UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

KELVIN JAMES DE SOUZA MARTINS

RAFAEL LAMMEL MARINHEIRO

LOCALIZAÇÃO DE TROPAS POLICIAIS POR MEIO DO GPS DE *SMARTPHONES*: UMA PROPOSTA UTILIZANDO REACT NATIVE

CURITIBA

2022

KELVIN JAMES DE SOUZA MARTINS
RAFAEL LAMMEL MARINHEIRO

LOCALIZAÇÃO DE TROPAS POLICIAIS POR MEIO DO GPS DE *SMARTPHONES*: UMA PROPOSTA UTILIZANDO REACT NATIVE

Police Troops Location Using Smartphone GPS: A Proposal Using React Native

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação do Curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Mauro Sergio Pereira Fonseca

CURITIBA
2022

KELVIN JAMES DE SOUZA MARTINS
RAFAEL LAMMEL MARINHEIRO

LOCALIZAÇÃO DE TROPAS POLICIAIS POR MEIO DO GPS DE *SMARTPHONES*: UMA PROPOSTA UTILIZANDO REACT NATIVE

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Sistemas de Informação do Curso de Bacharelado em Sistemas de Informação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 05/dezembro/2021

Ana Cristina Barreiras Kochem Vendramin
Doutora
Universidade Tecnológica Federal do Paraná

Hermes Irineu Del Monego
Doutor
Universidade Tecnológica Federal do Paraná

Mauro Sergio Pereira Fonseca
Doutor
Universidade Tecnológica Federal do Paraná

CURITIBA
2022

AGRADECIMENTOS

Aos nossos familiares e amigos, pela compreensão em relação a todos os momentos que precisaram ser dedicados à este trabalho, bem como o apoio moral.

Ao nosso orientador, professor doutor Mauro Sergio Pereira Fonseca, que nos guiou durante toda essa jornada, desde à definição do tema, até a última revisão deste documento.

Em especial, agradecemos à Claudia Roberta Bezerra, Leonardo Lammel Marinheiro, Lia Alflen, Mariangela Lammel Marinheiro e Sérgio Marinheiro, pela disponibilidade de seu tempo e ao compartilhamento de seus *smartphones*, ajudando imensamente com a verificação dos resultados do aplicativo.

RESUMO

A localização de forças policiais é um assunto extremamente importante dentro da esfera de segurança pública, já que o acesso a essas informações possibilitam atendimentos mais ágeis, eficientes e estratégicos. Essa foi uma das principais motivações para o desenvolvimento de um protótipo de sistema de baixo custo que dependa apenas do GPS de um *smartphone* como equipamento de transmissão da geolocalização atual, um servidor para o recebimento desses dados e uma interface *web* que ilustre o posicionamento dos policiais em um mapa, com tecnologias modernas e de código aberto, permitindo a fácil adoção e adaptação por outros desenvolvedores.

Palavras-chave: gps; geolocalização; policiais; desenvolvimento móvel.

ABSTRACT

The localization of police forces is an extremely important issue within the public security sphere, as the access to this information enables more agile, efficient and strategic services. This was one of the main motivations for the development of a low-cost system prototype that relies only on the GPS of a *smartphone* as a transmission device of the current geolocation, a server to receive this data and a web interface that illustrates the positioning of the police on a map, with modern and open source technologies, allowing easy adoption and adaptation by other developers.

Keywords: gps; geolocation; police; mobile app development.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	Interface de Programação de Aplicação, do inglês *Application Programming Interface*
AVL	Localização de Veículo Automática, do inglês *Automatic Vehicle Location*
AWS	Serviços *Web* da Amazon, do inglês Amazon Web Services
CBMSC	Corpo de Bombeiros Militar de Santa Catarina
CSS	Folha de Estilo em Cascata, do inglês *Cascading Style Sheets*
GPS	Sistema de Posicionamento Global, do inglês *Global Positioning System*
HAMPP	Assistente Portátil para Patrulhas Militares e Policiais, do inglês *Handheld Assistant for Military and Police Patrols*
HMR	Troca Rápida de Módulos, do inglês *Hot Module Replacement*
HTML	Linguagem de Marcação de Hipertexto, do inglês *Hypertext Markup Language*
HTTP	Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol*
HTTPS	Protocolo de Transferência de Hipertexto Seguro, do inglês *Hypertext Transfer Protocol Secure*
IPC	Comunicação Interprocesso, do inglês *Inter-Process Communication*
JSON	Notação de Objeto Javascript, do inglês *Javascript Object Notation*
JWT	*Json Web Token*
OHA	Aliança de Telefones Móveis Abertos, do inglês *Open Handset Alliance*
PMSC	Polícia Militar de Santa Catarina
REST	Transferência Representacional de Estado, do inglês *Representational State Transfer*
SDK	Kit de Desenvolvimento de Software, do inglês *Software Development Kit*
SISCOP	Sistema de Controle de Ocorrências Policiais
URI	Identificador Uniforme de Recursos, do inglês *Uniform Resource Identifier*

XML	Linguagem de Marcação Extensiva, do inglês *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A criminalidade no Brasil já é um problema alarmante há décadas, onde apesar de demonstrar índices variantes ao ser analisado regionalmente, em escala federal as taxas são altas e quase sempre em tendência de aumento (BORDIM; LIMA, 2012).

Analisando o anuário brasileiro de segurança pública, o ano de 2020 foi marcado por um crescimento de 4% em comparação com o ano anterior na taxa de mortes violentas intencionais em grupos de 100 mil habitantes. É importante mencionar que esse fator havia chegado no ápice em 2017 com uma taxa de 30,9% e que nos dois anos seguintes, o índice estava sofrendo uma queda consecutiva (Fórum Brasileiro de Segurança Pública, 2021).

Para informações regionais mais recentes do Paraná, existe o relatório estatístico criminal da Secretaria de Segurança Pública do Paraná, onde é apontado um crescimento de 3,32% em crimes contra a pessoa comparado ao ano de 2020. No mesmo relatório, também ficou evidente o grande aumento de 20,36% em Crimes contra o patrimônio, sendo Cascavel, Maringá, Toledo, Curitiba e São Mateus do Sul as cinco cidades mais afetadas, com taxas de crescimento de 31,71%, 30,09%, 26,57%, 25,29% e 23,12% respectivamente (Secretaria de Estado da Segurança Pública do Paraná, 2021).

Segundo levantamento bibliográfico e reflexões sobre os resultados do trabalho realizado por Figueiredo *et al.* (2021), há indícios de que a causa desse problema está ligada a alguns fatores sociais e econômicos. Dentre os fatores presentes no estudo, a desigualdade da população, representada principalmente pelo índice de empregabilidade é um dos principais motivadores para os altos índices de criminalidade. Em contra partida, os autores também abordam pesquisas, teorias e conclusões sobre possíveis soluções para auxiliar na diminuição da criminalidade, mesmo que, como o próprio autor comenta, algumas delas sejam um tanto quanto generalizadas e não muito específicas, são citados itens como a redução de índices de desemprego e melhorias na renda da população e na educação, pensando no longo prazo, mas também conta com algumas referências apontando que o aumento da força policial pode ser eficaz no curto prazo.

Entretanto, para este último item, existem alguns desafios que fazem com que sua implementação não seja tão simples. Para Bordim e Lima (2012) por exemplo, o aumento na compra de equipamentos, viaturas e outros itens policiais, assim como o aumento do número de policiais em si, não é suficiente para resultar em um aumento na efetividade do combate contra a violência e criminalidade. Os autores demonstram em seu trabalho que o uso de informações coletadas sobre os crimes, de forma centralizada e bem analisada, pode ser um método de fato muito efetivo, apresentando um estudo em cima de um projeto realizado pela Segurança Pública do Paraná de analisar os crimes do estado de forma espacial e temporal. Apesar disso, ainda existe uma dificuldade na parte da adoção e aceitação de novos métodos como esse, muitas vezes por falta de uma boa análise em cima dos dados coletados e que acaba levando de volta à prática e gastos em métodos que não demonstram uma efetividade satisfatória.

Além da análise de dados criminais para aplicação em melhorias da força policial, também existem pesquisas e propostas levantadas para utilização de sistemas de informação para um auxílio mais imediato às forças policiais. Um exemplo, é o aplicativo estruturado pela Polícia Militar de Santa Catarina (PMSC) chamado PMSC Mobile, disponível para *tablets* e *smartphones*, o qual possui diversas funções como: consulta de pessoas e veículos, recebimento de ocorrências geradas pelo serviço de emergência, elaboração dos boletins de ocorrência, solicitação de apoio, entre outras funcionalidades (Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC, 2019).

Atualmente no Paraná, uma das maneiras de rastreamento de veículos ocorre utilizando o Sistema de Controle de Ocorrências Policiais (SISCOP), o qual utiliza os dados de localização dos veículos enviados por rastreadores Localização de Veículo Automática, do inglês *Automatic Vehicle Location* (AVL). A Aplicação SISCOP apresenta um mapa e um conjunto de marcadores que representam a localização das ocorrências em andamento e a localização atual das viaturas, assim possibilitando que o operador de rádio faça o despacho de policiais de maneira estratégica (JUNIOR, 2021).

Pode-se dizer então, que apesar de ainda encontrar algumas barreiras, o uso de sistemas da informação, seja para análise de dados ou para um uso mais direto, pode ser considerado uma prática importante para auxílio das forças policiais no combate ao crime. Hoje, graças ao gigantesco número de aparelhos conectados à internet e outras tecnologias, como o computador, *smartphones* e TVs inteligentes por exemplo, o acesso à informação se torna fácil e rápido, quase que instantâneo.

Com base no que foi apresentado, se justifica cada vez mais a utilização de sistemas de informação no trabalho da polícia. Sendo assim, este trabalho visa propor uma solução computacional de código aberto e de baixo custo, utilizando tecnologias modernas para o desenvolvimento móvel e *web*, que pretende auxiliar as forças policiais em identificar facilmente a localização de suas tropas em campo, por meio do Sistema de Posicionamento Global, do inglês *Global Positioning System* (GPS) de um *smartphone* utilizado pelo policial, que envia seus dados periodicamente e automaticamente para um servidor e pode ser consultado por usuários previamente cadastrados, através de uma interface *web*.

Ressalta-se que este trabalho aborda o problema apresentado sob o olhar da área da computação. Não serão avaliados os impactos de tal sistema sobre a força policial, focando apenas na viabilidade computacional do mesmo.

Além disso, o presente trabalho será desenvolvido considerando apenas o sistema operacional Android, devido à inviabilidade de acesso a equipamentos Apple durante o período de execução deste trabalho, necessários para desenvolver aplicações para o sistema iOS.

1.1 Objetivos

O objetivo geral deste trabalho é o desenvolvimento e avaliação de um protótipo de aplicativo para Android que realiza a coleta de geolocalização de forças policiais, para que esses dados possam ser apresentados em uma interface *web*.

Como métodos para alcançar o objetivo geral, se destacam os seguintes objetivos específicos:

- Construção de um protótipo de um aplicativo Android para coleta e envio da localização dos usuários;
- Uma interface *web* para demonstração da posição dos policiais em um mapa;
- Servidor para intermediar a comunicação;
- Disponibilização do protótipo em repositório público.

1.2 Estrutura do Trabalho

O próximo capítulo aborda as principais áreas de conhecimento utilizadas e mencionadas durante o desenvolvimento do trabalho, além de apresentar outros trabalhos e projetos semelhantes.

O terceiro capítulo trata sobre os métodos que serão utilizados para o desenvolvimento do protótipo e da análise dos resultados, ferramentas e tecnologias escolhidas para a construção, arquitetura de *software* proposta e por fim, a descrição da construção do sistema.

No quarto capítulo os resultados das avaliações são apresentados, com foco na análise da precisão da geolocalização e consumo dos recursos de dados móveis e bateria do dispositivo, juntamente com um comparativo entre vários dispositivos diferentes onde os testes foram realizados.

O quinto e último capítulo apresenta a discussão e conclusão da pesquisa em cima dos resultados coletados, bem como sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

De forma a embasar o desenvolvimento deste trabalho, este capítulo apresenta tópicos importantes que são abordados no decorrer do texto. Ao fim, uma seção é reservada para citar trabalhos semelhantes, com o objetivo de visualizar soluções e propostas já existentes para resolver o problema abordado na introdução de formas parecidas.

2.1 Sistemas Distribuídos

Na área de desenvolvimento de *software*, é muito comum que sistemas de informação, especialmente aqueles que se conectam à internet ou alguma outra rede, sejam formados de dois ou mais processos que se comuniquem entre si. Segundo Coulouris *et al.* (2013) a definição de um sistema distribuído é a comunicação e coordenação de ações entre serviços de computadores dentro de uma rede, utilizando de troca de mensagens.

Existem diversas técnicas e conceitos existentes para a construção de um sistema distribuído. A seguir é apresentado o conceito de *web services*, um dos fundamentos utilizados neste trabalho.

2.1.1 *Web Services*

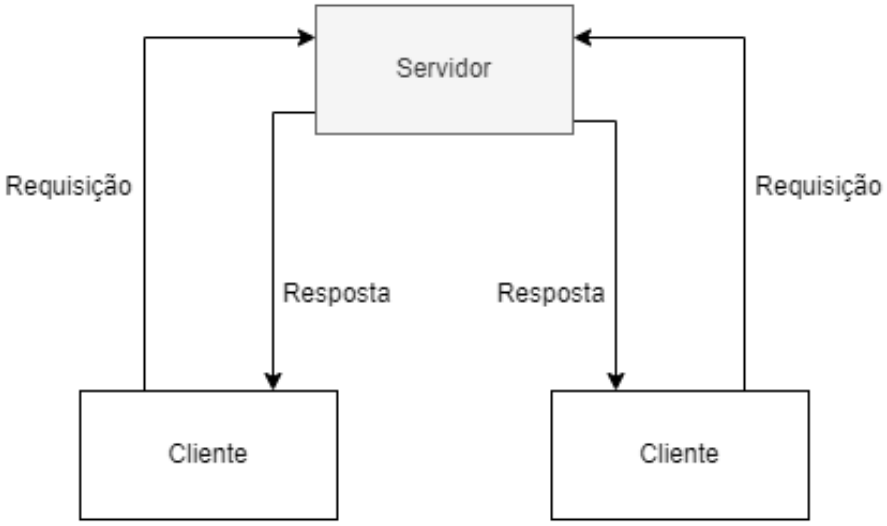
Um dos fatores fundamentais para o funcionamento de um sistema distribuído é que haja comunicação entre os serviços componentes do sistema. Para isso, é preciso decidir uma forma para a comunicação entre esses serviços. Utilizar de *web services*, é uma delas.

Antes de abordar o tema *web services*, é necessário mencionar que são comumente utilizados em uma arquitetura de *software* conhecida como cliente/servidor, que consiste da interação entre dois processos onde cada um recebe um papel. O servidor é o processo que detém os recursos e serviços da aplicação, enquanto o cliente é o processo que consegue acessar e utilizar tais recursos e serviços. Um fato interessante sobre essa arquitetura, é que tem-se aqui uma centralização dos recursos do sistema. Dessa forma, é possível construir diferentes tipos de clientes e todos teriam acesso aos mesmos recursos, desde que utilizem o mesmo servidor (MONTEIRO, 2020).

Vale o destaque de que apesar da explicação da arquitetura e o decorrer do texto mencionarem principalmente a interação entre apenas dois processos, essa arquitetura permite a existência de mais processos tanto clientes quanto servidores, podendo ter vários clientes acessando um ou mais servidores, e inclusive, um servidor acessar recursos e serviços de outro servidor.

A Figura 1 demonstra a utilização da arquitetura cliente/servidor com dois clientes fazendo requisições e recebendo respostas de um servidor.

Figura 1 – Exemplo da arquitetura cliente/servidor



Fonte: Autoria própria.

Para que os processos dessa arquitetura consigam se comunicar, é preciso que eles consigam trocar mensagens para solicitar recursos e serviços disponibilizados pelo servidor.

O conceito de *web services* é uma das possíveis implementações que podem ser utilizadas nesse caso. Coulouris *et al.* (2013) define que implementar *web services* é quando o servidor disponibiliza meios que permitem com que os clientes consigam interagir com algumas ou todas as operações presentes nele. Geralmente, as *web services* são usadas sob o protocolo Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol* (HTTP) de requisição-resposta, que utiliza o Identificador Uniforme de Recursos, do inglês *Uniform Resource Identifier* (URI), para que possam ser utilizados pelos clientes.

Com os meios estabelecidos e conhecidos entre os processos de cliente e servidor, são realizadas então as trocas de mensagens que utilizam um formato como o Linguagem de Marcação Extensiva, do inglês *Extensible Markup Language* (XML) ou o Notação de Objeto Javascript, do inglês *Javascript Object Notation* (JSON).

Um padrão arquitetural comumente utilizado com *web services* é um conhecido como Transferência Representacional de Estado, do inglês *Representational State Transfer* (REST). Segundo Pautasso (2013), o padrão REST propõe que apenas o conteúdo da mensagem é definido, confiando as demais informações ao protocolo HTTP, por meio das URIs para identificação do recurso, e dos métodos de requisição HTTP, como GET, POST, PUT, DELETE, etc. que definem o tipo de operação a ser realizada em cima do recurso. Isso tornou as mensagens mais leves e flexíveis, permitindo inclusive a utilização de outros tipos de formato para a mensagem.

