UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

BRUNO GIOVENARDI ESTEVE

App Cocar: Um aplicativo mobile para gerência de coletivos sociais

Orientadora: Profa. Dra. Renata Galante
Co-orientador: Prof. Dr. Luiz Fernando Bilibio

Porto Alegre
2020

AGRADECIMENTOS

Agradeço a minha família pelo apoio incondicional durante todo o percurso de minha edução no ensino superior, e aos meus amigos por me acompanharem e me ajudarem a sempre enxergar o lado mais leve das coisas.

RESUMO

Este trabalho trata do desenvolvimento de um aplicativo para dispositivos móveis para administração de cadastros de famílias por movimentos comunitários. O objetivo do aplicativo é prover uma maneira mais prática e unificada de criar e consultar informações utilizando o aparelho celular e a internet, substituindo processos que dependiam de caneta e papel e não viabilizavam centralização de dados. A necessidade desta melhoria, embora já existente, foi agravada com a chegada do COVID-19 em Porto Alegre, que tornou o papel deste tipo de organização ainda mais importante. Este trabalho propõe e implementa um aplicativo mobile chamado App Cocar, que auxilia no gerenciamento dos dados destas famílias e facilita o trabalho da organização. A aplicação é capaz de oferecer funcionalidades de cadastro, edição e consulta dos cadastros das famílias, bem como a visualização destas informações de maneira facilitada. Experimentos de usabilidade com usuários reais do aplicativo Cocar e com participantes voluntários mostraram que, de maneira geral, o aplicativo é capaz de suprir as necessidades citadas de maneira prática e acessível.

Palavras-chave: Coletivo social. Aplicativo móvel. União de Vilas.

Cocar App: Mobile aplication for social movements' management

ABSTRACT

This work is about the development of a mobile devices' application for management of families by social movements organizations. The objective of the tool is to provide a more practical and unified way of creating and searching for information using a smartphone device and the Internet, replacing processes that depended on paper and did not make data centralization possible. The need for this enhancement, although long existing, was made worse by the arrival of the COVID-19 in Porto Alegre, which made the social importance of such organization even bigger. This work proposes and implements a mobile application called App Cocar, which helps in managing the data of these families and facilitates the work of the organization. The application must be able to offer functionalities of registration, edition and searching of the families registers, as well as visualization of those information in a simplified way. Usability experiments with actual users of the application and with volunteers participants showed that, in general, the application is capable of providing for the aforementioned needs in a practical and accessible way.

Keywords:

. Social organization, Mobile Application, "União de Vilas".

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
REST	Representational State Transfer
PHP	Hypertext Preprocessor
PDO	PHP Data Objects
JS	JavaScript
SQL	Structured Query Language
SGBD	Sistema de Gerenciamento de Banco de Dados
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
CPF	Cadastro de Pessoa Física
GPS	Global Positioning System
XP	Extreme Programming
DSDM	Dynamic Systems Development Method
FDD	Feature Driven Development
SUS	System Usability Scale

SUMÁRIO

1 INTRODUÇÃO

Este trabalho descreve o desenvolvimento de uma aplicação móvel responsável por gerenciar os

Dispositivos móveis inteligentes, como *smartphones* e *tablets*, estão cada vez mais presentes no mercado e estão transformando a experiência das tele-comunicações em um passo bastante acelerado. Nesta área, já estão bem estabelecidos serviços de comércio, bancos, entretenimento multimídia, entre outros.

A União de Vilas da Grande Cruzeiro, associação criada a partir do movimento comunitário local, vem há anos fazendo um trabalho de cadastramento de famílias em situação de risco e registrando as necessidades que fazem falta para estes moradores. Com o avançar do COVID-19 na cidade e na região, o trabalho do coletivo se intensificou e ganhou novas proporções, e se teve a ideia de aproveitar a tecnologia dos aparelhos celulares e a internet para apoiar esta luta, de forma a torna-la mais clara, visível e organizada.

Existem diversos esforços no sentido de obter informações sobre o decorrer da pandemia em comunidades periféricas, e também esforços para orientar doadores e voluntários sobre como ajudar com esta crise, como é o caso do aplicativo RAH - Rede de Apoio Humanitário. No entanto, embora atuem com objetivos semelhantes, não foram encontradas ferramentas específicas para o cadastramento de famílias nos moldes necessitados pela União de Vilas. Além disso, ter um aplicativo próprio, pensado e desenvolvido com regiões específicas em mente permite uma especialização que traz muitos benefícios aos usuários.

O aplicativo Cocar, nome escolhido por remeter a Coletivos e Cartografia, visa suprir as necessidades de organização dos coletivos sociais no que se refere ao conhecimento reunido sobre os moradores de suas regiões. Com o aplicativo, é possível registrar informações das famílias locais, tais como a faixa etária de seus integrantes, quais deles possuem necessidades de cuidados especiais, por quais carências elas estão passando, entre outros. Além disso, é possível marcar e visualizar as localizações destas famílias em uma ferramenta de cartografia, de forma a possibilitar ao usuário uma compreensão da real situação das famílias da região. Os experimentos com usuários sugerem que todas as funcionalidades do aplicativo Cocar atendem os requisitos estabelecidos e proveem, de fato, uma alternativa viável para o gerenciamento deste tipo de dados.

Nos capítulos a seguir, é descrito em maiores detalhes todo o processo de desenvolvimento bem como o contexto do trabalho como um todo. O texto está organizado da

seguinte forma:

O capítulo 2 apresenta as tecnologias empregadas na construção da solução elaborada para o problema, além de outros conceitos básicos importantes para a compreensão do deste trabalho. São descritos as metodologias de organização utilizadas para o desenvolvimento do trabalho e quais as motivações por trás destas escolhas. É apresentado o framework Ionic, com que o aplicativo móvel foi construído, e explicitado porque este framework foi considerado adequado para a implementação deste trabalho. De igual maneira, são mostradas e explicadas as escolha das linguagens e sistemas utilizados na porção *server-side* da solução.

O capítulo 3 expõe trabalhos relacionados e são observadas as diferenças deles com este trabalho. Foram encontrados diversas ações realizadas por movimentos sociais para coletar informações e divulgar orientações acerca do avanço do corona vírus, principalmente nas regiões periféricas do Rio de Janeiro.

O capítulo 4 trata de todo o processo de desenvolvimento. É apresentada a União de Vilas, organização cuja necessidade dá origem ao projeto, e é descrito o processo de levantamento de requisitos em encontros virtuais com os representantes da mesma. Também aqui são explicitados os passos da implementação do aplicativo Cocar, aprofundando-se nas questões técnicas de programação, tanto na área do aplicativo quanto no desenvolvimento *server-side* da API e base de dados.

No capítulo 5 é realizada a apresentação de todas as funcionalidades como de fato implementadas e disponibilizadas. O capítulo conta com imagens retratando as funcionalidades, bem como explicações detalhadas sobre o modo de utilização que foi planejado para cada uma delas.

No capítulo 6 é descrito em maiores detalhes como aconteceu a disponibilização aos usuários selecionados e como foi a recepção dos mesmos à aplicação. É narrada a experiência de capacitação realizada com os primeiros usuários, ainda com uma versão inicial do aplicativo, e é exposta uma análise do experimento de usabilidade efetuado após o encerramento das atividades de desenvolvimento.

Por fim, o capítulo 7 contém as conclusões realizadas a partir deste trabalho e são analisados pontos de melhoria e possível expansão da usabilidade do aplicativo no futuro.

2 TECNOLOGIAS UTILIZADAS

Este capítulo tem como objetivo apresentar os conceitos básicos e as tecnologias empregadas no desenvolvimento do aplicativo Cocar.

2.1 Metodologias de Desenvolvimento

Em qualquer projeto de software, seja de pequeno ou grande porte, é aconselhável o uso de uma metodologia de implementação para tornar o processo mais eficiente e controlado, evitando a desorganização e as inevitáveis falhas decorrentes dela.

Neste projeto, o processo utilizado foi ditado pela forma de relacionamento com a associação, que, embora tivesse a necessidade, ainda não tinha seus requisitos plenamente definidos. Por existir a necessidade deste processo ser adaptável e dinâmico, a metodologia escolhida foi ágil, assumindo, principalmente, vários elementos da metodologia *Scrum*.

2.1.1 Métodos Ágeis

Desenvolvimento ágil de software, ou métodos ágeis, referem-se à um conjunto de práticas e processos utilizados na criação de produtos de software e a sua disponibilização para os clientes e usuários finais. Essas metodologias eram conhecidas anteriormente como “métodos leves”, pois surgiram como uma reação a outros modelos caracterizados por uma pesada burocracia, que iam contra a forma usual como os engenheiros conseguiam realizar seus trabalhos mais eficientemente. Entre outras, pode-se citar entre esses métodos as metodologias XP (BECK, 1999), DSDM (STAPLETON, 1997), SCRUM (SCHWABER, 1997) e FDD (PALMER; FELSING, 2001).

No ano de 2001, ocorreu um encontro entre alguns dos mais importantes defensores dos métodos leves, na cidade de Snowbird, Utah (EUA), onde aconteceu uma discussão sobre os pontos em comum mais relevantes entre todos os métodos. Desse encontro resultou o Manifesto para Desenvolvimento Ágil de Software (BECK, K, et al., 2001), ou simplesmente, o Manifesto Ágil, que formaliza os valores e princípios destes métodos, que ficaram então sendo conhecidos como Métodos Ágeis.

Os valores definidos no Manifesto Ágil como pontos chaves a serem seguidos por

todas as metodologias desse tipo são os seguintes:

- **Indivíduos e iterações mais que processos e ferramentas;**

  Em todas as metodologias ágeis, os projetos são divididos em iterações, que contém todas as etapas necessárias como levantamento de requisitos, codificação e testes.

- **Software funcional mais que documentação abrangente;**

  Usando um método ágil, o enfoque está em produzir produto com valor agregado, e a documentação é normalmente considerada minimalista em comparação aos métodos tradicionais.

- **Colaboração do cliente mais que negociação de contratos;**

  Ao invés da comunicação com o cliente se dar apenas em uma etapa específica anterior a codificação, os métodos ágeis apostam em uma relação mais dinâmica que possibilita consultas aos clientes quando houver necessidade.

- **Responder a mudanças mais que seguir um plano.**

  Talvez o mais importante ponto dos métodos ágeis, a adaptabilidade do projeto torna possível que a definição do projeto seja revista à cada iteração, o que diminui muito desperdício de tempo.

