Análise exploratória de dados geográficos de componente nativo para aplicativos híbridos

Bruno M. N. de Castro[1], **Fábio Henrique M. Oliveira**[1]

[1] Instituto Federal de Brasília

bruno.castro@estudante.ifb.edu.br, fabio.oliveira@ifb.edu.br

Abstract. *The development of hybrid mobile applications is accelerated and low cost, favoring the production of multiplatform prototypes. However, hybrid development can compromise the performance of components that utilize native resources. In order to validate the data collected by a geolocation feature available for the Ionic framework, a hybrid mobile application for component data collection and an exploratory analysis of the collected data, and its proximity to the location data of a native solution, were developed. Thus the Haversine formula was used to evaluate the data quality, which presented considerable distance between the records.*

Resumo. *O desenvolvimento de aplicativos móveis híbridos é acelerado e de baixo custo, favorecendo a produção de protótipos multiplataforma. Entretanto, o desenvolvimento híbrido pode comprometer o desempenho dos componentes que utilizam recursos nativos. Com o objetivo de validar os dados coletados com um recurso de geolocalização disponível para o framework Ionic, foi desenvolvido um aplicativo móvel híbrido para coleta de dados do componente e apresentada uma análise exploratória sobre os dados coletados, e sua proximidade com os dados de localização de uma solução nativa. Com isso, foi utilizada a fórmula de Haversine para avaliar a qualidade dos dados, que apresentaram distância considerável entre os registros.*

1. Introdução

Nos últimos anos, os telefones celulares melhoraram drasticamente em design e funcionalidade, desde dispositivos simples de ligação e mensagens de texto até sofisticados minicomputadores chamados *smartphones* [Kirwan et al. 2013].

Esses dispositivos têm sido fabricados com um rico conjunto de sensores embarcados, como acelerômetro, giroscópio, Sistema de Posicionamento Global (GPS), microfone e câmera, permitindo aplicações em diversos domínios, como assistência médica, segurança, monitoramento ambiental e transporte [Lane et al. 2010].

Os dados coletados por sensores são utilizados por aplicativos nativos para dispositivos móveis, impossibilitando a reutilização de código em outras plataformas [IBM 2012]. Deste modo, a avaliação do custo relacionado à manutenção de código nativo para produtos multiplataforma direcionou a indústria ao fomento de iniciativas de desenvolvimento de aplicativos móveis híbridos.

Os aplicativos híbridos são desenvolvidos com tecnologias da *web*, como *HTML* e *CSS*, e a comunicação com as APIs de cada plataforma é feita em *Javascript*

[Malavolta et al. 2015]. A abordagem simplifica o processo de desenvolvimento, porém é necessário considerar as restrições de acesso a recursos de hardware e a queda de *desempenho* que a alternativa pressupõe [Malavolta et al. 2015].

Com o objetivo de esclarecer o seu atendimento a requisitos de novas soluções, a academia tem se empenhado na produção de avaliações de *desempenho* dessas ferramentas híbridas. Caso a comunicação desses mecanismos seja relacionada a sensores de um dispositivo móvel, torna-se relevante a verificação do desempenho de componentes para aplicativos híbridos [Amatya and Kurti 2014].

Durante a validação de um aplicativo para monitoramento de alunos em eventos do Instituto Federal de Brasília, surgiu a necessidade de detalhamento dos registros de pontos geográficos localizados à borda do raio de presença. O experimento evidenciou que poderia ocorrer o registro de dados imprecisos.

O aplicativo foi desenvolvido com o conceito de aplicativos multiplataforma, com o *framework* para desenvolvimento de aplicativos híbridos *Ionic*, o que levou ao questionamento sobre a qualidade de dados de sensores gerados por aplicativos de natureza híbrida.

Deste modo, este trabalho propõe uma análise de dados exploratória (ADE) sobre os dados coletados pelo componente *Background Geolocation*, utilizado para o desenvolvimento da solução com o *framework* Ionic. Em conjunto à análise de exatidão, no decorrer do artigo são descritos os detalhes da arquitetura da solução e o método para avaliação do componente.

Na seção 2, são apresentados Materiais e Métodos para o *design* da solução. Na seção 3, complementa-se o estudo com Resultados e discussão acerca da avaliação do desempenho do componente. A Seção 4 trata-se da conclusão, contendo planos futuros de incremento à pesquisa.

