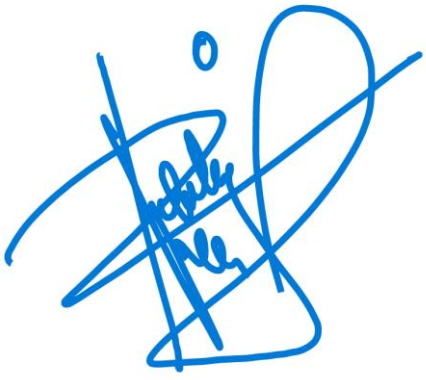
DANILO BASTOS HERNANDES
GIOVANNI ARAUJO CAVALCANTE
GUSTAVO SHUITI SAWADA THEODORO FERREIRA

RASTREAMENTO DE PACOTES EM TEMPO REAL



São Paulo
2018

DANILO BASTOS HERNANDES
GIOVANNI ARAUJO CAVALCANTE
GUSTAVO SHUITI SAWADA THEODORO FERREIRA

RASTREAMENTO DE PACOTES EM TEMPO REAL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro Elétrico com Ênfase em Computação

São Paulo
2018

DANILO BASTOS HERNANDES
GIOVANNI ARAUJO CAVALCANTE
GUSTAVO SHUITI SAWADA THEODORO FERREIRA

RASTREAMENTO EM TEMPO REAL DE PACOTES

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção do título de Engenheiro Elétrico com Ênfase em Computação

Área de Concentração:
Sistemas Digitais

Orientador:
Prof. Dr. Reginaldo Arakaki
Co-orientador:
Prof. Dr. José Kleber da Cunha Pinto

São Paulo
2018

Catalogação-na-publicação

Hernandes, Danilo Bastos Hernandes
Rastreamento de pacotes em tempo real / D. B. H. Hernandes, G. A. C. Cavalcante, G. S. S. T. F. Ferreira -- São Paulo, 2018.
52 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Rastreamento 2.Internet das Coisas 3.Sistemas Embarcados I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Cavalcante, Giovanni Araujo Cavalcante IV.Ferreira, Gustavo Shuiti Sawada Theodoro Ferreira

AGRADECIMENTOS

Ao Professor Reginaldo Arakaki e ao Professor José Kleber da Cunha Pinto, pela orientação, e o constante auxílio ao longo do projeto.

Aos amigos Diego Giliolli Soler dos Santos, Jeferson Ribeiro de Souza, Marina Scott do Val e a todos que colaboraram direta ou indiretamente na execução deste trabalho.

RESUMO

O objetivo do projeto foi o desenvolvimento de um sistema de rastreamento em tempo real de cargas em transporte através da utilização de dispositivos IoT(*Internet of Things*). A partir de uma análise dos sistemas de rastreamento que são utilizados hoje em dia, elaboramos uma arquitetura para um sistema dedicado a coletar dados em tempo real de um pacote em transporte, armazená-los, e em seguida enviá-los ao destinatário e a empresa responsável. Para elaborarmos a arquitetura foi realizado o levantamento dos requisitos funcionais e não-funcionais, o que serviu para direcionar a especificação do sistema e a implementação de uma montagem de testes. A arquitetura consiste em um sistema embarcado dentro do veículo utilizar *tags* RFID para informar o servidor sobre quais pacotes se encontram lá e a localização do veículo por meio de uma antena GPS, o servidor por sua vez armazena esses dados e os envia para os usuários quando esses utilizam um aplicativo para verificar o estado do pacote. Por meio dessa implementação buscamos verificar quanto um sistema de rastreamento que utiliza de tecnologias mais recentes consegue contribuir para um acompanhamento mais detalhado do transporte de cargas.

Palavras-chave: *Internet of Things*. Rastreamento de cargas. Tempo real.

ABSTRACT

The objective of this project is the development of a real time tracking system for packages in transport through the use of IoT (Internet of Things) devices. Through an analysis of the current tracking systems used by companies, we elaborated an architecture of a system dedicated to collecting data of the state of a package during transport in real time, storing this data, and later sending it to the recipient and the company responsible for the transport. To create this architecture, we listed the functional and non-functional requirements in order to aid in the specification of the system to be implemented and tested. The architecture consists of an embedded system inside the vehicle utilizing the RFID tags present in the packages to recognize what is inside, combining this information with the current location of the vehicle, which is obtained through a GPS antenna, and sending this group of data to the server. The server, in turn, stores this data to later send it to the users checking the current state of the package in an app. Through this implementation we strived to analyze how the utilization of modern IoT devices would contribute to a more detailed tracking of goods in transport.

Key-words: Internet of Things. Package tracking. Real time.

LISTA DE FIGURAS

Figura 1 – Esquema ilustrativo da arquitetura do projeto 18
Figura 2 - Diagrama de classes para dispositivos *mobile* 20
Figura 3 - Diagrama de classes para Web e API 21
Figura 4 - Diagrama de sequência do rastreamento de pacotes 23
Figura 5 - Diagrama de sequência do acesso à informações pela transportadora 24
Figura 6 - Diagrama de sequência do acesso à informações pelo destinatário 25
Figura 7 – Esquema ilustrativo da arquitetura implementada 31
Figura 8 - Tela de detalhes do pacote do aplicativo *Android* 32
Figura 9 - Leitoras RFID *Mfrc522 Mifare* com *tag* 34
Figura 10 - Leitora RFID *Acura Global Edge-50* 35
Figura 11 - Módulo SIM5230E para Arduino 36
Figura 12 - Placa *Arduino Uno R3* 37
Figura 13 - Placa *Raspberry Pi 3 Model B* 37
Figura 14 - Diagrama de classes para Web e API implementados 38
Figura 15 - Diagrama de classes para aplicativo Android 39
Figura 16 - Esquema ilustrativo da primeira montagem para testes 41
Figura 17 - Esquema ilustrativo da segunda montagem para testes 42
Figura 18 - Inversor/transformador utilizado para alimentação do hardware no carro 44
Figura 19 - Montagem para testes; da esquerda para a direita: leitora RFID e antena, *Arduino* com módulo SIM5230E e antenas, *Raspberry Pi* 44
Figura 20 - Trajeto amostrado durante os testes da segunda montagem 45
Figura 21 - Trajeto real percorrido durante os testes da segunda montagem 45
Figura 22 - Informações do pacote e trajeto amostrado pelos testes da segunda montagem no aplicativo Android 46
Figura 23 - Tabela (SQL) de ocorrências, vista através do MySQL Workbench 46
Figura 24 - Tabela (SQL) de amostras de localização, vista através do MySQL Workbench. 47

LISTA DE ABREVIATURAS

API *Application Program Interface*
IoT *Internet of Things*
RFID *Radio Frequency Identification*
SO Sistema Operacional
UHF *Ultra High Frequency*
GPS *Global Positioning System*
AWS *Amazon Web Services*
EC2 *Elastic Compute Cloud*
RDS *Relational Database Service*
SQL *Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO 8
1.1 Motivação 8
1.2 Justificativa 8
1.3 Objetivos 9
1.4 Metodologia do Projeto 9
1.5 Organização do trabalho 11
2 ASPECTOS CONCEITUAIS 12
2.1 Processo Logístico 12
2.2 Identificação e Rastreabilidade 13
2.3 Sistemas Embarcados 13
2.4 Internet das Coisas 14
2.5 Sistema Operacional 14
2.6 Computação em Nuvem 15
2.7 Arquitetura Distribuída 16
3 ESPECIFICAÇÃO DO SISTEMA 17
3.1 Negócio 17
3.1.1 Serviço de Rastreamento 17
3.1.2 Serviço de Controle de Logística 17
3.1.3 Serviço de Acompanhamento de Pacotes 18
3.1.4 Integração e Visão Geral 18
3.1.5 Arquitetura de Classes 19
3.2 Requisitos do Sistema 22
3.2.1 Requisitos Funcionais 22
3.2.1.1 Rastreamento 22
3.2.1.2 Transportadora 23
3.2.1.3 Destinatário 24
3.2.2 Requisitos Não-Funcionais 25
3.2.2.1 Rastreamento 25
3.2.2.2 Transportadora 26
3.2.2.3 Destinatário 27
3.3 Mecanismos de Engenharia 27
3.3.1 Servidor na Nuvem 27
3.3.2 Identificação por RFID 29
3.3.3 Sistema Embarcado 29
4 O SISTEMA IMPLEMENTADO 31
4.1 Descrição Arquitetural 31
4.1.1 Dispositivos Mobile - Android (Java) 32
4.1.2 API e Aplicação Web - .NET 32
4.1.3 Servidor em Nuvem - Amazon EC2 33
4.1.4 BD em Nuvem - Amazon RDS 33
4.1.5 Leitora RFID 1 - Mfrc522 Mifare 33
4.1.6 Leitora RFID 2 - Acura Global Edge-50 34
4.1.7 Sistema Embarcado - Arduino + Raspberry Pi 35
4.2 Arquitetura de Classes 38
5 TESTES E RESULTADOS 40

5.1 Primeira Montagem de testes 40
5.2 Segunda Montagem de testes 42
5.3 Resultados.....44
6 CONSIDERAÇÕES FINAIS.....48
6.1 Conclusões do Projeto de Formatura.....48
6.2 Comentários dos Integrantes 49
6.3 Perspectivas de Continuidade.....50
LISTA DE REFERÊNCIAS.....51

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

No início de 2018, o transporte de mercadorias pelas rodovias correspondeu a 61,1% do total de mercadorias transportadas no Brasil (CONFEDERAÇÃO NACIONAL DO TRANSPORTE, 2018), e 38% das compras no Brasil são realizadas pela internet (ECOMMERCE FOUNDATION, 2018). Interpretando esses dados, vemos que hoje em dia boa parte dos consumidores prefere o comércio pela internet e que eles dependem em grande parte do transporte rodoviário para adquirir seus produtos. Mas, além disso, o número de pessoas que interagem com o processo de transporte por meio de rastreadores de seus pedidos é muito elevado, já que o comércio eletrônico os utiliza em grande escala. Agora, se os sistemas atuais de logística de transporte forem analisados, observa-se que em sua grande maioria atualizam o estado da carga apenas quando essa chega em determinados pontos do trajeto a ser percorrido, normalmente centros de distribuição e galpões. Tal sistema é na realidade uma caixa preta entre qualquer um dos pontos tanto para os envolvidos no sistema logístico quanto para os destinatários da carga, possibilitando que algum indivíduo extravie a carga entre dois pontos do trajeto sem que seja possível saber o instante em que isso ocorreu, ou que tal informação não esteja disponível de maneira fácil.

Levando em conta a proliferação de dispositivos *IoT* e a grande precisão e disponibilidade de serviços de geolocalização, utilizar essas tecnologias mais recentes no processo de transporte de carga pode levar à solução de problemas antigos nessa indústria - realizando o rastreamento em tempo real da carga, desde o momento em que é empacotada até quando for entregue ao destinatário. Desse modo, a prevenção e detecção de atrasos, extravios e fraudes torna-se mais fácil, dando maior segurança à transportadora e ao destinatário. Além disso, o oferecimento de informações mais detalhadas gera uma melhor experiência de uso do serviço para o destinatário.

1.2 JUSTIFICATIVA

A possibilidade de se acompanhar um pacote em tempo real permite com que as empresas de logística consigam detectar um extravio no momento em que ele ocorre, tornando a apuração do que ocorreu mais fácil, possivelmente reduzindo os prejuízos decorrentes do desvio de carga. Além disso, com uma análise dos dados provenientes do rastreamento de todos

os pacotes em transporte, a empresa poderia averiguar se alguma rota utilizada está gerando um número descomunal de perdas de pacotes.

Com uma coleta mais constante de dados dos pacotes, os destinatários poderão visualizar de forma mais clara onde o pacote se encontra no trajeto, dando mais transparência ao processo de transporte do ponto de vista do cliente. Além disso, essa maior disponibilidade de dados sobre a movimentação do pacote pode ser utilizada em disputas jurídicas que envolvem o transporte de algum volume.

1.3 OBJETIVOS

O objetivo deste trabalho é projetar um sistema que acompanha o trânsito de pacotes e seus registros nas agências em tempo real, buscando manter o estado salvo no sistema mais próximo do estado real do pacote, além de criar uma interface que permita que os usuários visualizem esses dados.

1.4 METODOLOGIA DO PROJETO

O projeto foi originado a partir da observação das interfaces que agências de transporte oferecem ao destinatário para obter informações a respeito do pacote ou carga em questão; foi notado também que há um baixo nível de automação em serviços de transporte em geral no Brasil, o que se mostra um sério problema, considerando o tamanho e a importância desse setor. A partir desse panorama, decidiu-se fazer um projeto que busca corrigir os problemas encontrados, tanto para o destinatário e remetente, que desejam obter mais informações acerca da carga em trânsito, quanto para as transportadoras, que poderão realizar o serviço de forma mais eficiente e com acesso à mais dados que reflitam o estado atual das cargas em trânsito.

Portanto, foram levantados os pontos centrais a serem tratados de forma ao sistema atender aquilo que foi pensado:

- Um modo de identificar unicamente cada pacote registrado no sistema;
- Um modo de detectar ***em tempo real*** tais pacotes dentro de um contêiner/veículo e agências;
- Um modo de receber, armazenar e acessar as informações coletadas;

No início, o projeto foi desenvolvido dentro da disciplina de laboratório de Laboratório de Software 2, aonde realizamos as pesquisas iniciais e o levantamento inicial dos requisitos do projeto, além da elaboração da maquete de prova de conceito que serviu como a primeira montagem para testes. Com esses desafios em mente uma busca por soluções foi realizada, levantando as características de cada uma de forma que a escolhida fosse possível de ser implementada e testada fisicamente, levando em conta as limitações de dinheiro, tempo e pessoas que o projeto possui, além de ser interessante para os *stakeholders* de uma aplicação real do sistema proposto. Por meio dessas limitações na solução levantamos os requisitos funcionais e não-funcionais do sistema de forma a fundamentar a escolha da decisão em métricas que refletissem o cenário real de aplicação.