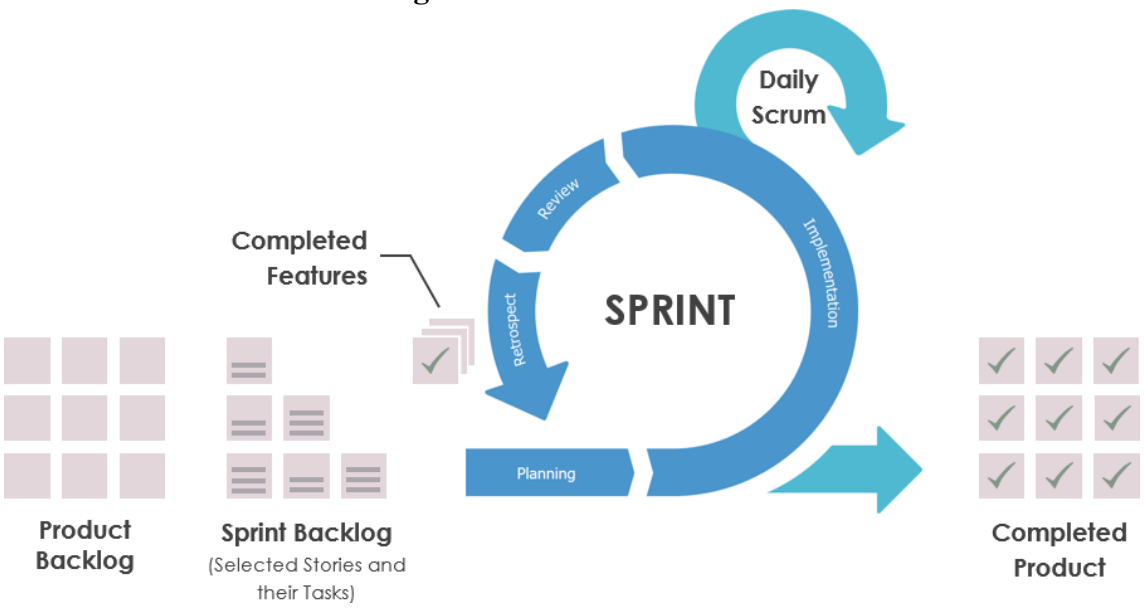
2.1.2 Scrum

Scrum é uma metodologia ágil para gestão e planejamento de projetos de software. No Scrum, as iterações em que os projetos são divididos são chamadas de *sprints*, e tem sua duração definida de acordo com a necessidade do projeto, sendo incomum ultrapassar o tempo de um mês. Ao final de uma *sprint*, os resultados obtidos são apresentados e avaliados, e é feito o planejamento para a próxima *sprint*. A maior vantagem dessa abordagem é que, por não haver uma fase engessada de planejamento, podem ser inseridos novos requisitos no projeto.

Em um ambiente de desenvolvimento em equipe, a equipe faz uma breve reunião diária (normalmente de manhã), chamada *Daily Scrum*. o objetivo é compartilhar o que foi feito no dia anterior, identificar e tratar problemas que estejam impedindo o progresso e priorizar o trabalho do dia que se inicia. Ao final de um *sprint*, a equipe apresenta as funcionalidades implementadas em uma reunião chamada *Sprint Review*. Finalmente, a

equipe parte para o planejamento da próximo *sprint*, e assim reinicia-se o ciclo. A Figura 2.1 ilustra a ordem dos processos ao utilizar a metodologia Scrum.

Figura 2.1: Fluxo do Scrum.



Fonte: https://beertech.ab-inbev.com/blog_detail/2

2.1.3 Kanban

Desenvolvido pela Toyota na década de 40, o Kanban (LIKER, 2004) (“cartão” em japonês) é uma prática simples e muito eficaz que pode ser usado por organizações de virtualmente qualquer segmento. Trata-se de uma técnica de organização ágil e visual que consiste em criar um cartão para cada tarefa, e anexar este em uma coluna que representa a fase atual em que se encontra essa tarefa. Em desenvolvimento de software, as típicas colunas “To-do”, “Doing” e “Done” deste método podem ser agrupadas nas diferentes etapas do projeto de software, como “Análise”, “Desenvolvimento” e “Teste”. Para melhorar ainda mais a visualização, os cartões podem possuir cores ou formatos diferentes para destacar características como prioridade e qual o membro da equipe é responsável por ele.

Existem disponíveis diversas ferramentas que implementam o Kanban de forma on-line e gratuita. Uma delas é o site Trello[1], que permite a criação e manutenção de quadros de forma cooperativa.

[1] https://trello.com/

2.2 Aplicativo

Aqui são apresentadas as tecnologias utilizadas no desenvolvimento do aplicativo móvel em si, e quais fatores foram levados em consideração para a decisão de escolher de cada uma.

2.2.1 Ionic Framework

A implementação de um sistema de software é um grande problema complexo que envolve a solução de incontáveis problemas menores, que, em sua maioria, não fazem parte dos requisitos do produto que se deseja criar. Funções de acesso à base de dados, segurança na troca de pacotes e renderização de componentes não são tarefas que são reimplementadas em todos os projetos, pois cada um destes é um projeto de alta complexidade por si só. Por isso, o reaproveitamento de códigos que foram desenvolvidos por especialistas experientes no campo, traz a vantagem de proporcionar um tempo maior para os programadores resolverem os problemas que fazem parte do escopo de seu projeto, além de contar com uma segurança maior ao utilizar artefatos de código já amplamente testados.

O aplicativo foi desenvolvido usando o Ionic Framework[2], que possibilita a construção de aplicativos híbridos, que poderão ser executados em diversos sistemas operacionais *mobile*, como o Android e o iOS. Esse *framework* é capaz de integrar-se com os mais populares *frameworks* de desenvolvimento web atuais, como o Angular, tornando a tarefa de desenvolver aplicativos muito mais fácil, principalmente para programadores com experiência prévia em web.

A linguagem de programação em que codificação é feita é a Typescript[3], uma das linguagem código-aberto que acrescenta funções ao JavaScript, que por sua vez é uma das linguagens dtye mais utilizadas atualmente. Por se tratar de uma linguagem de grande presença no mercado, ela conta com uma extensa gama de bibliotecas já implementadas para realizar tarefas comuns.

O *frontend* é escrito em HTML, como em qualquer aplicação web destinada ao consumo via navegador. Os elementos comuns de entrada e exibição de dados são todos fornecidos pelo framework, e já contam com ferramentas para validação de dados e

[2]https://ionicframework.com/
[3]https://www.typescriptlang.org/

tratamento de erros, o que ajuda a encurtar o tempo de implementação dos aplicativos.

Ionic Framework utiliza uma ferramenta chamada Capacitor[4] para criar um cenário onde os seus aplicativos tem acesso a todos os recursos do dispositivo onde está rodando. Assim, é possível acessar ferramentas como câmeras e GPS, além de reprogramar as funções dos botões virtuais do *smartphone* para melhor refletir as necessidades da aplicação.

2.2.2 Google Maps

O Google Maps (Google Team, 2020) é um dos sistemas de geolocalização mais amplamente utilizados no mercado de aplicações web, e seu aplicativo próprio é sem dúvidas o sistema deste segmento mais utilizado de todos. Sendo uma aplicação da Google, possui diversas integrações automáticas com o sistema operacional Android, sendo comum os aparelhos saírem da fábrica com o aplicativo já instalado. Além disso, possui uma versão *desktop* popular por facilitar o uso de ainda mais recursos.

O sistema também oferece suporte a diversas ferramentas como ícones personalizados, criação de linhas e polígonos sobrepostos no mapa, entre outros. A funcionalidade de geocodificação da Google é uma das mais poderosas do mercado, tornando a experiência de transformar coordenadas em endereços postais reais fácil e transparente.

Por esses e outros fatores, foi decidido usar o Google Maps para suprir os requisitos relacionados à geolocalização do aplicativo Cocar.

2.2.3 API RESTful

Para que os diferentes usuários do aplicativo Cocar possam ter acesso às informações coletadas pelos outros usuários, os dados não podem ser simplesmente gravados na memória do *smartphone*, devendo ser reunidos em um ponto de comum acesso para todos. Com essa necessidade em mente, foi criado um banco de dados MySql e uma RestAPI em PHP para receber e tratar as requisições de dados dos aplicativos.

API (Application Programming Interface) são conjuntos de instruções e padrões de programação que servem para fornecer dados e informações relevantes de uma determinada aplicação. A API funciona de forma passiva: ela permanece continuamente em

[4]https://capacitorjs.com/

espera até receber uma requisição de um programa. Ao receber a requisição, a API executará diversos testes para definir como deve tratá-la. As requisições podem ser dos tipos GET, POST, PUT, DELETE, entre outros. Estes tipos servem para indicar o tipo de operação que se pretende realizar com aquela requisição. A Tabela 2.1 explica brevemente os métodos mais importantes e amplamente utilizados de uma API RESTful.

Tabela 2.1: Métodos de uma API RESTful.

Método	**Utilização**
GET	Realização de consultas, obter informações da base de dados.
POST	Utilizado para guardar novas informações. É acompanhado de um pacote de informações que representam um objeto de dados complexo.
PUT	É comumente utilizado para atualizar informações.
DELETE	Remoção de informações da base de dados.

Fonte: O Autor

Uma API se torna RESTful (Representational State Transfer) (FIELDING, 2000) quando segue as diretrizes criadas por Roy Fielding em 2000. Essas diretrizes ajudam à implementar uma API. Neste modelo, todas as requisições para uma API específica são únicas e não alteram o estado do programa de modo algum. Isto é, nenhuma informação é salva da execução de uma requisição para outra. A aplicação deste conceito ajuda a diminuir consideravelmente a complexidade do código das APIs, e assim diminui o risco de gerar falhas no software.

2.2.4 PHP

A linguagem PHP[5] surgiu em 1995 e se consolidou como a linguagem de programação *server-side* mais popular. Embora atualmente outras linguagens mais novas tenham ganhado espaço, PHP continua sendo uma das linguagens com a maior comunidade ativa e possui diversos *frameworks* e bibliotecas recebendo atualizações constantemente. Além disso, trata-se de uma das linguagens mais fáceis de se obter hospedagem: virtualmente qualquer serviço de *hosting* da rede possui um plano básico para rodar aplicações em PHP, normalmente por preços muito acessíveis.

A partir da versão 5, a linguagem passou a contar com recursos da orientação a objetos, como herança, métodos protegidos e privados, classes, interfaces, construtores, entre outras funcionalidades. Estando atualmente na versão 7, a linguagem tem apre-

[5]https://www.php.net/

sentado evoluções de performance para adaptar-se às necessidades dos sistemas *web* mais atuais. Por essas vantagens, foi tomada a decisão de se implementar a API Rest necessária utilizando essa linguagem.

2.2.5 MySQL

O MySql[6] é um sistema de gerenciamento de banco de dados relacional. Ele utiliza a linguagem SQL para realizar consultas, inserções, remoções e atualizações dos dados armazenados. O sistema surgiu em 1994 de uma empresa sueca, pertencendo atualmente ao grupo Oracle, e sendo o SGBD mais amplamente utilizado nos dias de hoje. Por ser um dos sistemas mais utilizados do ramo, podemos encontrar uma instalação MySql na grande maioria dos serviços de hospedagem do mercado, o que torna seu uso ainda mais atraente. Além disso, MySql é facilmente integrado à uma aplicação desenvolvida em PHP, uma vez que a linguagem oferece suporte nativo para o gerenciamento de conexões.

