Implementação Utilizando o Aosp e Geolocalização para Bloquear Camera[1*]

Anthony de Pinho Martins, Jucelbia B. Campos, John Kinderman de F. Barros, Johnson T. Pinto, Thyago Alex C. Sampaio.

Universidade Estadual do Amazonas – UEA
Manaus – Amazonas – Brasil

Projeto Hefesto – HUB – EST

Anthony de Pinho Martins adpm.hef4@uea.edu.br, Jucelbia Brasil Campos jbc.hef4@uea.edu.br, Johnson T. Pinto jtp.tai@uea.edu.br, John Kinderman de Freitas Barros jkdfb.ele20@uea.edu.br, Thyago Alex Cardoso Sampaio tacs.hef4@uea.edu.br

Abstract. *This work aims to implement camera blocking in a given area using the Aosp framework for embedded mobile devices, combining processes, techniques and tools for physical and logical data acquisition from mobile devices. The proposed methodology integrates the use of Design Thinking techniques, Agile Management Methodologies (Scrum), and JAVA programming. For the purpose of validating the proposal, the experiments carried out demonstrate that the proposed methodology is feasible and provides new possibilities for data acquisition in devices running the Android Open-Source Project framework with information protection mechanisms. The techniques included in the methodology are shown to be ineffective, especially those with Google's API mechanism focused on geolocation. The studies show that the methodology is applicable to about 100% of mobile devices with Android Operating System (OS) in use today, preserving the integrity of the data, which is essential for a camera blocking process for the various market sectors.*

Resumo. *Este trabalho tem como objetivo implementar bloqueio da câmera em uma determinada área utilizando o framework Aosp para dispositivos móveis embarcados, combinando processos, técnicas e ferramentas para aquisição física e lógica de dados de dispositivos móveis. A metodologia proposta integra o emprego de técnicas de Design Thinking, Metodologias de gerenciamento ágil(Scrum), e Programação JAVA. Para efeito de validação da proposta, as experiências realizadas demonstram que a metodologia proposta é praticável e fornece novas possibilidades para aquisição de dados de bloqueio e geofence em dispositivos que executam o framework Android Open Source Project com mecanismos de proteção de informações. As técnicas incluídas na metodologia se mostram ineficazes, principalmente os que possuem mecanismo de API 's do Google voltadas para geolocalização. Os estudos demonstram que a metodologia é aplicável a cerca de 100% dos dispositivos móveis com Sistema Operacional (SO) Android em uso atualmente, preservando a integridade dos dados, o que é fundamental para um processo de bloqueio de câmera para os diversos setores mercadológicos.*

1. Introdução

Nosso objetivo foi desenvolver uma aplicação nativa para bloquear a câmera por Geolocalização (GPS) proposto pelo projeto Hefesto composto pelas instituições universidade do estado do amazonas (UEA), Instituto de Pesquisas Eldorado, Motorola, Flextronics e Suframa.

Este artigo traz uma visão geral dessa etapa, conceituando-o, apresentando os processos, funcionalidades, funções e uma das partes principais da elaboração de uma solução para segurança da informação que engloba instituições de todos os setores mercadológicos e governamentais.

A metodologia desenvolvida mostra todas as etapas que o projeto passa, apresentando os detalhamentos em suas distintas fases, com possíveis adaptações de acordo com a necessidade e as peculiaridades do projeto em questão. Também foi abordado a organização da equipe em relação às fases de desenvolvimento. O estudo faz uma introdução no desenvolvimento a respeito da tecnologia AOSP(Android Open Source Project).

2. Fundamentação Teórica

2.1. Design Thinking

O processo de design thinking precisa ser centrado no ser humano, pois ele integra o que é desejável do ponto de vista humano ao que é tecnológico e economicamente viável, e assim identificar quais são os reais problemas dos usuários ou quais problemas serão resolvidos. Com o design thinking é possível aplicar técnicas de design a uma ampla gama de problemas como representado na figura 1.

Figura 1: As 5 etapas do processo aplicada à construção do DT para o App GeoBlockCam.

2.2. Pesquisa Desk

Nesta seção foi desenvolvido uma pesquisa exploratória através da coleta e uso de informações disponíveis. Boa parte desse trabalho foi realizado via internet. Para uma pesquisa válida e confiável, foi fundamental realizar uma análise criteriosa na escolha das fontes a serem utilizadas:

1. Confiabilidade;
2. Disponibilidade;
3. Relevância;
4. Qualidade da informação;
5. Custo.

Nesse sentido, a pesquisa teve o objetivo de criar conhecimentos que auxiliaram na tomada de decisão em relação ao desenvolvimento do MVP(Minimum Viable Product) inicial.

Câmera

Um dos principais objetivos encontrados sobre como construir a aplicação foi descobrir como funciona o AOSP, a câmera e o GPS. Observou-se que no nível da estrutura do aplicativo está o código do aplicativo, que usa a API Camera 2 para interagir com o hardware da câmera. Internamente, esse código chama as interfaces Binder correspondentes para acessar o código nativo que interage com a câmera.

