ServicesLocker: uma solução de armário inteligente baseado em Internet das Coisas

Arthur de Bortoli[1], **Jeferson Gomes da Silveira**[1], **Robson Costa**[1]

[1]Instituto Federal de Santa Catarina (IFSC)
Câmpus Lages – 88506-400 – Lages – SC – Brasil

{arthurdb1999, jefersongomes0101}@gmail.com, robson.costa@ifsc.edu.br

Abstract. *Amongst an increasingly chaotic urban lifestyle, yet technological at the same time, an activity that may be harmed is the parcel delivery at residencies due to the recipient absence at the time of the delivery. Using the Internet of Things concept grounded in the 5 layers architecture, this work proposes a minimun viable product based on a smart locker, and in a Web system, applying a business model based on a intermediate service provisioning, focused on the clothes washing service and on the parcel delivery meanwhile enabling scalability to other types of services. It was developed using technologies such as Angular, Node.js, Thingsboard and Raspberry Pi as the hardware.*

Resumo. *Em meio a um cenário de vida urbana cada vez mais corrido e atarefado, mas ao mesmo tempo tecnológico, cada vez mais as pessoas buscam por serviços que facilitam as tarefas do dia a dia. Com a utilização do conceito de Internet das Coisas, este trabalho propõe um estudo de caso baseado em um armário inteligente. Partindo de um modelo de negócio baseado na prestação de serviços intermediários entre 2 partes com enfoque no serviço de lavação de roupa e ao mesmo tempo, possibilitando a escalabilidade para outros tipos de serviços como a entrega de mercadorias. Foi desenvolvido com a utilização das tecnologias Angular, Node.js, Thingsboard e Raspberry Pi como hardware.*

1. Introdução

Com a aceleração do desenvolvimento tecnológico nas duas últimas décadas, principalmente no contexto da computação ubíqua[1], grande parte da população mundial mantém-se conectada boa parte do dia na Internet. Como consequência desta grande acessibilidade aos recursos de informação e comunicação, observa-se também uma sobrecarga de afazeres diários, sejam estes laborais, acadêmicos ou até mesmo domésticos. Desta forma, o tempo acaba sendo um recurso cada vez mais valioso e limitado.

Neste contexto, a locomoção das pessoas para a execução de determinadas tarefas apresenta-se como um problema cada vez mais comum. Isto deve-se em grande parte pela impossibilidade destas estarem em determinados lugares em horários comerciais, bem como pelo trânsito e/ou falta de meio de transporte adequado, sendo estes dois últimos os principais motivos em grandes metrópoles.

Para tentar amortizar este problema e alavancar os seus negócios, as empresas procuram constantemente diferentes maneiras de aumentar a sua capilaridade. O método

[1]A computação ubíqua descreve a onipresença das Tecnologias da Informação e Comunicação (TICs) no cotidiano das pessoas.

tradicional, espalhar pontos de atendimento em diferentes locais de diferentes cidades, acaba por se tornar demasiadamente dispendioso. Como consequência, para suprir o aumento no custo operacional, as empresas acabam aumentando o preço dos seus produtos e/ou serviços, o que, em muitas situações, pode acarretar na perda de competitividade.

Como alternativa, observa-se o surgimento de empresas que lançam mão de um modelo de negócio baseado em uma plataforma multilateral. Neste modelo, uma empresa oferta um serviço intermediário para interligar as partes interessadas de um negócio, ou seja, os clientes aos efetivos comerciantes de produtos e/ou serviços (ex.: Uber, AirBnB, iFood e DeliveryMuch).

Nesta abordagem, ambas as partes do negócio são beneficiadas. Enquanto que aos clientes a plataforma permite simplificar e agilizar a tarefa de procurar, comparar e adquirir produtos e/ou serviços, aos comerciantes destes produtos e/ou serviços possibilita um serviço de propaganda e venda adicional além de muitas vezes permitir a redução da sua complexidade logística (ex.: serviço de entrega prestado por algumas empresas). No que tange as empresas que prestam este serviço intermediário de ligação entre o cliente e o efetivo fornecedor final, tal modelo lhes permite realizar a comercialização de produtos e/ou serviços que elas na verdade não possuem, eliminando assim toda a logística e *know how* necessários para mantê-los.

Ademais, outra mudança que a aceleração da evolução tecnológica está proporcionando é o aumento significativo de dispositivos conectados à Internet (Meulen, 2017), sejam computadores e *smartphones* ou elementos da vida cotidiana como *SmartTVs*, interruptores de luz e sensores diversos. Estes últimos, por sua vez, fazem parte dos dispositivos que ajudam a compor a Internet das Coisas (*Internet of Things* – IoT), a qual se caracteriza como um paradigma de comunicação o qual prevê que objetos da vida cotidiana sejam equipados com microcontroladores, transmissores para comunicação digital e pilhas de comunicação de forma a possibilitar que estes se comuniquem entre si e com usuários, tornando-se assim parte da Internet (Zanella et al., 2014).

Este paradigma permite a criação de aplicações em diferentes domínios, como automação residencial, automação industrial, cuidados de saúde, gestão inteligente de energia, controle de tráfego entre outros (Bellavista et al., 2013).

Neste contexto, lançando mão do conceito de prestador de serviço intermediário, o qual visa conectar clientes aos fornecedores de serviço de lavanderia, assim, integrando o conceito de IoT, e focando na solução do problema de tempo e locomoção da população, este trabalho propõe um estudo de caso de uma solução baseada em armários inteligentes (*smartlocker*). Este, por sua vez, além de disponibilizar uma plataforma para interação entre os prestadores de produtos e/ou serviços e os clientes, também visa implantar armários inteligentes em locais públicos e estratégicos das cidades, possibilitando a fácil e rápida inserção e retirada de roupas. Como exemplo de negócios que podem se beneficiar desta solução, é possível citar desde lavanderias (realização de entrega e retirada de roupas) até varejistas, para a realização da entrega de encomendas, sejam estas realizadas entre pessoas físicas ou até mesmo em um sistema *Business To Consumer (B2C)*[2]. Em suma, se uma empresa possui um serviço e/ou produto que pode se beneficiar da solução dos

[2]Sistema de comércio efetuado diretamente entre a empresa produtora, vendedora ou prestadora de serviços e o consumidor final.

espaços dos armários, é possível que a empresa se cadastre na plataforma para oferecer o seu serviço e interagir com os seus clientes, no sentido de receber pedidos, tirar dúvidas e manter a comunicação com o mesmo e vice-versa. Assim, visando trazer a solução para as pessoas que não possuem tempo disponível para receber determinado produto, objeto ou qualquer outro item que possa caber no armário. Além disto, há uma economia de tempo no fato das pessoas contratarem de maneira prática, os serviços disponíveis na plataforma e que façam uso do armário, como o exemplo da lavanderia ou de uma encomenda que poderia esgotar as tentativas de entrega e que não se faz necessário o cliente ter que buscar as roupas, ou a encomenda no correio, pois a empresa poderia deposita-los no armário mais próximo de sua localidade.

De forma a alcançar este objetivo geral os seguintes objetivos específicos foram definidos:

- estudar as tecnologias necessárias para seu desenvolvimento;
- conceber, integrar e configurar o *hardware* necessário para a prospecção do armário inteligente;
- desenvolver o *firmware* a ser utilizado pelo respectivo *hardware*, e;
- desenvolver as interfaces de gestão utilizando da solução proposta.

A metodologia utilizada neste trabalho lançou mão de um estudo transversal, exploratório/descritivo com abordagem mista, dividida em 3 (três) etapas (Fortin et al., 2009). Na primeira etapa foi realizado um estudo mercadológico da solução proposta, apontando potenciais concorrentes, parceiros e clientes de forma a refinar o escopo de atuação da mesma. Na segunda etapa, foram levantados os requisitos técnicos necessários para a solução tendo em vista um produto minimamente viável (*Minimum Viable Product* – MVP). A terceira e última etapa foi dividida nas seguintes atividades: i) integração e teste do *hardware*; ii) implementação e teste do *firmware*, iii) implementação e teste de uma interface de gestão, e; iv) implementação e teste de uma interface do cliente.

Este documento está organizado da seguinte forma: na seção 2 é apresentado o referencial teórico utilizado como base para o desenvolvimento deste trabalho; na seção 3 é descrita e detalhada a solução proposta; na seção 4 são apresentados os testes realizados bem como seus respectivos resultados; por fim, na seção 5, é retratada a conclusão obtida assim como direcionamentos para trabalhos futuros.

2. Referencial Teórico

2.1. Internet das Coisas

De acordo com Al-Fuqaha et al. (2015), a Internet das Coisas (IoT) traz a ideia de dispositivos físicos analógicos e digitais conectados à Internet. Os dispositivos físicos podem conter sensores que captam informações como a luz, a temperatura e a umidade do ambiente. A grande variabilidade de tipos de sensores permite a criação de sistemas de controle para diversas finalidades, sendo uma das mais conhecidas aplicadas à área da domótica, ou também conhecida como casas inteligentes (*Smart Homes*).

Dentre outras aplicações de IoT que podem ser elencadas, cita-se a predição de desastres naturais através de sensores que detectam mudanças no ambiente em que estão, podendo tomar decisões antecipadas. Aplicações industriais, monitoramento de escassez de água, aplicações médicas, aplicações na agricultura, inteligência em sistemas de transporte e cidades inteligentes (*Smart Cities*) (Khan et al., 2012).

Chaudhary et al. (2019) ratificam estas ideias sobre as soluções IoT e ressaltam alguns dos cenários onde estas podem ser aplicadas. Além disso, Chaudhary et al. (2019) comentam sobre sensores que conversam entre si para encontrar o indivíduo dentro de casa e avisá-lo que é necessário encher a garrafa de água da geladeira quando esta estiver em um nível baixo. Também citam aplicações IoT que podem ajudar pessoas com deficiência. Sensores que podem captar informações climáticas do solo e do ambiente para ajudar o agricultor a planejar qual a melhor data do ano que para plantar as sementes e também na detecção de doenças e pragas que podem atingir a plantação. Isto permite ao produtor reduzir o uso de defensivos agrícolas, visto que, se em diferentes áreas da plantação, pode não haver uma possível ameaça, logo não é necessário a sua aplicação.

Quando se fala em aplicações IoT, logo lembra-se que são dispositivos conectados à Internet. Porém, não se costuma imaginar o cenário ao qual estes dispositivos estão inseridos e qual o problema que estão tentando resolver. Há situações em que os dispositivos necessitam estar em locais que são distantes de pontos de fornecimento de energia e de uma conexão à Internet, como é o caso de algumas estações meteorológicas e de sensores em boias no mar. Os desafios são diversos e requerem abordagens diferentes para cada cenário. Routh e Pal (2018) citam alguns dos principais desafios enfrentados no desenvolvimento de aplicações IoT:

Segurança
: Tratar vulnerabilidades no *hardware* dos dispositivos e garantir a privacidade de dados pessoais como a localização;

Conectividade
: Garantir a comunicação de bilhões de dispositivos compartilhando uma significante quantidade de informações;

Compatibilidade
: Integrar as diferentes tecnologias de *hardware* e protocolos criados por diferentes empresas;

Complexidade
: Gerir a falta de uma padronização geral, o que torna a tarefa de integração entre as diferentes tecnologias e arquiteturas complexa;

Gerenciamento de grandes quantidades de dados
: Faz-se necessária a utilização de uma arquitetura de armazenamento e processamento de dados adequada;

Fluxo de dados
: A grande quantidade de dados transmitidos pelos dispositivos IoT e as escalas das aplicações fazem com que muitas vezes seja necessário redesenhar a arquitetura de comunicação para evitar a interrupção das comunicações;

Energia
: Muitas aplicações precisam fazer uso de baterias como fonte de energia, isto torna o seu desenvolvimento mais complexo, limitando a escolha das tecnologias empregadas desde o *hardware* até os protocolos de comunicação;

Com o crescimento acelerado do número de aplicações IoT sendo disponibilizadas no mercado, torna-se evidente a necessidade de uma padronização em sua arquitetura. Recentemente, em 10/03/2020, foi publicada a norma IEEE 2413 (IEEE, 2020a) a qual define uma padronização para a estrutura arquitetônica de aplicações IoT. Derivadas desta norma, e ainda em fase de desenvolvimento, estão sendo definidas as normas IEEE

P2413.1 (IEEE, 2020b), a qual define uma arquitetura de referência para *Smart Cities*, e a norma IEEE P2413.2 (IEEE, 2020c), a qual define uma arquitetura de referência para aplicações focadas na distribuição de energia.

