais e os não-relacionais.

Bancos de dados relacionais organizam os dados na forma de tabelas fixas. Utilizando esta estrutura é possível inter-relacionar os dados em diversas tabelas, garantindo assim a consistência dos dados. Um dado atualizado em uma das tabelas automaticamente refletirá em todos os outros elementos que apontam para ele. Este tipo de estrutura permite uma grande confiabilidade nos dados armazenados, porém exige um planejamento complexo e cuidadoso da estrutura do banco, pois é menos susceptível a mudanças e adaptações posteriores à sua criação.

Bancos de dados não-relacionais guardam os dados em um formato de arquivo dinâmico, geralmente JSON, que descreve um objeto através de chaves e valores. O acesso a dados deste tipo de arquivo utiliza a indexação das chaves de cada entidade. Estes índices também são definidos de maneira flexível através de arquivos de índices presentes na configuração do banco. Este tipo de estrutura de dados permite uma melhor distribuição dos dados no sistema de arquivos do servidor e simplifica a escalabilidade dos serviços que utilizam o banco.

Um dos maiores benefícios de bancos de dados não relacionais é a possibilidade de modificação da estrutura de dados conforme a demanda. Por exemplo, caso um cadastro de usuário passe a guardar informações de endereço que não estavam presentes anteriormente, todas as entidades que não apresentam esta informação respondem com o acesso a elas com um objeto de tipo nulo. Os novos cadastros que possuem este dado serão armazenados junto com os antigos, sem que isto afete os sistemas que já operavam sobre estas entidades. Estes bancos possuem uma consistência eventual dos dados armazenados, que significa que uma alteração em um dado irá eventualmente refletir em uma atualização do dado em outros locais.

Bancos não-relacionais normalmente são escolhidos para projetos pequenos ou startups de rápido crescimento, pois a estrutura dos dados não é totalmente definida, e requer constantes ajustes.

Porém, a eventualidade da consistência destes dados pode afetar a segurança do sistema, e, caso estes fatores não sejam tratados pelos sistemas que utilizam o banco, podem permitir que dados sejam duplicados ou desatualizados.

Nos casos citados, os benefícios no uso de bancos não-relacionais se sobrepõem aos prejuízos. Além disso, sistemas modernos de bancos de dados não relacionais possuem mecanismos que auxiliam na tratativa deste tipo de problema, melhorando a confiabilidade dos dados armazenados.

2.4. FIREBASE

O Google Firebase é uma plataforma de desenvolvimento *web* que apresenta diversos serviços essenciais para a criação de aplicações em *cloud*. As soluções integradas aceleram a velocidade de desenvolvimento de aplicações em geral, fornecendo ao usuário alta flexibilidade e escalabilidade.

A plataforma também apresenta planos gratuitos completos para que soluções possam ser desenvolvidas experimentalmente, passando a cobrar apenas quando seu uso é bastante elevado.

O Google Firebase permite que todo o ecossistema de aplicações *web* seja concentrado em um único serviço. O foco da plataforma é em providenciar ferramentas com alto poder de escalabilidade. Isso permite que pequenas empresas e startups forneçam seus serviços para qualquer quantidade de usuários automaticamente e em tempo real.

2.4.1. CLOUD FUNCTIONS

Soluções do tipo *functions*, como o AWS Lambda, Google Cloud Functions, ou o Firebase Functions apresentam uma metodologia de processamento de *back-end* sem a gestão de estados.

A requisição do cliente da aplicação corresponde à uma chamada de uma função presente na nuvem. Esta função executa seu processamento de maneira independente, sem haver uma necessidade de infra-estrutura complexa ou gestão de estados do servidor.

Na prática é como se cada requisição fosse encapsulada e processada em uma instância independente. Esse tipo de arquitetura também é conhecida como *stateless*.

Soluções do tipo *functions* são bastante adequadas para ambientes onde o servidor realiza consultas ou processamentos pontuais, que não dependem de requisições de outros usuários.

Nestes casos o *front-end* da aplicação pode ficar responsável pela gestão de estado do usuário. Caso exista a necessidade de manutenção do estado ou gestão de tarefas sequenciais em um ambiente de *functions*, recomenda-se utilizar ferramentas específicas para estes fins, ou utilizar entidades que armazenam o estado em um banco de dados.

2.4.2. CLOUD HOSTING

O Firebase Cloud Hosting apresenta uma solução de hospedagem de sites estáticos e dinâmicos, que propicia toda a infraestrutura necessária para gerenciar sites e aplicações *web*. Também estão disponíveis opções de controle de *cache* customizados para cada página, redirecionamento DNS automático, certificação *SSL* gratuita, redirecionamento de páginas simplificado e configurações avançadas de implantação ou *deploy*.

Os benefícios da utilização de uma solução de hospedagem ao contrário de hospedagens convencionais é principalmente a facilidade de *deploy*, métricas e integração com os demais sistemas, o que facilita a gestão da aplicação.

2.4.3. CLOUD FIRESTORE

Cloud Firestore é uma solução banco de dados não relacional em tempo real. Nesta solução é possível organizar os dados em coleções e documentos. Os documentos são entidades no banco de dados que possuem dados, e coleções. Coleções são entidades que possuem apenas documentos. Esta solução também apresenta a configuração de índices customizados, *queries*, notificações e integração com diversos sistemas de autenticação do usuário.

Os benefícios de utilizar um banco de dados não relacional estão principalmente ligados à flexibilidade e customização da estrutura do banco de acordo com a necessidade. Para a aplicação proposta é totalmente adequado, já que os dados do banco serão segregados por usuário, num cenário onde a

consistência eventual não afeta a usabilidade do sistema. Além disso, o tipo de dado sendo tratado não possui criticidade em questão de segurança.

2.5. BACK-END

O *back-end* da aplicação é a parte do software da aplicação que está no servidor. Nele são tratadas as requisições ao banco de dados, bem como a autenticação do usuário e processamento de dados.

2.5.1. JSON

O formato de arquivo de notação de objeto JavaScript, ou JSON é um formato de estruturação de dados minimalista focado em legibilidade e eficiência. O formato descreve objetos através de um padrão de combinações de chaves e valores.

Neste formato, todas as chaves são descritas por *strings*, e os valores são descritos na forma de seis tipos de dados: *string*, booleano, número, nulo, lista e objeto. Listas e objetos são composições dos demais tipos, onde listas possuem chaves enumeradas e objetos possuem combinações de chaves e valores. Ambas as chaves e objetos guardam valores de qualquer tipo.

As chaves são definidas como texto entre aspas. Para definir o valor relacionado à chave, utilizam-se dois pontos, seguidos pelo valor que se quer armazenar. Strings são caracteres alfanuméricos compreendidos por aspas. Booleanos são as palavras *true* ou *false*, sem aspas. Números são caracteres de números inteiros ou frações divididas por ponto, que podem ser seguidos por uma potência de base 10 denotada pela letra E, e não possuem aspas. Objetos são conjuntos de chaves e valores separados por vírgula e compreendidos por caracteres de chaves '{' e '}'. *Arrays* são conjuntos de valores separados por vírgulas e compreendidos por colchetes '[' e ']'.

Caracteres de formatação como *tabs*, espaços e novas linhas são ignorados. Para descrever estes caracteres especiais dentro de strings, é utilizada a barra invertida '\' seguida por um caractere de formatação.

Um exemplo de objeto descrito neste formato de arquivo está presente na FIGURA 5.

FIGURA 5 - OBJETO JSON

```json
{
    "somekey": "string value",
    "number": 3245.44,
    "bool": true,
    "array": [
        "first_value",
        "second_value",
        234.1,
        false
    ],
    "object": {
        "somekey": "value"
    }
}
```

FONTE: O Autor (2018)

Este padrão é amplamente utilizado como uma alternativa ao XML para a transmissão de dados, e é preferido sobre este por ser de mais simples leitura e conter menos informação redundante em sua descrição.

Os pontos positivos deste formato são sua flexibilidade e facilidade de interpretação humana. Algumas limitações do formato são a impossibilidade de referências internas de dados, *namespaces*, e metadados. Estes problemas são contornados com práticas como a adição de metadados como dados convencionais na estrutura e segregando a informação em objetos independentes de acordo com o contexto.

