FACULDADE DE TECNOLOGIA SENAC BLUMENAU
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Diego Weiler
Diogo Ottequir
Frederico Wuerges Becker

PROTÓTIPO PARA CONTROLE DE NÍVEL E VAZÃO DE ÁGUA UTILIZANDO CONCEITO DE INTERNET DAS COISAS EM RESERVATÓRIOS HÍDRICOS DE CONDOMÍNIOS

BLUMENAU
2017

Diego Weiler
Diogo Ottequir
Frederico Wuerges Becker

PROTÓTIPO PARA CONTROLE DE NÍVEL E VAZÃO DE ÁGUA UTILIZANDO CONCEITO DE INTERNET DAS COISAS EM RESERVATÓRIOS HÍDRICOS DE CONDOMÍNIOS

Trabalho de conclusão de semestre, apresentado à Faculdade de Tecnologia Senac Blumenau, como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadores: Cláudio Ratke Msc.
Fabiano Oss Esp.
Lariana Luy Peixoto Msc.

BLUMENAU
2017

Diego Weiler
Diogo Ottequir
Frederico Wuerges Becker

PROTÓTIPO PARA CONTROLE DE NÍVEL E VAZÃO DE ÁGUA UTILIZANDO CONCEITO DE INTERNET DAS COISAS EM RESERVATÓRIOS HÍDRICOS DE CONDOMÍNIOS

Trabalho apresentado à Faculdade de Tecnologia Senac Blumenau como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

_____
Cláudio Ratke, Msc.

_____
Fabiano Oss, Esp.

_____
Lariana Luy Peixoto, Msc.

_____
Luciano França, Msc.

Blumenau, 27 de junho de 2017.

RESUMO

Vivemos em um período de crescimento econômico, em que os grandes centros urbanos se modernizam e trazem melhores condições de moradia. O surgimento de novos condomínios de prédios é uma realidade. Porém, isso traz consigo alguns problemas como o aumento excessivo no consumo de água, recurso este essencial para a sobrevivência das espécies e da sociedade. As autoridades governamentais neste contexto convidam a sociedade para que façam uma reflexão e venham a utilizar este recurso com consciência e de forma sustentável. Para isto, a tecnologia da informação auxilia na realização através dos novos conceitos de Internet das Coisas, em que é possível gerar informações a partir de dispositivos eletrônicos de baixo custo. Utilizando estes conceitos e com a finalidade de controlar a água, foi escolhido condomínios de prédios para elaborar a arquitetura da solução de um protótipo utilizando Arduino, com a finalidade de controlar o uso excessivo da água pela sociedade moderna. Portanto, o principal objetivo é contribuir para que a sociedade se conscientize quanto ao uso da água em apartamentos residenciais através da visualização de dados estatísticos, possibilitando a redução de seu consumo e conscientizando o seu uso de forma disciplinada para que as gerações futuras possam usufruí-la. A solução é baseada no uso de um dispositivo Arduino, com sensores de nível e vazão de água instalados em reservatórios de condomínios, com o envio de dados para uma API, que armazena e disponibiliza as informações para uma interface *web*.

Palavras-chave: Água. Arduino. Condomínios Residenciais. Internet das Coisas.

ABSTRACT

We live in a period of economic growth, in which the great urban centers are modernized and bring better housing conditions. The emergence of new condominiums of buildings is a reality. However, this brings with it some problems such as the excessive increase in water consumption, an essential resource for the survival of species and society. Government authorities, in this context, invite society to reflect and use this resource conscientiously and in a sustainable way. For this, information technology helps in the realization through the new concepts of Internet of Things, in which it is possible to generate information from electronic devices of low cost. Using these concepts and with the purpose of using water in a sustainable way, condominiums were chosen to construct the architecture of the solution of a prototype using Arduino, in order to control the excessive use of water by modern society. Therefore, the main objective is to contribute to the society's awareness of the use of water in residential apartments through the visualization of statistical data, making it possible to reduce its consumption and making its use aware in a disciplined way so that future generations can enjoy it. The solution is based on the use of an Arduino device, with level and water flow sensors installed in reservoir reservoirs, sending data to an API, which stores and makes information available for a web interface.

Keywords: Arduino. Internet of Things. Residential Condos. Water.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

ABREVIATURAS E SIGLAS

API - *Application Programming Interface* ou Interface de Programação de Aplicativos
DNS - *Domain Name System* ou Sistema de Nome de Domínio
HTTP - *Hypertext Transfer Protocol* ou Protocolo de Transferência de Hipertexto
IAB SP - Instituto de Arquitetos do Brasil - Departamento de São Paulo
IDE - *Integrated Development Environment*
IoT - *Internet of Things* ou Internet das Coisas
JSON - *Javascript Object Notation* ou Notação de Objeto Javascript
MMA - Ministério do Meio Ambiente
REST - *Representational State Transfer*
SE - Sistema Embarcado
Sabesp - Companhia de Saneamento Básico do Estado de São Paulo
Sebrae - Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SPA - *Single Page Application* ou Aplicação de Página Única
TI - Tecnologia da Informação
URI - *Uniform Resource Identifier*
VAC - Volt Corrente Alternada
VDC - Volt Corrente Contínua
WS - Web Service ou Serviço Web
WWW - World Wide Web
XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da população mundial, há necessidade de aumentar o controle sobre o uso dos recursos naturais. Dentre estes, destaca-se a água doce, por ser um recurso não renovável, fundamental para a sobrevivência dos seres vivos e a manutenção das atividades econômicas que dão sustentação à sociedade. A luz desta informação, apenas 2,50% de toda a água existente no planeta é água doce, sendo que apenas 0,26% está disponível para ser utilizada por todos os seres vivos, em rios, lagos e depósitos subterrâneos, representando o disponível para consumo (MMA, 2016, p.27).

A diminuição da disponibilidade de água para o consumo humano vem sendo constatada há alguns anos (BRASIL, 2015), principalmente nos grandes centros, que têm sido considerados como causas o desmatamento e destruição de mananciais no campo e na cidade pela urbanização, a impermeabilização do solo, a retirada da mata ciliar, o consumo excessivo da água, a contaminação e a poluição de águas. Diante disso, inúmeras ações são tomadas pelas autoridades governamentais para que seja reduzido este fato.

As autoridades governamentais federais (IBAMA), estaduais (FATMA, em SC) e municipais (FAEMA em Blumenau), têm tomado algumas ações para que este dano seja reduzido. O Brasil conta com uma lei específica, conhecida como Lei das Águas - Lei nº 9.433/97, que é a Política Nacional de Recursos Hídricos. A lei destaca que a gestão dos recursos hídricos deve assegurar às gerações atuais e futuras a disponibilidade de forma sustentável (IRVING; OLIVEIRA, 2012).

Segundo dados mundiais publicados pela Sabesp (2017), o gasto médio de água em unidades consumidoras em condomínio pode chegar até 110 litros/dia, para suprir todas as suas necessidades de consumo e higiene. Tendo em vista o crescente aumento de condomínios nos conglomerados urbanos, o aumento no consumo de água será inevitável, em que deverá existir uma forma de medir e prevenir o seu uso de forma eficiente nesses condomínios.

A tecnologia da informação entra em cena como uma eficiente ferramenta de apoio para a gestão de informações, bem como uma ferramenta para a competitividade das organizações, fornecendo conteúdo e conhecimento que

circulam sem fronteiras, potencializando a importância intelectual para tomada de decisões baseada em dados coletados.

A evolução da tecnologia e a redução de custos de sensores e equipamentos Arduino têm impulsionado a adoção da internet das coisas a partir de um baixo custo de investimento. O Arduino é um conjunto de ferramentas de prototipagem eletrônica de código aberto que permite a criação de dispositivos e sensores eletrônicos, se tornando útil na automação de processos, sendo possível criar protótipos que permitem a comunicação de coisas: desde máquinas industriais até a integração total de residências (luzes, aparelhos eletroeletrônicos, ar-condicionado, etc.) com a internet.

Sendo assim, este trabalho traz uma proposta de controle do uso da água em condomínios prediais, buscando diminuir desperdícios e minimizar os custos. Apresenta um sistema de informação baseado em coletas de medição e histórico de consumo de água doce, utilizando a caixa d'água como reservatório, de forma a garantir dados precisos, utilizando conceitos de Internet das Coisas (IoT) e recursos da tecnologia da informação para sua execução.

1.1 Objetivos

O objetivo geral é desenvolver um protótipo de software para monitoramento de reservatório hídrico de condomínios, utilizando conceitos e recursos de internet das coisas (IoT).

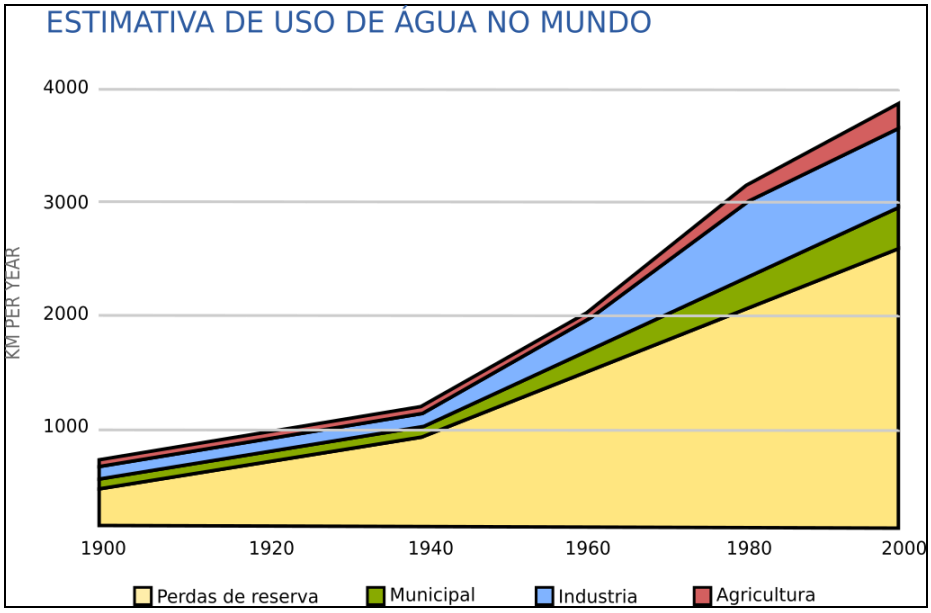
Para sua realização, os objetivos específicos são:

a) Desenvolver o protótipo do dispositivo de medição do reservatório.
b) Desenvolver a plataforma para recepção, disponibilização e armazenamento das informações enviadas através do equipamento de medição.
c) Desenvolver a interface gráfica para visualização das informações disponibilizadas pela plataforma de armazenamento.
d) Realizar uma pesquisa com condomínios para avaliar a viabilidade e aplicabilidade do projeto.
e) Contribuir com o desenvolvimento de tecnologia para a sustentabilidade de ecossistemas e da sociedade

1.2 Justificativa

Sabe-se que a água doce é um recurso indispensável para a sobrevivência da espécie humana. Segundo dados elaborados pela Unesco (BRASIL, 2015), através do artigo publicado no Portal Brasil, a estimativa é que o consumo da água doce cresça 55% até 2050. Neste artigo, é enfatizado que haverá água para suprir as necessidades desde que haja uma mudança consciente no uso, gerenciamento e compartilhamento. A Figura 1 demonstra claramente o consumo de água estimado no mundo.

Figura 1 - Consumo Estimado de Água no Mundo



Fonte: Modificado de *Food and Agriculture Organization of the United Nations* (2008).

Na Figura 1 pode-se observar o consumo de água no mundo, entre os anos de 1900 e 2000, medido em km³. Constata-se que o consumo no ano de 1900 era menor que 1000 km³ e que, um século depois, no ano de 2000, pulou para cerca de 2500 km³.

Atualmente, o consumo estimado em uma unidade consumidora é cerca de 27m³/mês, com um custo mensal de aproximadamente R$ 241,89 (CONDOR apud SAMAE, 2016). Portanto, a elaboração de um protótipo de *software* de controle deste consumo permite ações extras que reduzirão a chance de desperdícios, melhor eficiência energética e fornecer dados históricos para ações futuras,

colaborando para que toda uma comunidade possa ser beneficiada com a possibilidade de desenvolvimento sustentável, através do gerenciamento das informações do nível da água contido em um determinado reservatório existente e o compartilhamento destas informações.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão abordados os seguintes tópicos: água, gestão integrada de recursos hídricos, cenário atual da água, case da crise hídrica no estado de São Paulo e sua solução, internet das coisas (IoT), arquitetura da solução IoT, sistemas embarcados e internet como meio de consumo das informações.

2.1 Água

A água é um recurso essencial para que haja sobrevivência dos seres vivos e a existência da sociedade. Segundo dados da Unesco publicado pelo Ministério do Meio Ambiente (MMA, 2006), indicam que o planeta Terra é formado de 97,5% de água salgada, ou seja, apenas 2,5% é água doce. Além disso, a concentração desta água está 68,9% em geleiras e calotas polares, 29,9% em águas subterrâneas, 0,9% umidade do solo e pântanos e apenas 0,3% constitui a porção superficial presente em rios e lagos.

O consumo diário de água sofre uma variação muito grande ao redor do mundo. Por exemplo, um cidadão europeu, do continente que possui 8% da água doce no mundo, consome cerca de 150 litros de água por dia. Em contrapartida, a Unesco garante que uma pessoa necessita de pelo menos 40 litros de água para suas necessidades básicas (MMA, 2006, p.28).

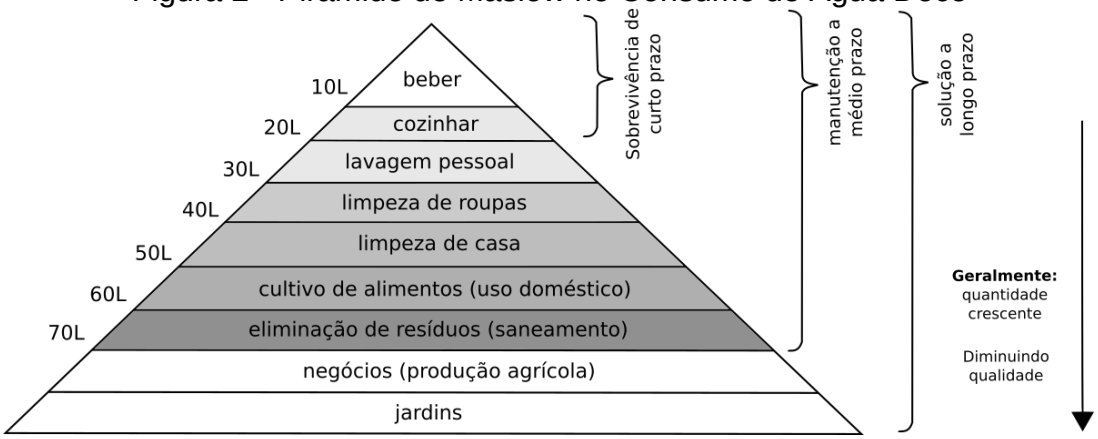
Em vista disto, as autoridades governamentais vêm tomando atitudes através da instituição de leis para que haja um consumo consciente e sustentável. No Brasil, foi criada a lei nº 9.433/97, conhecida como Lei das Águas, que considera principalmente que a água é um bem público e recurso natural limitado, dotado de valor econômico, em que deve priorizar o consumo humano e de animais, em especial em situações de escassez. Além disso, deve ser gerida proporcionando usos múltiplos (abastecimento, energia, irrigação, indústria) e sustentáveis com participação de toda a sociedade civil e do governo, assegurando acesso às gerações atuais e futuras (IRVING; OLIVEIRA, 2012).

Podemos destacar ações do governo do estado de Santa Catarina, exigidas no decreto nº 852 de 01/09/2016, que pôs em prática ações contidas na lei, promovendo a participação da sociedade com a distribuição de material educativo

sobre a gestão de recursos hídricos e o uso consciente da água; cursos de capacitação para os entes do Sistema Estadual de Gerenciamento dos Recursos Hídricos e usuários de água; palestras educativas, debates e seminários sobre a gestão dos recursos hídricos; e campanhas informativas de conscientização sobre o uso consciente da água (LEIS ESTADUAIS, 2016).

Neste contexto, a Figura 2 demonstra a aplicação da pirâmide de Maslow para o consumo de água doce.

Figura 2 - Pirâmide de Maslow no Consumo de Água Doce



Fonte: Modificado de WHO - World Health Organization (2005, p. 2, tradução nossa)

Pode-se observar que a necessidade ideal de água é de 70 litros/dia. E que a qualidade da água é reduzida, ao passo que esta quantidade aumenta. Por este motivo, o uso da água deve ser feito de forma consciente e sustentável. Ao contrário, ocorrerá a escassez, inviabilizando a manutenção da sociedade, comprometendo a sobrevivência das espécies. Isto nos pede que tenhamos ações emergenciais para que haja o devido controle deste recurso finito e precioso.

2.2 Gestão Integrada de Recursos Hídricos

Beekman (2017) explica que a natureza possui uma finita fonte renovável de água, sob o ponto de vista do crescimento populacional, sendo poucos os outros recursos essenciais à vida que estão restritos por limites de disponibilidade tão definidos quantos aos recursos hídricos. A disponibilidade média, de água potável, por habitante tende a diminuir, o que repercute sobre a saúde e os padrões de

qualidade de vida.

Entende-se que, a água é um recurso fundamental para a sobrevivência, em que na maioria das vezes não é utilizada de maneira sustentável, resultando na sua poluição e escassez. A gestão integrada da água, segundo Tera (2015) é um "projeto sistemático que tem como principal objetivo a alocação, monitoramento e o desenvolvimento sustentável dos recursos hídricos".

A gestão integrada tem como base que a água é um recurso finito e seu uso é interdependente. Essa prática promove o desenvolvimento coordenado dos recursos hídricos com o intuito de maximizar seu uso sem comprometer a sustentabilidade de ecossistemas (TERA, 2015).

Os principais objetivos da gestão integrada da água, segundo Tera (2015), são: ser eficiente ao utilizar os recursos; ter igualdade na alocação das águas entre todos os grupos socioeconômicos; proteção integrada dos ecossistemas e dos recursos hídricos.

Tal modelo de gestão desafia os modelos tradicionais de administração dos recursos hídricos, não tendo um ponto de partida e nem um fim fixo, visto que o meio ambiente está sujeito a mudanças constantes e, por este motivo, a gestão integrada dos recursos hídricos deve ser capaz de responder a mudanças e se adaptar às condições inéditas. (TERA, 2015)

O gerenciamento dos recursos pode ser considerado em todos os níveis, abrangendo: nacional, regional, estadual, bacia hidrográfica e local. Em cada um dos níveis, a gestão lógica da água como recurso único requer interações funcionais, em relação às responsabilidades, no que se refere a: uso da água e ocupação territorial (uso do solo); águas superficiais e subterrâneas; e qualidade e quantidade. O autor conclui que quantidade e qualidade são indissociáveis (BEEKMAN, 2017).

Nesse contexto, pode-se concluir que uma gestão integrada da água precisa necessariamente controlar quantidade e qualidade em seus ecossistemas. Contudo, neste trabalho é abordado apenas o controle de quantidade, pois conforme apontam Cassiolato e Orellana (2017), vazão e nível estão entre os principais controles de grandeza de medição industriais.

2.3 Cenário Atual da Água

Nos noticiários no Brasil e no mundo é comum relatos do crescimento populacional associados ao crescente consumo de água, muitas vezes a própria escassez, em que autoridades governamentais discutem medidas a minimizar os impactos da escassez de água.

A Unesco (BRASIL, 2015) descreve que a estimativa é que o consumo no Brasil cresça 55% até 2050, o que é preocupante, já que a lei das águas pede que o acesso deve ser assegurado às gerações atuais e futuras.

No Brasil, segundo dados do Instituto de Brasileiro de Geografia e Estatísticas (IBGE), a população total é de mais de 207 milhões de pessoas, espalhados nas 27 unidades federativas. Levando em consideração que o consumo médio de uma unidade consumidora, no estado de São Paulo, é de cerca de 110 litros/dia, segundo dados da Companhia de Saneamento Básico do Estado de São Paulo (SABESP, 2015), o consumo total por mês seria de 22,77m³/dia.

De acordo com a Unesco (BRASIL, 2015), o fator crescimento populacional é agravado quando olhamos para os grandes centros urbanos em que há grandes conglomerados de pessoas, gerando a destruição de mananciais, impermeabilização do solo, retirada da mata ciliar, consumo excessivo de água e poluição das águas.

