UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELLIPE FRATINI STUDART SOMBRA

Não cheguei!
Um aplicativo para aumentar a segurança dos deslocamentos cotidianos

RIO DE JANEIRO
2018

FELLIPE FRATINI STUDART SOMBRA

Não cheguei!
Um aplicativo para aumentar a segurança dos deslocamentos cotidianos

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Vinícius Gusmão Pereira de Sá

RIO DE JANEIRO

2018

CIP - Catalogação na Publicação

SS693n

Sombra, Fellipe Fratini Studart
Não cheguei!: um aplicativo para aumentar a segurança dos deslocamentos cotidianos / Fellipe Fratini Studart Sombra. -- Rio de Janeiro, 2018.
51 f.

Orientador: Vinícius Gusmão Pereira de Sá.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2018.

1. Não Cheguei!. 2. Computação. 3. Android. 4. Java. 5. Aplicativo. I. Sá, Vinícius Gusmão Pereira de, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a).

Não cheguei!
Um aplicativo para aumentar a segurança dos deslocamentos cotidianos

Fellipe Fratini Studart Sombra

Projeto Final de Curso submetido ao Corpo Docente do Departamento de Ciência da Computação) do Instituto de Matemática da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação).

Aprovado por:

_____

Prof. Dr. Vinícius Gusmão Pereira de Sá (Orientador)

_____

Prof. Dra. Adriana Santarosa Vivacqua

_____

Prof. Dra. Silvana Rossetto

RIO DE JANEIRO
2018

Dedico este trabalho à minha família: meus pais, Paulo Arthur Studart de Albuquerque Sombra e Teresa Cristina Fratini Sombra, que sempre me apoiaram em tudo que fiz desde que nasci, e se esforçaram ao máximo para me dar todo o suporte necessário para estar aqui hoje; a meu irmão, Flávio Fratini Studart Sombra, que me aguentou perturbando ele durante todos esses anos. Dedico também aos meus amigos que conheci ao longo da faculdade e aos que permaneceram na minha vida desde a época do colégio.

AGRADECIMENTOS

Agradeço aos amigos e familiares que me incentivaram e apoiaram na realização de mais esta etapa de minha vida pessoal e profissional. Agradeço aos professores desta universidade, em especial Vinícius Gusmão Pereira de Sá, pela orientação, paciência e disponibilidade durante o projeto final e Juliana Vianna Valério, pelos conselhos durante toda a minha jornada acadêmica. Agradeço também à Universidade Federal do Rio de Janeiro, por me proporcionar tamanha experiência e oportunidades.

RESUMO

A insegurança que assola as cidades e preocupa as famílias faz com que exista sempre um cuidado diário com todas as pessoas que nos são caras. Em razão disso, na atualidade, as famílias buscam impor o hábito de avisar o momento de saída e chegada aos seus destinos, buscando com isso um certo controle e mais tranquilidade. Porém, esse hábito nem sempre é efetivo, pois muitas vezes problemas ocorrem durante esses trajetos. Ocorrências que nem sempre são previsíveis, como acidentes, males súbitos, sequestros, dentre outros, que impedem o aviso e a chegada ao destino, seja trabalho, momentos de lazer ou compromissos sociais. Muitas vezes durante uma ocorrência, pode-se levar muitas horas ou até mesmo dias, até perceberem que algo deu errado e começarem a agir. E essas horas iniciais são cruciais para qualquer tipo de emergência.

Uma forma de minimizar esse problema seria existir um mecanismo que avisasse nossos entes queridos quando não chegarmos em algum destino planejado. Pois assim, poderiam tomar as medidas necessárias, seja simplesmente entrar em contato perguntando se está tudo bem ou até mesmo iniciar uma busca pela pessoa, em um espaço de tempo muito mais curto.

Assim surgiu a ideia do aplicativo denominado Não Cheguei!. O aplicativo tem o papel de avisar as pessoas de confiança do usuário caso este não chegue em seu destino. Além disso, também é disponibilizado um mapa com as localizações por onde o usuário esteve até o momento do aviso ser enviado. Esse sistema tem o intuito de facilitar e principalmente agilizar as resoluções de ocorrências que impossibilitam o usuário de contatar alguém para pedir ajuda.

Palavras-chave: Não Cheguei!. UFRJ. Computação. Android. Java. Aplicativo. Segurança.

ABSTRACT

The insecurity that plagues the cities and worries the families, makes that there is always a daily care with our loved ones. Because of this, families are now trying to impose the habit of warning the moment of departure and arrival to our destinations, seeking a certain control and peace of mind. However, this habit is not always effective, because problems often occur during these journeys. Occurrences that are not always predictable, such as accidents, sudden illness, kidnappings and so on... preventing us from warning about not having arrived at the destination, be it work, moments of leisure or social commitments. Often during an occurrence, it can take many hours or even days until someone realize something has gone wrong and begin to act. And those early hours are crucial to any kind of emergency.

One way to minimize this problem would be to have a mechanism that warns our loved ones when we do not reach our destination. That way they could take the necessary steps, be it simply to get in touch asking if everything is okay or even to start a hunt for the person in a much shorter period of time.

That's how was born the idea of the application Não Cheguei! The application has the role of notifying the user's trusted people if the user does not reach the programmed destination. In addition, a map is also provided with the locations where the user has been until the warning has been sent. This system is intended to facilitate and especially expedite the resolutions of occurrences that make it impossible for the user to contact someone for help.

Keywords: Não Cheguei!. UFRJ. Computing. Android. Java. App. Safety.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MVP	Minimum Viable Product
REST	Representational State Transfer
SQL	Structred Query Language
UI	User Interface

SUMÁRIO

1 INTRODUÇÃO

A vida atual gera um alto nível de stress nas famílias, devido a diversas preocupações diárias com que temos que lidar. Com isso, existe uma necessidade maior de controle com as pessoas que nos são importantes. Esse controle normalmente realizado através de inúmeras mensagens e ligações para saber se a pessoa já está voltando para casa, se já chegou no destino ou simplesmente para saber se ela está bem.

O problema é que muitas vezes a pessoa é impossibilitada de avisar alguém que aconteceu alguma coisa com ela, devido a assaltos, sequestros, acidentes, males súbitos ou, até mesmo, estar em uma área que não possui sinal de celular. O que leva a um distanciamento de tempo muito grande até que uma ação possa ser tomada, pois até que comecem a desconfiar que algo aconteceu com a pessoa já se passaram muitas horas ou até mesmo dias. E essas horas entre a ocorrência e a tomada de alguma ação por parte dos entes queridos são as mais importantes na maioria dos casos.

1.1 Motivação

O Não Cheguei! [1], aplicativo para dispositivos móveis que utiliza o sistema Android, foi criado pensando em reduzir esse tempo entre a ocorrência do usuário e a ação dos entes queridos. Pois estamos passando a responsabilidade de avisar a ocorrência para um sistema, que fará isso independente da situação que o usuário se encontra. Com isso, esperamos que o aplicativo se torne uma ferramenta eficaz de segurança para o usuário que deseja mais tranquilidade pessoal.

1.2 A ideia

A ideia é construir um sistema que possa gerar um alerta aos contatos do usuário sempre que ele não chegue ao seu destino na hora programada, fazendo com que uma tomada de ação imediata seja realizada. A solução desenvolvida neste trabalho foi projetar um aplicativo para o sistema móvel Android, tendo uma aplicação escrita em Java para ser responsável pelo controle das viagens que estão sendo realizadas pelos usuários.

O cliente Android também foi desenvolvido usando Java, já que é a linguagem de programação nativa para este sistema. Os sistemas de animação, cores e design patterns utilizados no cliente foram recomendados pela Google e pela comunidade ativa de desenvolvedores para sistemas Android.

O servidor Java foi criado com o intuito de ser o orquestrador do sistema. Ele é o responsável por receber as informações do aplicativo e utilizá-las para gerar os alertas quando forem necessários, eliminando assim a necessidade do dispositivo móvel estar com rede durante todo o processo de uso do aplicativo móvel. O sistema foi de-

senvolvido seguindo as especificações do JAVA EE 6 [2] e foi escolhido o framework Jersey[3] para implementação da referência do JAX-RS[4].

Com intuito de facilitar ainda mais qualquer tipo de ação a ser tomada após a ocorrência, enviamos junto com o alerta um link para uma página com as últimas posições do usuário até o momento do alerta. Para isso, foi desenvolvida uma aplicação estática em HTML5 [5] que é responsável por exibir o mapa através da Google Maps APIs [6].

