Smartrack - Sistema de Monitorização de Transporte de Mercadorias Delicadas

Leonan Fraga Leonardo - 38678

Trabalho realizado sob a orientação de

Prof. Dr. Paulo Jorge Teixeira Matos

Prof. Dr. Pedro Luiz de Paula Filho

Mestrado em Sistemas de Informação

2017-2018

Smartrack - Sistema de Monitorização de Transporte de Mercadorias Delicadas

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Sistemas de Informação

Leonan Fraga Leonardo - 38678

2017-2018

A Escola Superior de Tecnologia e de Gestão não se responsabiliza pelas opiniões expressas neste relatório.

Declaro que o trabalho descrito neste relatório é da minha autoria e é da minha vontade que o mesmo seja submetido a avaliação.

Leonan Fraga Leonardo - 38678

Agradecimentos

Agradeço aos Professores Paulo Jorge Teixeira Matos e Pedro Luiz de Paula Filho por partilharem parte de seu conhecimento, tempo e dedicação comigo.

Em seguida agradeço a Deus por abençoar toda minha trajetória e auxiliando na superação dos obstáculos da vida.

Gostaria de expressar aqui meu maior agradecimento à minha família, composta pelo meu pai (Nelson do Nascimento Leonardo), mãe (Inês Barbosa Fraga), irmão (Brucy Fraga Leonardo) e esposa (Daniela Prescila Minski), cujo apoio foi fundamental para o desenvolvimento do presente trabalho.

A seguir agradeço aos meus amigos de minha instituição de origem bem como os que conheci aqui. Em especial, gostaria de deixar minha enorme gratidão expressa neste documento a família dos meus amigos Lurdes, Pascoal e Rui, que me aceitaram de braços abertos em suas casas.

Farei também uma breve menção mais que merecida ao Professor Everton Coimbra pelas palavras positivas e confiança em minha pessoa, o qual sem dúvida foi deveras importante para minha carreira académica.

Gostaria de mencionar sem exceções, todo o Corpo Docente da Universidade Tecnológica Federal do Paraná - Medianeira (UTFPR-MD) que contribuiu para meu crescimento pessoal e profissional e que sem dúvidas em minha opinião é composto pelos melhores profissionais da área.

Resumo

No transporte de mercadorias as condições podem ser críticas para sua integridade. Como no caso de mercadorias vivas, em que os danos causados podem ser irreversíveis. A situação pode ser mais grave quando se trata de mercadorias de elevado valor, nomeadamente únicas, como é o caso de obras de arte. É assim natural que a contratualização do serviço de transporte, para determinados bens/mercadorias, seja feita em função das condições que são necessárias garantir. Tais condições podem estar relacionadas com a temperatura, humidade, luminosidade ou variações de movimento. No entanto, a contratualização do serviço não garante que o mesmo seja feito em tais condições, sendo por vezes impraticável para o cliente apurar imediatamente se tal aconteceu ou não e se houveram danos aos bens transportados. No que concerne às transportadoras é também importante ter uma solução sólida e substancial que permita comprovar ao cliente a qualidade do serviço prestado. Até porque muitas empresas subcontratam o serviço, podendo haver em um mesmo procedimento várias transportadoras envolvidas. É assim necessário garantir que o transporte é feito nas condições pretendidas, pelas várias entidades envolvidas e, quando há violação de tais condições, ter forma de apurar de quem é a responsabilidade. O projeto envolve ainda uma forte componente de segurança e de autenticação que valida que os dados recolhidos são fidedignos. Esta componente foi implementada com recurso à framework conceitual AAA (do inglês, *Authentication*, *Authorization* and *Accounting*) e com medidas preventivas para ataques MITM (do inglês, Man-in-the-Middle).

Palavras-chave: cadeia de suprimentos, internet das coisas, monitorização de mercadorias.

Abstract

In the transportation of goods, conditions may be critical to their integrity, specifically in the case of living goods because damages may be irreversible. The situation can be more aggravating when it comes to high-value goods, such as works of art. It is thus natural that the hiring of the transport service, for certain goods/merchandise be made according to the conditions that are necessary to ensure. Such conditions may be related to temperature, humidity, luminosity or variations in movement. However, the hiring of the service does not guarantee that it is done in such conditions, and it is sometimes impracticable for the customer to ascertain immediately if this happened or not and if there were damages to the goods transported. With regard to the provider of the transportation service, it is also important to have a solid and substantial solution that allows to prove the customers the quality of the service provided. Even because many companies subcontract the service, there may be in the same procedure several carriers involved. Faced with this fact, there is a need to ensure that transportation is carried out under the desired conditions, by the serveral entities involved and when there is a violation of such conditions, have a way of ascertaining who is the responsible. The project also involves a strong security and authentication component that certifies that the data collected is trustworthy. This component was implemented using the conceptual framework AAA (Authentication, Authorization, and Accounting) and includes preventive measures for MITM (Man-in-the-Middle).

Keywords: supply-chain, internet of things, goods monitoring

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

API Interface de Programação de Aplicativos - do inglês, *Application Programming Interface*. 25, 27, 30, 36, 47, 63, 71–74, 77, 79, 81

AS Apple Store. 26, 84

ATT *Attribute Protocol*. 21, 22

BD Banco de Dados. 30, 31, 62, 71, 72, 78, 80

BLE *Bluetooth Low Energy*. 9, 16, 17, 19, 22, 23, 34, 35, 57, 58, 67, 70, 74, 76, 77, 83, 85, 87, 93, 916

BR *Basic Rate*. 16

CS Serviço na Nuvem - do inglês, *Cloud Service*. 29, 30, 32, 43, 45, 46, 50–54, 56, 57, 60, 62, 71, 74, 77, 78, 80–82, 89, 97

DK Kit de Desenvolvimento - do inglês, *Development Kit*. 65

FIFO *First In First out*. 68

GATT *Generic Attribute Profile*. 22

GMT *Greenwich Mean Time*. 10

GP Google Play. 25

GPS Sistema de Posicionamento Global - acrónimo do inglês, *Global Positioning System*. 10, 24

GUI Interface Gráfica do Usuário - acrónimo do inglês, *Graphical User Interface* -. 25, 47, 73, 78, 83–86, 912

HCI *Host Controller Interface*. 18, 19

IDE Ambiente de Desenvolvimento Integrado - acrónimo do inglês, *Integrated Development Environment* -. 70, 72, 73, 93

IoE Internet de Tudo - do inglês, *Internet of Everything*. 12–15

IoT Internet das Coisas - do inglês, *Internet of Things*. 3, 12–15, 32, 37, 67

IPB Instituto Politécnico de Bragança. 1

ISM *Industrial, Scientific and Medical*. 16

LP Linguagem de Programação. 25, 27–29, 72, 73, 80, 915

MITM *Man-in-the-Middle*. 32, 33

MVVM Model-View-ViewModel. 73, 81

OTP *One-Time Password*. 44–46, 79, 80, 912

PC Computador Pessoal - do inglês, *Personal Computer*. 16

PDA Assistente Pessoal Digital - do inglês, *Personal Digital Assistants*. 16, 24

RAM *Random Access Memory*. 66

REST *Representational State Transfer*. 30, 63, 71, 77, 79, 81

RFID *Radio-Frequency Identification*. 14

RH Humidade Relativa - do inglês, *Relative Humidity*. 67, 69

RTC *Real Time Clock*. 50

SDK Kit de Desenvolvimento de *Software* - do inglês, *Software Development Kit*. 35, 70

SO Sistema Operativo. 24–27, 29, 44

SoC *System on Chip*. 19, 67

TS Toque na Tela - do inglês, *Touch Screen*. 24, 28

UI Interface do Usuário - do inglês, *User Interface*. 26

URI Identificador Uniforme de Recursos- acrónimo do inglês, *Uniform Resource Identifier*. 80, 81, 912

UTFPR-MD Universidade Tecnológica Federal do Paraná - Medianeira. vii, 1

UX Experiência do Usuário - acrónimo do inglês, *User Experience* -. 26, 85

WS *WebService*. 29, 77, 921

Capítulo 1

Introdução

Este documento tem como objetivo expor a análise e desenvolvimento de um serviço para monitorização do transporte de mercadorias delicadas, estas que podem ser definidas como sendo mercadorias sensíveis a determinadas condições de transporte. O presente documento foi supervisionado pelo Professor Doutor Paulo Jorge Teixeira Matos, Docente do Instituto Politécnico de Bragança (IPB), e pelo Professor Doutor Pedro Luiz de Paula Filho, Docente da UTFPR-MD.

Este capítulo introdutório está dividido em três secções. Na primeira secção é feita uma contextualização no que concerne ao trabalho. Na segunda secção são abordados os objetivos gerais e específicos respetivamente. Por fim, na terceira secção é apresentada a estrutura deste documento.

1.1 Contextualização

Esta dissertação tem como objetivo comprobatório construir uma solução de baixo custo para monitorização de mercadorias delicadas. Dentro deste nicho de mercadorias se enquadram os bens de alto valor acrescentado, animais vivos e alimentos perecíveis. Alguns exemplos destas mercadorias são obras de arte únicas, mercadorias de valor sentimental inestimável e até mesmo órgãos para transplantes.

Tais mercadorias podem estar sujeitas a diversos fatores que as podem colocar em

risco, como quedas, excesso de humidade e temperatura. Fatores estes suscetíveis de ocorrerem no transporte destas mercadorias e que podem implicar danos, perdas e, em casos mais severos, a morte de seres como plantas, animais e humanos.

Há muitas empresas que disponibilizam serviços de transporte, nomeadamente para mercadorias que requerem condições específicas, estas que são subjacentes ao serviço prestado, ou que podem ser contratualizadas especificamente para cada caso. No entanto, a garantia do cumprimento destas condições é, na maioria dos casos, apurada em função do estado da mercadoria entregue. O que eventualmente não permite que seja verificado se há danos menos imediatos ou evidentes. Provar a origem dos danos resultantes das condições de transporte pode ser uma tarefa difícil ou até mesmo impossível, no entanto, tal informação pode ser valiosa tanto para quem contrata, quanto para quem presta o serviço.

A tese desta dissertação de mestrado é demonstrar a possibilidade técnica e económica em mitigar a ocorrência destes tipos de problemas, utilizando como recurso algumas das tecnologias presentes no conceito de *Internet of Things*.

Diante de tais fatos, é assim proposta uma solução viável financeiramente e operacionalmente que faz prova do cumprimento das condições acordadas para o transporte das mercadorias ou, em caso de incumprimento, possibilita apurar quando e em que trajeto este ocorreu. Tal informação é devidamente autenticada para efeitos de prova do incumprimento.

A solução desenvolvida apresenta ainda aspetos inovadores, nomeadamente suporte para novos modelos de serviços, cujo foco está no transporte de mercadorias delicadas.

1.2 Objetivos Gerais e Específicos

Este documento tem como objetivo geral apresentar os processos utilizados durante o desenvolvimento de um serviço para monitorizar o transporte de mercadorias delicadas coletando dados como temperatura, humidade relativa, deslocamento e outros, nomeadamente quando estes estão fora dos limites aceitáveis, bem como contextualizar esses dados

em termos geográficos e temporais, no sentido de apurar responsabilidades.

Na sequência serão apresentados os objetivos específicos que se pretendeu alcançar:

- Desenvolver e testar uma solução para coletar informações referentes às condições de transporte;
- Contextualização geográfica e temporal das informações coletadas;
- Autenticar a informação coletada no sentido de garantir a veracidade da mesma e do contexto de recolha;
- Implementar toda base de suporte para comunicação de dados;
- Efetuar a integração de todos os componentes da solução, que estão relacionadas com o remetente, os possíveis operadores e o destinatário;
- Assegurar que a solução criada permite um custo residual por cada transporte efetuado;
- Não requerer alterações profundas dos procedimentos e processos utilizados pelas operadoras, nomeadamente em termos de recursos humanos e tempo despendido.

Em suma, com este trabalho pretende-se desenvolver um serviço capaz de monitorizar as condições de transporte com o auxílio de sensores, identificar os responsáveis por eventuais falhas ocorridas e garantir a fiabilidade dos dados visando minimizar os prejuízos e maximizar a qualidade do serviço prestado.

1.3 Estrutura do Documento

Esta dissertação está dividida em sete capítulos, dos quais, os Capítulos 1 e 2, apresentam uma introdução ao problema e uma visão geral da solução. No Capítulo 3, são explicadas com um nível superior de detalhes, baseando-se na opinião de outros autores, as tecnologias comummente utilizadas em soluções para a Internet das Coisas - do inglês, *Internet*

of Things (IoT), e também, algumas soluções similares a que está sendo proposta, assim como uma análise comparativa entre tais soluções. Por ser um capítulo extenso, ao fim de cada secção, objetivando tornar a leitura menos exaustiva, fez-se um breve resumo do conteúdo abordado em tais secções.

Quanto ao Capítulo 4, neste é explicado em detalhes, com auxílio de diagramas, o funcionamento da solução e como as tecnologias são utilizadas. Nos Capítulos 5 e 6, são apresentadas as ferramentas utilizadas para desenvolver a solução, os detalhes das etapas de implementação e testes. Em relação ao Capítulo 7, neste é feita uma conclusão no tocante ao desenvolvimento desta dissertação. Por fim, nos Apêndices 8 e 9, são apresentadas definições complementares de algumas tecnologias, assim como uma breve documentação e trechos de códigos utilizados.

Capítulo 2

Análise do Problema

Este capítulo será utilizado para explanar com mais detalhes a análise realizada no tocante ao problema e respetiva solução.

2.1 Transporte de Mercadorias Delicadas

Quando o objetivo é garantir a qualidade do serviço prestado, nomeadamente a integridade dos bens em questão, o processo de transporte deve ser realizado de acordo com as necessidades desse bem, sob a pena de provocar danos imediatos ou futuros, que podem inclusive ser irrecuperáveis.

2.1.1 Situações Factíveis

Uma situação possível é danificar componentes internos de equipamentos eletrónicos durante o transporte destes. A título de exemplo, têm-se os equipamentos utilizados no processo de fabricação de lentes oftálmicas.

Quedas ou variações bruscas de movimento, podem provocar danos nos componentes, impedindo que estes funcionem corretamente. Por vezes, tais danos podem ser imperceptíveis, por exemplo, quando resultam no descalibrar dos equipamentos. Nestes casos, os prejuízos podem não ser imediatos, mas a longo prazo, impactando de forma negativa

a produção de lentes, o que pode gerar prejuízos exponenciais para as empresas que as produzem.

De salientar que a identificação destes problemas muitas vezes só é possível no momento em que for realizada a instalação ou uso dos equipamentos - fato este que ocorre após a entrega da mercadoria. Podendo ser um processo complicado fazer prova da causa, ficando a dúvida se o equipamento foi remetido com defeito ou se ocorreram falhas no transporte.

Há casos mais críticos, onde os prejuízos causados pelo transporte inadequado podem ser permanentes, como danos à peças de arte únicas, morte de animais e até mesmo a perda de órgãos para transplantes, que pode implicar de forma direta ou indireta na morte de seres humanos.

Como exemplo, tem-se o transporte de animais domésticos cuja saúde é sensível a temperaturas negativas e, por este motivo não deve ser exposto a tais condições para que não adquira doenças, ou fique debilitado, em resultado de tal exposição. Caso isto ocorra, o animal pode não vir a entrar em óbito imediatamente, todavia, pode acontecer que, devido à doença contraída, este venha a falecer após a sua entrega. Portanto, caso não haja mecanismos para que o destinatário possa comprovar tal situação, não há como responsabilizar qualquer entidade.

Outro exemplo de uma situação similar é o transporte de alimentos perecíveis, tais como frutas, verduras, legumes, carnes, etc. As alterações de algumas condições como temperatura e humidade podem não impactar de forma negativa a aparência destes produtos, mas sim, sua qualidade, o que se pode traduzir na degradação precoce dos produtos, na perca de sabor ou alterações da textura.

Há casos mais severos em que se torna plausível a aplicação da solução proposta neste trabalho, especificamente nas situações onde há necessidade em monitorizar as condições de transporte de órgãos para transplante. Nestes casos, conforme afirmação de [1], é necessário não apenas que seja realizado um armazenamento adequado mas também que este seja entregue dentro de um limite de tempo pré-determinado pelo tempo de isquemia (tempo que o órgão fica sem vascularização). Caso ocorra atrasos neste processo, o órgão

pode ser prejudicado e em alguns casos pode ser levado à falência, tornando inviável a utilização do mesmo em cirurgias de transplante.

Mesmo no caso de mercadorias em que os danos possam ser facilmente detetáveis, como no transporte de peças em vidro, a utilização deste tipo de solução é da maior pertinência para a própria empresa prestadora dos serviços apurar internamente responsabilidades. De realçar que muitas empresas de transporte recorrem à subcontratação, seja de outras empresas, seja de profissionais liberais.

Diante de tais situações, é possível afirmar que quando o transporte é feito de maneira inadequada, nomeadamente quando os limites aceitáveis para que a carga se mantenha em condições são ultrapassados, diversos prejuízos podem ser causados, estes por sua vez podem ser imensuráveis. Portanto, torna-se assim importante que haja um meio de monitorizar tais condições, que promova um tratamento mais cuidadoso e responsável das mercadorias e, em caso de incumprimento das condições contratualizadas para o transporte, permita identificar os responsáveis por tais prejuízos. Para isso, foi elaborado um sistema cuja arquitetura pode ser observada na Figura 2.1.

2.1.2 Descrição do Modelo Operacional

De acordo com a Figura 2.1, é possível notar que o fluxo da solução proposta tem início a partir do remetente, que pode ajustar as condições que devem ser respeitadas no transporte da mercadoria. Posteriormente, o remetente contratualiza o serviço de transporte em uma prestadora de serviços, como, os CTT, FedEx, XPO Logistics, entre outros.

Em seguida, o prestador encaminha a mercadoria para o operador de transporte (motoristas, pilotos aeronáuticos ou marítimos, entre outros), podendo haver mais do que um operador envolvido, transitando a mercadoria de operador para operador, até ser entregue ao destinatário. A transição é feita nos postos de coleta, também designados por *milestones*. Ambos os operadores, seja o operador atual ou o próximo, podem verificar as condições da carga antes de a entregar ou a receber, afim de apurar eventuais problemas com a carga.

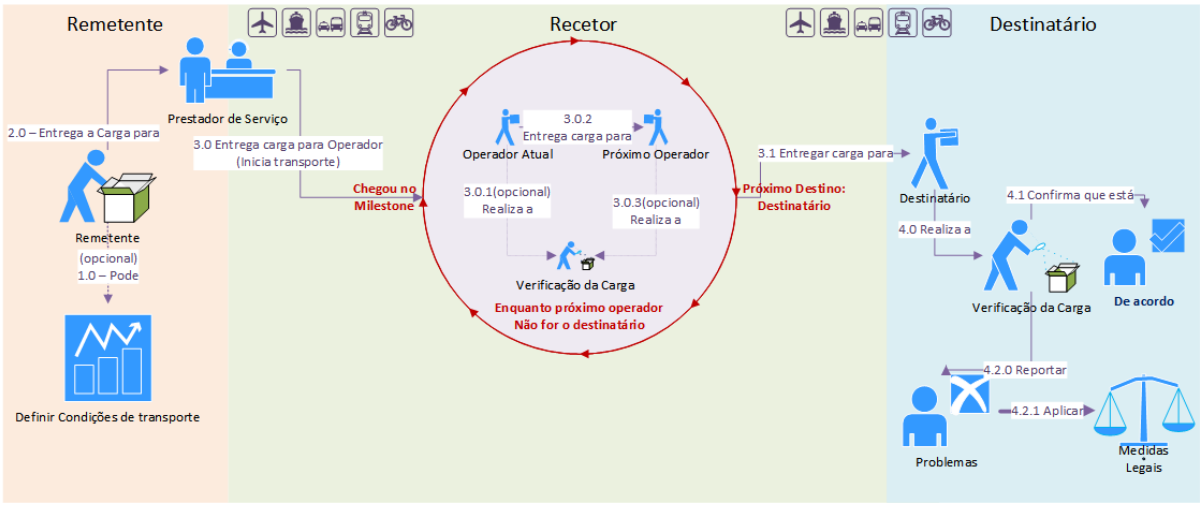


Figura 2.1: Fluxo de trabalho do processo.

Por fim, quando a mercadoria é apresentada ao destinatário, este ndeve averiguar as condições da mercadoria no momento da receção. Na falta desta, o destinatário pode receber uma mercadoria danificada, mesmo não tendo danos aparentes. Apurar a responsabilidade dos danos causados e fazer prova disso, pode ser uma tarefa complexa.

2.2 Arquitetura da Solução

A monitorização das condições de transporte é a base de toda a solução, especificamente no que diz respeito à temperatura, humidade e deslocação ou impacto. A recolha destes dados é utilizada para fins de identificação de situações que violem os limites acordados entre prestador de serviço e remetente.

Para este efeito, idealizou-se o **SmarTrack Device** - um dispositivo capaz de recolher variáveis do contexto de transporte, mas que também satisfaz os seguintes requisitos:

- Não ter peso e dimensões significativas comparativamente ao produto transportado;
- Ter autonomia energética elevada, isso porque o transporte da mercadoria pode demorar vários meses até chegar ao destinatário, mas também porque a cadência de substituição da bateria não deve ter um impacto significativo no custo de transporte;

- Ter acesso aos dados recolhidos de forma simples e fácil para quem o faz.

A autonomia do SmarTrack Device, além de cumprir o requisito mencionado anteriormente, também permite que este seja reutilizável. Fator este que reduz a emissão de lixo eletrónico e custos ao cliente. A autonomia é um requisito fundamental, tanto que influenciou inclusive a solução desenvolvida. Por exemplo, os dados não são transmitidos em tempo real porque, para tal, seria necessário utilizar um módulo específico para comunicação móvel, cujos consumos são demasiado elevados, o que poderia comprometer a autonomia do dispositivo. Além de que acrescentaria um custo operacional relativamente elevado ao transporte, haja vista que sua utilização implica em custos adicionais com o próprio equipamento e também com as comunicações.

Diante a inviabilidade em transmitir dados em tempo real, o processo de recolha dos dados é realizado aquando da transição da mercadoria. Isto é, no momento em que um operador de transporte inicia o processo de transição da mercadoria em um *milestone*, os dados armazenados no SmarTrack Device são recolhidos com recurso a uma aplicação para smartphone, desenvolvida especificamente para este efeito e com recurso a *Bluetooth Low Energy* (BLE).

A utilização do *smartphone* acresce um conjunto de benefícios importantes para o sucesso da solução. Por exemplo, o *smartphone* é amplamente difundido, o que implica na redução de custos operacionais, haja vista que não há necessidade em utilizar um módulo específico para este fim. O *smartphone* fornece informações importantes para fazer a contextualização geográfica do local da transição da mercadoria - onde se dá tramitação da responsabilidade sobre a mercadoria. Por fim, o *smartphone* faculta também ligação à Internet, importante para se conseguir fazer a contextualização temporal do ato de recolha de dados, assim como para remeter os dados para o repositório central.

Tal repositório é utilizado para armazenar todas as informações recolhidas pelos dispositivos e as distribuir de maneira adequada pelos envolvidos no processo, enviando as informações pertinentes para cada usuário.

2.3 Autenticidade das Informações

Garantir a veracidade dos dados recolhidos é parte fundamental para a credibilidade da solução a desenvolver. Para este feito, recorreu-se, entre outros mecanismos, à contextualização temporal e geográfica dos dados. Em outras palavras, isso significa que é possível identificar o trajeto e o momento em que a informação foi recolhida.

No que tange à vertente temporal, os dados são contextualizados pela hora absoluta - utilizando sempre a mesma referência geográfica (*Greenwich Mean Time* (GMT)+0), e pelo tempo relativo decorrido entre o início do transporte e o momento da recolha dos dados.

Assim sendo, estes conjuntos de dados temporais são complementares para garantir a autenticidade dos dados, associando estes ao momento do transporte que foram coletados.

No tocante ao contexto geográfico, os dados recolhidos pelo SmarTrack Device são associados ao local onde o operador de transporte faz a leitura, mais precisamente quando este chega em um *milestone* e se dá a transição da mercadoria entre operadores, ou entre operador e destinatário. Para o efeito, é obtida a localização GPS, que é convertida no nome da cidade, região (ou estado em alguns locais) e país. Como exemplo, cidade de Bragança, situada na região de Bragança, a qual pertence ao país Portugal.

Conforme mencionado anteriormente, o SmarTrack.IO não recolhe dados em tempo real e, por este motivo, não há necessidade em obter a localização precisa do local onde um problema ocorreu, mas sim, do trajeto onde este ocorreu.

Em cada *milestone*, tendo ocorrido violações das condições contratualizadas para o transporte, estas são reportadas para a central de dados por intermédio do aplicativo.

Por exemplo, um problema pode ocorrer no meio do oceano, no entanto, a informação mais relevante para o que está aqui proposto, é que este problema foi detetado quando a mercadoria chegou em uma determinada cidade - ou ainda, *milestone*. Pois, com isso é possível diminuir o consumo energético de forma significativa, visto que, não há necessidade em utilizar um módulo Sistema de Posicionamento Global - acrónimo do inglês, *Global Positioning System* (GPS) constantemente.

2.4 Suporte para Serviços Inovadores

No transporte de mercadorias delicadas, há situações onde o remetente e o destinatário podem solicitar evidências mais concretas para provar que a mercadoria está em conformidade com os requisitos a serem atendidos, por exemplo, no caso do transporte de animais de estimação ou de elevado valor económico ou sentimental. A solução idealizada permite sustentar um serviço inovador para este tipo de mercadoria, que passa por prestar provas do bem-estar da mercadoria ao longo do período de transporte que tal inovação, como exemplo, fornecer evidências do bem-estar do animal através da recolha de fotografias, ou mesmo vídeos.

As imagens podem, por exemplo, servir para demonstrar que o animal foi alimentado, o que em conjunto com a informação temporal, pode também permitir comprovar que a alimentação foi feita nos prazos acordados.

Essas imagens são obtidas a partir do *smartphone* do operador de transporte e serão contextualizados com dados temporais, geográficos e também do operador de transporte o qual capturou as imagens. Estes dados são contextualizados para garantir que as imagens são autênticas, mitigando ações de falsificação destas.

Importante realçar que, para a utilização deste recurso, o operador de transporte deve ter acesso à mercadoria, o que pode implicar em eventuais aberturas do contentor, baú do camião, porão do avião e afins. Neste caso, deve ser acordado entre o remetente e o prestador de serviços a necessidade em utilizar este recurso.

Para cumprir com os requisitos necessários para solucionar o problema abordado neste capítulo, é necessário a utilização de alguns conceitos e recursos tecnológicos. Os quais serão abordados a seguir.

Capítulo 3

Tecnologias em Soluções da Internet das Coisas

O presente capítulo apresenta uma sustentação teórica, objetivando auxiliar na compreensão do tema abordado utilizando como embasamento publicações de outros autores. O capítulo está dividido em sete secções, onde em cada secção um tópico relacionado ao presente trabalho será abordado.

3.1 IoE e IoT

Nesta Secção serão abordados os conceitos de Internet de Tudo - do inglês, *Internet of Everything* (IoE) e IoT, no entanto, a ordem utilizada para a explicação destes será inversa em termos temporais. Tal critério foi utilizado pois o conceito de IoE é mais abrangente que o IoT, sendo assim, será utilizada uma abordagem teórica partindo do conceito geral para o específico.

3.1.1 Uma breve introdução a *Internet of Everything*

A IoE pode ser considerada também como um conceito que vem emergindo nos últimos anos. A Cisco[1] referencia-o como algo que transforma informações em ações e para isso é feita uma combinação de diferentes conceitos como pessoas, dados e *things* (tradução livre, coisas) tornando conexões de rede mais valiosas, ou seja, fornecendo informações mais relevantes [2].

A IoE tem quatro pilares, sendo estes nomeados como pessoas, dados, *things* e processos. Visando uma melhor compreensão de tais pilares, a seguir será realizada uma explanação mais detalhada dos papéis de cada entidade:

- **Pessoas** poderão se conectar à Internet de outras maneiras além do uso de dispositivos eletrónicos como Tablets, Televisores e *smartphones*. Um exemplo de uma alternativa de conexão é a ingestão de pílulas que detetam e relatam algumas informações corporais a um médico por meio de uma conexão segura com a Internet;

- ***Things*** são objetos conectados uns com os outros através de uma plataforma de rede acessível, segura e confiável. Na IoE, os *things* serão mais sensíveis ao contexto e terão a capacidade de mudar como e quando as decisões são tomadas, fazendo com que tais decisões sejam mais relevantes e valiosas;

- **Dados** na IoT geralmente são coletados por um dispositivo e transmitidos pela Internet para um ponto central. Quanto maior a capacidade de processamento dos *things* conectados à Internet, mais inteligentes - dentro da perspetiva de transformar dados em informações úteis e relevantes - estes se tornarão. Portanto, na IoE os dados são considerados como informações relevantes utilizadas no processo de avaliação e tomada de decisão, tornando-o mais eficaz;

- **Processo** está relacionado no modo como esses pilares interagem uns com os outros. Com isso as informações são distribuídas de maneira mais segura, onde cada pessoa recebe apenas informações pertinentes a ela [2].

[1] https://www.cisco.com/

Portanto, a IoE pode ser interpretada como um conceito mais abrangente, haja vista que a IoT está associada apenas com *thingies* conectadas entre si, enquanto tais *thingies* fazem parte apenas de um dos quatro pilares da IoE.

3.1.2 Explicação do termo *Internet of Things*

No que concerne ao termo IoT, este vem sendo modificado com a evolução das tecnologias. Em 1999, Kevin Ashton tornou notável o termo introduzindo-o no contexto de gestão de abastecimento. Uma década depois, essa definição se tornou mais abrangente, incluindo aplicações nas áreas de saúde, logística, agricultura, etc [3].

Para os autores [4], IoT é um novo paradigma que está em voga no cenário de telecomunicações modernas sem fio. Em linhas gerais IoT é a presença de vários objetos ou "coisas" - como *smartphones*, sensores, etiquetas *Radio-Frequency Identification* (RFID), entre outros, que, por intermédio de esquemas de endereçamento único, podem interagir com outros dispositivos para alcançar um objetivo.

Conforme afirmação de [5], o conceito IoT não é novo, haja vista que muitos equipamentos industriais são supervisionados de maneira remota há anos e, por este motivo, o que torna o termo IoT como novidade é seu potencial para as mais diversas aplicações. Isto ocorre porque muitas limitações técnicas como capacidade de processamento, consumo energético, custos e dimensões vêm sendo eliminadas. Ainda segundo [4], as etiquetas RFID têm sido utilizadas amplamente para monitorização em tempo real de mercadorias no setor de transporte.

Algumas das aplicações da IoT em diferentes domínios podem ser visualizadas na Figura 3.1, onde é possível observar que uma das áreas relevantes de entre tais domínios é a monitorização de ambiente.

Nos domínios mostrados na Figura 3.1, e em muitos outros não mencionados, é deveras importante que haja mecanismos de segurança para garantir não apenas a fiabilidade dos dados, mas também a privacidade do usuário bem como sua segurança física em casos mais extremos.

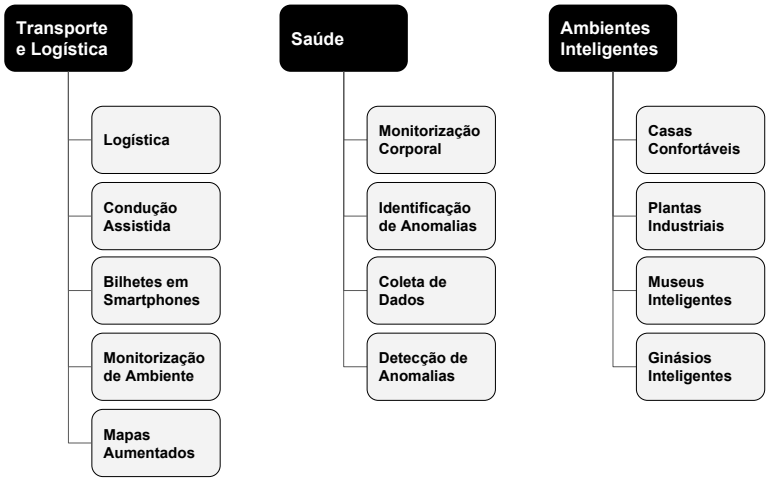


Figura 3.1: Aplicações da Internet das Coisas em algumas áreas.
Fonte: Adaptação de [4]

3.1.3 Sumário

No tocante à dissertação, com base no que foi apresentado nesta secção, é possível afirmar que solução proposta pode ser enquadrada no conceito de IoE, principalmente pelo fato de ter um **objeto** que recolhe **dados** através de um **processo** de comunicação com outros **objetos**. No entanto, há alguns aspetos que não são inseridos na solução, por este motivo, a solução se encaixa melhor no conceito de IoT. Quanto as formas de realizar a comunicação entre os objetos da IoT, há vários meios, no entanto, alguns se destacam, tal como o Bluetooth, que por sua vez é um protocolo de comunicação amplamente difundido.

