Victor Israel Anchieta de Medeiros

Proposta de um Sistema para Controle de Acesso Seguro e Inteligente de Baixo Custo

Alegrete

2021

Victor Israel Anchieta de Medeiros

Proposta de um Sistema para Controle de Acesso Seguro e Inteligente de Baixo Custo

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Telecomunicações como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Telecomunicações.

Universidade Federal do Pampa – Unipampa

Engenharia de Telecomunicações

Orientador: Prof. Dr. Lucas Compassi Severo

Alegrete

2021

VICTOR ISRAEL ANCHIETA

PROPOSTA DE UM SISTEMA PARA GESTÃO E CONTROLE DE ACESSO SEGURO, INTELIGENTE E DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Telecomunicações da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Telecomunicações.

Trabalho de Conclusão de Curso defendido e aprovado em: 04 de outubro de 2021.

Banca examinadora:

_____

Prof. Dr. Lucas Compassi Severo

Orientador

UNIPAMPA

_____

Prof. Dr. Alessandro Gonçalves Girardi

UNIPAMPA

_____

Prof. Dr. Bruno Boessio Vizzotto

UNIPAMPA

Assinado eletronicamente por **ALESSANDRO GONCALVES GIRARDI**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 04/10/2021, às 11:59, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **BRUNO BOESSIO VIZZOTTO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 04/10/2021, às 12:02, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **LUCAS COMPASSI SEVERO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 04/10/2021, às 12:03, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0626065** e o código CRC **9A2B35BF**.

Universidade Federal do Pampa, Campus Alegrete
Av. Tiarajú, 810 – Bairro: Ibirapuitã – Alegrete – RS CEP: 97.546-550

Telefone: (55) 3422-8400

Este trabalho é dedicado a Deus, que esteve sempre ao meu lado protegendo e amparando.

AGRADECIMENTOS

Primeiramente, agradeço à minha família por todo apoio durante esta trajetória.

Aos meus padrinhos por além de tudo, serem minha inspiração diária.

Aos meus avós, por todo amor e auxílio dado.

Agradeço à minha namorada que esteve ao meu lado nos mais difíceis momentos, tornando isso possível.

Sou grato ao professor Lucas Severo pela orientação neste e em outros trabalhos. Aos professores Dimas Irion e Cristian Müller, pela confiança, paciência, e orientação ao longo de vários anos. E também aos professores e colegas do GAMA, que tiveram importante papel no meu crescimento.

"Você pode encarar um erro
como uma besteira a ser esquecida,
ou como um resultado que aponta
uma nova direção."
(Steve Jobs)

RESUMO

A constante evolução tecnológica contribui para que cada vez mais a ciência seja aplicada aos mais diversos elementos do cotidiano, sendo a segurança e comodidade algumas das áreas mais beneficiadas. Deste modo, este trabalho propõe o desenvolvimento de um sistema que garanta acesso seguro, inteligente, e gestão de acesso simplificado aos laboratórios da Universidade Federal do Pampa. Este processo foi possível através do desenvolvimento e união de equipamento circuital e um conjunto de programas. O circuito fez uso do microcontrolador ESP8266, o qual vem possibilitando a criação de inúmeros dispositivos inteligentes, dada capacidade de conexão com a internet de forma nativa e simples. Como estrutura de apoio ao circuito eletrônico, tem-se um servidor com API escrita em PHP, acesso apenas através de requisições HTTPS, e autenticação, de forma a manter um controle seguro. Já para armazenar as informações do sistema, optou-se por banco de dados MySQL, o qual além de ser responsável pelos dados, também mantém diversas funções e procedimentos com objetivo de dirimir quaisquer possibilidades de falha ou injeção SQL. Além destes, para evitar comprometer o sistema, foi desenvolvida uma aplicação web onde baseado no login de acesso, são disponibilizadas funções de gerência. Funções estas como inserção ou modificação de permissão de acesso para alunos, e visualização dos acessos realizados e barrados. Com o desenvolvimento deste sistema completo, foi possível obter uma alternativa tecnológica com potencial de escalabilidade propícia a implementação em larga escala em ambientes com grande tráfego, e acesso restrito.

Palavras-chaves: Internet das Coisas. Segurança. Controle de acesso.

ABSTRACT

The constant technological evolution makes possible the application of science to the most diverse elements of everyday life, being security and convenience the most benefited areas. Thus, this work proposes the development of a system that guarantees secure, smart access, and simplified access management to the laboratories of the Federal University of Pampa. This study became possible through the development and union of hardware and software. The ESP8266 microcontroller is used as the main processor, which has to enable the creation of numerous intelligent devices, given the ability to connect to the internet in a native and simple way. As a support structure for the electronic circuit, there is a server with a PHP API, accessed only through HTTPS requests and authentication, in order to maintain secure control. In order to store the system information, we opted for a MySQL database, which, in addition to being responsible for the data, also maintains several functions and procedures in order to resolve any possible failure or SQL injection. In addition to these, to avoid compromising the system, a web application in which management functions based on login access was developed. With the development of this complete system, it was possible to obtain a technological alternative with the potential for scalability, able to be implemented on a large scale in environments with high traffic and restricted access.

Key-words: Internet of Things. Security. Access control.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Uckelmann Mark Harrison (2011) define Internet das Coisas (IoT [1]) como um conceito no qual o mundo virtual da Tecnologia da Informação (TI) é perfeitamente integrado com o mundo real. Neste mundo, máquinas tornam-se mais inteligentes a partir do momento em que a informação é utilizada para executar tarefas de forma mais independente da ação humana. IoT pode ser resumido como a interconexão massiva entre computadores embarcados através da internet.

Nos últimos anos pôde-se presenciar diversos dispositivos de IoT se tornando populares, como *Smartwatches* (Relógios inteligentes), *Smart Jewellery* (como anéis inteligentes), *Fitness Trackers* (aparelhos para medição de dados durante exercícios), *Smart Clothing* (Roupas equipadas com tecnologia e sensores), *Smart Glasses* (Óculos de realidade virtual ou realidade aumentada), e até mesmo implantes eletrônicos. Todos esses equipamentos vestíveis (do inglês *wearable*), representam o quanto caminhamos para o futuro IoT.

Contudo, não apenas o uso pessoal é beneficiado com a internet das coisas. A área de segurança (tanto física, quanto digital) tem muito a crescer com a tecnologia, e quando analisado o quanto algumas situações pecam neste quesito, percebe-se a importância do mesmo e o quanto novos dispositivos o auxiliariam.

No Brasil, sistemas públicos são frequentemente alvos de ataques cibernéticos, como nos casos recentes:

- Manson (2021), Ventura (2021b): Em janeiro de 2021, o laboratório de cibersegurança do PSafe relatou vazamento de uma base de dados brasileira contendo informações detalhadas de 104 milhões de veículos, em torno de 40 milhões de empresas, e em torno de 220 milhões de pessoas (como nome, RG, CPF, data de nascimento, estado civil, vínculo familiar, endereço, salário, score de crédito serasa, entre outros).

- Albuquerque (2021): Site do STF sofreu um ataque de negação de serviço (DOS) e ficou offline nos dias 6, 7, e 8 de maio de 2021. A falha se deu pelo acesso em massa de bots para raspagem de dados públicos no site.

- Alvarenga (2021): Dataprev e INSS são acusados de vazamento de dados de aposentados, infringindo a Lei Geral de Proteção de Dados (LGPD). “O vazamento de dados implicou no endividamento de milhões de aposentados e pensionistas do INSS”.

[1] IoT - Internet of Things

- Ventura (2021a): Site do DataSUS (Ministério da Saúde) foi invadido e sofre *defacement*[2] para alertar sobre a insegurança do sistema que estaria violando a LGPD.

- Dercoles (2021): "O sistema de segurança do Detran do estado do Rio Grande do Sul permitiu o acesso através de uma API para diversos dados sem pedir qualquer autenticação". Isso resultou na exposição de 5.1 milhões de motoristas, com informações como CNH, Renach, placa, modelo e Renavam dos veículos, etc.

- Alecrim (2021): Em março de 2021, foi relatada uma falha no endereço "Juventude Web", um sistema mantido pelo MTE (Ministério do Trabalho). O problema em questão era uma API que retornava sem autenticação, dados como CPF, nome completo, data de nascimento, endereço, CEP, município, estado, e nome da mãe.

Assim como nestes casos a falta de segurança foi um problema. Na Universidade também podemos identificar algumas situações com pouca segurança que podem vir a se tornar eventos indesejados. Entre elas está o acesso aos laboratórios de ensino e pesquisa e demais salas.

No Campus Alegrete da Universidade Federal do Pampa o acesso as salas é gerido por um porteiro terceirizado utilizando um livro de registros. Neste livro ficam registradas as informações de horário de retirada da chave, nome do responsável, e horário de devolução da chave. O que ocorre é que após aberta a sala, não existe controle de quem adentra além do responsável, e isso representa um enorme risco para os bens materiais e intelectuais encontrados nestes ambientes.

Para resolver este problema é necessário melhorar a segurança no acesso aos laboratórios (principalmente), de forma a controlar com menor ação humana possível, mantendo um registro mais confiável.

1.1 Motivação

Analisando os produtos disponíveis no mercado, de marca nacional, temos tanto modelos mais simples quanto mais elaborados. Abaixo relaciona-se o produto, custo, funcionalidades, pontos positivos e negativos para cada um dos modelos analisados.

- Modelo FD 1000
  - Abertura por senha, sendo uma senha de administrador, quatro senhas de usuários, e quatro senhas temporárias para visitantes.
  - Possibilidade de no máximo nove senhas simultaneamente.

[2] Ataque que consiste em mudanças visuais no site, de forma a chamar atenção para algo.

  - Gerenciamento de senhas, com inclusão e exclusão de um usuário por vez.
  - Alimentação por pilhas.
  - Custo R$ 539,90.

- Modelo FR 201
  - Teclado *touch screen*.
  - Abertura por tag RFID (aproximação) ou senha. Sendo possível cadastrar quatro senhas, e 100 chaves de proximidade.
  - Gerenciamento de senhas e tags, com inclusão e exclusão de uma por vez.
  - Alimentação por pilhas.
  - Custo R$ 919,90.

- Modelo FR 320
  - Teclado *touch screen* e design elegante.
  - Abertura por tag RFID (aproximação) ou senha. Sendo possível cadastrar quatro senhas de quatro a doze dígitos e 100 chaves de proximidade.
  - Gerenciamento de senhas e tags, com inclusão e exclusão de um usuário por vez.
  - Sensor de incêndio.
  - Alimentação por pilhas.
  - Custo R$ 1.589,90.

- Modelo FR 220
  - Teclado *touch screen*, e sensor biométrico capacitivo.
  - Abertura por biometria (digital) ou senha. Sendo possível cadastrar quatro senhas de quatro a doze dígitos, e até 100 biometrias.
  - Gerenciamento de senhas e digitais, com inclusão e exclusão de um usuário por vez.
  - Função não perturbe.
  - Alimentação por pilhas.
  - Custo R$ 1.739,90.

Quando analisa-se as possibilidades, todas tem como característica comum a configuração complexa e manual, o que inviabiliza a aplicação no ambiente universitário, que tem um grande fluxo de pessoas. Além disso, para cada funcionário terceirizado novo

(limpeza por exemplo) precisaria cadastra-lo em cada sala com auxílio de quem detivesse a senha administrativa. Outro ponto importante é o registro de acessos, pois como o fluxo de acessos é intenso, em algumas situações torna-se útil verificar a lista de presentes em determinado momento.

Para contornar todos os obstáculos, adequar a solução ao caso apresentado, propõe-se então o desenvolvimento de um sistema, composto por *hardware* e *software*, com autoidentificação, onde haja uma verificação e validação online para permitir ou barrar os acessos. Dessa forma, obtendo-se um fluxo mais seguro, mas ao mesmo tempo simplificado, não dependendo de livros com verificação e controle manual.

Também como requisito, tem-se o processo de aplicação deste sistema, dado que deve ser de fácil implantação na Unipampa, onde seja possível com o menor esforço a inclusão ou exclusão de acessos. Para possibilitar essa simplicidade, é imprescindível o uso de rede, pois torna possível configuração rápida de diversas salas, assim como comunicação entre as partes que garantem a segurança que deseja-se.

De forma simplificada pode-se afirmar que seria mais interessante o armazenamento dos dados em circuito local, assim como as diversas alternativas existentes no mercado, porém este diferencial facilita tanto o controle, quanto manter registros de segurança, aumentando a confiabilidade do controle em si.

1.2 Objetivos

O objetivo principal deste trabalho é desenvolver um sistema para controle de acesso aos laboratórios do campus Alegrete da Unipampa. Este sistema deve ser seguro, inteligente e apresentar um baixo custo de implementação. Como objetivos específicos propões-se:

- Manter relatório de acessos de forma a possibilitar a verificação dos acessos realizados sempre que necessário. A análise dos dados contidos nos relatórios também possibilitam obter outras informações, tais como verificar horários de maior tráfego e os membros que mais utilizam o ambiente. Dados estes que podem ter inúmeros propósitos futuramente;

- Possibilitar administração das permissões de acesso de forma rápida e simplificada através de uma aplicação com interface gráfica de usuário. Desta maneira, a inclusão, atualização, ou exclusão de dados para acesso, torna-se dinâmica e pode ser feita diretamente pelo gestor da sala ou chefe de laboratórios;

- Tornar o acesso rápido e seguro, pois a autenticação é realizada com o menor atraso possível, e ao adentrar na sala, a porta é fechada, e aberta novamente apenas com

nova autorização de acesso. Desta forma, acompanhantes e pessoas sem permissão só poderão entrar sob autorização e responsabilidade de um membro autorizado;

- Desenvolver um sistema genérico e estruturado, de forma que possa ser aplicado em diversos ambientes, além do ambiente universitário.

