**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**

**DEPARTAMENTO ACADÊMICO DE INFORMÁTICA**

**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**LUCAS RAFAEL SZULHA**

**MURILO RAFAEL SZULHA**

# DESENVOLVIMENTO DE UMA ARQUITETURA IoT PARA CONTROLE DE VAZAMENTOS EM TUBULAÇÕES DE ÁGUA

**TRABALHO DE CONCLUSÃO DE CURSO**

**PONTA GROSSA**

**2018**

**LUCAS RAFAEL SZULHA**

**MURILO RAFAEL SZULHA**

# DESENVOLVIMENTO DE UMA ARQUITETURA IoT PARA CONTROLE DE VAZAMENTOS EM TUBULAÇÕES DE ÁGUA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas do Departamento de informática da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Richard Duarte Ribeiro

**PONTA GROSSA**

**2018**

Ministério da Educação
**Universidade Tecnológica Federal do Paraná**
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Tecnologia em Análise e Desenvolvimento de Sistemas

# TERMO DE APROVAÇÃO

DESENVOLVIMENTO DE UMA ARQUITETURA IoT PARA CONTROLE DE VAZAMENTOS EM TUBULAÇÕES DE ÁGUA

por

LUCAS RAFAEL SZULHA
MURILO RAFAEL SZULHA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 7 de novembro de 2018 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof Dr. Richard Duarte Ribeiro
Orientador

_____
Prof MSc. Geraldo Ranthum
Membro titular

_____
Prof Dr. Augusto Foronda
Membro titular

_____
Profa Dra. Helyane Bronoski Borges
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof Dr. André Pinz Borges
Coordenador do curso

- O Termo de aprovação assinado encontra-se na Coordenação do Curso -

## AGRADECIMENTOS

Em primeiro lugar, não poderíamos deixar de agradecer a Deus pelo Dom da vida e sabedoria que nos foi concedido. Obviamente nestes poucos parágrafos é impossível dedicar a todos que de alguma forma ou outra estiveram envolvidos nessa etapa importante de nossas vidas. Portanto aqui vai nosso pedido de desculpas àqueles que não foram citados aqui, mas estejam certos de que estão em nossos pensamentos e fazem parte de nossa gratidão.

Agradecemos ao Professor Dr. Richard Duarte Ribeiro, pelo seu esforço em nos motivar e pela paciência dispensada para que o trabalho fosse realizado com sucesso.

Agradeço ao Professor MSc. Luís Alexandre Rauch, por partilhar conhecimentos que foram de grande valia para o nosso projeto.

Aos nossos Pais, esposa, familiares e amigos, por compreenderem nossas ausências que se fizeram necessárias, que nos apoiaram e motivaram.

## RESUMO

SZULHA, Lucas Rafael; SZULHA, Murilo Rafael. **Desenvolvimento de uma arquitetura IoT para controle de vazamentos em tubulações de água**. 2018. 40 f. Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018.

O desperdício de água no Brasil e no Mundo atingem números impressionantes, visto que muito poderia ser evitado com a redução do uso excessivo, manutenção periódica de tubulações. Este trabalho propõe uma arquitetura de Internet das coisas (do inglês *Internet of Things* (IoT)) que conecta sensores externos, como os de umidade e vazamento, a aplicações para dispositivos móveis. Foi realizado um estudo de caso utilizando uma aplicação implementada para dispositivos móveis e sensores de umidade, visando facilitar o monitoramento de vazamentos em tubulações de água.

Palavras-chave: Aplicativo. Água. Monitoramento. IoT. Arquitetura.

## ABSTRACT

SZULHA, Lucas Rafael; Szulha, Murilo Rafael. **Development of an IoT architecture for the control of leaks in water pipes**. 2018. 40 p. Course Conclusion Work (Technology in System Development and Analysis) - Federal University of Technology - Paraná, Ponta Grossa, 2018.

The waste of water in Brazil and in the World reaches impressive numbers, since much could be avoided with the reduction of the excessive use, periodic maintenance of pipes. This paper proposes an Internet of Things (IoT) architecture that connects external sensors, such as moisture and leakage, to mobile applications. A case study was carried out using an application implemented for mobile devices and humidity sensors, in order to facilitate the monitoring of leaks in water pipes.



Keywords: Application. Water. Monitoring. IoT. Architecture.

## LISTA DE ABREVIATURAS

| | |
|---|---|
| IoT | *Internet of Things* |
| RFID | *Radio Frequency Identification* |
| IBSG | *Internet Business Solutions Group* |
| HTML | *Hypertext Markup Language* |
| URL | *Uniform Resource Locator* |
| WSDL | *Web Services Description Language* |
| XML | *Extensible Markup Language* |
| HTTP | *HyperText Transfer Protocol* |
| PHP | *Personal Home Page* |
| SGBD | *Sistema de Gerenciamento de Bando de Dados* |
| IDE | *Integrated Development Environment* |

**SUMÁRIO**

# 1 INTRODUÇÃO

A poluição e o uso desregrado dos recursos hídricos estão, aos poucos, tornando a água imprópria para o consumo humano (BARROS, 2008). Além disso, existe o fato de que a água potável é desperdiçada tanto por problemas que impedem a água, por meio hidráulico, de ser utilizada com uma finalidade, ou então pelo simples fato de ser desperdiçada (DE OLIVEIRA, 2002). Segundo estudos do Instituto Trata Brasil[1] (ITB), em 2016, o consumo médio de água por habitante no Brasil era de aproximadamente 154 litros por dia, quando o estimado é de que 112,5 litros por dia são suficientes para suprir todas as necessidades diárias. O grande desperdício ocorre durante o processo de distribuição, que na média Nacional chega a 38% devido a estruturas envelhecidas pelo tempo, instalações precárias e desvios de água clandestino.

Segundo Ashton (2009) cada vez mais a procura por ferramentas que sejam capazes de executar tarefas no lugar de uma pessoa, tendo como uma de suas justificativas a falta de tempo hábil. A Internet das coisas sugere que a Internet seja estendida para objetos do nosso cotidiano, de modo a automatizar processor e ações realizadas antes apenas pela mão humana, por exemplo em cafeteiras, torradeiras, geladeiras, etc.

Unindo a preocupação com o desperdício de água e o estudo sobre a *Internet* das Coisas (do inglês *Internet of Things* (IoT)), este trabalho propõe uma arquitetura IoT para facilitar a aplicação do conceito de camadas, conectando sensores de captação externa como os de umidade, fumaça, temperatura ou presença, a uma interface gráfica implementada em um dispositivo móvel, e então aplicando os conhecimentos obtidos em um estudo de caso.

## 1.1 MOTIVAÇÃO E ESCOPO

A motivação do trabalho surgiu da necessidade de criar uma estrutura capaz de auxiliar na captura de sinais de vazamento de água em companhias de

[1] http://www.tratabrasil.org.br/institucional/quem-somos

saneamento e distribuição de água, como a SANEPAR[2] (Companhia de Saneamento do Paraná).

## 1.2 DESCRIÇÃO DO PROBLEMA

Elaborar uma arquitetura de Internet das coisas (IoT) capaz de englobar parâmetros e requisitos necessários para controlar sensores externos, como por exemplo, sensores de umidade, fumaça, temperatura, calor, vazamento, pressão, presença.

## 1.3 OBJETIVO GERAL

Propor uma arquitetura IoT para auxiliar e facilitar a implementação dos conceitos de Internet das coisas no monitoramento de vazamento de água. Para isso foi desenvolvido um aplicativo mobile para recepção e manipulação das informações fornecidas por um sensor de umidade conectado a uma placa Arduino.

## 1.4 OBJETIVOS ESPECÍFICOS

A partir do objetivo geral, foram definidos os seguintes objetivos específicos:

- Desenvolver um aplicativo *mobile* utilizando desenvolvimento híbrido (Ionic Framework[3]*) para* interação com o usuário;
- Desenvolver um *Web Service* utilizando a linguagem PHP para que seja feito a comunicação do Aplicativo com o Banco de dados;
- Criar uma base de dados (MySQL) para armazenar as informações coletadas;
- Adaptar o Arduino[4], utilizando uma placa ESP8266 para que seja possível que o mesmo se comunique com o *Web Service.*

## 1.5 ESTRUTURA DO TRABALHO

Esse trabalho está dividido em 5 capítulos, sendo descrito cada um a seguir.