Após a escolha da solução iniciou-se a definição da solução técnica a ser implementada junto com uma pesquisa das tecnologias a serem utilizadas, tais pesquisas foram realizadas com o apoio dos Professores Reginaldo Arakaki e José Kléber Da Cunha Pinto, o orientador e co-orientador deste projeto, respectivamente. Por meio dessa pesquisa escolhemos as tecnologias que atendiam aos requisitos levantados anteriormente, portanto a arquitetura passou por diversas versões até chegar em uma que além de atender os requisitos levantados, pudesse ser testada e apresentada. Nessa etapa também foi elaborada uma maquete de modo a simular as funções críticas do sistema final em pequena escala, desse modo quando a implementação de fato começasse teríamos como base esse protótipo.

Em seguida iniciou-se a implementação baseada na maquete elaborada anteriormente. Conforme a implementação foi se desenvolvendo, o modelo pensado anteriormente foi revisto de acordo com as mudanças que se revelaram necessárias após o início da implementação, além disso essas mudanças sempre ocorreram de forma a manter o sistema de acordo com os requisitos estabelecidos. Concomitantemente com a implementação, a elaboração da versão final da monografia se iniciou. Seguindo as orientações do orientador deste projeto, o documento que havia sido elaborado continuamente ao longo das apresentações parciais da disciplina de acompanhamento do projeto de formatura foi reestruturado. Conforme a implementação e a monografia foram se finalizando, realizamos algumas revisões e ajustes para certificar que tudo estava correto e consistente.

1.5 ORGANIZAÇÃO DO TRABALHO

No decorrer desta monografia será discutida a implementação dada ao projeto até o momento. Serão abordados os conceitos e as ideias em volta do projeto, suas motivações, usos e possibilidades, será feita uma análise das tecnologias escolhidas e utilizadas na implementação do protótipo do projeto e suas justificativas. Serão especulados os requisitos do sistema, definindo-os realisticamente, e será explicada a implementação dada ao projeto e como ela modela o funcionamento esperado do sistema. Testes serão feitos para obter dados a respeito do funcionamento e capacidade do protótipo e de como eles se comparam com o desempenho esperado na realidade. E por último será exposto as conclusões que foram obtidas a partir do projeto e dos resultados obtidos.

2 ASPECTOS CONCEITUAIS

A elaboração do sistema para o rastreamento de cargas passa pelo entendimento não só dos problemas a serem resolvidos, mas também dos conceitos e tecnologias que podem ser parte da solução para tais problemas. A seguir, serão introduzidos e justificados os principais conceitos que serão utilizados no projeto.

2.1 PROCESSO LOGÍSTICO

Para resolver o problema de rastreamento na cadeia logística, é necessário primeiro entender como essa cadeia funciona. Usualmente as empresas do ramo já utilizam sistemas logísticos para gerenciar os estoques e monitorar a entrada e saída de carga, sendo registradas todos os locais onde cada carga se encontra. Esses sistemas dependem dos funcionários que movimentam a carga para se manterem atualizados com o estado real em que se encontra o estoque, algo que abre caminho para inconsistências entre aquilo que realmente está no local com aquilo que o sistema registra. Além disso, a partir do momento que um veículo é carregado o estado da carga só é atualizado no sistema no momento em que o veículo chega em um outro ponto da cadeia logística.

O propósito do sistema é permitir o rastreamento da carga nesses trajetos em que os sistemas existentes não cobrem e criar um acompanhamento em tempo real do estado dos pacotes, dessa forma a localização de uma carga sempre estará à disposição dos destinatários e dos gerenciadores da transportadora. Portanto o sistema deve possuir um acompanhamento tanto dentro do veículo de transporte, quanto nas agências e galpões. O acompanhamento dentro do veículo é o foco do projeto já que é onde os sistemas atuais não alcançam, já nos pontos fixos esse rastreamento não será extremamente detalhado, sendo apenas necessário reconhecer que um pacote se encontra dentro da agência, e não uma posição exata onde ele se encontra, já que isso já é realizado pelos sistemas atuais.

Como já existe um sistema em uso pelas empresas de logística, é fundamental que a implementação do sistema ocorra de forma pouco intrusiva e atue mais como um complemento ao que já existe, de forma a aumentar no mínimo possível o trabalho realizado pelos funcionários. Isso significa que é necessário existir uma comunicação entre o sistema projetado e aquele que já é utilizado, ou seja, os pacotes já cadastrados no sistema anterior não precisariam

ser cadastrados novamente para que o rastreamento em tempo real passasse a reconhecê-lo. O ideal seria que essa comunicação dos pacotes cadastrados ocorresse de forma autônoma, mas isso dependeria do sistema que já é utilizado.

2.2 IDENTIFICAÇÃO E RASTREABILIDADE

O princípio de identificação e rastreabilidade dos pacotes é o que habilita o maior controle sobre a logística de transporte: ser capaz de diferenciar cada carga e ter informações sobre seu histórico são de grande ajuda para entender não só a situação da carga em si, como também de todo o sistema, que é um dos objetivos do projeto. Essas propriedades, a nível de pacote, permitirão ver suas condições (localização atual, tempo em transporte/parado, propriedades físicas como temperatura), enquanto que a nível de sistema permitirão controlar a lotação de estações de armazenamento e veículos de transporte, assim como otimizar rotas em função de tempo e/ou segurança.

Para que esses princípios sejam garantidos, há decisões arquiteturais tanto em *software* quanto em *hardware* a serem tomadas. Quanto a *software*, é preciso definir quais informações são pertinentes à rastreabilidade, e como salvá-las; isso leva à elaboração da arquitetura do banco de dados do sistema. Quanto ao *hardware*, deve-se escolher qual tecnologia será utilizada para identificar cada pacote, qual dispositivo realiza essa identificação, e como esse envia os dados para o sistema.

2.3 SISTEMAS EMBARCADOS

Um sistema embarcado pode ser definido como um sistema de propósito específico, dedicado a uma única tarefa, que utiliza de sensores e atuadores para interagir com o ambiente à sua volta. Usualmente os sistemas embarcados possuem um processador com um software, denominado *firmware*, que gerencia seus periféricos e executa a função para a qual o sistema foi desenvolvido. No caso desse projeto, o sistema embarcado deve ser capaz de identificar os pacotes presentes dentro do container de transporte e associar à identificação dele a localização do transporte. Essas informações devem ser enviadas pelo sistema ao servidor do qual essa informação é disponibilizada para o destinatário ou para a empresa responsável pelas mercadorias - conforme foi dito anteriormente.

A arquitetura do sistema embarcado destinado a esse projeto exige que haja um tipo de sensor capaz de identificar os pacotes dentro do containers, um dispositivo GPS para obter as coordenadas do sistema e uma interface de rede para conectar o sistema à rede e enviar as informações coletadas para o servidor. Controlando tudo isso é necessário um processador com um firmware desenvolvido no projeto para integrar os dispositivos conectados.

2.4 INTERNET DAS COISAS

O conceito de Internet das Coisas ("*Internet of Things*" ou "*IoT"*) trata-se de adicionar conectividade em dispositivos inteligentes, de modo que possam se comunicar e trocar informações entre si; Um dispositivo *IoT,* portanto, pode ser um pequeno sistema ou sensor, conectado com outros dispositivos *IoT*, formando uma rede via internet sem a necessidade de um servidor central, uma vez que cada um deles é capaz de realizar processamentos sobre os dados que tem e que recebem da rede.

Alternativamente, um dispositivo *IoT* pode ser um sistema que atua sozinho (no sentido de que não está numa rede com similares), utilizando sua capacidade de comunicação para trocar dados e informações com um servidor. Este caso, especificamente, é de interesse para o projeto, pois retrata uma possível solução para o problema de rastreamento de cargas, ao colocar um sistema embarcado munido de sensores que podem identificar pacotes no veículo de transporte; ainda com sua conexão à internet e/ou outras redes, o mesmo sistema embarcado ainda é capaz de adquirir a localização do veículo e então enviar esse conjunto de informações a um servidor.

De qualquer modo, não é necessário escolher entre os "modelos" apresentados de dispositivos IoT - por exemplo, seria perfeitamente possível criar um sistema em que os veículos de transporte são capazes de formar uma rede entre si e ainda se comunicar com o servidor.

2.5 SISTEMA OPERACIONAL

Grande parte dos componentes do projeto que rodam algum software contém uma aplicação que rege o funcionamento do dito componente; essa aplicação é denominada sistema operacional (SO), que é responsável pelo gerenciamento do *hardware* (memórias, processador,

dispositivos de interface) e de *software* (agendamento e priorização de tarefas, controle de permissão para execução ou acesso a recursos). Por exemplo, um programa que necessita de mais memória para continuar em execução não pode fazer a reserva de espaço - notifica-se o SO para que este libere o recurso desejado; essa hierarquia permite que uma máquina seja capaz de executar diversas aplicações simultaneamente (sejam programas diferentes, ou várias *threads* de um mesmo programa).

O SO rege o funcionamento de um dispositivo, logo é importante conhecer suas características, pois esse entendimento permite definir liberdades e restrições a serem consideradas no momento de se projetar a arquitetura do sistema. A partir disso, deve-se escolher os dispositivos (e, por consequência, o sistemas operacionais) que melhor se encaixam no escopo do projeto.

No caso do projeto discutido nesta monografia, são relevantes os SOs referentes à servidores, dispositivos móveis (*mobiles*) e, possivelmente, sistemas embarcados.

2.6 COMPUTAÇÃO EM NUVEM

O armazenamento em nuvem é uma forma de armazenar dados em servidores remotos, ao invés de localmente. Dessa maneira, o custo de manutenção de infraestrutura é significativamente menor, além de oferecer maneiras rápidas de expandir ou reduzir o tamanho dessa. É possível também, dessa maneira, fazer um gerenciamento eficaz de *load balancing*, que consiste em distribuir a carga de trabalho de uma determinada aplicação web entre várias máquinas, assim evitando perdas de desempenho em serviços muito requisitados. Há, no entanto, uma perda de controle sobre fatores como segurança, tanto em *hardware* quanto em *software*, uma vez que não é possível o acesso às máquinas da infraestrutura.

Embora ainda não tenha sido escolhida qual tecnologia será utilizada para esse armazenamento, ela será necessária para manter os dados consistentes entre as diversas interfaces da aplicação - sejam os dispositivos embarcados coletando e enviando dados para esses bancos remotos, seja para os aplicativos que acessarão essa informação e a disponibilizarão de forma *user-friendly* a um usuário final.

2.7 ARQUITETURA DISTRIBUÍDA

A ideia de uma arquitetura distribuída é distribuir o processamento dos dados em diversas entidades computacionais, também chamadas de nós, conectados por uma rede, dessa forma quando uma requisição chega ao servidor há um *cluster* de nós capaz de realizar as tarefas. Do ponto de vista arquitetural, os processadores devem estar implementados de forma a serem compatíveis com o processamento paralelo e distribuído, mas em uma camada mais alta, os processos rodando em cada um deles devem se comunicar por meio de um sistema de comunicação. Isso faz com que o sistema de comunicação utilizado e a rede na qual esse sistema se encontra são fundamentais para uma arquitetura distribuída. Para se aproveitar ao máximo as características dessa arquitetura os programas utilizados devem ser desenvolvidos especificamente para um processamento distribuído, sendo chamados de programas distribuídos, cujo desenvolvimento utiliza mecanismos criados especificamente para otimizar o processamento distribuído e paralelo.

Além de possuir múltiplas máquinas, uma arquitetura distribuída pode ter um número variável de máquinas além de uma topologia de rede modificável. Isso ocorre de forma a manter o tamanho do *cluster* suficientemente grande para não ocorrer uma sobrecarga nos nós, mas também não incluir nós demais a ponto deles ficarem ociosos por longos períodos. Essa mutabilidade da arquitetura é bastante utilizada por grandes *datacenters* que hospedam uma grande quantidade de programas simultaneamente.

Para o sistema deste projeto a utilização de uma arquitetura distribuída para os servidores responsáveis por receber as informações do rastreamento dos pacotes e enviar essas informações aos usuários é interessante devido ao grande fluxo de dados e aos requisitos funcionais e não-funcionais definidos. Pois, com uma arquitetura desse tipo, seria mais fácil conseguir tratar desse grande volume de requisições e ao mesmo tempo atender aos requisitos estabelecidos, mas com essa escolha a segurança do sistema. O projeto de uma arquitetura desse tipo está fora do escopo deste trabalho, portanto os servidores serão hospedados em um provedor de serviços de nuvem que utilizam uma arquitetura distribuída em seus *datacenters*. Com essa escolha, o cuidado com a segurança do sistema deve ser maior, já que ele estará em um ambiente em que diversos programas estão rodando e com muitas pessoas acessando seus servidores, tornando possível ataques de terceiros.

3 ESPECIFICAÇÃO DO SISTEMA

A seguir, será descrito o projeto do sistema, apresentando seus principais componentes e discutindo possibilidades de implementação, levando em consideração as definições e conceitos do tópico anterior.

3.1 NEGÓCIO

O sistema possui três pontos principais: o serviço de rastreamento de pacotes e cargas em tempo real, o serviço de controle da logística por parte das transportadoras e o serviço acompanhamento de pacotes por parte do destinatário; o primeiro serviço é o mais importante de todos, umas vez que os outros dependem dele para que possam realizar suas funções.

3.1.1 SERVIÇO DE RASTREAMENTO

Os pacotes, quando em transporte, devem apresentar seus dados de rastreamento (localização e propriedades físicas como temperatura) periodicamente para o servidor, que então guarda essas informações para formar um histórico para cada pacote; o período determinado é de entre 30s e 1min. Quando armazenados em depósitos, a leitura periódica pode ser mais esporádica, como algo em torno de 12h. O intuito é sempre ter dados atualizados acerca do pacote, assim como permitir a rápida descoberta de extravios ou roubos.

Para realizar esse rastreamento, considerou-se a possibilidade de utilizar tecnologias como RFID (etiquetas + leitoras) ou *beacons bluetooth* para identificar cada pacote, assim como o uso de sistemas embarcados como placas *Arduino* ou *Raspberry Pi* em veículos e depósitos para realizar as leituras e a comunicação com o servidor.