1.3 Trabalhos correlatos

Após pesquisas na internet e debates sobre a ideia entre conhecidos, foram encontrado três aplicativos que tem alguma semelhança com a ideia do Não Cheguei!. Todos visam a segurança pessoal e maior controle sobre pessoas que nos são caras, porém cada um tem uma abordagem diferente sobre o assunto.

1.3.1 Contatos de Confiança

Contatos de confiança[7] é um aplicativo de segurança pessoal criado pela Google que pouca gente conhece. A ideia principal do aplicativo consiste em você poder requisitar a localização de algum contato seu a qualquer momento ou então compartilhar sua localização caso seja necessário.

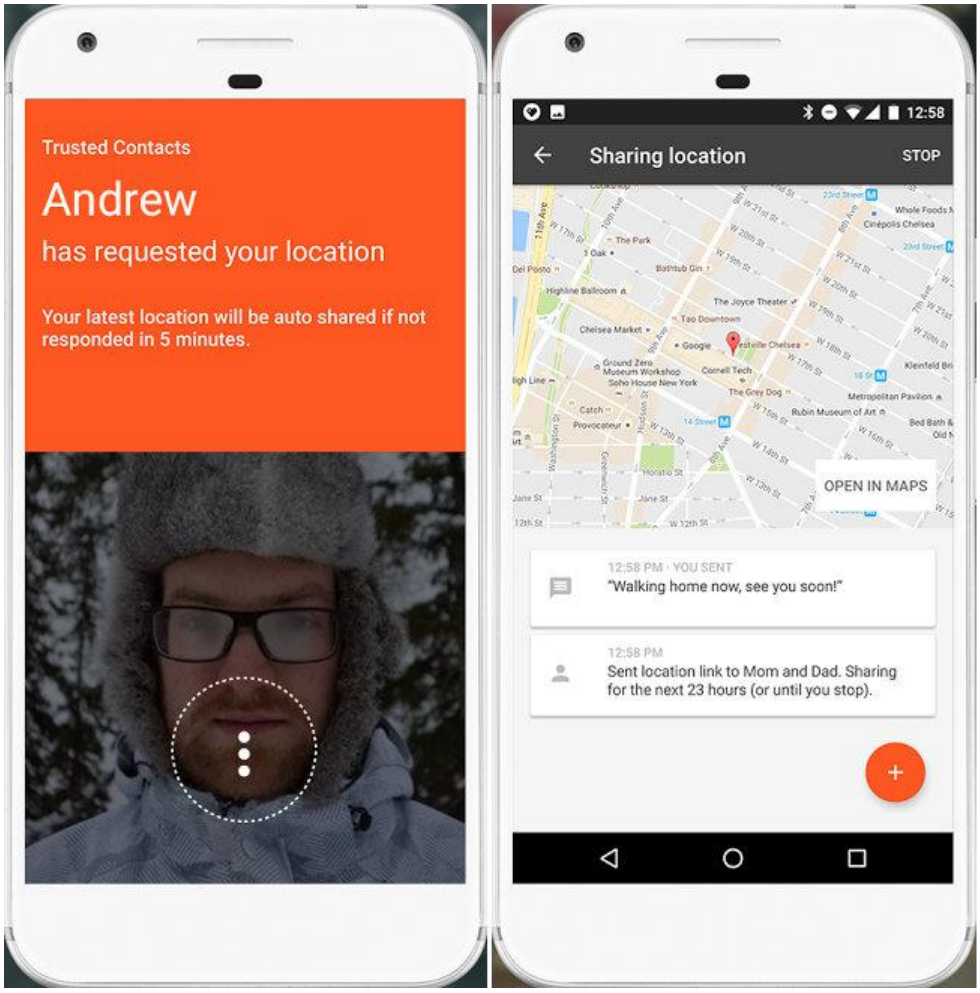

Figura 1.1: Telas de requisição e compartilhamento de localização do aplicativo Contatos de Confiança

Pontos altos:

- Os usuários podem requisitar a qualquer momento a localização de algum contato confiável;
- O usuário pode compartilhar sua localização com seus contatos caso se sinta em uma situação de perigo;
- Interface intuitiva;

- Não requer que seus contatos confiáveis tenham o aplicativo instalado também;

Pontos baixos:

- É necessário ativar a funcionalidade de histórico de localização do Google para o app funcionar da maneira correta;
- Depende da ação por parte de um usuário do aplicativo para enviar algum tipo de alerta ou requisitar alguma informação;
- Seus contatos podem podem ficar monitorando por onde você anda, caso você não rejeite a solicitação em até 5 minutos;
- Envio do alerta para os contatos que não possuem o aplicativo é através do e-mail;
- Não disponível para iOS;

O aplicativo tem grande potencial, é bem avaliado pelos usuários na Google Play[8] mantendo uma média próxima de quatro estrelas. É simples e direto ao ponto, e tem o nome da Google por trás, o que certamente o favorece. Mas o fato dele depender de alguma ação na hora do usuário para usufruir de seus benefícios é onde ele mais deixa a desejar, porque nem sempre o usuário poderá estar com o celular utilizável na hora de uma situação de perigo que esteja passando.

1.3.2 Clube da Segurança

Clube da Segurança[9] é um aplicativo da Graber[10] que possui funcionalidades para ajudar na segurança pessoal do usuário. O aplicativo não tem uma proposta objetiva e simples, já que se trata de uma composição de funcionalidades que juntas ajudam na proteção do usuário e seus bens.

Suas cinco funcionalidades são:

- Botão de emergência, que envia um alerta para seus contatos com sua localização quando acionado;

- Função armadilha, que tira uma foto e te envia por e-mail caso seu celular seja manuseado sem sua autorização;

- Cerca virtual, que avisa seus contatos caso você saia de um perímetro determinado durante um horário programado;

- Dicas, que é uma área destinada a vídeos e textos sobre segurança da TV Graber[11];

- Lembrete, que auxilia o usuário a encontrar o local onde estacionou seu veículo;

A cerca virtual e botão de emergência são as funcionalidades que são mais úteis para alguém que está buscando aumentar sua segurança através do uso de um aplicativo. As outras três funcionalidades são totalmente desnecessárias e poderiam ser facilmente descartadas.