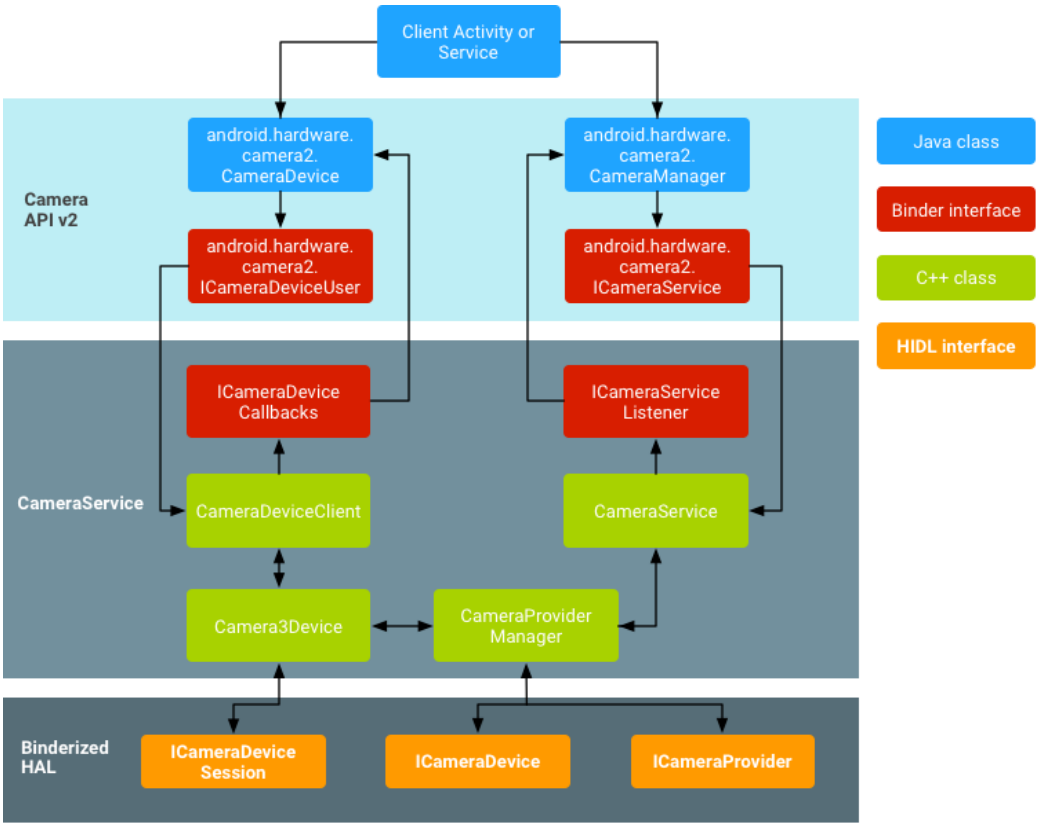


Figura 2: Arquitetura AOSP

GPS

No objetivo de desenvolver a geolocalização observou-se que o funcionamento de aplicações que consumam os dados do GPS nativo do aparelho, para então, desenvolver suas próprias aplicações que necessitem de tais dados.

Trabalhamos com um emulador baseado em software (O AVD – Android Virtual Device), e não com um dispositivo de verdade, a presença do hardware de GPS terá que ser simulada.Uma vez que o AVD tenha suporte a GPS, foi usado o console do ADB (Android Debug Bridge) para enviar comandos de localização, simulando que o usuário do dispositivo se locomoveu.

Geofence

Outro ponto importante a ser mencionado é em relação ao GPS, foi observado a fronteira geográfica desenvolvida para conhecimento da localização atual do usuário com o da proximidade do usuário em relação a locais que podem ser interessantes para ele. Para marcar o local de interesse foi especificada a latitude e longitude. Para ajustar a proximidade do local, foi adicionado um raio. A latitude, a longitude e o raio definem uma fronteira geográfica, criando uma área circular, ou fronteira, ao redor do local de interesse.

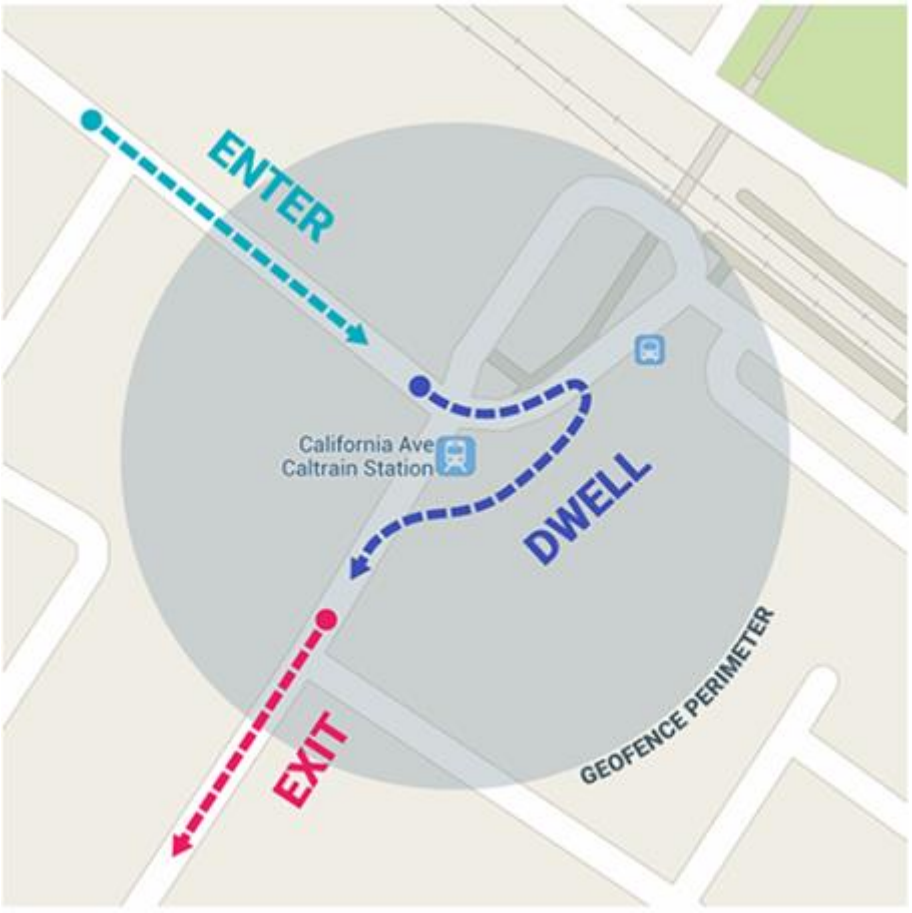


Figura 3: Cerca geográfica - Geofence

2.3. Desenvolvimento do Mapa de Empatia e Persona do público-alvo

O Mapa de Empatia foi desenvolvido descrevendo o que a persona (Gestor e Diretor de Inovação), pensa e sente, ouve, vê, diz e faz.

Mapa De Empatia

Como a empatia diz respeito a se colocar no lugar do outro compreendendo os seus sentimentos e necessidades o mapa de empatia tem a finalidade estrutural de conhecer e entender o usuário ao qual vai se desenvolver o produto, além disso, consegue identificar o comportamento do usuário bem como seus desejos e assim promovendo melhor entendimento do produto final como mostrado na figura 4.

