HÉRCULES ZAGURY NAKAI FERREIRA

PROJETO DE UM SISTEMA ADMINISTRATIVO E AUTOATENDIMENTO PARA COMERCIALIZAÇÃO DE CHOPE

LAVRAS – MG

2019

HÉRCULES ZAGURY NAKAI FERREIRA

PROJETO DE UM SISTEMA ADMINISTRATIVO E AUTOATENDIMENTO PARA COMERCIALIZAÇÃO DE CHOPE

Trabalho de conclusão de Curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

Prof. Daniel Furtado Leite

Orientador

LAVRAS – MG

2019

HÉRCULES ZAGURY NAKAI FERREIRA

PROJETO DE UM SISTEMA ADMINISTRATIVO E AUTOATENDIMENTO PARA COMERCIALIZAÇÃO DE CHOPE

DESIGN OF AN ADMINISTRATIVE SYSTEM FOR DRAFT BEER COMMERCE

Trabalho de conclusão de curso apresentado à Universidade Federal de Lavras, como parte das exigências do Curso de Engenharia de Controle e Automação, para a obtenção do título de Bacharel.

APROVADA 19 de novembro de 2019.

Prof. Dr. Daniel Furtado Leite	DAT UFLA
Prof. Dr. Felipe Oliveira e Silva	DAT UFLA
Profª. Drª. Tatiane Carvalho Alvarenga	IF SUDESTE/MG BARBACENA

Prof. Daniel Furtado Leite

Orientador

LAVRAS – MG

2019

RESUMO

Este trabalho tem como objetivo registrar o desenvolvimento de um sistema automatizado para comercialização de chope, no qual, é permitido ao consumidor a realização de toda operação de forma independente, desde a compra até a retirada, bem como a escolha da bebida de sua preferência e a forma do pagamento do produto. Além disso, o sistema contempla um painel administrativo que permite o gerenciamento de dispositivos e sistemas embarcados que fazem parte da máquina. Para o desenvolvimento deste projeto, utiliza-se um banco de dados, no qual, as informações são armazenadas e utilizadas por todos os componentes e sistemas, quais sejam; um *backend*, responsável pela lógica de funcionamento do sistema, fluxo e manipulação dos dados e informações; o *frontend* composto pelo painel administrativo e por um *tablet* com um aplicativo próprio, ambos responsáveis por fornecer as interfaces gráficas e facilitar comunicação e interações do sistema com os usuário e colaboradores do sistema, respectivamente; e por fim, a utilização de um sistema embarcado, controlado por um microcontrolador ESP-WROOM-32, que é responsável por fazer o controle do envase do chope e a comunicação com o *backend.* Todos os sistemas utilizam de serviços e servidores disponibilizados pela Amazon Web Service, sendo necessário o acesso à internet para funcionamento do mesmo. O projeto construído se mostrou satisfatório, atendendo as especificações propostas de ser um sistema autônomo. O *hardware* realizou as atividades propostas com sucesso, assim como o painel administrativo e o *tablet*. Estes forneceram interfaces simples que tornaram o processo de compra mais rápido e eficiente do que os processos já existentes. Tudo isso permitiu que o fosse disponibilizado um sistema robusto e capaz de contornar situações adversas, como, por exemplo, a interrupção no fornecimento de energia e internet.

Palavras-chave: Chopeira. V*ending machine*. *Backend. Frontend*. Autosserviço de chope. Dispositivos IOT. Automação.

ABSTRACT

This paper reports the development of an automated system for the draft beer commerce, in which the consumer is allowed to perform all operations independently, from purchase to picking-up, including the choice of the beverage and payment. In addition, the system has an administrative panel that allows the management of the embedded devices and systems that are part of the machine. For the development of this project, a database is used, in which the information is stored and used by all components and systems, namely; a backend, responsible for system logic, data and information flow and manipulation; the administrative panel frontend and a tablet with its own application, both responsible for providing graphical interfaces and facilitating the system communication and interactions with system users and collaborators, respectively; and finally, the use of an embedded system, controlled by an ESP-WROOM-32 microcontroller, which is responsible for controlling the draft beer filling and the communicating with the backend. All systems use services and servers provided by Amazon Web Service, requiring internet access to function. The built project proved to be satisfactory, meeting the proposed specifications of being an autonomous system. The hardware successfully performed the proposed activities, as did the administrative panel and the tablet. These provided simple interfaces that made the purchasing process faster and more efficient than existing processes. All of this allowed a robust system to be made available and able to overcome adverse situations, such as the interruption of power supply and internet.

Keywords: Vending machine. Backend. Frontend. Self-service. IOT devices. Automation.

LISTA DE FIGURAS

Figura 2.1 – Sistema de autosserviço EasyChopp 12
Figura 2.2 – ESP-WROOM-32 fabricado pela Espressif 13
Figura 2.3 – Banco de dados relacional 15
Figura 3.1 – Representação da troca de informações entre dispositivos 18
Figura 3.2 – Interface de configuração de instância EC2 da AWS 19
Figura 3.3 – Interface inicial do servidor. 20
Figura 3.4 – Projeto da PCI 22
Figura 3.5 – *Tablet* Voyager III 22
Figura 3.6 – Moderninha Plus 23
Figura 3.7 – Tabela *hardwares* do banco de dados 24
Figura 3.8 – Tabelas do banco de dados e suas relações. 25
Figura 3.9 – Requisição e feita ao servidor 27
Figura 3.10 – Resposta em formato JSON. 28
Figura 3.11 – *Token* JWT codificado. 29
Figura 3.12 – *Token* JWT decodificado. 29
Figura 3.13 – Fluxograma processo de *login*. 30
Figura 3.14 – Fluxograma para validação do *Token* JWT. 31
Figura 3.15 – Interface de edição de torneiras do painel administrativo. 34
Figura 3.16 – Fluxograma para compra das ações do usuário. 35
Figura 3.17 – Fluxograma para compra do dispositivo. 35
Figura 3.18 – Fluxo do processo de envase. 38
Figura 4.1 – Montagem final da chopeira 40
Figura 4.2 – Interface inicial do aplicativo do *tablet*. 41
Figura 4.3 – Interface de seleção de forma de pagamento no aplicativo 41
Figura 4.4 – Interface de copos disponíveis no código inserido. 42
Figura 4.5 – Interface de envase concluído. 42
Figura 4.6 – Interface inicial do painel administrativo (Listar torneiras). 44
Figura 4.7 – Interface dos registros de vendas e retiradas das torneiras. 44
Figura 4.8 – Interface de edição das torneiras 45

Figura 4.9 – Interface de edição do sistema embarcado. 45
Figura 4.10 – Uso médio da CPU do servidor BM-*Backend* 46
Figura 4.11 – Uso máximo da CPU do servidor BM-*Backend* 46
Figura 4.12 – Uso médio da CPU do servidor BM-AdminPanel 47
Figura 4.13– Uso máximo da CPU do servidor BM-AdminPanel 47

SUMÁRIO

1 INTRODUÇÃO 9
1.1 Objetivos 9
1.2 Estrutura do trabalho. 10
2 REFERENCIAL TEÓRICO 11
2.1 Fundamentos 11
2.1.1 Chopeiras e *Vending Machines* 11
2.1.2 Sistemas embarcados 12
2.1.3 *Backend* 13
2.1.4 *Frontend* 13
2.1.5 Computação em nuvem 14
2.1.6 Banco de dados 15
2.2 Revisão da Literatura. 15
2.2.1 Máquinas de Autoatendimento. 15
2.2.2 *Vending Machines* de Chope 16
3 MATERIAIS E MÉTODOS. 18
3.1 Materiais. 18
3.1.1 Servidores e serviços 18
3.1.2 Sistemas embarcados e IOT 21
3.1.3 Interfaces microprocessadas 22
3.2 Métodos 23
3.2.1 Banco de Dados 23
3.2.2 *Backend.* 26
3.2.3 *Frontend* 32
3.2.3 *Firmware* 36
4 RESULTADOS E DISCUSSÃO. 40
5 CONCLUSÃO 49
REFERÊNCIAS 51

1. INTRODUÇÃO

No contexto de comercialização de bebidas frias, um dos principais desafios é a qualidade de entrega da bebida em temperatura ideal para consumo. Equipamentos como refrigeradores, *freezers* e geladeiras são possíveis soluções para este problema quando se referem a venda de bebidas unitárias. Sendo assim, quando se deseja fazer a venda fracionada do produto, é necessário que seja utilizado um equipamento capaz de refrigerar o líquido e que forneça a possibilidade de se servir a bebida, como é o caso das chopeiras. Estes equipamentos atuam muito bem no processo de manter a bebida em sua temperatura ideal, porem também possuem suas limitações, sendo elas relacionadas principalmente à necessidade de um funcionário para executar a tarefa de servir o chope, aumentando assim o custo de operação do estabelecimento com mão de obra e consequentemente do valor final do produto. Além disso, o processo manual aumenta o desperdício e acaba por dificultar o gerenciamento de vendas e estoque, dificultando melhorias nos processos e na gestão do negócio.

Além da contratação de mão de obra adicional necessária para realização da venda do produto, o que encarece o custo final do produto, os donos de estabelecimento acabam por enfrentar também no processo uma possível formação de filas no estabelecimento. Com o intuito de solucionar parte destes problemas, máquinas automatizadas de vendas, ou *vending machines*, começaram a surgir. Estas máquinas dispensam a necessidade de um funcionário para realizar o atendimento, simplificando também, em grande parte dos casos, a interação com o usuário, o pagamento e retirada do produto. Um exemplo muito comum destes tipos de máquinas no contexto de comercialização de bebidas são as máquinas de refrigerante. O consumidor se aproxima, escolhe o produto de acordo com as opções disponibilizadas na máquina, realiza o pagamento, e a máquina libera uma unidade da mercadoria. Contudo, o mercado brasileiro atualmente carece de soluções de *vending machines* direcionadas à venda de bebidas fracionadas frias, com custo reduzido e fácil instalação.

Este trabalho foi realizado sob um contexto de trabalho na empresa Beer Mine, que possui os direitos de patente do produto aqui descrito. Vale ressaltar que este trabalho foi executado com o auxilio de colaboradores da empresa que participaram do projeto.

1.1. Objetivos

A proposta deste trabalho é apresentar, sob forma de relatório técnico, o projeto de desenvolvimento do sistema administrativo e gerenciamento de dispositivos IOT (*Internet of Things*) de uma chopeira de autoatendimento (uma *vending machine* de chope) que permite que

o cliente realize o pedido, o pagamento e a retirada de um copo de chope, sem a necessidade de auxílio de um empregado da empresa. Os principais requisitos são que os sistemas aqui desenvolvidos forneçam a segurança necessária para troca de informações entre os dispositivos, usuários e plataformas do sistema, permitindo que os funcionários realizem as operações de gerenciamento e controle do sistema e que os dispositivos IOT tenham total suporte para realizar suas tarefas.

1.2. Estrutura do trabalho

O capítulo 2 apresenta breves descrições a respeito das bases dos assuntos discutidos no projeto e das técnicas utilizadas para desenvolvimento do mesmo. Além disso, é realizada uma revisão da literatura como forma de determinar como tais aspectos foram abordados por outros autores assim como quais técnicas foram utilizadas e os resultados obtidos.

O capítulo 3 lista os principais materiais e métodos utilizados na elaboração do projeto. Apresenta os diagramas que exemplificam e detalham o funcionamento e forma com que os *softwares* se interagem uns com os outros.

O capítulo 4 aborda os resultados obtidos após concluída as partes de execução do projeto e como cada um dos sistemas se comportou individualmente.

O capitulo 5 contém a conclusão e cita brevemente quais são as metas e implementações futuras para continuidade e melhoria do projeto.

2. REFERENCIAL TEÓRICO

2.1. Fundamentos

2.1.1. Chopeiras e *vending machines*

Chopeiras são equipamentos que têm a função de resfriar o chope e mantê-lo em uma temperatura aceitável para o consumo. Basicamente existem duas maneiras de se alcançar este objetivo. A primeira maneira é utilizando o gelo para trocar calor. Tais chopeiras são denominadas chopeiras a gelo. Nestas chopeiras, o chope passa por uma tubulação chamada de serpentina, que possui diâmetro reduzido (como forma de aumentar a área de contato e permitir maior troca de calor), dentro de um recipiente cheio de gelo. À medida que o chope percorre o caminho do barril até o copo ele perde calor. Consequentemente ele se torna ideal para o consumo.

O segundo tipo, denominado chopeira elétrica, possui funcionamento semelhante ao das geladeiras, consistindo em um compressor, que circula um gás refrigerante por um condensador, e é responsável por resfriar o gás até ele entrar em estado líquido, e por um evaporador, que é onde o liquido troca calor com o ar dentro da geladeira, até voltar para o estado gasoso (STEFFANI; POGLIA, 2013). A única diferença entre a chopeira elétrica e a geladeira é que na primeira existe um evaporador mergulhado em um liquido, juntamente com a serpentina por onde passa o chope. Assim, este liquido é responsável por realizar a troca de calor com a bebida, resfriando a até temperaturas abaixo de zero.

Vending machines são sistemas de venda automatizados que funcionam vinte e quatro horas por dia com objetivo de fornecer ao usuário diversos produtos como salgados, lanches, ingressos, brinquedos, bebidas alcoólicas e outros, sem a necessidade de uma pessoa para atendê-lo (BABU, 2011). O produto é escolhido através de uma interface fornecida ao usuário, onde posteriormente é realizado o pagamento, podendo este ser através de uma máquina de cartão de crédito, sistemas de aproximação ou mecanismos para pagamento em dinheiro. Após confirmação do pagamento, um mecanismo normalmente eletromecânico permite a conclusão do processo de venda fornecendo o produto ao usuário.

Hoje no mercado brasileiro é possível encontrar algumas empresas que disponibilizam serviços de autosserviço de chopes, como a EasyChopp. Localiza-se em São Paulo, e de acordo com o próprio site oficial da empresa, consiste em fornecer um cartão de rádio frequência que ao ser aproximado de um leitor de cartões identifica o mesmo e libera a torneira, permitindo

assim que o usuário possa retirar seu chope (Figura 2.1). Dessa forma o sistema contabiliza a quantidade de chope envasado, e debita no cartão o valor correspondente.

Figura 2.1 – Sistema de autosserviço *EasyChopp*.

Fonte: *EasyChopp* (2019)

2.1.2. Sistemas embarcados

Sistemas embarcados são sistemas compostos de *hardware* e *software* projetados exclusivamente para a execução de uma determinada tarefa. Estes sistemas são usados em diversos dispositivos modernos, como carros, televisões, máquinas de lavar e outros. A complexidade de um sistema embarcado varia significativamente de acordo com a tarefa a qual foi projetado para executar. Seu projeto de *hardware* pode ser baseado em um simples microcontrolador ou um conjunto de *chips* com sensores, atuadores e acesso a rede e internet (EBERT; SALECKER, 2009). Atualmente, um dos microcontroladores mais utilizados é o ESP-WROOM-32 (Figura 2.2), microcontrolador de 32 bits com 2 núcleos de processamento que podem operar em uma frequência de até 240MHz. O microcontrolador é fabricado pela *Espressif*, que se destaca pelo baixo custo, alta capacidade de processamento, e recursos como conectividade sem fio e Bluetooth 4.0, além de possuir suporte a porta USB, 512KB de SRAM e 4 MB de memória *Flash*.