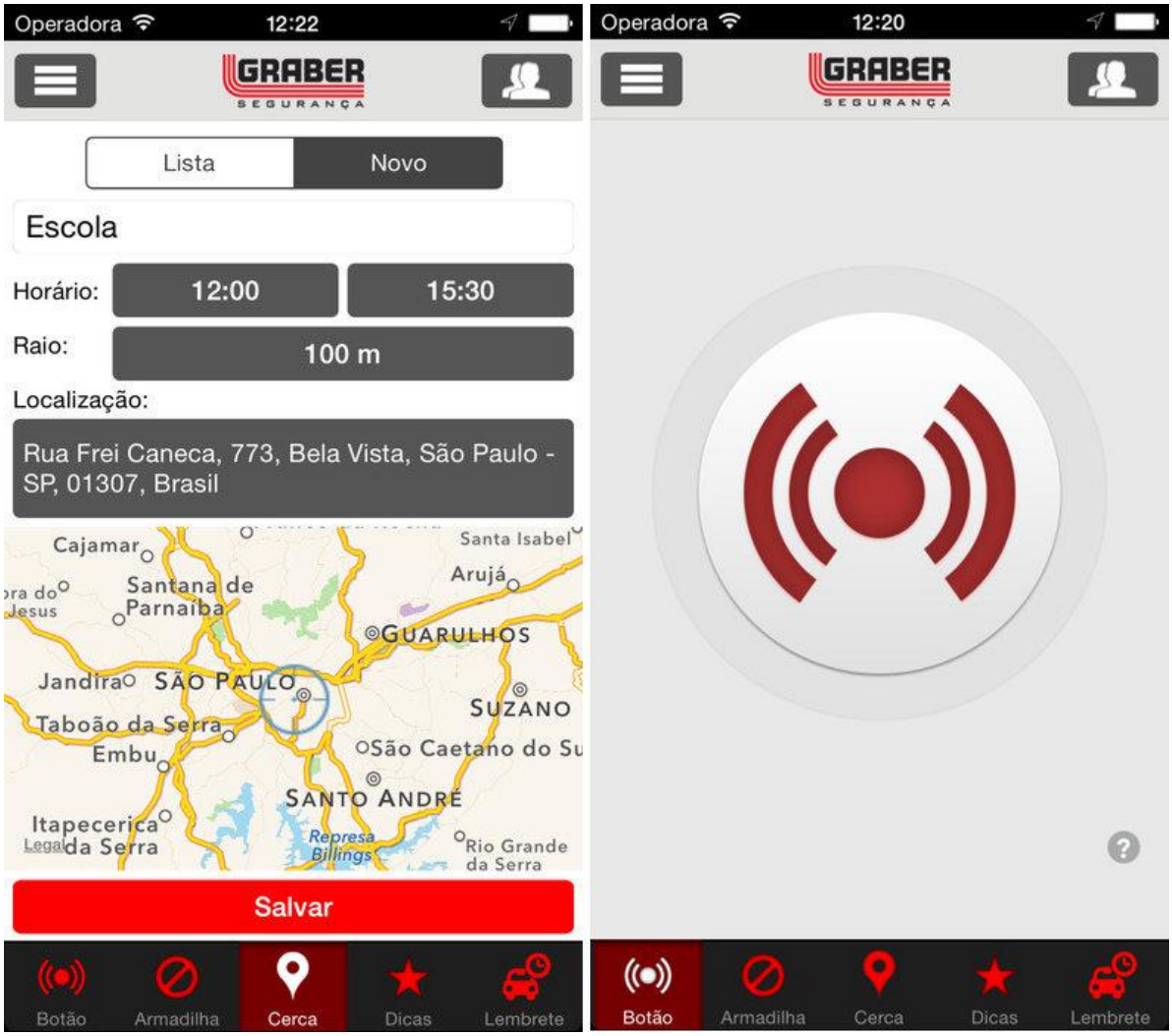

Figura 1.2: Telas de cerca virtual e botão de emergência do aplicativo Clube da Segurança

Pontos altos:

- Apertando apenas um botão você pode enviar um alerta com sua localização para seus contatos;
- Envio de alerta automático caso você saia do perímetro de um local e horário pré-determinados;

Pontos baixos:

- Interface mal feita;
- Aplicativo sem foco bem definido;
- Não disponível para Android;
- Aplicativo com diversos problemas;

Infelizmente esse aplicativo parece ter sido mal elaborado e está abandonado. A sua avaliação no iTunes[12] é de somente duas estrelas, e os comentários das pessoas que testaram o aplicativo são em grande maioria reclamando que o mesmo não cumpre sua proposta, falhando em quase todas suas funcionalidades.

Figura 1.3: Avaliaçãoes do aplicativo Clube da Segurança no iTunes

1.3.3 Localizador Familiar e Celular

Localizador Familiar e Celular[13] é um aplicativo com foco de monitoramento da sua família desenvolvido pela Life360[14]. A ideia principal dele é que o usuário possa saber onde estão todos os membros do seu grupo familiar, com uma visualização deles no mapa. Além disso, ele oferece recursos como chat entre os membros e a possibilidade de marcar lugares para saber quando alguém chegou ou saiu de um determinado lugar marcado.

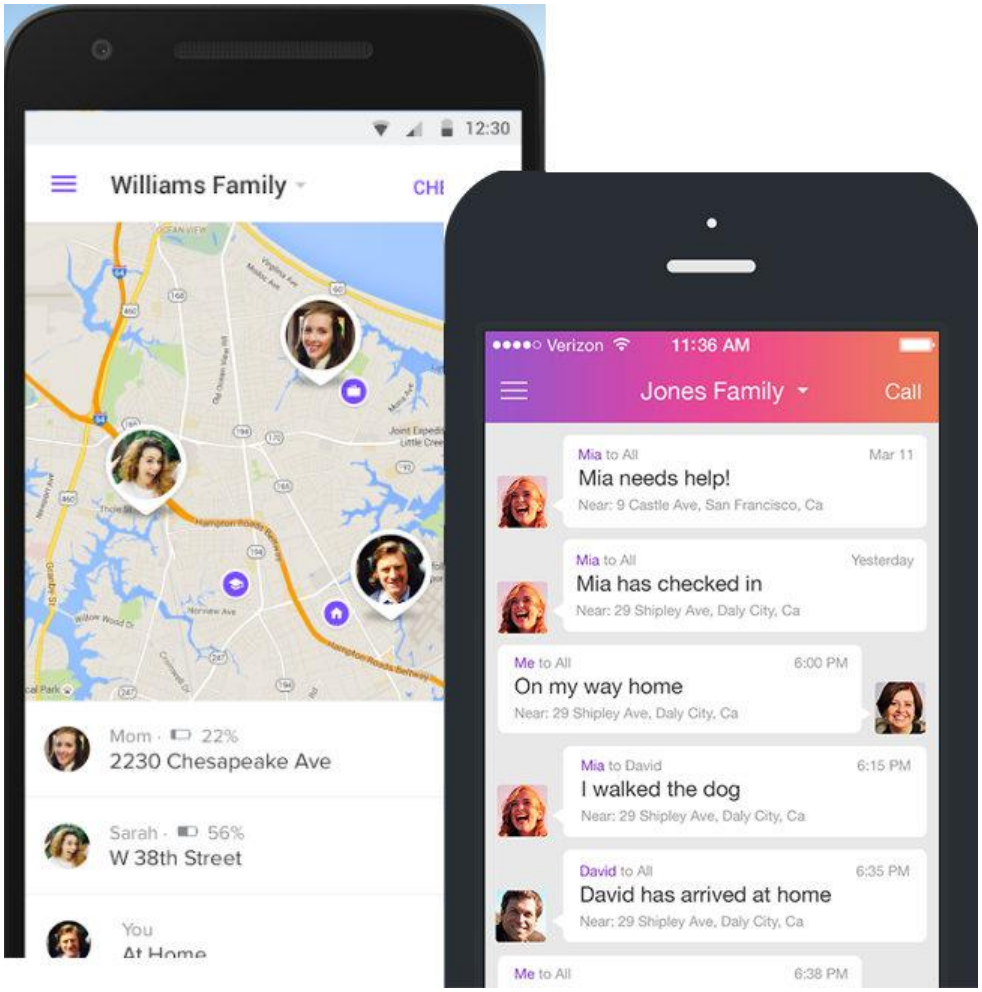

Figura 1.4: Telas do mapa de localização e chat do aplicativo Localizador Familiar e Celular

Pontos altos:

- Monitoramento pelo mapa dos membros do seu grupo;
- Criação de grupos diferentes para controle. (Ex: Um com a família e outro com os amigos);
- Possibilidade de desativar o compartilhamento de localização;
- Alertas para quando alguém do grupo chegar em um lugar marcado;

Pontos baixos:

- É limitado na versão grátis. Para usufruir sem limites precisa pagar;
- Botão pânico: Necessidade de o usuário abrir e executar a funcionalidade em uma emergência;

Apesar de diversas pessoas reclamarem da falta de precisão e da demora ao atualizar as posições das pessoas no mapa, o aplicativo cumpre bem o seu papel de monitoramento familiar. Podendo ser aquele almejado alívio para um pai ou mãe preocupados querendo saber onde o filho se encontra. Porém, o aplicativo não fornece um meio prático de avisar seus contatos em caso de necessidade, visto que o botão do pânico que ele fornece você precisa estar com o celular operacional e com rede para utilizar.

1.4 Público alvo

O aplicativo foi construído pensando nas pessoas que desejam se sentir um pouco mais seguras quando saem de casa. Logo, qualquer pessoa que saiba utilizar um smartphone é um possível público alvo.

2 O SERVIDOR

Figura 2.1: Desenho de arquitetura da aplicação

2.1 A criação

O servidor foi projetado para coletar informações do aplicativo, acompanhar as viagens que estão em andamento e fornecer informações para a aplicação web e móvel. Para que isto fosse possível, o aplicativo deveria se comunicar com o servidor através de uma interface web. A versão mais simples do servidor que poderia ser lançada com o menor tempo de desenvolvimento (Produto Viável Mínimo ou MVP, do original Minimum Viable Product) precisaria conter os pontos abaixo:

- Uma rotina para verificarmos se alguma viagem que estava em andamento já terminou.
- Serviço de envio de mensagens por parte do servidor. Para alertar os contatos do usuário.
- Uma interface web para recebermos informações das viagem que estivessem rodando no aplicativo e para que fosse possível a aplicação web requisitar informações sobre as viagens ao servidor.