1.3 Organização do trabalho

Este trabalho foi ordenado da seguinte forma:

O Capítulo 2 traz consigo as etapas de desenvolvimento do circuito eletrônico utilizado neste projeto, tais quais as motivações e detalhes responsáveis pelas escolhas tomadas.

O Capítulo 3 descreve os programas, códigos e a estrutura que permite toda comunicação entre os componentes do projeto. Nele é descrito também detalhes de segurança, criação e modelagem da base de dados e a relação entre os dados armazenados.

No Capítulo 4 são demonstrados os testes operacionais, teste prático, e quesitos que foram essenciais para a garantia de segurança que o projeto traz.

E o Capítulo 5 finaliza com as considerações finais a respeito do pesquisa, limitações que houveram e as recomendações baseadas na experiência obtida com o desenvolvimento desta trabalho.

2 IMPLEMENTAÇÃO EM NÍVEL DE HARDWARE

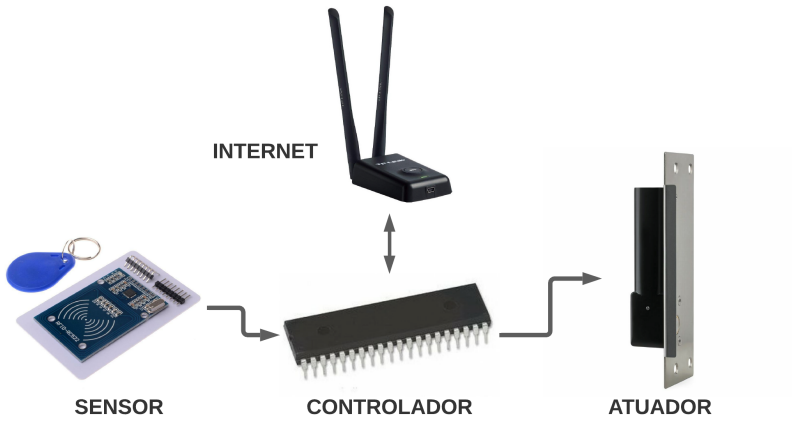
Hardware é o termo dado para a parte física de um computador ou sistema eletrônico. Esta é uma parte essencial para o sucesso deste trabalho, pois é a etapa responsável pela aquisição e tratamento de informações, comunicação entre os programas, e acionamento da ação desejada. Por estes motivos, deseja-se que o hardware contenha sensor, circuito de processamento, e atuador. Os itens do hardware proposto neste trabalho são apresentados na Figura 1.

O sensor tem como propósito possibilitar autoidentificação. É através do sensor que o sistema pode verificar e identificar quem deseja adentrar no local em questão. Este dispositivo pode ser também chamado de transdutor, pois é responsável pela conversão de informação física em sinais elétricos.

O circuito de processamento é utilizado para analisar o sinal do sensor, se comunicar com a internet, e controlar o atuador. O processamento e controle é tratado por um circuito integrado programável, que com a gravação de um *firmware* (programa que gerencia em baixo nível os comportamentos do circuito) possibilita a execução de ações como acionamentos elétricos, comunicação com outras máquinas, leitura de sensores, etc.

O atuador é necessário para dinamizar a ação de controle, liberando o acesso a sala quando necessário. O mesmo é um dispositivo responsável pela dinâmica física de um sistema, que tem como entrada a ação que deve ser executada, ou algum impulso que a signifique.

Figura 1 – Organização do hardware.



Fonte: o autor.

Nas próximas seções serão abordados com maiores detalhes os componentes do hardware desenvolvido.

2.1 Sensor de identificação

A tarefa de identificação, poderia ser feita de diversas maneiras, seja por sistemas biométricos, senhas, chaves de aproximação. Contudo, com a atual situação sanitária, por conta da pandemia do Covid-19, e a recomendação de evitar contato físico, utilizar tecnologia que necessite apenas aproximação (contact less [1]) se torna muito mais interessante.

Além disso, é muito mais prática a distribuição de tags RFID [2] (Radio-Frequency IDentification) do que requisitar cadastro de tantos alunos e funcionários, entre outros motivos, pela confidencialidade dos dados biométricos.

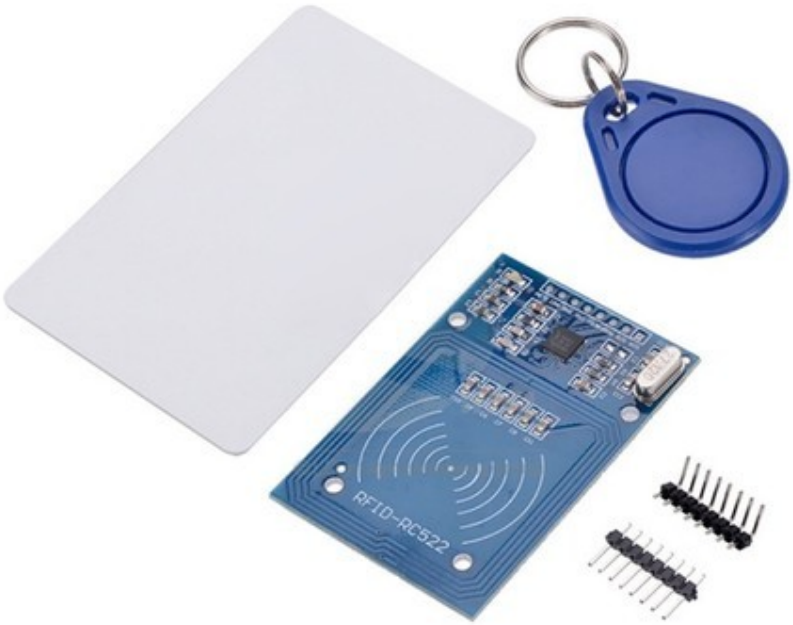
Ademais, o cadastro dos alunos e professores através de *tags* possibilita um dinamismo em outras situações, como acesso ao restaurante universitário (RU), evitando atrasos em filas, ou em chamadas durante as aulas, contabilizando presenças de forma automática, e garantindo que os alunos mantenham a presença necessária.

Desta forma, optou-se pelo sensor RFID da *NXP Semiconductors*, MFRC522 que é amplamente utilizado em diversos produtos, e tem uma gama de materiais de orientação (NXP SEMICONDUCTORS, 2016), o que facilitou muito durante o desenvolvimento. A

[1] Sem contato
[2] Identificação por Radio-Frequência

Figura 2 apresenta o leitor MFRC522, uma tag de aproximação em formato de cartão, e outra em formato de chaveiro.

Figura 2 – Leitor RFID MFRC522.



Fonte: Filipeflop.

Este sensor utiliza o padrão MIFARE Classic, o qual opera na frequência de 13.56 MHz para comunicação com as *tags*, conforme a norma ISO 14443. A norma em questão, trata-se de um padrão internacional, o qual visa regular e definir os padrões e protocolos de comunicação para circuitos de identificação por aproximação de até 10cm, e frequência de operação 13.56 MHz.

Trata-se de um sensor muito estável em termos de temperatura (-25ºC até +85ºC), que trabalha com 3.3V de alimentação, quando não está realizando leitura consome no máximo 10 μA (33 μW), e algumas dezenas de mA (podendo chegar ao máximo 100mA - 330 mW) quando realiza leituras e se comunica pela interface serial.

Para comunicação com o controlador, foi utilizado o padrão SPI, porém o circuito do sensor possibilita também uso por protocolo I2C.

Cada tag RFID tem um número único de identificação (UID [3]) formado por 32 b

[3] Unique ID

hexadecimal, e 32 kb de memória subdividido em 40 "setores". E é através desta memória que podemos identificar os diferentes cartões de cada indivíduo.

2.2 Controlador e Comunicação Wi-Fi

Esta seção tem por objetivo elucidar da escolha realizada em termos de controlador para o trabalho. Analisando as opções de dispositivo para garantir o controle temos: FPGA (Field Programmable Gate-Array [4]), microprocessador, SBC (Single Board Computer [5]), e microcontrolador.

Os FPGAs são circuitos muito aplicados em sistemas de comunicações, se tratam de placas robustas em termos de processamento, com capacidade de programação lógica (KOCAN, 2004). Normalmente são programadas em nível de Hardware, relacionando os barramentos e operações booleanas entre os dados processados. Tem um custo variável de acordo com o modelo e a aplicação, mas se comparadas com as opções SBC, e microcontrolador, são equipamentos de maior valor.

Já os Microprocessadores são os circuitos com frequência de clock mais alta da lista, e maior poder de processamento *single thread* (quando não realizado processamento paralelo). São utilizados nos mais diversos dispositivos, como celulares, computadores, auto-falantes com assistente de voz, televisores *smart*, computador de bordo para veículos automotores, entre outros. É a opção com custo mais elevado da lista em termos financeiros.

Por outro lado, os computadores de placa única (SBC) são equipamentos relativamente novos no mercado, mas que vem tomando espaço por seu custo-benefício e possibilitarem criação de dispositivos que antes seriam inviáveis. Foram criados inicialmente com intuito de promover ensino de computação e robótica com baixo custo de investimento para escolas, porém entusiastas foram os grandes responsáveis por promover este tipo de circuito, aplicando como firewall, servidor, media-center, e até mesmo substituindo computadores em equipamentos de medição em bancada.

Os Microcontroladores são computadores (compostos por CPU [6], memória, e interface de entrada e saída) de baixo custo, utilizados em quase todos equipamentos eletrônicos do nosso dia a dia. Inicialmente foram criados como computadores, contendo todos os componentes em um único chip (SHIRRIFF, 2016), porém o que diferencia atualmente um microprocessador e um microcontrolador é basicamente o poder de processamento e por consequência o custo de aplicação.

Para este projeto, o ideal é um microcontrolador, pois a escolha de quaisquer outros dispositivos (microprocessador, FPGA, SBC, ...) seria considerado um sobredi-

[4] Matriz de Portas Programáveis
[5] Computador de Placa Única
[6] Unidade Central de Processamento

mensionamento de projeto, dado que a capacidade de processamento necessária não é elevada.

As opções de microcontrolador levadas em conta foram:

- MCS-51 (Intel 8051) Por ser uma família de microcontroladores mais utilizados na industria desde a década de 80, quando lançado.
- PIC 16F877A ou 12F675 (Microchip) A segunda opção mais comum seriam os PIC [7] desenvolvidos pela Microchip. A opção pelo 16F877A seria por conta da experiência de uso, e caso optasse pelo 12F675, seria por seu baixo custo e tamanho reduzido, porém teria uma redução considerável de GPIOs [8], reduzindo possibilidades de projeto.
- AVR (ATMEL), STM32 (STMicroelectronics), e MSP430 (Texas Instruments) são também alternativas viáveis, pois vêm sendo utilizados em diversas aplicações nos últimos anos.
- As opções com melhor relação custo-benefício são as famílias ESP8266 e ESP32, desenvolvidos pela Espressif em 2014 e 2016 respectivamente. O diferencial destes microcontroladores foi inovarem ao trazer integrado interface Wi-Fi, e exclusivamente no ESP32 comunicação Bluetooth (BLE 4.0[9]). Este diferencial é uma boa opção, pois reduz a utilização de um segundo circuito responsável pela comunicação com a internet.

Para a comunicação poderíamos utilizar também o HLK-M30, o RTL8710, ou até mesmo o MT6625LN. Porém ao optar pelo ESP, reduzimos espaço físico, utilizando apenas um circuito integrado para o processamento e para a comunicação Wi-Fi, além de reduzir o custo. Também foi importante na decisão a simplicidade para programar o microcontrolador, que tem tanto um SDK [10](Kit de desenvolvimento de software) próprio desenvolvido pela Espressif, quanto opções alternativas utilizando o Arduino SDK, PlatformIO, MicroPython, Ruby, Lua, etc.

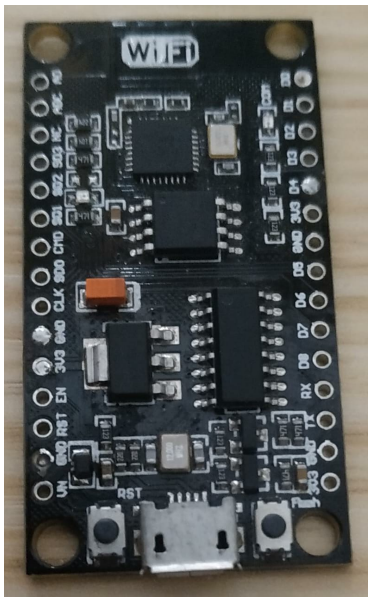
2.2.1 ESP8266

ESP8266 (Figura 3) é um controlador de 32 bits, frequência de 160MHz (clock), equipado para trabalhar com WiFi (balun de RF, LNA, amplificador de potência, etc), que roda um RTOS [11] (Sistema operacional em tempo real). Este microcontrolador permite

[7] Peripherical Interface Controller - Controlador de Interface Periférica
[8] General Purpose Input/Output - Entrada e Saída de Uso Geral
[9] Bluetooth Low Energy
[10] Software Development Kit
[11] Real Time Operational System

comunicação serial (I2C, I2S, SPI, UART, SDIO), atualização de firmware OTA (Over The Air), além de possuir modos de execução com baixo consumo de potência (Light-sleep, Deep-sleep, e Modem-sleep).

