CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB

PROGRAMA DE INICIAÇÃO CIENTÍFICA

RODRIGO PEREIRA DA ROCHA

AMFIT

APLICAÇÃO MÓVEL DE FISCALIZAÇÃO DE TRÂNSITO

BRASÍLIA – DF

2019

RODRIGO PEREIRA DA ROCHA

AMFIT

APLICAÇÃO MÓVEL DE FISCALIZAÇÃO DE TRÂNSITO

Relatório final de pesquisa de Iniciação Científica apresentado à Assessoria de Pós-Graduação e Pesquisa.

Orientação: Phd Paulo Rogério Foina

BRASÍLIA – DF

2019

RESUMO

Este projeto de pesquisa apresenta uma plataforma de denúncia móvel, automatizada e econômica capaz de complementar de maneira eficiente os processos de fiscalização de trânsito brasileiros, permitindo a punição e conscientização da sociedade a respeito de delitos que apesar de serem de fácil identificação, como estacionar em fila dupla e em frente a garagens, os agentes humanos são incapazes de oferecer atuação constante. E ainda, inserir a sociedade de forma competente no processo de policiamento de tráfego. O sistema foi desenvolvido de modo incremental, utilizando metodologias ágeis para levantamento dos requisitos de forma organizada, planejamento de curto a médio prazo dos procedimentos e correção acelerada de eventuais erros operacionais. A arquitetura elaborada integra uma aplicação mobile para android e iOS para captação das delações em formulário descritivo e imagem, um modelo de Backend as a service para estruturação e armazenamento de dados em um banco de dados NoSQL em tempo real e uma API estruturada segundo o padrão REST para distribuição dos dados para consumo de sistemas ligados ao serviço público. Os resultados obtidos foram promissores quanto a velocidade de transmissão das informações para consulta, que são realizados em segundos pelo Firebase database e também com relação a captação de meta informação, uma vez que em conjunto com a foto do delito são capturados: data, horário, posição geográfica e uma classificação prévia formulada pelo denunciante, viabilizando futuros estudos sobre locais e horários com maior quantidade de infrações e direcionamentos para políticas públicas. Também foram obtidos resultados interessantes em um sistema web, acoplado a API que instrui os usuários não técnicos a fazerem consumo da API. Por fim, não foram obtidos resultados satisfatórios quanto a utilização de uma técnica de reconhecimento óptico de caracteres para detectar automaticamente a placa de identificação do automóvel.

Palavras-Chave: Desenvolvimento mobile. Arquitetura REST. Fiscalização pública.

SUMÁRIO

INTRODUÇÃO

A engenharia de software (Sommerville, 2019), é definida como o estudo de métodos para o planejamento, concepção, organização e aprimoramento de sistemas tecnológicos capazes de automatizar ou suprir alguma necessidade humana ou de negócio, é um componente crítico para o crescimento de qualquer mercado ou órgão público. Neste contexto, o desenvolvimento de aplicações móveis busca uma aproximação na relação software e usuário por meio do acesso em plataformas como smartphones e tablets. Aproveitando a arquitetura única dos dispositivos como: câmera, tela sensível ao toque, GPS e captação de áudio para oferecer soluções de fácil uso e acesso independe do local em que o utilizador se encontra.

Apesar desse nível de tecnologia já estar disponível, a gestão pública brasileira ainda investe em modelos de fiscalização de trânsito ultrapassados, baseados na maioria dos casos em agentes humanos, que são incapazes de lidar com a demanda atual de infrações. Não só isso, os modelos atuais fracassam ao deixar a legislação vulnerável a desvios de caráter por parte dos fiscais que podem ser subornados por infratores ou até ameaçados em alguns casos. Mesmo o modelo de monitoramento automatizado atual, constituído por radares eletrônicos, é ineficaz contra este tipo de infração mais simples, uma vez que suas câmeras foram projetadas para captar apenas excesso de velocidade e transposição de semáforos vermelhos.

Mesmo que infrações simples inicialmente aparentem não gerar mais do que pequenos transtornos diários a vida da população como a lentidão no tráfego e atraso dos transportes públicos, seus impactos na sociedade são muito maiores. Economicamente, a falta de respeito ao código de trânsito reverbera em prejuízo a pequenos comércios locais que tem seu acesso restringido devido ao comportamento irresponsável de motoristas. Já na saúde pública, locais muito movimentados que é o caso de escolas, hospitais e estádios os condutores optam por deixar os veículos irregularmente nas proximidades, atrapalhando a mobilidade nas vias e levando a batidas e atropelamentos.

Incentivado pelo artigo (C. Oduor, 2014) no qual o desenvolvimento mobile de software foi empregado para elaboração de uma plataforma comunitária de delação de crimes e comunicação com a polícia, melhorando os processos e garantindo maior eficiência na segurança pública. E por (Scharff, 2010) onde a utilização da metodologia ágil scrum

obteve resultados expressivos na organização e entrega de aplicações móveis por estudantes universitários. Este projeto apresenta uma plataforma móvel de fiscalização totalmente desenvolvida seguindo a estrutura scrum, capaz de complementar de maneira eficiente os processo de monitoramento de trânsito. E ainda, inserir a população prejudicada diretamente no processo de vigilância enquanto reduz a demanda extrema para os fiscais do modelo atual.

FUNDAMENTAÇÃO TEÓRICA

Este capítulo expõe abrangentemente as principais tecnologias e tópicos teóricos empregados durante o desenvolvimento da plataforma mobile/web de delação e monitoramento de infrações de trânsito. Uma vez que a arquitetura é estruturada por quatro componentes: aplicação móvel, sistema de visão computacional (OCR), interface web e o modelo de BASS, esta revisão bibliográfica tratará de introduzir as abstrações referentes a estes três campos de conhecimento técnico.

