UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PEDRO VINÍCIUS BATISTA CLERICUZI

Desenvolvimento de um Sistema de Gerenciamento de Uso de Equipamentos em um Laboratório Baseado em RFID

Recife
2022

PEDRO VINÍCIUS BATISTA CLERICUZI

Desenvolvimento de um Sistema de Gerenciamento de Uso de Equipamentos em um Laboratório Baseado em RFID

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciências da Computação.

Área de Concentração: Engenharia da Computação.

Orientador: Profa. Dra. Edna Natividade da Silva Barros

Recife
2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C629d Clericuzi, Pedro Vinícius Batista
Desenvolvimento de um sistema de gerenciamento de uso de equipamentos de um laboratório baseado em RFID / Pedro Vinícius Batista Clericuzi. – 2022.
121 f.: il., fig., tab.

Orientadora: Edna Natividade da Silva Barros.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e apêndices.

1. Engenharia da Computação. 2. RFID. 3. IoT. 4. Cloud computing. 5. Controle de uso. I. Barros, Edna Natividade da Silva (orientadora). II. Título

621.39 CDD (23. ed.) UFPE - CCEN 2023 – 001

Pedro Vinícius Batista Clericuzi

“Desenvolvimento de um Sistema de Gerenciamento de Uso de Equipamentos em um Laboratório baseado em RFID”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Engenharia da Computação.

Aprovado em: 10/03/2022.

BANCA EXAMINADORA

_____

Prof. Dr. Adriano Augusto de Moraes Sarmento
Centro de Informática/UFPE

_____

Prof. Dr Victor Wanderley Costa de Medeiros
Departamento de Computação/UFRPE

_____

Profa. Dra. Edna Natividade da Silva Barros
Centro de Informática/UFPE
(**Orientadora**)

Dedico este trabalho à Anderson Paulo da Silva (in memorian), ex-doutorando do CIn e meu primeiro professor de programação, e à Angela Clericuzi (in memorian) que, mesmo longe, contribuiu para minha educação nos primeiros anos escolares.

AGRADECIMENTOS

Primeiramente agradeço à minha mãe que sempre esteve presente, nunca duvidou do meu potencial e fez de tudo para que eu pudesse ter todas as oportunidades que ela não teve. Agradeço também à Deborah Galhardo por todo o apoio, sobretudo durante o desenvolvimento desse trabalho.

À minha família e aos meus amigos que sempre me incentivaram e me deram todo o apoio nos momentos mais difíceis. Agradeço também a todos os meus professores que me ajudaram até aqui, sobretudo Alyson Campos que até os dias atuais me aconselha e me ajuda a tomar as melhores decisões.

Aos meus colegas de trabalho, em especial Zé, Mari, Tomoiti e Jayme que contribuíram diretamente para o meu desenvolvimento pessoal e profissional durante esse 1 ano de ZBRA. Não posso deixar de agradecer também ao Milton e ao Alê pela oportunidade de fazer parte desse time.

Ao Centro de Informática da UFPE, onde passei madrugadas e fins de semana realizando testes e pesquisas. A estrutura é incrível, não é à toa que o CIn é uma referência em ensino e pesquisa na área de computação em todo o Brasil. Tenho orgulho de poder dizer que eu estudo no CIn.

Aos meus professores do mestrado, pela atenção e pela dedicação em transmitir conhecimentos. Agradeço também à FACEPE pela oportunidade e incentivo concedido.

Agradeço a todos os envolvidos na minha pesquisa, com destaque a Lucas Amorim que sempre esteve disponível a me ajudar e sanar minhas dúvidas sobre a estrutura ou o projeto que originou seu TCC.

Por fim, eu não poderia deixar de agradecer à minha orientadora, Edna Natividade da Silva Barros, por toda a disponibilidade desde antes do início do mestrado até o presente momento, sempre acreditando no meu potencial e me dando todo o suporte necessário para conseguir seguir em frente. Teria sido muito mais difícil sem alguém como ela me orientando. Meu muito obrigado, professora!

RESUMO

Novas tecnologias voltadas para o controle no uso de bens têm surgido ano após ano para tornar o controle patrimonial cada vez mais ágil e confiável. Este trabalho teve como objetivo o desenvolvimento de um sistema de monitoramento de uso dos equipamentos de um laboratório usando a tecnologia RFID. O sistema foi desenvolvido a fim de possibilitar que os alunos do Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE) usem o laboratório de hardware a qualquer hora sem a supervisão de terceiros e os equipamentos contidos nele estejam sempre seguros e sob supervisão automatizada. No contexto atual isso não é possível, as baias são abertas apenas com chave e só quem tem acesso a elas são os monitores e funcionários. Isso é um problema porque o CIn funciona 24h todos os dias, sendo o laboratório de hardware um dos poucos que funcionam apenas em dias e horários específicos. A solução desenvolvida usa como meio de acesso às baias crachás com tags RFID, tecnologia essa que é amplamente utilizada no CIn. Para os equipamentos, também foram utilizados leitores RFID dentro das baias para garantir que os equipamentos estejam no local correto após seu uso. Todas as informações do monitoramento e log de uso ficam armazenadas em um sistema na nuvem, desenvolvido no contexto deste trabalho. O sistema em nuvem permite identificar os usuários dos equipamentos, enviar notificações de uso e efetuar reservas para uso do laboratório em horários não comerciais.

Palavras-chaves: RFID; IoT; cloud computing; controle de uso.

ABSTRACT

New technologies aimed to control the use of assets have emerged year after year to make asset control increasingly agile and reliable. This work aimed to develop a monitoring system for using equipment in a laboratory using RFID technology. The system has been developed to enable students from the Centro de Informática (CIn) of the Federal University of Pernambuco (UFPE) to use the hardware laboratory at any time without the supervision of third parties, and the equipment contained therein are always safe and under automated supervision. However, in the current context, this is not possible, the equipment cabinet is only opened with a key, and only those who have access to them are the monitors and employees. This situation is a problem because the CIn is open 24 hours a day, and the hardware laboratory is one of the few that only works on specific days and times. The solution developed here uses cards with RFID tags to access the equipment cabinet, a technology that is widely used in the CIn. For the equipment, RFID readers were also used inside the equipment cabinet to ensure that the equipment was in the correct place after use. All monitoring information and usage logs are stored in a cloud system, developed in the context of this work. The cloud system makes it possible to identify users' equipment, send usage notifications and make reservations for laboratory use during non-business hours.

Keywords: RFID; IoT; cloud computing; use control.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Controle de acesso é um termo bastante abrangente que é comumente usado para sistemas que limitam o acesso de pessoas em determinados ambientes, porém esse tipo de sistema também possibilita o controle de quem pode ou não ter acesso a compartimentos que guardam itens com um certo valor.

Sistemas de controle de acesso são uma evolução da forma tradicional, ou seja, que acontece usando chaves físicas que podem ser facilmente copiadas não oferecem a segurança que certos ambientes ou compartimentos necessitam. Com um sistema específico para isso os administradores não só podem transferir ou retirar permissões em frações de segundos como também pode acompanhar remotamente qualquer ocorrido em tempo real.

Mas às vezes, além de poder controlar o acesso também é preciso gerenciar o uso de bens dentro do determinado ambiente. O objetivo disso nada mais é do que acompanhar seu uso de modo que ações sejam tomadas em caso de detecção de alguma anormalidade, a fim de preservar a integridade do patrimônio.

Atualmente existem diversos produtos e tecnologias no mercado que facilitam o controle de acesso de usuários e gerenciamento de uso de bens, a tecnologia mais comum para a aplicação dessas soluções é o RFID (Radio Frequency Identification), que é baseado em *tags* com identificadores únicos, com um baixo custo financeiro e uma alta compatibilidade com plataformas de prototipação.

Em paralelo a isso o conceito do paradigma da Internet das Coisas (IoT) tem ganhado cada vez mais notoriedade sobretudo por conta dos avanços tecnológicos dos últimos anos e a necessidade de automatizar e otimizar tarefas rotineiras. A ideia deste paradigma é tornar possível a troca de informações entre sensores, atuadores e sistemas customizáveis desenvolvidos para atenderem às necessidades dos usuários. IoT não possui uma área de aplicação específica, inclusive diversas áreas de segmentos bem distintos já estão se beneficiando com a implantação de projetos baseados nisso, algumas delas são: agricultura, saúde, transporte, varejo e educação.

Segundo Evans (2011), a aplicação da Internet das Coisas representa a evolução da internet, dando um grande salto na capacidade de juntar, analisar e distribuir dados que podem ser transformados em informação, conhecimento e sabedoria.

O trabalho apresentado nesta dissertação tem como contribuição a integração de diversas tecnologias que juntas são capazes de controlar o acesso de pessoas e o gerenciamento de uso dos equipamentos em ambientes restritos e além de ter esse sistema funcional, ele também será escalável visando a evolução desse projeto em trabalhos futuros, como por exemplo a integração com o banco de dados do Centro de Informática.

Para o desenvolvimento do projeto foi levado em consideração a estrutura que já existe no laboratório (armário, cabos, compartimentos, etc). Além disso, foi levado em consideração também as limitações que essa estrutura poderia causar em relação aos requisitos, sendo a conectividade uma dessas limitações. Isso porque existe a comunicação entre sensores e um sistema

na nuvem e, pelo fato do microcontrolador estar dentro de um armário poderia ocorrer algum tipo de interferência, afetando o funcionamento do sistema e pondo em risco a integridade do patrimônio.

1.1 MOTIVAÇÃO

O Centro de Informática da UFPE além de ser referência em ensino, também conta com uma estrutura de alto nível com diversos laboratórios e computadores modernos disponíveis para alunos e funcionários 24 horas por dia, sete dias por semana, com a intenção de potencializar o desenvolvimento de todos.

Porém, embora seja um centro de referência em tecnologia com controles de acesso espalhados por suas dependências, nem todos os laboratórios dispõem de seus recursos para os alunos a qualquer momento, o laboratório de hardware é um exemplo. Este laboratório possui diversos equipamentos dentro de armários, onde também chamamos de baias, que só podem ser abertos por algum monitor e só podem permanecer em uso enquanto ele ou outro monitor estiver no laboratório, o que nem sempre é possível, restringindo o uso do laboratório aos horários comerciais.

Muito embora essa seja uma necessidade atual do CIn, durante a pesquisa do estado da arte ficou claro que essa é uma necessidade nos mais diversos âmbitos, sendo universidades, hospitais, centros comerciais industrias entre muitos outros. Alguns exemplos são os trabalhos de Freitas (2020) e Day (2016), onde ambos buscam automatizar o monitoramento e o gerenciamento no uso de equipamentos. O primeiro trabalho tem foco no controle dentro de um departamento de uma universidade pública e o segundo dentro de em uma empresa emissora de rádio e TV.

Este trabalho tem como motivação a criação de um sistema capaz de controlar o uso das baias e monitorar a presença dos equipamentos presentes nas baias do laboratório de hardware do Centro de Informática. A ausência de um sistema como esse limita os alunos a usarem os equipamentos apenas em momentos em que há um monitor presente no laboratório, isso atrapalha tanto quem quer aplicar os conceitos explorados em aula, como os próprios monitores que, por também serem alunos, também possuem demandas de outras disciplinas e eventualmente podem ficar sobrecarregados.

1.2 OBJETIVOS

1.2.1 Objetivos gerais

O objetivo geral deste trabalho foi desenvolver um sistema que realize o controle de acesso às baias e o gerenciamento de uso dos equipamentos presentes nas baias do laboratório de hard-

ware do Centro de Informática da Universidade Federal de Pernambuco (UFPE). Esse sistema desenvolvido realiza a coleta dos dados dos usuários e dos equipamentos via RFID, que é uma tecnologia amplamente utilizada no CIn, e realiza a comunicação com um banco de dados na nuvem para poder validar as informações enviadas, bem como para armazenar e atualizar as informações de uso das baias e dos equipamentos nela contidos.

Esses dados podem ser visualizados em um portal exclusivo para alunos e funcionários que terão acesso em tempo real às informações relacionadas às baias e aos equipamentos, emitindo alertas para os responsáveis do laboratório sempre que houver alguma anormalidade.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Desenvolver um sistema embarcado capaz de controlar o acesso às baias e gerenciar os equipamentos presentes nela;
- Implementar um *gateway* que se conecte ao sistema embarcado por meio de uma rede LAN e seja capaz de coletar, processar, armazenar dados e enviá-los para a nuvem via conexão Wi-Fi;
- Desenvolver um *backend* na nuvem que registre as solicitações feitas pelo usuário, que emita alertas em caso de extravio de equipamento e sirva de apoio para tomadas de decisões;
- Criar um *frontend* com uma interface amigável para acompanhamento de uso das baias e do sistema de modo geral, em tempo real.

1.3 ORGANIZAÇÃO DO TRABALHO

Após a introdução feita neste capítulo, serão apresentados no Capítulo 2 a teoria e os fundamentos básicos sobre Internet das Coisas e RFID.

No Capítulo 3, serão apresentados alguns trabalhos que tenham alguma relação com os objetivos propostos neste trabalho.

No Capítulo 4, serão apresentados os requisitos do sistema, os casos de uso, a arquitetura e as conexões entre os sistemas.

No Capítulo 5, serão apresentados os materiais, métodos e as tecnologias utilizadas no desenvolvimento do projeto.

O Capítulo 6 apresenta a metodologia aplicada no desenvolvimento do projeto, a implementação, as configurações feitas, os módulos usados e como foram feitos os tratamentos dos dados nos sistemas.

No Capítulo 7, são apresentados os testes e resultados detalhando as simulações feitas em laboratório e as simulações no *gateway*.

O Capítulo 8 é destinado para as considerações finais deste trabalho, bem como perspectivas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos importantes de tecnologias exploradas em capítulos futuros deste trabalho, com a finalidade de oferecer ao leitor um maior embasamento teórico.

2.1 INTERNET DAS COISAS

Internet das Coisas, ou simplesmente IoT, é um termo usado para um paradigma que vem ganhando força nos últimos anos graças aos recentes avanços tecnológicos e os vários benefícios vivenciados tanto no cotidiano das pessoas quanto no das organizações. De forma sucinta, o termo IoT se refere a "coisas"como objetos conectados à internet com o poder de conversar com outras objetos a fim de resolver um problema específico.

Inclusive, a origem desse paradigma surgiu no final do século XX quando o pesquisador Kevin Ashton buscava alternativas para realizar transações comerciais conectando dados das *tags* RFID à internet. Segundo ele, se os computadores fossem capazes de coletar informações sobre as coisas em geral o número de perdas e desperdícios seriam reduzidos drasticamente.

Para Coetzee e Eksteen (2011), IoT tem potencial para impactar no meio social, ambiental e econômico, pois permite decisões inteligentes e tomadas de decisões apropriadas baseadas em informações precisas sobre status, localização e identidade das coisas que fazem parte e impactam no ambiente.

Segundo Borgia (2014), um objeto inteligente é o bloco de construção da visão IoT. Ao colocar inteligência em objetos do dia-a-dia, eles se tornam objetos inteligentes capazes não apenas de coletar informação de um ambiente e interagir/controlar o mundo físico, mas também estar interconectados uns com os outros para trocar dados e informações através da internet.

Um exemplo prático e simples de como esses objetos inteligentes podem impactar no nosso dia-a-dia seria uma fechadura inteligente, onde é possível abrir e fechar uma porta remotamente desde que a mesma consiga se comunicar com um sistema externo e possua sensores e atuadores adequados para realizar essa tarefa.

Para Santos et al. (2016), a maioria de sistemas de IoT incluem sensores que coletam informações sobre o contexto onde os objetos se encontram e, em seguida, armazenam/encaminham esses dados para *data warehouse*, *clouds* ou centros de armazenamento. Atuadores podem realizar ações no ambiente ou reagir de acordo com os dados lidos.

Segundo Byers (2019), a maioria dos sensores e atuadores incluem tipos de transdutores. Transdutores são dispositivos que convertem uma forma de energia em outra. Para sistemas de IoT, quase todos os sensores usam alguns parâmetros físicos e os transformam em sinais elétricos. Da mesma forma, quase todos os atuadores nos sistemas de IoT captam sinais elétricos e os convertem em algum tipo de saída física.

Algumas vezes é possível fazer com que esses sensores influenciem de forma direta nas

ações dos atuadores, porém existem casos onde os valores capturados pelos sensores precisam passar por um processo de tratamento de dados em outros dispositivos, ou na nuvem, e para isso é necessário que haja uma comunicação adequada para cada situação.

Para que toda a comunicação seja realizada sem maiores problemas é muito importante que seja implementada a rede que melhor satisfaz às necessidades do projeto. Por exemplo, não há muito sentido adotar uma rede *ethernet* em um sistema onde todos os dispositivos estão em constante movimento. A tabela 1 mostra algumas das principais redes utilizadas em projetos de sistemas IoT, juntamente com informações da conectividade, os prós e os contras, além dos usos mais comuns para essas redes.

Tabela 1 – Algumas das redes usadas em projetos IoT

Rede	**Conectividade**	**Prós**	**Contras**	**Casos que se aplicam**
Ethernet	Com fio	Alta velocidade e segurança	Alcance limitado ao tamanho do fio e mobilidade também limitada	Câmeras de vídeo e equipamentos fixos
WiFi	Sem fio (curto alcance)	Alta velocidade e boa compatibilidade	Alcance limitado e alto consumo de energia	Casas inteligentes e dispositivos que podem ser recarregados facilmente
NFC	Sem fio (ultra curto alcance)	Confiabilidade e baixo poder de consumo	Alcance limitado e falta de disponibilidade	Sistemas de pagamento e casas inteligentes
BLE	Sem fio (curto alcance)	Alta velocidade e Baixo consumo de energia	Alcance limitado e alta latência	Pequenos dispositivos domésticos, acessórios e beacons
LPWAN	Sem fio (longo alcance)	Baixo consumo de energia e longo alcance	Baixa largura de banda e alta latência	Cidades e casas inteligentes
ZigBee	Sem fio (curto alcance)	Baixo consumo de energia e escalabilidade	Alcance limitado e problemas de conformidade	Automação residencial, assistência médica e locais industriais

Redes de Celular	Sem fio (longo alcance)	Alta velocidade, confiabilidade e cobertura global	Alto custo e alto consumo de energia	Drones enviando vídeos e imagens

Fonte: Altexsoft (2020)

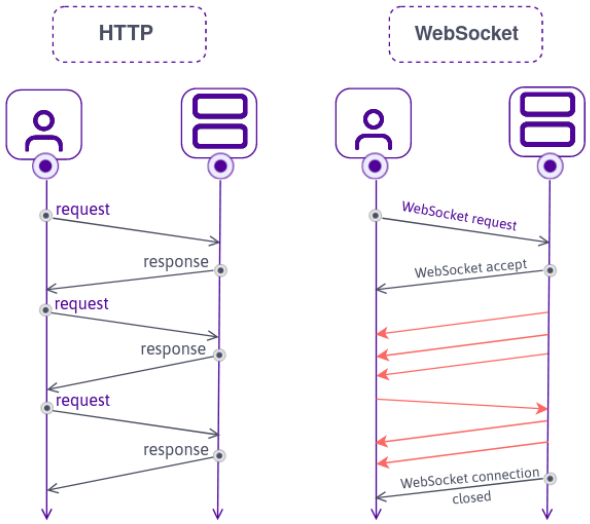
Além das redes, os protocolos responsáveis por realizar as trocas de mensagens entre os dispositivos também merecem uma atenção especial. Segundo Dizdarević J. (2019), os principais desafios para engenheiros de sistemas é escolher o protocolo apropriado e específico para cada requisito de um sistema IoT.

Dentre os vários protocolos existentes, vale destacar dois protocolos que são bastante usados, e assim, importantes para sistemas IoT, o HTTP e o WebSocket. A compreensão deles e de suas diferenças podem ajudar a entender melhor algumas informações em alguns capítulos dessa dissertação. Ambos são usados em diferentes partes do projeto desenvolvido e de acordo com Almeida (2019), o protocolo HTTP é a base de todas as comunicações em aplicações web existentes entre cliente-servidor. Este protocolo é definido como um tipo de comunicação caracterizado por uma troca de mensagens entre ambas as partes, através de um processo *request-response*. Ainda segundo Almeida (2019), o protocolo WebSocket, por outro lado, é definido como o protocolo que permite a comunicação bidirecional entre uma aplicação cliente num ambiente controlado e um servidor remoto.

A figura 1 ilustra a diferença entre esses dois protocolos: o HTTP, do lado esquerdo, para todas as requisições do cliente existe uma resposta do servidor; o WebSocket, do lado direito, segue o princípio que a partir do momento que é estabelecido um canal de comunicação entre cliente e servidor as mensagens podem ser transmitidas simultaneamente em ambos os sentido, sem uma ordem específica, e esse processo só é interrompido quando o canal de comunicação é encerrado.

Neste trabalho o WebSocket é usado na comunicação entre o módulo Baia e o *gateway* e o HTTP na comunicação do *gateway* e do *backend*. Em relação ao WebSocket, embora existam outras alternativas para uma comunicação bilateral entre cliente e servidor, o MQTT por exemplo seria uma alternativa até mais adequada em questão de performance, a escolha do WebSocket se sucedeu apenas por uma maior familiaridade de implementação e a facilidade de conseguir estabelecer a comunicação entre ambas as partes. Os testes apresentaram bons resultados com o WebSocket e, por isso, essa tecnologia foi mantida no projeto.

Figura 1 – Protocolos HTTP e WebSocket

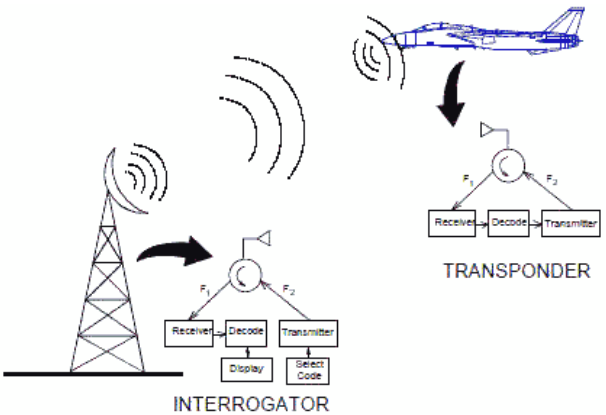


Fonte: Altexsoft (2020)

2.2 RFID

A ideia da tecnologia de identificação por radiofrequência (Radio Frequency Identification - RFID) foi concebida, ou pelo menos teve seu primeiro uso conhecido, na segunda guerra mundial e surgiu para possibilitar aos alemães a identificação das aeronaves aliadas, facilitando assim a distinção delas para as aeronaves inimigas. Cafe (2011) conta que a tecnologia se chamava *Identification - Friend or Foe*, ou simplesmente IFF, e cada aeronave tinha seu código exclusivo que era enviado para uma antena via ondas de rádio, como mostra a figura 2 e era dessa forma que eram feitas as identificações.

Figura 2 – Transponder IFF



Fonte: Cafe (2011)

Entretanto, embora tenha tido uma implementação inicial antes de meados do século XX, segundo Bertoncello (2018), a primeira etiqueta ativa de RFID recebeu a patente em janeiro de 1973, mesmo ano em que Charles Walton inventou o cartão magnético para abertura de portas

em hotéis sem necessitarem de chaves. A fechadura registrava os códigos contidos nos cartões e o receptor liberava a trava.

A partir de então, vários estudos e aplicações envolvendo a tecnologia RFID se tornaram frequentes na indústria. Pedroso M. C. e Souza (2009 apud MCKINSEY, 2003) citam que a utilização de RFID no varejo pode proporcionar várias melhorias, tais como oportunidades de aumento no faturamento (por meio de menores rupturas nas gôndolas e melhor planejamento de promoções), menores custos de distribuição (na forma de redução dos custos logísticos e das perdas) e menores custos de operação de loja.

Atualmente o cenário demonstra alguns avanços, tanto nas formas como essa tecnologia é utilizada, como nos diversos segmentos que ela é aplicada. Os correios, por exemplo, aderiram o uso do RFID para auxiliar no rastreamento das encomendas, e para que esse processo não se torne um gargalo, as leituras são feitas em grandes quantidades de maneira simultânea e numa distância que pode chegar até 10 metros, tornando todo o processo mais ágil e seguro. Outro uso que vem se tornando muito comum é o pagamento por aproximação, ou seja, atualmente não é mais obrigatório inserir o cartão na máquina para efetuar um pagamento, basta apenas aproximar o cartão ou o celular que o pagamento é computado.

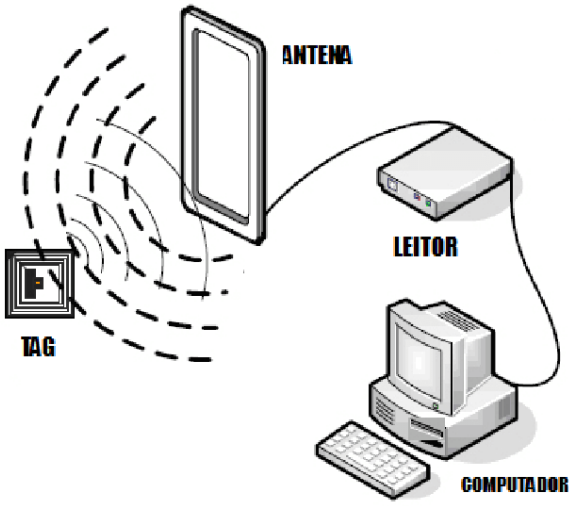
Mas para que tudo isso seja possível, é necessário de alguns componentes com tecnologias específicas para armazenar dados, transmiti-los via ondas de rádio em frequências específicas, e fazer a leitura correta desses dados. A figura 3 mostra os principais componentes para realizar tais tarefas e cada um é responsável por alguma tarefa dentro do processo que, segundo Liukkonen (2015), são dadas por:

- Tag: um microcircuito que inclui um código único e/ou outra informação;
- Antena: emite ondas de rádio e a tag envia de volta uma resposta com os dados contidos nela por meio dessa onda;
- Leitor: a antena é conectada em algum sistema de gerenciamento de informação que pode ser usado para processar a informação incluída na tag.
- Computador: mais voltado para o usuário final, identifica os dados e aplica para o uso que lhe é proposto.

Como foi dito no parágrafo anterior existem frequências específicas de operação, pois segundo Freitas (2020), as aplicações em RFID estão restritas apenas em escalas de frequências que foram reservadas. A forma de transmissão do sinal de RFID é particular para cada faixa de frequência, implicando equipamentos distintos para as diferentes faixas.

Ainda segundo Freitas (2020 apud FINKENZELLER, 2010), essas frequências eletromagnética são utilizadas para comunicação e obtenção da alimentação pelas tags/etiquetas. Atualmente existem quatro faixas de frequências que, Freitas (2020 apud FINKENZELLER, 2010) classificam em:

Figura 3 – Componentes do RFID



Fonte: Rodrigues, Borges e Barwaldt (2017)

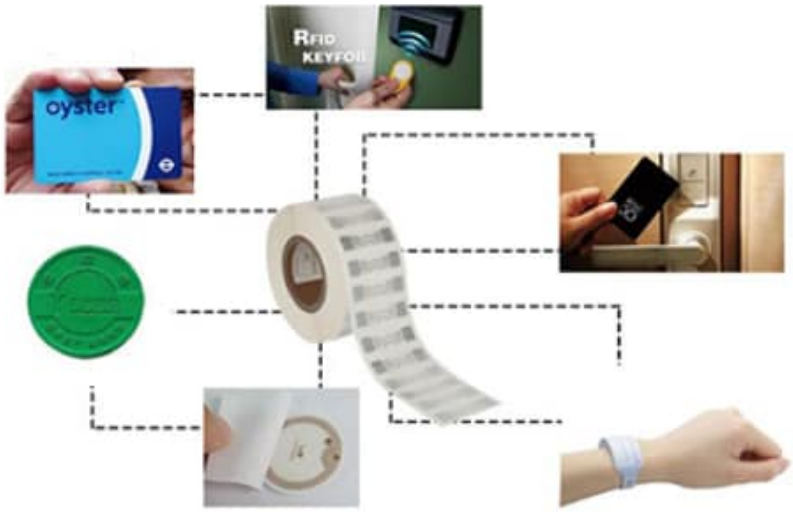
1. Baixa Frequência (LF – Low Frequency): faixa de frequência de 30 a 300 kHz. Nesta faixa a transferência de dados é bem lenta e o alcance de leitura é pequeno sendo de até 1 metro;
2. Alta Frequência (HF – High Frequency): faixa de frequência de 3 a 30 MHz. Geralmente, as informações nessa faixa podem ser lidas em até 1 metro de distância. Nesta faixa a transmissão é mais rápida que em faixas de baixa frequência, mas também consomem mais energia;
3. Ultra Alta Frequência (UHF – Ultra High Frequency): faixa de frequência de 300 MHz a 3 GHz, e tipicamente operam entre 866 e 960 MHz. *Tags* UHF possuem taxas de transferência mais altas e maior alcance do que as tags de alta e baixa frequência. No entanto, as ondas de rádio, nesta frequência, não passam por itens com alto teor de água. Em comparação às *tags* de baixa frequência, as *tags* UHF são mais caras e utilizam mais energia;
4. Micro-ondas: faixa de frequência acima de 3 GHz. *Tags* micro-ondas têm taxas de transferência muito altas e podem ser lidas a longas distâncias, contudo, elas usam uma grande quantidade de energia e são mais caras em comparação às demais.

Conhecendo todas essas faixas de frequência é possível concluir que cada uma é voltada para resolver problemas distintos e no caso do projeto desenvolvido neste trabalho foi utilizada a tag com uma frequência de 13.56MHz, que se enquadra na faixa classificada como *High Frequency*.

Como mostra a figura 4 essas *tags* podem ser encapsuladas de diferentes maneiras dependendo do uso que será destinado a ela.

Entretanto, nem todas as *tags* podem ser encapsuladas da mesma forma, isso porque podem existir três tipos diferentes de *tags*, classificadas por Cunha (2016) como:

Figura 4 – Etiquetas RFID



Fonte: Soluções (2020)

1. Passivas: Não necessitam de alimentação interna, sua energia vem do próprio sistema de leitura, através de indução magnética ou campo eletromagnético. Seu alcance típico dificilmente ultrapassa os 5 metros. Essas *tags* são as mais comuns e amplamente utilizadas por um simples motivo: o custo que, segundo Ivy e Cheatwood (2022), cada *tag* pode variar entre U$0.07 e U$0.50.
2. Semi-passivas: Têm uma fonte de alimentação interna (bateria), apenas para a recepção de dados. Isso permite que sejam lidas sem a energia do leitor externo, desta forma, podem ser usadas em ambientes com uma potência muito baixa de campo magnético. Isto reduz a quantidade de energia necessária para o sistema funcionar e também as interferências externas ao sistema. Como tem uma bateria interna, seu alcance pode chegar a 100 m. São mais caras e isso que dificulta o uso em larga escala, dependendo da aplicação. O preço por tag varia entre U$10 e U$25 dólares, também segundo Ivy e Cheatwood (2022).
3. Ativas: Tem uma fonte de energia interna (bateria) e um transmissor. O alcance, neste caso, pode chegar a alguns quilômetros. Essas *tags* são muito caras que as anteriores e são utilizadas apenas em sistemas específicos. O investimento mínimo por cada *tag*, segundo Ivy e Cheatwood (2022), pode partir de U$25 dólares e ultrapassar os U$100 dólares.

As mais comuns de serem utilizadas no mercado de fato são as *tags* passivas e ativas. As passivas estão mais presentes no nosso cotidiano, podemos citar como algumas de suas aplicações a identificação de itens e controle de acesso. As *tags* ativas, por sua robustez, podem ser usadas em aplicações que necessitam realizar leituras em longas distâncias como em construções, industrias e empresas logística.

Para facilitar o entendimento da diferença entre as duas a tabela 2 compara os principais aspectos a serem levados em consideração no momento da escolha das tecnologias.

Tabela 2 – Características Etiquetas Ativas e Passivas

Ativas	**Passivas**
Alimentadas por baterias	Alimentadas pelo leitor
Centenas de kb de memória	64 – 256 bits de memória
Custo alto	Custo baixo
Vida útil da bateria limitada	Vida útil mais longa
Transmissão Ativa	Transmissão reativa
Longo Alcance na leitura	Curto/Médio Alcance na leitura
Tag Talks First (TTF) Tag "fala" primeiro	Reader Talks First (RTF) Leitor "fala" primeiro

Fonte: Oliveira (2013)

Para um melhor entendimento da tecnologia RFID será destalhado, a seguir, como acontece o envio de dados entre a *tag* e dispositivo de leitura. As antenas emitem um campo magnético identificado pelas *tags*, que por sua vez respondem com uma alteração nesse mesmo campo magnético e essa alteração é convertida em informação, enviada para o leitor que irá processá-la e direcionar para algum sistema acessível ao usuário.

Assim como as *tags*, as antenas também possuem comportamentos distintos para cada faixa de frequência e até mesmo a direção dessas *tags*. Isso porque o comportamento dessas antenas estão ligados a algumas grandezas físicas que, de acordo com Freitas (2020), as principais são: polarização, diretividade e ganho.

Quanto à polarização, segundo Montalvão (2016), quando esta direção é constante no tempo, a onda é considerada linearmente polarizada. Entretanto, a antena é circularmente polarizada quando atua nos casos onde a orientação das *tags* em relação ao leitor não possui um padrão.

Segundo Freitas (2020 apud BROWN, 2006), a diretividade é a característica física da antena que define a direção de propagação do sinal de radiofrequência ou RF emitido pela mesma, por meio desta análise é definida a área de cobertura das antenas. Podendo assim as antenas serem classificadas da seguinte forma:

1. Antenas diretivas: são aquelas em que o campo de RF é propagado em um feixe bem restrito, com maior intensidade em uma direção do que em outra, delimitando uma pequena área. Sendo assim, a *tag* e antena ficam em posições predefinidas;
2. Antenas omnidirecionais: são aquelas em que o campo de RF tem propagação em 360 graus em relação à antena, ou seja, o sinal RF é propagado em todas as direções, di-

minuindo assim sua intensidade e facilidade de instalação. Sendo assim, a *tag* e antena podem estar em qualquer posição, mas com distâncias reduzidas entre elas;

3. Antena painel: aquela utilizada pela grande maioria de sistemas RFID, propagam sinal de RF com ação do campo em aproximadamente 160 graus, usada em montagem de portais.

A última grandeza física é o "ganho", que tem como unidade de medida o *dB* e de acordo com Teixeira (2017 apud DOBKIN, 2008) está relacionada com a diretividade da antena, ou seja, sua capacidade de transmitir a maior parte da energia recebida em uma direção específica. A vantagem de concentrar a maior parte da energia transmitida em uma única direção é o aumento do alcance de leitura da antena. Desse modo, *tags* mais distantes poderão ser identificadas na direção de maior radiação da antena.

No contexto do projeto desenvolvido neste trabalho o que melhor se enquadra nas características necessárias para um bom funcionamento do sistema proposto é o uso do módulo RFID MFRC522, apresentado na figura 5, que possui um leitor e uma antena integrada. Este módulo funciona numa faixa de frequência de 13.56MHz e é compatível com diversas plataformas baseadas em microprocessadores, incluindo o Arduíno. A *shield* para o Arduíno possui a interface SPI para comunicação, o que significa que é possível conectar mais de um leitor na mesma placa.

Figura 5 – Leitor MFRC522

Fonte: Rodrigues, Borges e Barwaldt (2017)

3 TRABALHOS RELACIONADOS

Este capítulo apresenta algumas análises de trabalhos com propostas de projetos semelhantes ao que foi desenvolvido neste e esses trabalhos são utilizados para fins de comparação.

3.1 SISTEMA DE MONITORAMENTO DO USO DOS EQUIPAMENTOS NO LABORATÓRIO DE HARDWARE BASEADO EM RFID

O trabalho de Amorim (2020) serviu como base para a esta dissertação, sendo assim, existem vários pontos em comum, incluindo o objetivo principal, que é automatizar a abertura das baias através dos crachá RFID e monitorar o uso de equipamentos. Esse trabalho foi bastante útil para compreender os problemas enfrentados na fase inicial e o que precisava ser melhorado para estar apto a se tornar um produto final.

O sistema desenvolvido utilizou como microcontrolador do módulo da baia o ESP32, pois a ideia era fazer com que a comunicação com o banco de dados na nuvem fosse feita inteiramente através de uma rede sem fio, todas as informações referentes às baias, os equipamentos e os usuários estão armazenadas nesse banco de dados.

Como foi dito anteriormente, esse sistema serviu de base para o projeto aqui desenvolvido e as funcionalidades de abrir e fechar baia possuem um fluxo de funcionamento muito semelhante, se diferenciando apenas em como ambos são implementados. Por exemplo, as estruturas de banco de dados são bem distintas, no trabalho de Amorim (2020) toda a infraestrutura já estava configurada, precisando apenas criar a lista dos tipos de dados que seriam incluídos. O problema desse serviço é que, além de deixar o projeto dependente de um sistema externo, a partir de um certo volume de dados seria necessário pagar e desde o princípio a intenção do presente trabalho foi fazer justamente o oposto, ou seja, criar uma estrutura própria, com o mínimo de dependências externas possíveis e com o menor custo financeiro.

Além disso, o sistema de Amorim (2020) não possui uma interface gráfica para os usuários e nem permite que certas regras de negócio sejam aplicadas, como a de precisar fazer uma reserva de baia fora de horário de aula. Além disso, outra distinção entre ambos os sistemas é em relação ao fluxo dos dados, como foi dito anteriormente o sistema desenvolvido no trabalho de Amorim (2020) realiza uma comunicação direta com o banco de dados na nuvem e não possui nenhuma forma de backup e nenhum tratamento em caso de falta de internet.