2.2 Dispositivos móveis

Um dispositivo móvel pode ser descrito como um computador ou dispositivo portátil capaz de performar tarefas, receber e/ou enviar dados e que seja de fácil mobilidade para o usuário. De acordo com Lee, Schneider e Schell (2005), essa descrição somente é alcançada quando o aparelho possui quatro específicas características, sendo elas: portabilidade, usabilidade, funcionalidade e conectividade. Fica a critério do usuário decidir quais dessas características são as mais importantes, podendo assim permitir que certas características estejam menos presentes em determinado dispositivo.

Atualmente, existe uma grande variedade de aparelhos que possuem essas características. Os principais exemplos são: *smartphones*, *notebooks*, *smartwatches* e *tablets*. É possível observar que cada um dos aparelhos citados são distintos no nível de implementação das características, tendo como um exemplo o *smartphone*, que possui um grau de portabilidade maior que um *notebook*, porém o *notebook*, dependendo da situação, pode oferecer maior funcionalidade.

Smartphones, em específico, possuem grande presença na sociedade brasileira. Segundo dados de Newzoo (2020), 55,4% da população total do Brasil são usuários de aparelhos *smartphones*. Na mesma pesquisa é possível observar que o Brasil é o quinto país com o maior número de usuários de *smartphone* dentro do ranqueamento desenvolvido pela pesquisa.

Um objeto essencial para os *smartphones*, assim como é para os computadores pessoais, é o sistema operacional, responsável por agir como um intermediário entre o *hardware* e as aplicações, além do gerenciamento dos recursos e o estabelecimento de uma interface ao usuário. Os sistemas operacionais para *smartphones* mais presentes conforme Alves (2018), são os sistemas Android com 82,8% de fatia do mercado e iOS com 13,9%.

2.2.1 Android

Quando houve um grande aumento na utilização de aparelhos móveis, ficaram evidentes as inúmeras oportunidades que poderiam surgir neste novo mercado emergente. Com esse pensamento e motivação, a Google e outras diversas empresas do ramo de tecnologia e dispositivos móveis se reuniram para formar um grupo chamado Aliança de Telefones Móveis Abertos, do inglês *Open Handset Alliance* (OHA) (SILVA; PEREIRA, 2009).

Atualmente 84 organizações formam essa equipe, todas conscientes das vantagens e inovações possíveis de serem alcançadas com um ambiente aberto para dispositivos móveis (OPEN HANDSET ALLIANCE, 2022).

A OHA então, desenvolveu a plataforma para dispositivos móveis de código aberto Android. Uma das características mais marcantes do sistema, era a possibilidade de instalar e desenvolver programas que possuíam a mesma liberdade e acesso de recursos das aplicações criadas e licenciadas pelo fabricante do aparelho. Isso possibilitaria uma customização e

liberdade semelhante ao que era encontrada nos computadores pessoais (SILVA; PEREIRA, 2009).

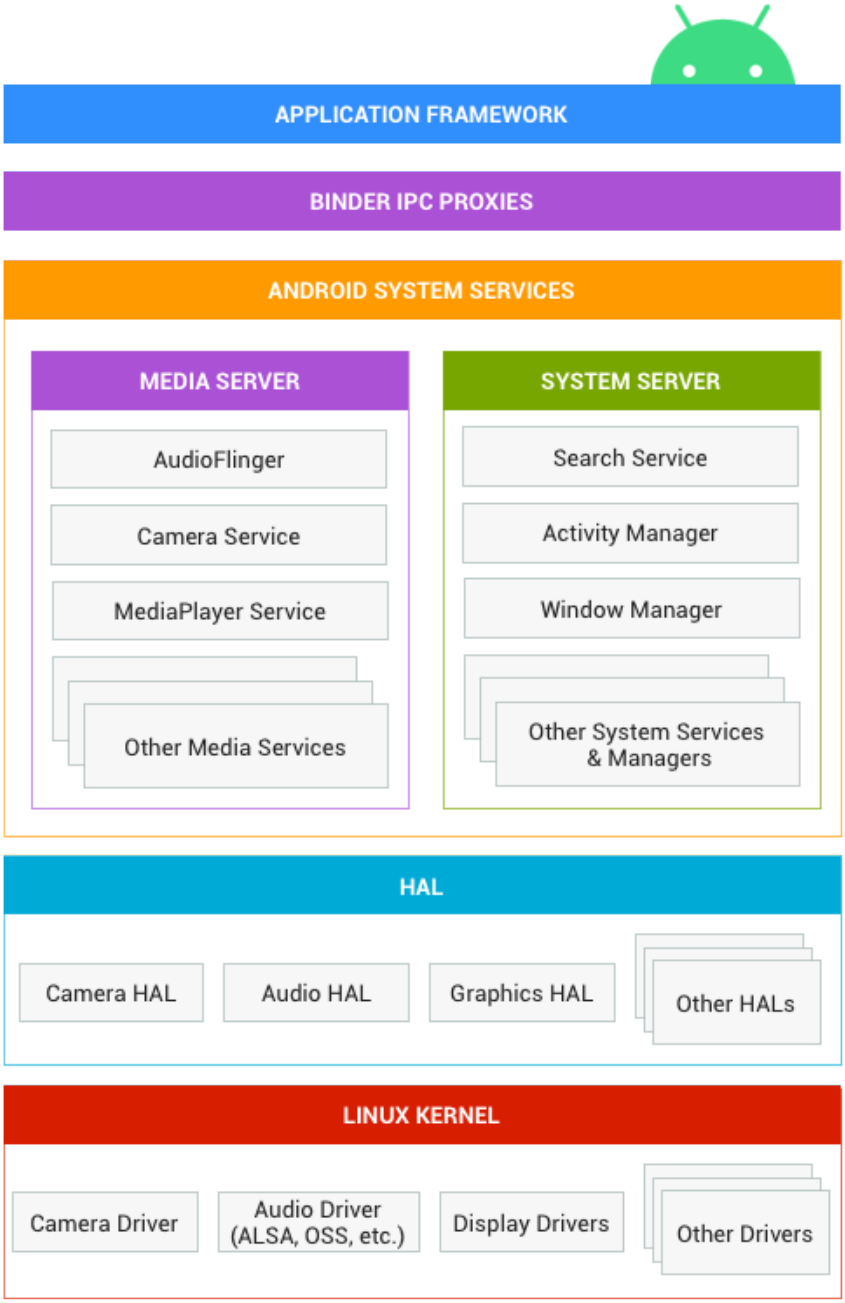
Segundo Burnette (2010), algumas das várias vantagens oferecidas pelo sistema Android são:

- **Plataforma aberta e baseada em Linux:** Por possuir um código aberto, o sistema pode estar sujeito a customizações criadas pela própria comunidade;
- **Grande número de funcionalidades embutidas:** Incluem, por exemplo, serviços de localização e armazenamento;
- **Manutenção automática do ciclo de vida de aplicações:** O sistema foi arquitetado pensando em baixo consumo de bateria e memória disponível limitada. Por conta disso, foi necessário a criação de um gerenciamento da execução das aplicações, assim não exigindo mais uma preocupação do usuário com a atividade de determinados processos. A partir deste atributo, também é oferecido o isolamento de aplicações através de camadas de segurança;
- **Portabilidade com diversos *hardwares*:** Diversos dispositivos de entradas, orientação e resolução são suportados. Os programas escritos em Java também facilitam a portabilidade.

A arquitetura da plataforma Android é apresentada na Figura 2.
As descrições dessas camadas, de acordo com Google (2022a), são:

- ***Framework* de aplicação:** Fornece Interface de Programação de Aplicação, do inglês *Application Programming Interface* (API), com funcionalidades úteis para desenvolvedores de aplicações;
- ***Binder Inter-Process Communication* (Comunicação Interprocesso, do inglês *Inter-Process Communication* (IPC)):** Possibilita que a camada de *framework* de aplicação se comunique com a camada de serviços de sistema do Android. Este processo na camada de *framework* de aplicação é invisível para o desenvolvedor de aplicações;
- **Serviços de sistema Android:** Possui diversos componentes, separados entre servidores de mídia e sistema, fornecem funções como: gerenciador de notificações, serviço de câmera, serviço de busca entre outros serviços. Essas funcionalidades são aproveitadas pela camada de aplicação para se comunicar com o *hardware*;

Figura 2 – Arquitetura Android



Fonte: (GOOGLE, 2022a).

- **Camada de abstração de *hardware*:** Define uma interface padrão para os fornecedores de *hardware* desenvolverem *drivers* de dispositivo;

- ***Kernel* do Linux:** O sistema Android utiliza um *kernel* do Linux com certas funcionalidades adicionais voltadas a dispositivos móveis como gerenciamento de energia e memória. Dessa maneira, desenvolver um *driver* de algum dispositivo para Android acaba sendo semelhante com o desenvolvimento para o *kernel* do Linux padrão.

2.3 React Native

Os desenvolvedores de aplicações móveis contam com algumas opções diferentes na hora de escolher quais ferramentas usarem. Existem as soluções conhecidas como “nativas”, que são as linguagens e Kit de Desenvolvimento de Software, do inglês *Software Development Kit* (SDK)s oficialmente recomendadas pelas empresas que mantém o sistema operacional do dispositivo móvel alvo, mas também existem soluções alternativas com diferentes formatos de desenvolvimento. Dentre as opções se encontra o React Native.

O React Native é uma ferramenta de desenvolvimento de código aberto, criado e mantido pela empresa Meta (anteriormente conhecida como Facebook), que tem como um de seus usos o desenvolvimento de aplicações móveis, utilizando Javascript e outras tecnologias que o tornam muito semelhante ao Linguagem de Marcação de Hipertexto, do inglês *Hypertext Markup Language* (HTML) e Folha de Estilo em Cascata, do inglês *Cascading Style Sheets* (CSS), tornando a construção de aplicações móveis com essa ferramenta parecida com o desenvolvimento *web* (META, 2022a).

Em sua introdução ao React Native, Zammetti (2018) explica que devido ao fato do código em Javascript se comunicar com uma API do sistema operacional para o qual está sendo desenvolvido, é possível acessar os componentes e funcionalidades do sistema móvel de forma abstraída pelo Javascript, e com isso, permite que o aplicativo seja executável em múltiplos sistemas operacionais móveis, precisando de pouca, ou em alguns casos, nenhuma alteração no código fonte.

De tal forma, quando falamos de desenvolvedores *web* que estão indo para o desenvolvimento móvel, o React Native acaba tendo uma vantagem em relação ao aprendizado e uso em comparação ao nativo, como mostra o estudo de Brito *et al.* (2018), citando que não só são tecnologias semelhantes ao desenvolvimento *web*, como também tem uma boa estrutura de documentação.

Em análises mais profundas de comparação de performance no sistema operacional Android, como os de Mahendra e Anggorojati (2020) e Biørn-Hansen *et al.* (2020), o React Native fica abaixo da tecnologia nativa. Porém Brito *et al.* (2018) destaca em seu estudo que o desempenho de aplicativos feitos com React Native tem uma performance semelhante ou até mesmo idêntica com a de outras ferramentas de desenvolvimento móvel, incluindo o desenvolvimento nativo, quando analisado o uso comum dos aplicativos. Além disso, vale a menção de que nas análises do uso da funcionalidade de geolocalização, que é o principal recurso utilizado neste trabalho, o estudo de Biørn-Hansen *et al.* (2020) apresenta resultados próximos ao da implementação nativa para o React Native em relação ao consumo de recursos do aparelho móvel.

2.4 Biblioteca Leaflet

Existem diversas APIs e bibliotecas que auxiliam desenvolvedores de software a construir funcionalidades que utilizam mapas e dados de geolocalização. Um exemplo de código aberto voltado para o desenvolvimento *web* com bastante destaque por conta de sua simplicidade de uso é a biblioteca Javascript Leaflet, que utiliza o projeto de mapeamento colaborativo **OpenStreetMap**, juntamente com HTML e CSS. A biblioteca é capaz de proporcionar uma solução com alta acuracidade e um aproveitamento de funcionalidades já oferecidas pelo HTML para necessidades de um projeto multimídia (EDLER; VETTER, 2019).

O projeto foi criado pelo engenheiro de software Agafonkin (AGAFONKIN, 2022), e se destaca pela facilidade e eficácia para apresentar mapas em telas pequenas de dispositivos móveis e por ter um código enxuto, que acabam sendo características bastante decisivas para a adoção da biblioteca (PETERSON, 2015).

Uma de suas funcionalidades mais básicas, como adicionar um mapa a um elemento HTML e a criação de marcadores, círculos e polígonos dado um conjunto de coordenadas como entrada, é exemplificado no guia de início rápido Leaflet (2022).

2.5 Estado da Arte

Nesta seção são apresentados trabalhos, propostas e protótipos com objetivos semelhantes ao do presente trabalho. Ao final de cada trabalho apresentado, haverá também uma pequena análise crítica de cada um. A última seção é reservada para considerações finais sobre o estado da arte, demonstrando o diferencial do presente trabalho em relação aos demais apresentados.

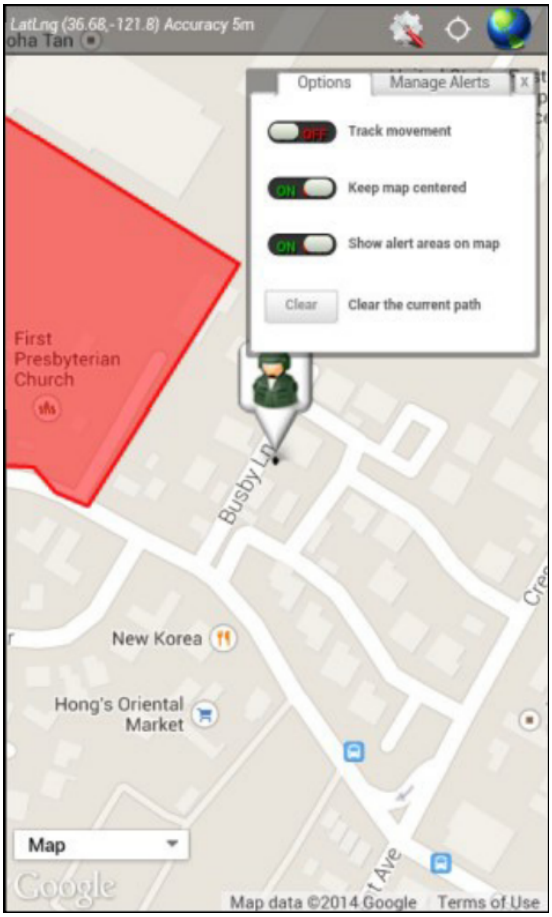
2.5.1 *Handheld Assistant for Military and Police Patrols*

Assistente Portátil para Patrulhas Militares e Policiais, do inglês *Handheld Assistant for Military and Police Patrols* (HAMPP), é um protótipo de sistema de informação desenvolvido por Seipel, Singh e Das (2014). A aplicação se propõe a fornecer às forças policiais informações sobre eventos (emergências, ocorrências e semelhantes) e rotas de patrulha compartilhados entre policiais e a central de comando, de forma a melhorar e monitorar as patrulhas policiais.

O aplicativo foi proposto tanto em uma versão *web* quanto em uma versão para dispositivos móveis, sendo usada pela central de comando e policiais respectivamente, que se comunicam através de um servidor. É através dele que os dispositivos móveis enviam sua localização atual e recebem notificações de eventos próximos, bem como atualizações da rota de patrulha, vindas da central de comando, que recebe e monitora a localização dos policiais.

A Figura 3 demonstra a aplicação sendo executada em um dispositivo móvel.

Figura 3 – HAMPP em execução em um dispositivo móvel



Fonte: (SEIPEL; SINGH; DAS, 2014).

O aplicativo móvel foi construído em cima do *framework* PhoneGap, que permite a criação de aplicativos com HTML, CSS e Javascript. A intenção, segundo os autores, foi de que dessa forma, seria possível utilizar a mesma base de código para múltiplas plataformas. Em adição a isso, o protótipo descrito no artigo utilizava a API do Google Maps em sua primeira versão, porém uma nova versão estaria sendo desenvolvida utilizando o Leaflet, pois o Google Maps acabou tendo limitações na sua versão gratuita, motivando os desenvolvedores a adquirir uma alternativa de código aberto.

A escolha do *framework* PhoneGap não tem mais uma justificativa sólida para ser utilizada em novos projetos ou mantida em projetos antigos, já que existem alternativas mais modernas com as mesmas características de serem amigáveis para desenvolvedores *web* e serem multiplataforma, como o React Native. Além disso, o PhoneGap foi descontinuado em 2020 (ADOBE, 2020), sendo recomendada a migração de quaisquer projetos para outra tecnologia.