3.2 Bluetooth

Nesta Secção será discorrida uma breve revisão literária no que tange ao Bluetooth, isto é, seu objetivo, conceitos técnicos e seus modos de operação.

3.2.1 Introdução ao Bluetooth

O Bluetooth tem como objetivo substituir a conexão por cabos entre dispositivos eletrónicos, sejam estes portáteis ou fixos, como *Personal Digital Assistants* (PDAs), Smartphones, Notebooks, *Personal Computers* (PCs), entre outros. Para cumprir com seu objetivo, ele opera numa banda *Industrial, Scientific and Medical* (ISM) não licenciada, a 2.4 GHz, em 79 canais com 1 MHz de largura de banda em cada, e alterna de canais até 1600 vezes em um intervalo de 1 segundo, utilizando um transcetor de salto de frequência para lidar com interferências [6].

Desde o lançamento do Bluetooth, diversas mudanças foram realizadas, historicamente[2] e tecnologicamente. Tais mudanças sempre buscaram realizar melhorias no protocolo, seja em segurança ou no modo de funcionamento. A Tabela 3.1 apresenta as principais evoluções do Bluetooth em termos de tecnologia, tendo como base as informações descritas em [7].

Conforme é possível observar, ainda na Tabela 3.1, muitas evoluções foram feitas no Bluetooth 4, dentre estas evoluções está a introdução ao BLE, que por sua vez é um modo de operação do Bluetooth, conforme afirmação de [6].

O BLE, ou Bluetooth Smart, tem como principal característica o baixo consumo energético e baixa complexidade quando comparado com o *Basic Rate* (BR). No entanto, o BLE tem uma taxa de transferência de dados relativamente baixa (até 2 Mb/s[3]) quando comparado com o BR, que por sua vez pode atingir até 54 Mb/s. A contrapartida deste modo de operação, está no consumo energético elevado [6].

Devido a algumas diferenças existentes entre os modos de operação (Tabela 3.2), uma aplicação projetada para utilizar apenas BLE não se comunica com dispositivos operando somente no modo BR.

Diante desta situação, cabe aos engenheiros analisarem a melhor opção para uma dada solução, tendo consideração o consumo energético e a taxa de transferência de dados.

[2]Um pouco da história do Bluetooth é abordada no Apêndice 8

[3]A taxa de transferência está relacionada com o uso de codificação dos pacotes da camada física. Este conceito é abordado na subsecção 3.2.2

Versões do Bluetooth	Recursos
Bluetooth 1.x	1. IEEE Standard 802.15.1; 2. Suporte ao FHSS; 3. Melhorias no processo de sincronização; 4. Taxa de transmissão de 721 kbps.
Bluetooth 2.x + EDR	1. Secure Simple Pairing (SSP); 2. Enhanced Data Rate (EDR); 3. Melhorias no consumo energético com Sniff Subrating; 3. Security Mode 4;
Bluetooth 3.x + HS	1. Melhorias nas taxas de Transferência com o novo modo de operação AMP.
Bluetooth 4.x	1. Introdução do BLE; 2. Menos consumo energético; 3. *Attribute Protocol*; 4. *Generic Access Profile*; 5. Criptografia AES; 6. *Security Manager*; 7. Conexões seguras nos modos de operação BR/EDR/LE; 8. Suporte à UUIDs de 32 bits. 9. Ajustes na Piconet; 10. *Advertising* em intervalos rápidos; 11. Autenticação GAP;
Bluetooth 5.x	1. Maior taxa de transferência; 2. Aprimoramentos na coexistência de outros equipamentos operando na mesma frequência com o Slot Availability Mask (SAM); 3. Melhorias no alcance do sinal.

Tabela 3.1: Evolução tecnológica do Bluetooth

BR	**LE**
Transferência de Arquivos, Streaming de Audio, etc	Aplicações que funcionem com taxas de transferência inferior a 2 Mbp/s.
Não é otimizado para operar em baixo consumo energético	Baixo consumo energético.
79 RF channels	40 RF channels.

Tabela 3.2: Principais diferenças entre o *Bluetooth Low Energy* e o *Basic Rate*.

Na presente dissertação, os dados transferidos são projetados para atender a taxa de transferência disponibilizada pelo BLE e, por este motivo, tal taxa de transferência é suficiente. Por fim, como a presente dissertação tem como um dos principais requisitos, obter uma solução de baixo consumo energético, foi optado pelo uso do BLE.

3.2.2 Conceitos Técnicos do Bluetooth Low Energy

Conforme afirmação de [6], o core ou, ainda, o sistema central do Bluetooth, compreende um *Host* e um, ou mais, *Controllers* (Figura 3.2), os quais são definidos como sendo as

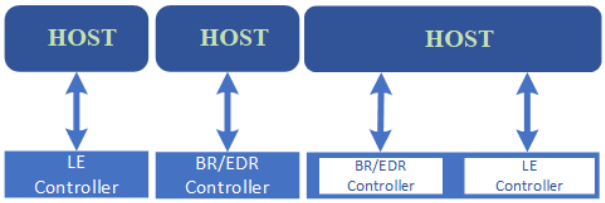


Figura 3.2: Associação entre *Host* e *Controllers*
Fonte: Adaptação de [6]

duas principais entidades lógicas dentro do *core* do Bluetooth.

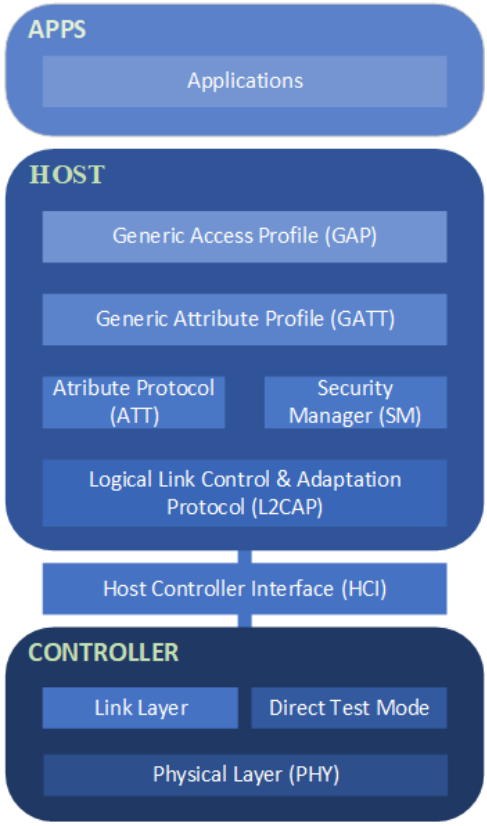


Figura 3.3: Arquitetura do *Bluetooth Low Energy*
Fonte: Adaptação de [8]

Em linhas gerais, é no *Host* onde estão todas as camadas abaixo dos perfis considerados como não essenciais, e acima do *Host Controller Interface* (HCI). Este, por sua vez, é um protocolo que permite a comunicação entre o *Host* e o *Controller*, isto é, no HCI são enviados os comandos do *Host* para o *Controller*, assim como os eventos do *Controller*

para o *Host*. Quanto ao *Controller*, este é definido como todas as camadas abaixo do HCI. Essa arquitetura, ou, pilha de protocolos, pode ser visualizada na Figura 3.3 [6].

3.2.3 Camadas de Baixo Nível

O *Controller* contém em si a *Physical Layer* (Camada Física, ou ainda PHY), a *Link Layer* (Camada de Enlace) e o *Direct Test Mode* (DTM), é no *Controller*, mais especificamente, na PHY, que comummente são implementados os *Systems on Chip* (SoCs) com um rádio integrado para comunicação e modulação/desmodulação dos dados.

No que diz respeito as taxas de transferência do BLE, utilizando o Bluetooth 4, é possível obter taxas de até 1 Mbp/s, isso porque é utilizada uma variante da PHY nomeada como LE 1M. Existem outras duas variantes, implementadas no Bluetooth 5, que são, a LE 2M, cuja taxa de transferência pode atingir 2 Mbp/s e a LE Coded que, por sua vez permite um alcance até quatro vezes maior que a LE 1M, mas com uma taxa de transferência de até 512 Kbp/s [8].

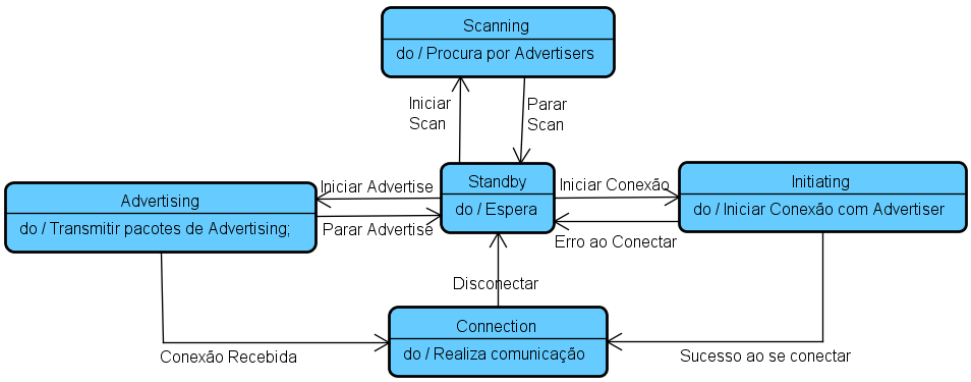


Figura 3.4: Diagrama de Estado de Máquina dos estados da *Link Layer*
Fonte: Adaptação de [9]

Na pilha de protocolos (Figura 3.3), há uma camada de enlace (*Link Layer*), a qual em linhas gerais interage com a PHY, e também é responsável por gerir algumas operações, tais como, geração de números aleatórios e aplicação de criptografia. A *Link Layer* ainda, pode ser explicada por uma máquina de estados contendo os seguintes estados,

Advertising[4], *Connection*, *Initiating*, *Scanning*[5] e *Standby* [9].

A Figura 3.4[6], representa as funções dos estados, assim como as interações entre cada estado, dentro do contexto da *Link Layer* [9].

- No ***Scanning State***, o *scanner*[7] faz varreduras com o objetivo de encontrar dispositivos que estejam realizando *advertise*;
- O ***Advertising State*** é um estado onde é feito o *broadcast* dos pacotes de *advertising*, e, eventualmente, a resposta aos pacotes de *scanning*;
- O ***Connection State*** é um estado acedido partir do *Initiating State* ou do *Advertising State*. Na primeira situação, quando a transição de estados é feita a partir do *Initiating State*, o *Connection State* assume funções de *Master*[8]. Caso a transição de estados ocorra a partir do *Advertising State*, o *Connection State* assume funções de *Slave*[9];
- No ***Initiating State***, o *scanner* estabelece uma conexão com o *advertiser*.
- No ***standby***, o dispositivo não recebe nem transmite dados.

Diante dessa visão geral no tocante à *Link Layer*, será dado sequência na explicação teórica abordando o L2CAP que, por sua vez, é um protocolo cujos requisitos funcionais são a multiplexação[10] de protocolo/canal, segmentação e remontagem, controle de fluxo por canal e de erro. É também neste protocolo que são garantidos que os contratos do *Quality of Service* são cumpridos [10].

A principal responsabilidade do L2CAP é fornecer serviços de dados para protocolos das camadas superiores, sejam estes serviços orientados à conexão ou sem conexão. É

[4]As variantes do verbo *advertise* são utilizadas para referenciar o ato de propagar dados (*broadcast*).

[5]As variantes do verbo *scan* são utilizadas para referenciar o ato de procurar dispositivos que estão propagando dados.

[6]Advertiser é um dispositivo que faz o *advertising*.

[7]Scanner é um dispositivo que faz o *scan*.

[8]Na *Link Layer* o *Master* tem como principal função definir alguns parâmetro de transmissão

[9]O *Slave* se comunica com um único *Master* atendendo os requisitos definidos pelos parâmetros do *Master*

[10]Reunir dados de diferentes origens e transmiti-los ao mesmo tempo, em um mesmo canal/protocolo.

através do L2CAP que os protocolos e aplicativos de alto nível podem enviar e receber dados da camada superior até 64 Kb de tamanho [10]. Estas camadas superiores serão abordadas logo na sequência.

3.2.4 Camadas de Alto Nível

O primeiro conceito a ser explicado é o *Security Manager*, que é um protocolo utilizado para gerar e gerir o armazenamento de chaves de criptografia e identidade. Este protocolo ainda fornece estas chaves armazenadas para que o *Controller* as possa utilizar no processo de encriptação e autenticação durante o emparelhamento [6].

Na sequência é explicado o *Attribute Protocol* (ATT) que, por sua vez, é um protocolo para descoberta, leitura e escrita de atributos em um dispositivo. O ATT compreende uma relação Cliente-Servidor, onde a função do servidor é manter um conjunto de atributos e seus possíveis valores, para que estes possam ser compartilhados com o cliente [11].

A Tabela 3.3 mostra um exemplo de como funcionam os atributos e seus valores. Nesta é possível perceber que cada atributo dentro da tabela é um valor ou parte da informação com algumas propriedades associadas.

ID do Atributo	**Descrição**	**Valor**	**Permissões**
0x7101	Valor da Temperatura	21	Leitura
0x7102	Valor da Humidade	43	Leitura
0x7103	Temperatura Minima	-5	Escrita
0x7104	Temperatura Máxima	5	Escrita
0x7105	ID do Operador Atual	abcdef01	Escrita
0x7103	ID do Remetente	123456ab	Escrita

Tabela 3.3: Exemplo de atributos organizados em formato de Tabela.

Portanto, para que um cliente saiba o valor de um atributo, é necessário consultar uma linha da tabela, por exemplo, para saber o valor do sensor de temperatura, é consultada a primeira linha desta tabela.

As permissões podem ser, em linhas gerais, para escrita, leitura, notificação ou indicação. A notificação e a indicação são mecanismos em que, sempre que o valor é da característica sofre alteração, é enviada uma notificação com o novo valor ao dispositivo

cliente. No entanto, a notificação envia dados sem que o cliente tenha que enviar qualquer pacote de confirmação de receção - análogo ao protocolo UDP. Enquanto a indicação, é preciso que na camada da aplicação do protocolo seja implementado mecanismos de confirmação de receção da mensagem.

O ATT é construído sob o *Generic Attribute Profile* (GATT) que, por sua vez, define com detalhe uma estrutura de como os dados são trocados entre *Profile* e usuários dentro de uma conexão BLE [12]. Os elementos são contidos nos atributos utilizados pelo ATT para transportar os dados do *Profile*, que por sua vez está no nível mais alto da hierarquia do GATT, conforme pode ser observado na Figura 3.5.

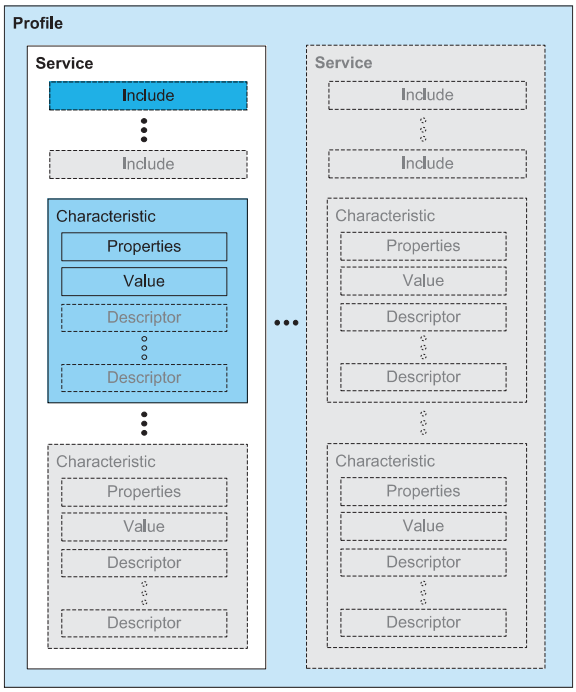


Figura 3.5: Hierarquia do GATT.
Fonte: **Core.Vol3G**

Na hierarquia, um ***Profile*** pode ser resumido como sendo uma aplicação BLE e pode conter um ou muitos *Services*, assim como estes podem conter uma ou mais *Characteristics* que, por sua vez, podem ter um ou mais *Descriptors*.

O ***Service*** é definido conceptualmente como sendo uma coleção de dados e comportamentos para implementar uma função ou um recurso. Uma ***Characteristic*** é, por definição, um valor utilizado em um *Service*, que contém *Properties*[11] e informações sobre a exibição ou representação deste valor. Por fim, o ***Descriptor*** é utilizado para incluir mais informações sobre a *Characteristic*, como exemplo, dica sobre os valores [12].

Em linhas gerais, os *Services* e as *Characteristics* existem como uma maneira de obter flexibilidade e compatibilidade entre diferentes plataformas, além de facilitar a implementação. Por exemplo, quando há um dispositivo fazendo *advertising* de um *Service* para lidar com informações de temperatura, e um *smartphone* deteta este serviço, é garantido que, independente da plataforma, será apresentado um valor padronizado referente à temperatura.

Como último conceito dentro do *Host*, tem o GAP, o qual permite que dispositivos BLE interajam entre si, por meio de uma estrutura, que qualquer implementação de BLE deve seguir para executar operações fundamentais, tais como transmitir dados e estabelecer conexão. No GAP há *Characteristics*, que contêm dados como nome do dispositivo e parâmetros de conexão [13].

Por fim, a camada da Aplicação é responsável por conter a lógica de negócio, a interface do usuário e também lidar com os dados relacionados com o dispositivo BLE.

3.2.5 Sumário

Desde o lançamento do Bluetooth até os dias de hoje, ocorreu uma série de evoluções tecnológicas no protocolo e novas melhorias estão sendo realizadas a cada dia. Melhorias estas que estão relacionadas com a distância máxima para o alcance do sinal, taxas de transferência, consumo energético e até mesmo a utilização do protocolo para serviços de localização.

[11] Uma tabela com as *Properties* de uma *Characteristic* pode ser visualizada no Apêndice 8.

Tais evoluções permitem que o Bluetooth seja utilizado na implementação de novas soluções, como exemplo, a solução aqui proposta. O Bluetooth ainda tem a vantagem de estar presente nos diversos dispositivos eletrónicos, como *Notebooks*, Televisores, *Smartwatches* e em especial nos *Smartphones*. Fator este que facilita a combinação dessas tecnologias. O conceito de *smartphone* é de seguida explicado em detalhe.

3.3 Smartphones e seus Sistemas Operativos

Em linhas gerais, *smartphones* são telefones móveis com recursos adicionais como Sistema Operativo (SO), acesso à Internet, câmara, GPS, Bluetooth, entre outros. Para [14], os *smartphones* foram originados por meio de funções como agenda de compromissos e visualizadores de documentos do PDA .

No que concerne aos SOs, para Smartphones atualmente há dois principais, sendo estes o Android e o iPhone OS (nome este que posteriormente fora alterado para iOS). Segundo uma pesquisa realizada pelo site [15], em 2016 os Smartphones com SO Android e iOS foram responsáveis por cerca de 1,5 bilhões de aparelhos vendidos. Ainda, segundo [16], é estimado que o número de usuários de Smartphones no planeta em 2019 ultrapasse os 2,5 bilhões.

Conforme [17], o Android teve seu desenvolvimento iniciado em 2003 pela *startup* Android Inc e em 2005 foi adquirido pela Google, que para [18] foi o responsável por gerir grande parte do desenvolvimento da plataforma.

Em Junho de 2006, os dois primeiros dispositivos físicos nomeados por “Sooner” e “Dream” foram utilizados para planear o desenvolvimento de *software* para o Android. O “Sooner"foi baseado em um smartphone com teclado QWERTY e sem entrada por Toque na Tela - do inglês, *Touch Screen* (TS), visando ter um produto inicial o mais rápido possível. Já o “Dream” foi projeto para executar o Android totalmente como previsto, o que inclui a entrada por TS, teclado QWERTY deslizante, acelerômetro, GPS, bússola e conexão móvel 3G [18].

Apenas em 2009 a Google lançou o Android que, segundo [19], é o líder mundial

no segmento de SOs para computadores portáteis. Ainda segundo [18], o mercado de Smartphones e Tablets está sendo dominado pelos SOs Android e iOS.

3.3.1 Fundamentos do Android

O primeiro SO a ser abordado neste documento é o Android que, de acordo com [18], é um SO baseado no *kernel* do **Linux** e introduz apenas alguns conceitos novos no próprio *kernel*. No que diz respeito ao iOS, [18] afirma que este é derivado do UNIX.

A segurança do Android é totalmente baseada na segurança do Linux. Cada aplicação no Android é executada em um único processo e cada processo tem uma thread dedicada. Quando uma aplicação é instalada, um usuário no SO é criado para ter acesso à estrutura de diretórios, assim nenhum outro usuário pode ter acesso a tal aplicação [19].

Sempre que um aplicativo único é iniciado, um novo processo no SO é disparado, alguns podem apresentar uma Interface Gráfica do Usuário - acrónimo do inglês, *Graphical User Interface* - (GUI) e outros apenas executar em segundo plano. Muitos aplicativos e processos podem ser executados de maneira simultânea, quem se responsabiliza por gerir a memória é o próprio *kernel* [19]. Mais adiante, na Secção 3.6 serão abordados mais alguns conceitos de segurança nos dispositivos Android.

Uma grande parte do SO Android é escrito em Java - a qual é uma Linguagem de Programação (LP) de alto-nível -, toda a API do aplicativo é escrita e publicada em Java, tendendo a seguir os padrões de orientação a objetos. O *kernel* e um número grande de bibliotecas de baixo-nível são escritos em LP C e C++ [18].

Para disponibilizar um local onde os programadores possam publicar suas aplicações para que os usuários realizem o *download* e instalação das mesmas, foi criada uma loja de aplicações online, designada por Google Play (GP), que foi inicialmente chamada de Android Market [19]. A GP ainda, tem um mecanismo para controlar a compatibilidade entre aplicativos e *smartphones*, por isso é relevante que o aplicativo utilize uma versão coerente à sua finalidade.

3.3.2 Fundamentos do iOS

O iOS é um SO desenvolvido e distribuído pela Apple Inc. e tem sido aprimorado para operar em dispositivos como iPad e Apple TV. O SO foi disponibilizado juntamente com o primeiro iPhone em 2007 e, diferente do Android, a instalação não é licenciada para dispositivos que não sejam oficiais da Apple, tal qual o OS X. Este que, por sua vez deu origem ao iOS e inclusive é construído também sobre o Darwin OS. Por tal motivo, este pode ser considerado como uma variação do UNIX [18], [20], [21].

Um dos conceitos relevantes do iOS é a maneira como a Interface do Usuário - do inglês, *User Interface* (UI) é tratada. A UI do iOS é desenvolvida com base em gestos multi-toque - nomeadamente *swipe*, *tap*, *pinch* e *reverse pinch*, ou seja, certos padrões são utilizados para realizar ações contextuais dentro do SO, por exemplo, ampliar imagens e minimizar uma aplicação colocando-a em segundo plano para execução. Além dos gestos multi-toque, é possível realizar ações por meio de botões, interruptores e até mesmo com o uso de sensores como acelerômetro e giroscópio. Este método de interação pode propiciar uma melhor Experiência do Usuário - acrónimo do inglês, *User Experience* - (UX) e por tal motivo é deveras importante garantir que toda ação realizada pelo usuário terá uma resposta, fornecendo assim uma interface fluida [21].

Com o propósito de garantir um nível de qualidade das aplicações disponíveis para o usuário na Apple Store (AS), a Apple desenvolveu um documento com *guidelines* (tradução livre, diretrizes) para que os programadores sigam com certo rigor. Portanto, para publicar uma aplicação desenvolvida para iOS, é necessário cumprir com as *guidelines*[12] expostas na documentação oficial da Apple [13], caso contrário a aplicação pode ser rejeitada e sua publicação na loja oficial da Apple - nomeadamente AS - não será realizada.

3.3.3 Arquitetura do iOS

De acordo com [22], por questões de segurança, os aplicativos desenvolvidos para iOS raramente se comunicam com o *hardware* do dispositivo de maneira direta, para isso foi

[12] A guideline pode ser conferida no Apêndice 8;
[13] https://developer.apple.com/documentation/;

desenvolvida uma API que possibilita o uso de certos recursos de *hardware* do iOS. Mais adiante, na Secção 3.6 serão abordados com mais detalhes alguns conceitos de segurança no tocante ao iOS.

No iOS há quatro camadas de abstração em que cada qual provê um conjunto de *frameworks* disponíveis para serem utilizados no desenvolvimento de aplicativos para o SO iOS. Tais camadas serão abordadas a seguir [21]–[25]:

Core OS é o *kernel* do SO, que gere recursos de baixo nível como suporte a *threads*, *sockets*, memória, criptografia, Bluetooth, processamento de imagem e áudio. Nesta camada também estão presentes *frameworks* como o

- *Accelerate* que disponibiliza métodos para computação de alto desempenho e eficiência energética na CPU. Este *framework* também contém bibliotecas que executam instruções apropriadas de acordo com a disponibilidade do processador em tempo de execução;
- *ExternalAccessory* que fornece interfaces para comunicação de acessórios conectados a um dispositivo baseado no iOS, como o Bluetooth.

***Core Service*:** contém os serviços de sistema fundamentais, que são divididos em frameworks e baseados nas LPs C e Objective C. É nesta camada que estão implementados aplicações de serviços básicos como Banco de Dados SQLite, Contatos, Calendário, *In-app Purchase* (em tradução livre, compras dentro dos aplicativos), Rede, Gerenciamento de Dados e Localização.

***Media*:** é uma camada que fornece *frameworks* de alto-nível para o uso de tecnologias como gráficos 2d, 3d, áudio e vídeo, simplificando o desenvolvimento de aplicativos que fazem o uso dos mesmos. Alguns exemplos de bibliotecas presentes nesta camada são *Core Graphics*, *Core Animation*, *Core Audio*, *Core Media* e *Assets Library*;

***Cocoa Touch*:** esta camada contém os principais *frameworks* para o desenvolvimento de aplicativos iOS, fornecendo um conjunto de métodos que facilitam a implementação de serviços como Apple Push Notification, Multitarefa, Acesso a dados do

dispositivo e TS.

Portanto, é possível afirmar que as camadas podem ser divididas por alto e baixo nível, onde a *Core OS* e a *Core Service* são caracterizadas como baixo-nível, enquanto a *Media* e *Cocoa Touch* de alto-nível.

No que concerne ao desenvolvimento de aplicações utilizando tecnologias nativas para iOS, podem ser utilizadas as LPs Objective C e Swift. Diante destas opções, cabe ao responsável pelo desenvolvimento fazer uma análise de ambas para selecionar a mais viável. O mesmo é válido no desenvolvimento de aplicativos Android. No entanto, se as tecnologias nativas não forem as mais adequadas, é possível ainda utilizar tecnologias que permitam criar aplicativos para várias plataformas utilizando o mesmo código-fonte para todas. Tais tecnologias serão abordadas na subsecção 3.3.4.

3.3.4 Desenvolvimento de Aplicativos para Android e iOS

Para desenvolver aplicações para *smartphones* é possível utilizar uma vasta gama de tecnologias, que podem ser categorizadas em *WebApp*, *Cross-platform* e Nativa, cada uma com seus prós e contras.

As tecnologias Nativas criam o aplicativo para uma plataforma específica, por isso, os custos de desenvolvimento podem aumentar, isso porque é necessário escrever códigos diferentes para cada plataforma. Em contrapartida, no desenvolvimento nativo é possível ter acesso antecipado aos novos recursos disponibilizados por cada plataforma, além de permitir o acesso a todas as funcionalidades do dispositivo.

Uma outra vertente do desenvolvimento de aplicativos é o uso das tecnologias WebApp, as quais permitem utilizar um mesmo código-fonte escrito em HTML, CSS e JavaScript, tendo como propósito criar aplicativos para cada plataforma. Os WebApps são executados em um ambiente Web, em alguns casos é preciso ter conexão com a Internet, em outros não[14] [26].

Com o uso das tecnologias *cross-platform* é possível criar aplicativos para diferentes

[14]Como no caso do Progressive WebApp, https://developers.google.com/web/progressive-web-apps/ ;

plataformas utilizando o mesmo código-fonte. Para isso, existem várias ferramentas com suas distinções, em algumas, o aplicativo é criado para ser executado em um ambiente específico, ou ainda, ambiente híbrido, enquanto outras, geram o código nativo para cada plataforma, utilizando como base, o mesmo código-fonte. Um exemplo é o Xamarin, que permite ainda, a escrita de código nativo específico para cada plataforma utilizando o C# como LP [26].

3.3.5 Sumário

Diante do que foi apresentado nesta secção, é possível afirmar que existem muitas diferenças no tocante aos SOs para *smartphones*. Tais diferenças impactam no desenvolvimento de aplicações para os mesmos e, por isso, existem várias ferramentas de desenvolvimento, tais como Cordova, Xamarin[15], Kivy[16], Beeware[17], Flutter[18], Ionic [19], etc.

Além das tecnologias para desenvolvimento de aplicações para *smartphones*, há também tecnologias para a comunicação entre estes dispositivos, tais como Bluetooth e Internet. No tocante a Internet, esta tecnologia permite realizar a comunicação com diferentes serviços, tais como os Webservices, servidores de transferência de arquivos, entre outros. Logo na sequência, serão abordados estes serviços.

3.4 Computação em Nuvem

De acordo com [27], a computação em nuvem é a disponibilização de uma plataforma de serviços via Internet, fornecendo armazenamento de banco de dados, acesso à aplicações e recursos computacionais. Nesta dissertação, o termo Serviço na Nuvem - do inglês, *Cloud Service* (CS) é utilizado para referenciar uma plataforma de computação em nuvem.

Quanto aos serviços, o *WebService* (WS) é uma tecnologia que permite aos dispositivos

[15]https://docs.microsoft.com/en-us/xamarin/
[16]https://kivy.org/doc/stable/
[17]https://pybee.org/
[18]https://flutter.io/
[19]https://ionicframework.com/docs/

conectados comunicarem-se entre si, independente da plataforma ou linguagem de programação. Neste processo de comunicação é necessário que haja uma interface padronizada para troca de mensagens, neste caso, pode ser utilizado o estilo arquitetónico *Representational State Transfer* (REST)[20] que, em linhas gerais, possibilita realizar operações nos dados utilizando os métodos GET, POST, PUT e DELETE.

Além da API REST, é necessário também que se tenha um Banco de Dados (BD)[21], que, por sua vez é importante para realizar operações em dados, mas, em alguns casos, não é adequado para lidar com arquivos. Por este motivo pode ser necessário que se tenha ,também, um serviço para lidar com transferência de arquivos, nomeadamente, um servidor FTP (*File Transfer Protocol*).

Portanto, todos os serviços mencionados nesta secção são, ou ao menos podem ser, utilizados no desenvolvimento de plataformas de computação na nuvem, ou ainda, o CS, conforme é possível observar na Figura 3.6.

Figura 3.6: Tecnologias em *Cloud Service*.

Geralmente, o CS é o principal componente para armazenar e distribuir os dados em muitas soluções. Neste caso, é importante que o CS tenha mecanismos para garantir a integridade de tais dados, evitando que estes sejam adulterados ou obtidos por usuários não autorizados. Conforme mencionado na secção 2.3, a solução SmarTrack.IO contém mecanismos para lidar com tal situação, no entanto, há possibilidade em acrescentar algumas medidas de segurança para fins de integridade dos dados, fazendo o uso de blockchain, que é explicado na sequência.

[20] A definição de REST pode ser encontrada no Anexo 8.
[21] A definição de BD pode ser encontrada no Anexo 8.

3.5 Blockchains

O blockchain é uma estrutura de dados em formato de lista ordenada e vinculada com blocos de transações e que pode ser armazenado em um BD ou até mesmo como um arquivo. Os blocos são identificados por um *hash* gerado por meio do algoritmo de criptografia SHA256 que vai no cabeçalho de cada bloco. Nestes blocos, também há uma referência ao *hash* do bloco anterior dentro da cadeia, que por sua vez é conhecido como sendo o bloco "pai". Esta estrutura básica pode ser visualizada na Figura 3.7 [28].