---

[2] http://site.sanepar.com.br/a-sanepar
[3] https://ionicframework.com/
[4] https://www.Arduino.cc/

O Capítulo 2 apresenta o Referencial Teórico, descrevendo as ferramentas utilizadas, fundamentando assim a construção do aplicativo móvel. O capítulo 3 descreve a arquitetura proposta para implementar o conceito de Internet das coisas. O capítulo 4 apresenta o desenvolvimento do estudo de caso, onde é apresentado tudo que foi desenvolvido utilizando as ferramentas descritas no capítulo 2. O capítulo 5 apresenta as considerações finais, juntamente com trabalhos futuros, com o intuito de aperfeiçoar o que foi desenvolvido nesse projeto.

# 2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado o levantamento bibliográfico que foi realizado para auxiliar na elaboração tanto da arquitetura de IoT quanto ao estudo de caso, bem como tecnologias utilizadas, suas vantagens e o motivo que levou a escolha de cada uma delas.

## 2.1 INTERNET DAS COISAS

A Internet das Coisas (do inglês *Internet of Things* (IoT)), segundo Santos (2016), sugere que a Internet seja estendida para os objetos do nosso dia a dia, a fim de automatizar processos feitos apenas pela mão humana, por exemplo em cafeteiras, torradeiras, geladeiras, etc. O assunto surgiu dos avanços nos estudos em áreas como microeletrônica, comunicação e sensoriamento.

Segundo Brock (2001), o termo propriamente dito só apareceu em 2001 no livro branco de Brock. Porém, Kevin Ashton (2009) afirma que foi ele quem usou a expressão pela primeira vez enquanto falava sobre RFID. Naquele momento, a Internet das Coisas era citada chamando a atenção de empresários, dizendo que os computadores poderiam fazer “coisas” melhores do que pessoas, quando essas não tinham a seu favor a precisão, atenção e tempo.

Segundo Santos (2016), a pesquisa e o avanço da IoT podem favorecer tanto a academia quanto as indústrias, que podem desfrutar de novas ideias como, por exemplo, cidades inteligentes (*Smart Citys*) e automatização de ambientes.

O primeiro eletrodoméstico que utilizava deste conceito surgiu em junho de 2000, onde a LG apresentou ao público a primeira geladeira inteligente durante um evento na Coreia do Sul (SINGER, 2012). Em 2010 foi constatado que o número de “coisas” conectadas a Internet já superava o número de habitantes na terra. Essa informação é amparada pelo levantamento da Cisco IBSG apresentado no *White Paper de Evans* (2011, p. 3).

## 2.2 PLATAFORMA DE PROTOTIPAGEM ELETRÔNICA DE *HARDWARE* LIVRE

*Hardware* livre (*Open Hardware*) são circuitos eletrônicos ou *Hardware* de computador que podem ser copiados livremente, já que o diagrama esquemático é disponibilizado pelo próprio desenvolvedor. Na grande maioria das vezes, não é necessário adquirir nenhuma licença para utilizar o *Software,* porém o desenvolvedor pode exigir que seu nome seja incluído nos créditos do projeto final (THOMSEN, 2014).

Segundo Thomsem (2014), quando se fala em *Hardware* livre, a primeira impressão que se tem é de que a principal vantagem de utilizar tal recurso é a redução de custos. Essa afirmação não deixa de ser correta, no entanto, outra vantagem não menos importante, é a que existe a possibilidade de se contribuir com o desenvolvedor, acrescentando funcionalidades extras, identificando e corrigindo possíveis problemas que possam surgir no *Hardware* original.

## 2.3 PLATAFORMAS EXISTENTES NO MERCADO

Existem várias plataformas de *Hardware* livre no mercado. Entre as mesmas é possível citar o Projeto RepRap, que surgiu em 2004 com o objetivo de criar impressoras 3D, o RONJA, dispositivo transmissor de dados utilizando espaço livre[5] com a mesma qualidade de uma conexão *ethernet.* Pode-se citar também o *Uzebox, Hardware* de código livre onde é possível que qualquer pessoa monte seu próprio console de jogos (THOMSEN, 2014).

Para quem procura algo voltado a aplicações científicas, é possível encontrar um projeto de *Hardware* de câmera, desenvolvido por Elphel Inc.[6]. Devido ao conceito de livre, mesmo que seu uso seja voltado às aplicações científicas, pode ser facilmente personalizado para muitos tipos de aplicações (THOMSEN, 2014).

---

[5] Diferente da fibra ótica, esse sistema utiliza como meio de transmissão o ar, ou em alguns casos o próprio vácuo. Sua principal vantagem está no custo-benefício, já que sua instalação é mais barata, mais rápida e mais fácil quando comparada à transmissão com fio em geral (TELECO, 2018).

[6] https://www.elphel.com/.

## 2.4 ARDUINO

O Projeto Arduino teve início em 2005, e desde então, pode-se ter uma estimativa de que provavelmente já foram vendidas mais de 500.000 placas Arduino (MCROBERTS, 2011). O Arduino é uma pequena placa eletrônica, que contém um micro controlador Atmel[7], e circuitos de entrada e saída. Através de uma linguagem baseada em C/C++, pode ser programada para executar diferentes tarefas do cotidiano remotamente (Figura 1) (THOMSEN, 2014).

Segundo McRoberts (2011), um exemplo prático de onde pode ser encontrado uma placa Arduino é nos locais que possuem lâmpadas com acionamento por controle remoto, ou até mesmo desligamento automático. Nos projetos mais complexos, é possível ligar/desligar a lâmpada através de um comando enviado pelo celular.

Nos casos onde a lâmpada possui desligamento automático, é possível programar o Arduino conectando-o à um computador. Uma vez pressionado o botão, a lâmpada irá acender, e o processo de contagem de tempo será iniciado. Ao término do tempo programado para desligamento, a lâmpada se apaga e a placa ficará em repouso até que o botão seja pressionado novamente (MCROBERTS, 2011).

**Figura 1 - Imagem representativa de um Arduino UNO.**


**Fonte: MCROBERTS (2011)**

[7] Os microcontroladores AVR da fabricante ATMEL são micro controladores de 8 bits, desenvolvidos sob a tecnologia *RISC - Reduced Instruction Set Computer* (Computador com Set de Instruções Reduzido) e arquitetura *HAWARD* que separa a memória de dados da memória de programa (ARNEROBOTICS, 2018).

## 2.5 PROGRAMAÇÃO DE DISPOSITIVOS MÓVEIS

É possível citar três tipos de desenvolvimento para dispositivos móveis, o Nativo, *Web* e Híbrido. Durante o decorrer desta seção, serão explicados brevemente cada um deles.

### 2.5.1 Desenvolvimento Nativo

O desenvolvimento nativo é aquele onde não há dependência de outros *Hardware*s e/ou *Softwares* para o seu funcionamento. Isso torna o desenvolvimento seja mais rápido (TAVARES, 2016).

### 2.5.2 *Web Apps*

Segundo Tavares (2016), a principal característica dos *Web Apps* é a de que são executados com o auxílio de um navegador, pois são escritos em HTML 5. Os usuários os acessam inicialmente como fariam com um *website*, através de determinada URL e tem a opção de instalá-los na tela principal de seu dispositivo. Na verdade, é criado um atalho para aquela página que hospeda o serviço a ser utilizado.

### 2.5.3 Desenvolvimento Híbrido

Os aplicativos híbridos recebem esse nome porque podem ser baixados através de um aplicativo de loja (na Internet), utilizam todas as funcionalidades do dispositivo, e além de serem capazes de executar em um navegador. Essas características tornam esse tipo de aplicativo popular, principalmente pela capacidade de ser desenvolvido em multiplataformas, facilitando o seu reaproveitamento (TAVARES, 2016).

## 2.6 *WEB SERVICE*

Nas décadas de 1960 e 1970, *softwares* eram vistos como ferramentas de apoio para rotinas específicas. Não havia acoplamento entre sistemas, de modo que cada *software* não projetava uma visão ampla da organização. Assim, haviam diversos sistemas para tarefas isoladas o que, pela adoção do paradigma da arquitetura estruturada, acarretava elevados custos de manutenção (SAUDATE, 2013).