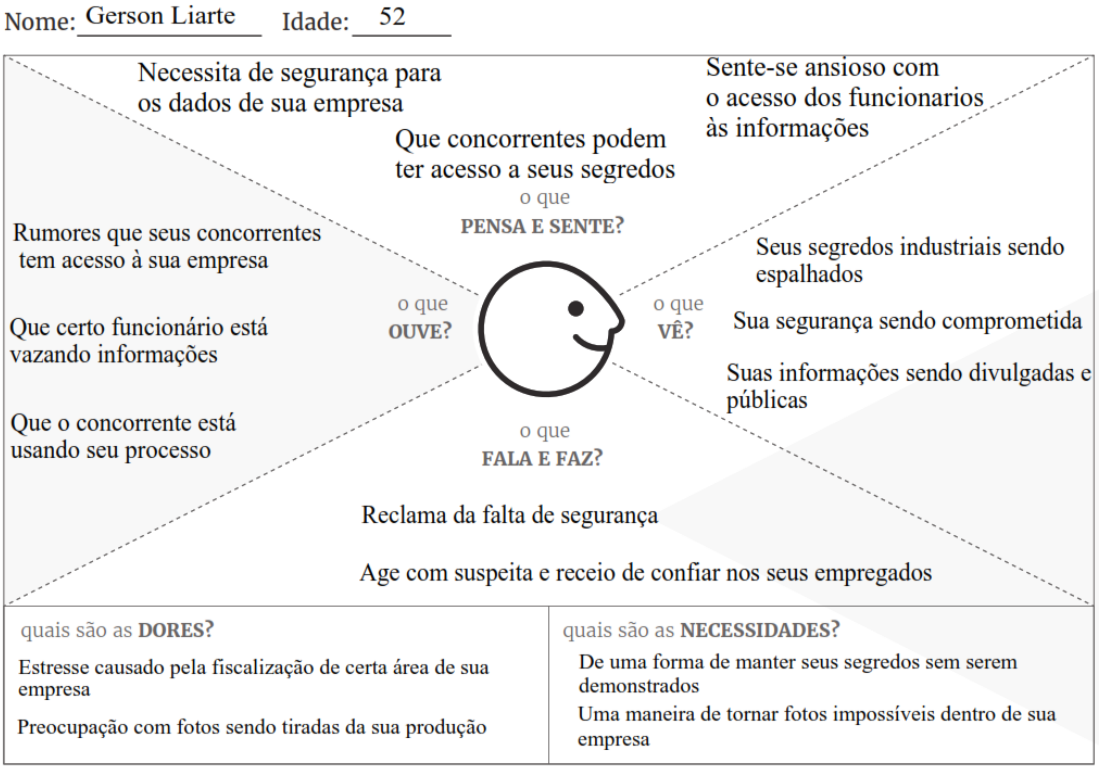


Figura 4: Mapa de empatia

A figura acima demonstra o mapa de empatia descreve o estado emocional da pessoa de um gestor de diretor de inovação a equipe saiu da mera ideia e colocou no papel dividindo por categorias previamente definidas nos quadrantes para uma melhor visualização de suas dores, necessidades e sentimentos. Com essas duas ferramentas o time de desenvolvimento pode se basear para a definição do backlog do produto.

2.4. Personas

É uma poderosa ferramenta utilizada durante as fases do design thinking, através desse método podemos identificar arquétipos (essa identificação pode variar desde aspectos demográficos, sexo, faixa etária e classes sociais) ou personagens fictícios conforme a figura 5.

Gerson

Gestor e Diretor de Inovação

Empresa: Indústria de Software de Cyber Segurança

Idade: 52 anos

Genêro: Masculino

Educação: Ensino superior

Mídias: Gerencia atividades por aplicativo

Objetivos: Privacidade para o desenvolvimento de seus produtos Evitar vazamento de informações

Desafios: Eu como gestor gostaria de gerenciar as funcionalidades do aplicativo para eu mesmo personalizar as áreas a serem bloqueadas. Eu como gestor desejo que seja implementado uma cerca (por GPS) para bloquear uma câmera do celular em uma área pré-definida. Eu como gestor gostaria de impossibilitar o desligamento do GPS para que não haja um furo na funcionalidade. Eu como gestor gostaria de Implementar o bloqueio da câmera por região em um determinado na área da fábrica de software.

Como minha empresa pode ajudá-la: Desenvolvimento de um produto para bloquear a camera por GPS

Figura 5: Persona idealizada para o desafio

2.5. Scrum

É um tipo de metodologia ágil que foi utilizada durante o processo de construção do GeoBlockCam, ele consiste na implementação de um ciclo para o desenvolvimento da atividade que é previamente definida com documentos chamados de backlogs como pode ser observado na figura 6.

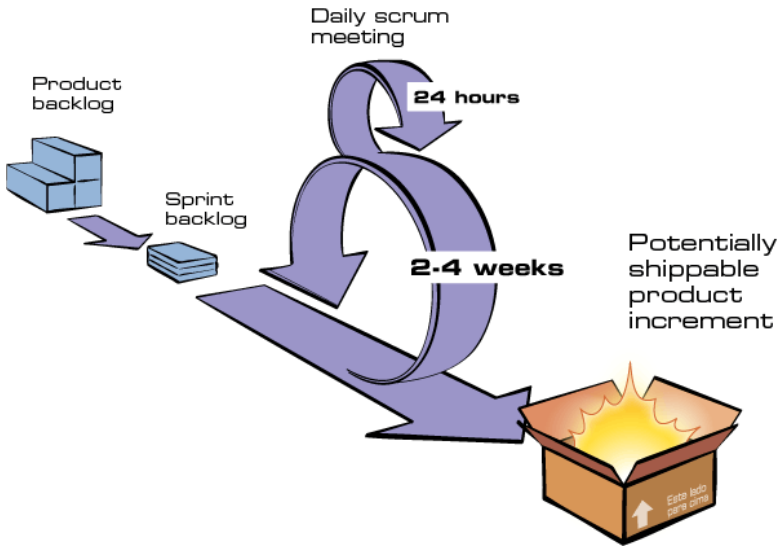


Figura 6: Metodologia Ágil - Scrum.

2.5.1. Sprint

Representa a janela de tempo dentro do qual um conjunto de atividades devem ser realizadas. No Scrum, as atividades são incrementam e moldam-se de acordo com a realização das sprints, no caso do desenvolvimento do GeoBlockCam foram utilizadas sprints com duração de 1 semana para realização das atividades de acordo com os critérios estabelecidos pelos product owners.