Figura 2.2 – ESP-WROOM-32 fabricado pela Espressif.

Fonte: Espressif (2019).

2.1.3. *Backend*

O termo *Backend*, comumente citado como *server-side*, em termos de desenvolvimento de *softwares*, se refere a parte de um sistema ou aplicação em que são processadas as requisições feitas pelo usuário, *softwares* de integração e até mesmo sistemas embarcados. O *backend* é também onde são determinadas as regras de negócio, logica de funcionamento, principais mecanismos de segurança, aonde são realizadas as operações de banco de dados. O *backend* de uma aplicação pode ser implementado utilizando diversos *frameworks* ou bibliotecas, cada qual com suas linguagens de programação e características. Os *frameworks* auxiliam no desenvolvimento de *softwares* impondo funcionalidades, regras e metodologias para a composição de um *backend* de uma aplicação, sendo o *Laravel* um dos mais populares *frameworks* escritos em PHP de código aberto.

2.1.4. *Frontend*

Já o conceito de *frontend*, ou *client-side* no contexto de desenvolvimento de *software*, se refere a tudo aquilo que o usuário vê e interage em uma aplicação, sendo desta forma durante o desenvolvimento priorizada a usabilidade, experiência e *design* gráfico do sistema. Ao contrário do *backend*, no *frontend* pouco se desenvolve a lógica de funcionamento de um sistema. Entretanto, é por meio dessas interfaces gráficas que as requisições são enviadas e processadas para o *backend*, assim como suas respectivas respostas são tratadas.

É comum um sistema completo ser composto de diversas plataformas cada qual com seu *frontend*, por exemplo: um painel administrativo onde apenas a equipe técnica e colaboradores relacionados a operação têm acesso; e uma ou várias interfaces para o usuário padrão e consumidores. Sendo assim, o usuário padrão têm acesso limitado ao sistema, podendo realizar operações básicas, como, compras, registro de cartões de credito, visualização de pedidos já feitos e outros, enquanto o painel administrativo possibilitaria o monitoramento do sistema e um tratamento mais completo dos dados e informações que auxiliam na gestão empresarial, como, por exemplo, relatórios de faturamento, informações sobre vendas, gerenciamento de estoque e fornecedores.

O Android Studio é um exemplo de ambiente de desenvolvimento integrado para aplicativos Android, podendo ser utilizado o Java ou o Kotlin como linguagem de desenvolvimento. O mesmo possibilita desenvolver aplicativos *frontend* que podem ser utilizados em *tablets* e disponibilizados como interface gráficas para o usuário. Outro exemplo é o Angular, *framework* construído inteiramente baseado em linguagem TypeScript, comumente utilizado para desenvolvimento de *Single App Applications*, ou aplicações *web* que podem ser utilizadas tanto em computadores como dispositivos móveis. Essas aplicações funcionam como sites comuns e podem ser acessadas tanto por usuários convencionais como por administradores, sendo o controle e o tipo de informação exibida de acordo com o nível de acesso do *login* fornecido pelo usuário. Caso o mesmo tenha um *login* administrativo, o mesmo acessará o painel administrativo. Do contrário o mesmo terá acesso as funcionalidades básicas do sistema.

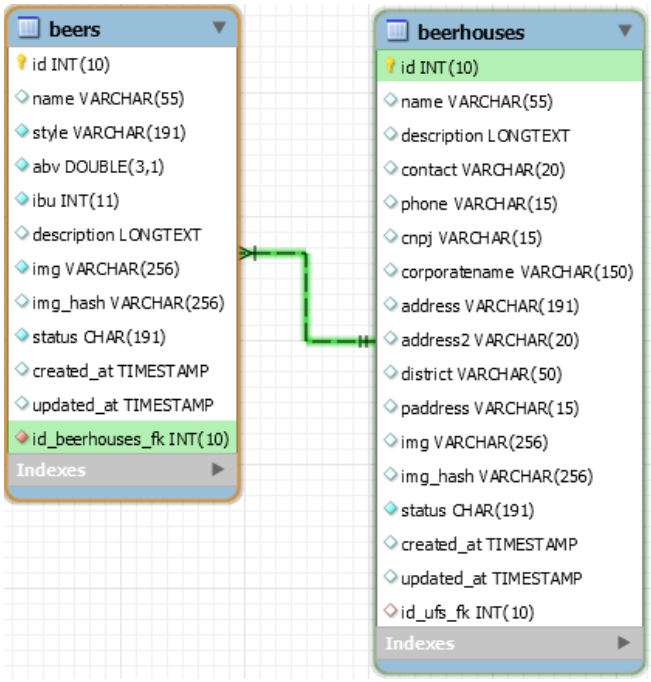
2.1.5. Computação em nuvem

A computação em nuvem é um modelo que permite o acesso virtual a qualquer mecanismo capaz de entregar e acessar serviços computacionais sobre demanda, sendo eles, servidores, armazenamento, banco de dados, aplicativos e diversos outros serviços via internet (MELL; GRANCE, 2001). A computação em nuvem é escalável e permite o rápido compartilhamento de recursos computacionais com o mínimo de esforço no gerenciamento do sistema, não sendo inclusive necessário que ocorra alterações ou interações físicas com o *hardware* em questão. O serviço de computação em nuvem escolhido pela empresa para hospedar todos os serviços mencionados anteriormente, são provenientes da Amazon Web Services (AWS). Serão detalhados cada um em seus respectivos aspectos e características.

2.1.6. *Banco de dados*

Todas as interações realizadas entre *backend* e *frontend* devem ser alimentadas por um conjunto de informações e dados que devem estar facilmente acessíveis, organizado e sempre atualizado. Esta coleção de informações e dados é chamada de Banco de Dados. Em um Banco de Dados Relacional as informações são organizadas em colunas e tabelas como forma de indexar e tornar fácil a procura e acesso às informações, como exemplifica a Figura 2.3, sendo o MariaDB um banco de dados relacional de código aberto, baseado no MySQL, que possibilita o controle de gravação e leitura das informações necessárias para o gerenciamento e funcionamento do sistema.

Figura 2.3 – Banco de dados relacional



Fonte: Do autor (2019)

2.2. Revisão da Literatura

2.2.1. Máquinas de autoatendimento

Sanders (2011) na Universidade de Brasília - Faculdade de Economia, Administração e Contabilidade, realizou um estudo empírico por meio de um questionário, no qual, analisou comportamentos de usuários ao utilizar sistemas de autoatendimento em aeroportos. O principal apontamento do estudo, demonstra que quando os usuários identificam a utilidade, facilidade, bem como o domínio sobre o equipamento, a abordagem e atitude dos consumidores são

positivas. Ainda que o estudo tenha sido realizado em um aeroporto, acredita-se que as pessoas tenham as mesmas atitudes positivas com relação à inserção de tecnologias, uma vez que exista uma demanda de atendimento.

Uma pesquisa de campo realizada por Hasegawa et al., (2008) na cidade de Guarapuava no Paraná, no Centro Universitário Campo Real, buscou analisar o nível de aceitação de inovações no comercio cervejeiro. A pesquisa foi realizada por meio de entrevistas, com caráter quantitativo, e apesar da amostra de entrevistados não ter sido escolhida de forma ampla, cerca de 92% dos entrevistados consideraram "muito interessante" ou "interessante" a presença de um sistema de *self-service* de chope, sendo atraídos pela facilidade de uso e por facilitar a forma de atender. No que se refere à Belo Horizonte, cidade em que o projeto em questão foi desenvolvido, não parece haver pesquisas com temas similares. No entanto, acredita-se que os resultados seriam semelhantes, uma vez que, não existem tantas distinções culturais em relação ao sul do país no que se refere ao uso de tecnologia em bares, além de que as faixas de idade e poder aquisitivo também se aproximam.

2.2.2. *Vending machines* de chope

Santos e Tanakaz (2017) desenvolveram uma mesa chopeira de autoatendimento, realizando o controle via Arduino e um sensor de vazão de efeito Hall, para aferir a quantidade de chope envasado. Por meio de um supervisório implementado no caixa, a mesa era liberada permitindo que os usuários servissem chope a vontade enquanto o sistema ia contabilizando a quantidade de chope servido. Após a utilização, os clientes deveriam se dirigir ao caixa e realizar o pagamento de acordo com o volume medido pelo sistema. Apesar do sistema contar com componentes não voltado ao chope, e as alterações no sistema de resfriamento, o sistema se apresentou funcional, atendendo aos objetivos propostos no trabalho.

Ortiz (2013), utilizando um Controlador Lógico Programável (CLP) e um sensor capacitivo para controlar o nível de chope, desenvolveu um projeto de chopeira automatizada. Apesar de funcional, o autor recomendou que fosse utilizada uma célula de carga ao invés de um sensor capacitivo, de forma a se obter "a harmonia perfeita entre o chope e a espuma". Além disso, o CLP mostrou-se um componente subutilizado para a execução do projeto, elevando o custo do projeto sem necessidade.

Gouveia e Lopes (2012), desenvolveram uma máquina de vazão de líquidos, utilizando um microcontrolador AT89S52. No cálculo de volume envasado, foram utilizadas somente funções de temporização, que pausam a execução do microcontrolador por um determinado

período de tempo. Apesar de satisfatórias, tal metodologia somente é funcional caso o fluxo da bebida envasada seja constante, o que não acontece quando se realiza o envase de chope extraído de um barril por meio de injeção de gases, uma vez que o fluxo depende de diversos fatores, como pressão do gás, comprimento e diâmetro da linha.

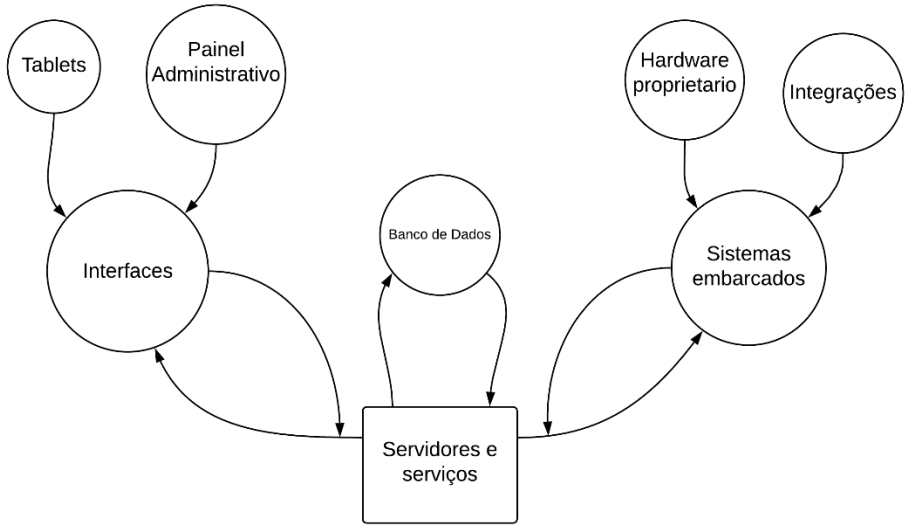
Em todos os sistemas acima mencionados, não existia um painel administrativo que possibilitasse o monitoramento em tempo real do sistema e o gerenciamento dos dispositivos.

3. MATERIAIS E MÉTODOS

3.1. Materiais

O sistema proposto neste trabalho é composto por dispositivos que podem ser subdivididos nos seguintes tópicos: servidores e serviços, sistema embarcado, e interfaces. Na Figura 3.1 tem-se o diagrama geral de funcionamento do sistema mostrando o fluxo de troca de informações com os sistemas e servidores.

Figura 3.1 – Representação da troca de informações entre dispositivos.



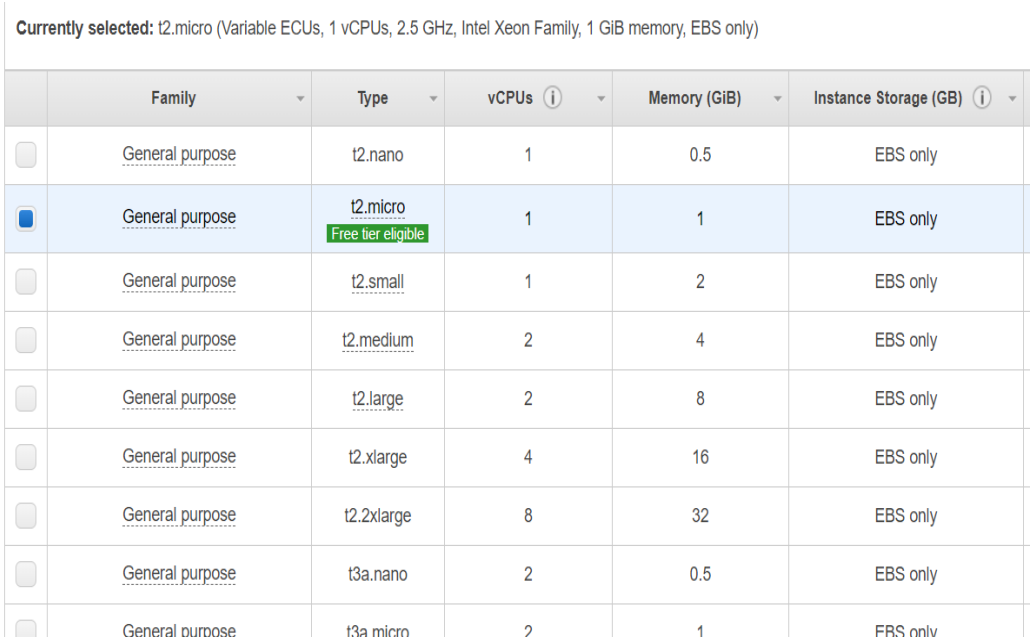
Fonte: Do autor (2019)

3.1.1. Servidores e serviços

Servidores podem ser definidos como computadores projetados para processar requisições e fornecer dados para outro computador por meio de uma rede local ou internet (MITCHELL, 2019). A capacidade de processar informações de um servidor está diretamente ligada ao *hardware*, ao tipo de requisição e serviços que estão sendo executados, e do sistema operacional que gerencia os recursos. Utilizando o serviço de computação em nuvem redimensionável fornecido pela Amazon Web Service, denominado *Elastic Compute Cloud* (EC2), é possível selecionar diversas configurações de servidores alterando os componentes que estão diretamente relacionados a sua capacidade de processamento como: quantidade de memória RAM, tipos de disco rígido (SSD ou HDD) e o espaço disponível nos mesmos, arquitetura do processador e sua quantidade de núcleos físicos ou virtuais, velocidade de conexão com a internet, e sistema operacional que será executado no servidor, como mostra a

Figura 3.2. Outras configurações estão disponíveis no console de gerenciamento da Amazon durante a criação de uma nova instância, como: selecionar as zonas e regiões onde o servidor será implementado, se ele será parte de uma sub-rede de computadores previamente implementados e se seus recursos de *hardware* poderão ser compartilhados com outros servidores ou se será uma instância dedicada.