2.2.6 XAMPP

Uma ferramenta de grande utilidade no desenvolvimento de um pequeno projeto de maneira local, o XAMPP[7] é um pacote que contém todos os utilitários necessários para simular um ambiente *web* real. Com execução de programas escritos em linguagens *web* populares através de seu servidor Apache e com uma instância pré configurada da base de dados MySql, a instalação do pacote tem o poder de habilitar o desenvolvimento de software *web* em um computador pessoal em poucos minutos.

O projeto, que teve seu lançamento inicial em setembro de 2002, é gratuito, código aberto e é disponibilizado para os sistema operacionais Windows, Linux e OS X das últimas versões, recebendo atualizações constantes pelo grupo que o mantém, o Apache Friends.

[6] https://www.mysql.com/
[7] https://www.apachefriends.org/

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados sistemas e trabalhos que surgiram para ajudar a diminuir os inevitáveis problemas originados pela crise gerada pela chegada da pandemia do coronavírus. São brevemente expostos três projetos, explicando seus objetivos principais, metodologias e resultados obtidos até o momento.

3.1 Painel Unificador COVID-19 nas Favelas do Rio de Janeiro

O Painel Unificador COVID-19 nas Favelas do Rio de Janeiro[1] é uma iniciativa da ONG Comunidades Catalisadoras (ComCat) que visa ampliar a visibilidade dos dados sobre a presença e o alcance do COVID-19 nas comunidades das favelas do Rio de Janeiro. Informações oficiais sobre os territórios de favelas, as áreas mais vulneráveis da cidade do Rio de Janeiro, ainda são escassas, e por isso dependem de iniciativas dos próprios moradores.

Os moradores das regiões contempladas pelo projeto são incentivados a preencher um formulário *on-line* que indaga sobre:

- os sintomas desta pessoa;
- se esteve em contato, ou reside, com alguém com suspeita de estar contaminado com o vírus;
- se esteve em uma área considerada de risco, como em uma unidade básica de saúde;
- se possui alguma condição pré-existente relevante;
- questões acerca do perfil social.

As informações autodeclaradas via formulário são então combinadas com as informações obtidas pelos órgãos oficiais afim de formar um painel mais completo, onde é possível obter os últimos números da pandemia na região e também verificar as densidades destes casos diretamente em um mapa, o que ajuda a se fazer um melhor julgamento sobre a situação real do avanço da pandemia.

[1] https://experience.arcgis.com/experience/8b055bf091b742bca021221e8ca73cd7/

3.2 RAH - Rede de Apoio Humanitário

A Rede de Apoio Humanitário nas e das Periferias (RAH[2]) lançou um aplicativo que permite ajudar as iniciativas que estão distribuindo produtos alimentícios e de higiene para famílias nas periferias da capital paulista, grande São Paulo e Baixada Santista. O aplicativo prove um mapa com cerca de 70 locais espalhados pelas regiões mencionadas. Esses polos, formados por associações, ONGs, instituições religiosas, entre outros, já atuavam nessas áreas antes do começo da crise de saúde do COVID-19. O aplicativo utiliza o sistema de GPS do *smartphone* do usuário para indicar quais os polos mais próximos que estão requisitando ajuda e doações, facilitando o deslocamento e ajudando a construir um vínculo do doador com organizações locais. As famílias necessitadas podem, então, ter acesso a estas doações ao conveniar-se aos diferentes polos que realizam as coletas. Embora tenha chegado neste momento de extrema importância, o projeto não se limita ao momento atual de pandemia e tem a intenção de continuar servindo como ponte indefinidamente.

3.3 Dados do Bem

Dados do Bem[3] é uma ferramenta desenvolvida pelo Instituto D'Or de Pesquisa e Ensino (IDOR) e pela Zoox Smart Data, contando com médicos, pesquisadores e cientistas da informação, que usa a inteligência de dados para analisar a evolução da imunidade na população.

Participantes do estudo fazem download do aplicativo e realizam uma autoavaliação, mesmo no caso de não apresentar nenhum sintoma. Essas informações ajudam a analisar de forma mais realista a curva de disseminação do COVID-19, e contribui na busca de uma melhor forma de combate-lo.

O projeto conta com o apoio direto do governo estadual do Rio de Janeiro e da prefeitura municipal da cidade de Niterói, o que amplia consideravelmente as possibilidades de inserção deste trabalho em toda a comunidade. Mais recentemente, o projeto também criou parcerias com o governo do Goias, onde mais de 61 municípios já utilizam a inteligência do aplicativo para selecionar pessoas para testagem.

As pessoas cujas respostas ao aplicativo indicam uma alta probabilidade de conta-

[2]https://play.google.com/store/apps/details?id=br.ong.rah&hl=en_NZ
[3]https://dadosdobem.com.br/

minação com o vírus são orientadas a comparecerem a um dos postos de testagem governamentais conveniados ao programa.

3.4 Análise Comparativa

É possível perceber que os movimentos sociais em diversos pontos do país estão apostando na tecnologia das telecomunicações para combater a desigualdade social. Estes esforços parecem estar sendo realizados de forma autônoma, ainda não existindo uma unificação de abordagens ou compartilhamento de informações entre os diferentes projetos.

Não foi encontrado nenhum projeto que sirva como alternativa para a questão específica de cadastramento de famílias, embora existam ferramentas disponibilizadas on-line para obtenção de dados que possibilitem criação de formulários. Entretanto, as mesmas precisariam de grandes ajustes e adaptações por parte de seus usuários, e não podem ser utilizadas de maneira prática para o mesmo objetivo buscado pelo aplicativo Cocar.

A Tabela 3.1 mostra uma comparação entre os elementos comuns entre estes esforços. Entretanto, é importante observar que embora os projetos tenham um claro objetivo comum: promover o bem estar social de suas respectivas regiões de atuação, suas abordagens são consideravelmente distintas, e qualquer comparação deve levar este fato em consideração antes de que sejam tiradas quaisquer conclusões.

Tabela 3.1: Análise comparativa de trabalhos relacionados.

	Painel Unificador	**RAH**	**Dados do Bem**
Sintomas COVID-19	Sim, autoavaliação	Não	Sim, autoavaliação
Avalia situação social	Sim	Não, pois conta com os polos para isso	Não, apenas dados demográficos
Emprega cartografia	Sim	Sim, para localizações de polos, não de casos	Sim
Disponibiliza relatórios	Sim	Não	Sim, do próprio app e dos dados do SUS
Oferece orientações sobre COVID-19	Não, não entra no escopo	Não	Sim

Fonte: O Autor

4 APLICATIVO COCAR: DESENVOLVIMENTO

Este capítulo descreve as atividades realizadas no desenvolvimento do aplicativo Cocar. É apresentado o contexto do trabalho e o processo de levantamento de requisitos juntamente aos clientes. Em seguida, são explicitados e explicados os processos de implementação utilizados na criação do aplicativo móvel, API e banco de dados. Por fim, o aplicativo Cocar é apresentado tal como disponibilizado aos usuários finais.

4.1 União de Vilas da Grande Cruzeiro

A União de Vilas da Grande Cruzeiro é uma associação comunitária existente desde 1979, criada a partir do movimento comunitário que supre uma carência deixada pelo Estado na região da Grande Cruzeiro, zona sul de Porto Alegre. A organização funciona através da atuação de lideranças das comunidades, numa lógica de fortalecimento coletivo. Dentro da estrutura do coletivo, as lideranças locais são responsáveis por mapear as necessidades dos moradores, e a coordenação se encarrega de conseguir doações para suprir essas necessidades, junto aos movimentos sociais que apoiam a organização. A coordenação da União de Vilas é composta atualmente por três membros que assumiram este papel no ano de 2019.

Além da entrega de doações, o coletivo também promove ações sociais na comunidade, principalmente em prol da saúde e da cultura, conscientizando a população sobre os interesses públicos da região, como a situação das escolas públicas locais e das unidades básicas de saúde. Uma dessas ações, por exemplo, e á movimentação de carros de som que passam informações sobre prevenção do novo corona vírus.

Com a chegada do corona vírus à Porto Alegre, a situação de muitas famílias da comunidade deteriorou-se rapidamente, tornando a atuação do coletivo na região ainda mais importante. Em decorrência disso, um maior número de empresas e organizações apresentou interesse em contribuir, e a coordenação do movimento foi capaz de obter ainda mais apoio para a obtenção de cestas básicas e produtos de higiene essenciais no combate ao avanço da pandemia.

4.2 Levantamento de Requisitos

A União de Vilas carecia de uma maneira unificada e sistematizada de guardar as informações reunidas sobre as famílias atendidas por eles, e essa dificuldade atrasava as operações de distribuição de doações.

O processo de levantamento dos requisitos foi feito através de uma sequência de reuniões com os três membros da coordenação da União de Vilas. Essas reuniões inciaram-se antes do início da implementação, e seguiram ocorrendo durante todo o processo de desenvolvimento. Todas essas reuniões foram realizadas via conferência *on-line*, uma vez que a pandemia do corona vírus impedia encontros presenciais.

Através destas reuniões foi possível identificar uma série de requisitos para o aplicativo Cocar.

- Possibilidade de registrar informações sobre uma família, sendo os campos disponíveis divididos entre obrigatórios e opcionais; Nome do membro responsável
- Acesso a lista de cadastros já existentes;
- Ser capaz de obter dados quantitativos acerca do montante de cadastros e suas características principais;
- Permitir registrar entrega de doação;
- Permitir o registro das informações de maneira *off-line*;
- Permitir acesso apenas aos usuários selecionados pela coordenação do coletivo;
- Criar dois tipos de usuários: coordenadores e referências locais, que possuem permissões diferentes.

Além destes, um requisito não funcional considerado extremamente importante pelos coordenadores era que sistema deveria ser intuitivo e com usabilidade facilitada. Em todas as funcionalidades criadas no aplicativo, uma consideração especial sempre foi feita para avaliar se os usuários compreenderiam naturalmente e não seria um fator que os desmotivassem a utilizar o programa.

Com o avanço da pandemia, o escopo do projeto com a Vila Cruzeiro se expandiu, e novas áreas foram incluídas para suporte a: a) cartografia: tarefas relacionadas ao mapeamento das famílias geograficamente; b) sociologia: organizações de movimentos sociais

de periferias. Essas duas áreas trouxeram considerações de grande valia para o avanço do trabalho.