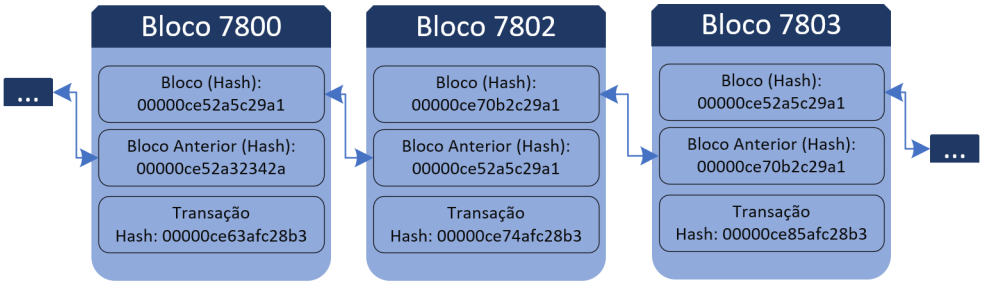


Figura 3.7: Estrutura básica do blockchain
Fonte: Adaptado de [29].

Os blocos ainda podem ter vários filhos, no entanto, cada filho tem apenas um pai. Neste caso, cada um dos "filhos" faz referência ao mesmo pai. Portanto, uma cadeia é criada a partir da vinculação do primeiro bloco com os demais, onde o primeiro bloco é identificado como sendo *The Genesis Block*, ou em tradução direta, o bloco de génese [28].

Um dos aspetos mais relevantes do blockchain é que este contém a prova devidamente verificada das transações realizadas e ainda elimina a possibilidade de adulteração dos dados por um usuário malicioso [29]. Este fator é importante para garantir a fiabilidade e integridade dos dados.

Há quatro conceitos chave no blockchain que permitem uma compreensão de seu funcionamento, são estes, *Shared Ledger*, *Permissions*, *Smart Contract* e *Consensus*. Os detalhes destes conceitos são apresentados no Apêndice 8.

O uso do blockchain vem revolucionando a segurança e a veracidade das informações

de diversas áreas, dentre estas está a logística. Com o blockchain é possível compartilhar informações do transporte da mercadoria com todos os participantes na rede de forma segura, aprimorar os mecanismos para garantir que os dados coletados são fidedignos e até mesmo aplicar os termos do seguro de transporte automaticamente. Porém, existem soluções mais simples, que podem cumprir objetivos parecidos. Há também algumas situações de segurança que devem ser analisadas, porque estas podem ser exploradas em soluções IoT, algumas destas são analisadas logo na sequência.

3.6 Segurança em Bluetooth Low Energy, Smartphones e *Cloud Service*

A presente dissertação tem requisitos relacionados com fiabilidade dos dados. Para isso, foi definido um ciclo com diversos atores que se relacionam. Cada autor tem suas funções na solução e só pode realizar a si associadas. Também foram definidos mecanismos para contextualização dos dados, afim de evitar ou detetar quando estes são adulterados. No entanto, é necessário que haja mecanismos de segurança para evitar ataques realizados por usuários maliciosos, não necessariamente envolvidos no processo de transporte, mas que podem colocar em risco a fidedignidade das informações recolhidas. Tais ataques podem ocorrer em quaisquer dos componentes desenvolvidos (SmarTrack Device, Aplicativo ou CS).

3.6.1 Tipos de Ataque

Há alguns tipos de ataques comuns que a solução está propensa a sofrer. Como exemplo o, Eavesdropping, *Man-in-the-Middle* (MITM), SQLInjection, Spoofing, Negação de Serviço e *Brute Force*.

O **Eavesdropping** é um ataque que pode ser dividido em *Passive Eavesdropping* e *Active Eavesdropping*. O *Passive Eavesdropping* utiliza código malicioso para obter informações das mensagens de *broadcast*. Enquanto o *Active Eavesdropping* ocorre quando

o invasor captura as informações através de consultas discretas [30].

Os ataques **MITM** ocorrem quando um usuário deseja realizar a conexão entre dois dispositivos e neste processo, há um terceiro dispositivo infiltrado, o qual recebe o pedido de conexão fazendo com que os dispositivos se conectem a este sem ter conhecimento. Este terceiro dispositivo coleta os dados transmitidos nesta conexão [6].

A diferença entre o Eavesdropping e o MITM é que no MITM é possível alterar dados, enquanto no Eavesdropping é possível apenas intercetar tais dados, por este motivo, o Eavesdropping é um ataque utilizado para dar início a outros ataques utilizando os dados coletados.

Outro ataque comum, é o Spoofing que, por sua vez é um ataque onde os criminosos falsificam informações dos usuários para obter acesso a informações confidenciais. As informações falsificadas podem ser endereço IP e MAC, e-mails, DNS, entre outros [31].

Um dos ataques mais destruidores em uma solução é o SQL Injection que, por sua vez, é uma vulnerabilidade que permite ao invasor alterar a execução de um determinado código SQL para realizar operações em dados confidenciais dos usuários, como dados bancários e *passwords*(palavras-passe). Importante ressaltar que não é apenas a partir de aplicações Web que podem ser realizados ataques de SQL Injection.

Por fim, um ataque, que em muitos casos, pode ser efetivo e simples de executar, é o *brute force*, ou força bruta, que se baseia no processo de tentativa e erro para obtenção de acesso não autorizado a dados confidencias do usuário.

3.6.2 Estratégias de Mitigação

No tocante ao Bluetooth, este tem suas vulnerabilidades e seus respetivos mecanismos de proteção contra ataques. Assim como os ataques, os recursos de segurança no Bluetooth tem evoluído. Atualmente, o modelo de segurança do Bluetooth contém cinco conceitos, que são [6]:

- *Pairing* (Emparelhamento) - processo de criação de uma ou mais *secret keys* compartilhadas;

- *Bonding* (Vinculação) - é um processo de armazenamento de *keys* durante o emparelhamento, tornando possível reutiliza-lo em conexões futuras;
- *Device authentication* (Autenticação do Dispositivo) - verifica a autenticidade de dois dispositivos fazendo o uso de duas *keys*, uma de cada dispositivo;
- *Encryption* (Encriptação) - ato de tornar as mensagens confidenciais utilizando o algoritmo E0;
- *Message integrity* (Integridade da Mensagem) - proteção contra falsificação de mensagens utilizada em conexões encriptadas.

Ainda no que diz respeito à segurança, existe um conceito designado por *security mode*. Em uma conexão BLE, o GAP define dois (*security modes*) diferentes, são estes o LE *Security Mode 1* e LE *Security Mode 2*, que compreendem em si o conceito de *Security Level*. Estes conceitos estão relacionados com procedimentos de segurança utilizados no protocolo. O *security mode* 1 aprimora a segurança com encriptação, enquanto o *security mode* 2, faz o mesmo utilizando assinatura nos dados, conforme mostra a Tabela 3.4.

LE Security Mode 1	
Level 1	Sem Autenticação e Criptografia
Level 2	Emparelhamento encriptado mas sem autenticação
Level 3	Emparelhamento encriptado e com autenticação
Level 4	Emparelhamento encriptado e utilizando LE Secure Connections[a]
LE Security Mode 2	
Level 1	Emparelhamento sem autenticação mas com assinatura nos dados
Level 2	Emparelhamento autenticado e com assinatura nos dados

[a]Os detalhes do LE Secure Connections pode ser visto em http://blog.bluetooth.com/bluetooth-pairing-part-4

Tabela 3.4: *Security Modes* e seus respetivos *Levels*. Adaptado de [13]

O *National Institute of Standards and Technology* recomenda a utilização do *Security Mode* 1 *Level* 3 por considerar que esta é a mais segura que as outras, além de não

recomendar o *Security Mode 1* Level 1 em hipótese nenhuma.

Algumas estratégias para mitigação de ataques são resolvidas a nível de implementação, seja por parte do fornecedor do Kit de Desenvolvimento de *Software* - do inglês, *Software Development Kit* (SDK), ou pelo próprio desenvolvedor do produto final. Algumas estratégias de mitigação de ataques no BLE, podem ser visualizadas na Tabela 3.5.

Ataque	**Estratégia de Mitigação**
BluePrinting	Manter oculto o Device Address
Reflection attack	Utilizar criptografia e manter oculto o Device Address
Repeatable authentication attempts	Limitar os pedidos de autenticação
Blueover	Manter oculto o Device Address
Static SSP pass keys	Passkeys para cada emparelhamento
No authentication	Aplicar segurança na camada da Aplicação
Bluesnarfing	Utilizar Non-Discover mode.
Pin Cracking	Utilizar pin codes aleatórios
MIM/Imperson action Attack	Criptografia na conexão e validação de chaves por combinação
Pairing Eavesdropping	Reduzir emparelhamento

Tabela 3.5: Tabela de ataques e estratégias de mitigação [32]

Em relação aos aplicativos para *smartphones* Android e iOS, uma das principais recomendações é não instalar aplicações de fontes externas, isto é, sempre instalar aplicações pela loja virtual de cada dispositivo. Isso porque estas têm mecanismos de segurança, como análise de código, experiência do usuário, testes de segurança do código, entre outros. Tais políticas tendem a mitigar a execução de códigos maliciosos por parte dos aplicativos, conforme mencionado no relatório de segurança do Google em [33].

Para isso, é fundamental que sejam aplicadas técnicas de segurança no código-fonte para mitigar tais ações, tais como, criptografia, dados assinados, limitação de permissões de acesso a funções, autenticação e autorização.

Outro mecanismo de segurança utilizado pelos dispositivos Android e iOS para mitigação de ataques, é o uso do *sandbox* que, por sua vez, é um ambiente que executa o código dentro de uma área restrita, permitindo apenas que um determinado aplicativo tenha acesso apenas aos dados associados com o identificador desta aplicação, conforme afirmação de [24].

Para dispositivos Android, o Google faz algumas recomendações para que os desenvolvedores possam mitigar alguns ataques, conforme pode ser visto em [34]. As dicas são, não armazenar dados confidencias no armazenamento externo, solicitar apenas permissões essenciais para o aplicativo, utilizar HTTPS e/ou SSLSocket, não confiar em dados oriundos de conexões HTTP, minimizar o uso de APIs que tenham envolvimento com os dados confidenciais do usuário, utilização de Tokens de autorização, criptografia, entre outras.

Quanto aos dispositivos iOS, assim como o Google, a Apple também fornece recomendações para mitigar alguns ataques. Dentre as recomendações estão, não armazenar dados em diretórios públicos do dispositivo, armazenar as *kernel extensions* em locais confiáveis e seguros, carregar apenas *kernel extensions* confiáveis, autorização para que os usuários tenham acesso apenas aos serviços pertinentes a ele, evitar tentativas excessivas de conexão em servidores com problemas de conexão (seja por ataques de negação de serviço, ou por indisponibilidade temporária), utilizar as ferramentas disponibilizadas pela Apple para lidar com *passwords*, nunca enviar dados confidenciais sobre protocolos considerados como não seguro, etc. Todas estas recomendações são encontradas em [35].

Por fim, há também medidas de segurança para mitigar os ataques em *Webservices*. Algumas recomendações sugeridas por [36] são a utilização de autenticação, autorização, criptografia em dados confidenciais, assinatura digital em mensagens, contextualizar dados em termos temporais evitando que a mensagem seja propagada mesmo sendo inválida, troca de chaves, etc. Além destas, há também medidas de segurança utilizadas por *frameworks* como EJB[22] e Hibernate[23] para mitigar ataques de SQL Injection. No entanto, há também algumas recomendações por estes que devem ser seguidas para mitigar alguns ataques, como evitar o uso de consultas SQL nativas.

[22]https://docs.oracle.com/cd/E24329_01/web.1211/e24446/ejbs.htm
[23]http://hibernate.org/orm/documentation/5.3/

3.6.3 Sumário

Diante do que foi apresentado nesta secção, é possível afirmar que uma solução para IoT está sujeita a inúmeros ataques que podem pôr em risco toda a solução, haja vista que o roubo de informações confidenciais podem impactar em prejuízos financeiros e morais tanto para quem faz o uso da solução, tanto para quem a desenvolve. Por este motivo, é importante realizar um estudo prévio para implementação de técnicas de segurança para mitigação, tanto dos ataques mencionados nesta secção, bem como outros que não foram mencionados explicitamente neste documento.

Além da segurança, muitas soluções existentes utilizam das tecnologias apresentadas neste capítulo para cumprir com seus requisitos. Desta forma, pode-se observar que enquanto este documento estava sendo desenvolvido, foram encontradas soluções similares à proposta neste documento, estes que também estavam em fase de desenvolvimento ou testes, e portanto, não estavam disponíveis para aquisição. Tais exemplos serão abordados a seguir.

3.7 Estado da Arte

Nesta secção serão discorridos alguns exemplos de soluções similares no que se refere à monitorização das condições de transporte da carga. Também serão explicadas as diferenças entre a solução proposta e as existentes no mercado levando em consideração a qual mercado a solução é destinada, a veracidade das informações e a topologia[24] utilizada.

3.7.1 Ambrosus

A solução mais similar encontrada no momento em que o presente trabalho estava sendo desenvolvido, foi a Ambrosus, que por sua vez é definida no whitepaper[25] como sendo um ecossistema para cadeia de suprimentos baseada numa rede blockchain.

[24] O conceito de topologia é abordado na secção 4.3
[25] https://ambrosus.com/assets/en/Ambrosus-White-Paper.pdf

O principal foco desta solução é aprimorar produtos alimentícios e farmacêuticos, no entanto, é afirmado no mesmo documento que o protocolo é escalável para qualquer cadeia de suprimentos. Os dados são coletados a partir de etiquetas, sensores, rastreadores, entre outros. Tais dados podem ser localização, tempo, temperatura, humidade, atributos de qualidade e segurança do produto, condições de transporte, entre outros.

Até o momento em que esta dissertação foi escrita, não foi possível identificar na documentação disponibilizada qualquer informação relacionada aos metodos utilizados para recolha dos dados da mercadoria, isto é, se há um dispositivo específico desenvolvido para este fim, assim como o SmarTrack Device, ou ainda, se será utilizado um *smartphone* dentro de cada carga para recolher tais dados, ou qualquer outro mecanismo com o propósito de recolha de dados de sensores para monitorizar a carga.

Portanto, como não há qualquer confirmação de que há um dispositivo, tampouco foi identificado a topologia utilizada por eventual dispositivo. Por este motivo, não foi possível comparar as soluções no quesito topologia.

Tangente ao mercado, foi possível perceber que a solução é escalável neste quesito, isto é, esta é uma solução teoricamente genérica, que deve funcionar para o transporte de inúmeras mercadorias, sejam estas as de alto valor acrescido, alimentícias, farmacêuticas, entre outras.

Portanto, é possível concluir que esta é uma solução equivalente à solução proposta nesta dissertação, no entanto, conforme mencionado anteriormente, até o presente momento, tal solução não apresentou nenhum produto específico para monitorização de mercadorias, apenas conceitos teóricos de como será feito ou, é feito, como no caso da integridade das informações.

3.7.2 MODSense

Uma outra solução similar encontrada, foi o produto designado por **MODsense**[26], cuja empresa responsável se chama Modum.

[26]https://modum.io/solution/products

O MODSense é considerado pela empresa Modum, como a primeira solução desenvolvida para monitorização de temperatura na indústria farmacêutica. Ele ainda atende aos últimos padrões definidos pela ISO/EN.

Também foi possível identificar que para monitorizar os medicamentos em distribuição, o MODSense faz uso de sensores de temperatura. Tal monitorização ocorre entre as empresas envolvidas, que podem ser empresas do setor de transporte e farmacêutico.

Uma outra característica identificada no tocante ao MODSense é que este realiza leituras sem que haja necessidade em abrir os pacotes, isto é, faz o uso da topologia 2, a qual é abordada na secção 4.3.

É utilizado também blockchain como proposta para garantia de autenticidade e integridade dos dados, tornando estes imutáveis. Além disso, o MODSense utiliza recursos de segurança a nível de *hardware*, os quais não foram detalhados na documentação enquanto o presente trabalho estava sendo desenvolvido.

Portanto, diante de tais informações, é possível afirmar que o principal foco do MODSense está em garantir a integridade de produtos farmacêuticos durante o processo de transporte, onde o público-alvo é a indústria farmacêutica. Por isso, a conclusão que se tem, no tocante ao MODSense, é que este é um serviço que faz uso de tecnologias semelhantes às adotadas nesta dissertação, mas cujos objetivos e modelo de negócio são distintos. Haja vista que o projeto proposto neste documento contém características de um serviço generalista, onde é possível com um mesmo produto, configurá-lo para monitorizar o transporte, seja de animais vivos, medicamentos, alimentos ou órgãos para transplante.

3.7.3 Roambee

Também foi encontrada uma solução nomeada como **Roambee**[27], em que é possível realizar um conjunto de ações para tornar mais eficientes determinados procedimentos logísticos.

[27]https://www.roambee.com/

Tal solução permite o uso de dispositivos nomeados como BeeBeacons em cada pacote/mercadoria para realizar a monitorização de maneira individual destes. Estes dispositivos ainda enviam as informações coletadas em tempo real para um dispositivo central, que são chamados de Bees, que ainda podem ser substituídos por um *smartphone*. Portanto, esta solução se encaixa na topologia 3, mencionada na secção 4.3.

No quesito integridade dos dados, o Roambee também utiliza blockchain, tendo como proposta, mitigar ataques e eventuais alterações dos dados recolhidos pelos sensores para fornecer informações no tocante ao transporte da mercadoria.

Por fim, esta solução assemelha-se tecnologicamente à proposta neste documento, no entanto, o foco do Roambee está em otimizar processos inerentes às empresas de transporte, especificamente, cujo foco está relacionado com a gestão de frotas e mercadorias, e sempre numa perspetiva interna. Portanto, não são processos relacionados diretamente com o cliente ou com a prestação de um serviço afim de obter um controlo efetivo da qualidade do mesmo.

3.7.4 Outros

Foi encontrada uma solução designada por **Averos**[28], que por sua vez é uma solução focada em otimizar procedimentos logísticos, assim como o Roambee. No entanto, a solução ainda não apresenta detalhes sobre seu funcionamento de forma a permitir uma comparação mais detalhada.

Também foram encontrados diversos artigos científicos ([37], [38], [39], [40], [41], [42], [43], [44]) que utilizam tecnologias como RFID, Bluetooth e Zigbee para monitorizar a temperatura de determinadas mercadorias. Dentro deste mercado estão a indústria farmacêutica e alimentícia.

[28]http://www.averos.com/solutions

3.7.5 Sumário

Portanto, foi possível perceber que os principais objetivos destes sistemas são divergentes ao que está sendo proposto neste documento, onde alguns têm como foco, otimizar procedimentos logísticos, enquanto outros estão limitados a um determinado nicho de mercado. Cabe ressaltar ainda que, no momento da escrita da dissertação, nenhum destes sistemas estava já em comercialização, alguns, em estágio mais avançado, estavam em fases de testes. Fato este que reforça a pertinência desta dissertação, no sentido de explorar o potencial tecnológico nesta área de atividade económica.

Após a explicação das tecnologias, no capítulo seguinte serão abordados onde elas se enquadram e como elas serão utilizadas.

Capítulo 4

Metodologia

Previamente à etapa de desenvolvimento, foi necessário aplicar técnicas da Engenharia de *Software* objetivando identificar possíveis falhas conceituais no SmarTrack.IO, evitando alterações, reduzindo a complexidade na compreensão da solução, além de estabelecer de maneira sucinta, as funcionalidades do sistema.

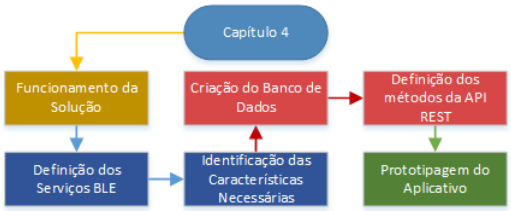


Figura 4.1: Fluxograma - Sequência Explicativa do Capítulo 4

Este capítulo será explicado seguindo a sequência lógica apresentada na Figura 4.1 em formato de fluxograma.

4.1 Visão Geral das Funções dos Atores

Em um primeiro momento, foram identificados os atores e suas respetivas funções dentro da solução (Figura 4.2). Os atores da solução são identificados por usuário, remetente, prestador, operador de transporte e destinatário.

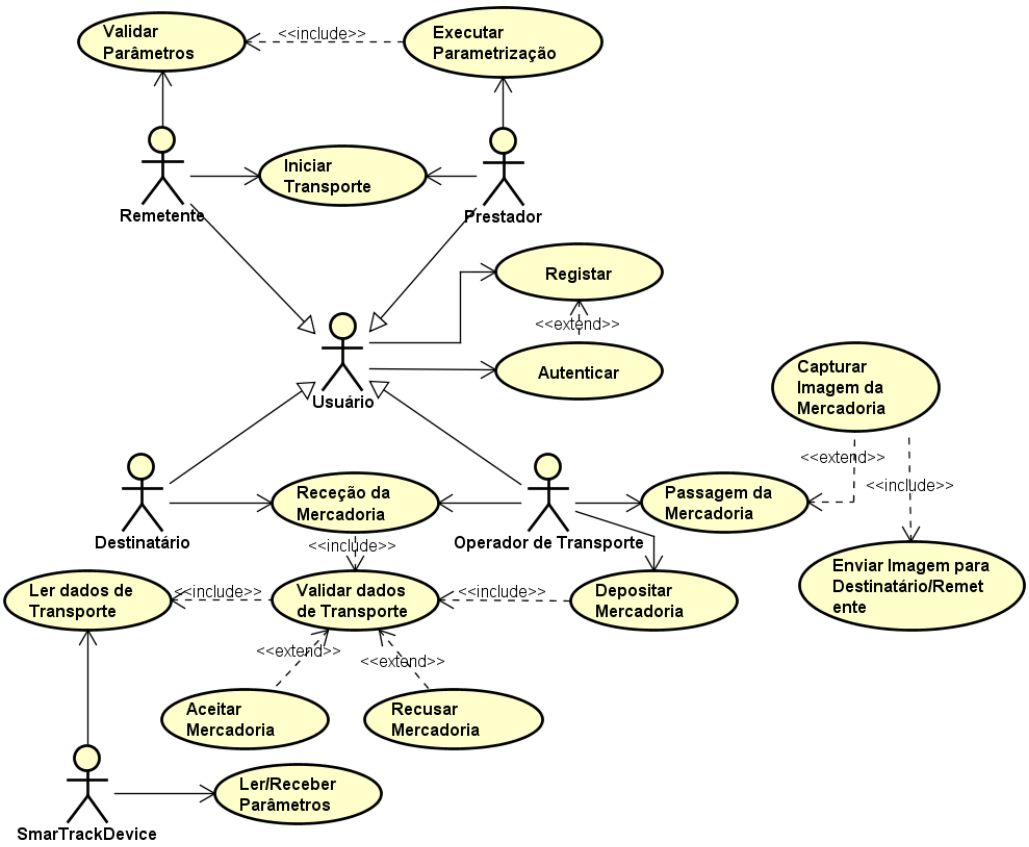


Figura 4.2: Diagrama de Caso de Uso - Atores e suas Ações

O **usuário** é um ator utilizado apenas para fins de simplificação, onde ele pode se registar ou se autenticar pelo aplicativo, utilizando seus dados pessoais, como endereço de e-mail, nome e palavra-passe. Tais características são atribuídas a todos os outros atores, tal como o **prestador**, que é o responsável por configurar as condições de transporte da mercadoria no SmarTrack Device. As condições definidas pelo prestador, devem ser lidas e validadas pelo **remetente** antes de confirmar o início do transporte. Esta validação é feita no sentido de garantir que as condições definidas são coerentes.

Após as condições serem verificadas e o serviço contratualizado por parte do remetente, o prestador entrega a mercadoria para um dado **operador de transporte**, que, por sua vez, sempre que alcançar um *milestone*, pode receber a mercadoria, ou deposita-la em um determinado local. Para ambas ações, o operador de transporte deve validar os dados de transporte recolhidos pelo SmarTrack Device, assim como os que estão no CS. Tal ação

é crucial para comprovar que, na passagem da mercadoria o operador atual entregou a mercadoria dentro das condições acordadas e também que o próximo operador recebe tal mercadoria dentro de tais condições.

Neste processo de passagem da mercadoria é possível, ainda, consoante as condições do contrato, que o operador de transporte obtenha imagens da mercadoria transportada para fins comprobatórios. Esta, e outras ações mencionadas anteriormente, serão explicadas em detalhes na sequência.

4.2 Descrição das Funcionalidades dos Atores

Para fins de segurança, foram implementados os conceitos de autenticação e autorização, sendo assim, para que um usuário possa autenticar-se no aplicativo, é necessário que este se tenha registado.

4.2.1 Registo e Autenticação

O primeiro passo para efetuar o registo, conforme mostra o Diagrama de Atividades representado pela Figura 4.3, é selecionar o perfil do usuário, que pode ser um destinatário, prestador, operador de transporte ou remetente. Em seguida, deve ser informado um endereço de e-mail que não tenha sido registado anteriormente. Após inserir o endereço de e-mail é necessário que o usuário informe seu nome e uma palavra-passe de força[1] média ou superior. Por fim, basta aguardar pelo recebimento da *One-Time Password* (OTP)[2] no endereço de e-mail registado, que será guardado localmente na memória de armazenamento local do próprio *smartphone*, nomeadamente em um local reservado pelo SO para armazenar dados do aplicativo.

Após o registo ser completado com sucesso, é dado início ao processo de autenticação, que pode ser visto no Diagrama de Atividades representado pela Figura 4.4, onde é possível

[1] A força é uma medida feita com base nos caracteres inseridos, considerando letras, números e caracteres especiais;

[2] Número de seis dígitos gerados aleatoriamente utilizado somente uma vez, o qual é necessário para comprovação de autenticidade do usuário;

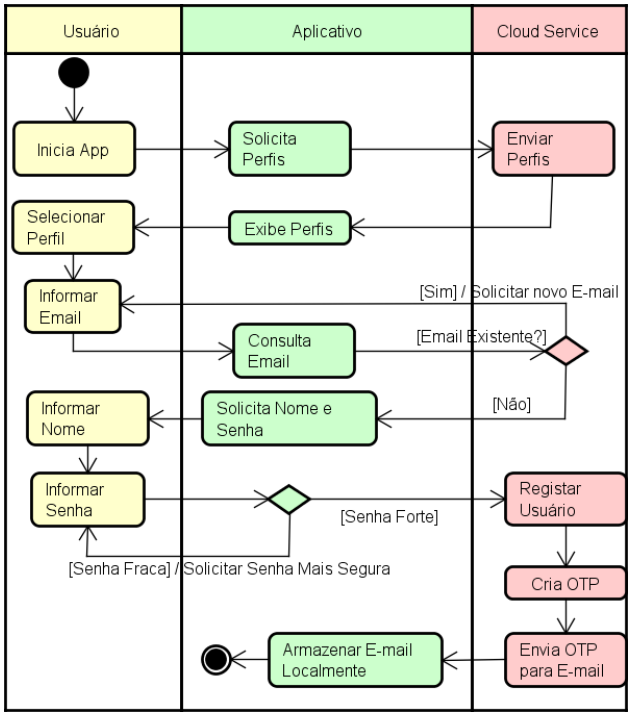


Figura 4.3: Diagrama de Atividades - Registo

perceber que tal processo é feito com base no endereço de e-mail do usuário. Este, que por sua vez, é o identificador único do **usuário**.

Em um primeiro momento, é verificado se há um endereço de e-mail guardado localmente no *smartphone* do usuário. Caso não haja, é necessário que o usuário informe seu endereço de e-mail, assim como sua palavra-passe. Se os dados inseridos forem iguais aos armazenados no CS, será enviada uma OTP para o endereço de e-mail informado. Entretanto, caso haja um endereço de e-mail guardado localmente no *smartphone*, será utilizado este e-mail para fins de autenticação, sem a necessidade em verificar o endereço de e-mail e palavra-passe no CS. Na sequência, é dado início à segunda etapa da autenticação, que, por sua vez, tem início quando o usuário insere a OTP recebida no endereço de e-mail registado.

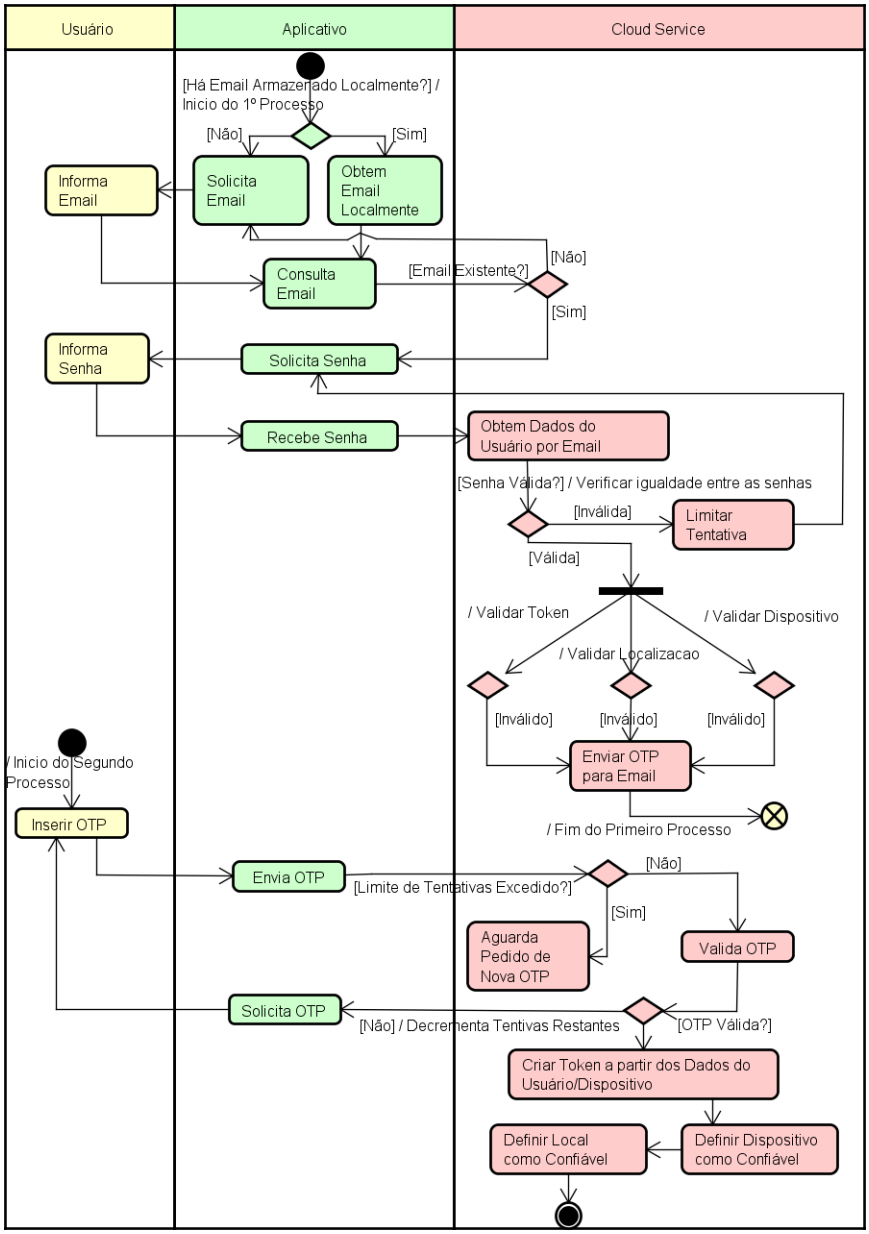


Figura 4.4: Diagrama de Atividade - Autenticação

A seguir a inserção, é feita a verificação da igualdade da OTP no CS. Quando confirmada a igualdade entre a OTP informada, e a OTP armazenada no CS, são realizados os procedimentos para criação de um *cs token*[3], inclusão do identificador único do

[3]Criado a partir dos dados de registo do usuário, e é utilizado para autenticação e autorização do usuário;

smartphone[4] na lista de dispositivos do usuário, e a inserção da localização atual[5] do *smartphone* na lista de locais confiáveis do usuário.

Após essa etapa inicial de autenticação, sempre que o usuário inicia o aplicativo, é feita a verificação da autenticidade do *cs token*, da localização atual do *smartphone*, assim como o identificador do *smartphone* que o usuário está utilizando.