Figura 3 – Circuito nodemcu com esp8266.



Fonte: o autor.

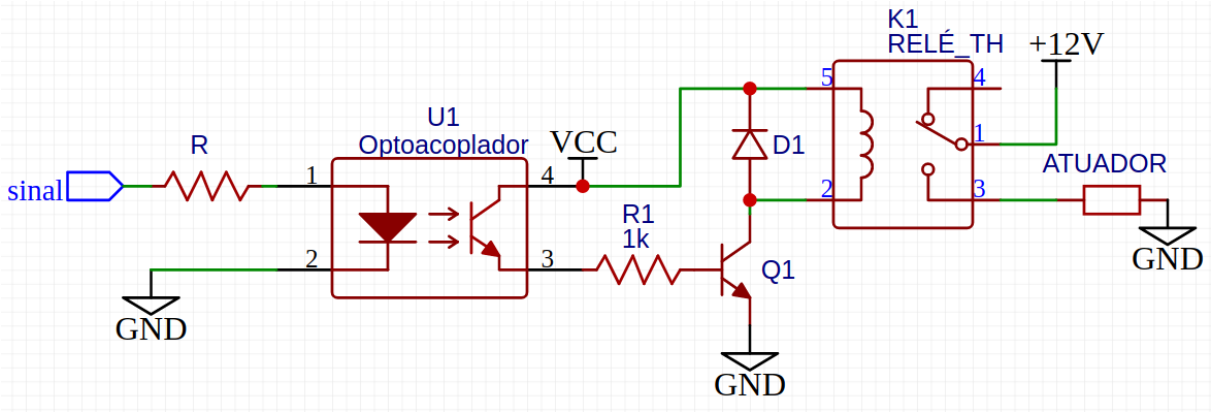
Os modos mais adequados para este uso são o Light-sleep e o Modem-sleep, onde respectivamente, toda parte de RF [12] fica desligada, e onde todo RTOS fica em pausa, acordando em caso de interrupção. Em testes após finalizado, o circuito completo consumiu em média 1.35 W de potência.

2.3 Atuador

Como interface de saída do circuito, foi utilizada uma fechadura solenoide, acionada por um relé, responsável pela abertura da porta. A Figura 4 demonstra a parte eletrônica responsável pelo acionamento do atuador. A utilização do optoacoplador é de suma importância para isolar o circuito de controle do circuito de potência, dado que mesmo que neste uso o atuador utilize corrente contínua, poderia ser facilmente utilizado com qualquer atuador, cancela, ou catraca, possibilitando aplicar não apenas na porta das salas, mas também o controle de eventos da faculdade, filas, etc.

[12] Radio-frequência

Figura 4 – Esquema utilizado para o acionamento do atuador.



Fonte: o autor.

2.4 Protótipo final

A circuito final (Figura 5) foi aplicado na porta do GAMA (Grupo de Pesquisa em Arquitetura de Computadores e Microeletrônica). Sendo necessário uso externo de uma fonte de tensão 12V para alimentação (tanto do circuito, quanto do atuador), e os conectores do atuador (Figura 7). Além disso, para evitar problemas com o WiFi, foi criada uma rede exclusiva para esta aplicação, com ssid oculto e permissão de conexão (controlada no roteador) apenas para os endereços físicos (MAC) dos circuitos em questão. Para evitar transtornos, a rede está dividida em duas VLAN, sendo a primeira através de cabo conectado no computador e no roteador, e a segunda através do WiFi, e todo acesso http ao gateway é bloqueado quando requisitado através do WiFi, evitando ataques mal intencionados de pessoas externas.

Figura 5 – Protótipo final.

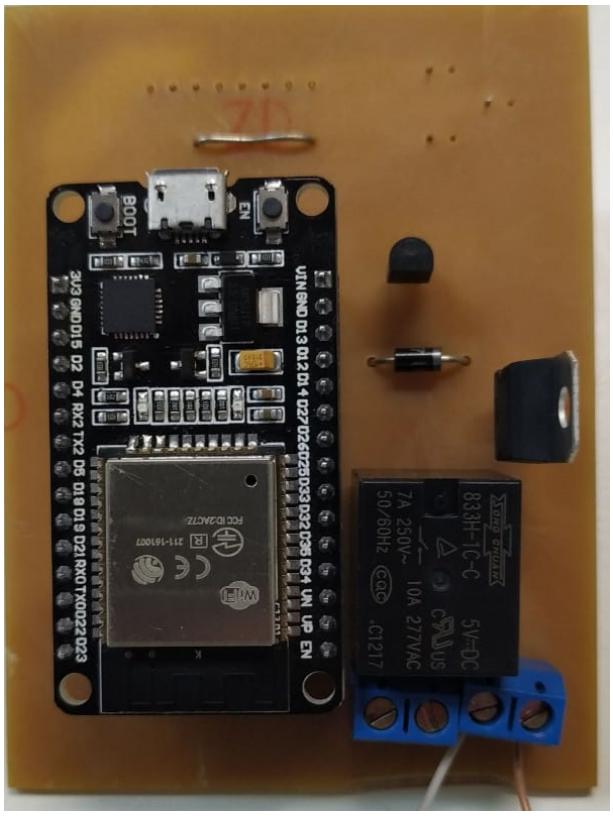
Fonte: o autor.

Os leds visíveis na Figuras 5, e 6 são para identificação visual, onde o led vermelho simboliza a falta de permissão, o verde simboliza o acionamento do atuador, pois foi permitida a entrada, e o amarelo é para verificar a conexão com a internet e a comunicação com o servidor. A Figura 7 representa o verso da placa, onde fica o regulador de tensão, o microcontrolador, e o relé utilizado para acionar o atuador.

Figura 6 – Placa de circuito impresso sem a proteção externa – Frente.

Fonte: o autor.

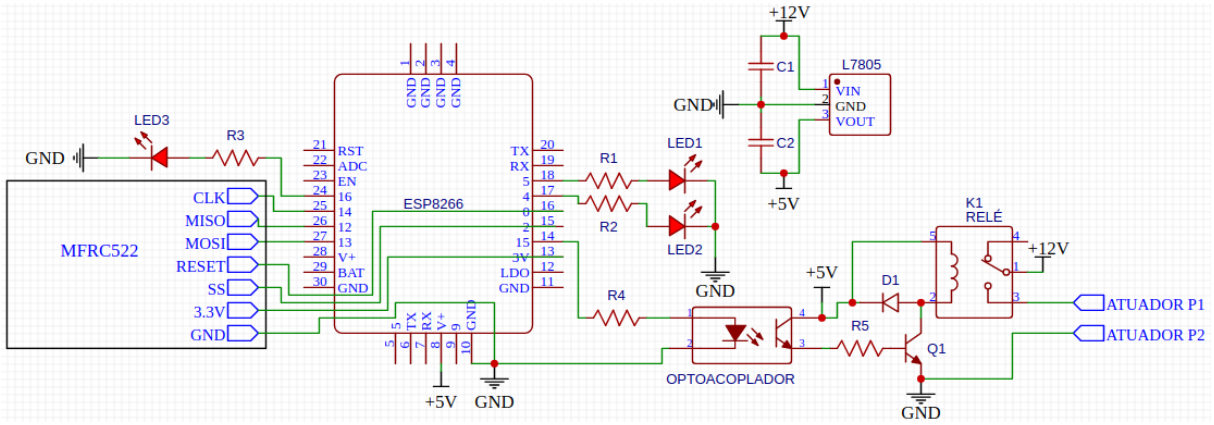
Figura 7 – Placa de circuito impresso sem a proteção externa – Verso.



Fonte: o autor.

A Figura 8, representa o esquemático e conexões da placa de circuito desenvolvida. Nela temos o microcontrolador ESP8266, o qual não foi soldado, apenas encaixado com soquete apropriado. Também temos conectado o leitor MFRC522, o qual ficará em contato com a parte externa da caixa de proteção (para ter maior alcance de leitura). Os leds *LED1*, *LED2*, e *LED3* são respectivamente os leds verde, amarelo, e vermelho utilizados para identificação visual. O optoacoplador utilizado é um PC817, e o regulador de tensão é um L7805. Este regulador é amplamente utilizado, e consegue suprir 5V para quase 5W de potência. O transistor utilizado após o optoacoplador foi um BC547, e o diodo 1N4007, o qual suporta até 1A de corrente elétrica.

Figura 8 – Esquemático da placa desenvolvida.



Fonte: o autor.

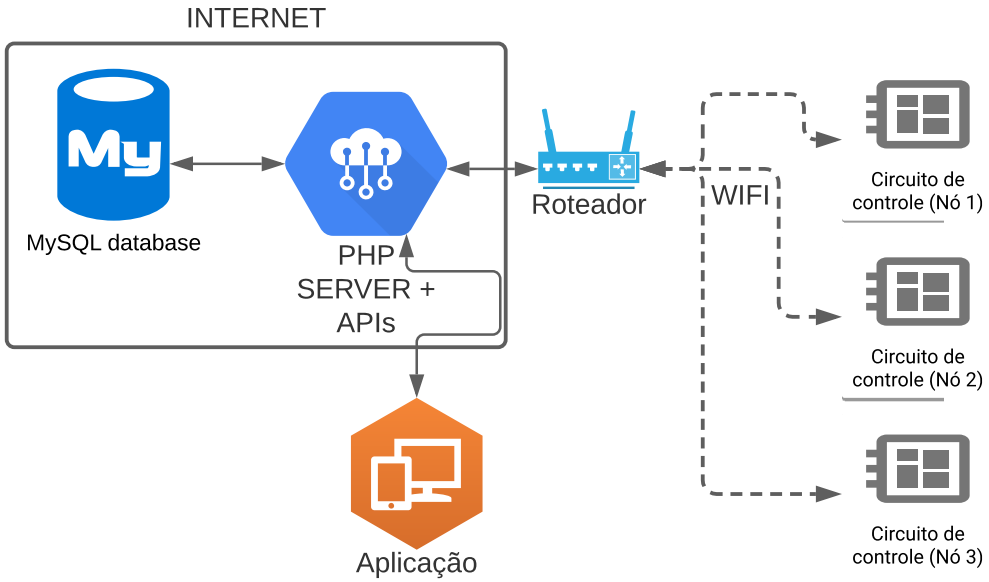
Em casos de queda de energia, dado baixo consumo de potência do circuito, poderia ser alimentado por uma bateria *Chumbo-Ácida Selada* de 12V e 7Ah, mantendo por dias o funcionamento do circuito, porém para isso, dependeria manter a rede local em pleno estado de funcionamento pelo mesmo período. Desta forma, optou-se por manter a possibilidade de uso com chave convencional, para estes excepcionais casos.

A placa desenvolvida atendeu os requisitos desejados, pois tem um custo de produção de R$ 100,00 baseado nos valores de materiais atuais, o que gera uma economia de no mínimo 80% quando comparado com as opções mais simples do mercado. Ademais, na situação de aceite para implantação em todo campus, pode-se requisitar os microcontroladores em lote com a fabricante, reduzindo significativamente o custo, além das cases personalizadas que podem ser impressas na impressora 3D da própria faculdade.

3 IMPLEMENTAÇÃO EM NÍVEL DE SOFTWARE

A etapa de software é responsável por grande parte do sistema desenvolvido. Nossa rede é composta pelos nós dos circuitos desenvolvidos, pela aplicação que possibilita a gerência dos nós e verificação dos dados de log[1], pelo servidor PHP hospedado em nuvem e pelo banco de dados MySQL acessado unicamente pelo servidor, sendo este também hospedado em nuvem. Na Figura 9 podemos visualizar graficamente o funcionamento da rede.

Figura 9 – Estrutura da rede utilizada no sistema.



Fonte: o autor.

3.1 Banco de Dados

> SQL significa *Structured Query Language* e é a linguagem padrão utilizada pelos bancos de dados relacionais. Os principais motivos disso resultam de sua simplicidade e facilidade de uso. (CARVALHO, 2015)

Para o armazenamento de dados foi escolhido um dos bancos de dados relacional

[1] Registro de informações

mais utilizado no mundo, o MySQL. A escolha do banco de dados relacional se deu pelas inúmeras vantagens em relação a um arquivo "flat" [2], como fornecer mais rapidamente os dados, sistemas pré-definidos de privilégio e pesquisa, e facilidade de consulta e extração de dados (WELLING, 2005). Já o MySQL foi escolhido por ser uma tecnologia livre, leve, rápida, e que é utilizada em diversos sistemas de hospedagem (BENTO, 2014).

Dentro do banco de dados, as informações são salvas em tabelas. As tabelas utilizadas neste trabalho possuem a seguinte estrutura:

Tabela 1 – Tabela referente aos acessos realizados.

IdSala	Momento	Usuario
1293	"2019-10-11 21:10:01"	"141150090"
1294	"2020-09-12 20:11:11"	"159231265"
1295	"2021-08-13 19:12:21"	"1654321987"

Fonte: Produzido pelo autor.