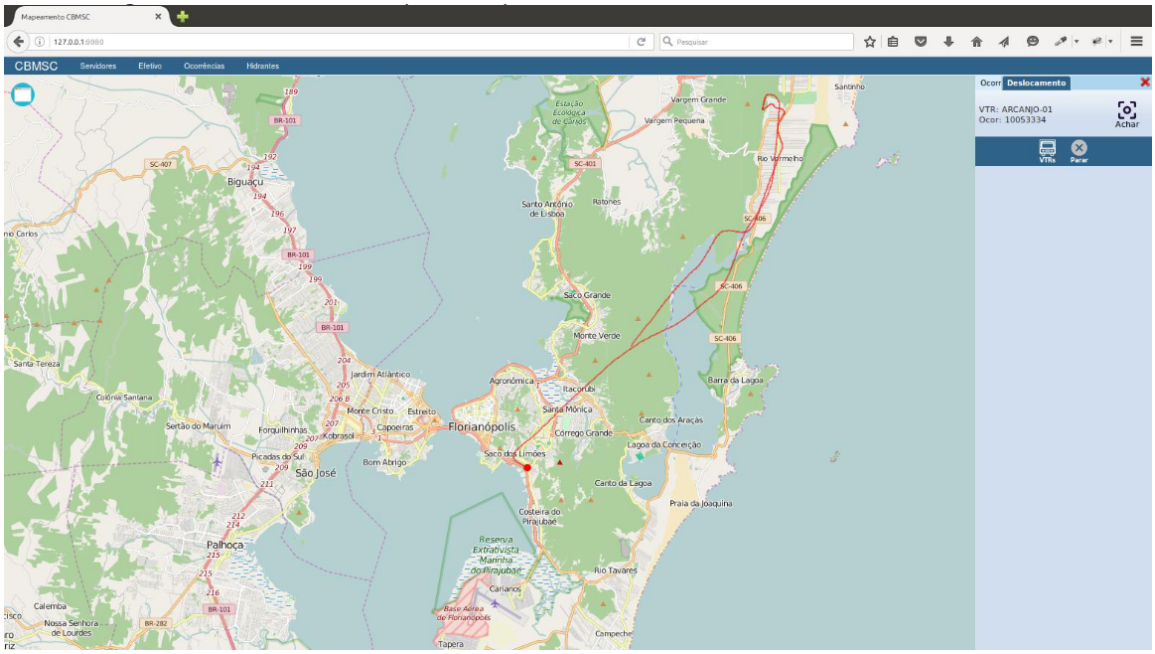
2.5.2 Firecast

O aplicativo Firecast foi desenvolvido pelo Corpo de Bombeiros Militar de Santa Catarina (Corpo de Bombeiros Militar de Santa Catarina (CBMSC)) sob o pretexto de utilizar a tecnologia dos dispositivos móveis de forma a auxiliar o trabalho dos bombeiros.

No artigo escrito por Borges, Frichs e Kretzer (2016), é descrito que o aplicativo funciona nos sistemas operacionais Android e contava primeiramente com as funcionalidades de monitorar o recebimento de ocorrências da viatura, atualizar a situação da viatura na ocorrência e abrir o aplicativo Google Maps com o endereço da ocorrência. Posteriormente, as funcionalidades de gravação de vídeo e transmissão da geolocalização da viatura em tempo real foram implementadas.

Com a adição da funcionalidade de localização em tempo real, os desenvolvedores também construíram um sistema para recepção dos dados de geolocalização e apresentação gráfica desses dados em um mapa, como demonstra a Figura 4.

Figura 4 – Demonstração do sistema auxiliar ao aplicativo Firecast



Fonte: (BORGES; FRICHS; KRETZER, 2016).

Os desenvolvedores escolheram utilizar recursos de transmissão da localização em tempo real, que necessita do aplicativo mantendo conexão contínua com o servidor. A alternativa proposta neste trabalho utiliza envios da localização em períodos de tempo pequenos ao invés de uma conexão constante, na expectativa de atingir um uso aceitável de rede móvel, sem afetar significativamente a eficácia da precisão do GPS.

Os autores também reforçam na conclusão que o aplicativo foi bem avaliado pelos usuários da CBMSC na época da implantação e testes, reforçando a necessidade e desejo de sistemas de informação como esse, para auxiliar a gestão no atendimento à ocorrências.

2.5.3 PMSC Mobile

O sistema para *tablets* e *smartphones* projetado pela Polícia Militar de Santa Catarina (PMSC), PMSC mobile, fornece diversas funcionalidades para suas forças de segurança. De acordo com Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC (2019), mais de 50% das guarnições de Santa Catarina já utilizam o aplicativo e algumas de suas utilidades são:

- Recebimento de notificações sobre ocorrências que chegam no canal de comunicação, juntamente com informações sobre o ocorrido e melhor rota até o local especificado;
- Observar no mapa acontecimentos de emergência e ocorrências já em andamento próximos da localização do policial. Também é possível observar guarnições e câmeras próximas;
- Consultar quadros de avisos, pessoas e veículos;
- Acesso a formulários e registro de documentos como boletim de ocorrências e problemas de ordem pública.

Todas essas funcionalidades com a adição de outras podem ser observadas na Figura 5, representando a tela principal do aplicativo.

Figura 5 – Tela principal do aplicativo PMSC Mobile

Fonte: (Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC, 2019).

O desenvolvimento dessa ideia foi motivado pela quantidade de etapas e informações necessárias no preenchimento de uma ocorrência. Esse problema é evidenciado quando uma grande parte dessas informações já está disponível em outras bases de dados acessíveis. A

partir disso, fica visível que há uma ineficiência que acaba gerando gastos de tempo e dinheiro e como esse processo pode ser agilizado através da utilização do aplicativo. O relato de experiência indica diversos impactos na utilização do aplicativo, sendo eles: redução do tempo de resposta às emergências e erros operacionais, agilidade no envolvimento de forças policiais nos atendimentos a ocorrências e economia de recursos como papel e serviços de gráfica (Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC, 2019).

Por ser um sistema bastante complexo e completo, houve uma necessidade de gastos bastante considerável. Segundo Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC (2019), foram usados R$ 505.200,00 para o desenvolvimento do sistema e um kit essencial para a utilização do aplicativo, constituído de um *tablet*, impressora térmica portátil e suporte veicular que gerou gastos de R$ 3.300,00 por kit.

2.5.4 Considerações Finais Sobre o Estado da Arte

Como visto nesta seção, outros autores já fizeram propostas e desenvolveram sistemas de informação que tratam do problema abordado por este trabalho. No entanto, o presente trabalho tenta se destacar dos demais pela sua intenção de ser um projeto de código aberto e fornecer ao próprio policial acesso à plataforma *web* onde pode verificar sua localização e de seus colegas, não sendo portanto de uso exclusivo da central de comando. O Quadro 1 demonstra um comparativo entre as soluções apresentadas no estado da arte e a solução proposta neste trabalho.

Quadro 1 – Comparativo do estado da arte

Projeto	**Rastreamento do Policial**	**Código Aberto**	**Tecnologia Utilizada Para o Aplicativo**
Este Trabalho	Sim	Sim	React Native
HAMPP	Sim	Não	PhoneGap
Firecast	Apenas viatura	Sim	Java (nativo)
PMSC Mobile	Não informado	Não	Java (nativo)

Fonte: Autoria própria (2022).

Em adição à isto, o presente trabalho apresenta uma solução em React Native para a aplicação móvel, uma tecnologia moderna e bem utilizada, além de ter semelhanças consideráveis com o desenvolvimento *web*, o que torna a colaboração de outros desenvolvedores muito acessível para continuar ou adicionar novas funcionalidades ao projeto proposto por este trabalho.

3 MATERIAIS E MÉTODOS

Este capítulo relata o desenvolvimento do trabalho, abordando os conceitos, tecnologias e metodologias escolhidas para a construção do protótipo. O sistema aqui relatado trata-se de uma base para a solução de um problema real das forças policiais apresentado no capítulo de introdução. O trabalho de pesquisa pode se caracterizar como exploratório, já que houve um desenvolvimento e avaliação de um protótipo que surge a partir do presente trabalho.

A pesquisa foca apenas na visão computacional do problema. Sendo assim, a avaliação do aplicativo será feita sob os aspectos de cumprimento dos requisitos propostos neste capítulo, utilizando a metodologia e ferramentas descritas a seguir.

Nas seguintes seções, são apresentados a modelagem de arquitetura do sistema, os requisitos propostos, as tecnologias escolhidas para o desenvolvimento, as metodologias de desenvolvimento e de avaliação e por fim o relato do desenvolvimento do sistema.

3.1 Arquitetura e Requisitos

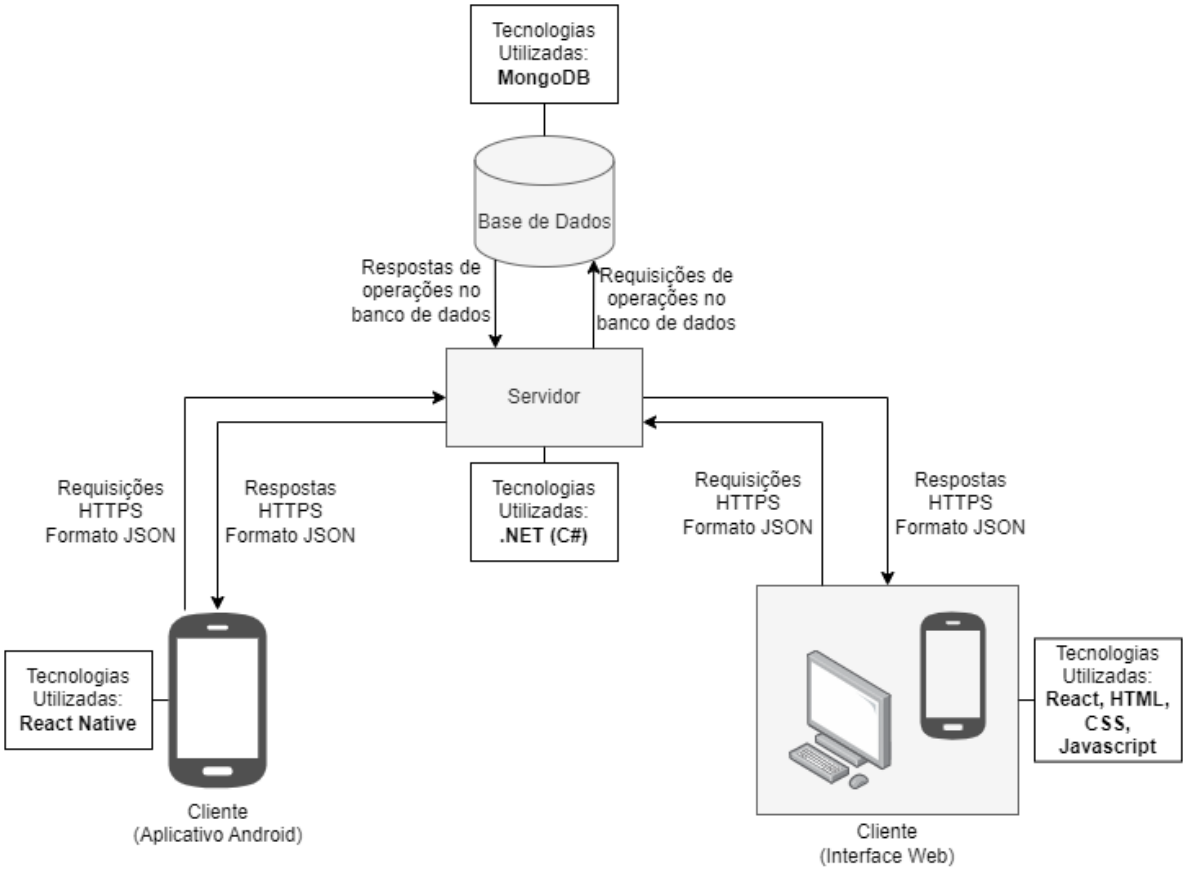
No quesito da arquitetura do sistema, o protótipo utiliza a arquitetura cliente/servidor, contando com um servidor central, que recebe e armazena a posição dos usuários, e dois clientes, o aplicativo que envia as informações da localização, e a interface *web* que consome essas informações e apresenta graficamente em um mapa.

A troca de mensagens entre os processos usa o protocolo HTTP no formato JSON, com o *web service* RESTful. No que diz respeito à segurança dos dados trafegados via rede, após a conclusão do desenvolvimento e implantação na Amazon Web Services, foi utilizado um certificado HTTP, gerado através do sistema da Amazon, **AWS Certificate Manager**, que gera e configura o certificado na aplicação de forma automática nos Serviços *Web* da Amazon, do inglês Amazon Web Services (AWS) (AMAZON, 2022). O Protocolo de Transferência de Hipertexto Seguro, do inglês *Hypertext Transfer Protocol Secure* (HTTPS), de forma resumida, criptografa os dados durante sua transmissão (CLARK, 2013), quesito de extrema importância, visto que os dados da localização do policial são extremamente sensíveis.

A Figura 6 ilustra a arquitetura do protótipo com as setas indicando as trocas de mensagens entre os processos. Nas Figuras 7, 8, 9 e 10 é possível observar exemplos das mensagens JSON trocadas entre os clientes e o servidor.

De forma a atender o propósito da solução apresentada, o Quadro 2 e o Quadro 3 demonstram os requisitos para o desenvolvimento do protótipo, apresentados na forma de requisitos funcionais do sistema.

Figura 6 – Ilustração da arquitetura do protótipo



Fonte: Autoria Própria.

Quadro 2 – Requisitos funcionais propostos para o aplicativo

Requisito Funcional	Descrição
RF01	O protótipo deve permitir a autenticação do usuário por utilização de login e senha
RF02	O protótipo deve coletar a localização do usuário periodicamente

Fonte: Autoria própria (2022).

Quadro 3 – Requisitos funcionais propostos para a interface *web*

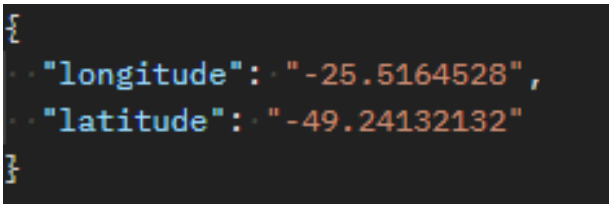
Requisito Funcional	Descrição
RF01	O protótipo deve permitir a autenticação do usuário por utilização de login e senha
RF02	O protótipo deve apresentar a localização em um mapa dos usuários do aplicativo que estiverem conectados
RF03	O protótipo deve permitir o cadastro de novos usuários

Fonte: Autoria própria (2022).

Figura 7 – Exemplo de um objeto JSON contendo todas as localizações cadastradas no sistema

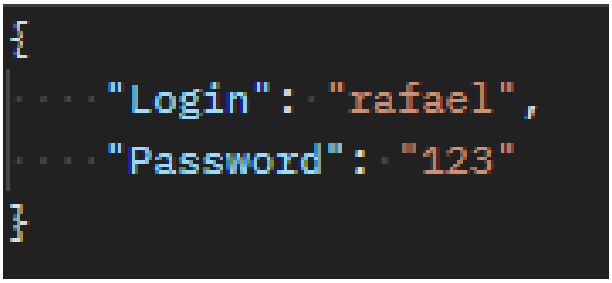
Fonte: Autoria Própria.

Figura 8 – Exemplo de um objeto JSON necessário para atualização de uma localização



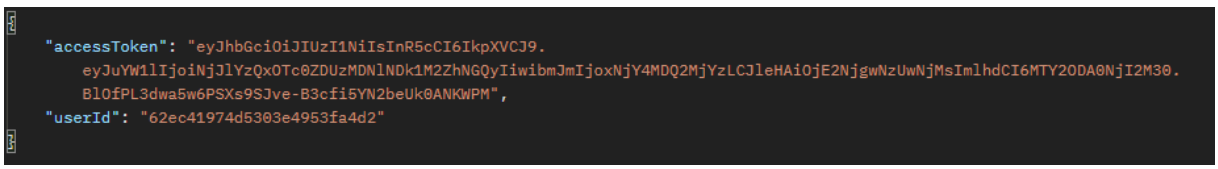
Fonte: Autoria Própria.

Figura 9 – Exemplo de um objeto JSON para dados de autenticação



Fonte: Autoria Própria.

Figura 10 – Exemplo de um objeto JSON recebido após autenticação no sistema



Fonte: Autoria Própria.

3.2 Metodologia de Desenvolvimento e Avaliação

Considerando que o desenvolvimento em questão se trata de um protótipo, construído por um time pequeno de desenvolvedores, opta-se por um processo de desenvolvimento que consiga misturar agilidade para otimização do tempo disponível, juntamente com simplicidade, para não criar tarefas desnecessárias e sobrecarregar o desenvolvimento do projeto.

Para atingir este objetivo, foram analisados os métodos ágeis descritos no livro de Prikladnicki (2014), *Métodos Ágeis para Desenvolvimento de Software*. Dentre as opções apresentadas, o método *Kanban* é o que melhor se enquadrou no contexto dos objetivos citados no parágrafo anterior.

De forma a utilizar o *Kanban* como processo para este desenvolvimento, tomando como base a descrição feita por Vale (2014), todo o trabalho de desenvolvimento é dividido em tarefas, que são realizadas em partes até que seja concluído um incremento do projeto, onde então são criadas novas tarefas até que o projeto seja concluído. Utilizando a ferramenta *online* Trello (2022) que fornece quadros virtuais, é possível obter a visão do andamento das tarefas, que são divididas entre os estados **A Fazer**, **Fazendo** e **Feito**. Além disso, para que não haja sobrecarga de trabalho, há um limitador do número de tarefas em cada estado, com exceção do estado **Feito**.

A Figura 11 demonstra um exemplo de um quadro criado com a ferramenta.

Figura 11 – Exemplo de quadro no Trello

Fonte: Autoria Própria.