A autenticidade do *cs token* é feita a partir de uma verificação de existência de um *cs token* válido no armazenamento local do *smartphone*, caso não exista um *cs token*, ou se este for inválido, o processo de autenticação irá falhar. Quanto a localização atual, sempre que o usuário tentar autenticar-se em um local onde não esteve nos três últimos meses, o processo de autenticação irá falhar. No tocante ao identificador do *smartphone*, este é utilizado para identificar tentativas de autenticação em dispositivos desconhecidos pelo usuário, caso isso ocorra, o processo de autenticação irá falhar.

Se o processo de autenticação falhar por qualquer um dos motivos mencionados anteriormente, faz-se necessário que o usuário repita todo o processo de autenticação.

Após o usuário ser devidamente autenticado, este será redirecionado para uma GUI consoante o perfil selecionado no processo de registo. Isto é, cada perfil de usuário contém GUIs pertinentes a tal perfil.

4.2.2 Contratualização e Parametrização

Para realizar a parametrização das condições de transporte da mercadoria, é necessário que o usuário seja um prestador, isto é, tenha definido seu perfil como tal. Enquanto que para verificar os parâmetros definidos pelo prestador, é necessário que o usuário seja um remetente. Na parametrização da mercadoria ocorrem algumas interações entre os diferentes atores do sistema, que podem ser observadas no Diagrama de Atividades representado pela Figura 4.5. Neste caso, o processo tem início com o prestador, que deve selecionar a mercadoria que será transportada no aplicativo. Após este momento, o operador, ainda pelo aplicativo, define e envia para o SmarTrack Device os limites máximos

[4]Gerado a partir de dados do *smartphone* e coletado por auxílio de uma API;
[5]Dados da cidade ou região, tais como nome, país;

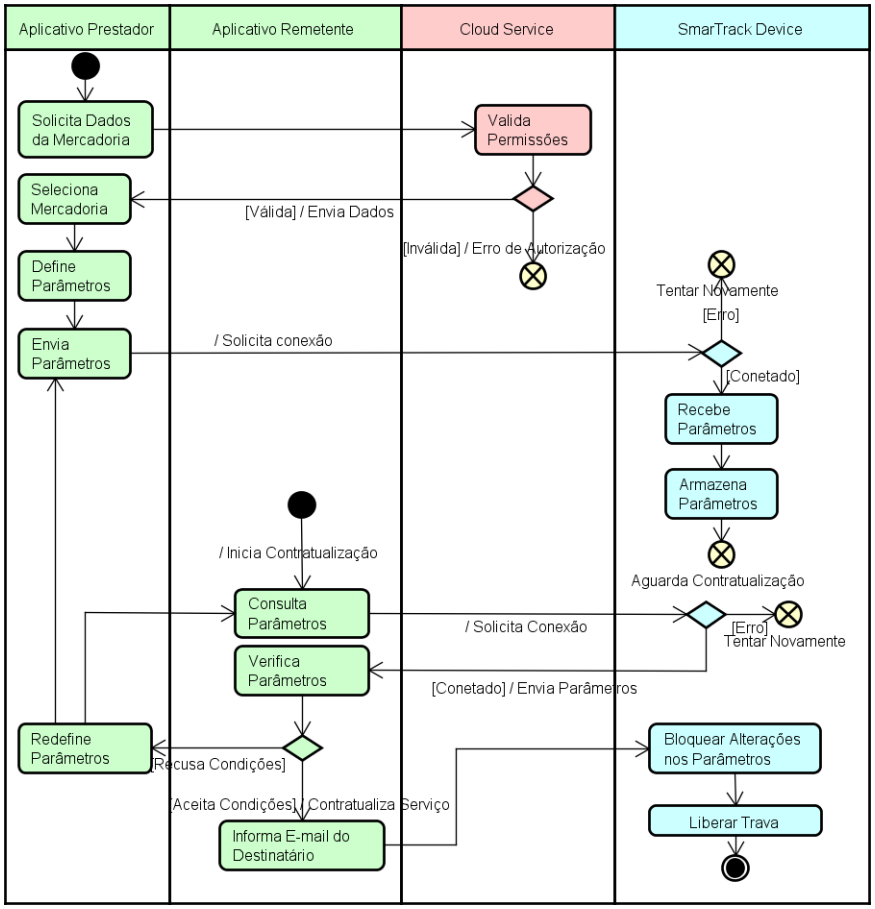


Figura 4.5: Diagrama de Atividades - Parametrização

e mínimos relacionados com as condições que devem ser atendidas durante o transporte, que podem estar vinculadas com a movimentação, humidade relativa e temperatura da mercadoria.

Após definir os limites, o prestador deve selecionar o modo de operação mais adequado para monitorizar tais condições. No que diz respeito aos modos de operação, foram definidos três modos, que são mecanismos relacionados com a maneira em que os dados são lidos. Tais modos têm por objetivo adequar às necessidades específicas do que está sendo transportado para aprimorar a eficiência energética do SmarTrack Device. Os três modos de operação podem ser vistos na Figura 4.6, que explica o funcionamento destes em forma de fluxograma. Além do diagrama, a seguir é também feita uma explicação mais abrangente dos modos de operação:

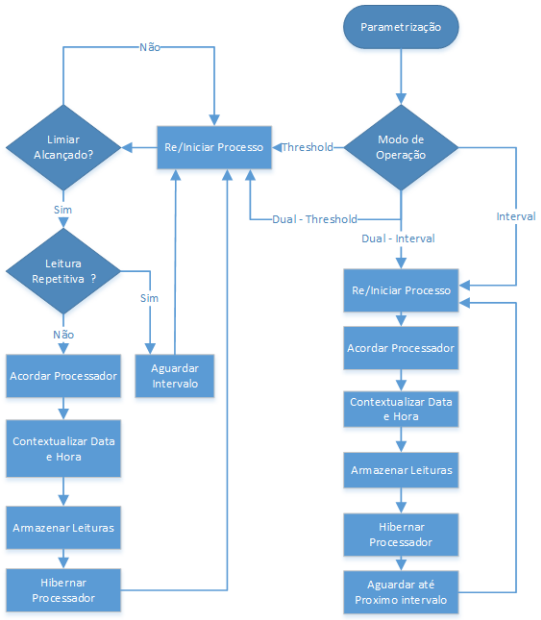


Figura 4.6: Fluxograma - Modos de Operação

Threshold Mode - neste modo, a coleta de dados dos sensores ocorre apenas quando um ou mais limiares são atingidos durante o transporte da mercadoria em questão, o que implica na ocorrência de uma anomalia no processo. Este modo pode ser exemplificado a partir de uma situação onde uma carga cuja temperatura máxima não pode exceder 10^oC, atinge um valor de 11^oC por uma breve falha no sistema de refrigeração, neste caso é detetada uma anomalia no transporte e o dispositivo se encarrega de efetuar o registo da mesma. Este modo é determinante para reduzir o consumo energético, isto porque o SmarTrack Device fica em modo de hibernação enquanto não houver dados anómalos para serem processados. Cabe ressaltar ainda que há um limite intervalar na coleta de dados anómalos, quando estes ocorrem. Este limite é fundamental para evitar o desperdício energético em situações onde há leituras repetitivas em curtos intervalos de tempo nas situações onde há problemas em todo o trajeto, ou até mesmo em parte dele;

Interval Mode é um modo onde a leitura dos dados dos sensores ocorre em um intervalo de tempo acordado previamente entre as entidades envolvidas no processo logístico.

Este modo pode não ser tão eficiente em termos energéticos, porque em determinadas configurações, pode ocorrer do dispositivo coletar dados quando não há necessidade;

Dual Mode pode ser utilizado em casos onde haja necessidade em coletar dados por intervalos de tempo e, também, quando há deteção de anomalias. A utilização deste modo ocorre apenas quando o Threshold Mode não é o suficiente, nomeadamente em casos onde não é possível controlar todos os limites estabelecidos por limitação dos sensores, havendo, assim, a necessidade de um modo auxiliar para coleta de dados.

Ainda nos modos de operação para leitura dos dados, sempre que o SmarTrack Device armazena os dados lidos dos sensores, é feita uma vinculação com a data e hora que representa o momento em que estes foram lidos. Estes dados temporais são obtidos através de um *Real Time Clock* (RTC).

Após o prestador definir os limites e o modo de operação, é iniciado o processo de confirmação dos parâmetros. Este, por sua vez, inicia com o remetente utilizando o aplicativo para verificar as condições definidas pelo prestador no SmarTrack Device. Se o remetente estiver de acordo com tais condições, ele deve confirmar a contratualização do serviço pelo aplicativo. Neste momento, o remetente informa o endereço de e-mail do destinatário, que será enviado em conjunto com o código identificador do transporte obtido a partir do CS. Na sequência, aquando da receção dos dados pelo SmarTrack Device, o *bit* de confirmação de contratualização para o SmarTrack Device é modificado.

Este *bit* de confirmação é utilizado para bloquear alterações nos parâmetros e fazer o fecho da trava de segurança, a qual permite ao remetente vincular o SmarTrack Device à sua mercadoria. Este vínculo só é desfeito no momento em que o operador de transporte entrega a mercadoria ao destinatário, o qual deve validar os dados de transporte antes de receber a mercadoria. Essa validação também deve ser feita pelos possíveis operadores de transporte. Processo este, que pode ser visto com detalhes na sequência.

4.2.3 Validação dos Dados de Transporte e Transação da Mercadoria

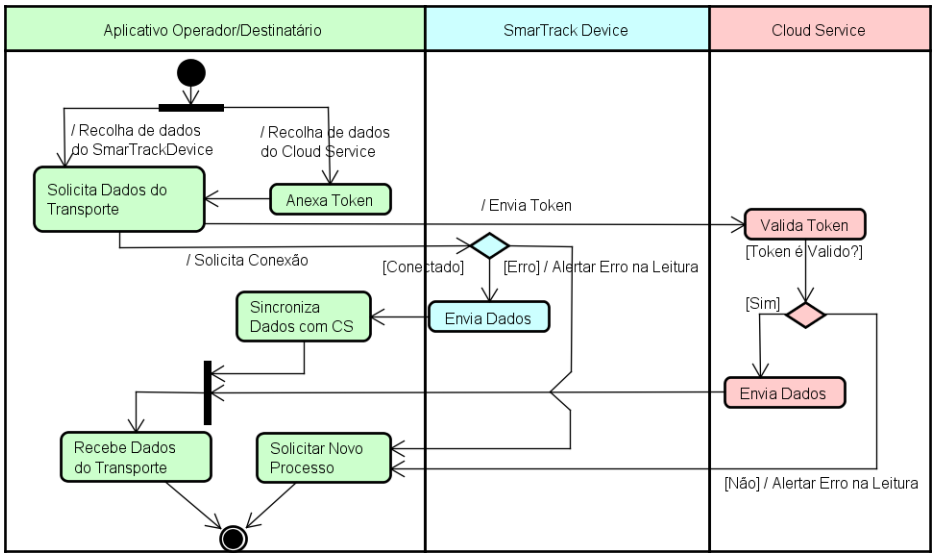


Figura 4.7: Diagrama de Atividades - Recolha de Dados

Para efetuar a validação dos dados de transporte, tanto o operador, quanto o destinatário, devem solicitar a leitura dos dados de transporte pelo aplicativo antes de receber a mercadoria. Este processo envolve um conjunto de interações entre aplicativos, SmarTrack Device e CS, conforme mostra o Diagrama de Atividades representado pela Figura 4.7.

Os dados a serem validados são descarregados do SmarTrack Device assim como eventuais dados armazenados no CS. A consulta em ambos os componentes faz-se necessária porque, sempre que um operador faz a leitura dos dados, é iniciado um procedimento de remoção dos dados lidos, objetivando liberar espaço de armazenamento no SmarTrack Device. Neste caso, pode acontecer que os dados não estejam sincronizados.

Há ainda outro inconveniente que pode ocorrer na remoção dos dados armazenados no SmarTrack Device. Se algum erro, seja técnico, ou até mesmo intencional acontecer, os dados do SmarTrack Device podem ser removidos sem que estes tenham sequer sido lidos, provocando a perda de informações de transporte, comprometendo assim, a fidedignidade dos dados.

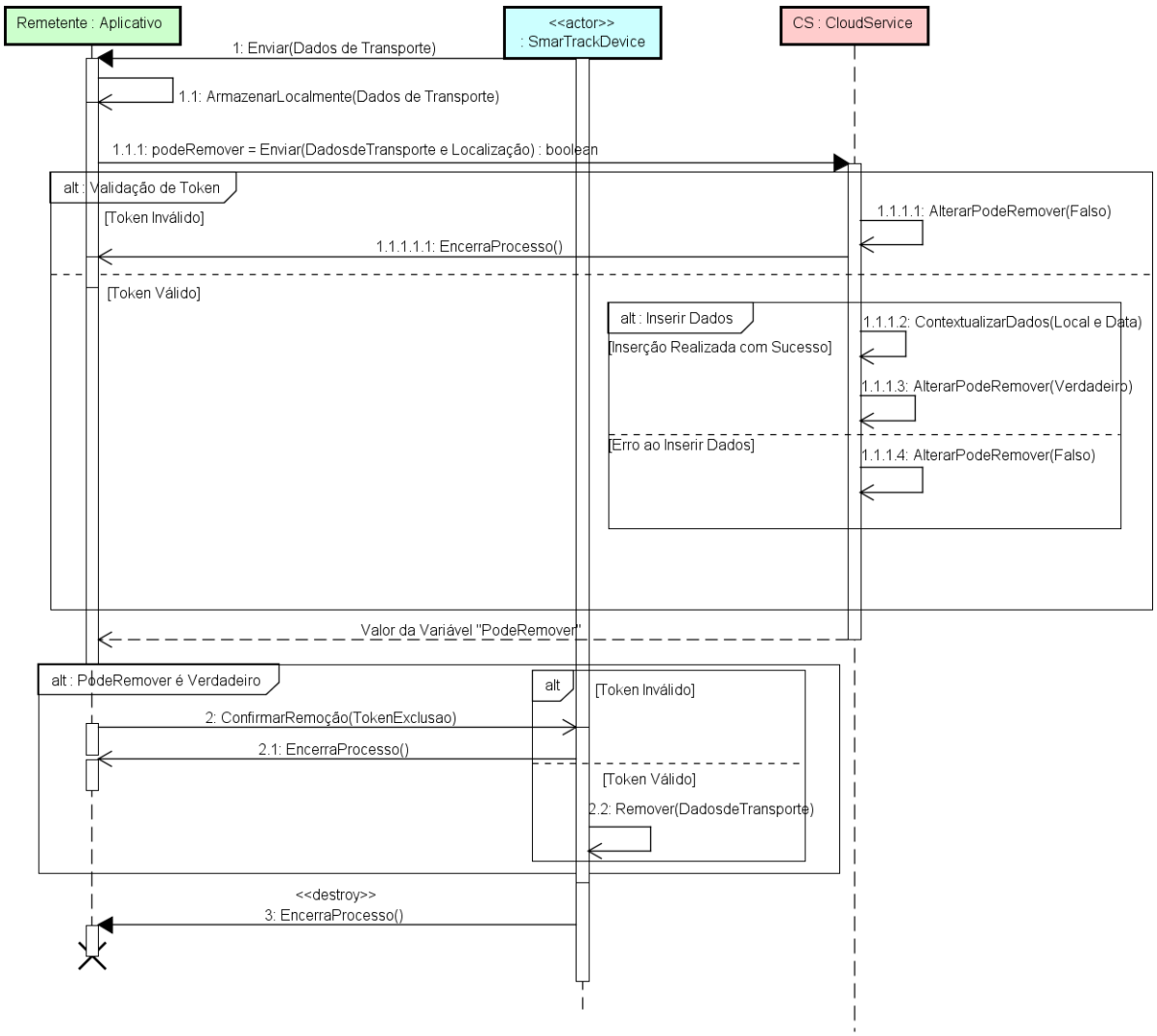


Figura 4.8: Diagrama de Sequência - Remoção segura dos dados armazenados no SmarTrack Device.

Para lidar com esta situação, os dados armazenados no SmarTrack Device, serão removidos se, e somente se, for recebida uma confirmação de exclusão de tais dados. Este processo é representado pela Figura 4.8 no formato de um Diagrama de Sequência.

Nesta diagrama, é possível notar que no momento em que o SmarTrack Device envia os dados para o aplicativo, o SmartTrack Device não realiza operações nos dados até ao momento em que o aplicativo recebe uma confirmação do CS que os dados podem ser removidos. Esta confirmação é enviada somente quando os dados enviados pelo aplicativo são armazenados no CS. O aplicativo envia também a localização atual do *smartphone*

para contextualizar os dados em termos geográficos. Além disso, a data e hora do CS é também contextualizada nos dados enviados pelo aplicativo.

Após a receção da confirmação, o aplicativo envia uma solicitação para remoção de dados do SmarTrack Device, juntamente com um *smt token*[6]. Este *smt token* é validado pelo SmarTrack Device e, caso seja válido, os dados do transporte serão removidos.

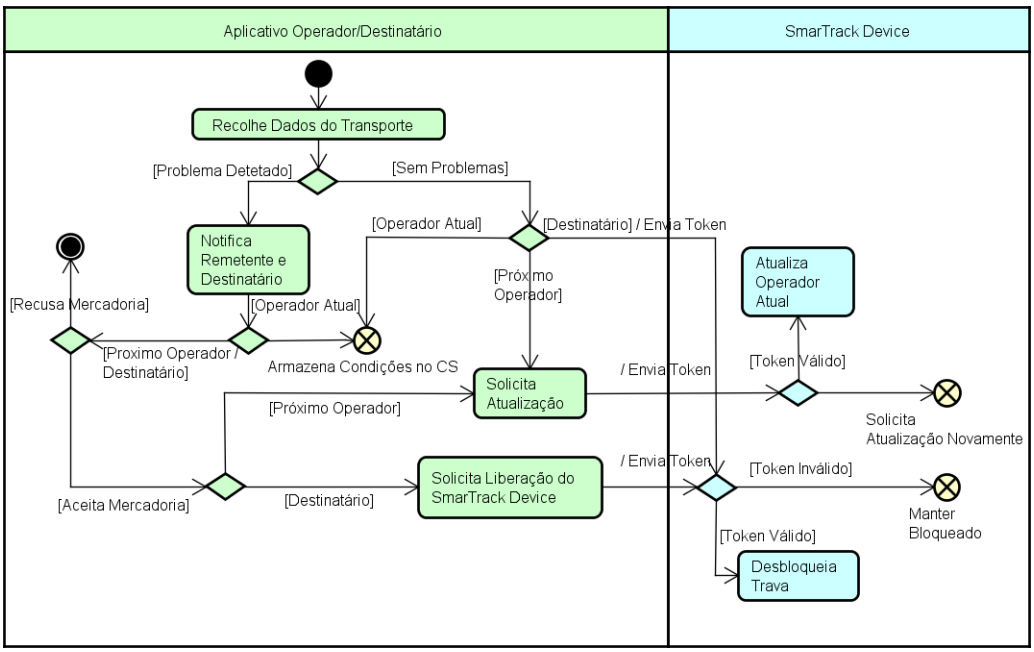


Figura 4.9: Diagrama de Atividades - Receção da Mercadoria

Após os dados serem descarregados e sincronizados de forma confiável, o aplicativo informa ao operador de transporte ou o destinatário, se as condições a serem cumpridas foram, ou não, violadas. Neste momento, cabe aos responsáveis por receber a mercadoria validar os dados recolhidos. Caso não o façam, há possibilidade em serem responsabilizados por eventuais falhas cometidas por outros operadores ou, ainda, no caso do destinatário, receber uma mercadoria danificada. O operador de transporte atual, deve também fazer essa verificação, aquando da passagem da mercadoria para comprovar que está entregando a mercadoria em condições.

Sendo assim, no processo de receção da mercadoria há três conjuntos de ações distintos,

[6]Gerado a partir de 11 dígitos, e é utilizado para autorização em operações de escrita no SmarTrack Device;

um para cada ator, conforme pode ser observado no Diagrama de Atividades da Figura 4.9.

Para o operador de transporte atual, a recolha de dados é feita para comprovar que este ator entregou a mercadoria em condições. Para isso, no momento em que o operador atual faz a recolha dos dados de transporte, o aplicativo faz a verificação se ocorreu violação das condições de transporte. Em caso positivo, o aplicativo notifica o remetente e o destinatário da violação, sincroniza os dados com o CS e o processo é finalizado. Se não forem detetados problemas, o aplicativo apenas sincroniza os dados de transporte, os quais informam a não ocorrência de problemas de transporte por parte do operador atual. Claro está que, em casos de falhas na sincronização dos dados com o CS, estes dados não serão perdidos, possibilitando que o próximo operador valide as condições da mercadoria antes de a receber.

Quando o próximo operador de transporte aceita receber a mercadoria, é feita a alteração do operador atual no SmarTrack Device. Para isso, o endereço de e-mail do operador que estava entregando a mercadoria é substituído pelos dados do operador que recebe a mercadoria.

Este processo de troca dos operadores pode ocorrer uma ou várias vezes no transporte de uma determinada mercadoria. Tais trocas ocorrem nos *milestones* que, conforme mencionado no Capítulo 2, são os pontos de recolha da mercadoria. Aquando da troca de operadores, podem ocorrer duas situações:

Situação 1 - Passagem por testemunho: o operador de transporte atual entrega a mercadoria para ser depositada em um armazém, isto é, não há troca imediata de mercadoria entre operadores. Neste caso, o responsável por receber a mercadoria não é necessariamente um operador de transporte, mas sim, é quem faz o testemunho das condições da carga, para que posteriormente, seja alterado o operador de transporte. No entanto, para fins de validação, quem recebe a mercadoria, deve seguir os mesmos procedimentos que um operador de transporte;

Situação 2 - Passagem direta: o operador de transporte atual, entrega a mercadoria

para um outro operador de transporte, esta passagem da mercadoria ocorre no mesmo instante da troca de operadores de transporte.

Na passagem da mercadoria, pode ser necessário que o operador de transporte atual faça a comprovação das condições da mercadoria com o uso de imagens. Por exemplo, no transporte de animais vivos, o proprietário deste animal pode solicitar evidências convincentes no tocante ao estado do animal.

4.2.4 Comprovação de Condições por Imagens

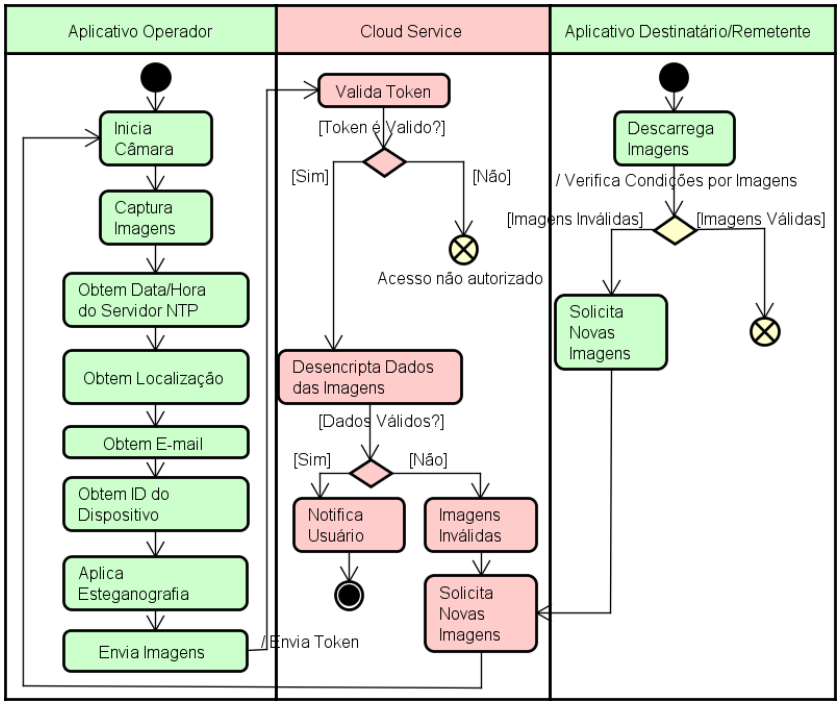


Figura 4.10: Diagrama de Atividades - Comprovação por Imagens.

Conforme mostra o Diagrama de Atividades representado pela Figura 4.10, para realizar este tipo de comprovação, o encarregado deve capturar, pelo aplicativo, imagens da mercadoria no momento em que for realizar a passagem da mercadoria. Para tal, o aplicativo não permite que seja feita a seleção de arquivos da galeria do *smartphone*, portanto, é permitido somente a obtenção de imagens a partir da câmara do *smartphone*. Fator este que minimiza ações fraudulentas neste processo.

Após a captura das imagens, o aplicativo obtém a data e hora atual[7], a cidade onde o operador está[8], o endereço de e-mail[9], nome do operador[10] e, por fim, o identificador único do dispositivo com o qual capturou as imagens. Estes dados são ocultados pelo aplicativo por meio de um processo de esteganografia, o qual é utilizado para fins de validação de autenticidade das imagens capturadas.

Depois de aplicar a esteganografia, a imagem é enviada para o CS, o qual desencripta e valida as informações contidas nas imagens. O processo de validação das imagens envolve a verificação da igualdade entre a cidade onde as imagens foram capturadas e a cidade onde a mercadoria está. Assim como a validação das informações do identificador do dispositivo, garantindo que a imagem foi capturada pelo operador de transporte responsável pela mercadoria. Por fim, os dados temporais para identificar se as imagens são coerentes ao momento em que foram capturadas, evitando o uso de imagens antigas. Caso alguma destas verificações de autenticidade falhar, o operador de transporte deve enviar novas imagens.

Em seguida, confirmada a autenticidade das imagens, o CS envia uma notificação para o remetente e o destinatário, para que estes possam visualizar as imagens. Sendo assim, através destas imagens, é possível que o remetente/destinatário verifique as condições da mercadoria, e pode, eventualmente, reportar os problemas para que sejam tomadas as medidas apropriadas, ou, solicitar novas imagens, caso estas não sejam suficientes, ou ainda, estiverem corrompidas.

Além da eventual validação por imagens, o destinatário também pode validar as condições da mercadoria antes de recebe-la, conforme já mencionado anteriormente.

4.2.5 Relação entre Operador de Transporte e Destinatário

Quando o destinatário recebe a mercadoria, este também pode verificar as condições da mercadoria utilizando o aplicativo. Essa operação tem uma lógica similar a que foi

[7]Obtida a partir de um serviço de consulta de data e hora online
[8]Obtida a partir da localização atual do *smartphone*;
[9]Armazenado no *smartphone*;
[10]Armazenado no *smartphone*;

explicada para que um operador de transporte receba a mercadoria (Figura 4.9). A diferença é que o destinatário, pelo aplicativo, ao confirmar se aceita, ou não a mercadoria, é enviado seu endereço de e-mail e um *smt token* para validar sua identidade para fins de confirmação da entrega.

Caso o destinatário aceite a mercadoria diante das condições apresentadas pelo aplicativo, é feita a sincronização dos dados do SmarTrack Device com o CS por intermédio do aplicativo e o processo é finalizado com sucesso. Para finalizar o processo, o aplicativo envia, juntamente com um *cs token*, uma confirmação de término do processo no CS, alterando o valor do *bit* de fechamento para 1. Após tal confirmação, o aplicativo envia para o SmarTrack Device, em conjunto com um *smt token*, uma solicitação de desbloqueio do dispositivo. Se o *smt token* for válido, a trava é desbloqueada, o SmarTrack Device pode ser desvinculado da mercadoria e todos os parâmetros definidos no SmarTrack Device são removidos.

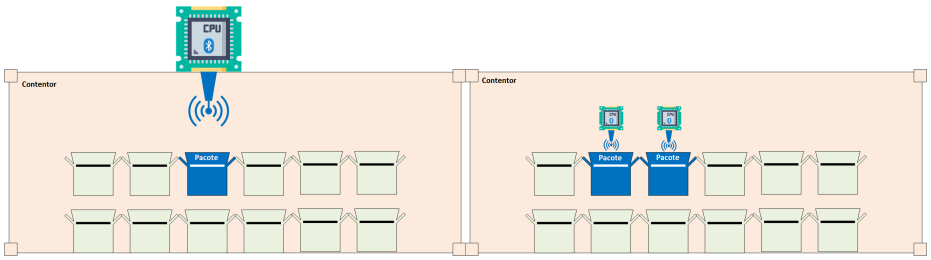
No que diz respeito à recusa da carga, esta pode ser realizada se, e somente se, houver problemas no transporte, ou ainda, se a mercadoria, ao ser entregue, não for a que o destinatário está aguardando.

Para confirmar se a mercadoria entregue é a mesma que o destinatário aguarda receber, é verificado se o endereço de e-mail enviado pelo destinatário no momento da receção é divergente do que foi definido pelo remetente no processo de parametrização, além de validar também se o código identificador do transporte definido nos parâmetros do SmarTrack Device, é equivalente ao que está armazenado no CS.

Logo, na situação onde há recusa da mercadoria, cabe aos envolvidos no processo entrarem em acordo, ficando a critério da solução apenas armazenar dados históricos, dados estes que podem ser utilizados como provas.

4.3 SmarTrack Device - Topologias

O SmarTrack Device pode ser utilizado em distintas topologias, atuando sempre como um dispositivo BLE *peripheral*, o qual é responsável por anunciar que está disponível para



(a) Exemplo 1 - externo ao contentor (b) Exemplo 2 - externo ao pacote

Figura 4.11: Representação gráfica da Topologia 1

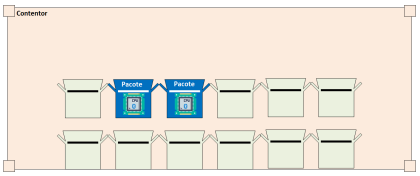


Figura 4.12: Representação gráfica da Topologia 2

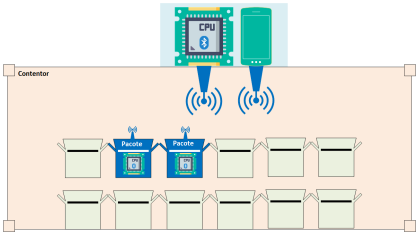


Figura 4.13: Representação gráfica da Topologia 3

receber conexões. Enquanto os dispositivos responsáveis por enviar pedidos de conexão ao SmarTrack Device, são dispositivos BLE do tipo *centrals*, como o *smartphone*. Para fins de exemplificação, algumas destas serão explicadas a seguir:

Topologia 1 (Figuras 4.11.a e 4.11.b) - nesta, o SmarTrack Device fica externamente a mercadoria, ficando acoplado ao contentor ou ao pacote;

Topologia 2 (Figura 4.12) - é uma topologia onde o SmarTrack Device é inserido internamente ao pacote da mercadoria, mesmo que esta esteja dentro de um contentor;

Topologia 3 (Figura 4.13) - é uma topologia que combina partes das topologias 1 e 2. Esta topologia contém um dispositivo externo responsável por coletar informações enviadas pelos vários SmarTrack Devices internos ao pacote da mercadoria.

Diante das topologias apresentadas foi necessário analisar os prós e contras entre cada uma destas, para que pudesse ser definida a topologia mais adequada à solução proposta nesta dissertação. Para a escolha da topologia, foram analisados fatores relacionados com a precisão das informações recolhidas, quantidade de dispositivos necessários, relevância dos dados coletados e consumo energético.

No que concerne a **topologia 1**, esta torna possível o acesso físico ao SmarTrack Device sem a necessidade em abrir o pacote ou o contentor. Por isso, em casos onde é necessário alguma interação física entre operadores de transporte e SmarTrack Device, esta topologia pode ser adequada. Outro fator relevante desta topologia, é que pode ser possível monitorizar uma ou mais cargas utilizando apenas um SmarTrack Device. Em contrapartida, algumas leituras podem não ser tão precisas, isto porque o dispositivo fica externo à mercadoria e ao sofrer variações, pode não ser possível as detetar.

Como exemplo, tem-se uma obra de arte sendo transportada em um contentor, e que sofre uma queda devido a uma variação de movimento no contentor, no entanto, tal variação pode não permitir realizar a prova de que tais fatos ocorreram. Outra consideração desta topologia, é ao monitorizar a temperatura e/ou humidade relativa. Mesmo que tenha um SmarTrack Device em cada pacote, pode não ser possível aferir certas medições de maneira coesa. Isto porque o que ocorre neste caso, são medições do ambiente, e não necessariamente do bem transportado.

Já na **topologia 2**, como o SmarTrack Device e a mercadoria ficam dentro do mesmo pacote, é possível realizar medições relacionadas com as condições da mercadoria de maneira mais precisa. Nesta topologia, as medições realizadas são em função da mercadoria, e não do ambiente, fator este que permite identificar quedas, variações de temperatura e humidade. Em contrapartida, tal topologia pode não ser tão escalável, haja vista que para cada mercadoria, é necessário ter individualmente um SmarTrack Device.