A principal facilidade do formato JSON é sua alta compatibilidade com objetos JavaScript, já que os objetos possuem a mesma estrutura. Interpretadores deste formato são amplamente encontrados em bibliotecas nas mais diversas linguagens de programação e plataformas de desenvolvimento. Por ser altamente flexível, é mais amplamente utilizado juntamente com linguagens de programação de tipagem dinâmica.

2.5.2. PROTOCOLO HTTP

HTTP é um protocolo de aplicação para comunicação de dados através de redes de computadores. Nele os dados são transmitidos no formato de texto, onde não é guardado estado a respeito de requisições anteriores.

O protocolo utiliza *sockets* TCP/IP para enviar e receber mensagens entre um cliente e um servidor. O formato geral do protocolo define campos de requisição, endereço, cabeçalho e corpo da mensagem. Sua formatação facilita bastante a inspeção humana das mensagens sendo enviadas e recebidas. O protocolo apresenta padrões de status de resposta de requisições, o que permite sua ampla utilização e interpretação por variados serviços e aplicações. A aplicação que utiliza o protocolo HTTP para a comunicação define o tipo de requisição e o formato de resposta. Algumas APIs e arquiteturas propõem formatos de utilização do protocolo HTTP para a comunicação de dados, por exemplo as arquiteturas SOAP e REST.

2.5.2.1. PROTOCOLO HTTPS

Os dados que trafegam através do protocolo HTTP são em formato de texto, e podem ser interpretados por qualquer usuário que tenha acesso a esta rede. Por isso, sistemas de segurança atuam sobre os dados transmitidos, principalmente o corpo da mensagem, para trafegar dados de maneira segura. O protocolo HTTPS é uma extensão do protocolo HTTP que criptografa os dados transmitidos, garantindo assim a segurança das informações trafegadas entre cliente e servidor.

A criptografia é bidirecional, e é utilizado um esquema de chaves que autentica o site sendo visitado, protegendo contra ataques do tipo "*man-in-the-middle*", um dos mais críticos para a segurança na rede. A autenticação do protocolo utiliza os protocolos de criptografia SSL e TLS para proteger os dados. A autenticidade da fonte da informação é verificada através da validação de chaves públicas, confirmadas por autoridades confiáveis de validação de certificado.

2.5.2.2. REST API

As mensagens enviadas através do protocolo HTTP(S) são de livre formatação, e são livremente definidas pelos desenvolvedores da aplicação. Um dos estilos de arquitetura utilizados para essa troca de mensagem é a Transferência de Estado Representacional, também conhecida como REST.

Nessa arquitetura, cada mensagem contém todo o dado necessário para compreender a requisição, seguindo a linha de comunicação *stateless* do protocolo HTTP.

A arquitetura abrange um conjunto de operações que define o tipo de operação a ser realizada. As rotas de destino estão sempre presentes no endereço da mensagem sendo enviada, que são a combinação da URL (destino) e URN (rota/recurso), em um endereço completo URI (identificador). Diversos frameworks de desenvolvimento *web front-end* e *back-end* apresentam APIs que fazem o redirecionamento das mensagens e rotas de maneira automática utilizando este padrão.

Esquemas que implementam uma arquitetura REST estritamente definida são conhecidos como RESTful. As APIs RESTful tratam coleções e elementos identificados através das rotas e apresentam métodos de comando padrões para o acesso dos dados. Os comandos mais comuns são "GET", "PUT", "POST" e "DELETE", para recuperar, substituir, criar e remover uma entidade do serviço, nesta ordem. A resposta pode estar nos formatos XML, HTML, JSON ou qualquer outro, porém, na *web* o formato JSON é mais amplamente utilizado.

2.5.3. NODE.JS

Node.js é um *runtime* de execução de código JavaScript de código aberto e multiplataforma fora do navegador. Seu foco é na criação de aplicações *web* como *back-end* do servidor. O ambiente permite que códigos e bibliotecas JavaScript diversas sejam utilizadas também na criação de aplicações como processamento de dados, aprendizado de máquina e gerenciamento de eventos e operações de entrada e saída, também conhecidas como *I/O operations*.

O código é executado em uma única *thread*, onde processamento concorrente é possível sem causar uma mudança de contexto do processador. Isso dá ao código uma imensa flexibilidade para atender centenas ou até milhares de operações concorrentes sem que a execução do código bloqueie suas operações.

Uma limitação deste tipo de arquitetura é a falta de capacidade de escalar o processo em diversos núcleos físicos do computador, porém existem algumas ferramentas que permitem que este problema seja contornado.

2.6. FRONT-END

O *front-end* da aplicação *web* é a parte do software que está no lado do usuário. Ela faz a tratativa dos dados recebidos pelo servidor e exibe as telas na interface. O *front-end* também é responsável pelo estado da aplicação e em fazer requisições ao banco de dados. A separação de funcionalidades entre o *back-end* e o *front-end* depende da arquitetura da aplicação.

Normalmente, principalmente quando se trata de dispositivos móveis, o *back-end* fica com a responsabilidade de tratar os dados e fornecer APIs. Desta forma o *front-end* tem um enfoque maior em consumir os dados e apresentá-los ao usuário com uma boa usabilidade.

2.6.1. HTML

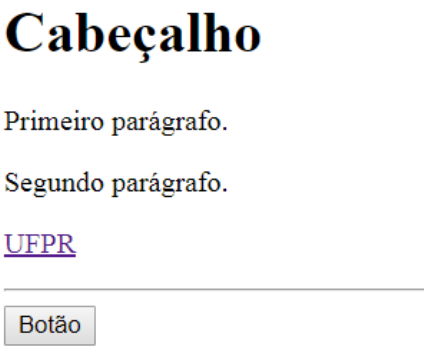
HTML, ou Hypertext Markup Language é o padrão *web* de descrição de aplicações. O padrão, com sintaxe igual ao padrão XML, é composto de identificadores compreendidos por por sinais de maior '<' e menor '>', possuindo uma identificação, identificadores de metadados e informação. Cada identificador possui um escopo, cujo fim é delimitado por um fechamento de escopo definindo o identificador que o abriu precedido pelo caractere de barra '/'. Um exemplo da sintaxe do padrão está presente na FIGURA 6, já o resultado deste exemplo está presente na FIGURA 7

FIGURA 6 - DOCUMENTO HTML

```
<!DOCTYPE html>
<html>
    <body>
        <h1>Cabeçalho</h1>
        <p>Primeiro parágrafo.</p>
        <p>Segundo parágrafo.</p>
        <p><a href="http://eletrica.ufpr.br/p/">UFPR</a></p>
        <hr>
        <button>Botão</button>
    </body>
</html>
```

FONTE: O Autor (2018)

FIGURA 7 - PÁGINA HTML



FONTE: O Autor (2018)

A renderização da aplicação é feita através do motor de renderização do navegador, que pode implementar elementos com sua própria aparência visual. Normalmente são utilizadas combinações de elementos para criar determinados padrões visuais em páginas *web*. Para configurar a apresentação destes elementos são associadas a eles classes *css*.

2.6.2. CSS

A linguagem de folhas de estilo em cascata ou *cascading style sheets* descreve a formatação de apresentação de um elemento HTML. Através destas folhas de estilo define-se o estilo de apresentação de elementos *web* como cor, tamanho, sombra e demais propriedades.

Sua sintaxe é similar a objetos JSON, porém com algumas características específicas, como a ausência de asteriscos e a precedência pelo tipo de *tag* HTML a que a classe se refere, ou nome da classe. A figura a seguir apresenta um exemplo de formatação do elemento de cabeçalho tipo h1. Um exemplo de classe *css* está presente na FIGURA 8.