A Tabela 1 tem como *IdSala* valores de tipo INTEGER, correspondentes a um número que representa a sala em questão, *Momento* valores do tipo DATETIME, que tem sua representação no formato "YYYY-MM-DD hh:mm:ss", e *Usuario*, que utiliza valores VARCHAR representando o número de matrícula do aluno ou do professor/funcionário. A escolha do tipo INTEGER se deu por ter uma precisão de 32 bits, do tipo DATETIME por ter variação de data até o ano de 9999, e por VARCHAR dado que as regras para tamanho do número de matrícula e mapeamento de caracteres para matrícula podem mudar tanto com o tempo, quanto por instituição.

A tabela referente as tentativas de acesso bloqueadas tem a mesma estrutura, apenas mudam os dados internos, sendo armazenados apenas usuários que tentaram acessar sem permissão de acesso.

Tabela 2 – Tabela de permissão de acesso.

Usuario	IdSala	HorarioEntrada	HorarioSaida	DiasDaSemana	Bloqueio
"141150090"	1293	"6:00:00"	"22:59:00"	"123456"	0

Fonte: Produzido pelo autor.

A Tabela 2 tem como *HorarioEntrada* e *HorarioSaida* valores de tipo TIME, que representa respectivamente o mínimo e máximo horário permitido adentrar a sala, *Bloqueio* é do tipo TINYINT(1) e armazena apenas valores 0 e 1. Já a coluna *DiasDaSemana* é

[2] Arquivo em texto plano para conter dados que devem ficar salvos.

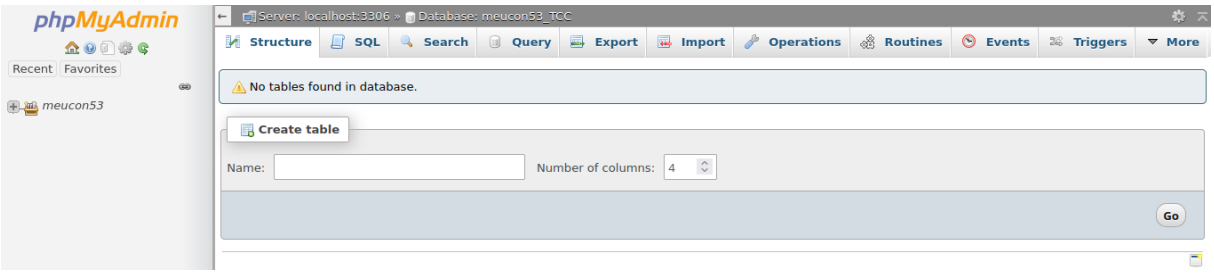
do tipo VARCHAR e é organizada da seguinte forma: relacionando os dias da semana (começando em domingo) com os números de 0 a 6, podemos montar uma string com os números referentes aos dias permitidos. Por exemplo, se um aluno tem permissão apenas de acessar entre a segunda-feira e sexta-feira, o campo *DiasDaSemana* será preenchido com o texto "12345". Assumindo outro aluno que tem permissão todos os dias da semana, o campo seria preenchido com "0123456". Por outro lado, se pudesse acessar apenas aos finais de semana, seria "06". Os demais campos seguem os mesmos tipos da Tabela 1.

Tem-se também uma tabela que relaciona a matrícula do *Usuário* com a tag cadastrada. Esta tag RFID é única, e cada aluno, professor, e funcionário tem apenas uma. A coluna que cadastra a tag é do tipo CHAR(8), pois cada tag é composta por oito caracteres hexadecimais.

3.1.1 phpMyAdmin

A criação e configuração das tabelas no banco de dados se deu de forma simplificada com o uso do *phpMyAdmin*, ferramenta esta em modo gráfico que permite edição de forma intuitiva. A Figura 10 apresenta uma captura de tela do sistema phpMyAdmin.

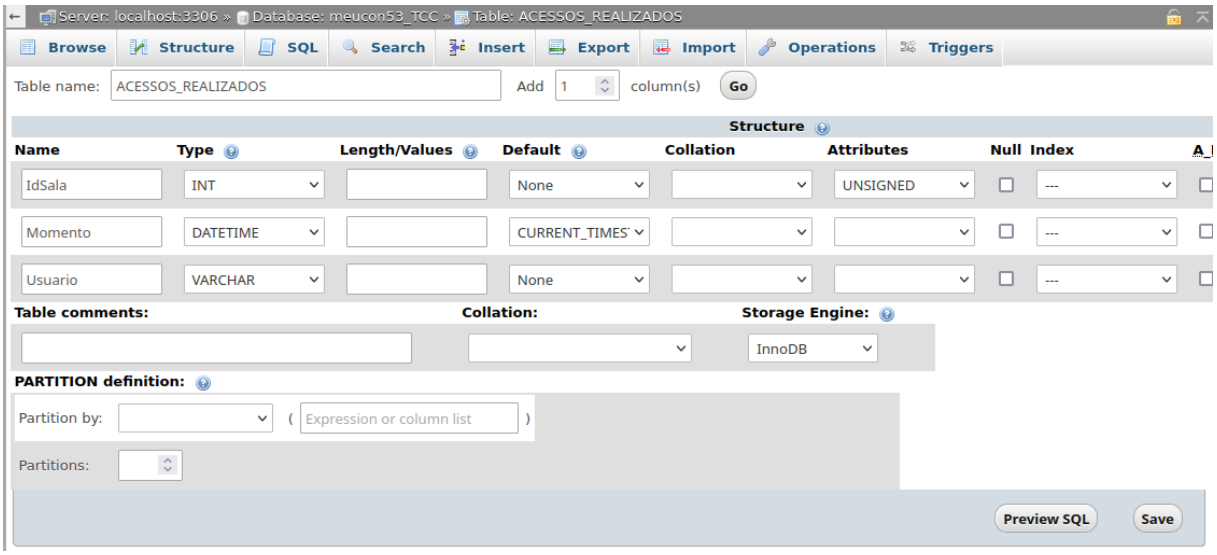
Figura 10 – Interface do *phpMyAdmin* no servidor utilizado.



Fonte: o autor.

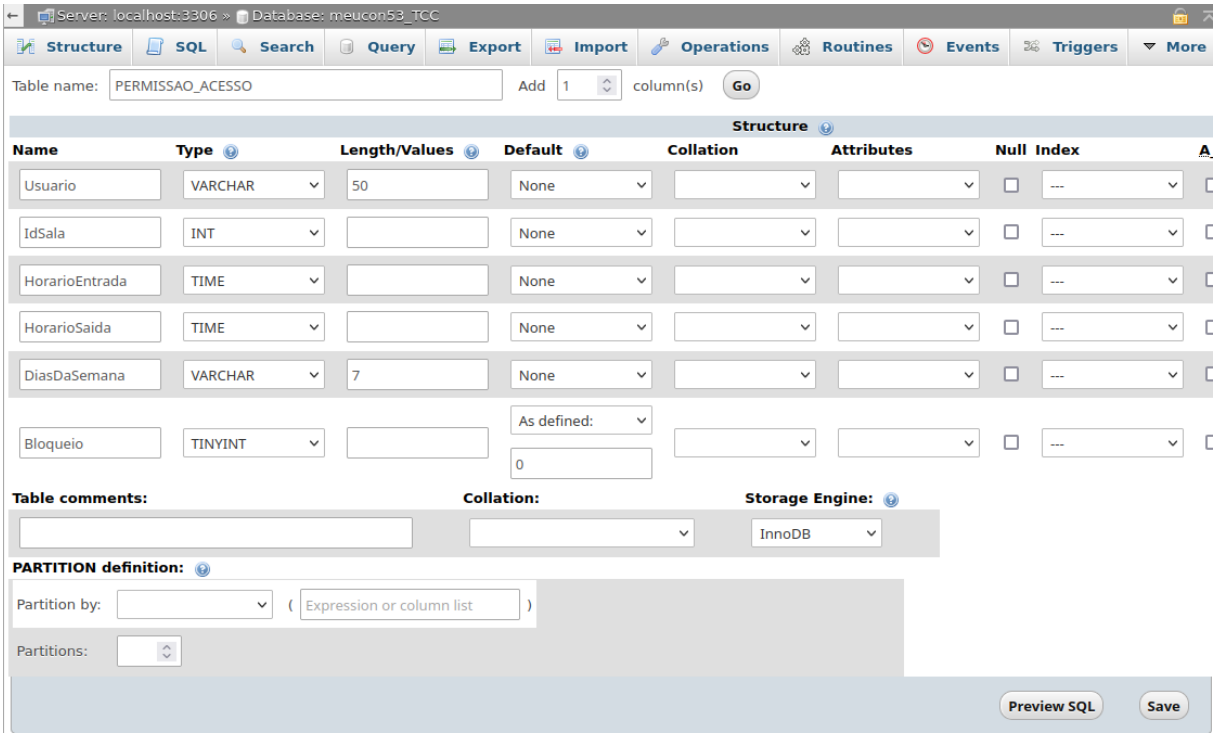
A criação das Tabelas de Acessos e Permissões (Tabelas 1, e 2) foi realizada conforme as Figuras 11 e 12 respectivamente.

Figura 11 – Processo de criação da Tabela de Acessos.



Fonte: o autor.

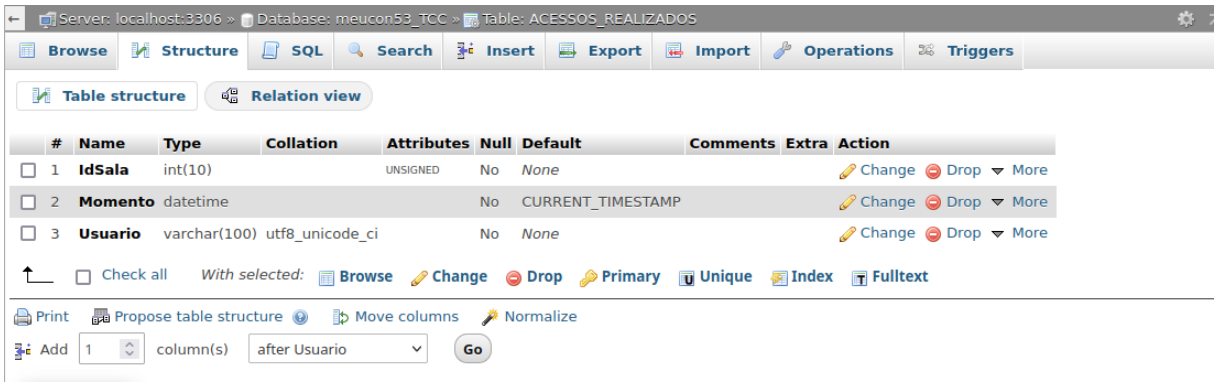
Figura 12 – Processo de criação da Tabela de Permissões.



Fonte: o autor.

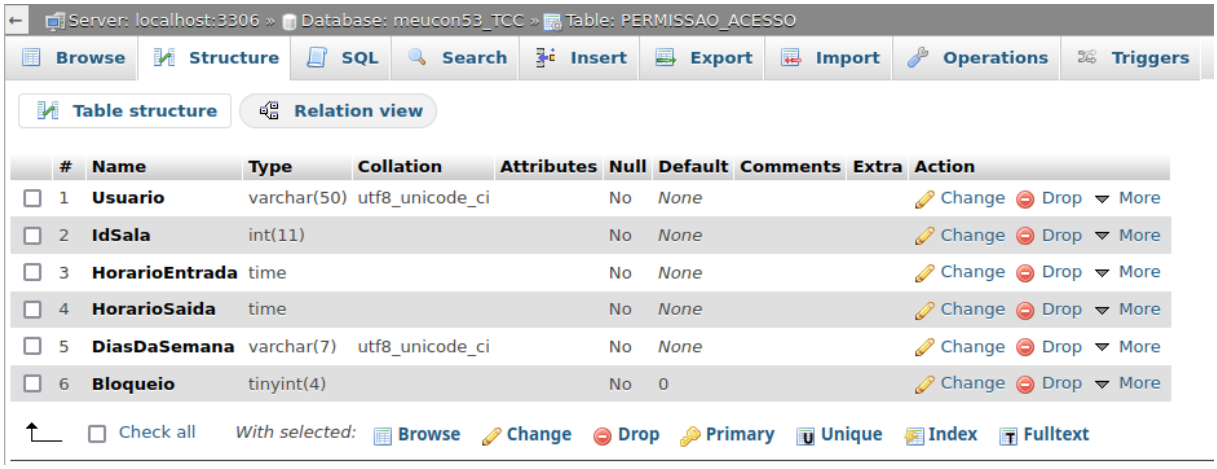
As estruturas das Tabelas 1 e 2 podem ser vistas de forma técnica nas Figuras 13 e 14.

Figura 13 – Estrutura da Tabela de Acesso.



Fonte: o autor.

Figura 14 – Estrutura da Tabela de Permissões.



Fonte: o autor.