A **topologia 3** é mais abrangente e escalável pelo fato de combinar alguns dos aspetos positivos das duas anteriores. Nesta topologia é possível monitorizar individualmente cada mercadoria e manter um dispositivo externo que permita o acesso físico a este.

Este dispositivo é responsável por receber as informações das mercadorias coletadas

pelos SmarTrack Devices e enviar para um CS que, em linhas gerais, é um componente da solução para lidar com os dados relacionados com a solução. Tal dispositivo pode ser algo desenvolvido para estes fins ou até mesmo um *smartphone*, desde que este forneça mecanismos para se comunicar com os SmarTrack Devices.

O uso deste dispositivo externo permite também recolher dados mais precisos da localização, além de recolher dados em tempo real, tornando possível eventualmente, alertar eventuais problemas com a mercadoria no exato momento em que estes estão ocorrendo.

Portanto, após identificar os aspetos positivos e negativos de cada topologia, foi escolhida a mais adequada à solução proposta para esta dissertação.

4.3.1 Comparação e Escolha da Topologia Adequada

Primeiramente, foi realizada uma comparação entre as topologias 1 e 2. Nesta comparação foi possível notar que a topologia 2 pode recolher informações da mercadoria de forma mais precisa que a topologia 1. No tocante ao fato da topologia 1 permitir acesso físico ao dispositivo, foi possível notar que é possível utilizar recursos tecnológicos disponibilizados por algumas tecnologias de comunicação sem fio para suprir esta necessidade. Por fim, no quesito consumo energético, ambas topologias tem sua base parecida, mudando apenas o local onde o dispositivo será incluído e portanto, não têm diferença no modo de operação, e consequentemente no consumo energético.

Assim sendo, o uso da topologia 1 foi descartado pelo fato de existir uma outra topologia que recolhe dados da mercadoria de forma mais confiável e não ter ganhos no quesito de consumo energético. No entanto, não há limitações de implementação que impeçam o uso do SmarTrack Device nesta topologia.

Quando comparada com a topologia 3, a topologia 2 é menos abrangente no quesito de dados coletados. Isso porque na topologia 3 há um dispositivo complementar que pode recolher dados da localização da mercadoria de maneira mais precisa, assim como enviar dados em tempo real por meio de uma interação entre dispositivos, isto é, sem que haja qualquer ação dos usuários envolvidos.

Para os objetivos propostos nesta dissertação, os dados coletados pela topologia 3 podem não ser tão relevantes, haja vista que o propósito desta dissertação é monitorizar as condições da carga para apurar responsabilidades, e para isso, é suficiente identificar o trajeto e o responsável. Em relação a coleta de dados em tempo real, há situações em que este recurso é viável, nomeadamente em casos onde é desejável notificar aos envolvidos que a mercadoria está prestes a sofrer danos. Porém, este recurso pode implicar no aumento do consumo energético da solução, considerando que o SmarTrack Device terá que recolher dados em momentos onde não há necessidade evidente. Este recurso ainda pode ser substituído por parâmetros extras adicionados no SmarTrack Device, permitindo que o remetente configure limites aceitáveis que identificam que a carga sofreu alterações, mas que estas não implicam diretamente em danos.

Sendo assim, diante as topologias apresentadas, foi possível concluir que a topologia que mais se adequa ao SmarTrack.IO, é a topologia 2 por questões de precisão dos dados, consumo energético, além da relevância dos dados recolhidos.

4.4 Planeamento do Desenvolvimento dos Componentes

Após a definição de todas as interações que ocorrem na SmarTrack.IO, foi dado início ao processo de elaboração dos conceitos técnicos das tecnologias.

4.4.1 SmarTrack Device - Definição dos Serviços e Características

Como o SmarTrack Device é o componente que faz a recolha dos dados, optou-se por fazer o planeamento conceptual deste primeiro. Isso porque, a partir deste planeamento, foi possível identificar que dados são transmitidos, assim como suas respetivas estruturas. Para isso, foram definidos os serviços e as características que compõem o SmarTrack Device, os quais podem ser visualizados na Figura 4.14.

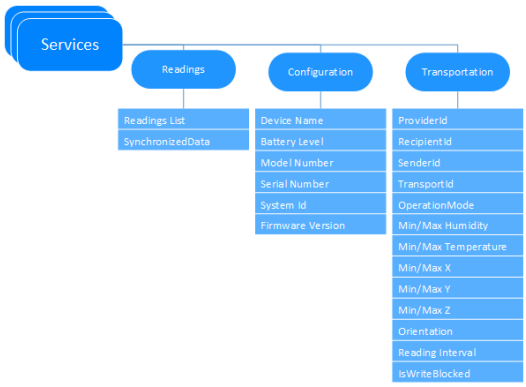


Figura 4.14: Visão geral dos Serviços e Características do SmarTrack Device

O *Configuration* é um serviço responsável por armazenar dados de configuração do dispositivo, que em linhas gerais, são utilizados para informar dados respeitantes ao dispositivo.

Quanto ao serviço *Readings*, este contém características responsáveis pela recolha dos dados dos sensores, que são armazenados durante o transporte da mercadoria.

Por fim, o serviço *Transportation* é responsável por armazenar os parâmetros definidos pelo processo de parametrização mencionados na subseção 4.2.2.

Os detalhes conceptuais de cada característica desenvolvida em tais serviços, podem ser observados na secção 9.1. Portanto, uma vez definido os serviços e as características necessárias para realizar as interações entre usuários e o SmarTrack Device, foi iniciado planeamento do CS.

4.4.2 *Cloud Service* - Armazenamento e Operações nos Dados

Após a identificação dos dados gerados pelo SmarTrack Device, foi necessário elaborar a estrutura do BD para armazenar os dados de transporte recolhidos pelo SmarTrack Device, assim como os dados dos usuários.

Um BD tem mecanismos para armazenamento de arquivos, no entanto, como há momentos em que muitos dados são recolhidos, optou-se pela utilização de um serviço dedicado para armazenamento de arquivos, em específico, as imagens obtidas para comprovação das condições da mercadoria.

Posteriormente à definição das estruturas para armazenamento de dados e arquivos, foi necessário disponibilizar um serviço para que os usuários pudessem realizar operações de leitura e escrita dos dados. Para isso, optou-se pela implementação de uma API utilizando o estilo arquitetónico REST.

4.4.3 Prototipagem dos Aplicativos para Android e iOS

Após a definição da estrutura para consumir os serviços disponibilizados pelo SmarTrack Device e, também, ter acesso a uma API REST, foi iniciado o processo de prototipagem dos aplicativos para Android e iOS, os quais têm como principal função, simplificar a interação entre os usuários e os componentes da solução.

Para a prototipagem, foram utilizados alguns *mockups*[11], buscando evidenciar as funcionalidades dos aplicativos. Alguns dos principais *mockups* podem ser visualizados nas Figuras, 4.15, 4.16 e 4.17.

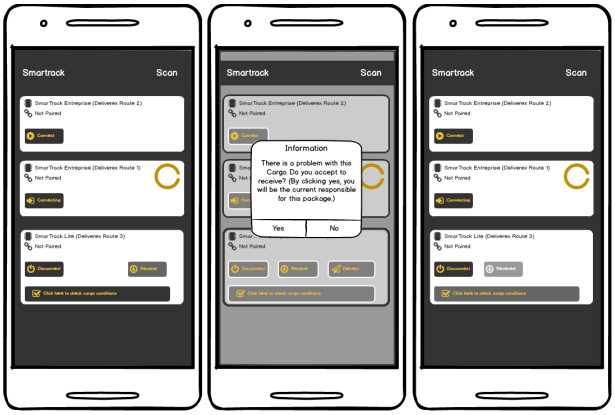


Figura 4.15: Mockup - Aceitação da Mercadoria feita pelo Operador.

[11]São, em linhas gerais, um protótipo da aplicação muito próximo do produto final a nível de *design*

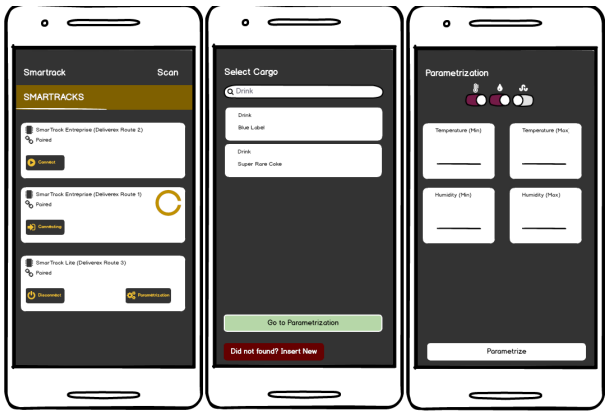


Figura 4.16: Mockup - Parametrização feita pelo Prestador

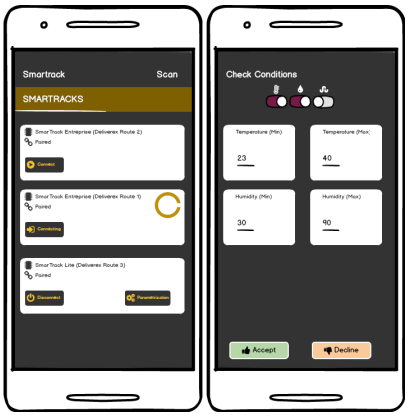


Figura 4.17: Mockup - Verificação dos parâmetros feita pelo Remetente.

Capítulo 5

Materiais

5.1 SmarTrack Device

Diante da necessidade em utilizar componentes de *hardware*, foi realizado um estudo comparativo entre alguns componentes disponíveis no mercado para, posteriormente, selecionar os componentes mais adequados para o desenvolvimento do SmarTrack Device.

5.1.1 Estudo comparativo dos Kits de Desenvolvimento

Neste trabalho, as características mais relevantes para a escolha do módulo foram o consumo energético, capacidade de armazenamento e processamento. Para tanto, foram analisados oito *Development Kits* (DKs) existentes no mercado que atendem aos requisitos de hardware, tais módulos podem ser vistos na Tabela 5.1.

Conforme pode ser observado na Tabela 5.1, há diversos módulos com características similares, alguns com maior destaque em um aspeto de desempenho, enquanto outros se destacaram com consumo energético relativamente baixo.

Em relação ao processamento, o maior destaque foi o módulo ESP32 da Espressif, o qual tem uma velocidade de relógio superior até aproximadamente 4 vezes mais rápida que o nRF52832 da Nordic Semiconductor, que, no entanto, nesta análise foi o segundo mais rápido.

	Processador	Flash	Ram	Consumo Energético
Nordic Semiconductor nRF 51822[a]	Cortex-M0 16 MHz	128 KB/ 256 kB	16 kB/ 32 kB	8.0 mA TX / 9.7 mA RX
Nordic Semiconductor nRF 52832[b]	Cortex-M4F 64 MHz	512 kB	64 kB	5.3 mA TX / 5.4 mA RX
Cypress PSoC 4 BLE[c]	Cortex-M0 48 MHz	256 kB	32 kB	18.7 mA
Texas Instruments CC2650[d]	Cortex-M3 Até 48 MHz	128 kB	20 kB	5.9 mA
Dialog Semiconductor 14580[e]	Cortex-M0 16 MHz	*32 kB	50 kB	4.9 mA
Qualcomm CSR101x[f]	Proprietário 16 MHz	128 kB	64 kB	∼20 mA
Atmel BTLC1000[g]	Cortex-M0 26 MHz	128 kB	128 kB	3.0 mA TX / 4.0 mA RX
Espressif ESP32 [h]	Tensilica LX6 Dual Core 80-240 MHz	16 MB	520 kB	100 mA

*Memória *One Time Programmable*;

[a]https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF51822
[b]https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832
[c]http://www.cypress.com/products/psoc-4-ble-bluetooth-smart
[d]http://www.ti.com/product/CC2650
[e]https://www.dialog-semiconductor.com/sites/default/files/smartbond_da1458x_family_product_brief_0.pdf
[f]https://www.qualcomm.com/media/documents/files/csr101x-product-brief.pdf
[g]http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel42493ATBTLC1000WLCSP-SOC_Datasheet.pdf
[h]https://www.espressif.com/en/products/hardware/esp32/overview

Tabela 5.1: Comparação entre alguns módulos que operam com *Bluetooth Low Energy*.

Ao analisar a memória Flash, o módulo ESP32 também foi superior aos outros, atingindo uma capacidade de armazenamento não volátil de aproximadamente trinta e duas vezes maior quando comparada ao segundo colocado, que neste caso foi novamente o módulo nRF52832.

Outro aspeto analisado foi a capacidade de armazenamento da memória *Random Access Memory* (RAM), e o destaque foi novamente para o ESP32, cuja diferença é de aproximadamente dezasseis vezes maior que o segundo colocado, o qual novamente foi o nRF52832.

Por fim, a última característica comparada foi o consumo energético dos módulos ao realizar operações de transmissão e receção (TX/RX). Neste caso, o destaque foi para o módulo BTLC1000 da Atmel, que consome 3.0 mA ao transmitir e 4.0 mA ao receber dados. Quando comparados estes valores com o segundo colocado, que neste caso foi o 14580 da Dialog Semiconductor, o módulo BTLC1000 chega a ser aproximadamente 63.3% mais económico ao transmitir dados e 22.5% ao receber dados.

5.1.2 A escolha do Kit de Desenvolvimento

Após realizar essa análise, optou-se por utilizar o SoC nRF 52832 da Nordic por ser o segundo melhor em termos de desempenho e o terceiro melhor em baixo consumo energético. Além disso, tal módulo permite operar em uma rede Mesh, que, em linhas gerais, permite que uma grande quantidade de dispositivos BLE sejam conectados e se comuniquem ao mesmo tempo utilizando alguns dispositivos para retransmitir a mensagem recebida para os outros. Portanto, este recurso permite a escalabilidade do projeto em trabalhos futuros que eventualmente possam necessitar a utilização de tal abordagem.

Importante ressaltar que durante o desenvolvimento do presente trabalho, novas versões de alguns dos módulos apresentados estavam em fase de produção e portanto não foram incluídos na Tabela 5.1.

Ao confirmar a seleção do módulo mais adequado ao projeto, foram adquiridos o nRF52[1] (Figura 9.4) e o Thingy52 (Figuras 9.5 e 9.3)[2] - ambos com o SoC nRF52832, os quais facilitaram a prototipagem deste projeto. A utilização de um módulo já disponível para IoT, que é caso do Thingy52, foi importante devido aos recursos já implementados, testados e disponibilizados pela Nordic Semiconductor, atendendo aos requisitos necessários e facilitando na escolha dos sensores utilizados para o projeto.

Após a aquisição dos módulos e a definição dos sensores utilizados no projeto, foi feita uma estruturação concetual do *hardware* do sistema desenvolvido em forma de um diagrama de blocos (Figura 5.1) que por sua vez representa as interações existentes entre os componentes de *hardware* e *software*, tais como aplicativos, serviços na nuvem, sensores e controladores. Este diagrama será expandido conforme o contexto do documento.

5.1.3 Sensores utilizados

Como o objetivo central do SmarTrack Device é a monitorização de alguns parâmetros, como exemplo, Humidade Relativa - do inglês, *Relative Humidity* (RH), temperaturas máximas e/ou mínimas, movimentação - eventualmente pressão atmosférica, qualidade

[1] https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52-DK
[2] https://www.nordicsemi.com/eng/Products/Nordic-Thingy-52

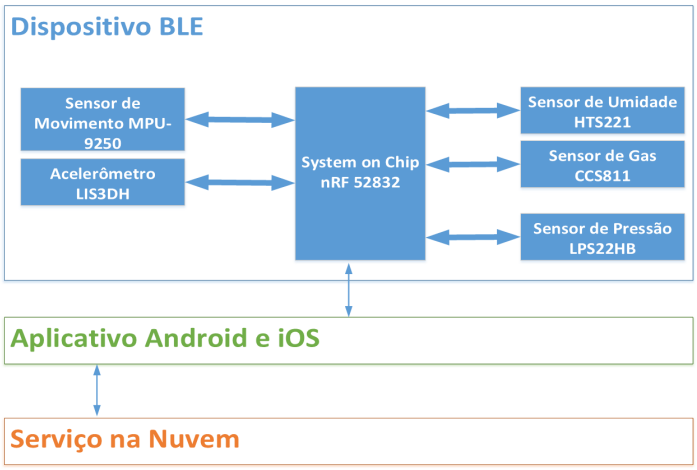


Figura 5.1: Arquictetura do *hardware* do sistema

do ar e luminosidade, foi necessário o uso de sensores que realizassem a medição de tais atributos. Alguns dos principais sensores para recolha de dados de temperatura, humidade e deslocamento são abordados com mais detalhes a seguir.

Sensor Acelerômetro LIS3DH : Diante da necessidade em monitorizar movimentos bruscos, nomeadamente os que caracterizam quedas, foi utilizado o LIS3DH, que por sua vez é um acelerômetro linear de três eixos de baixíssimo consumo energético - abaixo de 2 μA operando em modo de baixo consumo - e alta performance pertencente à família *nano*. O LIS3DH ainda mede acelerações em um intervalo de $1Hz$ a $5,3kHz$, além de ter escalas de $\pm 2g \pm 4g/ \pm 8g/ \pm 16g$ selecionáveis pelo programador de forma dinâmica. Este sensor pode operar em temperaturas dentro do intervalo de $-40^{o}C$ a $+85^{o}C$ [45].

O sensor ainda permite o uso de dois eventos inerciais independentes, que são o *wake-up* e o *free-fall*, os quais detetam respetivamente quando um dispositivo deve acordar por intermédio de movimentos e/ou quando está em queda livre. Ainda é possível identificar a posição em que o dispositivo se encontra [45].

Um outro recurso interessante deste sensor é o uso de um *buffer First In First out* (FIFO) integrado que permite armazenar dados utilizados para limitar a intervenção do processador [45].

No tocante à monitorização dos movimentos, neste projeto, os principais requisitos a serem cumpridos pelo sensor foram: ter um mecanismo de configuração para os limiares evitando o uso do processador quando não houver necessidade, gama de temperaturas de funcionamento, e baixo consumo energético;

Sensor de Humidade e Temperatura HTS221 : Quanto a coleta dos dados de temperatura e RH do ar, utilizou-se o sensor HTS221, que é um sensor digital ultra compacto para medir a temperatura e RH. Para o funcionamento adequado, as temperaturas não podem ser inferiores a -40^oC e superiores a $+120^oC$. No tocante à RH, este sensor fornece informações em um intervalo de 0 a 100% com uma margem de erro de $\pm 5\%$ podendo ser reduzida para $\pm 3.5\%$ num intervalo de 20 a 80% e com um tempo de resposta de 10s. Quanto à temperatura, é possível obter medidas num intervalo 0 a 60^oC com uma margem de erro de $\pm 1^o$, sendo o erro reduzido pela metade nas medidas entre $15^o e 40^oC$ e com um tempo de resposta de 15 s. Para outros intervalos de temperatura, não há uma precisão garantida pelo fabricante, sendo no entanto possível a leitura de valores. Em relação ao consumo energético, este sensor consome $2\mu A$ em funcionamento [46].

Diante da vasta gama de sensores existentes no mercado, foram optados por utilizar os sensores que já estavam sendo utilizados no Thingy52, os quais já estão devidamente implementados, calibrados e testados. Além disso, os sensores do Thingy52 permitem o desenvolvimento do protótipo do SmarTrack Device, cumprindo os requisitos de baixo consumo energético propostos pelo presente trabalho.

5.1.4 Tecnologias para Desenvolvimento do Código-Fonte

Finalizado os processos arquiteturais do *hardware* do sistema, foi identificada a necessidade em analisar o conjunto de ferramentas a serem utilizadas para dar início à implementação do SmarTrack Device. A Figura 5.2, ilustra em forma de diagrama de blocos, as tecnologias que foram utilizadas para o desenvolvimento do código-fonte do SmarTrack Device.

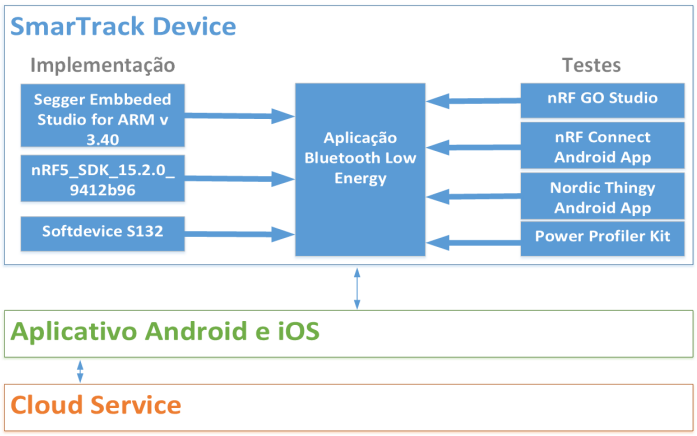


Figura 5.2: Tecnologias utilizadas na implementação.

Como Ambiente de Desenvolvimento Integrado - acrónimo do inglês, *Integrated Development Environment* - (IDE), foi utilizado Segger Embbeded Studio for ARM 3.40[3] por fornecer gratuitamente, um conjunto completo de funções que atenderam as necessidades deste projeto. Como SDK, foi utilizado a versão nRF5_SDK_15.2.0_9412b96[4] por ser a versão mais recente disponibilizada pela Nordic Semiconductor. Quanto à implementação da pilha de protocolos do BLE, foi optado por utilizar a Softdevice S132, por ser a mais recente disponibilizada e por permitir que o SmarTrack Device seja implementado, em um trabalho futuro, tanto como sendo um dispositivo *peripheral*, quanto um dispositivo *central*. Quanto ao uso do Segger Embbeded Studio, foi necessário realizar um ajuste nos parâmetros de configuração do projeto para que o SmarTrack Device pudesse funcionar adequadamente. Este ajuste encontra-se no Anexo 8.

Para realizar os testes de implementação do SmarTrack Device, foram utilizados, o *software* nRFGO Studio[5] e os aplicativos nRF Connect[6] e Nordic Thingy[7] para Android.

Também foi utilizado o Power Profiler Kit da Nordic[8] como ferramenta para realizar medições de consumo energético, tal como eventuais otimizações. Este dispositivo pode

[3]https://www.segger.com/products/development-tools/embedded-studio/
[4]https://www.nordicsemi.com/eng/nordic/Products/nRF52832/nRF5-SDK-zip/59012
[5]https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832
[6]https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=pt_PT
[7]https://play.google.com/store/apps/details?id=no.nordicsemi.android.nrfthingy
[8]https://www.nordicsemi.com/eng/Products/Power-Profiler-Kit

operar em simultâneo com o osciloscópio e ainda permite medições de consumo de corrente desde μA até dezenas de mA, além de realizar deteção de picos. Para realizar as medições basta conectar a entrada do acionador externo a um pino de E/S no dispositivo que está sendo testado. Conforme [47], este pino deve ser configurado para gerar um pulso de sincronização.

Após a escolha das tecnologias a serem utilizadas para implementação e testes do SmarTrack Device, foi dado início ao processo seletivo das ferramentas de desenvolvimento do CS, o qual é abordado em seguida.

5.2 *Cloud Service*

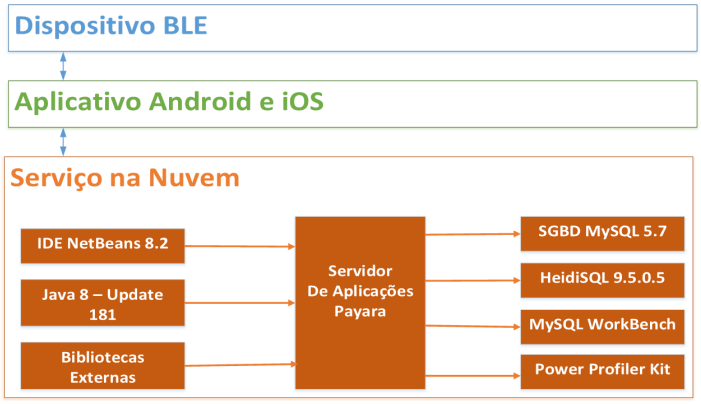


Figura 5.3: Tecnologias utilizadas na implementação do *Cloud Service*

Para o desenvolvimento do CS, há inúmeras tecnologias que podem ser utilizadas na implementação dos serviços que os compõe. A Figura 5.3, em formato de diagrama de blocos mostra as tecnologias utilizadas na implementação do CS, tecnologias estas, que são abordadas na sequência.

Optou-se por estruturar primeiro, a API REST que, em linhas gerais, fornece acesso aos dados. Assim sendo, o primeiro componente da API a ser estruturado, foi o BD. Para isso, foram utilizadas as ferramentas MySQL Workbench 6.3 CE[9], e a HeidiSQL

[9]https://dev.mysql.com/doc/workbench/en/

9.5.0[10]. A MySQL Workbench permite tanto o desenvolvimento de Diagramas Entidade Relacionamento, quanto a escrita de código SQL, assim como, gerar Diagramas Entidade Relacionamento a partir o código escrito em SQL e vice-versa. Quanto a HeidiSQL, foi optada pelo uso desta como uma ferramenta complementar, principalmente pela facilidade em testar códigos SQL.

No tocante ao Sistema Gerenciador de Banco de Dados (SGBD), foi optado por utilizar o MySQL 5.7, por questões de compatibilidade com o servidor de aplicações Payara 5[11].

Ainda nas tecnologias relacionadas com o BD, optou-se pelo uso da API JPA (Java *Persistence* API), por fornecer, principalmente, um conjunto de funcionalidades que permitem realizar o mapeamento de objetos relacionais, facilitando assim a implementação da API Rest, mitigando ataques de SQL *Injection* e, consequentemente, o desenvolvimento do código-fonte.

Quanto ao desenvolvimento do código-fonte, utilizou-se o IDE NetBeans 8.2[12], em conjunto com a LP Java 8 Update 181 por ser a versão mais recente, e estável, utilizada no momento do desenvolvimento do presente trabalho de dissertação.

Por fim, foram utilizadas algumas bibliotecas externas para auxiliar na implementação de algoritmos de segurança, tais como encriptação, criação e validação de *tokens* e chaves pública e privada.

5.3 Aplicativos para Android e iOS

Relativamente as tecnologias utilizadas para implementação no desenvolvimento dos aplicativos, a Figura 5.4, mostra, em formato de diagrama de blocos, uma visão geral das principais tecnologias utilizadas no desenvolvimento dos aplicativos.

Conforme mencionado no Capítulo 3, há diversas tecnologias que podem ser utilizadas para o desenvolvimento de aplicativos para *smartphones*. Para o presente trabalho,

[10]https://www.heidisql.com/
[11]https://www.payara.fish/software/payara-server/
[12]https://netbeans.org/community/releases/82/

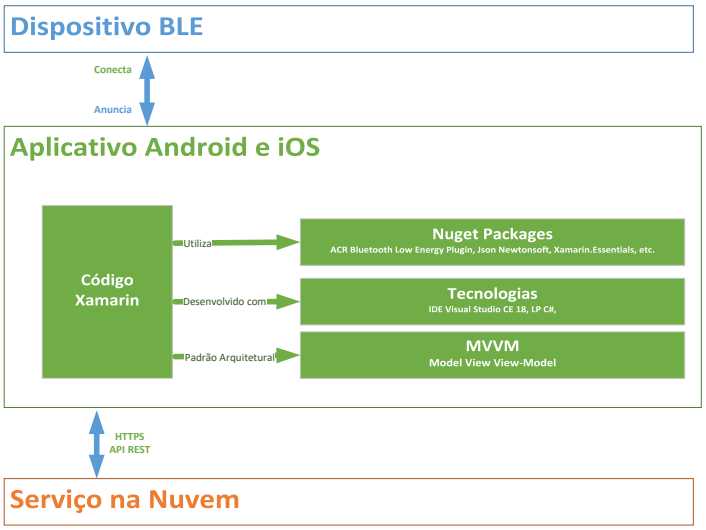


Figura 5.4: Arquitectura do aplicativo e seus principais componentes

utilizou-se o Xamarin[13], para o desenvolvimento dos aplicativos, isso porque o Xamarin, é uma tecnologia de desenvolvimento gratuita que fornece acesso à APIs nativas de cada plataforma de desenvolvimento, seja esta, Android, iOS ou Windows. Além disso, é possível encontrar uma variedade de *packages* que facilitam o desenvolvimento de aplicações utilizando um conjunto de métodos já implementados e testados.

Como LP, optou-se por utilizar o C# para desenvolver a lógica de negócios. Para desenvolver as GUIs do aplicativo, foi utilizada, a linguagem de marcação XAML. Assim como o IDE Visual Studio 2017 Community Edition, o uso destas duas linguagens foi optado por ser oficial no desenvolvimento de aplicações utilizando o Xamarin.

Para o desenvolvimento do aplicativo, foi optado por utilizar os conceitos do padrão de arquitetura de *software* Model-View-ViewModel (MVVM)[14] que, de acordo com [48], tem como propósito tornar possível a divisão das responsabilidades de cada classe separando as GUIs de suas respetivas lógicas de negócio, facilitando a manutenção e reutilização de código. Fatores estes que implicaram na redução da necessidade em alterar códigos escritos, tornaram o código mais legível e compreensível.

[13]https://visualstudio.microsoft.com/pt-br/xamarin/?rr=https%3A%2F%2Fwww.google.pt%2F
[14]https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm

Em relação a comunicação entre *smartphone* e SmarTrack Device, foi optado pelo uso do Bluetooth, atuando no modo de operação *Low Energy*. Já para consumir os serviços disponíveis no CS, foi necessário utilizar o Wi-Fi e/ou redes móveis.

Independentemente de estar utilizando uma tecnologia de desenvolvimento *cross-platform*, é necessário definir alguns requisitos da aplicação específicos para cada plataforma. Um destes requisitos é a versão mínima exigida para que a aplicação funcione adequadamente em dispositivos Android e iOS. Como o presente trabalho necessita utilizar a API do BLE, optou-se por utilizar a versão Android 6.0, que por sua vez é uma versão superior ao Android 4.3, a qual implementou recursos do BLE, no entanto, na versão 6.0, foram feitas melhorias no tocante ao BLE, além ainda de permitir o uso de *fingerprints*, que pode ser utilizado futuramente como recurso para otimização da segurança da aplicação.

No que diz respeito a versão do iOS, alguns dispositivos não receberam atualizações da versão 10 e, como as últimas versões não apresentaram mudanças significativas no tocante ao Bluetooth, foi optado por atingir um número maior de dispositivos com o uso da versão 10 do iOS.

Finalizada as etapas de análise e seleção das tecnologias utilizadas para o desenvolvimento do presente trabalho, deu-se inicio à implementação dos componentes. Este processo é descrito no Capítulo 6

Capítulo 6

Implementação e Testes

6.1 SmarTrack Device

Após identificar os serviços e suas respetivas características, definiu-se a estrutura de cada serviço, tais como permissões, e tipificação das características, conforme é mostrado nas Tabelas 6.1, 6.2, e 6.3, onde as permissões são representadas pelas letras L (leitura) e E (escrita) e os tamanhos são representados em bytes.

Configuration Service			
Nome	Permissão	Tamanho	Exemplo
Device Name	L/E	16	SmarTrack
Model Number	L	8	SAMPLE
Serial Number	L	4	A123
System ID	L	16	ABC0123XYZ
Firmware Version	L	3	1.2.3

Tabela 6.1: Detalhes das Características do *Configuration Service*

Readings Service			
Nome	Permissão	Tamanho	Exemplo
Readings List	L	20	[10.30,55,-2,721500]
Synchronized Data	L/E	1	0 - Não Sincronizado

Tabela 6.2: Detalhes das Características do *Readings Service*

Transportation Service			
Nome	Permissão	Tamanho	Exemplo
Provider ID	L/E	20	AXPNPG4POVIYKURE
Recipient ID	L/E	20	12NPG4POVIYKHGFX
Sender ID	L/E	20	42NPG4POVIYKHKOI
Transport ID	L/E	2	20
Operation Mode	L/E	1	0 - *Threshold Mode*
Temperature Thresholds	L/E	2 Cada	[-50;50]
Humidity Thresholds	L/E	1 Cada	[0;99]
X, Y, Z Thresholds	L/E	8 Cada	[-20;20]
Orientation	L/E	1	1 - Retrato
Reading Interval	L/E	4	18000 milissegundos
IsWriteBlocked	L	1	0 - Bloqueado

Tabela 6.3: Detalhes das Características do *Transportation Service*

Uma das principais características do SmarTrack Device é a *Readings List*, a qual é responsável por lidar com dados recolhidos pelos sensores e fazer a contextualização temporal destes dados. A estrutura desta característica é mostrada na Figura 6.1. Os detalhes das outras características são mostrados na secção 9.1, do Apêndice 9.