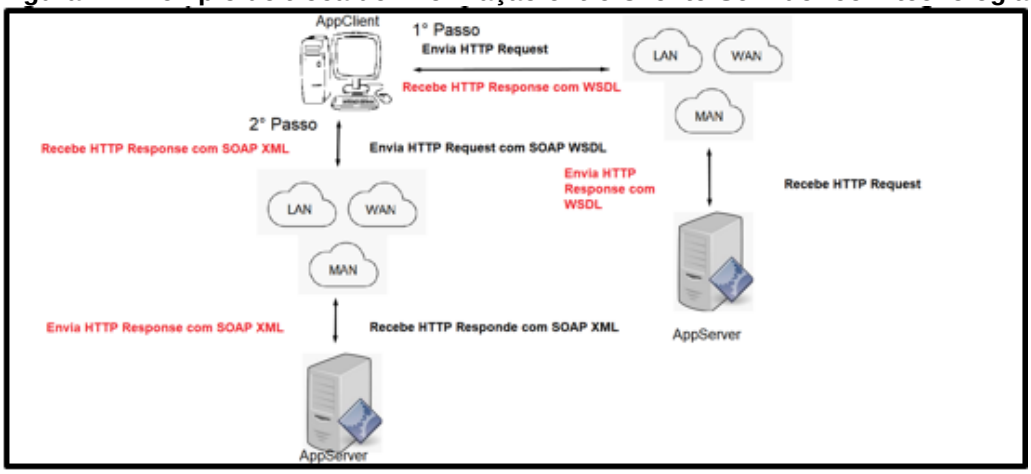
Neste universo de sistemas distribuídos, o termo serviço pode ser conceituado como a “execução de um trabalho ou realização de uma função de um prestador para um requisitante” (FUGITA; HIRAMA, 2012, p.2).

Na prática, o *Web Service* é responsável por fazer a comunicação entre o *front end* e o *back end*. Quando o usuário interage pela interface (*desktop*, aplicativo *mobile,* etc.) o *Web Service* encaminha as informações fornecidas pelo usuário até o Servidor. Existem dois tipos principais, o SOAP e o REST (RIBEIRO; FRANCISCO, 2016), explicados a seguir.

### 2.6.1 SOAP

Linguagem de anotação com a qual se pode descrever o protocolo de comunicação, responsável pela troca de mensagens de e para os *Web Services*. Uma mensagem SOAP é um documento XML (LOPES, 2004).

**Figura 2 - Exemplo de troca de informação entre Cliente-Servidor com tecnologia SOAP.**




**Fonte: Adaptado de Ferreira; Mota (2004)**

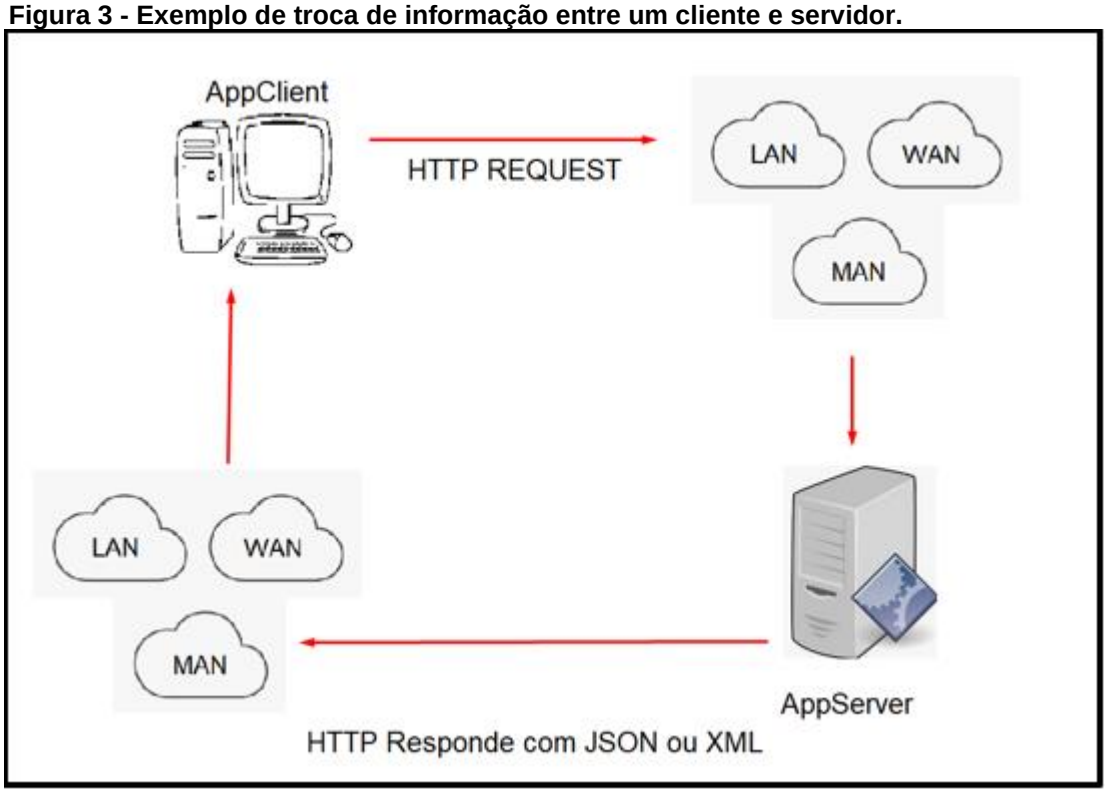
Na figura 2, é possível observar o funcionamento de um *Web Service* SOAP. A requisição é disparada do cliente (*appClient*), que pode ser de um celular, *notebook*, computador, etc. Em seguida essa requisição chega até o servidor (*AppServe*) que retorna então um arquivo *Web Services Description Language* (WSDL) (FERREIRA; MOTA, 2004).

WSDL é um arquivo que traz informações para o *AppClient* sobre como utilizar o serviço, tudo isso no formato XML. Dentro deste arquivo é possível encontrar informações importantes desde parâmetros até instruções de formatação de entrada e saída (FERREIRA; MOTA, 2014).

### 2.6.2 REST

Na arquitetura REST, não existe um descritor de funcionamento de serviço. A requisição realizada pelo cliente parte do princípio que a mesma sabe o que deve ser enviado para o Servidor, facilitando assim o processo de implementação (FERREIRA; MOTA, 2014).

**Figura 3 - Exemplo de troca de informação entre um cliente e servidor.**




**Fonte: Adaptado de Ferreira; Mota (2004)**

Foi escolhido REST para o trabalho devido ao fato de que estão pré-definidos todos os métodos e funções do sistema. Dessa maneira, houve total controle das informações que estavam sendo enviadas e das que estavam sendo recebidas.

## 2.7 LINGUAGENS DE PROGRAMAÇÃO *WEB*

Da mesma forma que nossas linguagens naturais, as linguagens de programação facilitam a expressão e comunicação de ideias entre pessoas. No entanto, em relação às linguagens naturais, as linguagens de programação possuem duas diferenças. A linguagem de programação também permite a comunicação de ideias entre pessoas e computadores. E também, possuem um domínio de expressão mais reduzido do que o das linguagens naturais. Isso significa que elas facilitam apenas a comunicação de ideias computacionais. Destarte, uma linguagem de programação deve satisfazer requisitos diferentes daqueles de uma linguagem natural (TUCKER, 2009).

As linguagens de programação *Web* são, de modo geral, utilizadas especificamente para desenvolvimento de sites, portais e aplicações *Web* em geral. A escolha de uma linguagem *Web* para iniciar o projeto ou até mesmo para aprender, depende de uma série de fatores. O principal fator é o objetivo: que resultado se espera dele? Para cada necessidade há uma opção mais adequada. São exemplos de linguagens de programação *Web:* Java, JavaScript, PHP, Python, Ruby (FINARDI; PREBIANCA, 2013).

### 2.7.1 Linguagem de Programação PHP

Originalmente denominada *Personal Home Page*, é uma linguagem interpretada livre bastante utilizada para implementação de funções dinâmicas e complexas aplicações *Web* e *Websites*. Sua grande vantagem é ser uma linguagem bastante simples para iniciantes. É uma poderosa ferramenta para desenvolvimento *Web* (ETTBRASIL, 2017).

Diferente do JavaScript, PHP tem seu código embutido no HTML e é executado diretamente no servidor, e para o cliente é enviado apenas o resultado

em HTML puro, tornando viável a interação com banco de dados e aplicações no servidor (ETTBRASIL, 2017).

A linguagem PHP cresceu e continua inovando rapidamente. Tem se baseado em dois princípios básico: se manter simples e ser fácil de aderir. Estatísticas mostram que cerca de 50% das aplicações *Web* pelo mundo são PHP, alcançando por volta de 240 milhões de *sites* (ETTBRASIL, 2017).