Embora o projeto tenha se iniciado a partir de uma necessidade do coletivo União de Vilas, o objetivo do trabalho foi se expandindo quando foi percebida a possibilidade de ajudar também outros coletivos deixando o aplicativo um pouco menos específico às necessidades da União de Vilas. Foi realizado um encontro *on-line* contando com a presença de representantes da União de Vilas, do coletivo Periferia Move o Mundo e da Associação dos Moradores e Amigos da Vila Tronco e Arredores (AMAVTRON), onde foi possível encontrar denominadores comuns entre as necessidades destes coletivos, e foi decido por realizar algumas alterações no sistema original. Além destes, também foram feitos contatos com o coletivo Somari, atuante no bairro Glória, e foi requisitado acesso ao aplicativo Cocar também por eles.

Foi decidido que, ao invés de separar os dados das famílias registradas por cada coletivo, as organizações que operam sobre uma mesma área teriam acesso às informações registradas a todas as famílias já cadastradas na região. Além da melhor organização dos dados, os coletivos esperam com isso ser capazes de identificar moradias já contempladas, e assim destinarem suas doações para os realmente mais necessitados. Desta forma, os coletivos União de Vilas, Periferia Move o Mundo e AMAVTRON trabalham sobre a mesma base de dados, referente à área da Cruzeiro, enquanto o coletivo Somari opera sobre outra base referente a região da Glória.

A principal mudança foi a retirada dos dois tipos de usuários do sistema, a partir deste momento, todos os usuários seriam iguais em termos de permissões. Além disso, também seria alterado um importante paradigma: até então cada usuário era vinculado à uma comunidade. Essa alteração, embora não interferisse muito com a experiência de uso do usuário, acarretava grandes mudanças estruturais no banco de dados bem como em ambos aplicativo e API.

4.3 Implementação

A construção do aplicativo Cocar se deu em duas partes principais: o aplicativo móvel em si, desenvolvido com o Ionic Framework, e a API, aplicação a ser executada em um servidor na *web* que atende as requisições das diversas instalações do aplicativo Cocar. Essa API foi inteiramente implementada utilizando a linguagem PHP. Também na parte do servidor, um banco de dados relacional MySql foi utilizado para armazenar

todas as informações relevantes do sistema. Essas duas tecnologias foram escolhidas pela grande disponibilidade de servidores que oferecem suporte a elas, o que permite que essa aplicação possa ser migrada para outra hospedagem sem nenhuma necessidade significativa de retrabalho.

De modo a facilitar e acelerar o trabalho do desenvolvimento, a implementação foi feita de maneira local. Utilizou-se o XAMPP[1] para criar um servidor Apache local e criar uma instância da base de dados MySql, dessa forma simulando perfeitamente o ambiente do servidor onde o sistema é de fato hospedado. Além do considerável ganho em praticidade, esta prática é imprescindível após uma versão inicial do aplicativo já estar em operação.

Para este projeto, um serviço de hospedagem de plano básico foi utilizado, que oferece servidor Apache capaz de executar a aplicação PHP, e uma instalação simples do MySql já incluída no pacote de serviços. O gerenciamento desta base de dados é feita através do *phpmyadmin*, ferramenta amplamente utilizada para simplificar as interações com o banco.

4.3.1 Desenvolvimento do Aplicativo Cocar

O aplicativo foi criado utilizando o Ionic Framework, através do qual é possível criar telas para aplicativos móveis da mesma maneira que um *webdesigner* criaria páginas de um site. O *framework* possui uma gama extensa de funções já implementadas, de forma a tornar mais simples o uso de elementos comuns, como botões, listas, campos de entradas de dados, etc.

A tela de login consiste em duas entradas de dados, uma para o nome de usuário e a outra para a senha. O campo de texto da senha foi configurado como tal para que os caracteres fiquem ocultos conforme sejam digitados. Além dos campos, um botão de login envia os dados para a API na web, onde é realizada a consulta ao banco por um usuário cujos dados confiram. Quando uma resposta é retornada, o sistema decide se avança para a próxima tela ou exibe erro de autenticação na tela. Caso prossiga, alguns dados do usuário são salvos na memória do celular, como id do usuário, nome e nome de usuário. No próximo uso do aplicativo, se não houver acontecido um log-out manual por parte do usuário, o sistema enviará esses dados para outra API, que realiza uma verificação destes dados e também se o usuário em questão ainda consta como ativo no banco de dados.

[1] https://www.apachefriends.org/

Primeiramente foi criada a tela da lista de cadastros, ainda sem a implementação da lista, com um botão flutuante com a *label* “+” levando à criação de um novo registro. A seguir, a tela do formulário de cadastro, onde encontra-se a maior parte da lógica do projeto. A maioria das entradas de dados desta página foi criada utilizando o componente *ion-input* do Ionic, que é capaz de criar os tipos mais populares de campos para formulário. Para implementar a validação dos dados no formulário conforme acontece o preenchimento, foi utilizado um módulo chamado *ReactiveFormsModule*, que permite a imediata exibição de valores inválidos em diversos campos, como Nome do Responsável, Endereço, CPF, entre outros. Uma função de validação de CPF foi implementada separadamente e fornecida aos métodos nativos do *ReacitiveForms* para que um número inválido digitado informe erro ao usuário imediatamente.

Uma classe chamada "Profile"foi criada para representar cadastros, localizada dentro da estrutura do aplicativo em "app/models/profile.ts"juntamente aos arquivos que guardam os demais tipos personalizados do projeto. A classe possui 22 atributos que guardam os valores inseridos pelo formulário, sendo 2 sendo para latitude e longitude da localização geográfica da família e 2 são listas que recebem os cadastros de moradores com sintomas de COVID-19 e com doenças pré-existentes.

Para a seleção das necessidades da família, visto que eram poucas, imutáveis e precisavam de uma maneira simples de seleção, foi criado um componente próprio, utilizando os componentes do Ionic *ion-card* e *ion-icon* e o evento de *click*. Desse modo, um ícone apropriado para cada necessidade foi escolhido e, ao tocar neste icone, o mesmo troca a cor de fundo para indicar que a necessidade em questão está selecionada para a família. A necessidade intitulada "Outros"possui o comportamento especial de, quando estiver selecionada, ativar a visibilidade de um componente *ion-textarea*, que nada mais é do que um campo preparado para receber textos mais extensos, onde o usuário pode opcionalmente expressar quais outros itens são necessitados por esta família.

Para informar o número de integrantes da família que são crianças, adultos ou idosos, também foi criado um novo componente personalizado. Trata-se de um número com dois botões para somar ou subtrair deste valor. As alternativas oferecidas pelo Ionic para este tipo de tarefa seriam um *ion-input* que aceitasse apenas números, ou um *ion-select*, que criaria uma caixa de seleção onde os números poderiam ser opções do sistema, mas ambas foram julgadas inadequadas para a tarefa. Neste caso, o novo componente foi definido formalmente, e foram criados um arquivo html, typescript e scss próprios para a implementação do mesmo. Após esta implementação, o componente pode ser

simplesmente utilizado pelas tags <*app-counter*></*app-counter*> no arquivo html da tela de cadastro, o que ajuda a deixar o código referente ao cadastro mais sucinto e legível.

Existem dois campos especiais na tela de cadastro que se referem ao cadastro de moradores da família que estejam apresentando sintomas de COVID-19 ou que possuam doenças pré-existentes. Ambos receberam um tratamento especial, pois na cada uma é na realidade composta de três campos por si só, e podem ser cadastradas diversos moradores deste tipo para cada família. Por isso, o seguinte mecanismo foi utilizado para estes campos: um botão "+" ao lado de da *label* de cada campo foi criado utilizando o componente *ion-icon*, combinado com o evento *click*, para levar o usuário a uma outra tela onde os campos pertinentes a cada tipo de morador podem ser encontrados. Cada tela conta com um botão voltar, que cancelaria o cadastramento deste morador, e um botão salvar, que efetivaria as informações e as passaria então para uma lista dentro do cadastro da família, que guarda esse tipo de informação. Estas listas são exibidas abaixo do *label* de cada campo, com cada item exibindo nome do morador, idade, e também dois botões que possibilitam a edição ou exclusão dos itens do cadastro. Importante notar que o efetivo envio destas informações para a API só é dado quando o cadastro da família é salvo, pois é então que estas listas são juntamente enviadas. Em ambas as telas, uma validação de dados é realizada para tornar impossível o registro sem o preenchimento do nome de morador ou uma idade valida.

Para representar um morador da residência familiar foi criada a classe "Person", que contem atributos para nome e idade. Essa classe é estendida por outras duas classes que representam moradores com sintomas de COVID-19 e com doenças pré-existentes, cada uma possuindo os atributos únicos necessários. São objetos destes tipos que integram as duas listas mencionadas anteriormente.

Na tela de moradores com sintomas de COVID-19, o terceiro campo, que é único a esta tela, pergunta ao usuário sobre quais sintomas estão sendo observados. Nas versões iniciais do aplicativo, eram exibidos aqui nove possíveis sintomas, como "tosse seca", "febre", "enjoo", entre outros. Porém, após orientações de profissionais da saúde repassadas ao professor Bilibio, foram decididos por apenas dois possíveis sintomas: "Qualquer quadro gripal ou febre persistente", cuja escolha pelo usuário imediatamente ativa a visibilidade de um curto texto informando os números telefônicos para triagem a distância ou unidade de saúde apropriada mais próxima, ou "Falta de ar", cuja escolha ativa a exibição da orientação de procurar emergência imediatamente. Foram utilizados os componentes *ion-item* de forma iterativa, através da diretiva **ngFor="let s of symptoms"*, para criar

cada sintoma, e o componente *ion-checkbox* para criar a caixa de seleção da opção.

Já na tela de moradores com doenças pré-existentes, o terceiro campo consiste de um simples *ion-textarea*, para que o usuário do sistema discorra sobre as condições do morador.

A lista de cadastros utiliza novamente uma diretiva **ngFor* para gerar código HTML para cada item na lista de cadastros. Cada cadastro aparece com o endereço da família e o nome do morador responsável pelo cadastro. A propriedade da cor deste nome recebe um *bind* com a propriedade "nameColor" da classe "Profile", que utiliza os atributos da classe para calcular a quanto tempo essa família não recebe uma doação que contemple todas suas necessidades e assim atribuir uma *string* correspondente a cor adequada. A direita do nome, três ícones são possivelmente exibidos, dois deles dependendo dos atributos "num_kids" e "num_seniors" serem maiores que zero, e outro ativado pela "symptomaticPeople" não estiver vazia.

Foram implementas cinco diferentes modos de ordenamento da lista. Estas opções estão acessíveis tocando um botão na barra superior da lista, que exibe *popover* criado pelo componente nativo do Ionic "PopoverController" contendo as opções. Cada opção utiliza atributos diferentes de "Profile"para efetuar o ordenamento. As opções de listar por data de criação e nome do morador de referência utilizam os atributos "created" e "ref_name", respectivamente. Já as opções de ordenar por comunidade e referência local não apenas atuam reordenando a variável de lista dos cadastros: elas alternam a visibilidade do código HTML para exibir outro tipo de lista, onde os itens são as comunidades ou referências locais e estes possuem subitens, que são os cadastros pertencentes a cada um. A visibilidade no HTML é ativada aqui utilizando a diretiva **ngIf*, que possibilita configurar uma condição onde o elemento é de fato desenhado na tela. De maneira similar, a quinta e última opção ativa ainda outro tipo de lista, onde os cadastros são acompanhados de um número que representa a prioridade que a família tem de receber uma doação. Essa prioridade é calculada utilizando os atributos "created", "last_delivery", "needs" e "deliveries".