Antes destas, várias propostas haviam sido realizadas na comunidade científica, sendo as principais baseadas 3, 5 e 7 camadas. Dentre estas, a arquitetura de 5 camadas é o modelo mais aplicável devido a sua simplicidade (Al-Fuqaha et al., 2015). Segundo Khan et al. (2012), a arquitetura de 5 camadas utiliza a estrutura apresentada na Figura 1:

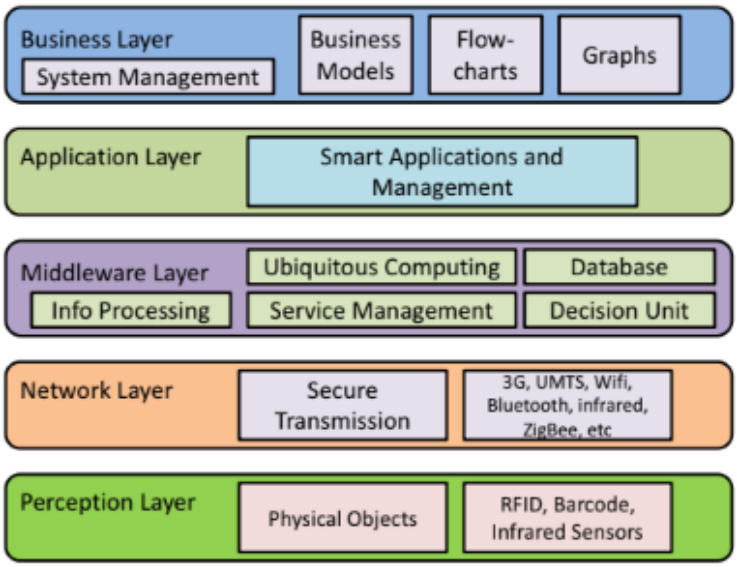

Figura 1. Arquitetura em 5 camadas (Khan et al., 2012).

1. Camada de Percepção (*Perception Layer*)
   Trata-se da camada mais baixa da arquitetura, nela encontram-se os dispositivos físicos, contendo microcontroladores, sensores e/ou atuadores, além de suas interfaces de comunicação;

2. Camada de Rede (*Network Layer*)
   Responsável por permitir a conexão do dispositivo com um *gateway* central ou diretamente com a Internet, nesta são definidas quais as tecnologias de comunicação (ex.: 3G, WiFi, Ethernet, Bluetooth, Zigbee, LoRaWAN) o dispositivo utilizará;

3. Camada de *Middleware* (*Middleware Layer*)
   Disponibiliza diversos tipos de serviços como *Global Sensor Networks* (GSN), *Data Distribution Service* (DDS) e *Link Smart*, estes que são responsáveis por possibilitar que diferentes sistemas operacionais e dispositivos possam se comunicar; esta camada possui conexão com a base de dados, sendo também responsável por receber os dados da camada inferior e armazená-los;

4. Camada de Aplicação (*Application Layer*)
   Responsável por fornecer um sistema de gerenciamento global das informações

processadas na Camada de *Middleware*, nesta também são definidos os protocolos de aplicação (ex.: MQTT, CoAP, HTTP) utilizados pela plataforma IoT;

5. Camada de Negócio (*Business Layer*)
   Trata-se da camada mais alta na arquitetura IoT, é responsável por gerenciar todo o sistema, gerando gráficos e modelos de negócio baseados nos dados recebidos pela camada inferior, nesta são aplicadas soluções de processamento de dados e inteligência artificial para otimizar a utilização dos dados possibilitando auxiliar na tomada de decisões acerca das estratégias de negócio;

2.2. Armários Inteligentes

O conceito de "armário inteligente" (*smart locker*) consiste em um armário que seja acessível ao usuário durante a maior parte do tempo, ou seja, o usuário deve ser hábil a utilizá-lo a seu próprio rigor, no horário que para ele for conveniente. Nesse contexto, o fato que designa o adjetivo "inteligente" é dado pela tecnologia implantada nesse armário, baseada no sistema *IoT* supracitado (Faugere e Montreuil, 2016). Esse conceito diz respeito a uma solução comercial, lucrativa, inovadora e recente, a qual já é explorada pelas empresas no cenário atual.

Valendo-se dessa veia comercial que a ideia apresenta, os seguintes pontos são essenciais para a consolidação dos armários inteligentes: i) capacidade de armazenar pacotes adequadamente para seus clientes, durante um certo período de tempo; ii) o armazenamento dos pacotes deve ser feito de forma segura, onde apenas o locatário (usuário) ou membros por ele autorizados possam efetuar a abertura de um armário; iii) localizar-se em local público e conveniente, para que possa atender adequadamente as necessidades do bairro em que se encontra (Faugere e Montreuil, 2016).

A arquitetura de 5 camadas pode ser utilizada para o desenvolvimento de uma solução de armário inteligente que atenda aos requisitos citados anteriormente. De forma a garantir seu funcionamento contínuo, uma vez que o terminal de armários encontra-se estático, a melhor forma de alimentar seu *hardware* é por cabeamento direto na energia elétrica, sem grandes impedimentos para a execução deste método. Todavia, não há obstáculos para uma implementação com baterias, apesar de tornar a solução menos eficiente no quesito suplementação energética. Da mesma forma, é necessário garantir a conexão do *hardware* com a Internet, isto é, através de cabeamento *Ethernet*, por rede *WiFi* ou infraestrutura de telefonia móvel. Este item relaciona-se diretamente com o local em que o terminal está posicionado, e pode-se adaptar conforme disponibilidade de rede.

Ademais, a grande característica que diferencia as empresas do ramo é a capacidade de fornecer ao cliente um sistema de personalização do uso da solução. Nesse sentido, torna-se imprescindível o desenvolvimento de plataformas e aplicações online que se comunicam com o sistema IoT dos armários. O cliente deve ter controle sobre o seu pacote, bem como do armário alugado. Deve ser hábil a visualizar o seu estado e personalizar como bem entender.

Dentre as soluções encontradas, Sa-ngiampak et al. (2019) descrevem o *LockerSwarm*. Trata-se de um sistema *IoT* de armários inteligentes de uso compartilhado, implementado com a justificativa de facilitar o processo de deslocamento de pertences e materiais dos estudantes em grandes campus universitários. A solução foi desenvolvida e

testada na Universidade de Chulalongkorn, Tailândia.

O *LockerSwarm* foi projetado para ser de utilização compartilhada entre os alunos com acesso, sem utilização de chaves físicas e abertura do armário realizada completamente por meio de um *smartphone*. Seu funcionamento é composto por 3 tecnologias: i) Arduino[3] equipado com uma fechadura eletrônica, responsável pela abertura do armário; ii) aplicação *front-end*[4] desenvolvida utilizando LIFF[5] e um LINE[6] *chatbot*[7]; iii) servidor *back-end*[8] que monitora e controla todos os armários ativos, desenvolvido utilizando NestJS[9].

Cada armário possui um botão que, ao ser pressionado, mostra o QR code referente ao armário em um visor. Além disso, o dispositivo envia alertas ao servidor para notificar baixa bateria. O usuário utiliza o aplicativo para efetuar a leitura do QR code e desbloquear o armário. Os administradores podem criar grupos de alunos que compartilham determinado armário. Cada estudante tem direito a uma certa quantidade de créditos mensais, que é decrementada após o aluguel de um armário.

Com o objetivo de focar em uma demanda diferente, David e Chalon (2018) descrevem um modelo de armário inteligente que suporta o armazenamento de alimentos frios ou congelados. Para isso, utiliza uma abordagem diferente comparado aos demais que existem no mercado:

> *"As múltiplas portas pertencentes a cada armário são substituídas por uma única porta. Isso se deve ao fato de os alimentos não serem armazenados em compartimentos estáticos, mas sim em cestos com diversos compartimentos menores. A posição vertical de cada um desses compartimentos menores é relacionada com sua temperatura correspondente (ambiente, fresco ou congelado). Os cestos deslocam-se em sentido circular para se posicionarem em frente à porta no momento do armazenamento ou retirada do alimento."*

O cenário de utilização dos armários é padrão comparado a outros serviços do ramo. O cliente/usuário realiza a compra de alimentos por meio de uma plataforma online e seleciona o terminal, data e hora de retirada desejado. Por sua vez, o serviço aciona o departamento de logística, que entrega o pedido ao funcionário encarregado pelo terminal de armários. Então, os alimentos são armazenados adequadamente nos cestos e uma notificação é enviada ao cliente final, informando a finalização do processo.

Alqahtani et al. (2020) descrevem uma solução de armários inteligentes similar ao *LockerSwarm*, apresentado acima. Com o intuito de fornecer maior segurança aos

[3]Microcontrolador para prototipagem eletrônica de *hardware* livre e placa única.

[4]Prática de conversão de dados em interface gráfica, de modo que permita ao usuário visualizar e interagir com a mesma.

[5]*Framework front-end* que permite a visualização de páginas *web* dentro do aplicativo LINE.

[6]Aplicativo *mobile* de troca de mensagens, amplamente utilizado na Tailândia, semelhante ao conhecido *WhatsApp*. Possui uma API pública que permite aos desenvolvedores construir novas aplicações utilizando o LINE. Link: `https://developers.line.biz/en/services/messaging-api`

[7]Programa de computador que tenta simular um ser humano ao conversar com outras pessoas, por meio de mensagens de texto.

[8]Parte do sistema que lida com as regras de negócio, segurança, conexão com banco de dados e rede.

[9]*Framework back-end* para construção de servidores escaláveis, utilizando *Node.js*, que por sua vez, é um interpretador Javascript assíncrono e orientado a eventos.

pertences de alunos universitários, o artigo expõe uma solução utilizando o protocolo de comunicação sem fio Bluetooth para desbloquear e obter o acesso ao armário. Nela o usuário utiliza um aplicativo *Android* instalado em seu *Smartphone*, desenvolvido através do MIT App Inventor[10], responsável por realizar a comunicação com o módulo Bluetooth acoplado a um Arduino e instalado nos armários da universidade.

Nesse sistema, o usuário realiza o *login* no aplicativo e seleciona a rede Bluetooth apropriada para conectar-se e realizar o desbloqueio da trava. Em casos de problemas com conexão ou com o *Smartphone* do usuário, um teclado numérico é disponibilizado no armário, possibilitando a digitação de uma senha para efetuar o desbloqueio do dispositivo. Quando um evento de *login* obtiver sucesso, um LED verde acenderá, caso contrário, ao digitar uma senha incorreta, um LED vermelho acenderá e permitirá ao usuário realizar mais 3 tentativas de senha para acesso ao armário.

Conforme exposto, nota-se uma gama de abordagens diferenciadas para utilização de soluções de armários inteligentes, principalmente quando se trata de métodos de acesso e desbloqueio, devido aos diferentes critérios de segurança que podem ser adotados.

Dentre estes critérios, pode-se citar o sistema descrito por Ze-hong e Guang-yuan (2015), que lança mão de uma abordagem baseada na tecnologia GSM[11]/GPRS[12]: o entregador deposita o pacote no compartimento alocado e digita os números de telefone dos usuários autorizados a retirar o pacote. Um SMS com código de verificação é enviado aos usuários, e uma senha é gerada para possibilitar a retirada da encomenda. O cliente digita a sua senha de autorização, retira o pacote, e o processo é encerrado.

Por outro lado, existem certas situações em que um sistema de segurança robusto é extremamente importante para a eficácia da solução, principalmente ao lidar com dinheiro em espécie. Este é o caso descrito por Chikara et al. (2020), cuja implementação abrange mais de uma camada de segurança, utilizando reconhecimento facial, leitura de impressão digital e senha de autenticação.

O caso de uso em questão contempla o acesso a cofres que encontram-se dentro de bancos, e seu fluxo se dá da seguinte maneira: uma câmera capturará a face do indivíduo na porta de entrada da seção de cofres, e então, a mesma é comparada através de um algoritmo HOG: *Histogram of Oriented Gradientes*, ou Histogramas de Gradientes Orientados, um algoritmo descritor[13] da área da computação gráfica e processamento de imagem, com o objetivo de detecção de objetos, desenvolvido com MATLAB[14]. Uma vez que a face é reconhecida, a porta se abre e o cofre é exposto. Nesta etapa, o usuário deverá digitar a senha fornecida pelo banco e escanear sua impressão digital. Se as informações fornecidas forem genuínas, o acesso ao cofre é concedido.

[10] Aplicação de código aberto criado pela Google e mantida pelo MIT (*Massachusetts Institute of Technology*). Permite ao programador iniciante arrastar e soltar blocos visuais de código para a criação de um aplicativo Android.

[11] *Global System for Mobile communications*, ou Sistema Global para Comunicações Móveis, popularmente conhecido como um sistema de telefonia de segunda geração (2G), possuindo sinal e canal de voz digitais. Utilizado posteriormente como base para o desenvolvimento da terceira geração (3G).

[12] *General Packet Radio Service*, ou Serviços Gerais de Pacote por Rádio, responsável por aumentar as taxas de transferência de dados nas redes GSM (varia de 40 a 144 *kbps*), utilizando a técnica de comutação de pacotes.

[13] Tipo de algoritmo que produz descrições de uma imagem, como formas, cores, textura ou movimento.

[14] *Software* interativo de alta performance voltado para o cálculo numérico com matrizes.

2.3. Funcionalidades Disponibilizadas

Há um grande leque de abordagens tecnológicas utilizadas pelo mercado atual, cada qual contendo seu propósito específico de existência, seja para aprimorar a segurança do processo ou prover serviços de confiança e qualidade. Por outro lado, alguns pontos são de comum senso entre os serviços existentes. Aqui, pode-se citar o método de acesso dos usuários aos compartimentos.