Obviamente a curva de aprendizado de uma linguagem depende da experiência e conhecimento do desenvolvedor. No entanto, quando comparada com outras linguagens bastante populares, a do PHP tende a apresentar menor curva de aprendizado. Outro fator importante é a extensa disponibilidade de documentações, tutoriais, artigos e até mesmo fóruns, que podem ser utilizados como guias para ajudar nos primeiros passos com o PHP (SCHMITZ, 2018).

## 2.8 BANCO DE DADOS

Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito. Por exemplo, considere nomes, números telefônicos e endereços de pessoas conhecidas. Esses dados podem ter sido escritos em uma agenda de telefones ou armazenados em um computador, por meio de programas eletrônicos. Essas informações são uma coleção de dados com um significado implícito, consequentemente, um banco de dados (ELMASRI, RAMEZ et al, 2005).

Um modelo de dados utilizado em Sistemas Gerenciadores de Banco de Dados (SGBD), é o modelo de dados relacional. Como exemplos de SGBD’s, é possível citar Oracle, MySQL, SQL Server, PostgreSQL (SCUDERO, 2018).

## 2.9 MYSQL

Um dos bancos de dados mais utilizados e populares, trata-se de uma tecnologia *Open Source*, ou seja, de código aberto. Isto permite que o desenvolvimento do mesmo seja realizado de acordo com as necessidades de uma organização (SCUDERO, 2018).

Para a realização deste projeto, o SGBD escolhido foi o MYSQL, devido ao fato de ser a ferramenta que mais foi utilizada durante o curso, consequentemente sendo a ferramenta em que os autores têm mais experiência e segurança para desenvolver.

# 3 ARQUITETURA PROPOSTA

Segundo Avelar (2010) a grande maioria dos produtos desenvolvidos utilizando do conceito de IoT, fazem uso de arquiteturas de 2 camadas (a de sensores e a de servidores), ou 3 camadas (as mesmas anteriores, com a adição de uma intermediaria, a de *Internet*).

Em ambas as estruturas, a primeira camada é onde são tratados os sensores e a segunda é responsável por armazenar os dados em um servidor. O que difere uma da outra, é que na arquitetura de três camadas, é usada uma camada intermediária que é responsável por escalonar a aplicação. Isso é possível usando, por exemplo, a Internet (AVELAR, 2010).

Baseado no conceito de arquitetura citado anteriormente, pode-se tomar como estrutura básica a de duas camadas, visto que em ambas, a camada de sensores e servidores são a mesma. Partindo desse conceito, fazem-se necessárias mais camadas para contemplar uma proposta mais robusta para gerenciar sensores que irão captar sinais externos (AVELAR, 2010).

A arquitetura proposta neste trabalho possui as seguintes camadas:

Camada de sensores: segue o padrão citado por Avelar (2010). É responsável pela captação dos sinais externos do ambiente para que outras camadas recebam as informações para um futuro tratamento. Esta pode representar qualquer tipo de sensor externo (umidade, fumaça, vazão de água, temperatura, movimento, presença, etc).

Camada de *front end*: recebe esse nome por ser o local onde ocorrerá a interação com o usuário. É responsável por apresentar, em uma interface gráfica, informações referentes aos comportamentos dos sensores e aos dados armazenados na camada de servidor.

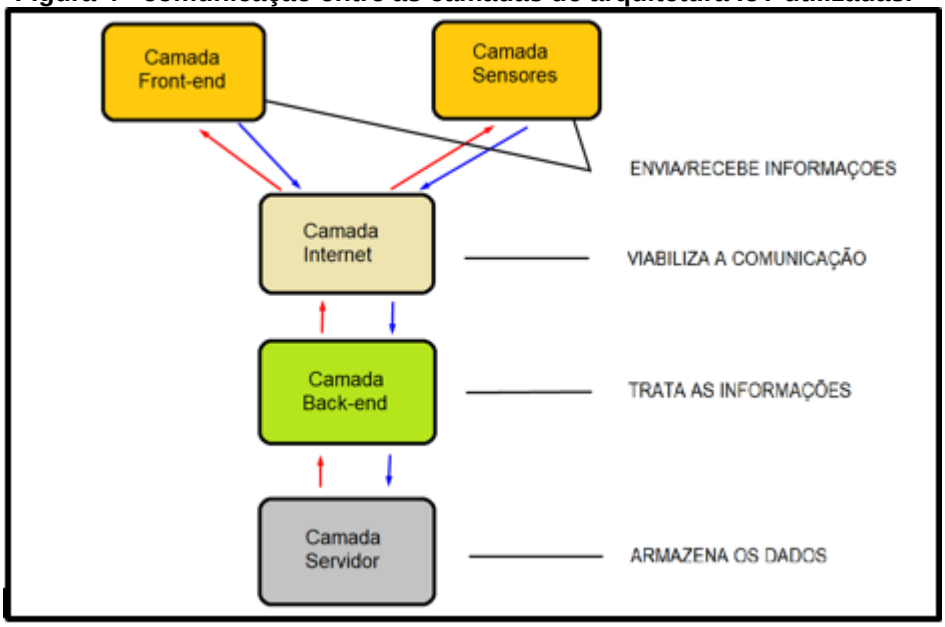
Camada de *back end*: responsável por receber todas as requisições vindas das camadas de sensores e front-end. Local em que a regra de negócio está armazenada, tornando mais fácil a manutenção e a escalabilidade.

Camada de Servidores: presente no padrão citado por Avelar (2010). Esta é responsável por armazenar todas as informações recebidas pela camada de *back end*. As informações são processadas e armazenadas nesta e, dentro do servidor, esses dados recebidos e armazenados podem acionar rotinas que tenham interação com os sensores.

Camada de Internet: responsável por realizar a comunicação entre todas as camadas e tornar o projeto escalável (com a adição de novos sensores externos e/ou novas ações).

A figura 4 ilustra a comunicação entre as camadas propostas neste capítulo.

**Figura 4 - comunicação entre as camadas de arquitetura IoT utilizadas.**




**Fonte: Autoria própria**

# 4 ESTUDO DE CASO

Este capítulo irá expor os detalhes referentes a implementação do projeto, a partir do conhecimento adquirido com as pesquisas mencionadas no capítulo 2, utilizando a arquitetura proposta na seção anterior.

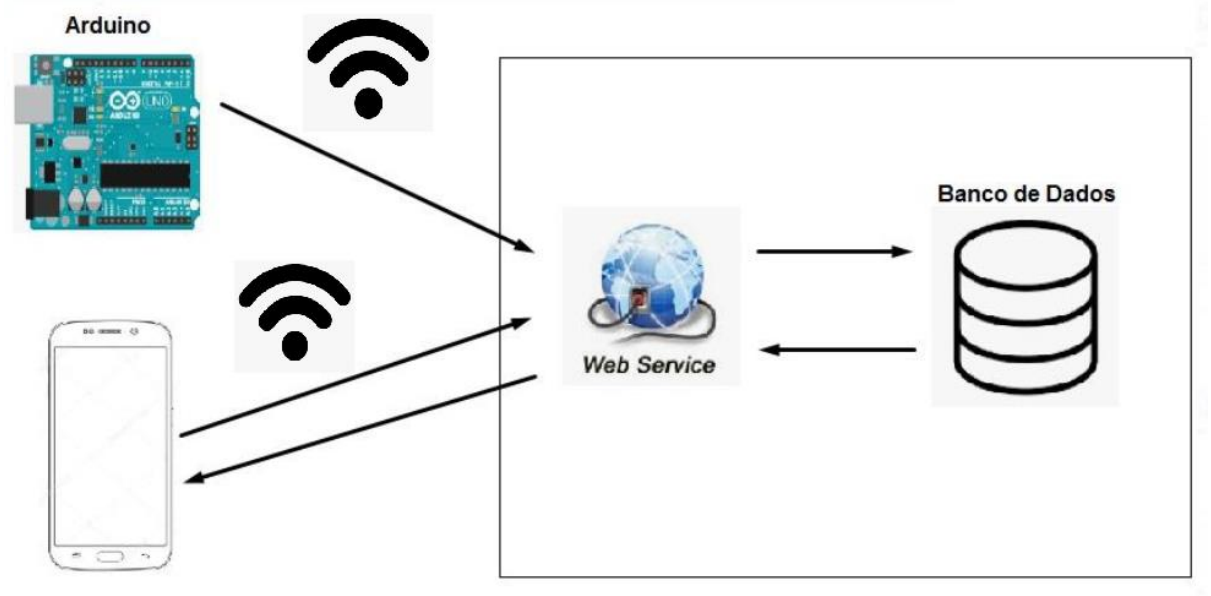
Para que o projeto fosse realizado, foram definidas algumas condições básicas para garantir que houvesse a troca de informações entre os dispositivos envolvidos *(smartphone* e Arduino).

