UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO
UNIDADE ACADÊMICA DE GARANHUNS

JOSÉ KELLISON DE ALMEIDA SILVA

SISTEMA DE GESTÃO PARA LOCALIZAÇÃO INDOOR UTILIZANDO WIFI FINGERPRINTS E MACHINE LEARNING

Garanhuns
2019

JOSÉ KELLISON DE ALMEIDA SILVA

SISTEMA DE GESTÃO PARA LOCALIZAÇÃO INDOOR UTILIZANDO WIFI FINGERPRINTS E MACHINE LEARNING

Trabalho de Conclusão de Curso submetido à Universidade Federal Rural de Pernambuco - Unidade Acadêmica de Garanhuns, como requisito necessário para obtenção do grau de Bacharel em Ciência de Computação, sob a orientação do Prof. Dr. Alixandre Thiago Ferreira de Santana e coorientação do Prof. Dr. Luis Filipe Alves Pereira.

Garanhuns
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Ariano Suassuna, Garanhuns-PE, Brasil

S586s Silva, José Kellison de Almeida
Sistema de gestão para localização *indoor* utilizando *Wifi Fingerprints* e *Machine Learning* / José Kellison de Almeida Silva. - 2019.
62 f. ; il.

Orientador: Alixandre Thiago Ferreira de Santana.
Cooorientador: Luis Filipe Alves Pereira.
Trabalho de Conclusão de Curso (Graduação em Ciência da Computação)-Universidade Federal Rural de Pernambuco, Departamento de Ciência da Computação, Garanhuns, BR-PE, 2019.
Inclui referências.

1. Sistema de localização *indoor* 2. Aprendizado do computador 3. Redes locais sem fio I. Santana, Alixandre Thiago Ferreira de, orient. II. Pereira, Luis Filipe Alves, coorient. III. Título

CDD 004

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

JOSÉ KELLISON DE ALMEIDA SILVA

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação, sendo aprovado em sua forma final pela banca examinadora:

Orientador: Prof. Dr. Alixandre Thiago Ferreira de Santana
Universidade Federal Rural de Pernambuco - UFRPE

Coorientador: Prof. Dr. Luis Filipe Alves Pereira
Universidade Federal Rural de Pernambuco - UFRPE

Prof. Dr. Tiago Buarque Assunção de Carvalho
Universidade Federal Rural de Pernambuco - UFRPE

Garanhuns, 17 de julho de 2019

Dedico este trabalho, em especial, à minha família por me apoiarem e fazerem o possível para eu ter a chance de conseguir ingressar e permanecer em um curso de graduação.

AGRADECIMENTOS

Sou grato a minha mãe e ao meu pai que fizeram tudo que estava ao seu alcance para que eu pudesse ter a chance de ter uma boa educação e oportunidades que eles nunca tiveram.

Aos meus avôs e avós que infelizmente não puderam ver a conclusão dessa jornada, mas que sempre me ajudaram.

Ao meu orientador, o professor Alixandre Thiago, que me orientou durante a construção deste trabalho, agradeço por todo o incentivo e pela oportunidade que me foi dada de trabalhar no projeto que deu origem aos resultados aqui descritos.

Ao professor Luis Filipe, que aceitou a tarefa de auxiliar e de ser um guia para que o projeto tivesse bons resultados. Também ao professor Tiago Buarque por ajudar na avaliação deste trabalho, compondo a banca.

Aos meus amigos Allyson, Marrone e Sérgio que conheço há muitos anos, eles que sempre me incentivaram a seguir em frente e me faziam levantar todas as manhãs para trabalhar.

Aos meus amigos Daniel, Eduardo, Isabelle que me acompanharam desde o inicio da graduação e que foram muito importantes durante este percurso. Agradeço, em especial, a Witássio, um amigo que sempre esteve presente e disposto a ajudar não somente à mim mas a todos que necessitavam de auxílio em determinadas disciplinas.

Aos meus amigos Mariana, Júlio César, Juan, Ane, Sandy e Júlio Monteiro, com os quais dividi o mesmo teto por tantos anos e que fizeram meus dias mais alegres e menos silenciosos.

Agradeço também a Fátima, Ciete e Hellen que atuaram como mães durante o período que residi na CAESA (Casa do Estudante de Sanharó).

Finalmente, agradeço a todos que de alguma maneira contribuíram tanto para meu crescimento pessoal quanto acadêmico em todos esses anos na instituição.

Mera mudança não é crescimento.
Crescimento é a síntese de mudança e
continuidade, e onde não há continuidade
não há crescimento.
(C.S. Lewis)

RESUMO

Soluções de software que são dependentes do sinal do Sistema de Posicionamento Global (GPS) podem não apresentar uma boa precisão em espaços fechados ou *indoor* (shoppings, aeroportos, complexos comerciais etc.), pois as variações do sinal são capazes de inviabilizar seu uso. Este trabalho tem como objetivo implementar uma solução de gestão de localização *indoor* utilizando sinais de redes sem fio locais para predizer a posição de um usuário. Para tanto, a solução utiliza WiFi *fingerprints* captados por aparelhos celulares com sistema operacional Android, para construir bases de instâncias representando posições reais de um usuário coletadas em um dos prédios da Universidade Federal Rural de Pernambuco - Unidade Acadêmica de Garanhuns. O aplicativo móvel determina a posição do usuário e uma ferramenta web de gestão de localidades de produtos permite a inserção de pontos de interesse do usuário no mapa *indoor*. Para completar o processo, o aplicativo móvel realiza a plotagem de um vetor da posição predita do usuário até a posição de um objeto específico de destino compondo assim uma solução completa e funcional de rotas *indoor*. Os algoritmos utilizados para predição da posição do usuário foram o *random forest*, *multi layer perceptron* e Adaboost. Ao final dos experimentos, o melhor resultado de localização *indoor* foi obtido com o Adaboost, apresentando um erro médio de pouco menos de 1 metro em relação à posição real e 98,64% dos resultados se apresentando dentro de uma margem de erro aceitável (até 2 metros).

Palavras-chave: Posicionamento *indoor*. Aprendizagem de máquina. *Wifi fingerprints*

ABSTRACT

Software solutions that are dependent on the Global Positioning System (GPS) signal may not display good accuracy in indoor spaces (malls, airports, commercial complexes, etc.), as the signal variations are likely to make it unreliable. This work aims to implement an indoor location management solution using signals from local wireless networks to predict the position of a user. To do so, the solution uses WiFi fingerprints, captured by mobile devices with Android operating system, to build bases of instances representing real positions of a user collected in one of the Federal Rural University of Pernambuco - Academic Unit of Garanhuns (UFRPE - UAG) buildings. The mobile application determines the user's position and a product location management web tool allows the insertion of user points of interest on the indoor map. To complete the process, the mobile application plots a vector from the predicted position of the user to the position of a specific target object, thus composing a complete and functional solution of indoor routes. The algorithms used to predict the position of the user were random forest, multi layer perceptron and Adaboost. The best indoor localization result was obtained with Adaboost, presenting an average error just under 1 meter from the actual position and 98.64% of the results were under an acceptable margin of error (up to 2 meters).

Keywords: Indoor positioning. Machine Learning. Wifi fingerprints.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O GPS (Sistema de Posicionamento Global, em português) é um meio de navegação baseado em satélites (EL-RABBANY, 2006), um dos mais usados para a geolocalização de objetos e pessoas. Apesar do seu alcance ser excelente, ele apresenta problemas para realizar o posicionamento com uma precisão aceitável dentro de espaços fechados, uma vez que a sua utilização requer *line of sight* (LOS, ou linha de visão) dos satélites (SHALA; RODRIGUEZ, 2011), como é o caso em shoppings, aeroportos, complexos comerciais etc. Outros métodos são usados em conjunto para tentar sanar a deficiência do sinal GPS, utilizando diversos tipos de sinais existentes localmente em um ambiente e em dispositivos portáteis, como por exemplo: *Bluetooth*, WiFi, dados de giroscópio, sinais emitidos por torres de rádio etc.

Em muitos estabelecimentos comerciais, onde há uma grande variedade de produtos, pode ser uma tarefa difícil para um cliente conseguir encontrar um determinado produto dentro de uma área extensa. Ampliando a visão da situação, o problema pode não ser exclusivo do lado dos clientes de uma empresa, ele pode ser estendido para a gestão dos produtos em um desses estabelecimentos. Um gestor pode encontrar dificuldade em administrar a disposição de seus produtos expostos aos visitantes, ou ainda os próprios funcionários podem ter dificuldades em encontrar um determinado objeto desejado por um cliente.

Devido a falta de confiabilidade do GPS em espaços fechados, é indispensável o uso de outras tecnologias para tentar chegar à uma precisão dentro de padrões aceitáveis, onde não haja grandes distorções da posição real dos objetos de um local. Contudo, devido ao grande número de tecnologias existentes, é necessário o estudo destas para escolha de uma que melhor atenda as condições dispostas: estrutura requerida, nível de precisão da geolocalização e custo.

Existe uma gama de métodos que visam tentar predizer a posição de um dispositivo com base em sinais de rede sem fio, como por exemplo: detecção de proximidade (FARID et al., 2013), técnica baseada em ângulo (AOA, do inglês *Angle Of Arrival*) (COSTA, 2014), técnica do menor polígono (COSTA, 2014), *fingerprinting* (ALTINTAS; SERIF, 2011; ZEGEYE et al., 2016) entre outros. Uma das soluções mais utilizadas na literatura é o uso de WiFi *fingerprints*, em outras palavras, o uso de sinais de redes WiFi recebidos por um dispositivo em uma determinada posição dentro de um espaço. Tais sinais são coletados em pontos de um espaço e armazenados em uma base de dados, durante uma primeira etapa para posteriormente serem usados com algoritmos de aprendizagem de máquina. Neste trabalho selecionamos alguns algoritmos para testes de acurácia, sendo eles: *Random Forest*, MLP (*Multilayer Perceptron*) e Adaboost.

O uso do sinais WiFi apresentam uma variância de predição entre 1 e 5 metros a depender da técnica utilizada, sendo que o uso de *fingerprints* é um dos com maiores níveis de

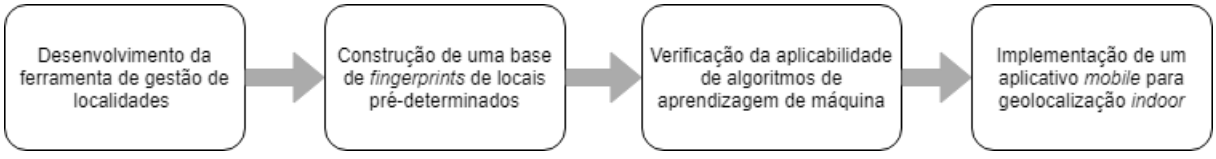
acurácia dentro desse intervalo (FARID et al., 2013). Além da acurácia apresentada pela literatura, outro fator que colabora para sua adoção é que na maioria dos lugares já existe uma rede com dispositivos WiFi instalados e prontos para uso. Deste modo, não há o custo de construir uma infraestrutura diferente para utilização de sistemas de geolocalização. Além do fato de que praticamente todo dispositivo móvel na atualidade é capaz de conectar-se à uma rede WiFi, o que implica na possibilidade de detecção desses sinais por um aplicativo para o uso em métodos de geolocalização. Para realizar a estimativa da posição atual de um dispositivo *mobile* pode-se adotar o uso de algoritmos de aprendizagem de máquina. Muitos trabalhos implementam algoritmos comumente usados para classificação (SHALA; RODRIGUEZ, 2011; FARID et al., 2013; ALTINTAS; SERIF, 2011), como o KNN e o *K-Means*. Entretanto, a análise do problema mostra-se mais próxima de um problema de regressão, uma vez que valores numéricos reais tentam ser preditos, por isso buscou-se validar os resultados de algoritmos de regressão, como os algoritmos Adaboost e MLP que são utilizados para experimentação neste trabalho.

Neste sentido, o objetivo geral deste trabalho é o desenvolvimento de uma solução de geolocalização *indoor* baseada no uso de algoritmos de *machine learning* e WiFi *fingerprints*. Os objetivos específicos estão listados abaixo:

- Desenvolver componente para gestão das posições de itens localizados no interior de um ambiente;
- Construir a base de dados de posições pré-determinadas com os sinais de rede em cada um desses ponto;
- Verificar a aplicabilidade de técnicas de aprendizagem de máquina;
- Implementar um aplicativo móvel para localização *indoor*.

A Figura 1 mostra o fluxograma dos objetivos específicos que regem o desenvolvimento deste trabalho. O detalhamento de como cada fase foi realizado e os resultados de cada uma são encontrados nos capítulos Capítulo 3 e Capítulo 4.

Figura 1 – Fluxograma dos objetivos específicos do trabalho



Fonte: O autor

Este trabalho está organizado na forma de cinco capítulos. O **Capítulo 2** apresenta a fundamentação teórica dos conceitos utilizados durante o desenvolvimento do sistema completo, incluindo tanto a plataforma web quanto a móvel, bem como os trabalhos correlatos. O **Capítulo 3**

apresenta a caracterização da pesquisa e a metodologia utilizada, o detalhamento as tecnologias utilizadas para construção do sistema, os requisitos para funcionamento dos componentes web e *mobile*, além de explicar o procedimento utilizado para coleta de dados e traz também um diagrama explicando a conexão entre os componentes. O **Capítulo 4** traz um detalhamento do resultado alcançado para cada um dos objetivos específicos que foram listados acima. O **Capítulo 5** apresenta as conclusões e trabalhos futuros que podem ser realizados a partir deste projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos básicos que foram utilizados para concepção do projeto. A seção 2.1 apresenta as variedades dos tipos de sistemas de localização existentes. Na seção 2.2 estão apresentados os conceitos de aprendizagem de máquina que foram utilizados no processo de escolha do método de predição que mais se adequava e que apresentava valores aceitáveis para o objetivo da aplicação. A seção 2.3 traz alguns trabalhos correlatos.

2.1 TÉCNICAS DE GEOLOCALIZAÇÃO

2.1.1 Sistema de Posicionamento Global - GPS

Assim como apresentado por El-Rabbany (2006), o Sistema de Posicionamento Global (ou GPS, sua sigla em inglês) é um sistema que funciona através do conjunto de 24 satélites que orbitam o planeta, sendo dispostos em grupos de quatro em seis planos orbitais da Terra. O GPS está atuante desde o início da década de 1990, tendo atingido potencial total no ano de 1995, fornecendo informações precisas e contínuas de posição tridimensionais em escala global para qualquer aparelho compatível (KAPLAN; HEGARTY, 2005). Inicialmente, foi criada para uso militar, contudo hoje é utilizado também por civis nos mais diversos serviços, como na navegação *outdoor* e no design de mapas.

Segundo Kaplan e Hegarty (2005), o GPS é dividido em três segmentos: controle terrestre, usuários e espaço. O segmento de controle terrestre é o responsável por realizar a manutenção dos transmissores (satélites) que estão em órbita, garantindo o seu funcionamento. O segmento de usuários é onde se enquadra quaisquer dispositivos que possam receber o sinal enviado por um satélite, sendo que esses dispositivos funcionam de forma totalmente passiva, apenas recebendo informações. Por fim, o segmento de espaço engloba a constelação de satélites em órbita do planeta.

Kaplan e Hegarty (2005) também explica que para ser possível obter a posição de um receptor na superfície do planeta, é necessário que ele esteja ao alcance do sinal de quatro satélites (no mínimo). Devido a este requerimento, o design da constelação de satélites foi construído para que seis satélites sempre estejam visíveis por um usuário. A posição obtida é calculada através do método de trilateração (SHALA; RODRIGUEZ, 2011).

Apesar desse meio de geolocalização possuir uma precisão bastante aceitável em ambiente abertos (*outdoor*), ele possui limitações que dificultam o uso confiável dentro de ambientes *indoor*. Como dito por Shala e Rodriguez (2011) e mencionado anteriormente neste trabalho, o GPS necessita que o usuário esteja em um campo que possibilite o alcance direto do sinal oriundo dos satélites que compõem o sistema, ou seja, é necessário que exista uma linha de visão (LOS,

do inglês *line of sight*). Sendo assim, o uso para sistemas de geolocalização *indoor* podem não apresentar resultados aceitáveis se utilizassem o GPS única como base de funcionamento.

2.1.2 Baseadas em Redes sem Fio

Sistemas baseados em redes sem fio utilizam os sinais de rádios provindos de aparelhos transmissores presentes em um ambiente para estimar a posição de um aparelho que recebe esses sinais (COSTA, 2014; ZEGEYE et al., 2016). Como apresentado por Altintas e Serif (2011), os algoritmos disponíveis para geolocalização *indoor* com uso de redes sem fio podem ser divididos em dois grupos gerais: *(a)* através de modelos matemáticos, como por exemplo, trilateração, que busca determinar a distância ou ângulo de três transmissores em relação ao receptor de sinais; ou, *(b)* através do mapeamento de locais, com o armazenamento de sinais em posições pré-definidas para uso posterior por algoritmos, como é o caso do uso do objeto de estudo principal deste trabalho, WiFi *fingerprints*.

Aparelhos que são usados para a transmissão de sinais podem ser dos mais diversos, dado os tipos de rede sem fio que existem, como por exemplo: WiFi utiliza-se de APs (do inglês *access points* ou pontos de acesso) que são roteadores, switches ou mesmo outros dispositivos móveis; ou Bluetooth, que utiliza *beacons*. Os sinais emitidos por esses transmissores são medidos através do indicador de intensidade de sinal recebido (*Received Signal Strength Indicator* ou RSSI (BENKIC et al., 2008; SHALA; RODRIGUEZ, 2011).