Figura 3.2 – Interface de configuração de instância EC2 da AWS.



Fonte: Do autor (2019)

Todos os servidores instanciados para processamento de informações e serviços da empresa foram configurados na zona denominada pela Amazon de *SA-EAST-1* (*South America East* 1) que compreende as instalações e infraestrutura localizados no estado de São Paulo, caso a região determinada não fosse atendida por um serviço em específico, o mesmo era redirecionado para US-EAST-1 (*United States East* 1), região do Norte da Virginia nos Estados Unidos da América. A escolha da região se deu pelo fato de grande parte da origem dos dados requisitados serem oriundos da região sudeste e sul do Brasil, e como consequência da menor distância, uma redução considerável nos tempos de resposta entre as aplicações e os servidores foi possibilitada, reduzindo assim o custo com *hardwares* mais robustos. Para atender as demandas dos serviços executados pela empresa, diversas instâncias de diferentes configurações de *hardware* foram lançadas. Apesar de ser possível executar todos os serviços utilizados pela empresa em uma única instância, decidiu-se pela subdivisão dos serviços em *hardwares* diferentes com o objetivo de avaliar a demanda operacional e de recursos de cada

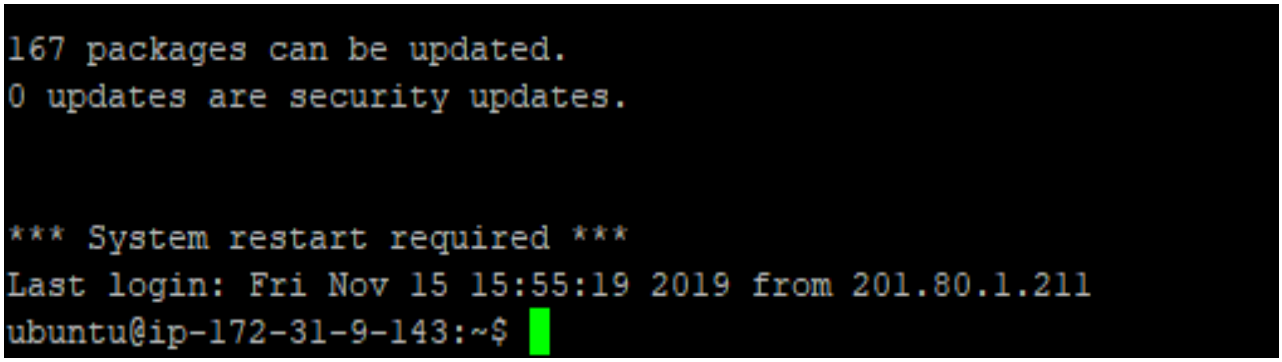
serviço, com o propósito de posteriormente ser possível dimensionar de forma mais eficiente o *hardware* em um servidor com maior recurso computacional.

A subdivisão dos serviços da empresa resultou em 4 instâncias, sendo elas alocadas da seguinte forma:

- Servidor BM-*Backend*, denominado *t2.micro* pela Amazon, é responsável pelo processamento do *backend* e relacionados, possui 1GB de memória RAM, processador Intel Xeon Family de 2.5GHz e 1 núcleo de processamento, 30GB de espaço de armazenamento do tipo SSD, e sistema operacional Ubuntu Server 18.04 LTS modificado pela Amazon;
- O servidor BM-AdminPanel responsável pela hospedagem do painel administrativo e o servidor BM-pwaServer responsável pelo site principal e a aplicação PWA da empresa, ambos denominados t3.micro pela Amazon, possuem o mesmo *hardware* composto de 1GB de memória RAM, processador Intel Skylake P-8175 de 2.5GHz e 2 núcleos de processamento, 30GB de espaço de armazenamento do tipo SSD, e sistema operacional Ubuntu Server 18.04 LTS modificado pela Amazon;
- Servidor BM-MailServer denominado *t3.small*, é responsável pela plataforma de e-mail da empresa e toda sua interface, possui 2GB de memória RAM, processador Intel Skylake P-8175 de 2.5GHz e 1 núcleo de processamento, 30GB de espaço de armazenamento do tipo SSD, e sistema operacional Ubuntu Server 18.04 LTS modificado pela Amazon.

A versão de sistema operacional utilizada não possui interface gráfica, com o objetivo de reduzir o gasto desnecessário de recursos computacionais, ao invés disto é fornecido uma *Command Line Interface* (CLI) como exibida na Figura 3.3.

Figura 3.3 – Interface inicial do servidor.



Fonte: Do autor (2019)

Outro serviço contratado da Amazon é denominado *Route 53*, que fez o gerenciamento de DNS (*Domain Name Server*) e rotas necessário para utilização dos domínios registrados em nome da empresa. A configuração de rotas e DNS permite que a requisição feita por meio de um endereço URL seja encaminhada para o servidor especifico. Este deve processar aquela informação e responder ao pedido. Um exemplo seria a requisição feita para o acesso ao site principal da empresa ao entrar no seguinte endereço da Beer Mine: (www.beermine.com.br). Como dito anteriormente, o site principal está hospedado no servidor BM-pwaServer. Sendo assim, as requisições devem ser encaminhadas ao endereço de IP do servidor em questão, o que não pode acontecer ao se acessar por exemplo o painel administrativo no endereço da Beer Mine (www.admin.beermine.com.br), onde a requisição deve ser repassada ao servidor BM-AdminPanel.

O Amazon *Simple Storage Service* ou Amazon S3, que fornece o armazenamento de objetos por meio de uma estrutura escalável, com gerenciamento de acesso e versionamento dos mesmos, é o último serviço contratado pela empresa do pacote fornecido pela Amazon. O S3 é utilizado como *bucket* para armazenamento e gestão dos arquivos de *firmware* dos *hardwares* utilizados pelos sistemas embarcados. Ele permite que os mesmos recebam as atualizações sempre que necessário.

3.1.2. Sistemas embarcados e IOT

Com o intuito de realizar as operações necessárias para funcionamento da chopeira e comunicação com a plataforma desenvolvida, um sistema embarcado foi especificamente projetado. O sistema embarcado conta um microcontrolador ESP-WROOM-32, um sensor de vazão de efeito Hall modelo YF-S301 para coleta do fluxo e quantidade de bebida que foi servida durante a operação de envase da bebida, uma válvula solenoide com bobina 220 Volts modelo EVR 6 da marca Danfoss que atua impedindo ou permitindo a passagem do liquido pela tubulação, e um servo motor modelo JX PDI-6221MG com torque máximo de 20,32 Kg.cm que atua no controle do posicionamento do suporte do copo durante o envase. Todo o sistema é alimentado utilizando uma fonte chaveada bivolt com capacidade de saída de 5 Volts e 5 Amperes.

Uma placa de circuito impresso (PCI), exibida na Figura 3.4, foi produzida especificamente para este projeto, contando com LEDs de identificação de energização da placa, acionamento da válvula solenoide e modo de configuração de rede. O modo de configuração é controlado através de uma chave alavanca de duas posições. Este modo é

utilizado quando se deseja alterar as configurações de rede WiFi na qual o sistema embarcado se conectará.

Figura 3.4 – Projeto da PCI.

Fonte: Do autor (2019)

3.1.3. Interfaces microprocessadas

Para servir como principal meio de interação entre o usuário e o sistema, foi escolhido um *tablet* modelo *Voyager* III (Figura 3.5), fabricado pela RCA, que conta com 16GB de espaço de armazenamento, 1 GB de memória RAM, processador Intel *Atom Quad Core* e *display* de 7 polegadas.

Figura 3.5 – *Tablet* Voyager III

Fonte: Notebookreview (2017)

Outro dispositivo microprocessado utilizado como interface foi uma máquina de cartões de crédito para realização de pagamentos (Figura 3.6). Devido a motivos financeiros, escolheu-se o modelo fabricado pela PagSeguro, podendo ser a máquina denominada Moderninha Pro, ou Moderninha *Plus*, ambas possuem conexão Bluetooth, 3G, e aceitam cartões de *chip*, tarja e NFC (sem contato) de diversas bandeiras.

Figura 3.6 – Moderninha Plus

Fonte: PagSeguro (2019)

3.2. Métodos

Para viabilizar a execução e processamento das informações necessárias para a automação das vendas de bebidas na chopeira, os seguintes métodos se fazem necessários: os métodos que ditam o correto funcionamento e a lógica para funcionamento do sistema (*backend* e banco de dados), métodos que fazem o controle do sistema embarcado, e os métodos responsáveis pela interação com usuário e seus colaboradores (*frontend*). Estes métodos serão detalhados nas subseções a seguir, sendo de suma importância ressaltar que os métodos citados foram desenvolvidos juntamente com colaboradores de outras empresas.

3.2.1. Banco de dados

É valido iniciar a abordagem técnica dos métodos e regras do sistema e mencionar inicialmente o banco de dados. O banco de dados possibilita manipular informações e dados que serão utilizados por todos os outros métodos e serviços do sistema. Para dar sequência a

estes procedimentos e gerenciar estas informações optou-se por utilizar o MariaDB, que consiste em um *fork* do MySQL, ou seja, um projeto independente baseado no já existente. O MariaDB é um banco de dados relacional, que consiste em um agrupamento de informações através de tabelas que representam objetos ou funções de um sistema e que podem se relacionar ou não entre si. Nas tabelas, os dados são organizados de maneira lógica. Elas são representadas em um formato de colunas e linhas, onde cada linha representa um registro daquele objeto e cada coluna representa uma característica ou informação. Como exemplo, a Figura 3.7 representa a tabela denominada *hardwares* do banco de dados que armazena todas as placas do sistema embarcado registrado no sistema. É possível ver que para cada registro as colunas armazenam informações como: id – código identificador do registro; *status* – situação ou estado do sistema embarcado (desativada, ativada, bloqueada, danificada); *flowsensor* – tipo de sensor que está sendo utilizado; restart – campo que armazena a demanda da placa reiniciar devido a uma requisição remota; *board_version* – armazena a versão da placa; *firmware_version* – armazena versão do *firmware* sendo executado na placa; *flag_version* – campo que define se será instalado um *firmware* de produção, testes ou desenvolvimento na placa; *created_at* – data de criação do registro; *updated_at* – armazena última atualização do registro; ip – armazena o endereço de rede que a placa está conectando; *id_taps_fk* – chave estrangeira que com qual torneira indica está placa está se relacionando; *id_users_fk* – chave estrangeira que indica para qual usuário a placa está registrada.

Figura 3.7 – Tabela *hardwares* do banco de dados.

id	unique	status	flowsensor	restart	board_version	firmware_version	flag_version	created_at	updated_at	ip	id_taps_fk	id_users_fk
1	30:AE:A4:73:07:C4	0	1	0	10	18	1	NULL	2019-10-25 13:54:36	0.0.0.0	2	4
2	b4:e6:2d:97:b6:bd	0	1	0	10	6	1	NULL	2019-10-23 18:36:28	0.0.0.0	1	4
3	B4:E6:2D:96:26:FD	0	1	0	10	18	1	2019-09-27 21:23:36	2019-10-23 18:36:36	0.0.0.0	3	4
4	B4:E6:2D:97:B7:29	1	1	0	10	23	1	2019-10-08 13:08:19	2019-10-30 14:46:24	0.0.0.0	4	4
5	B4:E6:2D:96:CE:45	0	1	0	10	20	1	2019-10-09 18:44:54	2019-10-25 14:16:00	0.0.0.0	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

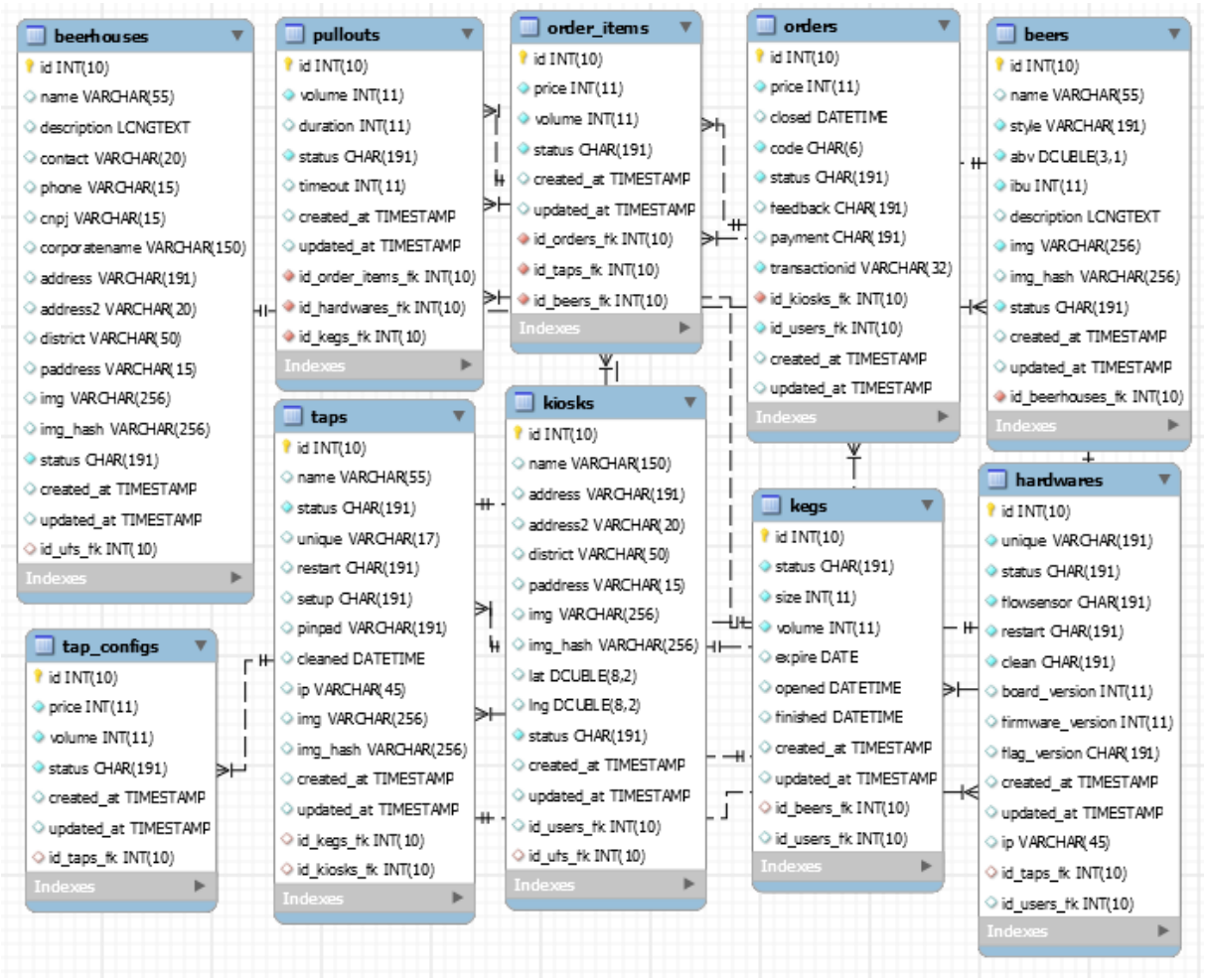
Fonte: Do autor (2019).