Desenvolvimento Mobile:

O desenvolvimento mobile em termos práticos é o processo pelo qual um app mobile é desenvolvido para um dispositivo mobile como celulares, tablets e smartwatches segundo (Antunes, 2019) existem três tipos principais de aplicação mobile:

- **Aplicativos Nativos:** são aplicações que foram desenvolvidos na mesma linguagem de seu sistema operacional sendo objective C e swift para sistema iOS e java e kotlin para sistemas Android, normalmente esse tipo de desenvolvimento desfruta de uma comunicação simplificada com o hardware do aparelho sendo capaz de extrair o máximo de performance do telefone. Normalmente são encontrados em lojas de aplicativos como AppStore e a PlayStore, entretanto em alguns casos o fato de cada versão ser desenvolvida de uma maneira diferente dificulta a padronização e cada versão sai com uma interface levemente diferente.
- **Aplicativos Híbridos:** Aplicações que são programadas em linguagens não nativas mas entregam o mesmo software para ambos os sistemas operacionais com

performance inferior. Em geral existem duas abordagens para o desenvolvimento híbrido:

- **Web-view:** adotada por frameworks como PhoneGap e Ionic, o sistema de web-view usa as tecnologias web para abrir um navegador por baixo dos panos, possibilitando utilização do sistema via essa interface indireta.
- **Transpilação:** adotada por Soluções mais modernas como o React-native (Câmara, 2018) em que as aplicações são programadas na linguagem do framework (Js) e são transpiladas para o mesmo código em funcionamento nativo, ganhando produtividade sem perder a interface direta do sistema operacional.

- **Aplicações Web Progressivas:** Uma coleção de ferramentas que geralmente são aplicadas nas versões mobile de sites jornalísticos para que quando o usuário aceitar os termos devidos, um ícone referente aquele site se torne disponível para acesso no celular do usuário. A partir disso, o sistema é capaz de interagir com os recursos do aparelho como câmera, GPS e etc, não sendo necessário nenhum tipo de download e possuindo uma performance mais fraca que os aplicativos nativos.

Para este projeto, foi escolhida a abordagem Híbrida por transpiração com a biblioteca Js React-Native criada pelo facebook, tendo em vista que ela possui a interface direta com o sistema operacional sem a necessidade de browser e agiliza a fabricação de um mesmo sistema tanto para Android quanto para iOS com produtividade acelerada.

OCR:

A inteligência artificial ou apenas I.A, é definida como estudo e projeto de agentes inteligentes por meio de sistemas de processamento de dados (programa) visando atingir capacidades antes restritas a cognição humana. Neste sentido, Optical Character Recognition (OCR) é pertencente a um subconjunto que busca reproduzir a capacidade de extração e interpretação de informações a partir de dados visuais, a principal característica dos reconhecedores ópticos de caracteres é identificar letras e números em arquivos de imagem, mesmo que escritos a mão.

Bibliotecas O.C.R para React-Native:

Existem diferentes bibliotecas que estendem os funcionamentos do react-native para permitir o reconhecimento de óptico de caracteres de interesse por meio da câmera do celular, os mais populares são a tesseract-ocr e a firebase MLKit que utilizam um sistema mantido pelo google para detecção de caracteres em fotos de documentos. Entretanto, como um dos objetivos finais deste projeto é a apuração e delação de infrações foi escolhida uma biblioteca OCR mais próxima do problema de negócio enfrentado a openal-pr, capaz de reconhecer a placa de identificação automobilística em imagens com carros o que agilizará o processo de queixa, uma vez que o utilizador não precisará inserir nenhuma informação que já esteja presente na foto durante o formulário de acusação.

Arquitetura REST:

A transferência representacional de estado ou REST (Fielding, 2000), é uma arquitetura de software para interoperabilidade entre serviços web, seguindo práticas padronizadas de comunicação a partir da versão 1.1 do protocolo http. Suas principais características são a modelagem de qualquer recurso do mundo real como um elemento web e a interlocução de aplicações baseada inicialmente em quatro verbos do protocolo, sendo eles:

- **GET:** método responsável pela solicitação de dados para o servidor, de longe o método mais usado em sistemas web. Em aplicações móveis é rotineiramente usado com ponte entre o aparelho celular e uma banco de dados mais robusto que armazena as informações de maneira mais estruturada.
- **POST:** dispositivo para criação de recursos em um servidor, em aplicações para smartphone, geralmente é usado para a inserção de dados no banco de dados externo.
- **PUT:** verbo encarregado de fazer alterações nos dados já existentes no servidor, capaz de manipular as informações armazenadas para refletirem o estado atual do recurso, gerando assim atualizações constantes dos dados inseridos.
- **DELETE:** claramente o termo mais perigos entre os quatro listados, este método é capaz de remover dados, seja de forma lógica como deixando inacessível ao consumidor ou removendo permanentemente aquela informação do sistema servidor.

Uma vez que um dos objetivos finais desta pesquisa é a disponibilização das denúncias para órgãos públicos, essa arquitetura foi escolhida como interface da aplicação por ser a mais adotada nas diversas áreas ligadas a web, facilitando a organização, planejamento, correção de erros e busca de referências para a modelagem de dados. Outro fator de peso nessa escolha foi a representação abstrata de estados poderosa que a arquitetura propõe, permitindo maleabilidade de representação para as mais diversas situações. Além disso, olhando pelo ponto de vista do consumidor a padronização clara e estabelecida no mercado favorece a adoção por parte de futuros sistemas clientes dos órgãos públicos para se conectarem as delações.

BASS:

O modelo de backend as a service ou “sistema interno como serviço” em tradução livre, é um modelo de negócio adotado por grandes empresas como google e microsoft onde essas companhias hospedam aplicações mobile em suas arquitetura de nuvem, oferecendo aos desenvolvedores a capacidade de conectar seus apps a grandes bancos e sistemas de processamento robustos.