Após a conclusão do desenvolvimento, o protótipo passa por uma avaliação que verifica se os requisitos conseguiram ser atendidos dada as tecnologias e arquitetura propostas. Em especial é analisado se a coleta da geolocalização apresentada na interface *web* conseguiu ser eficaz para demonstrar a posição do policial, bem como a utilização dos recursos de bateria e dados móveis do dispositivo. Estas avaliações são feitas através de testes funcionais no sistema e coleta de dados de bateria e dados móveis utilizados pelo aplicativo, fornecidos pelo próprio sistema operacional Android.

3.3 Recursos de Hardware e Software

Esta seção lista todos os recursos necessários para a construção do protótipo. Fazem parte desses recursos, linguagens e bibliotecas de programação e marcação, softwares de banco de dados, plataformas de serviços e dispositivos móveis Android:

- Tecnologia de desenvolvimento móvel:
  - React Native.
- Desenvolvimento do cliente *web*:
  - HTML;
  - CSS;
  - Javascript.
- Biblioteca utilizada para funcionalidades de geolocalização:
  - Leaflet.
- Desenvolvimento do servidor *web*:
  - .NET (C#).
- Software de banco de dados:
  - MongoDB.
- Infraestrutura:
  - Amazon Web Services.
- Dispositivos móveis;
  - Motorola Moto G7 - Android 10;
  - Samsung Galaxy A02 - Android 11;
  - Samsung Galaxy A32 - Android 12;
  - Samsung Galaxy A52s - Android 12;
  - Samsung Galaxy J4 - Android 8;
  - Samsung Galaxy J5 Prime - Android 8.

Além das ferramentas listadas, o sistema de controle de versões Git também é usado, e o código de todos os componentes do sistema está disponível na plataforma **Github** com acesso livre, nos seguintes endereços listados.

- *Web Service*: https://github.com/RafaelLammel/utfpr-policiamovel-backend
- Aplicativo móvel: https://github.com/RafaelLammel/utfpr-policiamovel-app
- Interface *web*: https://github.com/RafaelLammel/utfpr-policiamovel-web

3.4 Desenvolvimento do Protótipo

Com a definição inicial da arquitetura, em conjunto com a metodologia de desenvolvimento, deu-se início à construção do protótipo. De forma a organizar o fluxo do desenvolvimento, foi decidido a divisão da construção do protótipo em três partes.

A primeira foi a construção do servidor, peça fundamental do sistema, visto que os clientes dependem dele para poderem funcionar de acordo com o esperado. Depois, o aplicativo Android foi feito, que gera os dados da localização e os envia para o servidor. Por último, focou-se na construção da aplicação *web*, a interface na qual o usuário recebe um mapa da cidade e a representação gráfica dos dados coletados pelo aplicativo Android.

Nas seguintes subseções são apresentadas descrições detalhadas do desenvolvimento de cada uma das partes.

3.4.1 *Web Service*

Antes de dar início à escrita do código, foi decidido a utilização do conceito Arquitetura Limpa, do inglês *Clean Architecture*, com a intenção de que ao fim do protótipo o mesmo poderia ser usado como base para futuros desenvolvimentos.

Tal conceito descreve a divisão da aplicação em camadas, segregando as responsabilidades da aplicação. Seguindo a definição original de Martin (2017) com algumas modificações, o servidor do protótipo teve quatro camadas implementadas, sendo elas, da mais interna para a mais externa:

- **Domínio**: responsável por conter todos os objetos que representam as entidades da aplicação, ou seja, as representações dos dados que serão comunicados com a base de dados. O domínio dessa aplicação foi modelado com apenas duas entidades, *User* e *Location*, representando o usuário e a localização respectivamente;

- **Aplicação**: responsável pelas regras de negócio da aplicação. No caso deste protótipo, se trata da validação dos dados da autenticação. Além disso, é nessa camada que fica também a coordenação dos dados recebidos pelo banco de dados através da camada de infraestrutura, transformando-os de entidades para objetos mais simples à

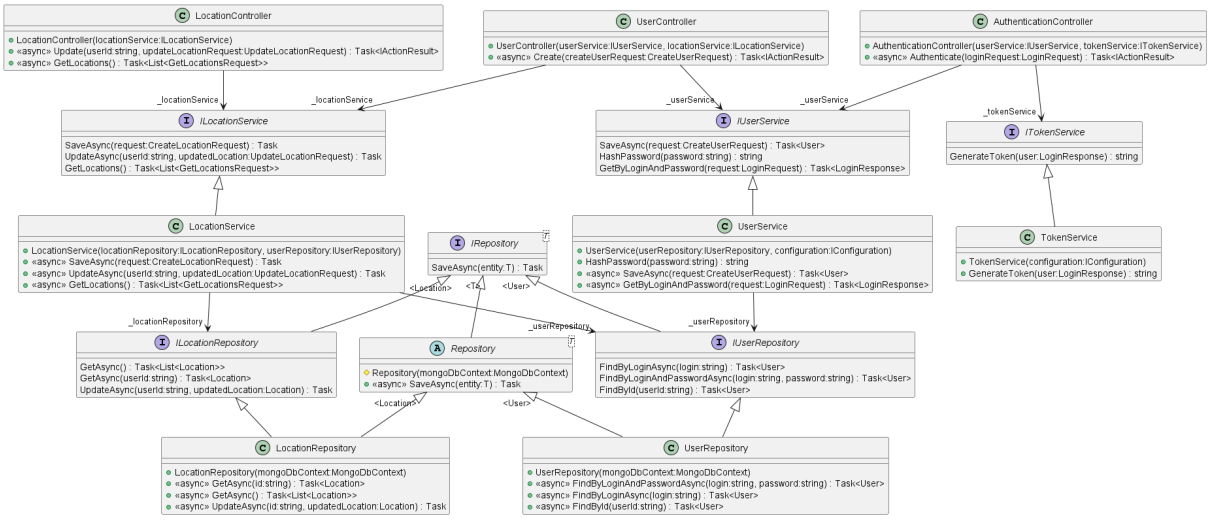
serem enviados em JSON, ou vice-versa;

- **Infraestrutura**: responsável por qualquer comunicação externa feita pela aplicação. No caso deste protótipo, se trata de qualquer operação que envolva a base de dados;

- **API**: responsável pela exposição da interface de comunicação com o servidor, seguindo o padrão REST. Disponibiliza recursos para serem utilizados nas aplicações clientes, por meio de chamadas HTTP nas URIs expostas.

O motivo da escolha e investimento em uma arquitetura robusta como essa se deve à intenção de facilitar ao máximo quaisquer adições ou modificações futuras ao projeto, visto o seu objetivo de servir como base para um sistema completo. A desacoplação de dependências e segregação de responsabilidades proporcionados por essa arquitetura ajudam a atingir este objetivo. Na Figura 12 é possível observar um diagrama de classe da aplicação, sem as classes de entidade que foram separadas para melhor visualização devido ao tamanho da imagem. É possível reparar pelo diagrama a grande vantagem da utilização da Arquitetura Limpa: nenhuma classe depende de outra classe, apenas de interfaces, tornando o acoplamento entre objetos baixo e a aplicação fácil de ser alterada, pois quaisquer alterações realizadas devem continuar seguindo o contrato das interfaces que implementam, garantindo assim que classes que dependam dessas interfaces continuem em funcionamento sem a necessidade de alterá-las.

Figura 12 – Diagrama de classes da aplicação sem as entidades



Fonte: Autoria Própria.

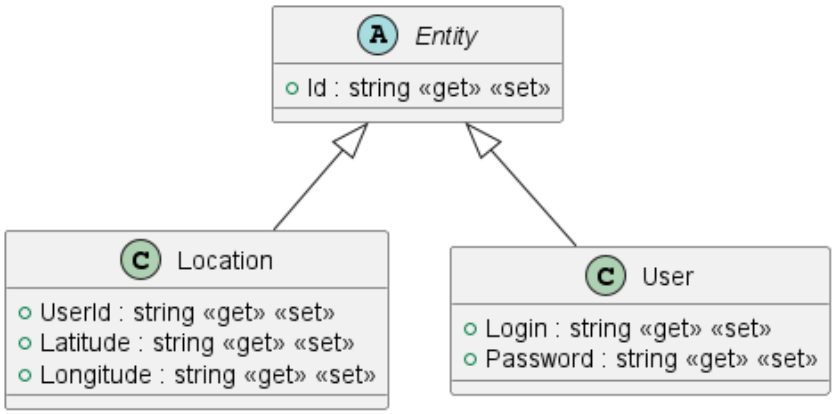
Após a definição da arquitetura, o código foi escrito na linguagem C# na plataforma .NET, pela afinidade dos autores com a tecnologia e por estar em constante evolução de performance e funcionalidades (MICROSOFT, 2022).

Com a arquitetura da aplicação definida, a estrutura do banco de dados foi construída seguindo os requisitos funcionais propostos. O MongoDB, um banco de dados não relacional e orientado a documentos (MONGODB, INC., 2022), foi escolhido por sua simplicidade e velocidade no desenvolvimento.

O domínio da aplicação contém as classes *User* e *Location*, representando os dados dos usuários da aplicação e os dados de geolocalização desses usuários. A entidade de usuário têm apenas *login* e senha por serem o mínimo necessário para a construção e funcionamento do protótipo. Entretanto, em um uso real, mais dados devem ser colocados, como por exemplo nome, número de registro e cargo. Tais adições devem ser simples de serem feitas, devido à arquitetura do sistema.

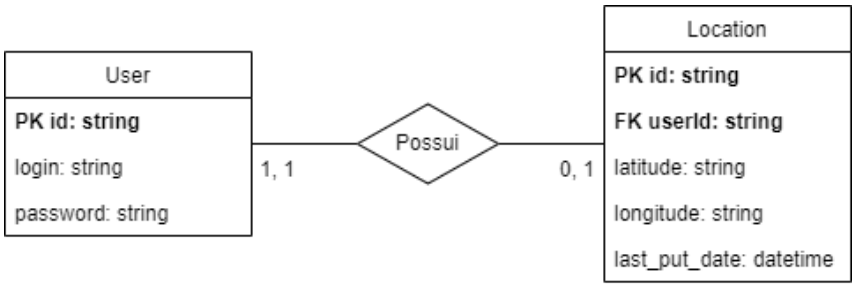
As classes do domínio são mapeadas e salvas em documentos no MongoDB. A Figura 13 apresenta as entidades da aplicação e complementa o diagrama de classes apresentado na Figura 12. A Figura 14 apresenta o diagrama entidade de relacionamento da aplicação, que é enxuto, devido à presença de apenas 2 entidades.

Figura 13 – Diagrama de classes do domínio do servidor



Fonte: Autoria Própria.

Figura 14 – Diagrama entidade relacionamento



Fonte: Autoria Própria.

Seguindo as definições apresentadas, o desenvolvimento do *web service* prosseguiu com a construção de quatro funcionalidades que estão expostas para consumo das aplicações clientes, disponíveis por meio de chamadas HTTP às URIs do *web service*. Dentro do *body*

destas chamadas se encontram os dados necessários para o sucesso da requisição, todos em formato JSON.

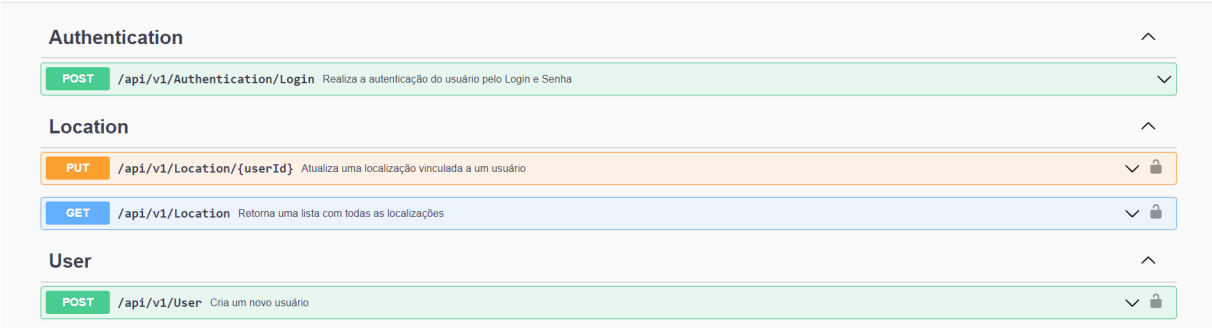
As funcionalidades desenvolvidas para o funcionamento do sistema foram as funcionalidades de atualização da localização e consumo de localização, à serem utilizadas pelo aplicativo móvel e pela interface *web*, respectivamente; bem como a autenticação que seria para o uso dos dois clientes e por fim a funcionalidade de cadastro de usuários. Não há uma funcionalidade de cadastro de localização, pois na hora que o usuário é cadastrado, uma localização nula é cadastrada junto no banco de dados, portanto há apenas a funcionalidade de atualizar a localização.

Visto que a aplicação se trata de um sistema distribuído, composto por um servidor e dois clientes, optou-se pela utilização do *Json Web Token* (JWT) para reforçar a segurança, um padrão de autenticação na qual um servidor emite um *token* criptografado e assinado com as informações necessárias para autenticação do usuário, onde é guardado na memória do cliente e enviado em cada requisição, para informar o servidor que o usuário tem permissão de realizar as ações (JONES; BRADLEY; SAKIMURA, 2015). Esse *token* também pode ter um prazo de validade ajustável, que quando vence, inutiliza o *token*, sendo necessário gerar um novo. Para os testes deste estudo, foi configurado para 8 horas, porém esse valor é facilmente alterado através dos parâmetros utilizados para a construção do objeto de atributos relacionado aos *tokens* emitidos.

Essa estratégia de autenticação foi escolhida principalmente por funcionar muito bem com sistemas distribuídos, visto que não há necessidade de manter um estado entre o cliente e o servidor, ficando tudo sob responsabilidade do *token*.

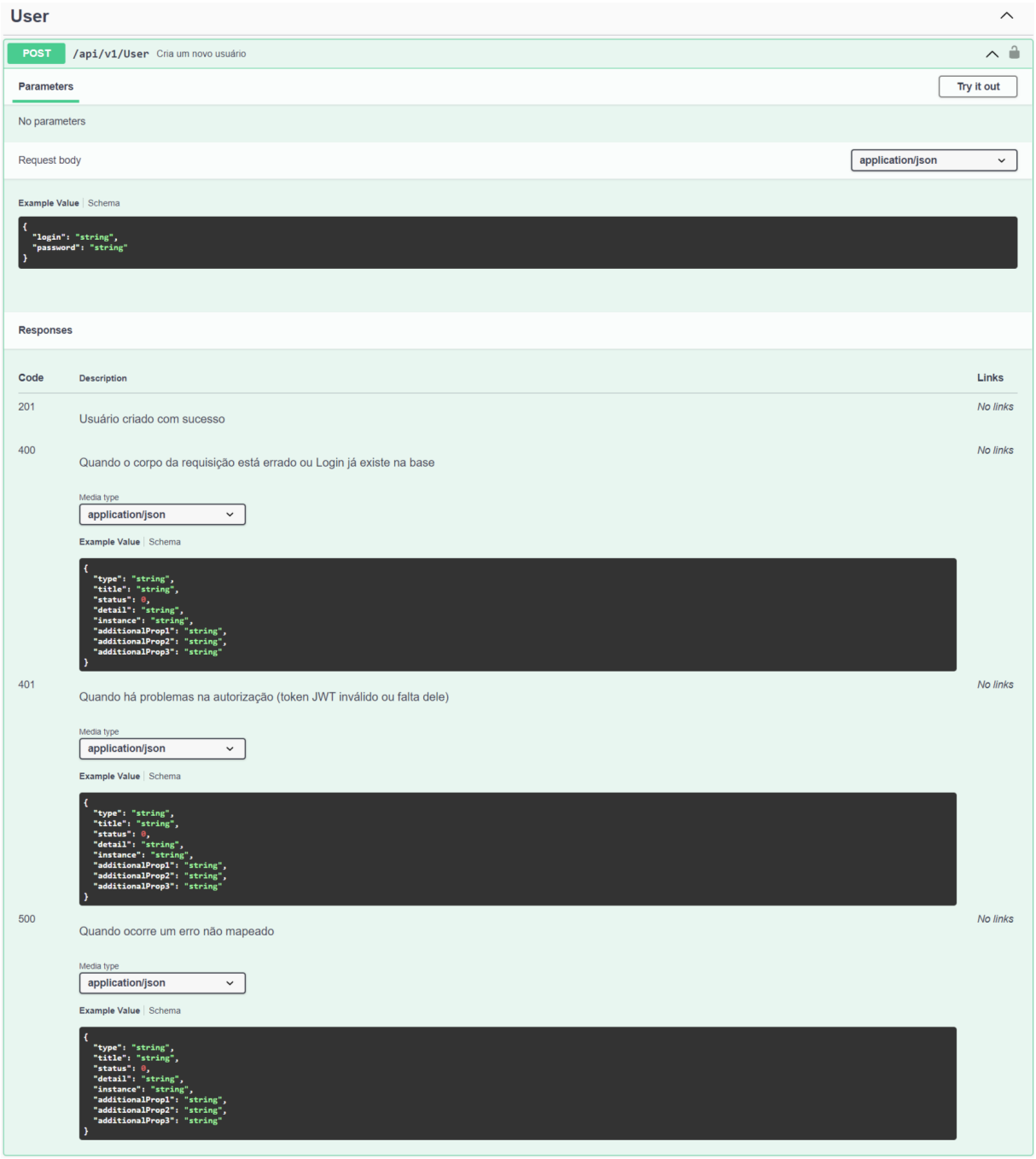
A Figura 15 representa as URIs das quatro funcionalidades desenvolvidas para o *web service* e nas Figuras 16, 17, 18 e 19 se encontra a documentação com mais detalhes dessas URIs.