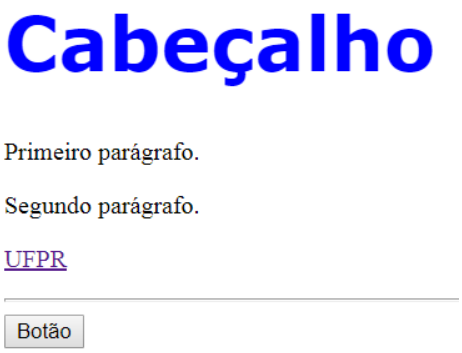
FIGURA 8 - CLASSE CSS

```
h1 {
    color: blue;
    font-family: verdana;
    font-size: 300%;
}
```

FONTE: O Autor (2018)

Existem centenas de propriedades diferentes que podem ser configuradas na classe *css*, o que permite uma grande liberdade na criação e configuração dos elementos na página *web*. Cada navegador é responsável por interpretar estes atributos e definir seu efeito visual, seguido os padrões de definição. A classe presente na *figura anterior*, aplicada ao documento de exemplo demonstrado anteriormente afeta sua aparência como é possível observar na FIGURA 9.

FIGURA 9 - FORMATAÇÃO DO DOCUMENTO



FONTE: O Autor (2018)

2.6.3. VUE.JS

Vue.js é um framework de desenvolvimento *front-end web* progressivo. O foco da estrutura do framework é prover ao projeto flexibilidade para um desenvolvimento incremental. Nele, itens da tela podem ser descritos na forma de componentes, que podem conter outros componentes, o que permite a criação de bibliotecas padronizadas e customizáveis. Todo o componente, incluindo a parte visual e scripts são contidas no mesmo arquivo, com extensão ".vue", e permitem uma melhor manutenção e descrição das telas e componentes.

Alguns benefícios da utilização deste framework em favor de outros frameworks como o React e Angular são a simplicidade, flexibilidade, facilidade de aprendizado e tempo de aprendizado. O framework apresenta uma infinidade de ferramentas e maneiras de realizar a mesma aplicação. Além disso, conta com um vasto ecossistema de componentes, bibliotecas e exemplos disponíveis *on-line* para consulta e utilização.

2.6.4. JAVASCRIPT

JavaScript é uma linguagem de programação interpretada de alto nível desenvolvida em 1995 por Brendan Eich. Desde sua concepção, o foco da linguagem foi o de ser implantada como código em navegadores *web*. Algumas das principais características da linguagem são tipagem dinâmica e suporte à múltiplos paradigmas, principalmente o paradigma de programação funcional.

Um dos grandes benefícios do JavaScript é a gama de caminhos possíveis para se alcançar uma mesma funcionalidade no código. Isso se mostra na prática como uma forma de acelerar o desenvolvimento, mas que pode gerar diversos problemas de legibilidade e manutenção. Objetos são flexíveis, assim como na linguagem Python, e podem ser alterados em tempo de execução. Código pode ser interpretado como texto para ser executado quando necessário, o que é absolutamente crucial para aplicações *web*, onde o carregamento das páginas acontece de maneira dinâmica.

Em sua utilização, tanto para aplicações *front-end* quanto *back-end*, o JavaScript apresenta diversos gerenciadores de pacotes. Com estes gerenciadores é possível instalar e utilizar bibliotecas de terceiros com extrema facilidade. Sua utilização é similar à instalações de aplicações utilizando gerenciadores de programas em ambiente Linux, com o benefício de que estes pacotes podem ser instalados tanto globalmente quanto especificamente para um projeto. Eles também separam a biblioteca da implementação do projeto, e permitem que todas as dependências sejam instaladas com um único comando.

Enquanto por um lado a linguagem fornece ao usuário enorme liberdade, por outro permite que o código seja estruturado de maneira bastante confusa e complexa, principalmente quando não há uma padronização no projeto. Outro problema também pode surgir quando o projeto implementa características conflitantes de diferentes paradigmas, o que é comum em linguagens flexíveis, principalmente nas de tipagem dinâmica. Outros problemas estão relacionados à segurança das aplicações e suporte de funcionalidades nos navegadores, já que cada navegador implementa suas próprias funcionalidades de maneira independente.

2.6.5. APLICAÇÃO WEB PROGRESSIVA - PWA

Aplicações *web* progressivas, também conhecidas como PWA são, em resumo, páginas *web* que podem ser utilizadas mesmo quando o usuário não possui acesso à internet. Neste tipo de aplicação, dados do usuário são salvos localmente no aparelho na forma de *cache*. Desta forma o usuário apenas receberá uma versão atualizada da página quando ela se modificar.

Os sistemas operacionais e navegadores de dispositivos móveis podem identificar sites PWA e apresentar ao usuário a opção de salvar esta página como uma aplicação no seu dispositivo. Quando isso é feito a experiência do usuário com a aplicação fica muito similar à utilização de aplicações nativas. O ícone da aplicação é apresentado no aparelho como o ícone de um *app*.

Alguns dos benefícios deste tipo de aplicação são a experiência melhorada da utilização, facilidade de acesso. Outros benefícios são as atualizações instantâneas, já que o site pode ser atualizado assim que o usuário abre a página, e o fato de não ser uma aplicação instalada no aparelho, apenas uma página *web* em *cache*.

2.7. VERSIONAMENTO

Softwares de controle de versão são cruciais para o desenvolvimento de softwares complexos, principalmente quando são desenvolvidos por mais de uma pessoa. Utilizando versionamento é possível verificar o estado do projeto em diversas etapas de implementação. Também é facilitado que o software seja revertido para um estado estável em caso de erro ou falha.

A forma mais básica de versionamento manual é salvando os arquivos do projeto em diversos estágios de desenvolvimento. Utilizando softwares de versionamento, além de uma maior eficiência para os softwares que guardam as modificações realizadas no arquivo, também é melhorada a integração entre versões, tagueamento e ramificação de versões, gerenciamento de grandes arquivos e melhora na rastreabilidade de modificações que possam introduzir falhas de segurança ou *bugs*.

O versionamento é adequado para qualquer tipo de projeto, mesmo que não seja código. Os sistemas de versionamento mais recentes tratam de maneira

eficaz com arquivos de texto, mas também existem extensões para outros tipos de arquivos. Alguns softwares de edição de imagem e vídeo também apresentam soluções customizadas específicas. Dentre os sistemas de versionamento mais comuns estão os softwares Git, Mercurial e SVN.

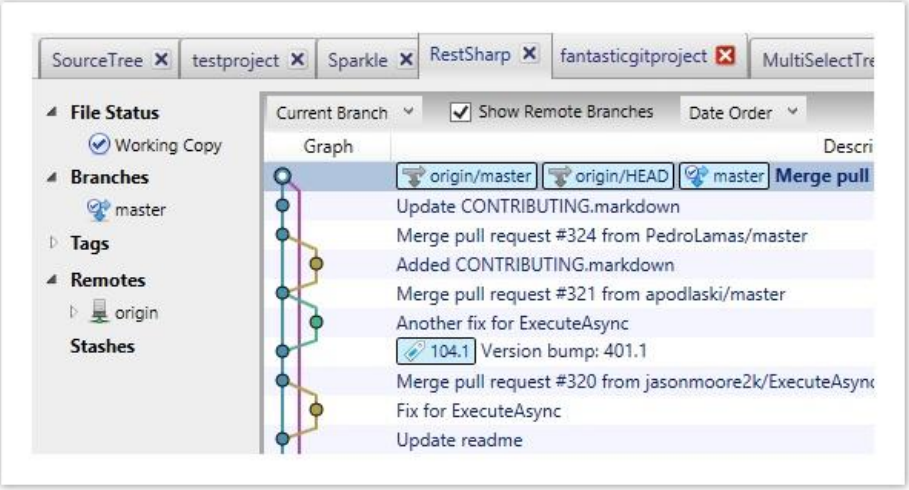
2.7.1. GIT

Git é o sistema de controle de versão de código gratuito e de código aberto distribuído sob a licença GNU. mais utilizado em projetos de pequeno e médio porte. Ele foi criado em 2005 para gerir o desenvolvimento do núcleo Linux. Nele, toda máquina que baixa o repositório para uso local possui o pacote completo das versões do projeto. Sendo assim, mesmo que ocorra pane em alguma máquina ou no servidor, o projeto e todas as suas versões pode ser recuperado a partir de qualquer máquina que o possua.

