CONIC SEMESP

18º Congresso Nacional de Iniciação Científica

TÍTULO: O USO DA IOT NO CONTROLE DE FREQUÊNCIA DOS ALUNOS DA UNIP

CATEGORIA: CONCLUÍDO

ÁREA: CIÊNCIAS EXATAS E DA TERRA

SUBÁREA: Engenharias

INSTITUIÇÃO(ÕES): UNIVERSIDADE PAULISTA - UNIP

AUTOR(ES): BRUNO ARAKAKI JUNQUEIRA SILVA

ORIENTADOR(ES): FILIPE ANTOINE KHATCHADOURIAN

UNIP UNIVERSIDADE PAULISTA

SEMESP

Vice-Reitoria de Pós-Graduação e Pesquisa

UNIVERSIDADE PAULISTA

VICE-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA

RELATÓRIO FINAL DE PESQUISA

O USO DA IOT NO CONTROLE DE FREQUÊNCIA DOS ALUNOS DA UNIP

AUTOR: Bruno Arakaki Junqueira Silva

ORIENTADOR: Prof. Me. Filipe Antoine Khatchadourian

2018

8º Semestre

São Paulo – SP

Pesquisa Financiada pela Vice-Reitoria de Pós-Graduação e Pesquisa da UNIP, no Programa “Iniciação Científica”.
É proibida a reprodução total ou parcial.

O USO DA IOT NO CONTROLE DE FREQUÊNCIA DOS ALUNOS DA UNIP

AUTOR: Bruno Arakaki Junqueira Silva

UNIVERSIDADE PAULISTA

Resumo:

A Internet das Coisas (*Internet of Things - IoT*) é a tecnologia que conecta os diversos objetos de nosso dia a dia à rede, proporcionando diversas funcionalidades que facilitem e melhorem nossos processos cotidianos. A tecnologia RFID, mais conhecidas como etiquetas inteligentes já é considerada uma tecnologia do futuro que começa a estar presente em nosso cotidiano. Atualmente as tecnologias de identificação em banco de dados, estão se tornando popular no mundo todo, e esse projeto segue essa atualidade de internet das coisas visando a integração de ambos para soluções mais práticas de desafios. As instituições de ensino realizam o controle de frequência de seus alunos por processos manuais, que depois devem ser inseridos em sistema. O processo manual, além de promover falhas, desperdiça o precioso tempo dos docentes em sala de aula. Este trabalho propõe uma ferramenta tecnológica que, com o uso de um microcontrolador conectado a um leitor *RFID*, registra a frequência dos alunos num banco de dados armazenado em nuvem, possibilitando a intervenção e o controle do professor por meio de aplicativo em *smartphone*. Tudo isso levando em consideração, que o projeto tem também foco voltado para que os custos financeiros com os materiais e componentes finais sejam os mais baixos possíveis, sendo provável adquirir todos eles por menos de R$50,00.

Palavras-Chave: *IoT; RFID; Microcontroladores ESP8266.*

ABSTRACT

Internet of Things (IoT) is the technology that connects the diverse objects of our day to day to the network, providing diverse functionalities that facilitate and improve our daily processes. RFID technology better known as smart tags is already considered a technology of the future that begins to be present in our daily lives. Currently, database identification technologies are becoming popular worldwide, and this project follows the current internet of things aimed at integrating both for more practical solutions to challenges. Educational institutions carry out the frequency control of their students through manual processes, which must then be inserted into the system. The manual process, in addition to promoting failures, wastes the precious time of teachers in the classroom. This work proposes a technological tool that, with the use of a microcontroller connected to an RFID reader, registers the frequency of the students in a database stored in the cloud, allowing the intervention and control of the teacher through a smartphone application. All this taking into consideration, that the project has also focused on the financial costs with the final materials and components are as low as possible, being likely to acquire all of them for less than $ 10 Dollars.

Key words: IoT; RFID; Microcontrollers ESP8266.

Lista de figuras

Figura 1: *Arduino UNO* 12
Figura 2: Módulo RFID RC522 12
Figura 3: Módulo *Ethernet* 13
Figura 4: ESP8266 14
Figura 5: *NodeMCU* 15
Figura 6: *Driver* ESP8266 16
Figura 7: Conexão *RFID NodeMCU* 17
Figura 8: *WAMP Server* 18
Figura 9: Acessos 18
Figura 10: Inserção 19
Figura 11: prof 20
Figura 12: Usuarios 20
Figura 13: *OK* 21
Figura 14: *NOTAG* 22
Figura 15: Formulário 22
Figura 16: Lista de Acessos 23
Figura 17: Excluindo 23
Figura 18: Cadastrando professor 25
Figura 19: Tela inicial prova 25
Figura 20: Editando 26
Figura 21: Imagens complementares do aplicativo 27
Figura 22: *Download* do app 27
Figura 23: Subindo arquivos 28
Figura 24: *Site* 29
Figura 25: listacess no link 29

SUMÁRIO

1. INTRODUÇÃO 6
2. OBJETIVOS 8
3. JUSTIFICATIVA 9
4. MATERIAIS 10
   4.1. MATERIAIS 10
   4.2. *SOFTWARES* 10
5. DESENVOLVIMENTO DO PROJETO 11
   5.1. PRIMEIRA ETAPA 12
      5.1.1. Primeiras impressões *Arduino* e *RFID* 13
      5.1.2. Conexão com a *internet* através do Módulo *Ethernet* 13
      5.1.3. Substituição do módulo *Ethernet* 14
      5.1.4. Substituição do microcontrolador 15
      5.1.5. Testes com *NodeMCU* e *RFID* integrados 17
   5.2. SEGUNDA ETAPA 17
      5.2.1. Primeiras impressões *WAMP Server* 17
      5.2.2. Criação do primeiro banco de dados 18
   5.3. TERCEIRA ETAPA 21
      5.3.1. Conectividade *MySQL* e *NodeMCU* através do *PHP* 21
      5.3.2. Teste do sistema geral 24
   5.4. QUARTA ETAPA 24
      5.4.1. Construção do aplicativo 24
      5.4.2. Funcionamento 26
      5.4.3. Instalação em sistema *Android* 27
   5.5. QUINTA ETAPA 27
      5.5.1. Hospedagem em nuvem 28
6. RESULTADOS 30
7. DISCUSSÃO 31
8. CONCLUSÃO 32
9. REFERÊNCIAS 33
ANEXO A 34
ANEXO B 39

ANEXO C **42**

ANEXO D **48**

ANEXO E **52**

ANEXO F **53**

ANEXO G **55**

ANEXO H **57**

ANEXO I **59**

ANEXO J **60**

ANEXO K **61**

ANEXO L **62**

ANEXO M **63**

1. INTRODUÇÃO

Imagine que você se encontra no trânsito e está atrasado para receber uma visita em casa. Enquanto você dirige, é possível executar comandos para que seu fogão se prepare para cozinhar, ou que seu aspirador de pó comece a limpar a sala. Isso tudo será possível com a Internet das Coisas. [6]

Internet of Things - IoT, popularmente conhecida no Brasil como “Internet das Coisas” é uma infraestrutura de rede dinâmica e global que nos possibilita realizar alterações de acordo com meios de comunicação que seguem uma linha padrão, de modo que “coisas” virtuais e físicas tenham sua própria identificação, formas físicas e identidades virtuais. A *IoT* é um modelo de evolução da tecnologia onipresente, que atua em diferentes dispositivos inteligentes no ambiente, por meio de sensores controlados, etiquetas *NFC*, etiquetas *RFID*, *Smartphones,* etc. Estes dispositivos têm a capacidade de comunicar-se uns com os outros para um objetivo específico facilitando a vida da sociedade. [1]

Segundo Kevin Ashton (1999) o termo *Internet of Things* é usado para designar métodos que projetem objetos conectados à *internet*, capazes de enviar informações em tempo real de maneira automática. [2]

De acordo com o Terra (2013), por volta de 2020, a Internet das Coisas vai atingir quase 50 bilhões de equipamentos conectados, ou seja, praticamente seis dispositivos por pessoa. Segundo o *McKinsey Global Institute* (2013), foi registrado um aumento de 300% no número de dispositivos conversando entre si nos últimos cinco anos. [5][7]

A definição de *IoT* não é novidade e vem tomando espaço na última década devido ao ponto de vista prático das grandes empresas tecnológicas que vem incentivando o consumo dessa nova tecnologia. Isso tudo se torna possível devido a grande evolução dos *hardwares* nos últimos anos e de: reduções de custos, maior eficiência do uso de energia, a evolução das memórias, o poder crescente dos microprocessadores e microcontroladores, e os avanços da telecomunicação. A definição de *IoT* reúne várias áreas de atuação e conhecimento, sendo extremamente importante para qualquer cadeia produtiva. A utilização do termo *IoT* é diferente em alguns países em razão dos seus limites nacionais. O termo é mais usual em países

da Europa e na China. Nos Estados Unidos são utilizados os termos *cloudcomputing*, *smart grid* e *smartobjects*. [3][4][8]

Este projeto foi desenvolvido em cinco etapas diferentes para melhor esclarecimento:

- A primeira etapa aborda a pesquisa e o aprendizado em relação ao Módulo RFID-RC522 e sua interação com o a plataforma microcontrolada *Arduino*. Também serão mostradas suas funcionalidades e porque foi escolhido substituir a placa de desenvolvimento *Arduino* pela *NodeMCU*.
- A segunda etapa consiste no primeiro contato com o pacote *Wamp Server* e suas principais ferramentas de uso para integração com o projeto. Dentre elas o *MySQL* (banco de dados).
- Na terceira etapa foi demonstrado todo o desenvolvimento e testes feitos com integração das etapas um e dois.
- A quarta etapa trata de mostrar a construção de um aplicativo para *smartphone android* desenvolvido no *MIT App Inventor 2* e também suas funcionalidades em interação com o banco de dados.
- A quinta e última etapa consiste na hospedagem em nuvem de todo o projeto.

2. OBJETIVOS

Dentre os objetivos deste trabalho, podemos destacar:

- Pesquisas da atualidade de aplicações com *IoT*;
- Desenvolver aplicações com microcontrolador ESP8266;
- Estudos e uso da tecnologia de leitores *RFID*;
- Criação de banco de dados em nuvem;
- Desenvolvimento de aplicativos para sistema operacional *Android*.

3. JUSTIFICATIVA

As instituições de ensino realizam o controle de frequência de seus alunos por processos manuais, que depois devem ser inseridos em sistema. O processo manual, além de promover falhas, desperdiça o precioso tempo dos docentes em sala de aula. Este trabalho propõe uma ferramenta tecnológica que, com o uso de um microcontrolador conectado a um leitor *RFID*, registra a frequência dos alunos num banco de dados armazenado em nuvem, possibilitando a intervenção e o controle do professor por meio de aplicativo em *smartphone*.

4. MATERIAIS

4.1. MATERIAIS

- *Protoboard* (400 pontos);
- Plataforma de prototipagem eletrônica *NodeMCU*, com microcontrolador ESP8266;
- Plataforma de prototipagem eletrônica *Arduino*, com microcontrolador ATmega328;
- Módulo *Ethernet*;
- Módulo RFID-RC522;
- *Tags RFID*;
- *Tags NFC*;
- Jumpers.

4.2. *SOFTWARES*

- *Notepad*++;
- *IDE Arduino*;
- *Fritzing*;
- *Wamp Server*;
- *MySQL* (Banco de dados);
- *MIT App Inventor 2*;

5. DESENVOLVIMENTO DO PROJETO

Para o desenvolvimento deste projeto, inicialmente foi realizado uma busca nas revistas e periódicos relacionados à área sobre as aplicações recentes de *IoT*.

Em seguida, foi necessária uma ambientação com as tecnologias a serem utilizadas no projeto, no caso: *Arduino*, *NodeMCU*, leitores e etiquetas *RFID*, banco de dados, acesso de dados em nuvem e desenvolvimento de aplicativos *Android*.

Depois, a instalação e uso dos *softwares* para as tecnologias citadas, buscando-se utilizar de licenças de *software* livre ou educacional.

A implementação do projeto se dará em partes distintas, divididas em cinco fases:

- 1 – Dispositivo microcontrolado em comunicação com leitor de RFID;
- 2 – Criação e manipulação de banco de dados;
- 3 – Integração das etapas um e dois;
- 4– Desenvolvimento de aplicativo para o sistema operacional *Android*;
- 5 – Interação do projeto em nuvem.