O fluxo que os dados percorrem também são diferentes, no trabalho de Amorim (2020) o dado vai direto do microcontrolador até o banco de dados na nuvem, no projeto desenvolvido nesta dissertação a comunicação entre os dois extremos é intermediada por um *gateway* que cria uma rede LAN e recebe os dados do módulo da baia independente se há ou não uma conexão com a internet.

3.2 CONTROLE E SEGURANÇA PATRIMONIAL POR RFID NO DEPARTAMENTO ACADÊMICO DE ENGENHARIA ELETRÔNICA DA UTFPR CAMPUS CAMPO MOURÃO

O trabalho desenvolvido por Freitas (2020) teve como objetivo implementar um sistema de segurança dos equipamentos patrimoniados dentro do almoxarifado do Departamento Acadêmico de Eletrônica por meio de um sistema de identificação baseado em RFID com o intuito de monitorar a entrada e saída dos equipamentos.

O sistema desenvolvido utilizou como hardware um microcontrolador ESP8266 e um leitor RFID, além disso uma *tag* RFID foi associada a cada equipamento para que o mesmo possuísse um identificador único e facilitando o controle do uso.

O projeto também inclui um sistema com uma interface Web que consiste em um banco de dados desenvolvido em *MySql* e uma interface simples para visualizar os *logs* do sistema e desativar possíveis alarmes causados pelo fato da *tag* não estar cadastrada ou simplesmente ela não estar liberada para retirada do equipamento.

Embora possua um banco de dados em um sistema web que guarda os logs e as *tags* cadastradas, o microcontrolador sempre faz as primeiras leituras e escritas na sua memória flash, a EEPROM. De forma prática, após a *tag* ser lida o microcontrolador verifica na sua memória flash se essa *tag* está cadastrada ou liberada pelo sistema, caso não esteja um alarme é acionado e um log é gravado na mesma memória flash. Esse log só vai para o banco de dados na web quando a rede WiFi estiver conectada e até que isso aconteça o alarme continua sendo disparado. Independentemente do alarme, o processo de leitura das *tags* continua ocorrendo normalmente e o fluxo segue da mesma maneira.

Quanto aos resultados, o principal objetivo que era detectar e desativar alertas foi alcançado, o sistema web desativa esses alarmes, cadastra e exibe os logs salvos anteriormente na EEPROM. Entretanto, esse sistema não foi concluído e realiza essas tarefas citadas.

O sistema desenvolvido nesse trabalho apresenta algumas desvantagens, são elas:

- **Controle de acesso:** esse sistema controla apenas a entrada e saída dos equipamentos do almoxarifado e não há uma identificação prévia de quem retirou esse equipamento, a única forma de identificação são câmeras de segurança do departamento;

- **Armazenamento:** os dados são centralizados na memória flash do microcontrolador, isso é um fator limitante pois essa memória flash tem um limite de quantidade de gravações de registros e os dados gravados nela são apagados após alguns anos;

- **Conexão:** A conexão utilizada é o WiFi, que não seria indicada para se usar na comunicação entre todas as camadas desse meu trabalho pois o microcontrolador é colocado dentro de um armário e rodeado por equipamentos metálicos e isso poderia causar instabilidades constantes na comunicação partido do Arduíno;

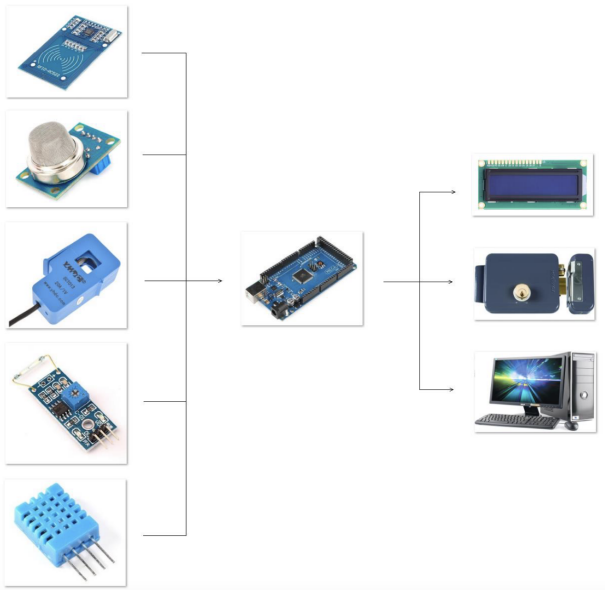
- **Interface web:** Não é possível fazer uma reserva e nem ter acesso à listagem dos equipamentos cadastrados. Além disso, só é possível acompanhar o status dos equipamento por meio do log, não existe uma página específica para isso.

3.3 AUTOMAÇÃO DE SEGURANÇA E CONTROLE DE ACESSO DE SALA EM EMISSORA DE RÁDIO E TV

O trabalho desenvolvido por Day (2016) teve como objetivo o desenvolvimento de um sistema de controle de acesso, de monitoramento de uso de equipamentos e do ambiente onde esses equipamentos estão localizados a fim de aumentar a segurança desse ambiente.

A figura 6 mostra a arquitetura do sistema que pode ser dividida em três partes: os sensores e o microcontrolador; a rede; e o sistema de monitoramento. Quanto ao funcionamento é possível observar como essas partes estão interligadas entre si por meio do apêndice **??**. É interessante ressaltar o protocolo utilizado para a comunicação entre o microcontrolador e o ScadaBr, o MODBus, que é amplamente utilizado por vários fabricantes em diferentes segmentos industriais para conectar um computador a terminais remotos e sistemas Supervisórios de Controle e Aquisição de Dados (SCADA).

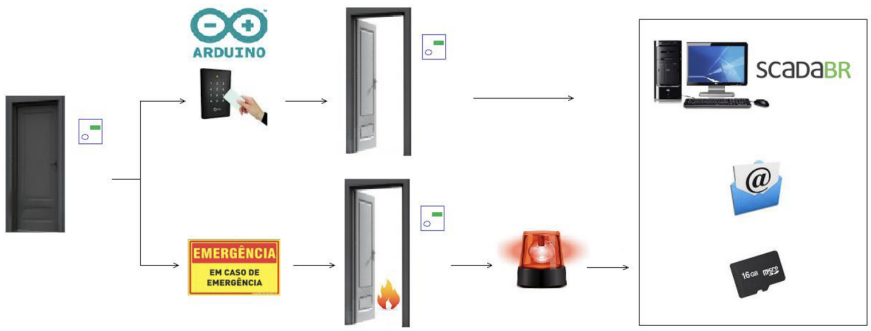
Figura 6 – Arquitetura proposta no projeto de Day (2016)



Fonte: Day (2016)

Dados os critérios de custo, compatibilidade e segurança, no caso da fechadura localizada externamente à sala dentro de uma caixa, foi feito o levantamento com três possíveis soluções onde todas elas possuíam sensores e atuadores com as mesmas finalidades, só mudando suas respectivas especificações. Após escolhidos os sensores, mostrados na figura 6, foi desenvolvido o circuito e implementado um sistema que tem o seu funcionamento ilustrado na figura 7.

Figura 7 – Funcionamento da solução de Day (2016)



Fonte: Day (2016)

A solução consiste em um sistema que controle o fluxo de entrada e saída com detecção de possíveis tentativas de burlar esse controle. Além disso dentro da sala ainda há o controle da corrente elétrica, de gás inflamável e fumaça, umidade e temperatura para, em caso de acidente, a porta seja aberta automaticamente. Para salvar os logs dos eventos, diferente do trabalho anterior que optou por salvar os dados na memória interna do microcontrolador, este utiliza um cartão de memória que é alimentado com os registros de entrada ou com os possíveis registros de acidente, esses logs também são enviados para o sistema de monitoramento ScadaBr para que sejam consultados localmente a qualquer momento.

Os resultados mostram um funcionamento esperado para a abertura da porta, com indicações visuais e sonoras, registro do log e alarme em caso de abertura incorreta e envio de relatório para os e-mails cadastrados através do sistema de monitoramento também foi realizado com êxito. Em relação aos demais sensores, o de temperatura e umidade tiveram uma margem de erro de 1 a 2ºC e 10%, respectivamente e o de gás e fumaça não foram testados por falta de um equipamento de medição com maior confiabilidade.

Algumas desvantagens foram identificadas, são elas:

- **Gerenciamento de uso dos equipamentos:** existe um cuidado em monitorar o ambiente mas não há algum sistema de gerenciamento de entrada e saída dos equipamentos da sala;
- **Armazenamento:** os dados são centralizados em um *SD Card* e isso é um fator limitante pois a biblioteca do Arduíno só permite que apenas 4GB de dados sejam lidos mesmo que a capacidade desse *SD Card* seja maior. Além disso essa não é uma forma adequada de armazenar dados que serão consumidos posteriormente com a finalidade de autenticação;
- **Acesso remoto:** esse sistema não permite nenhum tipo de configuração ou visualização remota, ele funciona conectado diretamente em um computador;

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

Embora este trabalho não tenha nenhuma relação com controle de uso equipamentos, a utilização de um RFID como uma forma de controle de frequência e toda a interação com um banco de dados na nuvem cria uma relação com o presente trabalho que está sendo desenvolvido e traz uma visão diferente de comunicação entre o hardware e a nuvem.

Sabendo disso, o objetivo do trabalho de SILVA (2018) é desenvolver um sistema de apoio aos professores com o intuito de registrar a presença dos alunos e diminuir falhas que são comuns de acontecer quando o processo é feito de forma manual.

O hardware utilizado foi um ESP8266 e um leitor RFID RC522 e sua arquitetura se assemelha ao trabalho da seção 3.3 onde possui apenas o sensor, a camada de rede e a central de controle e monitoramento, a diferença é que neste trabalho essa central é na nuvem e pode ser acessada de qualquer lugar, inclusive permite servir de base para outras aplicações como é o caso deste.

Esse sistema na nuvem conta com uma interface na qual, a priori, só pode ser acessada pelo coordenador e tem as funções de cadastrar, editar e excluir alunos e professores do sistema. O professor, por sua vez, só tem acesso aos dados dos alunos através de um segundo sistema que é um aplicativo para dispositivo móvel onde lá é possível realizar as mesmas tarefas que um coordenador com a limitação de conseguir cadastrar, editar e excluir apenas os alunos.

O objetivo principal do trabalho foi alcançado, visto que quando o microcontrolador realiza a leitura de um cartão RFID os dados são enviados para o sistema na nuvem e o registro de presença é concluído. Além disso, foram desenvolvidos os sistemas para os coordenadores e professores, possibilitando o acompanhamento da frequência dos alunos em tempo real.

Algumas desvantagens deste trabalho foram identificadas, que são:

- **Funcionamento:** realiza apenas o registro da entrada dos alunos, não faz verificações quanto à permanência deles;
- **Conexão:** A conexão utilizada é o WiFi, que não seria indicada para se usar em ambientes restritos, como é o caso do projeto desenvolvido neste trabalho, onde o equipamento é guardado em um lugar cercado de metais.
- **Sistema de backup:** não existe nenhum esforço para salvar um *backup* dos dados da nuvem;
- **Funcionamento offline:** em caso de falta de conexão com a internet o sistema fica inoperante;

3.5 ANÁLISE COMPARATIVA ENTRE OS TRABALHOS RELACIONADOS

A tabela 3 lista as características de cada trabalho com base em algumas das características consideradas importantes para o projeto desenvolvido nesta dissertação. Cada linha se está relacionada com um trabalho e cada coluna com uma característica, e cada uma dessas características estão descrita abaixo:

- **Interface Web Online**: Verifica se o projeto desenvolvido possui ou não um sistema web hospedado na nuvem;
- **BD na Nuvem**: Verifica se o projeto possui um banco de dados salvo na nuvem;
- **Possui cópia do BD externo**: Indica se o armazenamento local possui cópias das informações contidas no banco de dados externo;
- **Efetua ações offline**: Significa que o projeto consegue realizar um ou mais tarefas com segurança mesmo estando offline;
- **Protocolo da camada de aplicação**: Verifica qual o protocolo da camada de aplicação utilizado na comunicação entre o controlador e a central de monitoramento, seja ela local ou na nuvem;
- **Protocolo da camada física**: Verifica qual o protocolo da camada física utilizado na comunicação entre o controlador e a central de monitoramento.

Tabela 3 – Tabela de de comparação entre os trabalhos relacionados

Autor	**Interface Web Online**	**BD na Nuvem**	**Possui cópia do BD da nuvem**	**Efetua ações offline**	**Protocolo da camada de aplicação**	**Protocolo da camada física**
Freitas (2020)	Não	Não	Parcial	Sim	HTTP	WiFi
Day (2016)	Não	Não	Parcial	Sim	MODBus	Não Especificado
SILVA (2018)	Sim	Sim	Não	Não	HTTP	WiFi

Fonte: Autor (2022)

4 ESPECIFICAÇÃO DO SISTEMA PROPOSTO

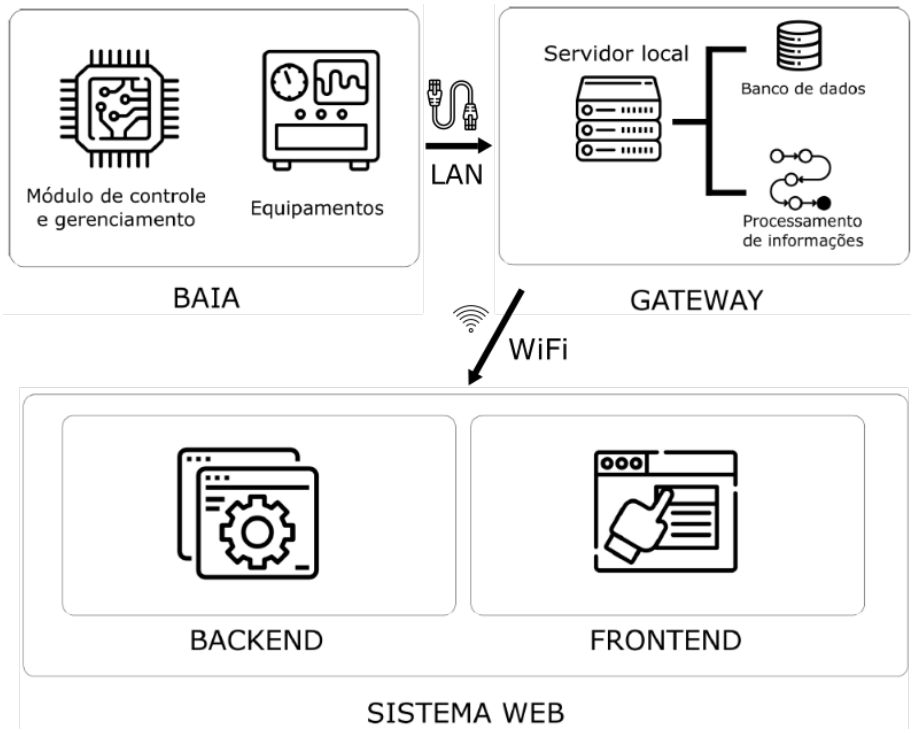
Este capítulo propõe expor as especificações do sistema que tem por objetivo mostrar onde serão aplicados os conceitos do referencial teórico, da visão geral, dos requisitos, casos de uso e arquitetura do sistema.

4.1 VISÃO GERAL

O objetivo do sistema desenvolvido neste trabalho é controlar o acesso às baias e gerenciar o uso dos respectivos equipamentos presentes nelas. As baias são abertas de maneira automática, apenas usando cartões com *tags* RFID, de forma que os alunos tenham um acesso facilitado aos equipamentos e consigam usá-los mesmo sem a presença de um monitor.

O sistema é dividido em 3 partes, sendo: subsistema do controle da baia, *gateway* e o sistema web. Entretanto, esse sistema web é composto por 2 subsistemas, o *backend* e o *frontend*. Todos esses sistemas são ilustrados na figura 8.

Figura 8 – Visão geral do sistema proposto



Fonte: Autor (2022)

A baia é o compartimento onde ficam os equipamentos e o módulo que controla o acesso dos usuários e que gerencia o uso desses equipamentos. Graças a esse módulo é possível capturar informações das *tags* RFID e transmitir os dados lidos para o subsistema responsável pelo processamento deles.

O *gateway*, é um servidor desenvolvido e executado em um micro-computador que aguarda continuamente dados enviados pelo módulo da baia através de uma rede LAN desenvolvida exclusivamente para a comunicação entre esses dois pontos. É possível conectar vários módulos de baia ao mesmo *gateway* por meio da conexão de todos eles a um mesmo *switch*. Para que esta comunicação seja estabelecida não é necessária uma conexão com a internet, a internet é utilizada apenas para enviar e receber as informações da *API Rest*, que nesse caso acontece via rede WiFi.

O sistema web engloba dois subsistemas distintos: o primeiro é o *backend*, que possui a *API Rest* responsável por receber solicitações dos outros sistemas e o banco de dados que centraliza os dados do sistema. O segundo subsistema, o *frontend*, é uma interface gráfica que permite a visualização dos dados contidos no *backend* de uma maneira amigável, ou seja, de fácil compreensão para o usuário.

4.2 REQUISITOS

Os requisitos definem o que o sistema precisa fazer para estar apto para resolver um determinado problema. Duas classes de requisitos foram elicitadas, os requisitos funcionais e os não funcionais.

Os requisitos funcionais definem as funcionalidades do sistema, em outras palavras, definem quais as tarefas que o *software* precisa desempenhar para atender às necessidades do usuário. Os requisitos não funcionais por sua vez são aqueles que dizem as características que um sistema deve ter sem se preocupar com a forma que será implementada.

A tabela 4 mostra a estrutura que será usada para apresentar os requisitos. Na primeira coluna está a referência e o título do requisito, a segunda possui duas linhas contendo apenas a indicação do que terá nas duas linhas da coluna seguinte. Na primeira linha da última coluna está a descrição daquele requisito enquanto na linha de baixo está a prioridade que aquele requisito deve ter para ser implementado.

Tabela 4 – Tabela de requisitos

[Identificador do requisito] Título do requisito	**Descrição**	Texto descrevendo o requisito.
	Prioridade	Descrição da prioridade do requisito

Fonte: Autor (2022)

As prioridades dos requisitos podem ser três: essencial, indicando que aquele requisito tem que ser implementado para que o sistema funcione; importante, ou seja, não precisa ser implementado mas o sistema não funcionará conforme é esperado; e desejável, que indica que o sistema funcionará perfeitamente sem ele, mas que esse requisito pode oferecer ganhos ao projeto.

4.2.1 Requisitos funcionais

As tabelas abaixo mostram os requisitos funcionais do sistema. Esses requisitos são divididos em: acesso, CRUD de baias e equipamentos, monitoramento, CRUD de monitores, controle e segurança.

Tabela 5 – Tabela de requisitos funcionais referente ao acesso

[RF 01] Liberação automatizada das baias	**Descrição**	As baias do laboratório devem ser liberadas automaticamente, apenas para alunos, monitores do laboratório ou funcionários do Centro de Informática.
	Prioridade	Essencial
[RF 02] Devolução de equipamentos	**Descrição**	O sistema deve permitir que o usuário devolva os equipamentos à baia após o uso.
	Prioridade	Essencial
[RF 03] Realizar fechamento da baia	**Descrição**	O sistema deve permitir que o usuário feche a baia e deve detectar automaticamente quando ela for fechada.
	Prioridade	Essencial

Fonte: Autor (2022)

Tabela 6 – Tabela de requisitos funcionais referente ao CRUD das baias e equipamentos

[RF 04] Cadastrar baias e equipamentos	**Descrição**	O sistema deve permitir que monitor/-funcionário cadastre novas baias e equipamentos.
	Prioridade	Essencial
[RF 05] Atualizar dados das baias e dos equipamentos	**Descrição**	O sistema deve permitir que o monitor/-funcionário edite informações das baias e dos equipamentos cadastrados.
	Prioridade	Essencial
[RF 06] Excluir/Descadastrar baias e equipamentos	**Descrição**	O sistema deve permitir que o monitor/-funcionário exclua os equipamentos da base de dados.
	Prioridade	Essencial

Fonte: Autor (2022)

Tabela 7 – Tabela de requisitos funcionais referente ao monitoramento

[RF 07] Mostrar uso das baias em tempo real.	**Descrição**	O sistema deve prover uma interface para visualizar o uso das baias e equipamentos em tempo real.
	Prioridade	Desejável
[RF 08] Mostrar logs de uso das baias	**Descrição**	O sistema deve ser capaz de mostrar o log de uso das baias.
	Prioridade	Desejável
[RF 09] Emitir alertas de possíveis extravios	**Descrição**	O sistema deve emitir alertas quando houver suspeitas de extravio de algum equipamento.
	Prioridade	Essencial

Fonte: Autor (2022)

Tabela 8 – Tabela de requisitos funcionais referentes ao CRUD dos monitores

[RF 10] Cadastrar monitores no sistema	**Descrição**	O sistema deve permitir que novos monitores sejam adicionados à lista de monitores.
	Prioridade	Essencial
[RF 11] Excluir/Descadastrar monitores no sistema	**Descrição**	O sistema deve permitir que os monitores sejam retirados da lista de monitores.
	Prioridade	Essencial

Fonte: Autor (2022)

Tabela 9 – Tabela de requisito funcional de segurança

[RF 12] Realizar reservas de baias.	**Descrição**	O sistema deve permitir que os alunos consigam reservar baias fora do horário de aula (seg - sex das 7h às 18h).
	Prioridade	Importante

Fonte: Autor (2022)

Tabela 10 – Tabela de requisitos funcionais referentes à segurança

[RF 13] Backup dos dados no gateway	**Descrição**	O sistema deve ser capaz de salvar uma cópia dos dados localmente.
	Prioridade	Importante
[RF 14] Fechar a baia com segurança mesmo estando offline.	**Descrição**	Mesmo offline o sistema deverá ser capaz de registrar o fechamento da baia e o enviar para a nuvem quando a internet voltar.
	Prioridade	Essencial

Fonte: Autor (2022)

4.2.2 Requisitos não funcionais

A tabela 11 mostra os requisitos não funcionais do sistema.

Tabela 11 – Tabela de requisitos não funcionais

[RNF 01] Identificação única via RFID	**Descrição**	As baias do laboratório devem ser liberadas através dos crachás RFID dos alunos, monitores ou funcionários do CIn. Além disso, os equipamentos devem possuir suas próprias tags RFID para garantir sua unicidade em relação aos outros equipamentos.
	Prioridade	Essencial
[RNF 02] Disponibilidade 24 horas	**Descrição**	O sistema deve estar em funcionamento 24 horas sem pausas.
	Prioridade	Essencial
[RNF 03] Dimensões	**Descrição**	O protótipo do módulo de monitoramento deve caber dentro da baia juntamente com os equipamentos.
	Prioridade	Essencial
[RNF 04] Escalabilidade	**Descrição**	O sistema deve permitir que várias baias possam estar conectadas simultaneamente ao gateway.
	Prioridade	Essencial

Fonte: Autor (2022)

4.3 REGRAS DE NEGÓCIO

As regras de negócio são todas as obrigações do sistema, ou seja, cada funcionalidade tem um padrão definido e qualquer coisa que fuja disso corresponde a uma falha de no sistema. As regras de negócio do projeto desenvolvido neste trabalho podem ser observadas nas tabelas 12, 13 e 14.

Tabela 12 – Tabela de regras de negócios para abertura de baia

[RN 01] Abrir uma baia livre	O usuário precisa estar portando um crachá fornecido pelo centro de informática e precisa estar em horário delimitado como horário de aula.
[RN 02] Abrir uma baia bloqueada	Apenas monitores, funcionários e o aluno que usou por último a baia poderá abrir uma baia bloqueada.
[RN 03] Abrir uma baia fora do horário de aula	Devido à quantidade de eventos que ocorrem no centro de informática, é necessário reservar uma baia com antecedência, isso deve ser feito pelo portal e precisa ser aprovada por um monitor. A resposta dessa solicitação deverá ser enviada automaticamente para o e-mail do aluno solicitante.

Fonte: Autor (2022)

Tabela 13 – Tabela de regras de negócios para fechar baia

[RN 04] Fechar baia	Antes de fechar a baia o aluno deve se certificar de que os equipamentos estão no lugar e, assim que a porta de fato for fechada, deverá chegar um e-mail para o aluno confirmando a presença ou ausência de algum equipamento. Em caso de ausência apenas o último aluno a usar e os monitores podem abrir a baia.

[RN 05] Fechar baia sem internet no gateway	Nenhuma informação pode ser perdida mesmo com o gateway estando sem internet, por isso, as informações sobre o fechamento de uma baia devem ser salvas em um banco de dados e enviadas para a API Rest assim que a conexão for reestabelecida.

Fonte: Autor (2022)

Tabela 14 – Tabela de regras de negócios do portal

[RN 06] Adicionar e remover itens no portal	Apenas quem pode alterar e excluir registros de baias, equipamentos e monitores são os próprios monitores e funcionários, essas funcionalidades devem ficar invisíveis pelos alunos.
[RN 07] Ver página de logs	A página de logs deverá estar visível apenas para monitores e funcionários.

Fonte: Autor (2022)

4.4 CASOS DE USO

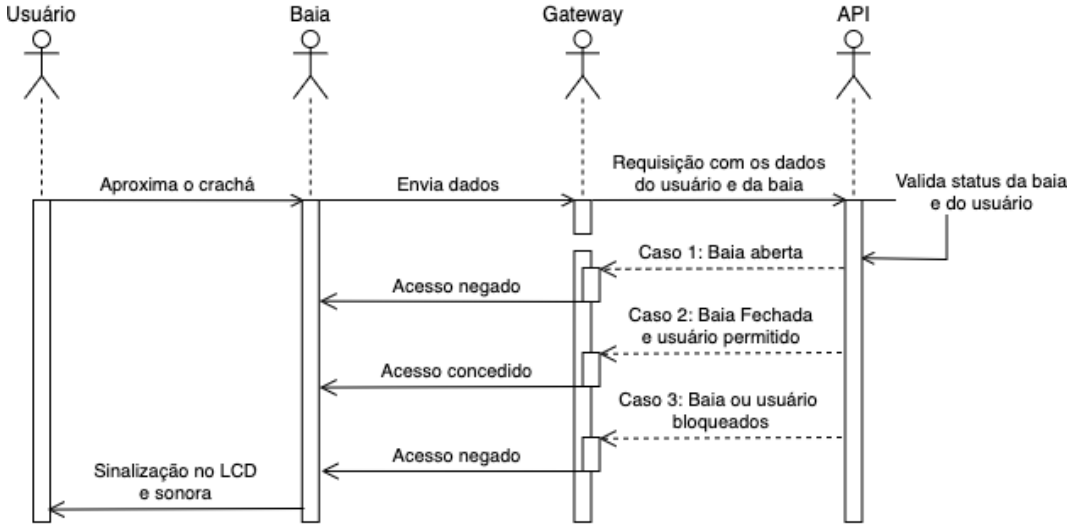
Um caso de uso é uma forma de detalhar como a funcionalidade de um software deve se comportar. Existe um conjunto de regras que devem ser seguidas para que haja um padrão e levem os desenvolvedores ao mesmo entendimento acerca do que deve e como deve ser implementado.

A estrutura usada para especificar os casos de uso é por meio de um diagrama de sequencia, que demonstra de uma maneira visual como o sistema deve se comportar em determinada situação. As figuras baixo correspondem aos diagramas de sequência de cada caso de uso, é possível observar que em cada figura existem blocos que se relacionam o que indica que é uma ação tomada para determinada a situação.

A figura 9 corresponde ao caso de uso de abrir uma baia com o cartão RFID e nesse caso de uso são aplicadas as regras de negócio RN-01, RN-02 e RN-03 da tabela 12. Essas regras de negócio são aplicadas na API e é possível observar três possíveis casos como resposta: o primeiro é quando a baia está aberta, nesse caso o acesso sempre vai ser negado; o segundo, ocorre

quando nem a baia e nem o usuário possui pendências, sendo assim, o acesso é concedido; e o terceiro caso acontece quando, ou a baia, ou o usuário possui alguma pendência e o acesso é negado.

Figura 9 – Caso de uso abrir baia com identificação via RFID



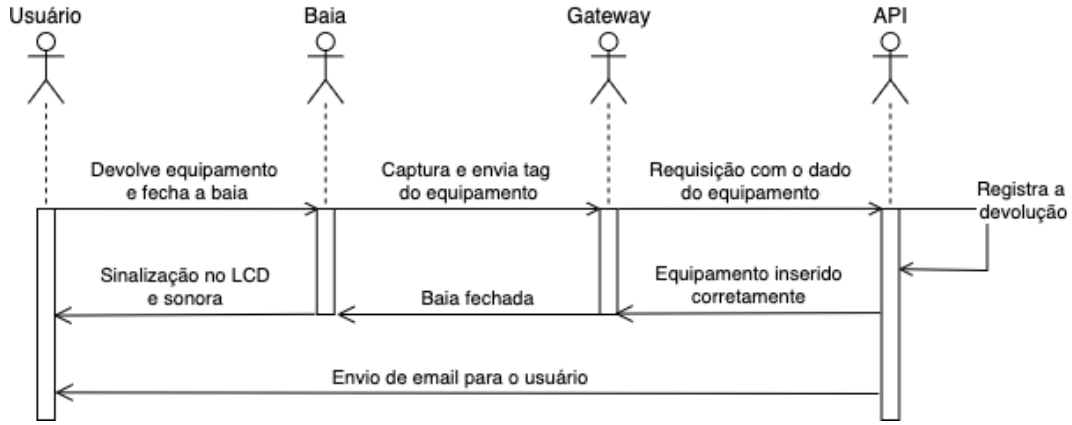
Fonte: Autor (2022)

A figura 10 corresponde ao caso de uso de detectar e registrar presença de equipamento após fechar a baia, a regra de negócio aplicada nesse caso é a RN-04 da tabela 13. Nesse caso o sistema deve permitir que o usuário devolva os equipamentos após o uso, além de reconhecer, automaticamente, que o usuário os devolveu. Existem quatro partes atuantes nesse caso de uso: o usuário, que coloca o equipamento no lugar e fecha a baia; a própria baia, que identifica quando foi fechada, captura as informações dos equipamentos e os envia para o gateway; o gateway, que por sua vez faz a requisição informando para a API os equipamentos presentes; e a própria API que registra que o equipamento foi inserido corretamente, responde ao gateway e, a partir daí, todos os outras partes atuantes receberá ao menos uma resposta referente a sua solicitação.

A figura 11 corresponde ao caso de uso de fechar a baia, a regra de negócio aplicada nesse caso também é a RN-04 da tabela 13. Esse caso é bem semelhante ao anterior, inclusive as partes atuantes são as mesmas, o fluxo que o dado percorre até a API também é o mesmo, a diferença é que aqui é especificado o que acontece em cada situação possível: no primeiro caso, quando há um equipamento faltando, o gateway envia para a baia que aquela a mesma está bloqueada; e no segundo caso, é registrado que a baia foi fechada corretamente e transmitido isso para todas as partes. Em ambos os casos, é enviando um e-mail para o aluno, porém, no caso de um possível extravio é enviado um e-mail também para os monitores.

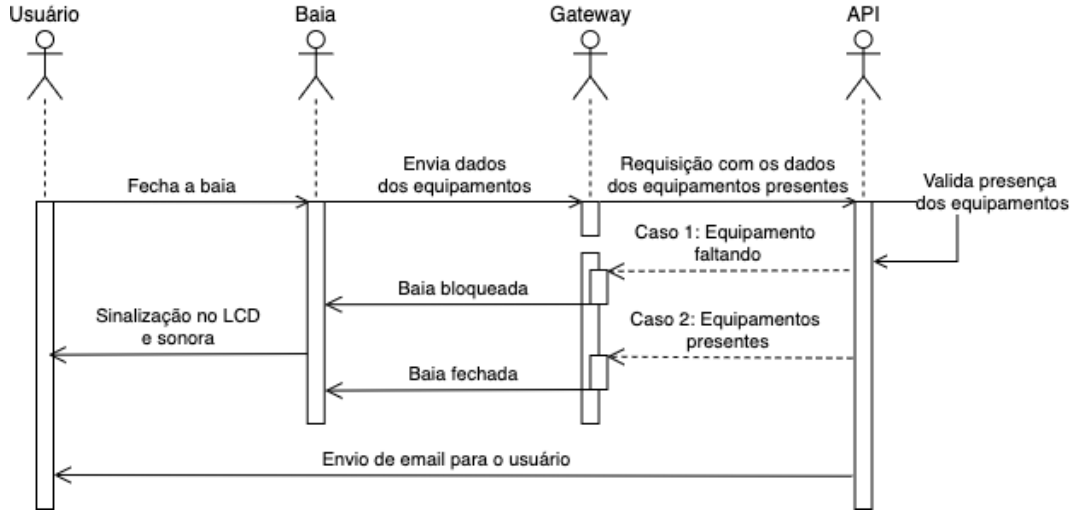
A figura 12 corresponde ao caso de uso de cadastrar um novo monitor, a regra de negócio aplicada nesse caso é o RN-06 da tabela 14. Nesse ceso, o sistema deve permitir que monitores e funcionários cadastrem novos monitores no sistema, para isso, possui quatro partes atuantes: monitor/funcionário, página de login e cadastrar monitores do portal web, e a API. O fluxo para

Figura 10 – Caso de uso de detectar presença de equipamento



Fonte: Autor (2022)

Figura 11 – Caso de uso registrar o fechamento de uma baia



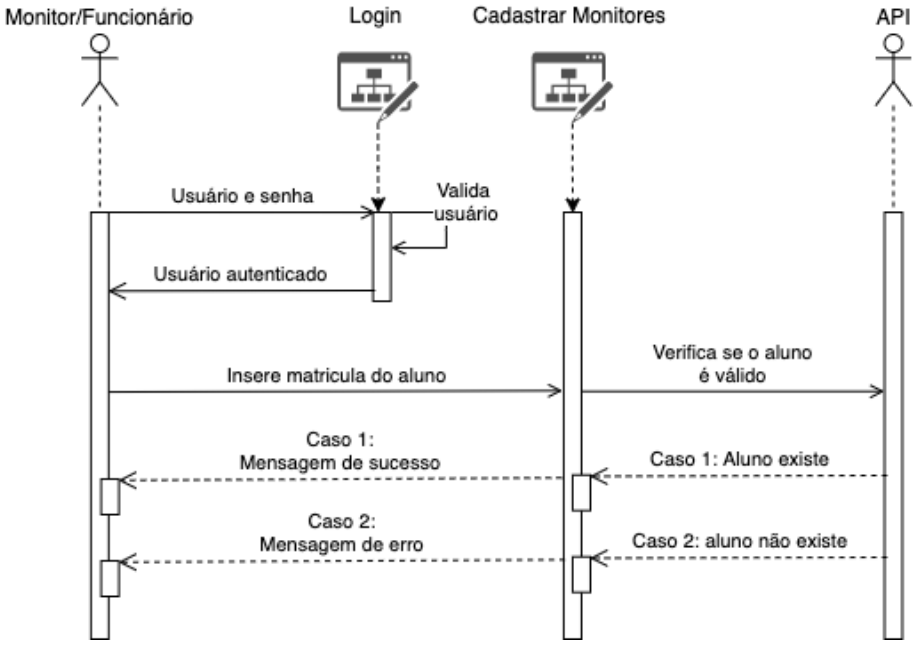
Fonte: Autor (2022)

realizar o cadastro se inicia com o usuário, sendo ele um monitor ou funcionário, realiza seu login, em seguida se dirige à página de cadastrar monitores, insere a matrícula do aluno e se esse aluno de fato existir, o cadastro é realizado com sucesso, caso contrário apenas é mostrada ao usuário uma mensagem de que não foi possível concluir tal operação.

A figura 13 corresponde ao caso de uso de cadastrar um novo equipamento, a regra de negócio aplicada nesse caso é a RN-06 da tabela 14. Nesse caso de uso o sistema deve permitir que novos equipamentos sejam adicionados ao banco de dados e associados a alguma baia. A figura 13 mostra três partes atuantes: o monitor/funcionário, a página de login e de cadastrar equipamentos, sendo esses dois últimos no portal web. Nesse caso de uso, antes de tudo o monitor ou funcionário deve se autenticar, após isso ir até a página de cadastrar equipamento, inserir as informações do mesmo e, por último, ele recebe uma mensagem informando se o seu cadastro foi realizado com sucesso ou não.