As informações contidas nas tabelas variam de acordo com o objeto e a lógica do sistema. Além disto, em cada campo deve ser especificado o tipo de dado que será armazenado. Os mais habituais são: *string*, *char*, *int*, *float*, *datetime* (representações de tempo e datas em geral). O banco de dados utilizado nos sistemas desenvolvidos conta com cerca de 30 tabelas, além das tabelas que relacionam objetos como cervejas, cervejarias, pedidos, *hardwares*,

fornecedores, usuários, existem outras tabelas que representam *tokens* de autenticação, erros do sistema, *resets* de *password*, *logins* executados e diversos outros.

As demandas computacionais destes serviços têm um custo razoavelmente baixo, principalmente para tabelas com menos de 2GB de informações. Os serviços têm seu desempenho é principalmente determinado pela velocidade de leitura e acesso dos discos, memorias e a capacidade que o processador tem de processar essas buscas. Desta forma, os processos são alocados no servidor BM-*Backend* que também processa as requisições solicitadas no *backend*. Logo, como compartilham os mesmos recursos de *hardware* as trocas de informações acabam sendo ligeiramente mais rápidas em comparação a alocações em *hardwares* diferentes, mesmo que na mesma rede. A Figura 3.8 mostra as principais tabelas do banco de dados e suas relações.

Figura 3.8 – Tabelas do banco de dados e suas relações.



Fonte: Do autor (2019).

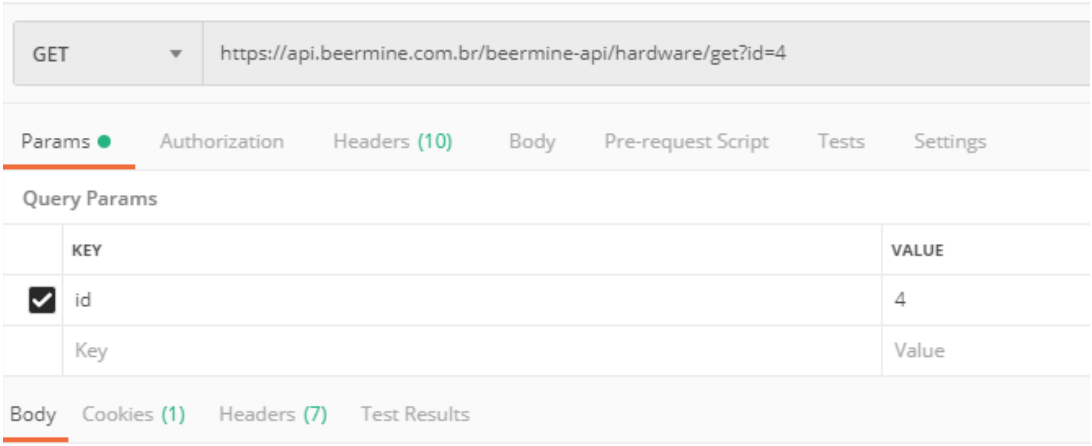
Algumas medidas de segurança são de suma importância para garantir a integridade dos dados e caso algum incidente ocorra, para restaurar uma versão atualizada dos dados perdendo a menor quantidade de informações. Posto isso, o acesso ao banco de dados só é permitido dentro da própria rede local. Além de usuários e senhas autenticados, é necessário que seja fornecida uma chave criptografada de 2048 bits. Desta forma, reduzem-se as chances de qualquer tentativa de acesso aos dados e informações ali contidas. Para garantir a integridade dos dados, serviços de *backup* são executados em intervalos de 1h, os quais salvam as alterações feitas em relação ao *backup* da hora anterior, outro serviço com intervalo de 24h salva todo o banco de dados, as copias e os *backups* são armazenados em um *bucket* do S3 com redundância dos dados. Vale ressaltar que estas medidas não protegem ou excluem totalmente o acesso indevido as informações e o risco de perda dos dados. Entretanto servem como medidas iniciais que aumentam a segurança e auxiliam na manutenção.

3.2.2. *Backend*

O *backend* é parte fundamental das operações realizadas por todas as aplicações. É ele quem vai ditar as regras, logicas e funcionamento geral do sistema. Todas as informações que são pesquisadas, inseridas, atualizadas ou deletadas são processadas por meio do *backend* e as funções nele implementadas. Desta forma, além de fornecer total suporte ao *frontend* e aos dispositivos embarcados no que diz respeito as operações que cada sistema deseja processar ou receber, o mesmo deve fornecer o máximo de segurança e controle de acesso de forma que somente usuários autorizados consigam acessar as informações.

O *backend* foi desenvolvido utilizando o *framework* Laravel, que utiliza o PHP como linguagem de programação. Os serviços necessários para execução do *backend* foram implementados no servidor BM-*Backend*. Para execução do mesmo foi instalado e configurado o Nginx, que é um servidor web, capaz de receber e manusear requisições HTTP (*Hypertext Transfer Protocol*). As requisições no *backend* são realizadas por meio de acessos a endereços URL convencionais. Os acessos podem ser realizados por meio dos dispositivos embarcados, móveis, navegadores, aplicativos e inclusive componentes do sistema. O protocolo HTTP define um conjunto de métodos de requisição. Cada métodos é responsável por indicar a ação que deve ser realizada. Os métodos mais utilizados nos sistemas são os verbos GET, que é a mais usual de todas e normalmente utilizada quando se quer obter um determinado recurso, e o verbo POST, utilizado quando se deseja executar alguma ação dentro do servidor, podendo ser a criação de um novo recurso ou atualização de um já existente.

Figura 3.9 – Requisição feita ao servidor.



Fonte: Do autor (2019).

Além das diferenças nas ações, o método GET se caracteriza por enviar os parâmetros da requisição junto ao endereço URL como mostra a Figura 3.9, feito utilizando o *software Postman*, foi feita a requisição ao endereço https://api.beermine.com.br/beermine-api/*hardware*/get?id=4, visando obter informações sobre o *hardware* com código identificador 4. Já no método POST, os parâmetros da requisição são passados no cabeçalho da requisição, não sendo possível sua identificação por meio do endereço URL.

Outra característica das requisições utilizando o protocolo HTTP são os códigos de *status* das respostas, que têm como objetivo auxiliar na identificação da operação e possíveis problemas. Os códigos das respostas são gerados pelo servidor e são caracterizados em 5 grupos, sendo eles: 1XX, respostas de informação, 2XX, respostas de sucesso, 3XX, redirecionamentos, 4XX, possíveis erros relacionados ao cliente, 5XX, possíveis erros relacionados ao servidor. Além disto, algumas respostas dentro de cada classe são, por via de regras, padronizadas como os códigos 200, indicando que a requisição foi bem-sucedida, 400 indicando que a requisição não é válida e possivelmente possui parâmetros inválidos, 401 indicando que o usuário não possui acesso autorizado e o famoso 404 indicando que o recurso não foi encontrado no servidor. Junto com o código de *status* da requisição, caso a mesma seja válida (*status* 200), é enviada a resposta da requisição em um formato conhecido como JSON (*JavaScript Object Notation*), que consiste em um formato padrão para troca de dados de forma simples, compacta e rápida.

Figura 3.10 – Resposta em formato JSON.

```
{
    "id": 4,
    "tap_id": 4,
    "unique": "B4:E6:2D:97:B7:29",
    "tap_name": "Alvaro's Growler Station",
    "updated_at": "2019-10-30 14:46:24",
    "flowsensor": "1",
    "board_version": 10,
    "firmware_version": 23,
    "flag_version": "1",
    "status": "1"
}
```

Fonte: Do autor (2019).

A Figura 3.10 mostra o formato de uma resposta em formato JSON, o qual é formada por um texto com diversos itens, e cada item por uma chave e um valor. Para todas as funções e chamadas definidas no *backend*, as respostas, quando no caso de um único objeto retornado, possuem o formato acima. Caso a resposta seja um conjunto de vários objetos, o servidor retorna uma matriz de vários objetos.

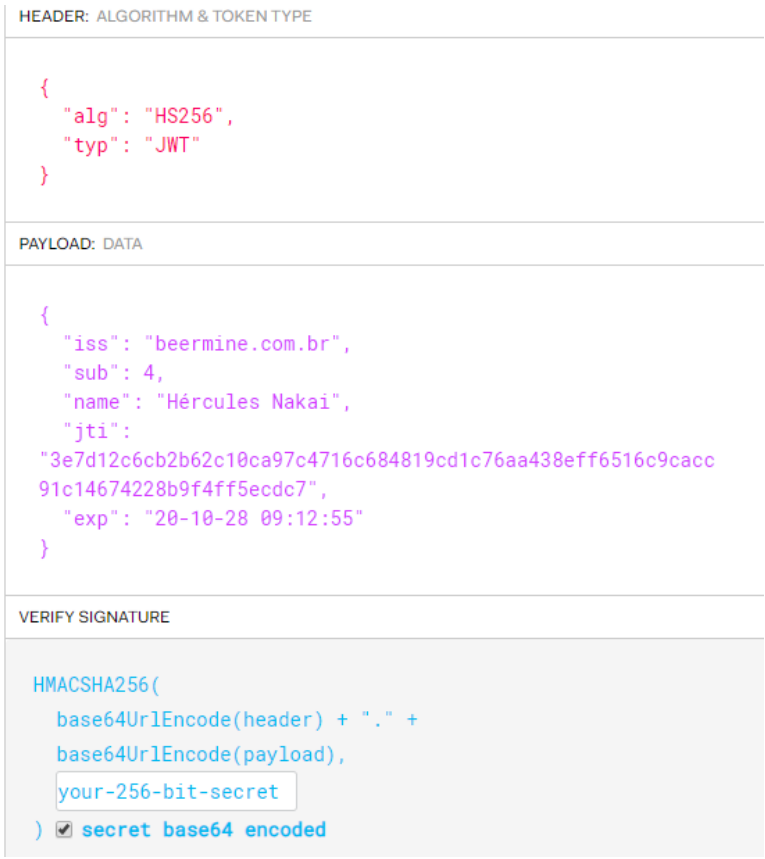
Sendo o *backend* o grande responsável pelo fluxo e manipulação de dados e informações do sistema, e em se tratando de requisições HTTP, onde basicamente qualquer pessoa pode acessar os endereços URL e realizar requisições para o servidor, é de suma importância que mecanismos de segurança sejam implementados de forma a permitir que somente aqueles usuários autenticados ou dispositivos autorizados tenham acesso ao conteúdo e as informações solicitadas. Desta forma foi desenvolvido um mecanismo de autenticação baseado em *tokens* assinados digitalmente proveniente do JWT (*Json Web Token*). Estes *tokens* permitem que o tráfego de dados entre cliente e servidor seja realizado de forma segura e no nível correto de acesso. O *token* gerado pelo servidor tem o formato ilustrado na Figura 3.11 e é composto de 3 partes, sendo elas: o *Header*, onde são armazenadas informações sobre o tipo de *token* e o algoritmo usado em sua criptografia; o *Payload*, que contém as informações sobre o usuário autenticado; e a *Signature*, que contém o conteúdo criptografado das partes do *Header* e *Payload*, junto de uma chave de autenticação ou certificado.

Figura 3.11 – *Token* JWT codificado.

Fonte: Do autor (2019).

Uma vez decodificado o *token*, este possui o conteúdo ilustrado na Figura 3.12, onde o *Header* indica que o algoritmo utilizado na encriptação é o HS256, e o *Payload* armazena informações sobre o domínio em que o *token* foi gerado, o nome do usuário, código identificador do usuário, código identificador do *token* e data de expiração do mesmo.

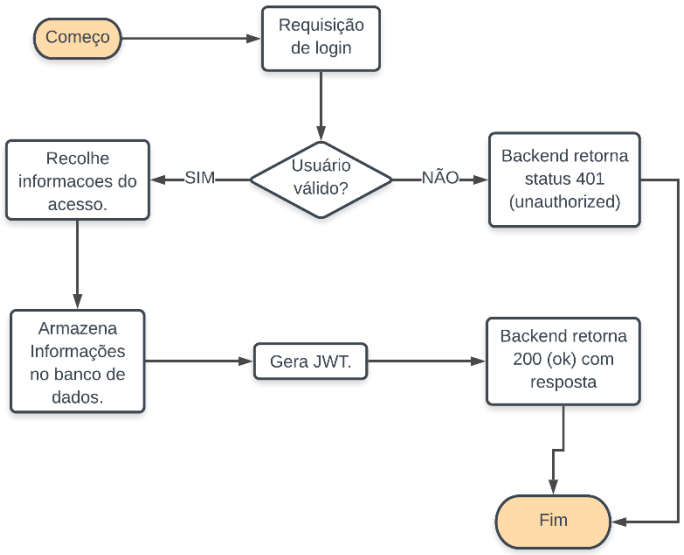
Figura 3.12 – *Token* JWT decodificado.



Fonte: Do autor (2019).

O fluxo de autenticação se inicia assim que usuário solicita a requisição de *login* no sistema, utilizando o método POST. O servidor verifica a autenticidade dos dados e uma vez corretas, são geradas informações a respeito do usuário e o *login* efetuado. Essas informações são armazenadas no banco de dados e utilizadas para gerar o *token* que é retornado como resposta tanto no corpo da resposta em formato JSON, como no cabeçalho da mensagem por meio do campo *authorization.* A Figura 3.13 resume as etapas dos processos.

Figura 3.13 – Fluxograma processo de *login*.