Por fim, a integração das partes e a realização de testes de funcionamento.

5.1. PRIMEIRA ETAPA

Para inicio do desenvolvimento do projeto foi adquirido um módulo RFID-RC522 e suas *tags*; também um *Arduino UNO* e *jumpers*.



Figura 1: *Arduino UNO*



Figura 2: Módulo RFID RC522

5.1.1. Primeiras impressões *Arduino* e *RFID*

Com a montagem do *Arduino* e o módulo *RFID* começaram-se os testes de funcionalidade do sistema.

O leitor *RFID* é um módulo bem simples de se usar, porém, é necessária a instalação de uma biblioteca própria do modelo RC522 para que a programação na IDE do *Arduino* possa funcionar com perfeição.

Foram escritos os comandos de programação para que, quando a *tag* fosse lida pelo leitor *RFID* o programa retornasse o número de identificação da *tag* com o nome de uma pessoa cadastrada no programa. De forma segura ocorreu tudo como o esperado e o sistema funcionou perfeitamente. A programação pode ser vista no ANEXO A.

5.1.2. Conexão com a *internet* através do Módulo *Ethernet*

A ideia era conectar o projeto a *internet*. Pelos estudos realizados com o *Arduino* foi encontrada uma forma de conectar-se a *internet* com o módulo *Ethernet* W5100.

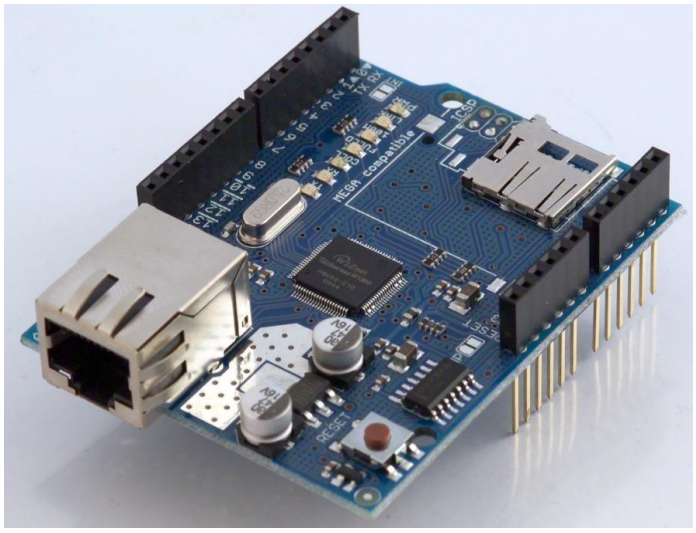


Figura 3: Módulo *Ethernet*

Este módulo possibilita a conexão do *Arduino* com a *internet* por meio do cabo *Ethernet* e roteador. Foram feitos os testes e também se obteve sucesso no que se diz respeito o objetivo de conectar-se a *internet*. A programação pode ser vista no ANEXO B.

5.1.3. Substituição do módulo *Ethernet*

A experiência com o módulo *Ethernet* de fato foi um sucesso, porém, foi constatado que não era a melhor maneira de se conectar a *internet* para este projeto. Um dos motivos para esta afirmação é de que seria necessária a conexão do módulo *Ethernet* há um roteador, e já que o objetivo do projeto é um protótipo em sala de aula, não seria possível a conexão do mesmo no roteador.

Após mais pesquisas, foi encontrado outro meio de conectar-se a *internet* com o *Arduino*, e seria através de uma adaptação com outro microcontrolador, o ESP8266, que é muito utilizado em aplicações *IoT*.

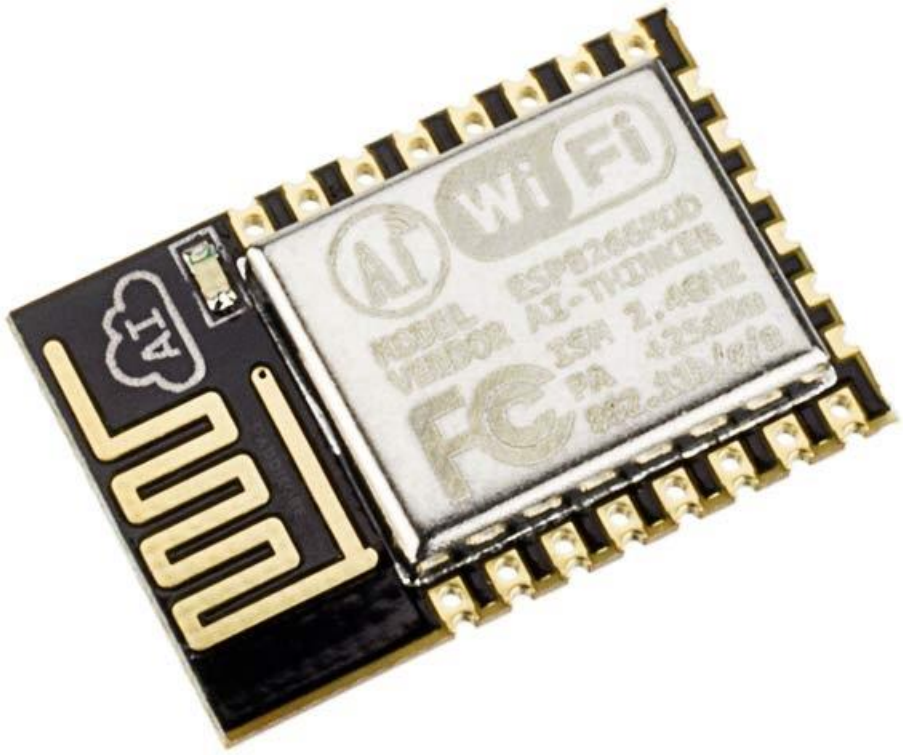


Figura 4: ESP8266

O ESP8266 seria perfeito já que se conecta a *internet* sem precisar de qualquer conexão via cabo. Concluído este processo, foi feita a substituição do módulo *Ethernet* pela adaptação do ESP8266 ao *Arduino*.

5.1.4. Substituição do microcontrolador

Para dar prosseguimento ao projeto, foi feito um estudo profundo sobre o ESP8266. Com este estudo, foi possível constatar que o ESP8266 tinha muitas variações de placas de desenvolvimento. Com base nesses estudos das diferentes placas, foi escolhido o *NodeMCU* por possuir mais portas *GPIO* (Portas programáveis de entrada e saída de dados) e não ser necessário o uso de um conversor de dados *FTDI* para programá-la. Além de substituir a função do módulo Ethernet, ele poderia também substituir a função do *Arduino UNO*.



Figura 5: *NodeMCU*

O *NodeMCU* utiliza uma linguagem de programação própria, porém, pode ser adaptado a IDE do *Arduino* através de bibliotecas e *driver*, assim podendo ser programado na linguagem C++ na própria IDE do *Arduino* facilitando muito o processo.

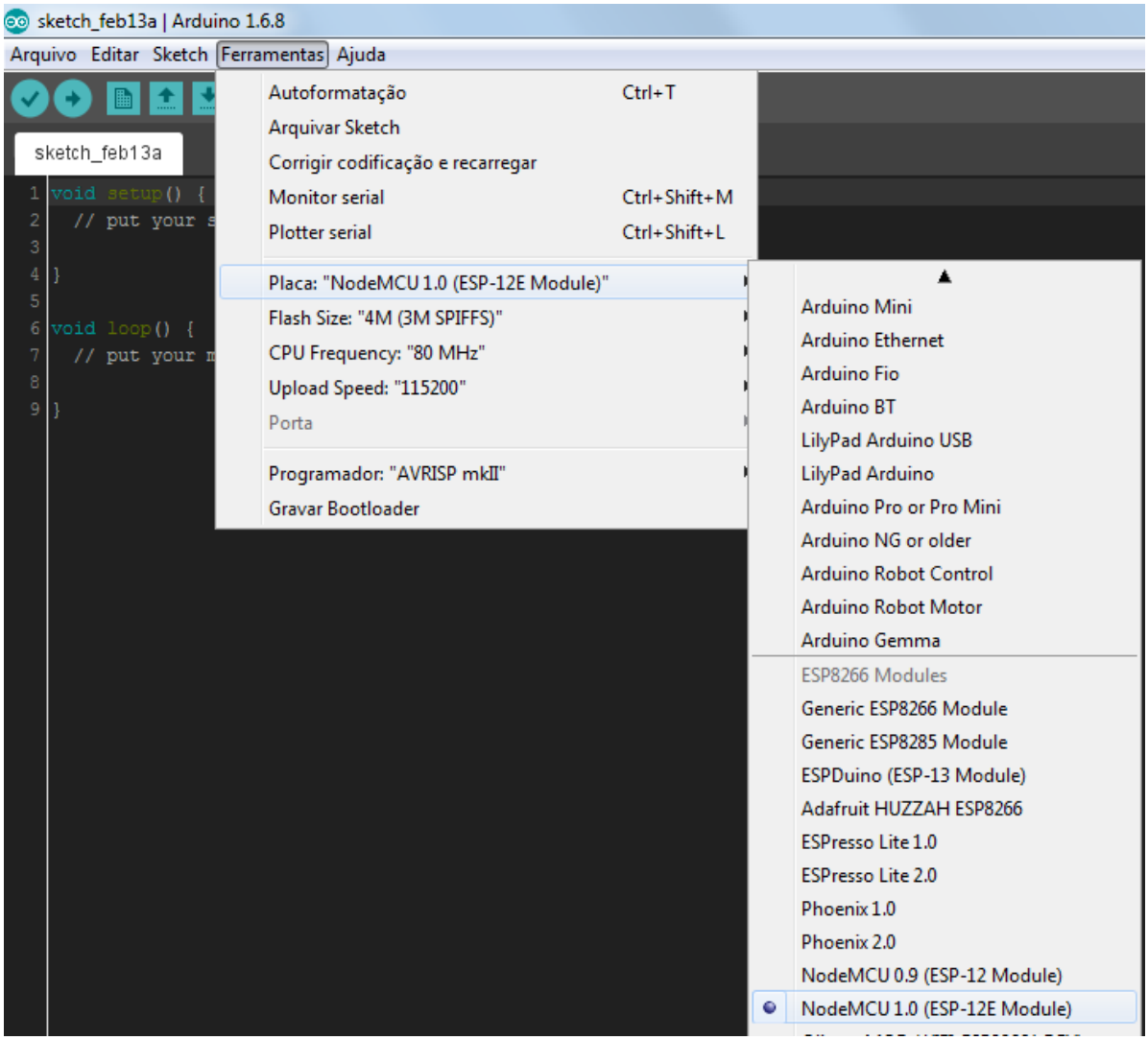


Figura 6: *Driver* ESP8266

Algumas vantagens do *NodeMCU* em relação ao *Arduino* UNO para este projeto:

- Conecta-se a *internet* via *wireless* (sem a necessidade de conexão de cabos)
- Dispensa qualquer tipo de *Shield* (módulos de *internet*)
- É muito menor que o *Arduino UNO*, assim, podendo se encaixar melhor a estrutura do protótipo
- Baixo custo (por volta de $3 Dólares)

5.1.5. Testes com *NodeMCU* e *RFID* integrados

Com o *NodeMCU* e leitor *RFID* em mãos, foi realizado o procedimento de conexão com a *internet* e *RFID*. Foram feitos os testes e o programa funcionou igualmente ao teste usando o módulo *Ethernet* e *arduino,* só que desta vez, sem a necessidade de conexão via cabo com o roteador, como o planejado.