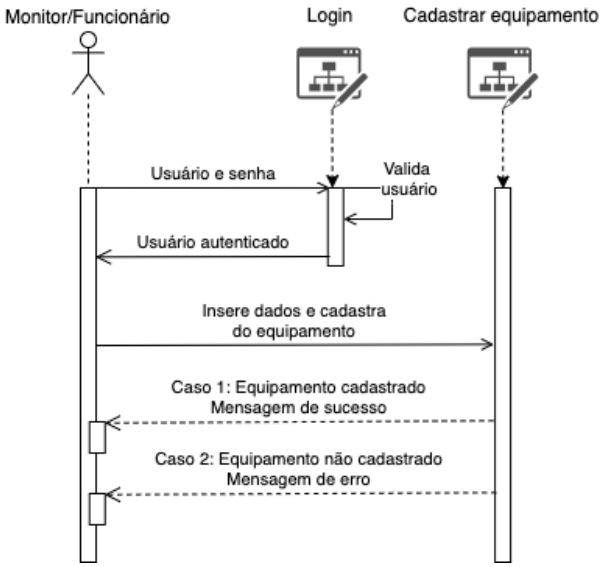
A figura 14 corresponde ao caso de uso de cadastrar uma nova baia no portal e a regra de

Figura 12 – Caso de uso cadastrar novos monitores



Fonte: Autor (2022)

Figura 13 – Caso de uso cadastrar novos equipamentos

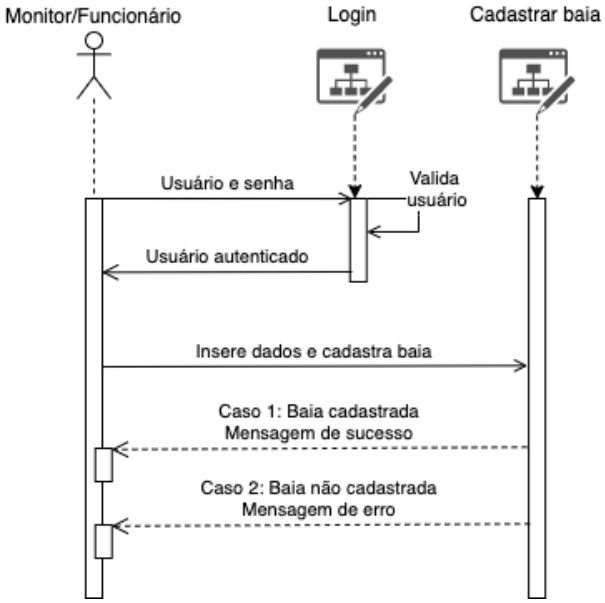


Fonte: Autor (2022)

negócio aplicada nesse caso também é a RN-06 da tabela 14. O fluxo e as partes atuantes são os mesmo do anterior, a diferença é que ao invés de se dirigir à tela de cadastrar equipamento, nesse caso ele precisa ir à tela de cadastrar baia. Após isso ele insere as informações da baia e clica em cadastrar, em seguida ele recebe uma resposta informando se seu cadastro foi realizado com sucesso.

Os casos de uso das figuras 15, 17, 16, 18 e 19 correspondem, respectivamente aos casos de uso de editar e remover um equipamento cadastrado, editar e remover uma baia e, por último,

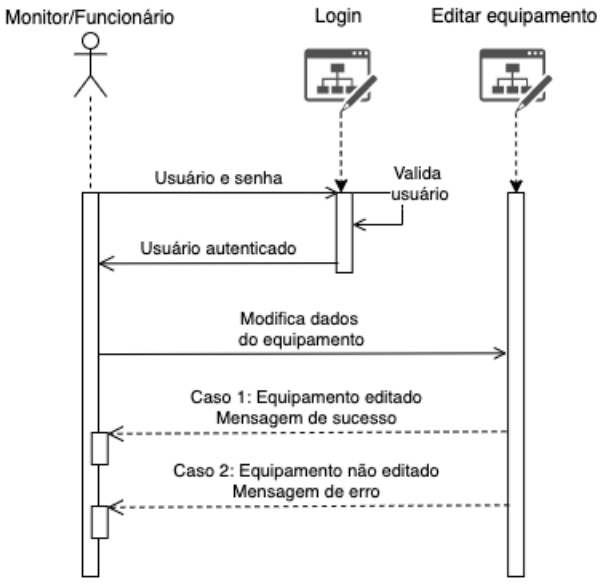
Figura 14 – Caso de uso cadastrar baia



Fonte: Autor (2022)

remover um monitor do sistema. A regra de negócio aplicada nesses casos também é o RN-06 da tabela 14. O fluxo e as partes atuantes são os mesmos do caso de uso de cadastrar um novo equipamento diferenciando apenas o que de fato o usuário, sendo ele um monitor ou um funcionário, modifica no sistema.

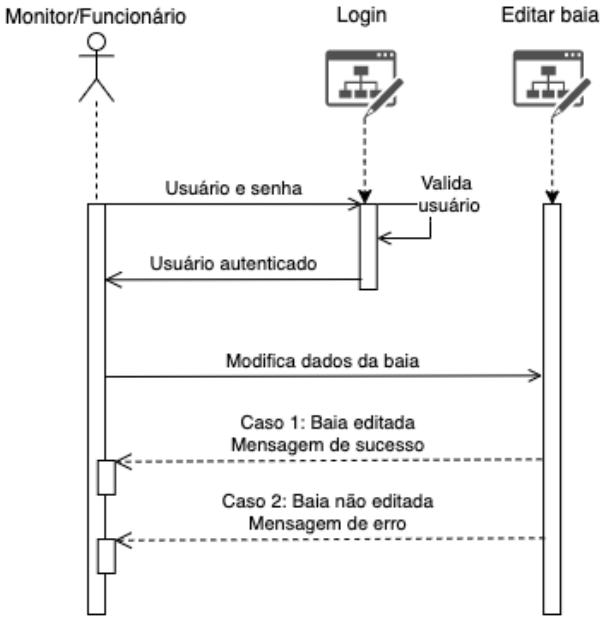
Figura 15 – Caso de uso editar equipamentos



Fonte: Autor (2022)

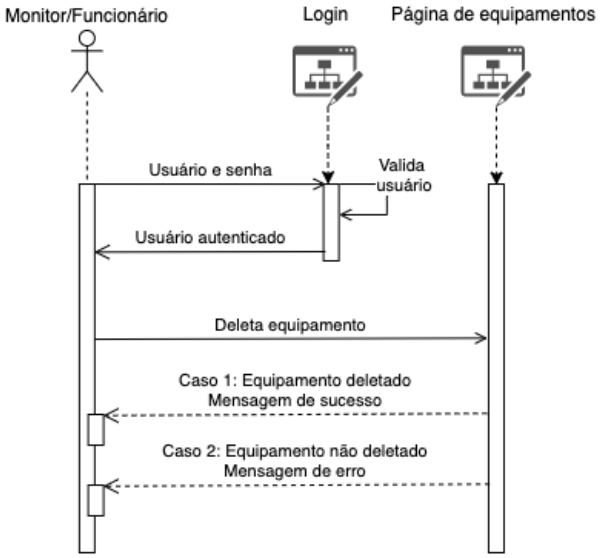
As figura 20 e 21 correspondem, respectivamente, aos caso de uso de visualizar uso das baias, aplicando a regra de negócio RN-06, e visualizar os logs do sistema, aplicando a regra e negócio RN-07, ambas as regras de negócio da tabela 14. O fluxo de ambos os casos de uso são

Figura 16 – Caso de uso editar baia



Fonte: Autor (2022)

Figura 17 – Caso de uso remover equipamentos

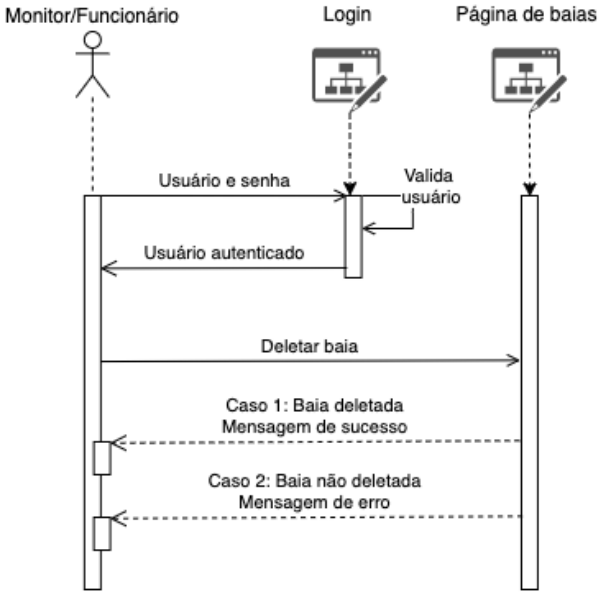


Fonte: Autor (2022)

os mesmo, o que difere são as partes atuantes, onde, o primeiro compreende todos os usuários, ou seja: alunos, monitores e funcionários. O segundo caso de uso é mais restrito, permite o acesso apenas para monitores e funcionários. O objetivo do primeiro é permitir que todos os usuários possam visualizar em tempo real o status das baias, enquanto o segundo permitir que os monitores e funcionários visualizem os logs do sistema.

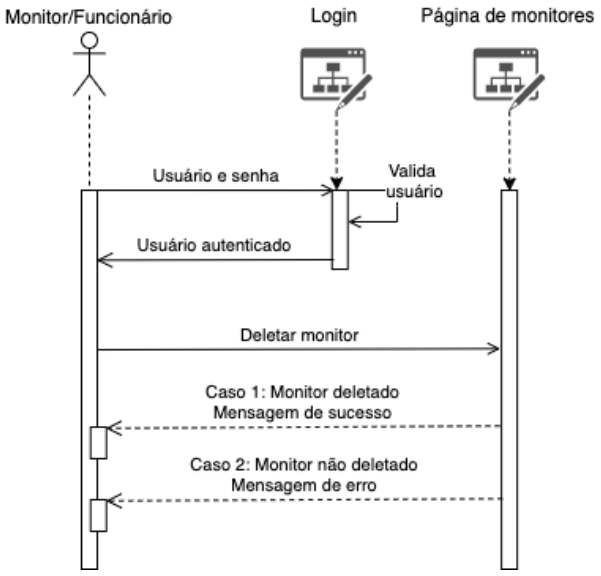
A figura 23 corresponde ao caso de uso de receber alertas de possível extravio e a regra de negócio aplicada nesse caso é a RN-04 da tabela 13. Nesse caso de uso o sistema deve enviar alertas de possíveis extravios ao aluno responsável e aos monitores. Existem quatro partes

Figura 18 – Caso de uso remover baia



Fonte: Autor (2022)

Figura 19 – Caso de uso remover monitores do sistema

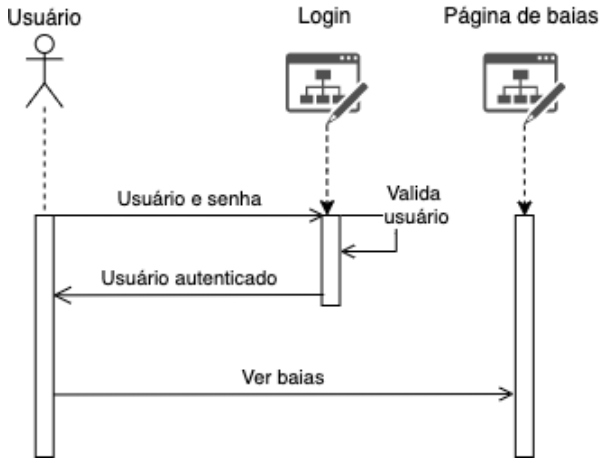


Fonte: Autor (2022)

atuantes nesse caso de uso: o usuário, que coloca o equipamento no lugar e fecha a baia; a própria baia, que identifica quando foi fechada, captura as informações dos equipamentos e os envia para o gateway; o gateway, que por sua vez faz a requisição informando para a API os equipamentos presentes; e a própria API que identifica que há um equipamento faltando, registra a baia como bloqueada e envia a respostas que será transmitida para todas as partes atuantes, inclusive o aluno e monitores via e-mail.

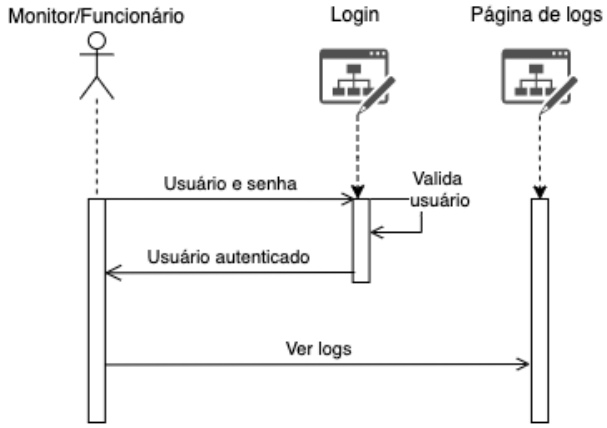
A figura 23 corresponde ao caso de uso de uso realizar reservas de baias e a regra de negócio aplicada nesse caso é a RN-03 da tabela 12. O sistema deve permitir que os alunos consigam

Figura 20 – Caso de uso visualizar uso das baias em tempo real



Fonte: Autor (2022)

Figura 21 – Caso de uso visualizar logs de uso



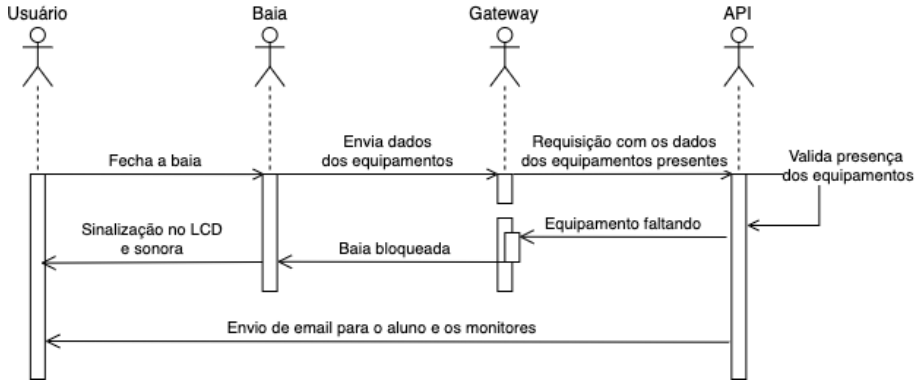
Fonte: Autor (2022)

reservar baias fora do horário de aula (segunda - sexta das 7h às 19h). Nesse caso o usuário precisa navegar até a página de reservar baia, verificar se no horário desejado há baias disponíveis, se houver aluno seleciona as datas e clica em realizar o agendamento, o sistema registra no banco de dados a data e a hora da reserva e o aluno aguarda a confirmação, a resposta será enviada por e-mail tanto no caso dela ser positiva como também negativa.

A figura 24 corresponde ao caso de uso da realização de *backup* dos dados no *gateway* diariamente, ou seja, o *gateway* deve ser capaz de buscar dados na nuvem todos os dias em um horário fixo, delimitado no projeto às 5 horas da manhã. Existem quatro partes atuantes nesse caso de uso: o *gateway*, o relógio, a API e o banco de dados local. Existe um evento que todos os dias às 5 horas da manhã é disparado e busca na API todos os dados salvos no dia anterior. Assim que a resposta é recebida pelo *gateway*, esses dados são salvos localmente.

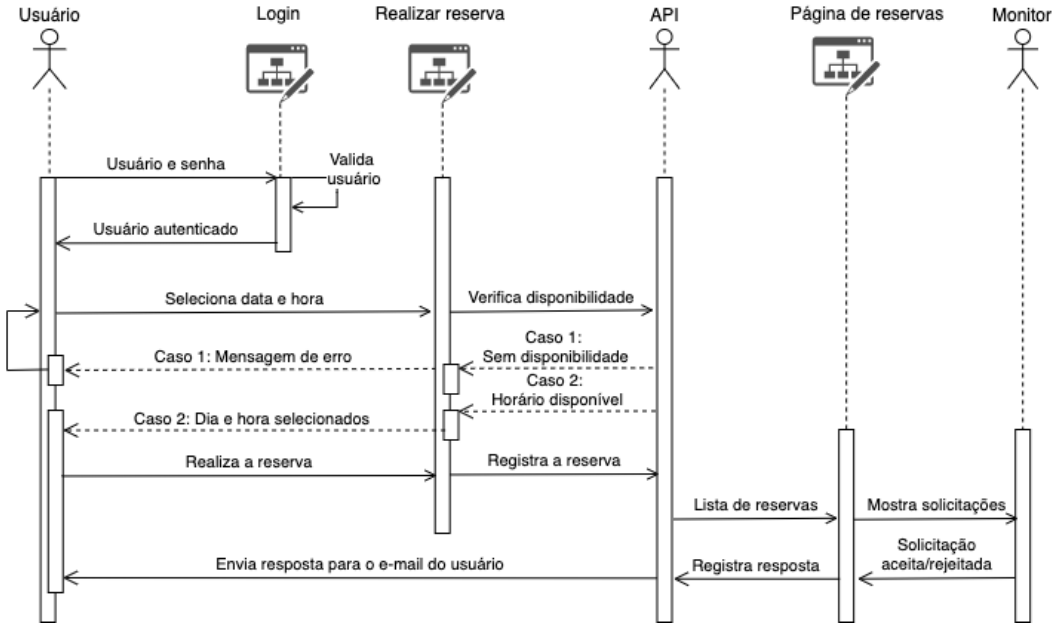
A figura 25 corresponde ao caso de uso fechar a baia com segurança mesmo estando offline e a regra de negócio aplicada nesse caso é a RN-05 da tabela 13. Esse é o caso que mais possui partes atuantes, sendo elas: o usuário, a baia, o *gateway*, a internet, o banco de dados do

Figura 22 – Caso de uso receber Alertas de Possível Extravio



Fonte: Autor (2022)

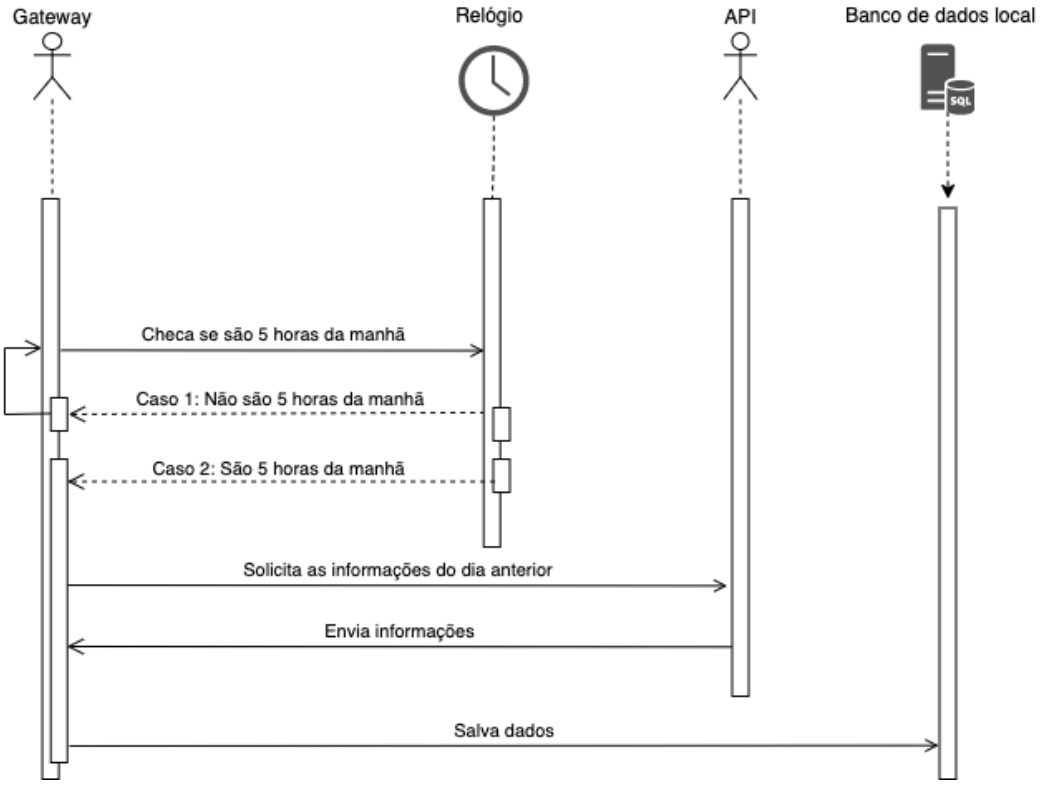
Figura 23 – Caso de uso realizar reservas de baias



Fonte: Autor (2022)

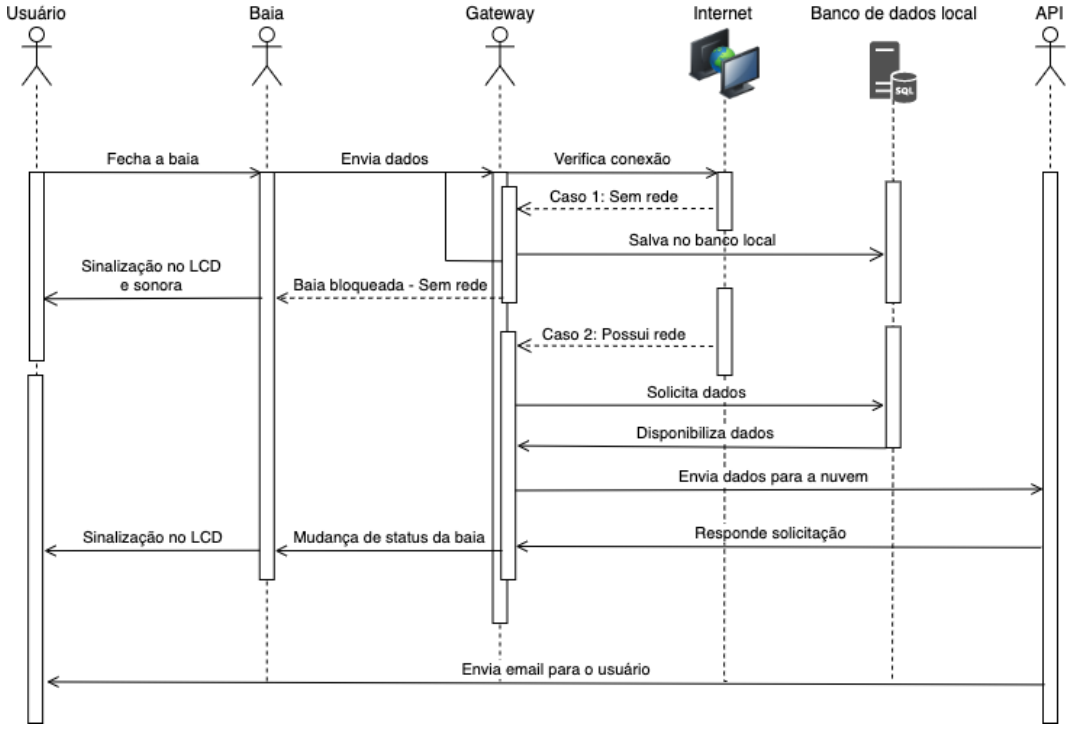
gateway e a API. O caso de uso se inicia com usuário fechando a baia, a mesma coleta e envia os dados das *tags* dos equipamentos presentes para o *gateway*, que recebe, verifica que está sem conexão com a internet, salva a solicitação no banco local e bloqueia a baia até que a internet seja reestabelecida e verifica continuamente a conexão. Quando a conexão é reestabelecida a tabela que guarda os dados salvos offline é lida e a solicitação de fechar a baia é feita para a API, que em seguida responde para o *gateway*, que muda o status da baia e que sinaliza no LCD da mesma seu novo status. Além disso a API também envia um e-mail para os respectivos destinatários.

Figura 24 – Caso de uso backup do banco de dados



Fonte: Autor (2022)

Figura 25 – Caso de uso fechar a baia com segurança mesmo estando offline

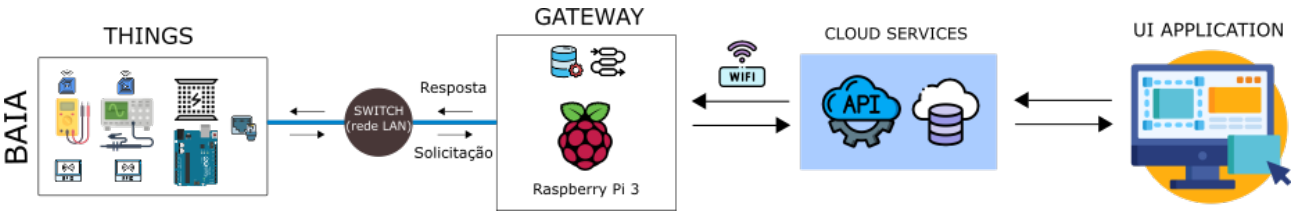


Fonte: Autor (2022)

4.5 ARQUITETURA

A arquitetura do projeto desenvolvido, mostrada na figura 26, é baseada em um sistema Iot com quatro camadas, sendo elas: os sensores, ou também chamados de "things", o *gateway*, o *back-end* e *front-end*.

Figura 26 – Arquitetura do sistema



Fonte: Autor (2022)

A camada de sensores recebe as solicitações do usuário, abrir ou fechar baia, cria um pacote de dados em um formato específico, envia para a camada seguinte e aguarda uma resposta para concluir a a solicitação. Enquanto essa resposta não é devolvida o usuário é informado que o sistema está processando a solicitação e que ele deve aguardar.

O *gateway*, que é um servidor local, atua como uma ponte na comunicação entre as duas camadas adjacentes, a de sensores e serviços na nuvem. Para isso, ele recebe e trata os dados vindos de ambas as camadas e por fim direciona a mensagem para seu devido destinatário. Além disso, o *gateway* também salva dados localmente afim de assegurar que os dados na nuvem tenham uma cópia e solicitações dos sensores não sejam perdidas.

A camada seguinte oferece os serviços na nuvem, que consistem em um banco de dados e uma *API Rest*. A função do banco de dados nesta camada é de centralizar todos os dados do projeto e servir de fonte principal de informação nos processos decisórios do sistema. A *API Rest* é a interface que recebe e lida as solicitações dos usuários levando em consideração as regras de negócio do projeto, ou seja, cada solicitação é direcionada para um *endpoint* que, a partir das informações recebidas, é possível saber se a solicitação pode ou não ser atendida.

Por último temos a camada de aplicação que consiste em um portal voltado para os alunos, monitores e funcionários que oferece uma visão geral de como as baias estão sendo utilizadas e dá aos responsáveis pelo laboratório a autonomia de adicionar ou retirar baias, equipamentos e monitores, decidir quais reservas poderão ser aceitas e visualizar todo o log do sistema.

5 MATERIAIS E MÉTODOS

5.1 ARDUINO

O Arduíno é uma plataforma de prototipação de sistemas eletrônicos baseado em microprocessadores de código aberto, ou seja, suas bibliotecas são disponíveis para qualquer pessoa interessada. Além disso, as próprias plataformas oficiais estimulam o crescimento de uma comunidade ativa, onde todos que quiserem consigam interagir e compartilhar conhecimentos.

Segundo o site oficial Arduino (2021a), tanto o hardware como o software estão disponíveis para serem usados por qualquer pessoa, permitindo que elas acessem facilmente tecnologias avançadas que interagem com o mundo físico. Os produtos são simples, fáceis de usar e possuem poder computacional para diversas aplicações, de forma que podem satisfazer as necessidades tanto de estudantes como de desenvolvedores profissionais. Por esses motivos foi escolhida essa plataforma para a criação do módulo de gerenciamento das baias desenvolvido neste trabalho.

Atualmente existe uma vasta gama de placas com diversas finalidades e dimensões físicas para atender às demandas dos mais variados projetos. No sistema desenvolvido neste trabalho, a placa utilizada foi um Arduíno UNO, que é uma das mais comuns e fáceis de encontrar disponível no mercado com um preço em torno de R$ 60,00.

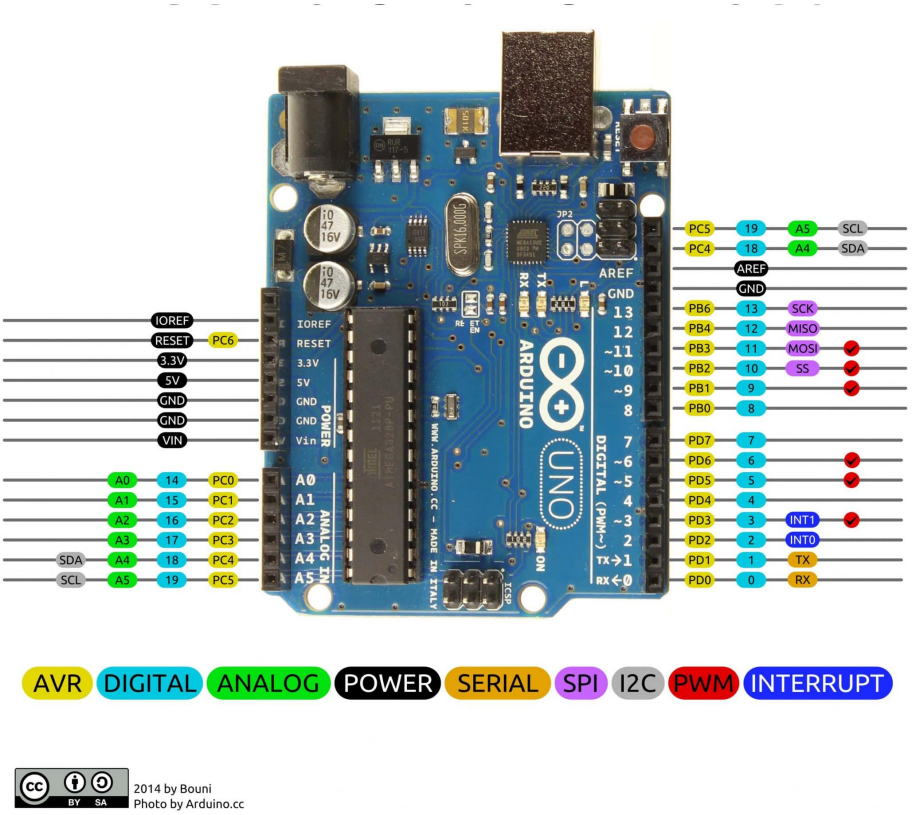
O Arduíno UNO é apresentado na figura 27 junto com uma relação do que cada pino representa. Segundo o *datasheet* da placa, disponibilizado no site oficial, em Arduino (2021b). A plataforma apresenta as seguintes características:

- Disponibiliza 14 portas digitais, possui ainda 6 portas analógicas e 6 tipos de pinos de energia;
- Seu microcontrolador é o ATmega328P que possui memória flash de 32KB, a SRAM de 2KB e a EEPROM de 1KB;
- A comunicação serial possui suporte aos seguintes protocolos de comunicação serial: UART, I2C e SPI;
- A tensão de entrada recomendada para um bom funcionamento da placa é entre 7V e 12V, e essa alimentação pode ser feita pela porta USB e ou por meio do conector de energia.

O ATmega328P é um microcontrolador de 8 *bits* fabricado pela Atmel, que é baseado no processador AVR RISC que garante um baixo consumo de energia, com uma alimentação de até 5V, e com uma arquitetura conhecida como *Arquitetura Harvard*.

Sendo assim, os motivos que contribuíram para a escolha do Arduíno como plataforma de prototipação foram: o baixo custo, a facilidade de encontrar tutoriais e tirar dúvidas nas comunidades, a facilidade de conectar novos componentes à placa e não é necessário ter conhecimentos avançados em eletrônica.

Figura 27 – Arduino UNO

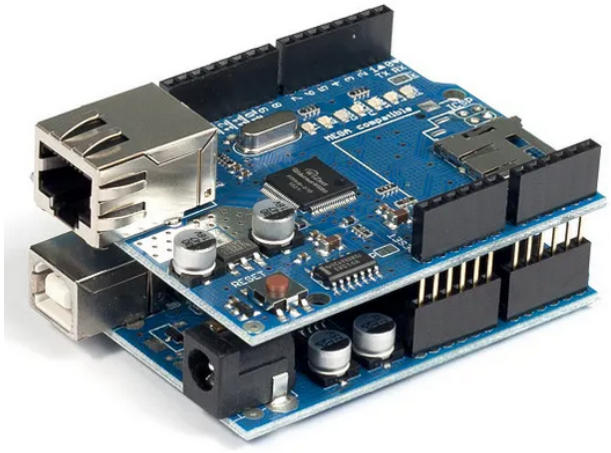


Fonte: Protto (2020)

5.1.1 Shield Ethernet

No sistema desenvolvido neste trabalho foi necessária a conexão a uma rede local via cabo ethernet do tipo RJ-45. Para esta conexão foi usado o *Shield Ethernet W5100* pela facilidade e praticidade de uso, precisando apenas encaixar ao Arduíno e sem a necessidade de plugar cabos.

Figura 28 – Shield ethernet conectado ao Arduíno UNO



Fonte: Caballero (2016)

Embora praticamente todas as suas portas estejam conectadas entre si, nem todas estão ocupadas, o *shield ethernet* permite que os pinos digitais, analógicos e de alimentação continuem disponíveis para que sensores sejam conectados e usados pelo microcontrolador.

A comunicação entre ambas as placas ocorre por meio dos pinos ICSP do Arduíno UNO, em destaque na figura 29. Esses pinos podem ser divididos em dois grupos: o grupo responsável pela tensão e controle (sendo os pinos VCC, GND e RST) e o grupo responsável pela conexão SPI (sendo os pinos MISO, MOSI e SCK) que realiza a troca de dados entre as placas.

Figura 29 – Arduíno UNO e pinos ICSP

Fonte: JGMaker (2020)

Falar brevemente sobre os outros componentes utilizados

5.2 SQL

Standard Query Language, ou simplesmente SQL, de acordo com RAMAKRISHNAN e Gehrke (2002), foi desenvolvida originalmente pela IBM na década de 70 e é a linguagem de banco de dados mais utilizada comercialmente até os dias de hoje.

O SQL é uma linguagem para especificação e desenvolvimento de banco de dados relacional sendo ela considerada uma linguagem declarativa e que pode ser usada tanto para análise quanto para a execução de tarefas dentro do banco de dados.

Os diversos comandos existentes no SQL podem ser classificados, segundo Andrade (2020), nas classes DQL, DDL e DML, que permitem consulta, definição e manipulação, respectivamente, conforme a seguir:

- DQL (*Data Query Language*)
  - SELECT: com ele é possível realizar consultas aos dados que pertencem a uma determinada tabela.
- DDL (*Data Definition Language*)
  - CREATE: no caso de um *create database* cria um novo banco de dados vazio e no caso de um *create table* irá criar uma nova tabela.

  - ALTER: comando utilizado para alterar uma tabela ou um banco de dados já existente;
  - DROP: utilizado para remoção de uma tabela ou do banco de dados por completo;

- DML (*Data Manipulation Language*)

  - INSERT: comando utilizado para inserir dados a uma ou mais tabela de um banco de dados;
  - UPDATE: comando utilizado para atualizar os dados de uma ou mais tabelas;
  - DELETE: utilizado para excluir os dados de uma ou mais tabelas no banco de dados.

No sistema desenvolvido neste trabalho a linguagem SQL é utilizada pelos sistemas de gerenciamento de banco de dados MySQL e MariaDB. A escolha pelo MySQL sucedeu-se porque possui suporte nos provedores mais populares de nuvem e quando o projeto estiver sendo colocado em produção isso pode impactar diretamente no preço da hospedagem. Por outro lado, o MariaDB é utilizado em um banco de dados local, ou seja, será implementado e usado em uma máquina do CIn, dessa forma o fator que mais foi levado em consideração foi o desempenho, que nesse caso o MariaDB se sobressai em relação ao MySQL.

5.3 TECNOLOGIAS UTILIZADAS

O projeto desenvolvido neste trabalho conta a implementação em três plataformas distintas e em cada uma dessas plataformas existem várias outras tecnologias que servem de apoio para execução de tarefas e abstração de código. Todas essas tecnologias serão explicadas com mais detalhes no capítulo 6 e o código fonte pode ser encontrado no endereço https://github.com/pedroclericuzi/Projeto-LabCin.

A começar pelo módulo da baia, o ambiente de desenvolvimento utilizado foi a própria IDE do Arduíno, logo a linguagem de programação utilizada foi o C++. Além disso foram usadas um conjunto de funções oriundas de diferentes bibliotecas, sendo elas: SPI, MFRC522, Ethernet.h, LiquidCrystal_I2C.

O *gateway*, executado em um Raspberry Pi 3, tem o seu sistema implementado em Node.JS, que é um o ambiente de execução JavaScript server-side. Para facilitar a implementação, são utilizadas as bibliotecas: *net*, *node-schedule*, *check-internet-connected*. Além disso, ainda foi criado um banco de dados todo baseado no sistema de gerenciamento MariaDB, que usa a linguagem SQL para a execução das *queries*.

A plataforma web é hospedada no Heroku, que é uma plataforma como serviço que já fornece a hospedagem, o domínio e o suporte ao banco de dados, de forma que a aplicação esteja rapidamente apta a ser acessada por usuários externos. Diferente do *gateway*, o sistema de gerenciamento de banco de dados implementado na plataforma web é o MySQL, que também

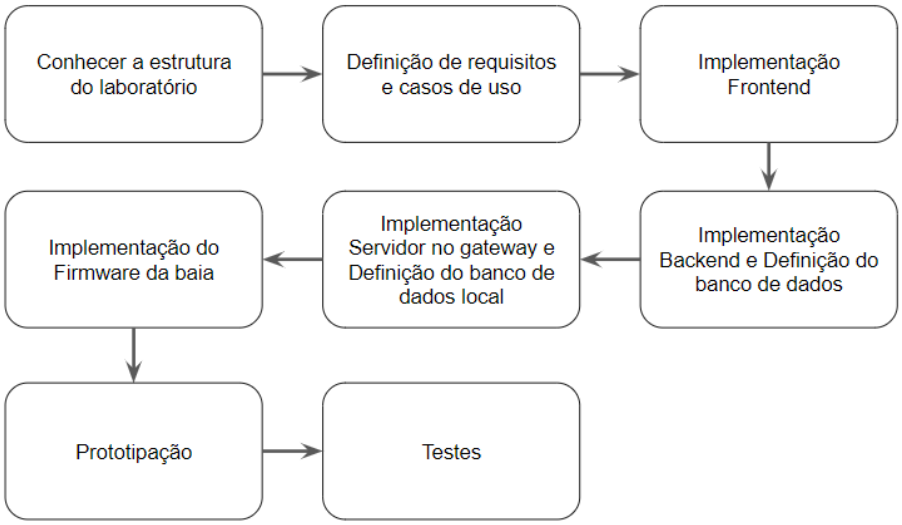
utiliza a linguagem SQL, e que é mais bem aceito em plataformas de hospedagem. Para o desenvolvimento do *front-end* e o *back-end* foi utilizado o framwork Laravel, que é um framework PHP que além de ter o suporte para o *back-end* também possibilita o desenvolvimento de um sistema *front-end* usando HTML, CSS e JavaScript.