2.1.1 Rotina de acompanhamento das viagens

Como não poderíamos depender da disponibilidade do celular do usuário, ou que ele tivesse com rede, ou até mesmo que o aplicativo estivesse rodando, foi decidido que toda a lógica do encerramento da viagem deveria estar no servidor. Logo, independente do aplicativo enviar o sinal indicando que a viagem terminou (sendo ela sucesso ou falha), o servidor fica verificando a cada dois minutos se alguma viagem das que estão ativas no momento já passou do tempo limite estipulado, para assim tomar as atitudes necessárias. E como ele faz essa verificação constante das

viagens? Para isso foi escolhido o Quartz Scheduler[15], um serviço de agendamento de tarefas que pode ser integrado, ou utilizado virtualmente, em qualquer aplicação Java. O motivo dele ter sido escolhido, foi devida a experiência prévia com essa biblioteca e saber que ela atenderia perfeitamente a situação.

Além do job que verifica se as viagens que estão em andamento já excederam seu tempo limite, também foi criado um job, que roda a cada 12h, para limpar os registros das viagens que terminaram há mais de três dias. A motivação da criação dessa funcionalidade foi a limitação do banco de dados. Pois como o serviço de banco que foi utilizado é grátis, ele tem um tamanho limite pequeno, o que inviabilizaria ficar acumulando registros indefinidamente. Além do mais, continuar mantendo registros de viagens antigas só serviria para extrair métricas da utilização da aplicação. Porque não faz parte do objetivo da aplicação a possibilidade de ficar vendo registros de viagens antigas.

2.1.2 Serviço de envio de mensagens

Era necessário ter algum meio para alertar os contatos confiáveis do usuário, sem que fosse necessário a utilização do aplicativo. Pois não queríamos adicionar essa restrição, de fazer com que a lista de contatos se limitasse às pessoas que possuem o aplicativo também. Ou até mesmo, evitar a situação do usuário ter que coagir a família a instalar o aplicativo.

Para isso teríamos que usar algum meio de comunicação global, utilizado em todos os lugares e pela grande maioria das pessoas. E qual seria a maneira mais fácil e rápida de entrar em contato com alguma pessoa específica ? Através do celular! O celular hoje em dia é uma ferramenta que possibilita que todos possam estar conectados com qualquer pessoa do mundo a qualquer instante. E segundo os dados

do site da GSMA[16], associação que representa as operadoras de telefones móveis, o número contas únicas nas operadoras de celular é de aproximadamente 5 bilhões, ou seja dois terços da população mundial. Sendo assim, irrefutavelmente o melhor meio para se comunicar com alguém atualmente.

Logo, foi concluído que a ferramenta de comunicação ideal para alertar os contatos seria o celular. Após pesquisas sobre soluções utilizando o mesmo, foi descoberto o Twilio[17], um serviço que disponibiliza uma API para envio de SMS global. A solução foi estudada, implementada e testada com sucesso, porém não existia maneira utilizar esse serviço em produção, pois o mesmo é pago e sua versão grátis só pode ser utilizada com um número fixo de celulares que necessitam de verificação prévia. Por isso essa solução foi inviabilizada para a situação atual desse projeto.

Com isso, outra possível solução seria através do envio de e-mail. Ela seria muito menos eficaz que a anterior, pois as pessoas consultam com menos frequência seus e-mails (mesmo podendo estarem conectadas nesse serviço pelo seu celular), porém para o MVP seria o suficiente. Devido a experiência prévia com essa solução, era conhecido que ela funcionaria sem o menor problema com nossa aplicação. Para a implementação, foi utilizada a API padrão de envio de e-mail do Java, o JavaMail[18]. Para seu uso, precisamos utilizar um provedor de correio eletrônico e ter uma conta no mesmo. Foi escolhido o Gmail, devido ao seu fácil uso e experiência com o mesmo. E foi criado e configurado a conta do naocheguei@gmail.com, para uso da aplicação. Após implementação da solução, a mesma funcionou perfeitamente e atendeu a necessidade de alertar os contatos confiáveis do usuário.

2.1.3 Interface web

Como o servidor não precisaria receber os cabeçalhos HTTP ou quaisquer outras abstrações adicionais dos protocolos baseados em padrões de trocas de mensagem, optamos em construir o sistema baseados nos princípios REST[19]. Como o aplicativo e a aplicação web não precisam conhecer a lógica por trás do servidor, foi criado uma interface de programação de aplicação (Application Programming Interface, API), que, em geral, é um conjunto de padrões e rotinas estabelecidos por um determinado software para que outros softwares possam utilizar suas funcionalidades, desconhecendo seu funcionamento. Com a criação de uma API REST, o aplicativo poderia enviar informações para o servidor e a aplicação web realizar consultas sobre as viagens, de maneira transparente para o usuário.

Foram desenvolvidos os seguintes serviços na nossa API:

Serviços de status do servidor:

- check -> Serviço utilizado para verificar se o servidor está ativo no momento. Necessário realizar essa verificação ao abrir o aplicativo, pois o mesmo não iria funcionar corretamente caso o servidor esteja indisponível;

Serviços de usuários:

- register -> Serviço utilizado para o cadastro da conta do usuário na aplicação. Necessário para identificar qual usuário está utilizando o aplicativo;
- login -> Serviço utilizado para validar a entrada de um usuário na aplicação. Retornando suas informações para serem utilizadas pelo aplicativo;

Serviços de contatos:

- register -> Serviço utilizado para o cadastro de contatos do usuário;
- getAllUserContacts -> Serviço utilizado para retornar todos os contatos de um usuário;
- delete -> Serviço utilizado para remover um contato da lista de contatos do usuário;

Serviços de viagem:

- getPathInfomration -> Serviço utilizado para retornar todos os registros do caminho realizado na viagem requisitada. Necessário para o uso da aplicação Web, para desenhar o caminho percorrido pelo usuário no mapa;
- startTrip -> Serviço utilizado para iniciar uma viagem de um determinado usuário. Guardando todas as informações iniciais definidas da viagem;
- endTrip -> Serviço utilizado para alterar o estado de uma viagem para "terminado" no banco. Necessário para parar de verificar a mesma, já que ela já foi concluída;
- delayTrip -> Serviço utilizado para adicionar mais tempo para a viagem. Necessário caso haja algum imprevisto, ou o usuário queira aumentar o tempo estimado inicialmente;
- getTrip -> Serviço utilizado para retornar as informações da viagem ativa do usuário. Necessário pelo aplicativo para verificar ao iniciar o aplicativo se o usuário já está com uma viagem em andamento, e poder continuar acompanhando a mesma;
- savePosition -> Serviço utilizado para salvar a posição atual do usuário periodicamente durante a viagem que está em andamento. Necessário para gerar o mapa com o registro de todas as posições por onde o usuário passou durante a viagem;

2.2 A infraestrutura

Em produção, o servidor está em funcionamento em um PaaS: Heroku [20], um serviço grátis, mas limitado, que permite que aplicações de diversas linguagens sejam executadas no container escolhido. O banco de dados também foi criado através de um serviço fornecido pelo Heroku.

A escolha desse serviço se fez necessária pois era um requisito ter um servidor com alta disponibilidade para hospedar a API de serviços e o banco de dados da aplicação.

2.2.1 Heroku

O Heroku se enquadra na categoria de serviços da computação em nuvem conhecida como Plataforma como Serviço (Platform as a Service, ou PaaS), no qual o fornecedor entrega para o cliente um ambiente pronto para receber a aplicação. Este tipo de solução abstrai o desenvolvedor dos detalhes de infraestrutura ao disponibilizar containers para instalação das aplicações, facilitando a manutenção, extensão e escalabilidade, além de oferecer maior agilidade para disponibilizar uma aplicação na web.

No Heroku, esses ambientes de execução que são fornecidos, são denomidados "dynos". As aplicações criadas para rodar no Heroku podem ser compostas por vários dynos. De acordo com a tarefa que realizam, os dynos podem ser classificados em três tipos:

- Web Dynos: Responsáveis por receber e atender as requisições web. Cada aplicação pode ter no máximo um dyno desse tipo;

- Worker Dynos: Executam tarefas em background, sendo recomendado para processamentos mais pesados. Cada aplicação pode ter vários dynos do tipo worker;

- One-Off Dynos: São criados apenas para uma tarefa eventual e logo são destruídos. Um exemplo clássico é a leitura de logs da aplicação.

Estamos rodando duas aplicações no Heroku, a API de serviços e a aplicação web, cada uma dentro de um Web Dyno. Dentro do Web Dyno que está rodando nossa API de serviços, foi contratado o plano grátis do JawsDB[21], um Banco de Dados como Serviço (Database as a Service, ou DaaS), como solução para hospedar nosso banco de dados. Porém, como é um plano grátis, nosso banco só pode possuir no máximo 5Mb de dados, que foi a motivação da criação do job no Quartz Scheduler para limpar os registros antigos.