Figura 15 – URIs do *web service*



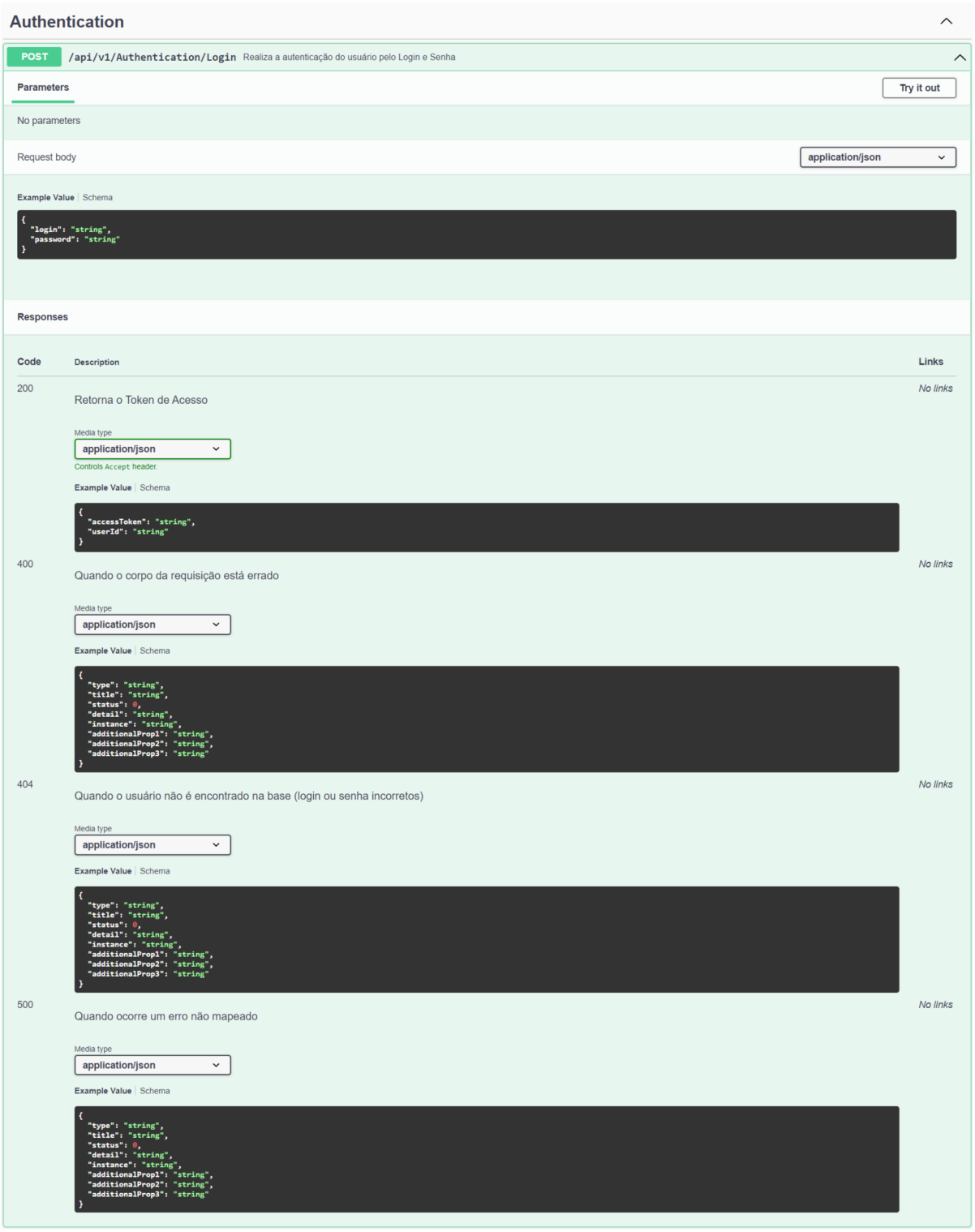
Fonte: Autoria Própria.

Figura 16 – Documentação da URI de cadastro de usuário



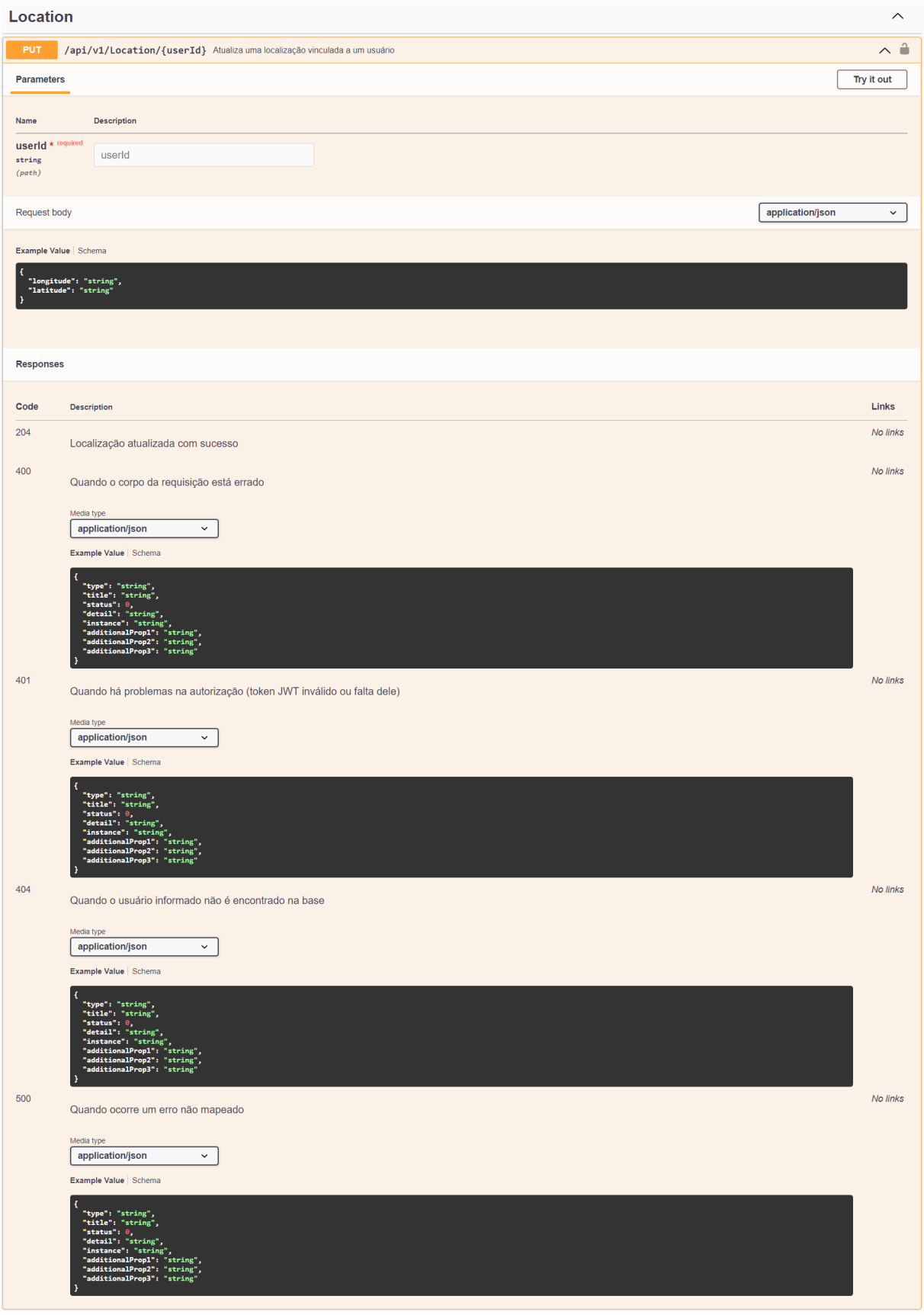
Fonte: Autoria Própria.

Figura 17 – Documentação da URI de autenticação



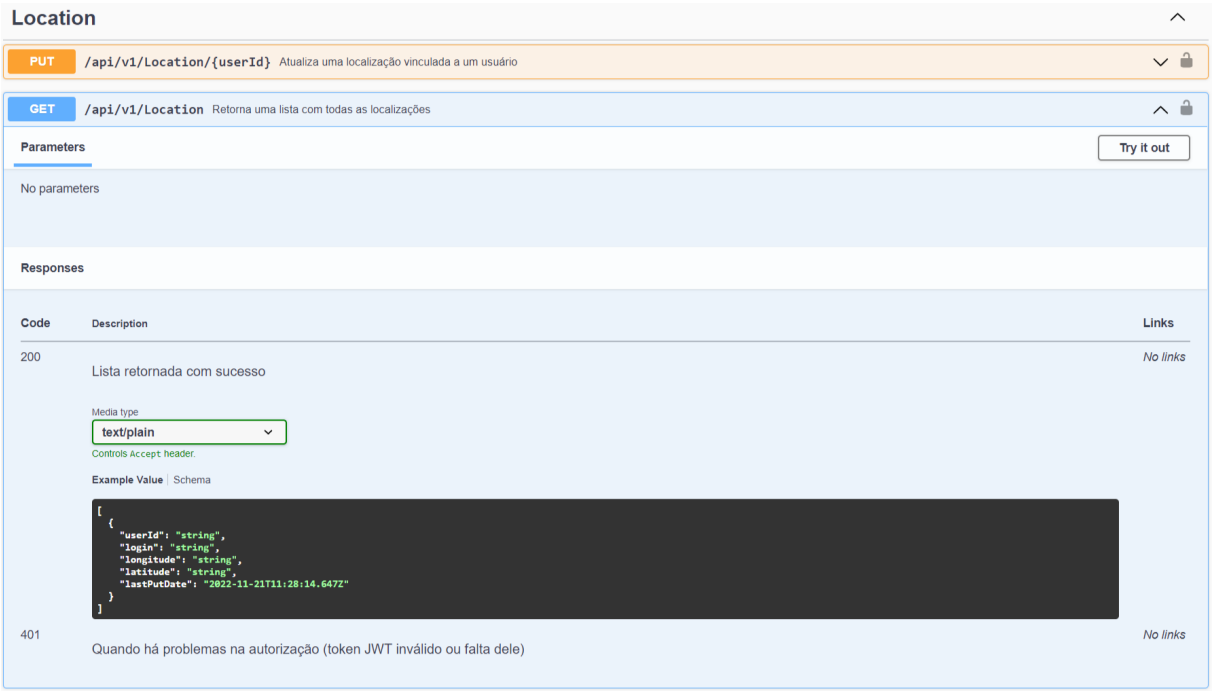
Fonte: Autoria Própria.

Figura 18 – Documentação da URI de registro de localização



Fonte: Autoria Própria.

Figura 19 – Documentação da URI de consumo de localização



Fonte: Autoria Própria.

3.4.2 Aplicação Móvel Android

Com a finalização do *web service*, deu-se início à construção da aplicação móvel que tem como função principal a coleta da geolocalização, juntamente com a autenticação do usuário, para informar ao servidor a identidade junto dos dados de localização.

Para cumprir todos esses objetivos, foram construídas duas telas para a aplicação. A primeira tela gerencia a autenticação do usuário, requisitando login e senha, que quando validados pelo servidor, gera e retorna um JWT para que possa ser armazenado na aplicação e enviado em futuras requisições para provar a identidade do usuário. A Figura 20 apresenta a tela de autenticação e a Figura 21 apresenta o fluxo em formato de diagrama de sequência.

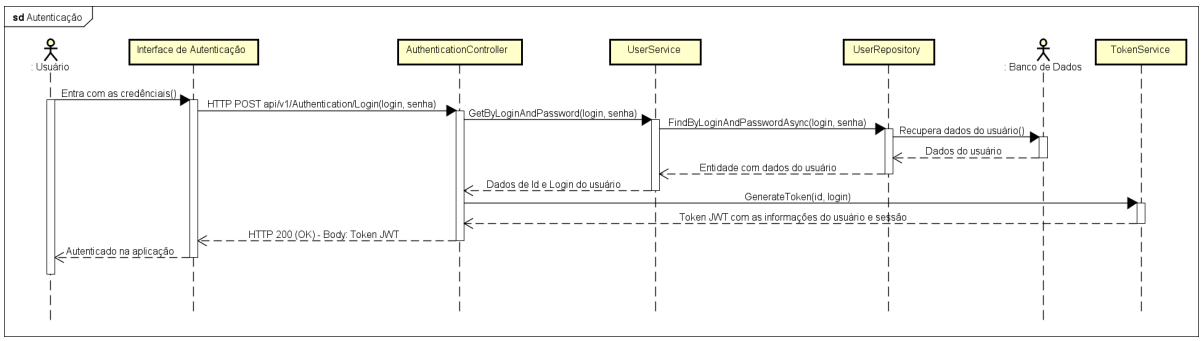
A segunda tela da aplicação pode ser acessada apenas após autenticação do usuário. Nela há apenas uma funcionalidade que o usuário pode realizar, que é a de encerrar sua sessão no aplicativo, apagando o JWT da memória do dispositivo móvel e retornando para a tela de autenticação.

Entretanto, existem outras ações que ocorrem em forma de tarefas que executam em segundo plano, sem necessidade da ação do usuário. Estas tarefas consistem na captura da posição atual do dispositivo móvel e o envio via chamada HTTP PUT para o *web service*, atualizando a localização do usuário logado na base de dados. Esses procedimentos são disparados caso o usuário se movimente mais que 5 metros em um intervalo de no mínimo 2 segundos. Estes foram parâmetros inicialmente definidos para os testes, porém estas configurações são facilmente customizadas através do objeto de opções utilizado na chamada ao método que re-

Figura 20 – Tela de autenticação do aplicativo móvel Android

Fonte: Autoria Própria.

Figura 21 – Diagrama de sequência do fluxo de autenticação da aplicação



Fonte: Autoria Própria.

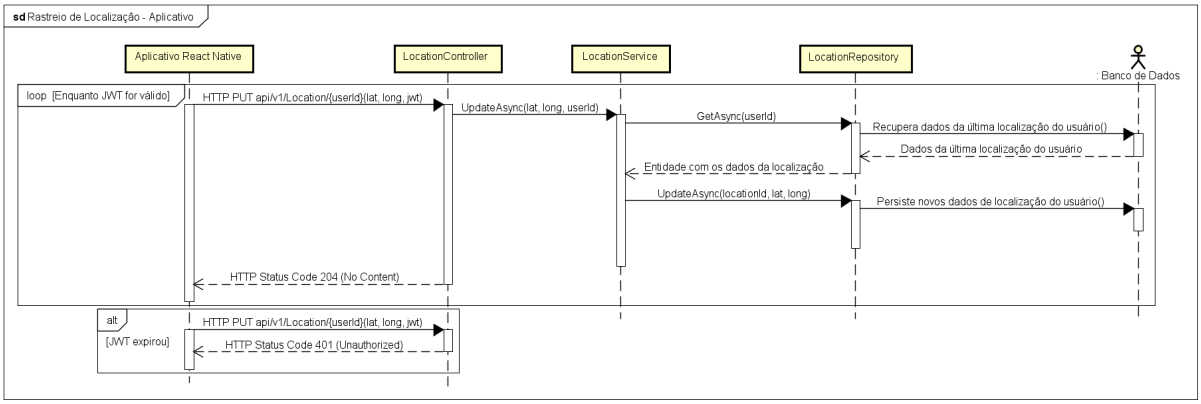
gistra a tarefa responsável pelas atualizações da localização. Essa estratégia foi escolhida para tentar ser o mais econômico possível no uso de dados móveis para a realização das chamadas ao servidor, sem perder muita precisão da geolocalização.

Esta tarefa é executada por meio da criação de um serviço de primeiro plano que mantêm a tarefa viva mesmo que aparelho esteja em modo soneca, com a tela apagada ou até mesmo com o aplicativo fechado. Por conta deste serviço, é possível observar uma notificação de que a tarefa está sendo executada no momento.

Dependendo da versão do Android, o comportamento desta notificação pode ser diferente. Por exemplo, a partir do Android 13, o usuário pode dispensar esta notificação. Porém, dependendo das versões anteriores, essa notificação somente é apagada quando a tarefa de primeiro plano for pausada ou acabada. Outro aspecto importante envolvendo serviços de primeiro plano e que possuem diferenças entre as versões do Android são as permissões de execução. Para aparelhos com Android 9 ou superiores, antes de iniciar a tarefa é necessário requisitar permissão e receber aprovação do usuário para a execução de um serviço de primeiro plano (GOOGLE, 2022b).

Após o tempo configurado na aplicação do servidor, ocorre o vencimento do *token* JWT. Com isso, uma requisição de atualização de uma localização para o *web service* recebe como resposta o código HTTP 401. Neste caso, a tarefa em primeiro plano será interrompida e a localização para de ser enviada, sendo necessário que o usuário faça a autenticação novamente para o retorno da atividade. A Figura 22 apresenta um diagrama de sequência representando este fluxo de envio da localização a partir do aplicativo.