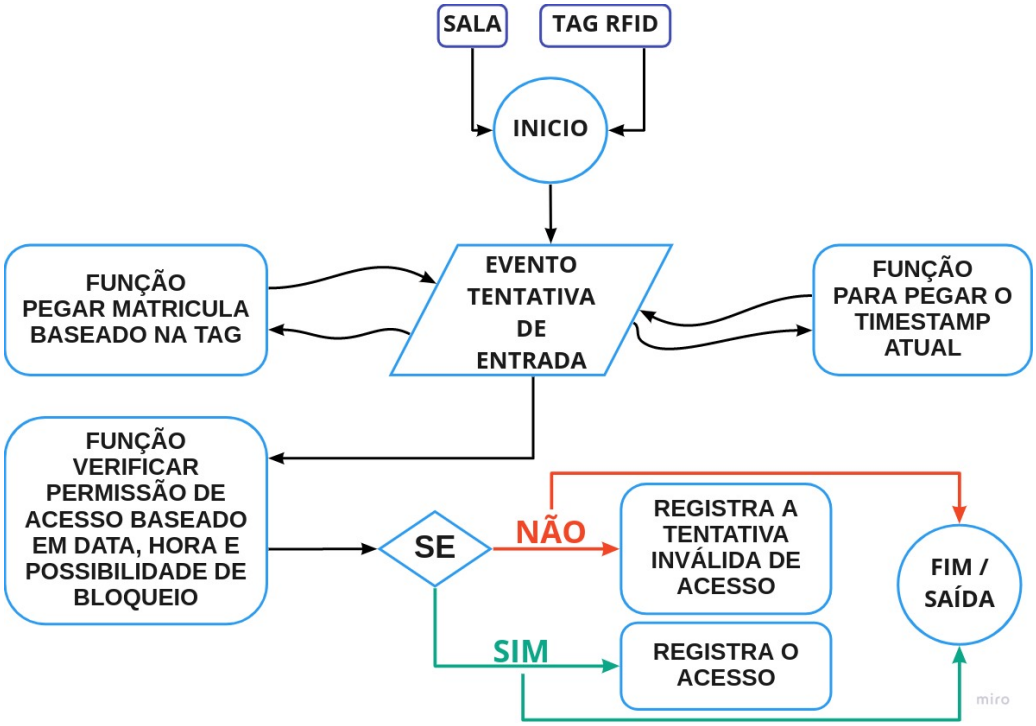
O armazenamento de tabelas não é a única responsabilidade do banco de dados, como disse (CARVALHO, 2015), as relações entre tabelas, pesquisa e operações são um dos pontos mais fortes de um banco de dados SQL.

Desta forma, tudo que for relacionado ao banco de dados, fica sob responsabilidade do mesmo [3] (Hunt (2002) explica que encapsular, consiste em ocultar todos os detalhes de um objeto que não contribui para suas características essenciais, ou seja, o interno é oculto, e apenas as interface externas são visíveis para outros). Para isso, utilizam-se funções e procedimentos (*functions* e *procedures*), que realizam as operações, comparações e leituras das tabelas de forma dinâmica, retornando apenas o valor desejado (sim ou não) para validar a requisição. A Figura 15 demonstra de forma gráfica o procedimento de

[3] Encapsulamento

verificação da permissão de acesso, com as requisições e inserções necessárias (inserção de log tanto em caso de acesso liberado quanto quando não permitido).

Figura 15 – Sequência de passos utilizada para validação em código SQL (Interno no banco de dados).



Fonte: o autor.

Vale destacar que apenas a criação das tabelas e funções de controle (etapas de desenvolvimento) utilizaram o PHPMyAdmin, pois a administração é realizada pela aplicação gráfica, que abstrai conhecimentos SQL, simplificando o uso.

3.2 Segurança da Informação

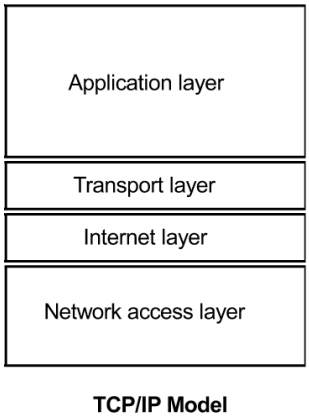
Como a proposta deste trabalho é justamente possibilitar uma melhor segurança para os laboratórios, é de suma importância não gerar outra falha de segurança através desta solução. Para proteger então a comunicação entre o circuito e o servidor, e a aplicação e o servidor, decidiu-se por permitir apenas tráfego no *PORT 443*, utilizado pelo protocolo TLS [4], que é uma camada de segurança para os dados.

Segundo (OPPLIGER, 2016), o protocolo de rede HTTP que conhecemos atua na Camada de Aplicação (Figura 16), e contém diversas vulnerabilidades, como ataques

[4] Transport Layer Security – Segurança da Camada de Transporte

MITM [5], já o protocolo TLS atua na camada de transporte, criptografando os dados e evitando espionagens.

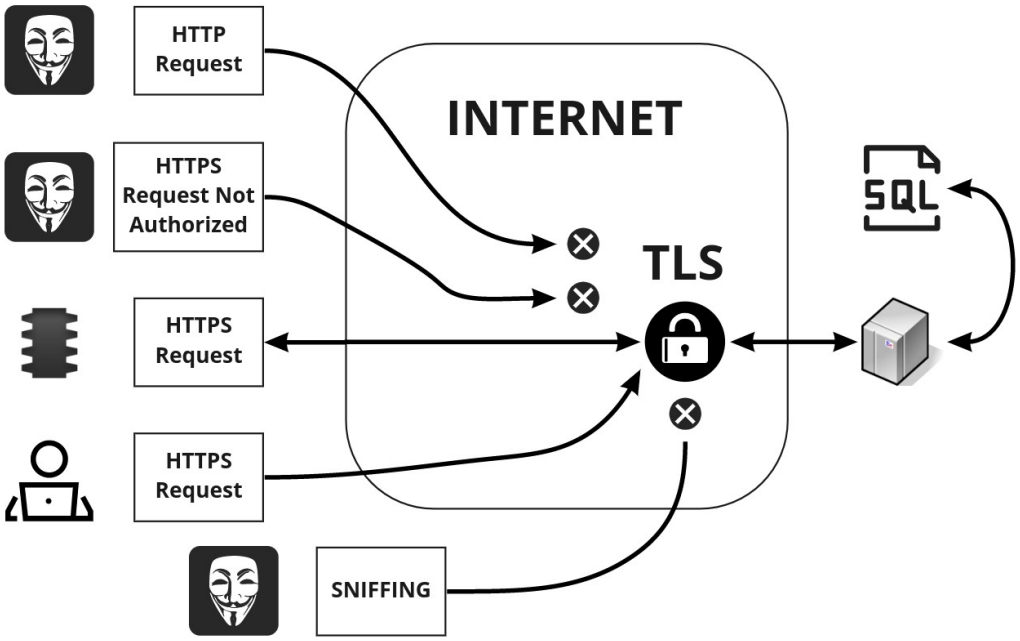
Figura 16 – Modelo de camadas TCP/IP.



Fonte: Oppliger (2016), p. 3.

No livro (HOFFMAN, 2020), temos exemplos de ataques MITM para captura de dados HTTP. A solução para evitar este tipo de ataque é bastante simples, utilizar o protocolo HTTPS, que nada mais é que uma união dos protocolos HTTP com TLS.

Figura 17 – Estrutura de comunicação entre o circuito e o servidor.



Fonte: o autor.

[5] Man In The Middle – Homem no meio

Para garantir acesso HTTPS, precisa-se de uma entidade certificadora para o servidor. Dessa forma, optou-se por hospedar o servidor na empresa Hostgator para fins de estudo. A hospedagem incluiu entre outras coisas, banco de dados MySQL, e servidor PHP com certificado digital. Com isso, foi possível bloquear acessos HTTP (Port 80), inseguros, e garantir estabilidade para grande demanda de acessos, assim como Firewall para proteção contra outras vulnerabilidades, como ataques de negação de serviço [6]. Entretanto, vale lembrar que a hospedagem externa se deu por ser um trabalho aplicado inicialmente apenas em um grupo de pesquisa. Caso opte-se pela implantação em outros ambientes da universidade, o servidor será hospedado localmente no sistema interno da universidade, evitando problemas de conexão caso hajam oscilações no acesso à internet.

Na Figura 17 é demonstrado graficamente que através do protocolo TLS não se tem acesso aos dados por *Sniffing* (captura de pacotes), além disso, o servidor foi configurado para não aceitar conexões inseguras (HTTP), e conforme será abordado na seção 3.3, mesmo requisições criptografadas (HTTPS) que não passarem em todas as etapas de segurança serão descartadas.

3.3 Servidor Utilizado

Em um modelo *Cliente-Servidor*, o servidor é caracterizado por um computador (hardware) ou programa (software) que provê um serviço através da internet (www [7]) (TANENBAUM, 2003). No modelo em questão o serviço é oferecido por meio de solicitações (requests) e respostas (reply), entre os clientes e o servidor, por meio da rede. A Figura 18 mostra a relação de comunicação entre o servidor e o cliente, através de requisição e resposta, na rede.

Figura 18 – Modelo cliente-servidor.

Fonte: Adaptado de (TANENBAUM, 2003)

Nos primórdios da internet, manter um serviço rodando exigia muito custo, pois requisitava uma máquina exclusiva para isso, contabilizando recurso e energia. Foi desenvolvida então a virtualização, que consistia em executar mais de um sistema operacional

[6] DOS – Denial of Service
[7] World Wide Web

na mesma máquina e ao mesmo tempo. Isso possibilitou um crescimento nos serviços de hospedagem, pois uma empresa de hospedagem poderia ter mais serviços rodando do que o número de máquinas ativas. Além disso, poderia se utilizar máquinas mais robustas e dividir o hardware entre os sistemas.

Todavia com a criação do *Docker*, tivemos uma redução de custo em termos de requisitos de hardware enorme para servidores, pois não era mais necessário criar tantos sistemas completos e dividi-los para cada aplicação. A ideia do *Docker* é a criação de *containers*, que rodam todos em cima da mesma plataforma (sistema), logo todos compartilham a mesma estrutura. Essa simplificação resultou em melhora de desempenho, velocidade de leitura em memória, escrita, redução em tempo de processamento e menor consumo de memória (POTDAR et al, 2020).

Com a transição dos sistemas virtualizados para os *containers*, hospedar serviços online ficou cada vez mais barato. Outrossim, hospedagem possibilita terceirização de ações como controle de firewall, acompanhamento do servidor (caso tenha problema com energia), já que conta com equipe especializada na administração das máquinas.

Neste trabalho o servidor consiste em APIs [8] (Interface de programação de aplicações) que são responsáveis por fazer a conexão entre as informações do banco de dados com o circuito físico, e com a aplicação de gerência. Estas APIs estão construídas com a linguagem de programação PHP, e a escolha da mesma se deu por ser uma linguagem interpretada pelo servidor, logo, as informações são processadas no servidor e apenas resultados de comunicação são repassados para o cliente.

3.3.1 Comunicação Circuito - Servidor

Tratar de segurança é sempre um *trade-off* (perde-e-ganha), e no caso da segurança de informação temos o tempo de processamento e custo em hardware em troca da segurança dos dados e da comunicação. Como o protótipo desenvolvido foi embarcado em um microcontrolador com poder computacional não muito elevado, e sabe-se que criptografia requisita muito recurso, foi mantida apenas a criptografia utilizada pelo protocolo HTTPS, sem redundâncias neste quesito.

Contudo, foram adicionadas outras camadas de segurança na comunicação no lado do servidor, já que o mesmo possibilita maior uso de recurso.

As requisições ao servidor para verificar a permissão de acesso utilizam o método POST, que envia o formulário através do corpo da mensagem, diferentemente do método GET que envia pela url da requisição.

Como dito anteriormente, o protocolo HTTPS evita ataques do tipo MITM e o Firewall reduz os danos por ataques DOS, porém, de forma a dirimir quaisquer danos por

[8] Application Programming Interface

sobrecarga de processamento do servidor e evitar atrasos na comunicação dos diversos nós, são utilizados como forma de validação para cada requisição o *User-Agent* e autenticação.

Deste modo, toda requisição que ocultar seu agente, ou utilizar qualquer agente não autorizado (os agentes são únicos para cada circuito, definidos no *firmware* como um número de identificação) será imediatamente descartada e terá um erro como resposta, evitando dificultando o trabalho de hackers.

Caso o agente seja válido, ou seja, alguém consiga se passar por algum nó do sistema, ainda há uma autenticação com usuário e senha que apenas os firmwares desenvolvidos para os circuitos conhecem, e novamente é retornado um erro em caso de comunicação descartada.

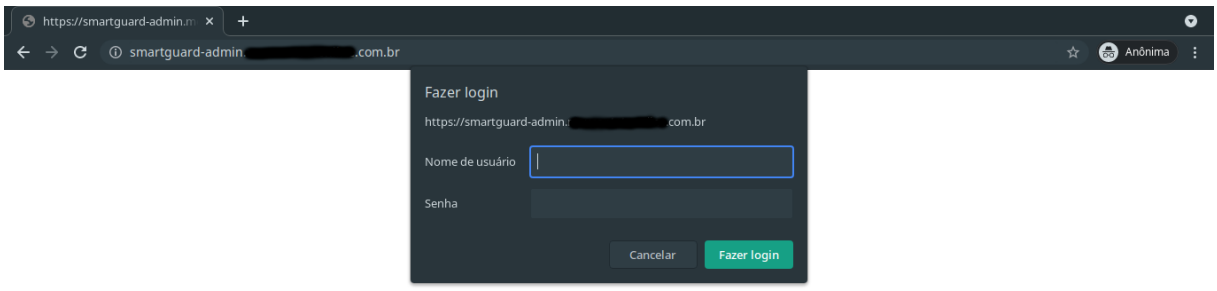
Como terceira etapa de verificação de autenticidade, existe o formulário contendo a chave RFID de representação pessoal, que só é validado caso esteja preenchido de forma correta e tenha passado pelas verificações anteriores.