Nossa legislação brasileira diz que todo o cidadão deve ter direito a moradia, conforme lei nº 11.124/2015 art. 2º. Isto é um fato agravante, já que os governantes devem permitir acesso à moradia, além de recursos essenciais para a vida, como por exemplo, saneamento básico, acesso à água potável, energia elétrica, etc. (PLANALTO, 2015).

Como solução, a sociedade faz a construção de conjuntos residenciais através de condomínios. O condomínio possui a vantagem de permitir maior número de famílias com acesso à moradia quando comparado a uma casa. Segundo dados do Censo Demográfico (2010), apresentados pelo IBGE (UCHINAKA, 2011), no Brasil, em média, 13% das moradias são condomínios, o que indica a importância do controle no consumo de água em condomínios residências.

Segundo Matsuki e Gomes (2015), a Austrália utiliza controle eletrônico para evitar perdas e gerenciar o ciclo hídrico. Os autores afirmam que o monitoramento é

realizado de forma integrada sobre o consumo, chuvas, desperdício e águas subterrâneas para garantir a captação e distribuição da água de forma inteligente e eficiente, onde soluções específicas possibilitam uma economia de 40 a 50% em cada setor.

O recurso de tecnologia empregado na Austrália foi através do desenvolvimento de novas tecnologias de inovação e investigação que apresenta os níveis de água por cidade, fornecendo informações precisas e detalhadas.

Pode-se destacar que a tecnologia da informação (TI) pode auxiliar a sociedade, propondo soluções de problemas recorrentes, como por exemplo, o controle de consumo de água doce. Na próxima seção, é apresentada a solução de TI empregada na crise hídrica em São Paulo, vivida nos anos de 2014 a 2016, em que foi possível adotar uma ferramenta que apoia a solução da falta de abastecimento, passando a controlar o consumo e utilização da água doce (OLIVEIRA, 2016, p. 84).

2.4 Case: Crise Hídrica no estado de São Paulo

A Sabesp adotou uma ferramenta que apoia a solução da crise hídrica vivida nos anos 2014 a 2016 no estado de São Paulo através do uso do conceito de IoT, em que passava a supervisionar, controlar, medir e obter dados à distância de mananciais que abastecem cerca de 365 municípios do estado (OLIVEIRA, 2016, p. 84).

As principais ações foram realizadas para aumentar o volume de água reservada, ampliar a capacidade de tratamento e interligar áreas de abastecimento. Contudo, o artigo revela que o elemento chave desta operação foi o emprego de tecnologia da informação (TI). Oliveira (2016, p. 85) destaca que Osvaldo Pazianotto, diretor de tecnologia da Sabesp, declarou o seguinte: “Esse foi um grande desafio. Montamos um plano de ação para mudar a situação por meio de tecnologia.” Este trabalho rendeu a Sabesp o prêmio “As 100+ Inovadoras no Uso de TI” na categoria *Utilities* e primeiro lugar no *ranking* geral.

Oliveira (2016) explica que o projeto girou em torno de prover uma plataforma utilizando o conceito de IoT, visando aprimorar o processo de automação, com telemetria e telecomando, com a finalidade de medir o nível da

água nos reservatórios, sem perder a segurança e com o menor custo possível. Além disso, possibilita realizar a medição do consumo de água em tempo real, permitindo identificar possíveis vazamentos por desvio padrão de consumo ou divergência entre o volume de água disponibilizado e consumido na região. Conta ainda com serviços de apoio através de tecnologia móvel, em que os cidadãos possam indicar irregularidades, facilitando o tempo de resposta aos chamados dos técnicos em campo.

O autor descreve o projeto, onde foram instalados sensores distribuídos nos vários reservatórios, que efetuam a coleta de informações e, em seguida, realizam a transmissão dos dados para uma central de controle, que tem a função de receber, acompanhar e realizar a análise e a gestão das informações. Desta forma, a empresa pública teve que investir em nova infraestrutura de *data center*, armazenamento, *cloud computing*, telecomunicações, mobilidade, RFID, segurança da informação e modernização dos sistemas corporativos.

Portanto a evolução da TI permite o surgimento de dispositivos de baixo custo conectados à internet, que formam um novo conceito, que permite expandir as funcionalidades, chamado de Internet das Coisas (IoT). A Sabesp utilizando este recurso da TI passa a controlar e monitorar a água, trazendo uma melhora para toda a sociedade contida neste contexto, assegurando acesso a água a toda população.

2.5 Internet das Coisas (IoT)

Segundo Meola (2016), “a IoT refere-se à conexão de dispositivos à Internet”. Assim sendo, a internet das coisas tem pela frente o desafio de facilitar a interação com o mundo real, envolvendo objetos que estão ao nosso redor e desconectados da rede, prometendo ser interativo e dinâmico (SEBRAE, 2016).

De acordo com um artigo publicado pelo Sebrae, dentro de 5 a 10 anos haverá mais de 100 bilhões de objetos conectados em rede, segundo Prof. Michael Nelson (2016) da Universidade de Georgetown *Comunication, Culture & Tecnology* e Diretor de Tecnologia Internet da IBM. A Figura 3, ilustra o funcionamento da Internet das Coisas em nosso dia-a-dia.

Portanto a proposta da IoT é “criar sistemas e ferramentas que emprestem mais inteligência aos objetos para que eles possam conversar entre si e tornar

nossa vida mais fácil”.

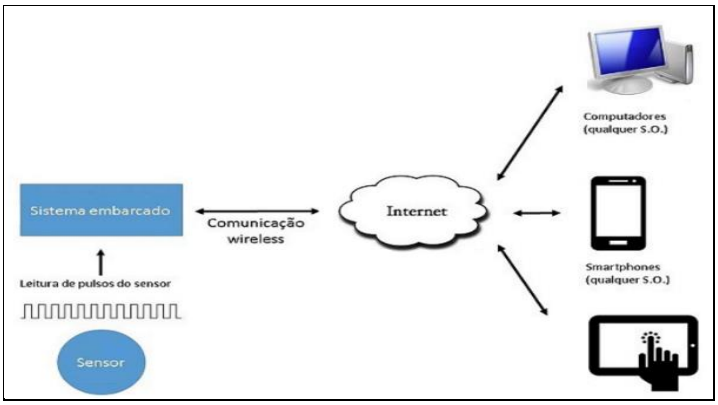
Figura 3 - Internet das Coisas (IoT)

Fonte: Cedro Technologies, 2017.

2.6 Arquitetura da solução IoT

Para que exista o conceito de IoT, é obrigatório que exista um dispositivo ligado à internet. A Figura 4 - Conceito de Funcionamento da IoT demonstra o conceito básico de funcionamento da IoT, no que podemos identificar a necessidade de um sistema embarcado, com conexão à internet para transmitir dados a outros dispositivos que possam consumir informações.

Figura 4 - Conceito de Funcionamento da IoT



Fonte: Embarcados (2015), adaptação dos autores.

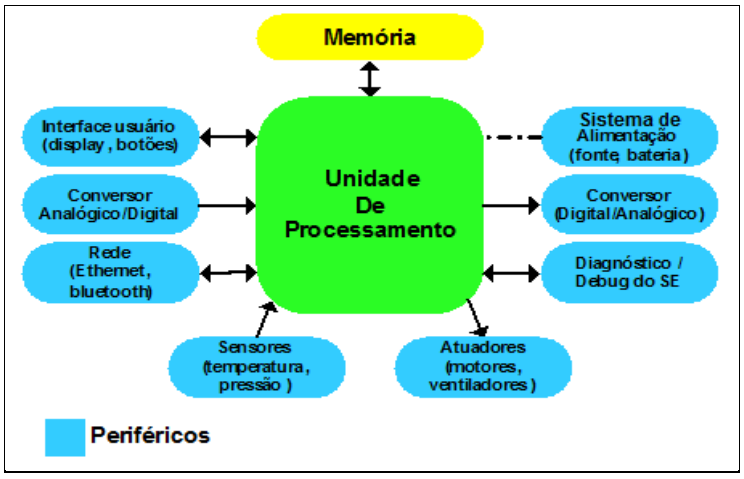
2.7 Sistemas Embarcados

Delai (2013), para um artigo ao site Hardware, define sistemas embarcados (SE) como relacionamento de *hardware* (eletrônica) e *software* (instruções), incorporados em um dispositivo com um objetivo pré-definido, possuindo dimensionamento de recursos a um domínio de objetivos bem menores, ou singular.

Os principais componentes de um sistema embarcado, segundo Delai (2013) são: Unidade de Processamento: executam as instruções (*software/firmware*) responsáveis por realizar cálculos, tomar decisões e tratar eventos. Possui normalmente a arquitetura elementar clássica de um processador de computador convencional, como a unidade lógica/aritmética (ULA), unidade de controle (UC), registradores, entre outros. Memória: armazena dados e instruções relacionados às operações da unidade de processamento. As instruções e dados podem dividir a mesma memória, como nos PCs (arquitetura Von Neumann) ou separados em memórias distintas (arquitetura Harvard), sendo a segunda a mais comum em sistema embarcado (SE). Periféricos: são as interfaces da unidade de processamento com o mundo externo, trazendo ou enviando informações para ele. Um exemplo de um periférico seria um conversor analógico/digital acoplado a um sensor térmico que converte a temperatura de um ambiente em números binários para que a unidade de processamento consiga interpretar e processar a informação.

A Figura 5 nos permite uma visão da subdivisão dos sistemas embarcados.

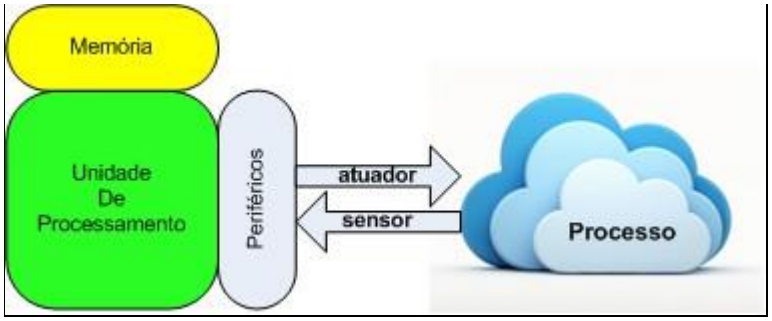
Figura 5 - Subdivisão de um sistema embarcado



Fonte: Hardware (2013).

Com isso, o principal objetivo de um sistema embarcado é controlar processos, ou seja, atuar sobre um problema. Isso é feito através dos periféricos, que são chamados e dimensionados com base no problema alvo. A Figura 6 ilustra o funcionamento dos periféricos nos processos (DELAI, 2013).

Figura 6 - Funcionamento dos Periféricos nos Processos



Fonte: Delai (2013)

Com isso, existem duas categorias diferentes de periféricos que são: os sensores e os atuadores.

Os sensores segundo Borges e Dores (2010), são componentes eletrônicos capazes de converter uma grandeza física em uma grandeza elétrica, com o objetivo de captar uma ação física como temperatura, umidade, vazão, nível, entre outros.

Os autores continuam comentando que estes componentes auxiliam na automação de processos manuais que seguem algum tipo de padrão, como por exemplo, uma dobradura de papel, desde de que se conheça os passos para efetuá-la, os sensores podem coletar o passo em que se encontra e efetuar o próximo passo, assim sucessivamente, até finalizar todo o processo, podendo repetir este processo por várias vezes do mesmo modo que a primeira vez.

> Sensores são responsáveis em adquirir informação do processo a ser controlado. Essas informações são primordiais para a unidade de processamento, pois com base nelas decisões podem ser tomadas. Bons sensores devem fornecer informação confiável e não promover alterações no processo alvo. Isso significa que um sensor não pode mudar os valores da grandeza física do qual é responsável por medir, como por exemplo diminuir a velocidade de um motor em monitoramento. Isso na prática pode difícil de conseguir dependendo da tecnologia do sensor (contato mecânico por exemplo). Exemplos desses tipos de periféricos são sensores de temperatura (termistores), pressão (piezos), contato (chaves mecânicas), toque (*touchscreen*), distância (sonar/infravermelho), movimento (acelerômetros), óticos (câmeras), etc. Esses são periféricos que enviam informação do processo para o SE (DELAI, 2013).

E os atuadores, segundo Brugnari e Maestrelli (2010), convertem grandezas, porém de caminho inverso aos sensores, transformando uma grandeza elétrica em uma grandeza física como movimento, magnetismo, calor entre outros.

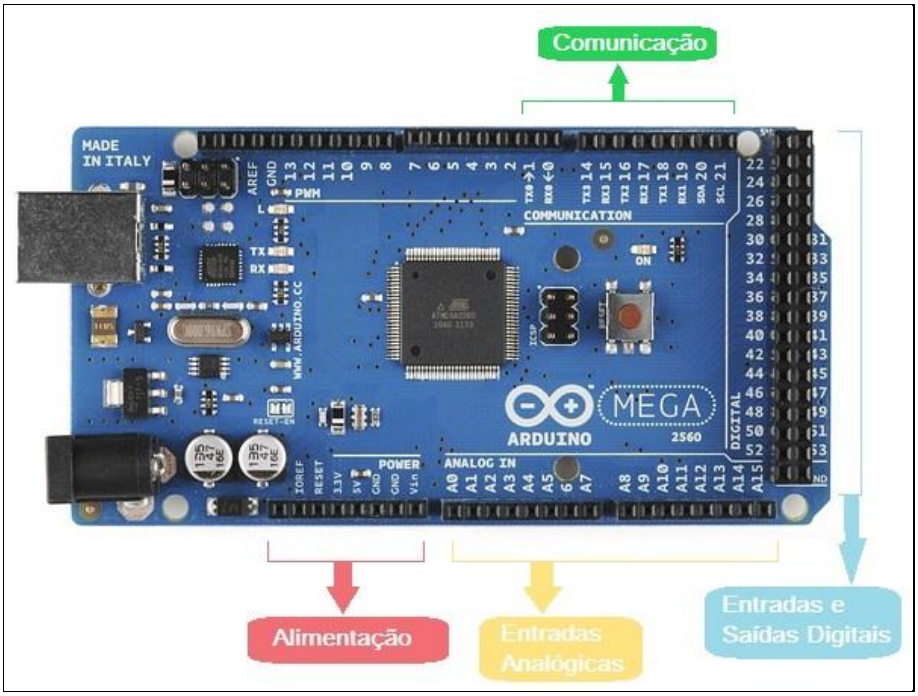
Podemos assim concluir que os sistemas embarcados facilitam tarefas do cotidiano das pessoas. Atualmente existem kits de microcontroladores que permitem avançar da teoria à prática em poucos minutos, como é o caso do dispositivo Arduino. O conceito de dispositivo Arduino e implementações de sistemas nesta plataforma é apresentada a seguir.

2.7.1 Arduino

Segundo o site oficial do fabricante (ARDUINO, 2017), o Arduino é uma plataforma de eletrônica de código aberto baseado em sistemas embarcados (SE). Esta tecnologia é capaz de receber dados de forma digital e analógica, processar os dados recebidos e apresentar os resultados de forma a facilitar a compreensão dos dados recebidos.

A Figura 7 retrata os recursos do SE Arduino.

Figura 7 - Recursos do SE Arduino.



Fonte: Modificado de GitHub (2016).

Existem outros microcontroladores e plataformas disponíveis como exemplo: PIC e dsPIC da *Microchip*, *Raspberry*, *Parallax Basic Stamp*, BX-24 da *Netmedia*, *Phidgets*, *Handyboard* do MIT entre outras.

Atualmente no mercado, existem vários outros modelos de dispositivos Arduino disponíveis, onde a principal diferença entre eles é a quantidade de pinos de entrada e saída de forma digital e analógica.

Com este dispositivo microprocessado, pode-se gerar informações e dados de qualquer tipo de origem captadas através de seus sensores e atuadores.

2.7.2 Tecnologias Utilizadas no Arduino

Por se tratar de um SE, ele possui dois principais componentes: *hardware* e *software*. Segundo Souza (2013), o *hardware* é composto por uma placa de prototipagem na qual são construídos os projetos. O *software* desenvolvido pela IDE (*Integrated Development Environment*) do Arduino é chamado de *sketch*, que é compilado pela própria IDE e transferido para o dispositivo eletrônico através de uma comunicação serial (USB).

A placa Arduino possui diversos conectores para a entrada e saída de dados que são processados e enviados para uma interface com o mundo externo, separadas em digitais e analógicas. A sua alimentação pode ser feita a partir da porta USB ou através de um adaptador AC, no qual é especificada pelo fabricante uma tensão de 7 a 12 Volts (ARDUINO, 2017).

A programação da placa é feita em linguagem C/C++, transmitida para o compilador AVR-GCC, que realiza a tradução dos comandos para uma linguagem que possa ser executada pelo microcontrolador.

O ciclo de programação do Arduino pode ser dividido da seguinte maneira: conexão da placa a uma porta USB do computador; desenvolvimento de um *sketch* com comandos para a placa; *upload* do *sketch* para a placa utilizando a comunicação USB; e aguardar a reinicialização, pois após ocorrerá à execução do *sketch* criado.

Depois de concluído, o Arduino executa o *sketch* criado, desde que seja ligado a uma fonte de energia. O Arduino também se comunica com vários periféricos que podem ser utilizados separadamente para ampliar a aplicação, assim

como é utilizado neste trabalho, um módulo para conexão com a internet para comunicação com uma aplicação externa.

2.8 Internet como Meio de Consumo de Informações

O desenvolvimento de novas técnicas e tecnologias de comunicação provocaram mudanças, principalmente ao longo do século XXI, nos modos de produção e distribuição da informação (AGNEZ, 2009).

O fator determinante para tal fato foi o surgimento da Internet. O termo Internet veio do inglês que significa rede mundial de computadores, a qual permite acesso a informações de todo tipo e muitas transferências de dados, além da variedade de recursos e serviços disponíveis. Sendo assim, revelado como um grande fator de comunicação, integração social, armazenamento de informações e globalização de produtos (MICHAELIS, 2017).

Com o aumento da Internet, surgiu a necessidade de organizar através de uma arquitetura, chamada *Web*. O IABSP (Instituto de Arquitetos do Brasil - Departamento de São Paulo, 2017) define arquitetura como “construção concebida com o propósito primordial de ordenar e organizar o espaço para determinada finalidade e visando a determinada intenção”.

Araújo (1997) explica que a arquitetura baseada na plataforma Web, modifica uma série de conceitos e paradigmas, em que pode se destacar como principais: computação distribuída, arquitetura cliente-servidor e multicamadas.

Para que haja o conceito de IoT, os dispositivos embarcados necessitam de acesso à internet. A internet entra como principal meio de comunicação entre sistemas, o qual requer uma infraestrutura própria para que suporte a recepção dos dados provindos de sistemas externos. Para isso, pode-se utilizar o recurso de uma Web API, que é uma plataforma de comunicação que utiliza a internet e permite que vários sistemas se comuniquem através de um formato comum, sem a necessidade de programação adicional em diferentes plataformas.

2.8.1 Interface de Programação de Aplicativos (API)

O conceito de API, segundo Canaltech (2017), é uma interface de

programação de aplicativos (*Application Programming Interface*), ou seja, “é um conjunto de rotinas e padrões de programação para acesso a um aplicativo de *software* ou plataforma”.

A API pode ser criada quando uma empresa de *software* tem a intenção de que outros criadores de *software* desenvolvam produtos associados ao seu serviço. Para exemplificar, o Google Maps permite que outros sistemas utilizem os dados, permitindo ser adaptado para a necessidade deste serviço (CANALTECH, 2017).

O Canaltech (2017) explica que através do uso de APIs, é possível complementar funcionalidade de um determinado programa, aumentando os seus recursos e suas funcionalidades.

> Os aplicativos podem se comunicar uns com os outros sem conhecimento ou intervenção do usuário. Eles funcionam através da comunicação de diversos códigos, definindo comportamentos específicos de determinado objeto em uma interface. A API liga as diversas funções em um site de maneira que possam ser utilizadas em outras aplicações. De modo geral, a API é composta de uma série de funções acessíveis somente por meio de programação (CANALTECH, 2017).

Ribeiro (2017) explica que a principal utilidade das APIs é a automatização de tarefas repetitivas, reduzindo custos e aumentando a sua produtividade. Desta forma, é possível concluir que, com o uso de APIs na construção de *softwares*, é possível de maneira simples, com baixo custo e de forma rápida, adicionar recursos e funcionalidades a um sistema.

2.8.2 API REST

Definido por Roy Fielding em sua tese de doutorado (2000), um dos principais autores da especificação do protocolo HTTP (*Hypertext Transfer Protocol*) que é utilizado por sites da internet, REST “é um design de arquitetura construído para servir aplicações em rede”, trazendo uma série de benefícios em sua construção e concepção (PLANSKY, 2014).