2. Materiais e Métodos

2.1. Arquitetura da solução

Para o estudo, foram selecionadas ferramentas para composição da arquitetura cliente-servidor que considerassem a natureza móvel que a solução necessitava (Figura 1). Para garantia da disponibilidade do serviço e da abstração da atividade de manutenção de infraestrutura, decidiu-se por hospedar o serviço em uma plataforma em nuvem.

A solução de cliente é composta por um aplicativo híbrido desenvolvido com o *framework Ionic* e os componentes *Background Geolocation*, *Device* e *HTTP* para, respectivamente: (1) coletar dados de localização em plano de fundo, (2) fazer registro único para cada dispositivo conectado, e (3) executar requisições para a API.

A solução de servidor, hospedada na plataforma em nuvem *Heroku*, é composta por uma API REST, desenvolvida com *Django Rest Framework* e versionamento de código integrado ao *GitHub*, e um banco de dados Postgre.

A escolha pelo desenvolvimento de um aplicativo caracterizou-se pela facilidade de acesso a sensores e dados proporcionada por esses dispositivos. Portanto, o aplicativo, utilizando recursos do *smartphone*, é responsável por coletar dados de localização do usuário e enviá-los para uma API REST. Em seguida, a API, ao receber a requisição do

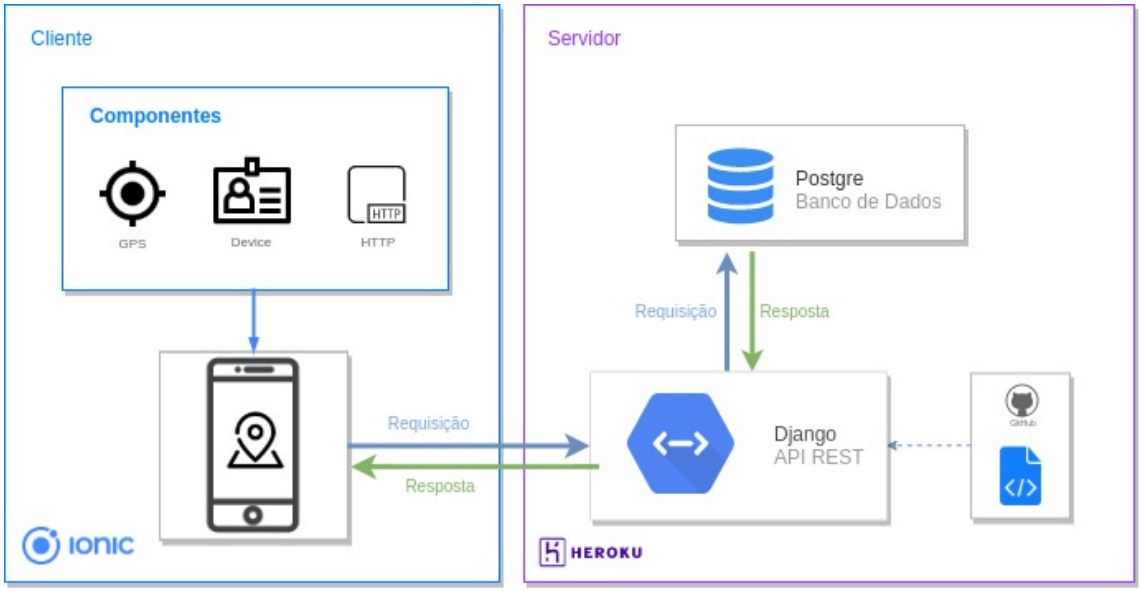

Figura 1. Arquitetura da solução

dispositivo, registra os dados de localização no banco de dados e retorna uma mensagem de status.

2.2. Django REST Framework

A API foi desenvolvida com o *Django Rest Framework*, uma ferramenta baseada no *framework Django* para facilitar o desenvolvimento de APIs e a adequação a normas de segurança. A solução apresenta painel com interface gráfica para visualização e interação com a API desenvolvida [Christie 2019].

2.3. GitHub

GitHub é uma plataforma de hospedagem de código-fonte que permite o desenvolvimento colaborativo de software e o compartilhamento de projetos [Dabbish et al. 2012]. A ferramenta é viabilizada pelo sistema de controle de versão de arquivos distribuído *Git*. As duas ferramentas foram utilizadas para versionamento de código e acompanhamento de melhorias da API e do aplicativo.