Com o objetivo de organizar as diferentes partes do aplicativo, foi utilizado o componente *ion-menu* para criar uma página onde é possível abrir um menu de opções a partir de um botão no cabeçalho. A partir dele, os usuários são capazes de alternar entre as três páginas principais do sistema, e também podem acessar a opção de *log-out*.

A tela de relatórios possui duas partes principais. A primeira contém uma tabela (*ion-grid*) que cria linhas (*ion-row*) de maneira iterativa a partir de nove itens pré defini-

dos no código. Estes itens apresentam informações estatísticas acerca dos cadastros das famílias. A seguir, em outra tabela, fica um resumo dos cadastros, que conta apenas com o nome do responsável, a data de criação deste cadastro e a data do último recebimento de doação pela família. No topo desta tela existe um *ion-select*, um campo para seleção de uma opção, que permite aos usuários filtrar estas informações e visualizarem as estatísticas e resumo apenas de uma comunidade específica dento da macro região. Ao selecionar uma opção, um evento do Ionic chamando "ionChange" é disparado, e uma função que recalcula todas as métricas é executada.

A funcionalidade de exibição de mapas é acessada em dois momentos diferentes: através do cadastro da família, quando é utilizada para registrar uma nova localização para esta família, e através do menu principal, onde o usuário poderá consultar o mapa e visualizar todas as famílias que possuem uma localização geográfica cadastrada. O *front* de ambas é praticamente o mesmo, a única diferença sendo os botões para registrar ou voltar na barra superior do primeiro caso. Fora isso, elas possuem um componente HTML *div* que possui uma diretiva "#mapCanvas", indicando que são relacionadas à um atributo da classe em Typescript, que por sua vez é utilizado como link para criação e atualização do mapa de forma programatica. Para a criação das telas de mapas, foi criado uma nova classe chamado "MapHandler", que contém toda a implementação necessária para ambas as telas onde o recurso de geolocalização é utilizado. Esta classe contém uma chave própria fornecida pela Google que é necessária para a utilização dos recursos do Maps apropriadamente. Aqui, o objeto do mapa "google.maps.Map" é inicializado utilizando as configurações especificadas para o aplicativo. São definidas as propriedades de em qual escala o mapa inicia ("zoom"). É utilizada a propriedade "mapTypeControlOptions" para personalizar os botões que aparecem sobre o mapa e qual a posição dos mesmos. A propriedade "center" é definida também no momento da inicialização, e seu valor normalmente dependo da posição do usuário. Para criação desta funcionalidade, foi necessário utilizar a classe nativa do *framework* "Geolocation". Utilizamos o método "getCurrentPosition" desta classe para obter as coordenadas geográficas do dispositivo, e repassamos as informações de latitude e longitude para o objeto de mapa do Google, de forma a centralizar o mapa na posição atual do usuário. Quando a tela de mapas está sendo acessada pelo cadastro de família e já possuir uma localização salva, o mapa será inicializado centralizado nesta localização.

Para representar localizações no mapa, são utilizados marcadores, objetos da classe "google.maps.Marker". Para criar um objeto deste tipo, é necessário passar três da-

dos ao construtor: a instância do mapa ao qual ele será vinculado, um objeto "google.maps.LatLng", com as informações de latitude e longitude indicando onde ele será exibido, e o endereço da imagem utilizada como ícone, que neste caso é apenas o nome do arquivo que é embutido dentro do aplicativo. O comportamento dos marcadores difere nos dois contextos diferentes de acesso ao mapa.

Quando o mapa é acessado pelo cadastro de família, existem apenas dois marcadores possíveis visíveis no mapa. Ao abrir o mapa, o mapa é centralizado na localização do usuário obtida pelo GPS, e um marcador é criado neste local. Porém, ao arrastar o mapa, este marcador também se move, de maneira a continuar estando no centro da tela. Esta funcionalidade é criada utilizando o evento do mapa "center_changed" e o vinculando à uma função que reposiciona o marcador. Ao salvar, as coordenadas deste marcador são guardadas no objeto "Profile" em edição como a nova localização desta família. Se o usuário abrir o mapa para localizar uma família e a mesma já possuir coordenadas salvas, um segundo marcador, com ícone diferente, será exibido de maneira fixa indicando onde era a localização correta anteriormente.

Acessando o mapa pelo menu principal, o comportamento dos marcadores é adaptado para exibir todas as famílias já cadastradas. Ao abrir, um marcador central também é exibido na localização do dispositivo do usuário, mas o mesmo não se move ao arrastar o mapa. Extraímos da lista de cadastros de famílias as informações necessárias para criar os marcadores, e criamos em "MapHandler" uma lista de marcadores que serão exibidos no mapa. Utilizamos as coordenadas da família para inserir o marcador na localização correta, e utilizamos a propriedade "nameColor" também aqui para definir de qual cor será o ícone representando esta família. No mesmo laço onde acontece a criação de cada marcador, já é vinculado o evento "click" do mesmo a uma função que cria um objeto "google.maps.InfoWindow". Para isso, também são extraídos os campos "ref_name", "address" e "phone" do objeto "Profile" deste cadastro, para que estas informações sejam exibidas em uma pequena janela sobre o mapa quando o usuário tocar em qualquer um dos marcadores.

Para guardar os dados do aplicativo que são utilizados em todos as telas, foi criada uma classe chamada *SharedDataService*, que pode acessada de qualquer outra classe.

Aqui acontece o carregamento dos dados do servidor acontece aqui, que ficarão em variáveis de listas acessíveis a partir desta classe, de modo que ao se movimentar entre as diferentes telas do software o usuário não seja obrigado a repetir os downloads de informações desnecessariamente. Para evitar o carregamento desnecessário de dados toda

a vez que o aplicativo é aberto, o mesmo salva os cadastros na memória do dispositivo através da nativa “Storage”. Desta forma, o comportamento padrão é carregar primeiro os dados salvos localmente e depois utilizar a API para realizar uma consulta menor, que atualiza os cadastros alterados e carrega os novos. Para isso, são passados como parâmetros para a API uma lista com os identificadores dos cadastros já existentes na memória e a data quando ocorreu a última atualização. Além disso, nesta classe também se encontram as informações sobre o usuário logado, que são utilizadas em diversos pontos do software.

Foi criada uma classe chamada *RegisterService* para agrupar todos os métodos de troca de mensagens com a API REST sendo executada no servidor. Esta classe é acessada ena inicialização para realizar as consultas, no login para a autenticação e também nos momentos onde dados são enviados para armazenamento no banco de dados. É utilizado aqui uma classe nativa chamada *HttpClient*, que já oferece os métodos de *post*, *put*, *delete* e *get* necessárias para a comunicação com a API. Todos os métodos recebem como argumentos o nome da API de destino, o endereço do servidor, que é mantido nesta classe de maneira estática e privada, e um conjunto de *headers* a serem enviados juntamento com o pacote. Os *headers* enviados pelo aplicativo são:

- “Content-Type”, que recebe o valor ’application/json’. Todos os objetos enviados às APIs são previamente convertidos para JSON, e este *header* indica o uso deste formato;
- “Authorization”, cujo valor passado é um GUID utilizado na API para testar se este pacote está vindo de fato de uma instância do aplicativo Cocar;
- “App-Version”, que transmite a informação de qual versão do aplicativo está fazendo a solicitação;
- “Db-Id”, outro GUID que informará a API qual *schema* da base de dados deve ser utilizada para realizar a operação requisitada.

4.3.2 Projeto do Banco de Dados

A estrutura da base de dados foi modificada frequentemente, conforme as reuniões com os coordenadores identificavam novas informações relevantes que deveriam ser mantidas nela. O modelo de dados foi elaborado com base nas informações que o coletivo já

possuía sobre as famílias. As definições de nomenclatura dos bancos de dados e tabelas foram feitas seguindo a experiência prévia do autor neste tipo de projeto: em inglês, com nomes de tabela no singular.

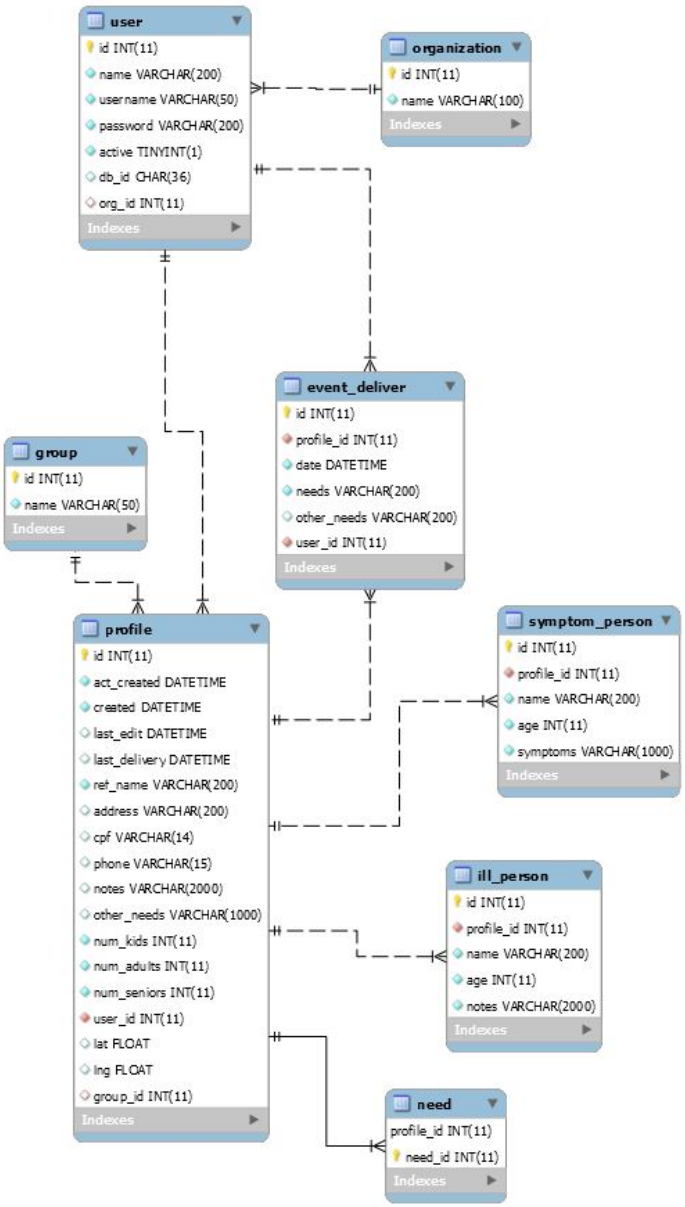
A Figura 4.1 ilustra o diagrama da base de dados. Em sua forma final, a base de dados foi divida: um *schema* com as tabelas "user" e "organization" guardam as informações de todos os usuários do sistema. Na tabela "user", uma coluna informa com qual dos demais *schemas*, onde estão salvas as informações registradas das famílias de uma região, este usuário deve se conectar. Desta forma, as informações de cada região ficam separadas, o que serve como mais um nível de proteção de dados e evita que um crescimento no volume de dados impacte severamente o funcionamento do aplicativo. A tabela "organization" registra o nome dos coletivos que utilizam o aplicativo. Uma chave estrangeira na tabela "user" aponta para uma entrada de "organization", para definir de qual organização o usuário é filiado, e essa informação é utilizada para informar o usuário final qual o coletivo que registrou uma doação.