6 IMPLEMENTAÇÃO DO SISTEMA PROPOSTO

O sistema desenvolvido neste trabalho conta com 3 subsistemas bem diferentes. O primeiro é o protótipo do *hardware*, que funciona como um subsistema de controle da baia. O segundo, é um servidor local implementado em um micro-computador e serve como um *gateway* para processar, buscar informações na nuvem e armazenar dados. O terceiro subsistema possui um *backend* e um *frontend*, que atende às necessidades do usuário e dos subsistemas citados.

O projeto foi desenvolvido seguindo o fluxo da figura 30, onde se inicia no conhecimento da estrutura do laboratório, passa pela definição dos requisitos e casos de uso, implementação, o desenvolvimento do protótipo e os testes.

Figura 30 – Procedimentos metodológicos do projeto desenvolvido



Fonte: Autor (2022)

Este capítulo irá detalhar o que e como foi implementado em cada subsistema e como eles se integram.

6.1 SUBSISTEMA DO CONTROLE DA BAIA

Para o subsistema de controle da baias foram escolhidos componentes visando a compatibilidade entre eles e o custo-benefício de cada um. A tabela 15 lista os componentes que fazem parte do subsistema de controle da baias:

Tabela 15 – Tabela com a listagem dos componentes escolhidos

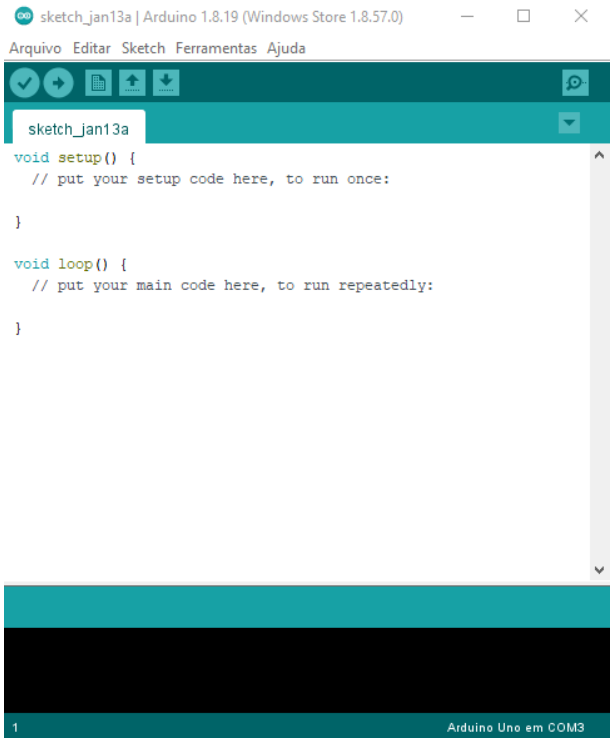
Função	**Componentes**
Controlador	Arduíno UNO

Leitor RFID	MFRC522
Display LCD	16x2
Sensor para porta	Sensor magnético 438
Auto-falante	Buzzer Ativo 5v
Acionamento de cargas	Módulo Relé 5V

Fonte: Autor (2022)

Existem diferentes linguagens e ambientes de desenvolvimento que tornam possível programar para Arduino. No presente trabalho todo o *firmware* foi desenvolvido no Arduino IDE que usa a linguagem C++. Na IDE todo novo projeto que é criado possui essa estrutura, com apenas duas funções, conforme observa-se na figura 31: *void setup()* e *void loop()*.

Figura 31 – Código padrão de um novo projeto na IDE do Arduino



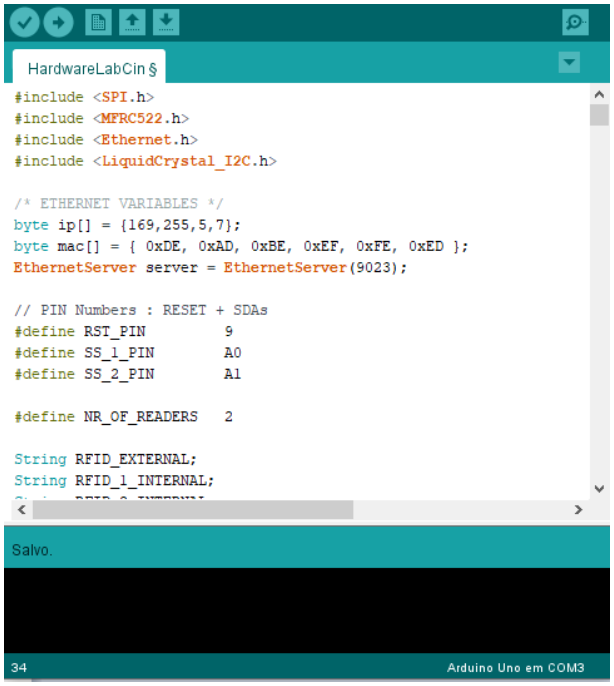
Fonte: Autor (2022)

A função *void setup()* é executada apenas uma vez sempre que uma aplicação é carregada com o intuito de inicializar funções, variáveis e os pinos utilizados na placa. O *void loop()*, por outro lado, é a função responsável por executar toda a parte lógica do projeto e roda continuamente até que haja alguma interrupção.

Além disso, para facilitar o desenvolvimento é possível utilizar bibliotecas, que são códigos desenvolvidos pela comunidade para tornar possível o acesso aos componentes de forma que não precise criar um código do zero sempre que for necessário usar um novo componente. No

projeto desenvolvido nesta dissertação foram usadas quatro bibliotecas conforme é mostrado nas quarto primeiras linhas da figura 32.

Figura 32 – Parte do projeto desenvolvido



Fonte: Autor (2022)

As duas primeiras bibliotecas trabalham em conjunto, pois a *MFRC522.h* se conecta na interface SPI através da biblioteca *SPI.h* para poder ler cartões RFID. Por meio dessa conexão um dispositivo mestre, nesse caso o Arduíno, se conecta aos escravos que são os leitores RFID e assim tem acesso às informações dos cartões dos usuários e/ou dos equipamentos.

A terceira biblioteca a *Ethernet.h* também se conecta na interface SPI através dos pinos ICSP e graças a ela é possível se conectar à internet, ou no caso do projeto desenvolvido, se conectar ao servidor local que está sendo executado no *gateway*.

A última biblioteca é a *LiquidCrystal_I2C.h*, que diferente das duas últimas se comunica através do protocolo I2C, que tem como uma das diferenças do SPI o fato de que permite apenas um mestre por barramento. Essa biblioteca é usada para que o Arduíno consiga realizar o envio do texto que será mostrado display LCD.

6.1.1 Leitores RFID

No protótipo desenvolvido neste trabalho houve a necessidade de utilizar mais de uma placa leitora de RFID porquê além da placa que fica do lado de fora da baia (para leitura do crachá do usuário), cada equipamento dentro dela também possui uma *tag* RFID para que, depois de utilizada, haja uma conferência se realmente todos os equipamentos desta baia estão presentes.

É válido destacar que o único papel do Arduíno é conseguir fazer a leitura das *tags* e gerenciar o uso das baias, todo o processo de tratamento e checagem dos dados são feitos no *gateway* e no *backend*, respectivamente.

A figura 33 mostra a configuração inicial para se utilizar mais de um leitor RFID. As linhas 12 a 14 são variáveis constantes que definem onde serão os pinos de *reset* e SS (*Slave Select*) que servem para colocar os módulos em modo de baixo consumo e para realizar a comunicação SPI, respectivamente. A linha 16 define o número de leitores RFIDs que serão conectados ao microcontrolador, enquanto nas linhas 18 até a 21 são instâncias para armazenar os valores lidos por esses leitores RFID. A linha 18 irá armazenar o que vai ser transmitido pelo leitor que ficará do lado de fora da baia, as linhas seguintes guardam as informações daqueles que ficarão dentro. Foram criadas três variáveis pois ficou definido que seria a quantidade máxima de equipamentos para esta versão do protótipo.

Nas linhas 23 e 25 são criados dois *arrays* sendo o primeiro contendo os pinos SS (*Slave Select*) de todos os módulos RFIDs e o segundo é a utilização da biblioteca MFRC522 delimitando o tamanho desse *array* com a quantidade de leitores que foram definidos, para que posteriormente cada módulo seja usado corretamente.

Figura 33 – Definições de variáveis RFID

```
12 #define RST_PIN         9
13 #define SS_1_PIN        A0
14 #define SS_2_PIN        A1
15
16 #define NR_OF_READERS   2
17
18 String RFID_EXTERNAL = "";
19 String RFID_1_INTERNAL = "";
20 String RFID_2_INTERNAL = "";
21 String RFID_3_INTERNAL = "";
22
23 byte ssPins[] = {SS_1_PIN, SS_2_PIN};
24
25 MFRC522 mfrc522[NR_OF_READERS];
```

Fonte: Autor (2022)

Para verificar se existe alguma *tag* RFID sendo lida, foi criada a função *reads_rfids()*, como mostra a figura 34. Nessa função, todos os leitores conectados ao microcontrolador salvam, em variáveis diferentes, as informações captadas pelas *tags*. No caso hipotético de um usuário que acabou de usar a baia, no momento em que ele fechá-la é nessa função que serão feitas as capturas dos dados das *tags* de todos os equipamentos presentes, para posteriormente esses dados passarem por algum tratamento e serem enviados para o *gateway*.

Esse tratamento é feito nas duas funções presentes da figura 35, a *check_equipament()* e *concatenate_rfid(String INTERNAL_RFID)*. A primeira retorna um texto que é a junção de todos os equipamentos lidos separados por um caractere específico e a segunda função é chamada

Figura 34 – Função que captura as leituras das *tags* RFID

```
232 void read_rfids(){
233   String uidTag = "";
234   for (uint8_t reader = 0; reader < NR_OF_READERS; reader++) {
235     mfrc522[reader].PCD_Init(ssPins[reader], RST_PIN);
236     if (mfrc522[reader].PICC_IsNewCardPresent() && mfrc522[reader].PICC_ReadCardSerial()) {
237       uidTag = dump_byte_array(mfrc522[reader].uid.uidByte, mfrc522[reader].uid.size);
238       if (reader == 0){
239         RFID_EXTERNAL = uidTag;
240       } else if (reader == 1) {
241         RFID_1_INTERNAL = uidTag;
242       } else if (reader == 2) {
243         RFID_2_INTERNAL = uidTag;
244       } else {
245         RFID_3_INTERNAL = uidTag;
246       }
247       mfrc522[reader].PICC_HaltA();
248     }
249   }
250 }
```

Fonte: Autor (2022)

apenas pela primeira e serve para concatenar o valor correspondente à *tag* lida com o separador, que neste caso é um sinal de ponto-e-vírgula.

Figura 35 – Funções para tratarem as *tags* lidas

```
144 String checkEquipament() {
145   String equipament = "";
146   if (RFID_1_INTERNAL != "") {
147     equipament.concat(concatRFID(RFID_1_INTERNAL));
148   }
149   if (RFID_2_INTERNAL != "") {
150     equipament.concat(concatRFID(RFID_2_INTERNAL));
151   }
152   if (RFID_3_INTERNAL != "") {
153     equipament.concat(RFID_3_INTERNAL);
154   }
155   return equipament;
156 }
157
158 String concatRFID(String INTERNAL_RFID) {
159   String equipamentInCorrectFormat = "";
160   const String semicolon = ";";
161   equipamentInCorrectFormat.concat(INTERNAL_RFID);
162   equipamentInCorrectFormat.concat(semicolon);
163   return equipamentInCorrectFormat;
164 }
```

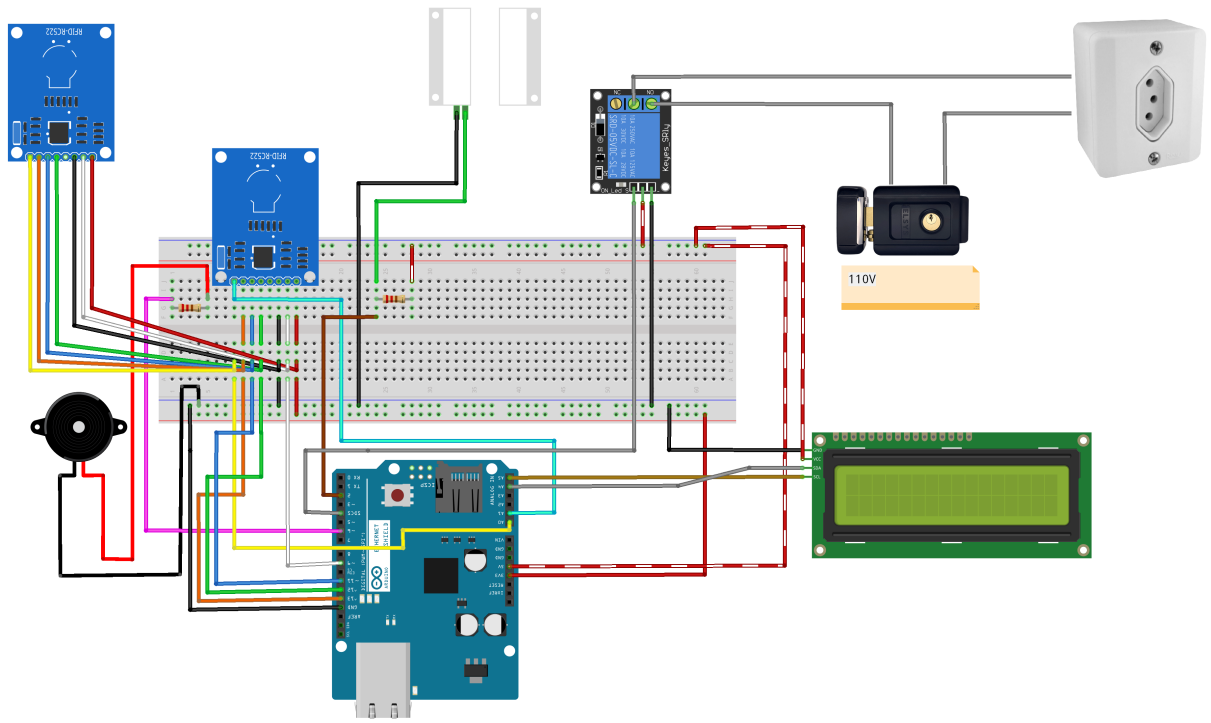
Fonte: Autor (2022)

6.1.2 Submódulo de Monitoramento da Baia

Como já dito anteriormente, o protótipo foi desenvolvido utilizando uma placa Arduíno e vários outros sensores. Em dados momentos esses sensores precisam conversar entre si e para isso eles precisam estar integrados da maneira correta, com a tensão adequada e com os pinos ligados às portas correspondentes do microcontrolador.

Sabendo disso, para conseguir organizar os componentes, reaproveitar portas e não precisar soldar nenhum dos módulos em alguma placa foi usada uma *protoboard* que, embora não sirva para a implementação final, é uma placa de ensaio muito útil para quem precisa testar vários componentes em um mesmo circuíto. A figura 36 demonstra como os componentes estão interligados ao Arduíno.

Figura 36 – Esquemático do protótipo proposto



Fonte: Autor (2022)

Dentre módulos conectados estão: dois leitores RFID, um sensor magnético, um módulo relê, um módulo *buzzer* e um display LCD. Ligado ao módulo relê está a fechadura eletrônica, que diferente do padrão de tensão do estado de Pernambuco, aceita apenas a voltagem de 110V por restrição de tensão do laboratório.

É possível observar também na figura 36 que os módulos RFID têm suas portas ligadas em paralelo, com exceção do SS (*Slave Select*). Isso porque o sinal do SS escolhe para onde vai a informação que circula em cada barramento, não permitindo o compartilhamento de sinais.

Os outros módulos têm ligações menos complexas, o *buzzer* e o sensor magnético por exemplo estão ligados em série ao GND e um resistor *pull-up* de 220 Ohms que altera o nível de resistência e o nível lógico das portas.

O display LCD por sua vez possui dois pinos de alimentação conectados na *protoboard*: o GND e o VCC, sendo esse último o pino com tensão de 5V. Os outros dois pinos são ligados direto no microcontrolador sendo eles responsáveis pela comunicação com a interface I2C com o objetivo de mostrar as mensagens no LCD.

Por último o módulo relê, que também tem dois pinos de alimentação conectados à *protoboard*, os mesmos do display LCD, mas conectado ao microcontrolador tem apenas um que realiza o envio de pulsos elétricos que acionam o interruptor do componente.

Na outra ponta desse componente estão os conectores parafusáveis que, embora tenham três, serão utilizados apenas dois, um que estará ligado a uma tensão de até 220V e o outro ao equipamento eletrônico que deverá ser acionado, uma fechadura neste caso. Na outra ponta desse componente estão os contatos de acionamento dos equipamentos eletrônicos. Conforme é possível observar na imagem, apenas dois desses contatos são utilizados: o comum e o NA (normalmente aberto), que impede a passagem de corrente elétrica enquanto o equipamento não é acionado.

6.1.3 Comunicação entre Arduíno e o gateway

Após realizar a leitura de um cartão, alguns passos são realizados para preparar um conjunto de dados que tem algum significado nas camadas superiores da arquitetura, esses passos serão detalhados em seções seguintes. Após realizada a preparação dos dados, eles são enviados para um servidor local e depois de feito todo o processamento e analise é devolvido um outro pacote deste servidor para o Arduíno com uma resposta da solicitação.

Toda essa transferência de pacotes é feita por meio de uma rede local com fio criada unicamente com essa finalidade. O protocolo que concretiza a comunicação é o WebSocket, que tem como ponto positivo, muito importante para a realização deste trabalho, a comunicação bidirecional, ou seja, as mensagens podem ser enviadas tanto pelo cliente como pelo servidor. Essa comunicação acontece de forma análoga à correspondências do mundo real, visto que também precisa de um endereçamento, que nesse caso é composto de uma porta e um IP. Nesse cenário um cliente envia para um servidor e um servidor envia para todos os clientes conectados na rede, desde que estejam conectados no mesmo endereço.

Na figura 37 é possível observar a estruturação, o envio e o recebimento dos pacotes. Esse código roda continuamente no microcontrolador e tem algumas condições para que sejam de fato executados. Para uma melhor compreensão, a figura 38 mostra o diagrama de blocos que corresponde à execução do código da figura 37.

A linha 105 mostra uma condição que só é verdadeira se o sensor magnético estiver no modo que indique que a baia está fechada, *LOW* neste caso, e se houver alguma *tag* sendo lida

Figura 37 – Estruturação, envio e recebimento dos pacotes no módulo da baia

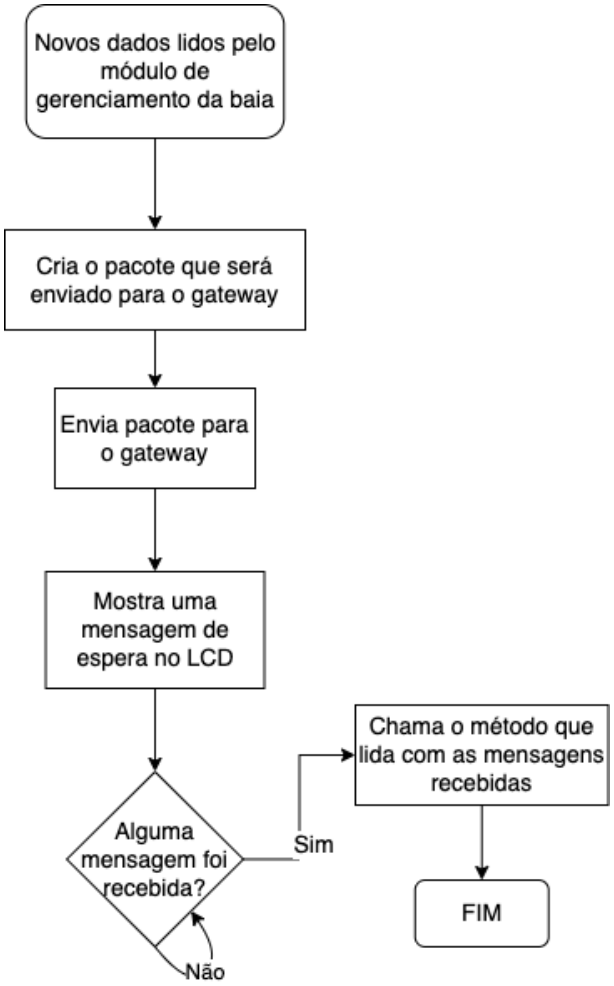
```
105  if(digitalRead(pinMagneticSensor) == LOW && RFID_EXTERNAL != "") {
106    strcpy(joinStringsToSend,"1#");
107    strcat(joinStringsToSend,BAIA.c_str());
108    strcat(joinStringsToSend,"#");
109    strcat(joinStringsToSend,RFID_EXTERNAL.c_str());
110    Serial.println(joinStringsToSend);
111    char tBuffer[50];
112    snprintf(tBuffer, sizeof(tBuffer), "%s", joinStringsToSend);
113    server.write(tBuffer);
114    RFID_EXTERNAL = "";
115    lcd_message("Aguarde um instante");
116    delay(1000);
117  } else if(digitalRead(pinMagneticSensor) == LOW && lastState==1 &&
118              !alreadStartedLock && server_connected){
119    char joinStringsToSend[50];
120    String equipamentos = check_equipament();
121    strcpy(joinStringsToSend,"2#");
122    strcat(joinStringsToSend,BAIA.c_str());
123    strcat(joinStringsToSend,"#");
124    strcat(joinStringsToSend,equipamentos.c_str());
125    char tBuffer[50];
126    snprintf(tBuffer, sizeof(tBuffer), "%s", joinStringsToSend);
127    server.write(tBuffer);
128    alreadStartedLock = true;
129    lcd_message("Fechando baia...");
130  }
131  EthernetClient client = server.available();
132  if(client){
133    receiveResponseFromServer(client);
134  }
```

Fonte: Autor (2022)

no leitor que fica do lado de fora, o que significa que alguém aproximou seu crachá para tentar abrir a baia. Se essa condição for satisfeita o programa concatena alguns valores separados pelo caractere *# (hashtag)*, conforme apresenta a figura 39, com a finalidade de enviar em um único pacote várias informações, que nesse caso são três. O primeiro é o tipo de solicitação, ou seja, o numeral 1 se for para abrir a baia e o numeral 2 se for pra fechar, nesse caso sempre será 1. Em seguida é concatenado o número correspondente à baia que é uma constante pré-estabelecida, e por último o valor que é lido pelo sensor RFID. Após esse processo, o pacote é enviado ao *gateway* através do código escrito na linha 113, *server.write(MENSAGEM)*.

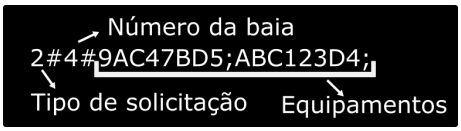
A condição seguinte, na linha 117, começa da mesma forma da anterior verificando se a baia está fechada, mas ela verifica também se o estado anterior dela era aberto, pela variável *lastState*, se não tem nenhuma solicitação de fechamento da baia em andamento e se ele está conectado ao servidor. Essa última verificação não ocorre na condição anterior porque se não

Figura 38 – Fluxograma da estruturação, envio e recebimento dos pacotes no módulo da baia



Fonte: Autor (2022)

Figura 39 – Pacote enviado para o *gateway*



Fonte: Autor (2022)

houver conexão com o servidor o usuário vai aproximar o cartão e nada vai acontecer, ou seja, a segurança dos equipamentos não será comprometida. Porém se o servidor não estiver conectado e o código dentro desta condição for executado alguma informação pode ser perdida.

Por último, as quatro últimas linhas da figura servem para verificar se há uma nova mensagem. Caso haja, a função dentro da condição da linha 132 é chamada, a mensagem recebida é devidamente tratada e a resposta da solicitação é dada ao usuário.

6.2 DESENVOLVIMENTO NO GATEWAY

O *gateway* tem uma função muito importante no projeto, é nele que passam todos os dados que entram e saem do microcontrolador e é através dele que os usuários conseguem acompanhar o uso das baias em tempo real. Mas além disso, o *gateway* também assegura que as informações contidas na nuvem terão uma cópia de segurança e também garante que nenhuma movimentação que possa comprometer os equipamentos seja perdida mesmo em situações adversas, como falta de internet por exemplo.

Esse *gateway* opera em uma placa com um poder de processamento maior que o Arduíno, isso porque essa placa, conhecida no mercado como Raspberry PI, é um mini computador projetado para as mais diversas aplicações, tais como: servidor, central de mídia, sistema de monitoramento, etc. Sendo assim, esse Raspberry PI, mostrado na figura 40, é ideal para o projeto desenvolvido neste trabalho pois consegue desempenhar a função de servidor fazendo toda a comunicação e processamento dos dados.

Figura 40 – Raspberry Pi 3

Fonte: www.filipeflop.com (2022)

6.2.1 Servidor Local

Para criar um servidor local que possa ser acessado por qualquer dispositivo da rede a partir de um endereço, é preciso de tecnologias adequadas e que tenham a capacidade de estabelecer um serviço que fique sempre ativo à espera de uma nova requisição. A tecnologia escolhida foi o Node.JS que executa um *serve-side* com a linguagem de programação *JavaScript* e é amplamente utilizado por grandes empresas em todo o mundo. O ponto forte dessa tecnologia

é o poder de escalabilidade, visto que diferente de outras já existentes ela consegue lidar com várias requisições simultâneas sem comprometer os recursos computacionais da máquina.

Para a criação desse serviço foi utilizada a biblioteca *Net* que estabelece um canal de comunicação baseado no protocolo WebSocket e possibilita o recebimento e envio de pacotes, como mostra a figura 41. É possível observar na linha 13 a criação do serviço que ficara ativo e na linha 16 habilitando o recebimento de pacotes.

Figura 41 – Código de conexão e troca de mensagens no *gateway*

```
13  const server = net.createServer((activeServer) => {
14      activeServer.setEncoding("utf8");
15      console.log('Uma baia foi conectada');
16      activeServer.on('data', function (data) {
17          const dataRequested = data + "";
18          const codigosEnviadosPlaca = dataRequested.split("#")
19          if (codigosEnviadosPlaca[0] == abrirBaiaCode) {
20              webserver.abrirBaia(codigosEnviadosPlaca[1], codigosEnviadosPlaca[2])
21                  .then(function (val) {
22                  var res = val.num_baia + "#";
23                  if (val.code == 200) {
24                      activeServer.write(res + 'allowed\n');
25                  }  else if (val.code == 10) {
26                      activeServer.write(res + 'blocked\n');
27                  } else if (val.code == 204) {
28                      activeServer.write(res + 'notfound\n');
29                  } else {
30                      activeServer.write(res + 'danied\n');
31                  }
32              });
33          } else if (codigosEnviadosPlaca[0] == fecharBaiaCode) {
34              webserver.fecharBaia(codigosEnviadosPlaca[1], codigosEnviadosPlaca[2])
35                  .then(function (val) {
36                  var res = val.num_baia + "#";
37                  if (val.code == undefined) {
38                      const insertInOfflineTable = query.scriptInsert_RunningOffline(val);
39                      database.runSqlScript(insertInOfflineTable);
40                      activeServer.write(res + 'no_network\n');
41                  } else {
42                      if (val.code == 200 || val.code == -1) {
43                          activeServer.write(res + 'locked\n');
44                      } else {
45                          activeServer.write(res + 'blocked\n');
46                      }
47                  }
48              });
49          } else {
50              console.log('Received: ' + dataRequested)
51          }
52      }
53  );
```

Fonte: Autor (2022)

Ao chegar uma nova mensagem, seu conteúdo é separado para facilitar a verificação das informações contida nele. Na linha 19, por exemplo, está sendo feita uma comparação entre o dado recebido e o dado que o sistema entende que corresponde a uma solicitação de abertura de baia. O mesmo acontece na linha 33, mas dessa vez a comparação é feita para verificar se a

solicitação é um fechamento de baia, e se não for nenhum dos dois casos o sistema simplesmente ignora e não faz nenhuma ação.

No caso de alguma dessas comparações seja positiva as outras informações presentes no pacote de dados serão concatenadas à URL da *API Rest* que, assim que receber a solicitação, fará as análises necessárias para a validação dessa solicitação e retornará uma resposta para o *gateway*. Essa URL é dividida em 3 partes: o *endpoint*, que é a URL base para qualquer outra requisição do sistema; a rota, que direciona a solicitação para a função correspondente; e os parâmetros, que correspondem às informações do usuário, ou do equipamento, e da baia. A figura 42, ilustra um exemplo de uma URL que solicita o fechamento de uma baia, nela é possível observar as marcações de onde estão a base, a rota e os dados.

Figura 42 – URL de requisição para fechar baia

Fonte: Autor (2022)

Em caso de falta de internet, especificamente quando a solicitação for para fechar a baia, essa URL será salva em um banco de dados local e consultada após a conexão ser reestabelecida. Todas as requisições pendentes serão feitas novamente e a URL será apagada no decorrer que essas requisições sejam bem sucedidas.

Após o momento que é identificado que não há conexão com a internet, o *gateway* de 15 em 15 segundos executa o algoritmo da figura 43, na qual, a linha 153 usa a biblioteca *check-internet-connected* para verificar se a conexão foi reestabelecida, caso tenha sido o *gateway* executa as requisições e avisa ao subsistema de controle de baia. Caso não tenha sido reestabelecida, apenas imprime uma mensagem na tela sinalizando que ainda está sem conexão.

6.2.2 Comunicação com o *backend*

Embora o *gateway* tenha poder computacional para ter um banco de dados próprio e realizar consultas e registros, os dados armazenados localmente não seriam de fácil acesso para um sistema web, por exemplo. Por isso, foi desenvolvido um *backend*, hospedado na nuvem, que centraliza os dados e os torna acessíveis por meio da *API Rest*.

As requisições para essa *API Rest* são feitas utilizando a biblioteca *Axios* que recebe como parâmetro apenas a URL que será usada para realizar a solicitação. É possível observar na 44 dois exemplos de requisições, uma utilizando o método *post*, na linha 39, e outra o método *get*, na linha 50.

A primeira função envia parâmetros para a *API Rest* referente a alguma solicitação do usuário, seja abertura ou fechamento de uma baia. A segunda função não envia parâmetros, apenas acessa uma rota e recebe informações dela, essa função é útil para checar se existem novos

Figura 43 – Algoritmo que verifica a conexão com a internet

```
var testeConectividade = function testeConectividade() {
  const config = {
    timeout: 5000,
    retries: 3,
    domain: "google.com",
  }
  checkInternetConnected(config)
  .then(() => {
    database.checkOfflineRegistersTable();
    if (estaOffline) {
      console.log("Conexão reestabelecida.");
      client.write('locked\n');
      estaOffline = false;
    }
  })
  .catch(() => {
    console.log("Sem internet.");
    estaOffline = true;
  });
}
```

Fonte: Autor (2022)

dados a serem salvos como *backup*, por exemplo. A funcionalidade de backup, inclusive, será mostrada com mais detalhes na seção seguinte.

6.2.3 Banco de dados local

Embora exista um banco de dados na nuvem onde serão feitas praticamente todas as principais consultas do sistema, se viu a necessidade de criar também um banco de dados local para que haja uma cópia de segurança desse banco na nuvem, e essa cópia é implementada no *gateway*.

As tabelas são praticamente as mesmas e banco de dados do *gateway* foi todo implementado em MariaDB. Foi usado esse banco de dados porquê além dele ser baseado no MySQL, ou seja, possuir uma alta compatibilidade com o banco de dados usado na nuvem, ele possui um método de busca e consulta mais otimizado, melhorando o desempenho na execução das consultas. Esse fator é muito importante tendo em vista que o Raspberry PI possui recursos computacionais bem limitados em comparação com muitos servidores na nuvem.

A implementação propriamente dita desse banco de dados será explorada com mais detalhes na sub-seção 6.3.3, nesta seção será mostrada apenas a diferença entre o banco de dados local e o banco de dados na nuvem. A figura 45 mostra o esquemático e o relacionamento entre as tabelas implementadas.

É possível observar que apenas duas tabelas estão destacadas, enquanto as outras não, isso porquê essas duas tabelas existem apenas no *gateway*, as outras estão no servidor e possuem os

Figura 44 – Código utilizado para se comunicar com a *API Rest*

```
37  var postAPI = function postAPI(url) {
38    return new Promise(function (resolve) {
39      axios.post(url).then(response => {
40        resolve(response);
41      }).catch(() => {
42        resolve("400_network_error");
43      })
44    });
45  };
46
47  var getAPI = function getAPI(path) {
48    return new Promise(function (resolve, reject) {
49      const fullUrl = URL + "" + path;
50      axios.get(fullUrl).then(response => {
51        resolve(response);
52      }).catch(error => {
53        reject("400_network_error");
54      })
55    });
56  };
```

Fonte: Autor (2022)

mesmos campos. A primeira dessas duas tabelas é a *registro_offline* que é usada para guardar as URL's das requisições que não puderam ser feitas por algum problema de comunicação, falta de internet por exemplo.

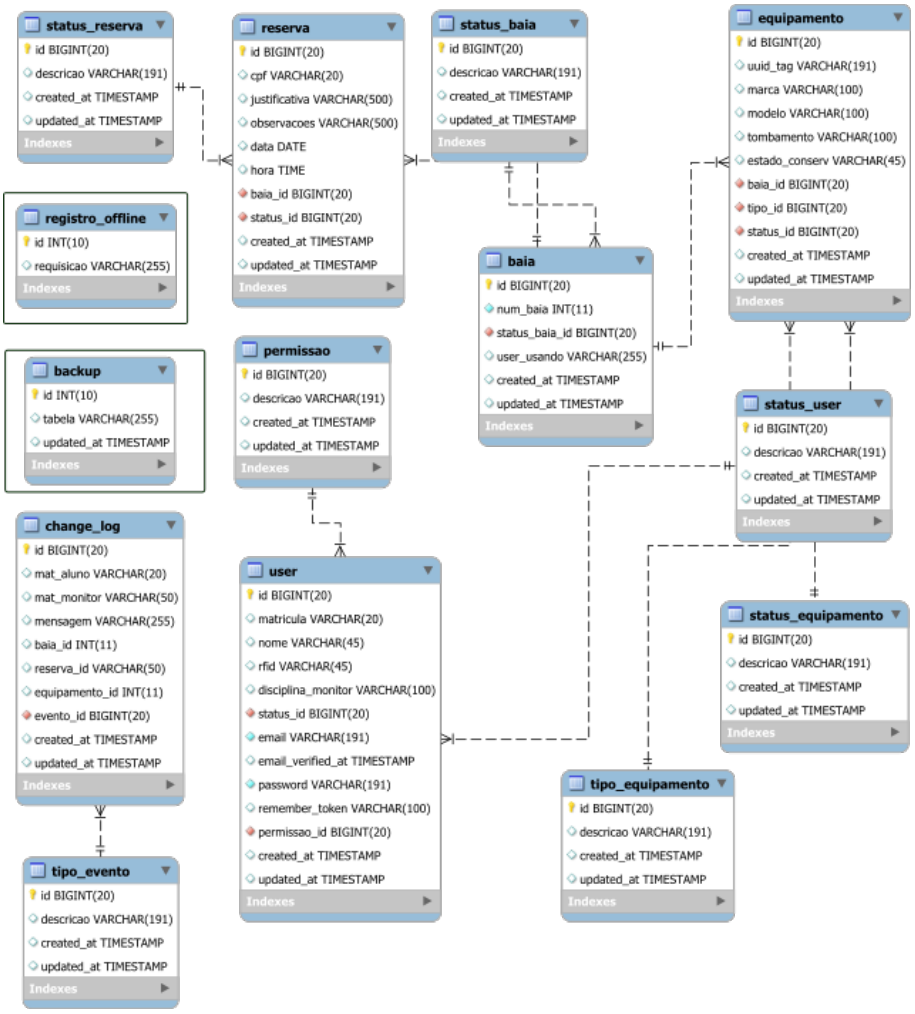
A segunda tabela é a de *backup* que é alimentada todos os dias em um horário predefinido com todas as novas informações contidas no servidor, a implementação da funcionalidade de recepção e atualização dessas novas informações pode ser vista na figura 46.

Essa função é chamada diariamente em um horário que normalmente há pouco ou nenhum movimento no laboratório, o que diminui a probabilidade de haver algum tipo de sobrecarga no *gateway* no caso de haver uma grande quantidade de dados vindos do servidor.

Logo na primeira linha, é chamada uma outra função responsável por pegar os dados da tabela de *backup*. Feito isso, na linha 103 o programa checa se a tabela está vazia e se estiver ele solicita na nuvem todos os dados contidos lá para salvar localmente. Caso haja dados, é feita uma requisição por tabela buscando os dados de cada uma a partir do horário da última atualização.

É válido ressaltar que, para não ocupar memória com dados que não serão mais usados ou precisam de atualização, os dados armazenados localmente nas tabelas "reserva", "change_log"e "user"são deletados a cada 5 meses, que o tempo médio que dura um semestre letivo. São deletados os dados apenas dessas tabelas porque elas são as que mais recebem novas entradas e, no caso da tabela de "user", a que passa por reformulação todo o semestre com a inserção de novos alunos e a exclusão dos antigos.