A montagem pode ser vista na imagem a seguir:

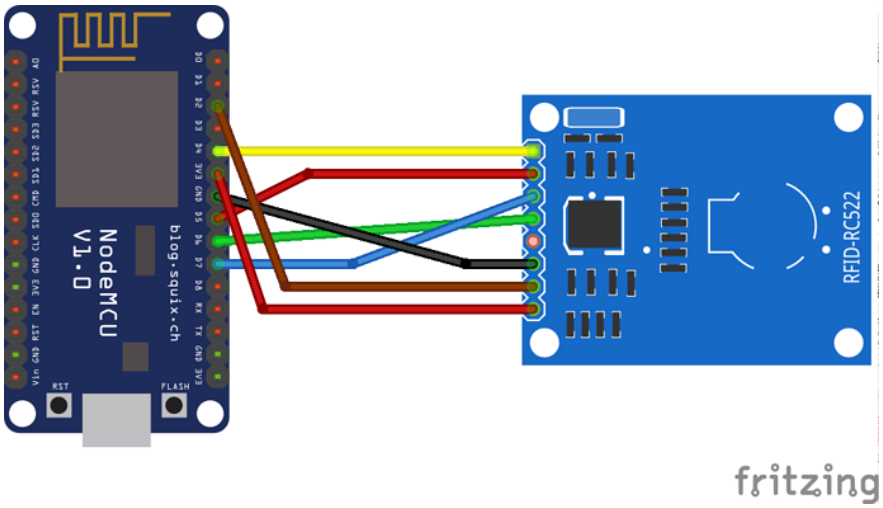


Figura 7: Conexão *RFID NodeMCU*

A programação pode ser vista no ANEXO C.

5.2. SEGUNDA ETAPA

A segunda etapa se inicia com a instalação do *WAMP Server (Windows Apache MySQL PHP)* que é um pacote de *softwares* que possibilita a criação de um "servidor" de banco de dados no próprio computador, afim de facilitar a criação de projetos sem precisar de uma hospedagem em nuvem. A princípio o *WAMP Server* foi utilizado para aprender a manipular suas ferramentas, dentre elas, o banco de dados. Depois, utilizar estes conhecimentos para aplicá-los no desenvolvimento do projeto em nuvem.

5.2.1. Primeiras impressões *WAMP Server*

O *WAMP Server* foi muito útil para a habituação de como utilizar um banco de dados. Ele funciona como uma espécie de servidor que já vem com um pacote para se trabalhar com um banco de dados no *Windows*. Com bastante conteúdo na *internet* foi relativamente fácil de aprender a manipulá-lo.

Graças a ele foi possível escrever códigos em *PHP* para o *NodeMCU* ter interação com o banco de dados. A programação do banco de dados (linguagem *SQL*) e arquivos em *PHP* será mostrada nos anexos que serão descritos conforme explicações.



Figura 8: *WAMP Server*

5.2.2. Criação do primeiro banco de dados

Foi criado um banco de dados pela plataforma *phpmyadmin*, que faz parte do pacote *MySQL*. O banco de dados recebeu o nome “rfidic”. Dentro deste banco de dados contém quatro tabelas nomeadas: acessos, insercao, prof e usuarios.

Dentro da tabela “acessos” contém as colunas: id, Tag e DataHora.

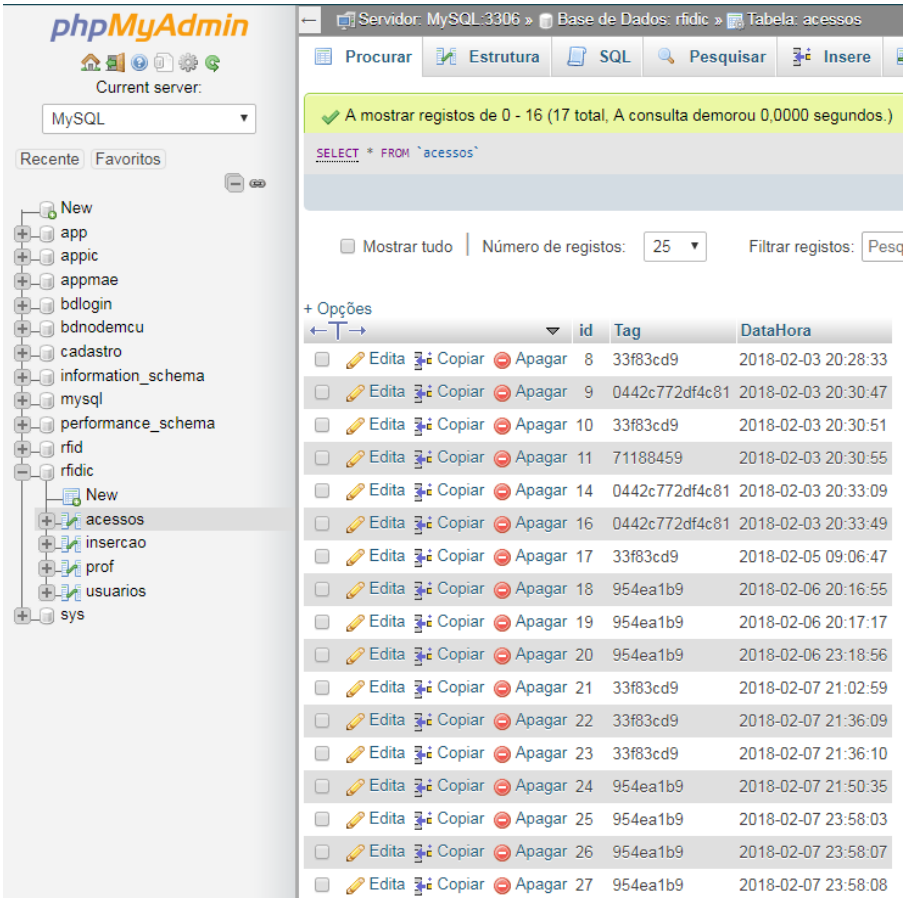


Figura 9: Acessos

Dentro da tabela de dados “insercao” contém as colunas: Tag e DataHora.

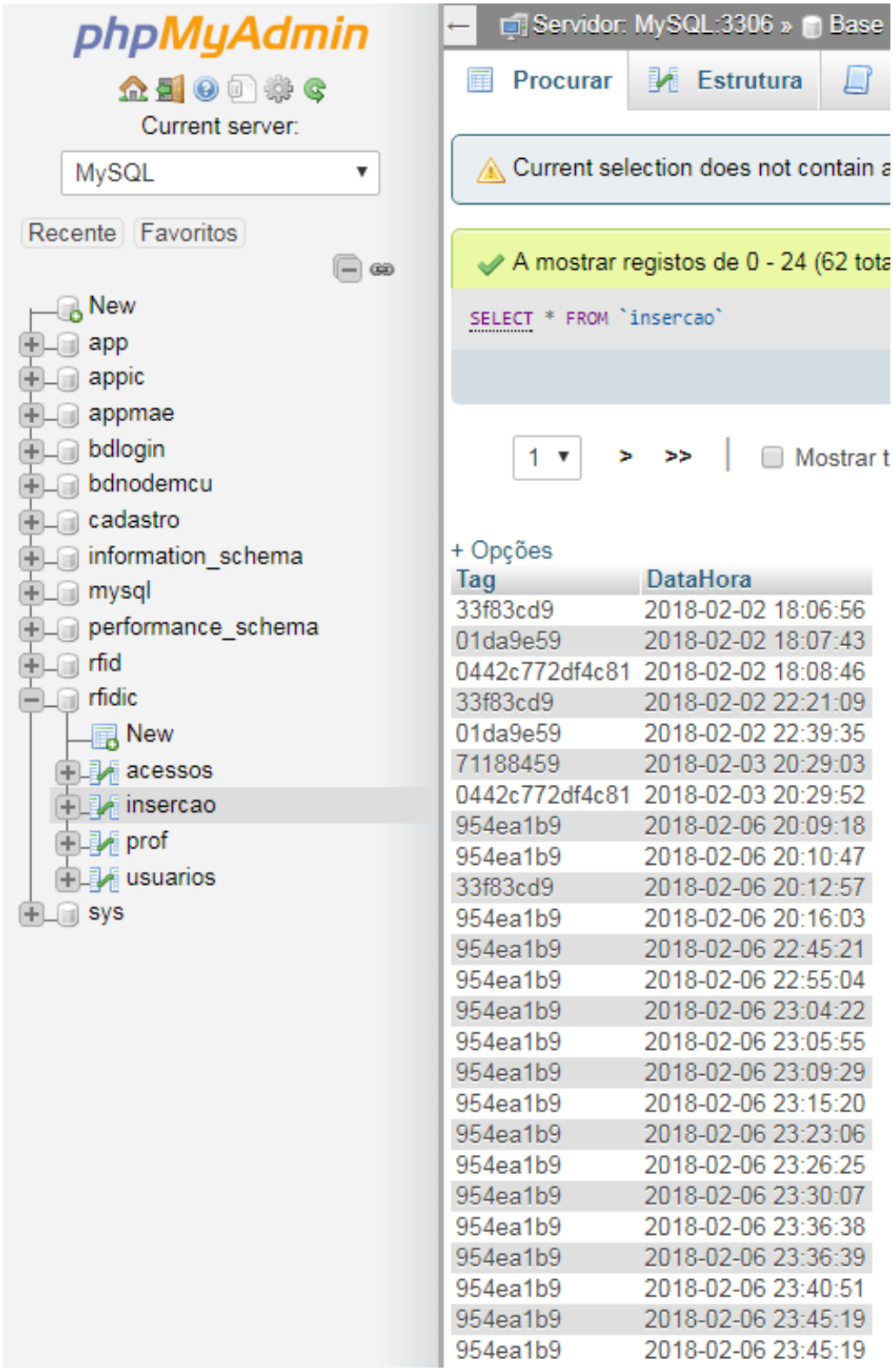


Figura 10: Inserção

Dentro da tabela de dados “prof” contém as colunas: id, nome, email e senha.

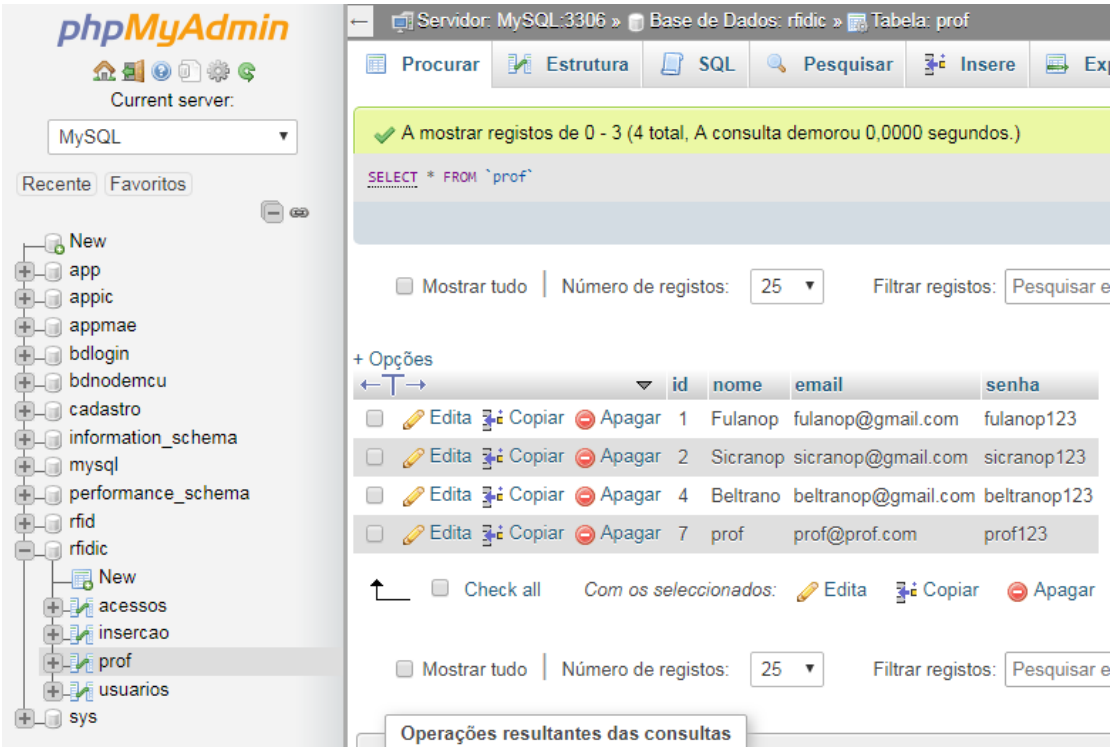


Figura 11: prof

Dentro da tabela de dados “usuarios” contém as colunas: Tag, Nome, RA, Turma, id.