O *Github* dispõe de soluções para *Integração Contínua* (IC) e *Automação de Processos*. A IC refere-se à automatização das etapas de construção e implantação de um *software* em uma *sandbox*, caixa-de-areia, para que seja possível executar uma coleção de testes quando uma contribuição for submetida, o que garante aumento de produtividade e qualidade de código [Vasilescu et al. 2015], sendo, portanto, incorporada ao projeto da API.

2.4. Heroku

Em desenvolvimento desde junho de 2007, o *Heroku* é uma plataforma de aplicação em nuvem que suporta hospedagem, monitoramento e implantação de projetos. A plataforma suporta projetos desenvolvidos em linguagens de programação como *Node.js*, *Ruby*, *Python*, *Java*, *PHP*, *Go*, *Scala*, *Clojure* e *Kotlin* [Ionic 2019].

Os projetos são organizados em aplicações *Heroku*, que são passíveis de integração com repositórios hospedados no *GitHub*. O recurso que pode ser utilizado tanto em repositórios públicos, quanto privados.

Para emular ambientes de produção, homologação ou desenvolvimento, seleciona-se a *branch* para cada fase do projeto no painel de controle da aplicação de acordo com a *branch* especificada para cada um e outras definições de projeto. Sendo uma Plataforma como serviço (PaaS) em nuvem, o ambiente oferecido assegura a abstração de diversas etapas de implantação e configuração dos serviços de sustentação das aplicações, garantindo mais força de trabalho para atividades da área de desenvolvimento.

No contexto de virtualização, o *Heroku* utiliza *containers Linux*, chamados *Dyno*. Cada *Dyno* dispõe de especificações adequadas ao tamanho do projeto (Tabela 1), sendo possível migrar a aplicação para outras configurações de *cluster*, de acordo com a necessidade do projeto.

Tabela 1. Especificações de containeres Dyno.

Tipo de Dyno	Memória RAM	CPU Share	Computação	Dedicado	Hibernação
free	512 MB	1x	1x-4x	não	sim
hobby	512 MB	1x	1x-4x	não	não
standard-1x	512 MB	1x	1x-4x	não	não
standard-2x	1024 MB	2x	4x-8x	não	não
performance-m	2.5 GB	100%	11x	sim	não
performance-l	14 GB	100%	46x	sim	não

Considerando o cenário de desenvolvimento de um protótipo, o tipo de *Dyno* escolhido foi o gratuito, por fornecer, além das especificações da Tabela 1, um banco de dados Postgre com capacidade de escrita de 10 mil linhas.

2.5. Ionic

Ionic é um framework *open-source* para desenvolvimento híbrido de aplicativos móveis e Aplicações da Web Progressivas (PWA) que proporciona alto desempenho *native-like* [Wang et al. 2017]. A sua codificação é feita em linguagem *TypeScript*, uma variação do *Javascript*, com tipagem forte.

2.5.1. Plugins

O *Ionic* possui uma biblioteca de plugins desenvolvidos pela comunidade para implementação de funcionalidades nativas e não-nativas, chamada *Ionic Native*. A biblioteca possui as edições comunitária, gratuita e mantida pela comunidade, e empresarial, paga e recomendada para desenvolvimento de aplicativos de missão crítica e de larga escala, com serviços de garantia de estabilidade e compatibilidade com outros sistemas e versões [Ionic 2019].

Sendo somente da edição Comunitária, os plugins utilizados no projeto são:

- Background Geolocation (versão 3.0.0-alpha.50): permite a execução do componente de localização do dispositivo em plano de fundo e a redução de recursos utilizados, contribuindo para o aumento de autonomia da bateria. Para evitar a interrupção durante a coleta de dados, o componente restringe o fechamento acidental do aplicativo;

- HTTP (versão 4.20.0): possibilita o estabelecimento de conexão com servidores HTTP, necessária para submissão dos dados de geolocalização registrados durante a coleta para a API;
- Device (versão 4.20.0): fornece informações do dispositivo, incluindo o número identificador único do dispositivo (uuid), utilizado para garantir que os dados separados para a análise sejam de um mesmo dispositivo.