Uma requisição web, que pode ser feita através do protocolo HTTP, é construída através de três pilares, sendo eles: recursos, operações e representação.

Os recursos são definidos como caminho de acesso a uma requisição, também conhecido como URI (*Unified Resource Identifier*), que é o identificador

único de acesso (CAELUM, 2017).

As operações são definidas como ações, o protocolo HTTP possui sete tipos de operações (métodos), sendo eles: GET, POST, PUT, DELETE, HEAD, OPTIONS e TRACE. Os mais conhecidos são os métodos GET e POST, que são comumente utilizados nas páginas da web através de links e processamento de formulários (CAELUM, 2017).

As operações geralmente são relacionadas a determinadas ações em que uma aplicação RESTful (aplicações que implementam o REST) deve executar na URI informada na requisição, sendo elas conforme descrito no Quadro 1.

Os recursos são manipulados através das suas representações, que são uma visão do estado de um recurso em determinado instante de tempo. Essas representações no protocolo HTTP podem ser manipuladas em diferentes formatos durante uma requisição, onde os mais comuns em uma aplicação RESTful são o JSON (*Javascript Object Notation*) e XML (*eXtensible Markup Language*) (CAELUM, 2017).

O JSON é um formato para troca de dados entre diferentes sistemas que foi inicialmente criado para ser utilizado apenas com *JavaScript*, que é uma linguagem de programação para web. Atualmente o formato não limita somente a esta linguagem, sendo um dos principais formatos para transferência de dados, devido a sua sintaxe simples, tamanho pequeno (comparado ao XML), consequentemente o tornando mais rápido para troca de informações (DPW, 2017).

Na Figura 8 é possível visualizar um exemplo de como o JSON deve ser utilizado, onde é constituído de uma coleção de pares de nome e valor, ou também por uma lista ordenada de valores. O objeto inicia com chaves de abertura e término, seguido do nome do atributo, dois pontos e o valor. Caso existam mais atributos, os mesmos devem ser separados por “,” (JSON, 1999).

Quadro 1 - Métodos HTTP Básicos

Método	**Descrição**
GET	Requisita dados de um recurso específico.
POST	Submete diversos dados para serem processados

	por um recurso específico.
PUT	Adiciona (ou modifica) um recurso na URI passada.
DELETE	Exclui um recurso específico.
HEAD, OPTIONS, TRACE	Mesma requisição de um GET, mas retorna apenas cabeçalhos HTTP.

Fonte: CAELUM (2017).

Figura 8 – Exemplo de formato JSON

Fonte: Elaborado pelos autores (2017).

Segundo Caelum (2017), quando a representação do recurso é enviada à aplicação, a aplicação provê uma resposta referente ao processamento desta representação, informando, por exemplo, se houve sucesso ou algum problema neste processamento. No caso do HTTP, essa informação é repassada através do da requisição, onde há um código que é formado por 3 dígitos. O primeiro dígito do código do status indica uma das cinco classificações de resposta e os restantes definem o status específico. As cinco classes de status são definidas no Quadro 2.

Quadro 2 - Status de retorno protocolo HTTP

Código	**Natureza**	**Descrição**
1XX	Informativa	Indica que a solicitação foi recebida e que o servidor está pronto para dar continuidade ao processo.

2XX	Sucesso	Indica que a solicitação foi recebida, entendida e que será processada com êxito pelo servidor.
3XX	Redirecionamento	Indica que será redirecionado a outra página. Isso acontece, por exemplo, quando o endereço da página (URI) pesquisada foi alterado, mas o site te redireciona para o novo endereço.
4XX	Erro do Cliente	Indica que o servidor não conseguiu processar a solicitação porque o cliente a fez de forma errada ou que não dependa dele, como por exemplo uma página excluída.
5XX	Erro do Servidor	Indica que, por um erro do servidor, a sua solicitação não pode ser atendida. Na maioria das vezes está relacionada a permissões dos arquivos ou pastas de software.

Fonte: Caelum (2017).

2.8.3 Ruby

Para a aplicação que intermedia o Arduino e a interface gráfica, foi definida a arquitetura de uma API REST utilizando o protocolo HTTP, de forma a integrar diferentes plataformas de maneira rápida e segura. Nesta aplicação foi utilizada a linguagem de programação *Ruby* com a ferramenta (*framework) Rails*.

Segundo OLIVEIRA JR. (2016), o *Ruby* é uma linguagem de programação interpretada multiparadigma, de tipagem dinâmica e forte, criada por Yukihiro Matsumoto no Japão, em 1995. Matsumoto, criou o *Ruby* pois queria uma linguagem de *script* que fosse mais poderosa do que Perl e mais orientada a objetos do que *Python*.

Ruby on Rails, também conhecido como apenas *Rails*, é um framework de desenvolvimento web escrito na linguagem *Ruby*. Desde sua criação em 2004, o *Rails* se tornou uma das mais populares e poderosas ferramentas para construção de aplicações web dinâmicas, sendo utilizado por empresas como Airbnb, Disney, GitHub, Shopify, Twitter, entre outras. O *Rails* é uma ferramenta de código aberto, disponibilizado através da licença de *software* MIT (*Massachusetts Institute of Technology*), o que a torna completamente gratuita (HARTL, 2016).

Além destas vantagens, o *Rails* possui uma comunidade de milhares de contribuidores, possui diversas conferências a nível mundial e uma vasta quantidade de bibliotecas para soluções de problemas comuns utilizando um gerenciador de pacotes chamado *RubyGems*, também conhecido como bibliotecas ou *gems*. Por fim, sua principal vantagem é que o *Rails* permite escrever menos código em relação a outras linguagens e *frameworks*, refletindo em alta produtividade e entregando mais soluções em menos tempo (RAILS GUIDE, 2017).

3 METODOLOGIA

A abordagem inicial foi através de uma pesquisa exploratória, buscando em mídias digitais, livros e artigos científicos, com a finalidade de identificar modelos existentes, para elaboração da aplicação.

O gerenciamento do projeto, optou-se na utilização da metodologia ágil no controle e na engenharia de *software*. No projeto foi empregado o método *Kanban*, que é um modelo visual de gerenciamento de processos, classificados em: *To Do* (à fazer), *Doing* (fazendo) e *Done* (concluído); em que é possível ter rapidamente um *feedback* da situação atual, utilizando a ferramenta *online* Trello.

Na elaboração da documentação de análise e engenharia do projeto, foram empregados o levantamento de requisitos funcionais e não-funcionais, utilizando desejos e estórias de usuário, diagrama de casos de uso e diagrama de classes.

A pesquisa elaborada foi de caráter quantitativo, para realizar a validação dos requisitos do sistema, obter uma noção de aceitação do mercado e viabilidade econômica deste projeto. Através de um questionário de múltipla escolha, disponibilizado na internet através da plataforma *Google Forms*.

4 DESENVOLVIMENTO

Durante a pesquisa exploratória realizada, permitiu entender como deve ser o desenvolvimento de um sistema que utiliza a arquitetura web, com a finalidade de controlar o nível e vazão de água em condomínios prediais, sendo apresentados nas próximas sessões.

4.1 Requisitos de Sistema

Os requisitos de sistema são as condições necessárias para satisfazer um objetivo ou expectativa dos clientes. Com isso, podemos entender que são as ações que um sistema deve fazer ou uma restrição no desenvolvimento do sistema, tendo por objetivo estabelecer e manter uma concordância com os clientes e envolvidos em cada etapa do projeto, definindo fronteiras, estimativa de custo e tempo de desenvolvimento, e por fim definindo uma interface de usuário (SOMMERVILLE, 2008).

Para classificar melhor, os requisitos estão subdivididos em duas etapas: Requisitos Funcionais (RF) e Requisitos Não-Funcionais (RNF).

4.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema se propõem a fazer, suas funções e informações. Abaixo segue relação de requisitos funcionais descritos em forma de *User Stories* (Estórias de Usuário), utilizando a metodologia ágil.

Para isto, foi utilizada a seguinte expressão para elaboração das estórias de usuário: “Como um [ator] eu quero [ação] para que seja possível [funcionalidade]”, facilitando a elaboração dos requisitos e permitindo estimar esforço para a realização de um objetivo. O Quadro 3 relaciona os desejos do usuário para o projeto e no Apêndice A relaciona-os com suas estórias.

Quadro 3 - Desejos do Usuário

Desejo 1: Eu como administrador do sistema, desejo realizar o controle de acesso

das requisições vindas até meu sistema.
Desejo 2: Eu como administrador do sistema, desejo realizar a manutenção dos reservatórios e equipamentos.
Desejo 3: Eu como cliente, desejo visualizar os dados exclusivos da minha conta.

Fonte: Elaborado pelos usuários (2017).

4.1.2 Requisitos Não Funcionais

Os requisitos não-funcionais tratam dos critérios de aceite e qualidade do sistema. Nos Quadros 4 a 7, são apresentados os requisitos não-funcionais da solução proposta.

Quadro 4 - RNF01, acessar o ambiente de forma rápida e eficiente.

Eu como cliente, desejo acessar o ambiente de forma rápida e eficiente.			
Identificador	RNF01	**Categoria**	Desempenho
Nome	Tempo de resposta limite de 500ms em 90% do tempo		
Versão	1	**Prioridade**	Essencial
Descrição	O sistema deverá prover recursos para processamento que possibilite o acesso simultâneo de 100 usuários de forma paralela, com o tempo de resposta limite de 500ms.		

Fonte: Elaborado pelos autores (2017).

Quadro 5 - RNF02, acessar o ambiente através da internet

Eu como cliente, desejo acessar o ambiente através da internet.			
Identificador	RNF02	**Categoria**	Usabilidade
Nome	Uso de Design responsivo nas interfaces gráficas		
Versão	1	**Prioridade**	Obrigatório
Descrição	A interface será construída para rodar em ambiente web e deverá possuir um design que se adapte a diferentes tamanhos de telas.		

Fonte: Elaborado pelos autores (2017).

Quadro 6 - RNF03, acesso utilizando navegador

Eu como cliente, desejo acessar o ambiente utilizando o navegador de um *smartphone*, *tablet* ou computador.			
Identificador	RNF03	**Categoria**	Compatibilidade
Nome	Compatível com principais browsers		
Versão	1	**Prioridade**	Essencial
Descrição	A interface será construída para rodar em ambiente web, sendo compatível com os principais navegadores disponíveis no mercado: Microsoft Edge, Internet Explorer 11, Google Chrome, Firefox e Safari.		

Fonte: Elaborado pelos autores (2017).

Quadro 7 - RNF04, separação em camadas usando padrão MVC

Eu como desenvolvedor, desejo realizar a separação em camadas utilizando o padrão MVC de programação.			
Identificador	RNF04	**Categoria**	Arquitetura
Nome	Divisão arquitetural do sistema em camadas para desacoplamento		
Versão	1	**Prioridade**	Essencial
Descrição	O projeto do software deverá ser fortemente orientado a baixo acoplamento e alta coesão, primando pela melhor separação de responsabilidades. Todo o projeto deverá ser feito utilizando uma arquitetura separada em camadas, onde cada camada conterá apenas os algoritmos relacionados à sua responsabilidade. Abaixo as camadas que deverão ser utilizadas, e suas responsabilidades: **Interface:** abrigar lógicas de tela, validação de campos, acionamento de comandos, códigos para design de interface etc. **Negócio:** abrigar lógicas de negócio, onde será codificado o escopo das regras de negócio associadas aos requisitos		

	funcionais pertinentes à funcionalidade. **Dados:** abrigar lógicas de acesso a dados, comandos SQL ou comandos para utilização de mecanismos de persistência utilizado, para o caso de uso de ORM. **Segurança:** abrigar lógicas de autenticação, auditoria, manutenção de usuários. **Infraestrutura:** abrigar lógicas não relacionadas a interfaces gráficas, regras de negócio, dados ou segurança, mas que poderão ser utilizadas em todas estas camadas. Conterá recursos para gravação de logs, transferência de arquivos, mensagens, envio/recepção de e-mails etc.

Fonte: Elaborado pelos autores (2017).

4.2 Descrição do Projeto

O projeto é uma solução mista de sistema embarcado capaz de monitorar de forma inteligente o nível e o consumo de água de reservatórios. Toda a interação com o morador / usuário do sistema é via internet, utilizando o conceito de IoT.

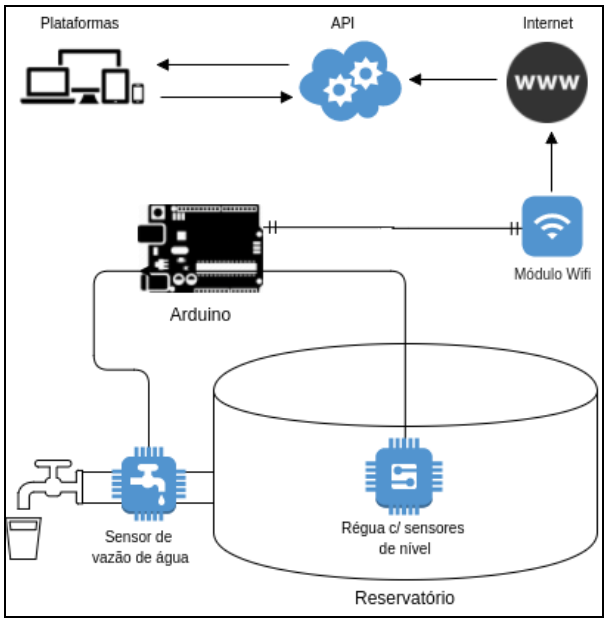
O projeto tem as seguintes funcionalidades: calcular o consumo e vazão de água utilizando um sensor de fluxo de água; identificar o nível de água utilizando uma régua com diversos sensores; informar esses dados ao sistema utilizando uma API RESTful através do dispositivo *wireless*; permitir interação do usuário com o sistema via Internet (utilizando qualquer navegador, inclusive através de dispositivos móveis), possibilitando a visualização da vazão instantânea (L/h) e nível de água (volume).

A Figura 9 permite entender o funcionamento do sistema, abordando as diferentes áreas da arquitetura proposta.

Para o controle do código fonte do projeto foi utilizado o *Git*, que segundo SCHMITZ (2015), é um sistema de controle de arquivos distribuído no qual diferentes pessoas podem contribuir simultaneamente no mesmo projeto, criando e alterando arquivos sem que haja risco de conflitos e perda de informações, mantendo um histórico de todas as mudanças realizadas. Essas mudanças são

armazenadas em um repositório, sendo este local ou remoto. Segundo a característica distribuída do Git, os repositórios locais são armazenados nos computadores de cada desenvolvedor, já os repositórios remotos geralmente estão na internet. No caso deste projeto foi utilizado o *GitHub*, que oferece diversas funcionalidades extras ao *Git*, além também de ser um serviço de uso gratuito.

Figura 9 - Visão Geral da Arquitetura



Fonte: Elaborado pelos autores (2017).

Por mais que o projeto tenha a proposta de controlar o nível e vazão de água de reservatórios, possui diferentes áreas, que são o desenvolvimento do dispositivo Arduino, a API e a Interface. Por este motivo, foram criados 3 repositórios distintos no *GitHub*, sendo eles:

a) API: https://github.com/fredw/senac-tcs-api.git

b) Arduino: https://github.com/diogoottequir/senac-tcs-arduino

c) Interface: https://github.com/fredw/senac-tcs-web

O projeto foi controlado através de metodologia ágil, utilizando um *Kanban* para que de forma rápida e simples pudesse obter o status de determinada cada etapa. A Figura 10 descreve como foi elaborado este quadro para controle e gerenciamento do projeto.

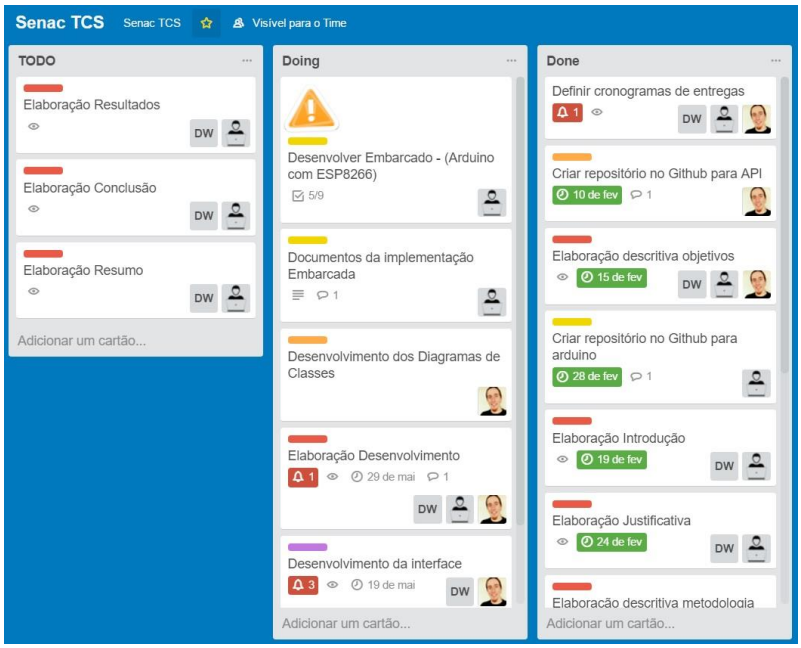
Ao optar pelo uso do desenvolvimento ágil, não se pode ignorar o fator de

integração contínua, que é um dos pilares do processo de desenvolvimento de software, garantindo que o sistema funcione a cada nova revisão de forma eficiente.

> Integração Contínua é uma prática de desenvolvimento de software onde os membros de um time integram seu trabalho frequentemente, geralmente cada pessoa integra pelo menos diariamente – podendo haver múltiplas integrações por dia. Cada integração é verificada por um build automatizado (incluindo testes) para detectar erros de integração o mais rápido possível. Muitos times acham que essa abordagem leva a uma significante redução nos problemas de integração e permite que um time desenvolva software coeso mais rapidamente (FOWLER, 2006, tradução nossa).

Além de processos, a metodologia ágil requer que documentos de suma importância sejam elaborados, um deles é o Diagrama de Caso de Uso, que descreve um cenário do projeto.

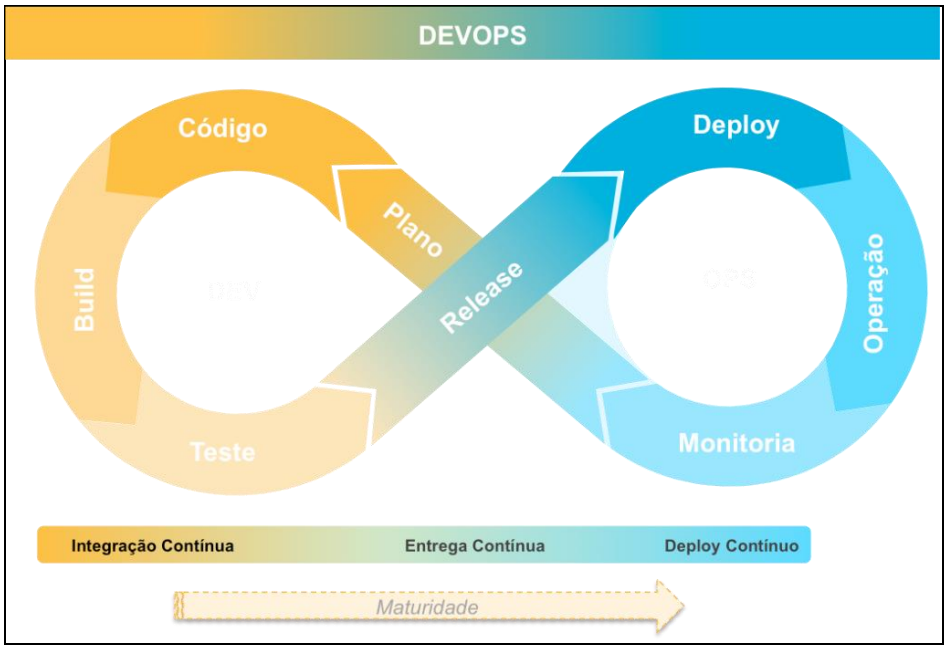
Figura 10 - *Kanban* para Controle e Gerenciamento do Projeto



Fonte: Elaborado pelos autores (2017).

Na Figura 11 é demonstrado o processo de integração contínua.

Figura 11 - Processo de Integração Contínua



Fonte: FERNANDES, 2015.

4.3 Tecnologias

As tecnologias adotadas no desenvolvimento do projeto foram divididas de acordo com o domínio de cada parte, bem como a infraestrutura requerida.