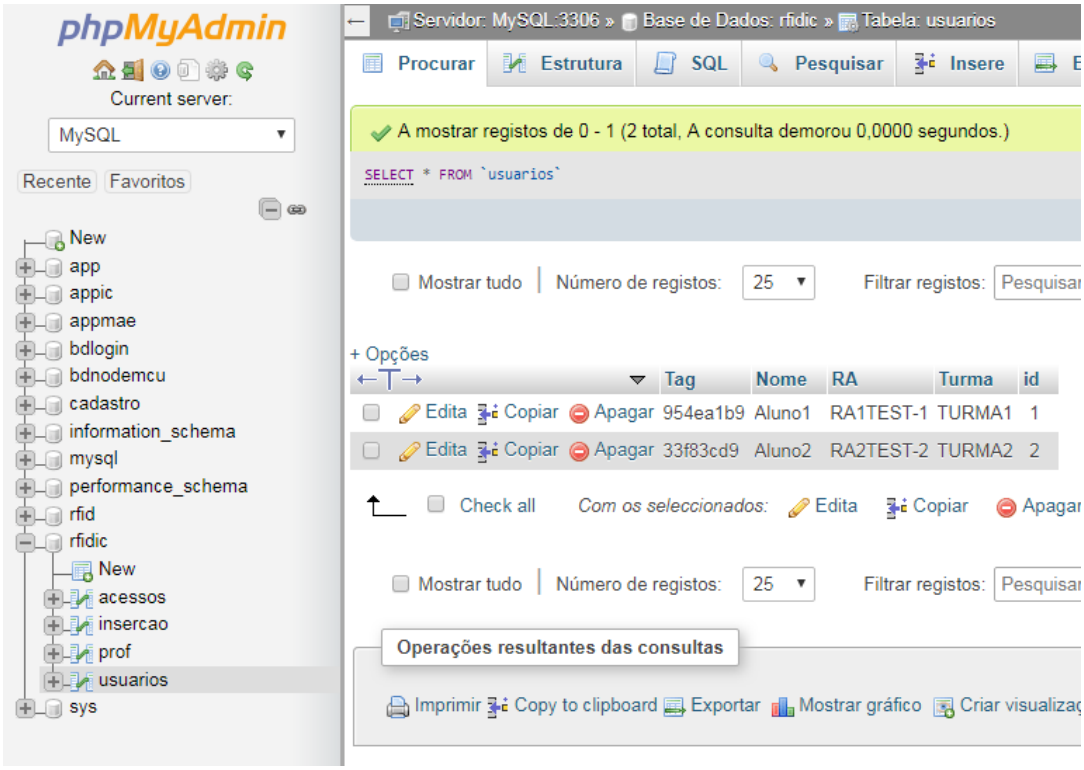


Figura 12: Usuarios

5.3. TERCEIRA ETAPA

Esse ponto foi o mais crítico do projeto, pois envolvia experiência no mínimo intermediaria em programação da parte da linguagem *PHP* e *HTML*. Para auxilio foi adquirido um livro do autor Sérgio de Oliveira intitulado “Internet das coisas com ESP8266, Arduino e Raspberry PI” que foi de extrema importância para esta etapa do projeto.

5.3.1. Conectividade *MySQL* e *NodeMCU* através do *PHP*

Na programação do *Arduino* que pode ser analisada no ANEXO D, nota-se que ele busca um endereço de um servidor e um arquivo específico. Neste caso ele busca o arquivo nomeado *consult.php*. Sua programação pode ser vista no ANEXO E.

5.3.1.1 Consulta de usuários

Ao passar a *tag* no leitor *RFID*, o *NodeMCU* acessa o endereço com o arquivo *consult.php* e faz uma consulta no banco de dados. Mais especificamente na tabela “usuarios”. Se a ID já existir no banco de dados, ele retorna a palavra “OK” para o arquivo, que por sua vez registra na tabela “acessos” a ID da *tag*, que por programação no próprio banco de dados insere a data e hora local.

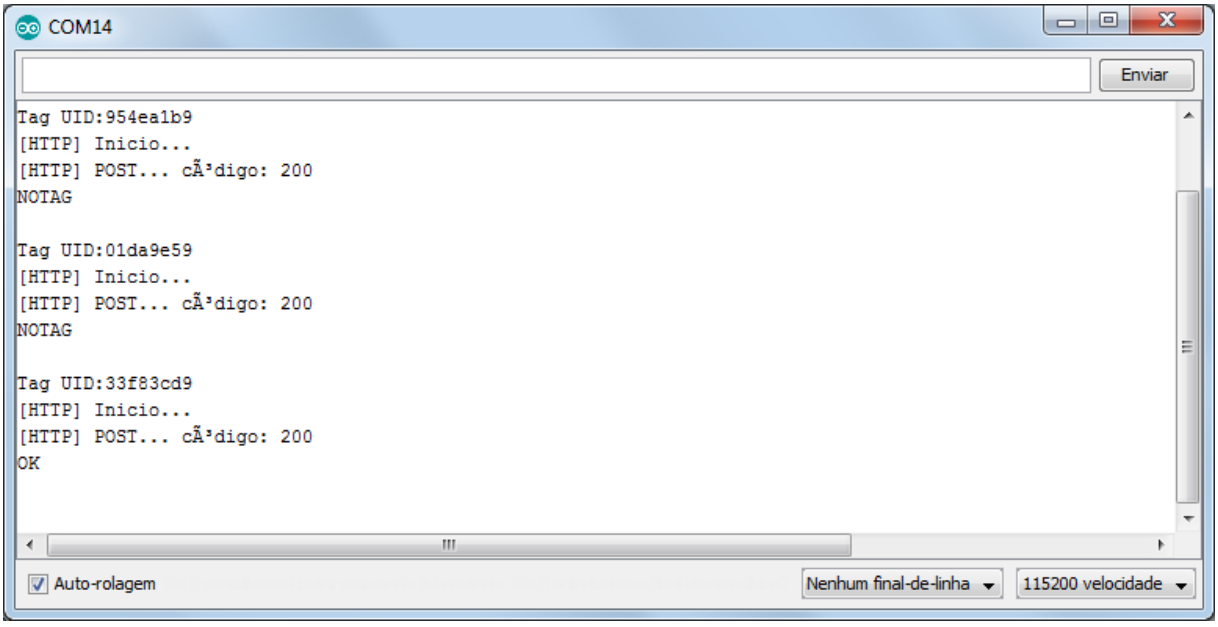


Figura 13: *OK*

Caso o ID da *tag* não exista no banco de dados, o *NodeMCU* retorna a palavra “NOTAG” para o arquivo, que por sua vez, insere o ID dessa *tag* na tabela “insercao”.

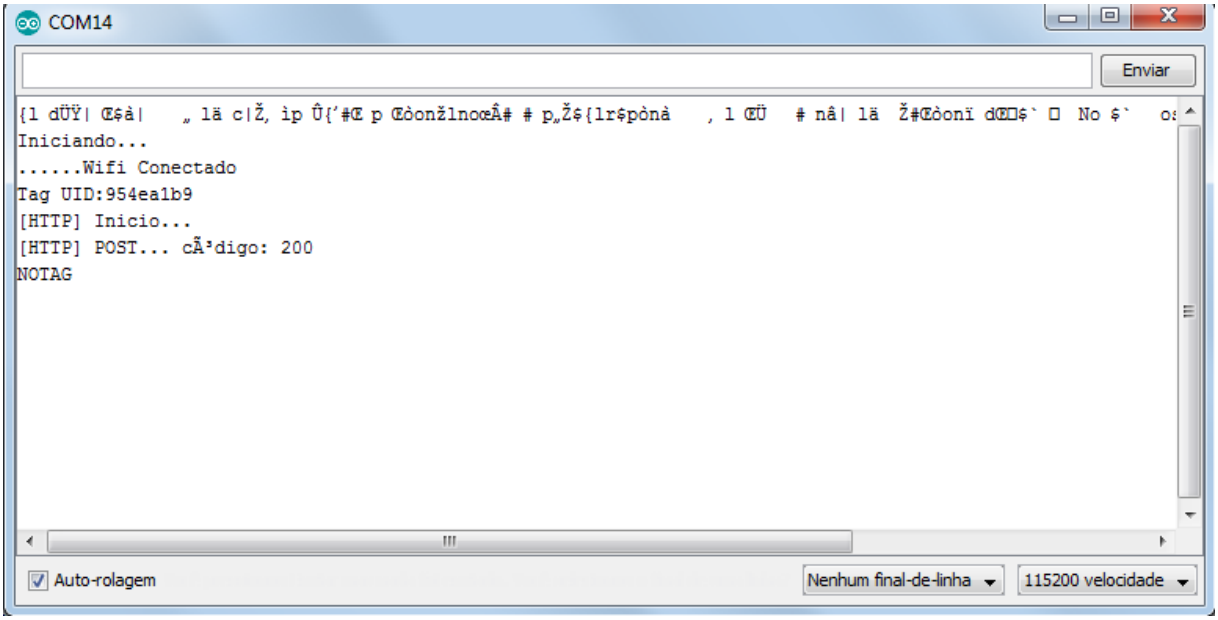


Figura 14: *NOTAG*

5.3.1.2 Cadastro de usuários

O arquivo *insere.php* consiste em registrar um usuário ainda não cadastrado através de um formulário, que criado dentro do próprio arquivo no formato *HTML* executa a ação como demonstrado na figura a seguir:



Figura 15: Formulário

Logo após a inserção da ID da *tag* na tabela “insercao”, o responsável pelo sistema de registro terá 3 minutos para fazer o cadastro, logo após esse período, o sistema exclui aquela *tag* e só será possível fazer o cadastro repetindo o processo. Essa programação pode ser vista no ANEXO F.

5.3.1.3 Controle de acesso

Para ter um controle de acesso dos usuários que passaram suas *tags* no leitor *RFID*, foi criado o arquivo *listacess.php*. Esse arquivo é responsável por mostrar o horário e a data que os usuários acessaram a sala de aula, funcionando como um controle de presença dos alunos. Pode ser visto por uma página na *internet* onde é possível selecionar entre qual data e hora os alunos acessaram o leitor. Essa página mostra o nome seguido do RA, Turma e data e hora do acesso. Para melhor ilustrar, veja a imagem a seguir:

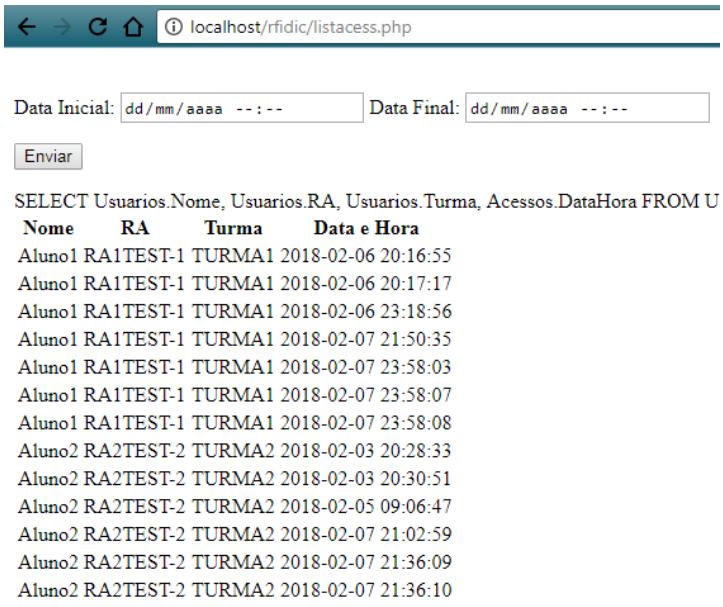


Figura 16: Lista de Acessos

Essa programação pode ser vista no ANEXO G.

5.3.1.4 Removendo usuários cadastrados

Ocasionalmente poderá haver a necessidade de se excluir algum usuário do banco de dados. Para isso foi criado o arquivo *remove.php*. Esse arquivo é responsável por excluir usuários que já estejam no sistema. Pode ser controlado em uma página de *internet*, onde basta apenas selecionar o nome do aluno e apertar o botão excluir.

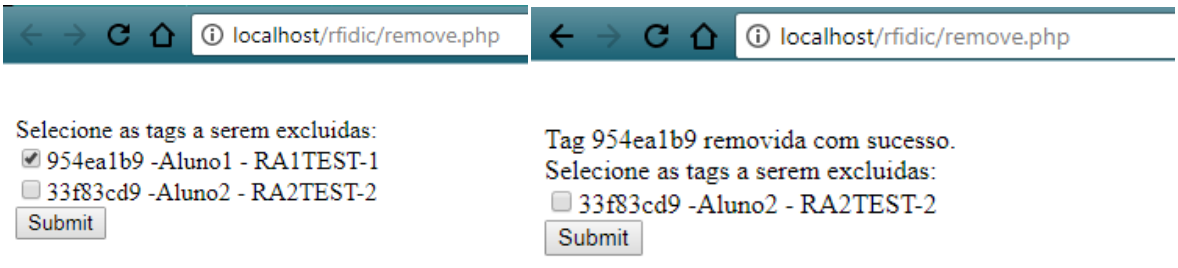


Figura 17: Excluindo

Sua programação pode ser vista no ANEXO H.