Detecção de Proximidade

Um dos métodos mais simples utilizados para geolocalização *indoor* é o método de detecção de proximidade. Esta técnica utiliza áreas (células) que possuem aparelhos transmissores de sinais, chamados de *beacons*, que identificam quando um aparelho móvel está dentro do seu alcance de sinal. Nos casos em que um ou mais *beacons* detectam um dispositivo dentro da sua área, aquele com o maior sinal de intensidade será o identificador da posição do dispositivo. Deste modo, é usado um algoritmo de vizinho mais próximo com os sinais que em que um dispositivo está ao alcance. Este tipo de detecção baseia-se em *beacons* que podem utilizar sinais de Bluetooth, RFID (identificador de rádio frequência), IR (radiação infravermelho), entre outros (FARID et al., 2013).

Fingerprinting ou Mapa de Sinais

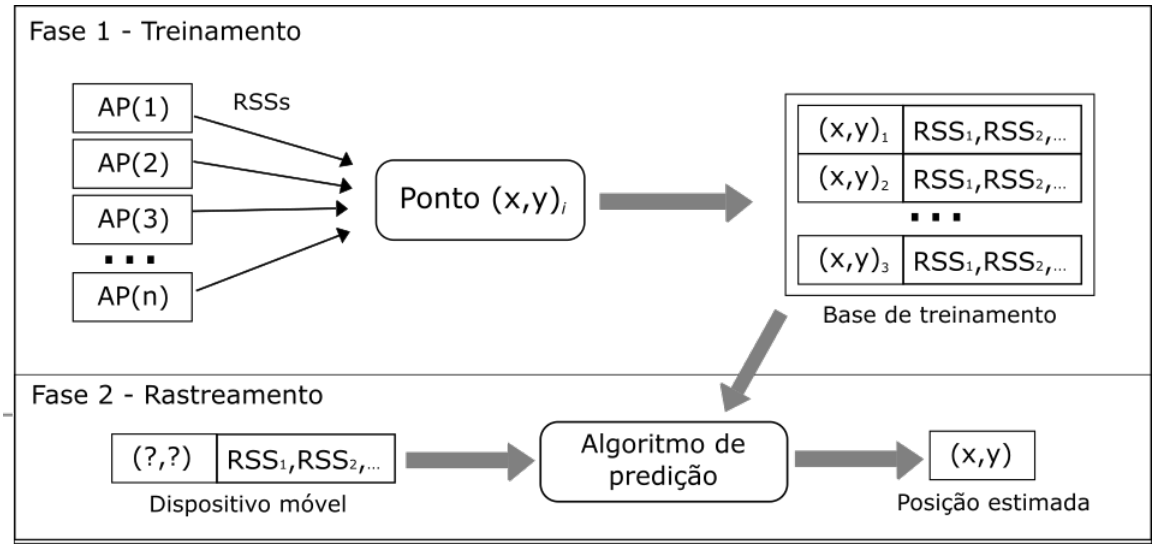
Esta é uma técnica que utiliza um vetor RSSI de múltiplos transmissores para formar um **mapa de sinais** (ALTINTAS; SERIF, 2011; ZEGEYE et al., 2016). *Fingerprints* funcionam como uma assinatura de um local, onde um conjunto de intensidades de sinais de redes são alcançáveis (COSTA, 2014). Cada indicador é associado ao identificador do aparelho que o transmite, o BSSID (*Basic Service Set Identifier*), sendo este o seu endereço do MAC. Os indicadores emitidos por dispositivos WiFi são medidos em dB (decibéis), variando entre 0 e

-100, sendo que quanto maior o valor melhor a qualidade do sinal o que significa uma maior aproximação com do dispositivo *mobile* (receptor) com o dispositivo transmissor. O processo utilizado neste método é subdividido em duas fases bem definidas: treinamento e rastreamento.

A Figura 2 apresenta o funcionamento do método que utiliza WiFi *fingerprints*, ou seja, os sinais emitidos por dispositivos de redes WiFi. A **Fase 1** do processo se dá pela **fase de treinamento** (também chamada de fase *offline* ou **calibração**), na qual, dado o mapa de um espaço, os RSSIs oriundos de APs são coletados em pontos pré-definidos (no formato de coordenadas [x,y]) e então armazenados em uma base para posterior treinamento. Situações que geram mudanças no ambiente (troca mudança de posição de *access points*, por exemplo) requerem que um novo treinamento seja realizado.

A etapa seguinte é a **fase de rastreamento** (*online*), nela os RSSIs detectados pelo dispositivo do usuário são coletados e usados como entrada para um algoritmo de aprendizagem de máquina, de modo que não há o conhecimento prévio da sua posição. O algoritmo usa então os dados coletados na primeira fase do processo para treinamento e os dados coletados na segunda fase, para gerar como saída valores estimados das coordenadas x e y que representam a posição atual do aparelho móvel.

Figura 2 – Fluxograma das fases que compõem o método de localização usando WiFi *fingerprints*



Fonte: Adaptado de Altintas e Serif (2011)

2.2 APRENDIZAGEM DE MÁQUINA

Aprendizagem de Máquina (*Machine Learning* ou ML) é um campo da ciência da computação que visa estudar métodos com os quais máquinas possam ser capazes de aprender baseando-se na observação de dados existentes, extraindo padrões e realizando predições, de modo a tentar se assemelhar a inteligência humana (RÄTSCH, 2004; NAQA; MURPHY, 2015). Abaixo seguem algumas definições de termos usados neste trabalho (REZENDE, 2003):

- Exemplo, padrão ou instância: conjunto de dados (ou vetores de características) que representa um objeto do mundo real, que serão usados por um modelo em algum método de aprendizagem de máquina.
- Característica, atributo ou *feature*: descrições de uma instância, informação específica de um objeto, podendo assumir valores numéricos, categóricos etc.
- Classe: atributo especial existente na categoria de algoritmos supervisionados, de modo que ele indica uma característica diferenciada permitindo o agrupamento com mais padrões.
- Conjunto de treinamento: conjunto de instâncias compostas por *n* características e *m* classes, usado para o treinamento de algoritmos de aprendizagem de máquina.
- Conjunto de teste: conjunto de instâncias que não apresentam classes (ou estas são desconsideradas) e que são usados para predição da sua classe com base nas informações da base de treinamento.
- *Outlier*: padrão que que se difere muito dos outros da sua base.

Na literatura, existem vários algoritmos que são utilizados nesse campo de pesquisa, os quais podem ser divididos em duas categorias distintas:

- **Supervisionados:** os conjuntos de dados de treino utilizados por estes algoritmos contêm uma classe para as instâncias, de modo que este é o resultado esperado quando um novo exemplo com classe desconhecida é processado pelo algoritmo (NAQA; MURPHY, 2015). Nesta categoria podemos citar os seguintes algoritmos que são descritos nessa pesquisa: *Random Forest*, MLP (*Multilayer Perceptron*) e Adaboost. Os algoritmos supervisionados podem ser utilizados resolver problemas que se dividem em dois tipos: **classificação**, onde o conjunto de instâncias pode ser divididos em classes categóricas; ou **regressão**, onde as instâncias possuem classes numéricas, por exemplo, para predição de preços de imóveis.
- **Não supervisionados:** diferente dos métodos supervisionados, os dados utilizados aqui não possuem rótulos. Os métodos que se encaixam nesta categoria são utilizados para identificar padrões ou anomalias através da identificação de *clusters*, por exemplo (NAQA; MURPHY, 2015).

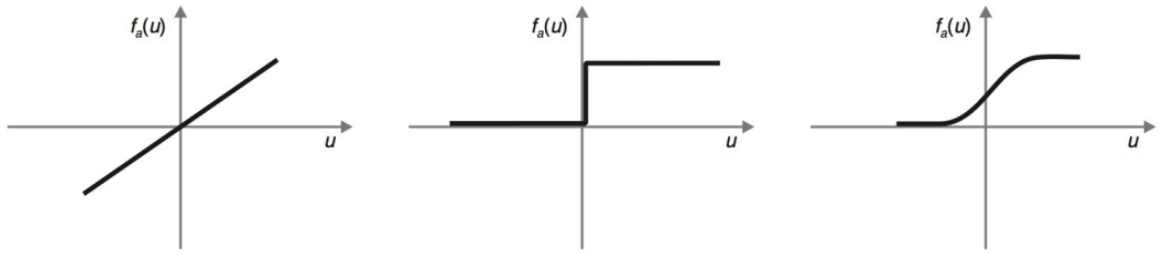
Como mencionado nos objetivos deste trabalho, a solução proposta funciona com base em geolocalização de objetos, portanto os algoritmos utilizados aqui tem o objetivo de identificar sua coordenada dentro de um espaço. Sendo assim, os esforços dos algoritmos de aprendizagem de máquina resultarão em dados numéricos. Neste caso, os métodos demonstrados são **algoritmos supervisionados de regressão**.

2.2.1 *Multilayer Perceptron* - MLP

Dentro da área de aprendizagem de máquina, existe métodos chamados de **Redes Neurais Artificiais** (ou RNA) cujo objetivo é simular a capacidade de aprendizagem do cérebro humano, utilizando como modelo a estrutura e o funcionamento de um sistema nervoso (FACELI et al., 2011).

Como explicado por Faceli et al. (2011), uma RNA é composta por uma série de unidades de processamento, os chamados **neurônios**, estes recebem um valor de entrada nos quais geralmente são aplicados pesos. Os valores de entrada ponderados de um neurônio são combinados à uma função matemática (função de ativação) na **camada intermediária** (ou oculta), dentre as funções que podem ser usadas, temos como exemplo: linear, limiar ou sigmoid (ver Figura 3). Ao final desse processo, na **camada de saída**, o neurônio gera um valor de saída de acordo com os valores recebidos anteriormente, sendo a saída a indicação do excitamento da função. Algumas RNAs possuem múltiplas camadas intermediárias, estas redes são classificadas como **multicamadas**. A Figura 4 contém a representação gráfica de uma RNA simples, onde a rede é alimentada por um conjunto de atributos de entrada (*input*) que são processadas pela camada intermediária resultando na camada de saída (*output)* que pode ter um ou mais neurônios.

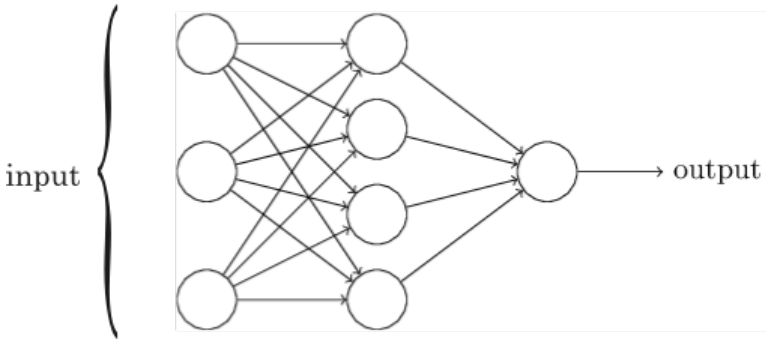
Figura 3 – Exemplo de funções de ativação de uma RNA



(a) Linear (b) Limiar (c) Sigmoidal

Fonte: Faceli et al. (2011)

Figura 4 – Representação gráfica de uma RNA artificial

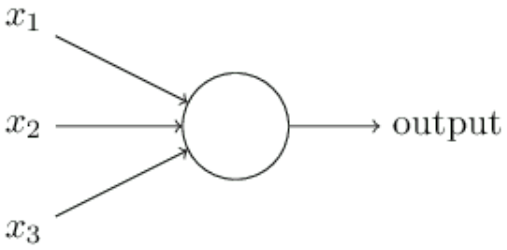


Fonte: Adaptado de Data Science Academy (2019)

Como citado por Faceli et al. (2011), a primeira rede neural, o *perceptron*, foi criado por Rosenblatt (1958). *Perceptrons* contém apenas os valores de entrada conectados à um neurônio

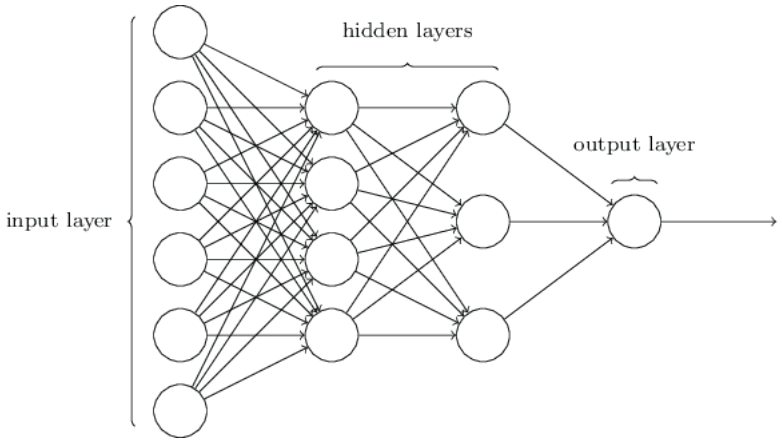
de saída e utiliza a função de ativação limiar (ver Figura 5). Uma RNA formada por camadas com múltiplos perceptrons é chamada de *multilayer perceptron* (MLP, ou perceptron multicamadas) (DATA SCIENCE ACADEMY, 2019), utilizada para resolução de problemas não linearmente separáveis, deste modo suas camadas ocultas utilizam funções não lineares (como a sigmoidal). Comumente, cada camada da MLP é conectada completamente à camada posterior (FACELI et al., 2011). A Figura 6 apresenta uma rede com duas camadas intermediárias.

Figura 5 – Representação gráfica de um perceptron



Fonte: Adaptado de Data Science Academy (2019)

Figura 6 – Representação gráfica de uma MLP com duas camadas intermediárias



Fonte: Adaptado de Data Science Academy (2019)

2.2.2 Árvore de Decisão

De acordo com Faceli et al. (2011), árvores são algoritmos que utilizam o método de divisão e conquista para resolver problemas complexos, dividindo problemas maiores em problemas menores e agrupando suas soluções em formato de uma árvore. Quando usadas com problemas de classificação, as chamamos de *árvores de decisão*, e para problemas de regressão elas são denominadas *árvores de regressão*, o que em suma não diferem muito. Como a seção de aprendizagem de máquina deste trabalho é baseado em um problema de regressão, será adotado durante o decorrer do trabalho o termo *árvore de regressão*.

Faceli et al. (2011) explica que uma árvore de regressão é um grafo acíclico onde cada nó pode ser classificado como: folha ou nó de divisão. Um **nó folha** é tido como uma função, podendo ser a constante que minimiza a função de custo, em problemas de regressão essa constante é : a média quando se quer minimizar o erro médio quadrático; ou a mediana, quando se

quer minimizar o desvio padrão. Já um **nó de divisão** possui um teste condicional baseado nos valores de um atributo.

2.2.3 Random Forest

Breiman (2001) apresenta *Random Forest* (ou floresta randômica) como uma coleção de árvores que são dependentes de vetores de características selecionados de forma randômica com a mesma distribuição para todas as árvores, tal que a formação de um vetor é independente de vetores utilizados em outras árvores. Após a formação de um número de árvores, o algoritmo realiza uma votação para identificar qual a classe mais popular dentro do conjunto.

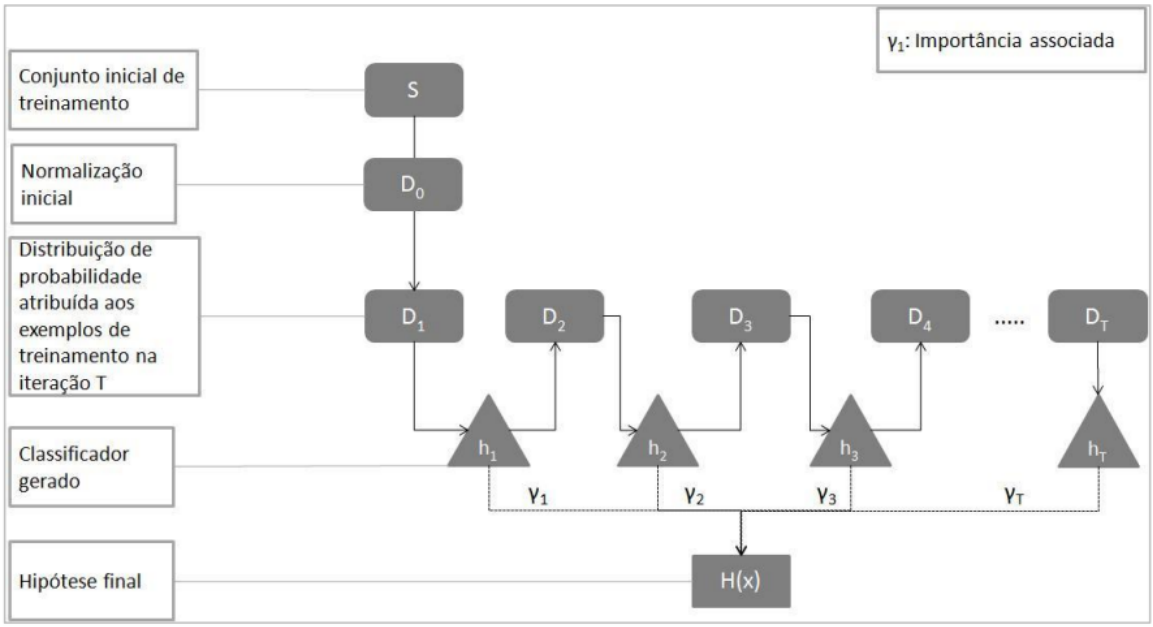
2.2.4 AdaBoost

Adaptive Boosting, ou AdaBoost, é um algoritmo de aprendizagem de máquina usado tanto para regressão quanto para classificação, apresentado em 1995 por Freund e Schapire (1999). Seu nome em tradução literal significa ”impulso adaptativo”, o qual descreve bem a funcionalidade do algoritmo: converter um algoritmo fraco de aprendizagem (um que não apresente bons resultados) em um algoritmo forte. Uma vantagem do uso desse método, é que ele não é tão custoso quanto outros existentes, sendo ele relacionado a programação linear (FREUND; SCHAPIRE, 1999; CHAVES, 2012).