Para tal comunicação existir, era preciso garantir que o projeto usasse a Internet. Com isso em mente, o mesmo foi desenvolvido em duas partes:

- Um aplicativo que fornece acesso às funcionalidades do sistema (cadastros, históricos, alertas).
- Uma plataforma que recebe as informações do *front end* (Aplicativo) e se comunica diretamente com o banco de dados, processando as informações que são enviadas e recebidas.

Esta comunicação é representada na figura 5.

**Figura 5 - Esquema de comunicação entre Cliente e Servidor**



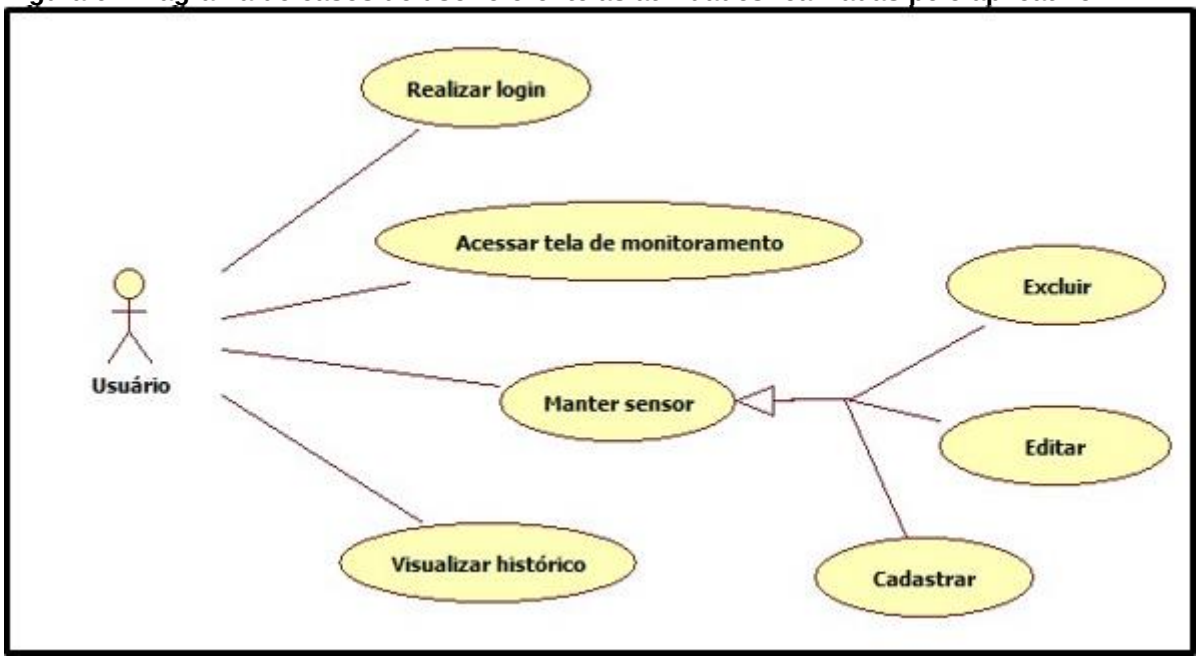

**Fonte: Autoria Própria**

O aplicativo se comunica diretamente com o *back end*. Este por sua vez, consulta no Banco de Dados e retorna a mensagem para o *front end* de uma maneira que possa ser interpretada.

O Arduino não necessita de retorno do *back end*, pois no contexto desta aplicação, o mesmo funciona como um “gatilho” que dispara uma requisição para o *back end*, que trata essa informação e retorna para o aplicativo.

Organizando a plataforma dessa maneira, torna a manutenção e/ou melhorias mais fáceis de serem implementadas. Na figura 6, pode-se observar as funcionalidades da aplicação do ponto de vista do usuário.

**Figura 6 - Diagrama de casos de uso referente as atividades realizadas pelo aplicativo.**




**Fonte: Autoria própria**

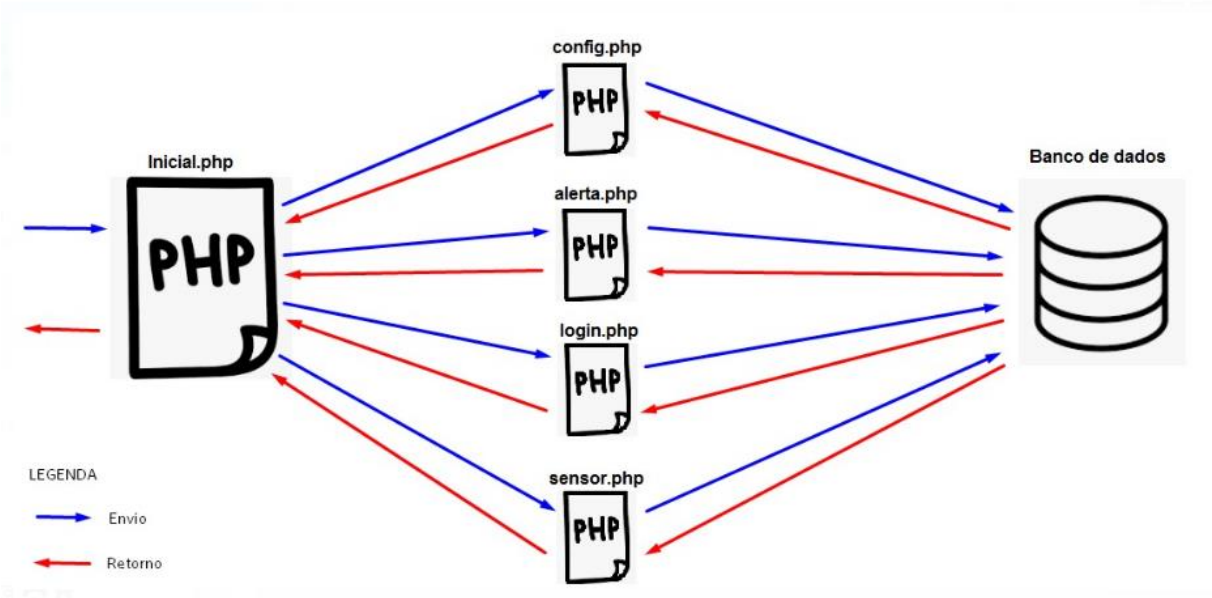
- Realizar Login: função responsável por garantir que apenas quem possui usuário cadastrado pode realizar alterações (cadastros, edições, exclusões de informações) e interações com o sistema.
- Manter sensor: o sensor é cadastrado pelo próprio aplicativo, tendo como parâmetros o número do sensor, latitude e longitude da localização que o sensor será instalado, e uma descrição da posição (nome de rua, referência próxima). O número do sensor deve ser o mesmo informado no Arduino.
- Acessar Painel: no painel, o usuário tem a opção de visualizar os vazamentos (através do aplicativo móvel) que ainda não foram verificados. O usuário tem a opção ainda de excluir um alerta de vazamento. Dessa forma, no painel é exibido apenas o que é crítico, o que está pendente.

## 4.1 SERVIDOR

A aplicação *back end* está alocada em um servidor, de modo a manter os recursos disponíveis a maior parte do tempo possível.

Os arquivos disponibilizados no servidor são responsáveis por receber as informações do aplicativo e do Arduino, encaminhar ao banco de dados e tratar o retorno para que seja encaminhado ao *front end*. As informações enviadas pelo *front end* são todas centralizadas a uma página chamada “inicial.php”. Nesta, estão todas as rotas do projeto. Cada rota, dá acesso a uma nova página que representa uma funcionalidade do sistema. A figura 7 ilustra a situação descrita.