5.3.2. Teste do sistema geral

O sistema funcionou como o desejado, sendo possível no futuro se necessário fazer algumas alterações desejadas para a adaptação ao sistema da UNIP.

5.4. QUARTA ETAPA

Pensando em tornar o projeto ainda mais completo, foi criado um aplicativo para celulares *Android* que possa ter as mesmas funções das páginas citadas anteriormente.

5.4.1. Construção do aplicativo

Para a construção do aplicativo optou-se usar o *software MIT App Inventor 2*, pertencente a *Google*, desenvolvido pelo *MIT* (Instituto de Tecnologia de Massachusetts). O *App Inventor 2* foi escolhido pela sua maneira simples e clara de criar um aplicativo para o sistema operacional de *smartphones Android*.

A ideia aqui é ter dois aplicativos. O primeiro seria o “mestre” que seria usado por um coordenador, onde seria possível fazer cadastros, edições e exclusões de professores e alunos do banco de dados. O segundo seria para os professores, que poderiam utilizar das mesmas funções do aplicativo mestre, só que apenas para funções dos alunos.

5.4.1.1 Conectividade do aplicativo com o banco de dados

A conexão se dá por meio do endereço de um servidor. O aplicativo inicia sua conexão buscando o endereço que foi colocado na programação, encontrando assim, o arquivo de conexão com o banco de dados. Para essa conexão aplicativo e banco de dados, foi criado os arquivos *Db.class.php* e o *query.php*, sendo que o primeiro faz a conexão com o banco de dados, e o segundo faz a decodificação de pacotes no formato *Json* que é usado na programação do *App Inventor 2*. As programações dos dois arquivos podem ser vistas no ANEXO I e ANEXO J respectivamente.

Para realizar os testes, foi usado o servidor no próprio computador.

5.4.1.2 Inserindo professores no banco de dados e *login*

Como dito anteriormente, foi criada uma tabela chamada “prof” que se destina ao cadastro de professores por meio do aplicativo. Para cadastrar um professor, basta selecionar o botão “Crie uma conta” na tela de *login*, e fazer o cadastro. Preenchendo com nome, email e senha os campos de cadastro, o sistema verifica através do arquivo conexão.*php* e *registrar.php* se já há algum email idêntico, caso haja, ele retorna a informação de que já existe o email. As programações dos dois arquivos podem ser vistas no ANEXO K e ANEXO L respectivamente.

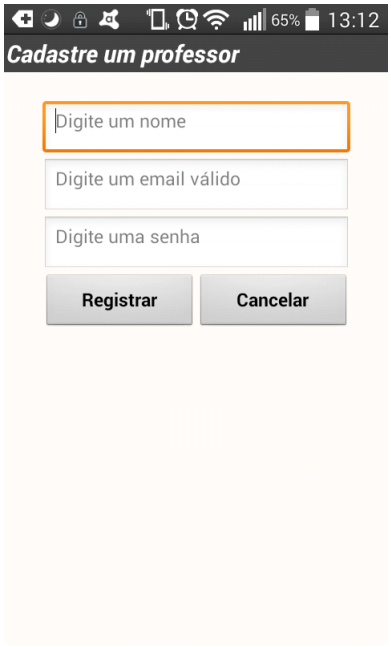


Figura 18: Cadastrando professor

Para o *login*, o aplicativo acessa o arquivo *login.php* que verifica se o email e senha são iguais aos do banco de dados, caso contrário, o programa retorna uma informação de que não foi possível fazer o *login*.

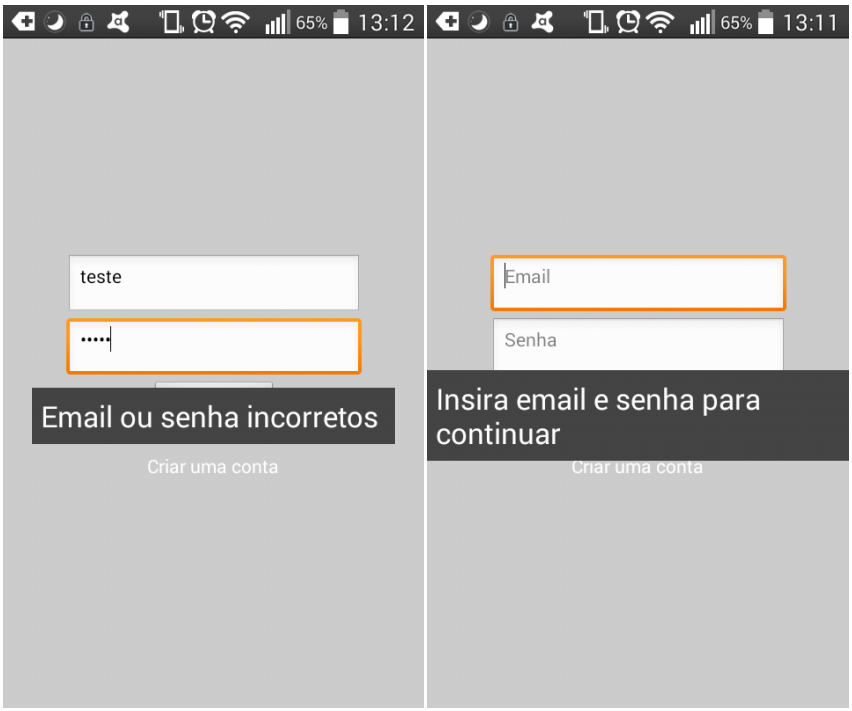


Figura 19: Tela inicial prova

Essa programação pode ser vista no ANEXO M.

5.4.1.3 Funções após *login*

Além de realizar as mesmas funções que já podem ser feitas no computador, também é possível realizar uma edição dos usuários que estão no banco de dados. Basta selecionar na tela principal do aplicativo a opção “EDITAR” e escrever o RA do Aluno para buscá-lo no banco de dados e assim fazer as modificações.

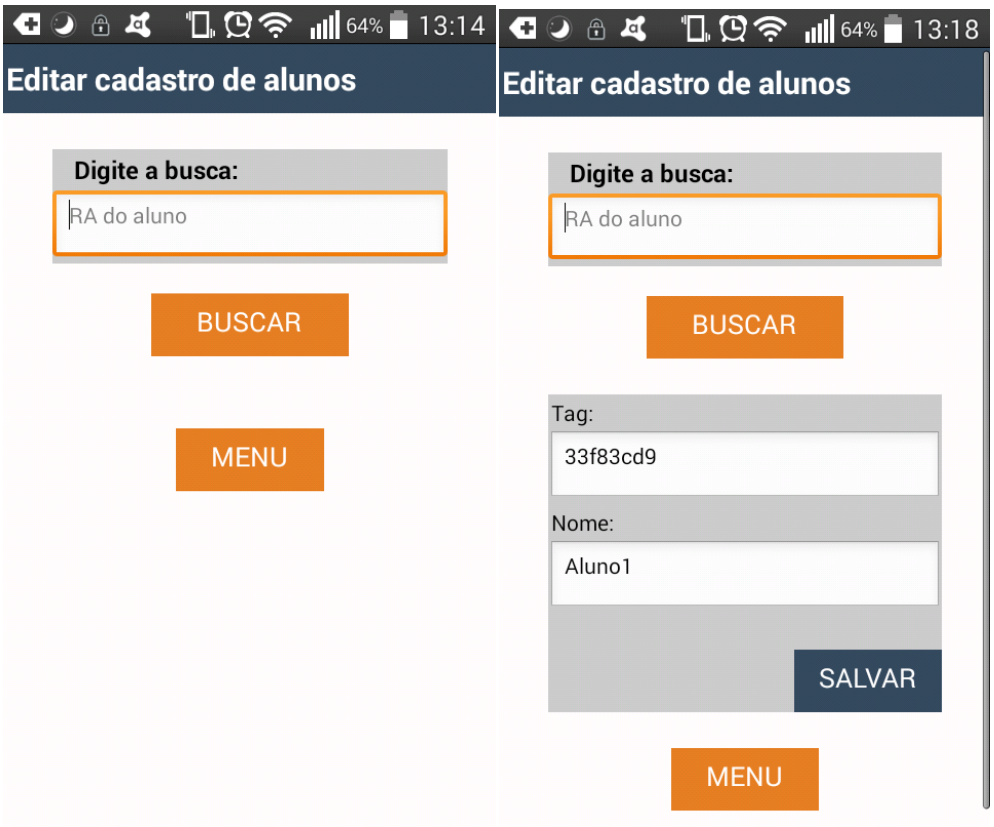


Figura 20: Editando

Também seria possível utilizar uma função “editar professor” para o aplicativo mestre.

5.4.2. Funcionamento

O aplicativo funciona perfeitamente e executa suas funções. A seguir, é possível analisar algumas das imagens ainda não ilustradas do aplicativo.

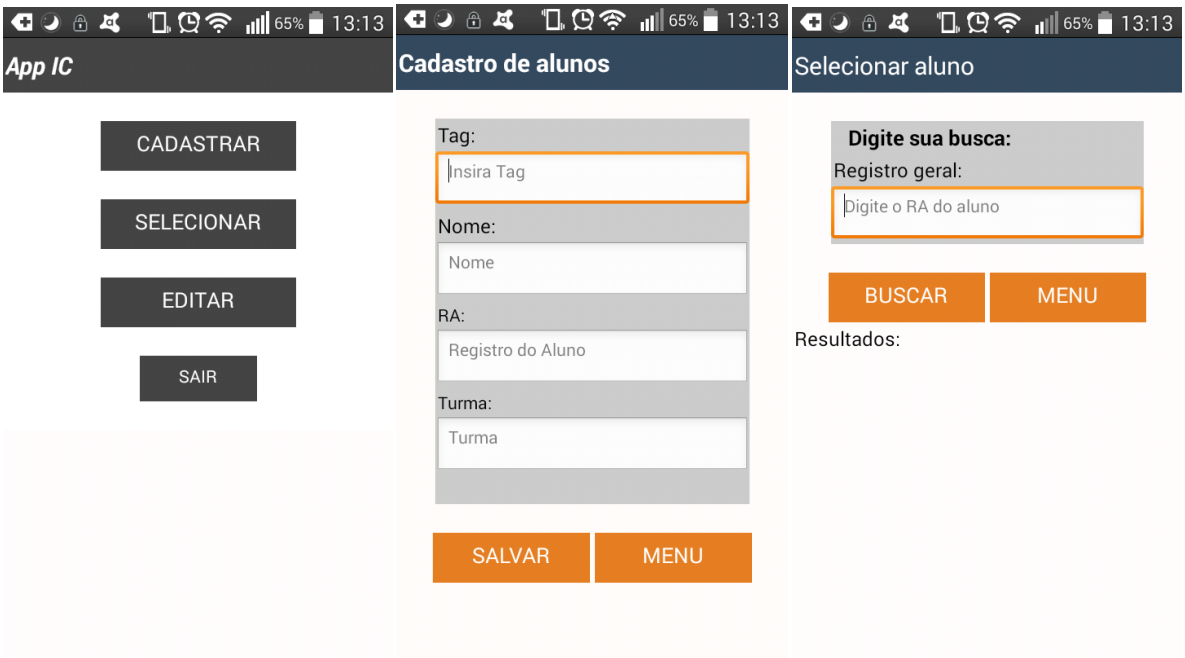


Figura 21: Imagens complementares do aplicativo

5.4.3. Instalação em sistema *Android*

A instalação do projeto do aplicativo no celular, pode ser facilmente executado a partir de um pacote de dados criado pelo *App Inventor 2*, ou até mesmo por *QR Code* que será normalmente instalado como um aplicativo comum.