4.3.1 Sistema Embarcado Arduino

As principais vantagens do Arduino identificadas foram: baixo custo, na qual as placas Arduino podem ser montadas manualmente ou adquiridas em versões pré-montadas que custam menos de US$ 50; multi-plataforma, na qual o *software* Arduino (IDE - *Integrated Development Environment*) é executado em sistemas operacionais Windows, Mac OS e Linux; ambiente de programação simples e claro, na qual o *software* Arduino (IDE) é fácil de usar para iniciantes, mas flexível o suficiente para que os usuários avançados também possam aproveitar; *software* de fonte aberta e extensível, na qual é publicado como ferramentas de código aberto, disponíveis para a extensão por programadores experientes, podendo ser expandido através de bibliotecas C/C++ e as pessoas que querem entender os detalhes técnicos podem fazer o salto do Arduino para a linguagem de programação AVR C

em que se baseia; código aberto e *hardware* extensível, na qual os planos das placas Arduino são publicados sob uma licença *Creative Commons*, para que designers de circuitos experientes possam fazer sua própria versão do módulo, estendê-la e melhorá-la.

Para que seja possível a aplicação do conceito de IoT ao dispositivo microprocessado Arduino, existe uma extensa lista de periféricos que adicionam recursos e funcionalidades. Os periféricos utilizados no dispositivo desenvolvido neste trabalho são: régua com sensores de nível de água, controle de vazão, módulo de Wi-Fi, módulo *display* OLED e o módulo SD.

Os sensores de nível de água permitem identificar qual o volume contido no reservatório, que é definido por uma régua, conforme demonstrado na Figura 12, construída com os sensores de nível e canos PVC de 20mm, posicionando os sensores em três níveis diferentes: o mais baixo em 7,5 litros (0,0075 m^3), o do meio em 11 litros (0,011 m^3) e por último o de 15 litros (0,015 m^3). Estes valores devem ser calculados conforme a régua construída e sensores posicionados, variando conforme a capacidade do reservatório.

Figura 12 – Régua com sensores de nível de água

Fonte: Elaborado pelos autores (2017).

O módulo de vazão permite coletar os dados de consumo na saída do reservatório. O sensor utilizado neste trabalho é o modelo YF-S201, que possui a

limitação de até 30 litros por minuto, podendo emitir até 450 pulsos por litro de água, suportando temperatura ambiente entre -25ºC até 80ºC e sua alimentação é de 5VDC (Volt Corrente Continua) consumindo 15mA. Conforme demonstrado na Figura 13, o sensor de vazão está ligado a um registro do tipo esfera de PVC com espessura de 20mm.

O módulo Wi-Fi, que permite acesso à internet através de uma rede sem fio, está demonstrado na Figura 14. Este módulo recebe os comandos emitidos pelo dispositivo Arduino através da comunicação serial, que envia os dados coletados para a API. O modelo utilizado neste trabalho é o módulo ESP8266-01, e suas características principais são: padrão 802.11b/g/n; Wi-Fi *Direct* (P2P), *soft-AP;* sensor de temperatura incorporado e consumo de energia em espera de menos de 1,0mW (miliwatt).

Figura 13 – Sensor de vazão

Fonte: Elaborado pelos autores (2017).

O módulo de *display OLED* permite a visualização dos eventos executados pelo Arduino, facilitando a manutenção e instalação do dispositivo embarcado. A resolução deste *display* é de 128 x 64 pixels e 0,96 polegadas, sendo que não possui luz de fundo pois este já contém luz própria, obtendo um consumo mais baixo do que os *displays* LCDs. Na Figura 15 é apresentada a sequência das informações disponibilizadas através do *display* na inicialização do equipamento, sendo elas:

iniciando o dispositivo (a); carregando módulo SD (b); módulo SD carregado com sucesso (c); definindo módulo Wi-Fi como modo de operação (d); módulo Wi-Fi definido com sucesso (e); conectando com a rede Wi-Fi (f); definindo módulo Wi-Fi como em modo Single/TCP (g); modo do módulo Wi-Fi definido com sucesso (h); tela final com a exibição dos dados de vazão e nível do reservatório (i).

Figura 14 – Módulo Wi-Fi

Fonte: Elaborado pelos autores (2017).

O módulo SD permite o armazenamento de arquivos suportando os formatos do tipo FAT16 e FAT32. No cartão de memória SD é definido um arquivo onde estão contidas as configurações para acesso à internet e as configurações dos sensores. Conforme demonstrado na Figura 16, este módulo está localizado logo abaixo do *Display OLED*.

Os sensores de nível de água e vazão são conectados ao dispositivo através de conectores do modelo STLZ950 4 vias, que facilitam a instalação e manutenção dos componentes. Estes conectores estão localizados na parte lateral do dispositivo, conforme demonstrado na Figura 17.

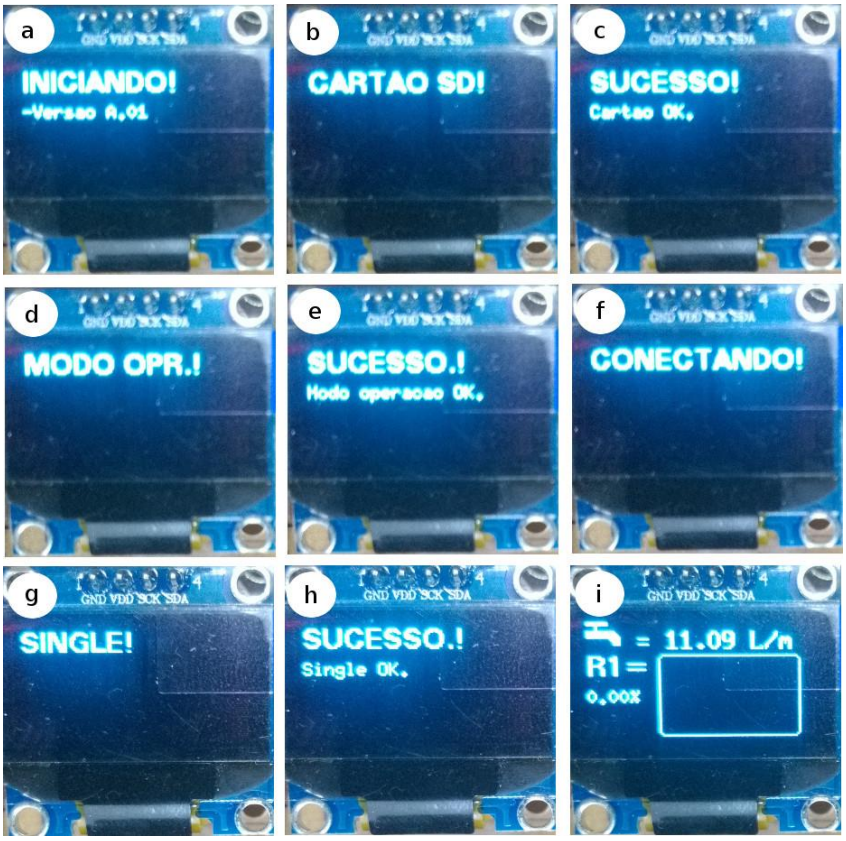
O Arduino é alimentado através de uma fonte bivolt 110V/220VAC (Volt Corrente Alternada) de rede convencional com saída de 9VDC (Volt Corrente Contínua) de tensão e 1A (Ampere) de corrente. Na Figura 18, estão demonstrados

o conector para a alimentação e o conector USB tipo B (semelhante à de uma impressora), que é utilizado para a instalação e atualização do *firmware* no SE.

O desenvolvimento foi realizado utilizando a ferramenta de programação (IDE) Visual Studio 2017 utilizando o *plug-in* Visual Micro, integrado ao GitHub para controle e versionamento dos códigos fonte. A linguagem utilizada para a geração do *firmware* do Arduino foi C/C++, utilizando a programação estruturada, dividida em funções. A

Figura 19 demonstra a IDE utilizada para a programação do dispositivo.

Figura 15 - Sequência de telas de inicialização no Display OLED

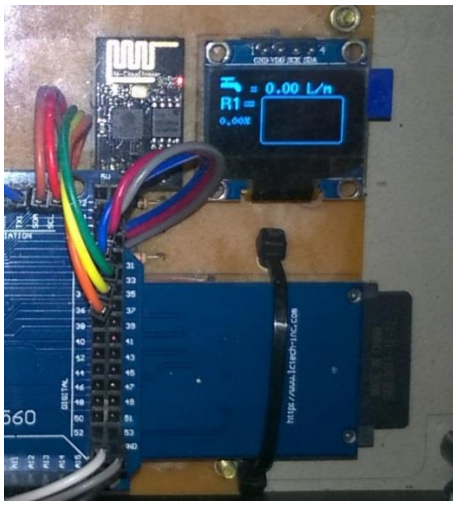


Fonte: Elaborado pelos autores (2017).

Para facilitar o controle dos periféricos instalados ao Arduino, foram utilizadas diversas bibliotecas, algumas disponibilizadas pela própria IDE do Arduino e outras disponibilizadas por terceiros. No caso de controle do módulo de internet sem fio utilizou-se a biblioteca ESP8266 disponibilizada por terceiros, que permite a comunicação com o módulo de forma mais simplificada através de métodos e

parâmetros. A Figura 20 apresenta um modelo padrão para conexão com o Wi-Fi utilizando a biblioteca ESP8266.

Figura 16 - Módulo de cartão SD



Fonte: Elaborado pelos autores (2017).

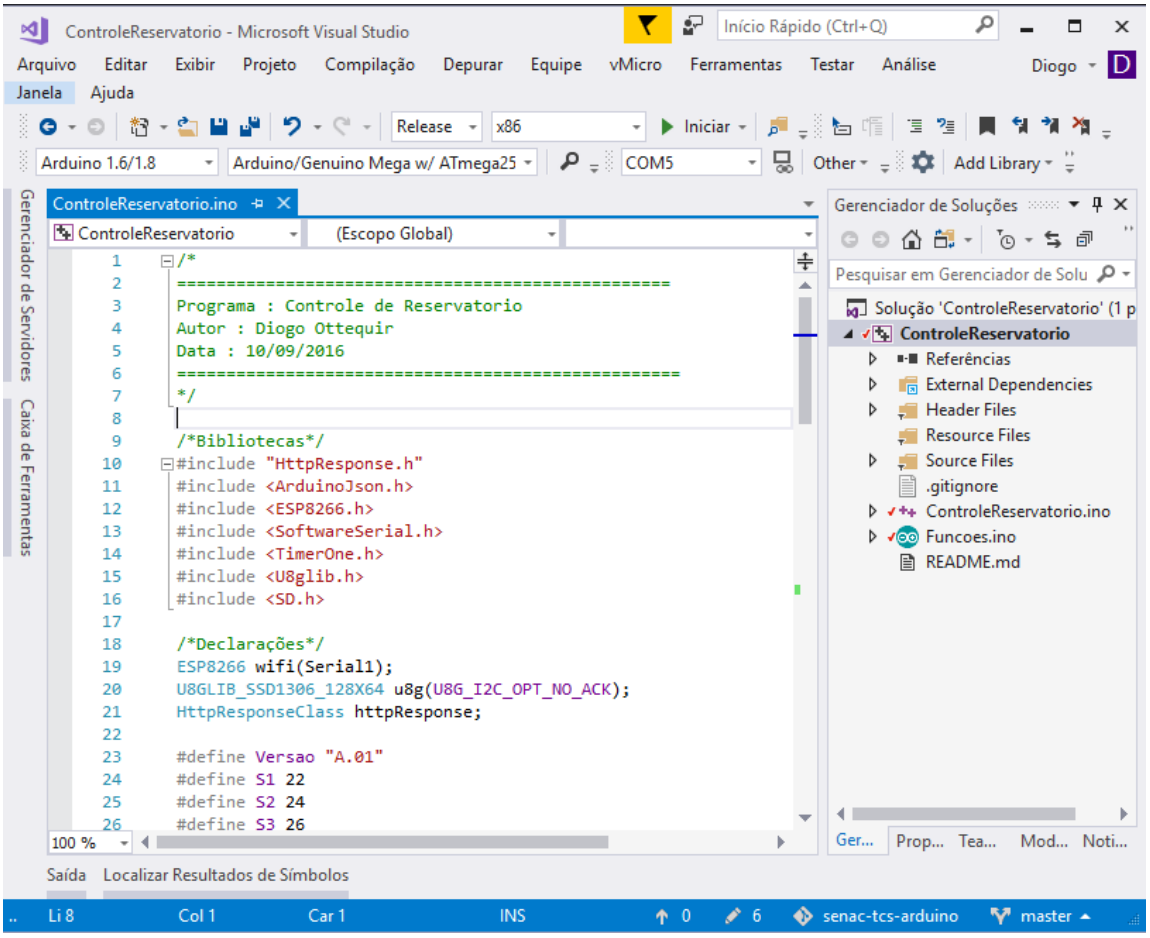
Figura 17 - Conexão dos sensores

Fonte: Elaborado pelos autores (2017).

Figura 18 - Entrada de alimentação e porta USB para atualização do *firmware*

Fonte: Elaborado pelos autores (2017).

Figura 19 - IDE Visual Studio 2017



Fonte: Elaborado pelos autores (2017).

Para o controle do módulo de cartão SD, foi utilizada uma biblioteca disponibilizada pela própria IDE, que facilita na leitura dos arquivos no dispositivo. Na Figura 21 é demonstrado um exemplo do arquivo de configuração do Wi-Fi (wifi.json.txt) contido dentro do cartão SD, que possui uma estrutura em JSON com os dados para conexão com a Internet.

Com a configuração inserida no cartão SD, a biblioteca efetua a leitura do arquivo e carrega o conteúdo em uma variável do tipo *string*. Na Figura 22 é apresentada a função que efetua a leitura do arquivo contido no cartão, e se caso o arquivo não for localizado, é apresentado no *display* a mensagem de este não foi encontrado.

Figura 20 - Modelo de utilização da biblioteca ESP8266

```
#include "ESP8266.h"
#include "SoftwareSerial.h"

// Seta a porta Serial1 do Arduino para comunicação com o modulo WI-FI
ESP8266 wifi(Serial1);

// Configuracao ID e senha da rede Wireless
#define SSID "NOME_DA_REDE"
#define PASSWORD "SENHA_DA_REDE"
void setup()
{
    Serial.begin(9600);
}
void loop(void)
{
    Serial.print("Inicializando modulo\r\n");
    Serial.print("Versao do firmware: ");
    Serial.println(wifi.getVersion().c_str());
    // Define modo de operacao como STA (station)
    if (wifi.setOprToStation())
    {
        Serial.print("Modo STA ok\r\n");
    }
    else
    {
        Serial.print("Erro ao definir modo STA !r\n");
    }
    // Conexao a rede especificada em SSID
    if (wifi.joinAP(SSID, PASSWORD))
    {
        Serial.print("Conectado com sucesso a rede wireless\r\n");
        Serial.print("IP: ");
        Serial.println(wifi.getLocalIP().c_str());
    }
    else
    {
        Serial.print("Erro ao conectar rede wireless !!!\r\n");
    }
    Serial.print("*** Fim ***\r\n");
    while (1) {}
}

```

Fonte: Elaborado pelos autores (2017).

Figura 21 - Arquivo de configuração wifi.json.txt

```
{
  "wifi": {
    "ssid": "NOME DA REDE",
    "password": "SENHA DA WIFI"
  }
}
```

Fonte: Elaborado pelos autores (2017).

Figura 22 - Leitura do arquivo de configuração wifi.json.txt

```
50  String lerArquivoConfigWifi() {
51      String json = "";
52      File dataFile = SD.open(F("wifi.json.txt"), FILE_READ);
53      if (dataFile)
54      {
55          while (dataFile.available())
56          {
57              char linha = dataFile.read();
58              json += linha;
59          }
60          dataFile.close();
61      }
62      else
63      {
64          mensagem(F("ERRO!"), F("arquivo wifi nao encontrado."), false);
65      }
66      dataFile.close();
67      return json;
68  }
```

Fonte: Elaborado pelos autores (2017).

A leitura dos sensores de nível de água do reservatório é feita de forma digital, onde os sensores funcionam como chave liga/desliga, da mesma forma que é feita a leitura do sensor de vazão, que emitem pulsos conforme a passagem de água dentro do cano conectado a saída do reservatório.

A requisição efetuada pelo Arduino, através do módulo de internet sem fio, utiliza os padrões definidos no protocolo HTTP para transportar os dados a serem enviados para a API, utilizando o formato de dados JSON. Para auxiliar e facilitar a montagem e leitura deste formato, utilizou-se a biblioteca *ArduinoJson* que é disponibilizada por terceiros.

Para coletar o valor de um determinado campo dentro desta estrutura, foi implementada a função *buscaAutenticacaoWifi*, que recebe como parâmetro uma

String com o conteúdo do arquivo JSON, conforme apresentado na Figura 23, definindo as variáveis *SSID* e *PASSWORD* de acordo com os parâmetros do arquivo JSON.

Para a utilização do display OLED foi utilizada a biblioteca *U8glib,* que permite imprimir textos e objetos gráficos, sendo possível representar formas geométricas como quadrados, círculos e linhas. Estas formas são redimensionadas através de parâmetros que definem a largura, a altura e o posicionamento, identificando a linha e coluna onde o objeto deve ser posicionado no *display*.

Figura 23 - Leitura dos dados do arquivo JSON

```
137  void buscaAutenticacaoWifi(String json)
138  {
139      SSID = "";
140      PASSWORD = "";
141      DynamicJsonBuffer jsonBuffer;
142      JsonObject& root = jsonBuffer.parseObject(json);
143      if (root.success())
144      {
145          SSID = root["wifi"]["ssid"].asString();
146          PASSWORD = root["wifi"]["password"].asString();
147      }
148      root.end();
149  }
```

Fonte: Elaborado pelos autores (2017).

Na Figura 24, entre as linhas 108 e 114, estão as especificações para o desenho da torneira do display [Figura 15(i)], representando o ícone do sensor de vazão. Na linha 116 é atribuído o tamanho da fonte do texto, sendo indispensável sua definição antes da exibição, e na linha seguinte é definido e exibido o valor da vazão atual.

Figura 24 - Montagem da visualização do consumo no display

```
106  void mostraConsumo(String vz)
107  {
108      u8g.drawBox(0, 0, 12, 2);
109      u8g.drawBox(0, 5, 16, 4);
110      u8g.drawBox(13, 6, 4, 4);
111      u8g.drawBox(14, 7, 4, 4);
112      u8g.drawBox(15, 8, 4, 3);
113      u8g.drawBox(15, 10, 4, 3);
114      u8g.drawVLine(5, 0, 9);
115
116      u8g.setFont(u8g_font_8x13B);
117      u8g.drawStr(20, 15, vz.c_str());
118  }
```

Fonte: Elaborado pelos autores (2017).

Na Figura 25 está ilustrado o trecho do código-fonte utilizado para desenhar o reservatório e seu respectivo nível atual.

Este conjunto de periféricos e bibliotecas, juntamente com o dispositivo microprocessado Arduino, efetuam a leitura e processamento dos dados emitidos pelos sensores, enviando-os a API, que é responsável por tratar, armazenar e disponibilizar estes dados.

Figura 25 - Montagem da visualização do nível no display

```
120  void mostraNivel()
121  {
122      float nivel = 0;
123
124      u8g.setFont(u8g_font_fub11r);
125      u8g.drawStr(0, 30, F("R1="));
126      u8g.drawRFrame(40, 18, 80, 46, 3);
127
128      if (digitalRead(S1) == 1)
129      {
130          u8g.drawBox(44, 55, 72, 5);
131          nivel = 16.67;
132      }
133      if (digitalRead(S2) == 1) { ... }
138      if (digitalRead(S3) == 1) { ... }
143      if (digitalRead(S4) == 1) { ... }
148      if (digitalRead(S5) == 1) { ... }
153      if (digitalRead(S6) == 1) { ... }
158
159      //PERCENTUAL
160      String n = F("");
161             n += nivel;
162             n += F("%");
163
164      u8g.setFont(u8g_font_6x10r);
165      u8g.drawStr(0, 45, n.c_str());
166  }
```

Fonte: Elaborado pelos autores (2017).