Esta categoria de serviço oferta auxílio no desenvolvimento do para pequenas empresas, que não terão que desenvolver sistemas complexos de backend para inserir sua aplicação no espaço competitivo.

Firebase:

Mantido pelo google o firebase é um dos mais populares sistemas de BASS da atualidade, segundo (Redação, 2018) mais de 1.5 milhões de aplicativos utilizam seus serviços, dentre eles os mais relevantes para este projeto são:

- **Firebase Authentication:** conjunto de funções com o intuito de descomplexificar a criação de sistemas de autenticação seguros, atribuindo uma experiência de login otimizada e compatibilidade com outros sistemas como contas de e-mail, facebook, twitter, github e até via número do telefone celular.
- **Cloud Functions:** grupo de ferramentas capazes de fazer a ponte entre aplicativo para telefones inteligentes e o sistema de nuvem, muito utilizado para fazer inserções e alterações em bancos de dados remotos.

- **Firebase Realtime Database:** um banco de dados NoSQL com formato de arquivos semelhante ao JSON cujo funcionamento em tempo real permite aos aplicativos conectados que possam sincronizar suas informações de maneira quase instantânea e compartilhá-las de maneira eficiente.

MÉTODO

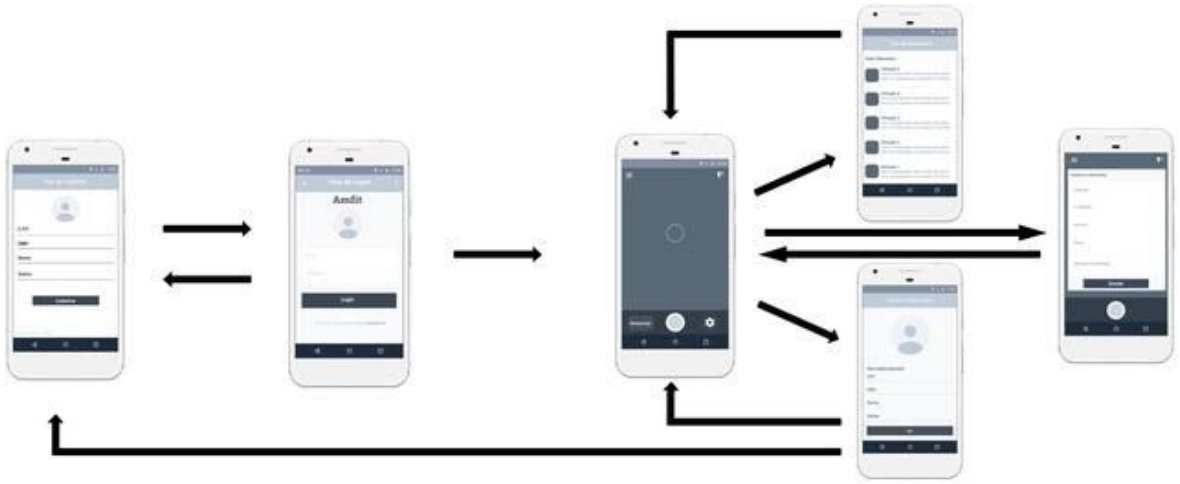
O desenvolvimento da plataforma iniciou com uma fase de estudo e planejamento da solução, houve um levantamento geral dos requisitos funcionais (capacidades que o software deve ter) e não funcionais (qualidades das funcionalidades) desejados para a solução do problema de pesquisa e pela identificação das restrições legais e regulatórias como proposto em (Sommerville, 2019). Em seguida, foi estudando o framework de gerenciamento de projetos Scrum e elaborada uma adaptação de sua estrutura para um para o meio acadêmico durante todo o programa PIBITI, onde o orientador realizava um papel misto entre product owner e scrum master, sanando dúvidas sobre o scrum e direcionando a entrega enquanto o orientando ficou responsável pelo papel do dev team, realizando a produção da lógica de programação e arquitetura interna da solução. Esta primeira fase chegou ao fim com a produção de user stories (Cohn, 2004) que descrevem características que geram valor para entrega final direcionadas pela visão do usuário que por sua vez encaminhou a formulação de um cronograma mais detalhado, com entregas constantes em ordem de prioridade e foco no resultado final.

A segunda fase foi marcada, a princípio, pelo aprofundamento no conhecimento da linguagem de programação javascript, onde foram testadas suas capacidades principais como a manipulação de DOM, o paradigma de programação funcional, a estrutura de software baseada em arquivos JSON e suas diferentes estruturas de dados (Groner, 2019), foram analisadas as principais ferramentas que compõem o ecossistema JS moderno por exemplo o transpilador babel.js, o runtime Node.js, o superconjunto typescript. A posteriori,, foi realizado um exame da biblioteca react.js, seus conceitos como a componentização de aplicações web, programação reativa, DOM virtual e sintaxe JSX foram assimilados e

adicionados ao cronograma do projeto como partes essenciais para produção tanto do mobile app quanto da solução em web. A segunda fase foi finalizada com o estudo da biblioteca react-native(Abbott, 2019), onde foram avaliados e testados seus artifícios em relação ao desenvolvimento móvel, como o sistema de banco de dados async-storage que só armazena dados em formato string a api responsável pela captação de dados do relógio/GPS do aparelho.