3.1.2 SERVIÇO DE CONTROLE DE LOGÍSTICA

Cada transportadora cadastrada no sistema terá acesso a um painel de controle, onde é possível realizar o agendamento de rotas, cadastrar pacotes, estações (depósitos de carga), veículos e motoristas; além disso, será possível acompanhar em tempo real o estado de uma rota em andamento, um pacote ou um veículo. O painel também terá um *dashboard*, em que o

resumo das informações será mostrado, assim como indicar anormalidades no fluxo dos pacotes, como por exemplo:

- rotas atrasadas;
- falha na atualização periódica do rastreamento de pacotes;
- pacotes parados em depósito por muito tempo;

Além do painel, haverá uma API disponível à transportadora para realizar o cadastro automático de pacotes, com a integração com sistemas de logística já existentes. A implementação do painel é feita por uma aplicação web; essa, junto da API, deve ser hospedada em um servidor nuvem, utilizando serviços como *Amazon Web Services* (AWS) ou *Microsoft Azure*.

3.1.3 Serviço de Acompanhamento de Pacotes

O serviço de rastreamento idealizado permite que o destinatário tenha acesso a mais informações sobre o pacote do que lhe é oferecido por outros sistemas de rastreamento existentes. Através de uma página web ou um aplicativo de *smartphone*/*tablet*, o destinatário visualiza essas informações.

3.1.4 Integração e Visão Geral

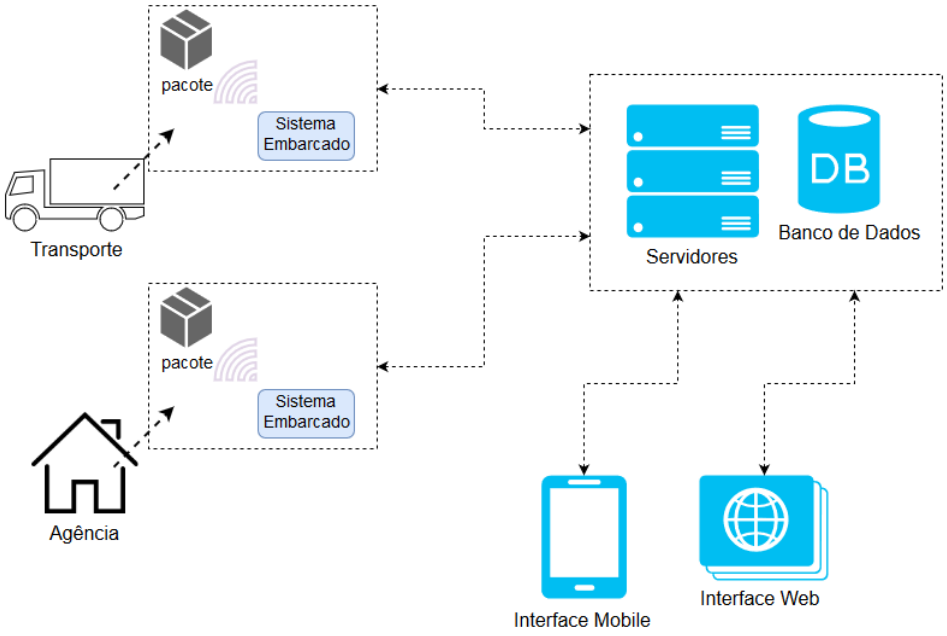


Figura 1 – Esquema ilustrativo da arquitetura do projeto. Fonte: autores

Sistemas embarcados nas agências e nos veículos de transporte realizam a identificação dos pacotes, através de sensores e leitores acoplados. A leitura pode ser periódica ou requisitada, e para isso os sistemas também possuem conexão com a internet para se comunicar com o servidor.

O servidor, por sua vez, faz a conexão com o banco de dados e também realiza a comunicação com os dispositivos utilizados pelos usuários finais (*mobiles* e páginas Web). Devido ao volume de tráfego esperado para os diferentes serviços do sistema, pode-se explorar a possibilidade de realizar uma divisão em dois servidores, cada um com seu respectivo banco de dados; o primeiro par servidor-banco seria responsável por lidar e armazenar os dados de rastreamento (pacotes, rotas, amostras de localização), enquanto que o segundo par trata dos serviços restantes (destinatários, transportadoras, estações, veículos e endereços). Alternativamente, a arquitetura projetada permite uma fragmentação maior do sistema, de modo a implantar o modelo de microsserviços.

3.1.5 ARQUITETURA DE CLASSES

A partir da descrição do sistema, foram elaborados as classes e o respectivo diagrama para melhor entender o relacionamento das informações. No caso, criou-se um diagrama para a parte Web, que engloba a aplicação Web e a API, e outro diagrama para a parte *mobile*, que pode ser utilizado de referência para *tablets*, *smartphones* e outros dispositivos móveis do tipo.

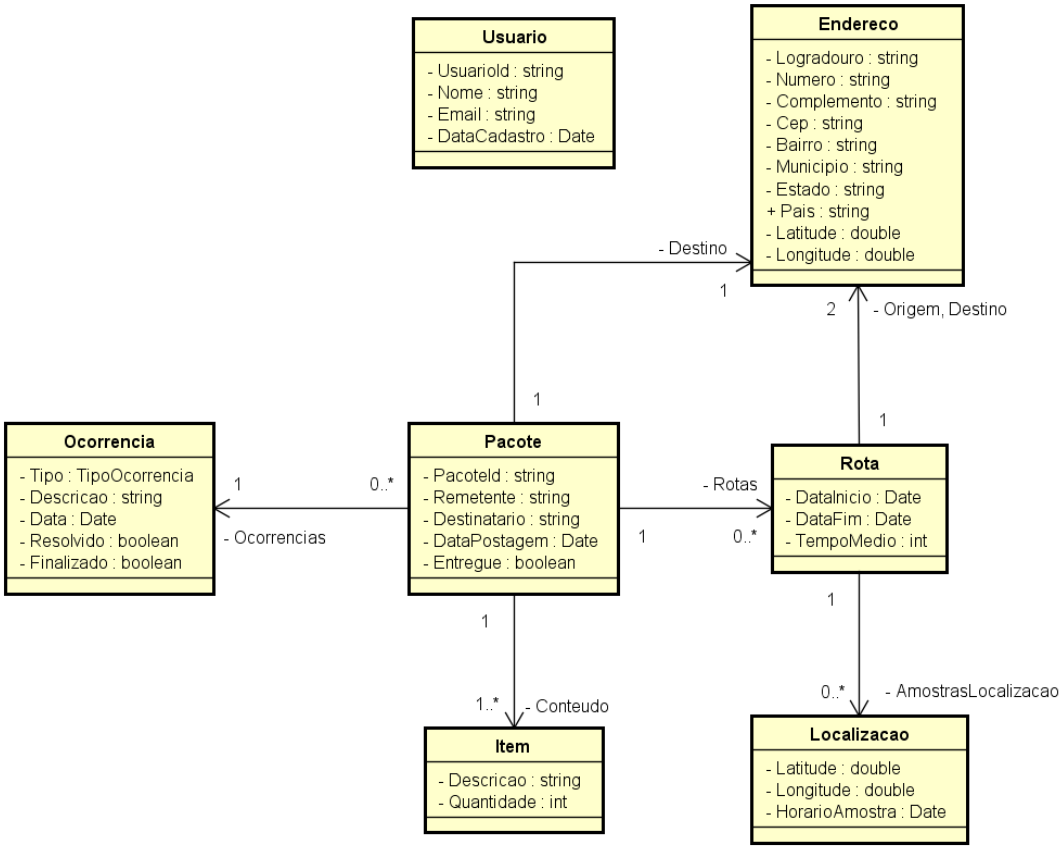


Figura 2 - Diagrama de classes para dispositivos *mobile*. Fonte: autores

O diagrama de classes para o código *mobile* reflete as informações que serão disponibilizadas ao usuário que seja um destinatário do pacote. Cada pacote terá um conjunto de rotas, que são o caminho realizado entre duas agências; cada rota, por sua vez, contém um conjunto de amostras de localização, que são as leituras periódicas realizadas dentro do veículo. Tais amostras trazem apenas o horário e localização. Além disso, o pacote também apresenta informações básicas, como seu conteúdo, remetente e eventuais ocorrências.

Um usuário só tem acesso aos próprios pacotes nesse caso, de modo que a relação direta entre as classes Usuario e Pacote não se faz necessária.

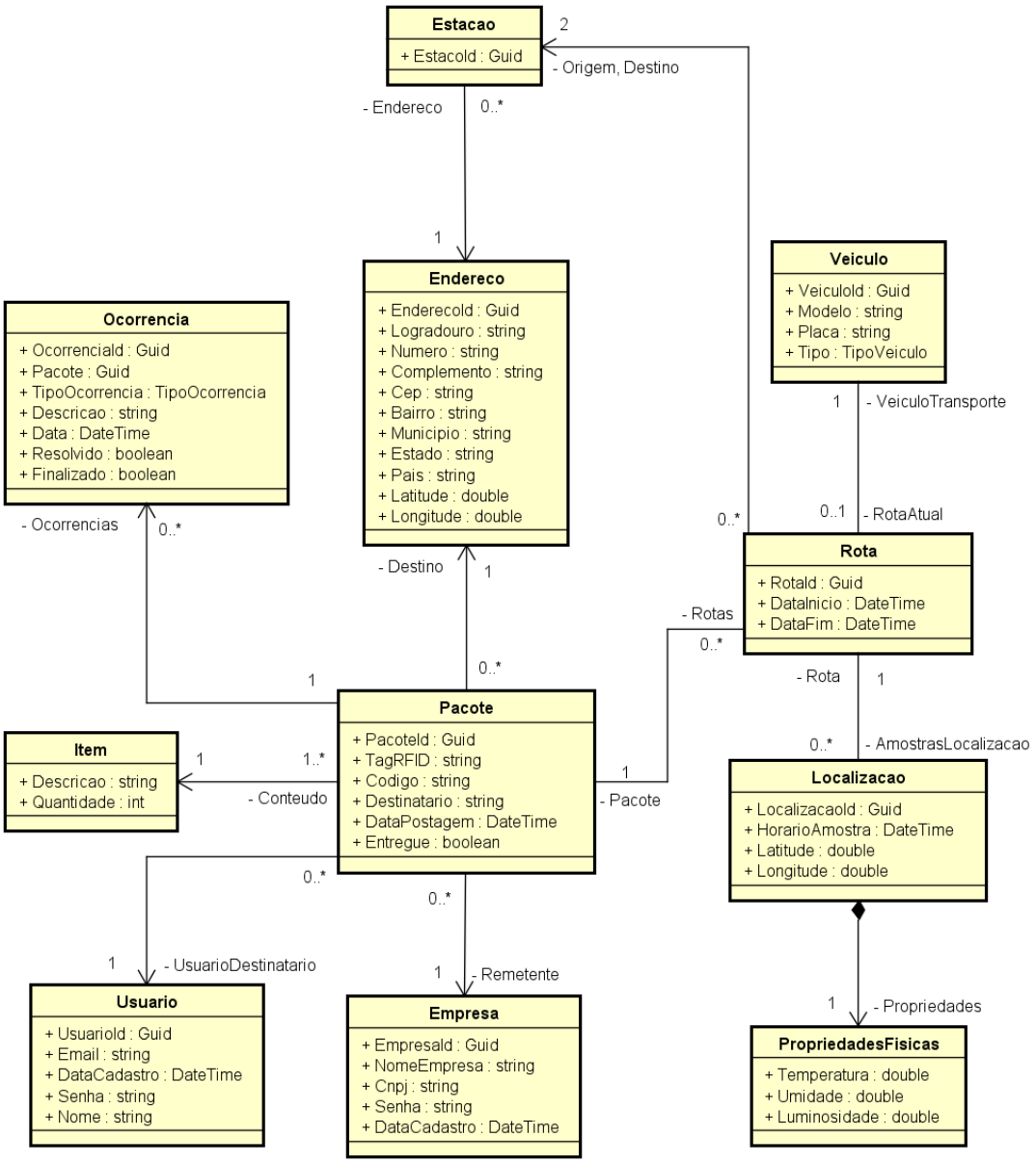


Figura 3 - Diagrama de classes para Web e API. Fonte: autores

O diagrama de classes para Web, por sua vez, apresenta todas as relações existentes entre as informações do sistema. Desse modo, é mais fácil garantir a rastreabilidade dos pacotes, além de simplificar o processo de análise de dados.

A partir do diagrama, pode-se observar que o pacote contém informações sobre seu conteúdo, ocorrências relacionadas, seu destinatário e remetente (no caso uma empresa); a informação sobre a movimentação possui também informações sobre o estado do pacote, explicitado pela classe *PropriedadesFisicas*, agregada a *Localizacao*. Como as propriedades

medidas podem variar, optou-se por seguir esse caminho em vez de adicionar atributos individualmente na classe *Localizacao*.

3.2 Requisitos do Sistema

A partir da descrição do negócio, levantou-se formalmente os requisitos do sistema. A seguir, os requisitos foram divididos entre funcionais e não-funcionais, e organizados de acordo com as áreas de atuação.