De modo geral, o termo *Boosting* presente na sua nomenclatura refere-se à um método utilizado para aperfeiçoar o desempenho de um algoritmo, através da combinação de algoritmos considerados mais fracos, chamado ao final do processo de algoritmo forte (FREUND; SCHAPIRE, 1999; CHAVES, 2012). Os algoritmos fracos podem ser quaisquer métodos de ML, como por exemplo uma **árvore de decisão**. A Figura 7 apresenta um fluxograma do funcionamento do algoritmo utilizado pelo AdaBoost, sendo seu funcionamento detalhado logo em seguida.

Inicialmente, o algoritmo recebe um conjunto de treinamento (S), que por sua vez é recebe uma distribuição de pesos (D_t) aplicados sobre os padrões do conjunto. Os pesos aplicados são distribuídos igualmente na primeira iteração, sendo atualizados a cada nova iteração (t) realizada pelo algoritmo. Após a execução do algoritmo fraco, uma hipótese h_t é gerada com base nos pesos da distribuição, e então os pesos associados aos padrões que tiveram uma pior classificação são incrementados. Deste modo, a próxima iteração utilizará os pesos atualizados sendo forçada a focar nos padrões mais difíceis. Para cada hipótese gerada, é calculado um valor de importância γ_t, no qual quanto menor o erro maior será o eu valor, sendo que se o valor de acurácia do algoritmo fraco for menor que 50% ele recebe uma importância negativa, caso contrário, uma importância positiva. Ao final do algoritmo, a hipótese final é escolhida com os valores de significância de cada uma das hipóteses dos algoritmos fracos sendo usados como pesos.

Figura 7 – Fluxograma de funcionamento do algoritmo AdaBoost



Fonte: Adaptado de Chaves (2012)

2.3 TRABALHOS CORRELATOS

Shala e Rodriguez (2011) apresentam uma avaliação de uma solução para geolocalização *indoor* utilizando WiFi *fingerprints* (assim como neste trabalho) em conjunto com sensores embutidos em dispositivos móveis com o sistema operacional Android, como é o caso do acelerômetro, giroscópio e outros. Os autores do artigo citado utilizaram um algoritmo de KNN (do inglês *k-nearest neighbour* ou *k* - vizinhos mais próximos) para estimar a posição do diapositivo móvel do usuário. Além disso, utilizaram apenas 4 APs fixas em posições conhecidas do mapa. A base de treinamento foi construída através da média de 4 leituras de RSSIs em um ponto do mapa, cada leitura é feita em 4 direções em relação ao centro do ponto, gerando um valor médio. Como meio de tentar contornar as variações de sinais que resultam em *outliers*, os autores utilizaram pesos atribuídos a outras técnicas implementados na aplicação: detector de movimento, contador de passos, bussola, entre outros. A técnica de fusão conseguiu chegar a um erro médio de menos de 2m entre a posição real e a estimada.

Altintas e Serif (2011) utiliza o método baseado no *K-Means Clustering* utilizando o RSSI de transmissores de redes sem fio, comparando seus resultados com o KNN comum. Pelos experimentos realizados, com diferentes variações de argumentos, o *k-Means* mostrou resultados que variam de 2,68m à 4,11m. Enquanto que Altintas e Serif (2011) utilizam técnicas de classificação e clusterização.

Zegeye et al. (2016) realizaram experimentos utilizando apenas WiFi *fingerprints* em um corredor de um prédio. Para os experimentos, foram utilizados 10 APs e 93 pontos de localização para treinamento no local. Realizaram dois tipos de experimentos: um inicial utilizando todos os dados coletados e posteriormente, realizaram alguns melhoramentos em como os dados eram

coletados e analisados. Para o primeiro experimento, obtiveram um distanciamento médio de 7,46m entre os pontos reais e estimados, sendo que em 67% dos casos a distância estava abaixo de 10m. O segundo experimento obteve uma distância de 3,46m e em 80% dos testes a distância estava abaixo dos 5m. Enquanto que o primeiro experimento se assemelhou ao usado aqui, eles não deixam claro se foi feita uma coleta com diferença temporal, e no segundo experimento realizaram mudanças na base que não foram feitas nesta pesquisa. Outra diferença se dá pelo fato de aqui ter sido usado algoritmos de aprendizagem de máquina, enquanto que eles buscaram resolver o problema configurando-o como um problema de otimização.

Gogolak et al. (2013), apresentam uma aproximação utilizando método de WiFi fingerprint juntamente com uma rede neural. Além disso, como meio de comparação, um algoritmo KNN com pesos (WKNN). A coleta de dados para treino da rede neural é feita em um espaço pré-definido com pontos colocados em uma coordenada (x, y) do ambiente.

Enquanto este trabalho utiliza apenas um tipo de tecnologia de sinais para estimar a posição de um dispositivo, Aparicio et al. (2008) propõe o uso da fusão das tecnologias de Bluetooth e WiFI para apresentar uma solução de geolocalização indoor. Para tanto, os autores utilizam mapas de sinais com os RSSIs coletados de APs de WiFi e Bluetooth. Ao final dos seus experimentos, a fusão das duas tecnologias apresentou um erro médio de 2,21m, o que representou um avanço em comparação com o resultado usando apenas WiFI, 2,73m.

3 ASPECTOS METODOLÓGICOS

Este capítulo apresenta como a solução, englobando todos os módulos do sistema, foi desenvolvida. A seção 3.1 como a pesquisa é caracterizada. A seção 3.2 descreve o método de pesquisa utilizado neste trabalho. A seção 3.3 mostra como os componentes que fazem parte do sistema se conectam entre si. A seção 3.4 detalha as principais tecnologias que foram utilizadas na construção do sistema como um todo. A seção 3.5 contém a descrição da interface administrativa do sistema, trazendo seus requisitos. A seção 3.6 apresenta um resumo sobre o aplicativo mobile. Por fim, a seção 3.7 apresenta como a coleta de dados foi realizada.

3.1 CARACTERIZAÇÃO DA PESQUISA

Usou-se a taxonomia de Wohlin e Aurum (2015) para classificar este trabalho, tomando oito pontos (que podem ser reduzidos a sete) de decisão para sua classificação. Deste modo, esta pesquisa pode ser considera de cunho **aplicado**, uma vez busca fornecer uma solução para um problema específico (geolocalização *indoor*) com base em conhecimento extraído de práticas conhecidas (WiFi *fingerprints e machine learning*). A lógica da pesquisa é do tipo **dedutiva**, onde o desenvolvimento parte do objetivo mais geral para os mais específicos, confirmando a hipótese de que uma das técnicas escolhidas para testes atendem apresentam um resultado satisfatório com base nos dados coletados. A pesquisa é de propósito **avaliativo**, uma vez que busca avaliar um método que pode solucionar o problema de geolocalização *indoor*.

Quanto à sua aproximação, esta pesquisa é de cunho **quantitativo**, uma vez que trabalha com métodos estatísticos e de aprendizagem de máquina. O método de pesquisa é classificado como ***design science research***, tal que tem como objetivo a geração de um artefato que solucione um dado problema. Durante a construção desse estudo, utilizou do método de **experimentação** para as coleções de dados trabalhados, utilizando dados reais para testes a fim de chegar nos resultados desejados. Por fim, os resultados obtidos foram analisados e tidos como aceitáveis através de um método **estatístico**.

3.2 DESIGN SCIENCE RESEARCH

Como denotado na seção anterior, a metodologia utilizada aqui foi o *Data Science Research* (DSR). Na área de Sistemas de Informação, existem dois paradigmas que caracterizam os métodos de pesquisa (ALAN et al., 2004): o chamado de ciência comportamental (do inlgês, *behavioral science*), que busca desenvolver e verificar teorias; e, ciência do design (do inglês, *design science*), cujo objetivo é o desenvolvimento de artefatos. No contexto da pesquisa feita aqui, o artefato gerado é uma implementação de um software.

Segundo Alan et al. (2004), como citado por Santana (2017), artefatos podem ser divididos em quatro tipos:

- Instância: implementação de um artefato no seu ambiente, como por exemplo, uma implementação de um software
- Construções: formação conceitual para caracterização de fenômenos investigados. Exemplos são: tipos de dados, objetos.
- Modelo: é uma representação de como algumas coisas são e se relacionam, como por exemplo: casos de uso, diagramas UML.
- Método: sequência lógica de passos para se cumprir uma tarefa

O DSR é dividido em seis passos (PEFFERS et al., 2007), os quais são contextualizados nessa pesquisa na Tabela 1.

Tabela 1 – Passos que compõem o DSR

Passo	**Contextualização**
Identificação do problema e motivação	Geolocalização através do GPS apresenta ineficiência quando usado em locais fechados, por isso há a necessidade de uma outra aproximação
Objetivo	Desenvolver uma sistema para geolocalização de usuários m um ambiente e traçar direção entre um usuário e um ponto localizados no mesmo ambiente
Design e desenvolvimento	A solução foi construída com base em conceitos de aprendizagem de máquina e técnicas de localização
Demonstração	Foram realizados experimentos com o método que apresentou resultados no estado da arte
Avaliação	Uso de métodos estatísticos para demonstrar que o resultado obtido se encontra dentro do esperado
Comunicação	Relatório de desenvolvimento no formato de monografia

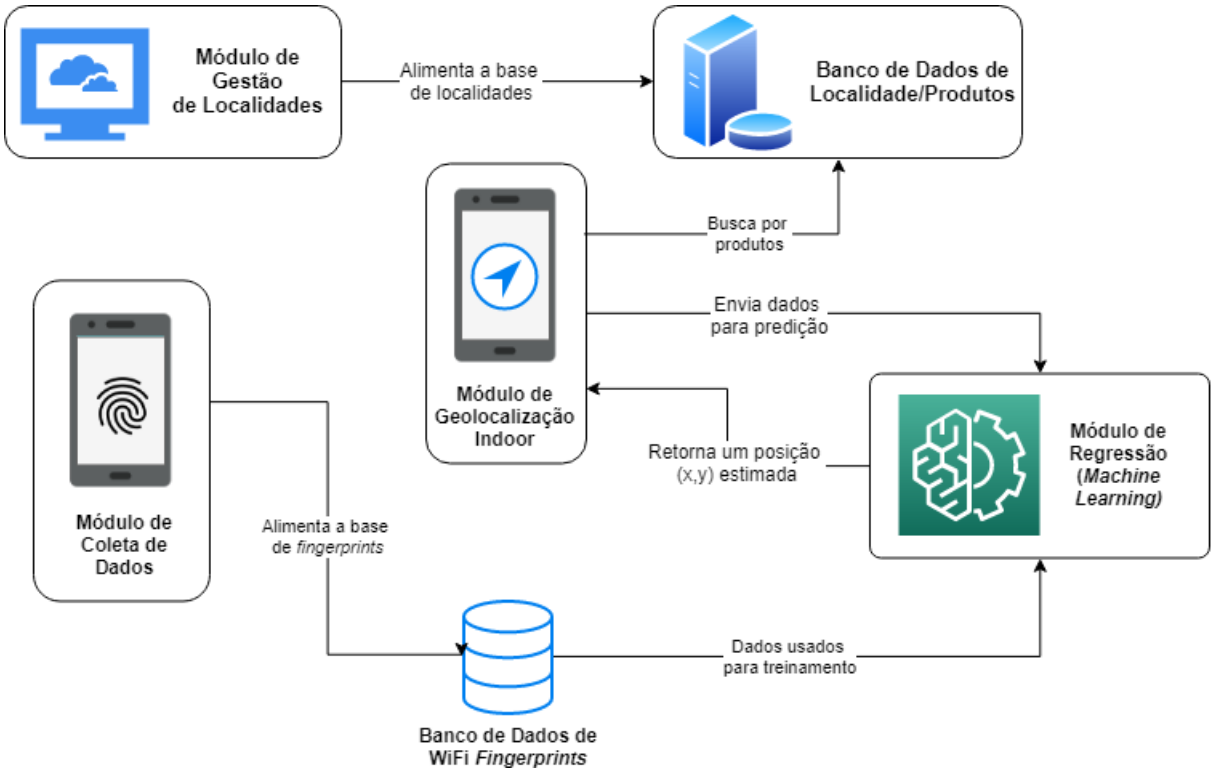
Fonte: Adaptado de Peffers et al. (2007)

3.3 DIAGRAMA DOS COMPONENTES

O sistema em sua completude é composto por quatro componentes e duas bases de dados. A Figura 8 traz uma representação visual das conexões entre estas bases e componentes do sistema. A ferramenta de gestão de localidades é responsável por alimentar a base de dados com as localidades dos produtos. A base de localidades é acessada pelo aplicativo móvel (versão cliente ou usuário final) durante a pesquisa por produtos. O aplicativo móvel, usado pelo cliente, captura os sinais de redes WiFi ao seu alcance em um momento t e envia como entrada para o algoritmo de aprendizagem de máquina responsável por estimar sua posição. O algoritmo é

alimentado também com a base de dados de WiFi *fingerprints* que foram coletados na fase de rastreamento (invisível para o usuário final). A fase de rastreamento é feita com o auxílio de um aplicativo móvel, que obtém e salva os dados das redes WiFi ao seu alcance.

Figura 8 – Diagrama de conexão entre os componentes que integram o sistema



Fonte: O autor

O funcionamento do sistema como um todo começa pelo **Módulo de Gestão de Localidades** (LocalFácil), onde os mapas dos ambientes são construídos com base nas plantas baixas de uma filial cadastrada pelo sistema. Dentro de uma filial é possível trabalhar com um ou mais mapas. A construção de cada mapa é feita como uma espécie de decalque da planta, em que elementos gráficos são posicionados sobre a imagem original, indicando locais de interesse. Após a construção de um mapa de locais, o **Módulo de Coleta de Dados** (implementado na forma de um aplicativo móvel) poderá ser utilizado para a construção da base de dados de treinamento que será utilizada pelo **Módulo de Regressão**. Este, por sua vez realiza a execução do algoritmo de predição da posição atual do usuário dentro de um ambiente, com base nos dados de sinais WiFi coletados pelo AcheFácil (nome dado ao **Módulo de Geolocalização Indoor**). O AcheFácil é um aplicativo móvel que realiza buscas por locais de interesse de acordo com termos informados pelo usuário. Os dados obtidos nas buscas, inclusive os mapas criados, são oriundos do banco de dados de localidades que é gerenciado pelo módulo de gestão. O detalhamento de cada um dos módulos é encontrado nas seções seguintes e no Capítulo 4.

3.4 TECNOLOGIAS UTILIZADAS

Um conjunto de tecnologias foram necessárias para o desenvolvimento do sistema completo. Sendo divido entre interface web e *mobile*, foi necessário pensar em uma solução para realizar a comunicação entre as duas plataformas de forma fácil. A seguir serão apresentados os principais conceitos que foram usados para o desenvolvimento dos componentes.

3.4.1 JavaScript

Criada em 1995 para o navegador Netscape, a linguagem de programação JavaScript atualmente está presente em todos os navegadores de internet. Sendo utilizada para melhorar a interação entre aplicações web e o usuário, a linguagem JavaScript mudou a forma como as aplicações são desenvolvidas nos dias de hoje. Apesar do seu nome, a linguagem não tem muitas semelhanças com a Java, seus criadores apenas aproveitaram o crescimento que esta última estava obtendo e resolveram tirar proveito da situação e chamar atenção com o nome (HAVERBEKE, 2018). Ela é considerada como uma linguagem de alto nível, dinâmica, sem tipagem de dados e interpretada, podendo ser trabalhada como orientada à objetos ou mesmo funcional (FLANAGAN, 2011).

Embora no início tenha sido utilizada apenas como um meio de melhorar a experiência do usuário em *websites*, a linguagem passou para um nível completamente diferente com o decorrer dos anos. Dentro da versatilidade de uso desta linguagem, podemos citar: meio de criação e gerência de servidores web robustos (Node.js[1] em conjunto com o Express[2]), frameworks para o desenvolvimento *frontend* de softwares (Angular[3]) ou ainda para desenvolvimento de aplicativos móveis multiplataforma (como é o caso do React Native[4]). Na subseção 3.4.2, são apresentadas tecnologias totalmente baseadas em JavaScript e que foram utilizadas no projeto.

A simplicidade no desenvolvimento de aplicação, mesmo robustas, é o fator determinante para escolha dessa linguagem quando o objetivo é trabalhar com uma plataforma web. A versatilidade e grande número de recursos disponíveis atualmente (frameworks, bibliotecas) colaboram para sua escolha. Como será mostrado adiante, é possível desenvolver todos os lados de um sistema utilizando apenas essa linguagem de programação como base, no caso deste trabalho, o **MEAN Stack** é o que exemplifica essa versatilidade. Dentro do diagrama dos componentes, o JavaScript está incluído dentro do módulo de gestão de localidades e do seu banco de dados (uma vez que é desenvolvido em MongoDB, visto mais adiante).

[1] https://nodejs.org/
[2] https://expressjs.com/pt-br/
[3] https://angular.io/
[4] https://facebook.github.io/react-native/

3.4.2 MEAN Stack

A **MEAN Stack** é uma coleção de tecnologias, dentre as muitas existentes, utilizadas atualmente para desenvolvimento de aplicações web baseadas na linguagem de programação JavaScript, abrangendo tanto o lado cliente quanto o lado servidor da aplicação. A nomenclatura dada a esta coleção é o acrônimo das seguintes tecnologias: **M**ongoDB[5], **E**xpress, **A**ngular e **N**odeJS. Dentro desse conjunto, o Angular é o responsável pelo *frontend* da aplicação, enquanto que os outros três componentes realizam o lado *backend*. Abaixo está um detalhamento de cada um deles. Ela foi utilizada no desenvolvimento do módulo de gestão web já que incorpora o lado cliente e o servidor de um sistema, a interação entre os submódulos se torna mais simples, facilitando o desenvolvimento da aplicação.