Durante a primeira etapa de codificação, foi confeccionado um esquema geral da interface do app e como suas diferentes telas deveriam se comunicar facilitando a usabilidade e a criação da identidade visual:

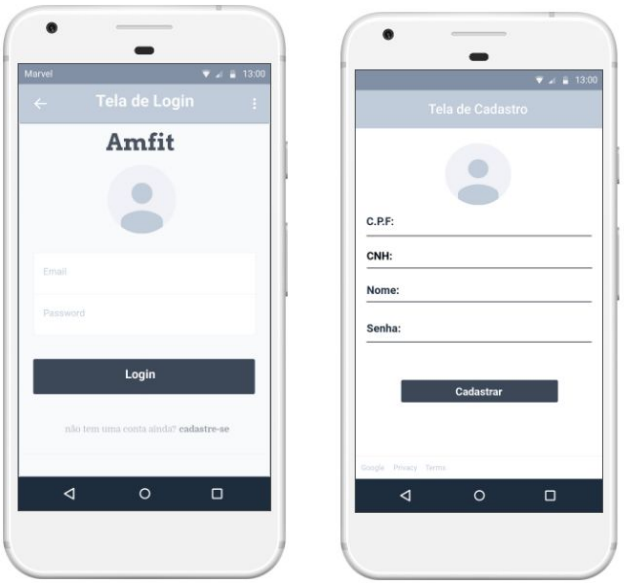
Figura 1: esquema geral da interface



Fonte: perfil pessoal em marvelapp.com

Em sequência, foi elaborada a tela de abertura como uma página de cadastro que pode se alternar com um tela de login caso o utilizador pressione o botão “já possuo cadastro”, ambas possuem conexão com o firebase authentication para criptografia das informações do operador quando necessário.

Figura 2 - tela de login e configuração do esquema



Fonte: perfil pessoal em marvelapp.com

Figura 3 - script, render da tela de login

```
render() {
  return(
    // Input E-mail
    <View style={ styles.container }>

      <Text style={ styles.titulo }>Tela de Login</Text>

      {/* Input e-mail */}
      <TextInput
        style={ styles.input }
        placeholder='Digite seu e-mail'
        value = {this.state.email}
        onChangeText={ email => this.setState( { email } ) }
      />

      {/* Input passord */}
      <TextInput
        style={ styles.input }
        placeholder='Digite sua senha'
        value = { this.state.password }
        onChangeText={ password => this.setState( { password } ) }
      />

      {/* Botão de Login */}
      <TouchableOpacity style={styles.button} onPress={this.login}>
        <Text style={styles.buttonText}>Logar!</Text>
      </TouchableOpacity>

      {/* tela de cadastro */}
      <TouchableOpacity onPress={this.TelaCadastro}>
        <Text style={styles.link}>Ainda não possui cadastro? cadastre-se aqui!</Text>
      </TouchableOpacity>

    </View>
  )
}
```

Fonte: Rodrigo Pereira da Rocha

Figura 4 - script, render da tela de cadastro

```
render(){

  return(
    <View style={ styles.container }>

      <Text style={ styles.titulo }>Tela de Cadastro</Text>

      {/* Input e-mail */}
      <TextInput
        style={ styles.input }
        placeholder='Digite seu e-mail'
        value = {this.state.email}
        onChangeText={ email => this.setState( { email } ) }
      />

      {/* Input passord */}
      <TextInput
        style={ styles.input }
        placeholder='Digite sua senha'
        value = {this.state.password}
        onChangeText={ password => this.setState( { password } ) }
      />

      {/* Botão de cadastro */}
      <TouchableOpacity style={styles.button} onPress={this.cadastrar}>
        <Text style={styles.buttonText}>Cadastrar</Text>
      </TouchableOpacity>

      {/* tela de login */}
      <TouchableOpacity onPress={this.TelaLogin}>
        <Text style={styles.link}>já possui cadastro? faça Login aqui!</Text>
      </TouchableOpacity>
    </View>
  )
}
```

Fonte: Rodrigo Pereira da Rocha

Após o login ou cadastro (caso seja a primeira entrada) os usuários são redirecionados para a interface de câmera (desenvolvida utilizando react-native-camera), onde três diferentes ícones: engrenagem, máquina fotográfica e lista de itens formam a interface em conjunto com a imagem captada pela câmera, cada um deles levando para uma parte diferente do sistema.

Figura 5 - tela de câmera do esquema inicial

Fonte: perfil pessoal em marvelapp.com

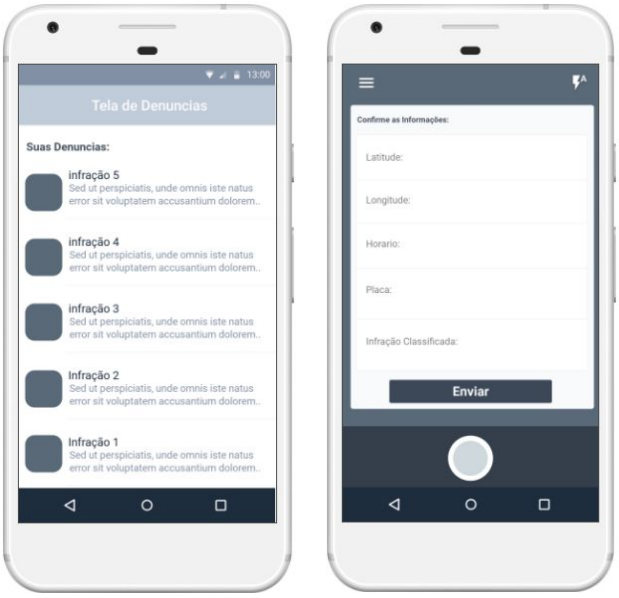
Figura 6 - script, render da tela de câmera

```
{/* Utilizando a classe que manipula a câmera */}
  <RNCamera

    // definindo a referencia do objeto câmera
    ref= { camera => { this.camera = camera } }

    // estilo da câmera
    style={styles.preview}

    // Prop para indicar se a câmera utilizada será a frontal ou traseira.
    type={RNCamera.Constants.Type.back}

    // Prop para indicar se ao capturar a imagem o Flash da câmera deve ser usado.
    flashMode={RNCamera.Constants.FlashMode.off}

  >

  {/* definição em formato de linha de varias opacidades tocaveis */}
  <View style={{ flex: 0, flexDirection: 'row', justifyContent: 'center' }}>

    {/* Implementação da biblioteca de icones vetoriais com o objetivo de criar um botão que leva para config */}
    <TouchableOpacity onPress={ this.TelaConfig } style={styles.capture}>
      <Text><Icon name="gear" size={30} color="#fff" /></Text>
    </TouchableOpacity>

    {/* Implementação da biblioteca de icones vetoriais com o objetivo de criar um botão que tira a foto */}
    <TouchableOpacity onPress={ this.TirarFoto } style={styles.capture}>
      <Text> <Icon name="camera" size={45} color="#fff" /></Text>
    </TouchableOpacity>

    {/* Implementação da biblioteca de icones vetoriais com o objetivo de criar um botão que leva para a lista  de fotos*/}
    <TouchableOpacity onPress={ this.TelaLista } style={styles.capture}>
      <Text><Icon name="list" size={30} color="#fff" /></Text>
    </TouchableOpacity>

  </RNCamera>
```