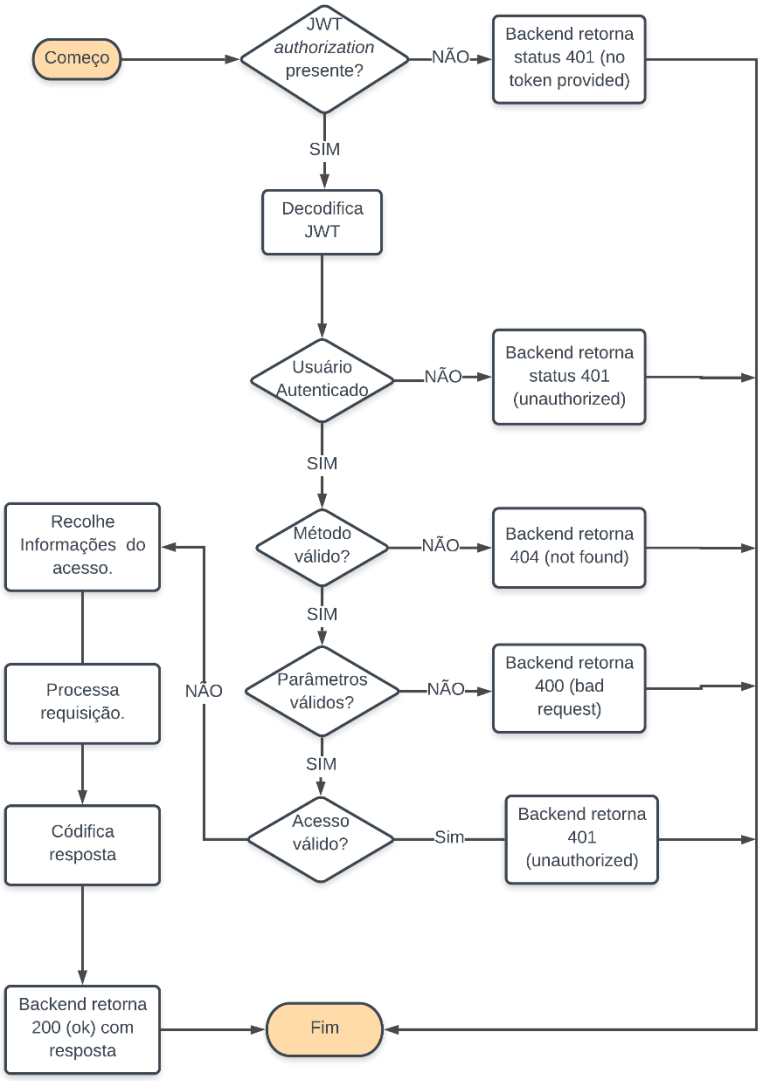


Fonte: Do autor (2019).

De posse do *token*, sendo um usuário ou dispositivo IOT via sistema ou aplicação *frontend*, todas as funções e chamadas registradas no *backend* fazem a verificação do *token* e autenticidade das informações, sendo obrigatório o envio das mesmas por meio do mesmo campo *authorization* presente no cabeçalho da requisição HTTP. Assim que recebe uma requisição, o *backend* verifica se existe um *token* presente no campo. Caso não exista, o mesmo retorna a mensagem "*no token provided*" junto do *status* 401 (*unauthorized*) na resposta. O mesmo *status* é retornado caso não seja possível validar as informações fornecidas no *token* com as informações presentes no banco de dados. Uma vez que o *token* é valido, verificações padrões são processadas antes de qualquer resposta ser retornada, a primeira delas se faz no tipo do método, onde o tipo da requisição deve ser o mesmo registrado no *backend*, caso contrário o usuário será redirecionado e terá uma resposta com o código 404 (*not found*). Se o tipo do método coincidir com o registrado, a requisição prossegue para o segundo mecanismo

de segurança, sendo este a validação dos parâmetros necessários para a requisição, onde somente os dados registrados no *backend* são considerados e levados em conta para posterior utilização pela função a ser processada, sendo todo o restante descartado. Neste caso, se os tipos de dados forem incompatíveis com os declarados ou faltantes, a resposta será uma mensagem de erro com o *status* 400 (*bad request*). A última etapa de segurança, comum a todas as funções implementadas no *backend*, é a verificação de permissão do nível e tipo de acesso do usuário, onde as informações presentes no *token* são mais uma vez verificadas e comparadas com os valores presentes no banco de dados. Caso o usuário possua um nível de acesso menor ou diferente do que o necessário para a execução da requisição, o sistema retornará o mesmo *status* 401 (*unauthorized)*. O fluxograma da lógica é apresentado na Figura 3.14.

Figura 3.14 – Fluxograma para validação do *Token* JWT.



Fonte: Do autor (2019).

Esta última camada de segurança garante, ao sistema, que usuários sem permissão fiquem impedidos de acessar qualquer tipo de informação e dados que seriam permitidos somente a níveis elevados do sistema. Ele também possibilita a integração com outros sistemas de terceiros e diferentes tipos de sistemas embarcados, especificando qual tipo de conteúdo o mesmo pode acessar, como, por exemplo; para o painel administrativo, são registrados dois níveis de acesso sendo, os administrados e clientes. Para uma mesma requisição que deveria listar todos os *hardwares* registrados no sistema, o usuário com nível de administrador teria todos os *hardwares* como resposta, independentemente para qual usuário o mesmo estivesse registrado, enquanto o usuário com nível de cliente receberia somente os *hardwares* registrados para o seu usuário. O mesmo conceito pode ser exemplificado em uma abordagem para os dispositivos IOT, onde são registrados três tipos de acesso no sistema, a saber: os sistemas embarcados proprietários, os *tablets* que funcionam como *frontend* para o usuário, e os dispositivos de integração com *softwares* de terceiros. Como cada um somente necessita e manuseia informações especificas no *backend*, se torna lógico o bloqueio a determinados acessos.

O *backend* desta forma realiza o controle e gerenciamento das requisições implementadas para prover o suporte, acesso aos dados e informações necessárias para o correto funcionamento dos sistemas e plataformas. São mais de 100 chamadas (rotas) externas disponíveis para aplicações e usuários fazerem a requisição dos dados, e mais de 50 chamadas internas onde somente o próprio *backend* pode realizar as operações. Vale ressaltar que o *backend* obriga que todas as trocas de dados sejam realizadas utilizando o protocolo HTTPS que fornece uma camada adicional de segurança, na qual as comunicações são criptografadas utilizando um certificado digital que é emitido por uma empresa certificada e que deve ser renovado periodicamente.

3.2.3. *Frontend*

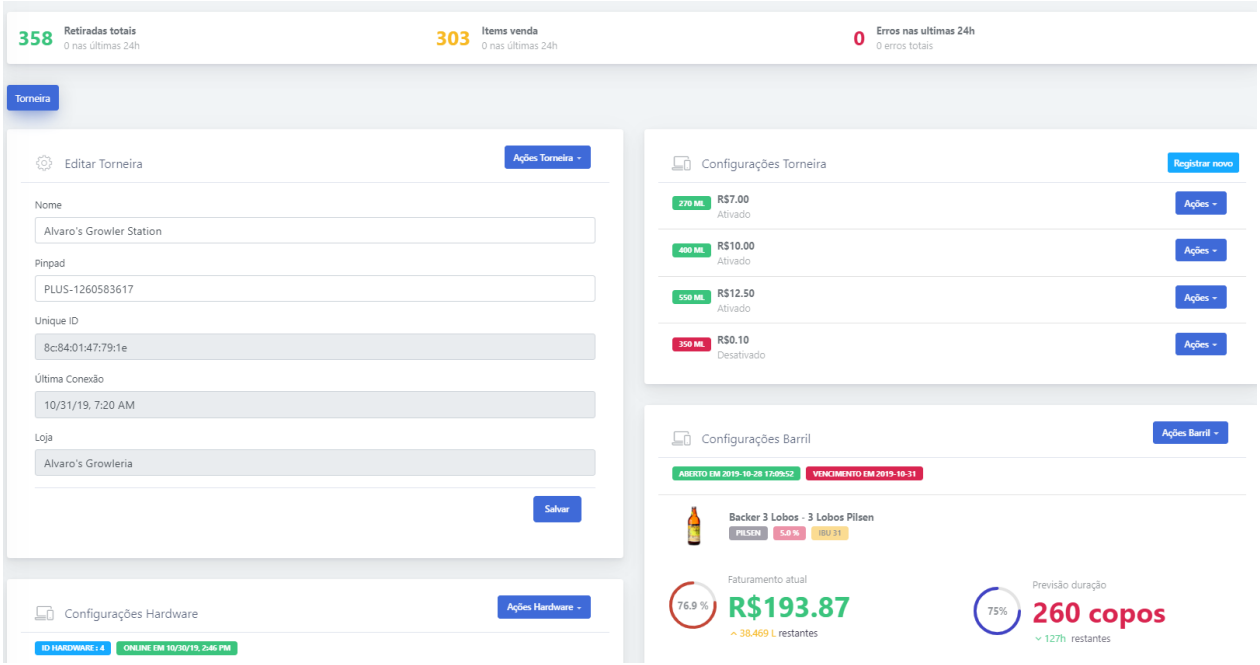
O *frontend* é a parte do sistema que permite que sejam realizadas as operações por meio de interfaces gráficas, sendo compostos por dois subsistemas; o painel administrativo e o *tablet* onde são realizadas as compras. Os aspectos mais importantes de ambos os sistemas estão relacionados a usabilidade e experiência, buscando simplificar e facilitar a comunicação entre os usuários e máquinas. No caso do painel administrativo, isto implica auxiliar no gerenciamento dos colaboradores das empresas que contratam o sistema, enquanto no *tablet,* o foco é o público consumidor das bebidas.

Para o desenvolvimento do painel administrativo, procurou-se implementar um sistema que funcionasse em uma ampla quantidade de dispositivos e apresentasse uma interface amigável e autoexplicativa. Sendo assim, foi utilizado o Angular como *framework* e plataforma de desenvolvimento, em conjunto com um *template* de interface gráfica que melhor se ajusta as funções de um painel de gerenciamento. O template permitiu que o tempo e recursos gastos com o desenvolvimento da interface fossem reduzidos de maneira significativa, uma vez que poucas implementações precisaram ser feitas.

O painel administrativo funciona como um site comum, onde o seu acesso pode ser realizado por meio de qualquer dispositivo móvel ou computador com acesso à internet utilizando um navegador, bastando para isso que o seguinte endereço URL seja acessado: https://admin.beermine.com.br. O acesso ao site só é permitido a usuários previamente registrados, onde deve ser inserido um usuário e senha para realização do *login* ao painel, isto garante que as informações exibidas sejam somente referentes aos *hardwares* e máquinas cadastradas para o seu usuário. Caso o perfil autenticado seja referente a um administrador, ao invés de um cliente, algumas informações e funcionalidades extras são exibidas de forma que os colaboradores da empresa (Beer Mine) possam realizar possíveis manutenções e gerenciamento dos clientes. Uma vez realizada a autenticação no sistema, a página exibida concentra as principais informações no centro da tela e exibe um menu lateral que permite que seja realizada a navegação entre os diversos conteúdos de gerenciamento de torneiras, *hardwares*, pedidos e outros.

Uma vez que definida a lógica e regras de funcionamento do sistema no *backend*, fica sob responsabilidade do *frontend* exibir corretamente as informações e garantir que as ações executadas pelos usuários sejam corretamente encaminhadas ao *backend*. De forma a exemplificar, a Figura 3.15 mostra a interface de gerenciamento das torneiras. Nela é possível verificar as informações das torneiras em si, configurações de opções das torneiras, o barril que está configurado nas mesmas, a quantidade de volume restante no barril em litros, o seu faturamento, a previsão de duração em horas, quantidade de copos restantes, e o *hardware* responsável por fazer o controle e gerenciamento da mesma.

Figura 3.15 – Interface de edição de torneiras do painel administrativo.



Fonte: Do autor (2019).

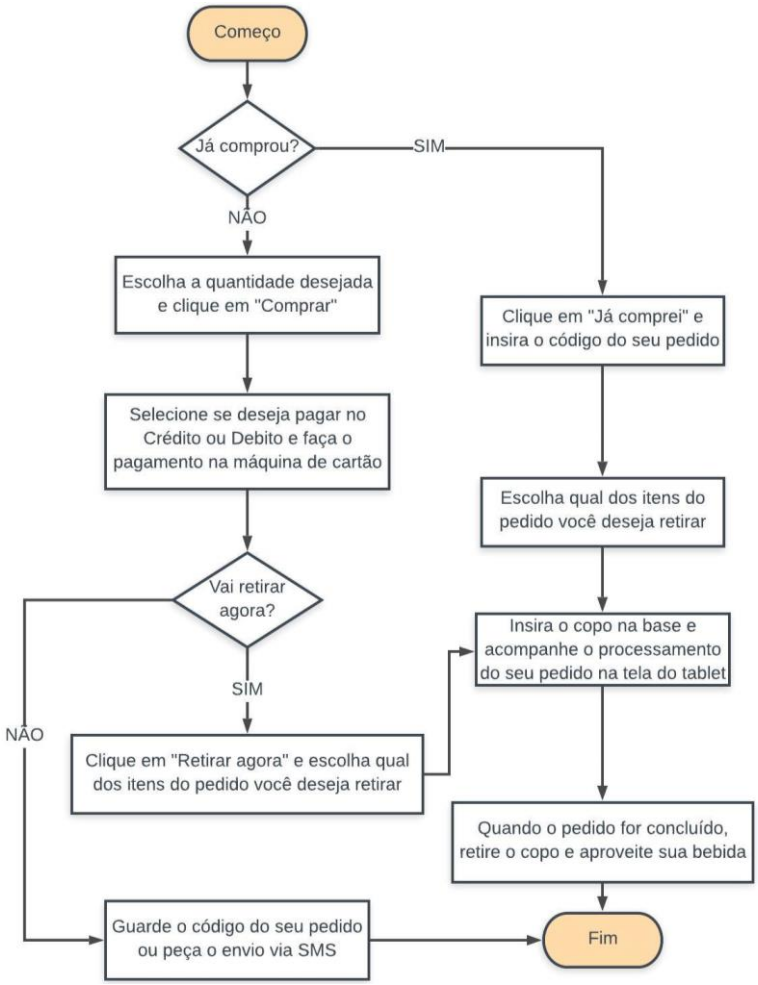
No sistema, cada página de configuração possui suas próprias ações. Além disso, a mesma fornece o monitoramento, em tempo real, com *logs* de atividades, *status* e operações realizadas por todos os dispositivos. `

No desenvolvimento do sistema do *tablet*, uma vez que o sistema operacional do mesmo é o Android na versão 8.0+, foi decidido que seria desenvolvido uma aplicação para funcionar neste sistema, o que permitiu aproveitar os outros recursos de *hardware* do *tablet*, como *bluetooth*, e assim auxiliar nos protocolos de comunicação com os meios de pagamento, no caso as maquininhas de cartão. Utilizando o Android Studio como plataforma de desenvolvimento, e o Java como linguagem de programação, priorizou-se uma aplicação com interface que permitisse ao usuário realizar a operação de compra no menor tempo ou menor número de etapas possíveis, visando tornar o processo simples e reduzir a formação de filas.

Com o propósito de ser um sistema de autoatendimento de uma *vending machine*, e tornar a aplicação mais autônoma, diversas configurações do sistema foram alteradas e implementadas durante a execução do *software*, como desabilitar as teclas físicas, impedir o usuário de alterar ou fechar a aplicação, desabilitar o áudio, configuração do brilho da tela sempre no máximo, e em caso de fechamento involuntário, a aplicação se reinicia automaticamente. Todas as etapas para realização da compra executadas pelo usuário podem ser observadas na Figura 3.16, e a troca de informações entre os componentes na Figura 3.17.