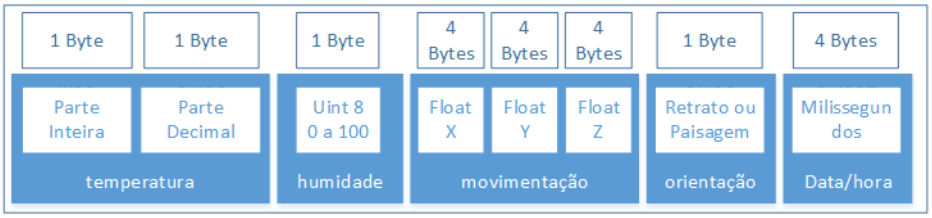


Figura 6.1: Estrutura da característica *Readings List*

Na sequência, iniciou-se o desenvolvimento do código responsável por transmitir e receber dados pelo protocolo BLE. Para isso, foi dividido o código em dois pacotes principais, sendo estes, o *Include* e o Source. O primeiro envolve os arquivos *headers*, os quais contém as declarações das funções, das variáveis, e dos macros. As classes do pacote *Source* contém a implementação da lógica de negócio de cada função declarada nos *headers* presentes no pacote Include. O diagrama que representa estas definições pode ser visto na Figura 9.1.

Uma das principais funcionalidades implementadas do SmarTrack Device, é o bloqueio

da parametrização quando é inicializado o processo de transporte. Para isso, optou-se por verificar o valor da característica *IsWriteBlocked* antes de escrever em quaisquer características do *Transportation Service*. A característica *IsWriteBlocked* tem o valor alterado para *true* no momento em que o SmarTrack Device recebe uma solicitação de escrita na característica *RecipientId*.

Portanto, uma vez iniciado o transporte, somente o destinatário pode finalizar. Este processo consiste basicamente em redefinir os parâmetros para o valor por defeito, e a característica *IsWriteBlocked* é alterada para *false*.

Ainda no tocante à implementação, alguns sensores enviam e recebem dados em formatos distintos, os quais estão definidos na documentação de cada sensor. Por este motivo, foi necessário implementar os códigos relacionados com a conversão de valores dos sensores para os serviços BLE. Para isso, foram utilizados os *drivers* já implementados e testados no Thingy52, os quais são disponibilizados no repositório da Nordic Semiconductor[1]. Estes *drivers* contêm toda a lógica de negócio necessária para a integração dos sensores com o SmarTrack Device, desde a leitura até conversões de valores e eventuais calibrações.

Implementou-se também uma solução para remoção dos dados quando estes foram sincronizados. Neste caso, a estrutura elaborada consiste em remover os dados do dispositivo se, e somente se, um *smt token* válido for escrito na característica *Synchronized Data*. Para isso, quando o *smt token* é recebido, um processo de validação deste valor é iniciado utilizando um algoritmo de validação[2].

Por fim, após ter a lógica do SmarTrack Device para recolha e configuração dos dados, iniciou-se a implementação do CS.

6.2 *Cloud Service*

No que concerne ao **CS**, optou-se por simplificar em um WS, o qual fornece acesso a recursos disponibilizados por meio de uma API REST. Esta é dividida em cinco pacotes

[1] https://github.com/NordicSemiconductor/Nordic-Thingy52-FW
[2] Este algoritmo é apresentado no Apêndice 9.4

principais, tais como, *Facade*, *Entities*, *Services*, *Security* e *Helpers* - pacote de classes utilizadas para auxiliar no processamento de determinadas funcionalidades. A distribuição dos pacotes e a interação entre eles pode ser visualizada no diagrama de pacotes do CS (Figura 9.6).

6.2.1 *Entities*

O pacote *Entities* é composto por classes que representam as tabelas do BD relacional, o qual pode ser observado na Figura 6.2 no formato de DER. A partir do BD, é possível obter dados históricos do transporte, como os operadores responsáveis pela mercadoria, local, data e hora da leitura, qual o identificador do dispositivo, valores recolhidos pelos sensores, etc. Dados esses que são utilizados tanto para detetar violações das condições de transporte, quanto para mitigar adulterações nos dados, buscando garantir que os dados recolhidos são fidedignos. Um código escrito em SQL, cujo objetivo é obter os dados da tabela **as_readings_data** e seu resultado, pode ser visualizado no Apêndice 9.

Além de dados de transporte, é possível também, pelo BD, controlar quais GUIs os usuários têm acesso, utilizando para isso, a tabela **as_role_pages** (Apêndice 9).

Portanto, os dados armazenados no BD podem influenciar no comportamento do aplicativo, além de fazer parte do processo de decisão dos usuários no tocante a receção da mercadoria. Para isso, foi implementado recursos de segurança como, autenticação, autorização e encriptação. Estes recursos estão nas classes contidas no pacote *Security*.

6.2.2 *Security*

Para autenticação e autorização, foi implementada uma adaptação do código-fonte elaborado por Tarcísio Carvalho[3], o qual faz o uso da biblioteca **jjwtt** (Java JSON Web Token) para gerar e validar o *cs token* (abordados na subsecção 4.2.1). Permitindo assim que seja feita a autenticação e autorização dos usuários, limitando o acesso aos recursos disponíveis no CS apenas para usuários devidamente autorizados. Exceto as requisições

[3]https://github.com/tarcCar/RestAutenticacaoPorToken;

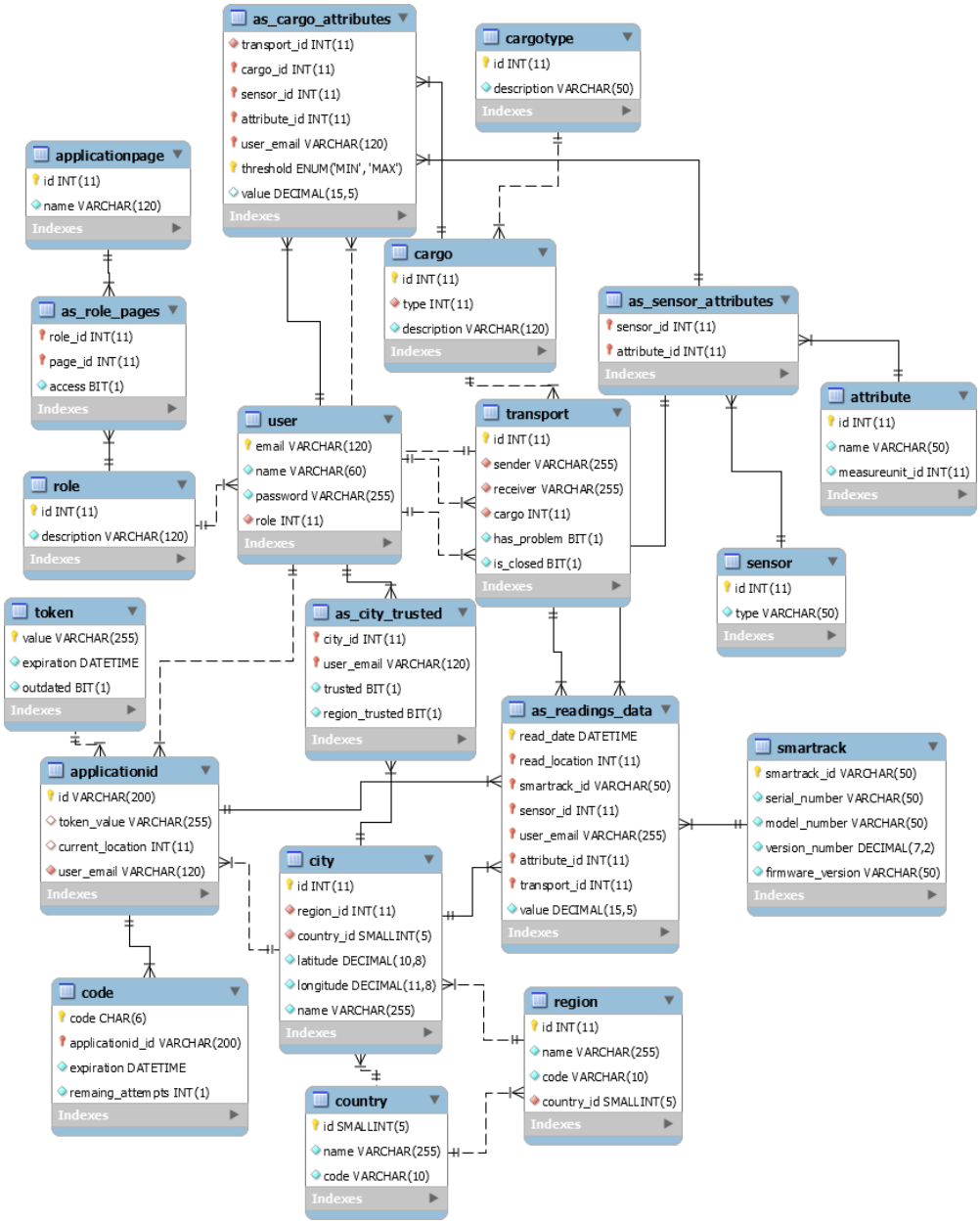


Figura 6.2: Diagrama Entidade Relacionamento - Banco de Dados

para registo, em todas as requisições da API REST, é feita a validação das credenciais do usuário (endereço de e-mail e palavra-passe), as quais são recebidas no cabeçalho do protocolo HTTPS.

Ainda no tocante ao processo de registo e autenticação, foi identificada a necessidade em enviar uma mensagem com uma OTP para o usuário. Para isso, optou-se pelo uso da

biblioteca *javax.mail* para enviar a OTP utilizando mensagens de e-mail.

Também foi necessário realizar o processamento nas imagens que passaram pelo processo de esteganografia para ocultar os dados do usuário no arquivo da imagem, para que, pudesse verificar a veracidade das imagens através dos dados ocultados nas imagens. Para isso, adaptou-se o algoritmo criado desenvolvido na LP C# por Hamzeh Soboh[4] para a LP Java.

6.2.3 *Facades* e *Services*

Estes dois pacotes contêm classes que auxiliam na definição dos serviços que serão disponibilizados para que os aplicativos possam consumir tais serviços, realizando operações de forma indireta no BD.

Serviços estes que estão relacionados com a inserção dos dados recolhidos do SmarTrack Device pelo aplicativo, registo de usuários, acesso aos dados, envio de OTP, etc. Tais dados são enviados por meio de requisições POST, e devidamente serializados com auxílio das bibliotecas *gson*[5], e *javax.json*[6]. Na secção 9.1, são apresentadas as *Uniform Resources Identifiers* (URIs) disponibilizadas para comunicação entre CS e aplicativos.

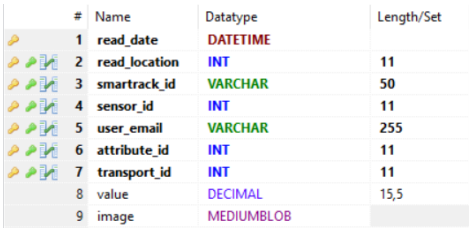
Além de lidar com os dados mencionados anteriormente, identificou-se a necessidade em implementar um recurso para armazenamento de imagens devido ao serviço que permite a comprovação das condições da mercadoria por imagem. Neste caso, foram consideradas duas opções, sendo a primeira, a utilização de um serviço de transferência de arquivos, enquanto a outra, armazenar as imagens em um conjunto de *bytes* no próprio BD.

Diante destas duas vertentes, optou-se pelo uso do BD para armazenamento de imagens, bastando apenas, acrescentar um novo sensor e seus respetivos atributos as tabelas

[4]https://www.codeproject.com/Tips/635715/Steganography-Simple-Implementation-in-Csharp?fid=1839244&df=90&mpp=25&sort=Position&view=Normal&spc=Relaxed&select=4721926&fr=37&prof=True

[5]https://github.com/google/gson

[6]https://docs.oracle.com/javaee/7/api/javax/json/package-summary.html



#	Name	Datatype	Length/Set
1	read_date	DATETIME	
2	read_location	INT	11
3	smartrack_id	VARCHAR	50
4	sensor_id	INT	11
5	user_email	VARCHAR	255
6	attribute_id	INT	11
7	transport_id	INT	11
8	value	DECIMAL	15,5
9	image	MEDIUMBLOB	

Figura 6.3: Banco de Dados - Tabela *as_readings_data atualizada.*

"*as_sensor_attributes*", "*sensor*" e "*attribute*"[7]. Inclui-se também uma nova coluna[8] responsável por lidar com dados do tipo *MediumBlob*. Sendo assim, a Figura 6.3, mostra a tabela "*as_readings_data*" atualizada.

Assim sendo, foram desenvolvidos novos métodos acessíveis através das URIs, como **data/insert/img**, **data/get/img**, respetivamente para receção/validação e consulta de tais imagens.

Finalizado o desenvolvimento do CS, iniciou-se o desenvolvimento dos aplicativos Android e iOS, os quais fazem o uso tanto dos dados do CS, quanto dos dados do SmarTrack Device.

6.3 Aplicativos

O **aplicativo** é organizado por cinco pacotes principais, entre estes estão, o *Services*, os pacotes do MVVM e *Helpers* - classes contendo códigos utilitários como conversão de valores, constantes e códigos relacionados com criptografia.

Inicialmente, implementou-se as funcionalidades das classes contidas no pacote *Services*, as quais contêm implementações para realizar a comunicação com a API REST. Os dados enviados e recebidos pelo CS, precisaram ser serializados, para tal, utilizou-se o plugin *NewtonSoft.Json*[9]. Na secção 9.6. é mostrado um exemplo de código utilizado para solicitar um recurso da API.

[7]Essa mudança pode ser observada no resultado das consultas apresentado no Apêndice 9.1

[8]Estudou-se a possibilidade em alterar o tipo de dados da coluna "value" para MediumBlob, podendo aceitar qualquer tipo de dados

[9]https://www.newtonsoft.com/json/help/html/Introduction.htm

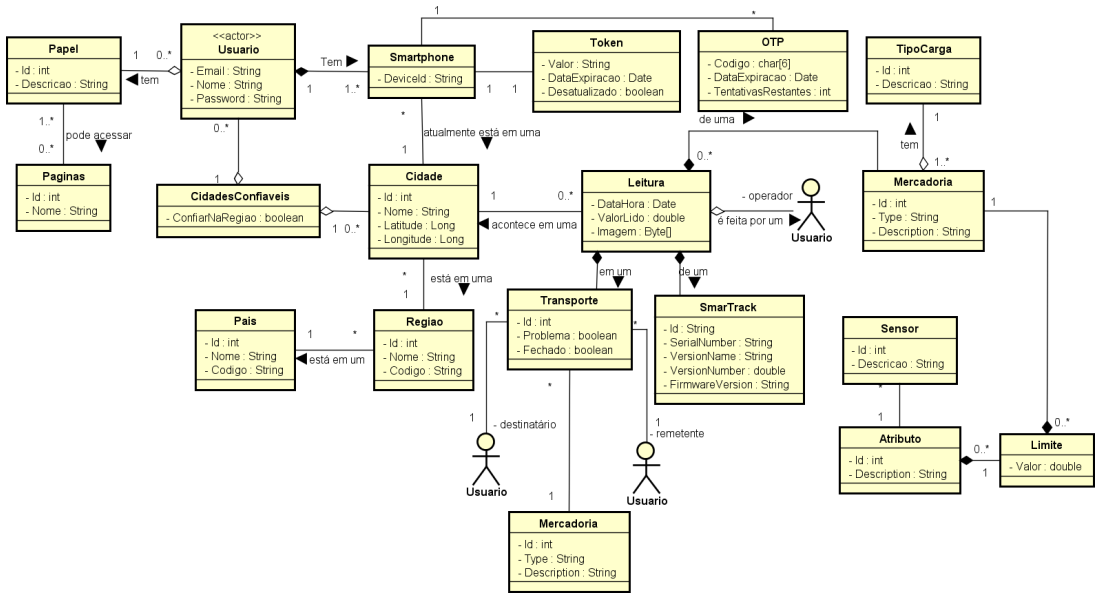
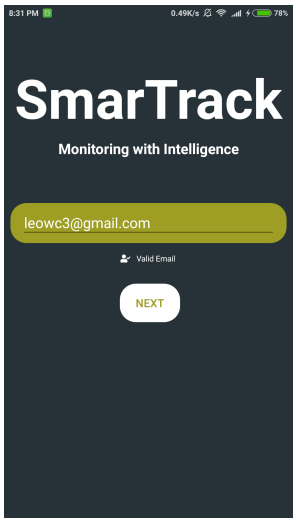
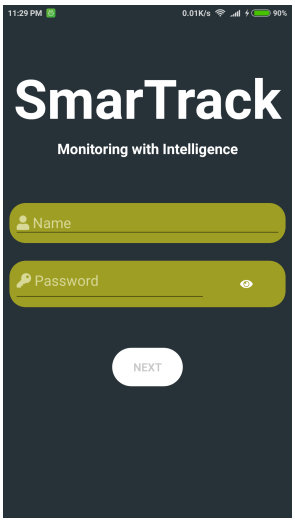


Figura 6.4: Diagrama de Classe - Aplicativos

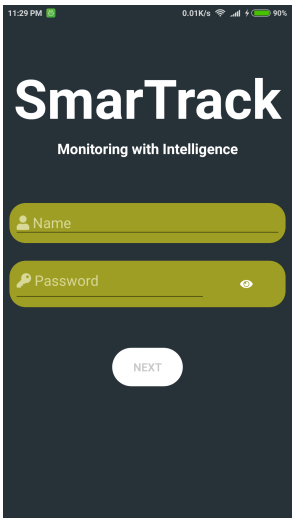
Após implementar as classes para utilização dos recursos disponibilizados pelo CS, implementou-se as classes do pacote *model*, as quais são utilizadas para manipulação de dados da solução e são representadas pelo Diagrama de Classes (Figura 6.4).



(a) GUI de validação do endereço de e-mail para registo/autenticação

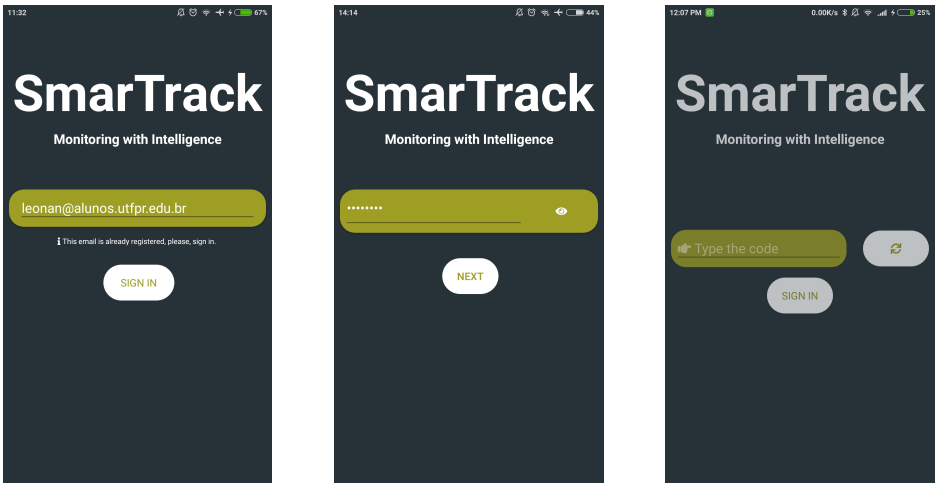


(b) GUI de inserção dos dados para registo



(c) GUI de inserção dos dados para autenticação

Figura 6.5: *Graphical User Interfaces* para registo de utilizadores.



(a) GUI para autenticação, etapa 1/3

(b) GUI de inserção dos dados para autenticação , etapa 2/3

(c) GUI de inserção dos dados para autenticação, etapa 3/3

Figura 6.6: *Graphical User Interfaces* para autenticação de utilizadores

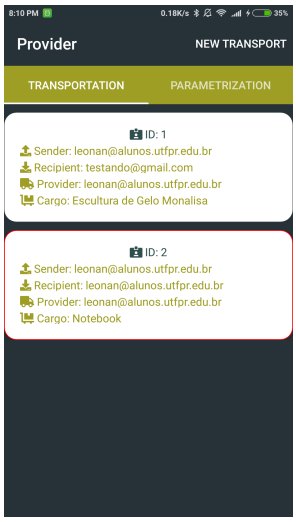
Na sequência, iniciou-se o desenvolvimento das GUIs e suas respetivas lógicas de negócio - presentes nos pacotes *Views* e *ViewModels*. Para o desenvolvimento das GUIs optou-se por seguir as boas práticas de design recomendadas pelo *Material Design*[10]. Logo, implementou-se as GUIs para registo (Figuras 6.5.a, 6.5.b, 6.5.c) e autenticação (Figuras 6.6.a, 6.6.b, 6.6.c), as quais são utilizadas por todos os usuários.

Uma vez registado, sempre que o aplicativo é iniciado, é feito o processo de autenticação (Figura 4.4), para tal, consulta-se os dados do utilizador por meio dos recursos, "auth/get/applicationid", "auth/code/app" e "auth/get/trustedcities".

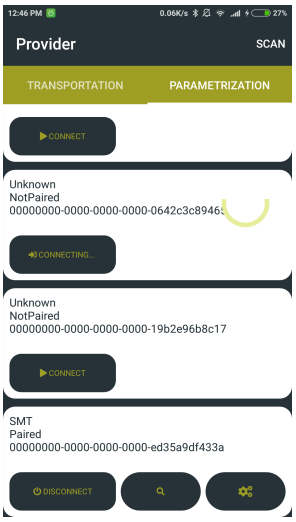
Em seguida, identificou a necessidade em diferenciar as GUIs, consoante os perfis dos usuários, para esse fim, foram criados outros quatro pacotes dentro dos pacotes *Views* e *ViewModels*, sendo estes, *ProviderPages*, *SenderPages*, *ReceiverPages*, e *OperatorPages*.

No pacote *ProviderPages* há GUIs (Figuras 6.7.a, 6.7.b, 6.8.a, 6.8.b) que fornecem métodos para que o prestador inicie um processo de transporte, faça a parametrização dos dados de transporte no SmarTrack Device, assim como os possíveis ajustes em tais

[10]https://material.io/design/

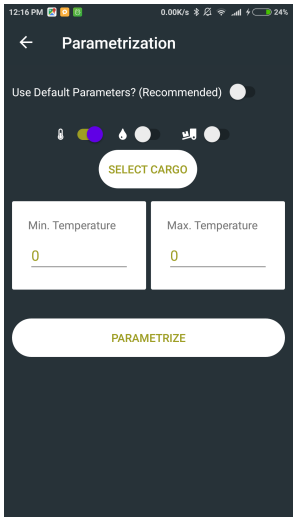


(a) GUI para consulta dos transportes contratualizados

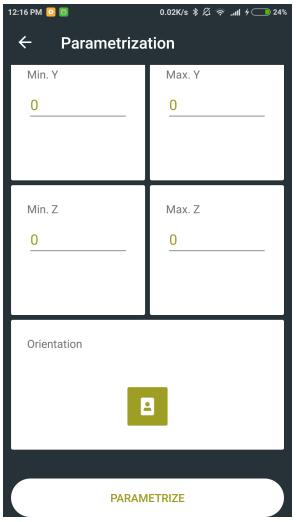


(b) GUI para parametrização, etapa 1/3

Figura 6.7: *Graphical User Interfaces* para Prestadores Parte 1



(a) GUI para parametrização, etapa 2/3



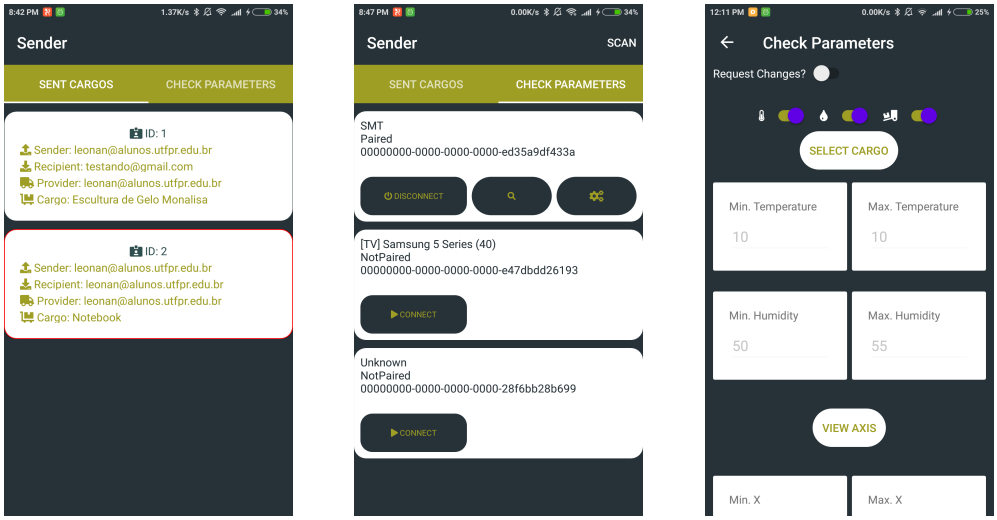
(b) GUI para parametrização, etapa 3/3

Figura 6.8: *Graphical User Interfaces* para Prestadores Parte 2

parâmetros. É possível observar na Figura 6.7.a que há um contorno em tons de vermelho nos dados do transporte, o qual representa que neste transporte, foram violadas as condições contratualizadas.

Para comunicar-se com o SmarTrack Device foi necessário implementar as regras de

um dispositivo *client* da arquitetura do BLE, o qual é responsável por ler e escrever, características disponibilizadas pelo SmarTrack Device. Para isso, utilizou-se o *package* *Plugin.BluetoothLE*[11], assim como métodos para conversão de valores enviados e recebidos pelo SmarTrack Device.



(a) GUI para consultar transportes contratualizados.

(b) GUI para verificação de parâmetros (1/5)

(c) GUI para verificação de parâmetros (2/5)

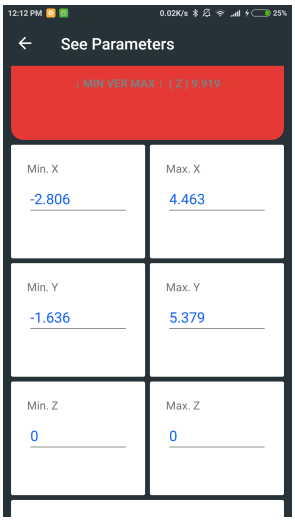
Figura 6.9: *Graphical User Interfaces* para Remetentes - parte 1

A comunicação entre aplicativo e SmarTrack Device ocorre também em outras GUIs, tais quais as dos remetentes (Figuras 6.9.a, 6.9.b, 6.9.c, 6.10.a, 6.10.b, e 6.10.c). Para este perfil de usuário, implementou-se GUIs que proveem funcionalidades para que os remetentes possam validar os parâmetros para monitorização da mercadoria, além de consultar os dados do histórico do transporte. Este processo de consulta dos dados é utilizado também para os operadores de transporte e os destinatários.

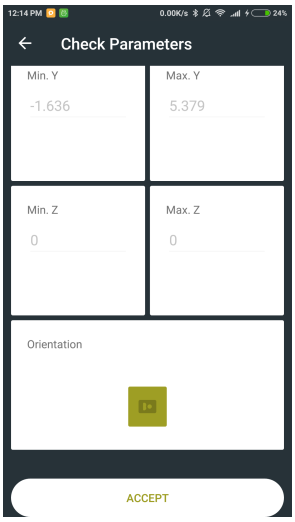
Para os destinatários, desenvolveu-se GUIs, para que estes, pudessem finalizar o ciclo de transporte da mercadoria, informando se aceita, ou não, a receção da mercadoria. Quanto aos remetentes e destinatários, desenvolveu-se também, uma GUI que permite a visualização dos históricos de transporte, tais quais eventuais imagens capturadas pelos operadores de transporte, e os dados recolhidos pelos sensores do SmarTrack Device.

[11]https://github.com/aritchie/bluetoothle

(a) GUI para verificação de parâmetros (3/5)



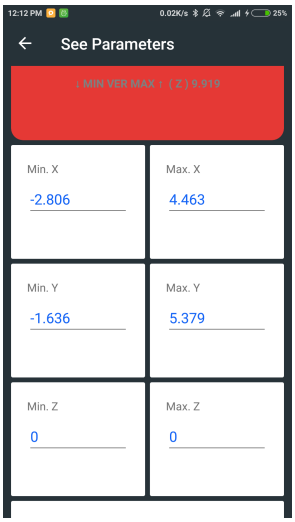
(b) GUI para verificação de parâmetros (4/5)



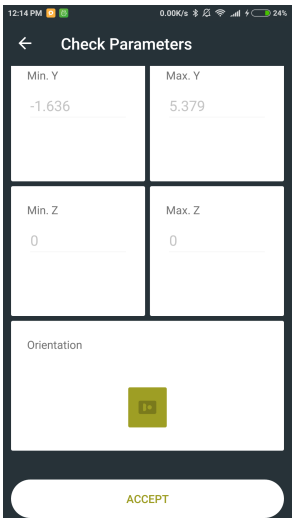
(c) GUI para verificação de parâmetros (5/5)

Figura 6.10: *Graphical User Interfaces* para Remetentes - parte 2

(a) GUI para verificação verificação das condições



(b) GUI para captura e envio de imagens (1/2)



(c) GUI para captura e envio de imagens (2/2)

Figura 6.11: *Graphical User Interfaces* para Operadores

Por fim, em relação aos operadores de transporte, foram implementadas GUIs, para que estes possam confirmar as condições da mercadoria (Figura 6.11.a) - aquando da sua passagem, além de outras duas GUIs, sendo uma para confirmação de receção da mercadoria, e outra, para captura das imagens da mercadoria (Figura 6.11.a e 6.11.b).

No momento em que o operador confirma a captura da imagem, é aplicado o processo de esteganografia na imagem, e na sequência, esta, é enviada para o servidor.

Após finalizar a implementação, realizou-se testes para validar o funcionamento da solução SmarTrack.IO.

6.4 Testes

Nesta secção, serão apresentados alguns testes, tais quais, o funcionamento dos algoritmos para leitura de dados de um dispositivo BLE, apresentação dos testes de consumo energético, e o envio de dados do aplicativo para o SmarTrack Device.

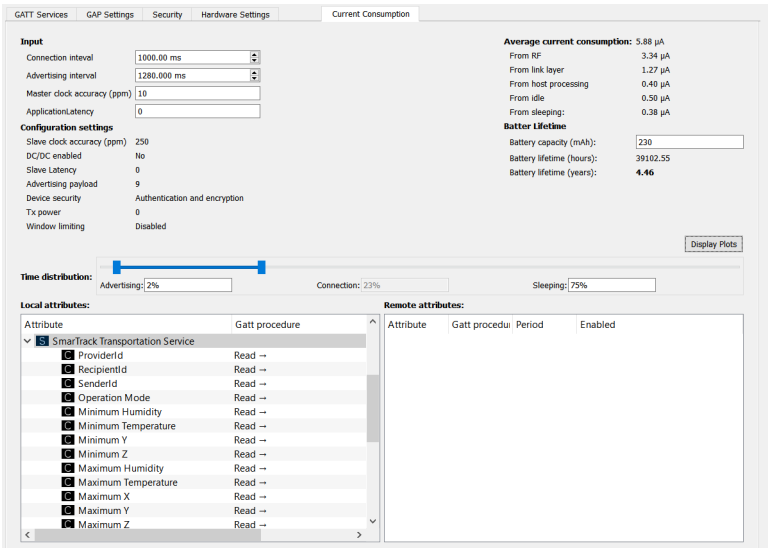
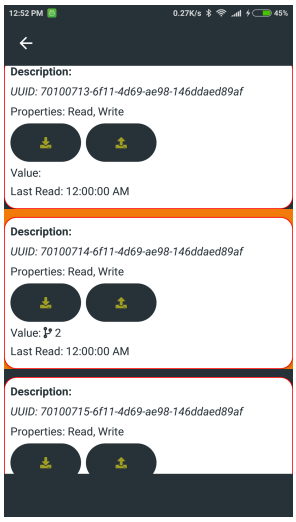
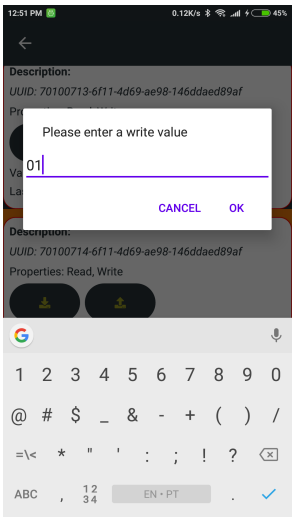


Figura 6.12: Estimativa de consumo energético do SmarTrack Device

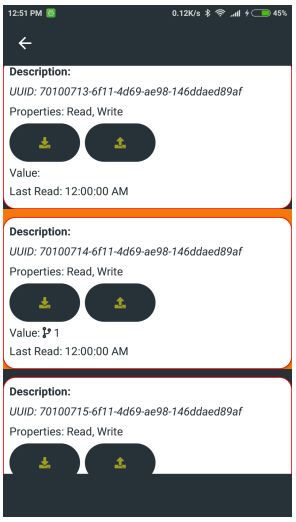
Para validar o consumo energético, foi realizada uma estimativa utilizando a ferramenta nRF GO Studio, o qual apresentou os resultados estimados para consumo de aproximadamente, 4 anos e 5 meses (Figura 6.12), considerando que o dispositivo, em todo tempo útil de vida da bateria, ficará em estado de conexão em 23% do tempo, e 75% em modo de hibernação. Em contrapartida, é importante ressaltar que a ferramenta não acresce o consumo dos sensores, os quais, mesmo assim, são de baixíssimo consumo energético.