3.2.1 Requisitos Funcionais

3.2.1.1 Rastreamento

- Coletar dados de localização e outras propriedades monitoradas de cada pacote, guardando essas informações no banco de dados com o respectivo horário de coleta;
- Alertar o sistema quando um pacote “some” do sistema:
    - Se o pacote estiver associado a uma rota em andamento, e na atualização mais recente de tal rota, o pacote não for encontrado;
    - Se o pacote estiver em uma estação, sem rota agendada, e na atualização mais recente do inventário da estação, o pacote não for encontrado;
- Alertar o sistema quando uma rota não enviar sua atualização periódica;

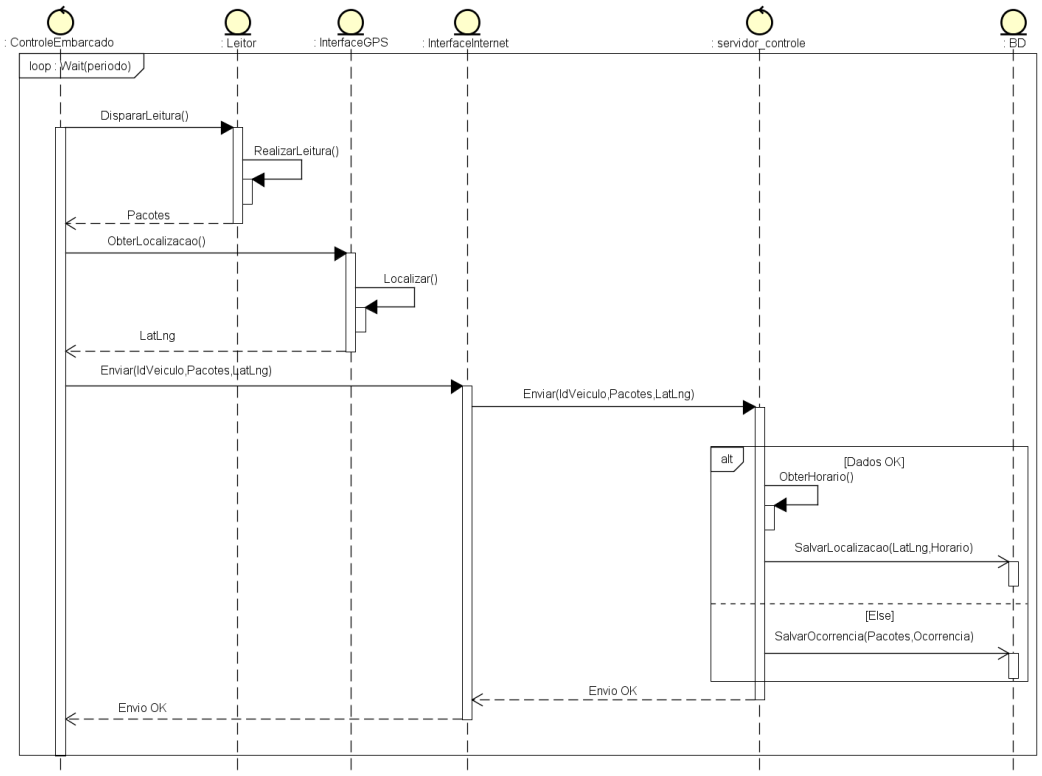


Figura 4 - Diagrama de sequência do rastreamento de pacotes. Fonte: autores

O diagrama de sequência acima descreve a funcionalidade da coleta de dados de rastreamento. Identificam-se à esquerda as entidades correspondentes ao sistema embarcado e leitor/sensor, e à direita o servidor e o banco de dados.

3.2.1.2 TRANSPORTADORA

- Realizar o cadastro da empresa no sistema, para ter acesso às funcionalidades subsequentes;
- Realizar o cadastro de pacotes no sistema;
- Realizar o cadastro de estações no sistema;
- Realizar o cadastro de veículos no sistema;
- Realizar o cadastro de motoristas no sistema;
- Visualização de um painel geral (*dashboard*) que contém um resumo de informações sobre a logística da empresa:
    - Quantidade de pacotes em circulação;
    - Quantidade de rotas/veículos em andamento;
    - Quantidade de pacotes com problemas (alertas);

- Gráficos e histogramas referentes aos dados acima ao longo do tempo;

- Visualização dos detalhes de uma rota;
- Visualização dos detalhes de um veículo;
- Visualização dos detalhes de um motorista;
- Visualização dos detalhes de um pacote;

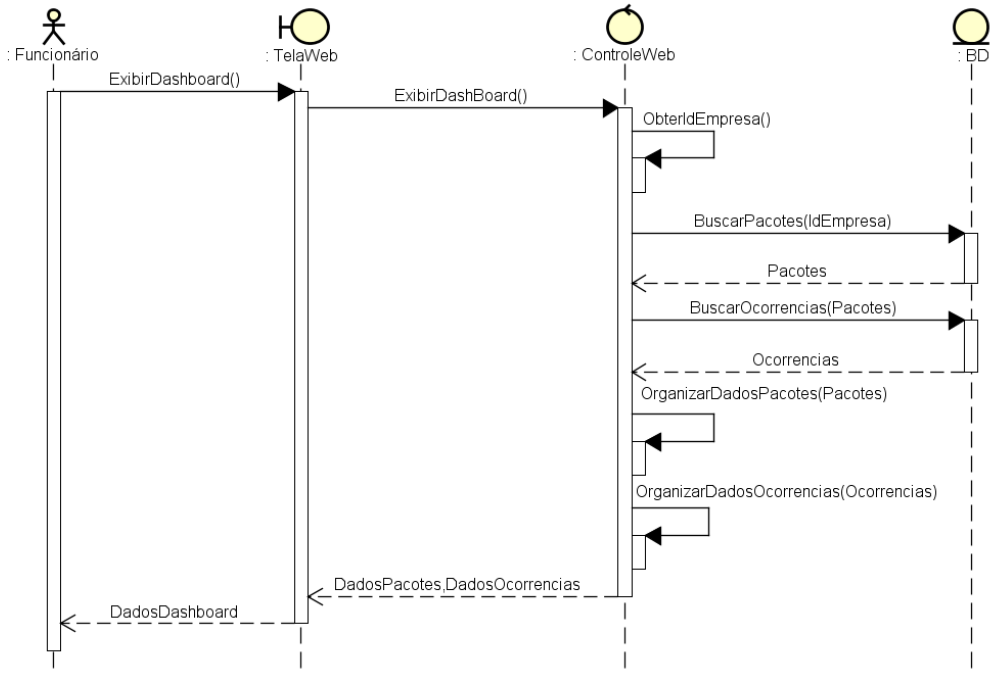


Figura 5 - Diagrama de sequência do acesso à informações pela transportadora. Fonte: autores

O diagrama de sequência acima descreve a funcionalidade do *dashboard*, acessado por um funcionário da transportadora. O caminho é simples, mas indica que o processamento de dados atualizados é feito na hora de acesso.

3.2.1.3 DESTINATÁRIO

- Realizar o cadastro do usuário no sistema, para ter acesso às funcionalidades subsequentes;
- Registrar um pacote em seu nome, através de um código referente ao pacote;
- Obter uma lista dos pacotes em circulação associados ao usuário;
- Obter uma lista dos pacotes entregues associados ao usuário;

- Obter os detalhes de um pacote específico, incluindo detalhes das rotas e coletas de localização;

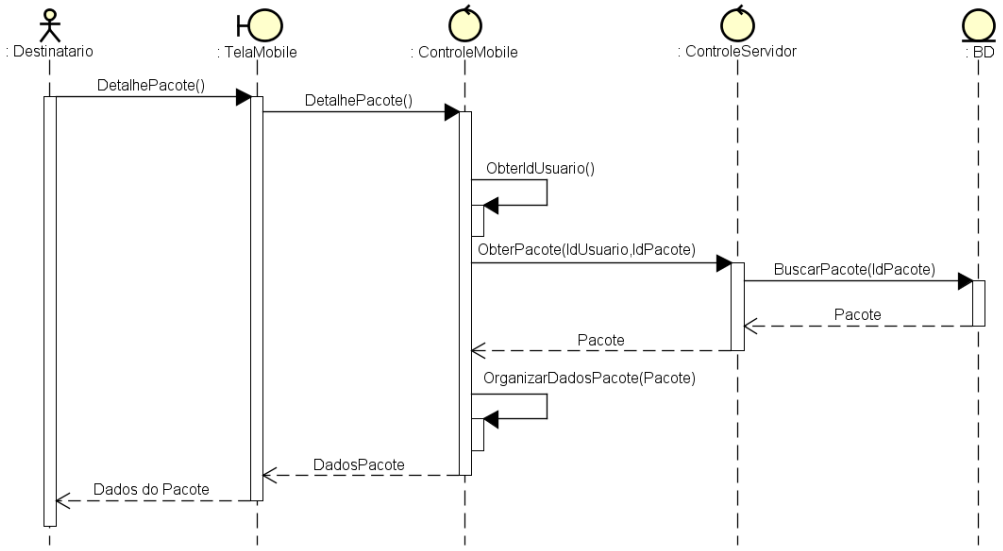


Figura 6 - Diagrama de sequência do acesso à informações pelo destinatário. Fonte: autores

O diagrama de sequência acima descreve a funcionalidade principal disponível para o destinatário, que é o acesso às informações de rastreamento em tempo real de um dado pacote. O diagrama evidencia que há pouco processamento de dados por parte do dispositivo *mobile*.

3.2.2 REQUISITOS NÃO-FUNCIONAIS

3.2.2.1 RASTREAMENTO

- Intervalo entre as atualizações da localização do pacote deve ser por volta de 2 a 5 minutos. Essa taxa de amostragem indica de quanto em quanto tempo a leitora RFID deverá ser acionada para verificar quais pacotes se encontram no veículo e obter a localização pela antena GPS.
    - Uma frequência muito alta de atualização leva a um gasto maior de energia sem grandes benefícios já que não há uma necessidade muito elevada de precisão do momento em que o algo ocorreu com o pacote, além disso as limitações energéticas são relevantes já que o sistema dependerá de uma fonte energética limitada como é uma bateria veicular.

- O rastreamento deve saber lidar com situações em que o sinal do GPS e/ou da rede celular não estejam disponíveis.
    - Tal situação pode por falta de cobertura da rede celular na região em que o veículo se encontra, pelo fato do satélite da rede GPS não conseguir se comunicar com a antena, por exemplo quando o caminhão passa por um túnel, ou até mesmo em uma tentativa de interferência no funcionamento do sistema, como ocorre em ataques em que se utilizam *jammers* de sinal.
- O tempo de notificação de uma situação anormal no transporte deve ser entre 1 e 2 minutos.
    - Uma situação anormal seria quando a *tag* de um pacote que estava sendo lida, parou de ser detectada pela leitora.
    - O tempo de notificação seria o tempo que demoraria a partir do instante que o sistema embarcado reconheceu a falta de uma *tag* até o momento em que o aviso aparece no painel de controle da transportadora.
- Segurança dos dados transmitidos e armazenados pelo sistema deve ser elevada.
    - A obtenção da localização e o conteúdo de todos os veículos de transporte é extremamente prejudicial aos *stakeholders*.

3.2.2.2 TRANSPORTADORA

- Painel de Controle da aplicação deve permitir uma rápida distinção das informações mais críticas do restante dos dados exibidos.
    - Isso é crucial para o caso de avisos de extravio de pacotes em transporte e outras anomalias que possam ocorrer.
- A atualização das informações exibidas deve ocorrer em tempo real, ou seja, em intervalos de tempo cuja duração minimize o número de inconsistências entre o verdadeiro estado dos pacotes e aquele registrado no sistema.
    - Considerando a natureza da aplicação do sistema, foi considerado que uma atualização a cada minuto seria aceitável.

3.2.2.3 DESTINATÁRIO

- Para o destinatário é importante que as informações exibidas na tela de rastreamento sejam corretas e que representem um estado recente do pacote.
    - Para o usuário a precisão da localização e a taxa de atualização do rastreamento podem ser menores do que as utilizadas na transportadora. Portanto uma precisão da localização de alguns metros e uma taxa de atualização de 5 minutos.
- As telas do aplicativo devem permitir uma fácil identificação de quais pacotes estão em transporte e aonde eles estão, além de indicar quais pacotes o usuário já recebeu, caso ele queira consultar alguma informação à respeito deles.

3.3 MECANISMOS DE ENGENHARIA

A partir da definição dos requisitos funcionais e não funcionais a escolha dos mecanismos que irão ser integrados a arquitetura planejada foi realizada. Como base para a escolha utilizamos um conjunto de critérios para auxiliar nessa escolha além dos requisitos levantados, que nesse caso foram a familiaridade dos membros do grupo com aquela tecnologia e o custo dela para a implementação planejada, além de quão adequada ela é para a satisfação dos requisitos. Dessa forma, ao escolhermos levamos em consideração o quão trabalhoso, custoso e adequado, do ponto de vista dos requisitos, seria utilizar um dado mecanismo na solução. Os tópicos seguintes detalham os componentes da arquitetura e os mecanismos presentes em cada um.

3.3.1 SERVIDOR NA NUVEM

Considerando o escopo da aplicação real, a quantidade de dados a ser processada pelo servidor poderá ser extremamente alta, principalmente nos dados referentes ao rastreamento. Para evitar que o servidor se sobrecarregue e perca os dados de maior importância em meio ao fluxo de pedidos a divisão do sistema em microsserviços, sendo um servidor responsável pelo rastreamento do pacote, outro pela aplicação Web utilizada pelas empresas e outro pelo aplicativo Android. Através dessa divisão conseguimos auxiliar no balanceamento do fluxo, já que esses três componentes do sistema são distintos o suficiente para que essa divisão não gere uma replicação excessiva entre os bancos de dados de cada servidor.

Além de uma divisão em microsserviços, a utilização de filas diferenciadas para cada tipo de requisição que chega a um servidor é um outro mecanismo que ajuda a manter os tempos de resposta do servidor de acordo com os valores dos requisitos, principalmente nos casos mais críticos como são os de detecção de ausência de pacotes no transporte. Portanto as filas mais prioritárias são aquelas ligadas aos avisos de anomalias no transporte ou de perda de conexão prolongada com um sistema embarcado de um veículo. Essas filas também devem existir para as comunicações entre os microsserviços, pois, por exemplo, a partir da detecção da ausência de um pacote, o aviso deve ser transmitido do servidor de rastreamento para o servidor da aplicação Web, e nesse trajeto inteiro a prioridade desse evento deve ser respeitada.

O fluxo intenso de dados recebidos pelos servidores varia conforme o horário, por exemplo, durante os dias da semana entre as 8 horas da manhã e as 6 horas da tarde a quantidade de mercadorias em transporte e o número de pessoas acessando o aplicativo para verificar o estado da carga é muito mais elevado do que durante um fim de semana à noite. Tal variação do fluxo geraria grandes períodos de ociosidade dos servidores, mas caso uma arquitetura distribuída com um número variável de máquinas seja utilizado no lugar de servidores dedicados convencionais, o desperdício de energia e recursos seria minimizado. Com isso, a escolha de hospedar o sistema em uma plataforma de computação nuvem que possui uma arquitetura distribuída altamente escalável permite que sejam implementados todos os mecanismos citados, além de manter os custos baixos.

Com os recursos disponíveis em uma nuvem e a quantia de dados gerados pelos sistemas de rastreamento, o sistema pode ser capaz de analisar essas informações por meio de técnicas de inteligência artificial, como *Machine Learning* e redes neurais, em busca de padrões, tendências e estatísticas que podem auxiliar as empresas de logística na tomada de decisões sobre mudanças no processo. Por exemplo, considerando um certo trajeto em uma rodovia, com um número considerável de amostras de rastreamentos realizados nele, podemos estimar um tempo médio para que um pacote atravesse tal trecho, agora caso algum pacote esteja nesse trajeto por um intervalo descomunal de tempo, é possível avisar a empresa sobre tal fato para que eles verifiquem se algo está errado. Além disso, com o registro de entrada e saída para transporte de cada pacote de uma agência específica, é possível identificar se ela está demorando demais para enviar os pacotes para o seu destino. Como podemos ver, existem diversas possibilidades para serem exploradas caso uma inteligência artificial seja aplicada sobre os dados coletados graças à coleta das localizações dos pacotes em tempo real.