**Figura 7 - ilustração da comunicação *back end* - Servidor.**




**Fonte: Autoria própria**

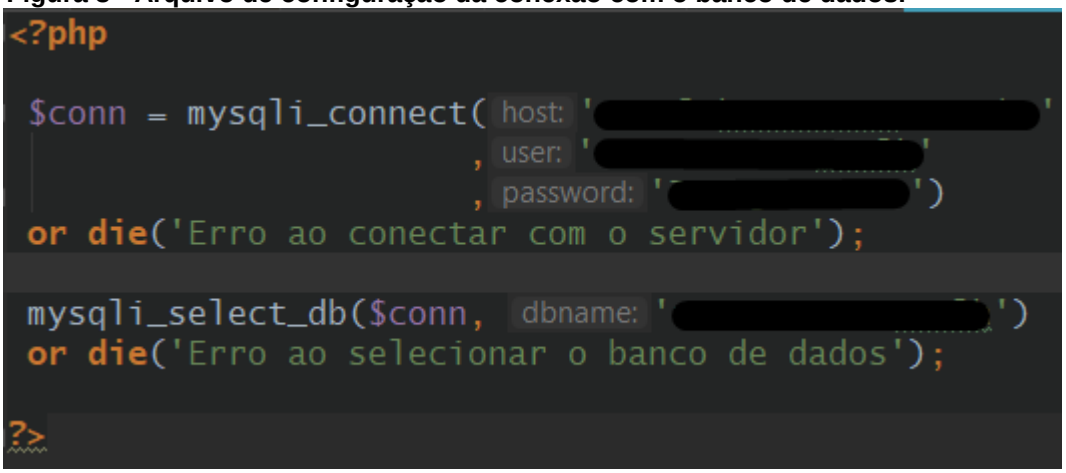
- inicial.php: neste arquivo, encontram-se as rotas que dão acesso às demais funcionalidades do sistema.

Para tratar as rotas, foi utilizada a função *explode* disponível no próprio PHP. Essa função recebe dois parâmetros: o delimitador e a *string* a ser particionada. O funcionamento da função *explode* pode ser descrito da seguinte maneira: a função localiza o delimitador (símbolo “?”) na *string* $_SERVER[‘REQUEST_URI’] (esse comando retorna a URL enviada para o *back end*), e divide a mesma em duas partes, uma antes do símbolo e uma depois. Caso existam mais símbolos, a URL é

dividida em maiores quantidades. No contexto deste projeto, foi sempre dividida em 2. Após isso, é atribuído a variável $URI apenas à primeira parte da *string*, conforme visto no código: "$URI = $URI[0];" e então é executado o comando *SWITCH* para verificar qual rota foi acionada. Dessa forma, apenas o recurso necessário é acionado, evitando assim a execução de código excessivo.

- config.php: aqui encontra-se apenas o método para conexão com o banco de dados, conforme pode-se ver na figura 8. Nesse arquivo é declarado a variável *$conn* que é usada no arquivo ***inicial.php*** e é passada como parâmetro para as demais classes, permitindo assim o acesso ao banco de dados nas outras funções do projeto.

**Figura 8 - Arquivo de configuração da conexão com o banco de dados.**




**Fonte: Autoria própria**

- alerta.php: neste arquivo, encontram-se os métodos responsáveis por manipular as informações que se encontram disponíveis no painel do aplicativo. Dentre elas:
  - InserirAlerta: essa função é acessada apenas pelo Arduino que, quando detecta umidade em um de seus sensores, dispara uma rotina que envia uma requisição HTTP para o *back end*, informando o número do sensor que detecta o vazamento. Nesse momento, é inserido um registro em uma tabela temporária chamada alerta, que é acessada pelo *front end* para tornar essa informação visível ao usuário.
  - excluirAlerta: essa função é outro exemplo de função que é acessada apenas pelo Arduino. Devido ao fato do sensor de umidade ser um

dispositivo de leitura analógica, existe uma porcentagem que precisa ser alcançada para ser considerado um vazamento. Logo, enquanto o *loop* da placa Arduino é executado, é validado em toda iteração se a umidade continua acima do nível de tolerância. A partir do momento que esse nível baixar, entende-se que o sensor não está mais detectando umidade. Nesse momento então, é disparado uma rotina que envia uma requisição para o *back end*, solicitando a exclusão do alerta.

  - consultarAlerta: responsável por consultar todos os alertas que estão pendentes (que não foram excluídos ainda, conforme citado no item anterior).

- login.php: nesse arquivo, encontra-se apenas o método para se logar no aplicativo. Ao realizar a consulta de usuário e senha no banco, e se o acesso for autorizado, é gerado um *token* que permite, por tempo limitado, o acesso às funcionalidades do aplicativo.

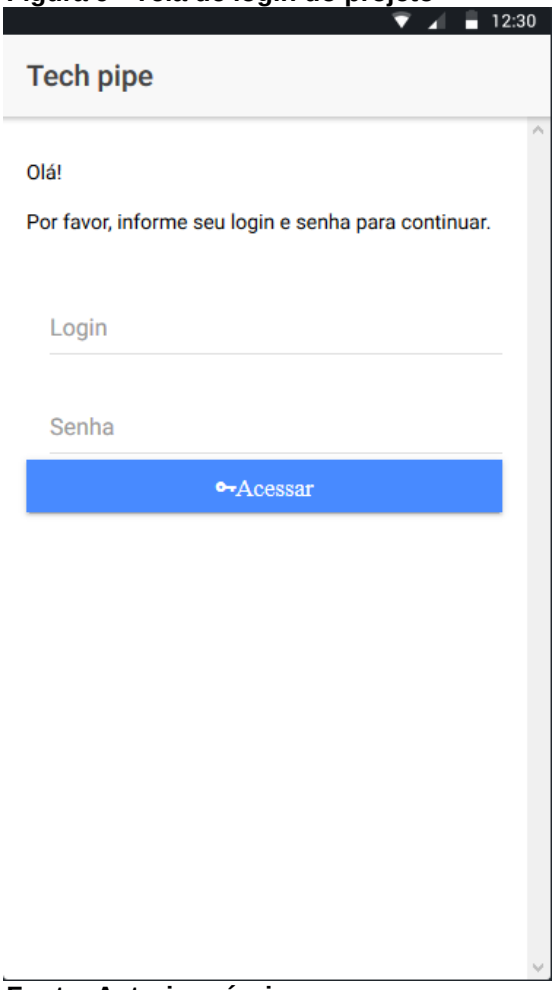
- sensor.php: neste arquivo, encontram-se os métodos responsáveis por manter o sensor (inserir, editar, excluir, consultar).

## 4.2 CLIENTE

A aplicação cliente (*front end*) tem seu ponto inicial, ou ponto de acesso, no arquivo ***home.html***, disponível dentro do diretório **src** do *framework* utilizado. Neste diretório, encontram-se todos os arquivos de código-fonte utilizados (referentes a aplicação cliente) para a execução deste trabalho.

No arquivo ***home.html*** foi disponibilizado um login para que seja feito o controle daqueles que possuem acesso ao aplicativo, conforme figura 9.

**Figura 9 - Tela de login do projeto**

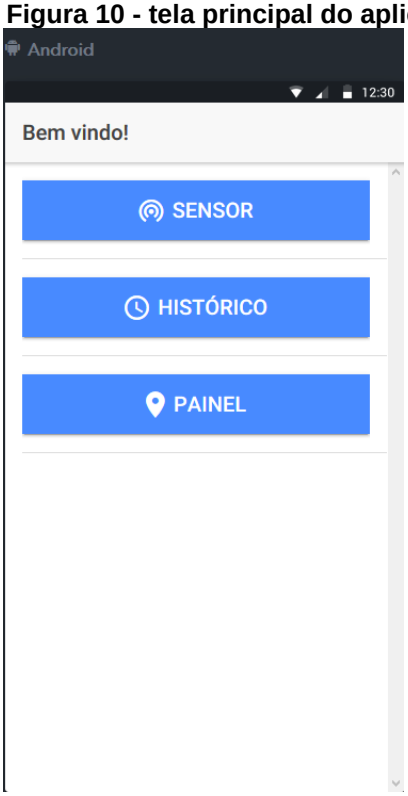



**Fonte: Autoria própria**

Após efetuado o *login*, é possível acessar a interface principal do projeto, através do arquivo chamado de ***principal.html***.

Nessa interface (figura 10), encontram-se os acessos para todas as funcionalidades do aplicativo.