(a) GUI para leitura do valor armazenado na característica, Modo de Operação

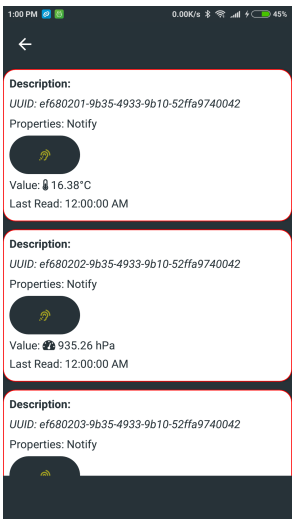


(b) GUI para escrita de valor na característica, Modo de Operação

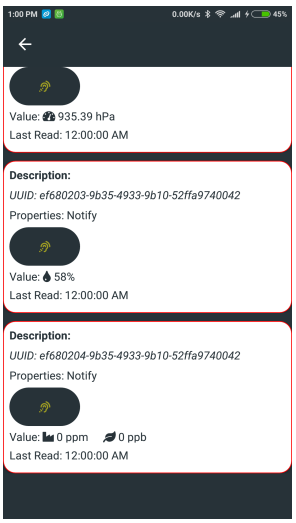


(c) GUI para leitura do valor escrito na característica, Modo de Operação

Figura 6.13: *Graphical User Interfaces* para Operação nos Dados do SmarTrack Device - parte 1



(a) GUI para leitura de dados dos sensores (1/2)



(b) GUI para leitura de dados dos sensores (2/2)

Figura 6.14: *Graphical User Interfaces* para Operação nos Dados do SmarTrack Device - parte 2

No tocante ao envio de dados do aplicativo, foi possível observar que as solicitações de escrita enviadas pelo aplicativo para o SmarTrack Device, foram realizadas com sucesso,

assim como a leitura dos dados das características, como exemplo, os dados dos sensores, conforme pode ser visto nas Figuras 6.13.a, 6.13.b, 6.13.c, 6.14.a, 6.14.b, e 6.14.c.

Quanto a implementação e testes das leituras de valores em um modo de operação, não foi possível realizar, porque o Thingy52, implementa alguns requisitos de segurança para compilação e execução de *firmwares* desenvolvidas por terceiros, como no caso do SmarTrack Device. Fator este que impediu a compilação do código no Thingy52.

No entanto, a lógica de negócios utilizada no Thingy52 para leituras intervalares, foi a base para o modo de operação *Interval Mode*, do SmarTrack Device. Neste sentido, foi possível realizar os testes para leitura dos dados dos sensores, configurando os intervalos de leitura do Thingy52, utilizando, para isso, os mesmos métodos do SmarTrack Device. Portanto, não foi possível apenas, testar os modo de operação *Threshold Mode*, e por consequência, o *Dual Mode*.

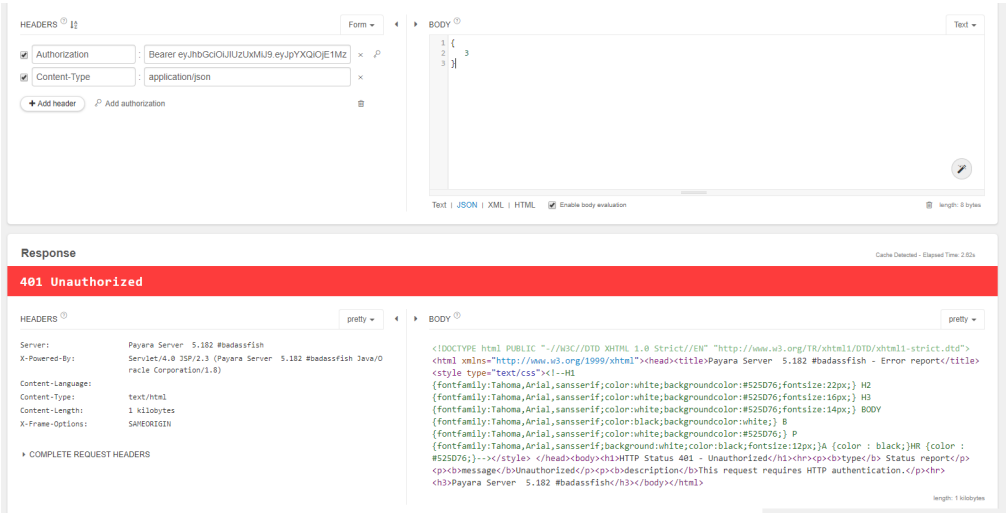


Figura 6.15: Exemplo de uma solicitação sem autorização para o *Cloud Service*

No tocante ao CS, foram feitos testes de autorização de acesso a determinados dados, os quais, só devem ser acedidos mediante envio do *cs token*, para validação e permissão de acesso, no cabeçalho do protocolo HTTPS, conforme pode ser visto na Figura 6.15.

Capítulo 7

Conclusão

No transporte de mercadorias, falhas podem ocorrer, estas que podem causar prejuízos a todas as entidades envolvidas neste processo. Considerando o fato, de que há soluções, tais quais as apresentadas neste documento, que buscam solucionar problemas parecidos, é possível afirmar que é viável o desenvolvimento de uma solução para deteção de falhas, apuramento de responsabilidades, que não altere o modelo operacional das empresas, sendo ainda, de baixo custo financeiro. Neste contexto, com esta dissertação pretendeu-se, analisar e desenvolver, uma solução de baixo custo, capaz de lidar com tais dados.

Conclui-se também, em uma análise preliminar baseada em outras soluções, que, o modelo proposto neste documento não impacta de forma significativa no modelo operacional, considerando que a solução foi elaborada de forma a ser possível efetuar sua integração no momento de fecho da mercadoria.

Diante dos testes apresentados, é de notar também que, todo o protocolo definido para recolha, armazenamento, e consulta dos dados, estão adequados aos objetivos propostos neste documento, sendo possível recuperar dados históricos do transporte, os quais identificam o momento, o local, quais condições eventualmente violadas, além do responsável pelo ocorrido.

Apesar de não ter sido validada a possibilidade de recolha dos dados pelos sensores, aquando da violação das condições, diretamente pelo SmarTrack Device, foi possível observar que, mesmo assim, o *firmware* do Thingy52 fornece implementações já testadas

para estes fins.

Diante da relevância dos dados coletados para fins de tomada de decisão, notou-se uma importância da análise, e implementação, de conceitos que permitissem que os dados pudessem ser validados, afim de garantir que tais dados são fidedignos.

Para isso, foi possível notar que, a contextualização temporal e geográfica dos dados, foi uma estratégia adequada para mitigação de possíveis adulterações nos dados recolhidos, haja vista o exemplo das fotografias, que por sua vez, são validadas utilizando dados temporais, geográficos, entre outros. No entanto, há possibilidades em ir além no tocante a fidedignidade dos dados, para isso, constatou-se que, a utilização de *blockchains* poderá dificultar a ação de adulteração em tais dados.

Além das soluções elaboradas para propiciar autenticidade dos dados, foi possível concluir que os conceitos de segurança aplicados nesta solução, permitiram ainda, mitigar um conjunto de ações que pudessem colocar os dados dos utilizadores em risco, ou ainda, comprometer a veracidade de tais dados.

Ademais, pode-se concluir também que, a partir da análise e implementação do conteúdo proposto neste documento, é possível produzir o *hardware* designado neste documento por SmarTrack Device, por intermédio de uma *Printed Circuit Board*, a um custo de até, quatro vezes menor, quando comparado ao custo em adquirir um Thingy52. Haja vista que, no momento da escrita deste documento, o preço final, para consumidor do Thingy52, poderia ser encontrado em um intervalo de preço entre 40 a 75 dólares.

7.1 Trabalhos Futuros

Por fim, identificou-se outros trabalhos futuros, que, mesmo não sendo fundamentais para o funcionamento da solução, podem agregar ainda mais valor. Tais trabalhos envolvem, a implementação de uma solução que permita realizar atualização no código final do SmarTrack Device - sem a necessidade em fazer a desassemblagem, utilizando para isso, o conceito de *Device Firmware Update Over the Air*. Implementação de mecanismos para ativar o SmarTrack Device à distância, sem necessariamente, ter de iniciar o processo de

advertising por um botão físico, utilizando por exemplo, a tecnologia *Near Field Communication*. Inclusão de um componente físico, que permita vincular o SmarTrack Device à mercadoria, de forma inviolável, ou, ao menos, identificar quando este componente é violado.

Bibliografia

[1] H. L. Hasegawa, D. Venanzi e O. R. Silva, "A CADEIA DE SUPRIMENTOS NO SETOR HOSPITALAR: TRANSPLANTE DE ÓRGÃOS", *Revista UNIABEU Belford Roxo*, vol. 7, nº 15, pp. 195–209, 2014.

[2] D. Evans, "The Internet of Everything", pp. 1–9, 2012.

[3] J. Gubbi, R. Buyya, S. Marusic e M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions", *Future Generation Computer Systems*, vol. 29, nº 7, pp. 1645–1660, 2013, ISSN: 0167739X. DOI: 10.1016/j.future.2013.01.010. arXiv: 1207.0203.

[4] L. Atzori, A. Iera e G. Morabito, "The Internet of Things: A survey", *Computer Networks*, vol. 54, nº 15, pp. 2787–2805, 2010, ISSN: 13891286. DOI: 10.1016/j.comnet.2010.05.010. arXiv: arXiv:1011.1669v3. endereço: http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568.

[5] T. Saarikko, U. H. Westergren e T. Blomquist, "The Internet of Things : Are you ready for what ' s coming ?", *Business Horizons*, vol. 60, nº 5, pp. 667–676, 2017, ISSN: 0007-6813. DOI: 10.1016/j.bushor.2017.05.010. endereço: http://dx.doi.org/10.1016/j.bushor.2017.05.010.

[6] Bluetooth SIG, D. Rui, P. Paiva e E. R. Jegundo, "Bluetooth Core Specification Version 5.0, Volume 1 Part A", *Bluetooth Core Specification Version 5.0*, nº December, pp. 161–264, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[7] ——, “Bluetooth Core Specification Version 5.0, Volume 0 Part C”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 90–157, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[8] B. SIG, “Bluetooth Low Energy Security Modes and Procedures”, 2018. endereço: http://blog.bluetooth.com/exploring-bluetooth-5-going-the-distance (acedido em 24/09/2018).

[9] Bluetooth SIG, D. Rui, P. Paiva e E. R. Jegundo, “Bluetooth Core Specification Version 5.0, Volume 6 Part B”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 280–292, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[10] ——, “Bluetooth Core Specification Version 5.0, Volume 3 Part A”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 161–264, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[11] ——, “Bluetooth Core Specification Version 5.0, Volume 3 Part F”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 161–264, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[12] ——, “Bluetooth Core Specification Version 5.0, Volume 3 Part G”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 161–264, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[13] ——, “Bluetooth Core Specification Version 5.0, Volume 3 Part C”, *Bluetooth Core Specification Version 5.0*, nº December, pp. 161–264, 2016. endereço: https://www.bluetooth.org/en-us/specification/adopted-specifications.

[14] C. e. Morimoto, *Smartphones - guia pratico*, SN. Porto Alegre, RS: Editora Sulina, 2009, p. 431, ISBN: 978-85-99593-14-1.

[15] Statista, *Market share worldwide smartphone shipments by operating system from 2014 to 2022*, 2018. endereço: https://www.statista.com/statistics/272307/

market-share-forecast-for-smartphone-operating-systems/ (acedido em 13/07/2018).

[16] ——, *Number of Smartphone users worldwide from 2014 to 2020*, 2018. endereço: http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (acedido em 13/07/2018).

[17] J. B. Monteiro, *Google Android - Crie aplicações para celulares e tablets*. São Paulo: Casa do Código, 2012, p. 307, ISBN: 9780874216561. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3.

[18] A. S. Tanenbaum, *Modern Operating Systems*, 4ª Edição. New Jersey: Pearson, 2014, p. 1137, ISBN: 9780133591620. DOI: 10.1142/S0129183108012261. arXiv: 0710.2023. endereço: http://www.worldscientific.com/doi/abs/10.1142/S0129183108012261.

[19] R. R. Lecheta, *Google Android - Aprenda a criar aplicações para dispositivos móveis*, 4ª ed., R. Prates, ed. São Paulo: Novatec, 2015, p. 1016, ISBN: 9788575224687.

[20] N. Sofiyanti, D. I. Fitmawati e A. A. Roza, "Stenochlaena Riauensis (Blechnaceae), A new fern species from riau, Indonesia", *Bangladesh Journal of Plant Taxonomy*, vol. 22, nº 2, pp. 137–141, 2015, ISSN: 10282092. DOI: 10.1007/s13398-014-0173-7.2. arXiv: arXiv:1011.1669v3.

[21] M. Nosrati, "Mobile Computing : Principles , Devices and Operating Systems", nº July, pp. 399–408, 2012.

[22] A. M. Rocha e R. M. F. Neto, "Introdução à Arquitetura Apple iOS", 2010.

[23] V. de Mendonça, T. Bittar e M. d. S. Dias, "Um estudo dos Sistemas Operacionais Android e iOS para o desenvolvimento de aplicativos", *Enacomp.Com.Br*, 2011. endereço: http://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011%7B%5C_%7Dsubmission%7B%5C_%7D54.pdf.

[24] T. Renner, "Mobile OS - Features , Concepts and Challenges for Enterprise Environments",

[25] Apple, *App Documentation*, 2018. endereço: https://developer.apple.com/documentation/ (acedido em 16/08/2018).

[26] E. Coimbra, *Everton Coimbra de Araújo*. São Paulo: Casa do Código, 2018.

[27] Amazon, *What is Cloud Computing*, 2018. endereço: https://aws.amazon.com/pt/what-is-cloud-computing/ (acedido em 16/08/2018).

[28] A. M. Antonopoulos, *Mastering Bitcoin*, 4. 2016, vol. 2, pp. 675–704, ISBN: 9781107671812. DOI: 10.1002/ejoc.201200111. eprint: arXiv:1011.1669v3. endereço: https://www.bitcoinbook.info/.

[29] M. Gupta, *Blockchain for Dummies*, 2ª ed. 2018, ISBN: 9781119545934.

[30] H.-N. Dai, Q. Wang, D. Li e R. C.-W. Wong, "On Eavesdropping Attacks in Wireless Sensor Networks with Directional Antennas", *International Journal of Distributed Sensor Networks*, vol. 9, nº 8, p. 760 834, 2013, ISSN: 1550-1477. DOI: 10.1155/2013/760834. endereço: http://journals.sagepub.com/doi/10.1155/2013/760834.

[31] Avast, *Spoofing*, 2018. endereço: https://www.avast.com/pt-br/c-spoofing (acedido em 16/08/2018).

[32] P. Luckett, J. T. Mcdonald e W. B. Glisson, "Attack-Graph Threat Modeling Assessment of Ambulatory Medical Devices", pp. 3648–3657, 2017.

[33] Google, "Android Security 2017 Year In Review", nº March, 2018. endereço: https://source.android.google.cn/security/reports/Google%7B%5C_%7DAndroid%7B%5C_%7DSecurity%7B%5C_%7D2017%7B%5C_%7DReport%7B%5C_%7DFinal.pdf.

[34] ——, *Security Tips*, 2018. endereço: https://developer.android.com/training/articles/security-tips (acedido em 16/08/2018).

[35] Apple, *Secure Coding Guide*, 2018. endereço: https://developer.apple.com/library/archive/documentation/Security/Conceptual/SecureCodingGuide/SecurityDevelopmentChecklists/SecurityDevelopmentChecklists.html (acedido em 16/08/2018).

[36] Oracle, *Fusion Middleware Security and Administrator's Guide for Web Services*, 2018. endereço: https://docs.oracle.com/cd/E17904_01/web.1111/b32511/intro_security.htm%7B%5C#%7DWSSEC1112 (acedido em 16/08/2018).

[37] J. Li, "Issues of food-related cold-chain logistics management in China", *2010 International Conference on Logistics Systems and Intelligent Management, ICLSIM 2010*, vol. 3, pp. 1319–1322, 2010. DOI: 10.1109/ICLSIM.2010.5461178.

[38] F. Li e Z. Chen, "Brief analysis of application of RFID in pharmaceutical cold-chain temperature monitoring system", *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering, TMEE 2011*, pp. 2418–2420, 2011. DOI: 10.1109/TMEE.2011.6199709.

[39] W.-J. Liao, T.-W. Lien, B.-R. Hsiao, H.-S. Wang, C.-F. Yang, J.-H. Tarng, C.-C. Nien e T.-C. Chiu, "Sensor Integrated Antenna Design for Applications in Cold Chain Logistic Services", *IEEE Transactions on Antennas and Propagation*, vol. 63, nº 2, pp. 727–735, 2015, ISSN: 0018-926X. DOI: 10.1109/TAP.2014.2384048. endereço: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6991530.

[40] Y. Bo e L. Danyu, "Application of RFID in cold chain temperature monitoring system", *2009 Second ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2009*, vol. 2, pp. 258–261, 2009. DOI: 10.1109/CCCM.2009.5270408.

[41] J. Zhang, L. Chen, X. Tang e Q. Gao, "A remote monitoring system for cold chain logistics by means of social networks", *Open Cybernetics and Systemics Journal*, vol. 9, pp. 888–893, 2015. DOI: 10.2174/1874110X01509010888. endereço: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84964344736%7B%5C&%7DpartnerID=40%7B%5C&%7Dmd5=abee6211e00d2d210fa81b32f798ef07.

[42] X. Zong e H. Shao, "Design of Cold Chain Logistics Remote Monitoring System based on ZigBee and GPS Location", *Advances in Materials, Machinery, Electronics I*, vol. 1820, 2017, ISSN: 0094-243X. DOI: 10.1063/1.4977397.

[43] G. Wang, W. Xu e M. Wang, "The grid-computing based instrumented monitoring platform for cold chain logistics", *2010 International Conference on Logistics Engineering and Intelligent Transportation Systems, LEITS2010 - Proceedings*, pp. 402–404, 2010. DOI: 10.1109/LEITS.2010.5664946.

[44] D. Ko, Y. Kwak, D. Choi e S. Song, "Design of cold chain application framework (CCAF) based on IOT and cloud", *Proceedings - 8th International Conference on u- and e-Service, Science and Technology, UNESST 2015*, pp. 11–13, 2016, ISSN: 17389984 (ISSN). DOI: 10.1109/UNESST.2015.8.

[45] STMicroelectronics, "Datasheet LIS3DH", nº May, pp. 1–48, 2015.

[46] ——, "Capacitive digital sensor for relative humidity and temperature", nº April, pp. 1–31, 2015. endereço: http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00116291.pdf.

[47] N. Semiconductor, *Power Profiler Kit*, 2018. endereço: https://www.nordicsemi.com/eng/Products/Power-Profiler-Kit (acedido em 24/08/2018).

[48] D. Britch, "Prism for the Windows Runtime : Developing a Windows Store business app using", nº May, 2013.

[49] Apple, *App Store Review Guidelines*, 2018. endereço: https://developer.apple.com/app-store/review/guidelines/%7B%5C#%7Dsafety (acedido em 16/08/2018).

[50] N. Semiconductor, *Thingy52 Documentation*, 2018. endereço: https://infocenter.nordicsemi.com/index.jsp?topic=%5C%2Fcom.nordic.infocenter.rds%5C%2Fdita%2Frds%5C%2Fdesigns%5C%2Fthingy%5C%2Fhw_description%5C%2Fhw_descr.html (acedido em 22/08/2018).

Capítulo 8

Definições Complementares das Tecnologias

8.1 Os Quatro Pilares do Blockchain

Além dos conceitos explicados no Capítulo 3, há ainda, quatro conceitos que são conhecidos como, os quatro pilares do *blockchain*, os quais são explicados na sequência [29]:

Shared Ledger: o *Ledger*, ou ainda, livro-razão, é um tipo de documento utilizado em contabilidade desde o século XV. No blockchain, o **Shared Ledger** é um conceito similar, que traz como novidade, o *Ledger* de uma forma compartilhada e distribuída, fornecendo um registo inalterável de todas as transações realizadas na rede, e que, todos os participantes desta mesma rede possam os aceder. As principais características dos *Shared Ledgers* são que, este é o único sistema de registo, é compartilhado entre todos os participantes através da replicação dos *Ledgers* e os participantes podem visualizar apenas as transações as quais estão autorizados a ver, utilizando identidades que os associam a tais transações;

Permissions*:** os blockchains podem ser ***permissioned ou ***permissionless***. No *permissioned*, cada participante da rede possui uma identidade exclusiva e uma autorização, para acessar as informações das transações. De modo oposto, no *permissionless*[1], qualquer usuário pode entrar na rede e efetuar operações, desde que este cumpra com as políticas desta rede.

No *permissioned* é possível também definir políticas de acesso às informações pertinentes para cada participante, isto é, os participantes da rede podem visualizar apenas as informações pertinentes a ele, nada além disso. Enquanto no *permissionless*, os detalhes da transação pode ser limitado para fins de confidencialidade dos dados. Como o blockchain *permissioned* tem políticas de restrição de participantes na rede, é possível cumprir as regulamentações de proteção de dados com mais facilidade, caso este venha a ser o caso.

Em algumas bibliografias, pode ser encontrado também o conceito de blockchain híbrido. Neste caso, pode ser utilizado tanto o *permissioned* quanto o *permissionless*, como exemplo, para realizar algumas ações, tais como leitura de transações, pode ser definido como *permissionless*, enquanto para ver os detalhes, ou, realizar uma transação, o blockchain é definido como *permissioned*;

Smart Contract: é um acordo que fica armazenado e é executado parte da transação. Este acordo é responsável por administrar uma transação. A título de exemplo, um *Smart Contract* pode conter as condições do seguro de transporte de uma dada mercadoria, para que, quando a mercadoria for entregue, se esta sofreu danos durante o transporte, o *Smart Contract* possa acionar o seguro automaticamente;

Consensus: Quando se tem a certeza de que os participantes da rede são confiáveis e conhecidos, as transações podem ser verificadas através de um acordo, ou ainda, ***consensus***. Há variantes de mecanismos *consensus*, no entanto, eles incluem *Proof of Stake*, *Multi-signature* e *Practical Byzantine Fault Tolerance*.

[1] Algumas bibiliografias também designam este conceito por *public blockchain*;

Em linhas gerais, o *Proof of Stake*, ou, Prova de Participação é utilizado para validar as transações. Para isso, quem os valida (validadores) deve ser o detentor de uma determinada percentagem do valor total da rede. No que tange ao *Multi-signature*, este é um mecanismo democrático, que valida uma transação somente se a maioria dos validadores concordarem a respeito da validade da mesma. Por fim, o *Practical Byzantine Fault Tolerance* é um algoritmo cujo seu objetivo é resolver incoerências nas saídas de cada nó, isto é, funcionar adequadamente mesmo que haja participantes propagando informações incorretas para outros pares.

8.2 Propriedades da *Characteristic*

Um dos conceitos mais utilizados no desenvolvimento de aplicações utilizando BLE, é o conceito das características, estas, por sua vez, contém algumas propriedades que são abordadas a seguir:

Properties	Value	Description
Broadcast	0x01	Permite realizar o *Broadcast* das informações da *Characteristic*
Read	0x02	Permite ao *Client* realizar leitura dos valores da *Characteristic*
Write Without Response	0x04	Permite ao *Client* realizar escrita nos valores da *Characteristic* sem resposta de sucesso da operação
Write	0x08	Permite ao *Client* realizar escrita nos valores da *Characteristic*, com resposta de sucesso da operação
Notify	0x10	Permite ao *Server* notificar os valores da *Characteristic* para os *Clients* sem resposta de sucesso da operação
Indicate	0x20	Permite ao *Server* indicar os valores da *Characteristic* com resposta de sucesso da operação.
Authenticated Signed Writes	0x40	Permite ao *Client* utilizar segurança na escrita do valor da *Characteristic*
Extended Properties	0x80	Permite ao *Server* adicionar propriedades extras na *Characteristic*.

Tabela 8.1: Propriedades da *Characteristic*

8.3 Guidelines para Publicação de Aplicativos na Apple Store

De acordo com [49], as *guidelines* tem como propósito oferecer mais segurança para usuários e fornecer uma oportunidade para que os desenvolvedores tenham mais sucesso em suas aplicações. Ainda segundo [49], mesmo que uma aplicação tenha sido aprovada e disponibilizada, ela ainda pode ser removida da AS em algumas situações que não cumpram com as *guidelines*.

Os tópicos a serem considerados para a publicação de uma aplicação na AS são [49]:

Segurança para proteger usuários e seu dispositivo de certos danos físicos, conteúdos ofensivos ou perturbadores. A seguir têm-se dois exemplos objetivando elucidar alguns infortúnios:

1. aplicativo médico que realiza cálculos incorretos para dosagem de medicamentos podendo instruir o usuário à realizar a aplicação de uma dosagem perigosa podendo causar prejuízos na saúde do mesmo;
2. aplicativo médico que tem como objetivo realizar o diagnóstico dos usuários sem uma base sólida podendo eventualmente diagnosticar incorretamente um paciente em estado crítico.

Desempenho objetivando evitar experiências negativas quanto ao uso da aplicação. Nesta etapa é feita uma avaliação no tocante às falhas que possam ocorrer bem como eventuais bugs que não foram previamente testados;

Negócios relacionados com a monetização do aplicativo, como exemplo nas transações financeiras o valor cobrado não deve ser abusivo e a explicação de como será cobrado deve estar bem definida, evitando assim possíveis vendas enganosas. Ainda nesta secção são feitas eventuais análises para detetar ações fraudulentas no que concerne a qualquer tipo de *feedback* da comunidade, por exemplo, compra de avaliações positivas pode implicar na expulsão do programa de desenvolvedores da Apple.

Design deve ser trabalhado de forma com que o aplicativo seja um produto atrativo, simples e refinado, conforme os padrões de UX e GUI estabelecidos pela Apple. Outro fator relevante é fornecer atualizações que evitem a degradação do mesmo. A Apple presa pela aparência de seus produtos e serviços, para isso há esta secção que tem por objetivo fornecer *guidelines* para auxiliar no desenvolvimento de um produto que se enquadre nos padrões exigidos pela empresa;

Legalidade está diretamente relacionada com os requisitos a serem cumpridos perante à lei em qualquer lugar que o aplicativo for publicado. Em alguns casos, pode ser necessário uma consulta com um advogado para evitar infringir qualquer lei local. Um dos aspetos relevantes desta secção é a privacidade do usuário, que deve ser protegida com mecanismos de segurança adequado com a finalidade de evitar que os dados sejam divulgados sem a autorização dos proprietários.

Capítulo 9

Documentação Complementar da SmarTrack.IO

9.1 Detalhes Complementares da Implementação do SmarTrack Device

A Figura 9.1 apresenta a estrutura de implementação do SmarTrack Device através de um diagrama de pacotes.

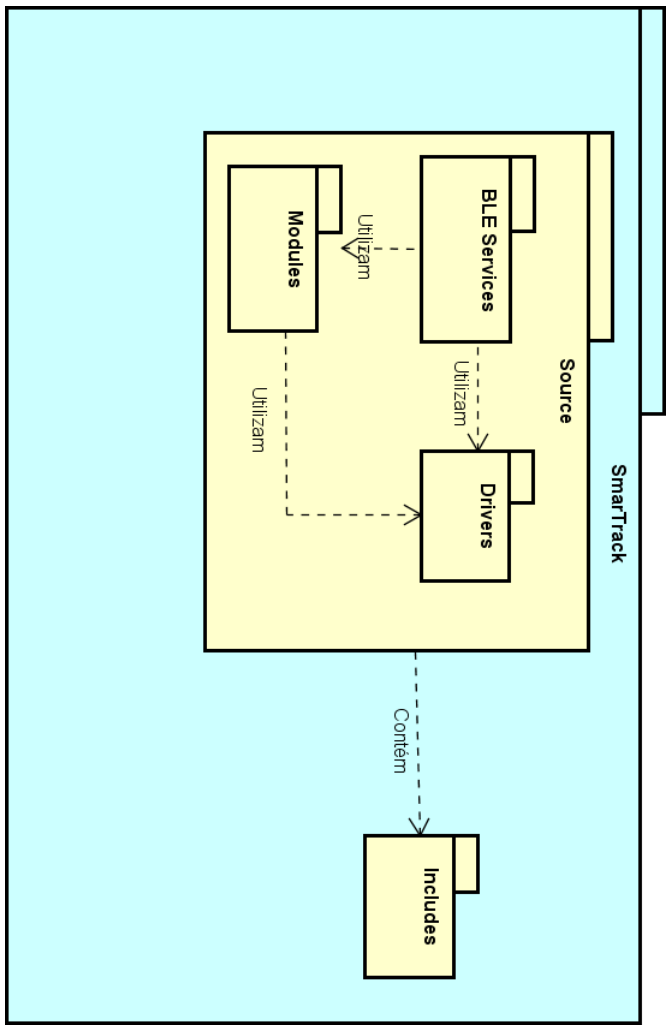


Figura 9.1: Diagrama de Pacotes - SmarTrack

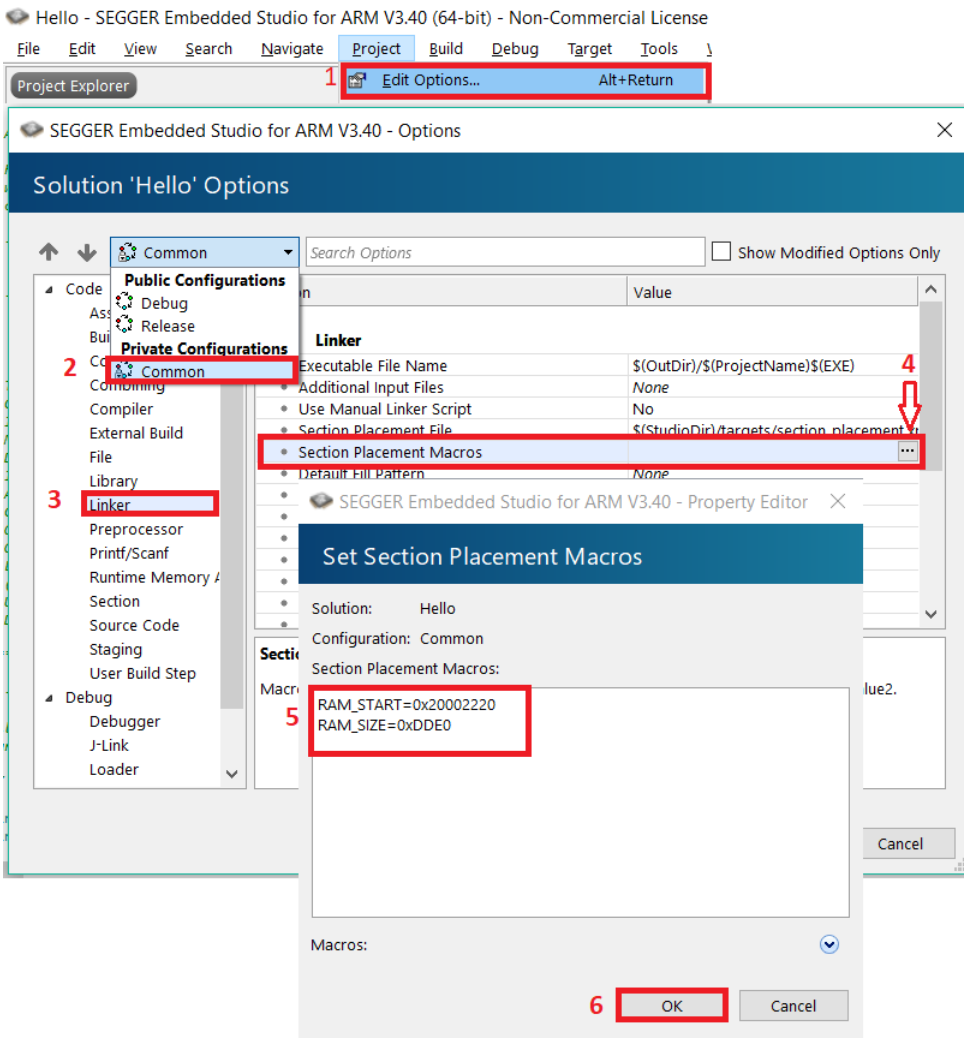


Figura 9.2: Ajuste de memória para a aplicação BLE

Ao desenvolver aplicações BLE, algumas IDEs podem solicitar ajustes no endereço de memória, para que a aplicação possa ser inserida na posição correta da pilha do BLE. Isso ocorre por alguns motivos, dentre ele está a adição de serviços à esta pilha, que implica no aumento de memória RAM bem como o tamanho do arquivo do código-fonte. Portanto, é necessário informar a posição de memória que a aplicação será inserida e qual o tamanho que ela irá ocupar. Para o presente trabalho, foi definido como posição inicial da memória RAM, o endereço 0x20002220 e o atributo RAM_SIZE como 0xDDE0. Os passos para realizar essa configuração na IDE Segger Embbeded Studio for ARM pode ser visto na Figura 9.2.