Figura 45 – *Schema* do banco de dados local



Fonte: Autor (2022)

Figura 46 – Função de *backup* diário

```
var dailyBackup = function dailyBackup() {
  runSqlScript(query.checkBackup()).then(data => {
    const responseJSON = JSON.parse(data);
    if (responseJSON.length == 0) {
      webserver.getAPI(pathToAllData).then(function(val) {
        var rows = val.data;
        feedDatabaseWithAllData(rows);
      }).catch(err => {
        console.log(err);
      });
    } else {
      responseJSON.forEach(function(response) {
        if (isMutableDataTable(response.tabela)){
          runSqlScript(query.removeOldRegister()).then(() => {
            console.log(`Dados antigos removidos`);
          }).catch(function(rej) {
            console.log(`Não foi possível remover
              os dados antigos. Motivo: ${rej}`);
          });
        }
        getUpdatedData(response.tabela, response.updated_at);
      });
    }
  });
}
```

Fonte: Autor (2022)

6.3 DESENVOLVIMENTO WEB

O desenvolvimento web foi a parte mais extensa e complexa do trabalho, isso porque foi preciso desenvolver duas aplicações para finalidades distintas. O primeiro a ser desenvolvido foi um portal, intitulado de *LABCIN*, que possibilitaria aos estudantes e funcionários acompanhar em tempo real o uso das baias. Nele muitas regras de negócio tiveram que ser aplicadas e o mesmo portal tem funcionalidades diferentes de acordo com os níveis de acesso.

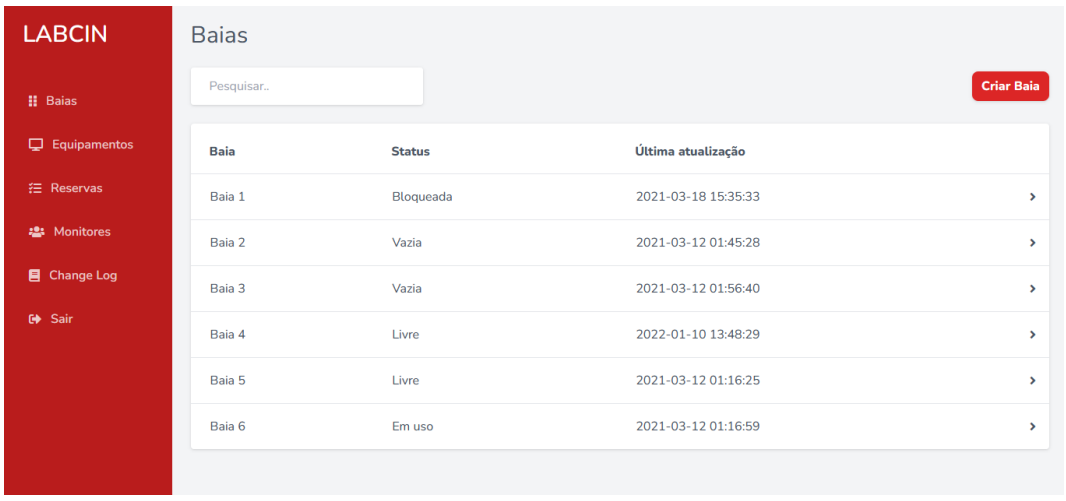
A segunda aplicação é o *backend*, que já foi citada na sessão anterior, é responsável por lidar com todas as requisições que vem do *gateway* e algumas feitas pelo portal. É através dos dados fornecidos por ela que o usuário consegue abrir e fechar uma baia em segurança, por exemplo.

6.3.1 Frontend

O portal pode ser usado por qualquer aluno e funcionário, entretanto existem níveis de acesso. A figura 47 mostra um usuário que é monitor ou um funcionário, enquanto na figura 48 o usuário era apenas um aluno, para entender a diferença basta observar o menu lateral onde a primeira figura possui mais opções que a segunda. Essa distinção é extremamente importante para permitir que apenas pessoas autorizadas adicionem, editem ou excluam alguma informação. Os alunos estão restritos à visualizarem apenas informações a respeito das baias e dos equipamentos, além de poderem fazer reservas das baias para dias e horários não úteis.

A tabela possui três informações importantes para quem deseja consultar o status de determinada baia, são elas: qual a baia, o status dela naquele momento e qual o último registro, seja ele cadastro, atualização ou uso.

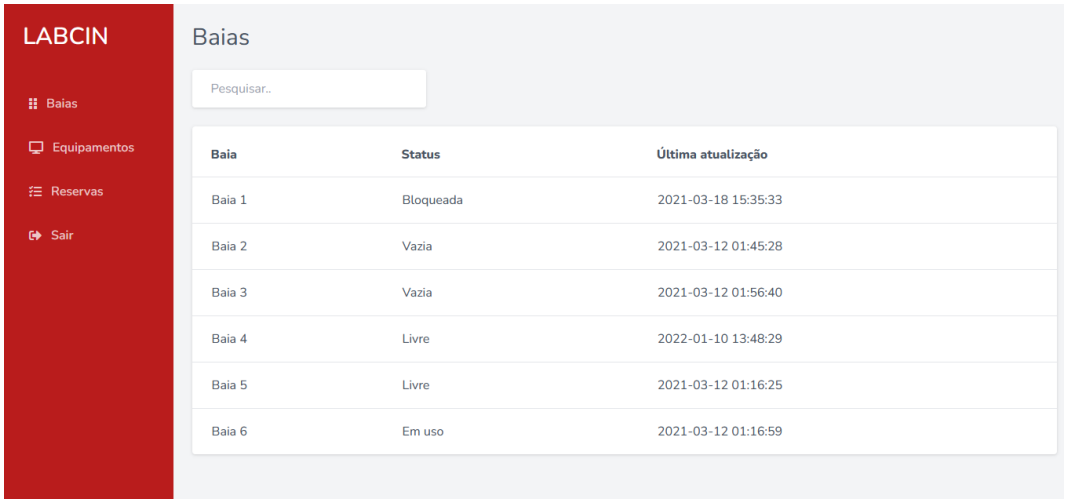
Figura 47 – Página de Baia para o monitor



Fonte: Autor (2022)

Ao clicar no botão ou em algum item da lista, o usuário é redirecionado para a página de adicionar ou editar uma baia, a figura 49 mostra esta tela com mais detalhes. Hoje cada

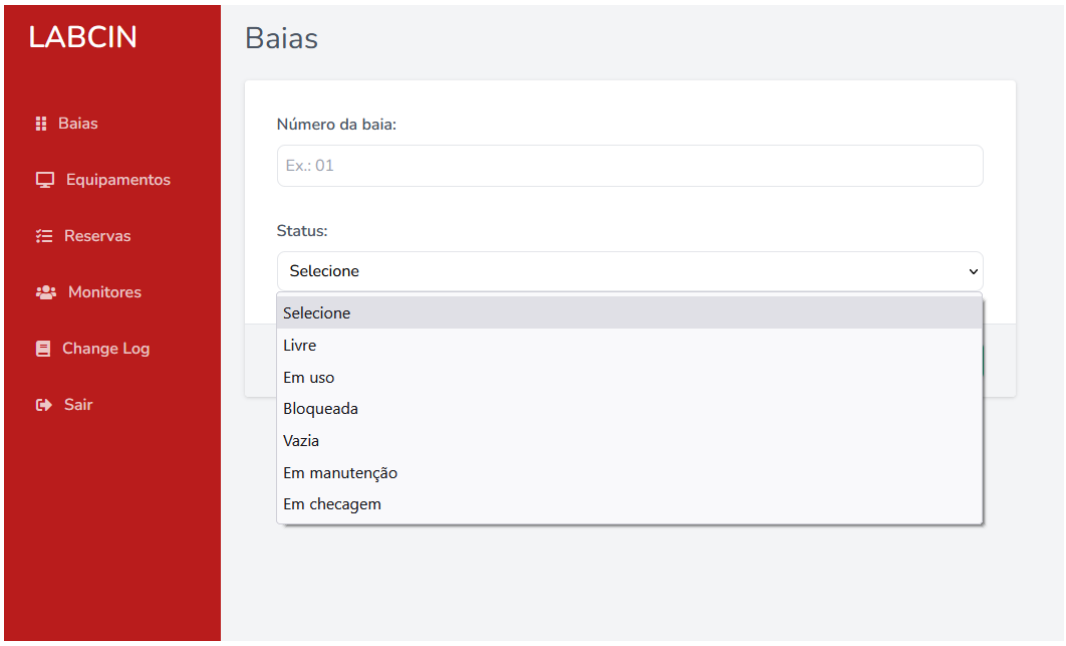
Figura 48 – Página de Baia para o aluno



Fonte: Autor (2022)

baia no laboratório possui uma numeração, e essa numeração é usada nesse primeiro campo para facilitar a identificação. O segundo campo é para selecionar o status atual dessa baia, pois podem existir baias vazias ou em manutenção e o cadastro apenas vai fazer com que elas estejam no sistema mesmo sem a possibilidade de utilização.

Figura 49 – Página de cadastrar uma nova baia

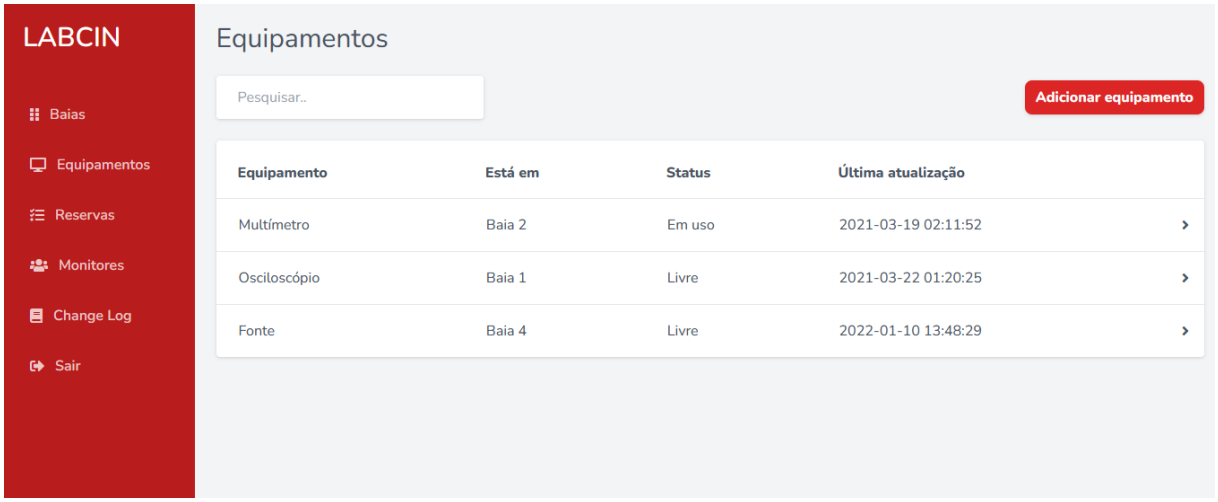


Fonte: Autor (2022)

A próxima página é a de equipamentos, mostrada na figura 50, que se assemelha com a anterior pela sua aparência e por conter os mesmos dados como a indicação da baia, o status e o último registro. A única diferença está na primeira coluna que informa qual é o tipo de equipamento, ou seja, cada linha diz para o usuário em qual baia cada equipamento está, o

status de uso dele e qual a última atualização.

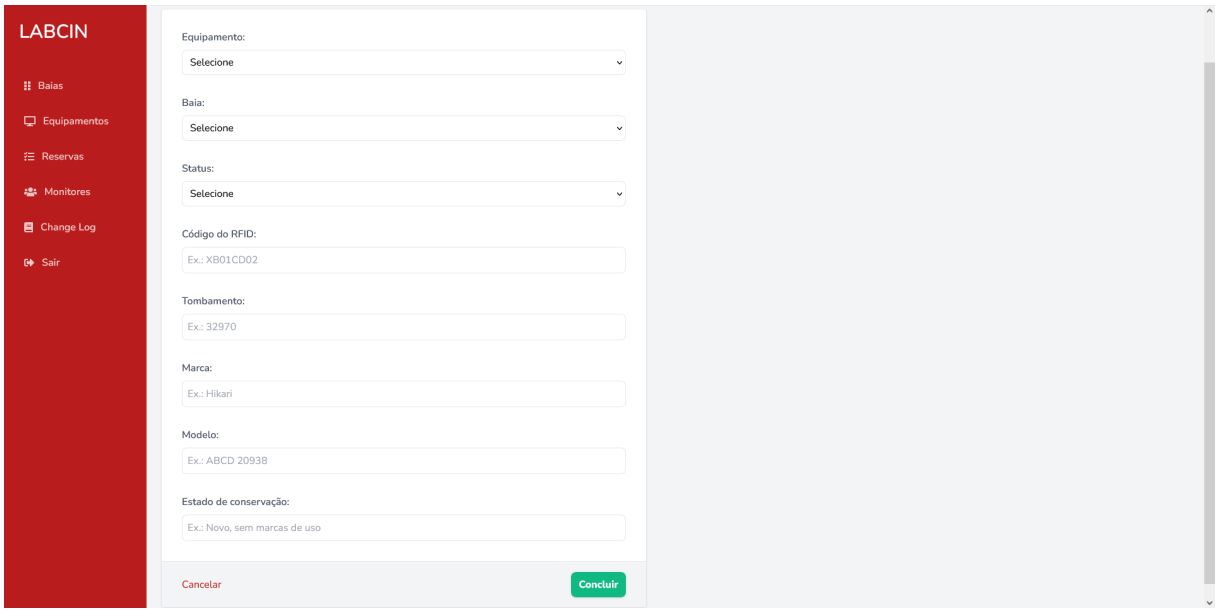
Figura 50 – Página de listagem de equipamentos



Fonte: Autor (2022)

Selecionando a linha do equipamento ou o botão de adicionar, o monitor ou funcionário é redirecionado para a página da figura 51 onde pode editar os dados que estão preenchidos, se ele clicou na linha do equipamento, ou inserir novos dados caso ele tenha clicado no botão. Nesse formulário deverão ser informado dados como: tombamento, marca, tipo, estado de conservação, o valor da *tag* RFID atrelado a ele, em qual baia ele está e o seu status.

Figura 51 – Formulário de cadastro de equipamento



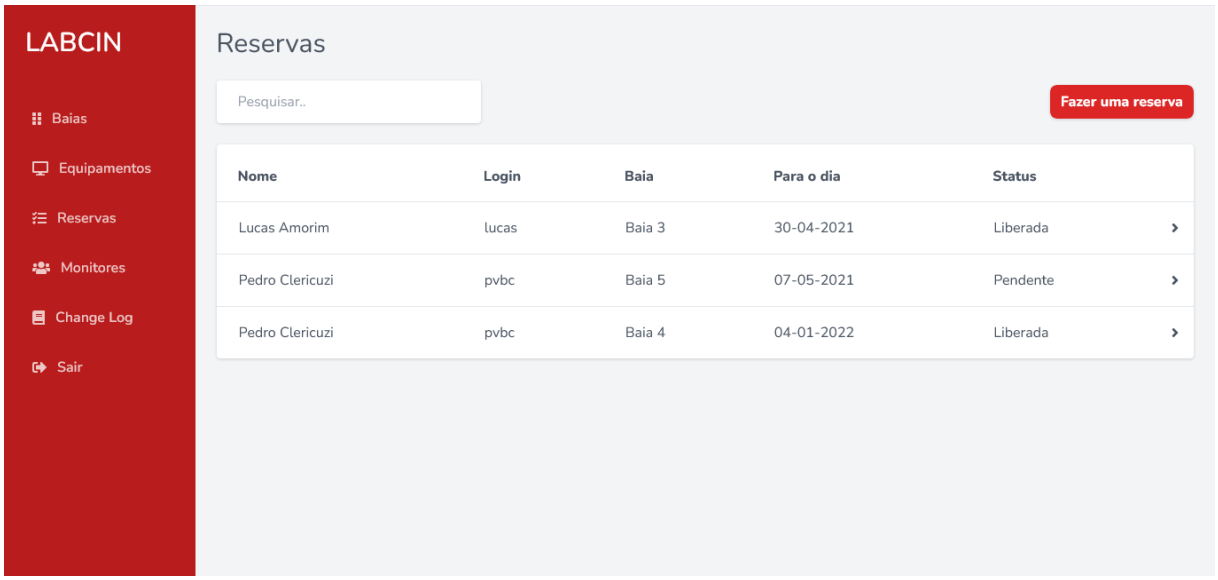
Fonte: Autor (2022)

A última página em comum com todos os usuários é a de reserva, que funciona apenas para manter um controle de uso fora do horário de aula. Seu *layout* segue o padrão e estrutura das

outras páginas, como mostra a figura 52, contendo uma tabela e um botão. Na tabela o aluno pode ver seu nome e login, a baia que ele reservou, o dia e a data da última atualização. No caso dos monitores e funcionários eles verão a mesma tabela só que as reservas de todos os alunos estarão sendo visualizadas.

Isso é necessário porque o laboratório conta com um número limitado de baias que não é suficiente para todos os alunos ao mesmo tempo, além disso o CIn é um centro que possui diversos eventos durante o ano e esses eventos eventualmente podem precisar ocupar uma parte ou todo o laboratório e, caso isso aconteça, os alunos poderão visualizar as baias que não estão disponíveis e escolher um outro momento para usá-las ou selecionar alguma outra que esteja livre.

Figura 52 – Página de reservas



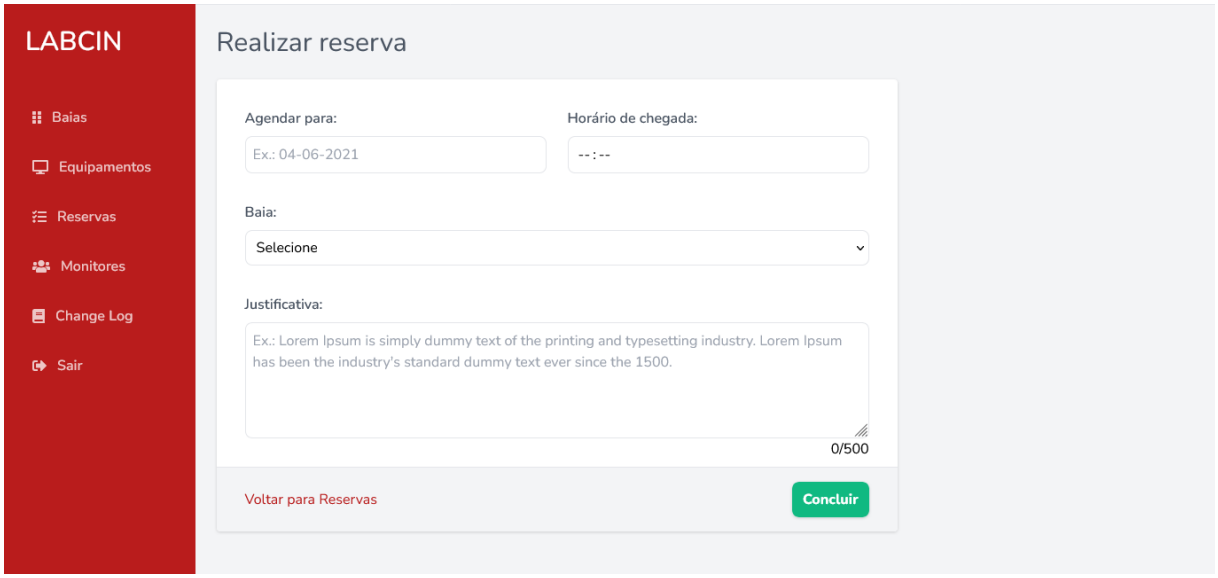
Fonte: Autor (2022)

Para criar uma reserva, basta clicar no botão no canto superior direito e o usuário será redirecionado para a página da figura 53, onde ele precisará preencher todos os campos e aguardar até que alguém autorizado aceite ou recuse, o usuário será notificado por e-mail assim que houver uma definição.

A página seguinte, mostrada na figura 54, lista todos os monitores e nessa listagem é possível encontrar as seguintes informações: nome, e-mail, a disciplina que está monitorando e, como nas outras páginas, a data da última atualização de cadastro desse monitor. A única diferença das linhas dessa tabela pra as outras é que o ícone do canto direito é uma lixeira, que deve ser usado sempre que for preciso excluir o respectivo aluno do quadro de monitores.

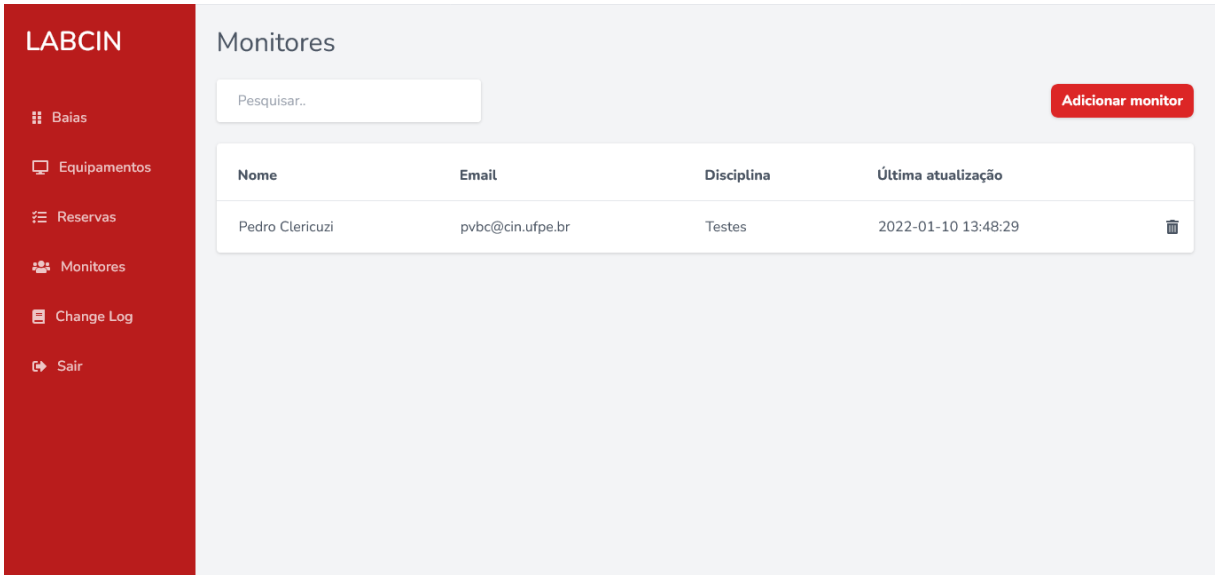
A página da figura 55 é acessada a partir da anterior, e sua finalidade é conceder a permissão de monitor a um aluno. Nela são mostrados dois campos: matrícula e a disciplina que ele monitora. Antes de salvar é feita uma checagem no *backend* se aquele número de matrícula (CPF)

Figura 53 – Página de criar uma nova reserva



Fonte: Autor (2022)

Figura 54 – Página de listagem de monitores

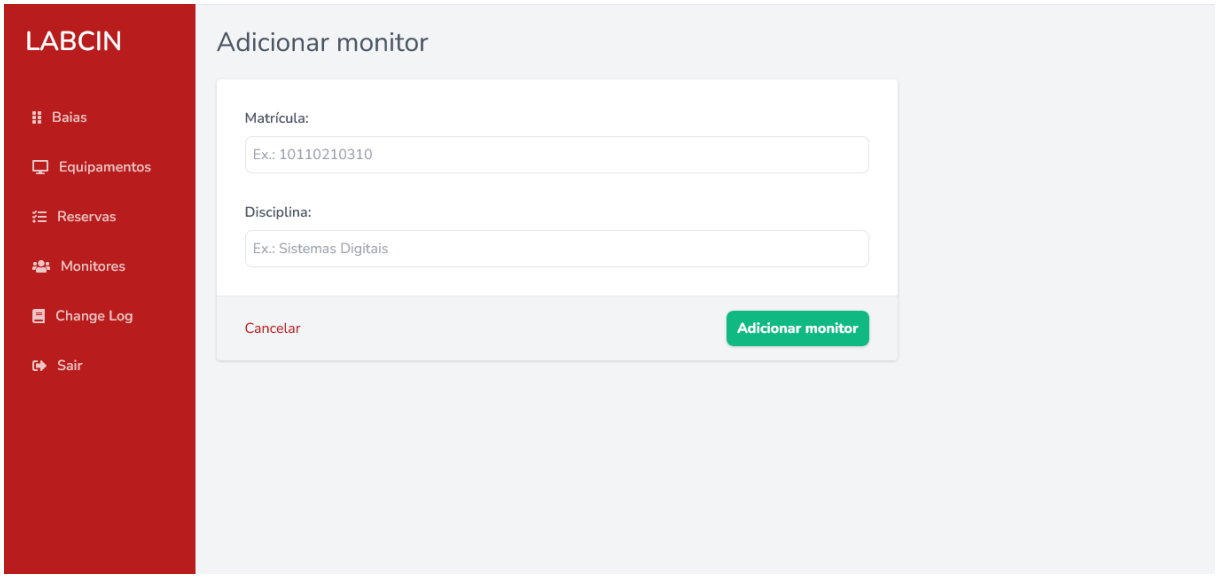


Fonte: Autor (2022)

realmente corresponde a um aluno e, se de fato for, a permissão é concedida e a partir de então esse aluno passa a ter o mesmo nível de acesso de todos os monitores.

Por último na página de log, mostrada na figura 56, é possível encontrar a listagem de todos os acontecimentos separados pelos seus tipos, que podem ser: baias, equipamentos e reservas. Ou seja, tudo que acontecer com cada um será feito um registro acessível para os monitores e funcionários que, se questionados, podem fazer uma consulta rápida nessa página e resolver o ocorrido. As informações contidas em cada linha correspondem a um evento que contém a

Figura 55 – Página de adicionar monitor



Fonte: Autor (2022)

matrícula de um responsável, seja ele aluno ou monitor, uma mensagem curta pra se ter uma noção do que ocorreu naquele evento, o tipo desse evento que pode representar um uso, uma atualização de dados, um erro ou um registro, tem também uma coluna que mostra quando esse evento ocorreu e a data de atualização que corresponde à criação dele.

Figura 56 – Página de Log

Fonte: Autor (2022)

6.3.2 Backend

O *backend* é um subsistema fundamental para gerenciar o fluxo do uso das baias com segurança, ela tem a importante tarefa de receber o dado, fazer as consultas necessárias para validar esse dado, atender ou não às solicitações realizadas e enviar a resposta ao solicitante. Inclusive os dados alterados neste subsistema influenciam diretamente nas informações mostradas no portal, visto que ambos compartilham do mesmo banco de dados.

Das várias funções existentes apenas quatro são chamadas diretamente pelo *gateway*, isso não quer dizer que apenas elas são importantes, cada uma delas é composta por um conjunto de outras funções que contribuem para melhorar a qualidade do código e ao mesmo tempo realizar todas as regras de negócio.

Para facilitar o entendimento de cada uma dessas funções serão mostrados fluxogramas que irão dar uma visão geral da lógica por trás delas, sem que seja necessário ler dezenas linhas de código.

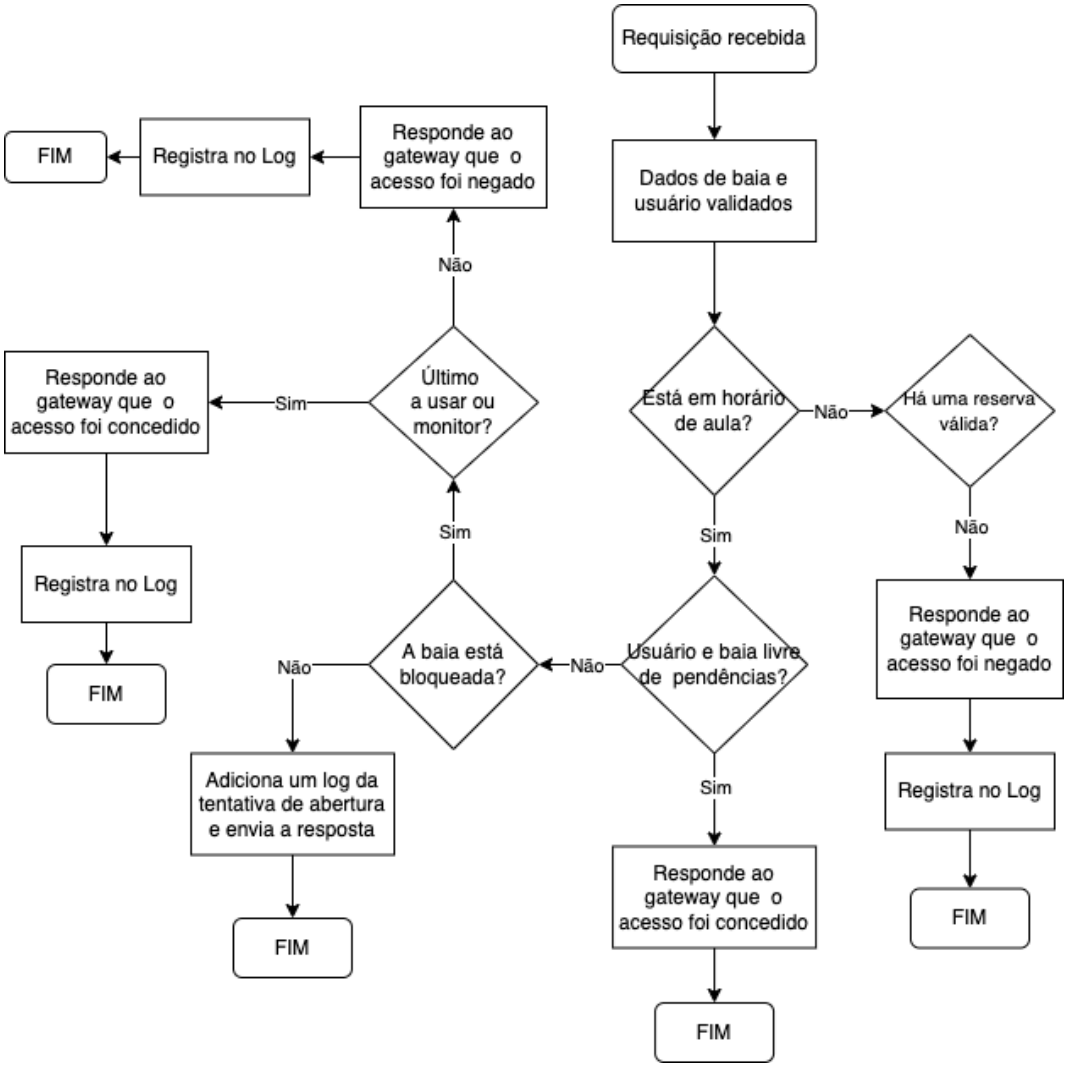
O fluxograma da figura 57 demonstra como é o processo para abrir uma baia. Logo após a função ser chamada são consultados os dados do solicitante e da baia, se algum desses não retornarem dado nenhum significa que o usuário ou a baia não estão cadastrados no sistema e é enviado para o *gateway* uma resposta negativa. Caso hajam dados, é feita mais uma verificação para checar se essa solicitação está sendo feita dentro do horário de aula e em dias úteis, caso não esteja, é feita uma busca de reserva para esse usuário, conforme a RN-03 da tabela 12, e se não retornar nada significa que não há reserva e por isso é enviada uma resposta negativa e é feito um registro no log. Por outro lado, caso esse usuário tenha feito uma reserva, é seguido o mesmo fluxo de alguém que está tentando abrir a baia em horário de aula.

Esse fluxo segue uma sequência com várias condições também. A primeira é o processo mais simples para se abrir uma baia, o usuário não pode ter nenhuma pendência e a baia tem que estar livre. Quando se fala em pendência quer dizer que ele não tem nenhuma baia bloqueada atrelada ao usuário dele. Caso essa primeira condição não seja verdadeira, é avaliada uma segunda condição, que verifica se a baia está bloqueada, se não estiver pode estar ocorrendo três situações: a baia possivelmente não está disponível ou o usuário pode estar usando ou ter uma pendência com alguma outra baia, de qualquer forma é enviada uma negativa como resposta ao *gateway*. Caso essa baia esteja bloqueada só quem pode abrir é um monitor ou o aluno responsável pelo bloqueio, se for um desses dois a baia é aberta, colocada no status de "em checagem"e enviada uma resposta de volta para o *gateway* informando que pode abrir a baia. Se não for nenhum dos dois, a resposta enviada é de que não pode abrir a baia.

O processo detalhado de uma solicitação referente ao fechamento de uma baia está representado no fluxograma da figura 58 e diferentemente do processo anterior, ele recebe apenas os dados capturados da respectiva baia, que são número da baia e equipamentos presentes.

Após o sistema de fato receber a solicitação de fechamento de baia, o processo se inicia identificando o usuário que utilizou a baia por último, após isso, é feita a checagem dos equi-

Figura 57 – Fluxograma de abrir baia



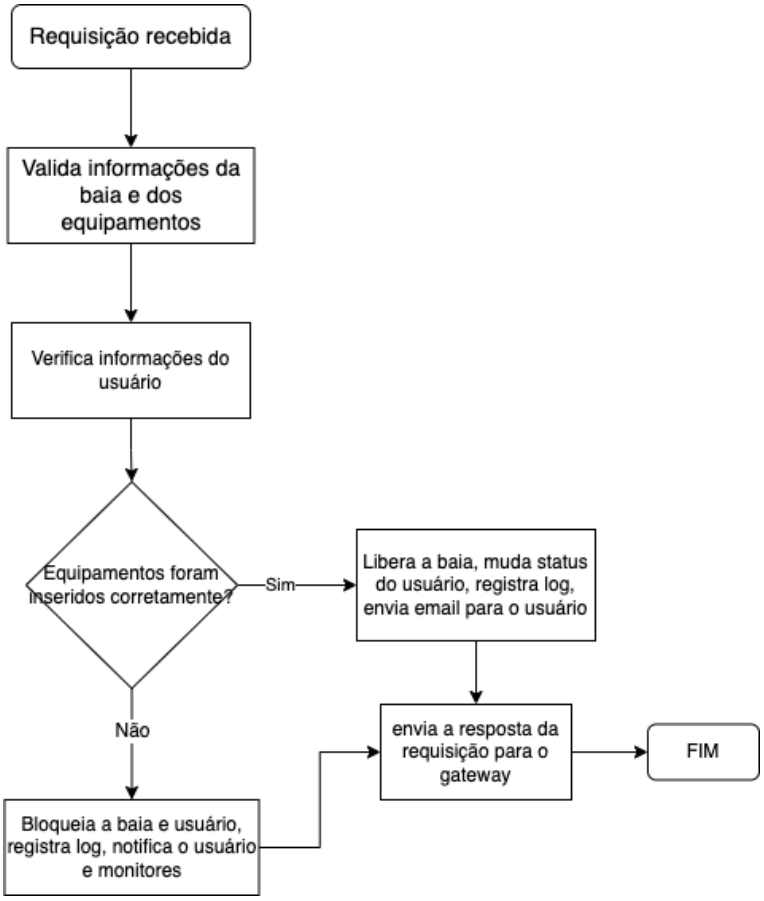
Fonte: Autor (2022)

pamentos presentes e se todos eles estiverem presentes é registrado que a baia está fechada e liberada para uso. Caso contrário, ela é bloqueada e é feita a notificação por e-mail para o aluno e os monitores informando a ausência de um ou mais equipamentos. O aluno só poderá abrir uma outra baia novamente quando essa situação for resolvida.

Por último, o fluxograma referente ao *backup* é mostrado na figura 59. Nele estão duas funções distintas que foram colocadas juntas porque possuem um código menos complexo que os anteriores e desempenham tarefas semelhantes diferenciadas apenas por algumas condições.

Ambas enviam dados das tabelas do banco de dados para o *gateway* mas cada uma com uma filtragem diferente. A primeira chamada retorna os dados de todas as tabelas de uma vez sem uma data de ponto de partida, ou seja, não existe uma filtragem por data. A segunda função possui um ponto de partida, ou seja, todos os dados a partir daquela data até o presente momento serão buscados, se não houver nenhum dado é retornada uma resposta vazia, caso contrário os dados são estruturados em um formato JSON e enviados de volta para o *gateway*.

Figura 58 – Fluxograma de fechar baia



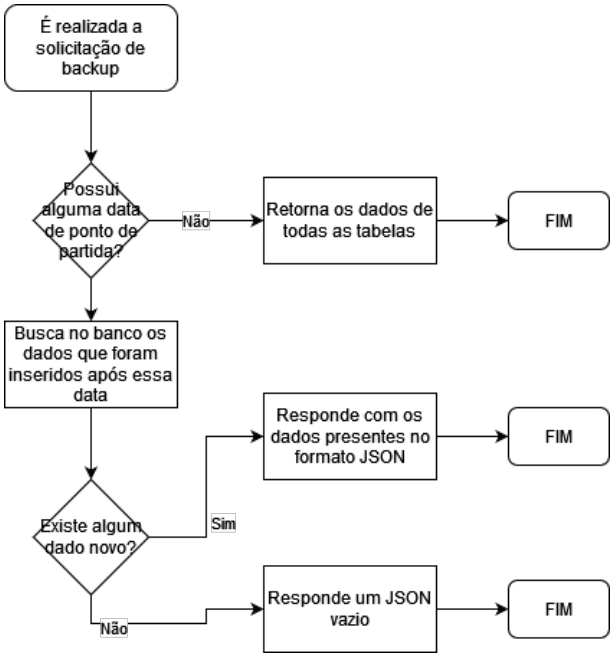
Fonte: Autor (2022)

6.3.3 Banco de dados na nuvem

O banco de dados na nuvem foi desenvolvido a partir do MySQL que é um sistema de gerenciamento baseado na linguagem SQL que possibilita criar relacionamentos entre tabelas. Esse relacionamento é importante porque algumas tabelas precisam de dados de outras para que não existam redundâncias. Por exemplo, toda reserva tem uma baia e essa baia precisa que todos os seus dados sejam acessados, mas não seria interessante ter todas as colunas dela na tabela de reserva, por isso existe uma referência particular de cada baia, chamada de *chave estrangeira*. Essa chave estrangeira corresponde a um identificador único daquela baia, ou seja, através dessa chave é possível ter acesso a todos os dados desejados da tabela referenciada. Vale ressaltar que essa referência é obrigatória porque no cenário descrito anteriormente uma reserva precisa sempre referenciar uma baia para que seja válida, mas uma chave estrangeira pode tanto ser obrigatória como não ser.