2.5.2. Product Owner

É a pessoa responsável por definir os itens que compõem o Product Backlog e durante a Sprint Planning os prioriza. Durante a realização de uma Sprint, é responsabilidade do product owner verificar os requisitos porém não poderá alterá-los, apenas após a realização deste ciclo.

2.5.3. Scrum Master

É quem procura assegurar que a equipe respeite e siga os valores e as práticas do scrum, atua como facilitador do Daily Scrum e torna-se responsável por remover quaisquer obstáculos que sejam levantados pela equipe durante essas reuniões.

2.5.4. Scrum Team

É a equipe de desenvolvimento. Não há necessariamente uma divisão funcional através de papéis tradicionais, todos no projeto trabalham juntos para completar o conjunto de trabalho com o qual colaboram em conjunto para um Sprint.

2.5.5. Product Backlog

É uma lista contendo todas as funcionalidades desejadas para o produto definidas pelo Product Owner. O product backlog não precisa estar completo no início, no decorrer do projeto ele cresce e muda à medida que se aprende mais sobre o produto e seu público alvo.

2.5.6. Sprint Backlog

É fruto da divisão das funcionalidades definidas no product backlog em tarefas que serão realizadas ao decorrer do sprint, o scrum master mantém o Sprint Backlog atualizado refletindo as atividades que são completadas e o tempo a equipe acredita ser necessário para finalizar aquelas que ainda não estão prontas.

2.5.7. Daily Scrum

Todos os dias durante uma sprint, há uma reunião onde a equipe definirá as atividades que serão realizadas, que já foram realizadas e levantar as dificuldades e o andamento das atividades no geral. As dificuldades encontradas devem ser trabalhadas pelo scrum master em conjunto com a equipe.

2.5.8. Sprint Planning

É uma reunião realizada antes da sprint ser realizada onde o product owner descreve as funcionalidades de maior prioridade para a equipe e deve-se dividir as funcionalidades em tarefas técnicas, após a reunião. Coletivamente, o scrum team e o product owner definem um objetivo para o sprint, que é uma breve descrição daquilo que se tentará alcançar no fim da sprint.

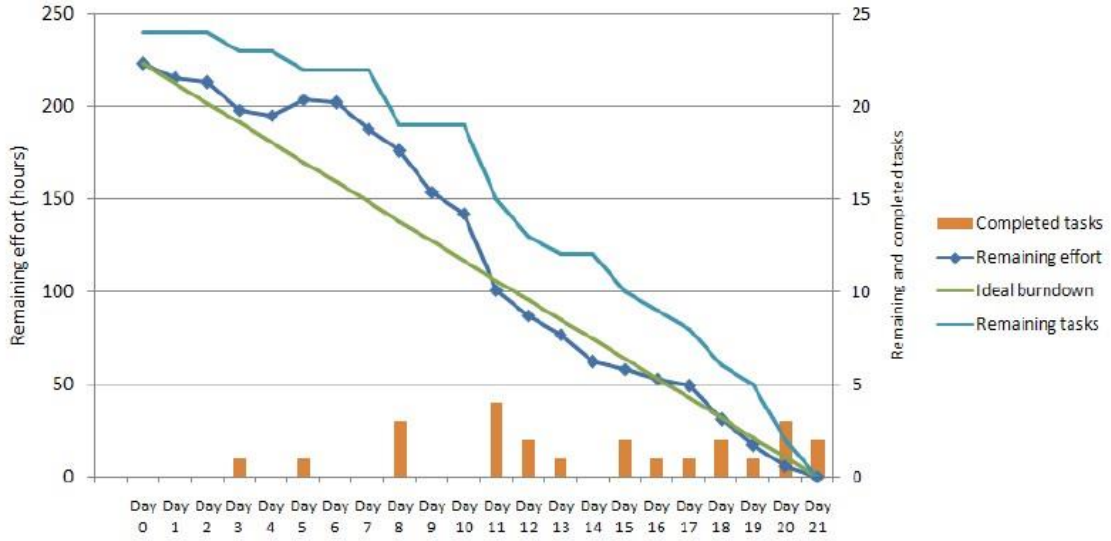


Figura 7: Gráfico Burndown.

2.5.9. Gráfico de Burndown

É uma forma de visualizar o progresso durante uma sprint, o scrum master realiza a verificação das atividades realizadas e monitora seu progresso, possuindo a habilidade de gerar uma linha do tempo do progresso da sprint, assim como visualizado na figura 7.

3. Metodologia

A metodologia proposta para este trabalho caracteriza-se pelo desenvolvimento de uma aplicação que possibilitasse o gerenciamento do uso da câmera em uma área predeterminada. A ideia central é que as etapas de gerenciamento e desenvolvimento sejam realizadas utilizando-se o framework AOSP executado no sistema operacional Linux. Nesta seção, serão apresentadas as ferramentas computacionais e a implementação para obtenção dos resultados.

3.1. Ferramentas computacionais

Nesta seção, descreve-se a seguir que aspectos as ferramentas computacionais devem contemplar para serem utilizadas para o desenvolvimento da aplicação.

1) *Android Open Source Project (AOSP)*: É uma versão do Android que é um sistema operacional de código aberto para dispositivos móveis com um projeto de código aberto correspondente liderado pelo Google, sem ferramentas ou aplicações proprietárias e que possui código personalizável.

2) *Sistema Operacional Linux*: Possibilita a execução de programas em um computador e outros dispositivos. Para o propósito da aplicação era importante um sistema operacional leve e seu kernel facilita o controle de como será usado o processador, a memória, o disco e dispositivos periféricos.

3) *Android Studio*: Ambiente de desenvolvimento integrado utilizado em mais alto nível para personalizar a visualização dos arquivos do projeto para se concentrar em aspectos específicos do desenvolvimento do app. Possibilitou personalizar, configurar e ampliar o processo de programação do APK.
4) *Gitlab*: Utilizado como gerenciador de repositório de software baseado em git, com suporte a Wiki, gerenciamento de tarefas e CI/CD.

5) Visual Studio Code: é um editor de código-fonte que inclui suporte para depuração, controle de versionamento Git incorporado, realce de sintaxe, complementação inteligente de código, snippets e refatoração de código. Ele é um software livre e de código aberto utilizado para edição dos arquivos do AOSP utilizados.