- MongoDB

  De acordo com Ihrig e Bretz (2015), MongoDB é um banco de dados NoSQL de código aberto (desde 2009), criado em 2007 pela MongoDB Inc (na época conhecida como 10gen). Este serviço de banco de dados é uma alternativa para bancos de dados SQL conhecidos como MySQL, Postegres etc. O MongoDB adota o padrão de orientação a documentos, onde os dados são armazenados no formato BSON (Bynary JSON).

  Comparando alguns conceitos do MySQL com MongoDB, a Tabela 2 traz algumas equivalências:

Tabela 2 – Equivalências de conceitos: MySQL x MongoDB

MySQL	**MongoDB**
Tabela	Coleção
Tupla (linha)	Documento
Coluna	Campo
Joins	Documentos embutidos, $lookup e $graphLookup
Group by	Pipeline de agregação

Fonte: Adaptado de MongoDB and MySQL Compared (MONGODB…,)

- Express

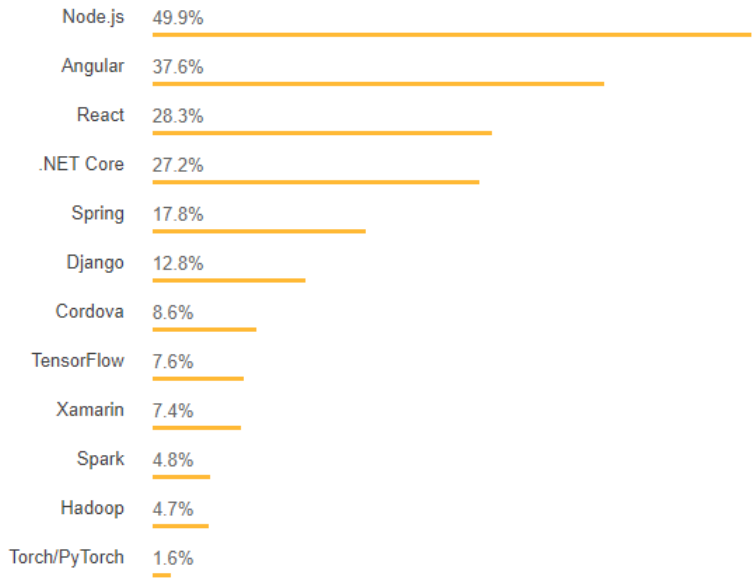
  Express é um framework web que funciona dentro do Node.js. Visto que o Node.js não suporta a manipulação direta de requisições HTTP, este framework fornece alguns mecanismos que visam suprir essa deficiência, sendo eles: manipulação de requisições de HTTP (GET, POST, PUT etc.) para diferentes rotas, servidor de arquivos estáticos, integração com mecanismos de renderização de *views* de respostas de dados (EXPRESS/NODE…, 2019) e entre outros. De forma resumida, o Express é o responsável por dar ao desenvolvedor uma solução para criação e manutenção de servidores web robustos (IHRIG; BRETZ, 2015; SHORT, 2018).

[5] https://www.mongodb.com/

- Angular

  O Angular é um framework utilizado para criação do *frontend* de aplicações web, sendo um dos mais populares atualmente (ver Figura 9). Inicialmente conhecido como AngularJS (Angular 1), o framework (até a data deste trabalho) encontra-se na sua versão 8 e é tratado apenas como Angular, tendo mudado drasticamente a sua estruturação quando evolui da sua primeira versão para o Angular 2.

Figura 9 – Frameworks mais utilizados por profissionais em 2018



Fonte: Stack Overflow Developer Survey 2018 (STACK…, 2018)

As aplicações desenvolvidas com Angular são classificadas como *Single Page Applications* (ou SPAs). SPAs partem do conceito que diz que tudo que é necessário para uso deve ser carregado na primeira requisição ao servidor, assim a navegação se torna mais rápida e fluida uma vez que o cliente não precisa esperar pelo carregamento das respostas de requisições para cada página carregada (IHRIG; BRETZ, 2015).

Ele utiliza como design pattern o MVC (*Model-View-Controller*), no qual os *models* do padrão são definidos como objetos JavaScript. As *views* são as representações visuais dos dados manipulados na aplicação, em formato HTML, incorporadas com elementos próprios da aplicação chamados de *directives*. E por fim, o *controller* é o responsável pela manipulação dos dados da aplicação.

Um recurso que facilita o desenvolvimento com este framework é o chamado *two-way data binding*, um recurso que permite a sincronização de dados entre os atributos/variáveis da camada model com a camada *view*, através do *controller*. Uma vez atualizado um valor do *model* no *controller*, a *view* é atualizada automaticamente, e vice-versa (IHRIG; BRETZ, 2015).

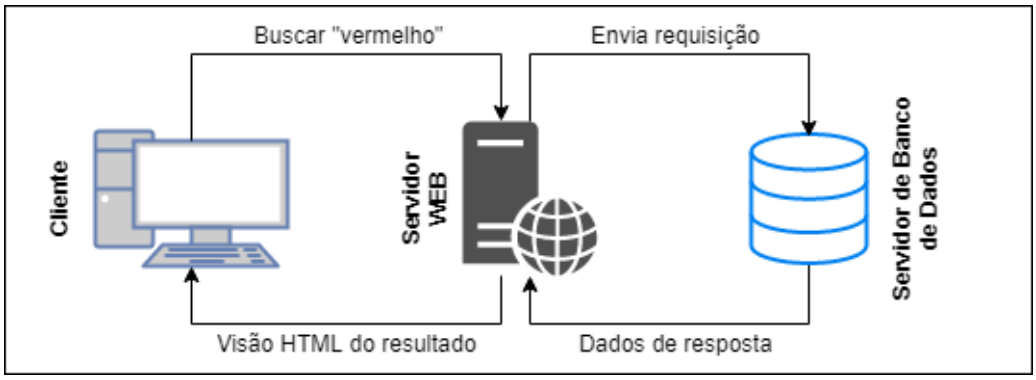
- Node.js

O Node.js é um *framework* criado com o intuito de permitir o desenvolvimento de aplicações escaláveis de rede (IHRIG; BRETZ, 2015), além de ser uma ferramenta de código aberta e multiplataforma (EXPRESS/NODE…, 2019). Sendo completamente baseado em JavaScript, este *framework* permite a criação de servidores de maneira rápida e simples, além de trazer a possibilidade de acesso a recursos do sistema que não era possível com a linguagem, como por exemplo: leitura de diretório e criação de sockets. No conjunto descrito aqui, o Node.js toma o papel de *web server*, contudo ele possui algumas diferenças em relação à *web servers* tradicionais feitos em linguagens como Java, PHP e outras.

A Figura 10 exemplifica como é o ciclo de vida de uma requisição realizada por um cliente à um servidor web.

1. O cliente deseja procurar todos os carros cadastrados com a cor ”vermelho”;
2. O servidor web recebe essa requisição e manda para o servidor de banco de dados;
3. A busca é processada e o resultado é enviado como resposta para o servidor web;
4. O servidor web processa o resultado em conteúdo HTML e envia para o cliente;
5. O resultado retornado pelo servidor web é processado pelo navegador de internet.

Figura 10 – Ciclo de vida de uma requisição web.



Fonte: Adaptado de Ihrig e Bretz (2015)

Como explicado por Ihrig e Bretz (2015), em servidores comuns, o segundo passo do processo pode gerar um problema conhecido como *blocking I/O*, que ocorre quando um processo bloqueia a execução do programa enquanto o seu término não é atingido, fazendo com que outros processos fiquem na espera por sua vez. Este problema é normalmente tratado com a execução de mais de uma *thread* simultaneamente, o que é uma solução cara já que precisa de aumento de hardware e poder de processamento na medida em que o servidor cresce.

Servidores baseados em Node.js funcionam do mesmo modo que o exemplo dado na Figura 10. Todavia, existe uma diferença básica e importante entre eles e servidores comuns: O Node.js consegue lidar com o blocking I/O de forma nativa, visto que o JavaScript é

por natureza uma linguagem tida como *non-blocking*, através da implementação dos *callbacks*. *Callbacks* são funções que são executadas quando uma operação é finalizada por sucesso ou erro na execução (IHRIG; BRETZ, 2015). Assim, quando um servidor que é executado em JavaScript recebe duas ou mais requisições, cada processo será executado concorrentemente, sem necessidade de espera pela finalização de um para executar o outro, de modo a não necessitar de recursos adicionais do sistema.

3.4.3 REST API

REST é um acrônimo para ***RE**presentational **S**tate **T**ransfer* (Transferência de Estado Representacional), e funciona como um modelo arquitetural para sistemas de hipermídia distribuídos (FIELDING; TAYLOR, 2000). Esse modelo segue alguns princípios que devem ser cumpridos para que alguma implementação possa ser considera REST (WHAT…,), sendo eles:

- **Cliente - servidor:** separação entre as interfaces de usuário e de armazenamento de dados, deixando-os independentes um do outro;
- **Stateless:** toda e qualquer requisição contém toda a informação suficiente para seu objetivo, ou seja, não é necessário nenhum contexto do lado servidor, apenas o cliente pode gerar e manter sessões;
- **Cacheable:** disponibilidade para permitir que dados sejam armazenados em cache ou não;
- **Interface uniforme:** permite que a arquitetura seja simplificada e assim melhorada;
- **Sistema em camadas:** meio de assegurar que um componente não tenha acesso a recursos que estão acima dele;
- **Código sob demanda:** permite a extensão das funções do cliente através da obtenção e execução de códigos em forma de scripts.

Recursos REST são manipulados através dos métodos do protocolo HTTP, que são identificados através de URI's (*Uniform Resource Identifier*, ou Identificador de Recurso Uniforme). Os métodos são identificados no momento em que uma requisição HTTP é enviada para uma URI., sendo os principais métodos listados na Tabela 3:

O uso do conceito REST foi utilizado neste trabalho para a construção das APIs de obtenção de dados abertos, funcionando com uma interface entre os módulos de localização *indoor* e o banco de dados de localidades. Uma vez que o aplicativo *mobile* de pesquisa e geolocalização do usuário necessita buscar por locais de interesse cadastrados, foram criados links que obtém uma lista no formato JSON (*JavaScript Object Notation*), que é um formato de troca de dados baseado em texto e independente de linguagem (CROCKFORD, 2006). Os dados buscados são tratados e introduzidos na aplicação.

Tabela 3 – Principais métodos HTTP

Método	**Descrição**
GET	Obtém recursos
PUT	Substitui os dados de um determinado recurso
POST	Cria um novo recurso
DELETE	Deleta um recurso existente
HEAD	Obtém o cabeçalho de um resposta, ignorando os dados que viriam normalmente

Fonte: REST: Princípios e boas práticas (FERREIRA, 2017)

3.4.4 Banco de dados - NoSQL

Ao contrário dos bancos de dados tradicionais (MySQL, Postgres, SQL Server etc.) que utilizam SQL para realizar todas as suas operações, bancos de dados NoSQL deixam de lado esse conceito para trabalhar de forma diferente. Segundo Ihrig e Bretz (2015), NoSQL representa qualquer tipo de armazenamento de dados que não utiliza SQL, sendo esses divididos em vários tipos de orientação de armazenamento. Essa categoria de banco de dados veio com a necessidade do gerenciamento de grandes volumes de informações, além da necessidade de alta escalabilidade e disponibilidade (LÓSCIO et al., 2011). Os modelos de armazenados são divididos em:

- **Chave-valor:** este é um modelo considerado simples, visto que de maneira básica, ele é pode ser visto como uma grande tabela *hash*, onde cada chave é associada a um único valor. Exemplos de bancos que utilizam esse modelo são: DynamoDB[6] e Redis[7].
- **Orientado a documentos:** tido como um modelo bastante flexível, ele é semelhante ao modelo chave-valor. Baseia-se em conjuntos de documentos que possuem conjuntos de campos (chaves) e valores. Permite a atualização da estrutura dos documentos sem que haja problemas com os dados já existentes.
- **Orientado a colunas:** neste modelo os dados são indexados por uma tripla (linha, coluna e timestamp), sendo que linha e coluna são chaves e o timestamp é usado para identificar versões diferentes de um dado. Exemplos de uso desse modelo são encontrados com o Cassandra[8] e o HBase[9].
- **Orientado a grafos:** neste modelo, o banco é estruturado por três elementos: nós (vértices), relacionamentos (arestas) e propriedades (atributos). De modo geral, um banco de dados com este modelo pode ser visualizado com um multigrafo direcionado, cada par de vértices pode ser conectado por uma ou mais arestas.

[6] https://aws.amazon.com/pt/dynamodb/
[7] https://redis.io/
[8] http://cassandra.apache.org/
[9] http://hbase.apache.org/s

De acordo com Lóscio et al. (2011), bancos NoSQL apresentam as seguintes características:

- Escalabilidade horizontal;
- Ausência de esquema ou esquema flexível;
- Suporte nativo a replicação;
- API simples para acesso aos dados; e,
- Consistência eventual.

A escalabilidade fornecida por bancos de dados NoSQL permitem que novas propriedades sejam adicionadas a entidades em tempo de gravação, o que diverge com a rigidez dos padrões dos formatos pré-definidos das tabelas de bancos relacionais. Além disso, esse tipo de banco de dados, diferente de sistemas relacionais, são desenvolvidos para funcionarem com distribuição de dados em múltiplos servidores sem perda de performance (GOOGLE…, 2017).

3.4.5 Sistema Operacional Android

O Android é um sistema operacional (SO) de código aberto e gratuito utilizado em dispositivos móveis, desenvolvido pela Android, Inc. e adquirida pela Google no ano de 2005, sendo mantido por esta desde então (DEITEL et al., 2015). De acordo com dados do website StatCounter (2019) no mês de maio, ele foi o SO mais utilizado em aparelhos portáteis, obtendo uma fatia de pouco mais de 75% do mercado mundial. Atualmente, é possível desenvolver aplicativos nativos para este sistema com o uso das linguagens de programação: Java (DEITEL et al., 2015) e Kotlin (JAMES, 2017).

A plataforma Android pode ser dividida em 5 grandes componentes (DEVELOPERS, 2019), sendo eles:

- **Aplicativos de sistema:** todos softwares desenvolvidos para execução em dispositivos móveis, como por exemplo: calendários, calculadoras, jogos, entre outros.
- **Java API framework:** todos os recursos utilizados por uma aplicação são acessíveis através de um framework na linguagem Java
- **Bibliotecas nativas C/C++:** alguns componentes e serviços são construídos com base em código nativo que necessitam de bibliotecas escritas em C ou C++, então o sistema dispõe de APIs Java para acesso a essas bibliotecas.
- **Android runtime:** responsável pela execução dos aplicativos no sistema,

- **Camada de abstração de hardware (HAL):** conjunto de módulos de bibliotecas usadas para implementação de interfaces para cada componente do hardware do dispositivo;
- **Kernel:** a base do sistema operacional é kernel do Linux.

3.5 MÓDULO DE GESTÃO DE LOCALIDADES

A proposta dessa aplicação é levar à empresa uma ferramenta que possibilite a organização de todos os seus produtos em exposição de forma prática e criar uma base de dados para o aplicativo de geolocalização ter acesso à pontos de interesse dos usuários. O sistema trata do cadastro de localidades em um ambiente, através de uma ferramenta gráfica de construção de mapas, como será apresentado no Capítulo 4. Neste contexto, uma localidade é um local ou ponto de interesse em que um objeto se encontra, a localidade é identificada por sua posição cartesiana. As localidades cadastradas são utilizadas como referencial de destino pelo aplicativo de geolocalização para traçar um vetor de direção entre elas e o usuário. A Figura 11 exemplifica a visualização (*pinpoint*, ou plotagem no mapa) de um local cadastrado, sendo indicado pelo ícone vermelho fixado no mapa.

Figura 11 – Exemplo do *pinpoint* realizado para localidade cadastrada e visualizada em um mapa do ambiente

Fonte: O autor

Com o intuito de garantir a portabilidade, e assim ser possível a sua utilização em qualquer sistema operacional, o `LocalFácil` (nome dado a este módulo) foi desenvolvido para ser executado a partir de navegadores web, sendo estruturado em forma de uma aplicação SPA. Como mencionado anteriormente, existe um conjunto de tecnologias chamado de MEAN *Stack* que permite o desenvolvimento de aplicações web completas (*client* e *server side*), e devido a simplicidade de desenvolvimento e vasto conteúdo para estudo, esta foi a escolhida para o projeto, além das outras vantagens detalhadas na fundamentação teórica desta pesquisa. Este sistema é representado na Figura 8 pelo módulo de gestão.

Para que seja possível a utilização do sistema, o ambiente de produção deve atender aos seguintes requerimentos:

- Node.js (na versão v10.x ou mais recente) para gerenciamento do servidor da aplicação;
- Gerenciador de pacotes NPM, para instalação de dependências da aplicação;
- Servidor MongoDB (instalação local ou acesso à um servidor remoto);
- Navegador de internet (Google Chrome, Mozilla Firefox, Opera) para acesso ao sistema.

3.6 MÓDULO DE LOCALIZAÇÃO INDOOR

O aplicativo móvel desenvolvido no trabalho, denominado **AcheFácil**, utilizado pelo usuário final cliente, foi desenvolvido para a plataforma Android devido ao fato desta ser a plataforma que domina o mercado mundial de *smartphones*, como foi dito no capítulo anterior. Além disso, a plataforma é código aberto e livre, o que facilita o seu uso para desenvolvimento de aplicativos, uma vez que é de fácil acesso e seu custo de desenvolvimento é mais barato do que outras plataformas existentes atualmente.