Segundo Faugere e Montreuil (2016), todas as empresas avaliadas no estudo utilizam a mesma forma de acesso para o entregador, isto é, a partir de um *login* com nome de usuário e senha, ou então por meio de escaneamento de código de barras ou *QR code*. Alguma forma de escaneamento sempre se faz presente nos terminais, dessa forma pode-se aferir seu depósito, reconhecer se o pacote está no terminal correto e principalmente distribuir essas informações aos interessados (remetente e destinatário). Ademais, ao depositar o pacote, o entregador provê uma identificação ou código de entrega, a qual poderá ser utilizada pelo destinatário final para retirada.

Por sua vez, o cliente final pode ter duas maneiras de acesso ao compartimento: a partir de um código de retirada, gerado e enviado ao seu email ou aplicativo utilizado para intermediação; ou então a partir de alguma informação pessoal anexada ao pedido pelo varejista, como documento de identificação, número de cartão de crédito ou ainda a partir de *login* e senha.

Para garantir a segurança, os terminais possuem câmeras que monitoram a infraestrutura do negócio. Porém, nenhuma abordagem de segurança é implementada no pacote em si, como sensores de presença internos ou rastreamento de localização, por exemplo.

Por fim, alguns recursos podem ser adicionados para suprir demandas de segurança que um sistema *Consumer to Consumer*[15](C2C) exige. Empresas como a Cisco proveem formas de pagamento *in loco*, permitindo que o sistema, juntamente com as câmeras de segurança, fiscalizem e garantam a realização do pagamento. Além disso, podendo também serem utilizadas em um sistema *B2C*, impressoras podem ser adicionadas aos terminais para impressão de notas fiscais ou recibos.

Em sistemas de armários inteligentes como este, é evidente a necessidade de uma solução eficaz e completa de conexão com a Internet, portanto, todas as empresas fornecem ampla cobertura de conexão às necessidades de seu negócio.

Quanto às características físicas dos terminais, segundo Faugere e Montreuil (2016), nota-se uma média de capacidade volumétrica de aproximadamente 108L por compartimento, e suportam pesos de em média 22Kg. Vale ressaltar a capacidade de 5Kg suportados e 47L de volume do mais famoso, porém com menor compartimento, serviço de armários inteligentes, o Amazon Locker[16]. Por outro lado, o maior compartimento dentre as empresas listadas, com 30kg suportados e aproximadamente 378L de capacidade, foi o do Post 24[17], serviço suíço de armários inteligentes.

[15] Sistema de comércio efetuado entre dois consumidores.

[16] `www.amazon.com/b?ie=UTF8&node=6442600011`

[17] `www.post.ch/en`

2.4. Mercado e Modelos de negócio

A grande maioria das empresas que utilizam a tecnologia de armários inteligentes buscam melhores alternativas para otimizar um processo chamado de *last mile delivery*, ou "a última milha de entrega". Este termo refere-se à última etapa do processo de entrega de uma encomenda, quando o pacote é entregue ao cliente final. Na logística, essa é uma das etapas mais importantes e desafiadoras do processo, pois pode ser um ponto crucial para a satisfação do cliente com o serviço. Ademais, a última milha, pode ser uma das mais custosas e lentas etapas de uma entrega, visto que envolve uma série de variáveis como o tráfego das grandes metrópoles; a inevitabilidade de deslocamentos a pontos rurais e afastados da cidade; e a necessidade de se realizar várias paradas entre os pontos de entrega.

Os armários inteligentes são uma excelente alternativa para otimizar a última milha, pois dessa forma, as empresas de logística têm a possibilidade de alugarem armários e depositarem a encomenda, para que o cliente possa retirá-la assim que puder. Além de poupar gastos para as transportadoras, esse método evita casos em que é notada a ausência do cliente de sua residência no momento da entrega.

Essa é uma tendência futura. Cada empresa adaptar-se-á à tecnologia de sua própria maneira, tanto utilizando uma abordagem mista entre labor humano e armários inteligentes, quanto elaboração de uma volumosa e vasta rede de armários capazes de cobrir grande parte das necessidades de entrega. Nas grandes potências mundiais, este último cenário já se faz parcialmente presente em determinadas localidades.

Faugere e Montreuil (2016) elaboraram um estudo baseado na avaliação das melhores práticas atualmente implementadas e experimentadas pelas empresas que propõem uma solução baseada em armários inteligentes. O estudo avaliou 12 companhias relevantes ao redor do mundo, analisou seus modelos de negócio e buscou responder o que a indústria logística está utilizando para resolver os problemas do *last mile delivery*.

Como anteriormente descrito, os chamados "terminais" de armários estão localizados em áreas frequentadas regularmente, como *shoppings*, estações de trem ou locais públicos. O serviço polônes Inpost[18] possui uma rede de mais de 7000 armários implantados nas mais diversas localidades ao redor do planeta.

Alguns serviços também possuem restrições de uso, como o Inpost e o DPD Groupe[19], os quais exigem que as compras sejam feitas em determinados sites parceiros. Já a limitação do Amazon Locker é que as compras sejam realizadas no site da Amazon.

No caso de uma expansão do modelo de negócios para suportar o serviço de entrega, três entidades principais podem ser identificadas nesse modelo de negócio, e cada uma terá a sua etapa na entrega da encomenda, são elas: varejistas, transportadoras e terminais de armários inteligentes. Nesse sentido, existem dois cenários de negócio reais e um ideal.

Um dos cenários reais descreve as transportadoras firmando parcerias e se atrelando a apenas um ou dois serviços de terminais inteligentes. Nesse cenário, pode-se observar uma limitação do serviço de armários, pois não se tem o direito contratual de

[18] `https://inpost.pl/en/about-inpost`
[19] `www.dpd.fr`

operar com mais de uma transportadora.

O outro cenário real descreve os varejistas se atrelando a apenas um ou dois serviços de terminais inteligentes. Nessa perspectiva, pode-se observar uma limitação da transportadora, pois mesmo sendo hábil para realizar entregas em terminais de melhor qualidade de serviço ou mais próximos, deve seguir as instruções dos varejistas e entregar no serviço atrelado a eles.

Ambas as situações descritas acima são danosas para o negócio. A consequência disso é a instalação da vários terminais concorrentes na mesma localidade, um para suprir as necessidades das transportadoras e outra para os varejistas. Essa prática custa espaço alocado, não é eficiente e necessita de mais capital injetado.

Portanto, o cenário ideal é onde não há limitações de transportadoras ou varejistas, assim, até mesmo outros negócios independentes conseguem utilizar os terminais. Além disso, não possuiriam amarras aos clientes, os mesmos poderiam optar por diferentes serviços para contratar. Como consequência, a taxa de utilização dos serviços poderia ser elevada e seria eliminado o problema de possuir mais de um terminal instalado na mesma localidade.

Além do sistema de comércio *B2C*, algumas empresas realizam *C2C*, como DHL PackRstation[20] (Alemanha), Dognposten[21] (Dinamarca) e a Zhilai[22] (China), anteriormente chamada de Webox. Um grande exemplo do sistema *C2C* no Brasil é o Mercado Livre, ao permitir que um usuário anuncie um produto novo ou semi-novo e o venda para outro usuário do site. Este é um cenário muito perigoso em uma solução de armários inteligentes, principalmente em países onde a expectativa e/ou qualidade de vida não é elevada. Ao contrário deste exemplo do Mercado Livre, onde pode-se aferir que todos produtos são legais e adequados, os terminais que permitem *C2C* não podem garantir qual conteúdo foi depositado no compartimento, fato que abre espaço para atividades ilegais, tais como tráfico de drogas, armamentos, ou até mesmo atentados terroristas.

Ao vislumbrar o futuro, um cenário *C2C*, ou até mesmo uma simples entrega de objetos entre usuários, sem transação monetária, pode ser sim uma alternativa interessante. Mas para isso, é necessário que a segurança seja reforçada, devido à necessidade de checagem do pacote antes de chegar ao compartimento, alternativa semelhante ao *check-in* de aeroportos.

2.5. Análise do Estado da Arte

Conforme elicitado, existem diversos produtos e modelos de negócio que podem ser elaborados a partir de uma solução de armário inteligente. Dentre os listados, Sa-ngiampak et al. (2019) e Alqahtani et al. (2020) propõem soluções semelhantes, pois em ambos, o público alvo são os estudantes de universidades, que são locais de grande circulação de pessoas, atingindo uma faixa etária mais jovem que é mais suscetível ao uso de novas tecnologias, principalmente quando elas são capazes de resolver seus problemas. Ademais, David e Chalon (2018) descreve uma solução com foco no armazenamento de alimentos, onde faz-se necessária a refrigeração ou aquecimento dos compartimentos, fato que causa

[20] www.dhl.de
[21] *www.postnord.dk*
[22] *www.smartelocker.com*

maiores gastos com energia elétrica e com infraestrutura. Já Chikara et al. (2020) propõe uma solução totalmente voltada à segurança de armazenamento, possuindo 3 camadas de bloqueio, fato que permite a circulação de unidades monetárias e produtos de alto valor dentro de seus compartimentos, porém ao mesmo tempo, acaba por tornar-se muito complexo e oneroso para uma solução de uso da grande massa populacional.

Dentre as funcionalidades disponibilizadas pelas empresas de armários inteligentes, pode-se citar duas essenciais para o funcionamento de uma solução do ramo, são elas: a necessidade de código ou senha para desbloquear o compartimento; e de uma solução eficaz para fornecimento de conexão com a Internet. Dentre as empresas avaliadas, as mais variadas funcionalidades podem ser listadas, como rastreamento do pacote, notificações ao cliente, câmeras de segurança nos armários, formas de pagamento *in loco* e diversos tamanhos de compartimentos. É importante destacar a relevância da adição de novas e variadas funcionalidades ao produto, pois este é o principal diferencial que pode ser agregado à solução.

Quanto ao modelo de negócio utilizado, assume-se que o ideal é o modelo sem restrições para com as transportadoras ou varejistas, devido aos motivos supracitados. Tal modelo que também será adotado na concepção do *ServicesLocker*, melhor detalhado na seção 3.1.

3. ServicesLocker

Este trabalho apresenta o *ServicesLocker*, uma solução de armário inteligente baseada no modelo de arquitetura de soluções IoT de 5 camadas. A sua concepção nasce com o objetivo de introduzir um nova forma de utilização para este gênero logístico: com o propósito de facilitar e flexibilizar o processo envolvido em uma ordem de serviço de lavação de roupas por terceiros (lavanderias). Além disso, o seu desenvolvimento será flexibilizado ao ponto de permitir seu uso padrão, assim como já executado por diversas empresas ao redor do mundo, isto é, permitirá a entrega de pacotes por empresas logísticas aos seus clientes de destino.

Nesse sentido, pode-se abstrair e simplificar estes dois processos por meio de dois infográficos. A Figura 2 apresenta as 5 etapas de um processo de lavação de roupas. A etapa 1 representa a realização de uma ordem de serviço, habilitando a opção de abertura de compartimento para depósito das roupas; na etapa 2, o usuário deposita as roupas no armário e a opção de abrir armário é habilitada para a retirada pela lavanderia contratada; a etapa 3 representa a execução da ordem de serviço, enquanto notifica o progresso da mesma ao usuário; na etapa 4, a lavanderia devolve as roupas ao armário para retirada do cliente; finalmente, a etapa 5 representa a retirada das roupas pelo cliente e a finalização da ordem de serviço.

Por outro lado, a Figura 3 apresenta as 4 etapas de uma ordem logística de entrega de encomendas. A etapa 1 representa a realização de uma compra online pelo cliente, optando pelo *ServicesLocker* como forma de entrega; na etapa 2, a empresa logística deposita o pacote no terminal de armários inteligentes; a etapa 3 representa a notificação ao cliente do depósito da encomenda no devido armário, e a opção de abrir o compartimento é habilitada; finalmente, na etapa 4, o cliente efetua a retirada da encomenda e a ordem é finalizada.

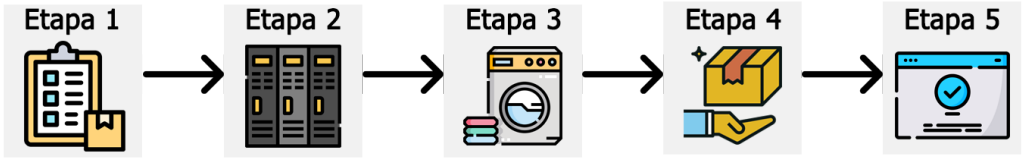

Figura 2. Etapas de uma ordem de lavação de roupas.

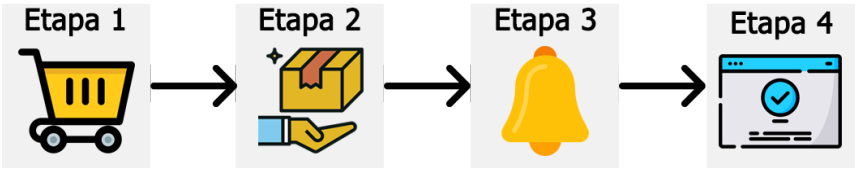

Figura 3. Etapas de uma ordem logística.