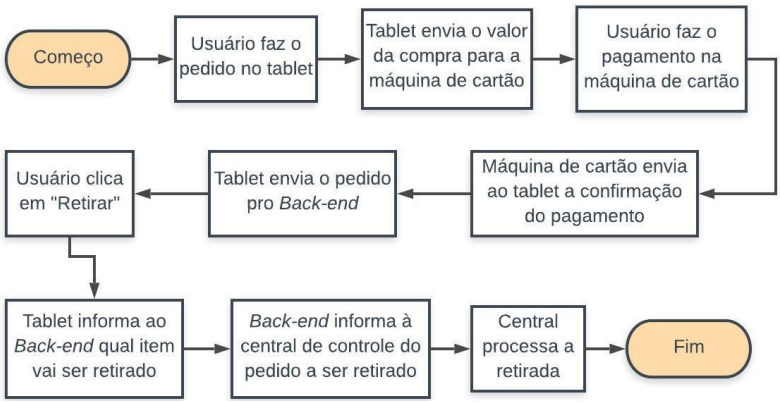
Todo o processo compra e retirada de chope para um copo de 300ml demora aproximadamente 50 segundos. Caso seja feita somente a retirada o processo reduz para cerca de 20 segundos.

Figura 3.16 – Fluxograma para compra das ações do usuário.



Fonte: Beer Mine (MENDONÇA, 2019).

Figura 3.17 – Fluxograma para compra do dispositivo.



Fonte: Beer Mine (MENDONÇA, 2019).

O painel administrativo fornece gerenciamento completo sobre as configurações do *tablet*, sendo possível alterar o tipo de bebida que é servida, valores, reiniciar o *hardware* envolvido e outras funcionalidades.

Toda a comunicação realizada pelo *frontend*, assim como no *backend*, utiliza o protocolo HTTPS, fornecendo uma camada extra de segurança. Vale ressaltar novamente que esta medida de segurança não garante a segurança completa do sistema e sim um adicional que torna o mesmo menos susceptível.

3.2.4. *Firmware*

Um sistema embarcado foi desenvolvido com o objetivo de controlar todas as operações que envolvem o processo de envase do chope. Sendo assim, um circuito elétrico e uma placa de circuito impresso foram desenvolvidas de forma a servir de suporte a todos os componentes necessários para execução das etapas. A confecção das placas foi de responsabilidade da Mazza G-Tec.

Com a PCI confeccionada e os componentes soldados, o *firmware* foi desenvolvido utilizando a IDE do Visual Studio Code utilizando a linguagem de programação C. Além do controle dos componentes que o sistema embarcado têm que realizar, tais como acionamento das válvulas solenoides, leitura do sensor de fluxo, controle de inclinação do copo pelos servo motores, este é de suma importância que o *hardware* em questão mantivesse as informações de funcionamento atualizadas no *backend*. Esta tarefa só é possível uma vez que o ESP-WROOM-32, microcontrolador utilizado, possuí 2 núcleos de processamento, permitindo que se separassem as atividades executadas pelo *firmware* em várias *threads* para cada um dos núcleos, sendo o núcleo 0 responsável pelo controle do *hardware*, e o núcleo 1, encarregado das tarefas e processamento das requisições de internet.

A lógica de funcionamento do sistema embarcado começa com a validação do *hardware* em questão por meio das chaves de acesso e o endereço MAC único do microcontrolador, que é enviado ao *backend* e validado (recebendo assim um *token*). De posse do *token*, o *hardware* começa a verificar se existe alguma solicitação de retirada de pedido. Caso exista uma solicitação de pedido, o mesmo recebe as informações necessárias para processar o pedido, sendo elas o volume do copo, utilizado para comparar com a leitura do sensor e controlar a abertura e fechamento da válvula solenoide, e o tempo limite de duração da retirada, valor este calculado pelo *backend* com base no tempo gasto nas últimas retiradas concluídas. Caso o processo ultrapasse o tempo estipulado pelo *backend*, o sistema entende que ocorreu um

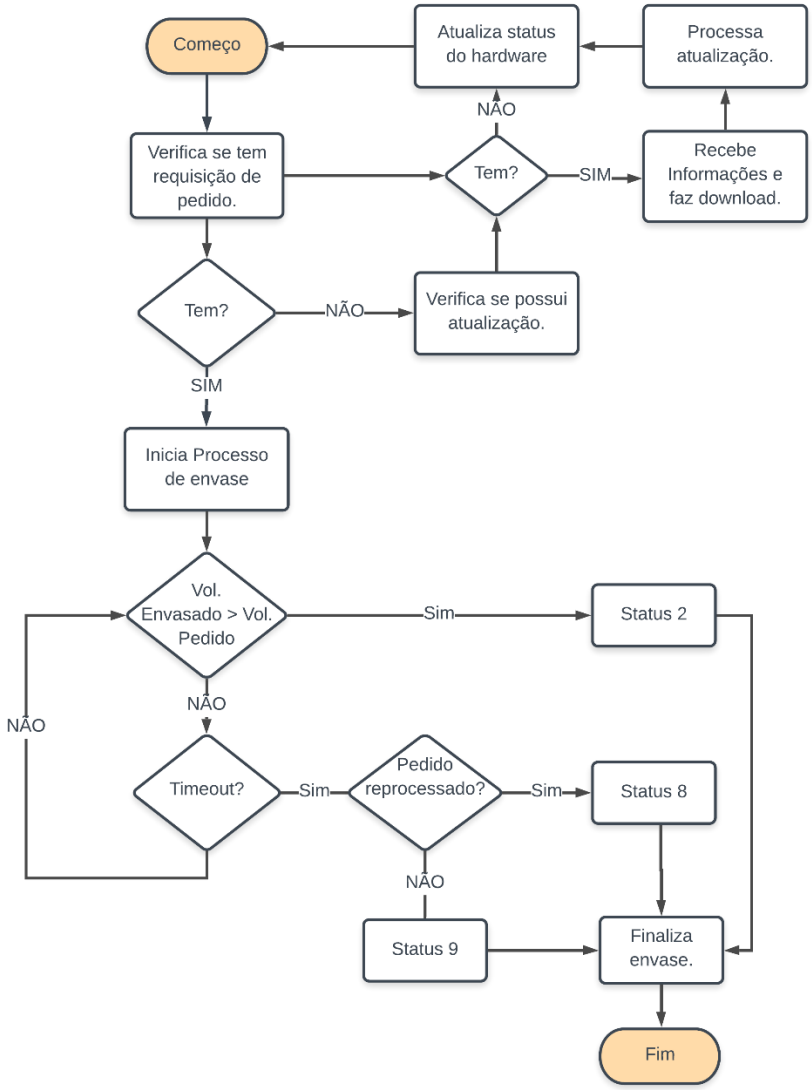
problema com o pedido e tenta processá-lo mais uma vez, caso o erro volte a ocorrer o sistema embarcado informa ao *backend*, que então trata as informações e cria os alertas para os responsáveis averiguarem o problema. Enquanto o núcleo 0 fica responsável pelo processo de envase e controle do *hardware*, simultaneamente o núcleo 1 fica atualizando o *backend* com as informações mais recentes do processo de retirada. No caso do *hardware* não ter nenhuma solicitação de pedido, ele verifica com o *backend* a presença de uma atualização de *firmware* e atualiza o mesmo sobre suas condições no momento.

Uma tabela no banco de dados denominada *pullouts*, faz o controle e gerenciamento dos processos de retirada de pedidos, o *status* é o principal campo desta tabela, representando a condição do pedido e o que o *hardware* deve fazer. Desta forma, assim que o usuário no *tablet* faz a solicitação para retirar o pedido, o *tablet* envia a requisição para o *backend*, que cria um novo registro nesta tabela com o *status* 0. Assim que recebe este registro o sistema embarcado começa a processar e tratar o mesmo, podendo então atribuir os seguintes valores:

- 0 – retirada pendente;
- 1 – retirada em andamento;
- 2 – retirada concluída com sucesso;
- 3: retirada estourou o tempo de limite, mas foi reprocessada e concluída com sucesso;
- 4 – retirada que estourou o tempo limite e já teve seu *status* reprocessado;
- 8 – retirada cancelada por estourar o tempo limite 2 vezes;
- 9 – retirada estourou o tempo limite 1 vez e será reprocessada.

Em uma situação ideal de funcionamento, assim que recebe o pedido, o sistema embarcado atualiza o *status* para 1 durante o procedimento e assim que termina atualiza para 2. Os valores de *status*, 5, 6 e 7 foram reservados para condições futuras de funcionamento. A Figura 3.18 mostra o fluxo do processo de envase.

Figura 3.18 – Fluxo do processo de envase.



Fonte: Do autor (2019).

Como forma de tornar mais fácil a manutenção do sistema e das máquinas comercializadas, foi necessário desenvolver um sistema que permitisse que o *firmware* fosse atualizado de maneira remota sem a presença de uma equipe técnica, seja por alguma melhoria no próprio código do *hardware* ou por necessidade de adaptação a mudanças realizadas no *backend.* Sendo assim, uma tabela no banco de dados gerencia o versionamento dos *firmwares* e as respectivas compatibilidades com as placas. Caso seja identificado a presença de uma atualização, que é informada pelo *backend,* o sistema embarcado recebe todas as informações necessárias para realizar a atualização remota conhecida como *Over The Air* (OTA). Como mencionado anteriormente, o procedimento recebe o auxílio do serviço S3 da Amazon, que armazena os arquivos que são disponibilizados para *download*. Assim que é concluído o *download*, o *hardware* realiza a atualização, armazenando o arquivo em uma parte separada da

memória, efetua as configurações necessárias para gravação do novo código, e em seguida já inicia com a versão atualizada do mesmo.

4. RESULTADOS E DISCUSSÃO

Concluído o desenvolvimento e implementação de todos os sistemas, foi possível realizar a montagem da máquina chegando ao produto final que pode ser visto na Figura 4.1. A máquina, como um todo, foi montada em um sistema de refrigeração da empresa MaxFilter. Foram realizadas algumas poucas adaptações para receber os equipamentos utilizados no projeto. A interface inicial do *tablet*, exibindo as opções “Comprar” e “Já Comprei”, pode ser observada na Figura 4.2. As demais interfaces do sistema se encontram nas Figuras 4.3 a 4.5.

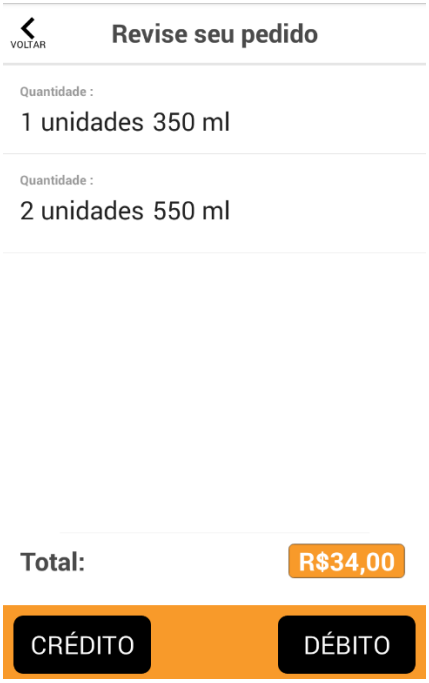
Figura 4.1 – Montagem final da chopeira.

Fonte: Beer Mine (2019)

Figura 4.2 – Interface inicial do aplicativo do *tablet*.

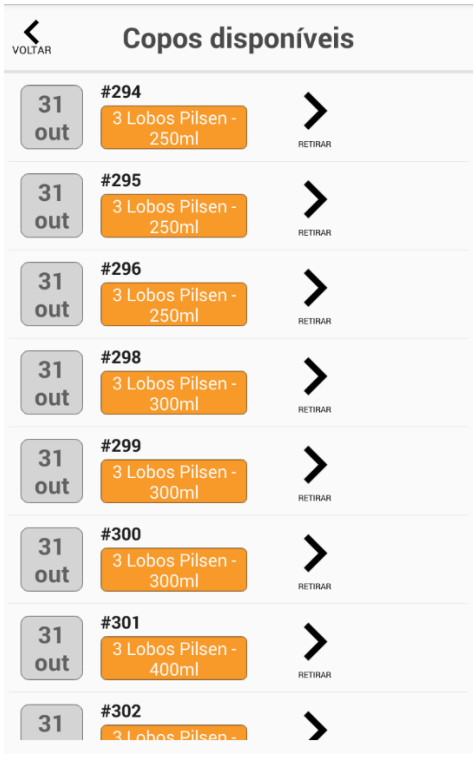
Fonte: Beer Mine (2019)

Figura 4.3 – Interface de seleção de forma de pagamento no aplicativo.



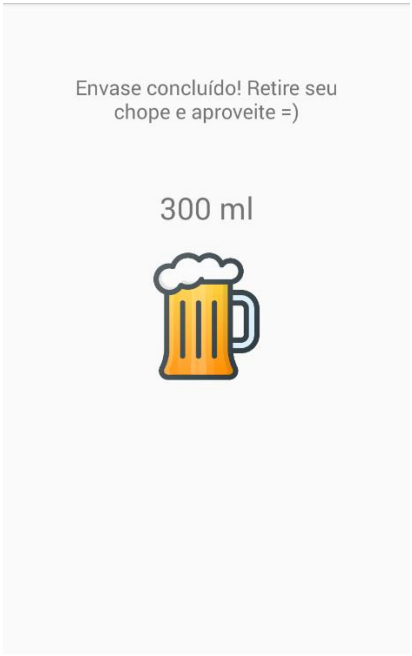
Fonte: Beer Mine (2019)

Figura 4.4 – Interface de copos disponíveis no código inserido.



Fonte: Beer Mine (2019)

Figura 4.5 – Interface de envase concluído.