3.2. Implementação do Algoritmo

A implementação deste trabalho foi realizada através da elaboração de um script em linguagem JAVA. Este script foi escrito na ferramenta Visual Studio Code.

Esse algoritmo propõe o bloqueio da câmera por geolocalização através da comunicação das instância do objeto Câmera e a instância do objeto de Localização(GPS). No nível da estrutura do aplicativo está o código do aplicativo, que usa a API Camera 2 para interagir com o hardware da câmera. Internamente, esse código chama as interfaces Binder correspondentes para acessar o código nativo que interage com a câmera. Para o GPS, os drivers de localização do usuário permitem que seu aplicativo publique atualizações na localização física do dispositivo por meio dos serviços de localização do Android recuperando os dados de latitude e longitude. Também foi necessário criar um método de integração da camada da aplicação com a camada onde se localizam os sensores e hardware.

4. Resultados Obtidos

Nesta seção o que se pode concluir sobre a aplicação desenvolvida é que os dados registrados de longitude, latitude e raio foram gerados com sucesso. Isso torna útil uma análise tanto dos desenvolvedores do sistema quanto da aplicação, sobre a programação dessas aplicações e seus impactos no dispositivo.

4.1. Contextualização do Desafio Proposto

4.1.1. Problema

Inicialmente nos foi proposto que fosse criado uma solução para implementar o bloqueio da câmera por geolocalização, deveria bloquear a câmera em um determinado local com a implementação de uma cerca por GPS e se o fosse desligado, mesmo assim a câmera continua bloqueada.

4.1.2. Solução

Foi obtido um bom resultado, mesmo com recursos limitados, foi desenvolvido o App GeoBlockCam. Foi elaborado um App para leitura dos valores de latitude, longitude e raio. Bloqueio por geolocalização: Indicar uma área de bloqueio circular com as coordenadas indicadas, Caso o GPS esteja desligado a câmera estará desligada.

Figura 8: MVP - Minimum Viable Product

O controle direto da câmera de um dispositivo requer um código muito maior do que o necessário para solicitar imagens ou vídeos de aplicativos de câmera. Para criar um aplicativo de câmera especializado ou algo totalmente integrado à IU foi necessário conseguir uma instância do objeto Câmera Manager que gerencia o serviço do sistema para detectar, caracterizar e conectar CameraDevices, sendo assim efetuando a comunicação com dispositivos de câmera.

Para obter mais detalhes sobre a comunicação com dispositivos de câmera, tendo como primeira etapa o processo de controlar a câmera diretamente. A maneira recomendada de acessar a câmera é abrir camera em uma linha de execução separada, iniciada a partir de onCreate(), como é feito pelo aplicativo Câmera do Android. Essa é uma boa abordagem, já que o acesso à câmera pode demorar um pouco e pode sobrecarregar a linha de execução de IU. Em uma implementação mais básica, o processo de abrir a câmera pode ser transferido para o método onResume() para facilitar a reutilização do código e manter o fluxo de controle simples.

Apesar deste algoritmo caminhar na direção correta para a correção de erros e proporcionar bons resultados, ele deixa um tanto a desejar na velocidade de convergência em busca da minimização dos erros. A principal dificuldade está relacionada com a falta de suporte às API 's do Google que está atrelada ao fato do AOSP ser open source.

A Classe CameraManager é a classe responsável pelo controle da câmera, e está disponível no diretório: frameworks/base/core/java/android/hardware/camera2/

Métodos utilizados elaborados e implementados pela equipe:

private void SetParams(Double latitudeblock, Double longitudeblock, Double raioblock)
private boolean IsConditionCamera()
private CameraDevice openCameraDeviceUserAsync()

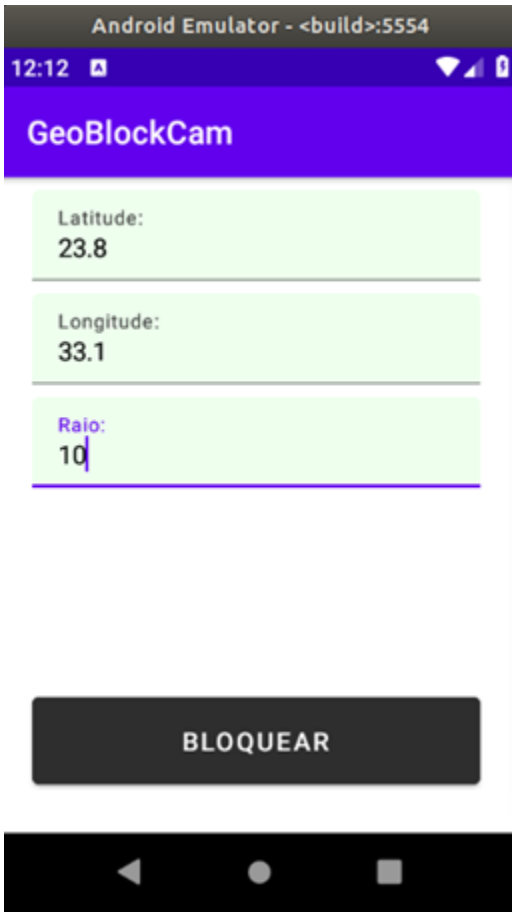


Figura 9: GeoBlockCam - Controle da camera.

Na implementação do GPS é necessário criar um driver que é responsável por se comunicar com o hardware conectado, monitorar as alterações de local válidas e relatar essas alterações à estrutura. Na construção do script foi implementado e personalizado da seguinte forma:

Method Location getLastKnownLocation (bestprovider): A chamada para getLastKnownLocation()não bloquear - o que significa que irá retornar null se não houver posição atual disponível.

Method boolean isLocationEnabled(): Retorna o estado atual habilitado / desabilitado do local. Para ouvir as mudanças, seu retorno é booleano executando verdadeiro se o local estiver habilitado(True) e falso se o local estiver desabilitado(false).

O método *SetParams* recebe os parâmetros de longitude e latitude vindos da camada da App.