3.1. Modelo de Negócio

A solução através de um armário inteligente, apresentada acima de forma geral, surge com a problemática da falta de tempo das pessoas em realizarem tarefas corriqueiras do dia a dia, como lavar roupas.

Para uma melhor compreensão da proposta, o modelo de negócios do *Services-Locker* é apresentado no formato de um CANVAS disponibilizado no Apêndice 1.

A proposta de valor visa facilitar a vida das pessoas que não dispõem de tempo para realizarem tarefas corriqueiras do dia a dia, mais especificamente, lavar roupas, mas não se limitando a isto, assim, podendo se estender a diversos outros serviços como a entrega de produtos, encomendas ou outros serviço que possam se beneficiar de um espaço do armário.

Assim, a solução proposta neste artigo visa facilitar o serviço de entrega e recolha de objetos otimizando o tempo gasto e a flexibilização de horário e redução de custo da última milha. Logo, as empresas de lavanderia podem se beneficiar do armário para guardar as roupas em uma localidade próxima ao cliente. Assim, evitando que o mesmo gaste tempo para buscar as roupas na lavanderia.

Neste contexto, abre-se algumas vantagens também para o lado dos prestadores de serviços quanto ao uso desta solução como a redução de custos por parte das empresas que oferecem serviços de lavação, pois haveria uma redução na quantidade de espaço para guardar as roupas, além de horário hábil disponível para que o cliente busque as roupas quando bem entender. Quando se fala na extensão para serviços de entrega, há o lado logístico onde as transportadoras parceiras deste tipo de serviço poderiam reduzir os gastos com combustível, visto que estas não precisariam voltar até o local para realizarem uma segunda tentativa de entrega, no caso da primeira tentativa a pessoa não estar no local para receber a encomenda. Assim, possibilitando a criação de uma logística mais

simplificada.

Em relação as vantagens para com as empresas concorrentes, pode-se citar que a principal vantagem desta solução é a diversificação de serviços prestados na plataforma. Visto que as empresas com propostas similares como a Amazon, prende seus clientes aos produtos vendidos em sua plataforma, não abrindo espaço para outras empresas atuarem. Já outras empresas trabalham com uma quantidade de serviços limitada, senão um único serviço. A proposta do Services Locker é expandir os serviços prestados na plataforma para qualquer empresa que pode se beneficiar dos espaços nos armários nas localidades em que estão.

Na parte de segmento de clientes, ou seja, o perfil de clientes em que a solução deseja atingir, destacam-se as empresas que prestam serviço de lavação e entrega de mercadorias, assim como os clientes das mesmas. Pessoas com o estilo de vida atarefado com pouco tempo disponível. Habitantes de grandes metrópoles e empresas de venda direta, como lojas físicas ou de *e-commerce*.

Com relação a estratégia para alcançar clientes (canais) pode-se citar representantes de vendas e canais de divulgação como redes sociais, anúncios onlines e físicos.

As métricas chaves utilizadas para medir o sucesso da solução são:

1. Número de reservas feitas por usuário durante um determinado período: servirá para medir a frequência de uso por usuário e tomar ações baseadas nisto;
2. Quantidade de serviços e empresas disponíveis na plataforma, visto que pode-se ter várias empresas provendo um mesmo tipo de serviço e também diversos serviços. Logo, de acordo com as necessidades existentes e que se encaixam na estrutura de negócio, poder ampliar esta rede de serviços, assim, fornecendo mais opções para os usuários existentes e atraindo novos perfis de usuários para utilizarem da solução.
3. Relação por armário: preço de custeio para manter o armário com o lucro gerado pelo mesmo no mês.

A estrutura de custos também é algo muito importante do modelo de negócios, pois é uma base que será utilizada para medir o mínimo que a solução deve alcançar para que dê lucro. Em razão disso, pode-se destacar:

- Custos fixos:
    1. aluguel do espaço em que o armário será alocado;
    2. servidor de hospedagem para manter o serviço online;
    3. salário dos empregados;
    4. impostos;
    5. serviço de limpeza dos armários.
- Custos variáveis:
    1. conta de energia utilizada pelos armários;
    2. reparo dos armários no caso de extravio;
    3. instalação de novos armários.
- Fontes de renda:
    1. Taxa fixa mensal para as empresas prestadoras de serviço estarem disponíveis na plataforma.

2. Taxa de uso e empréstimo (aluguel): O usuário será cobrado conforme realização de um pedido. Essa equação irá englobar uma porcentagem sobre o valor total do pedido + taxa diária de uso do espaço no armário;
3. Anúncios em geral, divulgação de empresas parceiras e relacionados aos serviços providos, ou seja, os serviços cadastrados na plataforma poderão fazer anúncio de promoções e de seus produtos na aplicação.

Ainda no quesito de fontes de renda, serão utilizadas métricas como o *Life Time Value* (LTV) cujo os resultados são usados para ajustar o valor cobrado e aumentar o tempo de permanência dos clientes. Junto a isso, se fará o uso do *Break Even Point* (BEP) utilizado para manter um controle sobre os custos do serviço, projetando o tempo para manter o equilíbrio entre gastos e o os lucros da empresa (Figura 4). Este, por sua vez, toma como base de cálculo a receita e lucros da empresa para encontrar o equilíbrio.

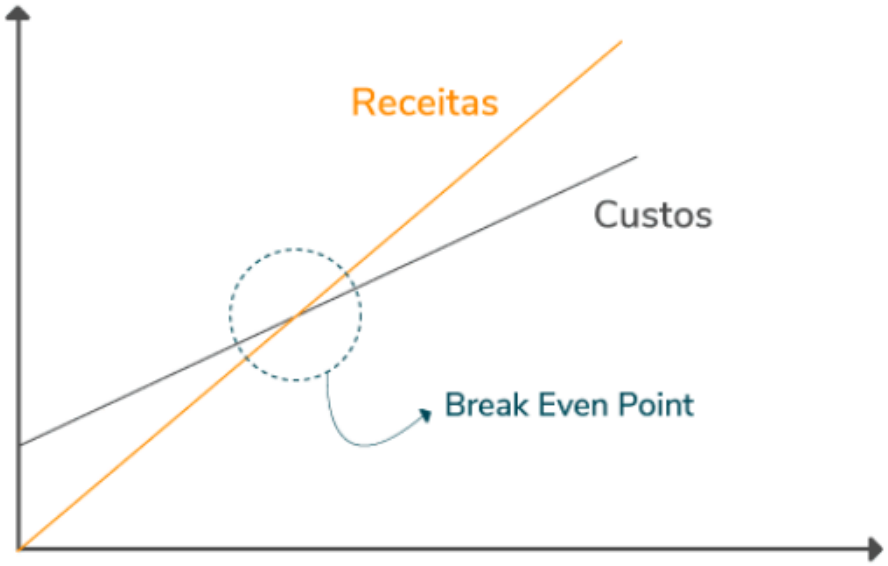

Figura 4. Break Even Point (Retorno, 2020).

Por fim, a margem bruta que é calculada para obter uma porcentagem de lucro sobre cada cada armário instalado. Por exemplo, se um armário custa mais caro para mantê-lo em determinado local, deve-se cobrar mais pela utilização dele para obter uma margem de lucro.

3.2. Arquitetura IoT

Nesta seção será apresentada a arquitetura da solução proposta, baseada em um modelo de arquitetura de 5 camadas.

3.2.1. Camada de Percepção

A Figura 5 apresenta o *hardware* utilizado na construção da solução proposta. Este, por sua vez, é composto de 5 partes:

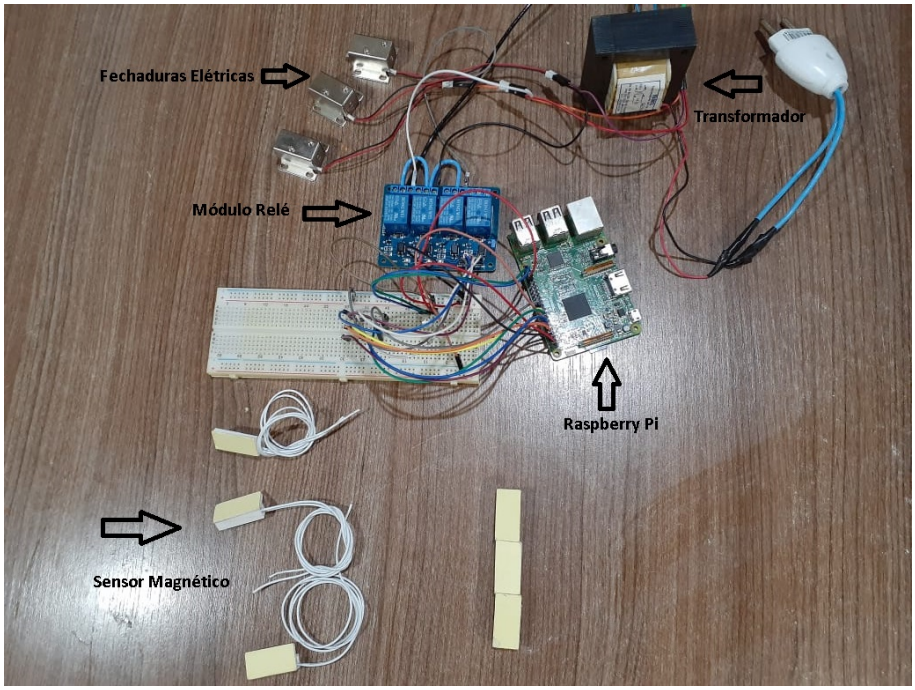

Figura 5. Camada de Percepção.

1. Raspberry Pi
   Responsável por realizar toda a lógica de operação do armário. Foi escolhido por ter uma ampla gama de interfaces de comunicação (*WiFi*, *Ethernet*, *Bluetooth* e USB), alto poder de processamento bem como a possibilidade de se utilizar um sistema operacional de forma a facilitar a implementação do seu *firmware*. No contexto deste trabalho, o Raspberry Pi é responsável por conectar-se ao servidor via *Ethernet* ou *WiFi*, receber as informações vindas do servidor e então executar a ação de liberar ou fechar uma porta do armário. Também é responsável por monitorar a situação das portas (se estão abertas ou fechadas) e enviar esta informação ao servidor;

2. Fechaduras elétricas
   As fechaduras das portas do armário são solenoides elétricas alimentadas em uma tensão de 12V. Serão controladas pelo Raspberry Pi através de um módulo relé;

3. Módulo relé
   Responsável por controlar o acionamento das fechaduras elétricas. Para este protótipo será utilizado um módulo com 4 relés o qual será conectado ao Raspberry Pi através de portas digitais (GPIO);

4. Sensor Magnético
   O sensor magnético de fim de curso é responsável por obter a informação do estado atual da porta (aberta ou fechada). Serão conectados nas portas digitais (GPIOs) do Raspeberry Pi. Este sensor é composto de duas partes, sendo uma fixada na parte móvel da porta e outra fixada na parte fixa do armário;

5. Transformador
   Responsável por transformar a tensão fornecida pela rede elétrica (110V ou 220V) para a tensão de operação interna da solução (12V). Esta tensão é utilizada para alimentar o microcontrolador e também para acionar as fechaduras elétricas instaladas nas portas do armário;

3.2.2. Camada de Rede

Os componentes de *hardware* supracitados, referentes à camada de percepção, necessitam de uma forma para se comunicarem com a infraestrutura da aplicação, isto é, precisam de alguma conexão com a Internet. Existem diversos protocolos diferentes de comunicação, tais como: *Sigfox, LoRa, Bluetooth, ZigBee* e *NB-IoT*. No contexto do *ServicesLocker*, o protocolo escolhido foi o IEEE 802.3, conhecido como *Ethernet*.

O *Ethernet* é um dos protocolos de comunicação mais comuns e usados no mundo. Juntamente com o WiFi (IEEE 802.11), encontra-se presente na maioria das residências da população através de portas disponibilizadas em APs (*Access Points*). Esse protocolo foi oficializado em 1983 e tornou-se o padrão para utilização em LANs. Ele possibilita altas taxas de transferência de dados, é ideal para dispositivos fixos e seu fornecimento se dá por meio de cabeamento par trançado ou fibra óptica.

A maioria das soluções de armários inteligentes implementadas, se dão pela instalação de terminais em locais fixos das cidades. Por esta razão, torna-se simples a instalação de uma conexão cabeada (LAN) em um terminal, sendo a sua implantação sempre preferida a outras abordagens.

Por outro lado, nem sempre a comunicação cabeada poderá ser implementada, devido a complicações do local que o terminal está posicionado. Nesses casos, opta-se pela utilização do IEEE 802.11, mais conhecido como *WiFi*, padrão *de facto* para as WLANs[23]. O *WiFi* permite uma alta taxa de transmissão de dados e opera na faixa dos 2.4GHz, 5GHz e 60GHz. Sua desvantagem em relação ao *Ethernet* é a limitação na área de cobertura e o alto consumo de energia. Este último não se apresenta como limitação para a solução proposta uma vez que, como discutido anteriormente, a alimentação do sistema será feita através de cabeamento direto na rede elétrica, não sendo necessário o uso de baterias.