Após o servidor garantir que os dados informados são verdadeiros, ele envia a chave RFID para o banco de dados. Este por sua vez contém diversas funções de automação para realizar as verificações de permissão e registrar o pedido. Após o registro, a função principal retorna a resposta para o servidor, que responde a requisição do circuito com a confirmação ou rejeição do acesso. A resposta é representada por 1 e 0, sendo 1 para confirmação e 0 para rejeição.

3.4 Aplicação para Controle do Sistema

A administração do sistema de acesso mantém-se sob responsabilidade de um gestor. Neste trabalho em questão, a gestão é encargo dos professores do grupo, diante disso, são necessárias credenciais de *login* para acessar o *sistema gerente*. Foi desenvolvido então um sistema web, de forma a facilitar todas as operações que pudessem ser relevantes, assim como proteger a integridade dos dados, pois no sistema são impostas regras, que buscam manter a padronização e segurança. A Figura 19 é uma captura de tela que mostra o pedido das credenciais para acessar o *sistema gerente*.

Figura 19 – Etapa de login para acessar o sistema que coordena as permissões e visualiza os eventos ocorridos.

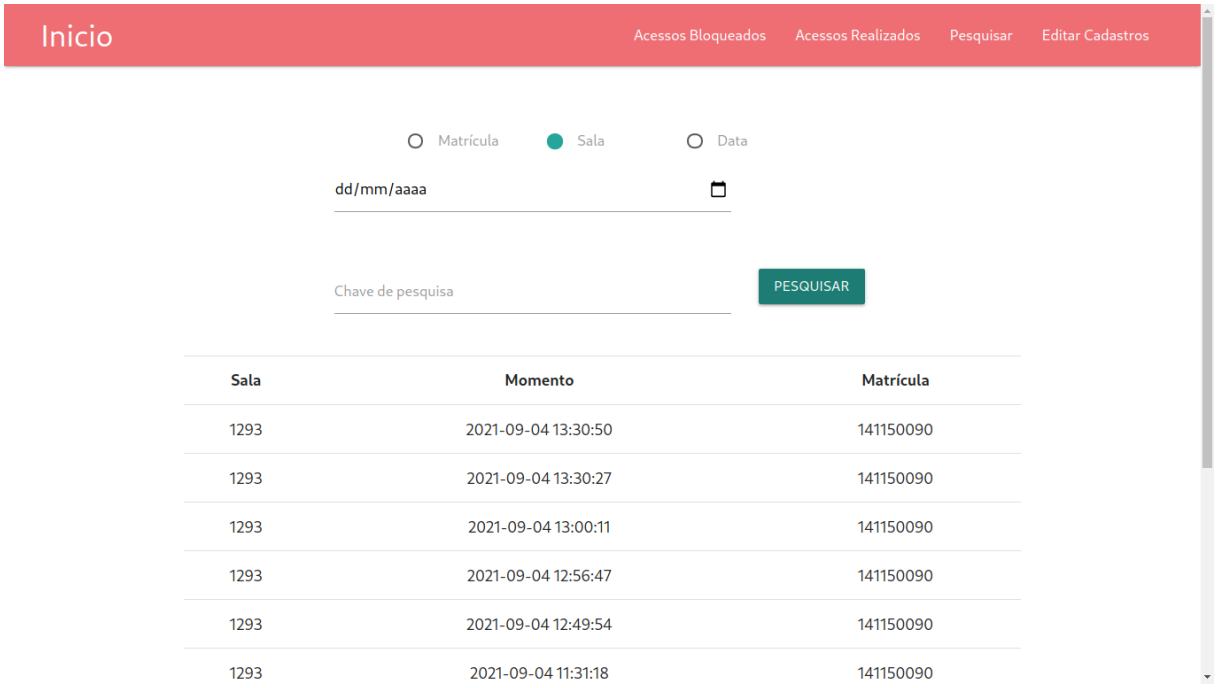


Fonte: Captura de tela realizada pelo autor.

Uma vez que conectado (login autorizado), é possível obter dados dos acessos permitidos e barrados, das permissões de acesso de determinado indivíduo, e inserir/atualizar cadastros.

Os acessos permitidos possibilitam verificar quem adentrou nas salas. Nesta página é possível filtrar por data, sala, ou matrícula, como pode-se ver na Figura 20.

Figura 20 – Sistema Gerente - Aba de visualização dos acessos realizados.

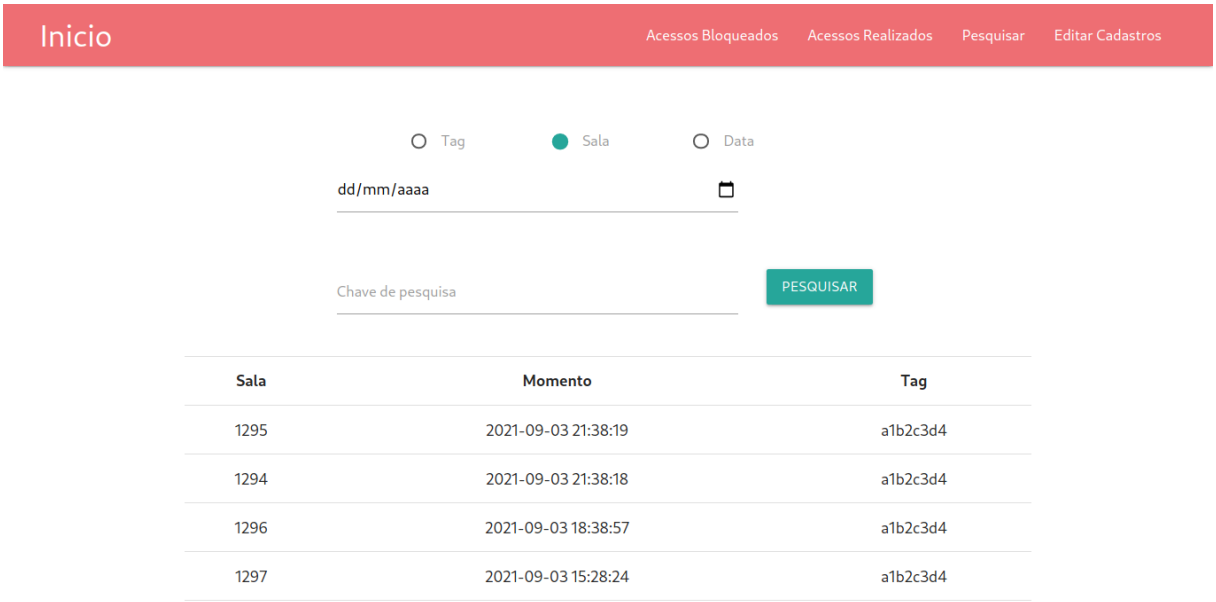


Fonte: Captura de tela realizada pelo autor.

Os acessos barrados também podem ser filtrados da mesma forma, e apresentam as chaves (tags) utilizadas na requisição inválida. Na Figura 21 pode-se ver um exemplo

de pesquisa baseado na tag.

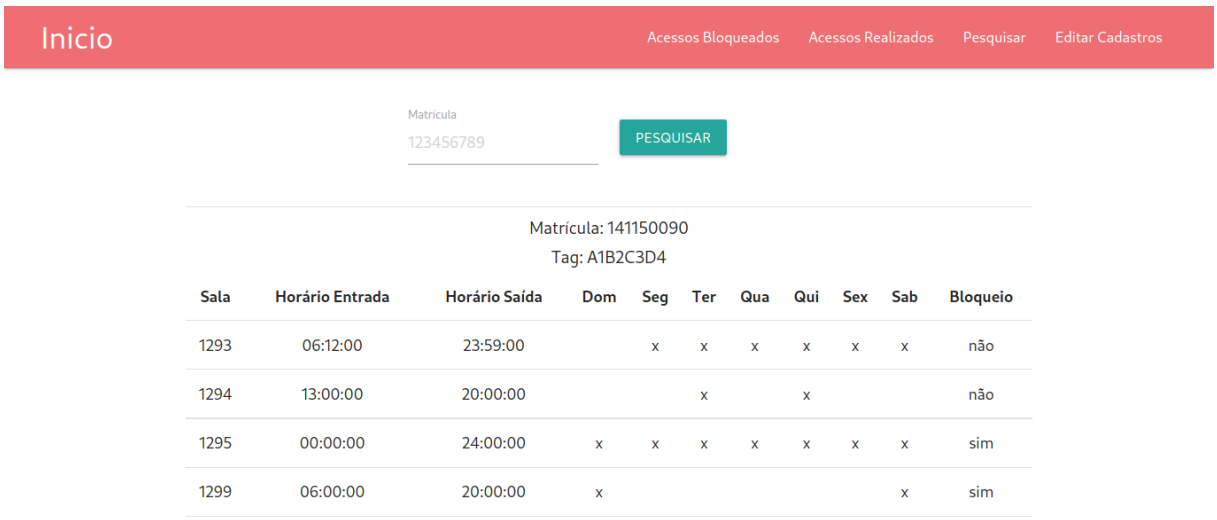
Figura 21 – Sistema Gerente - Aba de visualização das tentativas bloqueadas de acesso.



Fonte: Captura de tela realizada pelo autor.

A pesquisa de tag consiste em passar o número de matrícula, e são mostradas todas as permissões do indivíduo. Na Figura 22 tem-se o exemplo onde são apresentadas todas os dados de permissões para a matrícula “141150090”.

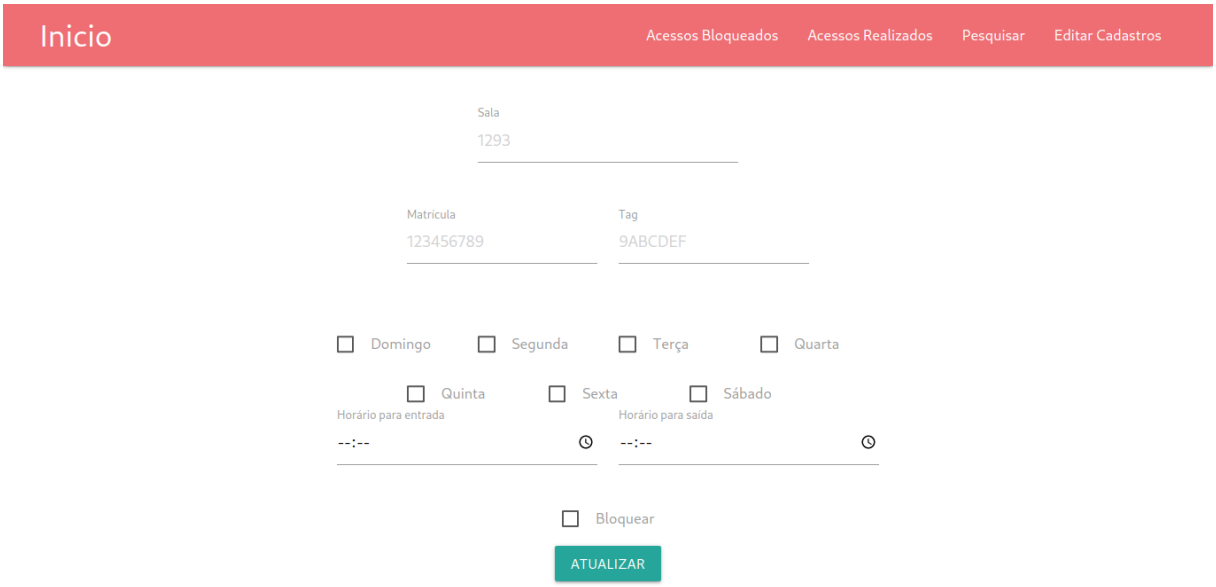
Figura 22 – Sistema Gerente - Aba de pesquisa das permissões cadastradas baseado na matrícula.



Fonte: Captura de tela realizada pelo autor.

Já na aba de inserção e atualização de cadastro, é possível modificar quaisquer dados do indivíduo. Porém a permissão para alterar estes dados deve ser compatível com o login utilizado, não sendo possível um professor de uma sala alterar os dados de salas que estejam sob responsabilidade de outro. Na Figura 23 tem-se a captura de tela da página de atualização cadastral.

Figura 23 – Sistema Gerente - Aba de inserção e/ou atualização de permissões de acesso.



Fonte: Captura de tela realizada pelo autor.

O programa web utilizado neste trabalho demonstra a aplicação do sistema e do hardware no ambiente universitário, porém, pela forma que foi modelado, poderia ser facilmente aplicado em diversos outros locais. Um hotel, por exemplo, poderia aproveitar muito para gerar dinamismo em *check in*, *check out*, bloqueio de quartos, assim como controle para o acesso de outros funcionários, evitando diversos transtornos.

4 TESTES DO SISTEMA PROPOSTO

Para verificar o funcionamento do sistema foram realizados alguns testes da comunicação entre os circuitos e o servidor. Estes testes foram divididos entre os testes de API e o teste do circuito final.

Os testes de API foram realizados variando parâmetros como User-agent, usuário e senha (da autenticação HTTPS), e tag RFID. Desta forma pode-se verificar diversos aspectos que devem estar em operação, além de verificar se as etapas de verificação adicionadas na comunicação estão funcionando conforme determinado.