O plugin *Background Geolocation* dispõe de parâmetros para ajustes de comportamento do componente (Tabela 2). Na coluna *valor definido* são apresentadas as configurações utilizadas para desenvolvimento do aplicativo, onde foi considerado o equilíbrio entre frequência e qualidade dos dados a serem coletados.

Tabela 2. Parâmetros do plugin Background Geolocation.

Parâmetro	Tipo	Plataforma	Descrição	Valor padrão	Valor definido
locationProvider	Numérico	Todas	Define o provedor de localização	DISTANCE _FILTER _PROVIDER	DISTANCE _FILTER _PROVIDER
desiredAccuracy	Numérico	Todas	Acurácia desejada, em metros. Outros Valores possíveis: *HIGH_ACCURACY*, *MEDIUM_ACCURACY*, *LOW_ACCURACY*, *PASSIVE_ACCURACY*.	MEDIUM _ACCURACY	10
stationaryRadius	Numérico	Todas	Raio estacionário em metros.	50	1
debug	Booleano	Todas	Emite sons para eventos do ciclo de vida do background geolocation.	false	false
distanceFilter	Numérico	Todas	Distância mínima (em metros) em que o dispositivo deve se mover para que um evento de atualização seja gerado	500	1
stopOnTerminate	Booleano	Todas	Força parada de registro quando a aplicação é finalizada	true	false
startOnBoot	Booleano	Android	Executa serviço de plano de fundo durante a inicialização do dispositivo	false	false
interval	Numérico	Android	Intervalo mínimo para atualizações de localização	60000	10000
fastestInterval	Numérico	Android	Taxa mais rápida, em milissegundos, em que o aplicativo pode receber atualizações de localização	120000	120000
activitiesInterval	Numérico	Android	Taxa, em milissegundos, em que uma atividade de registro ocorre.	10000	10000
notificationsEnabled	Booleano	Android	Ativa ou desativa notificações de local durante rastreamento e sincronização de localizações	true	true
startForeground	Booleano	Android	Permite que o serviço de sincronização de localização seja executado em estado de primeiro plano.	false	false

2.6. Dados Google

A coleta de dados do Google determina a localização em graus de precisão relativos à utilização de sensores, como GPS e demais sensores do dispositivo. Para adquirir melhor exatidão, dados complementares são coletados por meio de informações de pontos de acesso *Wi-fi*, torres de celular e dispositivos com *Bluetooth* ativado [Google 2019].

Em dispositivos com sistema operacional *Android*, a API de localização nativa registra dados de localização enquanto o componente de localização está ativado. Os dados são coletados sempre que o dispositivo estiver autorizado a fazer esta coleta, e são estruturados como disposto na Tabela 3.

2.7. Coleta de dados

A coleta de dados teve duração de 4 dias, ininterruptos, sendo executada por meio de duas fontes de dados em um único dispositivo, Asus Zenfone 4, modelo *ZE554KL* (Tabela 4).

Tabela 3. Estrutura da tabela de dados de localização Google.

Propriedade	Descrição
timestampMs	Data do registro no formato UNIX Timestamp, em milissegundos
latitudeE7	Latitude 10e7
longitudeE7	Longitude 10e7
accuracy	Acurácia do registro
activity	Predição de atividade executada durante o registro, como "EM_VEÍCULO", "PARADO", "EM_BICICLETA", "INCLINAÇÃO", "A PÉ" e "DESCONHECIDO"
velocity	Velocidade
heading	Posição
altitude	Altitude
verticalAccuracy	Acurácia vertical

Tabela 4. Especificações do *smartphone* Asus Zenfone 4 (ZE554KL)

Armazenamento interno	eMCP 64GB
Processador	Processador Qualcomm® Snapdragon™ 630 Octa-core com 14nm
GPU	Qualcomm® Adreno™ 508
Memória RAM	4GB
Câmera Principal	Sensor SONY IMX362 12 megapixels
Câmera Frontal	8 megapixels
Wi-fi	WLAN 802.11 a/b/g/n/ac
Bluetooth	Versão 5.0
Sensores	GPS, AGPS, GlONASS, BDS
Padrões de Rede	TD-LTE, FDD-LTE, TD-SCDMA, WCDMA/HSPA+/DC-HSPA+, CDMA 2000, GSM/GPRS/EDGE
Velocidade de transferência	LTE Cat6 300/50 Mbps DC-HSPA+: UL 5.76 / DL 42 Mbps

A primeira foi construída a partir da coleta feita por meio do aplicativo, com o componente *Background Geolocation*, e enviada para a API hospedada no *Heroku*, que, por fim, registra os dados em um banco de dados.