Fonte: Rodrigo Pereira da Rocha

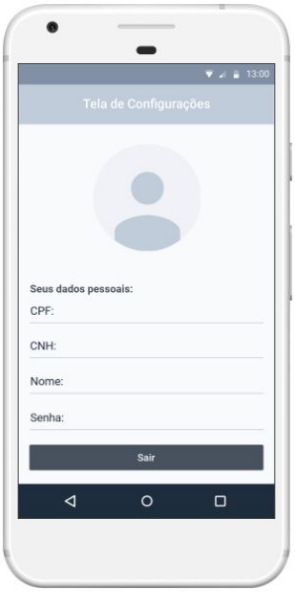
O ícone engrenagem direciona para uma seção de configuração, onde é possível mudar as informações seguindo um padrão CRUD, inclusive com a opção de remoção de conta. Já o emblema de lista, direciona para uma segmentação onde estão listadas todas as denúncias feitas, seus estados (em andamento/realizada) e suas datas. Por fim, o símbolo de máquina fotográfica grava a imagem da tela e abre um formulário de denúncia para que o operador escolha qual o tipo de infração.

Figura 7 - telas de lista e formulário do esquema inicial



Fonte: perfil pessoal em marvelapp.com

Figura 8 - tela de configuração do esquema inicial



Fonte: perfil pessoal em marvelapp.com

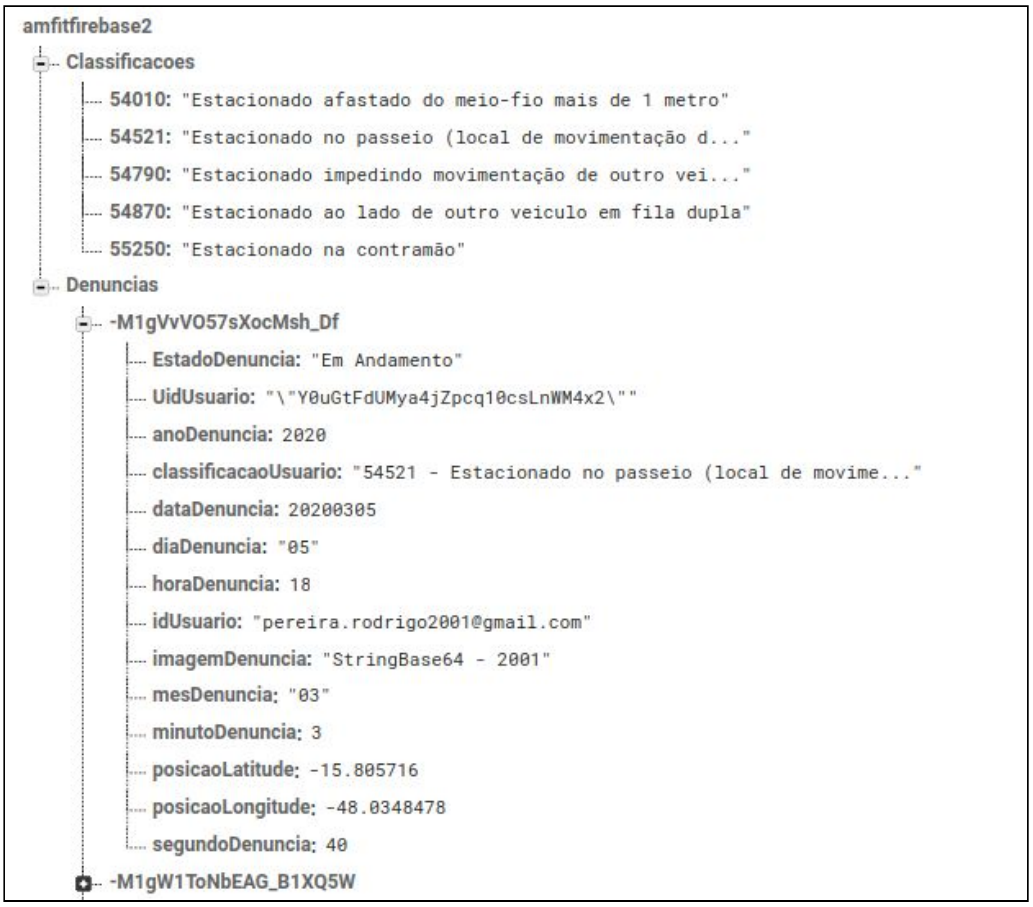
Figura 9 - script, render da tela de configuração

```
render() {
  return(
    <View style={ styles.container }>
      <View>
        <ModalEmail   ...
        />
        <ModalPassword ...
        />
      </View>
      <Text style={ styles.titulo }>Configurações</Text>
      <Text style={styles.subtitulo}>E-mail Cadastrado:</Text>
      <Text style={styles.info}>{this.state.email}</Text>
      <TouchableOpacity style={styles.subbutton} onPress={ this.showModalOne }>
        <Text style={styles.subbuttonText}>Alterar</Text>
      </TouchableOpacity>
      <Text style={styles.subtitulo}>Senha Cadastrada:</Text>
      <Text style={styles.info}>{this.state.password}</Text>
      <TouchableOpacity style={styles.subbutton} onPress={ this.showModalTwo }>
        <Text style={styles.subbuttonText}>Alterar</Text>
      </TouchableOpacity>

      {/* Botão de sair */}
      <TouchableOpacity style={styles.button} onPress={ this.logout }>
        <Text style={styles.buttonText}>Sair</Text>
      </TouchableOpacity>

      {/* Botão de deleção */}
      <TouchableOpacity style={styles.button2} onPress={ this.delete }>
        <Text style={styles.buttonText}>Deletar conta</Text>
      </TouchableOpacity>
    </View>
  )
}
```