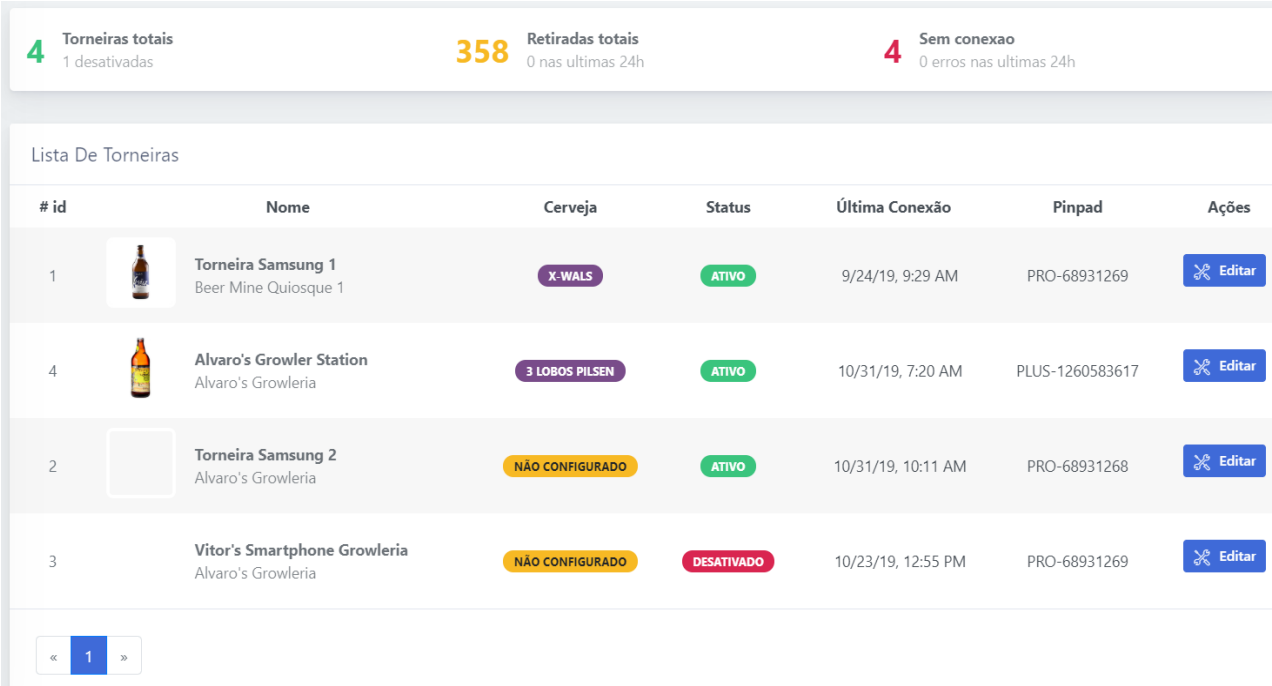
Fonte: Beer Mine (2019)

Inicialmente, com o intuito de validar o funcionamento do sistema, diversos testes foram realizados. Primeiramente, foi feita compra de um único copo de cerveja e selecionada a opção “Retirar Agora”, sendo o envase do copo finalizado com sucesso. Repetidos testes foram efetuados com diferentes quantidades de copos e o sistema se comportou conforme esperado. O *tablet* exibiu as informações de envase de maneira correta, indicando que o sistema embarcado estava fazendo as atualizações e por consequência o *backend* funcionava corretamente. Os procedimentos mecânicos também foram satisfatórios. A máquina inclinou corretamente o copo, imitando o movimento de um garçom ao servir um copo de bebida. Além das condições ideais de funcionamento, diversos testes foram executados em condições não ideais, simulando a falta de internet em todos os componentes, a queda de energia e outros. Quando ocorre uma queda na conexão de internet, o sistema embarcado, caso já esteja realizando o procedimento de envase, conclui o mesmo normalmente uma vez que para controle do *hardware* não depende da conexão, e após conclusão, o mesmo fica aguardando o retorno da conexão para atualizar o *backend* com as informações. O mesmo acontece com o *tablet*. Em caso de interrupção no fornecimento de energia, o *hardware* interrompe o envase (uma vez que a válvula solenoide utilizada é normalmente fechada), e assim que o mesmo retorna ele continua o processo de envase levando em consideração as últimas informações fornecidas antes da queda de energia.

O mecanismo de atualização *Over The Air* (OTA), presente no sistema embarcado também se mostrou eficiente. Após realizadas as configurações no banco de dados e no S3 da Amazon, o *hardware* rapidamente identifica que é necessário realizar o *download* da atualização e realiza todo o procedimento em normalmente não mais do que 60 segundos, tornando uma funcionalidade bastante eficiente e útil.

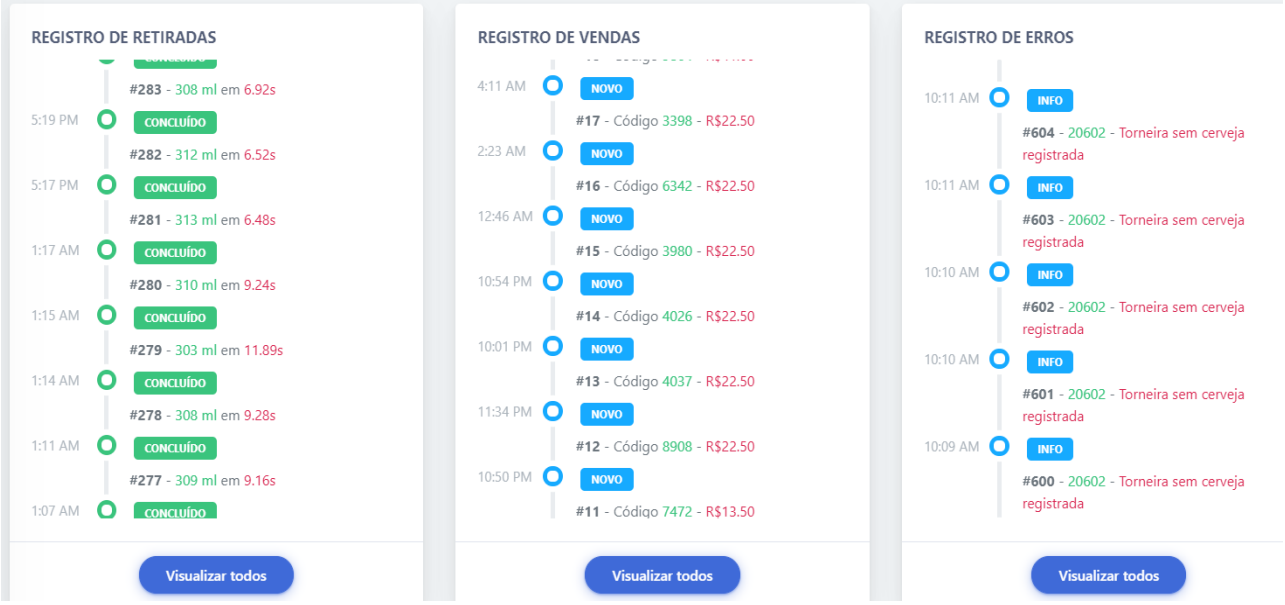
O painel administrativo durante todos os testes permitiu monitorar em tempo real o funcionamento do sistema, notificando sobre as retiradas, vendas, registros, alertas e outros. Além disso, possibilitou que as alterações pudessem ser realizadas de maneira rápidas tanto no *hardware* como no *tablet*. As Figuras de 4.6 a 4.9 mostram algumas interfaces desenvolvidas.

Figura 4.6 - Interface inicial do painel administrativo (Listar torneiras).



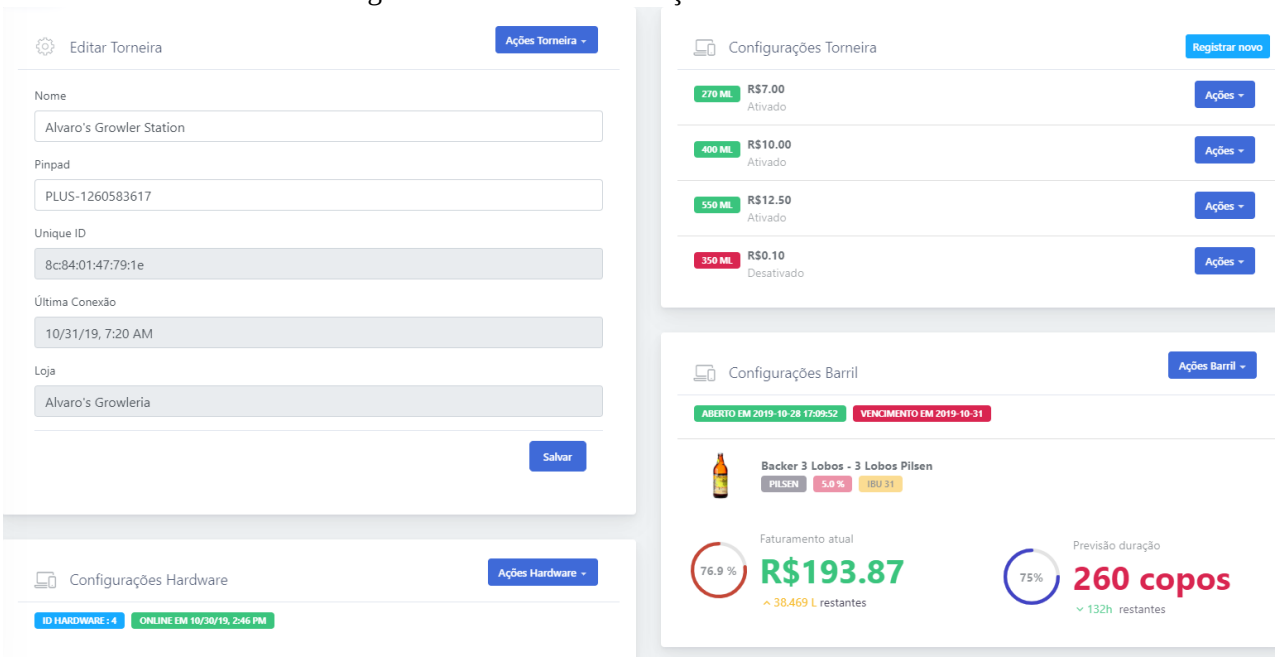
Fonte: Do Autor (2019)

Figura 4.7 - Interface dos registros de vendas e retiradas das torneiras.



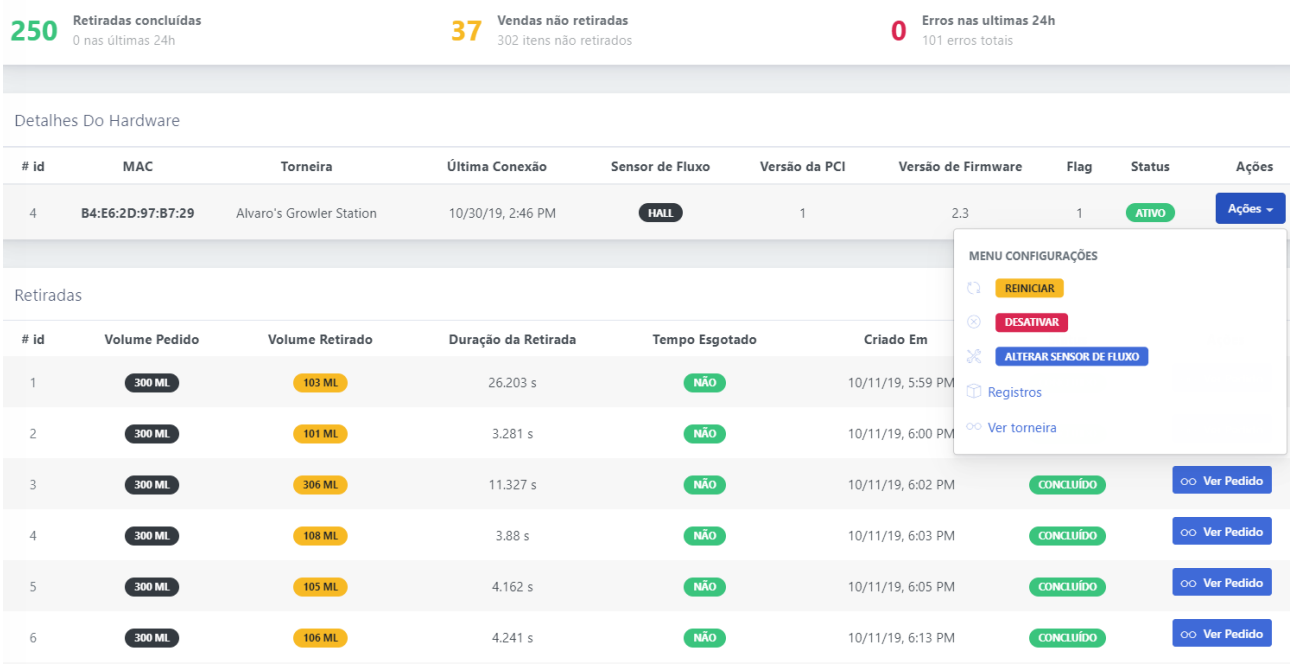
Fonte: Do Autor (2019)

Figura 4.8 - Interface de edição das torneiras.



Fonte: Do Autor (2019)

Figura 4.9 - Tela de edição do sistema embarcado.

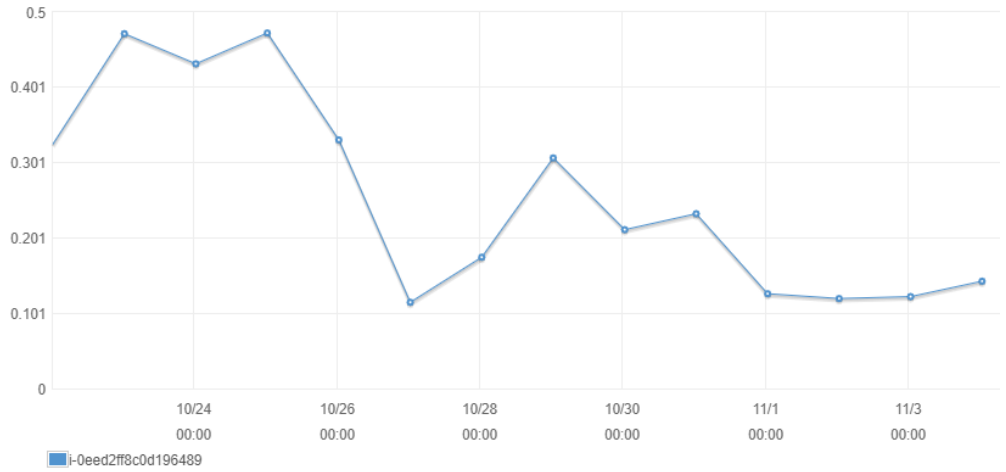


Fonte: Do Autor (2019)

A Figura 4.9 mostra a tela de edição do sistema embarcado com diversas informações sobre o *hardware*, os últimos processos de retirada que foram executados pelo mesmo e as ações que podem ser executadas, como, por exemplo, alterar o tipo de sensor de fluxo, reiniciar e desativar o mesmo.

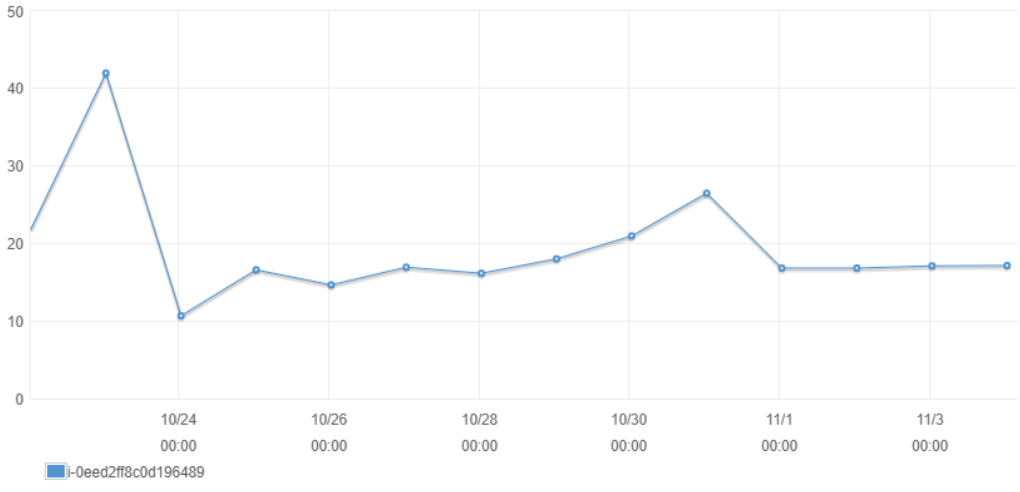
Com relação aos servidores utilizados, as Figuras 4.10 a 4.13 mostram a média e o máximo de utilização da CPU para os servidores BM-*Backend* e BM-AdminPanel para um período de duas semanas. É possível observar que apesar das configurações bastantes modestas, os mesmos atualmente se encontram superdimensionados, indicando que ainda existe margem para utilização e que uma futura junção dos serviços não acarretaria numa perda de desempenho das aplicações.