No Git, cada etapa de modificação de código pode ser salva em um novo passo chamado de *commit*. Para otimizar o espaço salvo do repositório são salvas apenas as modificações realizadas em cada arquivo, e estas modificações são compactadas em disco. Cada *commit* evidencia todas as mudanças em todos os arquivos que foram modificados nele. O *commit* recebe uma *hash* única, o nome do autor e um texto escrito pelo autor. Também é possível aplicar etiquetas ou *tags* nos *commits*, que podem ser usadas para diversas finalidades. O sistema também apresenta a funcionalidade de ramos, ou *branches*, que partem de uma determinada versão do código.

Em uma *branch* específica o código pode ser modificado de maneira independente de todas as outras, principalmente a fim de evitar conflitos durante o desenvolvimento. Após uma funcionalidade ter sido implementada, é possível aplicar todas estas modificações em outra *branch* de uma única vez, assim é possível garantir por exemplo uma *branch* onde existem apenas versões estáveis do código. Um exemplo visual de um projeto desenvolvido utilizando esta ferramenta está presente na FIGURA 10.

FIGURA 10 - EXEMPLO DE USO DO GIT



FONTE: https://3kllhk1ibq34qk6sp3bhtox1-wpengine.netdna-ssl.com/wp-content/uploads/visualize-original-windows.jpg

3. DESENVOLVIMENTO

Os requisitos mínimos da solução são definidos através do que se estima como a necessidade do usuário alvo. Neste caso, considerando principalmente usuários que pretendem utilizar a solução para perder peso, são definidos os requisitos de usabilidade e interação. Estes requisitos são então estendidos para requisitos de *software*, *firmware* e *hardware* que satisfazem suas necessidades mínimas.

Dentro da rotina do usuário é mais adequado que as pesagens sejam realizadas diariamente, aproximadamente no mesmo horário, no período da manhã. As necessidades básicas do usuário estão descritas a seguir:

- Configurar o equipamento para utilizar a rede Wi-Fi local
- Configurar usuário e associar equipamento
- Realizar medidas de peso
- Acompanhar histórico de medidas

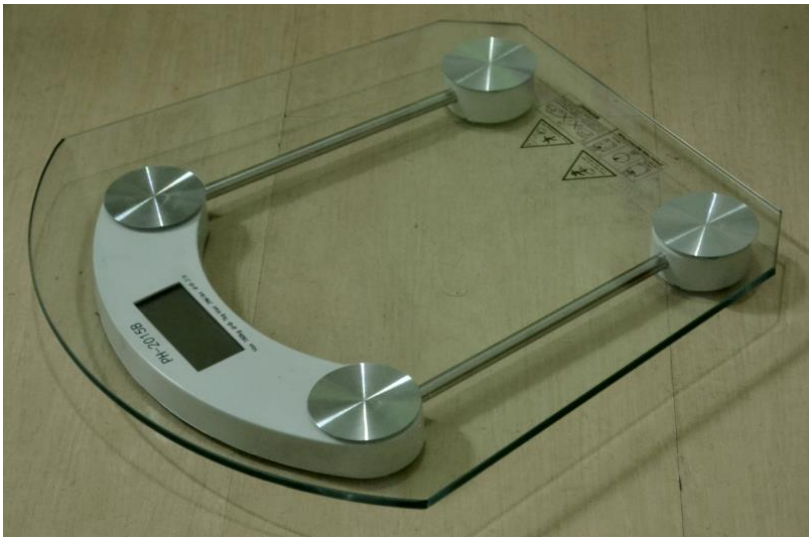
Na configuração do equipamento deve ser possível definir quais das redes locais o usuário gostaria de utilizar para o equipamento, informando SSID e senha.

A configuração do usuário deve permitir a configuração de qual equipamento será utilizado na medida e dados do usuário. A realização das medidas deve ser simples e transparente. O acompanhamento das medidas deve ser claro e objetivo, na forma gráfica.

3.1. HARDWARE

Para coletar os dados de medida de peso é utilizada uma balança convencional modelo PH-2015B (FIGURA 11). Dela são extraídos dois pontos de derivação, cuja tensão é proporcional ao peso medido. Esta razão é ajustada em hardware para que seja possível a coleta da medida.

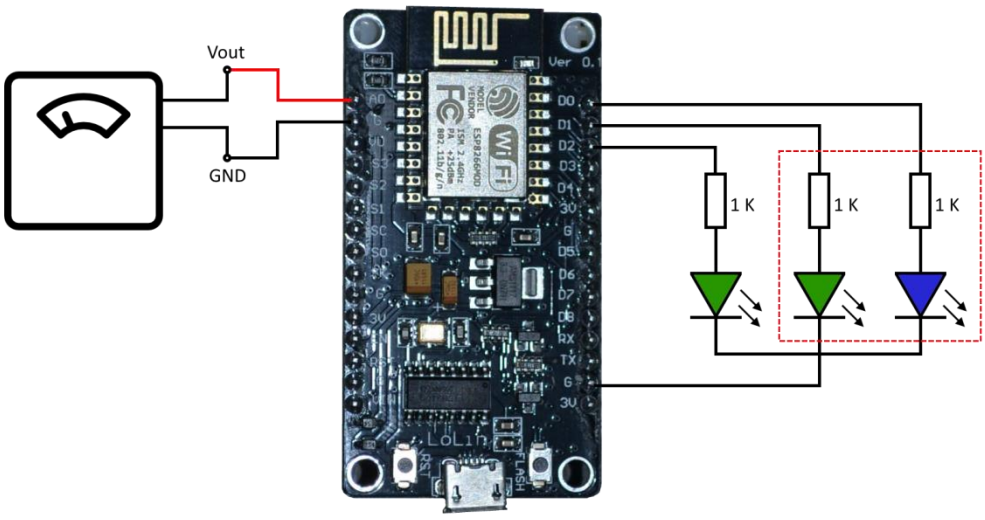
FIGURA 11 - BALANÇA PH-2015B



FONTE: O Autor (2018)

A topologia da montagem do equipamento presente na FIGURA 12 é aplicada de forma a coletar os dados gerados pela balança para transmissão e informar o status do equipamento ao usuário.

FIGURA 12 - TOPOLOGIA DO EQUIPAMENTO



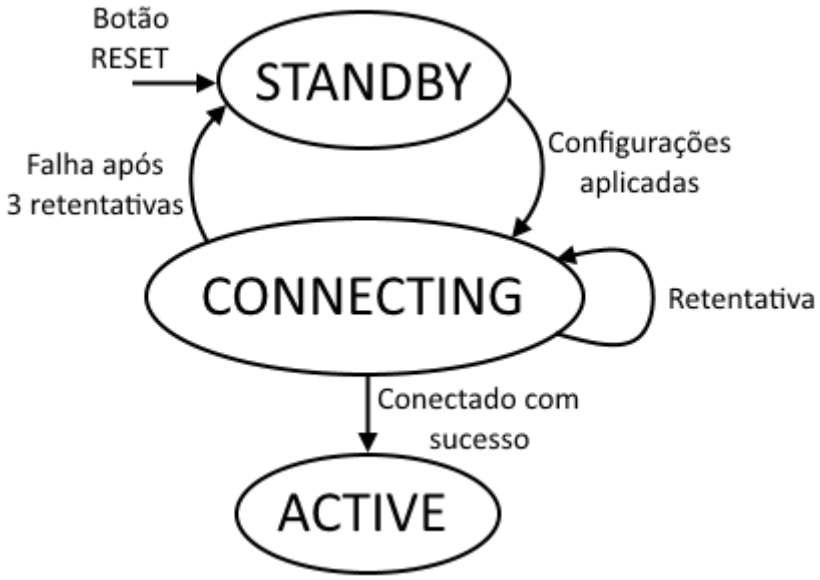
FONTE: O Autor (2018)

O sinal analógico de saída da balança é lido através do conversor analógico-digital presente na placa, que então é lido e enviado através da rede Wi-Fi. Para simplificar a solução e diminuir custos, as responsabilidades do módulo são reduzidas. Isso possibilita que futuramente seja utilizado um módulo ainda mais simples para o envio das medidas.