Fonte: Rodrigo Pereira da Rocha

Após o primeiro período de codificação, foi necessária uma nova etapa de estudos, focada no envio dos metadados tais como: horário, latitude, longitude e descrição da denúncia em conjunto com a imagem capturada pelo aplicativo para o banco de dados NoSQL no firebase. O principal desafio neste momento era como modelar o banco de dados não relacional para armazenar os diversos tipos de metadados necessários. Após experimentos e testes de usabilidade o modelo que melhor se adequou aos processos de inserção (por meio do formulário do app) e consulta (por meio da lista de denúncias do app) foi uma estrutura divida em que as classificações de infração que o usuário pode escolher no formulário compõem uma estrutura separada das denúncias em si e dentro do mapeamento da denúncia estão inseridos seus metadados incluindo a imagem em string base64.

Figura 10 - captura de tela da estruturação do banco NoSQL



Fonte: perfil pessoal em firebase.com

Finalizado o processo de codificação da aplicação para celulares o próximo estágio foi o desenvolvimento de uma API no modelo de arquitetura REST (Fielding, 2000) para que os sistemas consumidor pudessem ter acesso às fiscalizações enviadas pelos clientes via smartphones. A primeira opção de desenvolvimento foi utilizar mais uma vez os serviços oferecidos pelo firebase, entretanto a API disponibilizada pelo BAAS teve um desempenho muito fraco em conjunto com a modelagem elaborada anteriormente além de possuir uma complexidade muito elevada para configuração de privilégios (praticamente qualquer usuário pode alterar e excluir dados) antes da configuração. Para contornar essa situação, foi montado um sistema web dividido em duas camadas, a primeira é a camada visual, que apresenta para os sistemas consumidor de denúncias como a API funciona e como consultar os dados, já a segunda camada, é a API em si capaz de processar as consultas e responder com os dados do banco em formato JSON. Ambas as camadas da solução foram desenvolvidas usando um Framework da linguagem python o Flask, a comunicação com o

firebase foi realizada com uma biblioteca auxiliar chamada pyrebase e o painel visual foi abstraído para uma outra biblioteca de documentações visuais chamada flask_restplus.

Figura 11 - captura de tela da Interface web da API REST

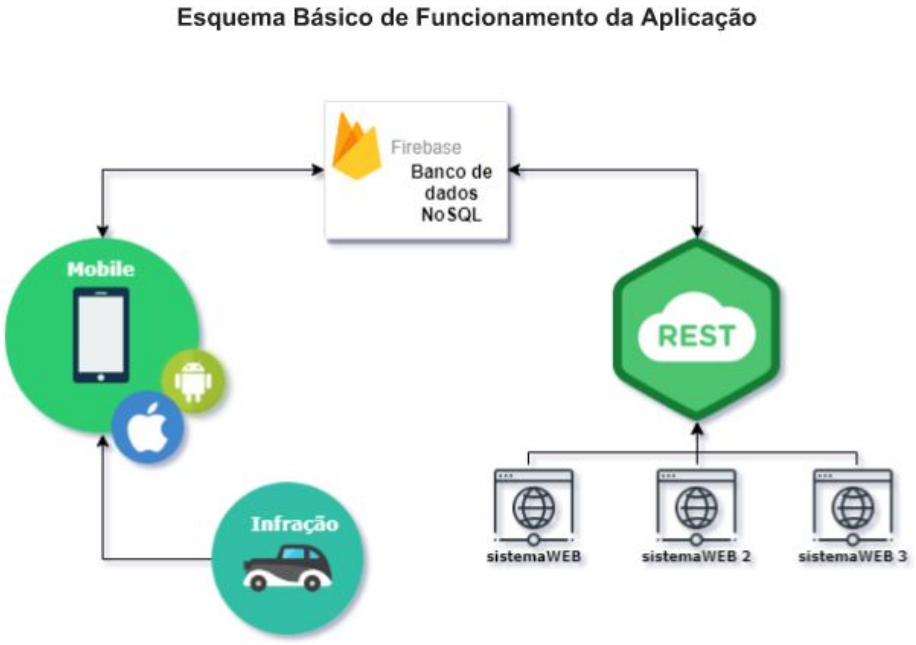
Fonte: sistema desenvolvido por rodrigo pereira da rocha

Por fim, foi pensado um sistema básico de visão computacional para o app. A idéia proposta era bem simples, uma das principais informações de interesse na imagem de delação é a placa de identificação do carro, uma vez que com a identificação única dela é possível localizar o veículo e aplicar a punição ao motorista responsável, para realizar essa atividade é necessário um sistema de OCR capaz de reconhecer os números da placa de identificação, converter esses números para string e enviá-los em formato JSON em conjunto com os demais metadados para o banco de dados em nuvem. Infelizmente dado o tempo investido nas demais atividades do projeto essa parte não foi completamente implementada, apesar dos inúmeros testes tanto com a biblioteca react-native-openal-pr quanto com o kit de machine learning do firebase em virtude do covid-19 e outros atrasos enfrentados durante o desenvolvimento os testes não puderam ser completados e a funcionalidade de OCR foi deixada de lado, mas todo o restante da plataforma de denúncia está implementado.