Entretanto, por tratar-se de um componente em fase de prototipação, notou-se a necessidade de etapa de validação, onde a coleta foi equiparada com dados do histórico de dados de localização coletados pelo Google, de fonte externa, sendo esta a segunda fonte de dados.

Durante o monitoramento, os momentos de deslocamento foram caracterizados pela utilização de meios de transporte, como ônibus e carro, e por passeios a pé. Além disso, de 22h às 5h o dispositivo foi mantido imóvel.

3. Resultados e Discussão

O impacto da pesquisa está em esclarecer o desempenho da ferramenta e qual a margem de exatidão que o componente de desenvolvimento híbrido pode garantir no que tange à coleta de dados de sensores de geolocalização.

Inicialmente, o arquivo de dados de localização do *Google* apresentava um total de 270.790 registros, considerando dados de 27 de maio de 2015 a 22 de setembro de 2019. Em contrapartida, a base de dados gerada a partir do aplicativo, coletada entre 17 e 20 de setembro de 2019, totalizava 467 registros.

Para permitir a equiparação dos registros, a primeira etapa consistiu na filtragem dos valores da base de dados *Google* com o critério de exclusão de valores registrados antes de 17 de setembro de 2019 e após 20 de setembro de 2019, implicando na remoção de 266.520 linhas, que passou a totalizar 4.270 registros.

Investigando o intervalo entre cada registro dos 4.270 remanescentes da base do *Google*, notou-se a existência de 1605 registros com valores discrepantes. Estes valores foram caracterizados por intervalos maiores do que 60 segundos, e tratam-se de casos em que houve (1) falha de conexão com a internet, (2) desligamento temporário do recurso de localização, e (3) momentos de longa parada. Com isso, na Figura 2 foi possível visualizar a distribuição temporal dos registros.

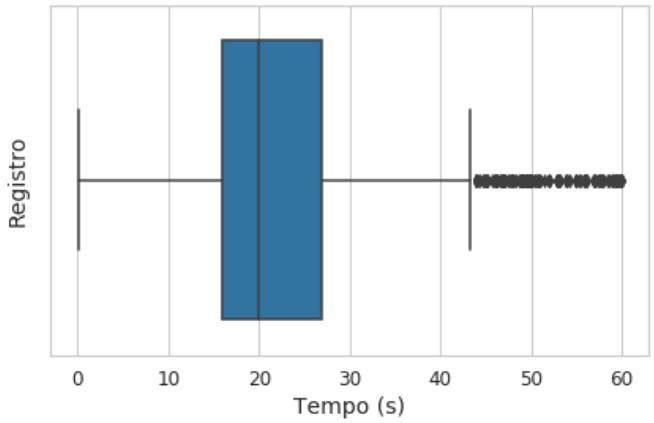

Figura 2. Distribuição de intervalo entre registros da base de dados Google

Em 2.665 registros, os dados acompanharam uma média de 22,73 segundos entre os registros, com desvio-padrão de 10,65 segundos, sendo metade da amostra sendo composta por valores abaixo de 20 segundos. Para redução de ruído, foram removidas 1.103 linhas com coordenadas duplicadas.

Com 3167 registros de localização da base do Google e 467 registros do banco de dados *Postgre*, foi possível visualizar tendências correlacionais na Figura 3, em que as coordenadas geográficas do aplicativo são representadas com a cor azul, e da base externa, em vermelho.