3.3.2 IDENTIFICAÇÃO POR RFID

A escolha por qual tecnologia seria utilizada para o rastreamento envolveu principalmente a limitação energética existente quando o rastreamento deve ser realizado dentro do veículo e o custo envolvido em identificar cada pacote, ou seja o quão caro seria equipar cada pacote com os componentes necessários para ele ser rastreado durante o transporte. Nesses quesitos a utilização de RFID se mostrou melhor que as outras, principalmente no custo envolvido, já que as *tags* que serão associadas aos pacotes são passivas, de baixo custo e descartáveis, apesar de que a leitora RFID e as antenas podem ter um custo elevado.

Devido às dimensões do compartimento de um veículo de transporte, a escolha da frequência de operação das *tags*, leitoras e antenas nos levou à faixa *Ultra High Frequency*, ou UHF, pois nessa faixa a distância entre as *tags* e a antena pode ser de 1 metro até 12 metros e o custo das *tags* é baixo, sendo portanto a mais adequada para ser utilizada nesse projeto (FINKENZELLER, 2010). Em contrapartida, as leitoras UHF consomem mais energia e são mais caras que as das outras faixas de frequência, mas considerando as vantagens listadas anteriormente, tal investimento é recompensado.

Uma outra possibilidade permitida pela utilização do RFID é a inclusão de sensores nas *tags* utilizadas nos pacotes. A possibilidade de se monitorar em tempo real alguma grandeza física seria bem aproveitada nos setores de transporte de cargas valiosas ou perecíveis, aonde o controle da temperatura, umidade ou alguma outra propriedade é crucial para a qualidade da carga transportada. Os trabalhos de Yin et al. (2010) e Vaz et al. (2010) mostram que a precisão oferecida por *tags* com sensores de temperatura seria suficiente para o controle de temperatura de materiais sensíveis à variações.

3.3.3 SISTEMA EMBARCADO

Levando em consideração as limitações energéticas presentes no caminhão, inicialmente a utilização de um sistema embarcado em conjunto com a leitora RFID parece ser desnecessária, pois a leitora por si só consegue enviar as *tags* lidas para um dispositivo simples com uma antena 3G, ou alguma outra antena que permita uma conexão a um rede para a transmissão dos dados, que apenas repasse os dados da leitora pela rede até chegar ao servidor. Com a utilização de um sistema embarcado mais complexo, existe a oportunidade de um processamento local dos dados antes da chegada deles ao servidor, portanto o sistema

embarcado pode realizar uma análise das *tags* lidas, e enviar os resultados da análise para o servidor. Por meio dessa análise local, a quantidade de dados transmitida ao servidor é reduzida, pois o sistema embarcado pode apenas notificar o servidor quando há uma mudança no estado dos pacotes e não a cada leitura realizada. Isso é vantajoso em dois aspectos: economia nos dados enviados e no número de mensagens enviadas ao servidor. Portanto, apesar do aumento no consumo de energia, a utilização de um sistema embarcado contribui para uma economia no gasto de dados e para um melhor desempenho do sistema.

Entre os requisitos não funcionais, está listado uma necessidade do sistema de rastreamento conseguir lidar com situações em que não há sinal da rede GPS ou da rede 3G, impossibilitando os rastreamento em um dado instante. Com um sistema embarcado dotado de algum dispositivo de armazenamento, como um cartão SD, por exemplo, pode-se criar um buffer de amostras de leitura com um *timestamp* que armazenará aquilo que foi lido até que seja possível enviar esses dados ao servidor. Tal mecanismo auxiliaria também em casos de ataques com a utilização de *jammers*, pois com a comunicação com o servidor bloqueada, o buffer entraria em efeito e armazenaria as mudanças na leitura dos pacotes, permitindo que os responsáveis pelo transporte consigam analisar o caso com uma maior quantidade de informações à respeito do que ocorreu.

4 O SISTEMA IMPLEMENTADO

Baseado na especificação e na arquitetura descrita na seção anterior, foi elaborada uma implementação que seria possível de ser realizada e testada considerando as limitações de recursos presentes no projeto, já que o escopo de uma aplicação real envolveria uma grande quantia de trabalho em diversas áreas de conhecimento para podermos satisfazer todos os requisitos listados. A implementação foi construída tendo em mente que os principais desafios que ela deveria enfrentar seriam conseguir ler as tags presentes no compartimento, adquirir a localização atual, enviar essas informações a um servidor, que por sua vez consiga armazenar os dados para depois enviá-los caso alguém consulte as informações sobre esse pacote.

4.1 DESCRIÇÃO ARQUITETURAL

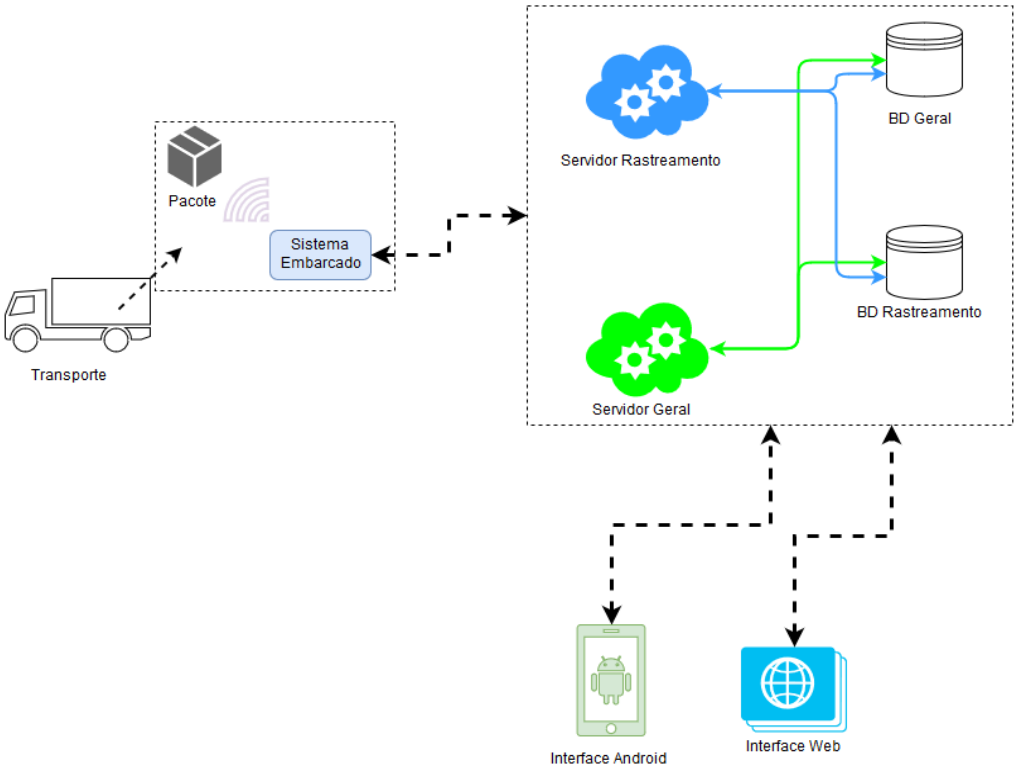


Figura 7 – Esquema ilustrativo da arquitetura implementada. Fonte: autores

A arquitetura implementada foca no rastreamento dos pacotes no veículo, na divisão dos servidores e no acesso à informação por parte dos usuários do sistema, com ênfase ao destinatário. A seguir, a listagem e explicação sobre a escolha de tecnologias para cada componente.

4.1.1 Dispositivos Mobile - Android (Java)

A interface do destinatário com o sistema será através de um aplicativo para sistemas *Android*; nesse caso, a escolha foi feita em função da familiaridade dos integrantes do grupo tanto com o SO quanto com a linguagem *Java*, uma das opções para desenvolvimento. Para efeito de testes, *smartphones Android* são muito mais fáceis de se obter.

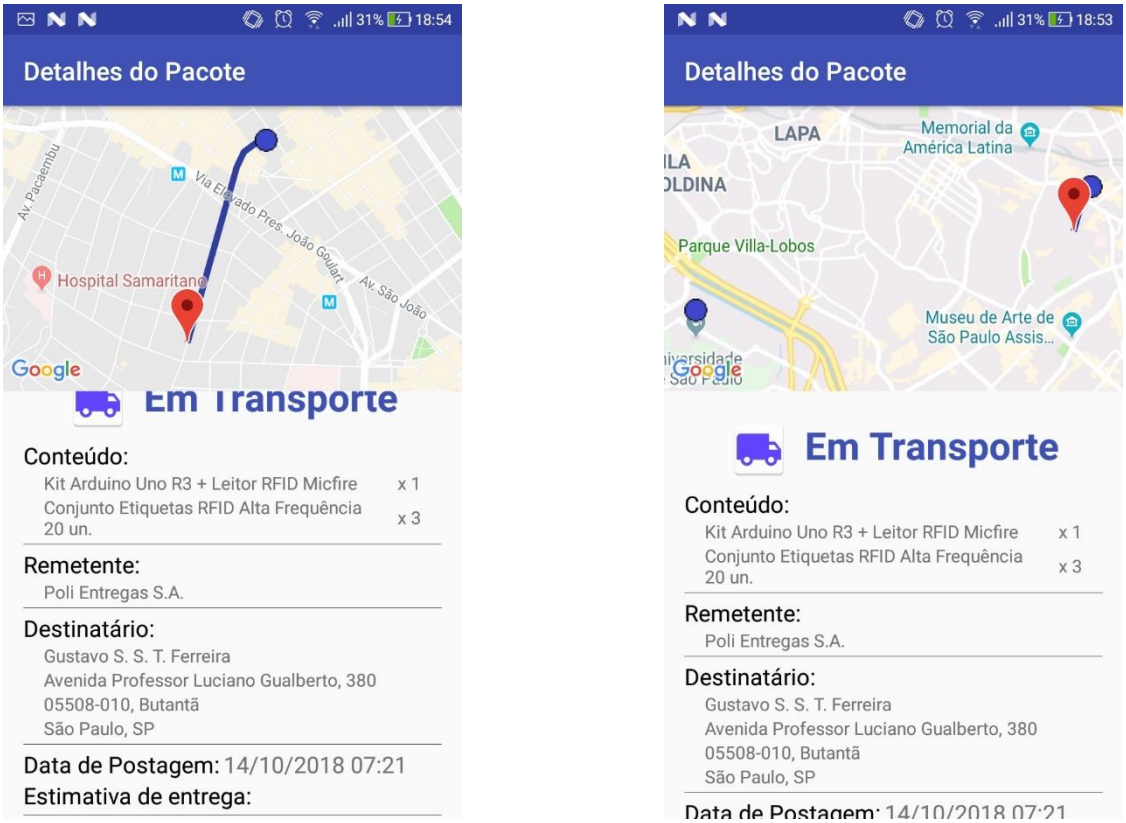


Figura 8 - Tela de detalhes do pacote do aplicativo *Android*. Fonte: autores

4.1.2 API e Aplicação Web - .NET

Tanto a API quanto a aplicação Web para os usuários foi desenvolvida dentro do *framework* .NET, utilizando C# para a seção do código correspondente à lógica de negócio. Dessa maneira, o desenvolvimento pode ser feito em conjunto, sendo possível aproveitar o mesmo código para API e aplicação. Como resultado, correções ou adições ao sistema nessa parte são realizadas com mais facilidade, reduzindo a possibilidade de duas versões de uma

mesma entidade estarem presentes no código. O *framework* .NET também proporciona um ambiente em que a implementação, teste e integração das partes se torna mais simples.

4.1.3 SERVIDOR EM NUVEM - AMAZON EC2

A escolha pela *Amazon Web Services* (AWS) dentre outras possibilidades foi feita levando em consideração a familiaridade dos integrantes do grupo com os serviços, assim como pelo acesso a uma conta cedida pelo professor Reginaldo Arakaki, o que nos permitiu o uso dos serviços sem a necessidade de pagamento.

Considerando as opções para hospedagem dentro da AWS, escolheu-se a EC2; desse modo, foi possível criar instâncias rodando o SO *Windows Server*, o que permite o fácil *deploy* de sistemas criados em .NET, uma vez que ambos foram desenvolvidos pela *Microsoft* com integração nativa. Embora não seja necessário para efeitos de teste do projeto, a EC2 apresenta funcionalidades úteis para o tipo de sistema desenvolvido, como o *Load Balancing*, que dinamicamente emprega a quantidade de máquinas necessárias para atender a todo o volume conexões à instância.

Foram criadas duas instâncias *t2.micro* com *Windows Server* 2016.

4.1.4 BD EM NUVEM - AMAZON RDS

O banco de dados foi projetado e implementado em SQL; por conta disso, foi necessário utilizar um serviço para bancos de dados relacionais. O *Amazon Relational Database Service* oferece o necessário e é de fácil utilização.

Foram criados duas instâncias *db.t2.micro* com *MySQL*.

4.1.5 LEITORA RFID 1 - MFRC522 MIFARE

Essa foi a primeira leitora RFID utilizada para testes. O conhecimento prévio dos integrantes da existência de um *shield* RFID compatível com dispositivos *Arduino* foi crucial para a escolha dessa leitora.

A leitora fez parte da primeira montagem e serviu principalmente para familiarizar os integrantes algumas das tecnologias escolhidas. Algumas limitações críticas do *shield* foram identificadas e forçaram a troca por uma leitora que oferecesse mais funcionalidades e mais flexibilidade.

As principais limitações encontradas com a leitora *Mfrc522 Mifare* foram:

- Baixo alcance: apenas *tags* a alguns centímetros da antena eram captadas;
- Ângulo das tags: embora não tão crítico, e afetado também pelo alcance da antena, as *tags* só eram captadas caso estivessem paralelas a ela, o que reduzia a flexibilidade da implementação;
- Leitura inflexível das *tags*: há dois problemas associados à leitura das *tags*. O primeiro deles é que apenas uma *tag* poderia ser lida por vez, e isso era um problema do próprio modelo de negócio que exigia que diversos pacotes fossem rastreados simultaneamente. O segundo problema é que a biblioteca que controla a leitora não suporta a leitura da mesma *tag* mais de uma vez sem sua remoção do campo de alcance da antena; isso também conflitava com o modelo de negócio pela exigência de amostrar a localização dos pacotes várias vezes durante o transporte sem sua remoção do container de transporte.