RESULTADOS E DISCUSSÃO:

A plataforma entregue é um app para celulares iOS e Android (com foco maior em Android) em que os utilizadores podem se cadastrar uma única vez e deixar o login automático, caso o utilizador se depare com uma infração que acredita ser imperceptível para os demais modelos de fiscalização ele pode sacar o aparelho telefone registrar a imagem da infração, preencher um formulário simples de delação e pressionar enviar. A partir deste momento a imagem e diversos metadados como: horário, latitude, longitude e etc, serão enviados para o banco de dados NoSQL dentro do sistema de backend as a service firebase onde poderão ser acessado em tempo real na tela de lista do app para ver o andamento da denúncia. Por fim, o sistema web em rede é capaz de consumir o banco de dados dentro do firebase e distribuir via API os dados das denúncias para diversos sistemas web ligados a órgãos públicos.

Figura 12 - captura de tela do diagrama visual da aplicação



Fonte: esquema básico desenvolvido por Rodrigo Pereira da Rocha com a ferramenta adobe illustrator

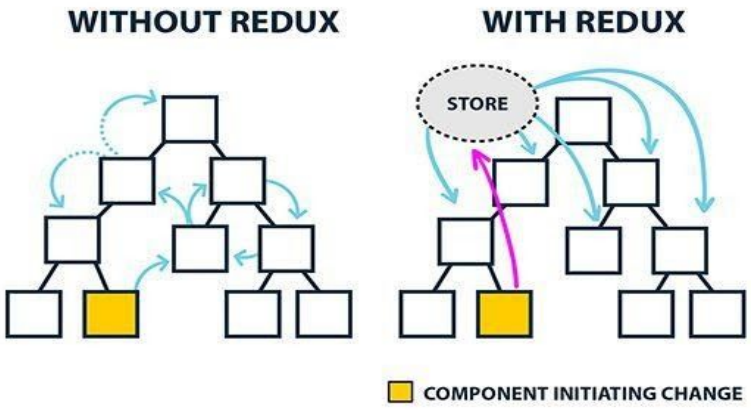
A respeito dos aspectos gerais do projeto, apesar da plataforma entregue suprir diversas carências enfrentadas pelos modelos atuais de fiscalização é primordial salientar que este projeto foi modelado para integração ao setor público, desde a documentação visual da API (com tutorial visual e instrutivo) até uma arquitetura que possibilita a mudança de estado de uma denúncia dentro do banco a partir de sistemas externos, como os de órgãos públicos fiscalizadores. Sendo assim, para que a plataforma continue sendo aprimorada e atinja seu pleno funcionamento, auxiliando investigações tomadas de decisões públicas, é necessária não só uma motivação dos colaboradores (orientando e orientador) e da academia científica, mas também é imprescindível a adoção por parte das instituições políticas, que em busca de uma melhora nos processos realizados, adotem a plataforma e implementem a solução em cenários reais para iterativamente melhorar seus resultados.

Sobre os aspectos estruturais de codificação do app, o desenvolvimento seguiu um processo de componentização dos recursos utilizados para reaproveitamento de código em diferentes situações. Para isso, os fragmentos foram segmentados em dois grupos, os que continham a lógica (stateful components) foram programados seguindo o padrão de classes do paradigma orientado a objetos, enquanto os demais, formados por elementos visuais foram desenvolvidos seguindo o padrão de funções puras do paradigma funcional. No entanto, durante o tempo de codificação da solução, houveram mudanças nas práticas componentização adotadas pela comunidade react-native e no momento o que é entendido como melhor prática é a abstração completa da lógica em segmentos funcionais, assim como os elementos visuais. Por enquanto, tais mudanças não possuem impactos diretos na solução mas é provável que em um futuro próximo isso interfira na manutenibilidade do software uma vez que o suporte a esse tipo de estrutura vá diminuir com o tempo.

Já com relação a usabilidade do app em si, o protótipo foi imaginado como uma versão inicial de um produto inovador, para isso, algumas funcionalidades foram removidas objetivando um enfoque na aplicação prática do conceito de ferramenta de denúncia comunitária. Uma das principais funções que foi deixada de lado, é a fila de ações em trabalho offline, ou seja, apesar de acessível sem internet o aplicativo não é capaz de cumprir todas as suas funções, como por exemplo enviar os dados para o banco de dados remoto (mostrando uma mensagem: “sem rede” ao usuário). É interessante o planejamento de uma futura extensão, em que a fila de ações sem rede seja implementada seguindo o padrão de redes sociais como o instagram e facebook, onde a conexão com internet é

monitorada constantemente e as fotografias e operações realizadas sem rede são armazenadas em um banco local do celular em formato de fila e enviadas para o servidor remoto em nuvem quando a rede é conectada.

Figura 13 - Fluxograma: funcionamento do Redux



Fonte: https://res.cloudinary.com/dyzj2erac/image/upload/c_limit/wjsojsu3qb6f1322sf5m.jpg

Ainda acerca da aplicação móvel, o controle de estado da aplicação foi inteiramente projetado fazendo uso do conjunto de instrumentos básicos da biblioteca react-native, isto é, sem nenhum framework para auxiliar a troca de informações entre as telas do sistema. Essa escolha durante o desenvolvimento se mostrou extremamente infrutífera, já que ambicionava um código mais limpo e organizado mas o que realmente ocorreu foi uma dificuldade extrema de comunicação, quando uma tela não possui conexão direta com a outra existe a obrigatoriedade de passar informações para telas intermediárias e só então após um longo caminho passar os dados para a tela que realmente irá utilizar tais informações (figura 8). Uma solução que foi estudada mas não foi colocada em produção foi o uso do framework redux para gerenciar e controlar os dados da aplicação de maneira externa, uma vez que o framework se comunica diretamente a qualquer tela do app (figura 8) ele pode agilizar a transferência de dados e melhorar a organização do código como um todo.