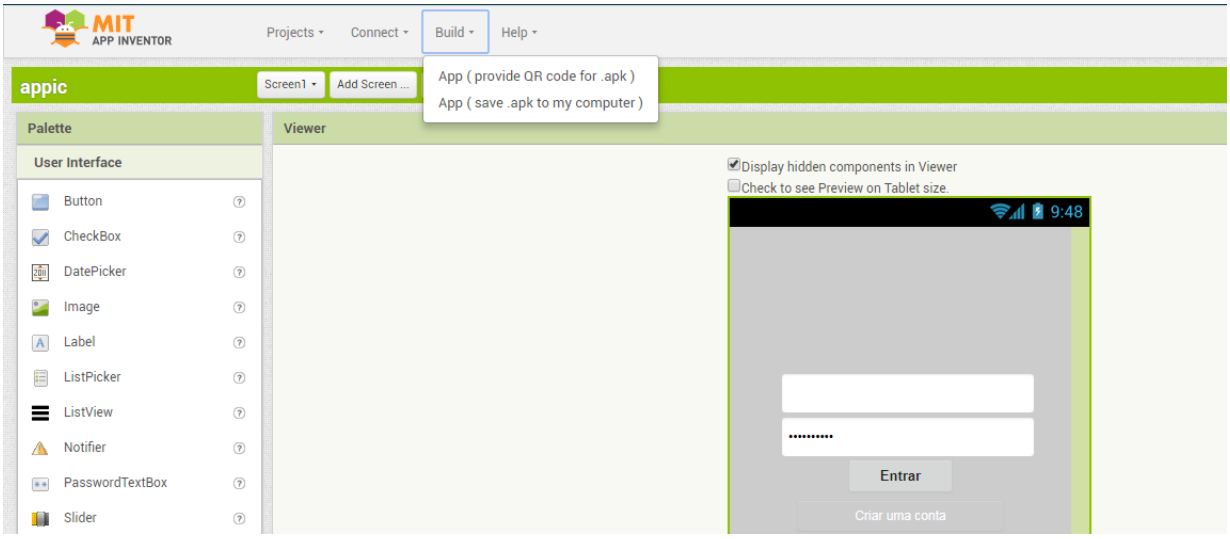


Figura 22: *Download* do app

5.5. QUINTA ETAPA

Um dos principais objetivos do projeto era colocar todo esse controle de acesso hospedado em um servidor na *internet* (nuvem), assim, tendo muito mais segurança com os dados da universidade.

5.5.1. Hospedagem em nuvem

Para hospedar todo o projeto na nuvem, foi feito um estudo sobre os servidores mais utilizados e seguros da *internet*, porém, que dispusesse de um suporte para banco de dados *MySQL* e hospedagem de arquivos. Após analises optou-se por utilizar o servidor gratuito do *site hostinger.com* nomeado *000webhost.*

5.5.1.1 Utilizando o *hostinger*

O primeiro passo no *site* foi fazer o cadastro e selecionar a opção gratuita do servidor, que nos dá a liberdade de utilizar até dois bancos de dados e fazer a criação de dois *sites*. Cada *site* tem um funcionamento de 23 horas por dia, após essas 23 horas, o *site* fica em modo de espera e não pode ser acessado. Isso se dá por utilizar uma conta gratuita, porém, é possível escolher o horário em que o *site* entrará em modo de espera.

5.5.1.2 Subindo os arquivos para a nuvem

Em relação ao banco de dados, foi bem simples transferir os dados do computador para o banco de dados do *hostinger*. Apenas foi necessário fazer uma exportação no banco de dados do computador em formato SQL, e logo em seguida no *site* do *hostinger* selecionar a aba “Gerenciar banco de dados”, e importar o arquivo para o banco de dados do *hostinger*.

Os arquivos de conexão no formato *php* também foram bem simples de serem transferidos para o *hostinger*. Foi necessário apenas selecionar a aba “Gerenciador de arquivos”, e em seguida “Fazer *upload* de arquivos agora” e selecionar a aba “carregar arquivos”. Pronto. Os arquivos já estão em nuvem.



Figura 23: Subindo arquivos

5.5.1.3 Executando o projeto em nuvem

Após mudar o endereço dos arquivos na programação do *NodeMCU* e no *App Inventor 2*, é necessário escrever na busca do navegador o endereço do *site* criado no *hostinger*. No caso desse projeto foi utilizado o link https://brunoiot.000webhostapp.com/.

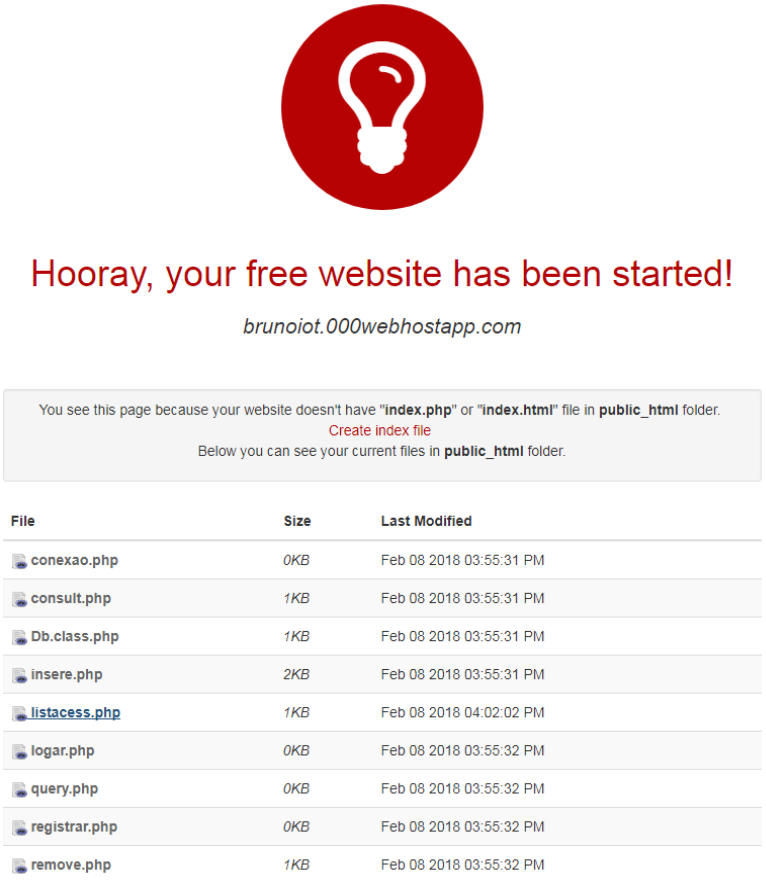


Figura 24: *Site*

Para ter acesso aos arquivos do projeto, basta apenas abrir o *site* e selecionar o nome do arquivo. Ou ainda, simplesmente adicionar o nome do arquivo no final do *link*. Exemplo: https://brunoiot.000webhostapp.com/listacess.php.

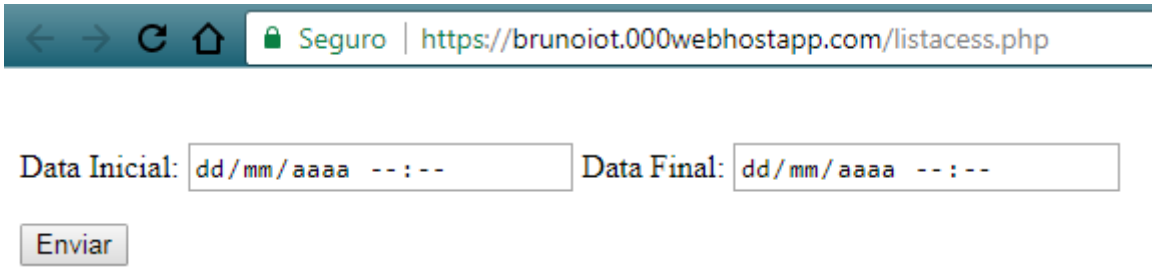


Figura 25: listacess no link

O Projeto em nuvem funcionou da mesma forma que no computador e pode ser utilizado o aplicativo da mesma maneira.

6. RESULTADOS

Como pode ser observado em todas as etapas descritas ao longo do desenvolvimento desse projeto, foi possível analisar que os resultados de ambas as etapas saíram como o planejado. Logo na primeira etapa, após a substituição da placa de controle, foi mais fácil obter um bom resultado no final do projeto, já que com o Arduino seria mais trabalhoso e seria necessário a aplicação de outros métodos para os resultados esperados. O módulo RFDI também se comportou muito bem com a placa *NodeMCU*, tal que foi necessário apensar instalar os drivers e bibliotecas na IDE do *Arduino* para que funcionasse com os mesmos algoritimos feitos para o *Arduino.*

O aplicativo para o sistema operacional *android* após muitos estudos e alguns testes, se mostrou completamente viável em nosso projeto. O aplicativo desenvolvido especialmente para os controles do banco de dados em nuvem foi um sucesso e atendeu as nossas expectativas como pode ser visto na etapa 4.

A criação do banco de dados e aplicação em nuvem, saíram como o pretendido. Uma vez aprendido como criar e programar o banco de dados, foi mais fácil integrar todo o projeto a nuvem. Como pode ser visto na etapa 5, apenas foi necessária a leitura do endereço no arquivo PHP. que já estava previamente conectado com o aplicativo.

7. DISCUSSÃO

Atualmente as tecnologias de identificação em banco de dados está se tornando popular no mundo todo, e esse projeto segue essa atualidade de internet das coisas. O projeto visa facilitar o método como funciona o registro de presença dos alunos nas universidades, e aqui prova-se que o sistema poderia ser facilmente implementado em uma sala de aula.

No inicio do projeto a ideia era usar um *Arduino* e o módulo *Ethernet* equipado com um *slot* cartão SD, porém, no decorrer do projeto foi observado que seria possível ter uma maior facilidade de desenvolvimento com a substituição desses itens pela placa *NodeMCU*. Além de maior facilidade, também foi possível utilizar menos recursos reduzindo muito os custos com materiais e componentes, e obtendo até mais desempenho do que na ideia inicial. Além disso foi possível observar pesquisando sobre a placa *NodeMCU* que, o material disponibilizado na internet e em livros é de muita qualidade e fácil entendimento, o que beneficiou muito este projeto. Também vale destacar que os custos financeiros com os materiais e componentes finais são muito baixos, sendo possível adquirir todos eles por menos de R$50,00.

Com todos os dados na nuvem, o aplicativo desenvolvido acrescentou ainda mais utilidade a este projeto, já que com ele é possível o controle de todos os dados do banco de dados estando em qualquer lugar do planeta, desde que se tenha acesso.

É bom ressaltar que apesar do *Arduino* e o *Wamp Server* não terem feito parte do projeto final, eles foram de suma importância para base de aprendizado para concluir o projeto. O *Wamp Server* ainda serve de base de estudos e da execução de projeto, porém, não seria possível colocá-lo em nuvem ainda, já que atua somente no computador.

8. CONCLUSÃO

Diversas tecnologias que fazem parte do universo da internet das coisas foram estudadas neste trabalho. Para que a aplicação pudesse atingir o melhor desempenho e praticidade, o comparativo de tecnologias se tornou fundamental, substituindo uma consolidada ferramenta, no caso o Arduino, pela NodeMCU, uma plataforma bastante recente.

A tecnologia de RFID embora muito utilizada em aplicações há muitos anos, permanece como uma excelente solução de automação de processos, permitindo controle de acesso e de usuários nas mais diversas aplicações.

Sem dúvidas a utilização de um banco de dados em nuvem, é o ponto de maior mudança nas arquiteturas dos sistemas de informação, que propões a internet das coisas. Os serviços de servidores tendem a migrar exclusivamente para este tipo de armazenamento com grande poder de compartilhamento.

Os aplicativos para plataforma Android criaram um mercado gigantesco de oportunidades para os desenvolvedores. Sinônimo de inovação, os aplicativos são hoje o modo mais fácil e usual de se criar novas funcionalidades para o dia a dia. São o pilar da internet das coisas, proporcionando integração das atividades de rotina e qualquer equipamento, à rede mundial de computadores.

Ao reunir todos estes recursos, este trabalho proporcionou um método inédito e inovador de controle de frequência dos alunos da UNIP nas aulas. Proporciona ao docente uma ferramenta de controle através de relatórios e consultas em tempo real através de aplicativo para smartphone.

Como sugestão de trabalhos futuros, pode-se estender a aplicação para outros sistemas operacionais, como IOS, e a integração das informações de frequência com os sistemas utilizados pela UNIP no controle oficial de frequência.