4.3.2 API

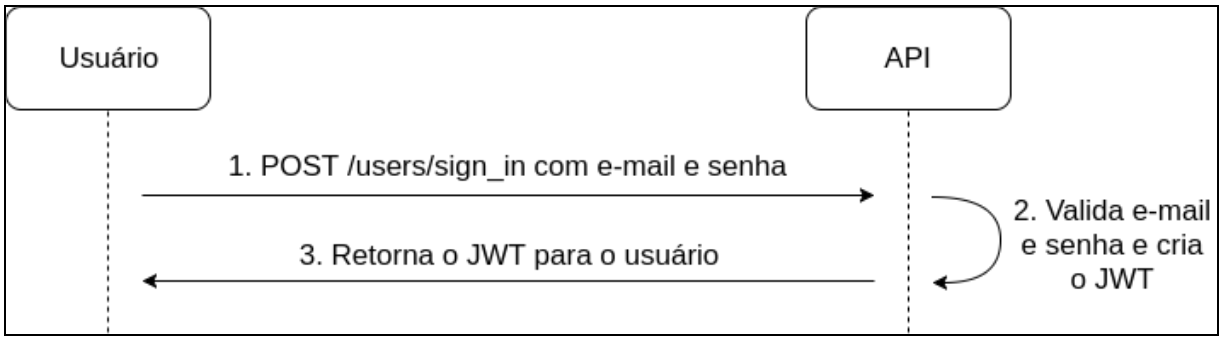
Uma API pública permite que qualquer pessoa possa enviar ou receber dados através dos endereços de acessos às rotas (URI), disponibilizada através da internet, o que impede a identificação da origem da requisição e consequentemente torna a aplicação insegura. A aplicação construída neste trabalho exige a identificação, pois conforme os requisitos funcionais, é necessário que se saiba quem está solicitando ou enviando os dados à API, identificando o cliente, usuário ou administrador. No caso do *Rails*, para este projeto foi utilizada a biblioteca chamada *Devise*, que é uma solução de autenticação flexível, incluindo

funcionalidades como: autenticação do usuário, registro de novos usuários, confirmação por e-mail de novos usuários, recuperação de senha do usuário, contagem e data e horários de acessos.

Para essa identificação é utilizado um conceito de *token* JWT (*JSON Web Token*), onde antes de requisitar quaisquer recursos, é necessário primeiramente que seja feita a autenticação na API, informado um e-mail e senha válidos para este endereço da rota de autenticação, conforme as rotas relacionadas no Apêndice B. Para este projeto, foi utilizado o *Devise* JWT, que é uma extensão do *Devise* que permite retornar um *token* na autenticação, funcionalidade que não é incluída por padrão no *Devise*.

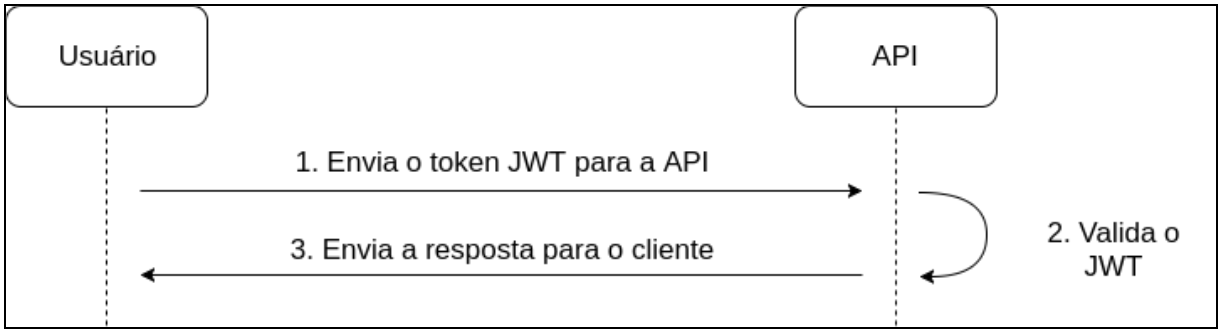
Segundo Jscrambler (2016), o JWT é uma estratégia simples e segura para autenticação de APIs REST. O mecanismo de autenticação funciona da seguinte maneira: o usuário faz uma única requisição enviando as credenciais de e-mail e senha e a API valida as credenciais e, se tudo estiver certo, ele retorna para o usuário um *token*, conforme representado na Figura 26; após receber o *token*, o usuário pode armazená-lo da forma que preferir, sendo que toda vez que necessitar acessar uma rota que requer autenticação, ele apenas envia esse *token* para a API, onde o servidor valida o *token*, permitindo ou negando a requisição ao usuário, exemplificado na Figura 27.

Figura 26 - Diagrama de Fluxo da autenticação na API



Fonte: Elaborado pelos autores (2017).

Figura 27 - Diagrama de Fluxo de uma requisição na API



Fonte: Elaborado pelos autores (2017).

Segundo FERREIRA (2015), a autenticação não é a única preocupação necessária em relação à segurança. Além da autenticação é necessário verificar também a autorização que um usuário tem em relação ao acesso a um recurso. Conforme os requisitos funcionais estabelecidos neste trabalho, há situações em que apenas administradores podem ter acesso a um recurso e usuários comuns não. Esta situação também se aplica em casos mais específicos, como por exemplo, um usuário pode ver apenas seus próprios dados, ou uma segunda situação onde usuários que pertencem a um cliente podem ver apenas os dados deste próprio, sem acesso às informações de outro cliente. Para esse tipo de restrição foi utilizado a biblioteca Pundit, que fornece um conjunto de funcionalidades para construir um sistema de autorização simples, robusto e escalável utilizando *Ruby*.

Afim de garantir a confiabilidade que a solução deve apresentar, foram empregadas no projeto técnicas de qualidade de *software* em todo o processo de desenvolvimento da API. As técnicas adotadas, seguindo a metodologia ágil de desenvolvimento foram, a elaboração de testes automatizados vinculado a um processo de integração contínua. Desta forma, ao longo do tempo o *software* poderá ser modificado e evoluído com a integridade e confiabilidade requeridas pelo cliente (MOLINARI, 2013).

Além disso, a qualidade em *software* garante um custo baixo e menor tempo de implantação e suporte. Por isso, os testes abrangem a busca de possíveis falhas nas alterações e implementações, garantindo que todos os requisitos sejam atendidos (RIOS, MOREIRA; 2013).

Para assegurar que os requisitos funcionais estabelecidos no projeto sejam

cumpridos, foi necessário definir diversos tipos de testes, englobando as diversas esferas de uma aplicação desenvolvida em *Rails*, tais como: domínio de negócios e validações, endereços das rotas, autorizações e por fim formatos e dados de retorno das requisições. Para implementação e verificação destes testes, foi utilizado o RSpec, que é uma ferramenta de Desenvolvimento Guiado por Comportamento que realiza estes testes utilizando uma linguagem de fácil entendimento.

A Figura 28 ilustra as especificações de negócio dos reservatórios, exigindo que a entidade tenha relacionamento com grupos de reservatórios, clientes e que tenha diversos dispositivos, além também de validar a obrigatoriedade e tamanho das informações.

A Figura 29 ilustra as especificações de autorizações dos reservatórios, declarando que um administrador pode listar, ver o detalhe, criar um novo registro, alterar e excluir.

A Figura 31 ilustra as especificações da requisição da listagem dos reservatórios, onde valida a quantidade de registros listados conforme o teste definido de 10 registros, bem como a validação do status da resposta HTTP, sendo o código 200 (Sucesso).

A Figura 31 ilustra as especificações das rotas dos reservatórios, garantindo que os endereços estabelecidos para este recurso estejam corretos.

A Figura 32 ilustra a execução de todos os testes da aplicação, mostrando que 419 testes foram realizados e houve 0 falhas.

Para o transporte dos dados entre o Arduino/API e API/Interface foi utilizado o formato JSON. Na Figura 33 é demonstrado um modelo de informações JSON da API, exibindo o retorno da rota de listagem dos reservatórios disponíveis.

Figura 28 - Teste no RSpec para a entidade de reservatório

```
1   require 'rails_helper'
2
3   RSpec.describe Reservoir, type: :model do
4     it { should belong_to(:reservoir_group) }
5     it { should belong_to(:customer) }
6     it { should have_many(:devices).dependent(:destroy) }
7     it { should validate_presence_of(:name) }
8     it { should validate_presence_of(:volume) }
9     it { should validate_length_of(:name).is_at_most(100) }
10    it { should validate_length_of(:description).is_at_most(250) }
11  end
12
```

Fonte: Elaborado pelos autores (2017).

Figura 29 - Teste no RSpec para as autorizações de reservatórios

```ruby
require 'rails_helper'

RSpec.describe ReservoirPolicy do
  subject { described_class.new(user, reservoir) }

  let(:reservoir) { create(:reservoir) }

  context 'with an admin user' do
    let(:user) { create(:user_admin) }
    it { is_expected.to permit_actions([:index, :show, :create, :update, :destroy]) }
  end

  context 'without an admin user' do
    let(:user) { create(:user) }
    it { is_expected.to permit_action(:index) }
    it { is_expected.to forbid_actions([:create, :update, :destroy]) }

    describe 'when user\'s customer is different' do
      let(:reservoir) { create(:reservoir, customer: create(:customer)) }
      let(:user) { create(:user, customer: create(:customer)) }
      it { is_expected.to forbid_action(:show) }
    end

    describe 'when user\'s customer is equal' do
      it { is_expected.to permit_action(:show) }
    end
  end
end
```

Fonte: Elaborado pelos autores (2017).

Figura 30 - Teste no RSpec para a requisição de listagem de reservatórios

```ruby
require 'rails_helper'

RSpec.describe 'Reservoir', type: :request do

  let(:reservoir) { create(:reservoir) }
  let(:reservoir_id) { reservoir.id }
  let(:customer) { create(:customer) }
  let(:headers_admin) do
    {
      'Authorization': sign_in(create(:user_admin)),
      'Content-Type': 'application/json'
    }
  end

  describe 'GET /reservoirs' do
    let!(:reservoirs) { create_list(:reservoir, 10) }

    context 'when user is an admin' do
      before { get '/reservoirs', headers: headers_admin }

      it 'returns reservoirs' do
        expect(json['data'].size).to eq(10)
      end

      it 'returns status code 200 Success' do
        expect(response).to have_http_status(:success)
      end
    end

    context 'when paginated' do
      before { get '/reservoirs?page=1&per_page=2', headers: headers_admin }

      it 'returns paginated reservoirs' do
        expect(json['data'].size).to eq(2)
      end

      it 'returns total records header' do
        expect(response.headers['Total']).to eq('10')
      end

      it 'returns total per page header' do
        expect(response.headers['Per-Page']).to eq('2')
      end
    end
  end
```

Fonte: Elaborado pelos autores (2017).

Além do retorno, o formato JSON também é utilizado para enviar dados e persistir novos registros, para isso é necessário informá-lo no corpo da requisição HTTP, conforme demonstrado na Figura 34, onde está sendo criado um novo reservatório através do método HTTP POST.

Para melhor entendimento da estrutura da API, na Figura 35, está ilustrada a estrutura das classes através do diagrama de classes.

A API, utilizando o formato de dados JSON, tem apenas responsabilidade de transportar os dados entre o Arduino e a Interface gráfica, mas esses dados necessitam estar de alguma forma armazenados, para que possam ser recuperados posteriormente, possibilitando a análise de históricos e etc. Para a persistência desses dados foi utilizado o PostgreSQL na versão 9.6.2, que é um Sistema Gerenciador de Banco de Dados (SGBD), onde são armazenadas todas essas informações arquitetadas no diagrama de classes, usando o conceito de banco de dados relacional, em que os dados são relacionados em tabelas, possuindo vínculo entre elas.

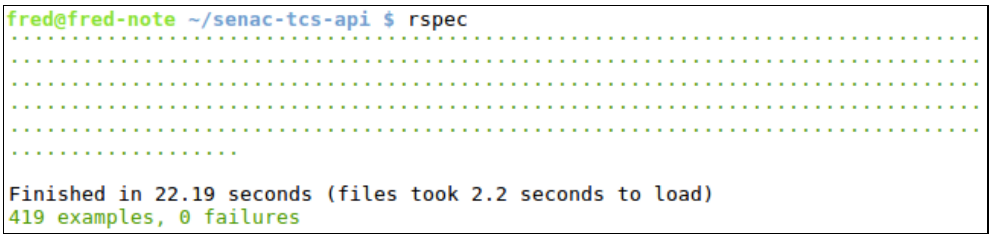
Figura 31 - Teste no RSpec para as rotas de reservatórios

```ruby
require 'rails_helper'

RSpec.describe V1::ReservoirsController, type: :routing do
  describe 'routing' do

    it 'routes to #index' do
      expect(:get => '/reservoirs').to
        route_to('v1/reservoirs#index')
    end

    it 'routes to #show' do
      expect(:get => '/reservoirs/1').to
        route_to('v1/reservoirs#show', :id => '1')
    end

    it 'routes to #create' do
      expect(:post => '/reservoirs').to
        route_to('v1/reservoirs#create')
    end

    it 'routes to #update via PUT' do
      expect(:put => '/reservoirs/1').to
        route_to('v1/reservoirs#update', :id => '1')
    end

    it 'routes to #update via PATCH' do
      expect(:patch => '/reservoirs/1').to
        route_to('v1/reservoirs#update', :id => '1')
    end

    it 'routes to #destroy' do
      expect(:delete => '/reservoirs/1').to
        route_to('v1/reservoirs#destroy', :id => '1')
    end
  end
end
```

Fonte: Elaborado pelos autores (2017).

No caso do *Rails*, para manutenção dessa estrutura de tabelas, é utilizado o conceito de *Migrations*, que é uma forma conveniente de alterar a estrutura de um banco de dados de forma estruturada e organizada, mantendo um histórico de alterações realizadas durante o desenvolvimento da aplicação, permitindo gerenciar a evolução de um *schema* usado pelo banco de dados físico. Na Figura 36 é possível visualizar a estrutura da *migration* inicial da estrutura de reservatórios no Rails, criando a tabela, seus respectivos campos e relacionamentos.

Figura 32 - Execução de todos os testes no RSpec



Fonte: Elaborado pelos autores (2017).

Figura 33 - Rota de retorno da listagem de reservatórios

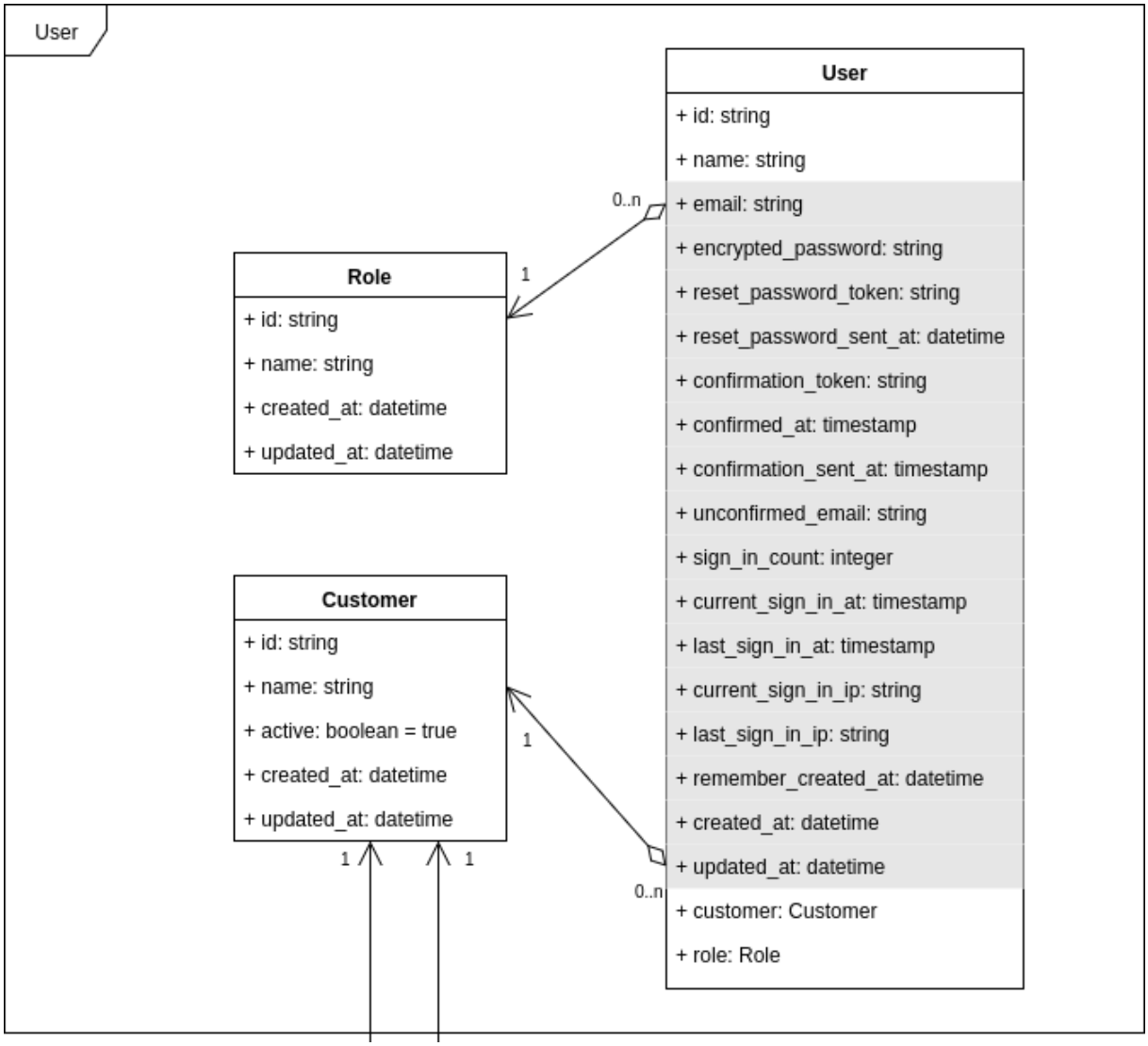
Fonte: Elaborado pelos autores (2017).