A figura 60 é um *schema* do banco de dados desenvolvido no presente trabalho. Um *schema* nada mais é do que uma representação gráfica das tabelas e os relacionamentos existentes. Nessa figura é possível notar a existência de doze tabelas, onde podemos dividi-las em dois grupos: as tabelas pais e as tabelas filhas.

Figura 59 – Fluxograma de pegar os dados para backup

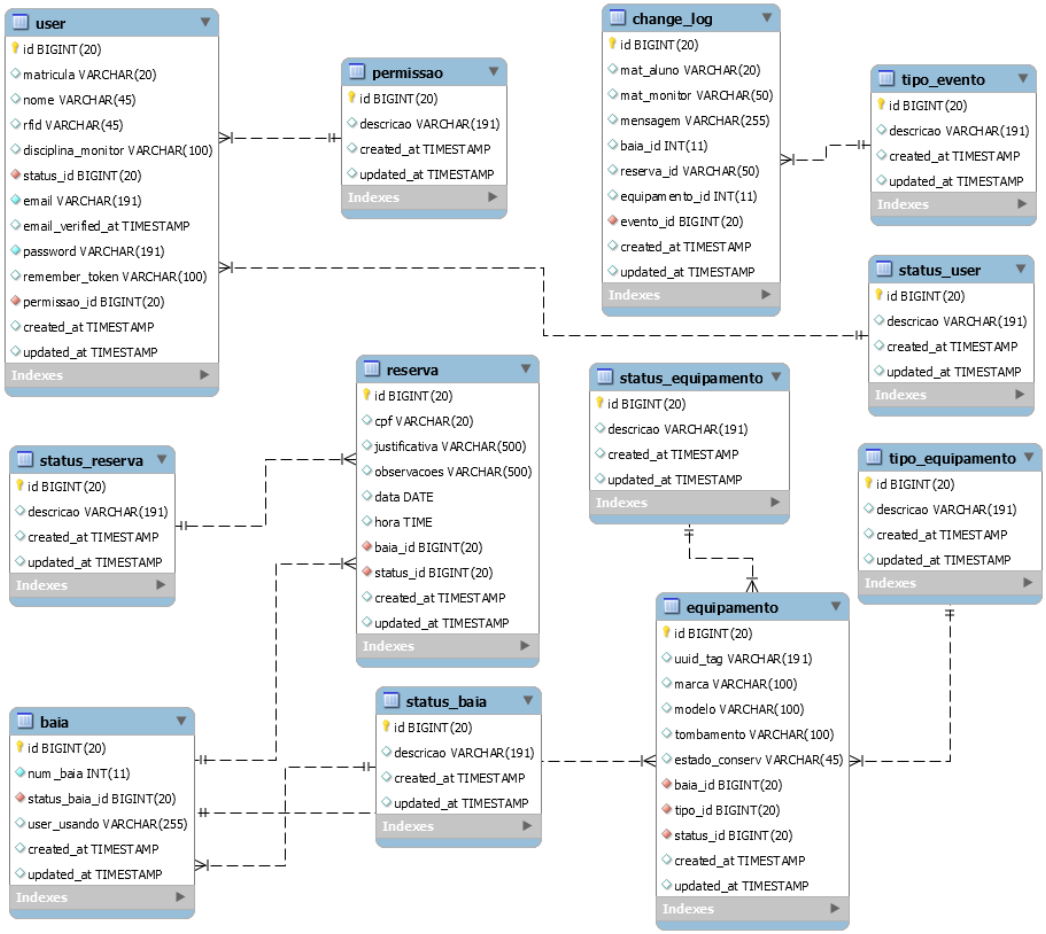


Fonte: Autor (2022)

O primeiro grupo são das tabelas onde os dados são variáveis, ou seja, é possível adicionar, editar e exclui-los quantas vezes for preciso. Em geral são as tabelas que possuem um maior fluxo de dados, que são: *user*, *baia*, *reserva*, *change_log* e *equipamento*.

O segundo grupo são as tabelas em que os dados são fixos, ou seja, uma vez cadastrados dificilmente são alterados ou são adicionados novos dados. Essas tabelas servem para garantir a integridade da informação nas tabelas pais. Por exemplo, toda reserva tem um status que pode ser pendente, liberado ou cancelado. Esses status estão todos pré-estabelecidos na tabela de *status_reserva* e ao invés da tabela reserva ter uma coluna com um campo de texto, terá uma coluna que receberá um valor inteiro que será a referência do seu respectivo status na tabela *status_reserva*. As tabelas desse segundo grupo são: *permissão*, *status_reserva*, *status_baia*, *status_equipamento*, *status_user*, *tipo_equipamento*, *tipo_evento*.

Figura 60 – *Schema* do banco de dados

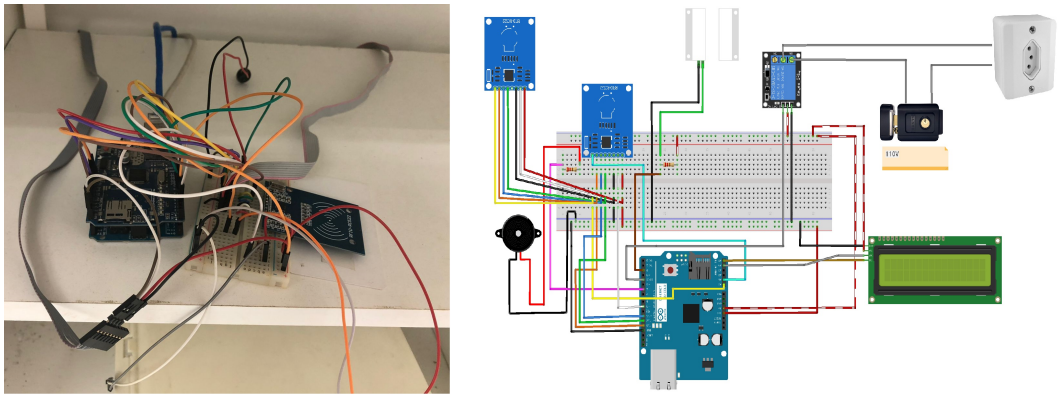


Fonte: Autor (2022)

7 TESTES E RESULTADOS

Neste capítulo são apresentados os testes e os resultados obtidos do sistema desenvolvido neste trabalho. Foram analisados tanto o funcionamento quanto o desempenho do sistema com testes e validações realizadas em um circuíto desenvolvido em uma *protoboard*. A figura 61 apresenta esse circuíto na imagem da esquerda e seu esquemático, detalhado no capítulo 6, na imagem da direita.

Figura 61 – Protótipo do hardware e seu esquemático



Fonte: Autor (2022)

A seção 7.1 irá mostrar como foi feita a conexão entre o Arduíno e o *gateway*, como foi colocado na baia e como é feita abertura e a identificação do fechamento da baia. A seção 7.2 irá explorar quais os testes que foram feitos e o tempo de execução de cada um para mostrar a viabilidade da implementação desse sistema.

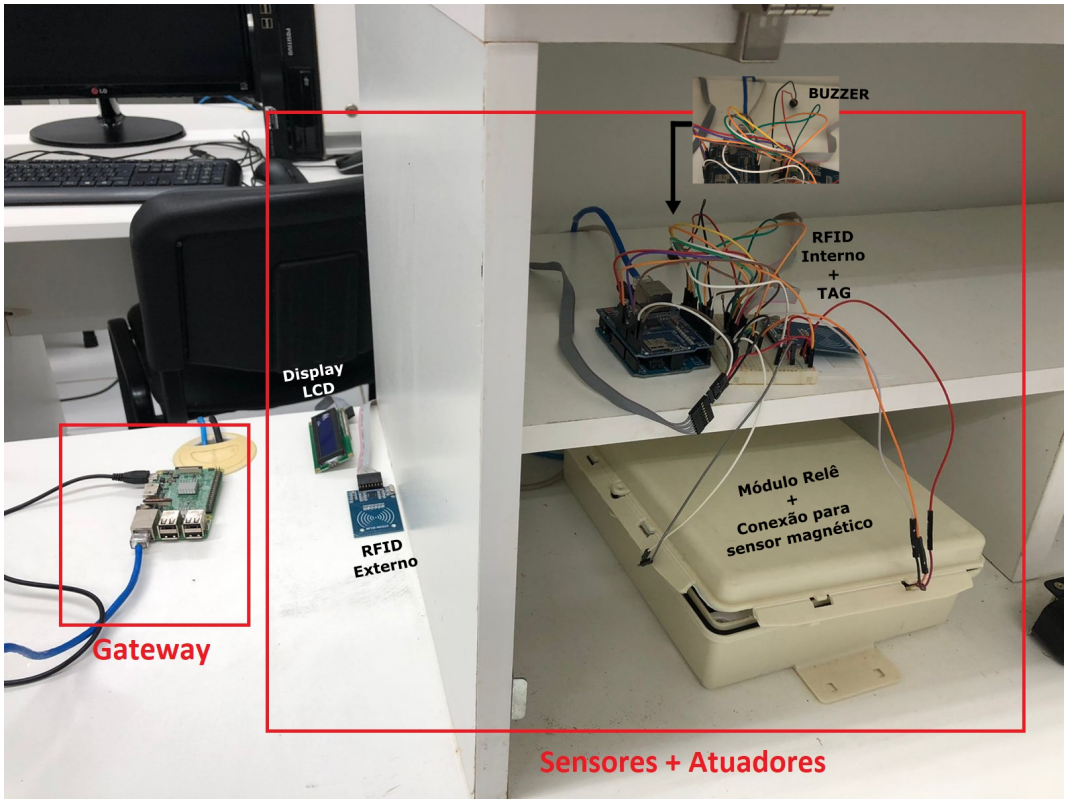
7.1 INFRAESTRUTURA

No capítulo 4 foi mostrada a arquitetura do sistema com as duas primeiras camadas sendo compostas do subsistema de monitoramento (incluindo leitores de RFID, microcontrolador, interface de rede) e o *gateway* de equipamentos que ocupam o espaço físico e estão conectados entre si via cabos de rede. Dito isso, a figura 62 apresenta a infraestrutura básica para realizar a comunicação entre uma baia, com seu microcontrolador e o *gateway*. Ambos estão conectados diretamente, porém em uma situação real terá um *switch* que liga as baias ao *gateway*.

No quadrado da esquerda está o *gateway* e no da direita está o microcontrolador, os sensores e os atuadores. No quadrado da direita é possível notar que do lado de fora da baia tem duas placas, uma delas é a do display LCD que é usado para dar uma resposta visual do que está acontecendo naquele momento e a outra é o leitor RFID que é responsável por capturar as informações do crachá dos usuários.

Na parte de dentro da baia está o microcontrolador com um leitor RFID conectado na *protoboard* de forma que seja possível realizar a leitura de um cartão simulando um equipamento.

Figura 62 – Infraestrutura de teste



Fonte: Autor (2022)

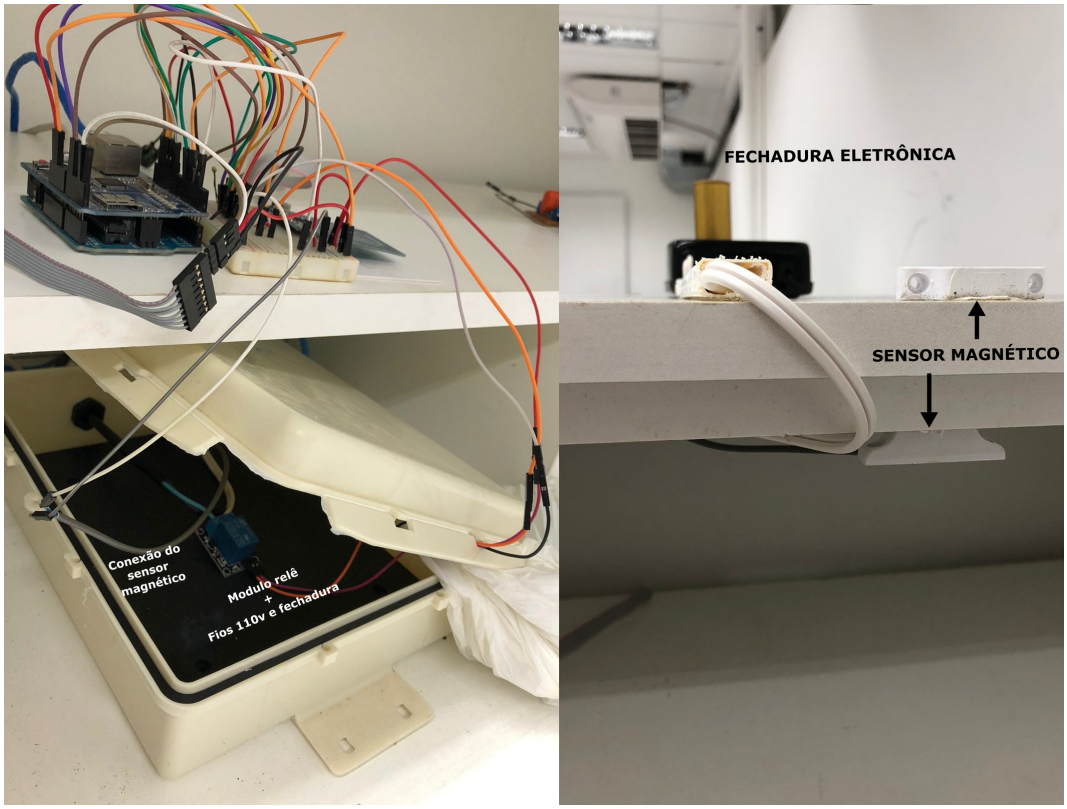
É importante salientar que essa *tag* não foi colada em um equipamento porque ainda não há um suporte que encaixe esse equipamento em uma posição no campo de contato do leitor RFID.

Ainda sobre o interior da baia, também estão o módulo buzzer, o relê e o sensor magnético, porém esses dois últimos só podem ser vistos com mais detalhes na figura 63.

A figura 63 possui duas imagens, a primeira mostra o que tem dentro da caixa, que na figura 62 estava fechada, e a segunda mostra a fechadura eletrônica e o sensor magnético instalados. Esse sensor magnético tem uma das suas partes presa no teto da baia e a outra presa no topo da porta, sendo identificado que a baia está fechada quando ambas partes entram em contato via campo magnético.

Dentro da caixa, que será melhor projetada para armazenar todo o sistema de monitoramento da baia, está o módulo relê e os fios que conectam a fechadura ao microcontrolador, dos fios conectados no relê um deles é o de alimentação plugado na tomada(o azul) e o outro é o da própria fechadura(o branco). O outro fio, o cinza, que está preso a dois *jumpers* é o que conecta o microcontrolador e o sensor magnético.

Figura 63 – Infraestrutura de teste



Fonte: Autor (2022)

7.2 TESTES DE VALIDAÇÃO DO COMPORTAMENTO DO SISTEMA

Os testes executados incluíram de seis possíveis ações do usuário relacionadas à abertura e ao fechamento das baias. A intenção desses testes é mostrar o funcionamento do sistema em situações reais de forma a medir o tempo que leva para cada ação ser concluída. Esses testes foram feitos no laboratório do Centro de Informática que além da estrutura, conta com uma internet de alta velocidade o que contribuiu para uma comunicação mais rápida e estável entre o *gateway* e a *API Rest*.

Além desses testes, foi feito um teste específico que realizou o envio sucessivo de vários tipos de pacotes para simular diferentes situações de uso e, assim, testar, a capacidade de resposta do sistema. Esse teste foi executado fora do Centro de Informática (residência do autor), onde a internet tem uma velocidade menor e uma maior oscilação que a do Centro de Informática da UFPE, o que pode influenciar diretamente no tempo total de resposta para o usuário.

As subseções seguintes irão descrever com mais detalhes cada um dos testes executados, o teste citado no parágrafo anterior e os que foram feitos em uma das baias do laboratório, que são: abertura da baia em horário de aula ou com reserva no sistema, a tentativa de abertura com uma *tag* não cadastrada, outra tentativa de abertura só que dessa vez fora do horário de aula e sem reserva, o fechamento da baia com todos os equipamentos presentes, o fechamento com algum equipamento ausente ou inserido incorretamente e o fechamento da baia com o *gateway*

sem acesso à internet.

7.2.1 Abertura de baia

Para um usuário cadastrado conseguir abrir uma baia livre existem duas situações, conforme é possível observar na tabela 12 referentes às regras de negócio: estar em horário de aula em um dia de semana e, caso esteja fora de horário de aula ou em fim de semana, ele precisa ter uma reserva. Uma vez que essas exigências estão sendo cumpridas, é necessário apenas o usuário aproximar seu crachá no leitor externo e aguardar o sistema processar a solicitação. A figura 64 mostra a mensagem do display LCD após a solicitação ser atendida e além dela o sistema também emite um aviso sonoro através do módulo *buzzer*.

Figura 64 – Mensagem de acesso concedido no Display LCD

Fonte: Autor (2022)

No sistema também é computado um *log* referente a essa solicitação informando a matrícula do usuário, a descrição desse log, a data e a hora que ocorreu conforme é mostrado na figura 65.

Figura 65 – Registro de abertura no portal

10314910409	Abriu a baia	Uso	2022-02-23 00:39:25	2022-02-23 00:39:25	›

Fonte: Autor (2022)

O *log* da figura 65 é gerado após a realização de algumas etapas, que serão detalhadas na seção 7.2.7. Entretanto, para uma maior transparência na visualização do *log*, a linha destacada na figura 65 mostra processo que antecedeu o envio do dado para o *backend*. Nessa linha, é possível verificar a data e a hora do início do processo e, se comparar esses mesmos dados da figura 65 concluí-se que ambos, de fato, são processos correspondentes.

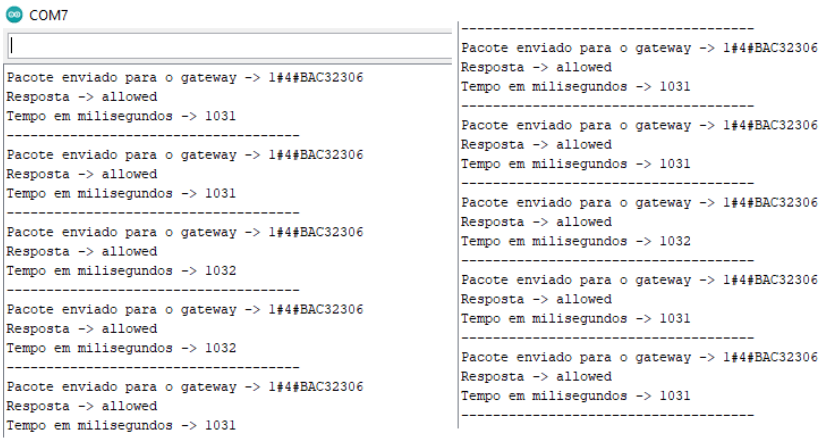
Figura 66 – Registro de abertura no portal no *prompt* de comando do *gateway*

```
Recebeu a mensagem 1#4#BAC32306 em 23/2/2022 às 0:39:24:189

Enviou pra API
|----- O tempo de processamento dentro da API foi de 56.81800842285156 milissegu
ndos
|----- Somente o tempo do transporte (Gateway -> API / API -> Gateway) durou 129
2.1819915771484 milissegundos
|----- O tempo total entre o envio, o processamento e a resposta da API foi de 1
349 milissegundos

Enviado de volta ao arduíno em 23/2/2022 às 0:39:25:543
------------ PROCESSO FINALIZADO -------------
```

Fonte: Autor (2022)

O teste de abertura de baia foi feito dez sucessivas vezes e todas apresentaram o mesmo comportamento tendo uma pequena variação no tempo de execução. A figura 67 apresenta a janela do monitor serial que mostra mensagens programadas para diferentes acontecimentos durante a execução do programa. Cada execução imprime três linhas de textos onde uma serve apenas como separador, que indica o fim da tarefa.

Figura 67 – Monitor serial da IDE do Arduíno com dados da abertura de baia



Fonte: Autor (2022)

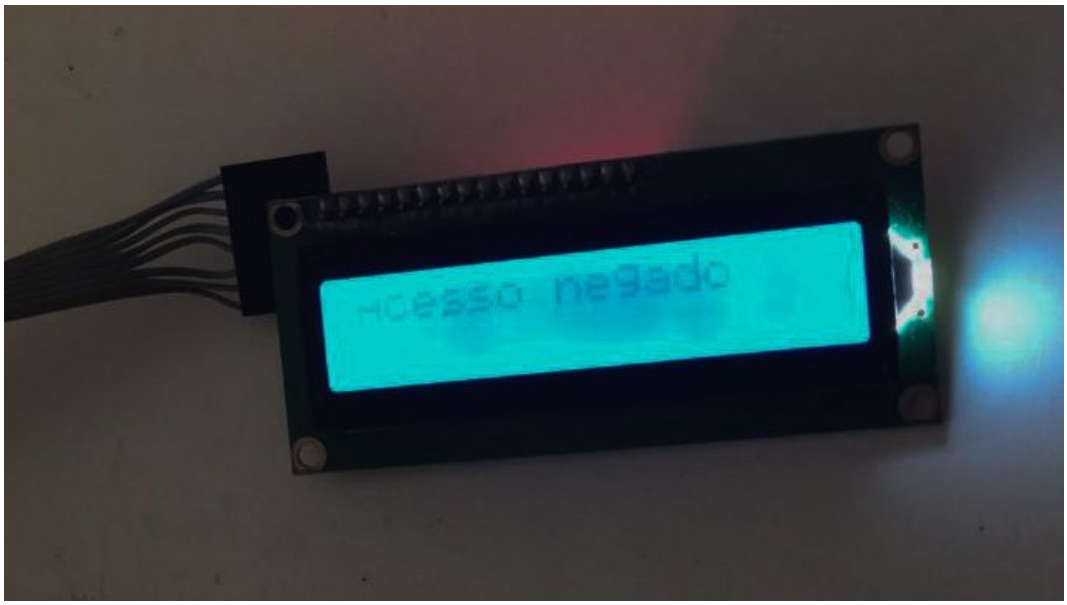
A primeira linha corresponde à mensagem que está sendo enviada para o *gateway*, a segunda já mostra a resposta da solicitação e a terceira mostra o tempo que essa solicitação levou para ser concluída contando a partir do momento que o usuário aproxima seu crachá até que uma mensagem seja recebida pelo *gateway*.

Em relação ao tempo de execução é possível notar que ele se manteve constante durante todos os teste, com um tempo um pouco maior que 1s e com uma variação mínima. Esse tempo pode ser considerado bom pela quantidade de camadas e de processamento que os dados precisam passar.

7.2.2 Tentativa de abertura de baia fora do horário de aula e sem reserva

Para este teste foi usado o crachá do autor com a permissão de aluno após às 19 horas, horário esse pré-configurado como o fim do expediente de aula, e como eu não tinha nenhuma reserva naquele momento naturalmente a solicitação foi negada, conforme apresenta a figura 68.

Figura 68 – Mensagem de acesso negado no Display LCD



Fonte: Autor (2022)

Além disso, semelhante à última solicitação, o sistema também registra um *log* referente à tentativa de abertura informando os mesmo dados: a matrícula do usuário, a descrição desse log, a data e a hora que ocorreu conforme é mostrado na figura 69.

Figura 69 – Registro de tentativa de abertura no portal

10111210310	Tentativa de usar a baia fora do horário de aula sem reserva.	Uso	2022-02-14 19:05:16	2022-02-14 19:05:16

Fonte: Autor (2022)

Esse teste também foi executado dez sucessivas vezes e novamente todas apresentaram o mesmo comportamento mas com algumas diferentes na variação do tempo de execução. A

figura 70 apresenta a janela do monitor serial e segue a mesma lógica da solicitação anterior com cada execução imprimindo três linhas de textos e um separador indicando o fim da tarefa.

Figura 70 – Monitor serial da IDE do Arduíno com dados da abertura de baia sem reserva

Fonte: Autor (2022)

A primeira linha corresponde à mensagem que está sendo enviada para o *gateway*, a segunda já mostra a resposta da solicitação, que dessa vez é diferente da anterior, e a terceira mostra o tempo que a solicitação levou para ser concluída.

Em relação ao tempo de execução é possível notar que o tempo médio dos testes foi aproximadamente 1.1s, com um aumento de 100ms, em relação ao teste anterior, o que não é suficiente para afetar a experiência do usuário.

7.2.3 Tentativa de abertura de baia com usuário sem cadastro

No caso da tentativa de abertura sem cadastro foi usado um crachá aleatório, não cadastrado no sistema, e conforme é apresentado na figura 71 o display LCD apenas informa que o usuário não está cadastrado e emite um efeito sonoro para alertar o usuário.

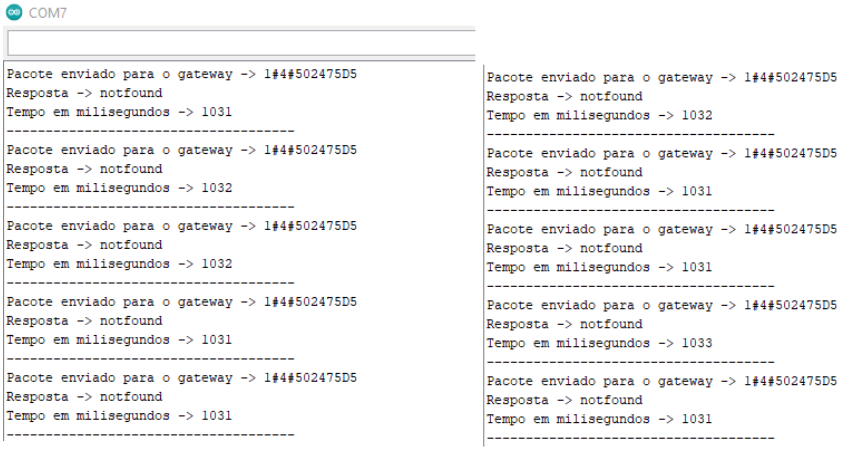
Novamente foram executados 10 testes e conforme mostra a figura 72 o formato da saída no monitor serial da IDE do Arduíno é o mesmo dos anteriores só mudando a resposta do *gateway* que dessa vez foi *notfound*.

O tempo de execução total se assemelha bastante com o primeiro teste, com uma variação muito pequena e praticamente com a mesma média de aproximadamente a 1s. Dito isso, pode-se dizer então que esse tempo também é consideravelmente bom pois o *delay* entre instante da aproximação do cartão e a resposta do *gateway* é muito curto e não afeta a experiência do usuário algo negativo.

Figura 71 – Mensagem de usuário não cadastrado no Display LCD

Fonte: Autor (2022)

Figura 72 – Monitor serial da IDE do Arduíno com dados da tentativa de abertura da baia de um cartão não cadastrado



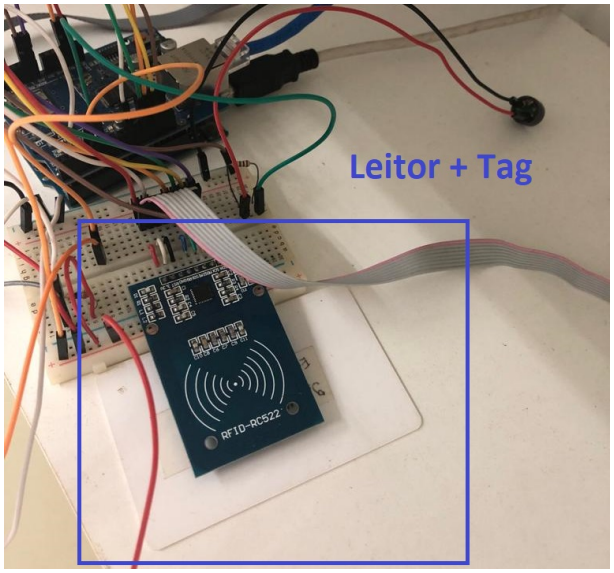
Fonte: Autor (2022)

7.2.4 Fechar baia com equipamentos presentes

Este teste teve o objetivo de verificar se todos os equipamentos estavam presentes ao fechar a baia, ele é baseado na RN-04 da tabela 13. Para isso, dentro dessa baia tinha um leitor RFID e abaixo dele, simulando um equipamento, uma *tag* conforme se pode observar no quadrado destacado em azul da figura 73. Essa *tag* estaria colada ao equipamento e de alguma forma em contato com esse leitor RFID.

A figura 74 mostra duas imagens, a da esquerda sinaliza que o sistema estava trabalhando na checagem dos equipamentos e a da direita que os equipamentos foram inseridos corretamente e

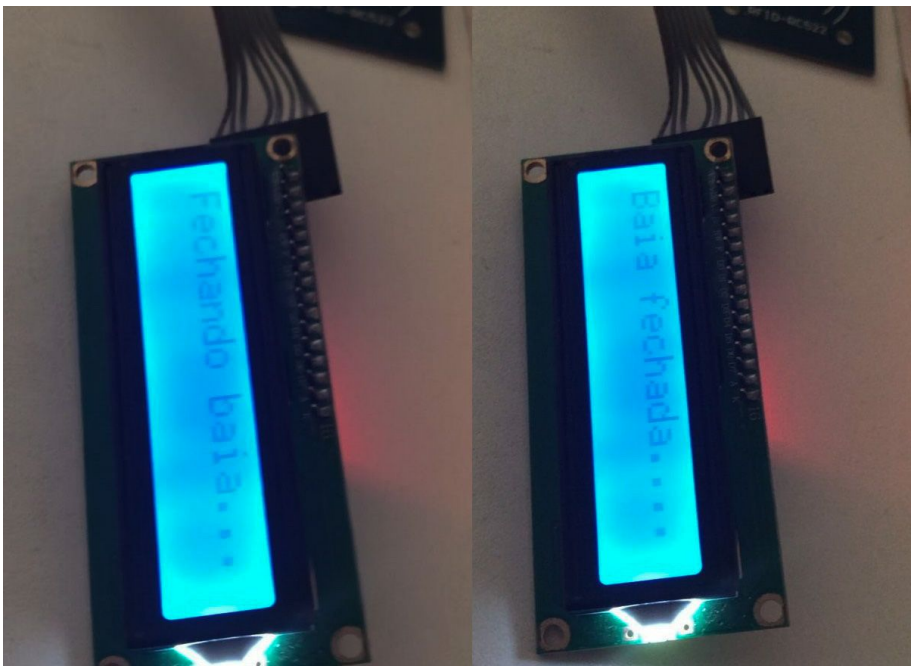
Figura 73 – Leitor e *tag* RFID dentro da baia



Fonte: Autor (2022)

a baia estará livre para uso.

Figura 74 – Display LCD com mensagens de fechamento de baia

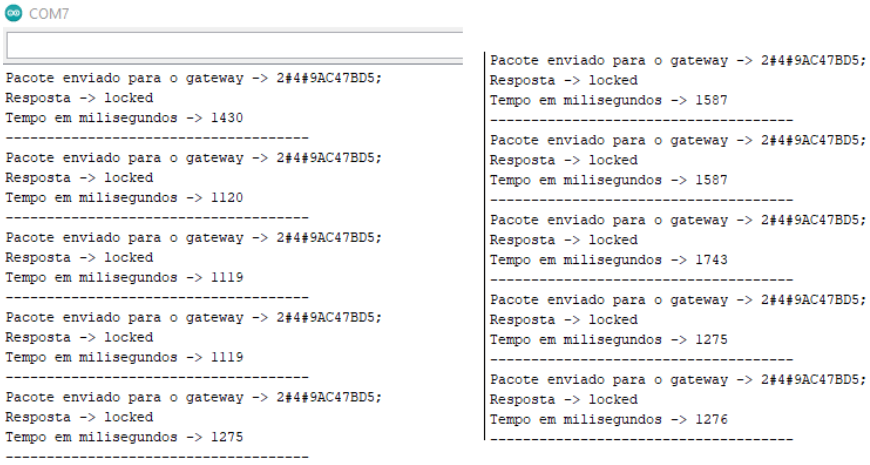


Fonte: Autor (2022)

O procedimento de testes se repetiu em relação aos testes anteriores, ou seja, foram feitos dez testes sucessivos e o andamento da solicitação pôde ser acompanhado pelo monitor serial da IDE do Arduíno e os dados foram impressos da mesma maneira dos testes anteriores, onde a primeira linha é a mensagem enviada para o *gateway*, a segunda é a resposta do *gateway* para o Arduíno e a terceira linha mostra o tempo em milissegundo que a operação demorou para ser

concluída. Essas informações podem ser visualizadas na figura 75.

Figura 75 – Monitor serial do Arduíno no processo de fechar a baia



Fonte: Autor (2022)

Em todos os testes, o Arduíno recebeu como resposta do *gateway* a mensagem *locked* que significa que todos os equipamentos estão presentes e a baia pode ser liberada para uso. Sabendo disso, conclui-se que todos os testes apresentaram o comportamento que era esperado, porém é possível observar que a média do tempo de execução para se fechar uma baia é superior às médias para abri-la. Isso acontece porque o *backend* precisa fazer um número maior de verificações até mandar a resposta de volta para o usuário. Ainda assim a média de todos os testes não ultrapassa os 1,5s o que mantém um tempo de espera curto e não torna a experiência do usuário algo desagradável.

Além disso, mais duas ações são realizadas durante o fechamento correto da baia, que são: a criação de um *log* no sistema, que informa quem foi o último a usar e a hora exata do ocorrido, e o envio de um e-mail para o aluno confirmando que a baia foi fechada corretamente. A primeira ação pode ser vista na figura 76 e a segunda na figura 77.

Figura 76 – Registro de baia fechada no portal

Fonte: Autor (2022)

7.2.5 Fechar baia com equipamentos faltando

Neste teste o usuário supostamente colocou o equipamento de uma forma errada, ou seja, sem o contato com o leitor RFID ou simplesmente pode não ter colocado esse equipamento

Figura 77 – E-mail recebido após o fechamento correto da baia

Fonte: Autor (2022)

de volta. Para simular essa situação foi retirado a *tag* que fica em baixo do leitor, mostrada na figura 73, e a baia foi fechada. O resultado é apresentado na figura 78 e além dele o módulo *buzzer* também emite um som para alertar o usuário que tem algo errado.

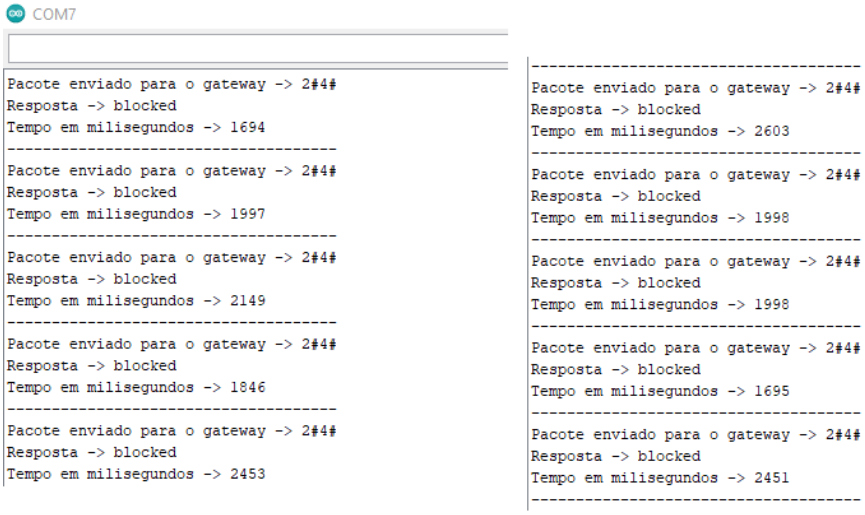
Figura 78 – Display LCD com a mensagem de baia bloqueada

Fonte: Autor (2022)

O procedimento de testes se repetiu em relação aos testes anteriores, ou seja, foram feitos dez testes sucessivos e o andamento da solicitação pôde ser acompanhado pelo monitor serial da IDE do Arduíno e os dados foram impressos da mesma maneira dos testes anteriores, onde a primeira linha é a mensagem enviada para o *gateway*, a segunda é a resposta do *gateway* para o Arduíno e a terceira linha mostra o tempo em milissegundo que a operação demorou para ser concluída. Essas informações podem ser visualizadas na figura 75.

A quantidade de repetições e mensagens impressas no monitor serial do Arduíno foram as mesmas do teste anterior, conforme é mostrado na figura 79. É possível ver também que assim como todos os outros testes, esse experimento funcionou conforme era esperado, sem nenhuma falha, porém o tempo médio subiu para um pouco mais de 2s, um aumento de aproximadamente 38% em relação ao teste anterior. Entretanto, mesmo com esse aumento, essa média ainda está dentro do esperado visto que 2s ainda é um tempo de espera baixo levando em consideração a quantidade de consultas que é feita durante todo o processo.

Figura 79 – Monitor serial do Arduíno no processo de fechar a baia com equipamento ausente



Fonte: Autor (2022)

Uma outra diferença para todos os outros testes está na mensagem enviada para o *gateway*. É possível notar que, nas primeiras linhas de todos os outros existiam três informações entre os caracteres de *hashtag*, porém nesse só existem duas. Isso porque só apenas um equipamento foi cadastrado nesta baia e o mesmo encontra-se ausente, nesse caso, a informação referente aos equipamentos é um campo vazio indicando que não possui nenhum equipamento naquela baia.

Depois que é constatado que algum equipamento está ausente, tanto os monitores quanto o último usuário que abriu a baia recebem um e-mail informando que algum equipamento não foi inserido corretamente, conforme mostra a figura 80.