**Figura 10 - tela principal do aplicativo, principal.html.**




**Fonte: autoria própria**

Sensor: permite o acesso para a tela de cadastro de sensor. Nesta tela, é informado os dados necessários para que o mesmo possa ser registrado no banco de dados. Dentre os dados necessários, existem a latitude e a longitude que, no contexto do aplicativo, são as informações mais importantes para que o aplicativo funcione com a precisão esperada. Foi implementado um recurso para que seja capturado a latitude e longitude da posição do aparelho no momento. Para tal, foi necessário instalar o *plug-in* nativo do *Android* chamado *Geolocation*[8]. No prompt de comando disponibilizado pela IDE utilizada, foram informados os seguintes comandos:

```
$ ionic cordova plugin add cordova-plugin-geolocation --variable
GEOLOCATION_USAGE_DESCRIPTION="To locate you"
$ npm install --save @ionic-native/geolocation
```

[8] https://ionicframework.com/docs/native/geolocation/

Esses comandos são responsáveis por baixar e instalar o *plug-in* de geolocalização nativo do sistema operacional *Android.* Após isso, foram implementados os seguintes métodos:

**Figura 11 - função getCoordenadas()**

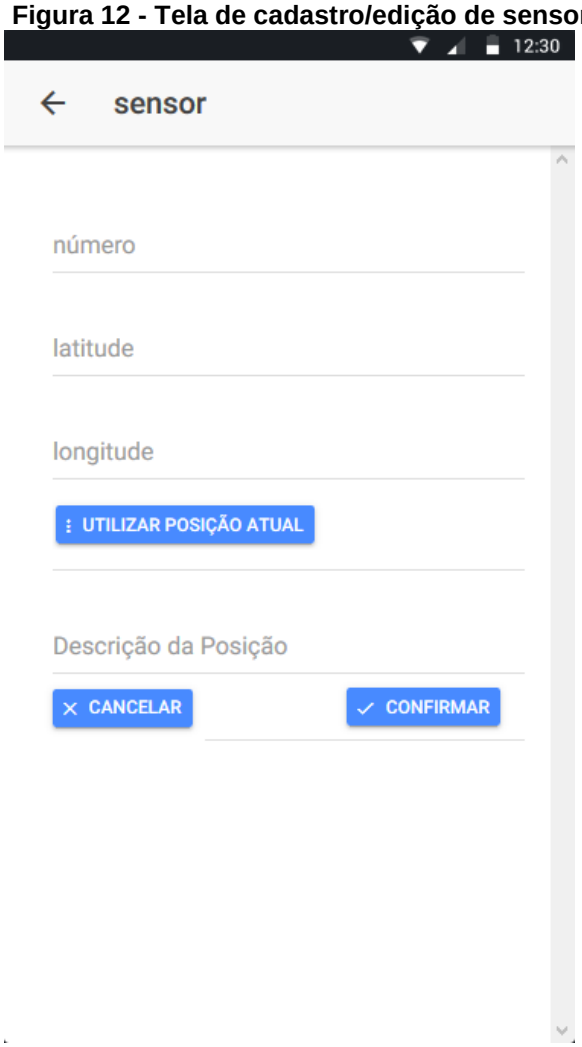
```
getCoordenadas(){
  let stream = this.geolocation.watchPosition();
  stream.subscribe(data => {
    this.coordenadas.latitude = data.coords.latitude;
    this.coordenadas.longitude = data.coords.longitude;
  })
}
```

**Fonte: Autoria própria**

A função implementada getCoordenadas() possui o método nativo do ***plug-in*** de geolocalização chamado *WatchPosition*(). Este, por sua vez, é responsável por capturar a latitude e longitude da posição atual do aparelho que realizou a chamada do método. Na implementação mostrada na figura 11, estão sendo atribuídos os valores da latitude e da longitude obtidos, à duas variáveis criadas durante o desenvolvimento: ***coordenadas.latitude*** e ***coordenadas.longitude***. Estas são apresentadas na tela e, posteriormente, salvas no Banco de dados caso o usuário decida por concluir o cadastro do sensor.

O método getCoordenadas() é opcional e é chamado através de um botão na interface disponibilizada através do arquivo ***sensor.html***, conforme figura 12.

**Figura 12 - Tela de cadastro/edição de sensor.**




**Fonte: Autoria Própria**

Histórico: nesse menu é possível visualizar os históricos de vazamentos dos sensores previamente cadastrados no banco de dados e no Arduino. Com isso, é possível que o usuário (que vem a ser o responsável pela manutenção dos sensores), possa analisar, durante uma inspeção periódica, os sensores que possuem a maior taxa de vazamentos. Com isso, podem ser tomadas precauções para que sejam evitados futuros vazamentos reincidentes.

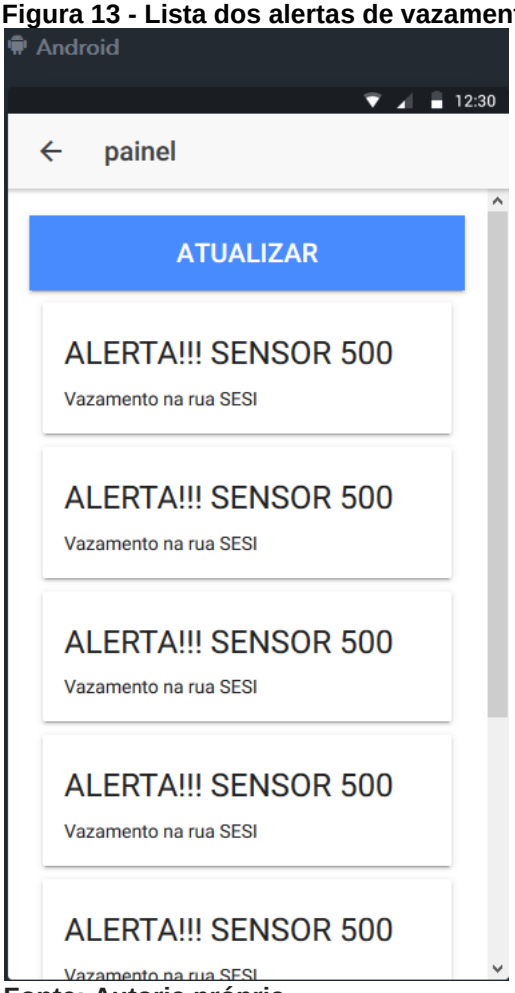
O histórico pode ser consultado utilizando três filtros individuais ou combinados. Dentre eles: número do sensor, período (data inicial e data final) e pode ser filtrado também ignorando-se as duas opções anteriores. Dessa maneira, o

sistema irá carregar todos os vazamentos registrados, e que já foram resolvidos, na base de dados.

Monitoramento: nesse menu encontra-se a funcionalidade principal deste projeto. Através dele é possível monitorar se existe algum sensor que acusou umidade acima do normal. É disponibilizado um botão para que possa ser atualizado a lista de forma manual para validar os vazamentos pendentes de verificação. Caso um sensor acuse umidade, o mesmo será apresentado na lista da interface ***painel.html***. Ao clicar no item dessa lista, o usuário é direcionado a uma tela onde é mostrado, em um mapa (através do *google maps*[9]), a posição onde se encontra o sensor, e nesta mesma tela, o usuário pode realizar a conferência do alerta.

Na figura 13 é possível verificar um exemplo de como ficaria a tela com vários sensores acusando vazamentos simultâneos.

**Figura 13 - Lista dos alertas de vazamentos.**

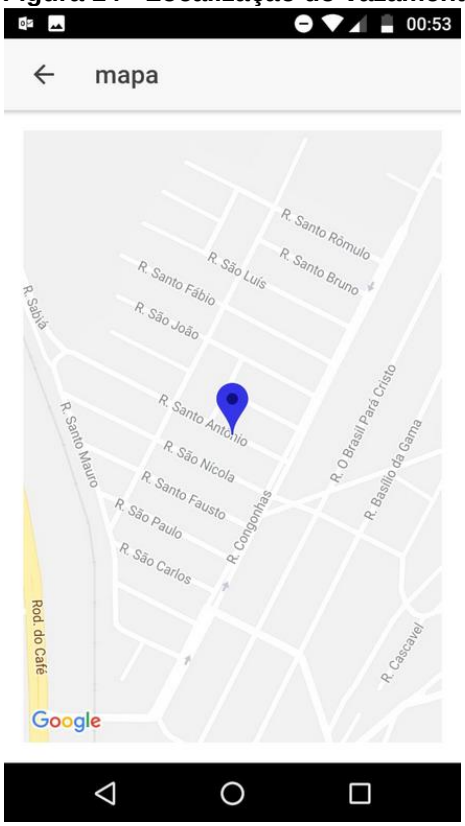



**Fonte: Autoria própria**

[9] https://www.google.com/maps

Na figura 14 é possível verificar o mapa com o ponto marcando o local do vazamento indicado.

**Figura 14 - Localização do vazamento.**




**Fonte: Autoria Própria.**

## 4.3 ADAPTANDO O ARDUINO A ARQUITETURA

Para implementação do estudo de caso referente a camada de sensores, foi utilizada uma proposta feita pelos alunos do colégio SESI da cidade de Irati, que implementaram um detector de umidade. O trabalho consiste em um sensor de umidade conectado a uma placa Arduino que, quando acusa umidade, dispara um sinal luminoso em um painel local. Coube aos autores desse trabalho, implementar a arquitetura proposta de modo a melhorar a confiabilidade nas informações captadas.