Figura 22 – Diagrama de sequência do fluxo de envio da localização



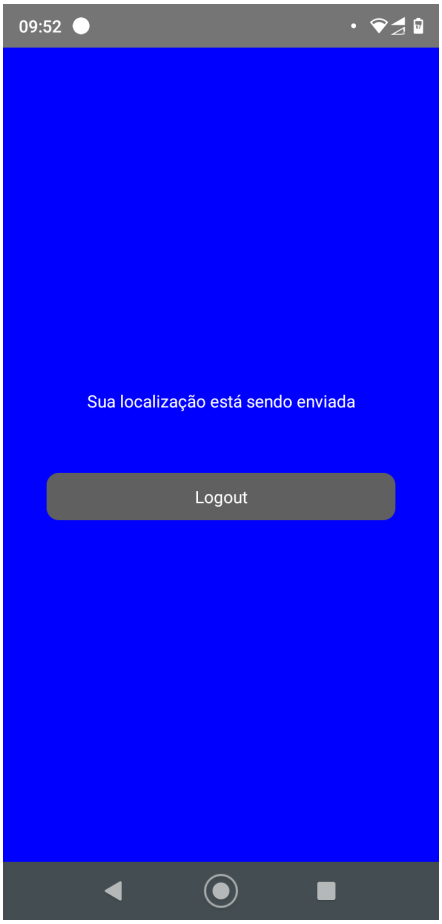
Fonte: Autoria Própria.

O resultado final da segunda tela é demonstrado na Figura 23 e o aviso da tarefa em execução na Figura 24.

Durante alguns testes iniciais após a conclusão da primeira versão do protótipo, foi identificado um comportamento do sistema Android que poderia se tornar um impeditivo para a continuação do trabalho.

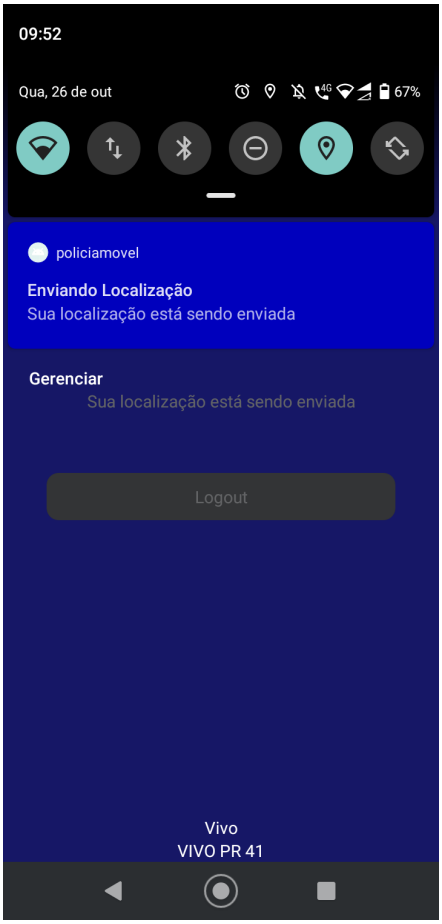
À partir da versão 6 do Android, o sistema operacional se tornou e vem se tornando cada vez mais rígido com a execução de tarefas em segundo plano, em prol da economia de bateria, fazendo com que as aplicações que são executadas em segundo plano sejam colocadas em

Figura 23 – Tela *home* do aplicativo



Fonte: Autoria Própria.

Figura 24 – Tarefa em execução



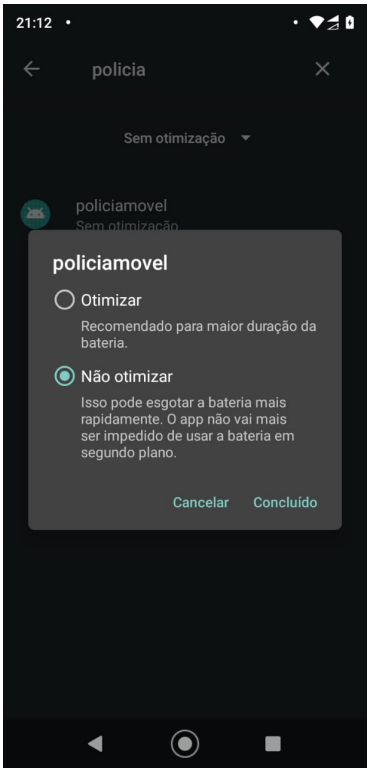
Fonte: Autoria Própria.

um estado dormente, na qual o próprio Android define a periodicidade da execução da tarefa (GOOGLE, 2022c). Este comportamento tornou a frequência de atualização da localização do aplicativo muito baixa e sem controle.

Felizmente, o Android permite que o usuário desative uma opção de restrição ou otimização de bateria, o nome depende do modelo e versão do Android, fazendo com que o aplicativo funcione da forma correta em segundo plano. A Figura 25 demonstra um exemplo dessa opção sendo desativada em um aparelho Android. Entretanto, com as ferramentas atuais disponibilizadas pela equipe do React Native e por sua comunidade, não é possível requisitar e realizar essa alteração de forma automática dentro do próprio aplicativo, mas é possível identificar se o aplicativo está com essa configuração ativada.

Com isso, a solução para este problema foi a criação de um aviso que aparece na tela *home* caso o aplicativo esteja com essa configuração, pedindo para que o usuário a desative. A Figura 26 mostra o aviso aparecendo na tela.

Figura 25 – Exemplo de diálogo de desativação da otimização bateria



Fonte: Autoria Própria.

Figura 26 – Aviso de otimização de bateria na tela *home* do aplicativo

Fonte: Autoria Própria.

3.4.3 Aplicação *Web*

Na aplicação *web*, para auxiliar no desenvolvimento das páginas, rotas e autenticação, foi utilizada uma biblioteca Javascript chamada React, bastante popular atualmente e com muitos pacotes e informações disponíveis fornecidas pela comunidade, sendo inclusive, a biblioteca que inspirou o React Native (META, 2022b).

Outra ferramenta utilizada para facilitar a criação e manutenção do projeto e ambiente de desenvolvimento, é a ferramenta de compilação Vite. Uma das funcionalidades principais que ela oferece, é uma API Troca Rápida de Módulos, do inglês *Hot Module Replacement* (HMR). Esta funcionalidade permite atualizações instantâneas e precisas durante o desenvolvimento da interface de uma aplicação *web* (VITE, 2022).

Dentre os objetivos da aplicação *web*, se encontram o cadastro de novos usuários, e a criação de uma interface para o usuário com a posição dos outros usuários cadastrados e autenticados no aplicativo móvel. Antes da implementação, também foi pensado nos usuários que irão acessar a aplicação através de dispositivos móveis. Por este motivo, a responsividade dos componentes da tela foi um aspecto levado constantemente em consideração durante o desenvolvimento. Com isso em mente, foram criadas um total de três telas, sendo elas a tela de autenticação, tela principal e tela de cadastro.

A tela de autenticação apresenta um formulário com um campo de texto para o *login* e outro para a senha. Ambos são enviados para o servidor, onde caso a autenticação seja válida, este enviará o *token* JWT necessário para que o usuário consiga navegar para tela principal e para a tela de cadastro de novos usuários. Esse *token* é salvo no armazenamento local do navegador e utilizado em todas as futuras requisições, uma vez que apenas a requisição de autenticação no servidor dispensa a necessidade de um *token* válido. Todas as outras funcionalidades retornarão erro caso o *token* não seja usado ou seja inválido. A tela de autenticação é representada na Figura 27. Já o fluxo para a autenticação é idêntico ao apresentado no aplicativo Android, portanto, a Figura 21 vale da mesma forma para a interface *web*.

Caso o usuário consiga terminar o processo de autenticação com sucesso, é feito o redirecionamento imediato para a página principal do sistema, contendo o mapa mundial fornecido pelo **OpenStreetMap**, sendo centralizado na cidade de Curitiba por conveniência deste estudo, mas isso é algo configurável no projeto através de alterações nas constantes do projeto. No mapa, é possível ver os marcadores no local onde se encontram os policiais cadastrados no sistema e dois botões, um para o cadastro de um novo usuário e o outro para *log out*, que apaga o *token* JWT do usuário do armazenamento local e faz o redirecionamento para a tela de autenticação. A localização de todos os usuários é obtida através de uma requisição que a aplicação *web* envia para o servidor em uma taxa de 2 segundos, valor também configurável. Se o *token* do usuário logado no momento não estiver vencido, o servidor retorna uma lista com a localização de todos os policiais, e com estes dados, os marcadores são colocados no mapa. Caso o *token* esteja vencido, é feito o *log out* automático. Quando o usuário clica em

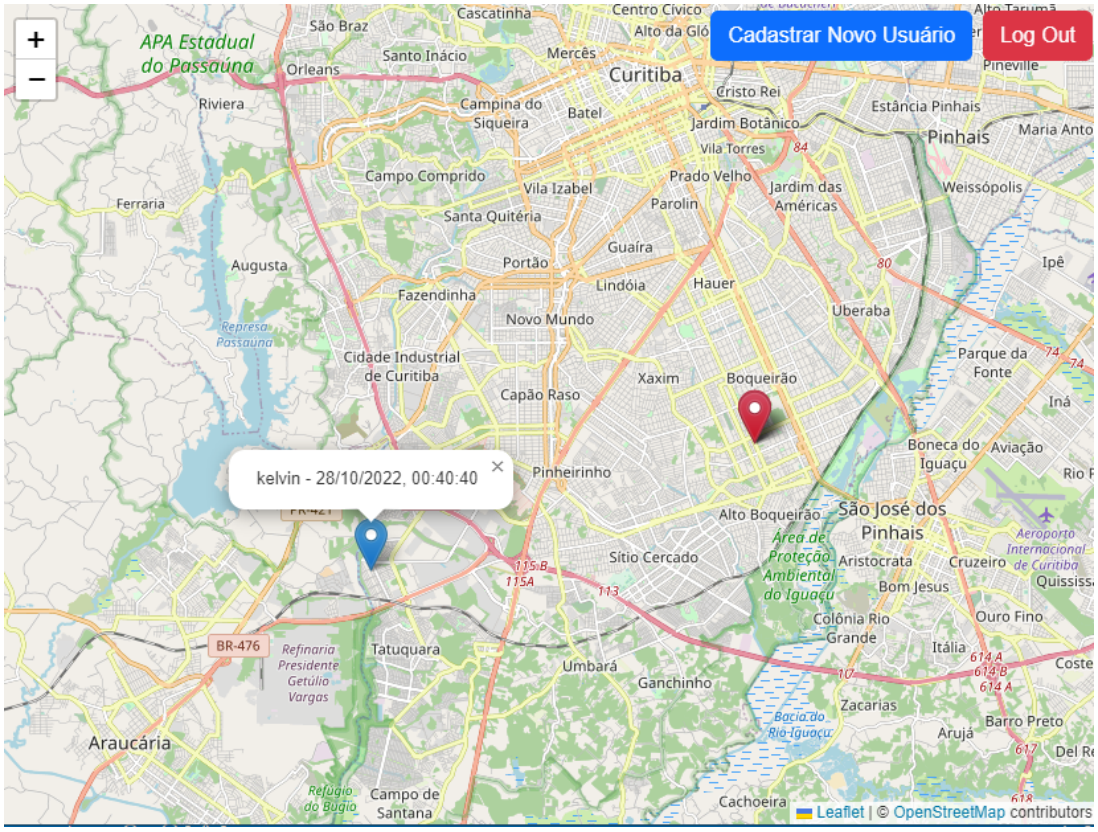
Figura 27 – Tela de login da aplicação *web*

Fonte: Autoria Própria.

algum marcador, uma caixa de diálogo é aberta acima do marcador em questão, informando o *login* do usuário que aquele marcador representa e a data da última localização dele enviada ao servidor. Outro detalhe que vale a pena mencionar, é que se a diferença da data da última localização com a data atual for maior do que 1 minuto, este respectivo marcador é considerado como inativo, sendo apresentado com a cor vermelha no mapa, diferente da cor padrão azul. O tempo para inativação de um marcador no mapa também é configurável alterando as constantes do projeto. A Figura 28 apresenta o resultado da tela principal, enquanto a Figura 29 apresenta o diagrama de sequência que representa este fluxo de execução.

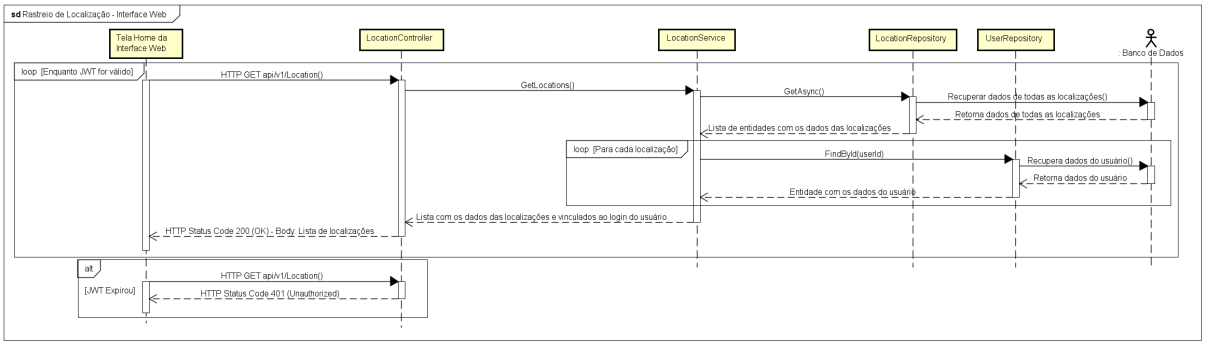
Na tela de cadastrar usuário, um formulário é exibido, solicitando *login* e senha para o novo usuário a ser inserido na base de dados. Após o preenchimento destes campos e a confirmação dos mesmos, os dados são enviados para o *web service*, onde ocorre de fato a criação deste novo usuário. Caso não ocorra nenhum problema no servidor e a interface *web* receba o código HTTP 201 como resposta, então um alerta de sucesso é apresentado, confirmando a criação do novo usuário. A Figura 30 mostra a tela de cadastro e o alerta de sucesso na criação do usuário e logo em seguida, a Figura 31 demonstra um diagrama de sequência que representa o processo de cadastro de usuário.

Figura 28 – Tela principal da aplicação *web*



Fonte: Autoria Própria.

Figura 29 – Diagrama de sequência do fluxo de coleta da localização

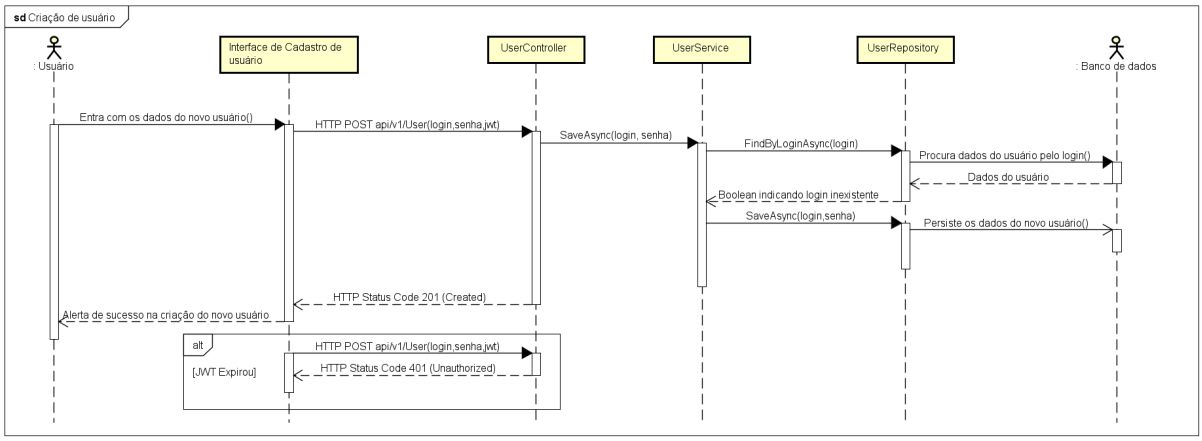


Fonte: Autoria Própria.

Figura 30 – Tela de cadastro de usuário da aplicação *web*

Fonte: Autoria Própria.

Figura 31 – Diagrama de sequência do fluxo de cadastro de usuário



Fonte: Autoria Própria.

4 RESULTADOS

Para a avaliação do sistema, optou-se pela utilização de testes de software funcionais, onde a verificação é feita em cima do cumprimento das funcionalidades propostas e do comportamento do software, sem olhar necessariamente para o código fonte, caracterizando-se como um teste “caixa preta”. Além disso, também é feita a validação da qualidade do serviço, ao verificar a utilização dos dados móveis e bateria consumidos durante a execução da aplicação móvel, ambas abordagens inspiradas pelas descrições feitas por Delamaro (2016). Além disso, devido ao uso da geolocalização do dispositivo, todos os testes foram executados em dispositivos reais de voluntários que puderam contribuir com esta pesquisa.

Tais testes focaram inteiramente na funcionalidade principal de geolocalização e o seu impacto no consumo de recursos do dispositivo, visto que isso é o objetivo primordial deste trabalho. Porém vale lembrar que por conta da forma que os fluxos da aplicação foram projetados e construídos, para testar essa funcionalidade e o consumo de recursos, os outros fluxos de criação de usuário e autenticação inevitavelmente são validados como complementos da avaliação principal.