3.2.3. Camada de Middleware

No modelo de 5 camadas, a camada de *middleware* caracteriza-se por ser uma abstração intermediária entre as camadas de comunicação e de aplicação responsável pelo gerenciamento de serviços, processamento de informações e conexão com o banco de dados, além de permitir a realização da computação ubíqua. Muitas vezes, essa abstração de *middleware* é conhecida como plataforma IoT: um software que permite a interconexão dos diversos dispositivos de *hardware* envolvidos na aplicação em questão.

No contexto deste trabalho, a plataforma IoT escolhida foi o *ThingsBoard*[24], a

[23] *Wireless Local Area Network*, ou Rede de Área Local Sem Fio.
[24] `https://thingsboard.io/`.

qual apresenta-se como uma das conhecidas e utilizadas plataformas IoT de código aberto (*open-source*) do mercado. Esta, por sua vez, permite a coleta de dados, processamento, visualização e gerenciamento de dispositivos. Algumas das funcionalidades disponíveis são citadas abaixo:

- compatibilidade com os protocolos MQTT, CoAP e HTTP;
- encriptação para MQTT e HTTP, autenticação e gerenciamento de credenciais;
- escalabilidade horizontal;
- tolerância a falhas;
- personalização da visualização de dados;
- suporte a microsserviços;

Outra importante funcionalidade disponibilizada pelo *ThingsBoard* é um conjunto de APIs (*Application Programming Interface*) que fornece um interface intermediária que possibilita a diferentes aplicações se comunicarem entre si, possibilitando a troca de informações entre o *back-end* do *ServicesLocker*, *front-end* e dispositivos físicos (através do *ThingsBoard*). Assim, por exemplo, a aplicação poderá disparar um evento de abertura de porta do armário que, através da comunicação com o *ThingsBoard*, propagará o evento até o *hardware*.

Ademais, a plataforma também fornece suporte a diversos bancos de dados, possibilitando a escolha do local de armazenamento de dados de telemetria[25] e do local de armazenamento dos dados das entidades do sistema. Na prática, os dados de telemetria são armazenados em um banco de dados NoSQL, mais especificamente o Cassandra[26](banco de dados NoSQL padrão do *ThingsBoard*). Enquanto que os dados das entidades (ex.: cadastro de usuários), são armazenados em um banco de dados SQL, mais especificamente o PostgreSQL[27]. Desta forma, o *ServicesLocker* utilizará uma abordagem híbrida de banco de dados, tanto SQL quanto NoSQL.

3.3. Camada de Aplicação

Para que o *ServicesLocker* seja implementado, é necessário que se estabeleça um padrão de comunicação entre os dispositivos físicos e o *back-end*, que por sua vez, repassa as informações para o *front-end*. No contexto deste trabalho, será usado o protocolo de comunicação *Message Queuing Telemetry Transport* (MQTT), o qual se baseia no conceito do padrão de projeto de *software* chamado *observable*, onde há um desacoplamento da ideia clássica de cliente/servidor.

Nesse sentido, o protocolo opera com publicadores (*publishers*) e inscritos (*subscribers*) em um determinado tópico ou assunto. As duas partes não tomam conhecimento uma da outra, portanto, um terceiro componente chamado *Broker* é responsável por gerenciar estes eventos e saber para onde enviar cada informação. O *ServicesLocker* utilizará o *Broker* fornecido pela plataforma *ThingsBoard*.

Na Figura 6, é apresentado um exemplo do modelo *publish/subscribe*: um carro (*publisher*) envia informações ao *Broker* sobre sua velocidade. O *back-end* da aplicação,

[25]Dados gerados pelo *hardware* que são transmitidos ao *ThingsBoard*, como os valores dos sensores, por exemplo.

[26]`https://cassandra.apache.org/`.

[27]`https://www.postgresql.org/`.

juntamente a um dispositivo móvel, estão inscritos no tópico velocidade. Portanto, toda vez que houver uma publicação ou envio de informação sobre velocidade por parte do carro, o *back-end* e o dispositivo móvel receberão esta informação.

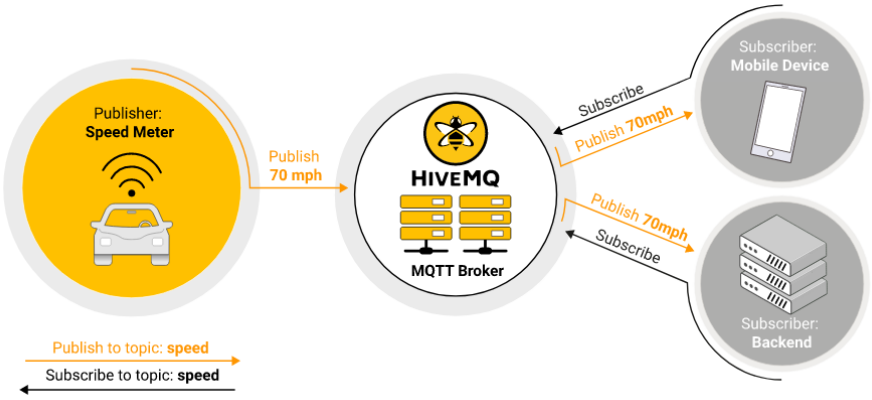

Figura 6. Comunicação MQTT (HiveMQ, 2015).

A decisão da utilização do protocolo MQTT é devido a sua baixa latência de banda de internet e segurança dos dados, visto que a encriptação dos dados é feita por meio de AEAD[28] (Sadio et al., 2019). Os dados serão encapsulados em arquivos JSON[29] para manter um padrão no formato da transferência de dados entre as diferentes partes do sistema. O *back-end* do *ServicesLocker* consumirá a API do *ThingsBoard* citada na seção 3.2.3, além também de se comunicar com a aplicação *front-end* para buscar e enviar os dados necessários através de JSON.

3.4. Camada de Negócios

Após a descrição do sistema IoT seguindo a arquitetura de 5 camadas, a última delas e também uma das mais importantes para qualquer solução que objetiva o lucro, é a camada de negócios. As seções 3.5 levantam tópicos acerca do *ServicesLocker* com suas ferramentas e funcionalidades mínimas para a criação de um MVP funcional, abordando seus fluxos de operações, lógica de negócio, requisitos funcionais e não funcionais, modelagem conceitual de banco de dados e casos de uso.

3.5. Requisitos Funcionais e Não Funcionais

Abaixo são listados 16 requisitos funcionais para o MVP do *ServicesLocker*, a ficha de detalhamento de cada um destes encontra-se no Apêndice 2:

- **RF 01** - Cadastro de pessoa;
- **RF 02** - Edição de pessoa;
- **RF 03** - Exclusão de pessoa;
- **RF 04** - Busca de pessoa;
- **RF 05** - Realização de *login* no sistema;
- **RF 06** - Criação de pedido/ordem de serviço;

[28] *Authenticated Encryption with Associated Data*, ou Criptografia Autenticada com Dados Associados
[29] *JavaScript Object Notation*

- **RF 07** - Edição de pedido/ordem de serviço;
- **RF 08** - Cancelamento de pedido/ordem de serviço;
- **RF 09** - Busca de pedido/ordem de serviço;
- **RF 10** - Atualização do estado do pedido/ordem de serviço;
- **RF 11** - Seleção de formas de pagamento;
- **RF 12** - Simulação de um pedido/ordem de serviço;

3.6. Modelagem Conceitual de Banco de Dados

O modelo conceitual do banco de dados da solução *ServicesLocker* é apresentado na Figura 7. Como descrito na seção 3, o desenvolvimento da solução proposta será flexibilizado para permitir dois tipos de fluxos de serviço, são eles: i) serviços de única via (cliente-fornecedor), como entregas de pacotes por empresas de logística, por exemplo; ii) serviços de duas vias (cliente-fornecedor-cliente), onde o cliente tem de entregar algo ao fornecedor para que ocorra a prestação de serviço, como por exemplo serviços de farmácia, cartório ou chaveiro. Para representar este comportamento, foi utilizado um relacionamento de generalização e especialização, dessa forma, a entidade genérica `Pedido` possui duas entidades especializadas (`Duas vias` e `Via unica`) que definem o escopo de serviço oferecido pela aplicação. Ainda, nos pedidos de duas vias, a entidade `Objeto` representa a discriminação do conteúdo depositado no compartimento, sejam estes roupas para lavanderias, ou documentos para cartórios, por exemplo.

A entidade `Pessoa` abstrai o conceito de cliente e de fornecedor, de forma que no cadastro de usuário seja possível escolher entre ser apenas um cliente ou então um cliente/fornecedor. Isso é feito através da entidade `Tipo` e `Serviço`, onde ”Serviço”representa qual serviço determinada pessoa fornece. Por sua vez, a entidade ”Tipo”pode assumir três valores no sistema, cada qual referenciando um ator diferente, são eles: comum, *e-commerce* ou transportadora.

- Comum: As pessoas (cliente/fornecedor) do tipo comum são prestadoras de serviços (lavação, cartório, entre outros), podendo ofertar 1 ou *n* serviços na plataforma.
- *E-commerce*: As pessoas (cliente/fornecedor) do tipo *e-commerce* oferecem aos seus clientes em seu *website* de vendas uma opção de entrega através do *ServicesLocker*. No cenário e modelo idealizado, o *e-commerce* envia, através de conexão com a API do *ServicesLocker*, o *e-mail* do cliente cadastrado em nossa plataforma e a localização do armário de depósito da encomenda desejada, juntamente com quaisquer outras informações do pedido que necessitarem ser enviadas. Após esta etapa, uma ordem de entrega é criada na conta do cliente com o *e-mail* correspondente, possibilitando ao cliente final o acompanhamento do processo pelo *ServicesLocker*.
- Transportadora: são pessoas (cliente/fornecedor) contratadas pelo *e-commerce* parceiro e são as responsáveis por realizar os depósitos dos pacotes nos armários. As transportadoras possuem uma tela de visualização com suas ordens de entregas pendentes, contendo o armário de entrega, número do compartimento e o *e-mail* do cliente (para associar com o *e-mail* escrito na etiqueta de transporte do pacote, e então tomar conhecimento de qual encomenda deve ser depositada no compartimento em questão).

- Na hipótese da inexistência de relacionamento entre ”Pessoa”e ”Tipo”(*null*), considera-se que a pessoa assume apenas o papel de cliente no sistema, não podendo fornecer serviços.

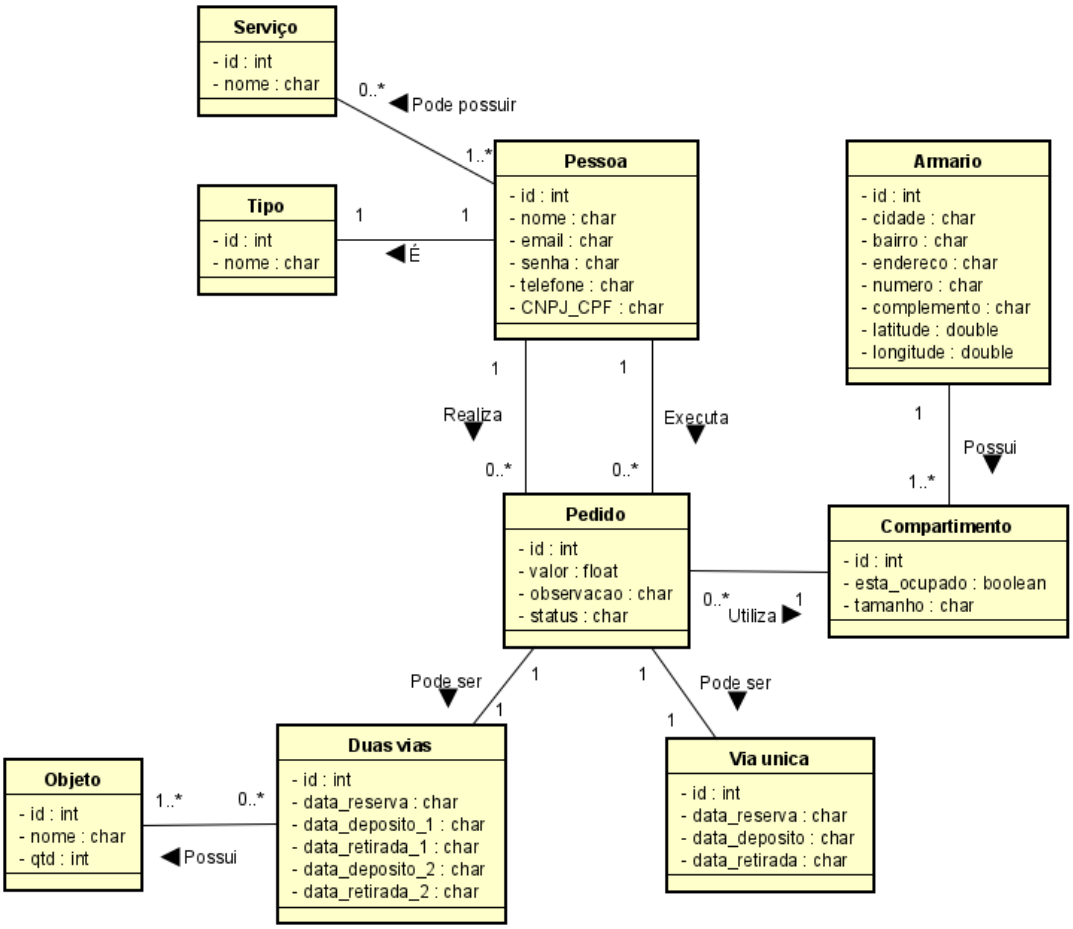

Figura 7. Modelagem Conceitual do Banco de Dados.