A tabela "group" serve para armazenar as comunidades presentes na região. Até o momento, a inserção destas comunidades na base de dados foi feita diretamente pelo autor após obter a informações dos coordenadores da União de Vilas.

Para representar os cadastros de famílias, foi criada a tabela "profile", que possui diversos atributos, como "name", "address", "cpf", "phone", "notes", "num_kids", "num_adults", "num_seniors", "lat", "lng", "last_edit", "last_delivery", "created", "user_id" e "group_id". Além das informações que já eram requisitadas anteriormente, essa tabela conta com colunas para armazenar informações de caráter mais técnico, como a última edição do cadastro e qual o usuário responsável por essa família. Também é salvo qual comunidade dentro da grande cruzeiro essa família pertence, pelo identificador da comunidade, que é salva previamente em outra tabela. As coluna "lat" e "lng" existem para armazenar as informações de latitude e longitude da família, como é possível serem informadas através do mapa no cadastro.

A tabela "ill_person" tem a finalidade de guardar informações relativas à uma pessoa com doenças pré-existentes. Trata-se de uma tabela simples, com os campos "name", "age" e "notes", além da chave primária e do campo "profile_id", que é a chave estrangeira que aponta para a família em que a pessoa pertence. Muito similar, a tabela "symptom_person" salva as informações de integrante da família que está apresentando sintomas relacionados ao vírus COVID-19. Diversas pessoas com doenças pré-existentes ou exibindo sintomas podem ser cadastradas para cada família.

Figura 4.1: Diagrama do banco de dados.



Fonte: O Autor

Cada doação que uma família recebe do coletivo União de Vilas fica armazenada na tabela “event_deliver”. O atributo “date” registra qual a data e hora que a entrega desta doação foi efetuada. A coluna “needs” guarda uma *string* com os códigos das necessidades que foram entregues. Por fim, o campo “other_needs” permite guardar a informação de uma doação que não se encaixa nas necessidades pré definidas.

4.3.3 API

O objetivo da API é prover uma forma de comunicação entre as instâncias do aplicativo e o banco de dados que armazena os dados utilizados por ele. Foram implementados ao todo sete APIs que são acessadas através da classe “RegisterService” do aplicativo. Cada uma destas é responsável por receber uma requisição, verificar a autenticidade da mesma e se os dados mínimos esperados estão presentes, processar o pedido utilizando as classes apropriadas, e retornar um resultado, seja este dados ou apenas o código HTTP apropriado indicando sucesso ou falha.

Toda a comunicação com o banco de dados acontece por meio de uma classe “Database”, que contém as informações necessárias para comunicação com a base. Aqui, um método chamado “getConnection” utiliza a classe nativa do PHP, “PDO”, para estabelecer uma conexão com a base. Este método recebe um parâmetro chamado “db-id”, que identifica qual *schema* deve ser passado no parâmetro “dbname” da *string* de conexão utilizada pelo “PDO”. Além desta *string*, são necessários um nome de usuário e senha válidos para a instância do banco de dados sendo acessado.

As APIs “read” (GET) e “save” (POST) criam instâncias da classe “Profile”, responsável pelas leituras e escritas na tabela de mesmo nome. O método “save” registra os dados na tabela através de um *INSERT* ou um *UPDATE*, dependendo se o cadastro for novo ou existente. Também neste método, são chamadas funções responsáveis por guardar os valores nas tabelas “need”, “symptom_person” e “ill_person”. No caso de um UPDATE do cadastro, os valores destas tabelas que são associados ao cadastro em questão são excluídos e os novos valores são então inseridos. O método “read” realiza uma consulta sobre as famílias cadastradas nesta base com base nos parâmetros recebidos, que consistem em uma lista de *ids* e a data da última atualização do requerente. São retornadas duas listas: uma com os cadastros novos e aqueles a serem atualizados, e uma lista de números, contendo os identificadores de cadastros que não existem mais na base de dados, e devem ser excluídos pelo aplicativo.

A API “delete” também atua sobre a tabela de cadastros, através do método de mesmo nome presente na classe “Profile”. O único parâmetro necessário aqui é o identificador do cadastro. A função exclui todas as informações relacionadas ao cadastro por chave estrangeira antes de prosseguir com a execução do *DELETE* da família em si. Para fins de reaproveitamento de código, todos os *deletes* são feitos por uma função genérica que recebe, além do valor identificador, os nomes da tabela e da coluna usada como chave.

As APIs “login” (GET) e “verify_session” (GET) utilizam a classe “User”. O método “login” recebe nome de usuário e senha e realiza um *SELECT* na tabela “user” para encontrar uma entrada onde ambos estejam corretos. Na clausula *WHERE* desta consulta também é testado se a coluna “active” tem valor verdadeiro, caso contrário o usuário não é considerado válido. Como resultado são retornadas informações do usuário, como “id”, “name” e “db-id”. Já no método “verify”, os parâmetros de entrada são apenas o identificador e o nome de usuário. A consulta apenas verifica se existe o usuário existe de fato e se continua como sinalizado como ativo, e o resultado retornado ao aplicativo não contém dados, apenas o código HTTP adequado para indicar sucesso ou falha da operação.

A API “save_delivery” (POST) conta com a classe “Delivery” para salvar uma entrega na tabela “event_deliver”. O método “save” desta classe implenta a *query* que insere na tabela os dados de data, quais itens estão sendo doados, de qual usuário partiu a doação e qual família está recebendo. Não é possível atualizar ou excluir doações da base de dados. O método “updateProfile” da mesma classe é invocado após o registro para atualizar a informação “last_delivery” na tabela “profile”. Novamente, apenas um código é retornado por esta API.

Por fim, “get_user_data” é utilizada para obter informações de usuários, organizações e grupos dentro do território. É utilizado o valor do *header* “Db-Id” para filtrar apenas os resultados que são relevantes para o pedido, e são retornadas três listas contendo os valores. São realizadas consultas nas tabelas “user”, “organization” e “group” do *schema* correto.

4.3.4 Publicação

Para facilitar a obtenção e atualização do aplicativo pelos usuários, foi decidido publicar o aplicativo Cocar na loja Google Play para que pudesse ser facilmente acessado por todos os usuários do sistema operacional móvel Android. Optou-se por não publicar o aplicativo para a plataforma iOS, primeiramente porque o publico alvo do sistema possui majoritariamente Android, mas também porque isso atrasaria o lançamento do aplicativo consideravelmente e acarretaria em custos adicionais que não seriam justificados.

Uma conta Google foi criada para o projeto, utilizada para cadastrar-se como desenvolvedor na plataforma Google Play Console e seguir com a publicação do aplicativo. Os lançamentos são realizados enviando uma nova versão do aplicativo assinado com uma

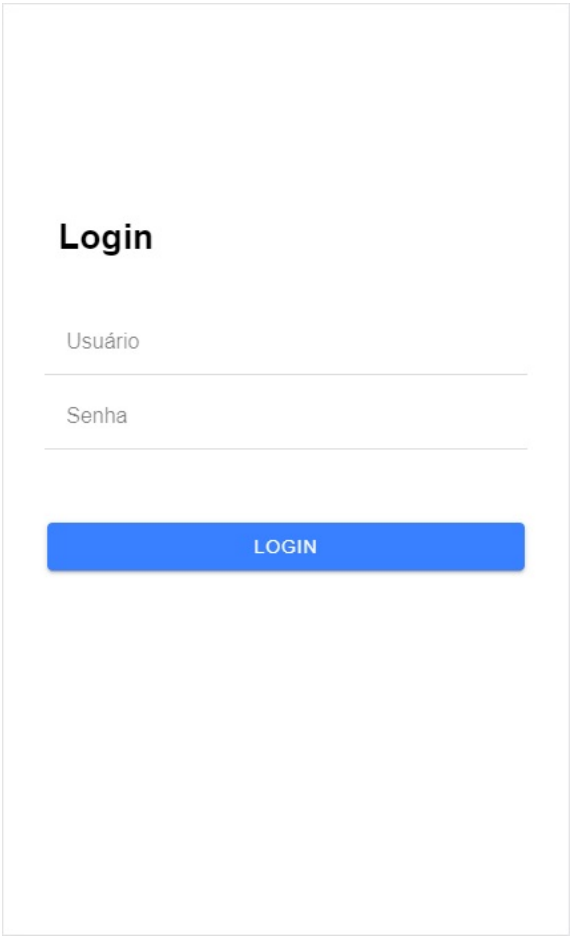
chave de *upload* própria, que deve ser guardada seguramente, pois a perda desta impede novos envios até que seja criada uma nova chave junto ao suporte do Google.

5 PRODUTO FINAL

Este capítulo é dedicado a apresentar a aplicação disponibilizada aos usuários do aplicativo Cocar. O capítulo é dividido em seções que visam cobrir as diferentes funcionalidades da ferramenta de forma a sinalizar as principais funcionalidades implementadas.

5.1 Login

Figura 5.1: Tela de login.



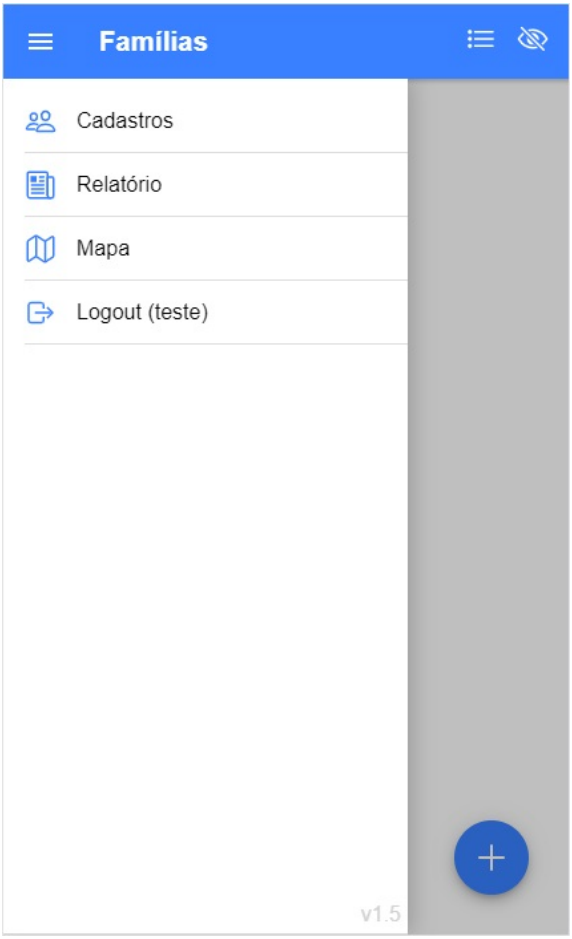
Fonte: O Autor

Apenas os colaboradores dos coletivos contribuintes são autorizados a editarem e consultarem dados das famílias cadastradas. Por isso, a primeira tela do aplicativo é a de autenticação de usuário, cuja interface é retratada na Figura 5.1. O usuário deve inserir nome de usuário e senha previamente cadastrados na base de dados para acessar os recursos do aplicativo. O login deve ser feito uma vez com sucesso, e após isso o aplicativo não irá mais exigir que seja feito, até que o usuário explicitamente faça log-out.