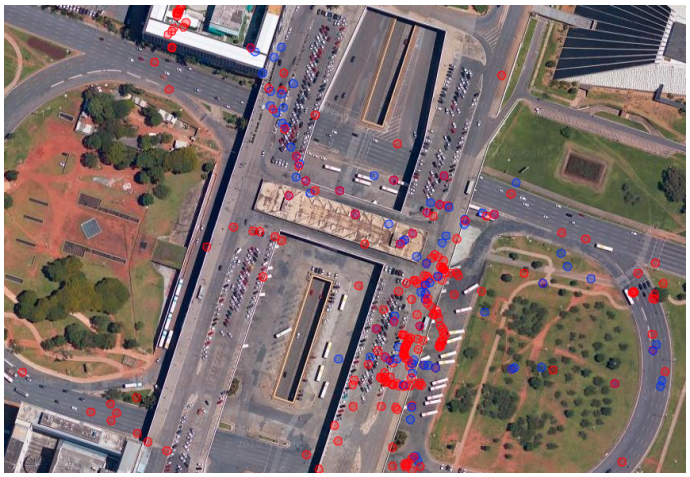

Figura 3. Relação entre dados da base externa e dados do aplicativo.

Com o objetivo de comparar dados relacionados entre as duas bases utilizadas na pesquisa, foram filtrados os pontos da base do *Google* que correspondessem ao intervalo

temporal de até 2 segundos para cada ponto coletado pelo aplicativo, considerando, no total, 50 registros com correlação de 0.89 para longitude e 0.81 para latitude.

Em casos registrados em atividade intensa, durante a utilização de veículos, o detalhamento de dados evidencia pontos com alta proximidade absoluta (Figura 4). Para verificar se o padrão se mantém no resto da amostra, foi calculada a distância entre os pontos relacionados de cada base com a fórmula de Haversine [Robusto 1957]:

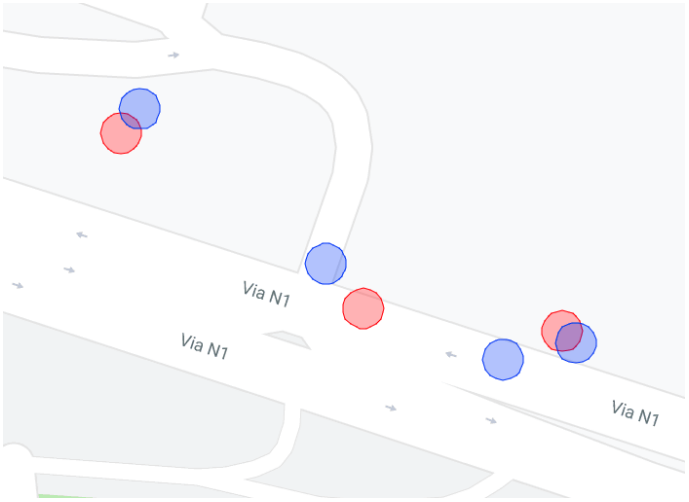

Figura 4. Proximidade entre os registros.

$$d = 2rsin^{-1}\left(\sqrt{sin^2\left(\frac{\Phi_2 - \Phi_1}{2}\right) + cos(\Phi_1)cos(\Phi_2)sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (1)$$

A fórmula de *Haversine* permite o cálculo da distância em uma esfera a partir de pontos de latitude e longitude. Por se aproximar do formato da Terra, a equação tem sido utilizada em diversas situações de cálculo de distância entre pontos geográficos [Winarno et al. 2017].

Com os valores de distância, na Figura 5 observa-se a distribuição da distância entre os pontos relacionados das amostras, que indica, aproximadamente, média de 2.83 Km e desvio-padrão de 2.54 Km. A distribuição compreende 50% de seus valores abaixo de 2.33 Km, evidenciando uma distância considerável entre os pontos mais próximos em temporalidade.

4. Conclusão

Neste artigo foi proposta uma avaliação experimental do componente *Background Geolocation* da biblioteca de plugins do framework para desenvolvimento de aplicativos híbridos *Ionic*. O objetivo foi comparar os dados coletados a partir do componente com a base de dados de geolocalização do Google.

Para coleta de dados, foi desenvolvida uma solução composta por um aplicativo *Android* com o componente, uma API e um banco de dados hospedados em uma plataforma em nuvem, que registrou dados durante 4 dias.