9. REFERÊNCIAS

[1] ATZORI, L.; IERA, A.; MORABITO, G. The Internet of Things: A survey. Computer Networks, v.54, n. 15, p. 2787–2805, out. 2010.

[2] ASHTON, Kevin. That 'Internet of Things' thing. RFID Journal, 2009. Disponível em <http://www.rfidjournal.com/article/view/4986> Acesso em: 26/03/2017.

[3] FLEISCH, E. WHAT IS THE INTERNET OF THINGS? AN ECONOMIC PERSPECTIVE. Economics, Management and Financial Markets, v. 5, n. 2, p. 125–157, 2010.

[4] KRANENBURG, R.; ANZELMO, E.; BASSI, A.; CAPRIO, D.; DODSON, S.; RATTO, M. The Internet of Things. 1st Berlin Symposium on the Internet and Society. Outubro de 2011. Disponível em <http://www.theinternetofthings.eu/sites/default/files/%5Busername%5D/The%20Internet%20of%20Things.pdf> Acesso em: 26/03/2017.

[5] MANIYKA, J. et al. Disruptive technologies: advances that will transform life, business, and the globaleconomy. Mckinsey Global Institute, May 2013. Disponível em <http://www.mckinsey.com/businessfunctions/digital-mckinsey/our-insights/disruptive-technologies> Acesso em: 26/03/2016.

[6] SINGER, T. SIMPÓSIO EM TECNOLOGIAS DIGITAIS E SOCIABILIDADE. 2012, Salvador. Tudo conectado: Conceitos e representações da internet das coisas. Salvador: Simsocial, 2012.

[7] TERRA, J. C. The internet of things. Terra FórumGlobant, Feb. 2013. Disponível em: <http://pt.slideshare.net/jcterra/internetofthings> Acesso em: 26/03/2016.

[8] YOO, Y.; HENFRIDSSON, O.; LYYTINEN, K. Research Commentary —The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. Information Systems Research, v. 21, n. 4, p. 724–735, dez. 2010.

ANEXO A

```
#include <SPI.h>

#include <MFRC522.h>

//Pinos de definição

#define SS_PIN 10

#define RST_PIN 9

MFRC522 mfrc522(SS_PIN, RST_PIN);

//Aqui colocamos as Tags que desejamos autorizar

//Lembrando que para acrescentar IDs deve alterar o numero de tagsCadastradas, e para cada uma cadastrada criar uma nova "funçao" nos acessos liberados e lembrar de alterar o numero do ID

String tagsCadastrada[3] ={"01 DA 9E 59","65 E6 94 89","33 F8 3C D9"};

// Num ha alterar na funçao:    0       ,      1      ,      2

void setup()

{

  // Inicia a serial

  Serial.begin(9600);

  //inicia a conexão SPI

  SPI.begin();

  // Inicia MFRC522

  mfrc522.PCD_Init();

   mensageminicial();

}

 void loop()

{

  //Aguarda proximidade do cartão / tag

  if ( ! mfrc522.PICC_IsNewCardPresent())

  {
```

```
  return;
}
 //Seleciona o cartão / tag
if ( ! mfrc522.PICC_ReadCardSerial())
{
  return;
}
//Mostra UID na serial
Serial.print("UID da tag :");
//String que armazenará o UID do cartão
String conteudo= "";
//variavel que coletará os bytes lidos
byte letra;
//dentro deste FOR, armazena byte a byte e
//concatena na variavel conteudo
for (byte i = 0; i < mfrc522.uid.size; i++)
{
   Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
   Serial.print(mfrc522.uid.uidByte[i], HEX);
   conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
   conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
}
//booleana que valida a tag lida
//como autorizada ou não
boolean tagVerificada = false;
//coloca o valor da variavel conteudo toda em Maisculo
conteudo.toUpperCase();
```

```
  Serial.println();
  //verifica se a tag lida coincide com uma das tags
  //cadastada no array
 for(int indice =0; indice < sizeof(tagsCadastrada); indice++){
// PARAMETROS PARA IDs CADASTRADAS
  //if (conteudo.substring(1) == "24 B4 8B 19") //UID 1
   if (conteudo.substring(1) == tagsCadastrada[0])
  {
   //Se sim imprime na Serial
    Serial.println("Ola Aluno!");
    Serial.println("Acesso Liberado!");
    //tag encontrada
    tagVerificada = true;
    delay(3000);
    //Incia o processo novamente
    mensageminicial();
    //limpa a String conteudo para fazer nova leitura
    conteudo= "";
  }
  //if (conteudo.substring(1) == "55 FC F9 0D") //UID 2
  if (conteudo.substring(1) == tagsCadastrada[1])
  {
    //Se sim imprime na Serial
    Serial.println("Bem vindo Aluno2!");
    Serial.println("Acesso Liberado!");
    //tag encontrada
    tagVerificada = true;
```

```
    delay(3000);
    //Incia o processo novamente
    mensageminicial();
    //limpa a String conteudo para fazer nova leitura
    conteudo= "";
  }
    //if (conteudo.substring(1) == "55 FC F9 0D") //UID 3
    if (conteudo.substring(1) == tagsCadastrada[2])
  {
    //Se sim imprime na Serial
    Serial.println("Olá Aluno3! ");
    Serial.println("Acesso Liberado!");
    //tag encontrada
    tagVerificada = true;
    delay(3000);
    //Incia o processo novamente
    mensageminicial();
    //limpa a String conteudo para fazer nova leitura
    conteudo= "";
  }
 // PARAMETROS PARA ACESSO NEGADO
  //caso o conteudo não seja "" e a tag Verficada
  //esteja falsa.
  //Informa o usuario que não tem acesso
  if((tagVerificada == false)&&(conteudo != "")){
    //Se ambas condições forem verdadeira
    //imprime na Serial uma mensagem de negação ao usuario
```

```
    Serial.println("Usuario desconhecido não cadastrado!");
    Serial.println("Acesso Negado!!");
    delay(3000);
    //Incia o processo novamente
    mensageminicial();
    //limpa a String conteudo para fazer nova leitura
    conteudo= "";
  }
}
}
// FUNÇOES DE COMANDO
//Metodo Inicial que informa ao usuario o que deve ser feito
void mensageminicial()
{
  Serial.println("Aproxime o seu cartao/chaveiro no leitor.");
 }
```

ANEXO B

```
#include <SPI.h>

#include <Ethernet.h>


byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };

IPAddress ip(192, 168, 0, 120); // Endereço IP que a Ethernet Shield terá. Deve ser alterado para um endereço livre da sua rede.

EthernetServer server(80);      // Cria um servidor WEB


void setup() {

    Ethernet.begin(mac, ip);  // Inicializa a Ethernet Shield

    server.begin();           // Inicia esperando por requisições dos clientes (Browsers)

} // fim do setup


void loop() {


    EthernetClient client = server.available();  // Tenta pegar uma conexão com o cliente (Browser)


    if (client) {  // Existe um cliente em conexão ?


        boolean currentLineIsBlank = true;

        while (client.connected()) {

            if (client.available()) {   // os dados do cliente estão disponiveis para serem lidos

                char c = client.read(); // lê 1 byte (character) do cliente
```

```
            // a ultima linha da requisição do cliente é branca e termina com o
caractere \n
            // responde para o cliente apenas após a última linha recebida
            if (c == '\n' && currentLineIsBlank) {

               // envia o cabeçalho de uma resposta http padrão
               client.println("HTTP/1.1 200 OK");
               client.println("Content-Type: text/html");
               client.println("Connection: close");
               client.println();

               // ENVIA A PÁGINA WEB
               client.println("<!DOCTYPE html>");
               client.println("<html>");
               client.println("<head>");
               client.println("<title>P&aacute;gina do Arduino</title>");
               client.println("</head>");
               client.println("<body>");
               client.println("<h1>Ol&aacute; sendo enviado pelo Arduino!</h1>");
               client.println("<p>Esta &eacute; a minha primeira p&aacute;gina web no
Ethernet Shield</p>");
               client.println("<p>#####  #####</p>");
               client.println("</body>");
               client.println("</html>");
               break;
            }
            // toda linha de texto recebida do cliente termina com os caracteres \r\n
            if (c == '\n') {
```

```
                // ultimo caractere da linha do texto recebido
                // iniciando nova linha com o novo caractere lido
                currentLineIsBlank = true;
            }
            else if (c != '\r') {
                // um caractere de texto foi recebido do cliente
                currentLineIsBlank = false;
            }
          } // fim do if (client.available())
        } // fim do while (client.connected())

        delay(1);      // da um tempo para o WEB Browser receber o texto
        client.stop(); // termina a conexão

    } // fim do if (client)
} // fim do loop
```

ANEXO C

```
#include <ESP8266WiFi.h>

#include <SPI.h>

#include "MFRC522.h"

//Pinos de definição

#define SS_PIN 4

#define RST_PIN 5

const char *ssid =  "BBRS";     // mudar de acordo com a internet

const char *pass =  "Arakakis";

MFRC522 mfrc522(SS_PIN, RST_PIN);

//Aqui colocamos as Tags que desejamos autorizar

//Lembrando que para acrescentar IDs deve alterar o numero de tagsCadastradas, e para cada uma cadastrada criar uma nova "funçao" nos acessos liberados e lembrar de alterar o numero do ID

String tagsCadastrada[3] ={"01 DA 9E 59","65 E6 94 89","33 F8 3C D9"};

// Num ha alterar na funçao:    0       ,      1      ,      2


void setup()

{

  // Inicia a serial

  Serial.begin(9600);

  //inicia a conexão SPI

  delay(250);

  Serial.println(F("Booting...."));

  SPI.begin();

  // Inicia MFRC522

  mfrc522.PCD_Init();
```

```
  WiFi.begin(ssid, pass);

   int retries = 0;
  while ((WiFi.status() != WL_CONNECTED) && (retries < 10)) {
    retries++;
    delay(500);
    Serial.print(".");
  }
  if (WiFi.status() == WL_CONNECTED) {
    Serial.println(F("WiFi conectado"));
  }
  Serial.println(F("Tudo funcionando!"));

  mensageminicial();
}

void loop()
{
  //Aguarda proximidade do cartão / tag
  if ( ! mfrc522.PICC_IsNewCardPresent())
  {
    return;
  }
   //Seleciona o cartão / tag
  if ( ! mfrc522.PICC_ReadCardSerial())
  {
    return;
```

```
  }
  //Mostra UID na serial
  Serial.print("UID da tag :");
  //String que armazenará o UID do cartão
  String conteudo= "";
  //variavel que coletará os bytes lidos
  byte letra;
  //dentro deste FOR, armazena byte a byte e
  //concatena na variavel conteudo
  for (byte i = 0; i < mfrc522.uid.size; i++)
  {
     Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
     Serial.print(mfrc522.uid.uidByte[i], HEX);
     conteudo.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " "));
     conteudo.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  //booleana que valida a tag lida
  //como autorizada ou não
  boolean tagVerificada = false;
  //coloca o valor da variavel conteudo toda em Maisculo
  conteudo.toUpperCase();
  Serial.println();
  //verifica se a tag lida coincide com uma das tags
  //cadastada no array
 for(int indice =0; indice < sizeof(tagsCadastrada); indice++){

// PARAMETROS PARA IDs CADASTRADAS
```

```
//if (conteudo.substring(1) == "24 B4 8B 19") //UID 1
 if (conteudo.substring(1) == tagsCadastrada[0])
{
 //Se sim imprime na Serial
  Serial.println("Ola Fulaninho!");
  Serial.println("Acesso Liberado!");

  //tag encontrada
  tagVerificada = true;
  delay(3000);
  //Incia o processo novamente
  mensageminicial();
  //limpa a String conteudo para fazer nova leitura
  conteudo= "";
}

//if (conteudo.substring(1) == "55 FC F9 0D") //UID 2
if (conteudo.substring(1) == tagsCadastrada[1])
{

  //Se sim imprime na Serial
  Serial.println("Eae Beltrano!");
  Serial.println("Acesso Liberado!");

  //tag encontrada
  tagVerificada = true;
```

```
  delay(3000);
  //Incia o processo novamente
  mensageminicial();
  //limpa a String conteudo para fazer nova leitura
  conteudo= "";
 }

  //if (conteudo.substring(1) == "55 FC F9 0D") //UID 3
  if (conteudo.substring(1) == tagsCadastrada[2])
 {

  //Se sim imprime na Serial
  Serial.println("Olá Sicrano! ");
  Serial.println("Acesso Liberado!");

  //tag encontrada
  tagVerificada = true;
  delay(3000);
  //Incia o processo novamente
  mensageminicial();
  //limpa a String conteudo para fazer nova leitura
  conteudo= "";
 }


// PARAMETROS PARA ACESSO NEGADO
```

```
  //caso o conteudo não seja "" e a tag Verficada
  //esteja falsa.
  //Informa o usuario que não tem acesso
  if((tagVerificada == false)&&(conteudo != "")){

    //Se ambas condições forem verdadeira
    //imprime na Serial uma mensagem de negação ao usuario
    Serial.println("Usuario desconhecido não cadastrado!");
    Serial.println("Acesso Negado!!");



    delay(3000);
    //Incia o processo novamente
    mensageminicial();
    //limpa a String conteudo para fazer nova leitura
    conteudo= "";
  }
}

}
// FUNÇOES DE COMANDO
//Metodo Inicial que informa ao usuario o que deve ser feito
void mensageminicial()
{
  Serial.println("Aproxime o seu cartao/chaveiro no leitor.");
 }
```