Figura 4.10 - Uso médio da CPU do servidor BM-*Backend*.



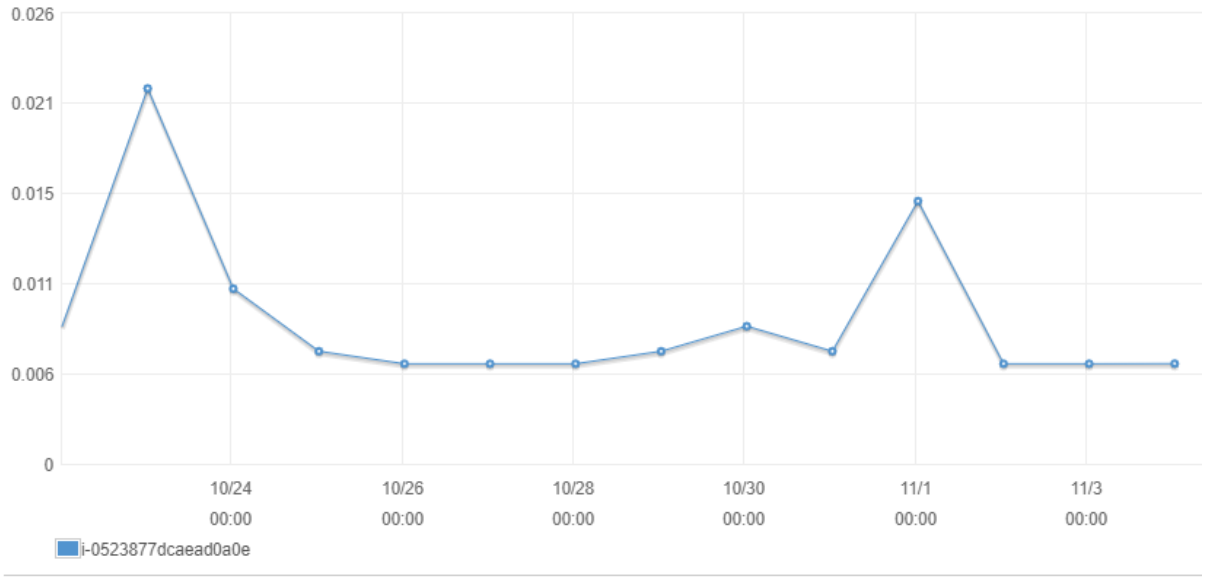
Fonte: Do Autor (2019)

Figura 4.11 - Uso máximo da CPU do servidor BM-*Backend*.



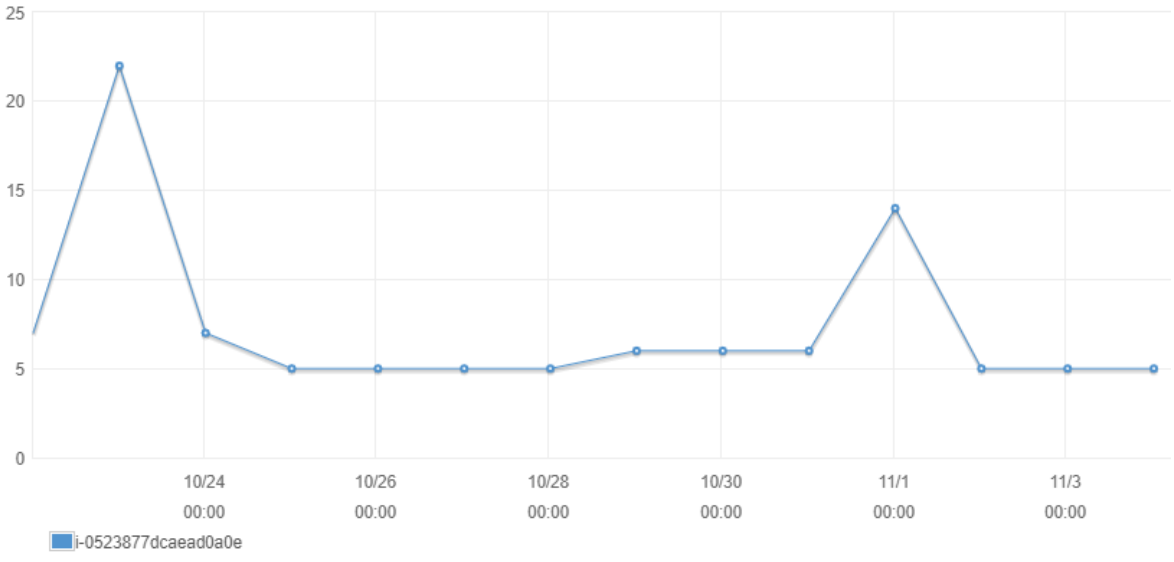
Fonte: Do Autor (2019)

Figura 4.12 - Uso médio da CPU do servidor BM-AdminPanel.



Fonte: Do Autor (2019)

Figura 4.13 – Uso máximo da CPU do servidor BM-AdminPanel.



Fonte: Do Autor (2019)

Analisando as ações executadas e os resultados obtidos durante os testes, pode-se afirmar que o sistema apresentou o comportamento esperado e foi validado como um todo. O processo de envase de chope realizado pela máquina também foi satisfatório, apresentando as seguintes vantagens:

- Permite o ajuste do fluxo de forma a reduzir a quantidade de espuma servida no copo e consequentemente apresentando funcionamento melhor do que a máquina de Ortiz (2013);
- Realiza a correta medição de fluxos variáveis, o que torna a leitura mais precisa do que o equipamento desenvolvido por Gouveia e Lopes (2012);
- Possui instalação mais simples que a máquina desenvolvida por Santos e Tanaka (2017), não necessitando realizar nenhuma adaptação, sendo suficiente apenas conectar o barril e realizar as operações de configuração.

O painel administrativo fornece o gerenciamento e monitoramento do sistema e dispositivos em geral, além de possibilitar a troca de dados de forma eficiente e com segurança. Estes são diferenciais não fornecidos por nenhuma das outras máquinas.

5. CONCLUSÃO

Com o objetivo de melhorar a venda fracionada de bebidas frias e reduzir os gastos com mão de obra, neste trabalho foi registrado o desenvolvimento de um sistema de autoatendimento integrado com a internet, construído com componentes de baixo custo. Para isto, foi desenvolvido um sistema embarcado utilizando um microcontrolador ESP-WROOM-32 capaz de realizar o controle do *hardware* responsável por realizar o envase da bebida, e fazer toda a comunicação com o *backend*, que é o responsável pela lógica de funcionamento e processamento de informações. Para completar a proposta de autoatendimento, um *tablet* com um aplicativo desenvolvido especificamente para o projeto e um painel administrativo foram incluídos na chopeira de autoatendimento da Beer Mine. Vale ressaltar, neste projeto houve a colaboração de outros funcionários da empresa.

O *backend*, responsável pelas manipulações no banco de dados, forneceu o suporte necessário para troca de informações dos sistemas e equipamentos envolvidos, garantindo, junto com os serviços prestados pela Amazon, que o sistema funcionasse de forma eficiente e segura. O aplicativo desenvolvido para funcionar no *tablet*, permitiu aos usuários que as operações realizadas fossem simples e autoexplicativas, tornando a usabilidade agradável e garantindo que o tempo de uso fosse rápido suficiente para reduzir o tempo total de compra. O painel administrativo desenvolvido para acesso dos colaboradores também apresentou resultados positivos, possibilitando o monitoramento, em tempo real, e análise dos registros das operações realizadas pelo sistema, além de fornecer diversas outras funcionalidades como controle de estoque, vendas e relatórios. Finalmente o sistema embarcado cumpriu seu objetivo, realizando a troca de informações com o *backend*, informando sobre as condições de uso e os processos que são executados, tudo isso em tempo hábil. Além disso, ele permite atualizações *Over The Air* (OTA), que auxiliam no caso de atualizações e melhorias no código, não sendo necessária a presença de um técnico para realizar a atualização do *firmware*.

Tendo então, cada elemento do sistema cumprido seu respectivo papel, é possível afirmar que o sistema, como um todo, atingiu com o objetivo proposto, sendo capaz de realizar toda a operação do pedido, pagamento e retirada do chope sem auxílio de nenhum atendente. Além disso, o sistema se mostrou bastante robusto, tanto nas operações realizadas remotamente pelos sistemas de gerenciamento, como no seu funcionamento físico em condições adversas, como interrupção no fornecimento de energia e possíveis quedas de internet. Já com relação às melhorias, os próximos passos são a integração com sistemas de pagamento que utilizam

cartões por rádio frequência (RFID), uma vez que diversos estabelecimentos adotam este sistema.

REFERÊNCIAS

AWS. **Amazon Elastic Compute Cloud**. 2019. Disponível em: <https://docs.aws.amazon.com/pt_br/AWSEC2/latest/UserGuide/concepts.html>. Acesso em: 01/11/2019.

AWS. **Amazon Simple Storage Service**. 2019. Disponível em: <https://docs.aws.amazon.com/pt_br/AmazonS3/latest/dev/Introduction.html>. Acesso em: 01/11/2019.

BADU, K. **Human Use Analysis**: Vending Machine. [2007?]. Disponível em: <http://web.mit.edu/2.744/studentSubmissions/humanUseAnalysis/keval/index.html >. Acesso em: 01/11/2019.

CHIPSKEY. **YF-S301 DC5～24V 0.3～10Lmin Water Flow Sensor**. 2019. Disponível em: < https://www.chipskey.cc/yfs301-dc524v-0310lmin-water-flow-sensor-p-7173.html>. Acesso em: 25/10/2019.

DANFOSS. Type EVR 2 - EVR 40 Version 2. 2019. Disponível em: <https://assets.danfoss.com/documents/DOC313934259602/DOC313934259602.pdf>. Acesso em: 01/11/2019.

EASYCHOPP. Apreciadores e Consumidores. 2019. Disponível em: <https://www.easychopp.com.br/apreciadores>. Acesso em: 01/11/2019.

EBERT, C.; SALECKER, J. Embedded Software – **Technologies and Trends**. 2009. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4814953 >. Acesso em: 02/11/2019.

EDX. **Embedded Systems**. 2019. Disponível em: < https://www.edx.org/learn/embedded-systems>. Acesso em: 02/11/2019.

ESPRESSIF. **ESP32-WROOM-32– Datasheet**. 2019. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf>. Acesso em: 31/10/2019.

FILIPEFLOP. **Servo JX PDI-6221MG Alto Torque 20Kg**. 2019. Disponível em: <https://www.filipeflop.com/produto/servo-jx-pdi-6221mg-alto-torque-20kg/>. Acesso em: 01/11/2019.

GURU99. **What is Database? What is SQL?**. 2019. Disponível em: <https://www.guru99.com/introduction-to-database-sql.html>. Acesso em: 01/11/2019.

GOUVEIA, A. A.; LOPES, G. M. **Automatização de uma Máquina de Vazão de Líquidos**. Curitiba, PR. 2012.

HARDY, E. **RCA Voyager III Review: Low on Price and Performance**. 2017. Disponível em: <http://www.notebookreview.com/tabletreview/rca-voyager-iii-review-low-price-performance/>. Acesso em: 03/11/2019.

HASEGAWA, G. et al. **Nível de Aceitação da Inovação em Um Comércio Cervejeiro de Guarapuava-Pr**: Estudo de Caso à Luz da Estratégia Competitiva. RESO, Guarapuava, v. 1, n. 2, p. 73-85, jul./dez. 2018.

HOFFMAN, C. **What Is HTTPS, and Why Should I Care?.** 2018. Disponível em: <https://www.howtogeek.com/181767/htg-explains-what-is-https-and-why-should-i-care/>. Acesso em: 04/11/2019.

IGHODARO, N. **How Laravel implements MVC and how to use it effectively**. 2018. Disponível em: < https://blog.pusher.com/laravel-mvc-use/>. Acesso em: 01/11/2019.

JWT. **Introduction to JSON Web Tokens**. 2019. Disponível em: <https://jwt.io/introduction/>. Acesso em: 03/11/2019.

MELL, P.; GRANCE, T. **The NIST Definition of Cloud Computing**. 2011. Disponível em: <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>. Acesso em: 01/11/2019.

Mitchell, B. **What Is a Server?**. 2019. Disponível em: <https://www.lifewire.com/servers-in-computer-networking-817380 >. Acesso em: 01/11/2019.

ORTIZ, L. M. **Projeto de Uma Chopeira Automatizada**. 2013. Disponível em: <http://fatecgarca.edu.br/uploads/documentos/tcc/monografias/mecatronica/2013/Leandro%20de%20Melo%20Ortiz%20-%20Projeto%20de%20uma%20chopeira%20automatizada.pdf>. Acesso em: 27/10/2019.

PAGSEGURO. **Moderninha Plus**. 2019. Disponível em: <https://pagseguro.uol.com.br/para-seu-negocio/maquininhas/moderninha-plus>. Acesso em: 01/11/2019.

RESTAPITUTORIAL. **HTTP Status Code**. 2019. Disponível em: <https://www.restapitutorial.com/httpstatuscodes.html>. Acesso em: 03/11/2019.

ROUSE, M; BISCOBING, J. **Relational Database**. 2017. Disponível em: <https://searchdatamanagement.techtarget.com/definition/relational-database>. Acesso em: 01/11/2019.

SANDERS, S. **Aceitação de Tecnologia**: Uma Análise da Atitude de Passageiros Diante de Totens de Autoatendimento em Aeroportos. Brasília, DF. 2011. Disponível em: <https://bdm.unb.br/handle/10483/3519>. Acesso em: 28/10/2019.

SANTOS, L.G.; TANAKA, M**. Desenvolvimento de uma Mesa Chopeira de Autoatendimento**. Ponta Grossa, PR. 2017. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/handle/1/8513>. Acesso em: 27/10/2019.

SILVEIRA, C. B . **Como Funciona a Válvula Solenoide e Quais São os Tipos Existentes?**. 2017. Disponível em: <https://www.citisystems.com.br/valvula-solenoide/>. Acesso em: 25/10/2019.

W3SCHOOLS. **What is JSON?**. 2019. Disponível em: <https://www.w3schools.com/whatis/whatis_json.asp>. Acesso em: 03/11/2019.

WODEHOUSE, C. **A Beginner's Guide to Back-End Development**. 2018. Disponível em: <https://www.upwork.com/hiring/development/a-beginners-guide-to-back-end-development/>. Acesso em: 02/12/2019.