3.7. Casos de Uso

Como citado na seção 3.6, a entidade `Pessoa` pode atuar tanto como fornecedora de produtos e/ou serviços quanto como o cliente final. No caso em que esta atua como fornecedora de produtos e/ou serviços, pode possuir até três atores, representados nas figuras 8 e 9. Além desses, o sistema possui um quarto ator, que é o cliente final, também representado pela entidade `Pessoa`, cujo caso de uso é representado na figura 10.

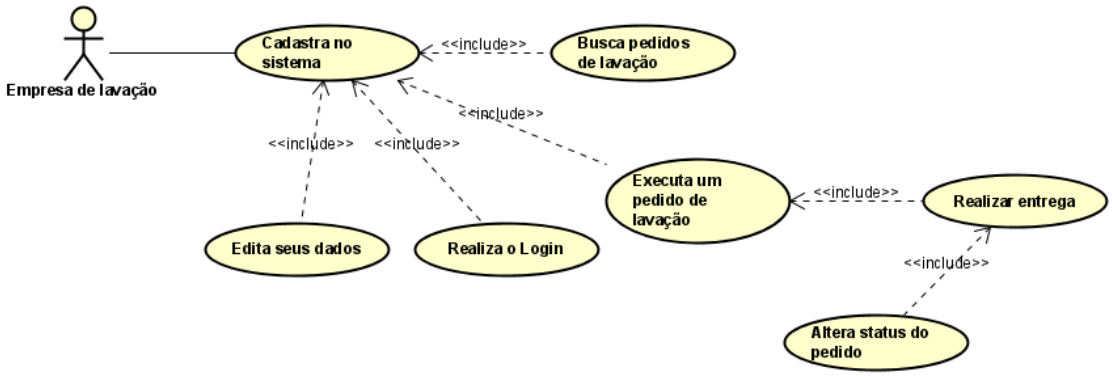

Figura 8. Caso de Uso - Empresa de Lavação Afiliada.

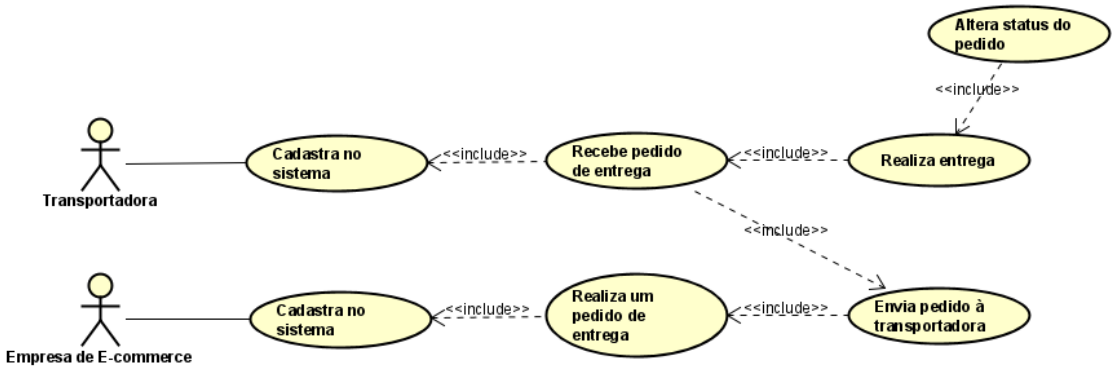

Figura 9. Caso de Uso - Entregas de Pedidos Feitos por *e-commerce*.

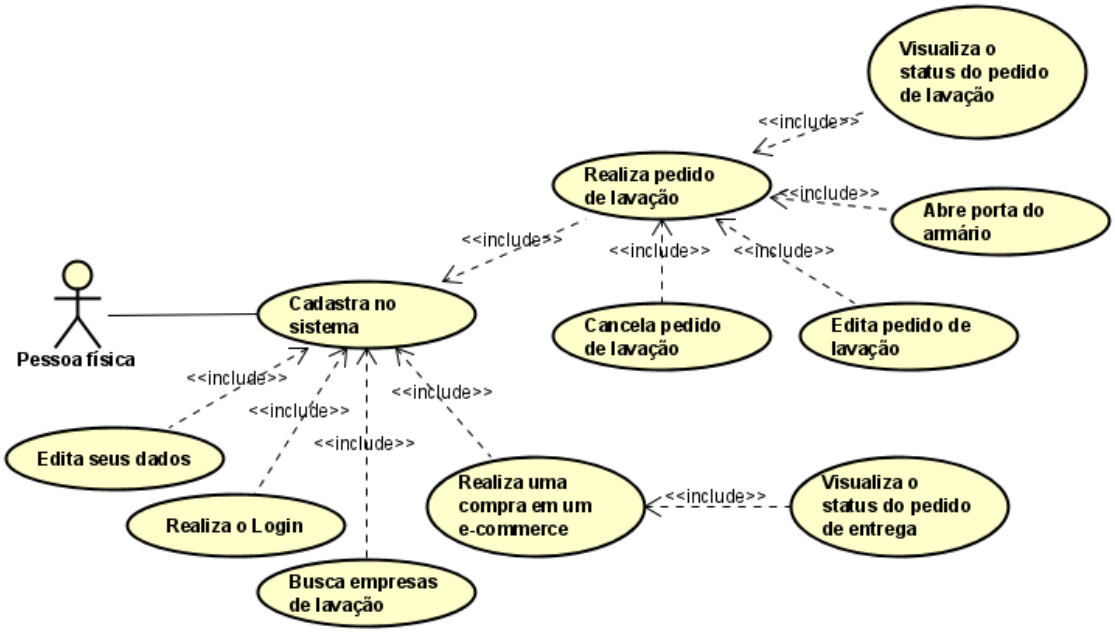

Figura 10. Caso de Uso - Cliente Pessoa Física.

3.8. Front-End

Responsável por exibir uma interface gráfica que possibilite a interação com o usuário. O *front-end* é o responsável por receber os dados de entrada do usuário e apresentar o estado atual da aplicação.

A arquitetura para a exibição e entrada de informação foi desenvolvida através do *HyperText Markup Language* (HTML), juntamente com *Syntactically Awesome Stylesheets* (SCSS) para estilizar as páginas e componentes utilizados. Estes, por sua vez, foram desenvolvidos utilizando-se o Adobe XD, a qual é uma ferramenta usada para prototipação de telas e componentes gráficos de uma aplicação.

Em conjunto com o SCSS, fora utilizado a metodologia de nomenclatura BEM para manter um padrão de nomes em classes SCSS. Para dar agilidade ao processo de construção da aplicação foi utilizado o *framework* Angular, o qual é mantido pela Google e fornece um conjunto de utilitários que permitem a produtividade na hora de gerenciar os módulos da aplicação, como por exemplo, o sistema de navegação entre páginas, identificação de eventos de mudanças, até módulos que permitem a interação com o *back-end*. Junto ao Angular, foi utilizado a linguagem de programação *Typescript*, responsável por gerenciar toda lógica de funcionamento das telas da aplicação. Além disso, foi utilizado o Angular Material, que é um pacote de componentes gráficos.

O Angular caracteriza-se por ser um *framework* voltado para a criação de *websites* com o conceito de *Single Page Application*(SPA), ou seja, Aplicação de Única Página. O conceito está baseado na dispensabilidade do redirecionamento para uma nova página ou de recarregamento da página a cada mudança do estado da aplicação. Nesse cenário, toda a estrutura da página é estática, onde apenas o conteúdo principal é atualizado através de um *back-end*.

Por fim, para gerenciar as dependências da aplicação, fora utilizado o gerenciador de pacotes *Node Package Manager* (NPM), e o sistema de controle de versões Git, que possibilita o versionamento do código, além de facilitar o trabalho em time por decorrência de alterações e conflitos de código.

No Apêndices encontram-se as telas desenvolvidas, que permitem que o fluxo de jornada do usuário dentro da aplicação seja realizada.

- Figura 24: representa a tela de login, que é a página inicial da aplicação. Nela, é possível acessar as funcionalidades da aplicação através do preenchimento da tupla de *e-mail* e senha. Caso o usuário não tenha cadastro, é possível se cadastrar através do clique no link ”cadastre-se”.
- Figura 26: representa a tela que permite a criação de um novo usuário no *ServicesLocker*. A tela de cadastro possui duas abas, que permitem o usuário alternar entre cadastrar uma conta pessoal (somente consumirá serviços disponíveis na plataforma) ou uma conta empresarial que fornecerá serviços na plataforma. A figura 25 representa a aba com o formulário dos dados que devem ser preenchidos em um cadastro pessoal.
- Figura 27: representa a tela de criação de pedidos para um serviço de lavanderia. Nesta tela, é possível selecionar qual empresa de lavação realizará o pedido. Junto a isto, o usuário pode adicionar qual o tipo e a quantidade de uma determinada peça de roupa que constará no pedido. Por fim, o usuário pode colocar uma

observação no pedido e enviá-lo.

- Figura 28: representa a página inicial da aplicação (quando o usuário já está logado). Nela, é apresentado os pedidos realizados pelo usuário, bem como o acompanhamento do progresso deles.
- Figura 29: representa o menu lateral, onde são apresentadas as opções disponíveis ao o usuário quando efetua o clique no ícone de menu, no canto superior esquerdo da tela. As opções apresentadas na Figura **??** são referentes a um usuário que já está autenticado na aplicação.

3.9. Back-End

No lado do servidor, possuímos uma aplicação que disponibiliza e fornece todos os dados e estados do sistema que o *front-end* requisitar, através de chamadas HTTP (*Hypertext Transfer Protocol*). O modelo de arquitetura tomado como base é o *Model View Controller* (MVC), ou seja, as *Models* mapeiam cada tabela criada no banco de dados, conforme Figura 7, e os *Controllers* são responsáveis pelo tratamento dos dados vindos das requisições, lógicas de negócio, inserções e atualizações no banco de dados e resposta para o *front-end*.

Como citado na seção anterior, o *front-end* caracteriza-se por ser uma SPA, fato que muda o conceito de *Views* em um modelo MVC. Nesse caso, o *back-end* é responsável por enviar apenas o conteúdo em formato JSON, para que o *front-end* utilize-o da forma que quiser. Essa abordagem do envio de dados por JSON difere de alguns *frameworks web*, como por exemplo PHP[30] ou Ruby on Rails[31], que enviam um conteúdo HTML novo a cada requisição feita pelo cliente, fato que exige o recarregamento da página. Em linhas gerais, as *Views* do MVC são substituídas pelo Angular, responsável pela montagem dos elementos em tela.

Quanto às tecnologias utilizadas, temos: *Javascript* e *Typescript* como linguagens de programação, o *framework Node.js* para a construção do modelo MVC e do *back-end*, *PostgreSQL* como banco de dados e a biblioteca *Knex.js*[32], um *query builder* em *Javascript* para SQL, que facilita a conexão com o banco de dados, bem como seu versionamento (através de *migrations*) e criação de *Models*. Ainda foi utilizada a biblioteca do *Thingsboard* para *Node.js*, servindo como *Middleware*, e possibilitando a comunicação do *hardware* com o *back-end* para o destravamento dos compartimentos, por exemplo.

O Knex aborda um conceito de versionamento de banco de dados, o qual é baseado em *migrations*, isto é, para cada alteração de tabela ou de relacionamento no banco de dados, deve ser criada uma nova *migration*. A Figura 11 mostra todas as *migrations* criadas para o *ServicesLocker*, cada uma é responsável pela criação de uma tabela. Em um cenário hipotético de um sistema em uso, onde se deseja alterar a tipagem de uma coluna, por exemplo, deve-se criar uma nova *migration* responsável apenas por esta alteração, desta forma, mantemos todas as versões do banco de dados ao longo do desenvolvimento do sistema, sendo possível retornar em um ponto específico caso necessário.

A Figura 12 mostra o conteúdo da *migration 01_create_pessoa*, onde é feita a criação da tabela pessoa através de código *Javascript*. Já a Figura 13 representa um

[30] `https://www.php.net/`
[31] `https://rubyonrails.org/`
[32] `http://knexjs.org/`

método disponível no *controller* de Usuários, o qual recebe uma requisição e retorna uma resposta ao *front-end*. Nesse caso, um comando SELECT é executado através do Knex na tabela ”pessoa”, realizando junções com a tabela ”service”e com a tabela intermediária ”provider_service”.