As informações obtidas através do app são oriundas de uma base de dados (alimentada pelo sistema de gestão, ver seção 3.5) que se encontra hospedada em servidores na nuvem, e obtidas através do uso de APIs REST. A escolha do uso de bancos remotos foi baseada no fato de que o armazenamento desses dados localmente (no dispositivo móvel) poderia ocupar muito espaço desnecessariamente, uma vez que os objetos desejados pelo usuário serão os únicos a serem obtidos. Hospedar os dados na nuvem possibilita a atualização das localizações de objetos em tempo real.

Após a obtenção da localização do objeto desejado, a aplicação captura os dados das redes sem fio ao seu redor para plotar um vetor de direção do usuário até o a posição do objeto, através da execução de um algoritmo de aprendizagem de máquina para predição da sua posição dentro do ambiente.

3.7 MÓDULO DE COLETA DE DADOS

A coleta de dados é uma etapa de suma importância para o funcionamento do sistema aqui descrito. Esta etapa equivale à fase de calibração mencionada na seção 2.1.2 (que trata dos métodos de geolocalização por sinais de redes sem fio), ou seja, essa é etapa em que dados serão coletados para alimentar a base de treinamento do algoritmo de aprendizagem de máquina.

As informações necessárias foram obtidas através do mapeamento do primeiro andar do Prédio II dos professores da UFRPE - UAG. Com o objetivo de obter o maior número possível de padrões de modo que fosse possível capturar os sinais de WiFi com mais precisão, foi construído

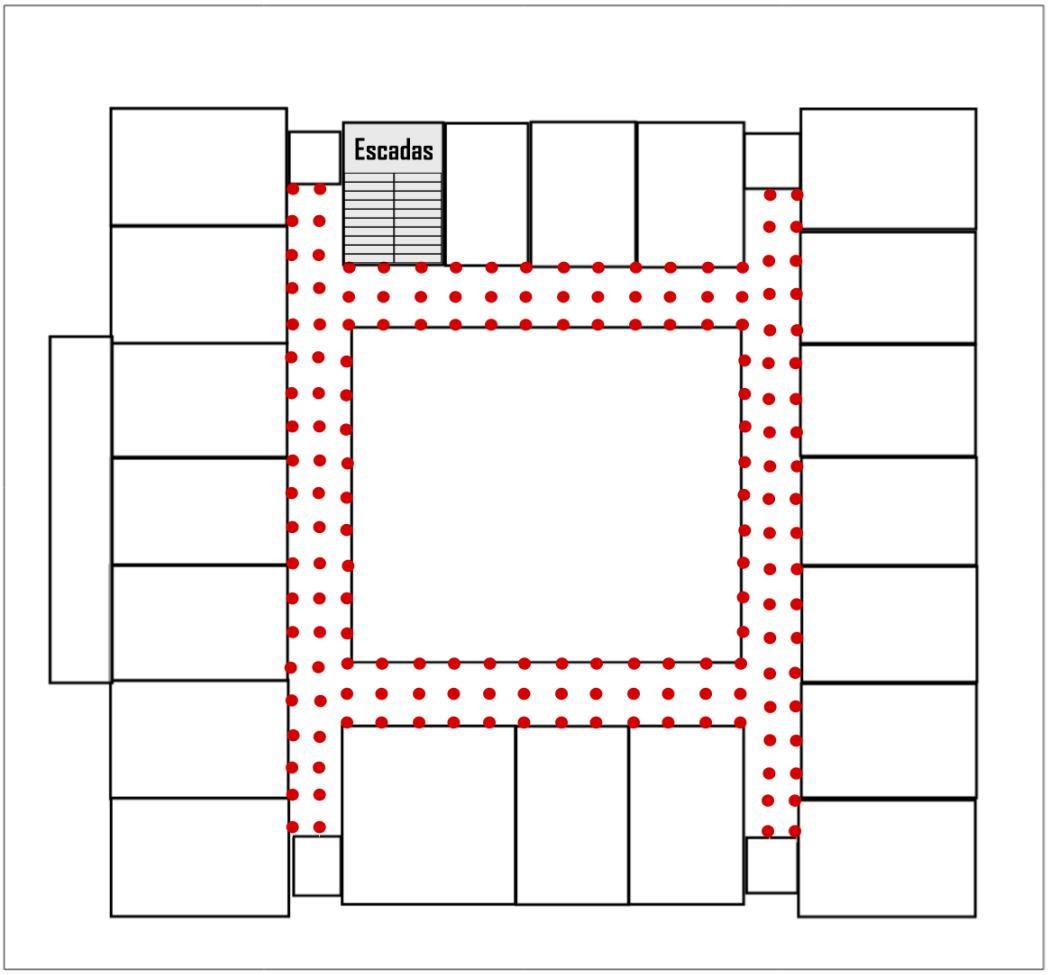
um mapa de pontos fixos a cada 1m (aproximadamente) nos corredores do andar. A Tabela 4 apresenta a equivalência das medidas em pixels para metros no mundo real, utilizando o mapa do andar utilizado como referência. A Figura 12 apresenta o layout de coleta de *fingerprints* por pontos pré-definidos, onde os pontos em vermelho indicam os locais em que é feita a obtenção de sinais em um dado momento.

Tabela 4 – Valores das medidas de distância e tamanhos utilizados nos experimentos

Valor em pixels (px)	**Valor em metros (m)**
19.75	0,5
39,5	1
79	2

Fonte: O autor

Figura 12 – Mapa de pontos pré-fixados do primeiro andar do prédio para a fase de calibração (coleta de *fingerprints*, ou fase offline)



Fonte: O autor

Foram construídas quatro bases de dados, sendo três delas compostas por dados obtidos em três dias consecutivos e uma foi construída através do cálculo da média dos valores dos RSSIs coletados anteriormente. As redes existentes em um ambiente podem sofrer alterações através do

tempo ou por perca de sinal por quaisquer que sejam os motivos, sendo assim decidiu-se realizar a construção do mapa de sinais em múltiplos dias e após isso realizar a média desses dias para que fosse possível analisar qual tipo de base se mostraria mais precisa na fase de aprendizagem.

As informações salvas nessas bases são instâncias compostas por valores numéricos, sendo eles: dois valores que representam a posição, x e y, de um ponto do plano cartesiano que representa o andar; e, uma quantidade *n* de valores de RSSI de todos os sinais de WiFi captados pelo aplicativo no momento da coleta, sendo associados ao BSSID (endereço MAC) do aparelho transmissor. Deste modo, na quarta base, a instância identificada pelo par (x_i,y_i) para o atributo MAC_p possui o valor obtido pelo cálculo $([MAC_{pDia1} + MAC_{pDia2} + MAC_{pDia3}]/3)$.

Caminhar por uma área coletando informações de sinais WiFi implica em, ao final do processo, não ter o mesmo número de sinais em todos os pontos visto que alguns locais podem estar fora do alcance de um determinado transmissor. Dado esse fator, para aqueles pontos que não possuem sinal lido para um transmissor (ou seja, um *missing value*) foi-lhe atribuído o valor -100. Como os valores de RSSI WiFi variam entre 0 e -100, onde quanto menor o valor menor o alcance/qualidade do sinal, essa atribuição foi considerada adequada.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos de acordo com cada um dos objetos específicos listados no Capítulo 1. A seção 4.1 apresenta a ferramenta construída para gestão das localidades dos produtos alocados em um ambiente. A seção 4.2 apresenta dados sobre da coleta de dados realizada na pesquisa. A seção 4.4 traz o resultado dos algoritmos de aprendizagem de máquina que foram obtidos com as bases da fase de coleta. A seção 4.3 apresenta o aplicativo gerado para posicionamento do usuário em relação à um produto em um ambiente.

4.1 MÓDULO DE GESTÃO DE LOCALIDADES

O sistema de gerenciamento de localizações recebeu o nome de `LocalFácil`, sendo ele um sistema web que tem como objetivo auxiliar o gerenciamento das localidades de produtos dentro de ambientes extensos, de modo que também serve como um fornecedor de dados para a aplicação móvel de geolocalização de usuários. Este componente possui os seguintes módulos principais:

Tabela 5 – Módulos do Sistema de Gestão de Localidades

Módulo	**Descrição**
Gerenciamento de filiais	Contém as operações CRUD (do inglês *Create, Read, Update and Delete*) para os dados de filiais do estabelecimento matriz
Gerenciamento de projetos de pisos	Engloba as operações de CRUD para os projetos (mapas) de pisos de um determinada filial
Interface de edição de projetos	Ferramenta de criação e edição de projeto criado para um piso. Engloba a adição de elementos gráficos em um mapa. Contendo também: • Bloco para adição de tipos de localidades à um elemento do mapa; • Alocação de produtos a um elemento do mapa; • Exportação do mapa criado para formatos arquivos de imagens.
Interface para pesquisa de produtos	Permite realizar a busca de produtos e realizar o *pinpoint* do produto na localidade em que ele se encontra na filial

Continua na próxima página

Tabela 5 – continuação da última página

Módulo	Descrição
APIs de acesso à dados	APIs utilizadas pelo aplicativo de geolocalização para acesso as localidades cadastradas

Fonte: O autor

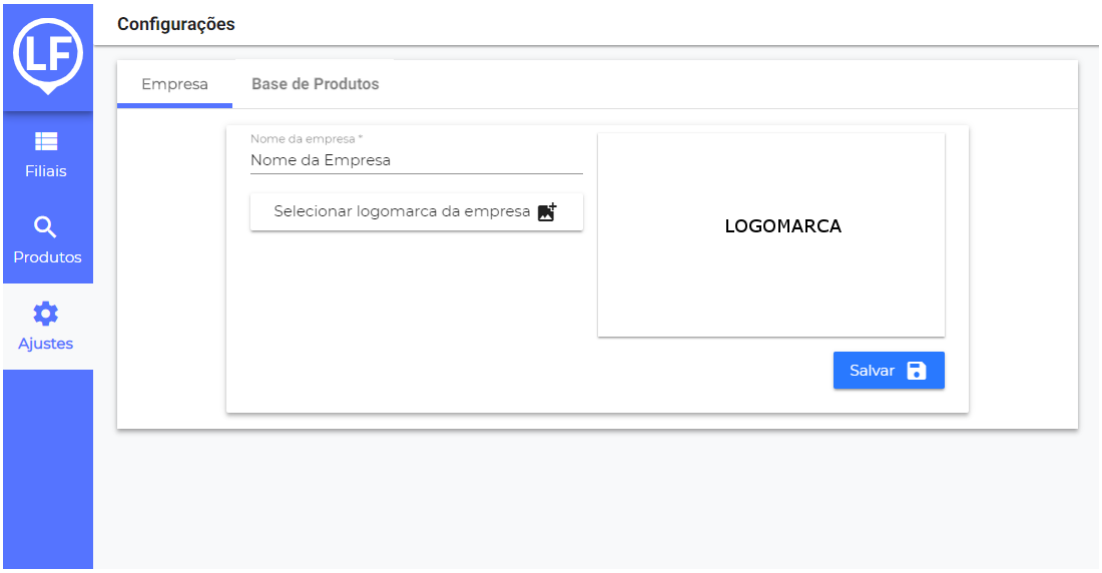
A tela principal do sistema pode ser vista na Figura 13, onde está localizado um menu com as opções dos módulos desenvolvidos. O menu principal pode ser acessado de qualquer parte do sistema.

Figura 13 – Tela inicial do sistema

Fonte: O autor

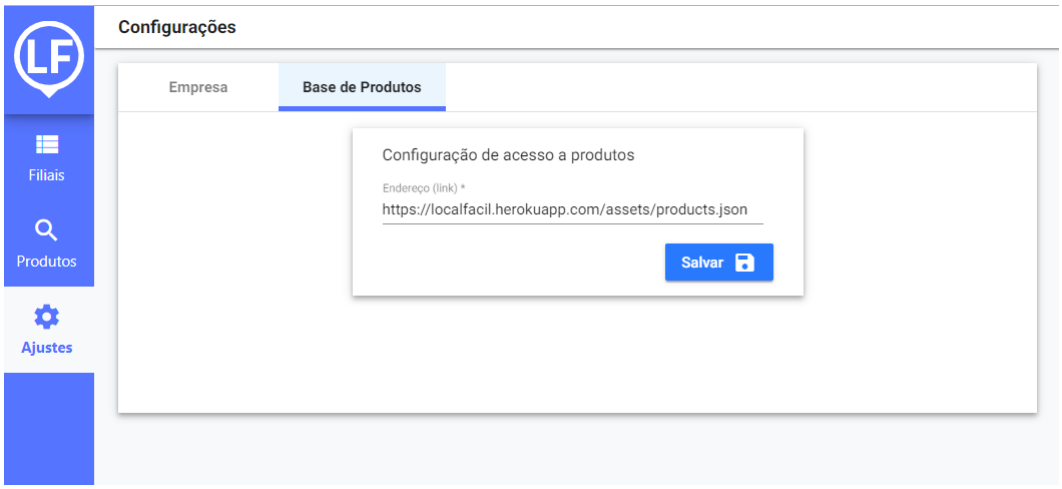
Seguindo um fluxo lógico de uso do sistema, a partir do primeiro uso, a primeira tela de interesse seria a tela de configurações (Figura 14), na qual o usuário pode: adicionar a logomarca e nome da empresa para customização do sistema (aba “Empresa”, Figura 14); e, inserir um link para a base de produtos (no formato JSON) para serem utilizados na alocação dos mesmos em um piso de filial (aba “Base de Produtos”, Figura 15).

Figura 14 – Tela de configurações do sistema - Dados da empresa



Fonte: O autor

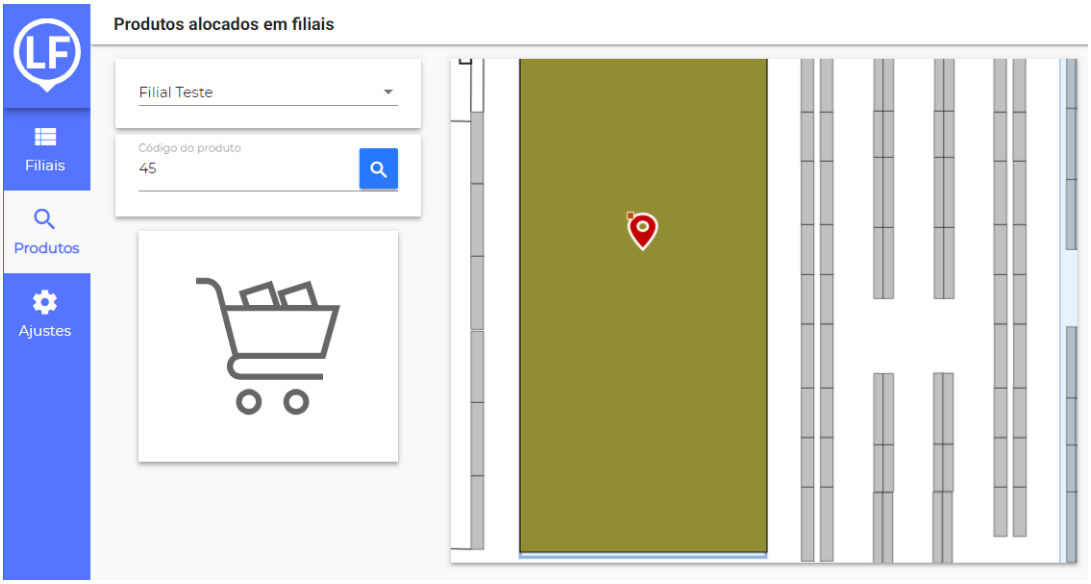
Figura 15 – Tela de configurações do sistema - Base de Produtos



Fonte: O autor

O usuário possui a opção de procurar por produtos alocados em uma filial, através do código do produto é possível verificar qual o andar em que o mesmo se encontra dentro da unidade, sendo exibido com um *pinpoint* no mapa do andar em que ele foi alocado. A tela de pesquisa e resultado é vista na Figura 16.

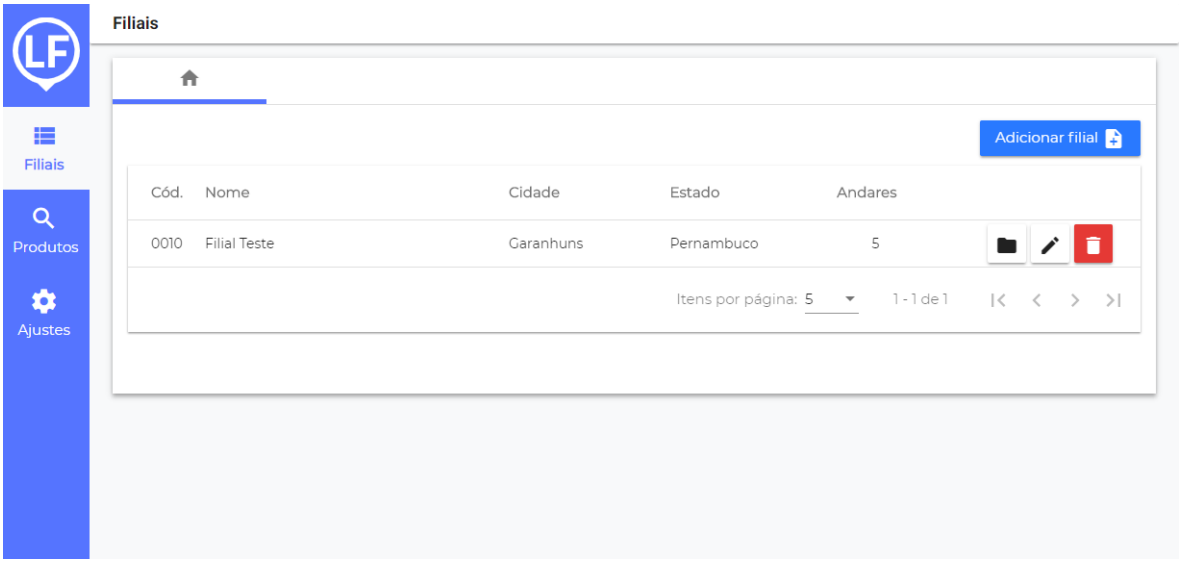
Figura 16 – Tela de pesquisa de produtos alocados em uma filial



Fonte: O autor

Levando em consideração que uma empresa pode possuir uma ou mais unidades de operação, além da unidade principal, foi desenvolvida uma área de gerenciamento de filiais, como pode ser vista na Figura 17. Aqui o usuário pode adicionar, editar, atualizar ou excluir uma filial. A exclusão de uma filial gera uma remoção em cascata de outras informações relacionadas a ela, sendo feita somente após a confirmação do usuário para evitar exclusões acidentais. Através desta tela também é possível chegar à tela de produtos alocados naquela filial.