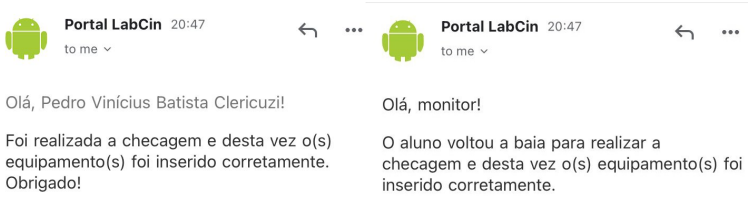
Figura 80 – E-mail recebido após o fechamento incorreto da baia

Fonte: Autor (2022)

Da mesma forma, depois que o aluno ou o monitor foi até a baia para verificar o ocorrido e devolveu o equipamento ou posicionou-o corretamente, foi enviado um e-mail novamente para ambas as partes notificando que os equipamentos foram devolvidos, como mostra a figura 81.

Além disso, também foi registrado um *log* no portal para cada um desses eventos. Esses

Figura 81 – E-mail recebido após a devolução dos equipamentos para baia



Fonte: Autor (2022)

logs podem ser vistos na figura 82, onde o da linha de cima corresponde ao fechamento com equipamentos ausentes e o *log* da linha de baixo corresponde ao registro de devolução desses equipamentos.

Figura 82 – Registro de baia fechada no portal

10099104090	Um ou mais equipamentos não foram inseridos corretamente, esta baia agora está bloqueada.	Uso	2022-02-15 20:16:42	2022-02-15 20:16:42
10099104090	Foi realizada a checagem e desta vez o(s) equipamento(s) foi inserido corretamente.	Uso	2022-02-15 20:47:24	2022-02-15 20:47:24

Fonte: Autor (2022)

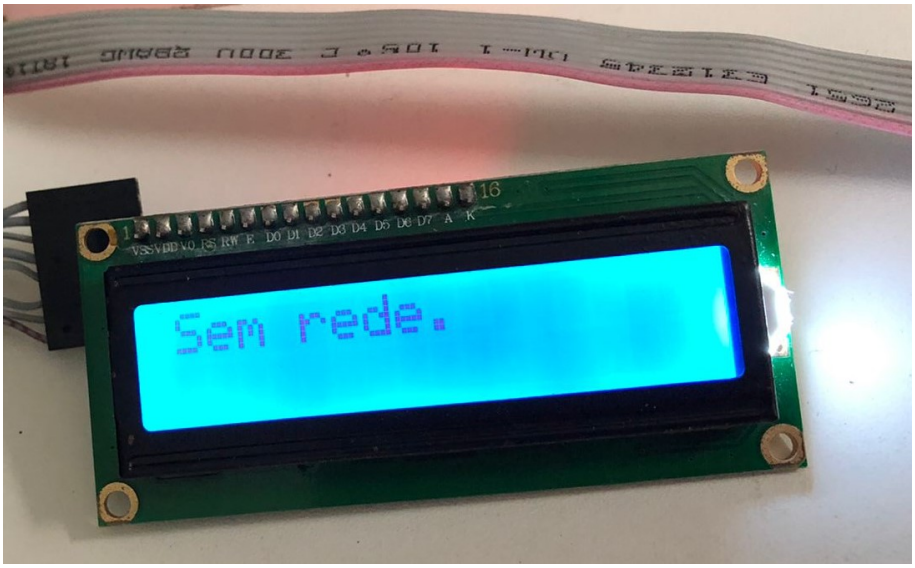
7.2.6 Fechar baia com o *gateway* sem internet

O último teste simulando uma situação real levou em consideração a possibilidade do *gateway* estar sem internet enquanto alguma baia estava em uso. Em uma situação como essa, como é definido na RN-05 da tabela 13, de alguma forma o dado tem que ser salvo para que o registro não seja perdido e a baia possa ser aberta novamente, afinal, não é possível abrir uma baia que já está aberta e o usuário fica impossibilitado de abrir outra visto que no sistema consta que ele já está usando uma.

Para a realização dos testes nesse cenário, após a baia ser aberta a conexão com a internet foi interrompida de forma intencional e, tendo passado alguns segundos, a baia foi fechada. Os dados foram enviados normalmente para o *gateway* que salvou o registro localmente e devolveu para o Arduíno a resposta de que não tinha conexão. Essa resposta foi mostrada para o usuário no display LCD conforme mostra a figura 83 e a baia só pode ser aberta novamente através do sistema depois que a conexão foi reestabelecida.

Os testes na baia tiveram um número menor de repetições, dessa vez foram 5 testes sucessivos e as mensagens no monitor serial permaneceram no mesmo formato. A figura 84 mostra as informações de cada teste, onde a primeira linha mostra a mensagem enviada para o *gateway* que sinaliza que a baia está sendo fechada com apenas um equipamento e que este possui como identificador os caracteres que sucedem o segundo *hashtag*. A segunda linha é a mensagem recebida de volta, sendo ela *no_network*, que indica que o *gateway* não possui conexão

Figura 83 – Display LCD mostrando a mensagem *sem rede*



Fonte: Autor (2022)

com a internet. Por último, o tempo de execução que como pode ser observado, do ponto de vista do usuário, a resposta aconteceu de maneira praticamente simultânea após o usuário ter aproximado seu cartão.

O tempo se mostrou bastante positivo, visto que praticamente não houve espera por parte do usuário e a informação foi registrada no banco de dados do *gateway* e todos os testes se comportaram da mesma maneira, como era esperado.

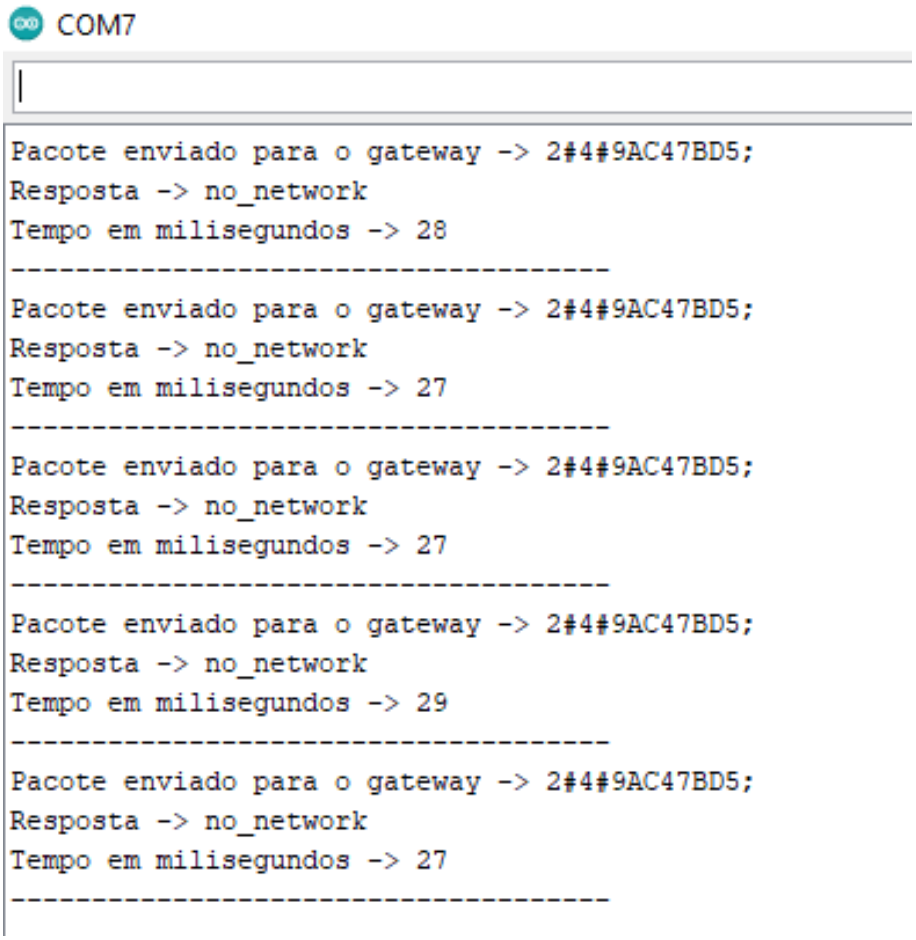
Do lado do *gateway* a figura 85 mostra como que o sistema se comporta nesta situação. De 15 em 15 segundos foram feitas verificações para checar se a conexão foi reestabelecida e após ter sido, o sistema checou a tabela *registro_offline*, recuperou todos os registros e fez o envio para a nuvem até que não houvesse mais nenhum. Feito isso, o *gateway* enviou uma mensagem para o Arduíno informando que a conexão foi reestabelecida e o mesmo voltou a ficar disponível para receber solicitações por parte do usuário.

7.2.7 Envios de pacotes

O teste de envio de mensagens, ou pacotes, teve o objetivo de medir o desempenho do *gateway* e do *backend* diante de várias solicitações sucessivas vindas do microcontrolador. Por isso, não foi usado nenhum dos sensores do protótipo real do projeto, apenas o arduíno e o *shield ethernet* conectados diretamente ao *gateway* via cabo de rede conforme apresenta a figura 86.

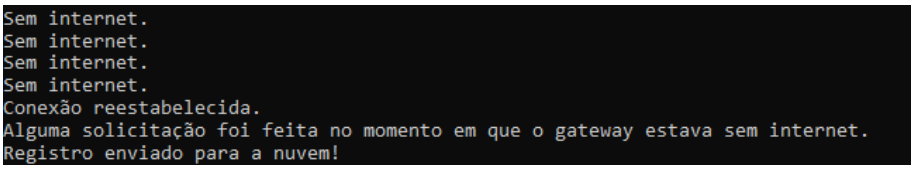
Os testes aconteceram de forma automática, ou seja, não foi necessário nenhuma intervenção manual para efetuar solicitações, a figura 87 mostra o trecho de código utilizado para automatizar o teste. A primeira linha é uma condição que se repete a cada 7 segundos e o que dita qual a solicitação que será feita naquele momento é uma variável *booleana* que alterna o tipo de teste todas as vezes que um pacote é enviado para o *gateway*. Como pode ser observado,

Figura 84 – Monitor serial do Arduíno no processo de fechar a baia com o *gateway* sem internet



Fonte: Autor (2022)

Figura 85 – Monitor serial do Arduíno no processo de fechar a baia com o *gateway* sem internet



Fonte: Autor (2022)

as mensagens já são predefinidas e efetuam as ações de abrir e fechar baia, pois o objetivo deste teste foi avaliar o comportamento do *gateway* com solicitações sucessivas e também avaliar o tempo de resposta em situações como essa.

A figura 88 apresenta o monitor serial da IDE do Arduíno mostrando a troca de mensagens com o *gateway* seguindo o mesmo padrão dos testes anteriores, na figura mostra apenas 12 solicitações, mas foram feitas 53 no total e suas respectivas imagens estão nos apêndices B e C. Assim como todos os outros testes, este não apresentou falhas durante a execução, entretanto teve tempos de conclusão bem distintos.

Essas diferenças no tempo de execução podem ser observadas com mais detalhes nas figuras 89 e 90. As duas imagens correspondem ao caminho que os dados fazem até retornar ao Arduíno

Figura 86 – Arduíno e *shield ethernet* ligadas ao *gateway* via cabo de rede

Fonte: Autor (2022)

Figura 87 – Trecho de código usado para enviar mensagens programadas para o *gateway*

```
if (millis() - lastMillis >= 7*1000UL)
{
  lastMillis = millis();
  if (aberto == false) {
    inicioRequisicao = millis();
    Serial.println("Solicitação -> abrir de baia");
    server.write("1#4#BAC32306");
    aberto = true;
  } else {
    inicioRequisicao = millis();
    Serial.println("Solicitação -> fechar de baia");
    server.write("2#4#9AC47BD5;");
    aberto = false;
  }
}
```

Fonte: Autor (2022)

e nelas mostram o rastreio de todos os 12 testes mostrados na figura 88, mantendo a mesma ordem. Para melhor visualização das informações, a tabela 16 apresenta os mesmos dados de uma forma mais simplificadas.

Tabela 16 – Tabela com a listagem dos teste com seus respectivos tempos, em milissegundos

	Processamento no backend	**Transporte (Gateway - Backend)**	**Total**
Teste 1	1555	940.7	2505

Teste 2	56.8	1292	1361
Teste 3	1216	637	1859
Teste 4	57	5592	5656
Teste 5	709	610.6	1325
Teste 6	110.9	561	678
Teste 7	671	547	1226
Teste 8	60	634	701
Teste 9	649	484.7	1140
Teste 10	50.5	517	574
Teste 11	667	589	1263
Teste 12	612	455	1074

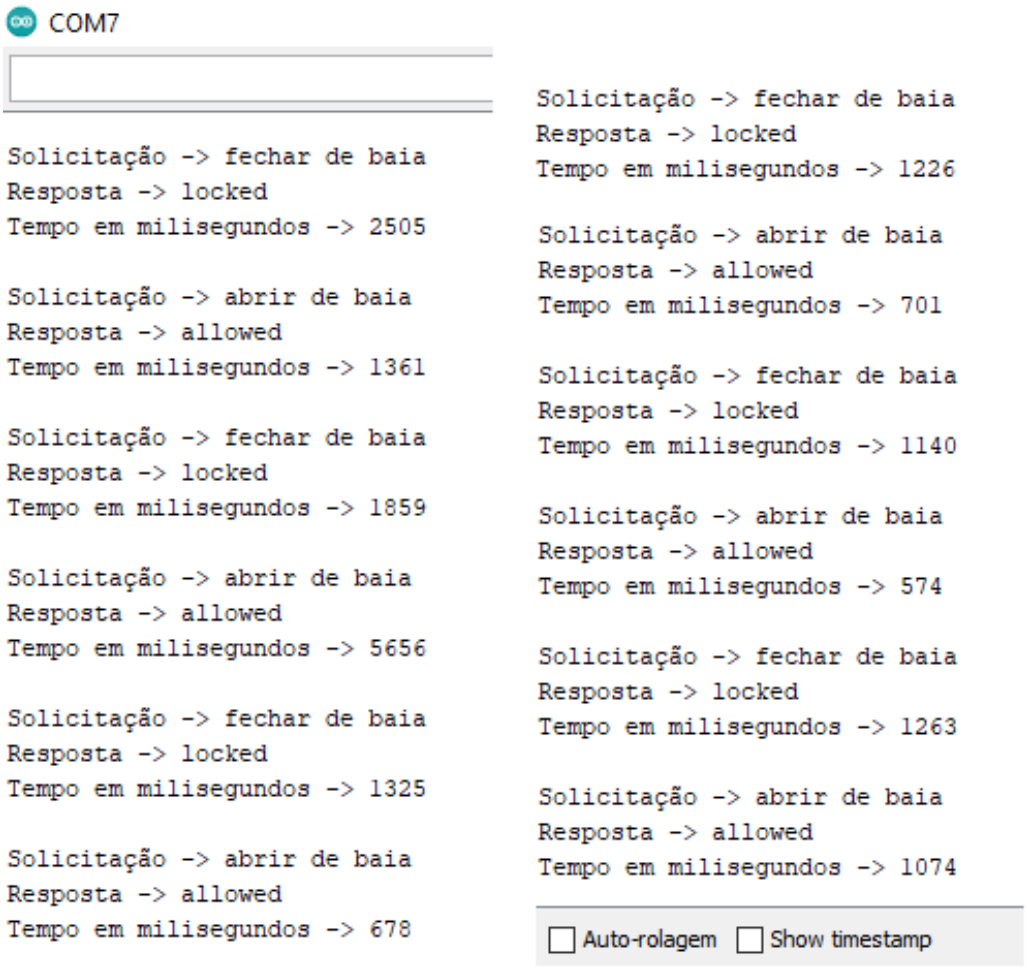
Fonte: Autor (2022)

Cada teste é separado por um conjunto de linhas tracejadas junto ao texto *PROCESSO FINALIZADO* e nesses testes é possível acompanhar o momento exato em que o dado entrou e saiu do *gateway* e o tempo gasto nas requisições para o *backend*.

A primeira e a última linha de cada teste correspondem ao dia e a hora de entrada e de saída das informações para o Arduíno, respectivamente. Os textos que ficam entre essas duas linhas são referentes ao tempo gasto na requisição para o *backend*, sendo a primeira linha após a mensagem *"Enviou para API"*, o termo API foi usado para referenciar o backend, o cálculo do tempo em que o sistema na nuvem levou para receber, processar os dados e transformá-los em informação. A linha seguinte mostra o tempo gasto no transporte dos dados durante a requisição HTTP. A terceira linha é uma junção das duas anteriores com a intenção de mostrar o tempo total usado por essa ação.

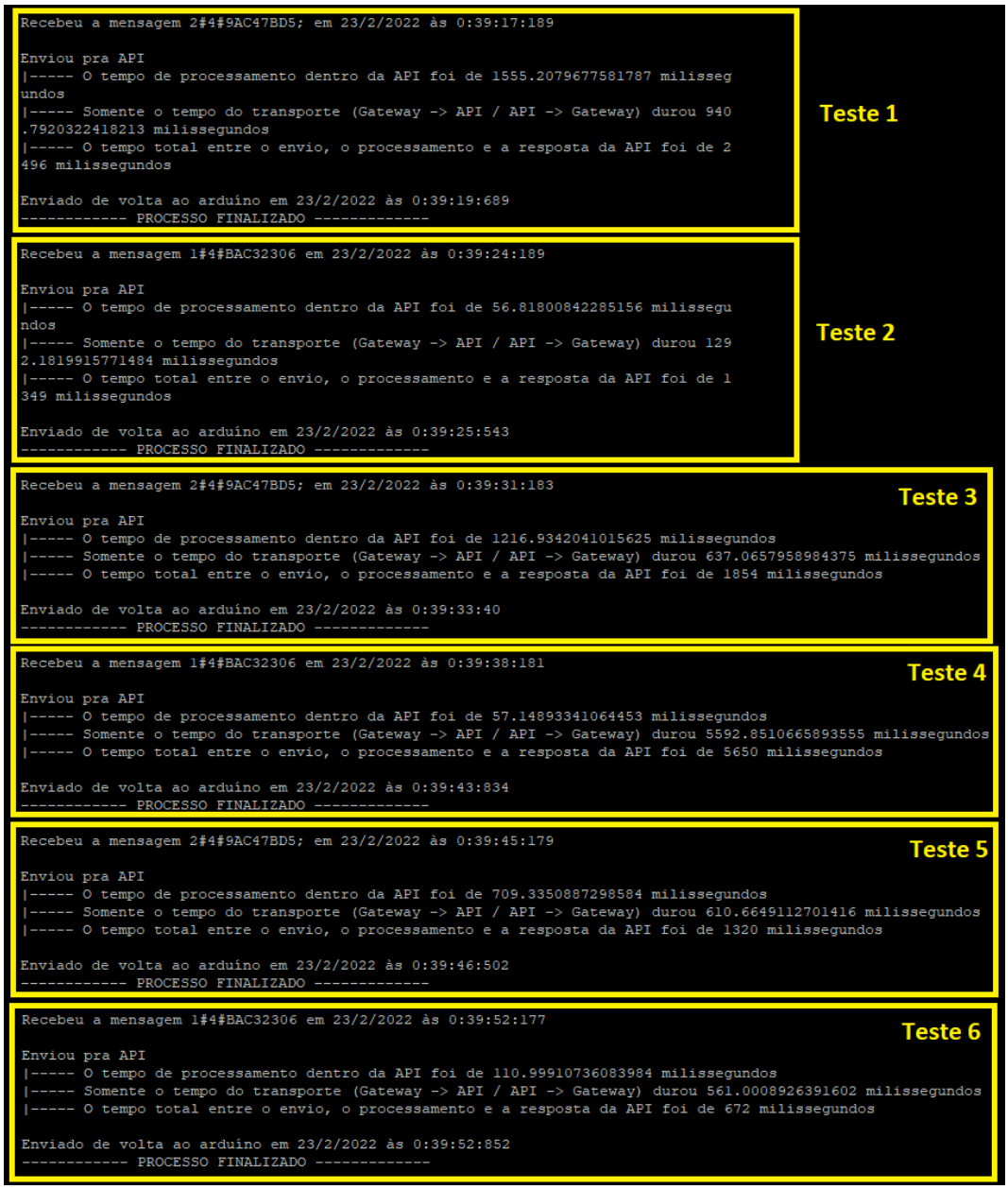
Um exemplo prático e que destoa de todas as outras solicitações é quarta delas. O tempo total ultrapassou os 5s, como é mostrado da figura 88, mas ao observar a figura 89 fica claro que o motivo de tamanha discrepância possivelmente é alguma oscilação na conexão à internet do local em que está o *gateway*.

Figura 88 – Monitor serial do Arduíno imprimindo os dados das solicitações



Fonte: Autor (2022)

Figura 89 – *Prompt* de comando do *gateway* - Parte I



Fonte: Autor (2022)

Figura 90 – *Prompt* de comando do *gateway* - Parte II

Fonte: Autor (2022)

8 CONCLUSÃO

Este capítulo apresenta os objetivos alcançados do projeto desenvolvido, uma breve análise dos resultados alcançados, as dificuldades apresentadas durante o desenvolvimento e, finalmente, algumas sugestões de melhorias para trabalhos futuros.

8.1 ANÁLISE DO PROJETO

Desde o princípio a ideia era desenvolver toda a infraestrutura para que só fosse terceirizado aquilo que realmente precisasse, como é o caso da hospedagem. Essa estratégia adotada logo no início evitou possíveis custos financeiros e que alguma parte do projeto ficasse dependente de algum serviço externo, o que em algum momento poderia inviabilizar todo o projeto.

O principal objetivo que é controlar o acesso e o gerenciamento de uso dos equipamentos foi alcançado, onde toda a comunicação é realizada conforme era esperado e os testes no laboratório mostram que a média do tempo total de execução das solicitações variam de 1 a 2 segundos, um tempo de espera satisfatório tendo em vista a quantidade de vezes que o dado é processado e enviado de um canto a outro, além a quantidade de verificações que são feitas até seja enviada a resposta para o Arduíno. Em relação ao segundo tipo de teste, o do envio de pacotes, embora tenha tido uma maior oscilação a média dos 53 testes se manteve dentro daquele intervalo de 1s e 2s com um tempo de 1,2s.

Os objetivos restantes também foram cumpridos:

- O sistema está emitindo alertas para os monitores e alunos nos casos de extravios de equipamentos;
- O sistema está gerando *logs* em todas as solicitações dos usuários cadastrados;
- Mesmo *offline* o *gateway* consegue registrar que a baia foi fechada com segurança;
- O *gateway* possui um banco de dados com as mesmas tabelas existentes na nuvem e diariamente realiza *backups* dos novos dados inseridos nela;
- O portal desenvolvido para os usuários possui uma interface gráfica agradável, é intuitivo e distingue as informações de acordo com os níveis de acesso do usuário.

Em relação às vantagens da abordagem escolhida nesse trabalho, usando como base a tabela 3, é possível visualizar na tabela 17 a relação entre as necessidades e as funcionalidades de fato implementadas nos trabalhos relacionados e neste.

Tabela 17 – Tabela de de comparação entre os projetos dos trabalhos relacionados e o projeto implementado neste

Autor	**Interface Web Online**	**BD na Nuvem**	**Possui cópia do BD da nuvem**	**Efetua ações offline**	**Protocolo da camada de aplicação**	**Protocolo da camada física**
Freitas (2020)	Não	Não	Parcial	Sim	HTTP	WiFi
Day (2016)	Não	Não	Parcial	Sim	MODBus	Não Especificado
SILVA (2018)	Sim	Sim	Não	Não	HTTP	WiFi
Projeto descrito neste trabalho	Sim	Sim	Sim	Sim	WebSocket e HTTP	Ethernet e WiFi

Fonte: Autor (2022)

8.2 DIFICULDADES

A principal dificuldade se deu pela quantidade de tecnologias utilizadas e a complexidade que o projeto foi ganhando à medida que funcionalidades iam sendo implementas. No total, foram milhares de linhas de código escritas, divididas em seis linguagens (PHP, HTML, CSS, JavaScript, SQL e C/C++) bem diferentes umas das outras e diferentes tecnologias para backend, frontend, cliente-servidor e hardware, como o NodeJS, Laravel, MariaDB, MySQL e Heroku. A tabela 18 mostra com mais detalhes informações da implementação.

Tabela 18 – Tabela com a listagem dos componentes escolhidos

	Linguagens	**Tecnologias**	**Informações de implementação**
Subsistema na baia	C/C++	Componentes eletrônicos e protocolos	Um arquivo com 306 linhas de código
Subsistema no gateway	JavaScript e MariaDB	Node.Js, Socket e Axios	Mais de 800 linhas de código, divididas entre 6 arquivos

Subsistema no backend	PHP e MySQL	Laravel, Banco de dados e Heroku	Mais de 2500 linhas de código, divididas entre 30 arquivos
Subsistema no frontend	PHP, HTML, CSS e JavaScript	Laravel, Livewire, SweetAlert e Heroku	Mais de 2500 linhas de código também, divididas entre 36 arquivos

Fonte: Autor (2022)

Outra grande dificuldade foi imposta pela pandemia, pois antes de iniciar a pesquisa aplicada neste trabalho, uma outra linha de pesquisa foi trabalhada durante longos meses até ter sido interrompida por conta de restrições que impossibilitava a aplicação de testes em laboratório.

Por último, juntar todos os sensores em uma única *protoboard* precisou de muita atenção e cuidado porque algumas vezes os sensores paravam de funcionar sem um motivo claro e na maioria dos casos a causa era algum mal contato, então, sempre que tinham falhas durante os testes era preciso parar a implementação, investigar qual era o sensor que estava falhando e testar diversas soluções para que esse problema não voltasse a acontecer.

8.3 TRABALHOS FUTUROS

Para dar continuidade ao trabalho de pesquisa descrito nesta dissertação, lista-se, nesta seção, propostas de trabalhos futuros a serem realizadas, que são:

- A criação do circuíto impresso e um compartimento seguro dentro da baia para que este circuíto não seja violado;
- A integração do *gateway* com o sistema do Centro de Informática com a finalidade de ter acesso a dados reais dos alunos e funcionários;
- A capacidade do *gateway* de conseguir abrir e fechar baia de maneira 100% *offline* sem comprometer a segurança dos equipamentos;
- Melhorias de interface e na responsividade do portal.

REFERÊNCIAS

ALMEIDA, P. M. O. *WebSockets e a sua aplicação no mundo Web*. Dissertação (Mestrado em Engenharia Informática) — Instituto Politécnico do Porto, Portugal, 2019.

ALTEXSOFT. *IoT Architecture: the Pathway from Physical Signals to Business Decisions*. 2020. Disponível em: <https://www.altexsoft.com/blog/iot-architecture-layers-components/>. Acesso em: 02 fev. 2022.

AMORIM, L. E. de. *Sistema de Monitoramento do Uso dos Equipamentos no Laboratório de Hardware baseado em RFID*. 51 p. Monografia (Trabalho de Conclusão de Curso) — Centro de Informática, Universidade Federal de Pernambuco, Recife, 2020.

ANDRADE, A. P. de. *Principais comandos SQL*. 2020. Disponível em: <https://www.treinaweb.com.br/blog/principais-comandos-sql>. Acesso em: 11 fev. 2022.

ARDUINO. *About Arduino*. 2021. Disponível em: <https://www.arduino.cc/en/about>. Acesso em: 10 fev. 2022.

ARDUINO. *Arduino UNO R3*. 2021. Disponível em: <https://docs.arduino.cc/hardware/uno-rev3>. Acesso em: 10 fev. 2022.

BERTONCELLO, C. C. S. A cadeia de suprimento do exército brasileiro: o uso do sistema rfid na gestão do suprimento classe ii. 2018.

BORGIA, E. The internet of things vision: Key features, applications and open issues. *Computer Communications*, v. 54, p. 1–31, 2014. ISSN 0140-3664.

BROWN, D. *RFID Implementation*. [S.l.]: Editora McGraw-Hill, 2006.

BYERS, C. C. *IoT Actuators and Sensors*. 2019. Disponível em: <https://br.mouser.com/blog/blog/blog/iot-actuators-and-sensors>. Acesso em: 02 fev. 2022.

CABALLERO, D. C. *How to use the Arduino Ethernet Shield*. 2016. Disponível em: <https://scidle.com/how-to-use-the-arduino-ethernet-shield/>. Acesso em: 11 fev. 2022.

CAFE, R. *IFF – Identification Friend or Foe. Erie*. 2011. Disponível em: <http://www.rfcafe.com/references/electrical/ew-radar-handbook/iff-identification-friend-or-foe.htm>. Acesso em: 06 fev. 2022.

COETZEE, L.; EKSTEEN, J. The internet of things - promise for the future? an introduction. p. 1–9, 2011.

CUNHA, A. *RFID – Etiquetas com eletrônica de ponta*. 2016. Disponível em: <https://www.embarcados.com.br/rfid-etiquetas-com-eletronica-de-ponta/>. Acesso em: 08 fev. 2022.

DAY, K. C. *AUTOMAÇÃO DE SEGURANÇA E CONTROLE DE ACESSO DE SALA EM EMISSORA DE RÁDIO E TV*. Dissertação (Mestrado em Mecatrônica) — Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Santa Catarina, 2016.

DIZDAREVIć J., C. F. J. A. . M.-B. X. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, p. 1–29, 2019.

DOBKIN, D. *The RF in RFID Passive UHF RFID in Practice*. [S.l.]: Oxford: Newnes, 2008.

EVANS, D. *The Internet of Things - How the Next Evolution of the Internet Is Changing Everything*. [S.l.]: Cisco Internet Business Solutions Group, 2011.

FINKENZELLER, K. *RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication*. [S.l.]: John wiley & sons, 2010.

FREITAS, R. B. d. *Controle e segurança patrimonial por RFID no Departamento Acadêmico de Eletrônica da UTFPR Campo Mourão*. Dissertação (Mestrado em Inovações Tecnológicas) — Universidade Tecnológica Federal do Paraná, Paraná, 2020.

IVY, B.; CHEATWOOD, E. *RFID – What You Need To Know*. 2022. Disponível em: <http://www.awareinnovations.com/2022/03/23/rfid-overview/>. Acesso em: 30 mar. 2022.

JGMAKER. *Flashing a new bootloader on the A5/A3S*. 2020. Disponível em: <https://jgmakerwiki.com/a5/bootloader>. Acesso em: 11 fev. 2022.

LIUKKONEN, M. Rfid technology in manufacturing and supply chain. *International Journal of Computer Integrated Manufacturing*, Taylor Francis, v. 28, n. 8, p. 861–880, 2015. Disponível em: <https://doi.org/10.1080/0951192X.2014.941406>.

MCKINSEY, Q. Why retail wants radio tags. 2003.

MONTALVãO, A. C. P. d. S. *Estudo da conversão de polarização linear-circular em antenas dual-band para leitores RFID portáteis usando metasuperfícies miniaturizadas*. Tese (Mestrado em Engenharia Elétrica e de Computação) — Universidade Federal do Rio Grande do Norte, Rio Grande do Norte, 2016.

OLIVEIRA, F. C. P. de. *Uso de RFID na gestão de artigos retornáveis em cadeias de distribuição tipo closed-loop*. Tese (Doutorado) — PUC-Rio, 2013.

PEDROSO M. C., Z. R.; SOUZA, C. A. d. Adoção de rfid no brasil: um estudo exploratório. ram. revista de administração mackenzie. v. 10, p. 12–36, 2009. ISSN 1678-6971.

PROTTO. *Entendendo os componentes e a pinagem dos Arduinos.* 2020. Disponível em: <https://protto.com.br/2020/10/19/entendendo-os-componentes-e-a-pinagem-dos-arduinos/>. Acesso em: 10 fev. 2022.

RAMAKRISHNAN, R.; GEHRKE, J. *Database management systems, 3rd Edition*. [S.l.]: McGraw-Hill, 2002.

RODRIGUES, A.; BORGES, E.; BARWALDT, R. Um estudo sobre o comportamento alimentar de frangos de corte utilizando a mineração de dados. *Scientia Plena*, v. 13, 2017.

SANTOS, B. P.; SILVA, L.; CELES, C.; BORGES, J. B.; NETO, B. S. P.; VIEIRA, M. A. M.; VIEIRA, L. F. M.; GOUSSEVSKAIA, O. N.; LOUREIRO, A. Internet das coisas: da teoria à prática. *Minicursos SBRC-Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuıdos*, 2016.

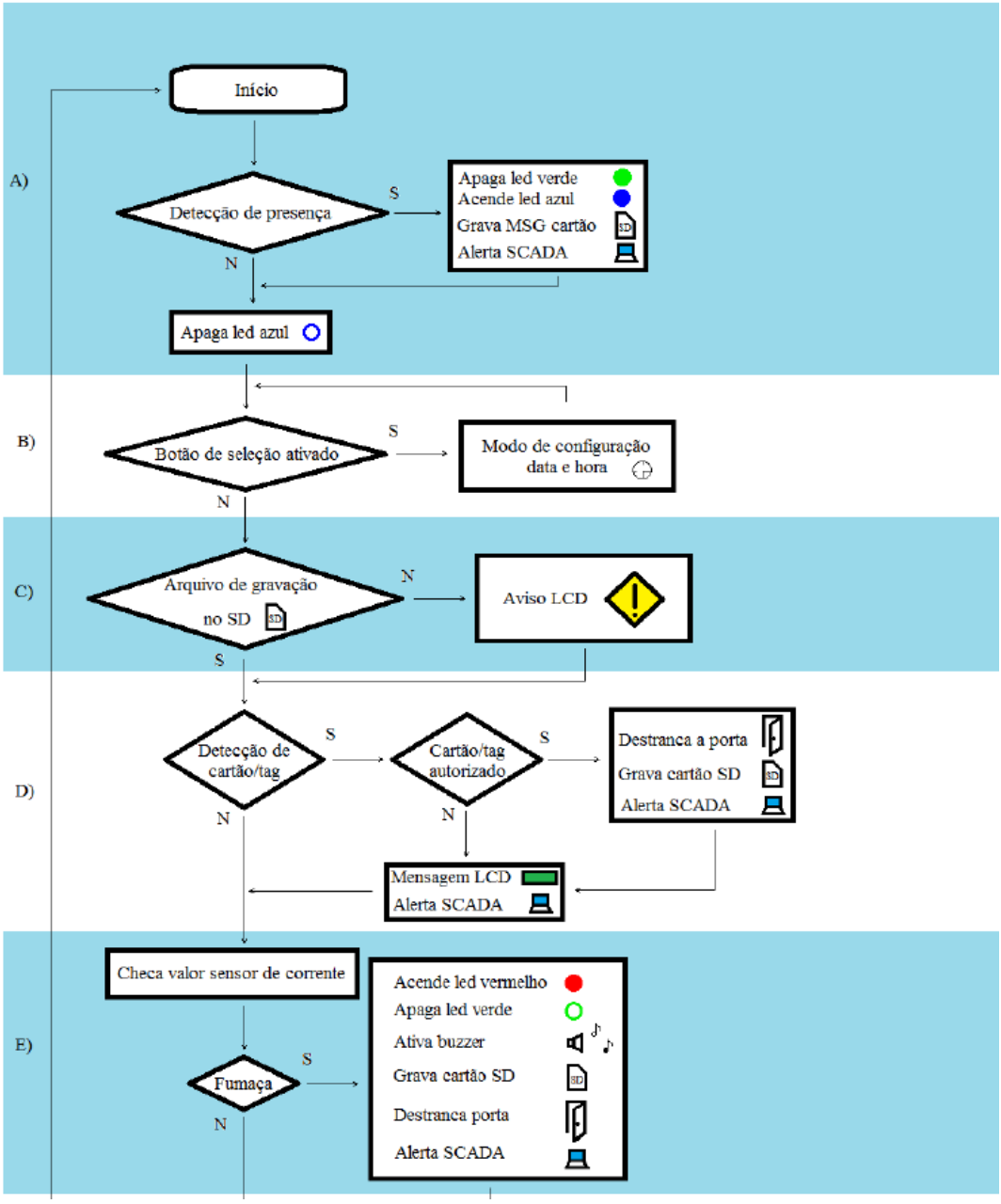
SILVA, B. A. J. O uso da iot no controle de frequÊncia dos alunos da unip. *18º Congresso Nacional de Iniciação Científica*, 2018.

SOLUçõES, I. *O que são Etiquetas RFID?* 2020. Disponível em: <https://i3csolucoes.com.br/o-que-sao-etiquetas-tags-rfid/>. Acesso em: 08 fev. 2022.

TEIXEIRA, T. *ANÁLISE DAS ANTENAS UTILIZADAS EM LEITORES RFId*. Monografia (Monografia) — Instituto Federal de Santa Catarina, Santa Catarina, 2017.