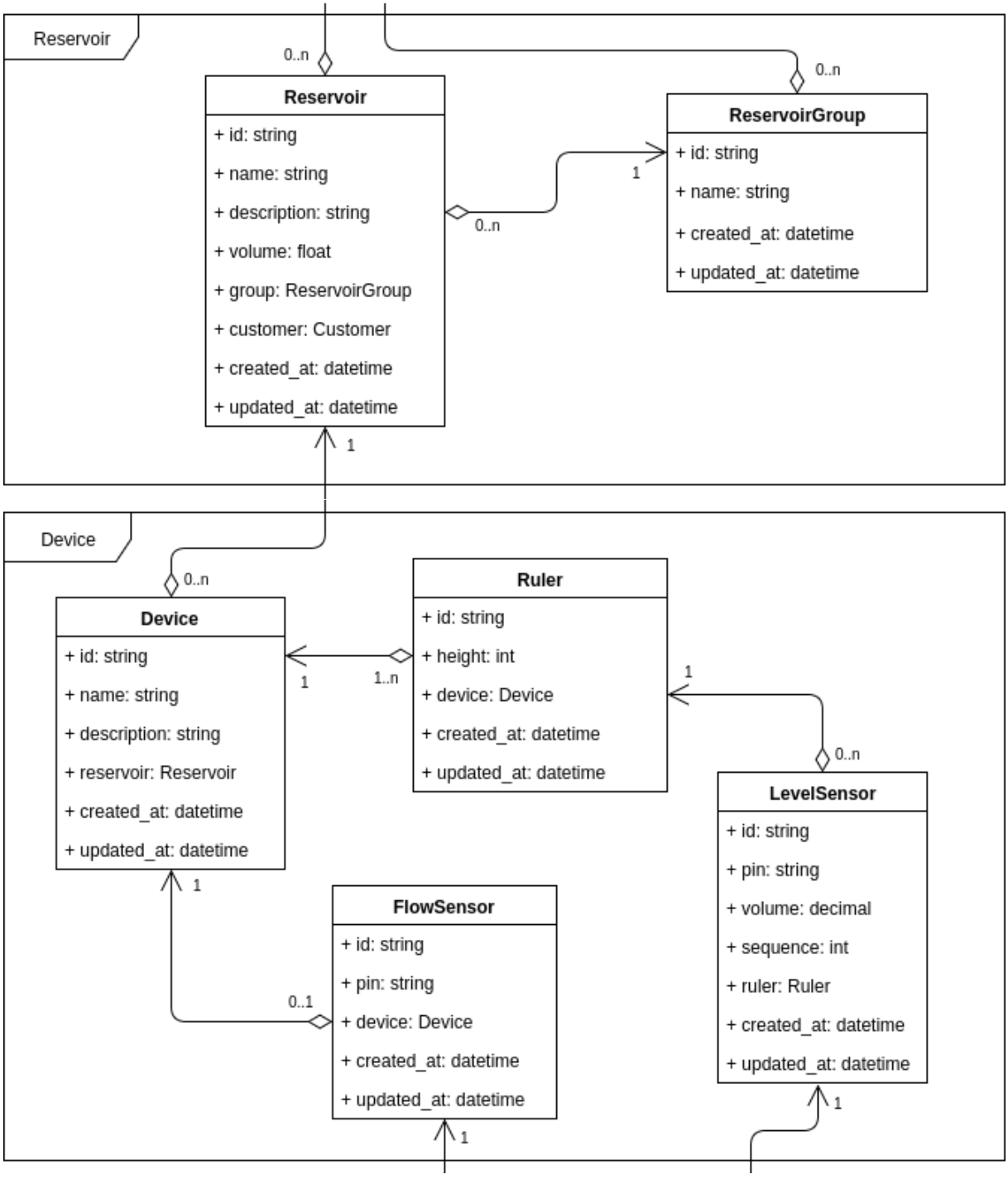
Figura 34 - Corpo da requisição para cadastro de novo reservatório

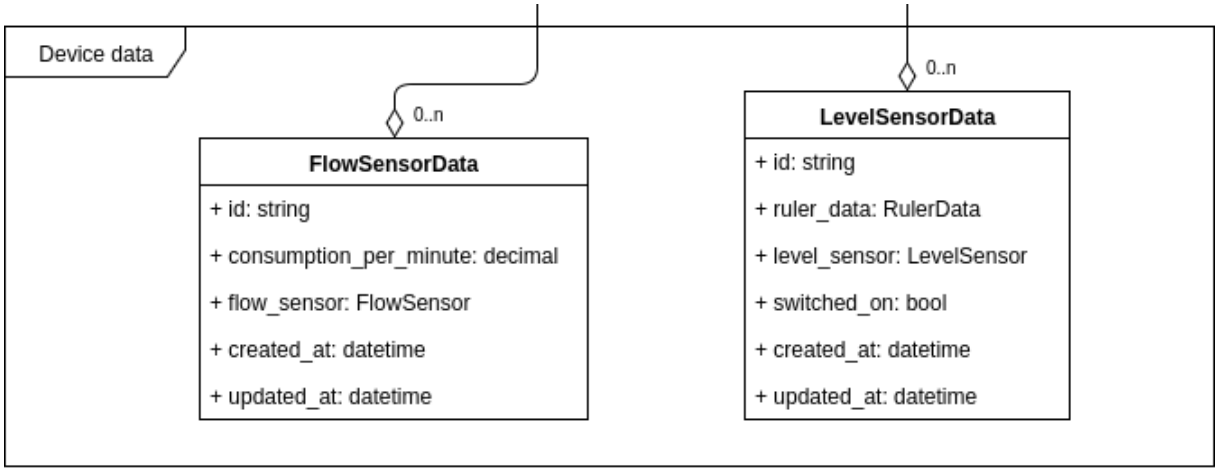
```
{
    "name": "Novo Reservatório",
    "description": "Reservatório localizado no último prédio do condominio",
    "volume": 140.5,
    "customer_id": "c03126c3-fe53-4ab8-91e9-b7b4dc631c0e",
    "reservoir_group_id": "bfff661d-bab0-4e4b-be3a-26b07d75ef0f"
}
```

Fonte: Elaborado pelos autores (2017).

Figura 35 - Diagrama de Classe da API




Reservoir
0..n
0..n
Reservoir
+ id: string
+ name: string
+ description: string
+ volume: float
+ group: ReservoirGroup
+ customer: Customer
+ created_at: datetime
+ updated_at: datetime
ReservoirGroup
+ id: string
+ name: string
+ created_at: datetime
+ updated_at: datetime
1
0..n
1
Device
0..n
Device
+ id: string
+ name: string
+ description: string
+ reservoir: Reservoir
+ created_at: datetime
+ updated_at: datetime
Ruler
+ id: string
+ height: int
+ device: Device
+ created_at: datetime
+ updated_at: datetime
1
1..n
1
0..n
LevelSensor
+ id: string
+ pin: string
+ volume: decimal
+ sequence: int
+ ruler: Ruler
+ created_at: datetime
+ updated_at: datetime
1
FlowSensor
+ id: string
+ pin: string
+ device: Device
+ created_at: datetime
+ updated_at: datetime
0..1
1



Fonte: Elaborado pelos autores (2017).

As *Migrations* são uma solução que visam auxiliar na adição de campos na estrutura do banco de dados, com o intuito de fazer um novo recurso funcionar. Sendo assim, as *migrations* "podem descrever as transformações em classes autônomas que podem ser verificadas em sistemas de controle de versão e executadas em outro banco de dados que pode estar com uma, duas ou cinco versões" (RUBY ON RAILS, 2017).

Figura 36 - *Migrations* de reservatórios

```
1  class CreateReservoirs < ActiveRecord::Migration[5.1]
2    def change
3      create_table :reservoirs, id: :uuid do |t|
4        t.string :name
5        t.string :description
6        t.decimal :volume
7        t.timestamps
8      end
9      add_column :reservoirs, :customer_id, :uuid
10     add_column :reservoirs, :reservoir_group_id, :uuid, null: true
11     add_foreign_key :reservoirs, :customers
12     add_foreign_key :reservoirs, :reservoir_groups
13   end
14 end
```

Fonte: Elaborado pelos autores (2017).

4.3.3 Aplicação Web

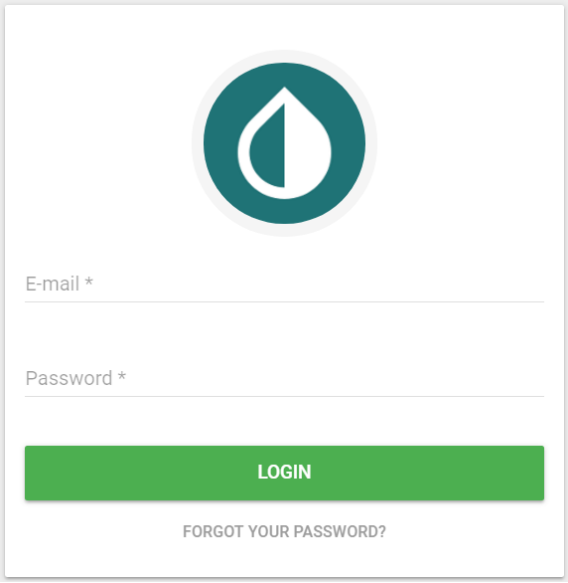
A parte visual do projeto foi desenvolvida através do conceito de SPA (*Single*

Page Application), em que uma única página é renderizada em tempo de execução através do próprio navegador, utilizando *JavaScript*, também conhecido como ECMAScript. O *framework* adotado para construção desta interface foi o Vue.js, que segundo SCHMITZ (2016), reúne os melhores recursos do Angular 2, bem como *React*, que são outras opções de frameworks SPA. A Figura 37 demonstra a interface inicial da aplicação, o formulário de *Login*.

Para auxiliar no desenvolvimento da interface e padronizar os elementos gráficos, foi utilizada a biblioteca Vuetify, que possui design responsivo e permite a visualização das informações em diferentes tamanhos de telas. Para a exibição das informações e relatórios do reservatório foram adicionados gráficos a interface utilizando a biblioteca VueCharts, demonstrado na Figura 38.

Outras tecnologias utilizadas foram: o *Webpack*, que é uma ferramenta para agrupar os recursos utilizados em um projeto JavaScript e transformá-los em um único arquivo; o Babel, que transforma o código de ECMAScript 6 para ECMAScript 5 (versão que a maioria dos navegadores suporta atualmente); e por fim o ESLint que é uma ferramenta que analisa o código e alerta sobre quaisquer problemas que encontrar.

Figura 37 - Interface de Login em Vue.JS



Fonte: Elaborado pelos autores (2017).

Figura 38 - Gráficos da Interface utilizando VueCharts

Fonte: Elaborado pelos autores (2017).

4.4 Qualidade de Software

Para garantir a integridade do projeto, foram empregados testes automatizados na API e na interface. A Qualister (2017) descreveu o seguinte texto sobre automatização de testes:

> O propósito da automação de testes pode ser resumidamente descrito como a aplicação de estratégias e ferramentas tendo em vista a redução do envolvimento humano em atividades manuais repetitivas. A automação possibilita a execução de testes regressivos com maior amplitude e profundidade (QUALISTER apud KANER, 2017).

Na API foram elaborados os testes unitários, que são executados através do serviço de integração contínua a cada nova alteração enviada ao repositório no *GitHub*, que em caso de sucesso, é enviado para o servidor de produção.

Para garantir qualidade ao dispositivo Arduino, foi elaborado o plano de testes disponível no Apêndice C.

4.5 Infraestrutura do Projeto

Em nível de tecnologias, para o projeto ser implantado adequadamente, é necessária uma infraestrutura mínima para a sua correta execução.

No caso do dispositivo Arduino, para que o mesmo possa enviar os dados através da internet utilizando o módulo Wi-Fi, é necessária a disponibilidade de uma rede sem fio com padrão 802.11 b/g/n, com o protocolo mínimo de segurança *WPA2_PSK*. Para que o dispositivo efetue a autenticação na rede e se comunique com a API, é necessário a instalação de um cartão SD contendo dois arquivos: o *wifi.json.txt*, incluindo os dados para a autenticação com a rede e o *settings.json.txt*, com os dados dos sensores e usuário para autenticação com a API.

Para a API, é necessário um servidor *web* baseado em *Ruby*, para o banco de dados é necessário o PostgreSQL na versão 9.2 ou superior e para a interface há um único requisito em relação ao servidor de aplicação, que deve ser um servidor *web* para arquivos estáticos.

4.6 Pesquisa de Campo

A pesquisa de campo foi elaborada com o intuito de validar os requisitos elicitados, comparando o volume obtido de respostas em cada questão, utilizando o método quantitativo. Além disso, as perguntas nortearam sobre a importância de um controle de água para um condomínio, o principal meio de acesso ao sistema dos usuários e a viabilidade econômica do projeto.

A ferramenta escolhida foi a *Google Forms*, que permite enviar um *link* para acesso ao questionário e procede com a elaboração de uma planilha eletrônica, que computa os resultados apurados. A lista das perguntas empregadas está disponível no Apêndice D.

4.7 Estimativa de Custos

Tendo em vista a viabilidade econômica do negócio, foram levantadas as estimativas de custos para infraestrutura e custos operacionais.

Para a infraestrutura, contemplando os serviços da API e da Interface, foi considerada a utilização dos serviços da *Amazon AWS*, que é uma plataforma de serviços em nuvem, oferecendo poder computacional de forma simples, preço acessível e escalável. Os serviços considerados para esta estimativa foram os seguintes: EC2, para o *web server*; S3, para armazenamento de arquivos; RDS,

para banco de dados relacionais; *Cloud Front*, para distribuição de conteúdo; SES, para envio de e-mails transacionais. Para a estrutura inicial foram definidos recursos mínimos de funcionamento, no qual o valor estimado é, em média, R$200,00 mensais.

O valor definido por reservatório seria calculado utilizando os seguintes itens: R$60,00 do Arduino; R$50,00 dos sensores de nível (considerando 5 níveis de monitoramento); R$35,00 do material para montagem da régua dos sensores de nível; R$30,00 do sensor de vazão. Deveriam ser consideradas também a mão de obra de montagem e configuração do equipamento Arduino, e da configuração da API para o cliente. Esses valores poderiam ser menores no caso da compra através de atacado ou com consulta prévia de fornecedores.

Em relação aos custos operacionais poderiam ser considerados os custos de uma equipe como desenvolvedores e suporte, tendo em vista a prestação de serviço mensal em eventuais manutenções na aplicação e atendimento aos clientes.

Portanto, o custo total estimado por reservatório seria de R$ 175,00 para aquisição dos equipamentos e de R$200,00 mensais para a infraestrutura de *software*.

4.8 Resultados

Observa-se que o dispositivo Arduino com acesso à internet pode ser utilizado como exemplo prático do conceito de Internet das Coisas. Utilizando este conceito, é possível adicionar recursos pré-definidos ao dispositivo tornando possível que o protótipo, efetuasse a leitura de sensores de nível e vazão de água em um reservatório e transmitisse os dados para a API.

Durante o desenvolvimento do *firmware* do dispositivo foram encontrados diversos obstáculos, sendo o primeiro e principal o funcionamento e a forma de utilização das bibliotecas disponibilizadas pela IDE do Arduino e de terceiros. Estas bibliotecas facilitaram a utilização de periféricos, como no caso do módulo de acesso à rede sem fio que utiliza a biblioteca ESP8266, do display OLED que utiliza a biblioteca *U8glib* e do módulo de cartão SD que utiliza a biblioteca SD. Para manipular o JSON nas requisições e respostas HTTP da API, foi necessário a

utilização de uma biblioteca denominada *ArduinoJson*, que permitiu montar o corpo das requisições e manipular do retorno no corpo das respostas.

Outro ponto identificado no processo é o tamanho de alocação do *firmware* dentro do dispositivo, que não pode exceder o tamanho máximo de 256 kB, sendo necessário que a implementação do *software* fosse estruturada em funções, diminuindo a utilização de variáveis globais e passando a utilizar variáveis locais.

O dispositivo foi instalado em uma caixa plástica com aberturas e uma tampa de acrílico, possibilitando a visualização do *display* e facilitando acesso para eventuais alterações no cartão SD.

A API exerceu um papel fundamental durante a elaboração do projeto, já que não foi necessária preocupação adicional na elaboração de outras formas de comunicação, sendo que foi um recurso para a persistência dos dados. Neste caso, a API foi responsável pela autenticação dos usuários e armazenamento dos dados. Desta forma, atendemos o objetivo de desenvolver o modelo da plataforma para recepção e disponibilização da informação enviada através do equipamento de medição.

A experiência em desenvolver uma API com o *Ruby on Rails* foi extremamente produtiva e gratificante, confirmando as características fundamentadas sobre a ferramenta, em que se permite escrever pouco código, refletindo em alta produtividade e solucionando diversos problemas em pouquíssimo tempo, principalmente quando se tem domínio da linguagem. A variedade de bibliotecas do *Ruby*, também conhecidas como *gems*, facilitaram o processo de desenvolvimento, permitindo oferecer diversas funcionalidades sem a necessidade de preocupar com o desenvolvimento de rotinas para autenticação e autorização, por exemplo.

Os testes implementados na API foram fundamentais para garantir que os requisitos funcionais estabelecidos fossem cumpridos, tendo papel de extrema importância. Além disso, facilitou o processo de ajustes durante o desenvolvimento, garantindo que as alterações realizadas continuassem de acordo com os testes.

Tendo em vista facilitar a configuração dos dispositivos Arduino, foi criada uma rota na API para gerar a configuração necessária a ser armazenada, evitando o trabalho manual de configuração dos sensores.

A interface gráfica permitiu a visualização dos dados coletados pelo dispositivo Arduino e disponibilizados através da API. Entende-se que isto atinge o objetivo de desenvolver a interface gráfica para visualização da informação contida na plataforma de armazenamento.

A escolha do *Vue.js* para desenvolvimento da interface foi de suma importância, pois facilitou a integração com a API, além também de facilitar o processo de desenvolvimento. A biblioteca *Vuetify* permitiu que a interface fosse padronizada sendo possível disponibilizar também uma versão responsiva, facilitando o acesso em diferentes tamanhos de telas.

A implementação do processo de Integração Contínua para a API e a Interface, utilizando a plataforma do *CodeShip*, auxiliou no desenvolvimento ágil, garantindo que todos os testes fossem executados durante o processo de desenvolvimento, além também de automatizar o processo de envio dos arquivos para o servidor de produção.

Por fim, a realização da pesquisa tinha por finalidade a validação dos requisitos elicitados durante a fase de desenvolvimento do projeto, bem como a viabilidade econômica e possíveis adições de recursos ao projeto.

O resultado da pesquisa permite-nos a elaboração de alguns indicadores quanto aos principais itens: utilização, recursos, forma de acesso e valor de investimento mensal, que permeia viabilidade econômica.

A pesquisa foi elaborada com 35 pessoas da região de Blumenau, sendo encaminhado o link através de e-mail em que consta o formulário da pesquisa. Nos resultados, que estão disponíveis no Apêndice E, pôde-se concluir que 88,6% das pessoas, considerando as respostas Sim, acreditam que um controle eletrônico nos reservatórios de água seria importante. Desta forma, a construção de uma aplicação comercial teria seu verdadeiro valor econômico.

Quanto aos recursos iniciais implementados, eles atendem plenamente os desejos dos usuários, confirmando o que Cassiolato e Orellana (2017) apontam que vazão e nível estão entre os principais controles de grandeza de medição industriais, onde em prédios residenciais também teriam seu verdadeiro valor. Aqui os dados apontaram que 66% das pessoas que responderam a pesquisa, afirmaram que vazão e nível são essenciais para se analisar e controlar.

Sob esta ótica, os dados estatísticos dão subsídios suficientes para concluir que a implementação de outros sensores de leitura seria um diferencial comercial do protótipo, destacando a qualidade da água e a falta de abastecimento de água na rede.

Continuando com a análise dos resultados obtidos na pesquisa, com a intenção de validar a forma de acesso à plataforma, 88,6% das pessoas indicaram que o principal meio de acesso seria através de dispositivos *mobile*. Desta forma, a construção de um *design* responsivo se torna obrigatória, validando desta forma os requisitos não funcionais descritos no projeto.

Um ponto que deveria ser avaliado é sobre a forma de comercialização, visto que as pessoas estariam dispostas a investir um valor de até R$ 50,00/mês. Para isto, a estratégia comercial deveria ser analisada, visto que possui uma infraestrutura mínima requerida para o funcionamento satisfatório da aplicação.

Além disso, a pesquisa se mostra uma alternativa de explorar ideias futuras de novas funcionalidades. Sendo assim, pode-se perceber que o objetivo de realizar uma pesquisa com condomínios para avaliar a viabilidade do projeto foi atendido.

5 CONSIDERAÇÕES FINAIS

A água é um recurso finito, não renovável e essencial para a existência dos seres vivos e da sociedade. Por este motivo, a preservação e uso sustentável se fazem necessário, para que não haja o seu uso excessivo, garantindo que todos tenham acesso à água de boa qualidade e em quantidade que atendam as necessárias diárias.

O aumento dos grandes centros urbanos é uma realidade atual. Conforme dados apresentados, pode-se observar que a construção de moradias em condomínios garante que em um espaço possam viver mais pessoas, comparado a uma casa. Isto traz consigo o crescimento populacional e problemas urbanos tais como: o uso indisciplinado da água, o aumento no conglomerado de pessoas gerando a destruição de mananciais, a impermeabilização do solo, a retirada da mata ciliar, o consumo excessivo e a poluição da água.

Assim sendo, ações e atitudes sobre o uso da água devem ser tomadas. Os principais órgãos de defesa mundiais já iniciaram ações para mobilização. Os governantes do nosso país e dos estados, também já iniciaram atitudes para conscientização e seu uso sustentável. E isto se faz necessário para que não venham a acontecer problemas sérios, como o caso da crise hídrica no estado de São Paulo, entre os anos 2014 a 2016, que deixou milhares de famílias sem acesso à água.

Neste quesito, a tecnologia da informação entra como apoio a outras áreas. Na crise hídrica vivida em São Paulo, através da sua utilização de conceitos de Internet das Coisas (IoT), pode-se mapear as áreas críticas, passando a controlar os níveis dos mananciais e trazer a disponibilização da água novamente às famílias atingidas.

O conceito IoT é a comunicação de qualquer sistema embarcado com a internet para a geração de dados e informações. Observou-se durante as pesquisas que é uma área de estudo e desenvolvimento a ser explorada em trabalhos de cunho científico.

A popularização da IoT se deu no surgimento de sistemas embarcados de baixo custo, como exemplo o Arduino, NodeMCU, entre outros. Sistemas embarcados são dispositivos eletrônicos, programáveis e customizáveis para

exercerem uma função específica com a adição de sensores ou atuadores.

Com a finalidade de contribuir com a sociedade, com o emprego da TI, utilizando o conceito de IoT, foi realizado o projeto para controle de nível e vazão de água em condomínios residenciais.

O projeto teve a proposta de armazenar registros das aferições, sendo que o dispositivo Arduino está instalado em um reservatório, gerando registros de nível e vazão, onde posteriormente, outras ações sejam acrescentadas, permitindo assim, propostas para trabalhos futuros.