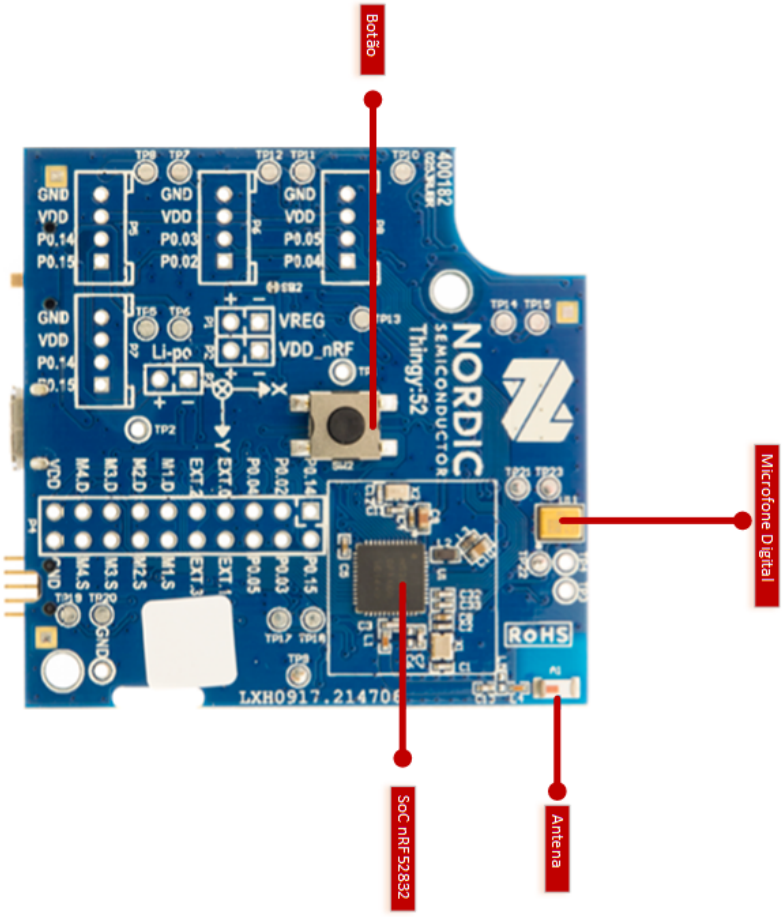


Figura 9.3: Parte superior Thingy52
Fonte: Adaptado de [50]

Para o desenvolvimento do SmarTrack Device, foram utilizados alguns componentes de *hardware*. As Figuras 9.3, 9.4, 9.5, mostram as arquiteturas destes componentes de *hardware*.

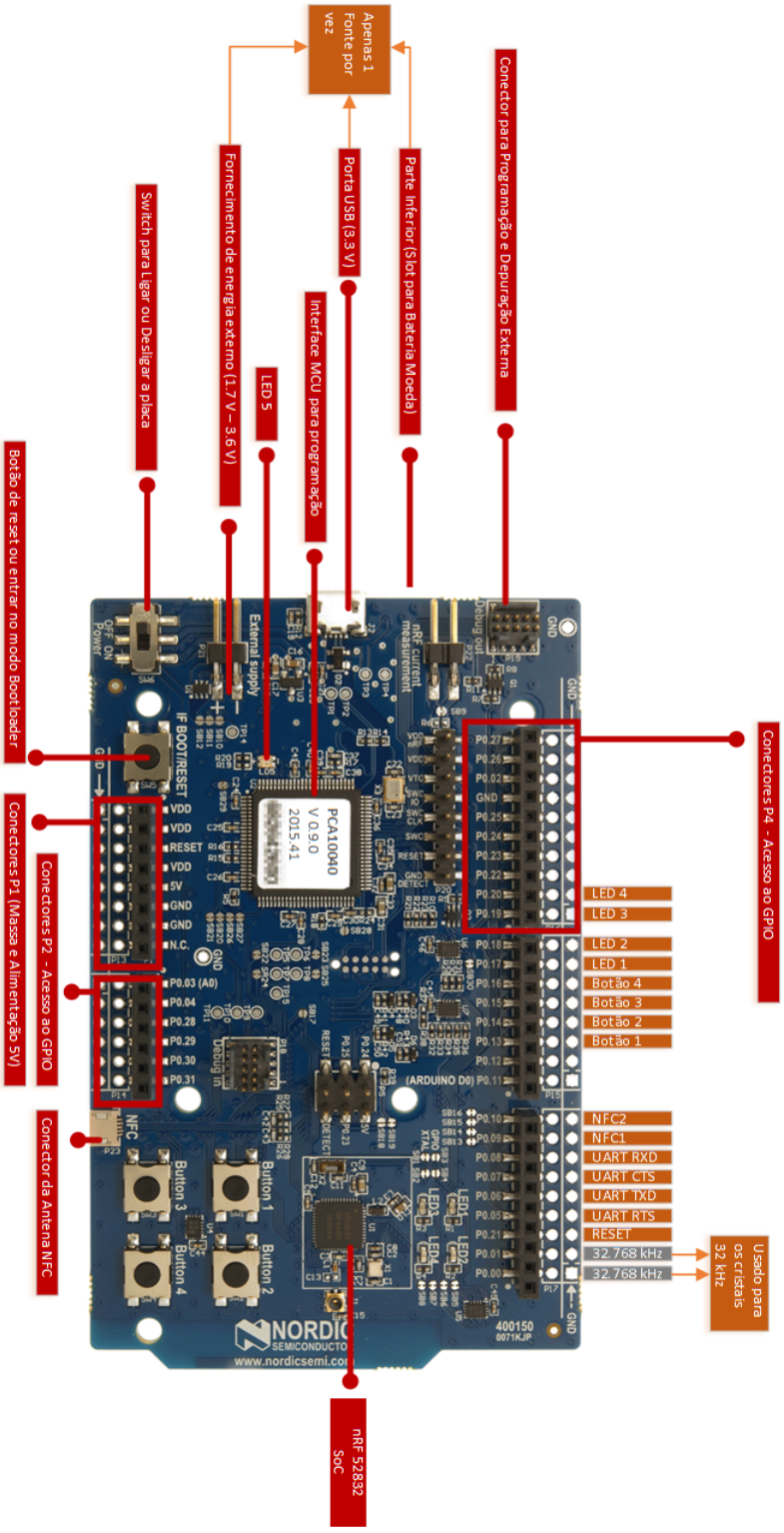


Figura 9.4: nRF52832 DK
Fonte: Autoria Própria

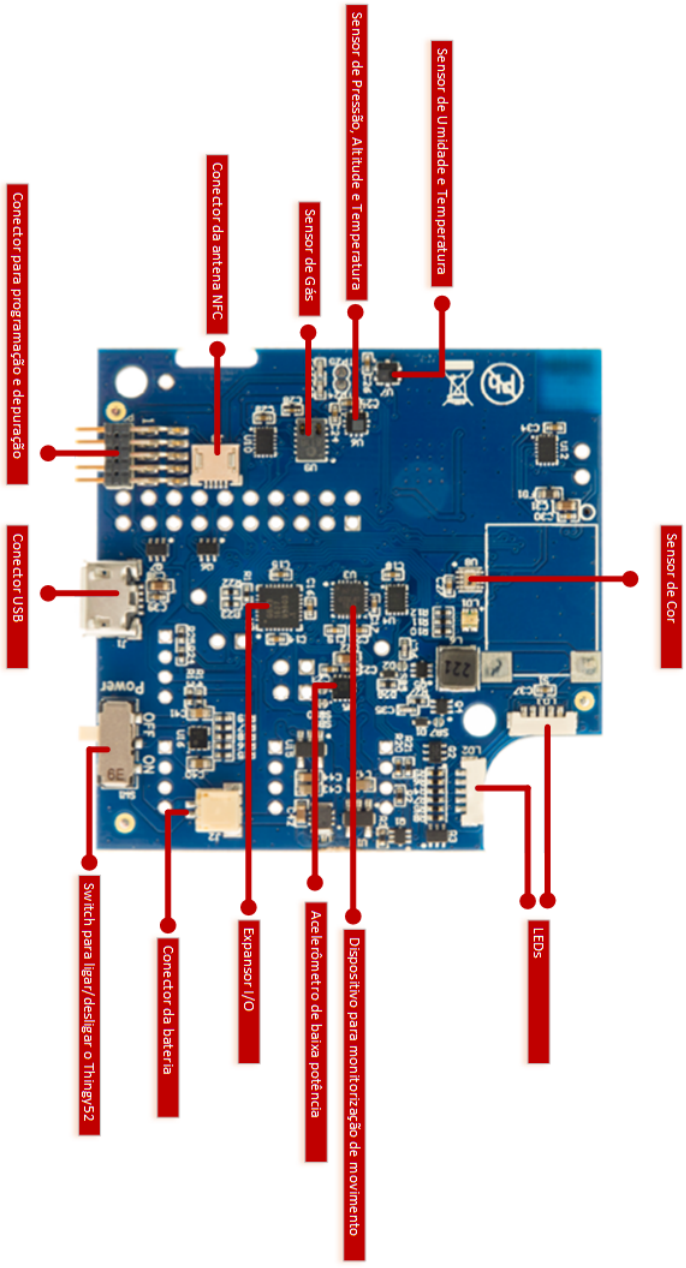


Figura 9.5: Parte inferior do Thingy52
Fonte: Adaptado de [50]

Quanto as características, a seguir são explicados os detalhes destas, dando início pelas características do *Configuration Service*:

Device Name : identifica o dispositivo através de um nome "amigável"[1];

Battery Level : indica o percentual restante da bateria, sendo útil para alertar o usuário da necessidade em substituir a bateria por uma nova;

Model Number : identifica o modelo do dispositivo, que é utilizado para diferenciar possíveis versões do SmarTrack Device;

Serial Number : especifica a versão fabricada, utilizada, por exemplo, para identificar os componentes utilizados em um determinado lote;

System Id : identificador único do SmarTrack Device, útil nos processos de validação dos dados;

Firmware Version : indica a versão do código-fonte que está em execução no SmarTrack Device.

Quanto ao *Readings Service*, este contém duas características, as quais são explicadas na sequência:

Readings List : lista de valores lidos pelos sensores durante o transporte, tais como temperatura, humidade, movimentação nos eixos, x, y e z, orientação - retrato ou paisagem, data e hora da leitura;

Synchronized Data : valor que indica se os dados recolhidos foram sincronizados com sucesso, isto é, quando o CS envia a confirmação para o aplicativo de que os dados recolhidos foram armazenados, após este momento, o aplicativo envia um valor de confirmação para remoção dos dados armazenados no SmarTrack Device.

Quanto ao *Transportation Service*, este contém as seguintes características:

[1]Evitando que este seja identificado por uma sequência de caracteres que não possuem uma semântica clara ao usuário final;

ProviderId*, *RecipientId*, *SenderId*, *TransportId : respetivos identificadores, prestador, remetente, destinatário e transporte;

Operation Mode : valor do modo de operação a ser utilizado pelo SmarTrack Device no transporte da mercadoria;

Min/Max Humidity : limites mínimos e máximos para deteção de violações das condições da humidade relativa durante o transporte;

Min/Max Temperature : limites mínimos e máximos para deteção de violações das condições da temperatura durante o transporte;

Min/Max X : limites mínimos e máximos para deteção de violações das condições de movimentação no eixo X, durante o transporte;

Min/Max Y : limites mínimos e máximos para deteção de violações das condições de movimentação no eixo Y, durante o transporte;

Min/Max Z : limites mínimos e máximos para deteção de violação das condições de movimentação no eixo Z, durante o transporte;

Orientation : orientação que a mercadoria deve se manter durante o transporte;

Reading Interval : intervalo de tempo para o SmarTrack Device realizar leituras dos dados dos sensores;

IsWriteBlocked : valor de bloqueio de alterações nos parâmetros de transporte, o qual, uma vez bloqueado, é desbloqueado apenas pelo destinatário.

Na sequência, é mostrado os detalhes de implementação das características, isto é, sua semântica para o SmarTrack Device, bem como o tipo de variável que as características suportam:

SynchronizedData : booleano, sendo 0 como não sincronizado, 1 sincronizado;

Device Name*, *Model Number*, *Serial Number*, *System Id : são características com seus tamanhos variados, em que, cada byte representa um valor Hexadecimal relacionado com o seu respetivo elemento na tabela ASCII, por exemplo, a palavra SMART, é representada por 0x53(S), 0x4D(M), 0x41(A), 0x52(R), 0x54(T);

***ProviderId*, *RecipientId*,*SenderId*, *TransportId* :** funciona de forma similar as características *Device Name*, *Model Number*, *Serial Number*, *System Id*;

Battery Level : um byte que representa um inteiro em um intervalo de 0 a 100;

Firmware Version : é uma estrutura de versionamento da *firmware*, a qual contém 1 byte para representar a versão MAJOR[2], 1 byte para representar a versão MINOR[3] ,e a versão do PATCH[4], assumindo uma estrutura similar a, versão 1.2.3;

Operation Mode : 1 byte que pode assumir os valores, 0x00 - *Threshold Mode*, 0x01 - *Interval Mode*, ou 0x02 - *Dual Mode*;

Min/Max Humidity : 1 byte para cada limite, o qual é um valor inteiro dentro de um intervalo de 0 a 99, e deve ser interpretado como um percentual, isto é, 0 a 99%;

Min/Max Temperature : 2 bytes para cada limite, sendo que 1 byte representa a parte inteira da temperatura, podendo ser positiva ou negativa, enquanto o outro byte, representa a parte decimal, mas ambos são dois valores inteiros. Por exemplo, o primeiro byte assume o valor 21, e o segundo byte 79, portanto, a temperatura será $21.79^{o}C$;

[2]Quando há mudanças na API
[3]Implementação de novas funcionalidades mantendo a compatibilidade.
[4]Correção de bugs que mantém a compatibilidade com versões anteriores.

Min/Max X, Y, e Z : 4 bytes para cada limite, sendo representado por uma variável *float*;

Orientation : 1 byte que representa um valor que pode ser 0x00 - Retrato, ou 0x01 - Paisagem;

Reading Interval : 4 bytes que representam um intervalo de leitura em milissegundos;

IsWriteBlocked : 1 byte, somente de leitura, que é iniciado em 0x00 - Permite Parametrização, e no momento em que o remetente confirma a contratualização do serviço (4.2.2), o valor é alterado para 0x01 - Bloqueia Parametrização.

9.2 Detalhes Complementares da Implementação do *Cloud Service*

A Figura 9.1 apresenta a estrutura de implementação do *Cloud Service* através de um diagrama de pacotes.

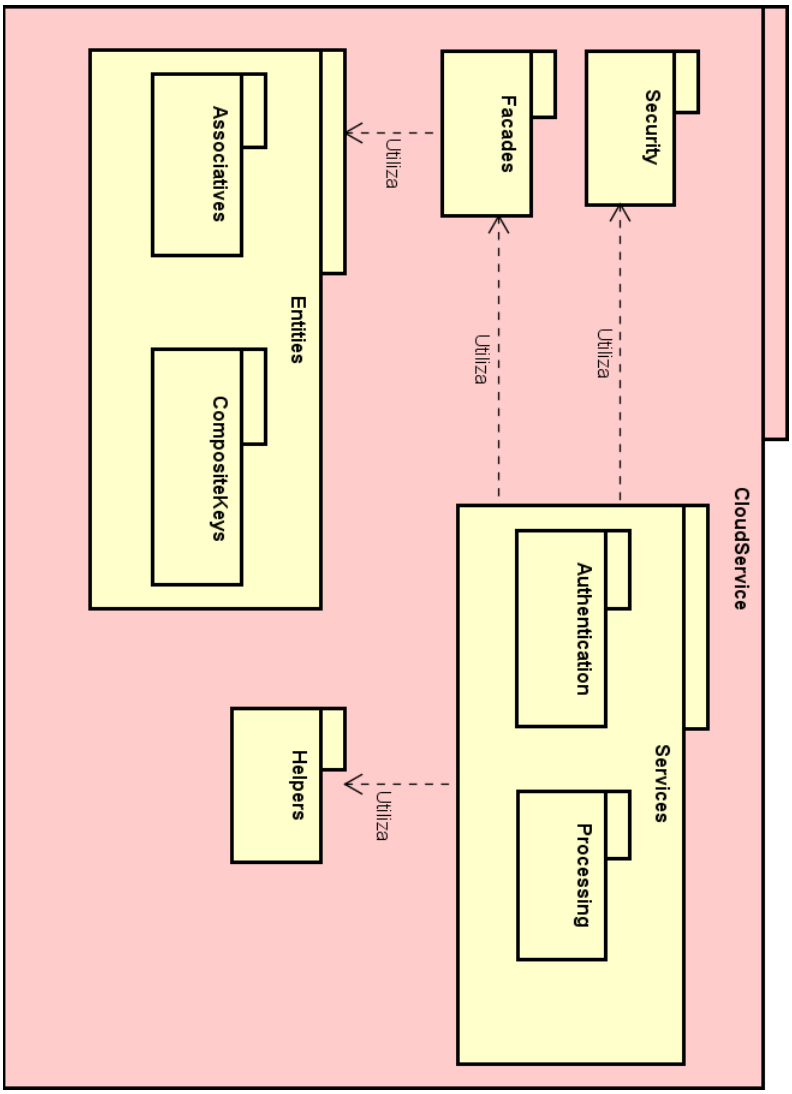


Figura 9.6: Diagrama de Pacotes - Cloud Service

A lista a seguir, mostra quais são as URIs disponíveis, assim como suas respetivas funcionalidades:

auth/get/email : verifica se existe um endereço de e-mail já registado na base de dados;

auth/register : regista um usuário, a partir dos dados recebidos no método.

auth/resendcode : cria, e envia uma nova OTP para o endereço de e-mail recebido no método;

auth/login : reinicia o processo de autenticação do usuário, verificando os dados recebidos no método - este método é utilizado somente aquando da deteção de tentativas de autenticação caracterizadas como suspeita;

auth/validate/code : faz a validação da OTP enviada pelo usuário, caso a OTP seja correta, é criado um *cs token* e localização atual do usuário é adicionada à lista de locais confiáveis;

auth/get/applicationid : método utilizado no primeiro passo de verificação de autenticidade do usuário, o qual verifica se código identificador do dispositivo que o usuário está autenticando, existe na lista de dispositivos confiáveis do usuário - método utilizado na primeira etapa de autenticação;

auth/code/app : verifica se o usuário tem uma OTP pendente - método utilizado na segunda etapa de autenticação;

auth/get/trustedcities : retorna uma lista com as cidades confiáveis do usuário - método utilizado na terceira etapa de autenticação;

business/get/profiles : retorna uma lista com os perfis existentes no servidor - remetente, prestador, operador, etc;

business/get/allowedpages : retorna uma lista das GUIs, das quais o usuário pode aceder;

transportation/get/readings : retorna uma lista com o histórico dos dados do recolhidos durante o transporte da mercadoria;

transportation/all: retorna uma lista com os dados dos transportes;

data/insert/conditions : armazena as condições de transporte da mercadoria, definidas pelo prestador de serviço, aquando da contratualização do serviço;

data/insert/cargo : armazena os dados de uma mercadoria, como nome e o tipo - sensível a quedas, alimento perecível, produto eletrónico, etc.;

data/insert/transport : armazena os dados do transporte, aquando da contratualização do serviço. Dados como, endereço de e-mail do remetente e do destinatário, e mercadoria a ser transportada.

9.3 Detalhes Complementares da Implementação do Aplicativo

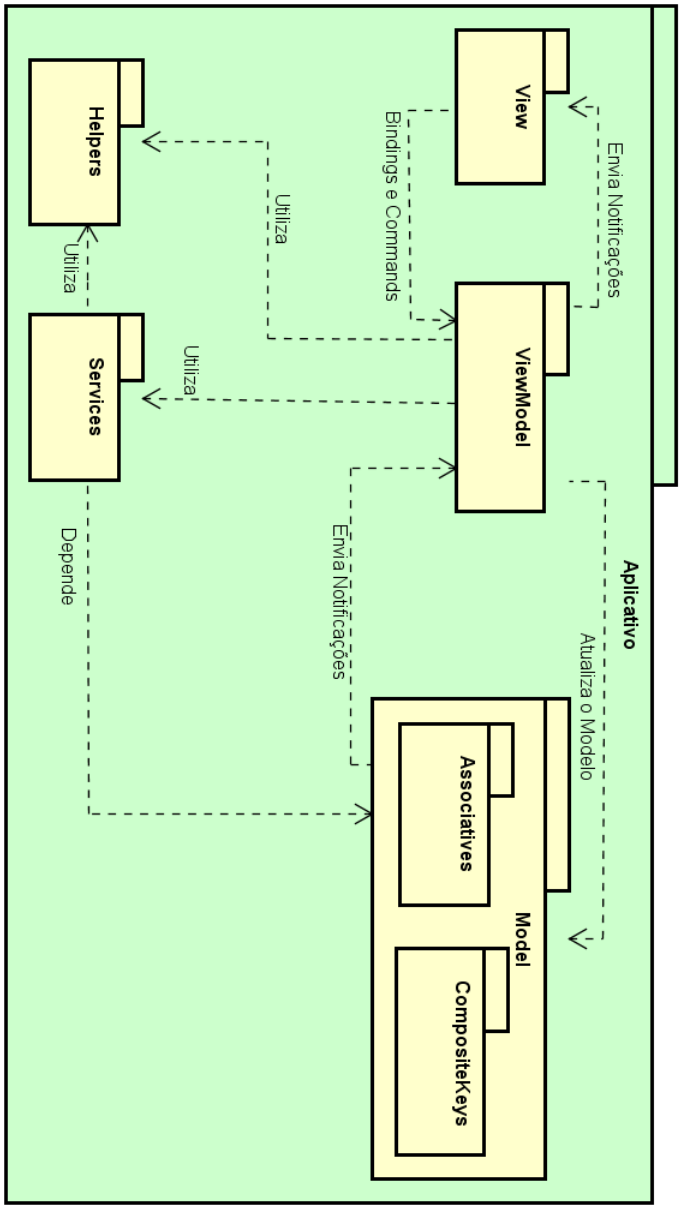


Figura 9.7: Diagrama de Pacotes - Aplicativo

9.4 Códigos relacionados ao SmarTrack Device

Nesta secção serão mostrados alguns trechos de códigos-fonte utilizados no desenvolvimento desta dissertação. O primeiro código apresentado, está relacionado com a validação do *smt token*, o qual é uma adaptação para a LP C, do algoritmo para validação do Cadastro de Pessoa Física (CPF) - documento brasileiro, desenvolvido por Ronaldo Faria Lima.

```
#define TOKEN_SIZE 11
tokenIsValid(const char * token){
  int ckDigit, calculatedCkDigit;
  if (token == NULL) {return 0;}
  if (strlen(token) != TOKEN_SIZE) {return 0;}
  if (hasAlpha(token)) {return 0;}
  if (isSingleDigit(token)) {return 0;}
  ckDigit = getCheckDigit(token);
  calculatedCkDigit = checkDigit(token, 10) * 10 + checkDigit(token, 11);
  return (ckDigit == calculatedCkDigit);
}
```

Este algoritmo verifica se o *smt token* é nulo, ou o tamanho é diferente de 11 dígitos, ou contém letras, e se são dígitos repetidos (11111111...). Após estas validações iniciais, são obtidos e calculados os dígitos verificadores para uma validação final. Além do algoritmo implementado, também foram encontradas alternativas, tais como a fórmula **ISO 7064 Mod 97,10**[5], e o algoritmo de *Verhoeff*[6].

Como retorno da validação do *smt token*, obteve-se o seguinte resultado, onde 1 é válido, e 0 é inválido:

```
int x = tokenIsValid("79932461059");
```

[5]https://usersite.datalab.eu/printclass.aspx?type=wiki&id=91772
[6]https://www.programmingalgorithms.com/algorithm/verhoeff-algorithm?lang=C

```
resultado: 1
int y = tokenIsValid("11111111111");
resultado: 0
```

Devido ao uso de funções específicas das bibliotecas do BLE, optou-se por apresentar um pseudocódigo do funcionamento da função para permitir ou restringir alterações nas características responsáveis por parametrizar as condições de transporte.

```
autorizacao_requisitada(parametros_omitidos){
        bool autorizado;
 obter_valor_escrita_bloqueada();
        se(escrita_bloqueada == 1){
            autorizado = falso;
        }senao{
            autorizado = verdadeiro;
            permitir_escrita();
        }
}
```

A função **autorizacao_requisitada(...)** é invocada sempre que o SmarTrack Device receber uma solicitação de escrita nas características do *Transportation Service*.

Quanto a função **obter_valor_escrita_bloqueada()**, esta é responsável consultar o valor da característica *is_write_blocked* antes de realizar qualquer operação de escrita. A alteração do valor da característica *is_write_blocked* ocorre no momento em o SmarTrack Device recebe uma solicitação de escrita na característica *Recipient ID*, como pode ser observado no pseudocódigo a seguir:

```
escrita(parametros_omitidos){
se(caracteristica == identificador_do_destinatario){
        bool valor_valido = validar_valor_escrito();
        se(valor_valido){
```

```
        alterar_valor_escrita_bloqueada();
        }
    }
}
```

Algumas características, tendem a ser mais complexas, pois envolvem uma estrutura que compreende em si, outras estruturas. Como no caso estrutura dos dados a serem lidos pelos sensores, a qual pode ser observada a seguir:

```
struct{
    int8_t  integer; //Parte inteira da temperatura - 29 de "29.77"
    uint8_t decimal; //Parte decimal da temperatura - 77 de "29.77"
} ble_srs_temperature_t;        //2 Bytes
uint8_t ble_srs_humidity_t;     //1 Byte
struct{
    float x;
    float y;
    float z;
} ble_srs_gravity_t;            //12 Bytes
uint8_t ble_srs_orientation_t;  //1 Byte
union{
    uint32_t                current_time;
    ble_srs_temperature_t   temp_value;
    ble_srs_humidity_t      humidity_value;
    ble_srs_gravity_t       gravity_value;
    ble_srs_orientation_t   orientation_value;
}) ble_srs_readings_t; //16 Bytes
```

9.5 Códigos relacionados ao *Cloud Service*

Nesta secção são mostrados alguns códigos em SQL, e seus respetivos resultados. O código a seguir, tem como objetivo, consultar as informações de transporte armazenadas.

```
//Código SQL
SELECT dt.read_date 'Data e Hora', c.name 'Cidade', r.name 'Região',
dt.smartrack_id 'SmarTrack', ap.id 'Dispositivo', ap.user_email 'Email do
Operador', u.name 'Nome do Operador', s.`type` 'Sensor', a.name 'Medida', dt.value
FROM as_readings_data dt
inner join sensor s on s.id = dt.sensor_id
inner join attribute a on a.id = dt.attribute_id
inner join transport t on t.id = dt.transport_id
inner join applicationid ap on ap.id = dt.user_email
inner join user u on u.email = ap.user_email
inner join city c on c.id = dt.read_location
inner join region r on r.id = c.region_id
//Resultado em formato JSON
{"table": "as_readings_data",
    "leituras":[{
       "Data e Hora": "2018-10-21 20:42:19",
       "Cidade": "Bragança",
       "Região": "Braganca",
       "Transporte": 1,
       "Mercadoria": "Escultura de Gelo Monalisa",
       "SmarTrack": "XDA5325X-5113S",
       "Dispositivo": "DUNFVG4POVIJSKMF",
       "Email do Operador": "leonan@alunos.utfpr.edu.br",
       "Nome do Operador": "leonanan ",
       "Sensor": "Temperature",
```

```
        "Medida": "Room Temperature",
        "Valor": 13.72000}]}
```

A seguir, é mostrado o código e resultado de uma consulta escrita em SQL, cujo objetivo é consultar as informações , tais quais: perfis de usuários e permissões de acesso a página de registo.

```
//Codigo SQL
select p.name, r.description, rp.access from as_role_pages rp
inner join role r on r.id = rp.role_id
inner join applicationpage p on p.id = rp.page_id
where p.name = 'LoginPage'

//Resultado da Consulta
{"table": "as_role_pages",
"permissoes":[{
        "name": "LoginPage",
        "description": "Sender",
        "access": 1},
        {
        "name": "LoginPage",
        "description": "Recipient",
        "access": 1},
        {
        "name": "LoginPage",
        "description": "Provider",
        "access": 1},
        {
        "name": "LoginPage",
        "description": "Operator",
```

```
        "access": 1}]}
```

No próximo código, é possível notar a escalabilidade do projeto, haja vista que ao inserir apenas alguns dados, foi possível substituir um servidor de armazenamento de arquivos, para suportar armazenamento de imagens.

```
//Codigo SQL para o novo Sensor : Câmara
select type, name from as_sensor_attributes sa
inner join sensor s on s.id = sa.sensor_id
inner join attribute a on a.id = sa.attribute_id
where type = 'Camera'
//Resultado da Consulta
{"table": "as_sensor_attributes",
"sensores":[{
"type": "Camera",
"name": "Picture"
},
{
"type": "Camera",
"name": "Video"}]
}

//Resultado de uma Consulta na Tabela as_readings_data
    {"table": "as_readings_data",
"rows":[
{
"read_date": "2018-08-22 14:34:25",
"read_location": 1995164,
"smartrack_id": "XDA5325X-5113S",
"sensor_id": 4,
```

```
"user_email": "DUNFVG4POVIJSKMF",
"attribute_id": 9,
"transport_id": 1,
"value": 0.00000,
"image": "0x1234"}]}
```

9.6 Códigos Relacionados com os Aplicativos

No aplicativo, para enviar requisições para o WS, é necessário que seja enviado o *cs token* para validação da requisição. Para isso, implementou-se códigos para este tipo de solicitação, conforme mostra o algoritmo abaixo.

```
GetWithSecurityAsync(dynamic id, string resource, string authValue){
    HttpClient client = ConfigureHeader(authValue, address);
    try{
        var json = JsonConvert.SerializeObject(id);
        var content = new StringContent(
                  json, Encoding.UTF8, "application/json");
        var url = Constants.CONN.COMPLETEADDRESS + resource;
        HttpResponseMessage response = await client.PostAsync(url, content);
        if (response is null)
            throw new HttpRequestException();
        var resultString = await response.Content.ReadAsStringAsync();
        return JsonConvert.DeserializeObject<T>(resultString);
    }
    catch (Exception ex){
        InformarErro(ex);
    }
}
```

O parâmetro enviado para autorização (authvalue), é armazenado utilizando a classe *SecureStorage*[7], a qual fornece métodos para armazenar e consultar dados de forma segura[8], e também é utilizada para armazenar informações sensíveis dos usuários.

[7]https://docs.microsoft.com/en-us/dotnet/api/xamarin.essentials.securestorage?view=xamarin-essentials-android

[8]Utiliza as ferramentas, Android KeyStore e KeyChain para armazenar uma chave de criptografia usada para encriptar o valor antes de ser armazenado na memória local.