Essa integração do Heroku com outros serviços, facilita a vida do desenvolvedor a ponto dele somente ter que se preocupar em desenvolver a aplicação, pois toda parte de configuração de infraestrutura é fornecido de forma fácil e rápida.

2.2.2 Banco de Dados

O banco de dados foi desenvolvido utilizando a linguagem SQL[22], e o sistema de gerenciamento de banco de dados (SGBD) escolhido foi o MySQL[23], pela sua simplicidade de uso e experiência prévia.

A modelagem do banco do NãoCheguei! foi evoluindo conforme o desenvolvimento da aplicação era realizado e era percebido novas necessidades, até o término do desenvolvimento da mesma, tendo como resultado esse diagrama.

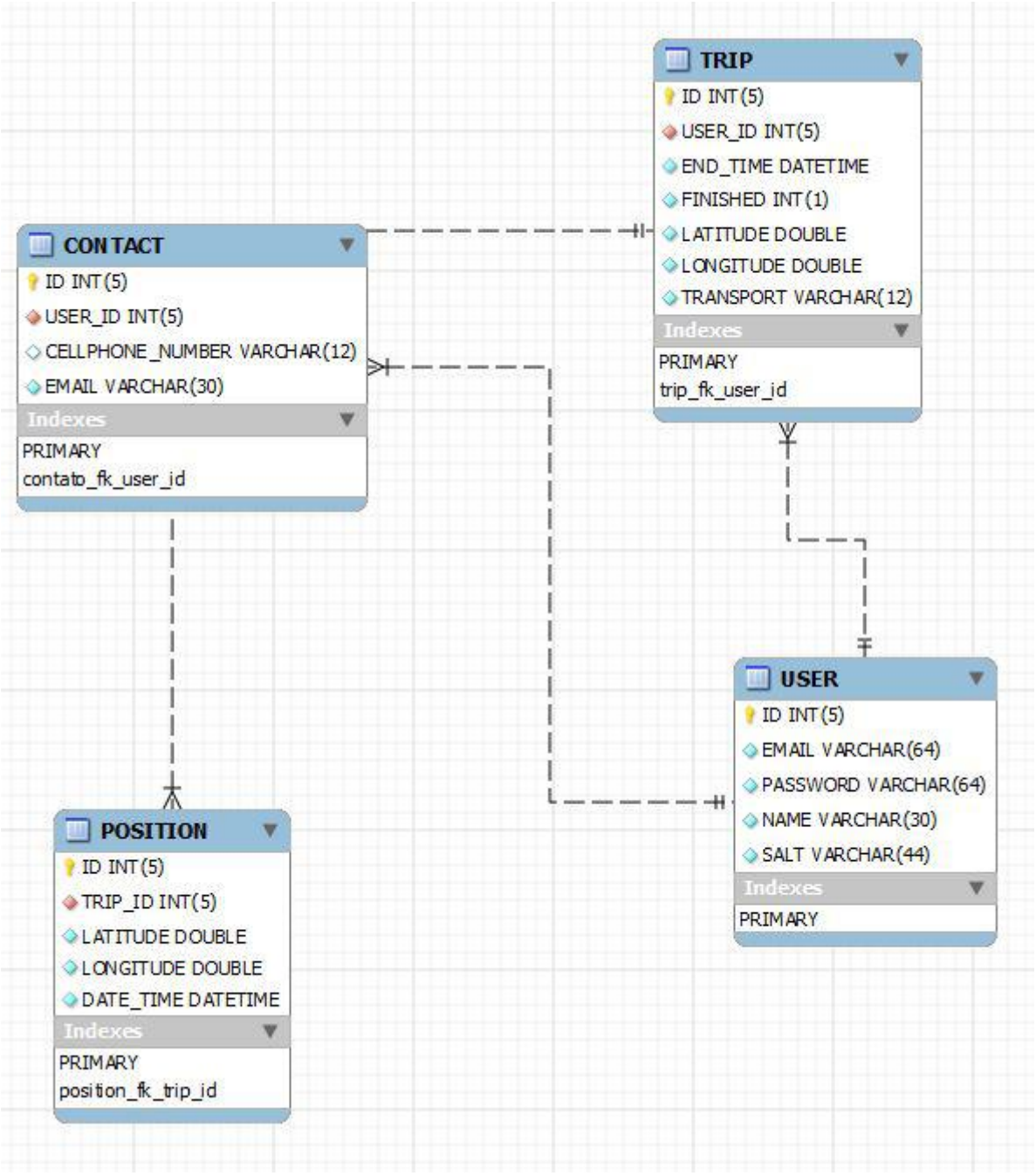

Figura 2.2: Diagrama Entidade Relacionamento

3 O APLICATIVO

3.1 A criação

Na criação do aplicativo, foi necessário tomar algumas decisões: Android ou iOS? Nativo ou ferramentas Multi-Plataforma, como Xamarim [24], Adobe Air [25] ou PhoneGap [26]? Elementos do framework nativo de interface do usuário ou elementos de frameworks customizados?

Após pesquisas sobre soluções multiplataformas, foi constatado o problema de performance e adaptação do código que estas ferramentas produziam a cada atualização de sistema, seja Android ou iOS. Além disso, como seriam usadas funcionalidades sensíveis do sistema, como o GPS, já teríamos acesso à todas as interfaces, sem o uso de bibliotecas de terceiros, que poderiam ter problemas de implementação ou nem oferecer suporte oficial. Por estes motivos, foi escolhido a produção de código nativo.

Agora era necessário decidir se o aplicativo seria desenvolvido para Android, iOS ou ambos. Como se trata do início de uma ideia, não há motivos para gastar tanta energia desenvolvendo o aplicativo nas duas plataformas. Devido ao maior uso e a

disponibilidade de aparelhos para testes, foi escolhido produzir um aplicativo nativo Android, com a mesma interface e padrões de projeto utilizados pelos aplicativos padrões do Google. Usar os mesmos elementos do framework nativo de interface de usuário, sem customizações, traria ganhos por conta de dois motivos: A facilidade de implementação e similaridade da aplicação com os aplicativos previamente instalados no smartphone. Com isso, a experiência do usuário seria mais rica, pois ele já estaria familiarizado com os estilos e temas impostos pela fabricante dos smartphones.

3.2 Funcionalidades

Para que o aplicativo pudesse cumprir com seu papel e assim ser publicado na Google Play, teria que ser implementado, no mínimo, estas funcionalidades:

- Cadastro e Login dos usuários, para podermos ter o registro da viagem e saber quem são seus contatos confiáveis;
- Integração com o Google Maps, para que possamos obter a posição de destino que o usuário almeja chegar, além de poder dar uma estimativa de tempo até a mesma;
- Possibilidade do usuário adicionar ou remover contatos confiáveis;

3.2.1 Cadastro e Login

Com a ideia de tentar fazer o aplicativo o mais prático possível, no início do desenvolvimento, foi descartada a ideia de existir as funcionalidades de cadastro e login. Pois, era sabido a possibilidade de obter um indentificador único do celular, e através dele seria possível saber quem é o usuário que está utilizando o aplicativo.

Porém, apesar de tornar mais prático o uso do aplicativo, essa abordagem possuía alguns pontos negativos. Não seria possível informar no alerta enviado aos contatos nenhuma informação que identificasse o usuário que está em uma possível situação de perigo, logo, os contatos não conseguiriam saber a quem o aviso se referenciaria. Essa foi a principal motivação que levou a implementação do cadastro. Fora isso, também havia o fato do da perda de todas as informações de contato caso o usuário mudasse de celular.

Após definido a utilização de um sistema de cadastro e login, foi desenvolvido o cadastro com os seguintes campos: e-mail, senha e nome. Sendo o "e-mail" e a "senha" os campos necessários para realizar o login, e o "nome" somente para podermos utilizar nos avisos como forma de identificar a pessoa para seus contatos.

Quando o usuário se loga pela primeira vez no aplicativo, é salvo seu identificador único nas Preferências de Usuário (ou SharedPreferences) para evitar ter que inserir suas informações na próxima vez que for aberto o aplicativo. As SharedPreferences consiste em uma interface que permite acessar e modificar dados de preferência de usuário. O valor armazenado apresenta-se sob formato chave-valor ou key-value, ou seja, cada preferência armazenada possui uma identificação ou chave e associada a ela está um valor. Foi criado uma chave chamada "userId" e o valor dela é o identificador do usuário que se logou no aplicativo. Assim, toda vez que é aberto o aplicativo, é possível obter todas as informações do usuário que estão no servidor sem a necessidade dele preencher nada novamente.