Para isto, foi necessária a elaboração de uma API, seguindo o padrão de comunicação REST, com o objetivo de reduzir os custos finais do projeto. Adicionar ainda módulo de Wi-Fi ao Arduino, para que fosse possível a realização do conceito de IoT. Além disso, o resultado da pesquisa aplicada demonstrou que o projeto tem potencial e viabilidade prática, podendo desta forma gerar uma aplicação que atendam às necessidades do mercado, bem como trazer uma melhoria para a sociedade.

Interessante destacar que a visualização dos dados exerceu uma tarefa importante na usabilidade do usuário do protótipo. Isto possibilitou elaborar gráficos que demonstram na prática a situação atual dos reservatórios, o que permite a tomada de decisão de forma ágil.

É importante frisar que este trabalho não se limita ao escopo aqui demonstrado, podendo dar subsídio a futuros trabalhos acadêmicos de diversas áreas. Algumas ideias identificadas foram: adicionar ao dispositivo Arduino o sensor para medição de qualidade da água, adicionar o recurso para criação de alertas conforme definido pelo usuário, cálculo de desvio padrão, cálculo da média de consumo de água para exibição nos gráficos, elaborar o custo do projeto final e considerar o consumo individual das pessoas.

Assim, destaca-se que os objetivos propostos neste trabalho foram atendidos. Considerando as fases do desenvolvimento do protótipo (a), a elaboração da API (b), a visualização dos dados na interface gráfica (c) e aplicação da pesquisa de campo (d), o estudo apontou claramente que propostas para o uso sustentável da água devem ser tomadas e estudadas para que a sociedade possa usufruir deste recurso finito e essencial (e).

REFERÊNCIAS

AGNEZ, Luciane F. Consumo da informação na sociedade contemporânea. **Intercom – Sociedade Brasileira de Estudos Interdisciplinares da Comunicação**. Natal, 2009. Disponível em: <http://www.intercom .org.br/premios/2009/ LucianeAgnez.pdf>. Acesso em: 01 mar. 2017.

ARAÚJO, Joao G. R. O Desenvolvimento de Aplicações Web. **RNP - Rede Nacional de Ensino e Pesquisa**. 1997, v. 1, n. 5. Disponível em: <https://memo ria.rnp.br/newsgen/9710/n5-3.html>. Acesso em: 01 mar. 2017.

ARDUINO. **Introduction**. Disponível em: <https://www.arduino.cc/en/Guide/Introduc-tion>. Acesso em: 28 fev. 2016.

_____. **Technical specs**. Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardMega2560#techspecs>. Acesso em: 29 mai. 2017.

BEEKMAN, Gertjan B. Gerenciamento integrado dos recursos hídricos. **Instituto Interamericano de Cooperação para a Agricultura - IICA**. Brasília, 2017, p. 7, 12.

BORGES, Lucilia Pereira; DORES, Rodrigo de Carvalho. **Automação predial sem fio utilizando Bacnet/ZigBee com foco em economia de energia**. 2010. TCC (Graduação) - Universidade de Brasília, Brasília, 2010, p. 88. Disponível em: <http://www.ene.unb.br/adolfo/Monographs/Graduation/TG10%20Luc%C3%ADlia%2 0P.%20Borges%20e%20Rodrigo%20C.%20Dores.pdf>. Acesso em: 29 de abr. 2017.

BRASIL. **Segundo Unesco, mundo precisará mudar consumo de água**. Meio Ambiente. 2015. Disponível em: <http://www.brasil.gov.br/meio-ambiente /2015/03/segundo-unesco-mundo-precisara-mudar-consumo-de-agua>. Acesso em: 22 fev. 2017.

BRUGNARI, ARTHUR; MAESTRELLI, LUIZ HENRIQUE MUSSI. **Automação Residencial via WEB**. 2010, 36f. Trabalho de conclusão de curso - Curso de Graduação em Engenharia de Computação - PUC-PR, Curitiba, 2010. Disponível em: <http://www.ppgia.pucpr.br/~laplima/ensino/pfec/concluidos/2010/autores.pdf>. Acesso em: 29 de abr. 2017.

CAELUM. **Web ágil com VRaptor, Hibernate e AJAX**. Disponível em: <https://www.caelum.com.br/apostila-vraptor-hibernate/rest/#11-3-o-triangulo-do-rest>. Acesso em: 22 fev. 2017

CANALTECH. **O que é API?** Disponível em: <https://canaltech.com.br/o-que-e/ software/o-que-e-api/> Acesso em: 16 fev. 2017.

CASSIOLATO, César; ORELLANA, Evaristo. **Medição de Vazão**. Disponível em: <http://www.smar.com/newsletter/marketing/index40.html>. Acesso em: 18 abr. 2017.

CEDRO TECHNOLOGIES. **Internet das Coisas:** O que você precisa saber. Disponível em: <http://blog.cedrotech.com/internet-das-coisas-o-que-voce-precisa-saber/>. Acesso em: 08 mai. 2017.

CONDOR. **Consumo de Água**. 2016. Disponível em: <http://www.condorlrscondominios.com.br/consumodeagua>. Acesso em: 26 mar. 2017.

DELAI, Andre Luiz. Sistemas Embarcados: A Computação Invisível. **Hardware**. 2013. Disponível em: <http://www.hardware.com.br/artigos/sistemas-embarcados-computacao-invisivel/conceito.html>. Acesso em: 28 fev. 2017.

EMBARCADOS. **Monitoramento de água com IoT - Parte 1**. 2015. Disponível em: <https://www.embarcados.com.br/monitoramento-de-agua-com-iot-parte-1/>. Acesso em: 28 mai. 2017.

FERNANDES, Sérgio Luis Peixoto. **Maturidade da Integração Contínua**. 2015. Disponível em: <http://blog.octo.com/pt-br/maturidade-da-integracao-continua/>. Acesso em: 25 mai. 2017.

FERREIRA, Rodrigo. **Protegendo sua API via Shared Key Authentication**. 2015. Disponível em: <http://blog.caelum.com.br/protegendo-sua-api-rest-via-shared-key - authentication/>. Acesso em: 23 mai. 2017.

FOWLER, Martin. **Continuous Integration**. 2006. Disponível em: <https://martinfowler.com/articles/continuousIntegration.html>. Acesso em: 25 mai. 2017.

GITHUB. **HTTP Specs**. 2015. Disponível em: <https://http2.github.io/http2-spec/>. Acesso em: 01 mar. 2017.

HARTL, Michael. **Ruby on Rails Tutorial:** Learn Web Development with Rails. Cover Rails 5. 2016. Disponível em: <https://www.railstutorial.org/book>. Acesso em: 23 mai. 2017.

IABSP. **O que é Arquitetura**. Disponível em: <http://www.iabsp.org.br/ oqueearquitetura.asp>. Acesso em: 01 mar. 2017.

IBGE - Instituto Brasileiro de Geografia e Estatísticas. Projeção da população no Brasil e das Unidades Federativas. Disponível em: <http://www.ibge .gov.br/apps/populacao/projecao/>. Acesso em: 06 abr. 2017.

IRVING, Marta de Azevedo; OLIVEIRA, Elizabeth. **Sustentabilidade e Transformação Social**. Rio de Janeiro: Senac Nacional, 2012.

JSCRAMBLER. **API para Autenticar usuários com JWT e Passport**. 2016. Disponível em: <https://tableless.com.br/autenticar-usuarios-com-jwt-e-passport/>. Acesso em: 23 mai. 2017.

JSON. **Introdução ao JSON**. 1999. Disponível em: <http://www.json.org/json-pt.html>. Acesso em: 24 mai. 2017.

LEIS ESTADUAIS. **DECRETO Nº 852, de 1º de Setembro de 2016**. Santa Catarina. Disponível em: <http://leisestaduais.com.br/sc/decreto-n-852-2016-santa-catarina-regulamenta-a-lei-n-16699-de-2015-que-institui-a-semana-estadual-do-uso-consciente-da-agua-no-estado-de-santa-catarina>. Acesso em: 22 fev. 2017.

MATSUKI, Edgard; GOMES, Gustavo. Conheça Soluções para falta de água em diversos países. **Portal EBC**. 2015. Disponível em: <http://www.ebc.com.br/noticias/meio-ambiente/2015/03/falta-de-agua-lista-de-10-solucoes>. Acesso em: 08 mar. 2017.

MEOLA, Andrew. O que é a Internet das Coisas (IoT)? **Business Insider**. 2016. Disponível em: <http://www.businessinsider.com/what-is-the-internet-of-things - definition-2016-8>. Acesso em: 28 fev. 2017.

MICHAELIS. **Internet**. 2017. Disponível em: <http://michaelis.uol.com.br/busca?r=0&f=0&t=0&palavra=internet>. Acesso em: 11 mai. 2017.

MMA - Ministério do Meio Ambiente. **Água um Recurso cada vez mais Ameaçado**. 2006. Disponível em: <http://www.mma.gov.br/estruturas/sedr_proecotur/_publicacao/140_publicacao09062009025910.pdf>. Acesso em: 06 abr. 2017.

MOLINARI, Leonardo. **Testes de Software:** Produzindo sistemas melhores e mais confiáveis. São Paulo: Érica, 2013.

OLIVEIRA, Déborah. Inteligência que vem da Água. **Revista IT Forum**. Dez/2016, p. 84-85.

OLIVEIRA JR, Eustácio Rangel de. **Conhecendo Ruby**. 2016. Disponível em: <https://leanpub.com/conhecendo-ruby/read>. Acesso em: 23 mai. 2017.

PLANALTO. **Lei Nº 11.124, de 16 de junho de 2015**. Disponível em: <http://www.planalto.gov.br/ccivil_03/_ato2004-2006/2005/lei/l11124.htm>. Acesso em: 06 abr. 2017.

PLANSKY, Ricardo. Definição, Restrições e Benefícios do Modelo de Arquitetura REST. **iMasters**. 2014. Disponível em: <http://imasters.com.br/desenvolvimento/definicao-restricoes-e-beneficios-modelo-de-arquitetura-rest/> Acesso em: 20 fev. 2017.

QUALISTER. **Introdução à Automatização de Testes**. Disponível em: <http://www.qualister.com.br/blog/introducao-a-automacao-de-testes>. Acesso em: 26 mai. 2017.

RAILS GUIDE. Getting Started with Rails. 2017. Disponível em: <http://guides.rubyonrails.org/getting_started.html>. Acesso em: 23 mai. 2017.

RIBEIRO, Marcus. O que é API? **Pluga**. Disponível em: <https://pluga.co/blog/api/o-que -e-api/> Acesso em: 16 fev. 2017.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software**. 3. ed. Rio de Janeiro: Alta Books, 2013.

RUBY ON RAILS. **Migrations**. 2017. Disponível em: <http://api.rubyonrails.org/classes/ActiveRecord/Migration.html>. Acesso em: 25 mai. 2017.

SABESP. **Dicas de Economia**. Disponível em: <http://site.sabesp.com.br/site/interna/Default.aspx?secaoId=140>. Acesso em: 24 mar. 2017.

SCHMITZ, Daniel. Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar. **Tableless**, 2015. Disponível em: <https://tableless.com.br/tudo-que-voce-queria-saber-sobre-git-e-github-mas-tinha-vergonha-de-perguntar/>. Acesso em: 29 mai. 2017.

_____. Conheça o Vue.js, um framework javascript para criação de componentes web reativos. **Tableless**. 2016. Disponível em: <https://tableless.com.br/conheca-o-vue-js-um-framework-javascript-para-criacao-de-componentes-web-reativos/>. Acesso em: 14 mai. 2017.

SEBRAE. **Inovação e Tecnologia:** Internet das Coisas. 2016. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/artigos/inova-cao-e-tecnologia-internet-das-coisas,05d99e665b182410VgnVCM100000b272010aRCRD>. Acesso em: 28 fev. 2017.

SOMMERVILLE, Ian. **Engenharia de Software**. 8 ed. Rio de Janeiro: Prentice-Hall, 2008.

SOUZA, Fábio. Arduino - Primeiros Passos. **Embarcados**, 2013. Disponível em: <https://www.embarcados.com.br/Arduino/>. Acesso em: 01 mar. 2017.

TERA. **A importância da gestão integrada das águas com a crise hídrica**. 2015. Disponível em: <http://www.teraambiental.com.br/blog-da-tera-ambiental/a-importancia-da-gestao-integrada-das-aguas-com-a-crise-hidrica>. Acesso em: 09 mar. 2017.

UCHINAKA, Fabiana. Em uma década, número de moradias aumenta mais que o dobro que o crescimento populacional. **UOL Notícias**. 2011. Disponível em: <https://noticias.uol.com.br/cotidiano/ultimas-noticias/2011/04/29/em-uma-decada-

numero-de-moradias-aumenta-mais-que-o-dobro-que-o-crescimento-da-populacao.htm>. Acesso em: 06 abr. 2017.

WHO - World Health Organization. **Minimum water quantity needed for domestic uses**. 2005. Disponivel em: <http://ec.europa.eu/echo/files/evaluation/watsan2005/annex_files/WHO/WHO5%20-%20Minimum%20water%20quantity%20needed%20for%20domestic%20use.pdf>. Acesso em: 06 abr. 2017.

ZEMEL, Tárcio. **JSON: JavaScript Object Notation**. DPW. 2011. Disponível em: <http://desenvolvimentoparaweb.com/javascript/json-javascript-object-notation/>. Acesso em: 24 mai. 2017.

APÊNDICE A - Estórias de Usuário

Desejo 1

Estória 01: Como um administrador do sistema, eu quero controlar o acesso dos usuários para que seja possível realizar o gerenciamento de acesso de acordo com cada nível de acesso proposto.

Estória 02: Como um administrador do sistema, eu quero permitir a manutenção dos usuários para que seja possível a completa administração do sistema.

Estória 03: Como um administrador do sistema, eu quero realizar o cadastro de clientes para que seja possível a utilização do sistema.

Estória 04: Como um administrador do sistema, eu quero permitir acesso apenas a clientes ativos para que seja possível validar o status para consumo dos dados.

Estória 05: Como um administrador do sistema, eu quero que apenas usuários com privilégios exclusivos realizem o cadastro de usuários adicionais para que seja possível controlar os níveis de acesso dos usuários.

Estória 06: Como um administrador do sistema, eu quero que meus clientes ativem sua conta para que seja possível confirmar o endereço de e-mail.

Desejo 2

Estória 01: Como um administrador do sistema, eu quero realizar a manutenção dos reservatórios para que seja possível realizar o cadastro do equipamento vinculado ao mesmo.

Estória 02: Como um administrador do sistema, eu quero realizar o agrupamento dos reservatórios para que seja possível a melhor visualização por parte dos meus clientes.

Estória 03: Como um administrador do sistema, eu quero realizar a manutenção dos dispositivos para que seja possível a manutenção dos sensores vinculados ao mesmo.

Estória 04: Como um administrador do sistema, eu quero realizar a geração do arquivo de configuração para que seja possível a utilização do dispositivo Arduino no local de sua instalação.

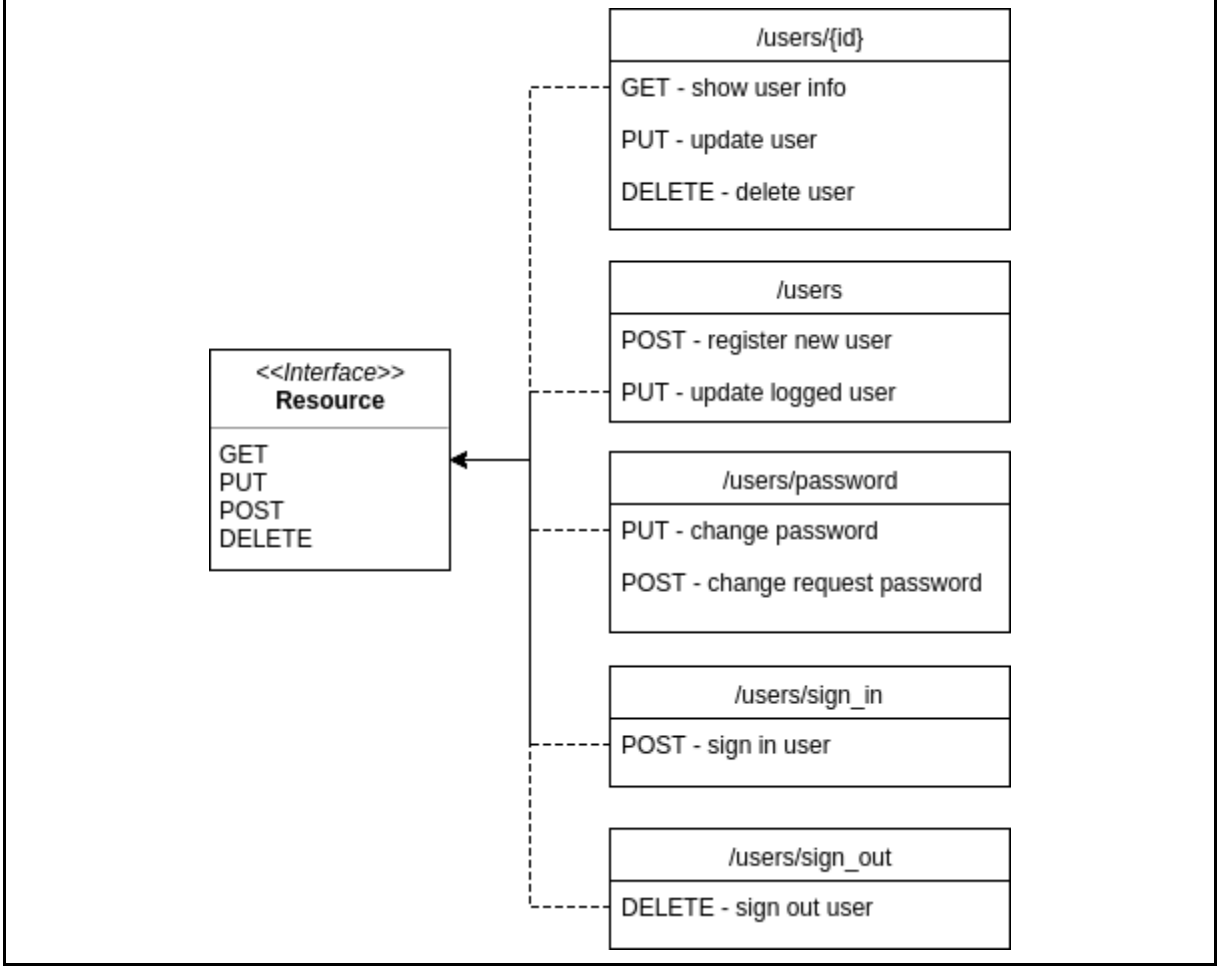
Desejo 3

Estória 01: Como um cliente, eu quero realizar a troca da senha de acesso para que seja possível aumentar a segurança de acesso da minha conta.

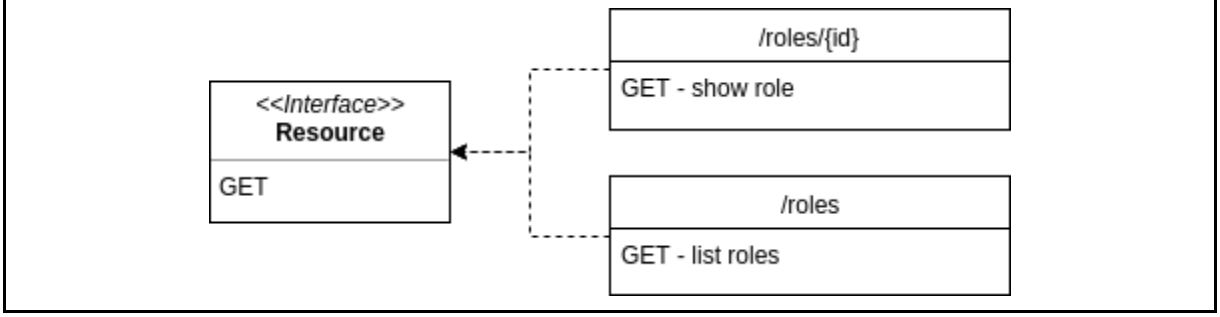
Estória 02: Como um cliente, eu quero visualizar dados apenas minha conta para que seja possível realizar a consulta dos dados enviados pelos meus dispositivos.