O import da classe LocationManager:
import android.location.;*

Em seguida, instanciados da seguinte forma:

LocationManager locationManager = (LocationManager) mContext.getSystemService(Context.LOCATION_SERVICE);

Para completar a implementação e obter coordenadas de localização no (CameraManager) foram implementados:

```
Double latitude = location.getLatitude();
Double longitude = location.getLongitude();
boolean condition = locationManager.isLocationEnabled();
```

Método *set* criado para implementar a persistência dos parâmetros recebidos utilizando o método set.

```
SystemProperties.set("persist.systemlatitude",
Double.toString(latitudeblock));

SystemProperties.set("persist.systemlongitude",
Double.toString(longitudeblock));

SystemProperties.set("persist.systemraio",
Double.toString(raioblock));
```

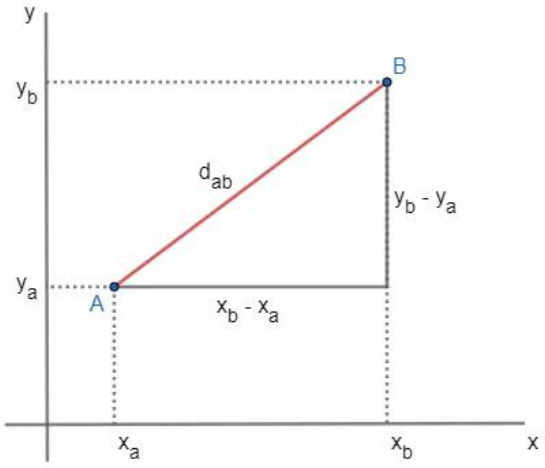
Método IsConditioncamera(): É Invocado no launch da câmera. Retorna um valor booleano, caso seja True a Câmera poderá ser utilizada, ou seja False a Câmera não poderá ser utilizada.

boolean ConditionOnCamera = IsConditionCamera();

Este método também foi personalizado para verificar se o GPS está ativo ou não, caso esteja On a Câmera poderá ser habilitada, e se estiver Off a Câmera será desabilitada. Recebe os dados de localização atuais do usuário final (latitude e longitude). Recebe os dados de localização e raio de bloqueio da câmera, vindos da app através das SystemProperties. E executa a validação da distância do usuário final para o centro da zona de bloqueio, caso a:

distância <= raio: A Câmera será desabilitada.
distância > raio: A Câmera poderá ser habilitada

Implementação da Geofence para Bloqueio de Câmera - Distância de Pontos: conforme a figura abaixo foram elaborados cálculos matemáticos para a precisão da geofence.



$$D^2 = (x_b - x_a)^2 + (y_b - y_a)^2$$
$$\sqrt{D^2} = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$
$$D = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$$

Figura 10: Cálculos matemáticos para a precisão da geofence – Distância de Pontos.

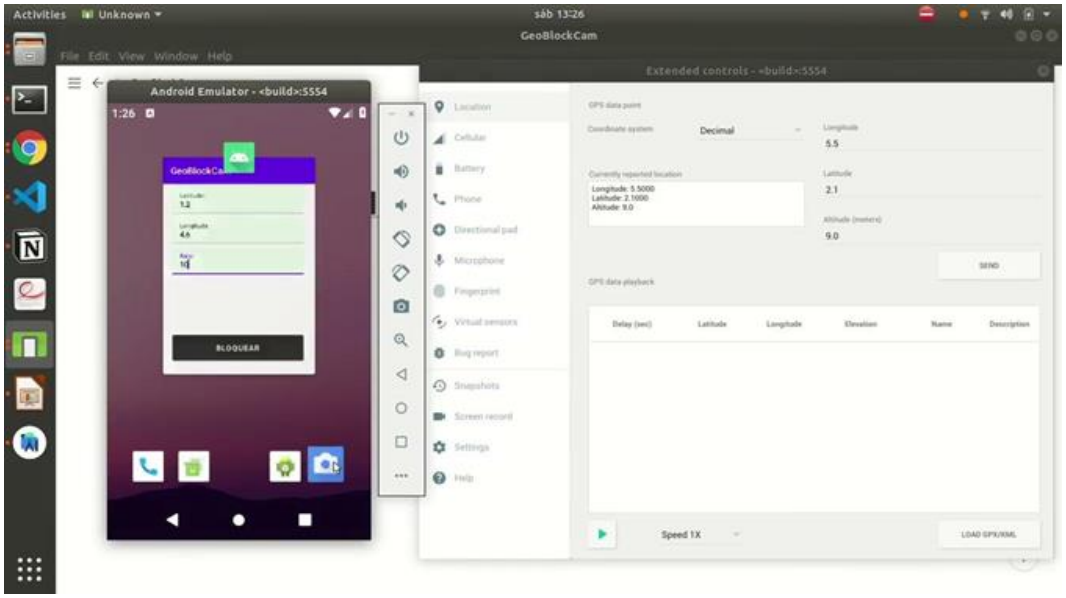


Figura 11: Demostração da lógica da distância>raio implementada.

4.1.3. Hipóstese 1 - Bloqueio por geolocalização:

latitudeAtual = 2.1 longitudeAtual = 5.5
latitudeblock = 10.2 longitudeblock = 11.6 raioblock = 5.0
isLocationEnabled : True
distância entre os pontos = 10.140019723846695

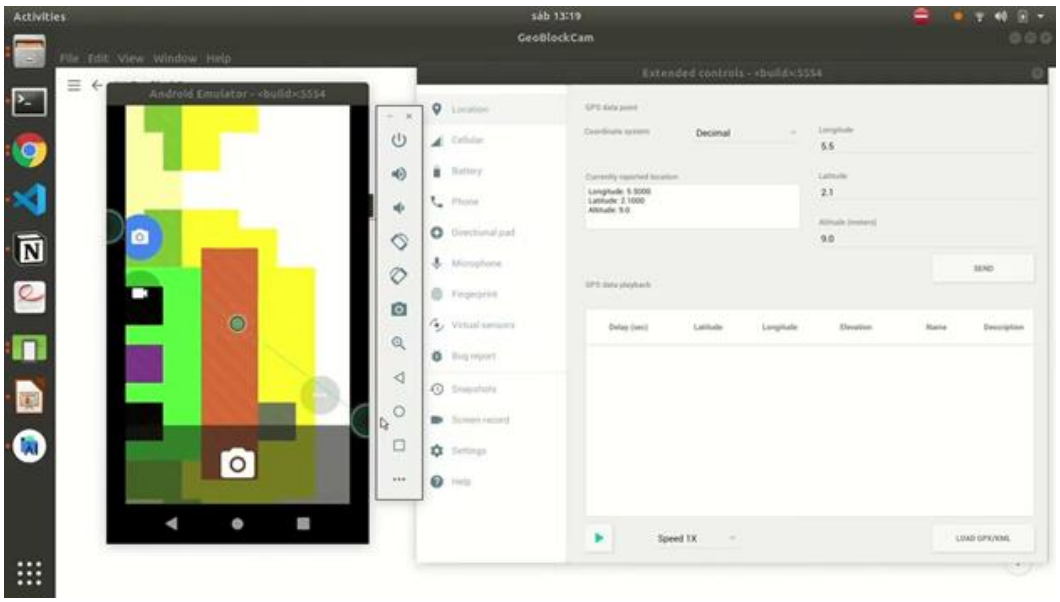


Figura 12: Evidência prática da hipótese 1.