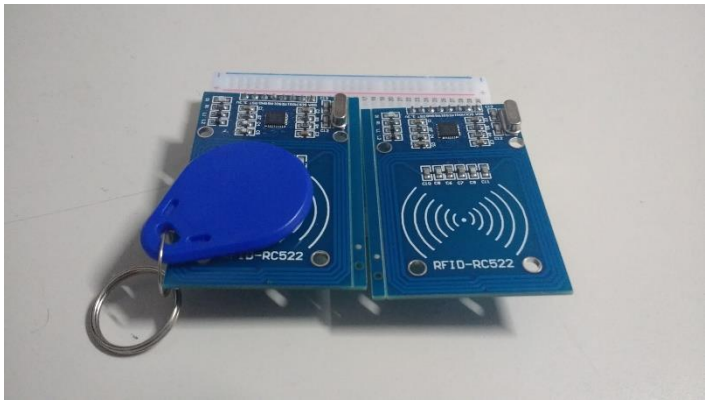


Figura 9 - Leitoras RFID *Mfrc522 Mifare* com *tag*. Fonte: autores

4.1.6 LEITORA RFID 2 - ACURA GLOBAL EDGE-50

O co-orientador do projeto, o professor José Kleber, disponibilizou para o grupo a leitora RFID *Acura Global Edge-50*. Ao trabalhar com o novo modelo de leitora os problemas citados

anteriormente foram resolvidos, pois a leitora já disponibiliza suporte para as funcionalidades necessárias e oferecia antenas com maior potência e alcance.

A *Acura* disponibiliza uma API, chamada *Mercury API*; através desta API é possível criar uma aplicação que configure a conexão da leitora e execute a leitura das *tags* dentro do alcance da(s) antena(s) conectada(s). Através do uso da API, a elaboração de uma aplicação para computador que oferecesse as funcionalidades do exigidas pelo projeto exigiria pouco esforço.

Embora a API possua compatibilidade com dispositivos *Arduino*, o modelo utilizado (*Uno R3*) não possui o espaço de armazenamento necessário para comportar a biblioteca. O dispositivo embarcado conhecido pelo grupo que tornaria possível o uso da API era a *Raspberry Pi*, que se encaixa na categoria de sistemas *Linux* embarcados aos quais a API oferece compatibilidade.

Figura 10 - Leitora RFID *Acura Global Edge-50*. Fonte: autores

4.1.7 Sistema Embarcado - Arduino + Raspberry Pi

Na montagem final do projeto, o sistema embarcado é composto por um *Arduino Uno R3* equipado com o módulo SIM5230E para conexão 3G/GPS, e um *Raspberry Pi 3 Model B*. O *Arduino* é usado para obter conexão com a internet e acesso à dados de localização, enquanto o *Raspberry Pi* controla a leitora RFID. O módulo SIM5230E, por sua vez, possui duas antenas

conectadas para permitir a comunicação com a rede 3G e satélites GPS; é necessário inserir um cartão SIM no módulo para utilizar a rede 3G.

Inicialmente, os testes realizados antes de se obter a leitora RFID definitiva foram feitos utilizando apenas o *Arduino*. Após a troca de leitoras, no entanto, verificou-se a incompatibilidade do modelo *Uno R3* com a biblioteca *Mercury API* da leitora *Edge-50*, pois a placa em questão não possui espaço de armazenamento suficiente para o programa compilado com tal biblioteca. Uma vez que o módulo SIM5230E já havia sido comprado e tinha-se acesso a um *Raspberry Pi*, optou-se por utilizar os dois em conjunto.

A placa *Raspberry Pi* é um pequeno computador que, ao contrário de sistemas embarcados, possui um SO, usualmente o *Raspbian*, que é baseado no *Debian* e foi criado especificamente para o *hardware* da placa. Por ser um pequeno computador, a *Raspberry Pi* conta com uma arquitetura mais completa, sendo capaz de realizar praticamente qualquer tarefa que um computador convencional pode realizar. Mas, devido a essa arquitetura, a *Raspberry Pi* consome mais energia e pode ser mais lenta que um sistema embarcado, pois sua arquitetura não está otimizada para a realização de um conjunto mais restrito de atividades.

O código para o programa inserido no Arduino foi feito em C++, enquanto que o programa inserido no *Raspberry Pi* foi feito em *Java*.

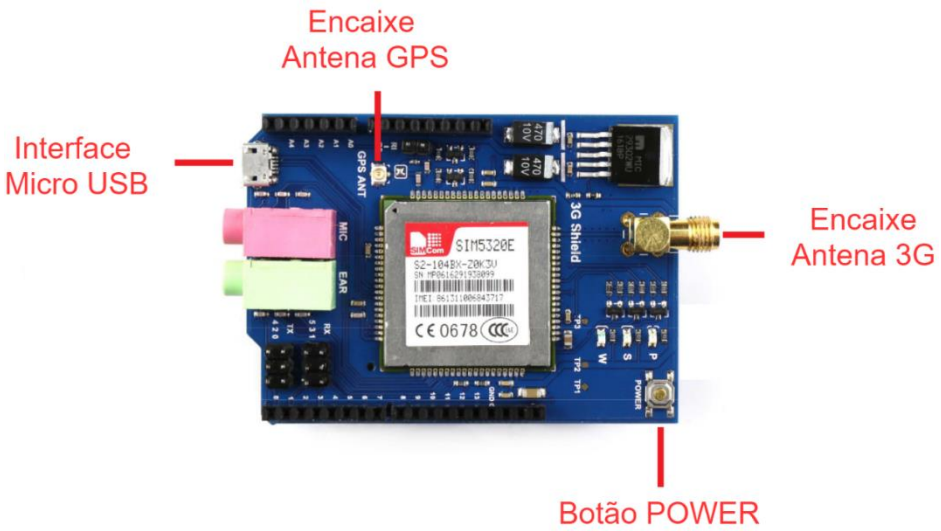


Figura 11 - Módulo SIM5230E para Arduino. Fonte: Tiny Sine (2017) (adaptado)

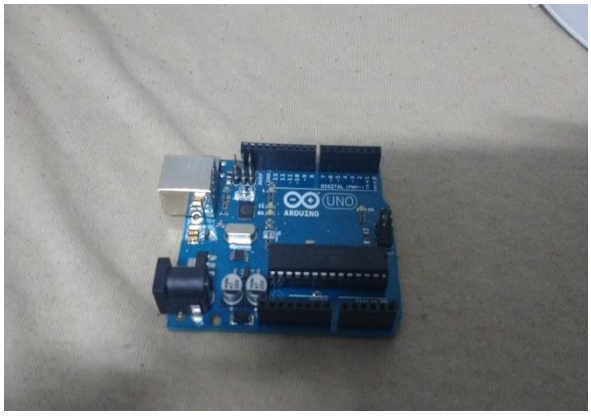


Figura 12 - Placa *Arduino Uno R3*. Fonte: autores

Figura 13 - Placa *Raspberry Pi 3 Model B*. Fonte autores

4.2 ARQUITETURA DE CLASSES

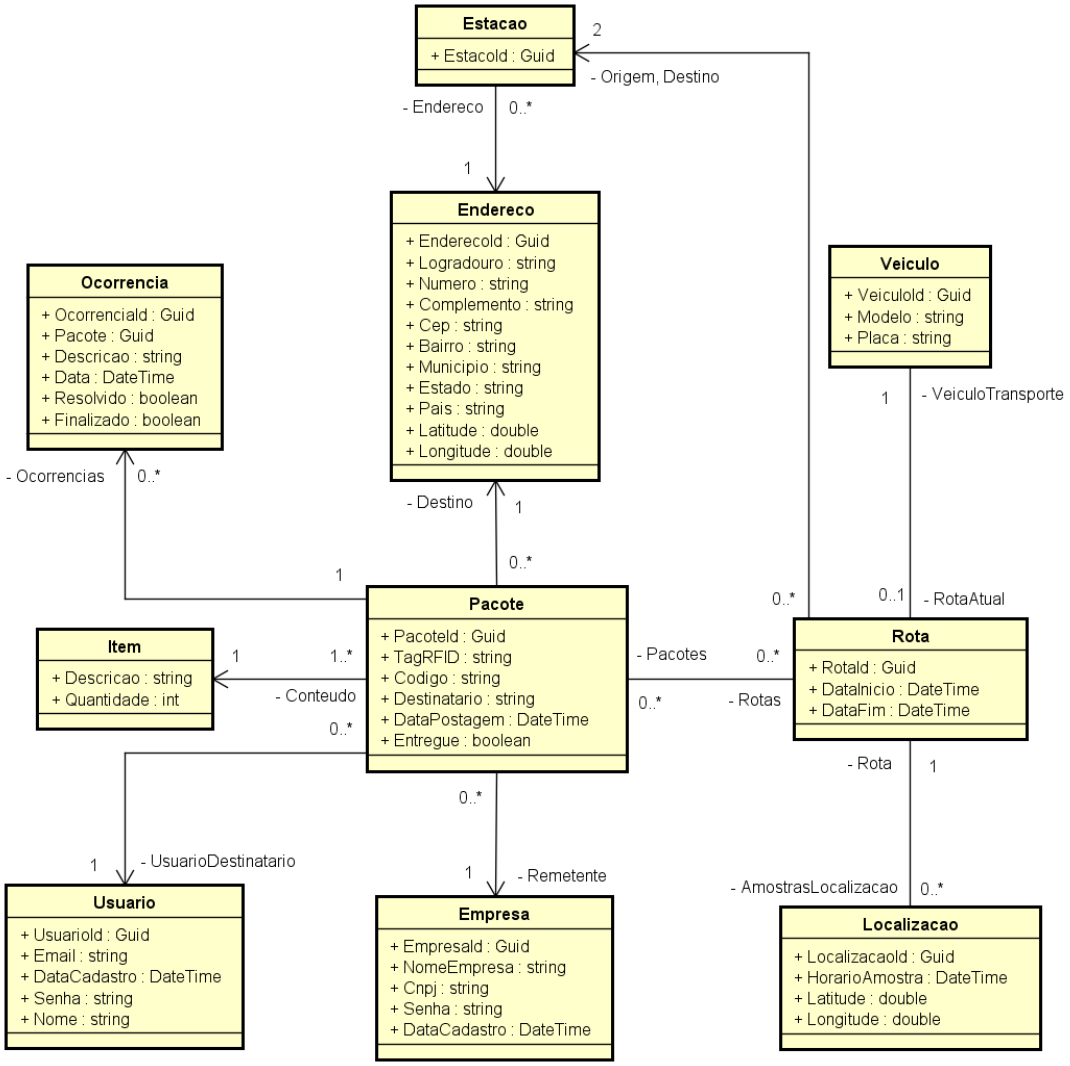


Figura 14 - Diagrama de classes para Web e API implementados. Fonte: autores

A principal diferença da arquitetura implementada em relação à teórica é quanto às amostras de localização - uma vez que é medido apenas a posição dos pacotes, não há necessidade de se implementar as propriedades físicas no sistema. Além disso, considerando que a amostra de posição de todos os pacotes dentro de um mesmo veículo serão as mesmas, apenas uma única entidade de rota é criada para tais pacotes (no sistema teórico, cada pacote teria sua rota separada, mesmo que compartilhando o veículo com outros pacotes).

Devido à separação de serviços e funções em dois servidores e dois bancos de dados, algumas das relações presentes no diagrama foram implementadas de maneira indireta; por exemplo, as classes *Pacote* e *Endereco* são salvas em bancos distintos - para relacionar o destino

do pacote com um endereço, então, o atributo *Destino* em *Pacote* é do tipo *Guid* que guarda o identificador da instância de *Endereco* correspondente. Classes relacionadas que compartilham o mesmo banco tem suas relações implementadas da forma usual.

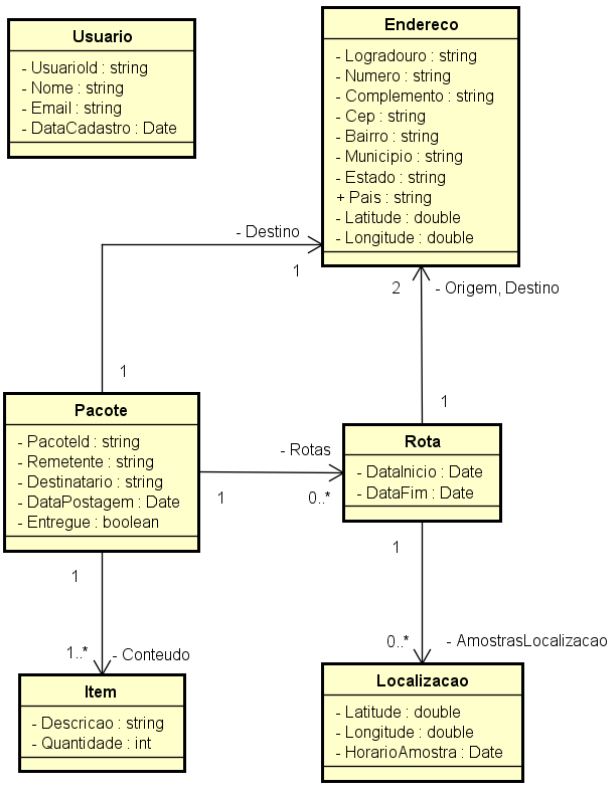


Figura 15 - Diagrama de classes para aplicativo Android. Fonte: autores

As mudanças na arquitetura de classes para mobile foram menores, removendo o envio de informações sobre ocorrências de pacotes, e a informação do tempo médio de duração de rotas entre duas estações específicas.