De forma a se dar inicio à avaliação, primeiramente foram configurados os projetos do servidor e da interface *web* para que estivessem em execução exposta na internet. Para isso, foi escolhida a utilização da infraestrutura da Amazon, a AWS. Utilizando serviços de criação de máquinas virtuais e hospedagem de sites, os dois projetos foram colocados na AWS, rodando nos ambientes mais simples disponíveis, visto que a intenção seria apenas para as avaliações desta pesquisa. Depois, o aplicativo foi instalado nos dispositivos móveis que seriam usados e foram cadastrados usuários na base para cada um dos dispositivos.

Os testes da aplicação foram divididos em dois. O primeiro teste foi feito para realizar a verificação do consumo de recursos em um cenário intenso de constante atualização, enquanto o segundo foi focado em um uso mais próximo ao uso cotidiano.

Vale ressaltar que infelizmente, os trabalhos correlatos que foram citados no estado da arte não forneceram os dados necessários para realizar um quadro comparativo entre os trabalhos nos sentidos que esta pesquisa se propôs em realizar. Sendo assim, o atual capítulo tem o foco de demonstrar o uso do aplicativo em diferentes dispositivos e cenários, para no capítulo seguinte discutir seu uso levando em conta a viabilidade da aplicação ser utilizada por um longo período de tempo, se aproximando do cotidiano do policial, público alvo deste trabalho.

4.1 Avaliação Técnica do Tamanho dos Pacotes

Como parte da proposta deste trabalho foi o desenvolvimento de uma aplicação para dispositivos móveis, fica evidente a importância de medir a quantidade de dados móveis que serão gastos na utilização do aplicativo. Para extrair estes dados, foram analisados os pacotes JSON de requisição e resposta enviados e recebidos pelo dispositivo móvel, ou seja, a requisi-

ção do método PUT que atualiza a localização do usuário e outra do método POST responsável pela autenticação.

Para a captura e envio dos pacotes, foi utilizado o *software* **Postman**, uma ferramenta muito presente no desenvolvimento de APIs. Com ele é possível enviar requisições e receber respostas de pacotes HTTP além de analisar o tamanho destas mensagens.

Através do **Postman**, foi possível observar que os pacotes de requisição PUT, exemplificado na Figura 8, possuem um *body* de 65 *bytes* e *headers* de 512 *bytes*, totalizando 576 *bytes*, dependendo da precisão dos valores de latitude e longitude, este valor pode apresentar pequenas variações. O response, quando o *web service* recebe os valores e atualiza a localização sem nenhum problema, assim retornando o código HTTP 204, apresenta uma resposta onde os *headers* são de 110 *bytes* e o *body* de 0 *byte*, visto que não há conteúdo retornado nessa resposta.

Como a autenticação é um processo que ocorre normalmente uma única vez até a expiração do *token* JWT, ele tem uma contribuição bastante pequena no gasto dos dados móveis. O *body* e os *headers* da requisição apresentam um tamanho de 280 *bytes* e 54 *bytes* respectivamente e a resposta contêm um *body* de 250 *bytes* e *headers* de 170 *bytes*.

A partir destes resultados, é possível traçar algumas estimativas que podem se tornar úteis na especificação do aplicativo. Por exemplo, em um cenário de envio constante de localização a cada dois segundos, de acordo com o tamanho de pacote definido anteriormente, 1,0368 *megabytes* seriam enviados para o *web service* em uma hora, com 8 horas de uso seriam 8,2944 *megabytes*. Já a resposta do *web service* para o aplicativo seriam de 0,198 *megabytes* em 1 hora e em 8 horas totalizariam 1,584 *megabytes* recebidos.

4.2 Avaliação em Cenário Real Intenso Simulado

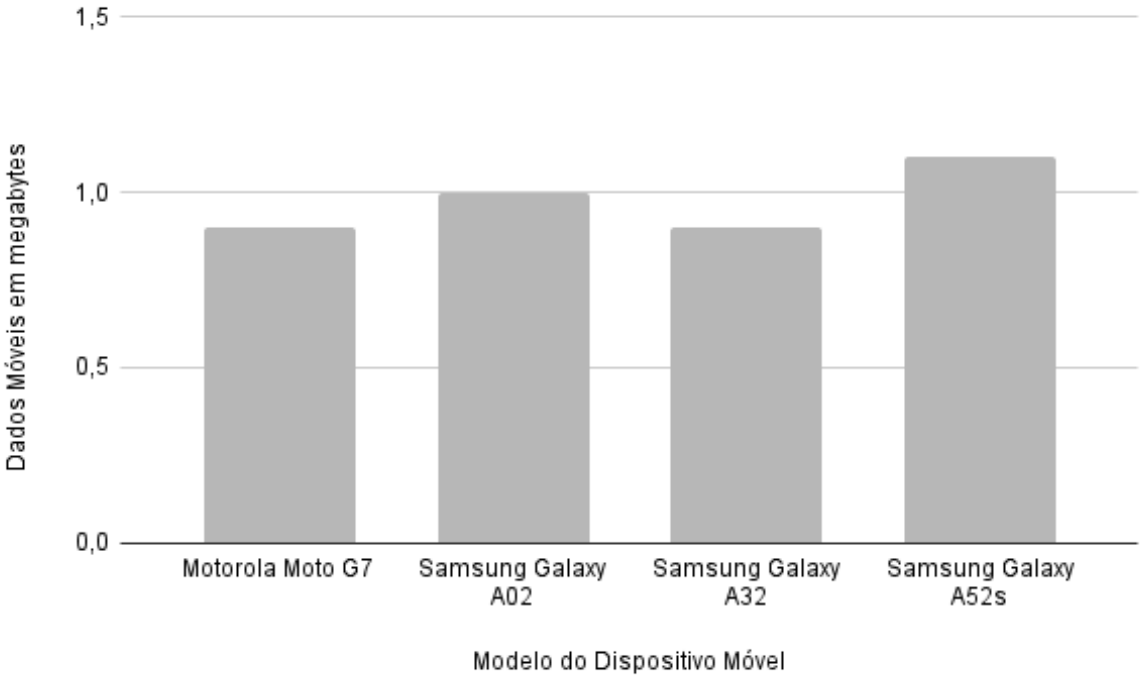
A primeira avaliação em cenário real teve foco no uso de dados móveis e da bateria em um uso intenso simulado, e foi realizada em 4 dispositivos móveis, de modelos e versões diferentes do sistema operacional Android.

Como método de avaliação, foi decidido a criação de um cenário de teste que atualizasse a localização com a maior frequência possível, para simular um uso intenso dos recursos dos dispositivos. Conforme explicado no capítulo anterior, o aplicativo envia dados da localização apenas quando há movimentação de no mínimo 5 metros com intervalo de pelo menos 2 segundos.

Com essas informações e objetivos em mente, o cenário montado foi um onde o usuário estaria em movimento constante, no qual uma rota de 5,4 quilômetros dentro dos limites da cidade de Curitiba foi percorrida no período de aproximadamente 1 hora, com cada dispositivo móvel utilizando dados móveis durante todo o percurso para enviar a localização ao servidor. Os resultados de cada dispositivo móvel podem ser observados nas Figuras 32 e 33, tendo sido

coletados do próprio sistema operacional Android que permite a visualização do consumo de recursos de forma aproximada de cada aplicativo instalado no dispositivo.

Figura 32 – Resultado do consumo de dados móveis em cenário de uso intenso

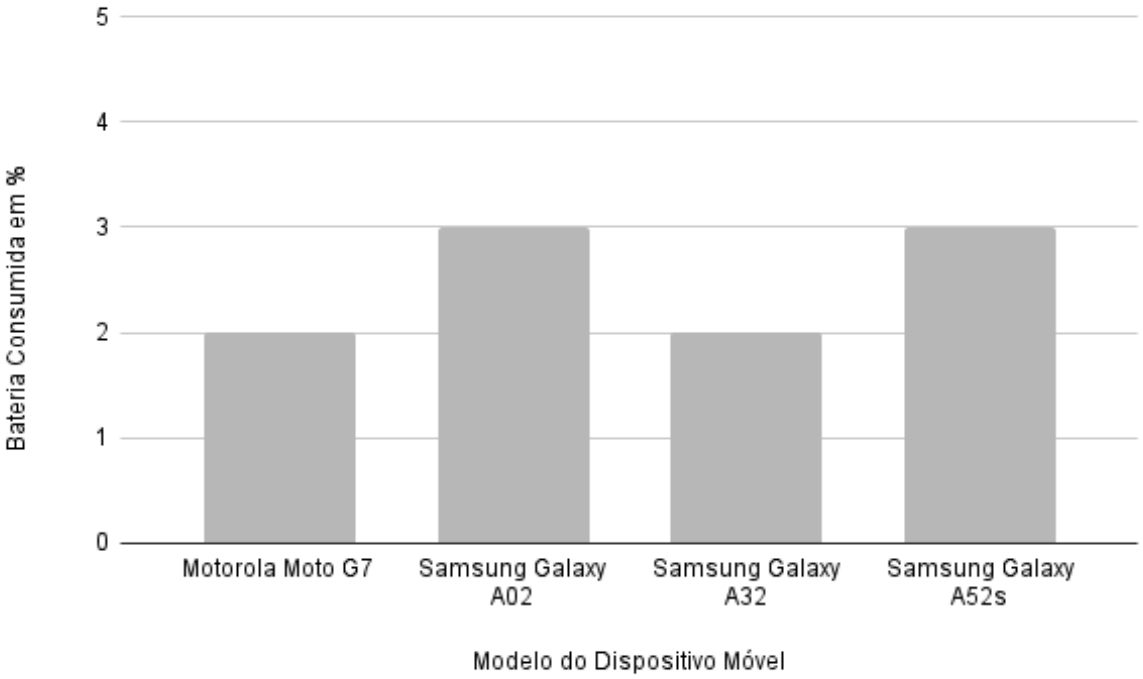


Fonte: Autoria Própria.

Visto que independente do modelo e da versão do Android o aplicativo executa da mesma forma em todos os dispositivos, o resultado se deu conforme o esperado, no qual todos os dispositivos obtiveram um número próximo no consumo de recursos.

As pequenas variações se devem por conta da precisão da localização de cada dispositivo móvel, sendo confirmado durante esse teste que varia um pouco de dispositivo para dispositivo. Isso faz com que, além das variações de tamanho da mensagem JSON que impactam o uso de dados móveis, ocorram também pequenas atualizações da localização, mesmo quando o usuário não tenha se movido necessariamente os 5 metros programados, mas isso não comprometeu a precisão da localização, no máximo fazendo com que o marcador na interface *web* aparecesse um pouco fora de posição, por exemplo o usuário estar na calçada, e o marcador mostrar que o usuário está na rua. Essa distância errada entre o apresentado na interface *web* e a realidade, segundo as informações da biblioteca de localização do React Native (EXPO, 2022) e do que foi observado nos dispositivos testados, é de aproximadamente 3 metros de erro, mas pode variar para mais ou para menos, dependendo da qualidade do GPS do dispositivo utilizado. Apesar disso, em geral, a localização se manteve muito fiel à trajetória que foi percorrida pelos usuários durante o teste.

Figura 33 – Resultado do consumo de bateria em cenário de uso intenso



Fonte: Autoria Própria.

Finalmente, aplicando o cálculo da média entre os resultados e realizando arredondamento, é possível observar que após uso do aplicativo na forma mais intensa, ou seja, com movimento constante, o uso de dados móveis e bateria por hora fica em 1 *megabyte* e 3%, respectivamente.

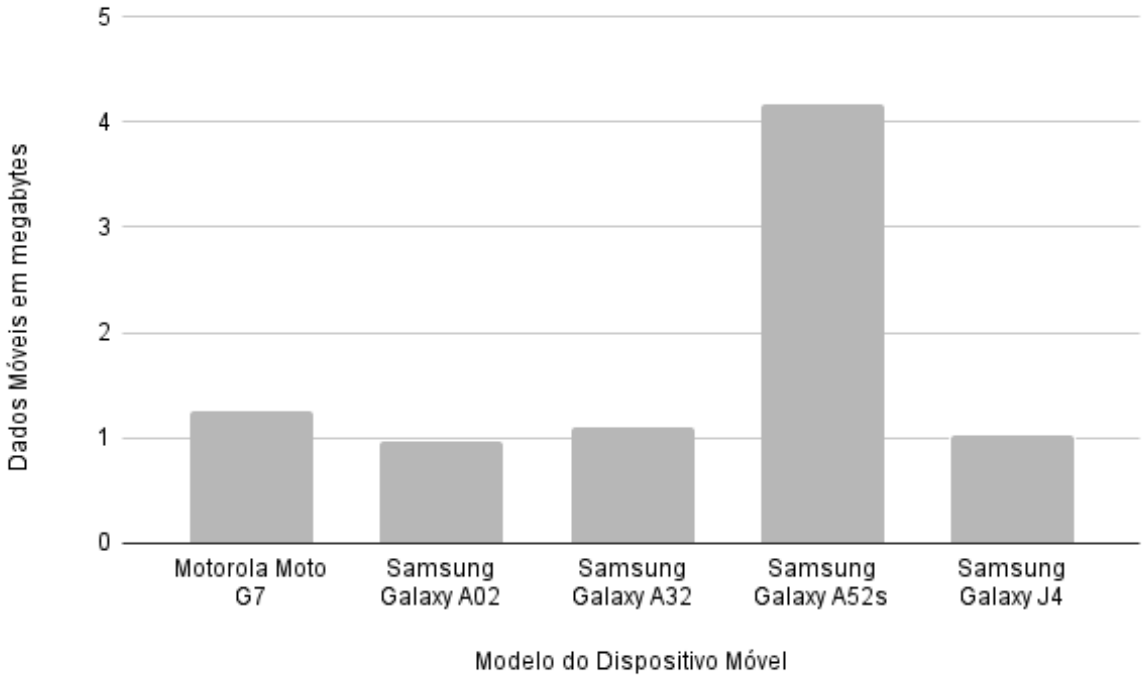
4.3 Avaliação em Cenário Real Normal Simulado

A estratégia do segundo teste foi a utilização do aplicativo durante um período de 8 horas, porém, desta vez, sem uma rota traçada ou quaisquer outras pré-determinações. O aplicativo apenas ficaria executando em segundo plano, enquanto os usuários realizavam suas atividades diárias, que é exatamente a forma que se espera que o aplicativo seja utilizado pelas forças policiais.

Para este segundo teste de cenário real, mais 1 dispositivo conseguiu ser utilizado para coleta dos dados, visto que para este teste não era necessário nenhuma tarefa especial dos usuários, o que permitiu com que mais 1 voluntário pudesse contribuir com esta pesquisa.

Os resultados finais podem ser observados nas Figuras 34 e 35, que assim como o teste anterior, foram coletados via o próprio sistema operacional.

Figura 34 – Resultado do consumo de dados móveis em cenário de uso normal



Fonte: Autoria Própria.

Diferentemente do teste anterior, onde todos os dispositivos foram utilizados sobre as mesmas condições, aqui temos uma variação grande, tanto no uso de dados móveis quanto no consumo de bateria.

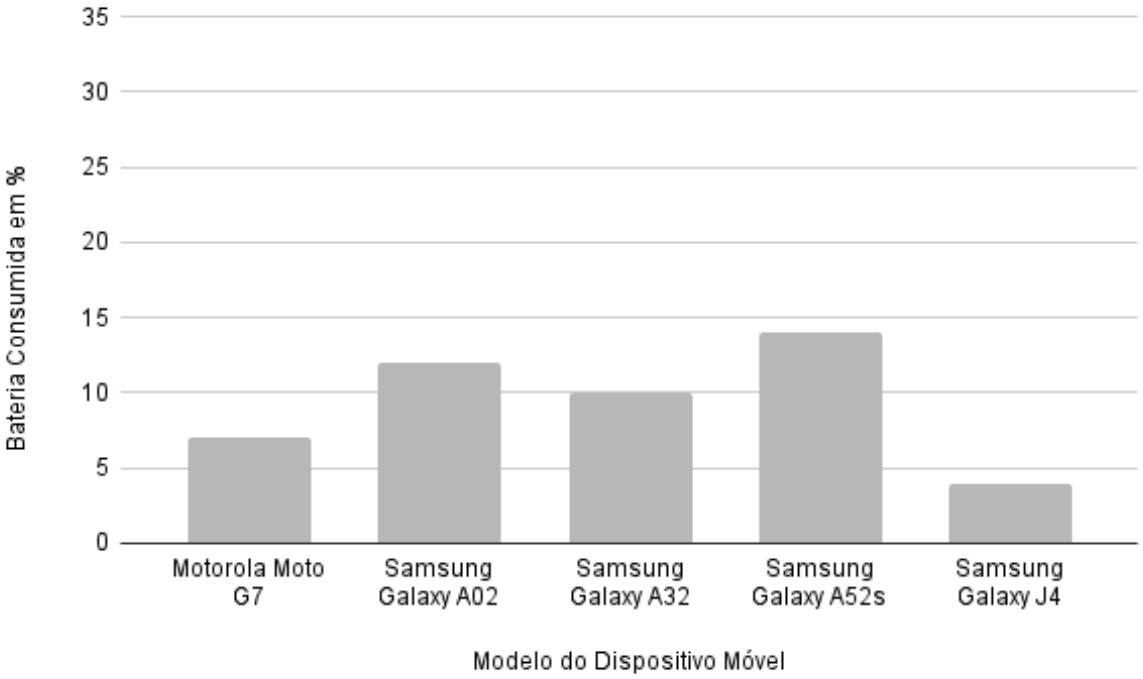
Isso se deve ao fato de que cada pessoa se moveu mais ou menos durante o período de teste, o que impacta diretamente na quantidade de requisições feitas ao servidor, causando a variação de dados móveis.