4.1 Testes funcionais de API

Os testes a seguir foram realizados utilizando a linguagem de programação *Python*, em sua versão 3.8. Python é uma linguagem de programação interpretada, multiplataforma, orientada a objetos, e com sintaxe simples, com escrita de código focada na legibilidade humana.

Para os testes realizados, alguns dados foram fixados.

- User-agent: Para realizar os testes, de forma a não utilizar os padrões verdadeiros (por motivo de segurança), foi estipulado que todo User-agent dos circuitos terão padrão "Porta/X", onde X é o Id da porta específica (gravado no firmware do circuito).

- Usuário: O usuário liberado para a autenticação HTTPS durante os testes foi "esp_sistemalocal".

- Senha: A senha utilizada para a autenticação HTTPS é "semsenha".

- Tag: A tag utilizada por padrão nos testes é "a1b2c3d4".

4.1.1 User-agent

O teste do agente funcionou da seguinte forma, foi criada uma lista com agentes comuns de navegadores web, smart-phones, kindle, entre outros, e adicionado a esta lista três agentes válidos. Os outros parâmetros seguiram os dados fixos.

Tabela 3 – Testes de segurança em relação ao *User-Agent* da requisição para a API.

Agente	Status HTTP	Resposta
AppleWebKit/537.36	407	Erro
Chrome/62.0.3202.84	407	Erro
Mobile Safari/537.36	407	Erro
CriOS/69.0.3497.105	407	Erro
Mobile/15E148	407	Erro
Safari/605.1	407	Erro
FxiOS/13.2b11866	407	Erro
Edge/12.10536	407	Erro
SamsungBrowser/3.3	407	Erro
Silk/47.1.79	407	Erro
Roku4640X/DVP-7.70	407	Erro
Dalvik/2.1.0	407	Erro
NX/3.0.4.2.12	407	Erro
Kindle/3.0	407	Erro
Porta/1293	200	{'status': '1'}
Porta/1294	200	{'status': '0'}
Porta/1295	200	{'status': '0'}

Fonte: Produzido pelo autor.

Nota 1: *Erro* é demonstrado por uma página de erro quando a requisição é realizada, ou corpo (body) vazio na resposta (quando utilizado script para fazer a requisição).

Nota 2: O *status* 1 significa que foi autorizado o acesso e registrado já no sistema. O *status* 0 significa que o acesso não foi autorizado, e foi registrado nas tentativas inválidas.

A Tabela 3 relaciona os agentes utilizados nos testes, com a resposta de retorno e o *status code*. Percebe-se que todos os agentes que não seguiram o padrão especificado no teste retornaram um *status-code* 407, e uma mensagem de erro. Já nos agentes existentes o *status* foi 200 (Ok) e houve uma resposta no formado JSON. O valor 1 e 0 (significando respectivamente verdadeiro e falso para a permissão de acesso) se deu pois, mesmo que todas as portas estivessem cadastradas, a tag utilizada só tinha permissão de acesso na sala 1293.

Abaixo tem-se o código utilizado para o teste (A Tabela 3 representa o arquivo exportado pelo teste):

```
import requests, json, time, dados
import pandas as pd
from teste_default import teste
url = dados.url
tag = dados.tag
usuario = dados.usuario
senha = dados.senha
wrong_agents = ["AppleWebKit/537.36",
                "Chrome/62.0.3202.84",
                "Mobile Safari/537.36",
                "CriOS/69.0.3497.105",
                "Mobile/15E148",
                "Safari/605.1",
                "FxiOS/13.2b11866",
                "Edge/12.10536",
                "SamsungBrowser/3.3",
                "Silk/47.1.79",
                "Roku4640X/DVP-7.70",
                "Dalvik/2.1.0",
                "AppleTV6,2/11.1",
                "NX/3.0.4.2.12",
                "Kindle/3.0"]
write_agents = ["Porta/1293",
                "Porta/1294",
                "Porta/1295"]
def executar_testes(url,usuario,senha,tag,agentes):
    print("Iniciando testes")
    status_list = [0]*len(agentes)
    resposta_list = [0]*len(agentes)
    for i,agente in enumerate(agentes):
        time.sleep(0.1)
        status_list[i],resposta_list[i] = teste(url,agente,usuario,senha,tag)
    df = pd.DataFrame({'Agente':agentes,
        'Status':status_list,
        'Resposta':resposta_list})
    return df
tabela = executar_testes(url,usuario,senha,tag,wrong_agents+write_agents)
tabela.to_csv('teste_agente.csv', index=False)
```

4.1.2 Usuário

Para o teste do usuário, foi criada uma lista com o usuário real e usuários aleatórios, e fixados os outros parâmetros. Para quaisquer usuários incorretos a requisição deve retornar erro. Os dados reais de autenticação "user" e "password" são mantidos apenas no firmware do circuito, então servem para verificar se a requisição vem do circuito ou de outro computador.

Tabela 4 – Testes de segurança em relação ao *Usuário* utilizado na autenticação da requisição para a API.

Usuarios	Status HTTP	Resposta
esp_sistemalocal	200	{'status': '1'}
admin	401	Erro
root	401	Erro
usuario	401	Erro
tyzdfmmwoz	401	Erro
yvibrb	401	Erro
fczfrq	401	Erro
xigki	401	Erro
ytiemzl	401	Erro
sgpwjx	401	Erro
nxrzf	401	Erro
xrbgoes	401	Erro

Fonte: Produzido pelo autor.

Nota: *Erro* é demonstrado por uma página de erro quando a requisição é realizada, ou corpo (body) vazio na resposta (quando utilizado script para fazer a requisição).

A Tabela 4 relaciona os usuários utilizados nos testes, com a resposta de retorno e o *status code*. Percebe-se que todos os usuários incorretos retornaram um *status-code* 401, e uma mensagem de erro. Já o usuário verdadeiro recebeu *status* 200 (Ok) e houve uma resposta no formado JSON. O valor 1 significa que foi permitido o acesso.

Abaixo tem-se o código utilizado para o teste (A Tabela 4 representa o arquivo exportado pelo teste):

```
import json, time, dados
import pandas as pd
from teste_default import teste
```

```
from random import randrange,choice
from string import ascii_lowercase
url = dados.url
tag = dados.tag
usuario = dados.usuario
senha = dados.senha
agente = dados.agente
def random_name():
    return ''.join(choice(ascii_lowercase) for _ in range(randrange(5,12)))
usuarios = [usuario,'admin',
    'root','usuario']+[random_name() for _ in range(8)]
def teste_usuario(url,usuarios,senha,tag,agente):
    print("Iniciando teste")
    status_list = [0]*len(usuarios)
    resposta_list = [0]*len(usuarios)
    for i,usuario in enumerate(usuarios):
        time.sleep(0.1)
        status_list[i],resposta_list[i] = teste(url,agente,usuario,senha,tag)
    df = pd.DataFrame({'Usuarios':usuarios,
        'Status':status_list,'Resposta':resposta_list})
    return df
tabela_teste_usuarios = teste_usuario(url,usuarios,senha,tag,agente)
tabela_teste_usuarios.to_csv('teste_usuarios.csv', index=False)
```

4.1.3 Senha

O teste de senha foi uma variação do teste 4.1.2, entretanto a variação de dados para as requisições foi na senha utilizada para a autenticação HTTPS. Foi criada uma lista com a senha correta e diversas outras senhas definidas de forma aleatória, e mantendo fixos os outros parâmetros corretos.

Tabela 5 – Testes de segurança em relação a *Senha* utilizada na autenticação da requisição para a API.

Senhas	Status HTTP	Resposta
semsenha	200	{'status': '1'}
s4ytyo4fip	401	Erro
xuon6cxv	401	Erro
nckaiuc04xl	401	Erro
jr7ax4o6hy	401	Erro
8853ioi2cyk	401	Erro
d6c02ye25srv9	401	Erro
u1u1jjpvnx	401	Erro
w54ra6s12e	401	Erro
ppnq4jn6	401	Erro

Fonte: Produzido pelo autor.

Nota: *Erro* é demonstrado por uma página de erro quando a requisição é realizada, ou corpo (body) vazio na resposta (quando utilizado script para fazer a requisição).

A Tabela 5 relaciona as senhas testadas, com a resposta de retorno e o *status code*. Todas requisições retornaram um *status-code* 401, e uma mensagem de erro, com exceção apenas da senha correta. Assim como no teste 4.1.2, a senha correta recebeu *status* 200 (Ok) e houve uma resposta no formado JSON. O valor 1 significa que foi permitido o acesso.

Abaixo tem-se o código utilizado para o teste (A Tabela 5 representa o arquivo exportado pelo teste):

```
import json, time, dados
import pandas as pd
from teste_default import teste
from random import randrange,choice
from string import ascii_lowercase,digits
url = dados.url
tag = dados.tag
usuario = dados.usuario
senha = dados.senha
agente = dados.agente
random_password = lambda : ''.join(choice(ascii_lowercase+digits) for _ in range(ra
```

```
senhas = [senha]+[random_password() for _ in range(9)]
def teste_senha(url,usuario,senhas,tag,agente):
    print("Iniciando teste")
    status_list = [0]*len(senhas)
    resposta_list = [0]*len(senhas)
    for i,senha in enumerate(senhas):
        time.sleep(0.1)
        status_list[i],resposta_list[i] = teste(url,agente,usuario,senha,tag)
    df = pd.DataFrame({'Senhas':senhas,'Status':status_list,'Resposta':resposta
    return df
tabela_teste_senhas = teste_senha(url,usuario,senhas,tag,agente)
tabela_teste_senhas.to_csv('teste_senhas.csv', index=False)
```

4.1.4 Tags

O teste funcional de tag teve como intuito apenas a validação no servidor. As respostas consistem apenas em 1 para permissão concedida, e 0 para negada independente do motivo. Como este foi apenas um teste funcional, todas as informações de segurança estavam corretas, logo o *status-code* de resposta limita-se apenas a 200.

Para realizar o teste, foi adicionado em uma lista uma tag com cadastro existente e permissão independente do horário, assim como outras nove tags aleatórias sem permissão de acesso. Assim como esperado, na Tabela 6 tem-se, todas tags não cadastradas receberam resposta 0, e a tag permitida obteve como resposta o valor 1.

Tabela 6 – Testes de segurança em relação a *Tag* enviada por método *Post* na requisição para a API.

Tags	Status HTTP	Resposta
a1b2c3d4	200	{'status': '1'}
249b3c92	200	{'status': '0'}
26340996	200	{'status': '0'}
e33b18b5	200	{'status': '0'}
666ca073	200	{'status': '0'}
e51ae570	200	{'status': '0'}
8d2ac4bb	200	{'status': '0'}
4521677a	200	{'status': '0'}
2741ea42	200	{'status': '0'}
88ef2178	200	{'status': '0'}

Fonte: Produzido pelo autor.

Nota: O *status* 1 significa que foi autorizado o acesso e registrado já no sistema. O *status* 0 significa que o acesso não foi autorizado, e foi registrado nas tentativas inválidas.

Abaixo tem-se o código utilizado para o teste (A Tabela 6 representa o arquivo exportado pelo teste):

```
import json, time, dados
import pandas as pd
from teste_default import teste
from random import randrange,choice
from string import ascii_lowercase,digits
url = dados.url
tag = dados.tag
usuario = dados.usuario
senha = dados.senha
agente = dados.agente
random_tag = lambda : ''.join("%1x"%randrange(0,16) for _ in range(8))
tags = [tag]+[random_tag() for _ in range(9)]
def teste_tags(url,usuario,senha,tags,agente):
    print("Iniciando teste")
    status_list = [0]*len(tags)
    resposta_list = [0]*len(tags)
```

```
        for i,tag in enumerate(tags):
            time.sleep(0.1)
            status_list[i],resposta_list[i] = teste(url,agente,usuario,senha,tag)
        df = pd.DataFrame({'Tags':tags,'Status':status_list,'Resposta':resposta_lis
        return df
    tabela_teste_tags = teste_tags(url,usuario,senha,tags,agente)
    tabela_teste_tags.to_csv('teste_tags.csv', index=False)
```

4.2 Teste do circuito final

Por conta da atual situação pandêmica e do ensino remoto, não pode-se realizar os testes físicos com a placa instalada na sala do grupo de pesquisa GAMA. Para então permitir o teste físico do sistema proposto, foi desenvolvida outra placa com a mesma funcionalidade. A mesma foi encapsulada, como pode-se ver na Figura 24, e instalada em ambiente residencial.

Figura 24 – Protótipo feito durante a pandemia para os testes.

Fonte: o autor.

A Figura 25 demonstra um teste realizado, quando aproximada a tag o led amarelo acende indicando conexão com o servidor. Após a verificação e resposta do servidor, o led verde é aceso (Figura 26) e é acionado o atuador.

Figura 25 – Leitura do cartão e conexão com a internet para verificar permissão de acesso.

Fonte: o autor.

Figura 26 – Led verde ligado indicando de forma visual o acionamento do atuador e a permissão para a entrada.

Fonte: o autor.