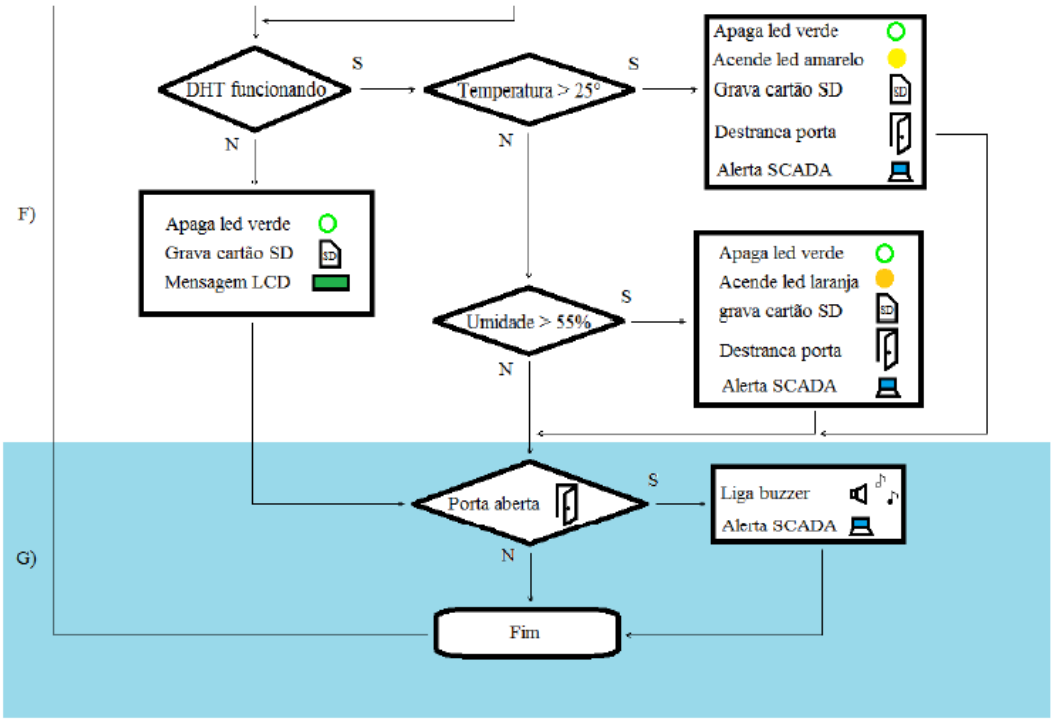
APÊNDICE A – FLUXOGRAMA DO FUNCIONAMENTO DO PROJETO NO TRABALHO DE DAY (2016)

Figura 91 – Fluxograma Arduíno parte 1



Fonte: Day (2016)

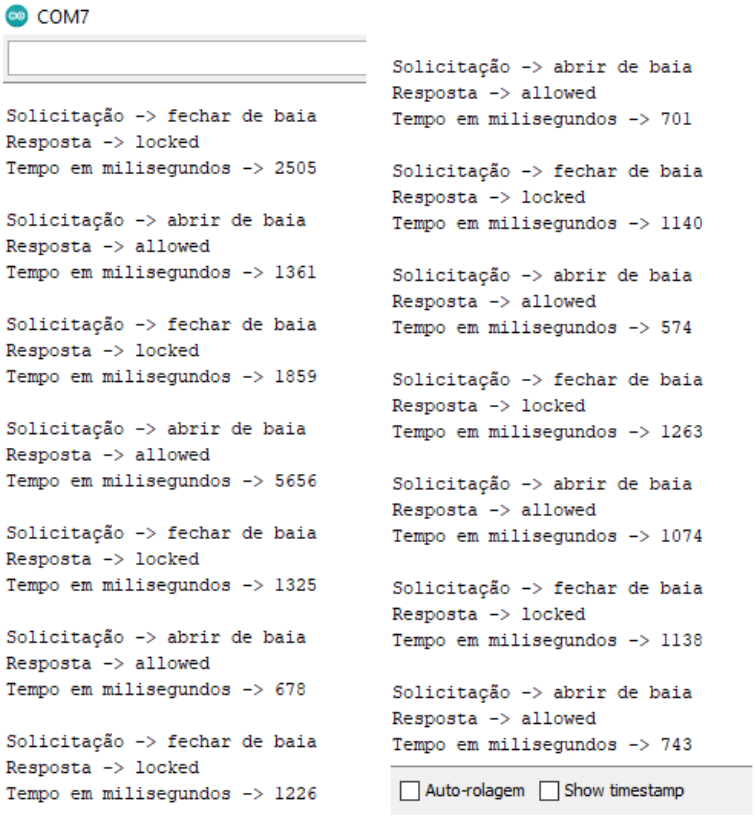
Figura 92 – Fluxograma Arduíno parte 2



Fonte: Day (2016)

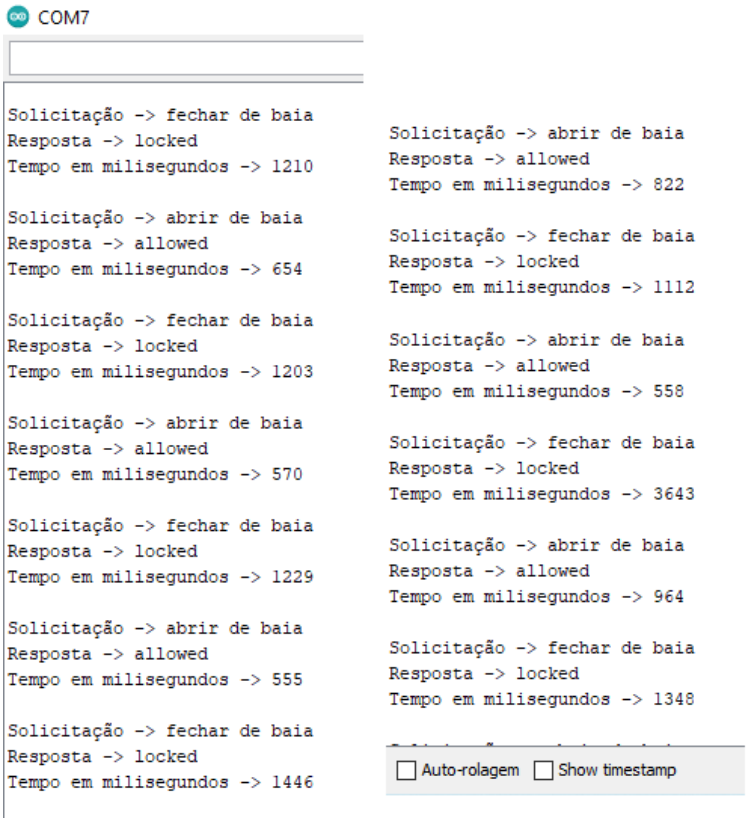
APÊNDICE B – RESULTADOS DAS SIMULAÇÕES NO ARDUÍNO

Figura 93 – Monitor serial do Arduíno imprimindo os dados a 1ª até a 14ª solicitação



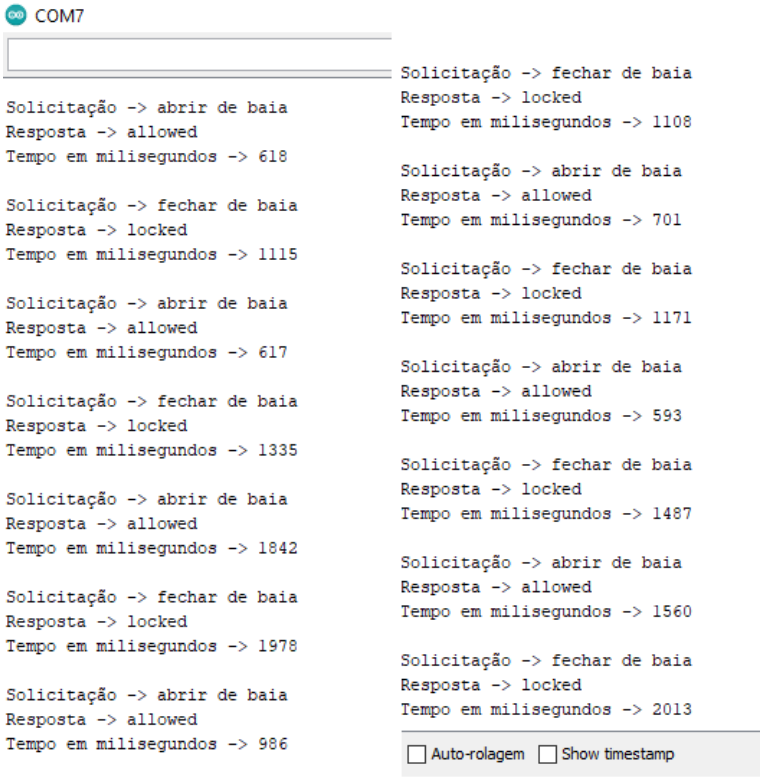
Fonte: Autor (2022)

Figura 94 – Monitor serial do Arduíno imprimindo os dados a 15ª até a 27ª solicitação



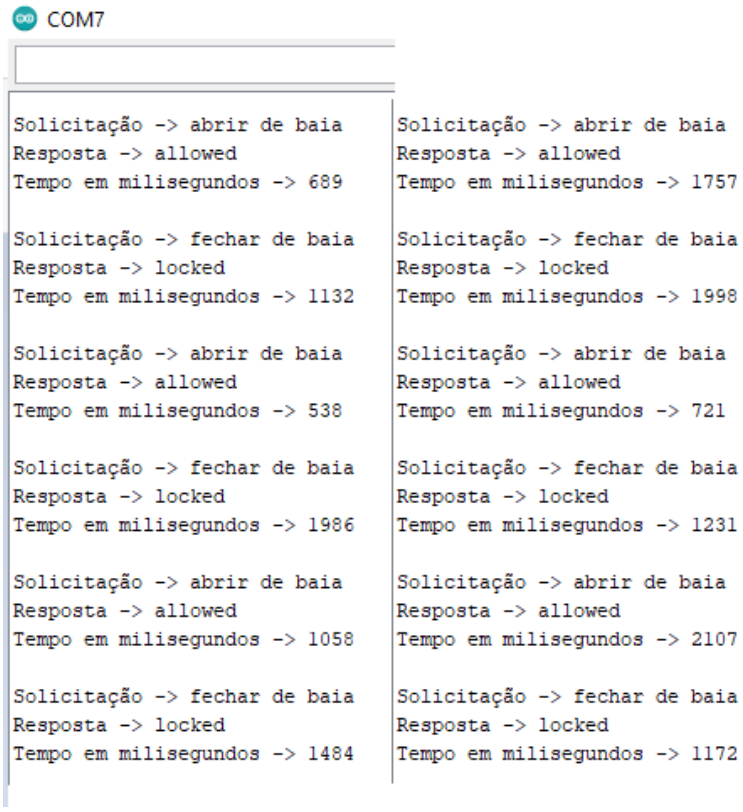
Fonte: Autor (2022)

Figura 95 – Monitor serial do Arduíno imprimindo os dados a 28ª até a 41ª solicitação



Fonte: Autor (2022)

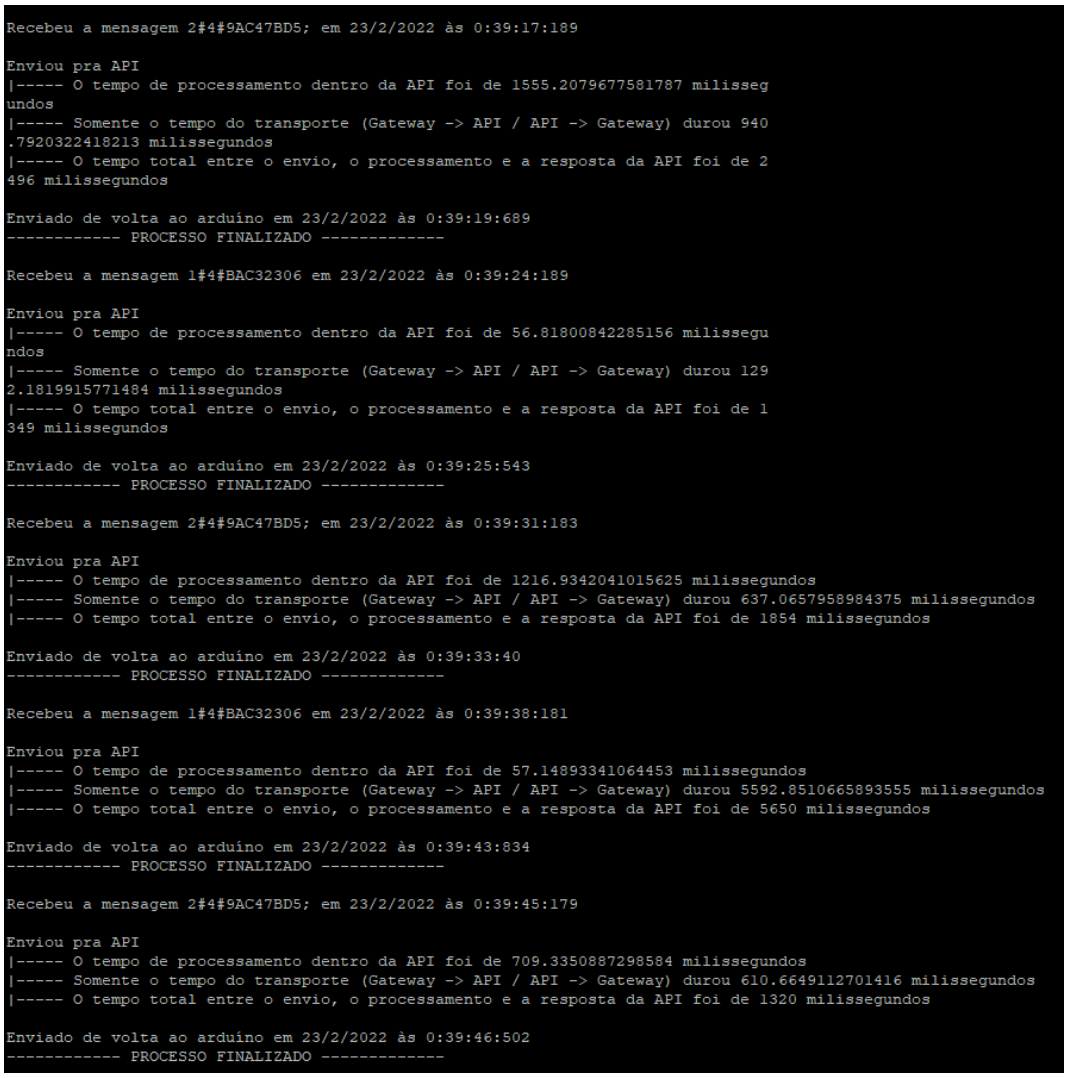
Figura 96 – Monitor serial do Arduíno imprimindo os dados a 42ª até a 53ª solicitação



Fonte: Autor (2022)

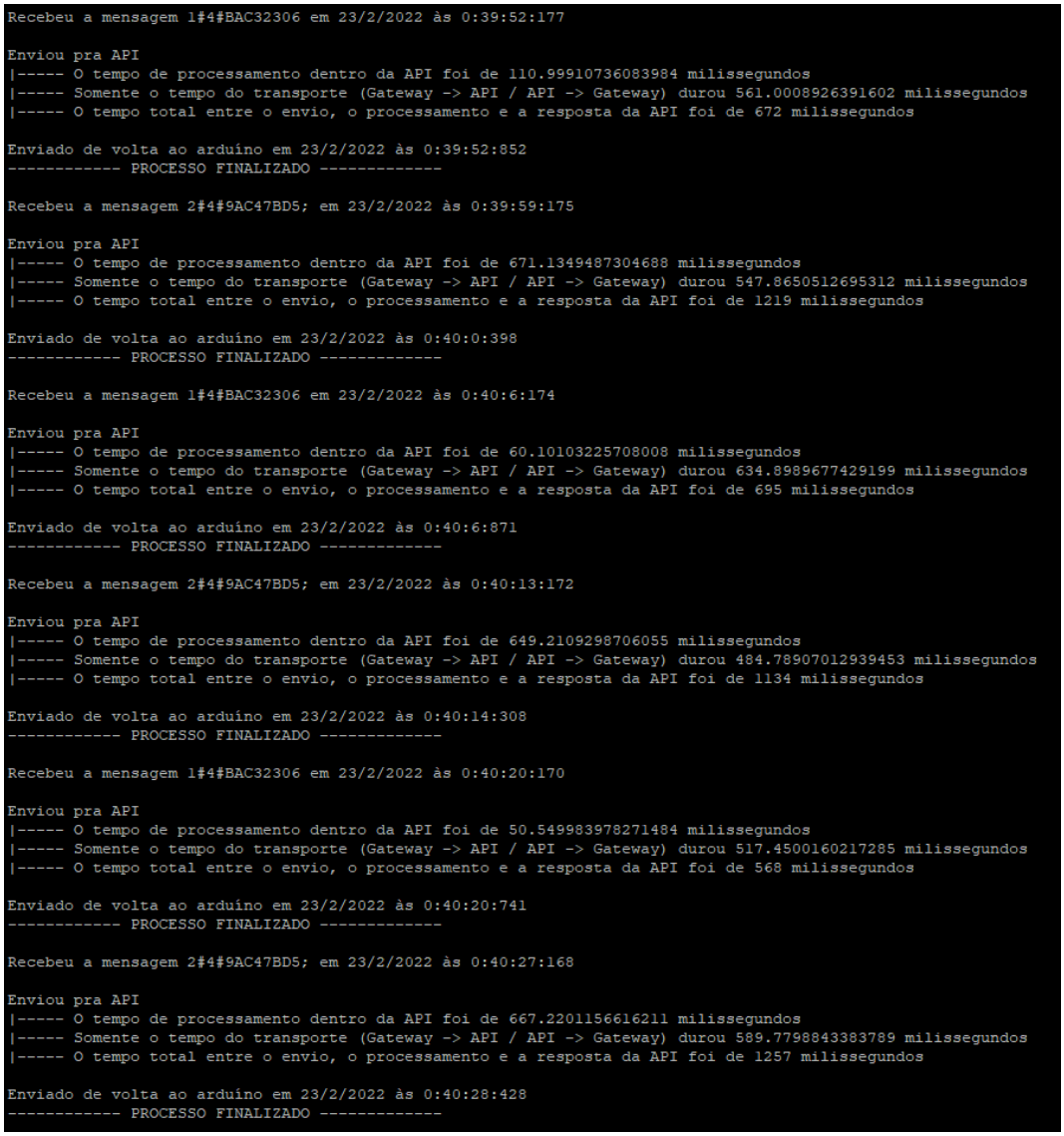
APÊNDICE C – RESULTADOS DAS SIMULAÇÕES NO GATEWAY

Figura 97 – *Prompt* de comando do *gateway* com a 1ª até a 5ª solicitação



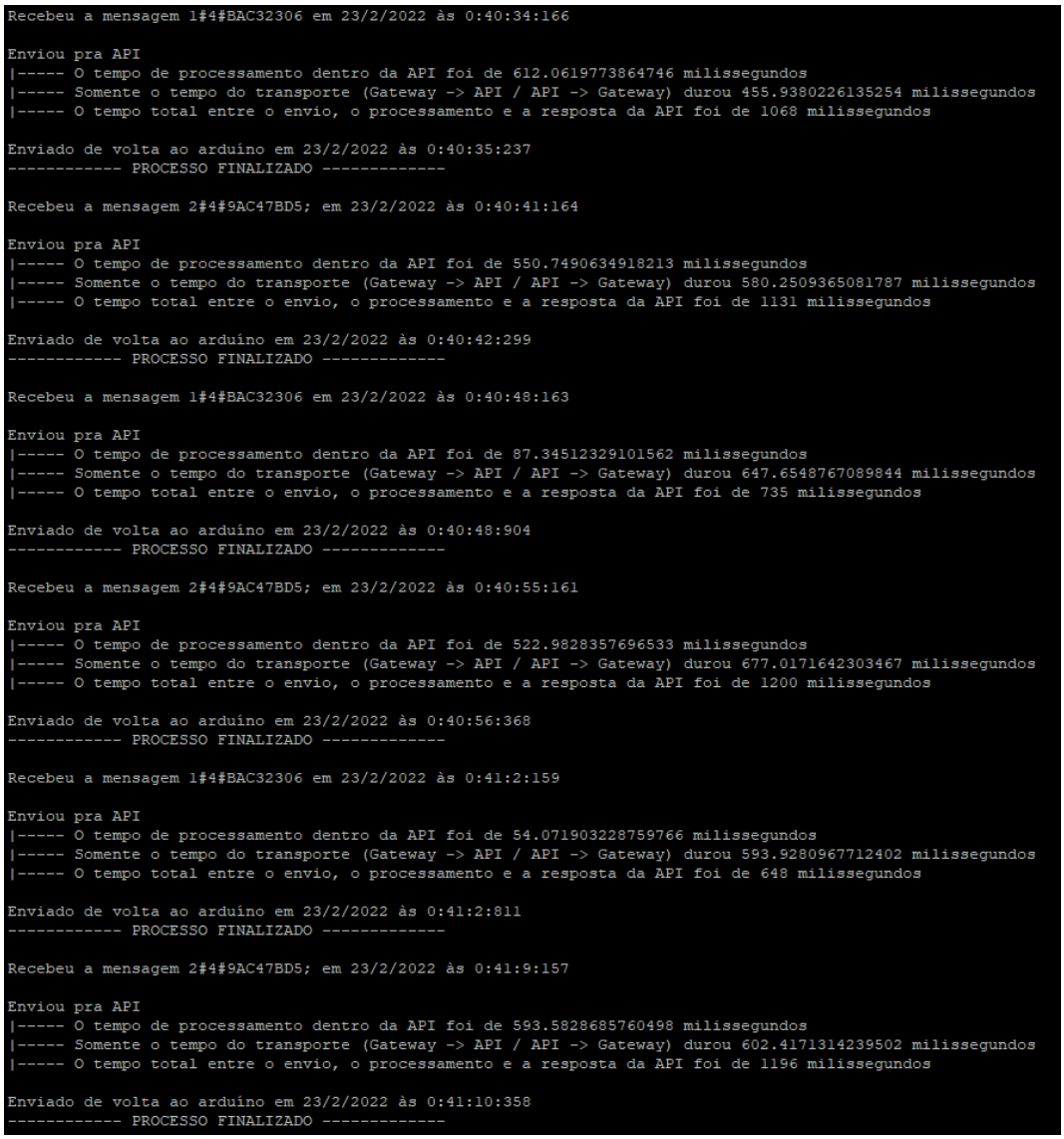
Fonte: Autor (2022)

Figura 98 – *Prompt* de comando do *gateway* com a 6ª até a 11ª solicitação



Fonte: Autor (2022)

Figura 99 – *Prompt* de comando do *gateway* com a 12ª até a 17ª solicitação



Fonte: Autor (2022)

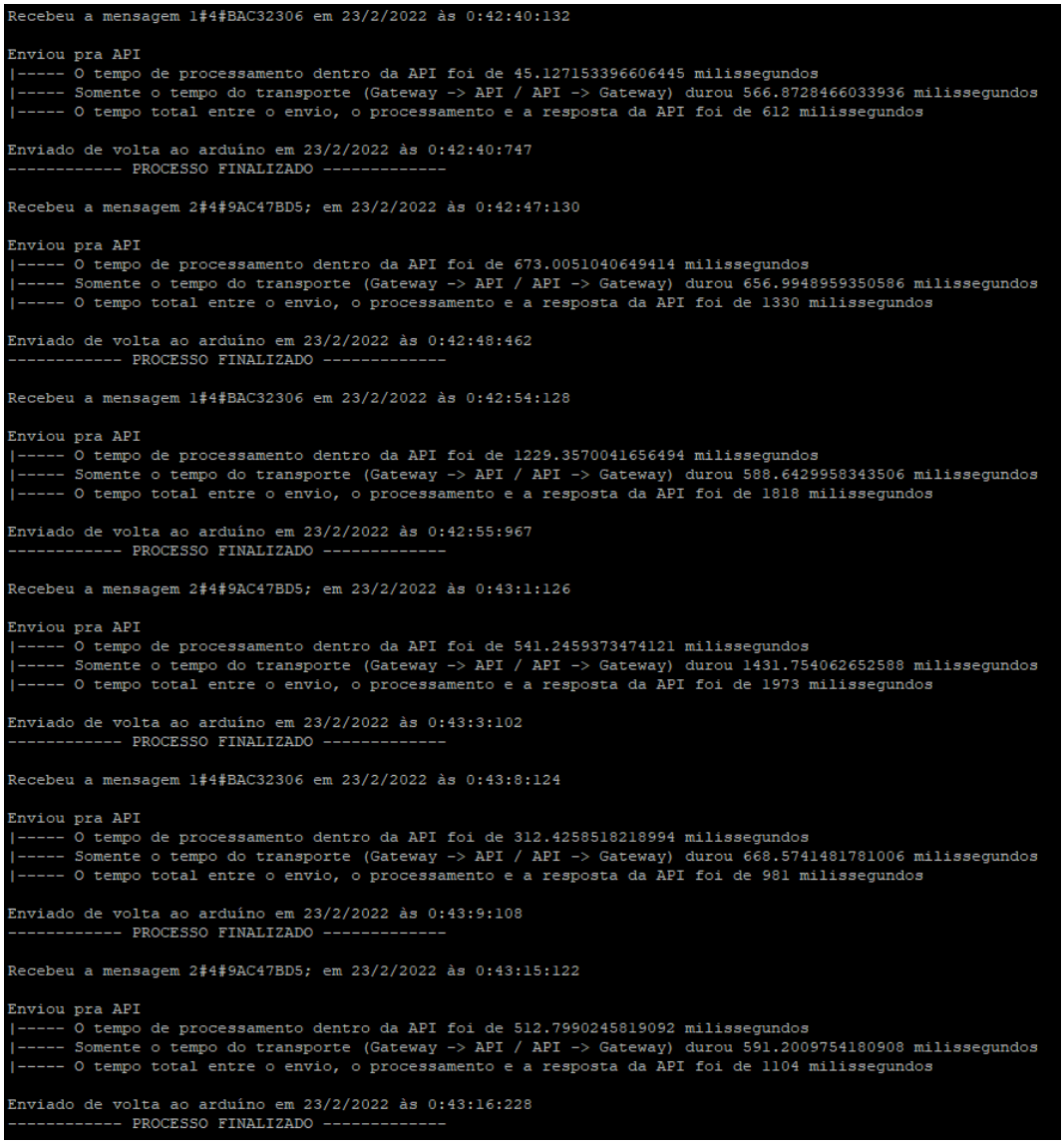
Figura 100 – *Prompt* de comando do *gateway* com a 18ª até a 23ª solicitação

Fonte: Autor (2022)

Figura 101 – *Prompt* de comando do *gateway* com a 24ª até a 29ª solicitação

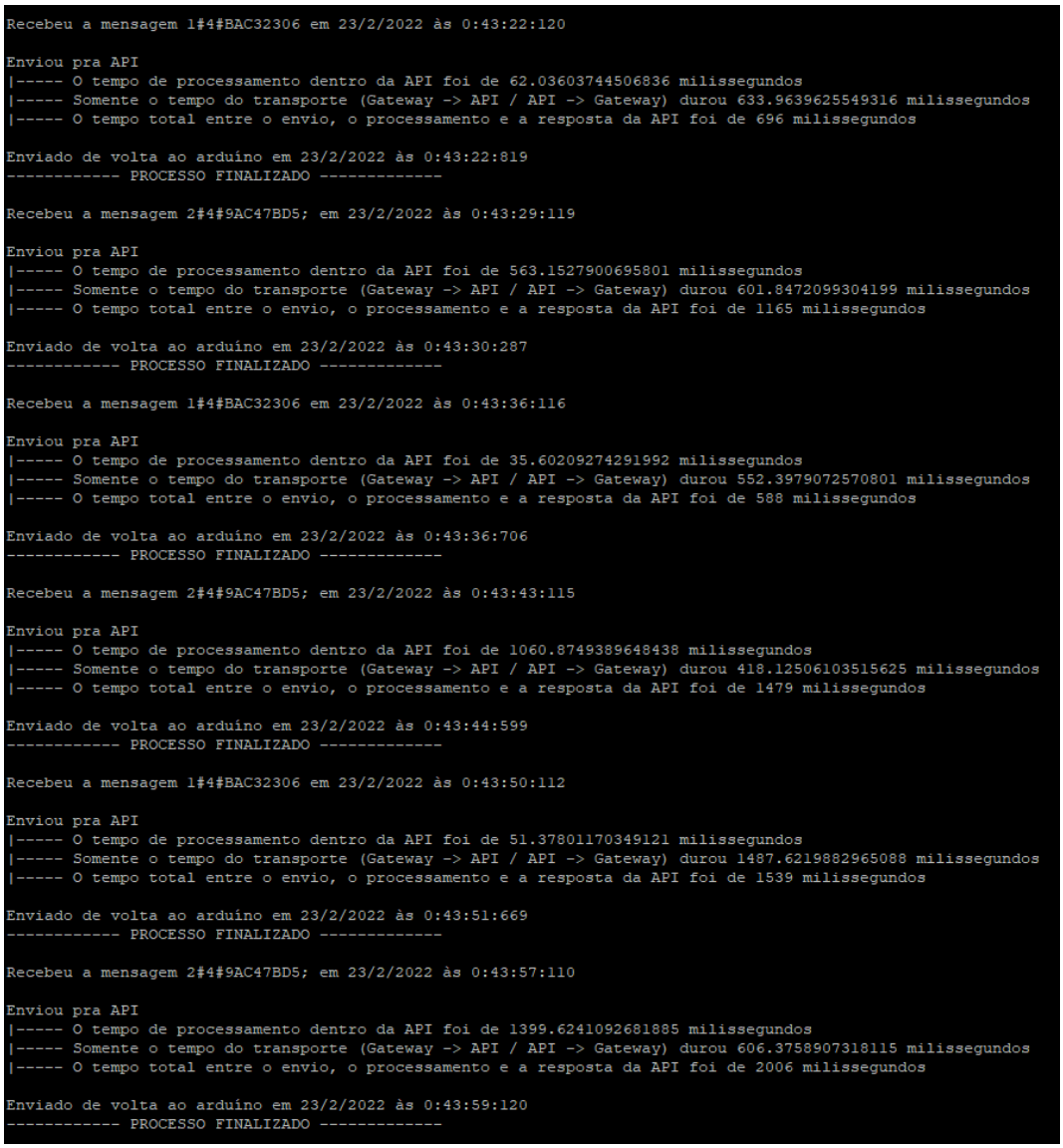
Fonte: Autor (2022)

Figura 102 – *Prompt* de comando do *gateway* com a 30ª até a 35ª solicitação



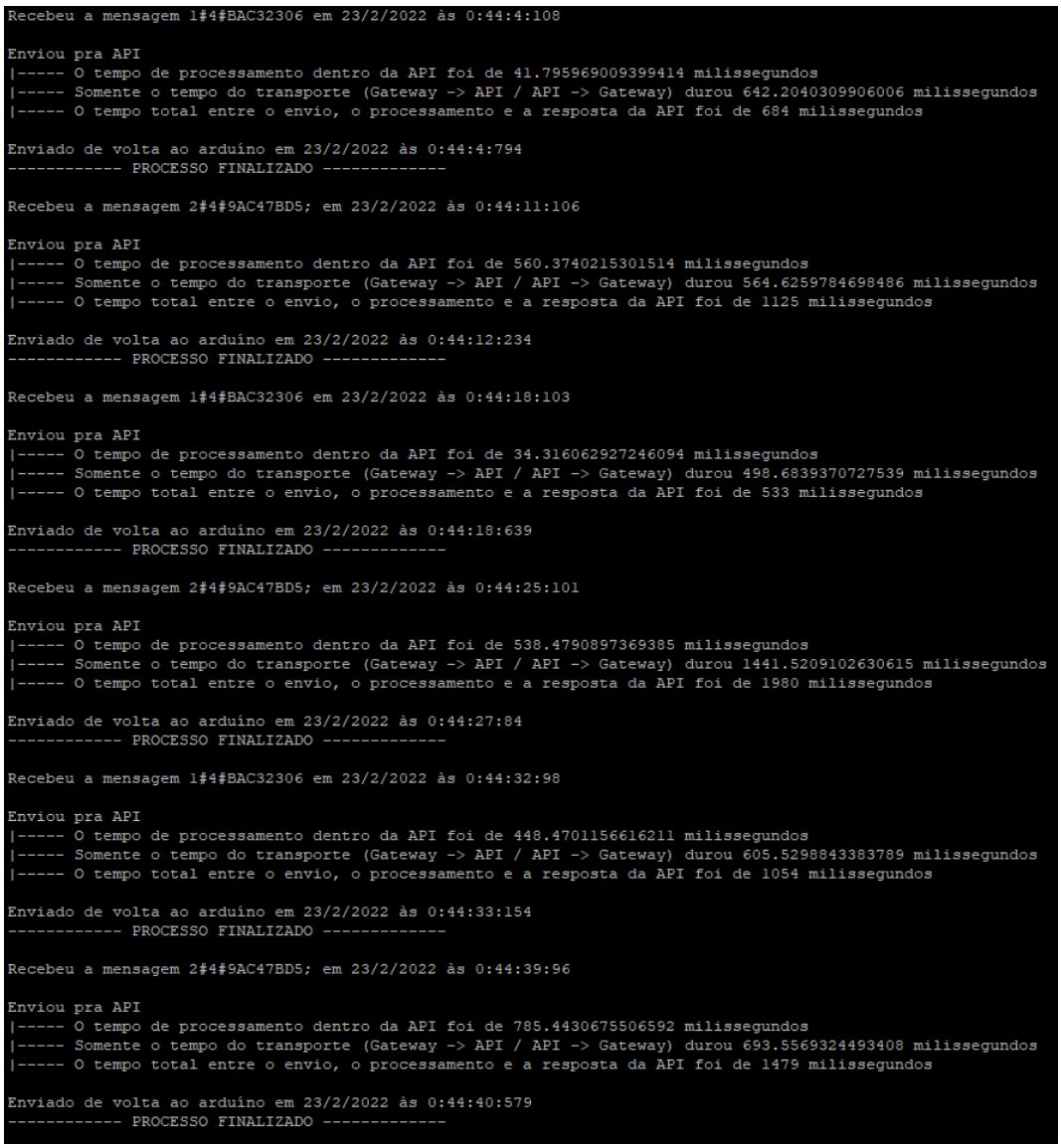
Fonte: Autor (2022)

Figura 103 – *Prompt* de comando do *gateway* com a 36ª até a 41ª solicitação



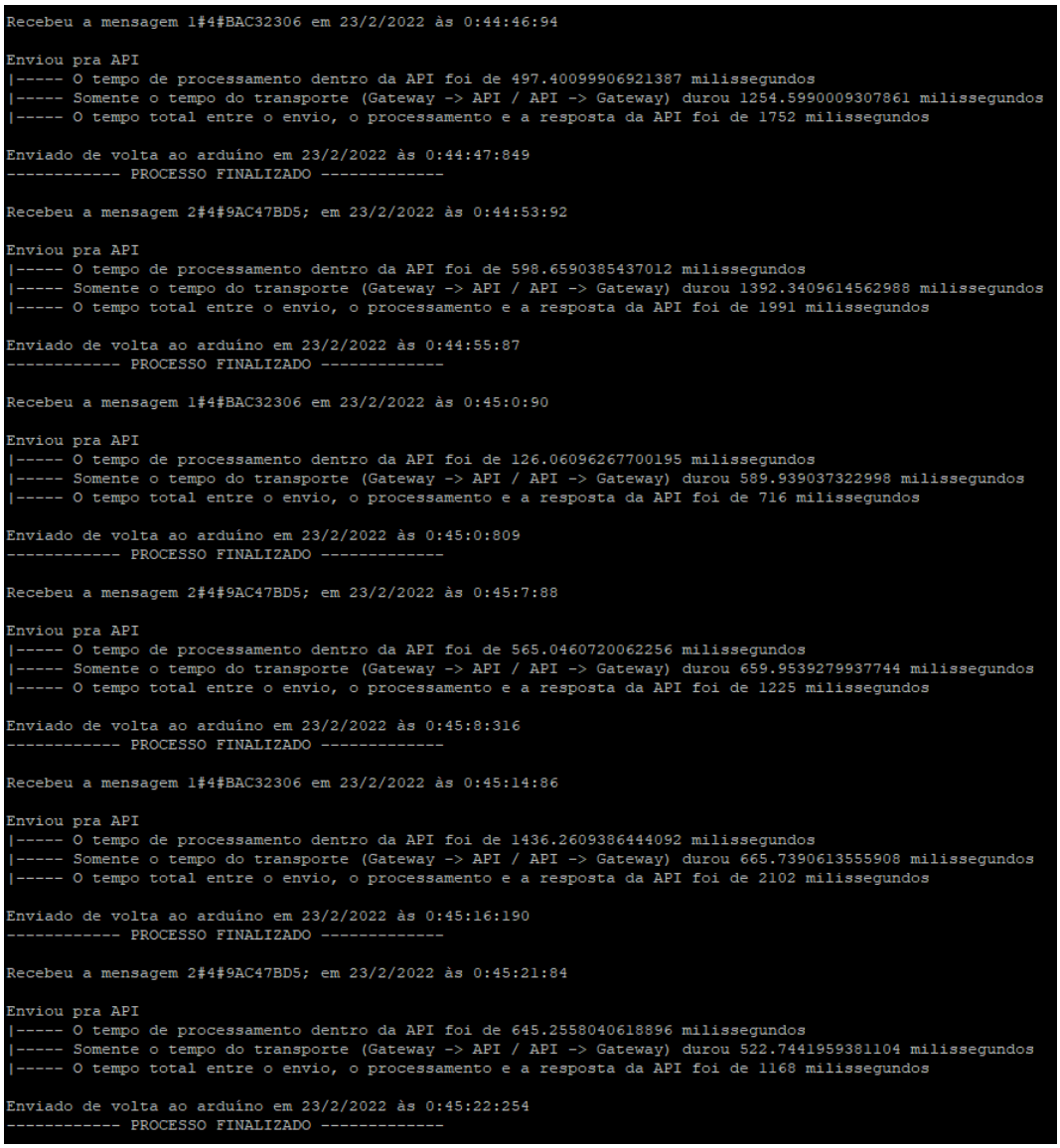
Fonte: Autor (2022)

Figura 104 – *Prompt* de comando do *gateway* com a 42ª até a 47ª solicitação



Fonte: Autor (2022)

Figura 105 – *Prompt* de comando do *gateway* com a 48ª até a 53ª solicitação



Fonte: Autor (2022)