Figura 17 – Tela de gerenciamento das filiais

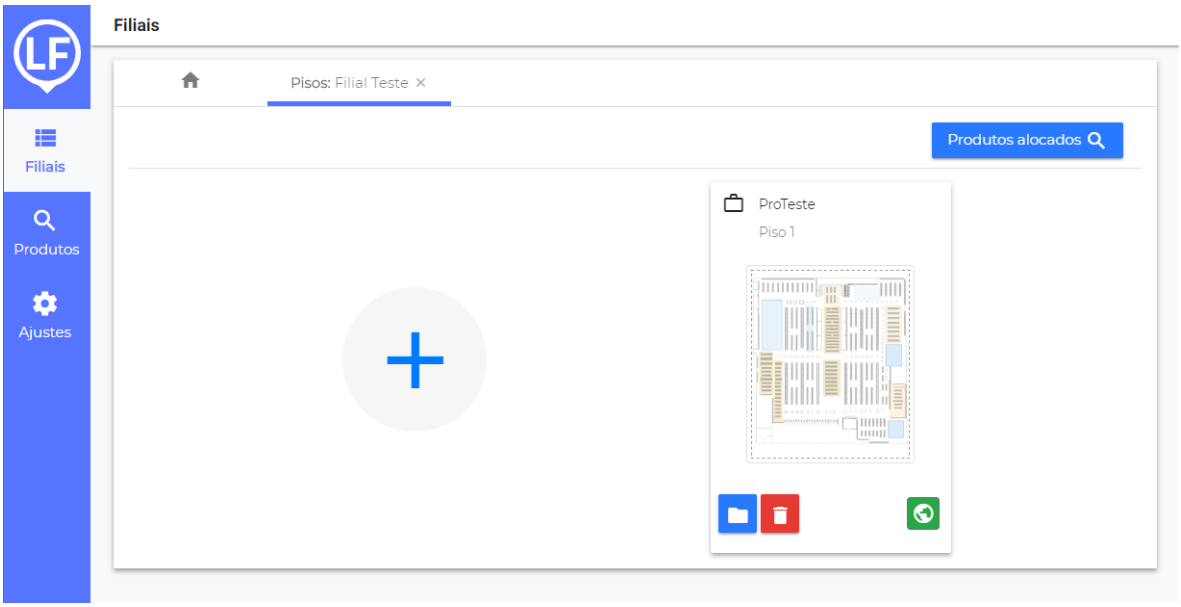


Fonte: O autor

A partir da lista de filiais, o usuário pode abrir a lista dos projetos daquela filial, através de um botão que leva à uma aba com tais projetos. Como visto na Figura 18, a lista de projetos exibe o nome do projeto, número do piso e uma miniatura da imagem que serve de esboço

do mapa (normalmente a planta baixa do andar). Além disso, o item possui um indicador se o projeto foi liberado para acesso fora do sistema, quando não liberado, os dados do projeto não podem ser vistos pelo aplicativo móvel.

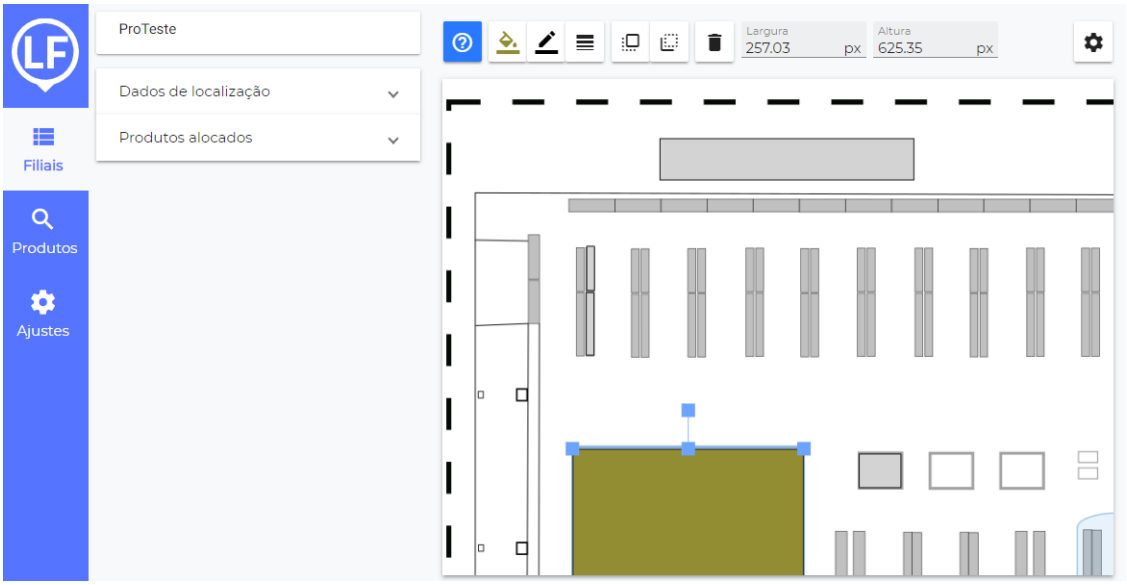
Figura 18 – Tela de gerenciamento de projetos de pisos



Fonte: O autor

A interface de edição de mapas é vista na Figura 19. Esta interface detém o principal propósito do sistema, nela o usuário pode adicionar aos mapas dos andares da empresa elementos customizáveis que abstraem os blocos (ou estantes) que armazenam os produtos na empresa. Cada elemento adicionado é identificado por um código gerado pelo sistema no momento em que o mesmo é salvo. Esses elementos são responsáveis por guardar a posição dos produtos dentro da loja, sendo que a localização é salva no formato de plano cartesiano (R^2) com coordenadas *(x, y)*. Como funcionalidades adicionais, o usuário pode exportar o mapa atual como uma imagem, publicar o mapa para que o mesmo fique disponível para acesso através do link da API, ou ainda, liberar os dados do projeto para uso pelo aplicativo móvel.

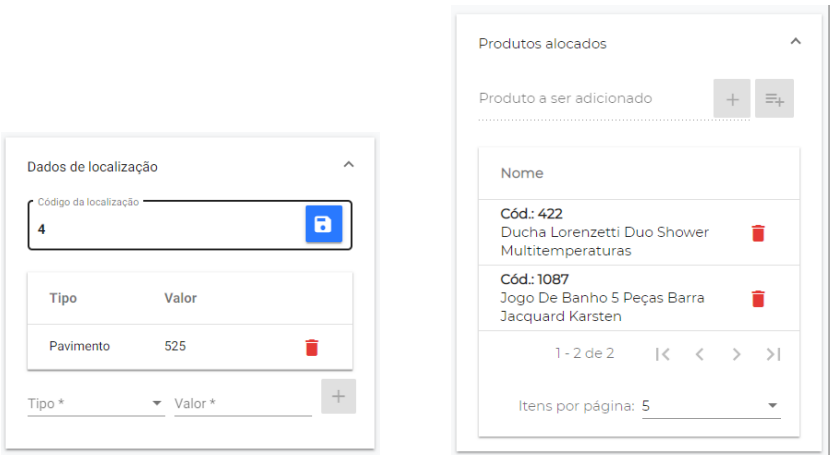
Figura 19 – Tela de edição do mapa de um projeto de piso



Fonte: O autor

Como visto na Figura 19 existem um estrutura *dropdown* contendo dois blocos com seus conteúdos retraídos: “Dados de localização” e “Produtos alocados”. Tais blocos são apresentados na Figura 20, sendo: *(a)* uma seção para visualização do código da localidade selecionada e suas classificações; *(b)* é a área para alocação de produtos à uma localidade/local de interesse.

Figura 20 – Aplicação web: tela de edição de mapas - características de uma localidade no mapa: (a) Bloco de edição de tipos de localidade; (b) Bloco de edição de produtos alocados

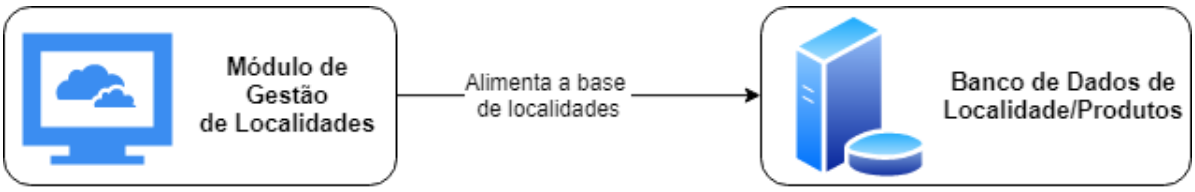


(a) (b)

Fonte: O autor

A Figura 21 ilustra como este módulo se comunica com outros elementos que compõem o sistema. Sendo utilizado apenas para gerenciamento das localidades de locais de interessa, as informações geradas neste módulo são armazenadas em um banco de dados para serem acessadas por outros módulos.

Figura 21 – Comunicação do módulo de gestão com outros módulos



Fonte: O autor

4.2 MÓDULO DE COLETA DE POSIÇÕES INDOOR

Como explicado na seção 3.7, foram construídas quatro bases para os testes dos algoritmos utilizados na pesquisa. De acordo com o mapa de pontos construídos, para cada base foram coletados os *fingerprints* de 169 locais espalhados no ambiente. O conjunto de características da base de dados "Médias" é formado pela interseção das caraterísticas das bases diárias (Dia 1, Dia 2 e Dia 3). O resumo das informações coletadas são descrito na Tabela 6.

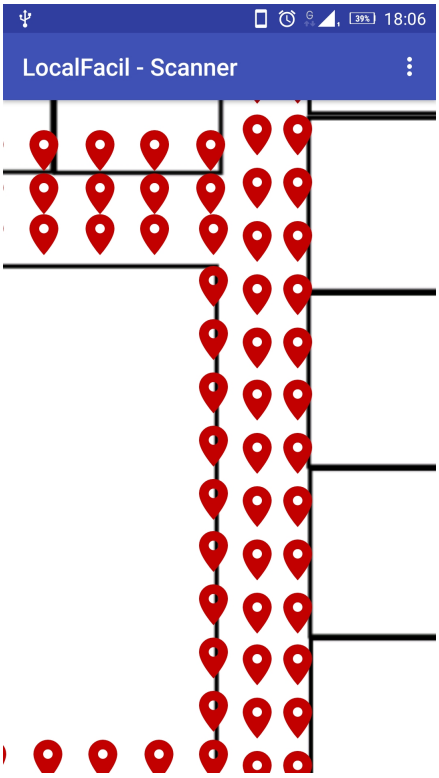
Tabela 6 – Quantidade de caraterísticas (BSSIDs) coletados em cada base

Base	**Nº de Características**
Dia 1	18
Dia 2	18
Dia 3	23
"Médias"	27

Fonte: O autor

O aplicativo responsável pela coleta das informações acima, possui uma tela com o mapa do andar sobreposto com marcadores para indicar os locais onde a coleta deve ser realizada. Ao clicar em um dos marcadores, o processo de obtenção de RSSIs é realizado e o marcador é removido do mapa para que não haja a sobreposição de dados, uma vez que o dado é identificado pela posição onde o marcador é posicionado. A Figura 22 mostra um *screenshot* da tela do aplicativo coletor de *fingerprints* (na fase de rastreamento).

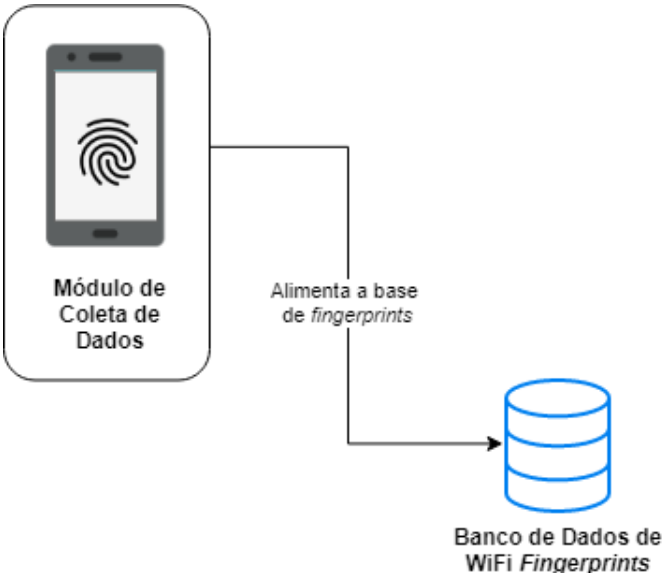
Figura 22 – Tela do módulo de coleta com os marcadores posicionados sobre o mapa



Fonte: O autor

A Figura 23 ilustra como este módulo se comunica com outros elementos que compõem o sistema. Sendo utilizado apenas para coleta de dados para alimentação da base de treinamento (calibração), ele se comunica apenas com a base que armazena tais dados.

Figura 23 – Comunicação do módulo de coleta com outros módulos



Fonte: O autor

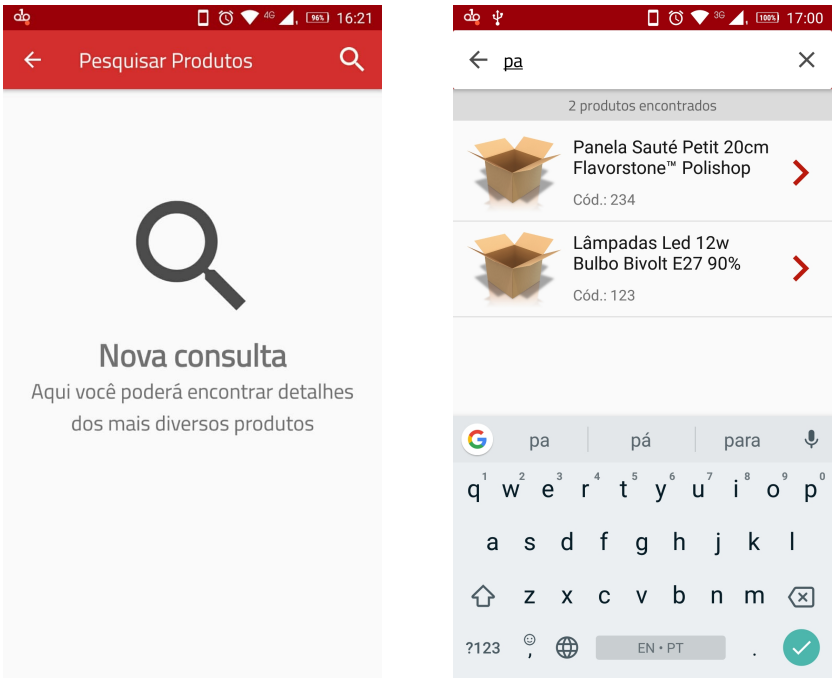
4.3 MÓDULO DE LOCALIZAÇÃO INDOOR

Além do LocalFácil, o usuário pode utilizar uma aplicação móvel desenvolvida para a plataforma Android, chamada de **AcheFácil**. Nessa aplicação é possível pesquisar por produtos existentes em uma filial. Pesquisando por sua descrição ou código, o usuário pode identificar o local exato do produto na filial. Na pesquisa, é exibida uma lista dos produtos que correspondem a sua pesquisa, a partir dessa lista é possível acessar o mapa do andar em que o produto está alocado, sendo indicado por um ícone de localização no local em que se encontra.

O processo de pesquisa e exibição dos resultados pelo aplicativo móvel (módulo de geolocalização) podem ser vistos de pesquisa e resultado podem ser vistos na Figura 24. No processo exemplificado, os produtos que contém o valor "pa" na sua descrição é exibido na tela, com um ícone que o leva para a tela de *pinpoint* no mapa.

A Figura 25 mostra o vetor direcional entre a posição do usuário dentro do ambiente em relação ao ponto da localização do produto (outrora cadastro pelo módulo de gestão) selecionado na tela de resultados (Figura 24b).