O código fonte gerado pela equipe do colégio SESI, por si só, já estava pré-programado para que, quando o sensor captasse umidade, disparasse um sinal em um painel local. Coube também aos autores deste trabalho adaptar o código para que, nesta situação, fosse enviado uma requisição HTTP para o *back end*, contendo o número do sensor que acusou o vazamento.

Para isso, inicialmente, foi necessário conectar ao Arduino uma placa de rede sem fio, para possibilitar a conexão da mesma com a camada de Internet, para realizar tal ação, foi utilizada a placa ESP-8266[10]. Após isso, dentro da condição de checagem de umidade, que se encontra dentro do método *loop* do código fonte, foi adicionado uma requisição POST enviando o número do sensor que acusou vazamento para a URL “...xyz/inicial.php/alerta/inserir”. A partir disso, os processos presentes na arquitetura são desencadeados, gerando todo o processo logístico visto neste capítulo.

[10] https://www.filipeflop.com/blog/guia-do-usuario-do-esp8266/

# 5 CONCLUSÃO

Foi desenvolvido e testado um aplicativo mobile utilizando desenvolvimento híbrido, em conjunto com uma base de dados que foi criada para armazenar todas as informações fornecidas pelo dispositivo móvel e pela placa Arduino. A comunicação entre o banco de dados, Arduino e o aplicativo móvel foi possível devido à implementação de um ***Web Service***. A adaptação do código fonte utilizado no Arduino para se comunicar com a Internet, e a configuração de uma placa Wi-Fi, possibilitaram o envio de dados da placa Arduino para o ***Web Service***.

## 5.1 CONSIDERAÇÕES FINAIS

Utilizando os conhecimentos obtidos com as pesquisas realizadas nos capítulos 1 e 2, juntamente com a arquitetura proposta no capítulo 3 e o desenvolvimento do estudo de caso apresentado no capítulo 4, chegou-se ao resultado que será apresentado neste capítulo.

A integração das ferramentas apresentadas, unindo-se com as pesquisas realizadas para aprofundamento do assunto e também o conhecimento adquirido ao longo do curso, possibilitou o desenvolvimento do aplicativo e da arquitetura IoT, integrando as duas e garantindo sua funcionalidade.

O aplicativo foi validado em ambiente de testes, que consistiu em simular vários casos de umidade no sensor para que fosse enviado o sinal para o *Web Service* e, em seguida, verificado no painel de monitoramento se o alerta condizia com a realidade (devido às informações das posições geográficas).

## 5.2 TRABALHOS FUTUROS

O futuro deste projeto consiste em torná-lo mais robusto, adicionando novas funcionalidades para que aumente a utilidade do mesmo perante a sociedade.

Uma melhoria interessante para o projeto, seria o desenvolvimento de uma aplicação (Java por exemplo), separando o projeto em gerencial e operacional. Dessa maneira, a parte gerencial (relatórios, históricos) ficaria na aplicação Java (instalado em computadores e operado por funcionários da empresa que contratar o serviço), deixando para o funcionário responsável pela instalação e manutenção

apenas as informações cadastrais e o acesso ao painel. Isso é possível, pois com a arquitetura dividida em camadas, um novo componente na camada *front end* não irá impactar negativamente nas outras camadas, pelo fato de ser facilmente escalável. Um processo que pode e precisa ser revisto é uma maneira de automatizar o processo de cadastro de sensores, para evitar que o código fonte executado no Arduino seja alterado manualmente, conforme explicado na subseção anterior. Para isso, seria necessário realimentar o sensor com informações, e não somente retirar dados. Com isso desenvolvido e concluído, a arquitetura proposta nesse trabalho se torna uma arquitetura 100% fiel ao conceito de IoT.

## REFERÊNCIAS

ASHTON, Kevin et al. That ‘internet of things’ thing. **RFID journal**, v. 22, n.7, p. 97-114, jul. 2009.

AVELAR, Edson Adriano M.; AVELAR, Lorena M.; DIAS, Kelvin Lopes; SILVA, Diego dos Passos. Arquitetura de Comunicação para Cidades Inteligentes: Uma proposta heterogênea, extensível e de baixo custo. In: XXXIX Seminário Integrado de Software e Hardware, 2012, Curitiba, **Anais** do CSBC, 2012.

BARROS, Fernanda Genes Nunes; AMIN, Mario M.. Água: um bem econômico de valor para o Brasil e o mundo. **Revista Brasileira de Gestão e Desenvolvimento Regional**, Taubaté (SP), v. 4, f. 75, 2008.

ELMASRI, Ramez et al. **Sistemas de Banco de dados**: Fundamentos e Aplicações. 4. ed. Addison Wesley, 2005.

FERREIRA, Cleber de F.; MOTA, Roberto Dias**.** Comparando aplicação *web service* rest e soap. In: XVI SEINPAR- Semana de Informática e XII Mostra de trabalhos de Iniciação científica de Paranavaí, 2014, **Anais...**Paranavaí - PR. Artigos de pós-graduação, 2014.

FINARDI, Kyria Rebeca; PREBIANCA, Gicele Vergine; MOMM, Christiane Fabiola. Tecnologia na Educação: o caso da Internet e do Inglês como Linguagens de Inclusão. **Cadernos do IL**, Porto Alegre (RS), v. 46, f. 193-208, 2013.

LOPES, Carlos J. Feijó; RAMALHO, José Carlos. Web Services: Metodologias de Desenvolvimento. In: XML, aplicações e tecnologias associadas (XATA). 2004, Porto, **Anais** Porto: FEUP, 2004. p. 1-15.

MCROBERTS, Michael. **Arduino básico.** 2. ed. São Paulo: Novatec, 2015.

ETTBRASIL. **Por que o PHP é a melhor opção?** 2017. Disponível em: <http://ettbrasil.com.br/portal/blog/por-que-o-php-e-a-melhor-opcao/>. Acesso em: 10 nov. 2018

RIBEIRO, Michel F.; FRANCISCO, Rodrigo E. WEB SERVICES REST CONCEITOS, ANÁLISE E IMPLEMENTAÇÃO. **Educação, Tecnologia e Cultura-ETC**, v. 14, 2016.

SANTOS, Bruno P. et al. Internet das coisas: da teoria à prática. In: SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos. 2001, Santa Catarina: **Anais** UFMG, 2016. p. 2-51.

SAUDATE, A. **SOA aplicado**, integrando com web services e além. 1. ed. São Paulo: Casa do Código, 2014

SCHMITZ, Mateus. **Linguagem PHP: Porque Aprender a Utilizar**. Disponível em: <https://king.host/blog/2018/06/linguagem-php-porque-aprender-a-utilizar/>. Acesso em: 03 dez. 2018.

SCUDERO, Erick. **TOP 10 principais SGBDs do mercado global**. Disponível em: <https://becode.com.br/principais-sgbds/>. Acesso em: 03 dez. 2018.

SINGER, Talita. Tudo conectado: conceitos e representações da internet das coisas. In: SIMSOCIAL - Simpósio em Tecnologias digitais e sociabilidade. 2012, Salvador. **Anais** do SIMSOCIAL, 2012.

SOARES, Márcio José. **OS MICROCONTROLADORES AVR ATMEL - NOÇÕES BÁSICAS**. Disponível em: <http://www.SOARES.com.br/eletronica/Microcontroladores_AVR_basico.htm>. Acesso em: 25 nov. 2018.

TAVARES, Henrique Leal. Introdução a Desenvolvimento de Aplicações Híbridas. **Revista Eletrônica e-F@tec**, Garça (SP), v. 6, n. 1, p. 11-11, 2016.

THOMSEN, Adilson. **O que é Arduino?** Disponível em: <https://www.filipeflop.com/blog/o-que-e-Arduino/>. Acesso em: 04 dez. 2018.

THOMSEN, Adilson. **Vamos falar de Open Hardware?** Disponível em: <https://www.filipeflop.com/blog/open-hardware-livre/>. Acesso em: 03 dez. 2018.

TUCKER, Allen; NOONAN, Robert. **Linguagens de Programação:** Princípios e Paradigmas**.** 2. ed. Porto Alegre, AMGH, 2009.

VILELA, Dênio Ferreira de Lima et al. **Estudo da Viabilidade de Comunicações Ópticas no Espaço Aberto – I**. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialfsoeab1/default.asp>. Acesso em: 03 dez. 2018.