Relativo ao recurso de reconhecimento óptico de caracteres, é pertinente salientar que sua implementação não pode ser contemplada dentro do prazo de entrega deste projeto, mas que a atividade de detecção e transcrição de caracteres presentes em placas automobilísticas obteve resultados satisfatórios em todos os testes realizados. Ao longo dos experimentos com esta tecnologia, também foram levantados tópicos de evolução e

investigação interessantes para futuras pesquisas, como a aplicação visão computacional para extrair mais informações e substituir o formulário de denúncia completamente. Neste cenário, uma análise automatizada seria inteiramente realizada por algoritmos de aprendizados de máquina aptos a localizar a posição de objetos como: carros, meios-fios, placas e postes para então, a partir do processamento de tais informações inferir a infração representada na cena e deste modo, destituir o utilizador da responsabilidade de preencher o formulário de delação evitando erros operacionais humanos.

Como abordado na discussão anteriormente, em busca da entrega de um protótipo funcional e capaz de representar a idéia geral da plataforma, alguns artifícios não foram implementados a fundo e tirem seus escopos reduzidos. Dentro da aplicação web, o principal deles foi a robustez da API que em razão do escopo foi limitada a consumos gerais de informação completa ou filtros de data, incapaz de agrupar os dados por geolocalização, horário, ou número da infração. Para futuras pesquisas é interessante que os demais filtros estejam disponíveis e que um formulário de cadastro seja criado para que usuários de responsabilidade (como os vinculados a estruturas estatais) possam usufruir de acessos privilegiados com capacidade de remoção lógica de dados do banco (o que já é possível atualmente mas não via API) e alteração de informações ligadas as delações. Além disso, é interessante que todo o visual da aplicação web seja melhorado com estruturas de design personalizadas e coerentes com estudos de interação homem máquina.

Por fim, o último tema de discussão levantado pela pesquisa realizada está relacionado a apresentação das informações coletadas e analisadas para a comunidade civil não técnica. Como uma plataforma comunitária de queixa, é valoroso que não somente os órgãos governamentais e pessoal capacitado tenham acesso aos dados processados o que implica na própria plataforma dispor estes dados tratados e de maneira intuitiva a população. Para isso, uma concepção inicial imaginada durante o desenvolvimento foi a da criação de um dashboard interativo dentro do sistema web que ofereceria em gráficos a quantidade de queixas, quais as infrações mais cometidas e em mapas do país ou de estados quais os locais mais afetados por cada transgressão, favorecendo assim o uso da plataforma pela comunidade civil e pela mídia em geral, como jornais revistas e sites.

CONSIDERAÇÕES FINAIS

Este projeto de pesquisa, compreendendo a ciência da computação enquanto ciência que estuda técnicas e metodologias que automatizam resoluções de problemas buscou conceber uma plataforma completamente nova de fiscalização, capaz de complementar de maneira eficiente, os processos atuais de monitoramento de delitos simples, e ainda possibilitar a participação da sociedade neste processo. Para isso, foi desenvolvido uma plataforma em três partes, um aplicativo para celulares capaz de capturar fotos das infrações, um serviço remoto de banco de dados em tempo real e uma interface web de disponibilização comunitária das informações. Apesar da plataforma entregue suprir diversas carências enfrentadas pelos sistemas de fiscalização pública, para que a mesma consiga solucionar definitivamente o problema de fiscalização do estacionamento em fila dupla e outras infrações simples é necessário uma adoção dos órgãos públicos e da população para que a ferramenta seja ampliada e melhorada em cada um dos três pontos desenvolvidos.

REFERÊNCIAS

Antunes Ana. **apps nativos, híbridos ou pwa? qual o melhor para a minha solução?** 2019. Disponível em: <https://gobacklog.com/blog/nativos-hibridos-pwa/>. Acesso em: 20 set. 2019.

C. Oduor, F. Acosta and E. Makhanu, **"The adoption of mobile technology as a tool for situational crime prevention in Kenya,"** 2014 IST-Africa Conference Proceedings, Le Meridien Ile Maurice, 2014, pp. 1-7, doi: 10.1109/ISTAFRICA.2014.6880669.

Câmara Rafael . **O que você deve saber sobre o funcionamento do React Native 2018**. Disponível em: <https://medium.com/tableless/o-que-voce-deve-saber-sobre-o-funcionamento-do-react-native-7e3c610aa268> Acesso em: 14 jul. 2020.

Devin Abbott. 2019, **Fullstack React Native: Create beautiful mobile apps with JavaScript and React Native.** Independently published (January 11, 2019).

Fielding, Roy Thomas**. "Architectural styles and the design of network-based software architectures. 2000."** University of California, Irvine (2000): 162.

Loiane Groner. 2019, **Estruturas de Dados e Algoritmos com JavaScript: Escreva um Código JavaScript Complexo e Eficaz Usando a Mais Recente ECMAScript**. Novatec Editora (March, 11, 2019) Brasil.

Manifesto for Agile Software Development. http://agilemanifesto.org/ (accessed April 2009).

Mike Cohn. 2004. **User Stories Applied: For Agile Software Development**. Addison-Wesley Professional; 1 edition (March 1, 2004), USA.

Redação, Google atualiza plataforma de desenvolvimento de apps Firebase. 2018. Disponível em: <https://computerworld.com.br/2018/10/31/google-atualiza-plataforma-de-desenvolvimento-de-apps-firebase/> Acesso em: 03 Ago. 2019

Scharff, C., & Verma, R. (2010). **Scrum to support mobile application development projects in a just-in-time learning context**. Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering - CHASE ’10. doi:10.1145/1833310.1833315