4.1.4. Hipóstese 2 - Distância < Raio

latitudeAtual = 2.1 longitudeAtual = 5.5
latitudeblock = 1.2 longitudeblock = 4.6 raioblock = 10.0
distância entre os pontos = 1.272792206135786

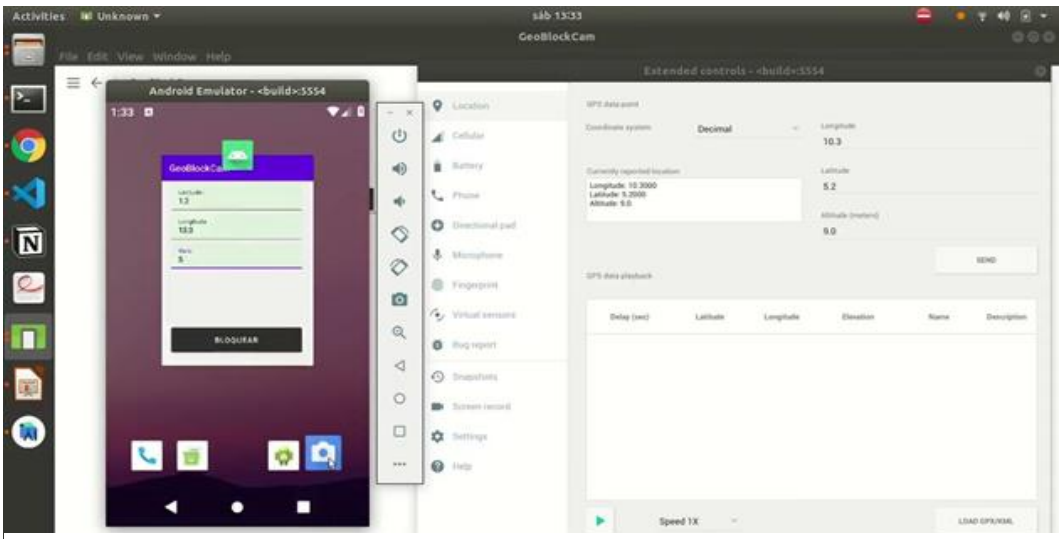


Figura 13: Evidência prática da hipótese 2.

4.1.5. Hipóstese 3 - DIstância = Raio

latitudeAtual = 5.2 longitudeAtual = 10.3
atitudeblock = 1.2 longitudeblock = 13.3 raioblock = 5.0
isLocationEnabled distancia entre os pontos = 5.0
isLocationEnabled : True

4.2. Metas e Desempenho da Equipe

Em geral, é possível expressar o desempenho ou performance do ente que se pretende avaliar utilizando-se uma métrica, função ou índice de desempenho em relação às metas, requisitos ou expectativas previamente definidos. Nosso time teve um desempenho considerável alcançando 86,49% das metas cumpridas em relação ao total de 37 tarefas. Foi elaborado uma timeline para demonstrar todos os resultados obtidos em cada etapa do desenvolvimento, conforme mostra a figura 14.

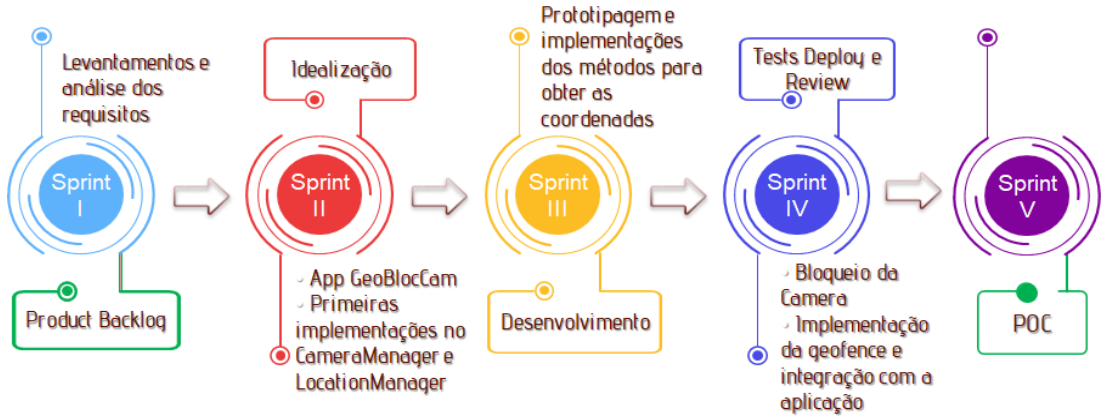


Figura 14: Timeline das etapas do desenvolvimento nas Sprints.

Para a elaboração das tarefas o time realizava uma reunião para planejar o que deveria ser feito, pois nossas entregas eram semanais, na ferramenta excell eram listadas e analisadas para a definição do esforço e tempo de realização, conforme mostra a figura 15.

Backlog					
#ID	Atividades	Responsáveis	Tempo Estimado	Esforço	Goals
#T1	Implementar no system proprieties método para salvar as coordenadas recuperadas	Juce | Johnson | John Kinderman	15h	Alto	0%
#T2	Testes e implementação da POC no Pixel	Juce | Johnson | John Kinderman	02h	Médio	0%
#T3	Persistência dos dados no aosp	Juce | Johnson | John Kinderman	15h	Alto	5%
#T4	Parametrizar os dados se latitude e longitude da app no aosp	Juce | Johnson | John Kinderman	15h	Alto	20%
#T5	Pesquisas e estudos para o layout da app	Thyago | Anthony	15h	Médio	0%
#T6	Preparar Apresentações de Sprint Planning e Defesa IV	Juce	04h	Médio	50%
Total de Estimativas de horas para atividades: **66h**					

Figura 15: Sprint Backlog para entregas semanais.