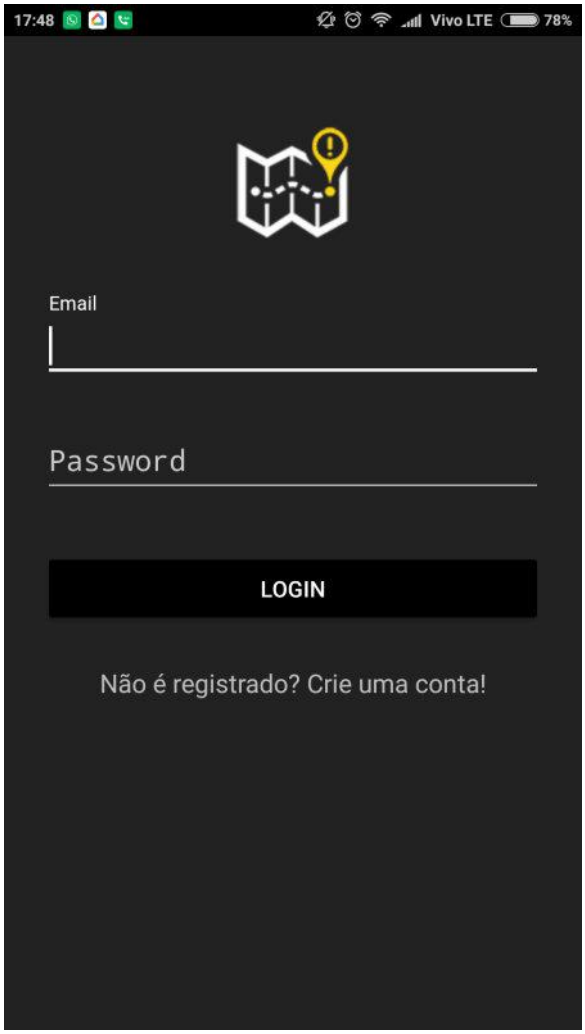

Figura 3.1: Tela de login do aplicativo

3.2.2 Integração com Google Maps

Desde a consolidação da ideia do NãoCheguei!, sua interface principal sempre foi imaginada preenchida com um mapa, que o usuário pudesse interagir para fazer suas configurações de viagem. Inspirado em aplicativos como Uber e Google Maps, era sabido que seria necessário a utilização das APIs de mapa do Google para que

o aplicativo pudesse ter vida.

Três funcionalidades eram essenciais para um bom funcionamento da ideia:

- Um mapa, para o usuário poder ter a visualização do local escolhido, tornando assim mais fácil a escolha do destino;

- A busca dos lugares no mapa. Para que fosse possível a busca do destino pelo nome;

- O tempo médio que levaria para chegar ao destino escolhido, baseado no tipo de transporte que fosse escolhido;

Era sabido que todas essas funcionalidades estariam disponíveis, pois as mesmas estão presentes no aplicativo Google Maps, e é de conhecimento público que suas APIs são abertas para consumo.

Com a Google Maps Android API, você pode adicionar mapas baseados em dados do Google Maps ao aplicativo. A API processa automaticamente o acesso aos servidores do Google Maps, o download de dados, a exibição de mapas e a resposta a gestos de mapa. Também é possível usar chamadas a APIs para adicionar marcadores, polígonos e sobreposições a um mapa básico e para alterar a visualização de uma determinada área de mapa pelo usuário. Através dela, foi obtido o resultado da primeira funcionalidade relacionada a mapa que era necessária. Foi implementado um mapa que o usuário pode interagir e instalar o marcador no mapa para seu destino.

Figura 3.2: Tela principal do aplicativo com uma marcação de destino no mapa

Com a API do Google Places[27], foi possível adicionar um campo de busca em que o usuário pode digitar o local que ele deseja encontrar e o Google retorna a lista de todos os locais que tem cadastrado e sua base relacionados com sua pesquisa, o mesmo mecanismo de pesquisa que utilizamos para encontrar um lugar no Google Maps. Para isso funcionar, foram implementadas 2 funcionalidades do Google Places:

- Place Searches: retorna uma lista de locais com base na localização ou no texto de pesquisa de um usuário;
- Place Autocomplete: pode ser usada para preencher automaticamente o nome e/ou o endereço de um local à medida que o usuário digita;

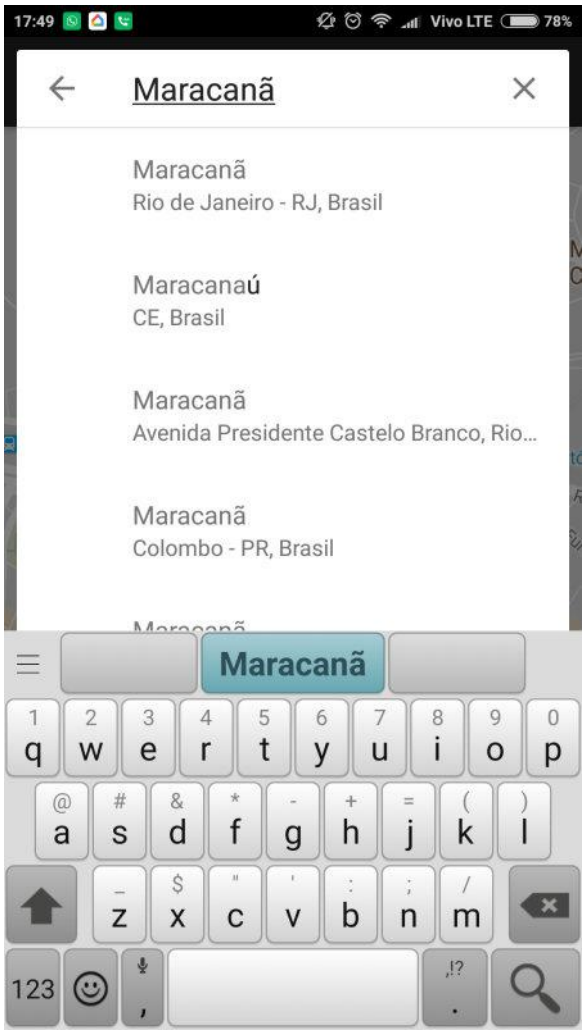

Figura 3.3: Funcionalidade de pesquisa de lugares

Para exibir o tempo estimado do trajeto do usuário foi utilizado a Google Maps Distance Matrix API[28], que é um serviço que fornece a distância e o tempo de

percurso para uma matriz de origens e destinos, de acordo com a rota recomendada entre os pontos de partida e chegada. Quando o usuário já tem escolhido seu destino e o meio de transporte desejado, são passadas as três informações (localização atual, destino e meio de transporte) para a API, e assim é obtido o tempo em milissegundos que é convertido para um formato amigável e exibido na tela para o usuário.

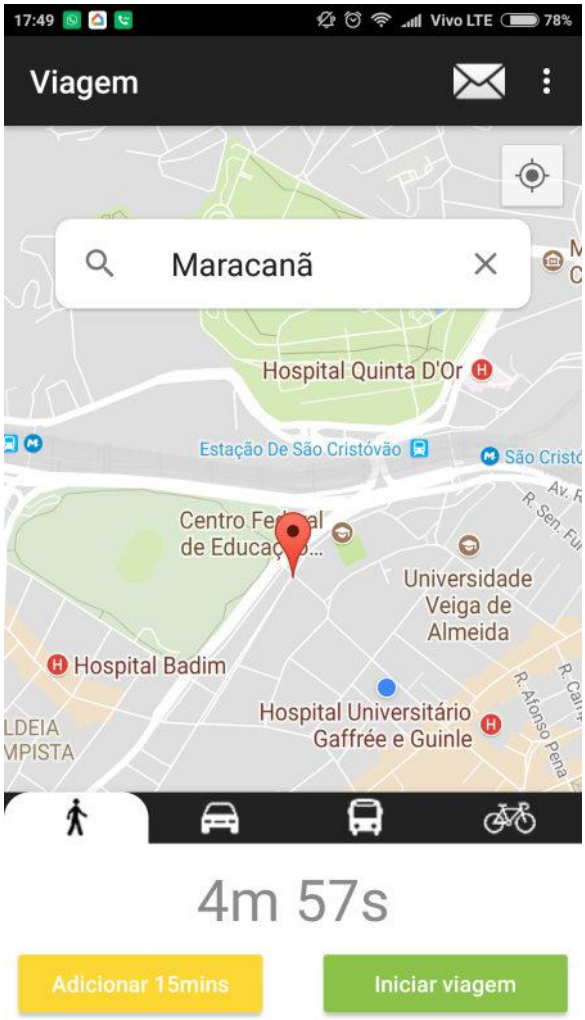

Figura 3.4: Tela com o tempo estimado ao destino selecionado

Foi considerado a utilização da Directions API[29], que indicaria o trajeto até o

destino mo mapa, porém sua implementação não foi concebida pois o foco do aplicativo não é direcionar o usuário até seu destino. Em sua ideia, não é previsto que o usuário fique com o aplicativo aberto na tela principal de seu celular. Sua ideia é somente que ele inicie a viagem e deixe o aplicativo rodando em segundo plano, dando liberdade para o usuário fazer o que desejar e se mantendo protegido. Outro ponto é que já existem aplicativos mundialmente conhecidos e bem consolidados para fornecer direções para algum lugar, como o próprio Google Maps e o Waze, então não havia nenhuma vantagem em projetar o trajeto para o usuário no mapa.