Destaca-se aqui que o o resultado do Samsung Galaxy A52s ficou com o consumo de recursos excepcionalmente alto, visto que ele ficou aproximadamente 4 das 8 horas em movimento devido a diversas viagens feitas pelo usuário voluntário, acarretando em atualizações frequentes de localização, que, assim como previsto e demonstrado nos resultados deste teste, gerou um maior consumo de recursos.

Conforme é possível olhar pelos gráficos, os dispositivo móveis com maior consumo de dados móveis não necessariamente ficaram com um uso maior de bateria, podendo concluir com base nesse teste, que o consumo desses dois recursos não são diretamente proporcionais, o que tem lógica, visto que o serviço de localização executa a todo momento, independente se teve ou não atualização da localização, pois precisa observar a todo instante se teve ou não movimentação.

Além disso, como quase todos os dispositivos não estavam em movimentação constante como era o caso da primeira avaliação, é possível observar que o consumo de dados

Figura 35 – Resultado do consumo de bateria em cenário de uso normal



Fonte: Autoria Própria.

móveis foi proporcionalmente e significativamente menor, visto que os números estão próximos ao resultado da primeira avaliação, mesmo este teste tendo sido executado com 7 horas a mais.

Como cada usuário submeteu o aplicativo a diferentes cenários, com diferentes quantidades de movimentação, optou-se por não concluir esta avaliação com uma média, mas sim apenas com a análise dos pontos de interesse e devidas explicações de algumas exceções, como o caso do Samsung Galaxy A52s.

5 CONCLUSÃO

Neste último capítulo, são expostos uma visão geral sobre os resultados, dificuldades encontradas e que poderão ser enfrentadas no futuro, funcionalidades e ferramentas que poderão aumentar a qualidade e utilidade dos projetos, tanto de uso quanto para a manutenção. Essas melhorias poderão ser incorporadas à qualquer um dos projetos através dos repositórios **Git** que estão disponíveis nos endereços:

- *Web Service*: https://github.com/RafaelLammel/utfpr-policiamovel-backend
- Aplicativo móvel: https://github.com/RafaelLammel/utfpr-policiamovel-app
- Interface *web*: https://github.com/RafaelLammel/utfpr-policiamovel-web

5.1 Discussão dos Resultados

Com base no que foi visto no capítulo de resultados, o aplicativo construído em React Native demonstrou possuir a capacidade necessária para atingir os requisitos propostos por esta pesquisa. Com uma tecnologia moderna e com muito apoio de sua comunidade, já era esperado que as ferramentas e bibliotecas disponíveis conseguissem cumprir os objetivos do aplicativo. O que se torna o ponto principal dessa pesquisa é a verificação da eficácia e viabilidade em relação ao consumo de recursos, visto que de nada adianta um aplicativo que funciona, mas consome mais recursos do que o disponível.

Com isso em mente, é possível observar nos resultados demonstrados por esta pesquisa que o aplicativo teve bons resultados de consumo de dados móveis e energia, visto que aguentaria um tempo de uso superior a 8 horas, podendo chegar ao tempo de dois períodos de trabalho antes de precisar ser recarregado.

No quesito de dados móveis, a abordagem utilizada na construção do aplicativo teve o gasto em dados móveis de aproximadamente 1 *megabyte* por hora, em um cenário de constante movimentação do usuário. Ainda mais, na segunda simulação que propôs um uso mais moderado, teve um resultado ainda melhor, onde de todos os aparelhos avaliados nenhum passou de 4,2 *megabytes* durante as 8 horas de avaliação. Tal característica permite que as mensagens entre o servidor e o aplicativo transitem na rede de forma rápida, e graças ao tamanho, conseguem ser também muito econômicas no quesito financeiro, visto que a faixa de preços costuma ser cobrada em gigabytes, conforme indica a reportagem **Mesmo com inflação, preço da internet móvel cai no Brasil**, da revista Exame (2022).

Em relação à bateria, o aplicativo demonstrou uma variação dependendo do modelo e versão do Android, porém nada que dispense seu uso. Com a faixa variando entre 4% e 14% no segundo cenário de teste, é plausível que se utilize o aplicativo durante um longo período sem que haja a necessidade de carregamento do aparelho. Ressalta-se que essa afirmação leva em

consideração a utilização do aplicativo e do aparelho em forma regular, não considerando o uso paralelo de outras aplicações com consumo intenso de bateria.

O último ponto a ser discutido nesta seção é a questão da restrição do Android para com as aplicações em execução em segundo plano. Essa característica se demonstrou um potencial problema para a viabilidade do aplicativo. Não no quesito técnico, pois este comportamento pode ser desabilitado, embora não seja tão trivial, conforme explicado anteriormente neste trabalho, mas sim no quesito da implementação em uma corporação de polícia. Como já foi dito, é necessário fazer um processo manual, que difere de aparelho para aparelho, para desativar as restrições que o sistema operacional impõe no serviço que executa em segundo plano. Este processo pode vir a se tornar uma complicação, se levado em conta o número de dispositivo móvel em que o procedimento deve ser feito dentro de uma corporação policial.

Apesar deste contraponto, tendo em vista que o aplicativo obteve bons resultados nos gastos de recursos de dados móveis e bateria, o protótipo demonstrou que o React Native é capaz de ser utilizado para construir uma opção viável para o auxílio das forças policiais a fim de melhor coordenação de tropas em campo, sendo encorajado seu uso para a continuação do desenvolvimento do protótipo em um projeto completo.

5.2 Trabalhos Futuros

Durante o desenvolvimento do projeto, diversos conceitos e funcionalidades foram levantados, porém devido ao escopo proposto, não foram implementados. A intenção com esta pesquisa foi de fornecer a base e confirmar a viabilidade do aplicativo construído em React Native, além de deixar um projeto altamente adaptável e incremental.

Naturalmente, com isso, diversas sugestões ficam para trabalhos futuros, como um botão que ativaria um sinal de reforço na aplicação *web* que notificaria todos os usuários em um certo raio de distância. Esta notificação seria composta pela localização do policial que acionou o alarme, dessa maneira agilizando todo o processo de pedidos de reforços.

Outra sugestão seria uma melhoria na segurança, com a implementação de acesso baseado em funções, onde seria criada uma função para policiais que fazem as patrulhas, uma para os que ficam na estação designando e manejando as tropas e outra para o administrador do sistema que seria responsável pela manutenção dos dados, como a inserção e remoção de um usuário. Também baseado nessas funções, poderiam ser impostas limitações de visibilidade dependendo da função do usuário. Por exemplo, o policial de patrulha só conseguiria ver os policiais próximos dentro de uma área definida, já os que ficam na estação e o administrador teriam uma visão sem restrição, podendo ver a localização de todos os policiais com o aplicativo ligado. Esta restrição também reforça a segurança, pois caso ocorra a perda do dispositivo móvel de um policial de patrulha ou caso alguém mal intencionado encontre o dispositivo já autenticado no sistema, até o vencimento do *token*, esta pessoa teria acesso a localização de todos os outros policiais que estivessem com o aplicativo aberto caso não existisse a restrição.

No quesito de qualidade de código, também foi pensado na utilização de testes unitários e de integração, principalmente para o *web service*.

Todas estas funcionalidades e quaisquer outras que forem julgadas pelas forças policiais como úteis, poderão ser implementadas e incorporadas ao projeto futuramente por qualquer pessoa interessada na continuação de algum dos três projetos, sendo necessário somente a criação de um *pull request* ao repositório no **GitHub** do projeto ou caso prefira, pode-se também criar uma cópia do projeto, com as modificações desejadas. Tal prática é comumente chamada de *fork*.

REFERÊNCIAS

ADOBE. **End of service for Adobe PhoneGap**. 2020. Disponível em: https://helpx.adobe.com/experience-manager/kb/adobe-phonegap-end-of-service.html. Acesso em: maio 2022.

AGAFONKIN, V. 2022. Disponível em: https://agafonkin.com/. Acesso em: abr. 2022.

ALVES, M. A. Análise de fatores relacionados a satisfação de uso dos sistemas operacionais android, ios e windows phone. **Sistemas & Gestão**, v. 13, n. 1, p. 97–106, 2018. Disponível em: https://doi.org/10.20985/1980-5160.2018.v13n1.1269. Acesso em: abr. 2022.

AMAZON. **AWS Certificate Manager**. [*S.l.*], 2022. Disponível em: https://aws.amazon.com/pt/certificate-manager/. Acesso em: nov. 2022.

BIØRN-HANSEN, A. *et al.* An empirical investigation of performance overhead in cross-platform mobile development frameworks. **Empirical Software Engineering**, v. 25, p. 2997–3040, 2020. Disponível em: https://doi.org/10.1007/s10664-020-09827-6. Acesso em: abr. 2022.

BORDIM, M.; LIMA, R. P. Mapeamento do crime e análise criminal: A experiência do estado do paraná. **Geografares**, n. 10, p. 156–175, 2012. Disponível em: https://periodicos.ufes.br/geografares/article/view/1666. Acesso em: mar. 2022.

BORGES, V. S.; FRICHS, M. H. B.; KRETZER, J. Aplicativo Firecast CBMSC. **Ignis: Revista Técnico Científica do Corpo de Bombeiros Militar de Santa Catarina**, v. 1, n. 2, p. 170–179, 2016.

BRITO, H. *et al.* Javascript in mobile applications: React native vs ionic vs nativescript vs native development. *In*: **2018 13th Iberian Conference on Information Systems and Technologies (CISTI)**. [*S.l.*: *s.n.*], 2018. p. 1–6.

BURNETTE, E. **Hello, Android: Introducing Google's Mobile Development Platform**. [*S.l.*]: Pragmatic Bookshelf, 2010. ISBN 9781934356562.

CLARK, T. HTTPS. *In*: **Encyclopedia of Systems Biology**. New York, NY: Springer New York, 2013. p. 926. ISBN 978-1-4419-9863-7. Disponível em: https://doi.org/10.1007/978-1-4419-9863-7_1571. Acesso em: mai. 2022.

COULOURIS, G. *et al.* **Sistemas Distribuídos: Conceitos e Projeto**. Porto Alegre: Bookman, 2013. ISBN 9788582600542.

DELAMARO, M. E. **Introdução ao teste de software.** GEN LTC, 2016. (SBC (Sociedade Brasileira de Computação)). ISBN 9788535283525. Disponível em: https://search.ebscohost.com/login.aspx?direct=true&db=edsmib&AN=edsmib.000017068&lang=pt-br&site=eds-live&scope=site. Acesso em: nov. 2022.

EDLER, D.; VETTER, M. The simplicity of modern audiovisual web cartography: An example with the open-source javascript library leaflet.js. **KN - Journal of Cartography and Geographic Information volume**, v. 69, p. 51–62, 2019. ISSN 2524-4965. Disponível em: https://doi.org/10.1007/s42489-019-00006-2. Acesso em: abr. 2022.

EXAME. **Mesmo com inflação, preço da internet móvel cai no Brasil; confira os planos mais baratos**. 2022. Disponível em: https://exame.com/invest/minhas-financas/mesmo-com-inflacao-preco-da-internet-movel-cai-no-brasil-confira-os-planos-mais-baratos/. Acesso em: nov. 2022.

EXPO. **Location**. [*S.l.*], 2022. Disponível em: https://docs.expo.dev/versions/latest/sdk/location/. Acesso em: out. 2022.

FIGUEIREDO, S. O. *et al.* Fatores determinantes do controle da criminalidade em gestão de políticas de segurança pública. **Revista De Administração Pública**, v. 55, n. 2, p. 438–458, 2021. Disponível em: https://doi.org/10.1590/0034-761220200058. Acesso em: mar. 2022.

Fórum Brasileiro de Segurança Pública. **Anuário Brasileiro de Segurança Pública**. [*S.l.*], 2021.

GOOGLE. **Android Architecture**. [*S.l.*], 2022. Disponível em: https://source.android.com/devices/architecture. Acesso em: abr. 2022.

GOOGLE. **Foreground Services**. [*S.l.*], 2022. Disponível em: https://developer.android.com/guide/components/foreground-services. Acesso em: out. 2022.

GOOGLE. **Optimize for Doze and App Standby**. [*S.l.*], 2022. Disponível em: https://developer.android.com/training/monitoring-device-state/doze-standby. Acesso em: out. 2022.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **Json web token (jwt)**. [*S.l.*], 2015.

JUNIOR, E. N. Localização de frota na Polícia Militar – AVL x Rádio. **Brazilian Journal of Development**, v. 7, n. 9, p. 88954–88970, 2021.

LEAFLET. **Leaflet Quick Start Guide**. [*S.l.*], 2022. Disponível em: https://leafletjs.com/examples/quick-start/. Acesso em: abr. 2022.

LEE, V.; SCHNEIDER, H.; SCHELL, R. **Aplicações Móveis. Arquitetura, Projeto e Desenvolvimento**. [*S.l.*]: Pearson, 2005. ISBN 9788534615402.

MAHENDRA, M.; ANGGOROJATI, B. Evaluating the performance of android based cross-platform app development frameworks. *In*: **2020 the 6th International Conference on Communication and Information Processing**. New York, NY, USA: Association for Computing Machinery, 2020. (ICCIP 2020), p. 32–37. ISBN 9781450388092. Disponível em: https://doi.org/10.1145/3442555.3442561. Acesso em: abr. 2022.

MARTIN, R. C. Clean architecture. *In*: **Clean Architecture: A Craftsman's Guide to Software Structure and Design**. [*S.l.*]: Pearson, 2017. cap. 22. ISBN 978-0-13-449416-6.

META. **Architecture Overview**. [*S.l.*], 2022. Disponível em: https://reactnative.dev/architecture/overview. Acesso em: mai. 2022.

META. **Introdução**. [*S.l.*], 2022. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: out. 2022.

MICROSOFT. **What is .NET?** [*S.l.*], 2022. Disponível em: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet. Acesso em: nov. 2022.

MONGODB, INC. **O Que É O MongoDB?** [*S.l.*], 2022. Disponível em: https://www.mongodb.com/pt-br/what-is-mongodb. Acesso em: maio 2022.

MONTEIRO, E. R. Modelos de arquitetura para sistemas distribuídos. *In*: **Sistemas Distribuídos**. Porto Alegre: SAGAH, 2020. p. 61–82. ISBN 9786556901978.

NEWZOO. **Top countries by smartphone users**. 2020. Disponível em: https://newzoo.com/insights/rankings/top-countries-by-smartphone-penetration-and-users/. Acesso em: abr. 2022.

OPEN HANDSET ALLIANCE. **Overview**. 2022. Disponível em: https://www.openhandsetalliance.com/oha_overview.html. Acesso em: abr. 2022.

PAUTASSO, C. RESTful Web Services: Principles, Patterns, Emerging Technologies. *In*: **Web Services Foundations**. Nova York, NY: Springer, 2013. p. 31–51. ISBN 978-1-4614-7518-7. Disponível em: https://doi.org/10.1007/978-1-4614-7518-7_2. Acesso em: abr. 2022.

PETERSON, M. P. Evaluating Mapping APIs. **Modern Trends in Cartography**, p. 183–197, 2015. Disponível em: https://doi.org/10.1007/978-3-319-07926-4_15. Acesso em: abr. 2022.

PRIKLADNICKI, R. **Métodos ágeis para desenvolvimento de software.** Bookman, 2014. ISBN 9788582602089. Disponível em: https://search.ebscohost.com/login.aspx?direct=true&db=edsmib&AN=edsmib.000005386&lang=pt-br&site=eds-live&scope=site. Acesso em: maio 2022.

Santa Catarina. Polícia Militar. 7ª Seção do Estado Maior Geral da PMSC. PMSC Mobile: tecnologia móvel para gestão de atendimentos policiais. Escola Nacional de Administração Pública (Enap), Junho 2019. Disponível em: http://repositorio.enap.gov.br/handle/1/4118. Acesso em: abr. 2022.

Secretaria de Estado da Segurança Pública do Paraná. **Relatório Estatístico Criminal**. [*S.l.*], 2021.

SEIPEL, P.; SINGH, G.; DAS, A. HAMPP: A handheld assistant for military and police patrols for ha/dr missions. **Procedia Engineering**, v. 78, p. 71–77, 2014. ISSN 1877-7058. Humanitarian Technology: Science, Systems and Global Impact 2014, HumTech2014. Disponível em: https://doi.org/10.1016/j.proeng.2014.07.041. Acesso em: abr. 2022.

SILVA, M. L.; PEREIRA, L. C. O. **Android Para Desenvolvedores**. [*S.l.*]: Brasport, 2009. ISBN 978-8574524993.

TRELLO. 2022. Disponível em: https://trello.com/about. Acesso em: maio 2022.

VALE, A. Kanban. *In*: **Métodos ágeis para desenvolvimento de software**. [*S.l.*]: Bookman, 2014. cap. 8, p. 119–145. ISBN 9788582602089.

VITE. **Getting Started**. [*S.l.*], 2022. Disponível em: https://vitejs.dev/guide/. Acesso em: out. 2022.

ZAMMETTI, F. React native: A gentle introduction. *In*: **Practical React Native**. Berkeley, CA: Apress, 2018. p. 1–32. ISBN 978-1-4842-3939-1. Disponível em: https://doi.org/10.1007/978-1-4842-3939-1_1. Acesso em: abr. 2022.