APÊNDICE B - Rotas API

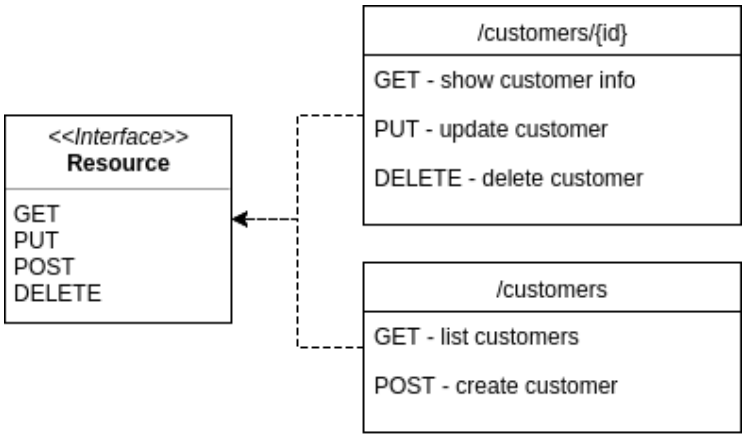
Nome: Autenticação e Usuários



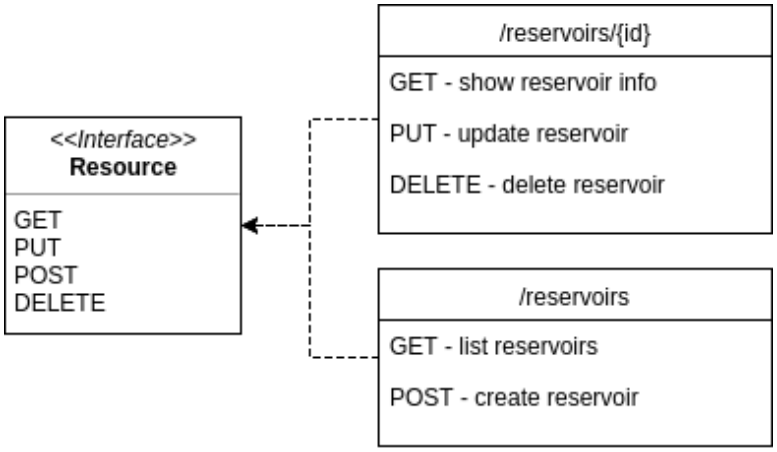
Nome: Papéis de usuário



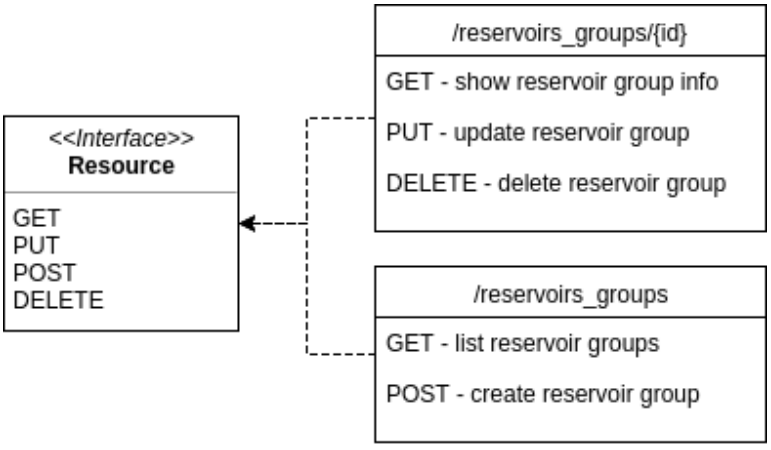
Nome: Clientes



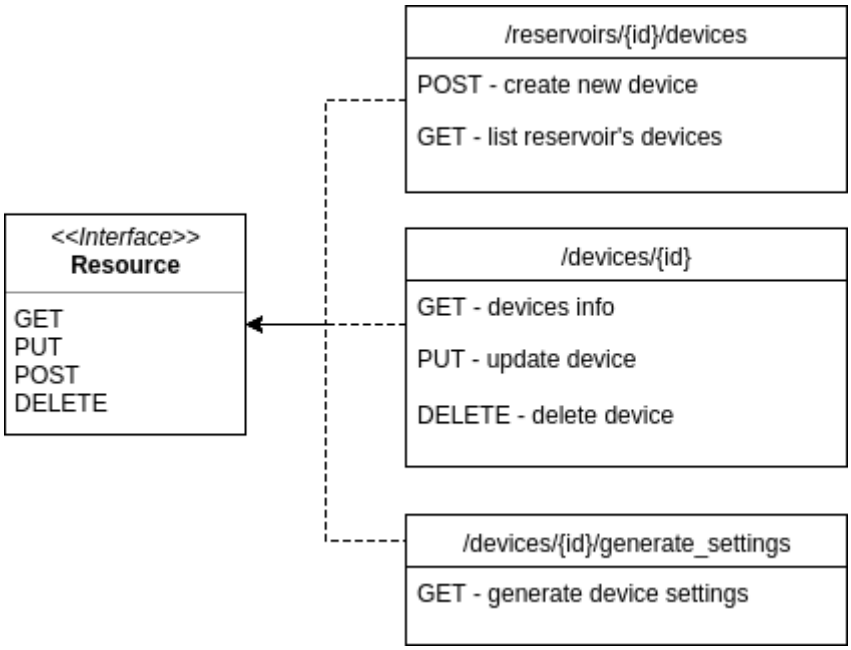
Nome: Reservatórios



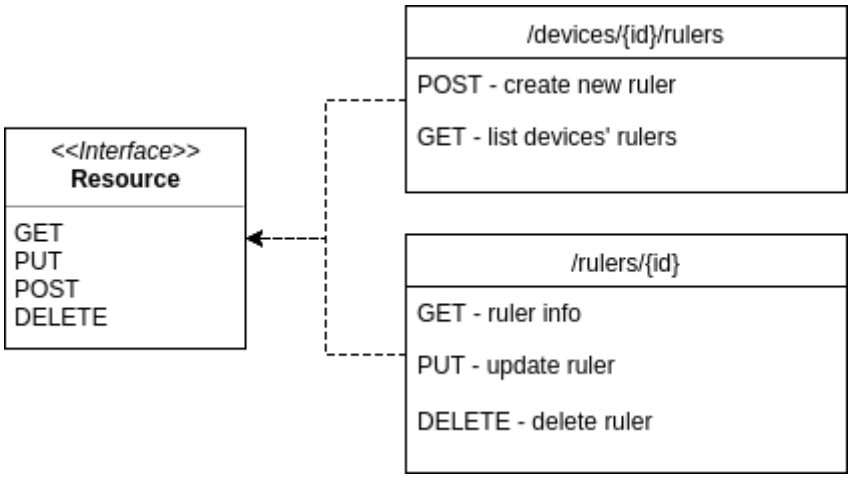
Nome: Grupos de Reservatório



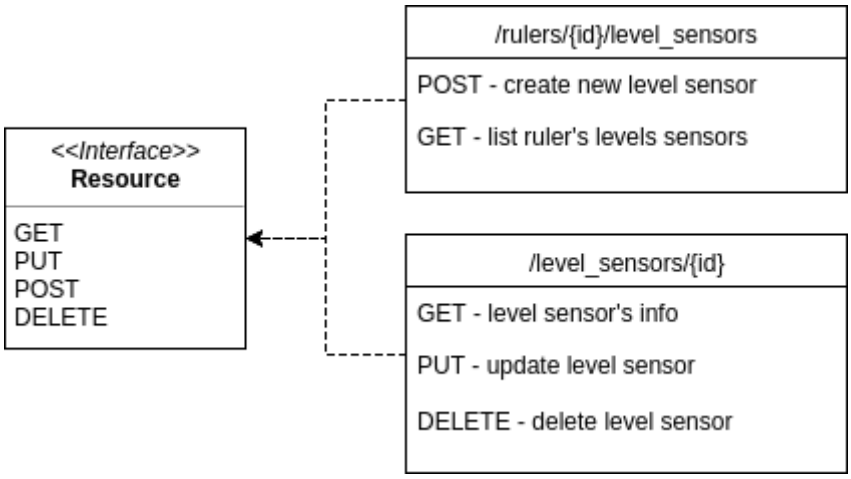
Nome: Dispositivos



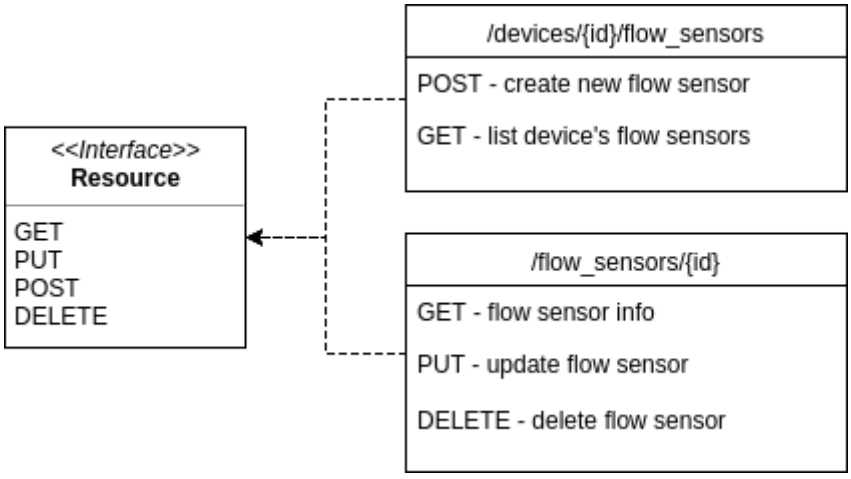
Nome: Réguas de Medição



Nome: Sensores de Nível



Nome: Sensor de Vazão



APÊNDICE C - Plano de Testes para o Dispositivo Arduino

Caso de Teste TCS-1: Alimentação do Equipamento		
Autor:	Diego Weiler / Diogo Ottequir	
Objetivo do Teste: Descreve os passos para ligar o equipamento.		
Pré-condições: Disponibilidade de uma fonte de alimentação na rede convencional.		
#:	**Ações do Passo:**	**Resultados esperados:**
1	Identificar a tensão de saída da tomada da rede convencional.	Encontra uma tomada da rede convencional com tensão de saída 110V ou 220V de corrente alternada.
2	Identificar a voltagem da fonte do equipamento.	Ajustar voltagem da fonte do equipamento com a tensão de saída da rede convencional.
3	Ligar na tomada o equipamento.	O equipamento liga luzes indicativas na cor vermelha.
4	Acionar a chave on/off do equipamento	O equipamento ligar e apresentar em seu display a mensagem “Iniciando...”
Tipo de execução:	Manual	
Tempo estimado (min):	5.00	
Prioridade:	Alta	
Relações:	Bloqueia - TCS-2: Funcionamento do Equipamento.	
Requisitos:	Alimentação Bi-Volt 110/220V	

Caso de Teste TCS-2: Funcionamento do Equipamento	
Autor:	Diego Weiler / Diogo Ottequir
Objetivo do Teste: Descreve os passos para preparar o funcionamento do equipamento	

Pré-condições: Equipamento deverá estar operante. Disponibilidade de internet Wi-Fi. Assinatura ativa no ambiente da API.		
#:	**Ações do Passo:**	**Resultados esperados:**
1	Equipamento fará as verificações dos periféricos.	O display irá apresentar o status dos periféricos.
2	Inserir cartão SD com os arquivos de configurações.	Cartão SD será reconhecido e os dados de configurações serão lidos pelo dispositivo e a rede será conectada.
3	O equipamento efetua a leitura do nível do reservatório.	O display irá apresentar o nível do reservatório.
4	Abrir o registro de vazão do reservatório.	Avaliar se o display indica a vazão do reservatório.
5	Repita o teste 5.	Avaliar o display indicador conforme o nível do reservatório.

Tipo de execução:	Manual
Tempo estimado (min):	10.00
Prioridade:	Média
Relações:	Dependência - TCS-1: Alimentação do Equipamento Bloqueia - TCS-3:Integração entre equipamento e API
Requisitos:	Licença do Equipamento vinculado a uma Assinatura

Caso de Teste TCS-3: Integração entre equipamento e API	
Autor:	Diego Weiler / Diogo Ottequir
Objetivo do Teste: Descreve os passos para comunicação entre o equipamento e a API	
Pré-condições: Equipamento deverá estar operante. Acesso à rede de internet via Wi-Fi. Assinatura ativa no ambiente da API.	

#:	Ações do Passo:	Resultados esperados:
1	Equipamento enviará um pacote para a API, efetuando login.	Equipamento receberá uma resposta do servidor com a chave de autenticação.
2	Equipamento enviará um pacote para a API, enviando os dados do sensor de vazão com o ID informado no cartão SD.	Equipamento receberá uma resposta do servidor com o status da gravação.
3	Desligar a internet.	Internet desligada.
4	Equipamento sem acesso a internet.	Equipamento no display será exibida a informação “Rede Desconectada”.
5	Tentativa de conexão com a internet	Equipamento efetuará 5 tentativas de conexão e no display será exibida a informação “Tentativa 1/5”.
6	Falha na tentativa número 5 de conexão com a internet	Equipamento no display será exibida a informação “Falha na conexão com a internet Reinicie o Equipamento”
7	Ligar a internet	Internet ligada.
8	Resetar o equipamento	Equipamento reiniciado.
9	Efetuar os passos 1 e 2 para conferência.	O equipamento receberá a resposta do servidor.

Tipo de execução:	Manual
Tempo estimado (min):	25.00
Prioridade:	Média
Relações:	Dependência - TCS-2: Funcionamento do Equipamento Bloqueia - TCS-4: Utilização da API
Requisitos:	Integração com API

Caso de Teste TCS-4: Utilização da API	
Autor:	Diego Weiler / Diogo Ottequir
Objetivo do Teste:	

Descreve os passos para acesso e utilização da API		
Pré-condições: Assinatura ativa no ambiente da API. Pacotes enviados pelo equipamento.		
#:	**Ações do Passo:**	**Resultados esperados:**
1	Acessar link de acesso à API	Link acessado.
2	Informar usuário e senha incorreto	API retorna mensagem de exceção de login.
3	Informar usuário e senha correto	API retorna página restrita.
4	Informar no filtro um período	API retorna posição do reservatório no período informado.

Tipo de execução:	Manual
Tempo estimado (min):	10.00
Prioridade:	Média
Relações:	Dependência - TCS-3: Integração entre equipamento e API
Requisitos:	API

APÊNDICE D - Pesquisa de Campo

1. Você acha importante um controle eletrônico dos reservatórios de água de seu condomínio? Opções:
   a. Sim,
   b. Não, ou
   c. Talvez.

2. Quais das opções abaixo, você gostaria que um sistema eletrônico permitisse controlar? Opções:
   a. Nível de água,
   b. Vazão ou Consumo de água,
   c. Temperatura da água,
   d. Qualidade da água (pH),
   e. Falta de abastecimento de água na rede,
   f. Eficiência energética, ou
   g. Outro.

3. Qual(is) a(s) principal(is) forma(s) que você acessaria este sistema?
   a. Computador/notebook,
   b. Smartphone,
   c. Tablet, ou
   d. Outro.

4. Quanto você estaria disposto a investir para obter este sistema eletrônico em seu condomínio?
   a. Até R$ 50,00/mês,
   b. De R$ 50,00 a R$ 100,00/mês, ou
   c. Mais de R$ 100,00/mês.

Link de acesso: https://goo.gl/forms/CkDxMLzOvYwCAarW2

APÊNDICE E - Resultado da Pesquisa de Campo

Computação das Respostas Individuais

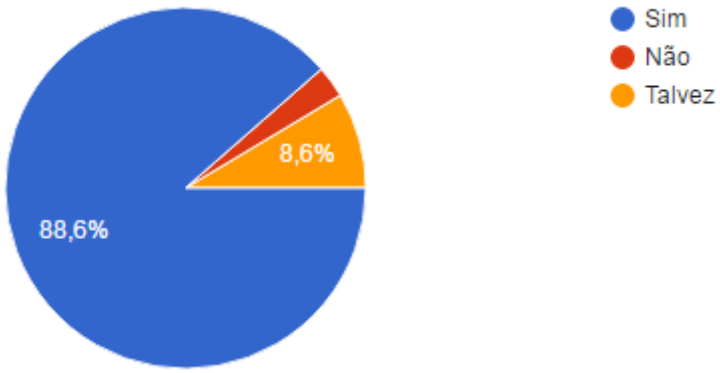
Total de pessoas: 35

Região: Blumenau

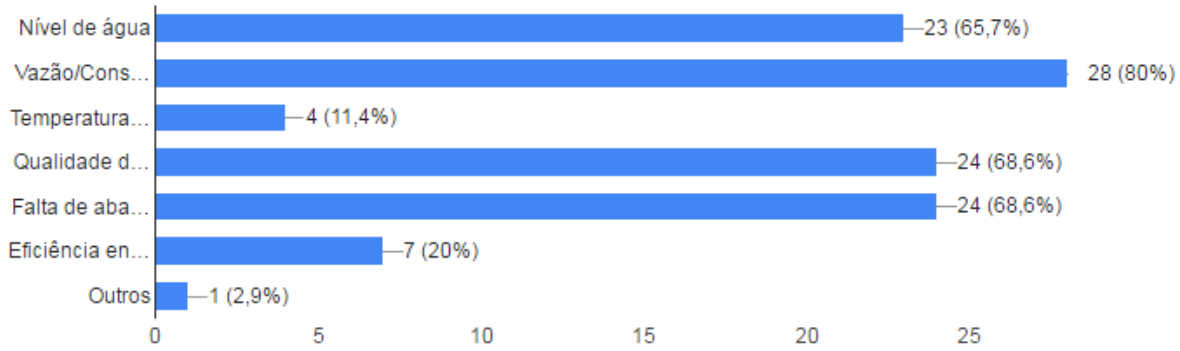
Pessoa	Pergunta 1	Pergunta 2	Pergunta 3	Pergunta 4
1	A	A, B, D, E	A, B	B
2	A	A, B, D, E	A, B	A
3	A	A, B, D, E	A, B	B
4	A	A, B, D, E	A, B, C	A
5	A	A, D, E	A, B	B
6	A	A, B, D, E, G	B	A
7	A	B, D, E	A, B	A
8	C	B	A	A
9	A	B, D, F	B	A
10	C	B	C	A
11	A	A, B, D	B	A
12	C	A, B, D, E, F	A, B	A
13	A	A, B, E	B	A
14	A	A, B, D, E	A, B	A
15	A	D	B	A
16	A	B, D, E	A, B	A
17	A	A, B	B	A
18	A	A, B, C, D, E, F	B	A
19	B	A, B, E	B	A
20	A	D	B	A
21	A	A, B, C, D, E	B	A
22	A	A, B, D, E, F	A, B, C	A

23	A	D	B	B
24	A	A, B, D, E	B	C
25	A	A, E	A	A
26	A	B, D	A, B	A
27	A	A, B, D, E	A, B	A
28	A	A, B, C, D, E, F	A, C	B
29	A	A, B, D, E, F	A, B	A
30	A	A, B, D, E, F	B	C
31	A	B, C	A, B	A
32	A	A, B, E	A, B	B
33	A	B	B	C
34	A	A, E	B	A
35	A	E	B	B

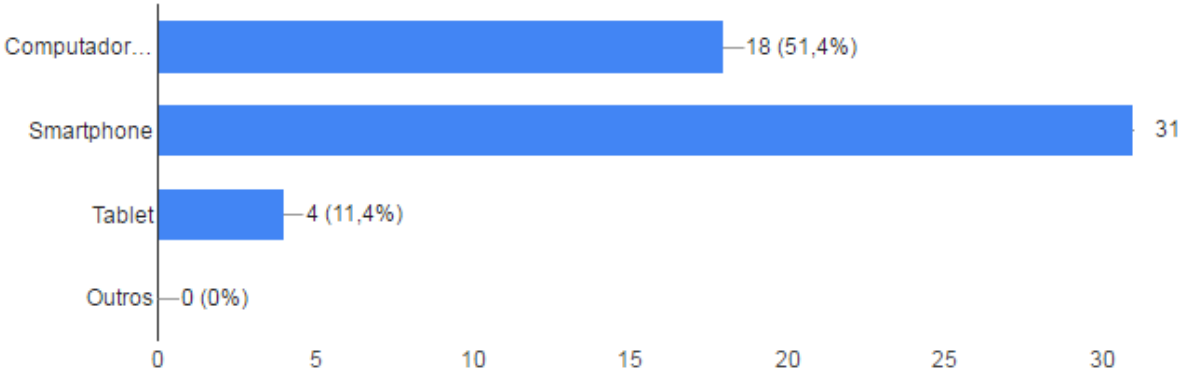
Você acha importante um controle eletrônico dos reservatórios de água de seu condomínio?



Quais das opções abaixo, você gostaria que um sistema eletrônico permitisse controlar?



Qual(is) a(s) principal(is) forma(s) que você acessaria este sistema?



Quanto você estaria disposto a investir para obter este sistema eletrônico em seu condomínio?