Após todas essas considerações relacionadas ao mapa e ao início da viagem, a parte final do uso seria relacionado ao término de uma viagem. Existem três formas de uma viagem iniciada chegar ao fim: se o usuário decidir encerrar a viagem por conta própria, apertando no botão para finalizar a viagem, se ele chegar no destino selecionado ou se o tempo estimado chegar ao fim antes de um dos dois casos anteriores serem acionados.

Caso o usuário decida cancelar a viagem ou simplesmente considerar que chegou no destino, existe um botão para ele finalizar a viagem atual. Após finalizada, o aplicativo volta ao seu estado normal, recomeçando o ciclo de sua utilização. Para saber se o usuário chegou no destino, a cada vinte segundo é verificado baseado na localização atual do usuário se o mesmo está dentro de um raio de cinquenta metros do destino, caso seja verdade a viagem é terminada automaticamente como sucesso. Por último, se a localização do usuário ainda não estiver dentro do raio estabelecido de seu destino quando o tempo se esgotar, a viagem será finalizada sem sucesso, fazendo com que o servidor durante suas checagens automáticas perceba sua falha e dispare os e-mails para os contatos desse usuário.

E é importante mencionar que caso haja algum imprevisto ou o tempo não seja considerado adequado, também há um botão para adicionar mais quinze minutos

ao tempo atual estimado. Esse botão pode ser utilizado antes mesmo de começar a viagem ou durante a mesma. O tempo de quinze minutos foi um valor escolhido arbitrariamente pois para facilitar o uso, era ideal que com somente um clique o usuário já conseguisse aumentar o tempo, não sendo necessário ficar perdendo tempo escolhendo um tempo específico que ele deseja adicionar. Porém, como a ideia é que o aplicativo ficasse em segundo plano, não havia como o usuário saber que seu tempo estava acabando, para assim poder adicionar mais tempo. Com isso foi implementado uma notificação que aparece no celular do usuário quando faltam 5 minutos para o término de sua viagem. Assim, ele pode entrar no aplicativo e adicionar mais tempo caso perceba que não conseguirá chegar no destino dentro do tempo restante, evitando um envio de alerta desnecessário.

Figura 3.5: Notificação para avisar que está acabando o tempo da viagem

3.2.3 Lista de contatos confiáveis

A ideia original do aplicativo era que ele fosse somente uma tela em que pudesse ser feito a gerencia de uma viagem, sendo o mais prático possível, evitando assim ficar muito complexo para o utilizador. Mas não era possível atingir esse objetivo devido as necessidades que foram surgindo, uma delas era ter um espaço que possibilitasse

a gerência dos contatos confiáveis.

Nesse espaço teria que ser possível visualizar, adicionar e remover os contatos confiáveis do usuário. Para inseri-los, no inicio, foi pensado em adicioná-los a partir dos contatos do próprio celular, visto que já teríamos acesso aos números dos mesmos. Porém essa ideia foi deixada de lado quando a decisão de usar e-mail para esse projeto foi tomada. Como só seria necessário o e-mail do contato, e como na maioria das vezes essa informação não está vinculada ao registro do contato nos celulares, foi decidido deixar o usuário inserir manualmente o e-mail nessa tela de gerência de contatos. Assim como poderia visualizar os e-mails inseridos e excluí-los facilmente também.

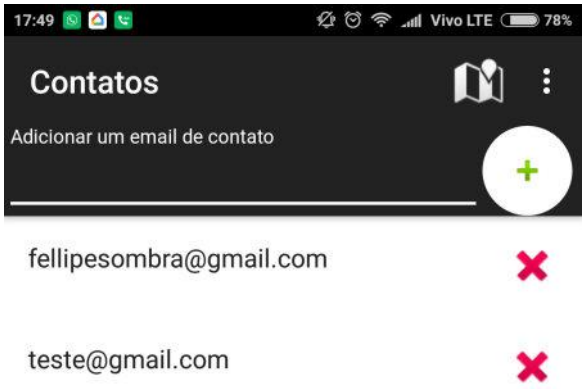

Figura 3.6: Tela de gerência dos contatos confiáveis

4 DESENVOLVIMENTO

4.1 Organização das tarefas

Para que as tarefas pudessem ser organizadas e priorizadas, optamos pelo fluxo de trabalho utilizando um método ágil: Kanban.

O Kanban, desenvolvido por um executivo da Toyota, é um modelo de organização de tarefas. É feito um quadro com raias, e cada raia representa um estado possível da tarefa: A Fazer, Em Desenvolvimento, Pronto, dentre outros. O quadro é customizável de acordo com o funcionamento linha de produção.

A ferramenta que tornou isto possível foi o Trello [30] (ver Figura 4.1). Com o sistemas de tarefas organizado e, principalmente, visível em todo o processo de desenvolvimento, foi possível perceber quais tarefas deveriam ter um destaque maior do que outras.

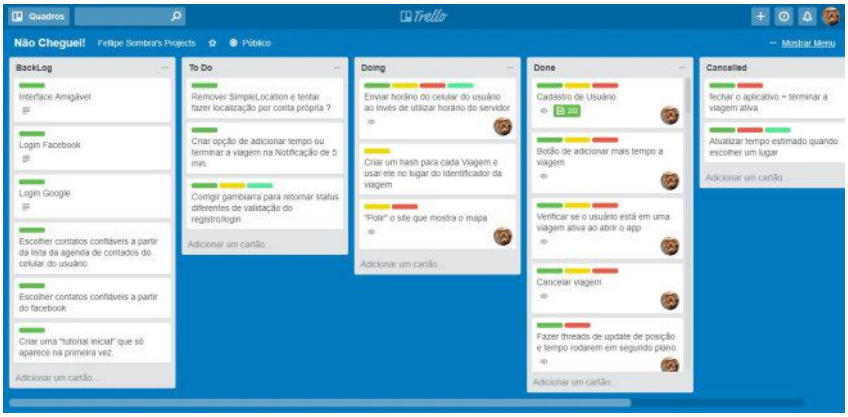

Figura 4.1: Quadro kanban no Trello

Foi escolhido trabalhar da seguinte maneira: Foram criados etiquetas coloridas para que o contexto de uma tarefa pudesse ser facilmente entendido (Como, por exemplo: FIX, App, MVP, etc). O responsável pela tarefa deveria movê-la para a coluna durante o processo de desenvolvimento, e atualizar o membro da tarefa com o seu avatar cadastrado no Trello. O processo de desenvolvimento foi divertido, pois era nítida a sensação de progresso a cada mudança de coluna. Abaixo, um exemplo mais claro de uma tarefa (Figura 4.2):