5 TESTES E RESULTADOS

5.1 PRIMEIRA MONTAGEM DE TESTES

Durante a disciplina de . O conjunto está descrito na figura 8 e utiliza um microcontrolador *Arduino Uno R3*, duas leitoras RFID de baixa potência e um aparelho celular com o sistema operacional *Android*. A utilização do celular permitiu que se conectasse o microcontrolador à rede para assim alcançar o servidor, além de poder utilizar o GPS do celular, dessa forma simulamos o conjunto do sistema embarcado de forma simplificada. Além disso, configuramos uma máquina na nuvem da AWS para ser o nosso servidor e plataforma Web, portanto configuramos o celular para enviar as informações transmitidas pelo microcontrolador para o endereço de rede dessa máquina. Essa montagem inicial busca simular apenas o que consideramos como o caminho crítico do projeto, portanto deixamos as funcionalidades de consulta do estado do pacote pelo destinatário e pela empresa para uma segunda etapa de implementação.

Antes de começar a desenvolver o *software*, a equipe se reuniu para definir a divisão das tarefas entre os membros e definimos padrões para as informações que cada componente deveria trocar. Por exemplo, o microcontrolador deve enviar a lista das *tags* que a leitora identificou anteriormente, mas que não foram detectadas nessa última leitura. Essa informação em seguida seria enviada por comunicação serial em um formato específico e acrescida das geocoordenadas obtidas pelo celular e enviadas à nuvem. Com essa padronização especificada, a divisão das tarefas entre os membros evitou a incompatibilidade entre componentes feitos por pessoas distintas e permitiu que os componentes possam ser alterados sem que haja uma necessidade de se alterar os outros componentes do sistema, desde que o novo componente seja compatível com o método de comunicação previamente utilizado. Tal isolamento entre as tecnologias adotadas entre os componentes também possibilitou que ao elaborarmos a próxima etapa de implementação não fosse necessário um grande esforço para adaptarmos aquilo que já foi feito nessa primeira etapa.

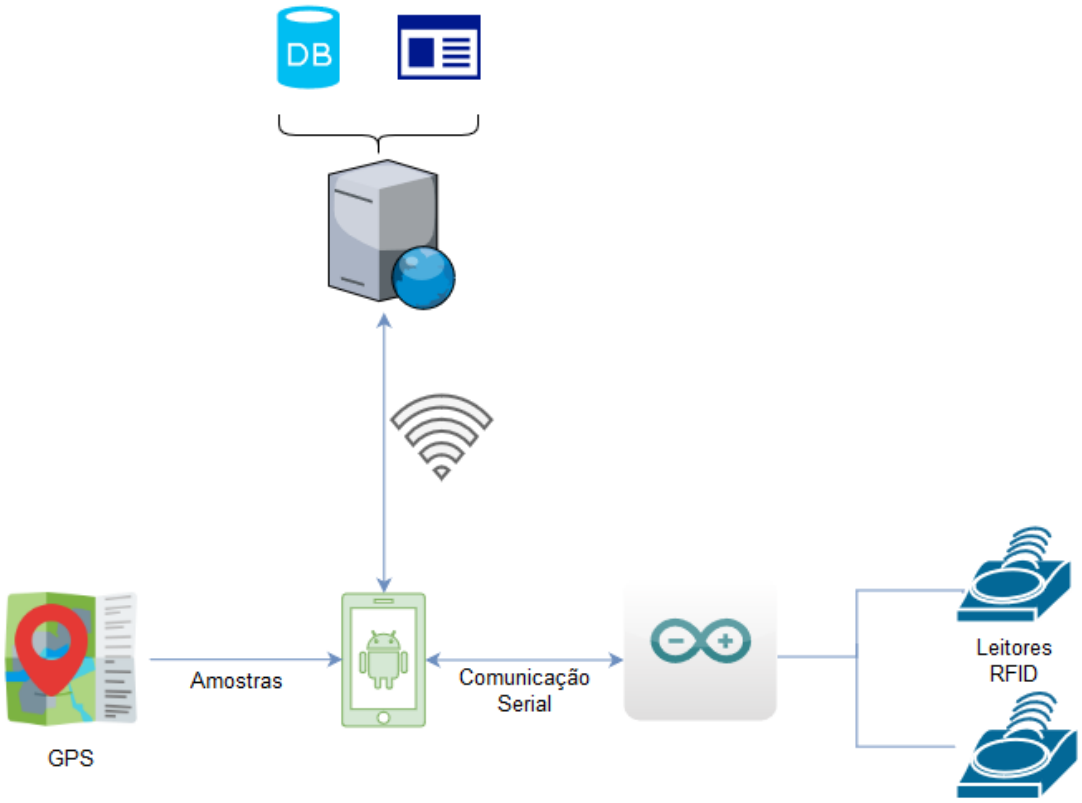


Figura 16 - Esquema ilustrativo da primeira montagem para testes. Fonte: autores

Com o trabalho dividido entre os membros, coube à cada um realizar os testes necessários para assegurar que o funcionamento daquele componente do sistema estaria correto e que ele seria capaz de enviar dados para os outros componentes seguindo o padrão definido. Desse modo, pode-se verificar que o *Arduino* era capaz de acionar as leitoras e recuperar as duas *tags* lidas, e que essa informação era lida pelo celular e acrescida das coordenadas obtidas pelo GPS e enviadas para o servidor. Já no lado do servidor enviamos requests que simulavam as informações dos pacotes e vimos que elas eram armazenadas corretamente no banco de dados. Nos testes das leituras das *tags* nos foi constatado o problema das leitoras, pois elas não estavam realizando as leituras corretamente em alguns casos. Uma investigação mais profunda revelou que isso ocorria principalmente quando a mesma *tag* era lida múltiplas vezes pela mesma leitora, e que a leitora armazena a identificação da *tag* em um registrador e caso o valor armazenado seja igual ao que foi lido, a leitora ignora o valor recebido e indica que não há uma *tag* presente. Isso se deve ao caso de uso mais comum das leitoras RFID em que uma *tag* é aproximada da leitora por um curto período de tempo e depois é retirada. Para isso ser contornado foi necessário procurar na biblioteca de instruções da leitora por um comando que removesse o valor armazenado no registrador.

5.2 SEGUNDA MONTAGEM DE TESTES

Com os testes concluídos na montagem descrita anteriormente, foi definido que o próximo passo seria realizar testes em um cenário mais próximo de como seria na aplicação real, portanto adicionamos um *shield* GPS e 3G ao *Arduino* e retiramos a necessidade do celular ser o intermediário entre o *Arduino* e o servidor. Além disso, as leitoras RFID utilizadas anteriormente foram substituídas por uma antena RFID UHF e uma leitora RFID; com tal composição é possível cobrir uma área maior e detectar um número considerável de tags simultaneamente. Com a utilização dessa nova leitora tornou-se necessária a utilização de uma API fornecida pelo fabricante para podermos realizar a interface entre o *Arduino* e a leitora. Ao tentar iniciar a integração, foi constatada a deficiência do modelo *Uno R3* descritas anteriormente, o que tornou necessário a utilização de um outro componente entre a leitora e o *Arduino* para conseguirmos transmitir as *tags* lidas. Verificada a compatibilidade da API com uma placa *Raspberry Pi*, essa foi inserida na montagem.

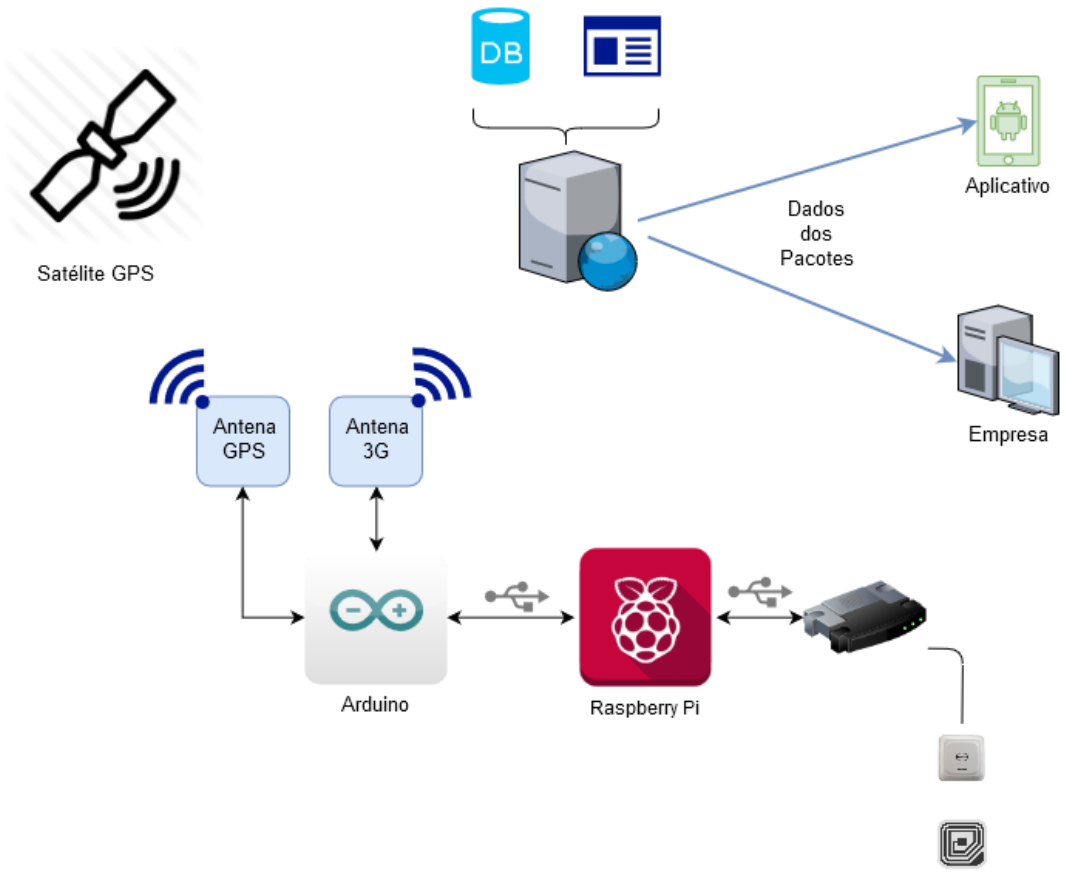


Figura 17 - Esquema ilustrativo da segunda montagem para testes. Fonte: autores

Com isso em mente, foi criado um *software* básico em *Java* e realizados testes de leitura utilizando a API em um laptop conectado a leitora, para em seguida os testes serem realizados utilizando a *Raspberry Pi*. Os testes seguintes exigiram um trabalho extra devido à necessidade de se adaptar o software anterior ao ambiente da *Raspberry*, pois como escolhemos a linguagem *Java* para a criação do *software*, não era possível compilar o código dentro da *Raspberry*. Portanto era enviado um executável já compilado para ela, mas devido à diferenças nas arquiteturas dos processadores do computador em que o *software* foi compilado e a da *Raspberry*, foi necessário procurar ajuda com os fabricantes da leitora. Com a assistência deles foi possível executar os testes com o *software* na *Raspberry*.

Uma vez que o *Raspberry Pi* se comunicava com a leitora RFID, e o *Arduino* realizava as conexões com GPS e internet, foi necessário estabelecer o método de comunicação entre as duas placas. Decidiu-se por realizar apenas a comunicação no sentido *Raspberry - Arduino*, considerando que o número de *tags* seria reduzido e portanto a quantidade de informação passada entre as placas seria pequena. Nota-se que em um cenário com várias *tags*, seria necessário realizar comunicação de duas vias, de modo a confirmar o recebimento de informação de uma placa para outra.

A placa *Arduino*, por sua vez, roda uma simples rotina que espera pelo recebimento das *tags* (no formato de *strings*) para então coletar dados de localização e enviar os dados ao servidor. Devido às limitações de capacidade do modelo *Uno R3* e do módulo SIM5230E, foi necessário criar uma segunda versão do serviço de recebimento de amostras de localização; no caso, essa nova versão recebe os dados em um formato mais reduzido para compactar o tamanho da mensagem enviada.

A montagem foi realizada no carro de um dos integrantes do grupo, para simular um veículo de transporte de carga, com duas *tags*; os equipamentos foram alimentados através de um inversor/transformador conectado ao carro. O tempo de amostragem configurado foi de 30s, controlado pelo *Raspberry Pi*. O percurso escolhido teve como origem o estacionamento da Escola Politécnica e destino a Avenida Professor Lineu Prestes, na altura 900.

Figura 18 - Inversor/transformador utilizado para alimentação do *hardware* no carro. Fonte: autores

Figura 19 - Montagem para testes; da esquerda para a direita: leitora RFID e antena, *Arduino* com módulo SIM5230E e antenas, *Raspberry Pi*. Fonte: autores

5.3 RESULTADOS

Após a realização dos testes, realizou-se uma comparação dos dados de coleta com o trajeto real:

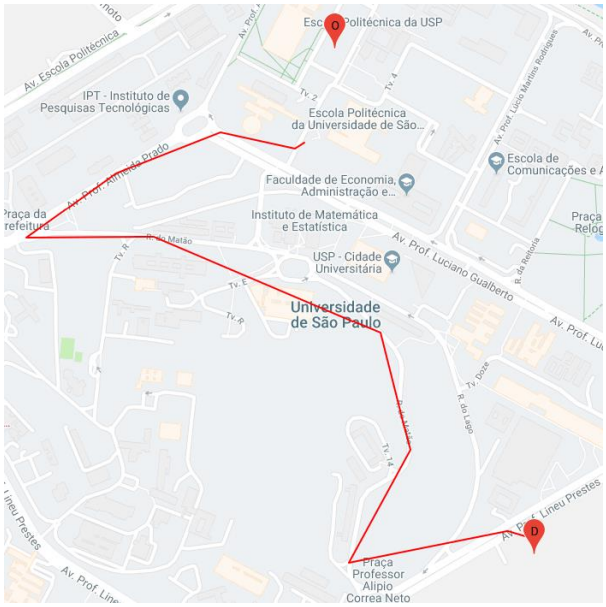


Figura 20 - Trajeto amostrado durante os testes da segunda montagem. Fonte: autores

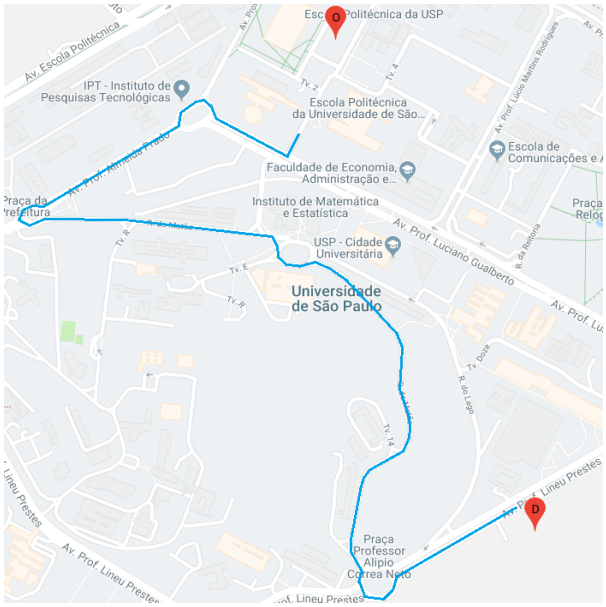


Figura 21 - Trajeto real percorrido durante os testes da segunda montagem. Fonte: autores

Figura 22 - Informações do pacote e trajeto amostrado pelos testes da segunda montagem no aplicativo *Android*. Fonte: autores

O aplicativo apresentou o resultado esperado; observa-se, no entanto, uma inconsistência não observada pelo sistema, uma vez que a rota traçada para o pacote vai na direção contrária de seu destino final.