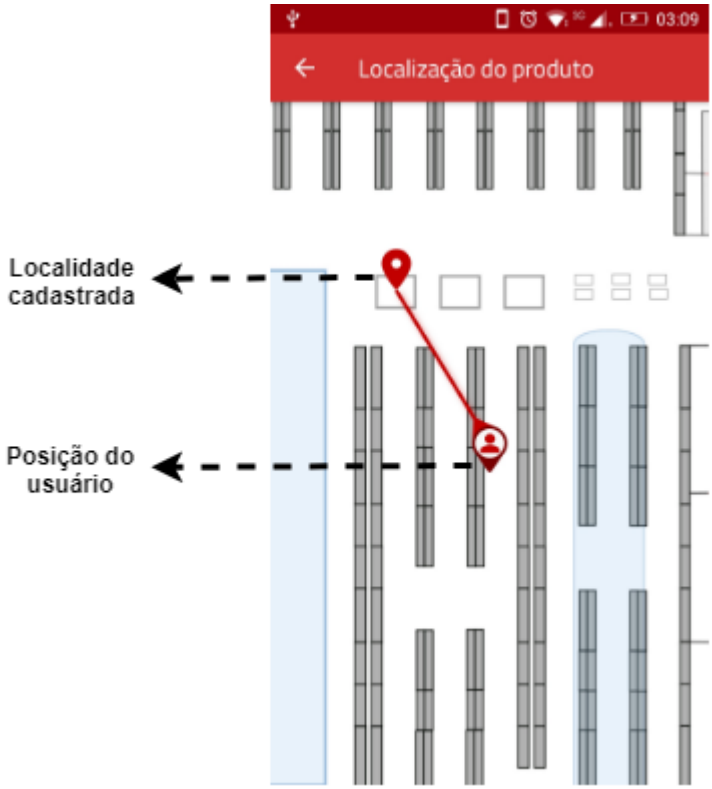
Figura 24 – Aplicativo móvel: tela do processo de pesquisa no aplicativo móvel: (a) Tela de pesquisa de produtos no aplicativo móvel; (b) Tela de resultados de pesquisas de produtos



(a) (b)

Fonte: O autor

Figura 25 – Aplicativo móvel: tela com o vetor direcional da posição do usuário até uma localidade cadastrada



Fonte: O autor

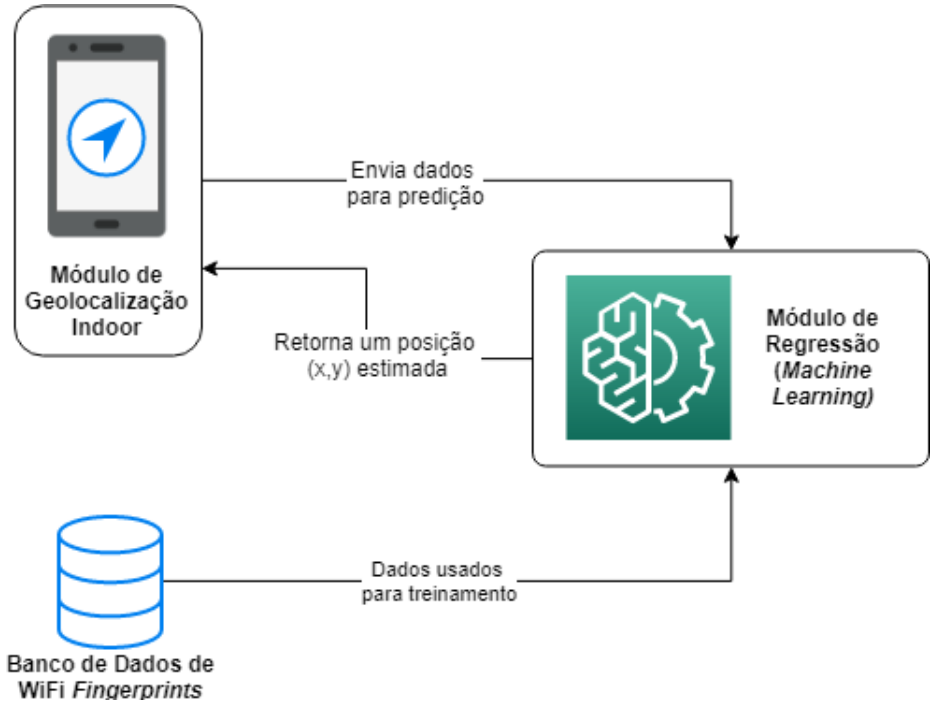
4.4 MÓDULO DE REGRESSÃO

As bases construídas durante a fase de calibração, possuem duas classes que identificam cada padrão usado para treinamento dos modelos primitivos, sendo elas: **x** e **y**. Utilizando os algoritmos AdaBoost, *Random Forest* e MLP, os modelos preditivos foram treinados e testados utilizando a técnica de validação cruzada *Leave One Out*. Esta técnica foi escolhida devido ao fato de que as bases usadas possuem poucos dados, e deste modo, compensou a falta de padrões em quantidade satisfatória para treinamento dos algoritmos.

Os experimentos executados foram feitos com os valores das classes em pixels (px). Para aceitação dos resultados obtidos, o valor da distância da entre o a posição real e posição estimada de um ponto (x, y) deve ser menor ou igual a 79 pixels, ou em uma escala do mundo real, aproximadamente 2 metros levando em consideração a imagem da planta baixa do primeiro andar do local de testes.

A Figura 26 ilustra como este módulo se comunica com outros elementos que compõem o sistema. O módulo tem acesso a base de dados da fase de calibração e executa duas vias de comunicação com o aplicativo de geolocalização, recebendo dados para predição de posições e enviando os resultados obtidos.

Figura 26 – Comunicação do módulo de regressão com outros módulos



Fonte: O autor

4.4.1 Resultados dos Métodos

As Tabelas 7, 8, 9 e 10 mostram os resultados de predição para as classes obtidos com cada base de dados. Os valores apresentados nas tabelas são: MSE (média do erro quadrático, do inglês *mean square error*), MAE (média do erro absotluto, do inglês *mean absolute error*)

Tabela 7 – Resultado dos testes realizados com base de dados "Dia 1"

Método	**Classe X**		**Classe Y**	
	MSE	**MAE**	**MSE**	**MAE**
AdaBoost	2350,922	29,007	2358,084	28,131
Random forest	6321,431	54,836	4436,222	42,302
Neural Network	9832,279	63,995	7154,644	55,822

Fonte: O autor

Tabela 8 – Resultado dos testes realizados com base de dados "Dia 2"

Método	**Classe X**		**Classe Y**	
	MSE	**MAE**	**MSE**	**MAE**
AdaBoost	2218,797	26,131	771,139	18,590
Random forest	3894,990	41,448	3115,381	38,729
Neural Network	5019,841	46,306	4015,228	41,111

Fonte: O autor

Tabela 9 – Resultado dos testes realizados com base de dados “Dia 3”

Método	**Classe X**		**Classe Y**	
	MSE	**MAE**	**MSE**	**MAE**
AdaBoost	1556,850	22,311	897,890	18,446
Random forest	3052,945	37,000	1802,758	27,335
Neural Network	2193,997	32,787	2888,768	34,863

Fonte: O autor

Tabela 10 – Resultado dos testes realizados com base de dados “Médias”

Método	**Classe X**		**Classe Y**	
	MSE	**MAE**	**MSE**	**MAE**
AdaBoost	936,100	20,272	553,492	16,314
Random forest	2086,023	31,268	1356,877	23,970
Neural Network	2239,775	34,301	3519,694	43,539

Fonte: O autor

Após a execução dos algoritmos de regressão, os resultados das classes foram analisados formando pontos em um plano cartesiano para comparação com a posição real do padrão testado. As Tabelas 11, 12, 13 e 14 apresentam os erros médios e os desvios padrões das distâncias entre os pontos real e predito de cada uma das bases. As tabelas citadas também mostram a quantidade de pontos com uma distância maior do que 2 metros (ou 79px).

Tabela 11 – Resultado da comparação entre os pontos real e predito para os padrões da base “Dia 1”

Método	**Erro médio (px)**	**Desvio padrão (px)**	**Erro real > 2m**
AdaBoost	48,074	48.961	20/169
Random Forest	78,232	69,324	57/169
MLP	96,576	87,574	70/169

Fonte: O autor

Tabela 12 – Resultado da comparação entre os pontos real e predito para os padrões da base “Dia 2”

Método	**Erro médio (px)**	**Desvio padrão (px)**	**Erro real > 2m**
AdaBoost	39,345	37,983	8/169
Random Forest	63,648	60,228	35/169
MLP	69,383	64,995	43/169

Fonte: O autor

Tabela 13 – Resultado da comparação entre os pontos real e predito para os padrões da base "Dia 3"

Método	**Erro médio (px)**	**Desvio padrão (px)**	**Erro real > 2m**
AdaBoost	36,306	33,728	5/169
Random Forest	48,377	40,747	22/169
MLP	59,022	49,638	40/169

Fonte: O autor

Tabela 14 – Resultado da comparação entre os pontos real e predito para os padrões da base "Médias"

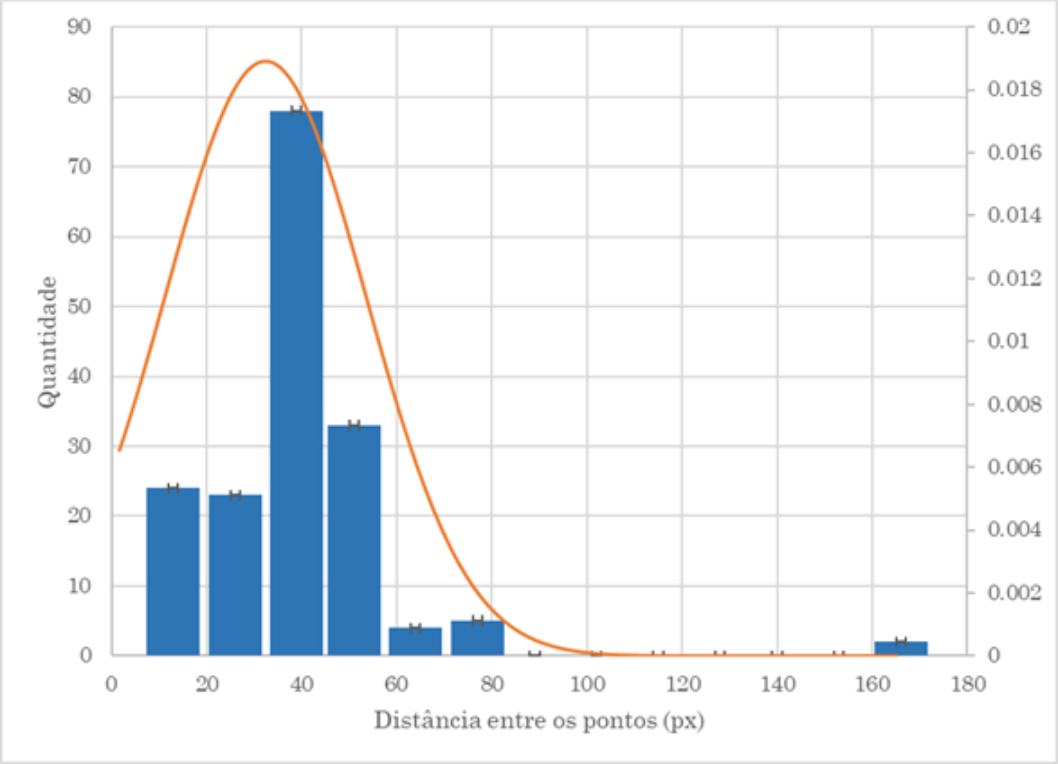
Método	**Erro médio (px)**	**Desvio padrão (px)**	**Erro real > 2m**
AdaBoost	32,379	21,089	2/169
Random Forest	44,460	38,477	12/169
MLP	62,719	42,871	43/169

Fonte: O autor

Com base nas informações contidas nas tabelas 7 - 14, pode-se constatar que o algoritmo de aprendizagem de máquina AdaBoost foi o que obteve melhores resultados em todos os casos de testes. Dentre as bases utilizadas, o algoritmo quando executado com a base "Médias" apresentou resultados extremamente significativos, com um erro médio de menos de 1 metro (menor que 39,5px) e um desvio padrão de pouco mais de meio metro (maior que 19.75px). Além disso, o resultado obtido com a base "Médias" apresentou apenas 2 pontos estimados acima do erro aceitável por este trabalho.

A Figura 27 mostra a distribuição dos erros de distância dos valores estimados da base "Médias", deixando nítido que a maior parte dos pontos estimados se encontra abaixo do erro tolerado (79px).

Figura 27 – Distribuição dos erros de distâncias entre os pontos estimados (com AdaBoost) e reais para a base “Médias”



Fonte: O autor

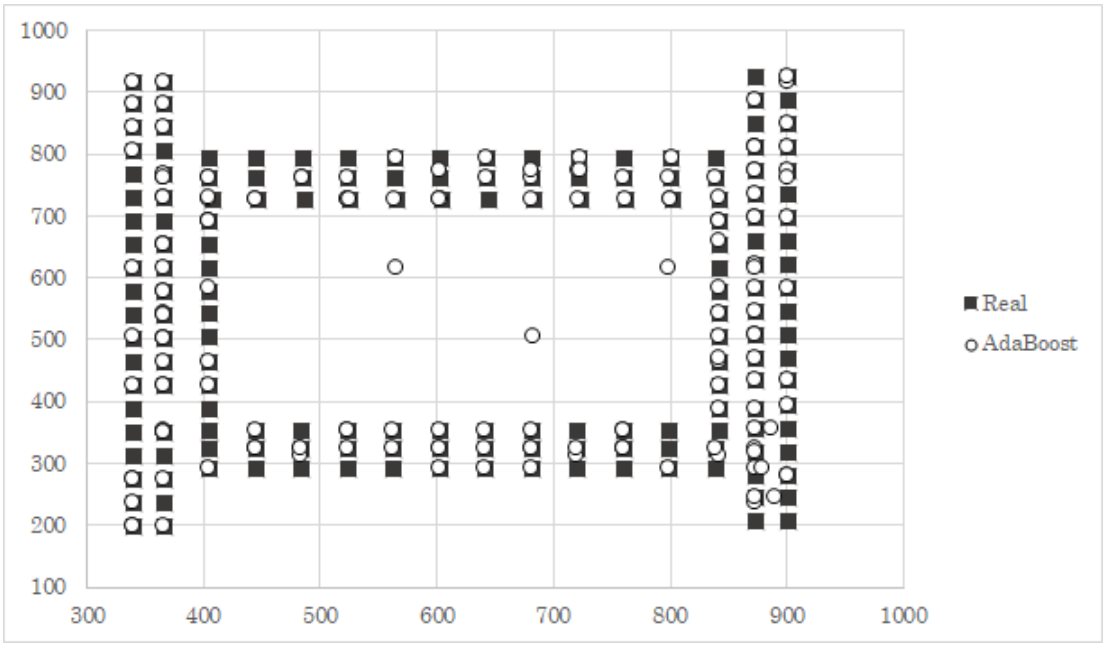
Dada a distância tolerável de dois metros obteve-se um resultado ótimo com o Adaboost e a base de médias. com objetivo de verificar qual seria a quantidade de erros gerados quando esse limiar de tolerância é diminuído, verificou-se os resultados para um metro (a) e um metro e meio (b), sendo que:

(a) Foram obtidos 36 erros de predição, o que é considerado um resultado ruim, chegando a mais de 20% do número de exemplos de treinamento; e,

(b) Foram obtidos 9 erros de predição, marginalizando um resulto bom, visto que apenas representa apenas 5% da base de treinamento.

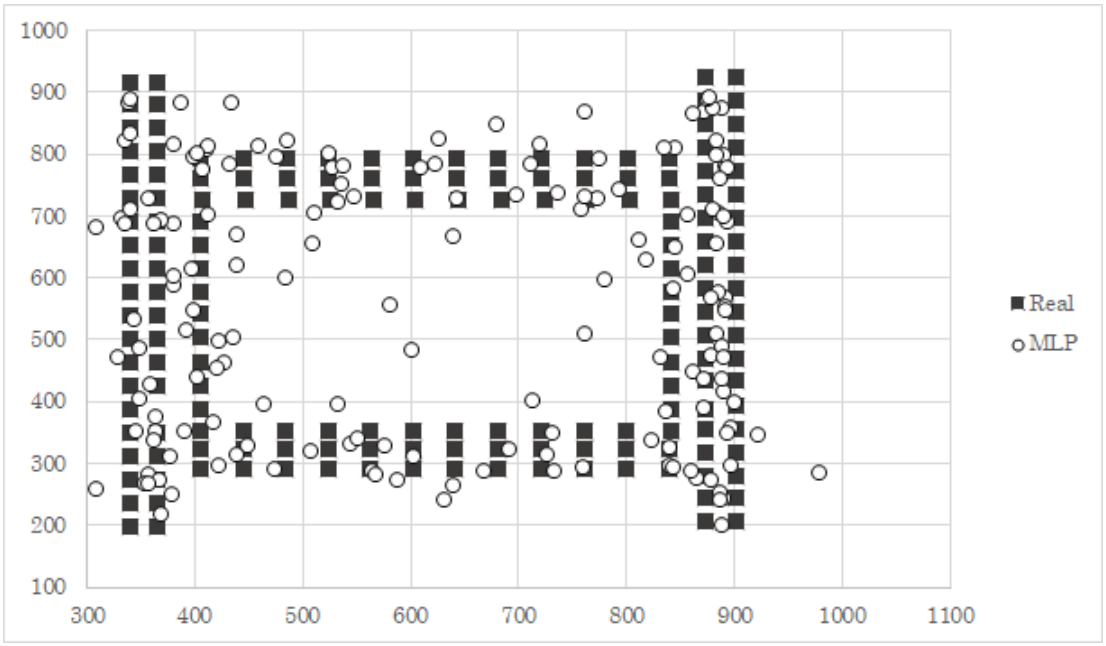
A Figura 28 mostra o mapa de pontos pré-definidos reais (quadrados) sobrepostos pelos pontos estimados (círculos) pelo AdaBoost na base de dados “Médias”, mostrando o quão próximo o modelo conseguiu chegar do cenário real, contando apenas com poucos *outliers*, diferente do resultado obtido pelo MLP, como visto na Figura 29. Os *outliers* visto na na figura (os três pontos posicionados no centro da área do mapa) estão destacados para indicar quais os pontos reais para os quais eles deveriam ser estimados, sendo que cada par (real e estimado) estão na mesma cor.