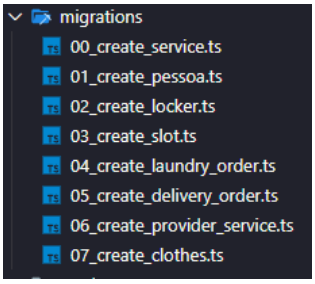

Figura 11. Migrations

```
import Knex from 'knex'

export async function up(knex: Knex) {
    return knex.schema.createTable('pessoa', table => {
        table.increments('id').primary()
        table.string('nome').notNullable()
        table.string('CNPJ_CPF').notNullable()
        table.string('fone').notNullable()
        table.string('tipo')
        table.string('email').notNullable()
        table.string('senha').notNullable()
    })
}
```

Figura 12. Criação da tabela pessoa

```
import db from "../database/connection"
async providerIndex(_req: Request, res: Response) {
    const fornecedores = await db('pessoa')
        .join('provider_service', 'provider_service.provider_id', '=', 'pessoa.id')
        .join('service', 'service.id', '=', 'provider_service.service_id')
        .select('id', 'nome', 'CNPJ_CPF', 'fone', 'tipo', 'email')
        .where('tipo', '=', 'comum')

    return res.json(fornecedores)
}
```

Figura 13. Consulta SQL utilizando *Knex.js*

4. Testes e Resultados

A aplicação foi avaliada em nível qualitativo, em aspectos que contemplam desde flexibilidade e usabilidade até custo de implantação de uma unidade do *ServicesLocker*. Em um primeiro momento, buscou-se a viabilização da solução proposta, e para isso, testes de integração foram realizados, são eles: integração entre o *back-end* e *front-end*, onde podem ser realizadas ações como cadastro de usuário, login na aplicação, criação de pedido de lavanderia com alocação em um compartimento existente, listagem de pedidos tanto por parte de cliente quanto fornecedor, e por fim, listagem de empresas disponíveis para realização da ordem de serviço.

Para simular a criação de um pedido de entrega, foi utilizado o Insomnia[33], um cliente que consegue realizar chamadas HTTP e obter uma resposta. Essa simulação tem como objetivo exemplificar o cenário em que um *e-commerce* parceiro consuma a API do *ServicesLocker* para criação de um pedido de entrega. A Figura 14 mostra à esquerda os dados que o cliente envia (peso da encomenda, e-mail do destinatário, ID do fornecedor, ID do cliente e o tamanho do compartimento). À direita é mostrado a resposta que o servidor devolveu ao cliente (data da reserva, ID do compartimento, entre outros). No topo, pode ser observado a URL requisitada e o método HTTP utilizado (POST).

[33] *https://insomnia.rest/*

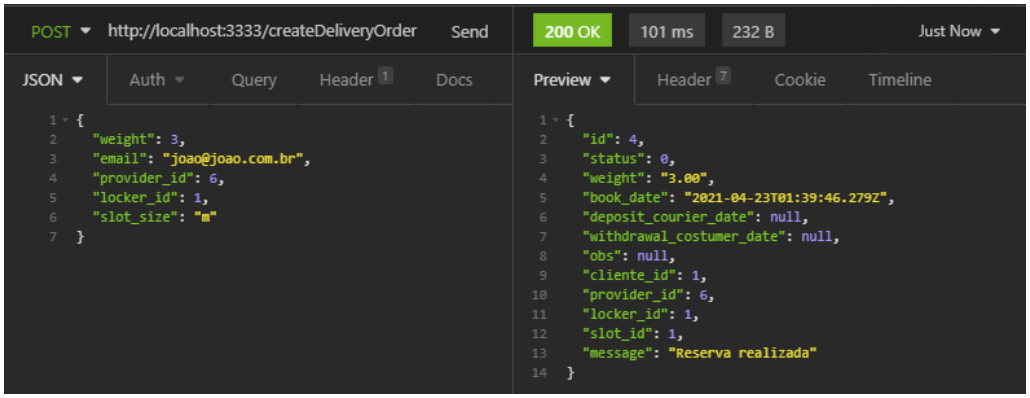

Figura 14. Criação de Pedido de Entrega

O próximo teste realizado foi a integração do *back-end* com o *Thingsboard*. Neste caso, era necessária que uma chamada RPC[34] fosse realizada para que a abertura da fechadura eletrônica seja realizada. O cenário em questão também foi exemplificado através do Insomnia, conforme Figura 15. À esquerda estão representados os parâmetros enviados pelo cliente (ID da ordem de entrega ou de lavação), e à direita, a resposta retornada pelo servidor, onde pode-se notar a alteração do status, da mensagem retornada e do armazenamento da data de depósito do entregador no compartimento reservado. Nesse caso, o resultado obtido foi aquém do esperado, pois a chamada RPC não pôde ser executada com sucesso.

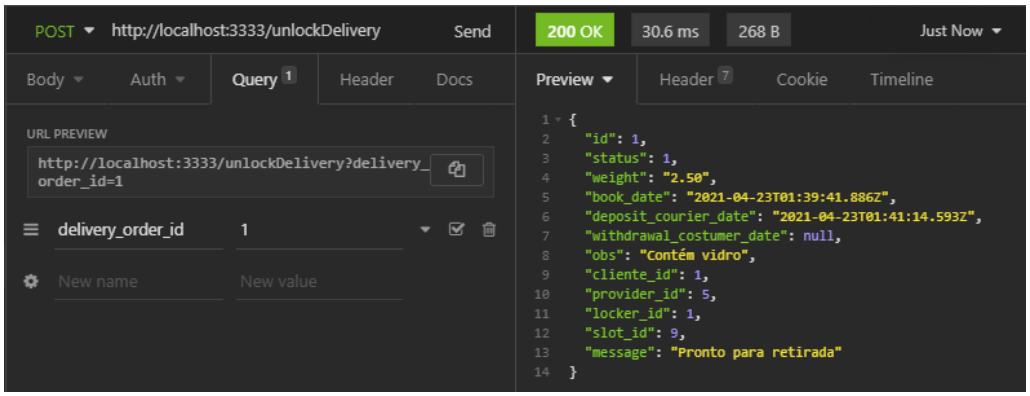

Figura 15. Criação de Pedido de Entrega

Por outro lado, a integração do *hardware* com o *Thingsboard* foi realizada com sucesso, pois a trafegabilidade de dados entre esses dois pontos foi observada, podendo assim destravar o armário manualmente pela plataforma do *Thingsboard*.

Alguns requisitos funcionais foram definidos para o desenvolvimento desta aplicação, os mesmos encontram-se na seção 3.5. Chegado ao estágio final de desenvolvimento, os requisitos completos com sucesso foram: **RF 01, RF 04, RF 05, RF 06, RF 09, RF 10 e RF 12**. Por outro lado, a funcionalidade de listagem de ordens de en-

[34]Chamada de Procedimento Remoto - *Remote Procedure Call*. É uma tecnologia de comunicação entre processos que permite a um programa de computador chamar um procedimento em outro espaço de endereçamento

trega não foi possível ser realizada no lado do *front-end*, todavia fora implementado com sucesso no *back-end*, utilizando do Insomnia para a realização dos testes.

Buscando a análise de outros pontos qualitativos, podemos citar a facilidade de troca de *hardware*. O utilizado foi o Raspberry PI, cujo valor de compra está na casa dos R$500,00 atualmente, e o *firmware* fora escrito utilizando a linguagem de programação Python. Devido à utilização de Python, o *hardware* pode facilmente ser trocado por um Arduino, pois existem bibliotecas que permitem a utilização desta linguagem de programação. Esta troca pode ocasionar um impacto econômico na solução muito grande, principalmente quanto posto em larga escala, pois o valor de um Arduino Uno hoje está na faixa dos R$80,00.

A aplicação foi desenvolvida utilizando uma abordagem chamada de *Mobile First*, que consiste no desenvolvimento adaptado para dispositivos móveis anterior à implementação para versões *desktop*. Este conceito permitiu a criação de um MVP fidedigno às necessidades do usuário do *ServicesLocker*, pois permite a utilização do *smartphone* para o desbloqueio do compartimento justamente no momento em que o mesmo encontrar-se em frente ao armário.

A solução desenvolvida pode ser flexível em muitos aspectos, como por exemplo na troca do *middleware* utilizado. Esta atividade demanda pouco tempo de configuração e permite a adaptabilidade para qualquer plataforma que se deseja utilizar. Além disso, as tecnologias utilizadas para o desenvolvimento da aplicação *Web* são conhecidas e populares em toda a comunidade de desenvolvimento de software, fato que possibilita a escalabilidade das funcionalidades através do desenvolvimento realizado por outros programadores que desejarem participar da construção de um produto mais robusto. Por fim, com base no avanço do desenvolvimento até o presente momento, é possível concluir que a solução por parte tecnológica é viável e passível de ser executada.

5. Conclusões e Trabalhos Futuros

Este trabalho fora apresentado como um estudo de caso para avaliar a construção de uma solução baseada em armários inteligentes, visando um mínimo produto viável. Com base no que fora executado ao longo deste trabalho, é possível afirmar que a construção da solução baseada em armários inteligentes é passível de ser executada.

Alguns objetivos foram definidos para a execução deste trabalho e podemos tirar algumas conclusões à respeito dos mesmos:

1. O estudo de tecnologias necessárias para o desenvolvimento da solução focou em verificar quais tecnologias seriam necessárias para a construção de um mínimo produto funcional. É válido citar que, além das tecnologias escolhidas serem escaláveis e populares, foram selecionadas pela afinidade do grupo, visto que o intuito do trabalho é uma viabilidade da solução. Assim, reduzindo o tempo de desenvolvimento, visto que não há uma curva de aprendizado tão grande para a execução da solução, e que posteriormente pode ser migrada para utilização de outras ferramentas. Foi escolhido *Node.js* para o *back-end* juntamente com Angular para o *front-end*, fato que trouxe uma facilitação na manutenibilidade, pois ambos utilizam Javascript e Typescript como linguagem de programação, logo, o desenvolvedor pode trabalhar nas duas pontas do projeto;

2. conceber, integrar e configurar o *hardware*: Os equipamentos utilizados para os testes não necessariamente são os ideais por motivos de custo ou qualidade, mas sim podem ser trocados, permitindo flexibilidade, como já citado na seção anterior;
3. desenvolver o *firmware* a ser utilizado pelo *hardware*. O desenvolvimento do *firmware* para o controle das portas do armário e comunicação com o *Thingsboard* foi feito com a linguagem de programação Python. Python vem sendo utilizado amplamente na programação de microcontroladores, logo, é possível uma fácil migração de *hardware*;
4. desenvolver as interfaces de gestão. A primeira etapa foi a de prototipação para validar as telas com a ajuda da ferramenta Adobe XD[35], o que resulta em uma visão geral de como ficará a aplicação, além da definição de como será a estrutura das páginas. A construção da interface, em primeiro momento, focou em uma abordagem para acessos via *smartphone*, visto que a maior parte dos acessos à Internet são feitos por dispositivos móveis. Logo, para alcançar um público maior e viabilizar a abertura das portas do armário, focou-se em dar suporte para acessos via *mobile*.

Trabalhos futuros devem ser feitos para que a solução possa entrar em produção no mercado. Como por exemplo: parceria com uma montadora de armários inteligentes, aperfeiçoamento das funcionalidades de integração, desenho da solução para outras resoluções como *desktop* e *tablet*, e posteriormente um aplicativo nativo para *smartphones*, parcerias iniciais com empresas de lavação de roupas, implementação de certificado TLS[36] para garantir a segurança de trafegabilidade dos dados, implementação dos requisitos funcionais faltantes, e por fim, a validação do modelo de negócios com pesquisas feitas com o público alvo desta solução, com a finalidade de realizar pesquisas quantitativas à respeito do *ServicesLocker*.

Referências

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., e Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376.

Alqahtani, H. F., Albuainain, J. A., Almutiri, B. G., Alansari, S. K., AL-awwad, G. B., Alqahtani, N. N., Masaad, S. M., e Tabeidi, R. A. (2020). Automated Smart Locker for College. In *3rd International Conference on Computer Applications Information Security (ICCAIS)*, pages 1–6.

Bellavista, P., Cardone, G., Corradi, A., e Foschini, L. (2013). Convergence of MANET and WSN in IoT Urban Scenarios. *IEEE Sensors Journal*, 13(10):3558–3567.

Chaudhary, S., Johari, R., Bhatia, R., Gupta, K., e Bhatnagar, A. (2019). CRAIoT: Concept, Review and Application(s) of IoT. In *4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4.

Chikara, A., Choudekar, P., Ruchira, e Asija, D. (2020). Smart Bank Locker Using Fingerprint Scanning and Image Processing. In *6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 725–728.

[35] `https://www.adobe.com/br/products/xd.html`

[36] O *Transport Layer Security* é um protocolo de segurança projetado para fornecer segurança nas comunicações sobre uma rede de computadores

David, B. e Chalon, R. (2018). Box/Lockers: Contribution to Collaborative Economy in the Smart City. In *IEEE 22nd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 802–807.

Faugere, L. e Montreuil, B. (2016). Hyperconnected City Logistics: Smart Lockers Terminals & Last Mile Delivery Networks.

Fortin, M. F., Côté, J., e Filion, F. (2009). *Fundamentos e Etapas do Processo de Investigação*. Lusodidacta.

HiveMQ, T. (2015). *Publish and Subscribe - MQTT Essentials: Part 2*. Disponível em: `https://www.hivemq.com/blog/mqtt-essentials-part2-publish-subscribe/` - Acesso em 30 de Setembro de 2020.