3.2. FIRMWARE

O firmware permite que o usuário configure a rede em que o equipamento irá se conectar. Quando estiver conectado, permite que as medidas sejam realizadas. Para isso é necessário um controle de status, que será apresentado através dos dois LEDs do equipamento. Aqui são utilizados três status, STANDBY, CONNECTING e ACTIVE. O primeiro LED verde estará aceso sempre enquanto o aparelho estiver ligado. A máquina de estados está presente na imagem a seguir (FIGURA 13)

FIGURA 13 - MÁQUINA DE ESTADOS DO CONTROLADOR



FONTE: O Autor (2018)

O status de STANDBY representa quando o equipamento não está conectado a nenhuma rede Wi-Fi com acesso à internet. Nele, o equipamento entra no modo ponto de acesso, com uma rede padrão, que permitirá que o usuário se conecte a ele. Neste status, o LED de status estará aceso na cor azul. Para acessar o equipamento utiliza-se a rede com SSID "iot-scale", que não necessita de senha. Isso pode ser feito utilizando *smartphone* ou outro dispositivo com acesso Wi-Fi. Ao acessar o equipamento, o usuário é prontificado com uma tela contendo a identificação do equipamento, uma lista de redes disponíveis e um campo para a senha desta rede

Após a configuração de uma rede, o aparelho entrará no modo CONNECTING. Nele o ponto de acesso será desligado, e o aparelho tentará se conectar com a rede configurada. Durante as tentativas de conexão, o LED de status piscará na cor azul. Este será sempre o status inicial do aparelho quando for ligado, caso exista uma rede configurada.

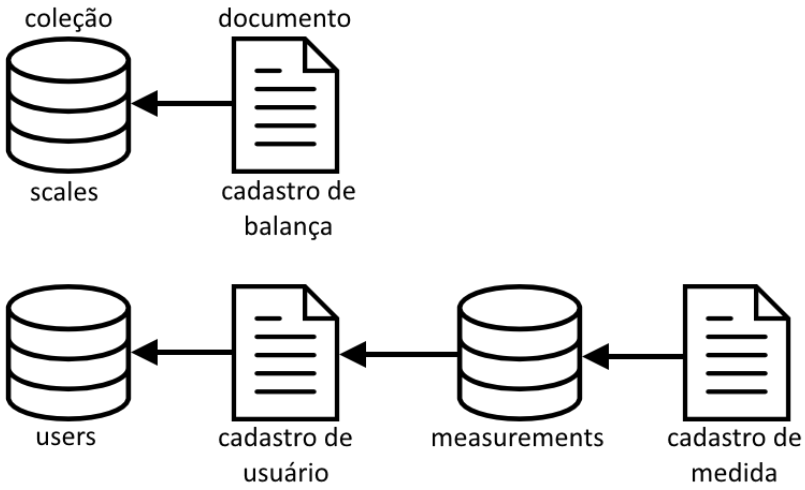
Caso após três tentativas o aparelho não consiga se conectar, retornará ao estado STANDBY para uma nova configuração. Também é possível forçar o aparelho a ir para o status de STANDBY pressionando o botão RESET. Caso o aparelho consiga se conectar com sucesso a uma rede Wi-Fi, passará para o status ACTIVE, acendendo o LED de status na cor verde. Neste status será possível realizar medidas. Para isto, basta subir na balança.

Neste momento o ESP8266 irá realizar uma medida utilizando o conversor analógico-digital. Quando a medida for realizada, será enviado ao servidor uma mensagem contendo o peso medido e a identificação da balança que realizou a medida.

3.3. ESTRUTURA DO BANCO DE DADOS

O banco de dados (FIGURA 14) foi estruturado de forma a conter três tipos de informação. O cadastro de um usuário, o cadastro de uma balança e as medidas realizadas.

FIGURA 14 - ESTRUTURA DO BANCO DE DADOS



FONTE: O Autor (2018)

As balanças são cadastradas em uma coleção independente. Nesta coleção estão todas as balanças já cadastradas, de acordo com o seu identificador único. O identificador único é composto por 4 letras, um traço '-' e 4 números.

O índice dos documentos que contém o cadastro é o identificador único em si, o que acelera bastante a recuperação dos dados da balança. Neste cadastro está presente o identificador único do usuário para o qual a balança está cadastrada e uma referência do banco de dados ao documento do usuário.

Os usuários são cadastrados numa coleção de usuários. Como podem existir mais do que um usuário com o mesmo nome, os documentos desta coleção possuem identificadores únicos aleatórios. Poderíamos utilizar o *e-mail* ou CPF do

usuário para definir o cadastro, porém isso não permitiria a troca de email, e seria necessário pedir o dado de CPF no momento do cadastro.

As medidas referentes a um usuário específico estão presentes em uma coleção de medidas dentro do documento do usuário. Sendo assim, recuperar todas as medidas realizadas se torna uma tarefa trivial, tanto para o DBMS quanto para o *back-end* da aplicação. No documento de uma medida estão presentes o identificador único do equipamento, o *timestamp* da medida e o valor de peso medido.

3.4. CONFIGURAÇÃO DO BACK-END

O *back-end* da aplicação é minimalista, e trata principalmente das requisições para o banco de dados. Existem três tipos de requisições que são tratadas, as requisições do tipo "usuário", "balança" e "medida". Seguindo o modelo REST, é proporcionada uma API com métodos para a requisição e criação de dados no banco para os três tipos de entidades descritos.

O QUADRO 2 apresenta todas as chamadas implementadas seguidas por seu endereço, comando e suas funcionalidades. Em seguida cada chamada é descrita e exemplificada individualmente.

QUADRO 2 - API DA APLICAÇÃO

Nome	Endereço	Comando	Função
Recuperar Usuário	/user/[userId]	GET	Retorna dados do usuário
Recuperar Medidas	/measurement/[userId]	GET	Retorna medidas do usuário
Recuperar Balança	/scale/[scaleId]	GET	Retorna dados da balança
Cadastrar Usuário	/user/	POST	Cria cadastro de usuário
Cadastrar Medida	/measurement/	POST	Cria cadastro de medida
Cadastrar Balança	/scale/	POST	Cria cadastro de balança

FONTE: O Autor (2018)

3.4.1. RECUPERAR USUÁRIO

Este comando tem o formato "[URL]/user/[userId]", onde "URL" é o endereço que hospeda a aplicação, e "userId" é a chave identificadora única do usuário cadastrado. Esta chamada retornará todos os dados presentes no cadastro do usuário, em formato JSON, assim como presente no banco de dados. Uma chamada à um usuário inexistente retornará um documento em branco. A descrição da resposta a esta chamada está presente no QUADRO 3.

QUADRO 3 - RESPOSTA DA REQUISIÇÃO DE RECUPERAR USUÁRIO

Chave	Tipo	Descrição
active	booleano	Usuário está ativo na plataforma
sex	*string*	Sexo do usuário, pode ser "male" ou "female"
name	*string*	Nome do usuário
scales	lista	Lista conténdo strings representando a identificação de cada balança associada à este usuário
height	número	Altura do usuário em centímetros
birthdate	*string*	Data de nascimento do usuário no formato "dd/mm/aaaa"
email	*string*	*E-mail* do usuário

FONTE: O Autor (2018)

3.4.2. RECUPERAR MEDIDAS

Este comando tem o formato "[URL]/measurement/[userId]", onde "URL" é o endereço que hospeda a aplicação, e "userId" é a chave identificadora única do usuário cadastrado. Esta chamada retornará um objeto contendo uma lista com chave "weightData" que possui todas as medidas do usuário em formato JSON. Uma chamada à um usuário inexistente retornará uma lista vazia. A descrição da resposta a esta chamada está presente no QUADRO 4.

QUADRO 4 - RESPOSTA DA REQUISIÇÃO DE RECUPERAR MEDIDAS

Chave	Tipo	Descrição
weightData	lista	Lista contendo objetos medidas realizadas pelo usuário
scale	*string*	Identificador único da balança que realizou a medida
weight	número	Peso medido
timestamp	objeto	Objeto de timestamp contendo o momento em que a medida foi realizada

FONTE: O Autor (2018)

3.4.3. RECUPERAR BALANÇA

Este comando tem o formato "[URL]/scale/[scaleId]", onde "URL" é o endereço que hospeda a aplicação, e "scaleId" é a chave única identificadora da balança cadastrada. Esta chamada retornará todos os dados presentes no cadastro da balança em formato JSON. Uma chamada à uma balança inexistente retornará uma lista vazia. A descrição da resposta a esta chamada está presente no QUADRO 5.