Figura 28 – Plotagem dos pontos estimados (com AdaBoost) em relação aos pontos reais utilizando a base “Médias”



Fonte: O autor

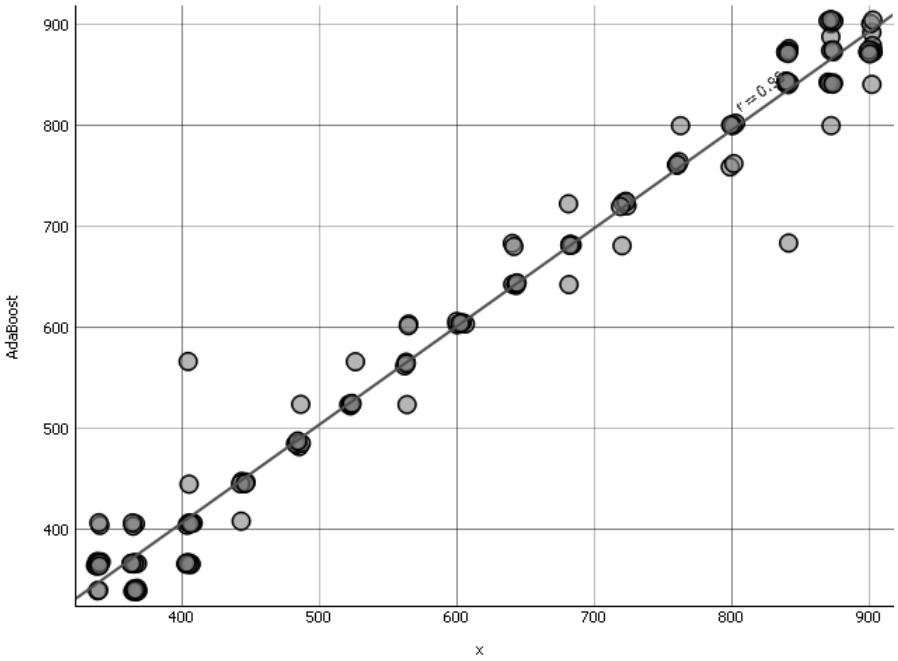
Figura 29 – Plotagem dos pontos estimados (com MLP) em relação aos pontos reais utilizando a base “Médias”



Fonte: O autor

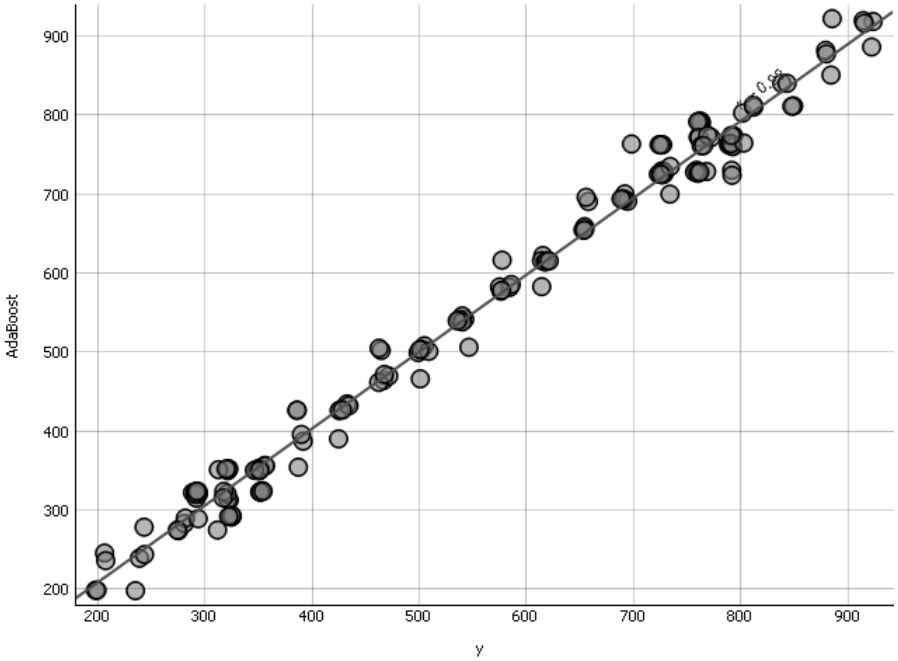
As Figura 30 e 31 também mostram que os valores preditos não se distanciam muito do valor real, ficando bem próximos da linha de regressão.

Figura 30 – Comparação do valor de “x” do ponto real com o do ponto estimado - Base “Médias” com AdaBoost



Fonte: O autor

Figura 31 – Comparação do valor de “y” do ponto real com o do ponto estimado - Base “Médias” com AdaBoost



Fonte: O autor

4.4.2 Testes de Hipótese

Como meio de validar os resultados obtidos nos experimentos, foi utilizado o Teste Z sobre os valores das distâncias entre os pontos reais e estimados para cada uma das bases usadas. O Teste Z é utilizado para testar a média de uma distribuição em que a variância já é conhecida

(MASSEY; MILLER, 2006). Para obtenção de um resultado precisamos fixar duas hipóteses: a nula e alternativa. Consideramos como hipótese nula, a afirmação de que a média amostral se iguala ou está acima de 2 metros (Equação 4.1), enquanto que a hipótese alternativa (a que queremos provar) afirma que a média amostral é inferior a 2 metros (Equação 4.2).

$$H_0 : \mu \leq \mu_0 \tag{4.1}$$

$$H_1 : \mu > \mu_0 \tag{4.2}$$

O teste foi executado de modo a considerar um intervalo de confiança de 95%. Assim, temos que o valor critico (c) sendo z_α = 1,65 (de acordo com a Tabela de Distribuição Normal Z), ou seja, valores a direita (maiores) que esse levam a rejeição da hipótese nula do teste. O cálculo do Teste Z é feito com a Equação 4.3:

$$Z = \frac{(\mu - \mu_0)}{\sigma} \tag{4.3}$$

Sendo, μ o valor da distância média máxima considerada aceitável (79px ou aproximadamente 2m) por este trabalho, μ_0 é a média da população amostral testada e σ indica o desvio padrão amostral. A hipótese nula será rejeitada quando $Z \geq z_\alpha$.

- **Base "Médias"**: apresentou como resultado o valor $Z = 2,21$. Logo, rejeitamos a hipótese nula ao nível de 95% de confiança e consideramos como verdadeira a afirmação sobre a distância média ser aceitável para a solução do problema aplicado nesta base.
- **Base "Dia 1"**: apresentou como resultado o valor $Z = 0,01$. Logo, a hipótese nula não é rejeitada e consideramos como falsa a afirmação sobre a distância média ser aceitável para a solução do problema aplicado nesta base.
- **Base "Dia 2"**: apresentou como resultado o valor $Z = 0,25$. Logo, a hipótese nula não é rejeitada e consideramos como falsa a afirmação sobre a distância média ser aceitável para a solução do problema aplicado nesta base.
- **Base "Dia 3"**: apresentou como resultado o valor $Z = 0,63$. Logo, a hipótese nula não é rejeitada e consideramos como falsa a afirmação sobre a distância média ser aceitável para a solução do problema aplicado nesta base.

Com base nos resultados estatísticos obtidos e apresentados anteriormente, podemos concluir que a base formada pela médias dos valores RSSI coletados em momentos distintos podem ser uma melhor opção para uso no modelo de regressão, em relação ao dados oriundos de uma base construída apenas com os valores de um único dia.

5 CONCLUSÃO

O trabalho dedicou-se a estudar métodos de regressão para solucionar o problema. A aplicação destes algoritmos possibilita estimar uma posição de acordo com dados já conhecidos do ambiente e com informações em tempo real do usuário. Dito isto, o algoritmo com maior nível de aceitação foi o Adaboost, apresentando um erro médio de menos de 1 metro e com a grande maioria dos testes resultando em uma margem tolerável de erro (até 2 metros) para o contexto da solução. Testes estatísticos mostraram que o resultado alcançado pelo Adaboost é aceitável, alcançando uma taxa de aceitação de 98,61%, com nível de confiança de 95% pelo teste de hipótese.

Além da ferramenta móvel desenvolvida, este trabalho gerou um componente para gestão da localização de objetos e ou locais de interesse para usuários em um ambiente, o LocalFácil. Esta ferramenta de gestão pode ser utilizada para diversos usos, como por exemplo, para: gestão de produtos alocados em uma empresa de varejo, identificação de obras em uma exposição de arte, além de aplicações para localização de estabelecimentos em shoppings ou aeroportos.

O uso do RSSI transmitido por redes WiFi, pode gerar distorções na predição dos resultados de acordo com os obstáculos existentes no ambiente. Além disso, em estruturas que possuem mais de um andar, é necessário realizar a identificação do mesmo de forma automática em uma aplicação móvel. Sensores nativos de dispositivos *mobile* podem ser usados para obter a altitude aproximada de um dispositivo, adicionando um meta dado para identificação do andar na base de treinamento e no momento de predição. Idealmente, as bases de treinamento deveriam possuir o máximo possível de instâncias para uma fase de calibração ter bons resultados. Contudo, neste trabalho o ambiente de testes não possuía uma extensão grande o suficiente, e devido a isto foram coletadas poucas instâncias, e mais tarde utilizou-se do *Leave One Out* para tentar contornar esse fato.

Possíveis trabalhos futuros incluem o aperfeiçoamento das fases de calibração e rastreamento. Apesar deste trabalho ter apresentado a prova de conceito com sucesso, a solução ainda precisa ser verificada em um ambiente com proporções consideráveis. A fase de calibração pode ser aperfeiçoada com o objetivo de torná-la mais ágil, uma vez que para ambientes muito extensos essa tarefa se torna cansativa e demanda muito tempo. Outro trabalho que pode ser realizado é o uso da fusão de tecnologias (como os sensores embutidos de dispositivos móveis) para melhorar os resultados alcançados com o Adaboost ou mesmo com outros algoritmos. Além destes, um possível caso de estudo seria a análise sobre o número de características da base de dados, buscando encontrar qual seria um tamanho ótimo para diminuição do erro obtido nos algoritmos utilizados. A forma de tratamento utilizada para os *missing values* nas bases também pode ser utilizada para estudo, buscando encontrar a melhor forma de tratá-los. Como o módulo de geolo-

calização utiliza a representação de um vetor para indicar a direção do usuário até um ponto de interesse, uma melhoria seria a possibilidade do módulo exibir um caminho dentro do ambiente.

REFERÊNCIAS

ALAN, R. H. V. et al. Design science in information systems research. *MIS quarterly*, Springer, v. 28, n. 1, p. 75–105, 2004. Citado 2 vezes nas páginas 27 e 28.

ALTINTAS, B.; SERIF, T. Improving RSS-based indoor positioning algorithm via k-means clustering. In: VDE. *17th European Wireless 2011-Sustainable Wireless Technologies*. [S.l.], 2011. p. 1–5. Citado 5 vezes nas páginas 15, 16, 19, 20 e 25.

APARICIO, S. et al. A fusion method based on bluetooth and wlan technologies for indoor location. In: IEEE. [S.l.], 2008. p. 487–491. Citado na página 26.

BENKIC, K. et al. Using RSSI value for distance estimation in wireless sensor networks based on ZigBee. In: IEEE. *2008 15th International Conference on Systems, Signals and Image Processing*. [S.l.], 2008. p. 303–306. Citado na página 19.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Citado na página 24.

CHAVES, B. B. *Estudo do algoritmo AdaBoost de aprendizagem de máquina aplicado a sensores e sistemas embarcados.* Tese (Doutorado) — Universidade de São Paulo, São Paulo, 2012. Citado 2 vezes nas páginas 24 e 25.

COSTA, L. J. S. *Técnica de localização em ambientes fechados utilizando padrões de redes sem fio*. Dissertação (Mestrado) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2014. doi:10.11606/D.3.2016.tde-13072016-161143. Citado 2 vezes nas páginas 15 e 19.

CROCKFORD, D. The application/json media type for javascript object notation (json). 2006. Citado na página 34.

DATA SCIENCE ACADEMY. *Deep Learning Book*. 2019. Disponível em: <http://deeplearningbook.com.br/a-arquitetura-das-redes-neurais/>. Acesso em: 5.7.2019. Citado 2 vezes nas páginas 22 e 23.

DEITEL, H.; DEITEL, P.; DEITEL, A. *Android: Como programar*. [S.l.]: Bookman Editora, 2015. Citado na página 36.

DEVELOPERS, A. *Platform Architecture*. 2019. Disponível em: <https://developer.android.com/guide/platform>. Acesso em: 2.7.2019. Citado na página 36.

EL-RABBANY, A. Introduction to gps: The global positioning system. Artech house, Londres, 01 2006. Citado 2 vezes nas páginas 15 e 18.

EXPRESS/NODE introduction. 2019. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction>. Acesso em: 20.01.2019. Citado 2 vezes nas páginas 31 e 33.

FACELI, K. et al. Métodos baseados em otimização. In: ____. *Inteligência Artificial: Uma abordagem de aprendizado de máquina*. Rio de Janeiro: LTC, 2011. p. 108–122. ISBN 9788521618805. Citado 2 vezes nas páginas 22 e 23.

FARID, Z.; NORDIN, R.; ISMAIL, M. Recent advances in wireless indoor localization techniques and system. *Journal of Computer Networks and Communications*, Hindawi, v. 2013, 2013. Citado 3 vezes nas páginas 15, 16 e 19.

FERREIRA, R. *REST: Princípios e boas práticas*. 2017. Disponível em: <http://blog.caelum.com.br/rest-principios-e-boas-praticas/>. Citado na página 35.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Irvine, USA, 2000. v. 7. Citado na página 34.

FLANAGAN, D. *JavaScript: The Definitive Guide Activate Your Web Pages*. 6th. ed. [S.l.]: O'Reilly Media, Inc., 2011. ISBN 0596805527, 9780596805524. Citado na página 30.

FREUND, Y.; SCHAPIRE, R. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, JAPANESE SOC ARTIFICIAL INTELL, v. 14, n. 771-780, p. 1612, 1999. Citado na página 24.

GOGOLAK, L.; PLETL, S.; KUKOLJ, D. Neural network-based indoor localization in wsn environments. *Acta Polytechnica Hungarica*, v. 10, n. 6, p. 221–235, 2013. Citado na página 26.

GOOGLE NoSQL: as principais vantagens de um banco de dados não relacional. 2017. Disponível em: <https://www.qinetwork.com.br/google-nosql-e-suas-vantagens/>. Acesso em: 31.01.2019. Citado na página 36.

HAVERBEKE, M. *Eloquent JavaScript: A Modern Introduction to Programming*. No Starch Press, Incorporated, 2018. ISBN 9781593279509. Disponível em: <https://books.google.com.br/books?id=ZwL9swEACAAJ>. Citado na página 30.

IHRIG, C. J.; BRETZ, A. *Full Stack JavaScript Development With MEAN*. 1st. ed. [S.l.]: Sitepoint, 2015. ISBN 0992461251, 9780992461256. Citado 5 vezes nas páginas 31, 32, 33, 34 e 35.

JAMES, M. *Android Programming in Kotlin: Starting With An App*. 1st. ed. [S.l.]: I/O Press, 2017. ISBN 1871962544, 9781871962543. Citado na página 36.

KAPLAN, E.; HEGARTY, C. *Understanding GPS: principles and applications*. [S.l.]: Artech house, 2005. Citado na página 18.

LÓSCIO, B. F.; OLIVEIRA, H. R. d.; PONTES, J. C. d. S. Nosql no desenvolvimento de aplicações web colaborativas. *VIII Simpósio Brasileiro de Sistemas Colaborativos*, v. 10, n. 1, p. 11, 2011. Citado 2 vezes nas páginas 35 e 36.

MASSEY, A.; MILLER, S. J. Tests of hypotheses using statistics. Mathematics Department, Brown University, Providence, RI 2912, p. 1–32, 2006. Citado na página 57.

MONGODB and MySQL Compared. Disponível em: <https://www.mongodb.com/compare/mongodb-mysql>. Acesso em: 25.01.2019. Citado na página 31.

NAQA, I. E.; MURPHY, M. J. What is machine learning? In: *Machine Learning in Radiation Oncology*. [S.l.]: Springer, 2015. p. 3–11. Citado 2 vezes nas páginas 20 e 21.

PEFFERS, K. et al. A design science research methodology for information systems research. *Journal of management information systems*, Taylor & Francis, v. 24, n. 3, p. 45–77, 2007. Citado na página 28.

RÄTSCH, G. A brief introduction into machine learning. *Friedrich Miescher Laboratory of the Max Planck Society*, 2004. Citado na página 20.

REZENDE, S. O. *Sistemas inteligentes: fundamentos e aplicações*. [S.l.]: Editora Manole Ltda, 2003. 89-114 p. Citado na página 20.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, American Psychological Association, v. 65, n. 6, p. 386, 1958. Citado na página 22.

SANTANA, A. T. F. d. Enterprise architecture analysis based on network paradigm: a framework proposal and empirical evaluation. Universidade Federal de Pernambuco, 2017. Citado na página 28.

SHALA, U.; RODRIGUEZ, A. *Indoor Positioning using Sensor-fusion in Android Devices*. Dissertação (Mestrado) — School of Health and Society, Department Computer Science, Kristianstad University, Kristianstad, 01 2011. Citado 5 vezes nas páginas 15, 16, 18, 19 e 25.

SHORT, M. *What is Express? A Guide for Beginners*. 2018. Disponível em: <https://www.coursereport.com/blog/what-is-express>. Acesso em: 20.01.2019. Citado na página 31.

STACK Overflow Developer Survey 2018. 2018. Disponível em: <https://insights.stackoverflow.com/survey/2018#technology>. Acesso em: 26.01.2019. Citado na página 32.

STATCOUNTER. 2019. (Global Stats). Disponível em: <http://gs.statcounter.com/os-market-share/mobile/worldwide>. Acesso em: 1.7.2019. Citado na página 36.

WHAT IS REST – Learn to create timeless RESTful APIs. Disponível em: <https://restfulapi.net/>. Acesso em: 25.01.2019. Citado na página 34.

WOHLIN, C.; AURUM, A. Towards a decision-making structure for selecting a research design in empirical software engineering. *Empirical Software Engineering*, Springer, v. 20, n. 6, p. 1427–1455, 2015. Citado na página 27.

ZEGEYE, W. et al. Wifi RSS fingerprinting indoor localization for mobile devices. *2016 IEEE 7th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, p. 1–6, 10 2016. doi:10.1109/UEMCON.2016.7777834. Citado 3 vezes nas páginas 15, 19 e 25.