Com este acompanhamento era possível elaborar o gráfico de Burndown que era atualizado diariamente, e teve um papel muito importante no gerenciamento das atividades, proporcionando para a equipe trabalhar com mais segurança e alvos definidos, conforme figura 16.

		Base Line	Realizado
14/6	dia 0	6	6
15/6	dia 1	5	5
16/6	dia 2	4	3
17/6	dia 3	3	1
18/6	dia 4	2	2
19/6	dia 5	0	0

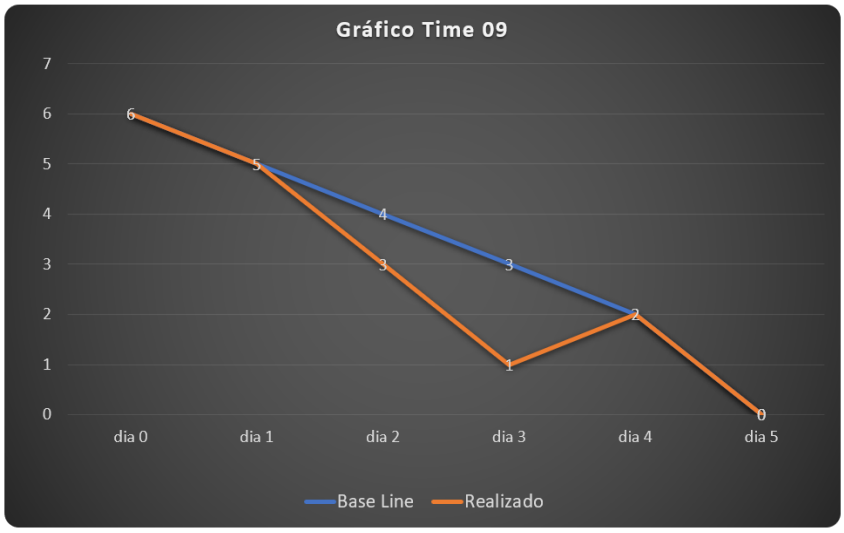


Figura 16: Gráfico Brurndown

Utilizando o Gitlab ao aplicarmos este processo, os desenvolvedores ao acessar a issue têm uma visão de tudo o que foi pesquisado, desenhado e testado até o desenvolvimento, aumentando o seu conhecimento sobre o que é e o porquê esta atividade deve ser realizada. Em alguns casos, devido ao tamanho do escopo, o time de desenvolvimento realizava uma sprint review um planejamento e "quebra" o projeto principal em outras issues são abertas em menores partes, sendo referenciada a issue principal em todas estas e também vinculada a milestone do projeto, conforme a figura 16\7.

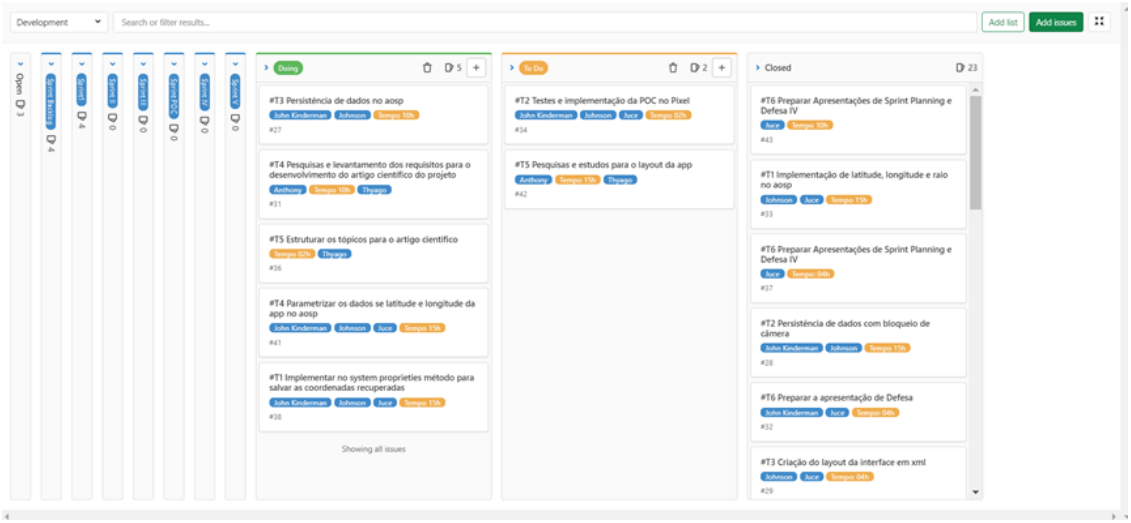


Figura 15: Gerenciamento no Kanban na plataforma Gitlab.

5. Dificuldades Encontradas

Como este projeto está sendo desenvolvido para inovar abrangendo vários setores, tivemos muitas dificuldades e não encontramos trabalhos relacionados e a documentação ainda é muito escassa. Também foram enfrentados problemas computacionais e de conexão remota. Outro ponto muito importante foi que devido

ao trabalho remoto ficou muito difícil a comunicação da equipe para reuniões e aplicar as implementações nos dispositivos físicos.

6. Considerações finais

Em relação aos resultados obtidos, a janela de tempo disponibilizada para o desenvolvimento foi o suficiente para a implementação inicial. As técnicas implementadas foram limitadas pela indisponibilidade de ferramentas para monitoramento posicional e mapas. O GeoBlockCam está implementado em o mais baixo nível, sendo possível para futuras iterações poder utilizar-se de recursos como o Open Google Apps Project para uma implementação mais completa com as funcionalidades incluídas nas bibliotecas do Google para o desenvolvimento do projeto.

6. References

Disponível em: CameraManager <https://developer.android.com/reference/android/hardware/camera2/CameraManager> Acesso em: 14 mai 2021.

Disponível em: LocationManager <https://developer.android.com/reference/android/location/LocationManager> Acesso em: 14 mai 2021.

Disponível em: Criar e monitorar fronteiras geográficas virtuais <https://developer.android.com/training/location/geofencing?hl=pt-br> Acesso em: 12 mai 2021.

Disponível em: http://androidxref.com/ </frameworks/base/core/java/android/hardware/location/> Acesso em: 12 mai 2021.