QUADRO 5 - RESPOSTA DA REQUISIÇÃO DE RECUPERAR BALANÇA

Chave	Tipo	Descrição
userId	*string*	Identificador único do usuário utilzando esta balança
user	objeto	Objeto do banco de dados referente ao documento que contém os dados do usuário

FONTE: O Autor (2018)

3.4.4. CADASTRAR USUÁRIO

Este comando tem o formato "[URL]/user/", onde "URL" é o endereço que hospeda a aplicação. Esta chamada envia os dados do usuário no corpo da mensagem HTTP, no formato JSON, e registra um novo usuário utilizando estes dados.

O servidor verifica apenas as informações mínimas para a criação de cadastro, mas também é possível salvar mais informações caso necessário, dada a natureza da estrutura do banco de dados não-relacional sendo utilizado.

Caso alguma dessas informações não tenha sido definida no corpo da mensagem, o servidor retornará uma mensagem de erro. Caso o cadastro seja realizado com sucesso, o servidor retornará o identificador único do usuário no banco de dados. A descrição do corpo da chamada está presente no QUADRO X.

QUADRO 6 - CORPO DA REQUISIÇÃO DE CADASTRAR USUÁRIO

Chave	Tipo	Descrição
sex	*string*	Sexo do usuário, pode ser "male" ou "female"
name	*string*	Nome do usuário
scales	lista	Lista conténdo strings representando a identificação de cada balança associada à este usuário
height	número	Altura do usuário em centímetros
birthdate	*string*	Data de nascimento do usuário no formato "dd/mm/aaaa"
email	*string*	*E-mail* do usuário

FONTE: O Autor (2018)

3.4.5. CADASTRAR MEDIDA

Este comando tem o formato "[URL]/measurement/", onde "URL" é o endereço que hospeda a aplicação. Esta chamada envia os dados da medida no corpo da mensagem HTTP, no formato JSON, e registra uma nova medida utilizando estes dados. A mensagem contém o peso medido e a identificação da balança.

O servidor será responsável por inserir a informação de *timestamp*. Para salvar a medida, o servidor irá acessar o dado da balança e recuperará a informação de qual usuário a está utilizando. Depois disso, irá utilizar a informação do usuário para salvar o documento de medida na coleção de medidas deste usuário.

Caso alguma dessas informações não tenha sido definida no corpo da mensagem, o servidor retornará uma mensagem de erro. Caso o cadastro seja realizado com sucesso, o servidor retornará o identificador único da nova medida no banco de dados. A descrição do corpo da chamada está presente no QUADRO X.

QUADRO 7 - CORPO DA REQUISIÇÃO DE CADASTRAR MEDIDA

Chave	Tipo	Descrição
weight	número	Peso medido
scale	*string*	Identificador único da balança que realizou a medida

FONTE: O Autor (2018)

3.4.6. CADASTRAR BALANÇA

Este comando tem o formato "[URL]/scale/", onde "URL" é o endereço que hospeda a aplicação. Esta chamada envia os dados da balança no corpo da mensagem HTTP, no formato JSON, e registra uma nova balança utilizando estes dados. A mensagem contém o identificador único do usuário e a identificação da balança.

Caso a balança já tenha sido cadastrada anteriormente, o comando irá removê-la da lista de balanças cadastradas do usuário original, em seguida irá cadastrar a balança no novo usuário, atualizando a informação do usuário no cadastro.

Na hipótese de que alguma dessas informações não tenha sido definida no corpo da mensagem, o servidor retornará uma mensagem de erro. Caso o cadastro seja realizado com sucesso, o servidor retornará o identificador único da nova balança no banco de dados. A descrição do corpo da chamada está presente no QUADRO 7.

QUADRO 8 - CORPO DA REQUISIÇÃO DE CADASTRAR BALANÇA

Chave	Tipo	Descrição
user	*string*	Usuário que possui a balança
scale	*string*	Identificador único da balança que realizou a medida

FONTE: O Autor (2018)

3.5. CONFIGURAÇÃO DO FRONT-END

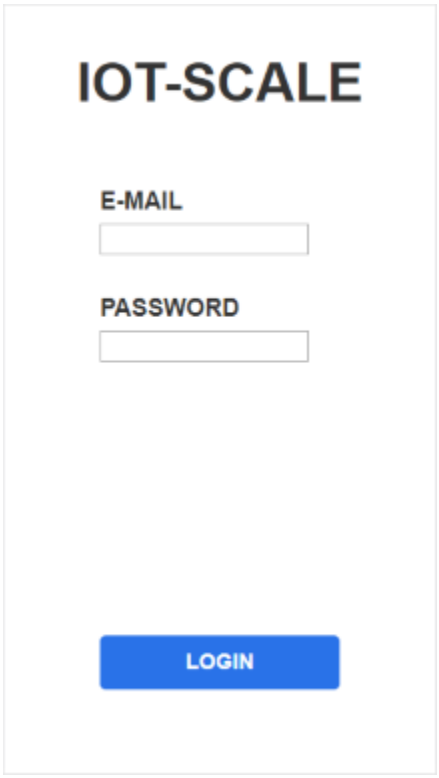
3.5.1. APLICAÇÃO

A aplicação, implementada utilizando o framework Vue.js, foi criada para permitir uma fácil manutenção e atualização. De acordo com os requisitos, o usuário fará *login* utilizando senha e *e-mail*, para então ir para o menu principal, onde terá acesso ao seu histórico de medidas. No histórico, é apresentado o gráfico das últimas medidas.

3.5.1.1. LOGIN

Na tela inicial da aplicação o usuário será prontificado com um formulário de *login* (FIGURA 15). Para realizar o *login*, o usuário deverá entrar com suas credenciais de e-mail e senha. Através destes dados, a aplicação saberá qual usuário está conectado, podendo assim acessar os dados referentes a este usuário.

FIGURA 15 - TELA DE LOGIN

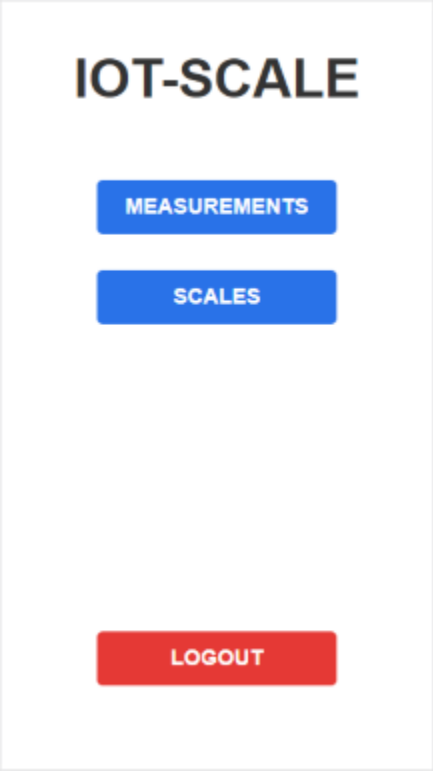


FONTE: O Autor (2018)

3.5.1.2. MENU PRINCIPAL

Após o *login*, no menu principal (FIGURA 16), é possível escolher entre configurar a balança ou visualizar as medidas. Nesta tela também é possível fazer *logout*.

FIGURA 16 - TELA DO MENU PRINCIPAL

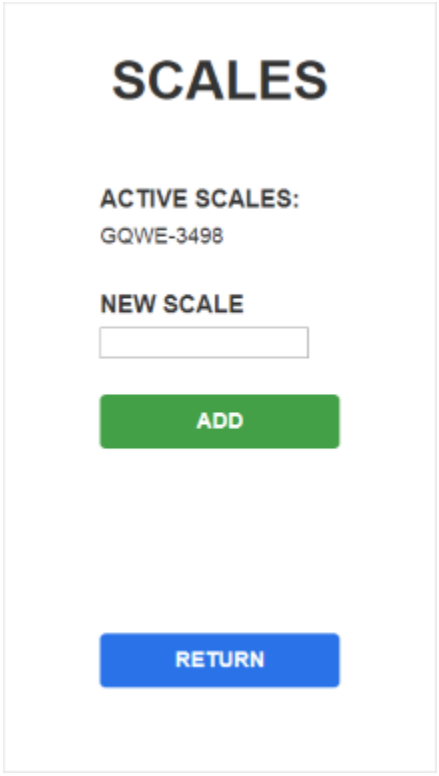


FONTE: O Autor (2018)

3.5.1.3. CONFIGURAÇÃO DE BALANÇA

Na tela de configuração de balança (FIGURA 17), o usuário poderá entrar com o identificador único do aparelho. A configuração não exige verificação, e é transparente para o usuário. Nesta tela também é possível visualizar todas as balanças já configuradas para utilização. Caso mais de uma pessoa utilize a balança, antes de realizar a medida o usuário precisará acessar esta tela para configurar a balança a seu próprio usuário.