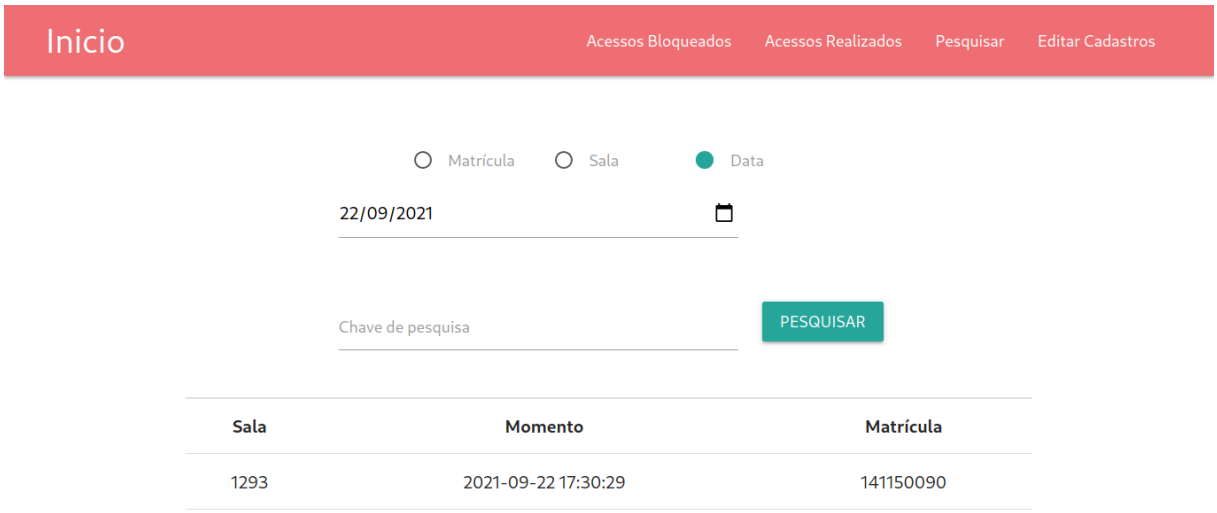
Para execução do teste, por ter em mãos apenas uma tag, alterou-se a permissão da mesma e executou-se o teste em horário permitido e fora do horário permitido. Alterou-se então o acesso para apenas as quartas-feiras, das 14:00 até as 18:00 (Figura 27), e testou-se primeiro as 17:30, e depois as 18:01.

Figura 27 – Captura de tela da permissão configurada para o teste.

Fonte: o autor.

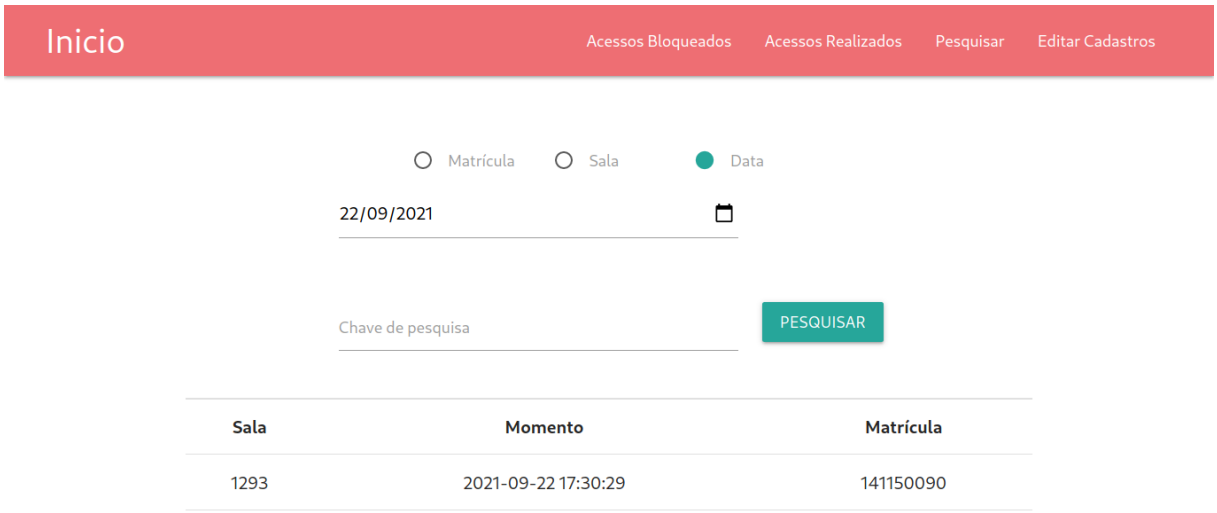
Quando testado as 17:30 o acesso foi liberado conforme o esperado (Figura 28). Já posteriormente, quando testado depois das 18:00, o acesso foi negado (Figura 29), demonstrando que o sistema está em correto funcionamento.

Figura 28 – Captura de tela do acesso realizado no teste.



Fonte: o autor.

Figura 29 – Captura de tela do acesso bloqueado no teste.



Fonte: o autor.

Nas Figuras 28 e 29 tem-se os registros dos testes, obtidos através da aplicação

gráfica.

Também foi testado o bloqueio do sinal WiFi utilizado pelo circuito, e quando requisitado acesso ele verificou a falta de conexão e sinalizou com piscadas rápidas através do led vermelho. Nestes casos o acesso se da por meio de chave convencional.

5 CONSIDERAÇÕES FINAIS

Diante do crescimento acelerado no desenvolvimento de dispositivos inteligentes, o setor de segurança tende a conquistar diversas melhorias quando utilizadas estas novas tecnologias. Entre as inúmeras situações benéficas do emprego de IoT, está o controle seguro e autônomo de acesso aos ambientes e salas tanto no meio acadêmico, quanto na maioria dos setores empresariais.

Em vista disso, esta pesquisa teve como objetivo o desenvolvimento de um sistema completo que garantisse acesso simples e seguro aos ambientes de laboratório e grupos de pesquisa da Universidade Federal do Pampa (Unipampa) - Campus Alegrete. Percebe-se então que o fim foi parcialmente atendido, dado que o sistema, consistindo em *software* e *hardware*, realiza a tarefa desejada, porém devido a pandemia, o sistema não chegou a ser testado *in loco*.

Por certo, a segurança em todas as etapas da elaboração foi um dos fatores mais importantes, e como tal, foi atendida, pois não falhou em nenhum teste realizado. Ademais, mesmo com as etapas de segurança e garantia dos dados, as requisições de acesso se mantiveram ágeis e funcionais.

Outro ponto de interesse com a produção do trabalho foi o registro dos incidentes, como acessos e tentativas indevidas. Esta etapa foi facilmente conquistada com o uso do banco de dados. Além disso, a aplicação gráfica para gestão do sistema garantiu facilidade tanto no controle de permissões de acesso (como inserção, atualização ou remoção de dados), como na visualização e aquisição dos dados e informações.

Diversos eventos foram limitadores diante as ideias e ambições iniciais, entre os quais o afastamento presencial e adaptação ao ensino remoto. Este impossibilitou testes massivos com alunos e professores, assim como aplicações em diversas salas e ambientes. Era de interesse também o cadastro do máximo de indivíduos possível, para poder verificar o comportamento do banco de dados e o tempo de resposta em um caso mais realista. Além disso, o afastamento comprometeu o uso dos materiais de laboratório, dificultando possíveis modificações e melhorias em termos de *hardware*.

É desejável, por fim, que futuramente sejam realizadas algumas modificações que não foram possíveis, como uma verificação no servidor de número anormal de acessos e seu possível bloqueio automático. Também seria interessante a criação de um procedimento de limpeza que periodicamente remova permissões não utilizadas mais, pois provavelmente são permissões de acesso antigas que foram esquecidas de serem removidas.

Outras duas sugestões são:

- Adição de uma verificação do endereço MAC requisitante para a API dos circuitos, porém atualmente em PHP isso não seria possível, necessitando o uso de outra linguagem.
- Modificar a estrutura de organização dos dados, de forma a separar os dias da semana, ou seja, em lugar de armazenar todos os dias permitidos em um *VARCHAR* que contém números concatenados representando os dias, armazenar uma linha de permissão por dia de acesso. Isso permitiria maior possibilidade de variação nos horários de acesso, limitando diferentemente para cada dia. Além disso, poderia representar uma redução no tempo de verificação dos dados.

Adicionalmente, foram publicados resumos e artigos deste projeto nos eventos SIEPE e SCIPROT. Também participou-se do Desafio Negócios Inovadores 2019, onde foi possível analisar os aspectos econômicos do projeto, e os impactos que traria ao mercado. Com isso, diversas possibilidades de ramificações e aplicações futuras surgiram, e pretende-se dar prosseguimento, tanto para a possibilidade de aplicação no Campus Alegrete, quanto para outros Campi ou Universidades que assim desejarem.

REFERÊNCIAS

ALBUQUERQUE, R. *Site do STF sofre ataque de "bots"de informação*. 2021. Site do STF sofre ataque de "bots"de informação. Disponível em: <https://safesrc.com/video/8288ff20-d03e-4268-89ac-0754cd571f43>. Acesso em: 19 ago 2021. Citado na página 21.

ALECRIM, E. *Após falha no MTE, dados de 9 milhões de brasileiros são liberados de graça*. 2021. Após falha no MTE, dados de 9 milhões de brasileiros são liberados de graça. Disponível em: <https://tecnoblog.net/425553/dados-9-milhoes-brasileiros-expostos-mte-forum/>. Acesso em: 19 ago 2021. Citado na página 22.

ALVARENGA, L. *Dataprev e INSS são acusados de vazamento de dados dos pensionistas, e agora?* 2021. Dataprev e INSS são acusados de vazamento de dados dos pensionistas, e agora? Disponível em: <https://fdr.com.br/2021/07/02/dataprev-e-inss-sao-acusados-de-vazamento-de-dados-dos-pensionistas-e-agora/>. Acesso em: 19 ago 2021. Citado na página 21.

BENTO, E. J. *Desenvolvimento web com PHP e MySQL*. [S.l.]: CASA DO CODIGO, 2014. Citado na página 40.

CARVALHO, V. *MySQL: Comece com o principal banco de dados open source do mercado*. [S.l.]: CASA DO CODIGO, 2015. v. 1. ISBN 978-85-5519-079-7. Citado 2 vezes nas páginas 39 e 43.

DERCOLES, R. *Falha do Detran-RS expõe dados de 5,1 milhões de motoristas*. 2021. Falha do Detran-RS expõe dados de 5,1 milhões de motoristas. Disponível em: <https://www.nextpit.com.br/falha-do-detran-rs-expoe-dados-de-5-1-milhoes-de-motoristas>. Acesso em: 19 ago 2021. Citado na página 22.

HOFFMAN, H. H. H. *Ethical Hacking With Kali Linux: Learn Fast How To Hack Like A Pro*. [S.l.: s.n.], 2020. Citado na página 45.

HUNT, J. *Java and Object Orientation: An Introduction*. 2. ed. [S.l.]: Springer-Verlag London, 2002. ISBN 9781852335694; 1852335696; 9781447101253; 1447101251. Citado na página 43.

KOCAN, K. E. D. K. F. Fpga technology to minimize extended life-cycle development. *Bell Labs Technical Journal*, v. 9, 2004. Citado na página 30.

MANSON, L. *Giant leak exposes data from almost all Brazilians*. 2021. Data exposed from almost all Brazilians. Disponível em: <https://www.databreaches.net/giant-leak-exposes-data-from-almost-all-brazilians/>. Acesso em: 19 ago 2021. Citado na página 21.

NXP SEMICONDUCTORS. *Data Sheet, Standard Performance MIFARE*. [S.l.], 2016. Rev. 3.9. Citado na página 28.

OPPLIGER, R. *SSL and TLS: Theory and Practice*. 2. ed. [S.l.]: Artech House Publishers, 2016. (Computer Security). ISBN 1608079988, 9781608079988. Citado 2 vezes nas páginas 44 e 45.

POTDAR et al. Performance evaluation of docker container and virtual machine. *Procedia Computer Science vol. 171*, v. 171, 2020. Citado na página 47.

SHIRRIFF, K. The surprising story of the first microprocessors. *IEEE Spectrum*, IEEE, v. 53, p. 48–54, 2016. ISSN 0018-9235. Citado na página 30.

TANENBAUM, A. S. *Redes de computadores*. [S.l.]: Editora Campus, 2003. (4). Citado na página 46.

UCKELMANN MARK HARRISON, F. M. D. *Architecting the Internet of Things*. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2011. ISBN 3642191568; 9783642191565. Citado na página 21.

VENTURA, F. *DataSUS é invadido de novo e hacker reclama: "continua uma b****"*. 2021. DataSUS é invadido de novo e hacker reclama: "continua uma b****". Disponível em: <https://tecnoblog.net/413354/datasus-e-invadido-de-novo-e-hacker-reclama-continua-uma-b/>. Acesso em: 19 ago 2021. Citado na página 22.

VENTURA, F. *Vazamento que expôs 220 milhões de brasileiros é pior do que se pensava*. 2021. Exclusivo: Vazamento que expôs 220 milhões de brasileiros é pior do que se pensava. Disponível em: <https://tecnoblog.net/404838/exclusivo-vazamento-que-expos-220-milhoes-de-brasileiros-e-pior-do-que-se-pensava/>. Acesso em: 19 ago 2021. Citado na página 21.

WELLING, L. *PHP E MYSQL: DESENVOLVIMENTO WEB, 3a EDIÇÃO*. 3. ed. [S.l.]: CAMPUS, 2005. ISBN 8535217142,9788535217148. Citado na página 40.