Além da trajetória, verificou-se o registro de ocorrências pelo sistema, e foi o sucesso na indicação de falhas de leitura:

Pacote	TipoOCorrencia	Descricao	Data	Resolvido	Finalizado
0212e037-bce3-4731-bcad-a9a801795947	NaoEncontrado	Pacote não encontrado no veículo	2018-12-04 21:53:02	0	0
0212e037-bce3-4731-bcad-a9a801795947	NaoEncontrado	Pacote não encontrado no veículo	2018-12-04 21:51:43	0	0
0212e037-bce3-4731-bcad-a9a801795947	NaoEncontrado	Pacote não encontrado no veículo	2018-12-04 21:51:09	0	0
0212e037-bce3-4731-bcad-a9a801795947	NaoEncontrado	Pacote não encontrado no veículo	2018-12-04 21:50:33	0	0
NULL	NULL	NULL	NULL	NULL	NULL

Figura 23 - Tabela (SQL) de ocorrências, vista através do *MySQL Workbench*. Fonte: autores

Nota-se que foi um único pacote que foi marcado nas ocorrências, em quatro ocasiões com um intervalo próximo a 30s entre uma ocorrência e a próxima. A partir disso, pode-se inferir que uma das etiquetas esteve fora da área de cobertura da leitora RFID por cerca de 2min. Também foi observado o comportamento das amostras de localização, buscando informações sobre o período de amostragem:

HorarioAmostra	Latitude	Longitude	RotaId
2018-12-04 21:49:20	-23.557446	-46.731567	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:49:57	-23.557442	-46.731567	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:50:33	-23.557442	-46.731567	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:51:09	-23.557438	-46.731567	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:51:43	-23.557438	-46.731567	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:53:02	-23.557568	-46.731804	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:53:40	-23.557194	-46.733677	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:54:13	-23.558149	-46.736286	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:54:48	-23.55962	-46.738533	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:55:24	-23.559599	-46.735352	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:56:38	-23.561808	-46.729664	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:57:15	-23.564489	-46.728916	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:57:49	-23.567095	-46.730461	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 21:59:05	-23.56637	-46.726475	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:00:31	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:01:07	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:01:42	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:02:17	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:02:53	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:03:28	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:04:03	-23.566366	-46.726463	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:05:15	-23.566492	-46.726032	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:05:50	-23.566492	-46.726032	dfb58896-31b2-44d7-b027-a9ac0155d26c
2018-12-04 22:06:25	-23.566492	-46.726032	dfb58896-31b2-44d7-b027-a9ac0155d26c

Figura 24 - Tabela (SQL) de amostras de localização, vista através do *MySQL Workbench*. Fonte: autores

Pode-se observar que a maioria das amostras se separam umas das outras por um intervalo próximo a 30s; no caso, não há informações suficientes para determinar a causa específica dos atrasos ocorridos. Como primeira hipótese, há a perda de conexão 3G devido ao sinal fraco; como segunda hipótese, considera-se falha na comunicação entre as placas *Raspberry Pi* e *Arduino* ou ainda uma falha interna em qualquer uma das placas - para esse caso, muito provavelmente se trata de um erro de *software* ao encontrar um caso não antecipado, dada a intermitência da ocorrência de erros.

6 CONSIDERAÇÕES FINAIS

6.1 CONCLUSÕES DO PROJETO DE FORMATURA

Comparando os objetivos propostos no início desta monografia com o desenvolvimento da arquitetura do sistema e os resultados obtidos por meio dos testes, concluímos que o sistema criado e testado é capaz de realizar o rastreamento em tempo real de forma aceitável para o escopo dos testes que foram possíveis realizar, já que para uma aplicação real os requisitos levantados seriam mais exigentes e em consequência os testes deveriam cobrir uma variedade muito maior de cenários. Apesar do levantamento de diversos requisitos que abrangem os diferentes componentes da arquitetura, o projeto teve que se dedicar ao desenvolvimento do caminho crítico da aplicação mais do que foi planejado inicialmente, portanto a usabilidade das telas criadas, a criação do *dashboard* completo para as empresas, entre outros aspectos que não fazem parte do escopo da implementação final e dos testes, não foram desenvolvidos como havia sido inicialmente idealizado. Acreditamos que tal fato se deve a termos subestimado o trabalho necessário para se desenvolver o caminho crítico da aplicação, pois lidamos com diversos problemas de compatibilidade entre os componentes do sistema embarcado e outros componentes do software que necessitaram de muito tempo de trabalho para serem resolvidos.

Tendo em vista as as dificuldades encontradas na montagem e os resultados provenientes da segunda montagem de testes, é possível realizar críticas a algumas decisões de projeto tomadas ao longo do desenvolvimento:

- *Arduino Uno R3*: o modelo em questão se provou insuficiente para cumprir as expectativas iniciais, especialmente no quesito de capacidade - sua pequena memória se mostrou um empecilho significativo ao projeto, em um primeiro momento impossibilitando o uso desse para a comunicação com a leitora Edge-50, e posteriormente limitando a capacidade de processamento necessária para uma comunicação efetiva com os serviços do servidor. Uma mudança necessária seria a utilização de um modelo mais robusto, como o *Arduino Mega 2560*, o que permitiria inclusive descartar o *Raspberry Pi* da implementação.

- *Raspberry Pi*: a placa *Raspberry Pi* se mostrou mais versátil e prática no momento do desenvolvimento; no entanto, foi empregada apenas como “solução” às limitações do *Arduino* utilizado. Por fim, é uma opção viável como placa única do sistema, mas se mostra uma pior escolha no quesito custo.

6.2 COMENTÁRIOS DOS INTEGRANTES

De forma geral, o projeto serviu para que os alunos colocassem em prática os conhecimentos de desenvolvimento de software tanto nos aspectos arquiteturais quando de programação e implementação do software. A elaboração desse projeto deu uma visão mais prática do esforço necessário para desenvolver um projeto e de como lidar com trabalho em grupo, erros e dificuldades ao criar um projeto desde sua concepção.

A parte do projeto que gerou mais problemas e imprevistos foi a integração entre os dispositivos de hardware utilizados. Havia um erro de configuração da *Mercury API* que a princípio não permitiu o uso dela na *Raspberry Pi*; houve dificuldades para utilizar a conexão via internet com o *Arduino*, além de dificuldades genéricas para implementar a comunicação entre os dispositivos - nessa parte do projeto, tarefas que pareciam simples se tornaram problemas, em parte pela qualidade material de apoio e consulta aquém do esperado, dificultando o desenvolvimento do protótipo.

Após um pouco de autoanálise, os integrantes do grupo chegaram à conclusão de que parte do motivo das dificuldades enfrentadas foram a ausência de uma experiência prática anterior de elaboração de um projeto verossímil de início ao fim no escopo da área de computação e da pouca familiaridade do uso de sistemas embarcados para propósitos mais complexos. A escolha dos dispositivos também aumentou a complexidade do projeto, pois com o crescimento de dispositivos *IoT* poderia se encontrar soluções de dispositivos embarcados que já possuam interfaces de conexão com a internet e entre si nativamente; nesse caso, no entanto, a escolha de um dispositivo mais sofisticado poderia simplificar demasiadamente o projeto, prejudicando o exercício de engenharia esperado do projeto de formatura.

6.3 PERSPECTIVAS DE CONTINUIDADE

Comparando a especificação do sistema da seção 3, com o sistema implementado na seção 4, fica evidente que os aspectos que foram descritos mas não implementados poderiam compor outros trabalhos no futuro, principalmente em relação à utilização de *tags* RFID que possuem sensores. Explorar a utilização das *tags* capazes de pegar uma amostra de temperatura em um sistema que se comunica em tempo real com um servidor traria um controle de qualidade muito maior no transporte de produtos sensíveis a variações de temperatura, como vacinas e alimentos. Em uma outra frente, a possibilidade de se utilizar os dados gerados pelo rastreamento com algoritmos de *machine learning* para parametrizar o desempenho de uma cadeia logística e assim buscar uma otimização do processo também seria um desenvolvimento interessante para um projeto futuro.

Algo a ser considerado seria a substituição de algumas tecnologias que foram utilizadas na implementação por uma outra tecnologia capaz de realizar a mesma função mas cuja utilização possa resolver alguns problemas existentes na arquitetura atual. Por exemplo, podemos considerar a utilização de dispositivos Bluetooth Zigbee, aonde os pacotes passariam a possuir um pequeno sensor que estabeleceria uma conexão com sensores instalados dentro do veículo e assim realizar o rastreamento. Tal mudança seria interessante para as empresas já que a detecção de um sensor Bluetooth não depende de fatores como o ângulo entre ele e a antena, como ocorre com RFID, mas agora os sensores *Bluetooth* são ativos e, portanto, dependem de uma fonte de energia, além de serem individualmente mais caros que as *tags* RFID. Graças a esses pontos acreditamos que um outro trabalho envolvendo uma análise dessa implementação seria interessante para o desenvolvimento dos sistemas de rastreamento.

Lista de Referências

ANDROID DEVELOPERS. **Sobre a Plataforma.** 2018. Disponível em: <https://developer.android.com/>. Acesso em: 21 mai. 2018.

ARDUINO. **Introduction**. 2018. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em 21 mai. 2018.

FINKENZELLER, Klaus. **RFID handbook: fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication**. John Wiley & Sons, 2010. Disponível em: <https://books.google.com.br/books?hl=pt-BR&lr=&id=jAszZEQYa9wC&oi=fnd&pg=PT6&dq=RFID&ots=2Jmyni_UMk&sig=NL1L7dr2GtuvqeDGUdloUSS4-lA#v=onepage&q=RFID&f=false>. Acesso em: 13 mai. 2018

GODFREY, Bill. **A Primer on Distributed Computing**. 2002. Disponível em: <http://billpg.com/bacchae-co-uk/docs/dist.html>. Acesso em: 20 jul. 2018

AL-FUQAHA, Ala et al. Internet of things: A survey on enabling technologies, protocols, and applications. **IEEE Communications Surveys & Tutorials**, v. 17, n. 4, p. 2347-2376, 2015. Disponível em: <https://ieeexplore.ieee.org/abstract/document/7123563>. Acesso em: 12 jul. 2018

MATTERN, Friedemann; FLOERKEMEIER, Christian. From the Internet of Computers to the Internet of Things. In: **From active data management to event-based systems and more**. Springer, Berlin, Heidelberg, 2010. p. 242-259. Disponível em: <http://www.vs.inf.ethz.ch/publ/papers/Internet-of-things.pdf>. Acesso em: 12 jul. 2018

CHASE, Otavio; ALMEIDA, F. J. Sistemas embarcados. **Mídia Eletrônica.** Disponível em: <http://www.lyfreitas.com.br/ant/pdf/Embarcados.pdf>. v. 10, n. 11, p. 13, 2007. Acesso em: 22 nov. 2018

ACURA; **Guia de Usuário - EDGE-50 TCPIP**. 2014. Disponível em: <http://www.rfidsystems.com.br/pdf/100.255_EDGE-50_TCPIP_Guia_Usuario_pr_pt_rev6.pdf> Acesso em: 22 nov. 2018. Rev.1

THINGMAGIC; **Mercury API Programmer's Guide**. Disponível em: <https://www.jadaktech.com/download-product-pdf-sf?download_file=https://www.jadaktech.com/wp-content/uploads/2018/02/MercuryAPI_ProgrammerGuide_for_v1.27.3.pdf&prodID=14668>. Acesso em: 22 nov. 2018. v1.27.3

CONFEDERAÇÃO NACIONAL DO TRANSPORTE; **Boletim Estatístico - CNT - Maio 2018.** Disponível em: <http://cms.cnt.org.br/Imagens%20CNT/BOLETIM%20ESTAT%C3%8DSTICO/BOLETIM%20ESTAT%C3%8DSTICO%202018/Boletim%20Estati%CC%81stico%20-%2005%20-%202018.pdf>. Acesso em: 27 nov. 2018.

ECOMMERCE FOUNDATION; **Latin America Ecommerce Report 2018.** Disponível em: <https://www.ecommercewiki.org/reports/742/latam-ecommerce-report-2018>. Acesso em: 27 nov. 2018

YIN, Jun et al. A system-on-chip EPC Gen-2 passive UHF RFID tag with embedded temperature sensor. **IEEE Journal of Solid-State Circuits**, v. 45, n. 11, p. 2404-2420, 2010. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5593894>.Acesso em: 1 dez. 2018

VAZ, Alexander et al. Full passive UHF tag with a temperature sensor suitable for human body temperature monitoring. **IEEE Transactions on Circuits and Systems II: Express Briefs**, v. 57, n. 2, p. 95-99, 2010. Disponível em: <https://ieeexplore.ieee.org/abstract/document/5411829>. Acesso em: 1 dez. 2018

NXP; **MFRC Standard Performance Mifare and NTAG Frontend**. 2016. Disponível em: <https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf>. Acesso em: 13 mai. 2018. Rev. 3.9

TINY SINE; **3G/GPRS/GSM Shield Datasheet**. 2017. Disponível em: <https://www.tinyosshop.com/datasheet/3G%20Shield%20Datasheet.pdf>. Acesso em: 22 nov. 2018. v1.1