FIGURA 17 - TELA DE CONFIGURAÇÃO DE BALANÇA



FONTE: O Autor (2018)

3.5.1.4. ACOMPANHAMENTO DE MEDIDAS

Ao acessar a tela de acompanhamento de medidas (FIGURA 18), o usuário pode visualizar as últimas medidas. O gráfico apresenta até uma medida por dia, e para os dias em que nenhuma medida foi realizada, o valor é extrapolado linearmente combinando as medidas adjacentes. Por padrão, é possível visualizar as medidas dos últimos 7 dias.

FIGURA 18 - TELA DE ACOMPANHAMENTO DE MEDIDAS

FONTE: O Autor (2018)

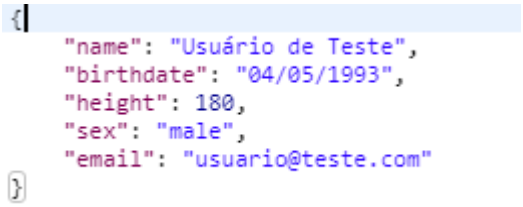
4. TESTES

A API da solução foi testada utilizando o software Postman, que simula requisições REST do cliente. O software permite a customização do corpo da mensagem HTTP enviada, e permite uma fácil visualização da resposta.

Os testes foram realizados verificando a mensagem retornada de cada uma das requisições presentes na API, tanto com o servidor rodando localmente quanto no serviço da nuvem. As mensagens e respostas utilizadas nos testes estão presentes a seguir.

Inicialmente, é cadastrado um usuário, para isso é chamado o comando de criação de usuário passando os parâmetros presentes na FIGURA 19.

FIGURA 19 - PARÂMETROS PARA CRIAÇÃO DE USUÁRIO

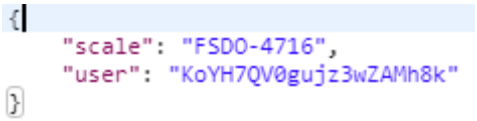


FONTE: O Autor (2018)

Verifica-se que a resposta é a chave do novo usuário no banco de dados "KoYH7QV0gujz3wZAMh8k".

Em seguida, utiliza-se esta chave para registrar uma nova balança a este usuário (FIGURA 20)

FIGURA 20 - PARÂMETROS PARA CRIAÇÃO DE BALANÇA

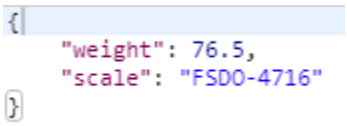


FONTE: O Autor (2018)

Verifica-se que a resposta é a chave da nova balança criada no banco de dados, "FSDO-4716".

Em seguida, utiliza-se esta chave para criar uma nova entrada de medida de peso (FIGURA 21).

FIGURA 21 - PARÂMETROS PARA CRIAÇÃO DE MEDIDA DE PESO



FONTE: O Autor (2018)

Verifica-se que a resposta é a chave da nova entrada de medida de peso para este usuário, " udTBxsYq1OxDBlw8BeYx". Este processo é repetido mais duas vezes com valores aleatórios para aumentar a coleção de medidas realizadas.

Em seguida, são requisitados os dados do usuário utilizando seu identificador único (FIGURA 22).

FIGURA 22 - DADOS DO USUÁRIO CADASTRADO


```
{
    "height": 180,
    "birthdate": "04/05/1993",
    "active": true,
    "sex": "male",
    "email": "usuario@teste.com",
    "name": "Usuário de Teste",
    "scales": [
        "FSDO-4716"
    ]
}
```


FONTE: O Autor (2018)

Verifica-se que o usuário está corretamente cadastrado, e a balança cadastrada consta em seu registro.

Então, são requisitados os dados de medida deste mesmo usuário (FIGURA 23).

FIGURA 23 - DADOS DE MEDIDA DO USUÁRIO


```
{
    "weightData": [
        {
            "timestamp": {
                "_seconds": 1543210855,
                "_nanoseconds": 993000000
            },
            "scale": "FSDO-4716",
            "weight": 78.2
        },
        {
            "scale": "FSDO-4716",
            "weight": 74.9,
            "timestamp": {
                "_seconds": 1543210849,
                "_nanoseconds": 529000000
            }
        },
        {
            "scale": "FSDO-4716",
            "weight": 76.5,
            "timestamp": {
                "_seconds": 1543210643,
                "_nanoseconds": 120000000
            }
        }
    ]
}
```


FONTE: O Autor (2018)

Verifica-se que os dados de medida estão presentes de acordo com o esperado. Não é necessária a verificação dos dados da balança, pois devem estar corretos para que o registro de medidas funcione corretamente.

Já a integração do sistema com a balança apresentou severas instabilidades. Então, para poder prosseguir com os testes, foi utilizado um potenciômetro regulando a entrada de tensão do conversor analógico-digital. Para os testes foi criado o cadastro de um novo usuário, e, após a coleta de dados, foram verificados o gráfico e as medidas salvas no banco de dados (FIGURA 24).

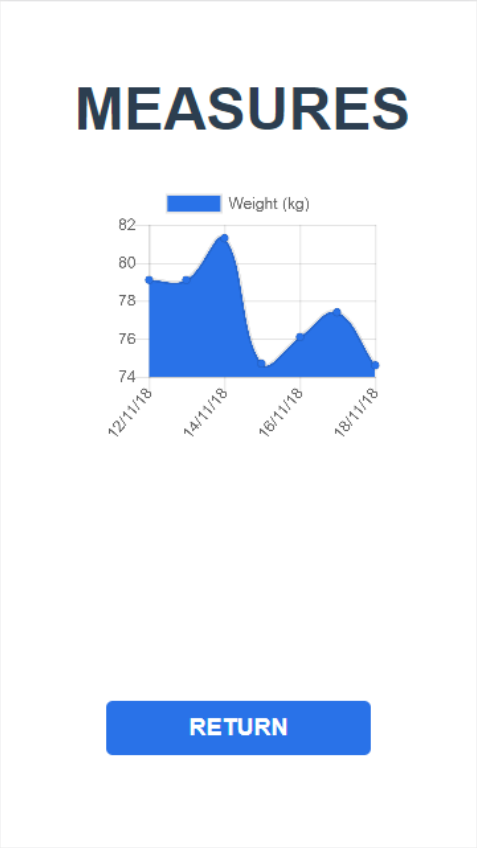
FIGURA 24 - DADOS COLETADOS UTILIZANDO O ESP8266


```
{
    "weightData": [
        {
            "weight": 74.7,
            "timestamp": {
                "_seconds": 1543211706,
                "_nanoseconds": 116000000
            },
            "scale": "AYGU-2987"
        },
        {
            "weight": 76.1,
            "timestamp": {
                "_seconds": 1543211719,
                "_nanoseconds": 452000000
            },
            "scale": "AYGU-2987"
        },
        {
            "weight": 81.3,
            "timestamp": {
                "_seconds": 1543211690,
                "_nanoseconds": 985000000
            },
            "scale": "AYGU-2987"
        },
        {
            "timestamp": {
                "_seconds": 1543211634,
                "_nanoseconds": 916000000
            },
            "scale": "AYGU-2987",
            "weight": 76.2
        },
        {
            "timestamp": {
                "_seconds": 1543211657,
                "_nanoseconds": 760000000
            },
            "scale": "AYGU-2987",
            "weight": 79.1
        },
        {
            "timestamp": {
                "_seconds": 1543211748,
                "_nanoseconds": 517000000
            },
            "scale": "AYGU-2987",
            "weight": 74.6
        },
        {
            "weight": 77.4,
            "timestamp": {
                "_seconds": 1543211731,
                "_nanoseconds": 721000000
            },
            "scale": "AYGU-2987"
        },
        {
            "timestamp": {
                "_seconds": 1543211668,
                "_nanoseconds": 414000000
            },
            "scale": "AYGU-2987",
            "weight": 79.1
        }
    ]
}
```


FONTE: O Autor (2018)

Verifica-se que os dados foram coletados como o esperado. Então, a informação de data é editada no banco de dados, para simular medidas realizadas em diversos dias diferentes. Finalmente observa-se o gráfico mostrado ao usuário (FIGURA 25).