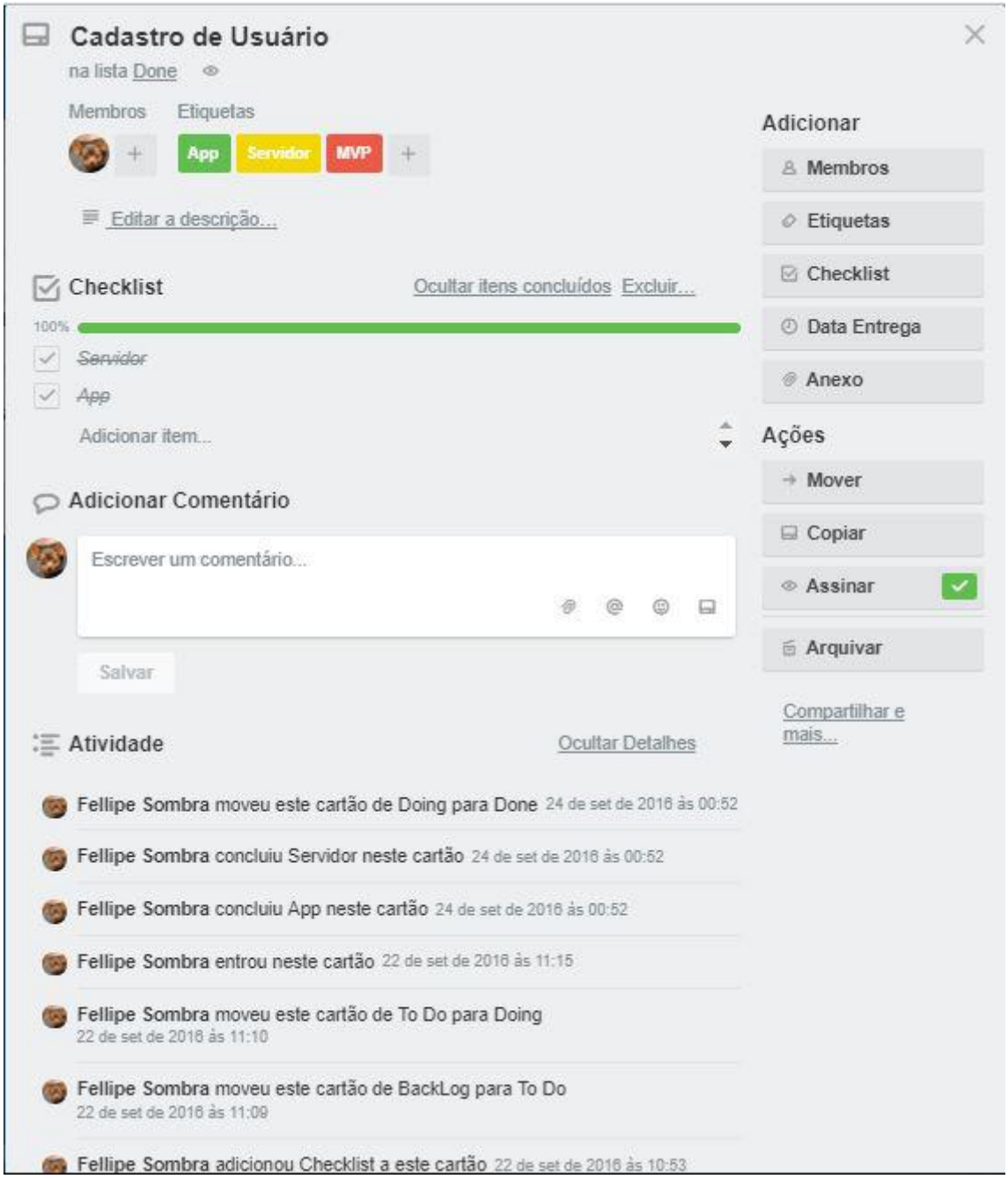

Figura 4.2: Tarefa no quadro do Trello.

4.2 Licença, versionamento e distribuição

O código possui licença GNU General Public License v3.0 [31]. A GNU GPL é a licença de software livre mais utilizada e tem uma forte exigência de copyleft, isto é, ao distribuir trabalhos derivados, o código-fonte do trabalho deve ser disponibilizado sob a mesma licença. Esta licença foi escolhida porque nos importamos com o desenvolvimento colaborativo, e gostaríamos de ver nossas soluções aplicadas a outras cidades ou estados.

O código foi versionado usando o Git[32], que é um sistema livre e de código aberto distribuído de controle de versão projetado para lidar com projetos pequenos e grandes com rapidez e eficiência. Para os repositórios remotos, utilizamos o GitHub [33], um serviço com planos pagos e gratuitos que fornece repositórios para projetos versionados usando o sistema Git. Os códigos, tanto do servidor quanto do aplicativo, se encontram disponíveis através dos repositórios encontrados em `https://github.com/fellipesombra/`.

O aplicativo está sendo distribuído de forma gratuita pela loja online Google Play, da Google. Ele pode ser instalado a partir deste link: `https://play.google.com/store/apps/details?id=br.com.fellipe.naocheguei`.

5 CONCLUSÃO

5.1 O aprendizado

Este trabalho proporcionou um enorme aprendizado: Jamais havia desenvolvido uma aplicação em Android nativo. Apesar de já possuir experiência com a linguagem Java, foi necessário um estudo prévio para entender o ciclo de vida de uma aplicação no sistema móvel, boas práticas e padrões de projeto.

Entender todo o fluxo de desenvolvimento, e participar ativamente de todo o processo de planejamento e tomada de decisão do aplicativo foi um grande aprendizado na área de gerência de projeto. Priorizar tarefas e invalidar funcionalidades foram práticas regularmente utilizadas durante o processo de desenvolvimento para que o produto final tivesse a melhor qualidade possível.

5.2 Dificuldades encontradas

A dificuldade mais marcante foi o fato do projeto todo ter sido desenvolvido por uma pessoa. A idealização da ideia, criação do banco de dados, aplicativo móvel, página

web, api de serviços e configuração do servidor são trabalhos para diversos times. Mas nesse trabalho uma pessoa foi responsável pelo trabalho de diversas áreas de conhecimento.

Outra dificuldade foi a falta de experiência prévia na programação de aplicativos móveis, visto que foi algo que eu nunca havia feito. O design do mesmo foi algo complicado também, visto que requer um conhecimento muito específico. E até o aplicativo ficar com uma aparência aceitável levou um certo tempo e número de tentativas.

5.3 Os próximos passos

Caso seja realizado um investimento no aplicativo, o primeiro e principal passo a ser tomado seria mudar a forma de alerta para envio de SMS, que era algo planejado desde a concepção da ideia.

Fora isso algumas funcionalidades extras como integração com a lista de contatos do celular do usuário, possibilidade de login com Facebook e Google eram desejados também para a versão final do aplicativo.

REFERÊNCIAS

[1] Página do aplicativo Não Cheguei! na Play Store. https://play.google.com/store/apps/details?id=br.com.fellipe.naocheguei.

[2] JAVA EE 6. http://www.oracle.com/technetwork/java/javaee/tech/javaee6technologies-1955512.html.

[3] Jersey. https://jersey.github.io/.

[4] JAX-RS. https://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services.

[5] HTML5. https://pt.wikipedia.org/wiki/HTML5.

[6] Google Maps APIs. https://developers.google.com/maps/?hl=pt-br.

[7] Aplicativo do Contatos de Confiança. https://play.google.com/store/apps/details?id=com.google.android.apps.emergencyassist&hl=pt-br.

[8] Google Play Store. https://play.google.com/store.

[9] Aplicativo do Clube da Segurança. https://itunes.apple.com/br/app/clube-da-seguran%C3%A7a/id722853756?mt=8.

[10] Graber. https://www.graber.com.br/.

[11] TV Graber. http://www.graber.com.br/seguranca/TVGraber.html.

[12] iTunes. https://www.apple.com/br/itunes/.

[13] Aplicativo Localizador Familiar e Celular. https://play.google.com/store/apps/details?id=com.life360.android.safetymapd.

[14] Life360. https://www.life360.com/.

[15] Quartz Scheduler. http://www.quartz-scheduler.org/.

[16] GSMA. https://www.gsma.com/.

[17] Twilio. https://www.twilio.com/.

[18] JavaMail. http://www.oracle.com/technetwork/java/javamail/index.html.

[19] Web Services REST. https://pt.wikipedia.org/wiki/REST.

[20] Heroku. https://www.heroku.com/.

[21] JawsDB. https://www.jawsdb.com/.

[22] SQL. https://www.w3schools.com/sql/.

[23] MySQL. https://www.mysql.com/.

[24] Xamarin, . https://www.xamarin.com/.

[25] Adobe Air. https://get.adobe.com/br/air/.

[26] Adobe PhoneGap. http://phonegap.com/.

[27] Google Places API. https://developers.google.com/places/web-service/?hl=pt-br.

[28] Google Maps Matrix API. https://developers.google.com/maps/documentation/distance-matrix/intro.

[29] Google Maps Directions API. https://developers.google.com/maps/documentation/directions/?hl=pt-br.

[30] Trello Boards. https://trello.com/.

[31] GNU General Public License v3.0. http://www.gnu.org/licenses/gpl-3.0.en.html.

[32] Git Version Control System. https://git-scm.com/.

[33] GitHub Free Git Repositories. https://github.com/.