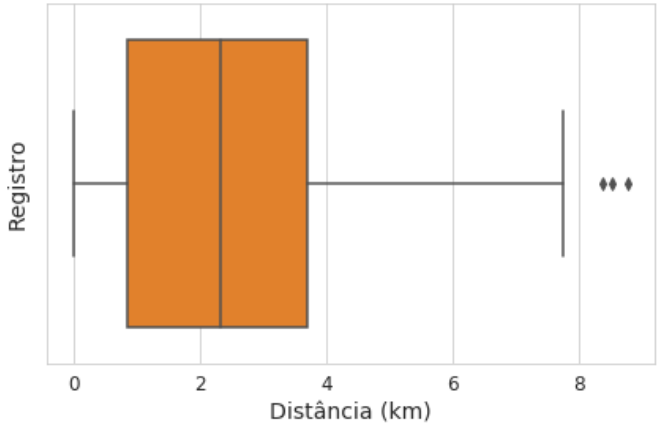

Figura 5. Distribuição da distância entre os pontos.

A filtragem de dados relacionados com as duas bases foi executada a partir de métricas das distribuições, seguindo critérios de exclusão para remoção de valores repetidos, discrepantes e não relacionados na linha temporal. Com dados relacionados, foram apresentadas visualizações em mapa e gráficos para exploração das distribuições.

A qualidade dos registros foi definida a partir da fórmula de *Haversine* para cálculo da distância entre coordenadas geográficas associadas, que mostrou que os pontos relacionados, em maioria, apresentavam distâncias consideravelmente altas.

Para trabalhos futuros, há necessidade de melhora no processo de coleta, que poderia armazenar os registros no dispositivo, e considerar outras características, como velocidade e altitude, em cada registro para a análise, para que com esses dados complementares seja possível observar com maior grau de confiabilidade o fluxo de deslocamento e o comportamento do componente em determinados contextos durante o processo de coleta.

5. Agradecimentos

A temática de pesquisa não teria sido levantada sem o comprometimento dos alunos em experimentar o aplicativo de registro de presenças em eventos, Ponto IFB, trabalho realizado anteriormente a este. Com os dados coletados durante o evento CONECTAIF, foi possível analisar o comportamento do componente de geolocalização e, portanto, propor a sua avaliação.

Por me desafiar a desenvolver o projeto do aplicativo durante a disciplina de Programação para Internet II e por me orientar durante toda esta fase de grande importância acadêmica, registro meu agradecimento ao professor Fábio Oliveira.

Agradeço ao professor Tiago Segato, por contribuir para minha inserção no cenário de desenvolvimento *mobile*, e à professora Ana Régia Neves, por me ensinar Inteligência Artificial, área em que estou me especializando. À professora Jaline Mombach, pelo incentivo à produção de artigos científicos e pelo notável comprometimento com o ensino, à minha família e a todos os professores e colegas que, direta ou indiretamente, contribuíram para a minha formação.

Referências

Amatya, S. and Kurti, A. (2014). Cross-platform mobile development: Challenges and opportunities. In *ICT Innovations 2013*, pages 219–229. Springer International Publishing.

Christie, T. (2019). Django rest framework.

Dabbish, L., Stuart, C., Tsay, J., and Herbsleb, J. (2012). Social coding in github: Transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1277–1286, New York, NY, USA. ACM.

Google (2019). Política de privacidade – privacidade e termos.

IBM (2012). Native, web or hybrid mobile-app development. *White Paper*.

Ionic (2019). Ionic native - ionic documentation.

Kirwan, M., Vandelanotte, C., Fenning, A., and Duncan, M. J. (2013). Diabetes self-management smartphone application for adults with type 1 diabetes: randomized controlled trial. *Journal of medical Internet research*, 15(11):e235.

Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., and Campbell, A. T. (2010). A survey of mobile phone sensing. *IEEE Communications magazine*, 48(9):140–150.

Malavolta, I., Ruberto, S., Soru, T., and Terragni, V. (2015). Hybrid mobile apps in the google play store: An exploratory investigation. In *Proceedings of the second acm international conference on mobile software engineering and systems*, pages 56–59. IEEE Press.

Robusto, C. C. (1957). The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40.

Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., and Filkov, V. (2015). Quality and productivity outcomes relating to continuous integration in github. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, pages 805–816, New York, NY, USA. ACM.

Wang, N., Chen, X., Song, G., Lan, Q., and Parsaei, H. R. (2017). Design of a new mobile-optimized remote laboratory application architecture for m-learning. *IEEE Transactions on Industrial Electronics*, 64(3):2382–2391.

Winarno, E., Hadikurniawati, W., and Rosso, R. N. (2017). Location based service for presence system using haversine method. In *2017 International Conference on Innovative and Creative Information Technology (ICITech)*, pages 1–4.