FIGURA 24 - VISUALIZAÇÃO DOS DADOS COLETADOS



FONTE: O Autor (2018)

Verifica-se que, como o esperado, apenas as 7 medidas mais recentes estão presentes, e que os valores estão demonstrados corretamente.

5. CONCLUSÃO

Foi obtido êxito no desenvolvimento da aplicação, o que permitiu que a ideia inicial fosse validada com sucesso, apesar do empecilho na coleta de dados de uma balança real.

As ferramentas escolhidas para a execução do projeto se mostraram altamente confiáveis e eficientes para sua prototipagem. Sem este fator não seria possível realizar o projeto dentro do prazo estimado.

O versionamento do projeto permitiu que fossem experimentadas algumas alternativas para solucionar os desafios do projeto sem o risco de perda do trabalho. Este foi um fator fundamental que auxiliou na melhora do resultado final visto que dentre as alternativas implementadas fora utilizada a mais adequada.

A usabilidade do sistema, no que se diz respeito à configuração da conexão de rede da balança se mostrou mais simples que o esperado. As telas de acesso se mostraram satisfatórias, apesar de simples, pecando apenas em seu design.

Já a coleta de dados diretamente da balança se mostrou um grande desafio. Os dados coletados não apresentaram confiabilidade suficiente para a utilização no fim proposto. Uma solução possível para este problema é desenvolver o próprio sistema de balança para aquisição de dados, ou tratar o sinal de entrada utilizando filtros analógicos ou por amostragem.

5.1. TRABALHOS FUTUROS

Como prova de conceito, o projeto desenvolvido apresenta um enorme potencial para desenvolvimento futuro. Funcionalidades extras como a configuração do gráfico de medidas apresentado agregariam bastante valor à experiência do usuário na utilização da balança. Outras possibilidades são mostrar o gráfico de variação de peso, e permitir o compartilhamento destes dados.

Para além disso, essa solução se mostra muito adequada para a implementação por exemplo em academias, onde o usuário poderá acompanhar seu progresso e ter as medidas realizadas tanto na academia quanto em casa.

Outras possibilidades incluem a implementação do sistema em balanças que possuem leitor de bioimpedância, enriquecendo ainda mais a gama de dados coletados. Nestes casos, a bioimpedância provê informações ricas sobre muitos parâmetros corporais, sendo bastante difícil o seu acompanhamento sem a utilização de ferramentas como a que fora desenvolvida.

O sistema em si, ignorando sua aplicação específica, demonstra a coleta de dados em tempo real e monitoramento. Uma solução de monitoramento IOT, por exemplo na indústria, pode ser de imenso valor – como uma interface onde o engenheiro responsável pode controlar calor e umidade de caldeiras ou fábricas inteiras em tempo real.

REFERÊNCIAS

About Node.js. NODE.JS FOUNDATION. Disponível em <https://nodejs.org/en/about/>. Acesso em: nov. 2018.

ALMEIDA, Regis Rodrigues de. **"Obesidade no Brasil"**; Brasil Escola. Disponível em <https://brasilescola.uol.com.br/saude-na-escola/obesidade-no-brasil.htm>. Acesso em: set. 2018.

Balancing Strong and Eventual Consistency with Google Cloud Datastore. GOOGLE FIREBASE (2018). Disponível em <https://cloud.google.com/datastore/docs/articles/balancing-strong-and-eventual-consistency-with-google-cloud-datastore/>. Acesso em: set. 2018.

Custo Brasil. BÚSSOLA DO INVESTIDOR (2018). Disponível em <https://www.bussoladoinvestidor.com.br/abc_do_investidor/custo-brasil/>. Acesso em: nov. 2018.

CHACON, S.; STRAUB, B. (2014). **Pro Git 2a ed. 2014.** Disponível em <https://git-scm.com/book/en/v2>. Acesso em: nov. 2018.

CHRISTENSSON, P. (2006) **CSS Definition**. Disponível em <https://techterms.com/definition/css>. Acesso em: nov. 2018.

ESP8266EX Datasheet. ESPRESSIF SYSTEMS (2018). Disponível em <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: ago. 2018.

Estudo "Internet das Coisas: um plano de ação para o Brasil". BNDES (2018). Disponível em <https://www.bndes.gov.br/wps/portal/site/home/conhecimento/pesquisaedados/estudos/estudo-internet-das-coisas-iot/estudo-internet-das-coisas-um-plano-de-acao-para-o-brasil> Acesso em: set. 2018.

ESP8266 NodeMCU WiFi Devkit User Manual V1.2. HANDON TECHNOLOGY (2017). Disponível em <http://www.handsontec.com/pdf_learn/esp8266-V10.pdf>. Acesso em ago. 2018.

Firebase API Reference. GOOGLE FIREBASE (2018). Disponível em <https://firebase.google.com/docs/reference/>. Acesso em: set. 2018.
GARCIA, M. (2018). **A revolução da IoT.** BLOG DA ARCON TECNOLOGIA Disponível em: <https://www.arcon.com.br/blog/a-revolucao-iot>. Acesso em: nov. 2018.

HTML5: A vocabulary and associated APIs for HTML and XHTML. WORLD WIDE WEB CONSORTIUM (2017). Disponível em < https://www.w3.org/TR/html5/>. Acesso em: nov. 2018.

HOMAN, J. (2014). **Relational vs. non-relational databases: Which one is right for you?** Disponível em <https://www.pluralsight.com/blog/software-development/relational-non-relational-databases>. Acesso em: set. 2018.

LUETH, K. L. (2018). **IoT Investments 2018: $3.3B annual funding, record number of startup exits.** Disponível em <https://iot-analytics.com/iot-investments-m-and-a-market-update-2018/>. Acesso em: out. 2018.

LUND, D. et al. (2014). **Worldwide and Regional Internet of Things (IoT) 2014–2020 Forecast: A Virtuous Circle of Proven Value and Demand**. International Data Corporation, Framingham, Massachusetts. Disponível em: <https://www.business.att.com/content/article/IoT-worldwide_regional_2014-2020-forecast.pdf> Acesso em: out. 2018.

MONUS, A. (2018). **SOAP vs REST vs JSON comparision [2018].** Disponível em <https://raygun.com/blog/soap-vs-rest-vs-json/>. Acesso em: nov. 2018.

OSMANI A. (2015). **Getting started with Progressive Web Apps**. Disponível em <https://addyosmani.com/blog/getting-started-with-progressive-web-apps/> Acesso em nov. 2018.

PAULA, R. B. de. **O excesso de peso e suas consequências**. Disponível em <https://sbn.org.br/publico/doencas-comuns/o-excesso-de-peso-e-suas-consequencias/>. Acesso em: out. 2018.

SHAN, P. (2014). **Reasons do use JavaScript, with Pros and Cons!** Disponível em <http://voidcanvas.com/reasons-to-use-javascript-with-pros-and-cons/>. Acesso em nov. 2018.

The HTTP protocol. IBM (2018). Disponível em <https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.internet.doc/topics/dfhtl29.html>. Acesso em: nov. 2018.

Vue.js User Guide. VUE.JS. (2018). Disponível em <https://br.vuejs.org/v2/guide/index.html>. Acesso em: set. 2018.

What is JSON? SQUARESPACE (2018). Disponível em <https://developers.squarespace.com/what-is-json>. Acesso em: nov. 2018.

What is HTTPS? COMODO CA. (2017). Disponível em <https://www.instantssl.com/ssl-certificate-products/https.html>. Acesso em: nov. 2018.