IEEE (2020a). IEEE 2413 - Standard for an Architectural Framework for the Internet of Things (IoT).

IEEE (2020b). IEEE P2413.1 - Standard for a Reference Architecture for Smart City (RASC).

IEEE (2020c). IEEE P2413.2 - Standard for a Reference Architecture for Power Distribution IoT (PDIoT).

Khan, R., Khan, S. U., Zaheer, R., e Khan, S. (2012). Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *10th International Conference on Frontiers of Information Technology*, pages 257–260.

Meulen, R. V. (2017). Gartner says 8 billion connected things will be in use in 2017 up 31 percent from 2016. Disponível em: `https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-` - Acessado em 06 de Fevereiro de 2021.

Retorno, M. (2020). *Breakeven*. Disponível em: `https://maisretorno.com/blog/termos/b/breakeven-point` - Acesso em 31 de Agosto de 2020.

Routh, K. e Pal, T. (2018). A survey on technological, business and societal aspects of Internet of Things by Q3, 2017. In *3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–4.

Sa-ngiampak, J., Hirankanokkul, C., Sunthornyotin, Y., Mingmongkolmitr, J., Thunprateep, S., Rojsrikul, N., Tantipiwatanaskul, T., Techapichetvanich, K., Pongsawang, A., Prayoonkittikul, T., Wattanakulchart, U., Prompoon, N., Ratanamahatana, C., e Pipattanasomporn, M. (2019). Lockerswarm: An Iot-based Smart Locker System with Access Sharing. In *IEEE International Smart Cities Conference (ISC2)*, pages 587–592.

Sadio, O., Ngom, I., e Lishou, C. (2019). Lightweight security scheme for mqtt/mqtt-sn protocol. In *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, pages 119–123.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., e Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, 1(1):22–32.

Ze-hong, S. e Guang-yuan, Z. (2015). Multi-functional Parcel Delivery Locker System. In *2015 International Conference on Computer and Computational Sciences (ICCCS)*, pages 207–210.

Apêndices

Apêndice 1 - CANVAS

Apêndice 2 - Requisitos Funcionais e Não Funcionais

RF01 - Cadastrar pessoa		**Prioridade:** Alta
Descrição	Cadastro de uma pessoa	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Nome, CPF ou CNPJ, Telefone, Email, Senha, Tipo e Serviços.	
Informação de Saída	Redirecionamento do usuário para tela home do sistema	
Requisitos Não Funcionais		**Categoria**
RNF 1.1	CPF ou CNPJ e Email devem ser validados	Integridade
RNF 1.2	Todos os campos do formulário de cadastro são de preenchimento obrigatório	Integridade
RNF 1.3	Contempla a especificação do tipo da empresa, conforme casos de uso das figuras 9 e 10 deste artigo.	Especificação
RNF 1.4	Os tipos podem ser: comum, transportadora ou e-commerce	Especificação
RNF 1.5	Os serviços ofertados só serão preenchidos caso o tipo seja comum. Os serviços podem ser lavação, farmácia, cartório, entre outros	Especificação

Figura 16. RF01 - Cadastrar pessoa

RF02 - Editar pessoa		**Prioridade:** Média
Descrição	Edição dos dados de uma pessoa no sistema	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Nome, Telefone, Email, Senha, Tipo e Serviços	
Informação de Saída	Mensagem informando que a edição foi efetuada com sucesso	
Requisitos Não Funcionais		**Categoria**
RNF 2.1	CPF ou CNPJ e Tipo não podem ser editados	Integridade
RNF 2.2	CPF ou CNPJ e Email devem ser validados	Integridade
RNF 2.3	Todos os campos do formulário de edição são de preenchimento obrigatório	Integridade
RNF 2.4	O campo de senha deve ter seus caracteres ocultos ao digitar, e deve possuir um campo de confirmação de senha.	Segurança
RNF 2.5	Um serviço não pode ser removido caso haja algum pedido pendente ou em progresso	Integridade

Figura 17. RF02 - Editar pessoa

RF03 - Excluir pessoa		**Prioridade:** Baixa
Descrição	Possibilidade do usuário excluir sua conta.	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Id do banco de dados do registro da tabela que deve ser excluído ou inativado	
Informação de Saída	Conta desativada	
Requisitos Não Funcionais		**Categoria**
RNF 3.1	Ao concluir a exclusão do usuário com êxito, este, deve ser redirecionado para a página home da aplicação	Especificação
RNF 3.2	A exclusão realizada é uma exclusão lógica. O registro permanece no banco, porém inativado	Manutenibilidade
RNF 3.3	A exclusão não pode ser habilitada caso a pessoa jurídica tenha algum pedido/ordem de serviço em progresso.	Integridade

Figura 18. RF03 - Excluir pessoa

RF04 - Buscar pessoa		**Prioridade:** Média
Descrição	Busca as pessoas	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Sistema e Pessoa	
Informação de Entrada	-	
Informação de Saída	Retorna uma lista de pessoas ou de uma pessoa em específico	
Requisitos Não Funcionais		**Categoria**
RNF 4.1	Este RF será utilizado ao finalizar um pedido/ordem de entrega através de um *website* parceiro, geralmente buscando por apenas uma pessoa	Especificação
RNF 4.2	A busca por chave de uma pessoa não pode retornar mais de um registro	Integridade
RNF 4.3	Caso não haja uma pessoa com o nome especificado, deve-se apresentar uma mensagem informado o usuário	Usabilidade

Figura 19. RF04 - Buscar pessoa

RF09 - Buscar Pedidos/Ordens de Serviço		**Prioridade:** Alta
Descrição	Lista em tela os pedidos/ordens de serviço em aberto	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Nenhuma, ou então utiliza filtros de data ou status	
Informação de Saída	Mostra os pedidos/ordens que satisfazem o critério de busca	
Requisitos Não Funcionais		**Categoria**
RNF 9.1	Os atores do sistema podem ter um histórico de pedidos, bem como mais de um pedido ativo ao mesmo tempo	Especificação

Figura 20. RF09 - Login no sistema

RF10 - Atualização de status do Pedido de lavação		**Prioridade:** Alta
Descrição	Quando o cliente/fornecedor entrega ou retira as roupas do armário, o status do pedido deve ser atualizado	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/fornecedor	
Informação de Entrada	-	
Informação de Saída	-	
Requisitos Não Funcionais		**Categoria**
RNF 10.1	Para alterar o status do pedido, deve-se clicar sobre um botão de status que deve aparecer junto ao pedido	Especificação
RNF 10.2	O botão de status pode assumir 2 descrições que variam de acordo com o status do pedido (Retirar/Depositar)	Especificação
RNF 10.3	Se a pessoa ainda não depositou a encomenda, o botão que altera o status do pedido deve ter a descrição 'Depositar'	Especificação
RNF 10.4	Se a pessoa ainda não retirou a encomenda, o botão que altera o status do pedido deve ter a descrição 'Retirar'	Especificação
RNF 10.5	Ao clicar no botão de status, a porta do armário referente ao pedido deve abrir	Especificação

Figura 21. RF10 - Criação de Pedido de Lavação

RF11 - Forma de Pagamento		**Prioridade:** Baixa
Descrição	Requisito necessário para o futuro do projeto. Porém, na situação atual de MVP, não se faz necessário.	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Tipo de cartão utilizado, bandeira, número, data de vencimento, nome completo, código de segurança, entre outros.	
Informação de Saída	Pagamento realizado	
Requisitos Não Funcionais		**Categoria**
RNF 11.1	Um compartimento do armário é liberado para a execução do pedido/ordem após a confirmação do pagamento	Especificação
RNF 11.2	Utilizar-se-á uma API de pagamentos para o desenvolvimento deste RF	Especificação
RNF 11.3	Todas as medidas de segurança, privacidade e execução de transações devem ser implementadas	Segurança
RNF 11.4	É oferecido ao usuário a opção de salvar o cartão para usos futuros	Usabilidade

Figura 22. RF11 - Edição do Pedido de Lavação

RF12 - Simular um Pedido/Ordem de Entrega		**Prioridade:** Média
Descrição	Um cliente pode realizar uma compra em um *website* parceiro, e optar como forma de entrega pela utilização do *ServicesLocker.*	
Fonte de Informação	Alunos/desenvolvedores responsáveis pelo projeto	
Usuários	Cliente/Fornecedor	
Informação de Entrada	Email do cliente, endereço do armário, data de realização do pedido e observações opcionais	
Informação de Saída	Um pedido é criado na conta do usuário cujo email fora digitado no momento da finalização do pedido no *website* parceiro.	
Requisitos Não Funcionais		**Categoria**
RNF 12.1	Ao finalizar o pedido, o *website* parceiro se comunica com a API do *ServicesLocker* para criar uma ordem de entrega.	Especificação
RNF 12.2	Ao optar pelo *ServicesLocker,* o cliente digita seu email na finalização do pedido no *website* parceiro e escolhe em qual armário deseja que sua encomenda seja entregue. Todas as informações são enviadas à API do *ServicesLocker*	Especificação
RNF 12.3	Esta simulação será feita através de um JSON contendo os dados que seriam enviados à API em uma futura implementação	Flexibilidade e Reusabilidade

Figura 23. RF12 - Cancelar Pedido de Lavação

Figura 24. *Login*.

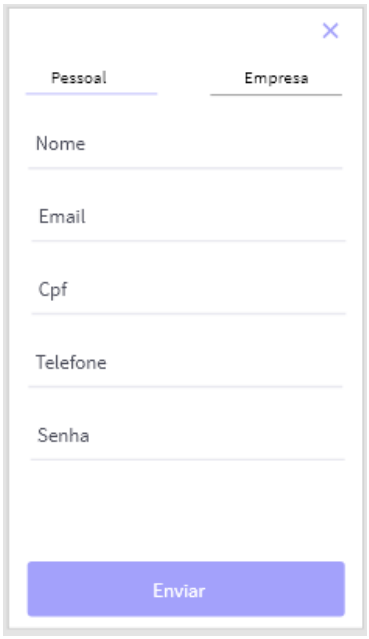

Figura 25. Pessoa física.

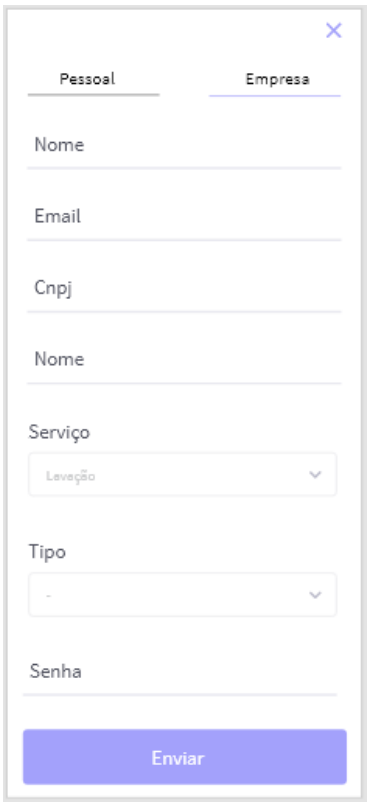

Figura 26. Pessoa jurídica.

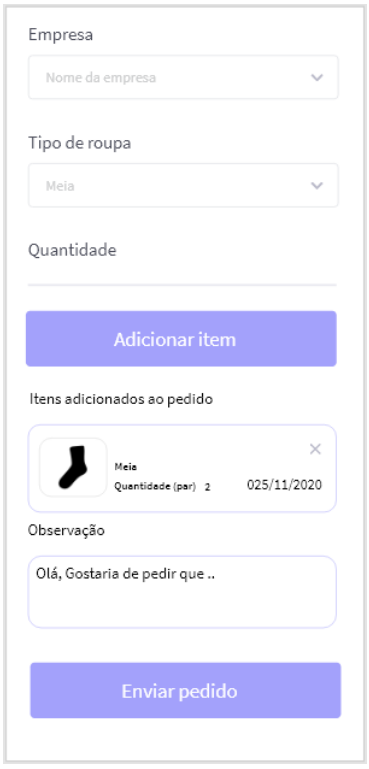

Figura 27. Criar pedido.

Figura 28. Lista de pedidos.

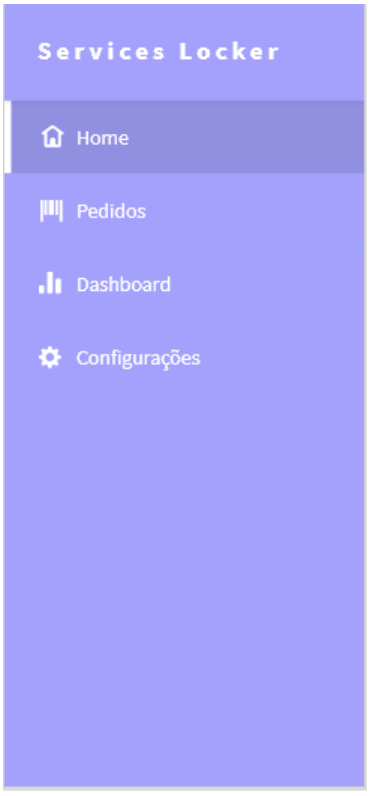

Figura 29. Menu lateral.