ANEXO D

```
#include <MFRC522.h>
#include <SPI.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WiFi.h>
#define RST_PIN 2
#define SS_PIN 4

MFRC522 mfrc522(SS_PIN, RST_PIN); // cria acesso MFRC522

void setup() {
  Serial.begin(115200);
  delay(250);
  Serial.println();
  Serial.println("Iniciando...");

  SPI.begin(); // inicia serial SPI no leitor
  mfrc522.PCD_Init(); //inicia o leitos rfid
  pinMode(D0, OUTPUT); // seta o pino D0 como saida

  WiFi.begin("BBRS", "Arakakis"); // dados net
  int tentativas=0;
  while ((WiFi.status() != WL_CONNECTED) && tentativas++ < 20) {
    delay(500);
    Serial.print(".");
  }
  if (WiFi.status() == WL_CONNECTED)
```

```
    Serial.println("Wifi Conectado");
  }

  char * read_RFID(char *buffer) { // função para ler as tags
    // verifica se ha cartoes presentes e le um cartao
    if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial()) {
    buffer = dump_byte_array(mfrc522.uid.uidByte, mfrc522.uid.size, buffer);
    //mostra o cod da tag em hexadecimal
    Serial.printf("Tag UID:%s\n", buffer);
    return buffer;

  } else
  return NULL;
}

  // função para converter o cod rfid para char [] em hexadecimal
  char * dump_byte_array(byte *buffer, byte bufferSize, char * result) {
    for (byte i=0; i < bufferSize; i++) {
      char num[3];
      itoa(buffer[i], num, 16);
      if (buffer[i] <= 0xF) strcat(result, "0");
      strcat(result, num);
  }
  return result;

  }
```

```
void loop() {
  char code_RFID[20] = "";
  if (read_RFID(code_RFID)) { //verificar se o catao foi lido
    if(WiFi.status() == WL_CONNECTED) { //verifica wifi
      HTTPClient http;
      Serial.print("[HTTP] Inicio...\n");
      //configura url, cabeçalhos e monta o metodo POST com o cod da tag
      http.begin("http://brunoiot.000webhostapp.com/consult.php"); // aqui coloca o
servidor/site
      http.addHeader("Content-Type", "application/x-www-form-urlencoded");
      int httpCode = http.POST(String("Tag=") + code_RFID);
      //httpCode sera negativo em caso de erro
      if(httpCode>0) {
        //obteve resposta do servidor
        Serial.printf("[HTTP] POST... código: %d\n", httpCode);
        // achou o arquivo
        if(httpCode == HTTP_CODE_OK) { //conectou, atualiza os arquivos
          String payload = http.getString();
          Serial.println(payload);
          if (payload == "OK") {
            digitalWrite(D0, HIGH);
            delay(1000); //tempo suficiente para acionar o acesso
            digitalWrite(D0, LOW);
          }
        }
      } else
      Serial.printf("POST Erro: %s\n", http.errorToString(httpCode).c_str());
```

```
      http.end();
      delay(300); // tempo suficiente para acionar o acesso
    }
  }
}
```

ANEXO E

```
<?php
	//Cria conexao
	$conn = new mysqli("localhost", "id4359326_rfid", "30040910bia", "id4359326_iotdb");
	// verifica se foi criado com sucesso
	if ($conn->connect_error) {
		die("Falha na conexão: " . $conn->connect_error);
	}
	$sql = "SELECT * FROM usuarios WHERE Tag = '" . $_POST[Tag] . "'";
	$result = $conn->query($sql);
	if ((!$result) || ($result->num_rows == 0)) { //tag nao cadastrada
		echo "NOTAG\n";
		$result->free();
		$sql = "INSERT INTO insercao (Tag) VALUES ('". $_POST[Tag] . "')";
		$result = $conn->query($sql);

	} else { //tag existente, registra com sucesso
		echo "OK\n";
		$result->free();
		$sql = "INSERT INTO acessos (Tag) VALUES ('". $_POST[Tag] . "')";
		$result = $conn->query($sql);

	}

	$conn->close();
?>
```

ANEXO F

```
<HTML><BODY>


<?php

	//cria conexao

	$conn = new mysqli("localhost", "id4359326_rfid", "30040910bia", "id4359326_iotdb");

	// verifica se foi criado com sucesso

	if ($conn->connect_error) {

		die("Falha na conexão: " . $conn->connect_error);


	}


	if (!$_POST[Nome]) { //1 etapa: se nao tem os dados, verifica se uma nova tag for lida


		$sql = "SELECT * FROM insercao

			WHERE DataHora > NOW() - INTERVAL 3 MINUTE

			ORDER BY DataHora DESC LIMIT 1";

			// recupera a ultima tag enviada

		$result = $conn->query($sql);


		if ((!$result) || ($result->num_rows == 0))	{

			echo "Não foi identificada nenhuma tag. Aproxime uma tag no leitor <BR>";

			header("Refresh:3"); //tag nao lida... recarrega a pag a cada 3s ate a leitura
```

```
		} else { // 2 etapa: tag lida, monta um formulario
			$row = $result->fetch_assoc();
			echo "<form action=insere.php method=POST>
			<p>id: <input type=text name=id /></p>
			<p>Nome: <input type=text name=Nome /></p>
			<p>RA: <input type=text name=RA /></p>
			<p>Turma: <input type=text name=Turma /></p>
			<p>Tag: <input type=text name=Tag value=" . $row['Tag'] . "
/></p>
			<p><input type=submit /></p>
			</form>";
		}
	} else { //3 etapa: se os dados ja foram enviados, insere no banco

		$sql = "INSERT INTO usuarios VALUES('".$_POST['Tag']."','

	".$_POST[Nome]."','".$_POST[RA]."','".$_POST[Turma]."','".$_POST[id]."')";
		$result = $conn->query($sql);
		if ($result) {
			echo "Inserção realizada com sucesso<BR>";
			} else {
				echo "Ocorreu um erro<br>";
			}
		}

?>

</body></html>
```

ANEXO G

```
<html><body>
	<form action=listacess.php method=POST>
	<p>Data Inicial: <input type=datetime-local name=DataI />
		Data Final: <input type=datetime-local name=DataF /></p>
	<p><input type=submit /></p>
	</form>


<?php
	if ($_POST['DataI']) {
		$conn = new mysqli("localhost", "id4359326_rfid", "30040910bia", "id4359326_iotdb");
	// verifica conexao
	if ($conn->connect_error) {
		die("Falha na conexão: " . $conn->connect_error);


	}
	$sql = "SELECT usuarios.Nome, usuarios.RA, usuarios.Turma, acessos.DataHora
			FROM usuarios, acessos
			WHERE acessos.DataHora > '" . $_POST['DataI'] . "' AND
			acessos.DataHora < '" . $_POST['DataF'] . "' AND
			acessos.Tag = usuarios.Tag";
	echo $sql;
	$result = $conn->query($sql);
	if ($result->num_rows > 0) {
		// Imprime os dados de cada linha da tabela
		echo "<TABLE><tr><th>Nome<th>RA<th>Turma<th>Data e Hora";
```

```
                while($row = $result->fetch_assoc()) {
                        echo "<tr><td>" . $row["Nome"] . "<td>" . $row["RA"] . "<td>" .
$row["Turma"] . "<td>". $row["DataHora"] . "<BR>";


                }
        } else {
                echo "Resultado Vazio";
        }
        $conn->close();


        }
?>
</html></body>
```

ANEXO H

```
<html><body>

<?php

	//cria conexao

	$conn = new mysqli("localhost", "id4359326_rfid", "30040910bia", "id4359326_iotdb");

	// verifica se foi criado com sucesso

	if ($conn->connect_error) {

		die("Falha na conexão: " . $conn->connect_error);


	}


	foreach($_POST as $val) {

		$sql = "DELETE FROM usuarios WHERE Tag='" . $val . "'";

		$result = $conn->query($sql);

		if ($result) echo "Tag " . $val . " removida com sucesso. <br>";


	}


?>


Selecione as tags a serem excluidas:

<form action=remove.php method=post>


<?php

	$sql = "SELECT Tag,Nome,RA FROM usuarios";

	$result = $conn->query($sql);
```

```
        if ($result->num_rows > 0) {
                //imprime os dados de cada linha da tabela

                while($row = $result->fetch_assoc())
                        echo "<input type=checkbox name=check ".$row["Tag"]." value=
".
                        $row["Tag"].">".$row["Tag"]." -". $row["Nome"]." - ".
$row["RA"]."<br>";

        } else {
                echo "Resultado vazio";
        }
        $conn->close();
?>
<input type="submit" value="Submit">
</form>
</html></body>
```

ANEXO I

```
<?php

	/**
	*  conexão com o banco de dados usando mysqli
	*/
	class Db{

		private $con;

		// Faz a conexão com o banco assim que o objeto é criado
		function __construct(){
			$this->con = mysqli_connect('localhost', 'id4359326_rfid', '30040910bia', 'id4359326_iotdb');
			if (mysqli_connect_errno($this->con)) {
				echo "Problemas para conectar no banco. Verifique os dados!";
				die();
			}
		}

		// Faz a consulta sql
		public function query($sql){
			return mysqli_query($this->con, $sql);
		}
	}

?>
```

ANEXO J

```
<?php

        require "Db.class.php";

        $db = new Db();


        $sqlBusca = $_POST['sql'];// Pega o código sql enviado pelo aplicativo

        $resultado = $db->query($sqlBusca);


        // Transforma o resultado da consulta em um array associativo

        while ($array = mysqli_fetch_assoc($resultado)) {

                $dados[] = $array;

        }


        echo json_encode($dados);// Retorna o resultado da consulta no formato
JSON


?>
```

ANEXO K

```
<?php

		$host = "localhost";

		$usuario = "id4359326_rfid";

		$senha = "30040910bia";

		$banco = "id4359326_iotdb";


		$dbcon = new MySQLi("$host","$usuario","$senha","$banco");


		if($dbcon->connect_error) {

			echo "conexao_erro";

		}/* else {

			echo "conexao_ok";

		}*/




?>
```

ANEXO L

```
<?php


        include_once 'conexao.php';


        $nome = $_POST['nome'];
        $email = $_POST['email'];
        $senha = $_POST['senha'];
        $sql1 = $dbcon->query("SELECT * FROM prof WHERE email='$email'");
        if(mysqli_num_rows($sql1) > 0) {
                echo "email_erro";
        } else {
                //echo "vai continuar rsrs";
                $sql2 = $dbcon->query("INSERT INTO prof(nome,email,senha)
                VALUES('$nome','$email','$senha')");
                if($sql2) {
                        echo "registro_ok";
                } else {
                        echo "registro_erro";
                }
        }


?>
```

ANEXO M

```
<?php

        include_once 'conexao.php';

        $email=$_POST['email'];
        $senha=$_POST['senha'];

        $sql = $dbcon->query("SELECT * FROM prof WHERE email='$email' AND senha='$senha'");

        if(mysqli_num_rows($sql) > 0) {
            echo "login_ok,";
            while($dados = $sql->fetch_array()) {
                echo $dados['id'];
                echo ",";
                echo $dados['nome'];
            }
        } else {
            echo "login_erro";
        }

?>
```