Enquanto estiver logado, o sistema irá apenas realizar uma verificação junto ao banco de dados, confirmando que o usuário ainda está marcado como “ativo”. Caso não esteja, a tela de login é apresentada novamente para que o usuário possa tentar acessar com outro nome de usuário.

5.2 Layout do Aplicativo

Figura 5.2: Tela principal com o menu aberto.

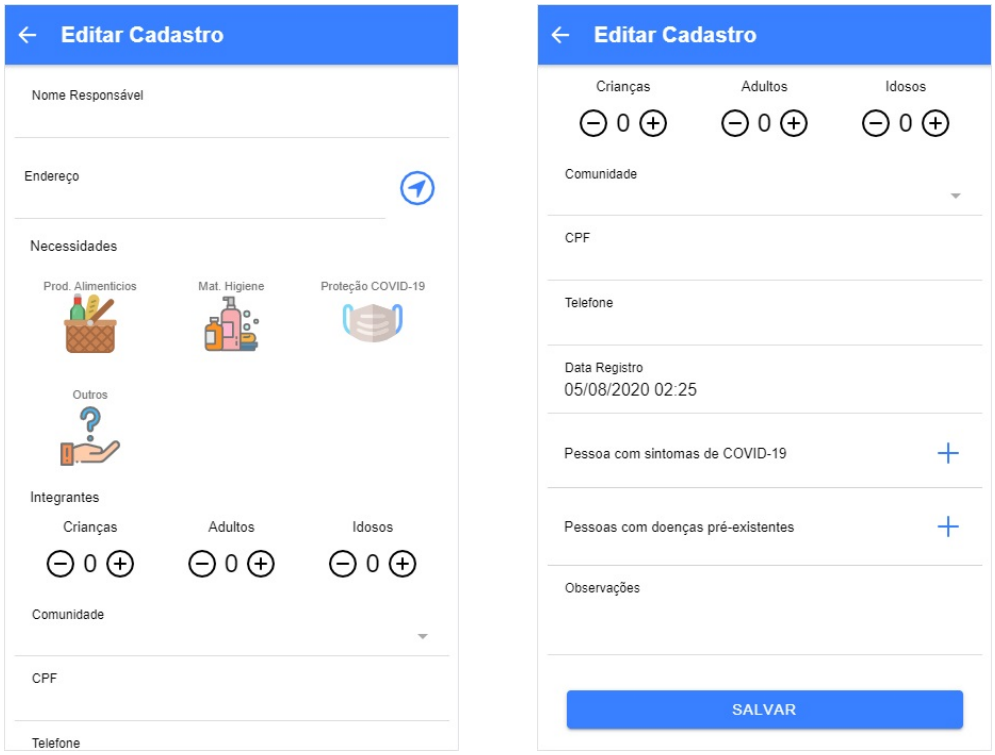


Fonte: O Autor

O aplicativo possui 3 recursos principais: Cadastros, Relatório e Mapa. Para navegar entre estas funcionalidades, o aplicativo conta com um menu lateral que pode ser acessado por um botão na barra superior, como pode-se observar na Figura 5.2. Nesse menu também é possível acessar a opção de log-out para remover as informações do usuário atual. Por padrão, o aplicativo é iniciado na primeira opção: Cadastros.

5.3 Formulário de Cadastro

Figura 5.3: Formulário de cadastro de família.
(a) Parte 1 **(b)** Parte 2



Fonte: O Autor

Na tela de cadastros há um botão no canto inferior direito para adicionar um novo cadastro. Tocando nesse botão, o usuário é levado a tela do formulário de entrada de dados da família, que é explicitado em duas partes na Figura 5.3. A seguir, explica-se como é a utilização esperada de cada campo e funcionalidade dentro da tela de cadastro de família.

Campos obrigatórios: Os seguintes campos devem obrigatoriamente ser informados para que o registro da família seja devidamente efetivado. Na falta de qualquer uma dessas informações, uma mensagem de alerta será exibida na tentativa de salvar, e o cadastro não será enviado.

Nome Responsável: Nome e sobrenome da pessoa responsável pelo contato da família com o coletivo que a estiver cadastrando.

Endereço: Endereço da moradia da família como normalmente referenciado pelos habitantes da comunidade. É importante notar que esse endereço nem sempre corresponderá com os endereços de registro, podendo até mesmo se situar em ruas e vielas não presentes na cartografia oficial do município.

Necessidades: Esse campo possui opções pré-prontas cuja seleção é feita tocando nos ícones da tela. As opções vieram da discussão com os representantes do coletivo União de Vilas sobre as demandas mais comuns encontradas nas comunidades da região. A opção "Outros", quando selecionada, exibe um campo de texto que permite ao usuário explicitar uma demanda particular daquela família.

Integrantes: Neste campo o usuário deve informar quais os grupos etários que compõem a família. Através dos botões de mais e menos ele informa quantas crianças, adultos e idosos essa família é composta. Essas informações auxiliam os coletivos nas decisões que tocam a distribuição dos seus recursos entre as famílias cadastradas.

Comunidade: Esse campo proverá uma lista ao usuário de comunidades previamente cadastradas na região onde o usuário atua. O propósito deste campo é permitir filtrar os cadastros por micro-comunidades dentro das macro-regiões onde os coletivos atuam.

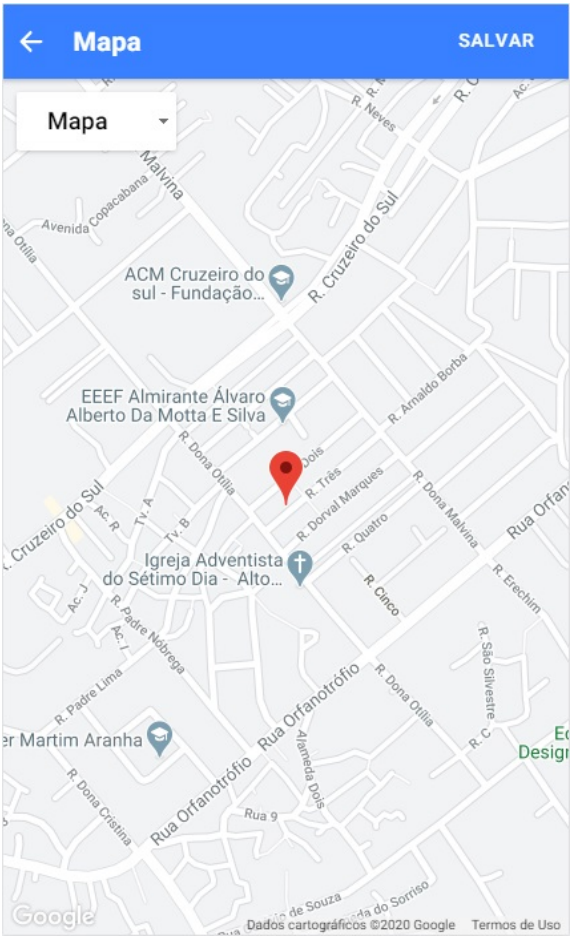
Campos opcionais: Os seguintes campos não são necessários para a conclusão do formulário, mas são informações úteis que agregam conhecimento aos coletivos sobre as famílias.

Localização: A direita do campo Endereço há um botão com ícone de bússola que permite registrar a localização geográfica da família, de forma independente do endereço digitado. Ao tocar este botão, o usuário é levado à uma tela com o GoogleMaps, centralizado em sua posição atual obtida pelo GPS de seu *smartphone*, tal como exposto na Figura 5.4. O usuário pode deslocar manualmente o marcador para melhor refletir a localização da família, caso o GPS não esteja aferido. Ao entrar novamente em um cadastro existente e tocar no botão de Localização, o usuário é capaz de ver a localização salva previamente no mapa, e pode atualiza-lá se for necessário. Embora o endereço seja uma informação obrigatória, a localização não é, por depender de ferramentas de mapa que podem estar indisponíveis para um usuário utilizando o aplicativo de maneira *off-line*.

CPF: Campo que espera os 11 dígitos do CPF do responsável pela família (a mesma pessoa cujo nome foi inserido). O campo pode ser deixado em branco, ou receber um número de CPF válido. A entrada de um número inválido ou incompleto acarretará na exibição de um alerta se o usuário tentar salvar, para evitar que dados incorretos não possam chegar até a base de dados.

Telefone: Um telefone da família, para que o coletivo possua um meio de contato com a mesma e possa repassar qualquer comunicado que veja como relevante. O campo espera código de área (insere os parênteses automaticamente) e tem tamanho máximo de

Figura 5.4: Tela de cadastro de localização da família.



Fonte: O Autor

11 dígitos.

Data de Registro: Esse é um campo de entrada de data e hora que já vem automaticamente preenchido com a data e hora atual. Ele permite que o usuário, caso esteja registrando no aplicativo uma família que visitou previamente, insira uma data anterior à atual como hora do registro e mantenha a informação de quando a demanda da família foi recebida pelo coletivo correta. Isso é importante pois a data em que a demanda foi feita é utilizada como parâmetro na ordem de distribuição dos recursos da maioria dos coletivos.

Pessoa com Sintomas de COVID-19: Esse campo permite o cadastro de informações sobre integrantes da família que estejam apresentando sintomas relacionados ao COVID-19. É uma informação valiosa para os coletivos, que além da distribuição de doações, também promovem ações públicas junto aos postos de saúde da região. Esse campo abre uma nova tela onde são requisitadas três informações: Nome, idade e a seleção de sintomas entre as opções “Faltar de ar” e “Qualquer quadro gripal ou febre persistente”, como exposto na Figura 5.5 De acordo com o sintoma escolhido, uma orientação é repas-

Figura 5.5: Tela do cadastro de pessoa com sintomas de COVID-19.

Fonte: O Autor

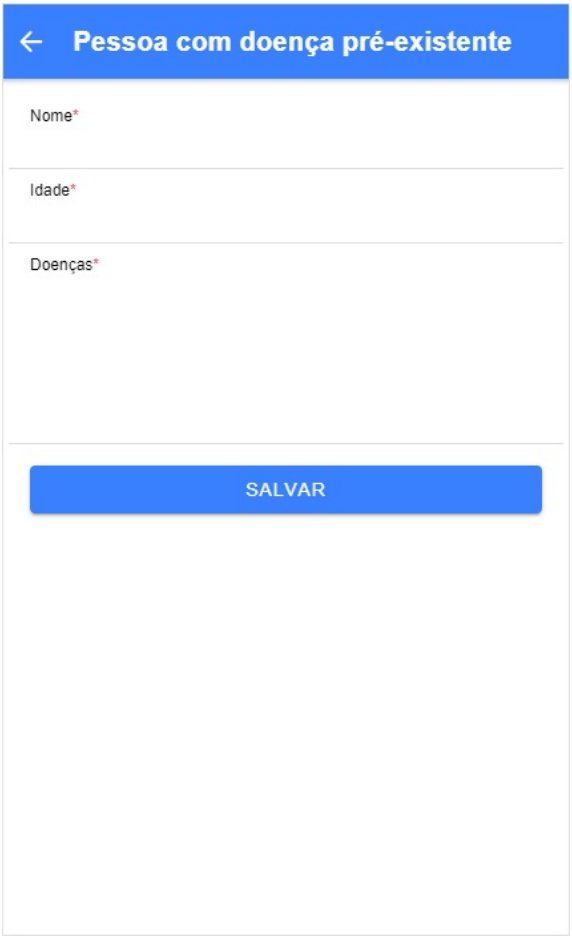
sada ao usuário para que este informe a família.

Pessoa com doença(s) pré-existente(s): Possibilita o cadastro de informações sobre integrantes da família que tenham alguma doença crônica ou problema de saúde anterior relevante. A tela exibida por este campo pede três informações: Nome, idade e doenças (campo de texto onde o usuário pode escrever de maneira livre sobre os problemas daquela pessoa), como demonstrado na Figura 5.6.

Observações: Campo de texto que permite ao usuário informar qualquer outra informação que julgar relevante sobre a família.

Após preencher o formulário de maneira correta, o usuário toca no botão Salvar, que enviará as informações para o servidor e criará as devidas novas entradas no banco de dados. O usuário será então redirecionado à tela da lista de cadastros das famílias, que será atualizada e incluirá a família recém registrada.

Figura 5.6: Tela do cadastro de pessoa com doenças pré-existentes.



Fonte: O Autor

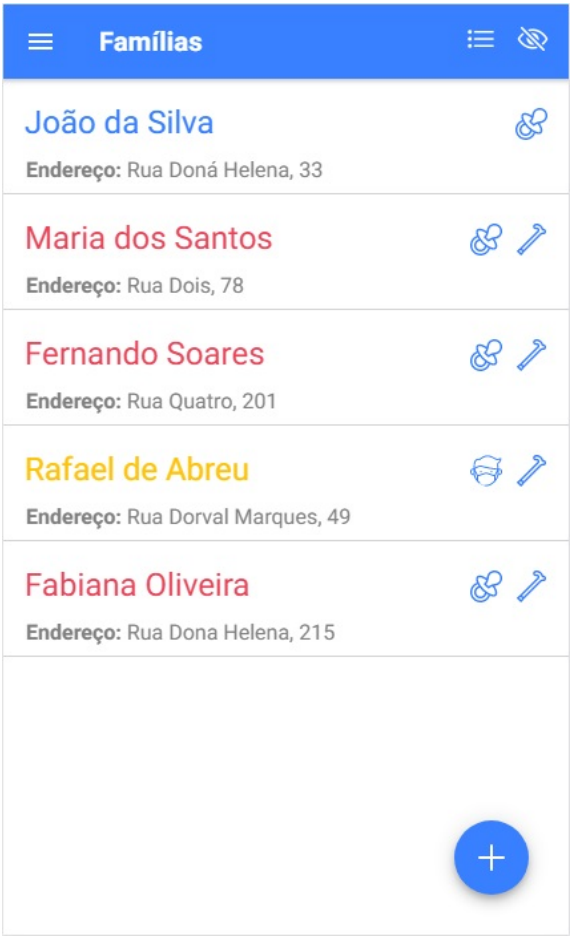
5.4 Lista de Cadastros

A tela da lista de cadastro exibe as famílias identificadas pelo nome do responsável registrado no cadastro. Abaixo do nome, é exibido o endereço de residência informado para a família. Além disso, existem três ícones que podem aparecem a direita do nome que informam se a família possui crianças (ícone de chupeta), idosos (ícone de bengala), ou pessoas que apresentam sintomas de COVID-19 (ícone de pessoa com máscara). O nome do responsável na lista de cadastros pode ser exibido em três possíveis cores, que representam o status do recebimento de doações da família. As cores são as seguintes:

- **Vermelho**: significa que a família ainda não recebeu nenhuma doação, ou recebeu doações a mais de 30 dias.

- **Amarelo**: a família recebeu doação nos últimos 30 dias, mas os produtos entregues não supriam todas as necessidades selecionadas no perfil da família.

Figura 5.7: Tela da lista de cadastros com alguns exemplos.



Fonte: O Autor

- **Azul**: a família recebeu doação que contemplava todas as suas necessidades nos últimos 30 dias.

A Figura 5.7 possui algumas famílias cadastradas como exemplos, não representando dados de famílias reais, e serve o propósito de evidenciar as características descritas nesta seção.

Na barra superior da tela da lista de cadastros existem dois botões juntos a margem direita. O primeiro (mais a esquerda), ao ser tocado, exibirá uma lista de opções de exibição da lista. Opções de exibição:

Nome: organiza a lista de cadastros por ordem alfabética a partir do nome do do membro responsável pela família. Data do Registro: ordena pela data e hora de criação do cadastro. Importante notar que essa hora nem sempre é a hora que o cadastro chegou de fato no banco de dados, mas sim a hora que foi preenchida no campo “Data de Registro” do formulário de criação.

Por Comunidade: utilizando essa opção, os cadastros ficaram agrupados de acordo

com a comunidade em que pertencem. As comunidades registradas aparecerão em ordem alfabética, bem como as famílias dentro delas.

Por Ref. Local: a opção Por Ref. (referência) Local funciona de maneira semelhante à opção anterior, mas agrupando por usuários do aplicativo. Assim, é possível saber quais cadastros foi criados por cada usuário.

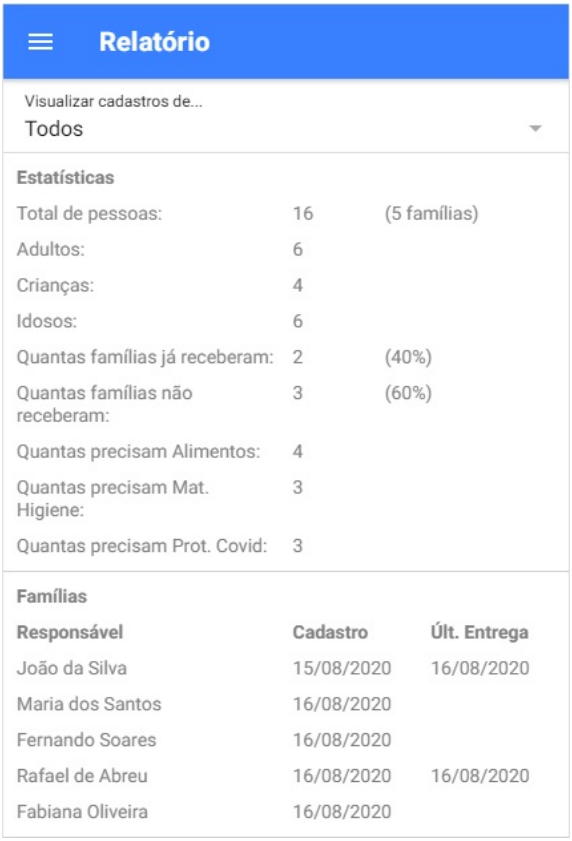
Prioridades: ordenará a lista de registro por uma ordem que indica quais as famílias devem ser contempladas a seguir. Em primeiro aparecem as famílias que foram registradas a mais tempo e ainda não receberam nenhuma doação. Após, aparecem as famílias que já receberam a doação, por ordem de quem recebeu a mais tempo. Nessa opção, um número é exibido ao lado do cadastro indicando qual a posição da família na lista de espera. É importante salientar, entretanto, que os coletivos não se comprometem a utilizar de fato essa lógica na escolha de doações, e podem utilizar outros dados que criem agravantes na situação da família na hora de decidir os destinos de suas doações.

O segundo botão, com o ícone de um olho, serve para alternar a exibição da lista entre todos os cadastros, ou exibir apenas os cadastros que foram criados pelo usuário. A finalidade dessa opção é disponibilizar aos usuários uma maneira mais fácil e prática de encontrar as famílias pelas quais são responsáveis. É importante notar que, embora os cadastros de todos os usuários sejam visíveis para todos os demais usuários, a atualização dos dados destes registros não podem ser realizados por ninguém além do usuário que criou o cadastro. Ao entrar em um cadastro que não é seu, um usuário será capaz de ler todas as informações, incluindo a localização geográfica no mapa, mas não será capaz de alterar nenhuma delas. Essa escolha visa evitar a inserção de informações errôneas.

5.5 Relatório

A tela de Relatório exibe de forma concisa algumas informações que refletem a situação das famílias cadastradas em geral. Nessa tela é possível verificar quantas famílias estão registradas no aplicativo, e por quantas pessoas essas famílias são compostas, assim como discriminar entre crianças, adultos e idosos. Também pode-se obter a informação de qual o percentual de famílias registradas que já receberam doações e quantas ainda estão esperando. Além disso, é exibida uma lista de todas as famílias registradas visualizando apenas nome, data de cadastro e data da última entrega de doações para aquela família. A Figura 5.8 demonstra como a tela de relatório exibe os dados das famílias utilizadas como exemplos na seção anterior.

Figura 5.8: Tela de relatórios.



Fonte: O Autor

5.6 Mapa

Nessa opção, podemos visualizar a localização geográfica de todas as famílias cuja localização foi salva por um usuário do aplicativo. Ao abrir, a tela exibira o mapa centralizado na posição atual do usuário, obtida pelo GPS do *smartphone* do usuário. As localizações das famílias no mapa são representadas por *pins*, cuja cor segue os mesmos critérios da cor do nome do responsável na lista de cadastros, como é possível observar na Figura 5.9. Dessa forma, espera-se que os usuários sejam capazes de identificar de maneira intuitiva quais são as comunidades que mais necessitam doações no momento. Ao tocar em um *pin*, um balão é exibido contendo algumas informações da família, como nome do responsável e endereço inserido manualmente.

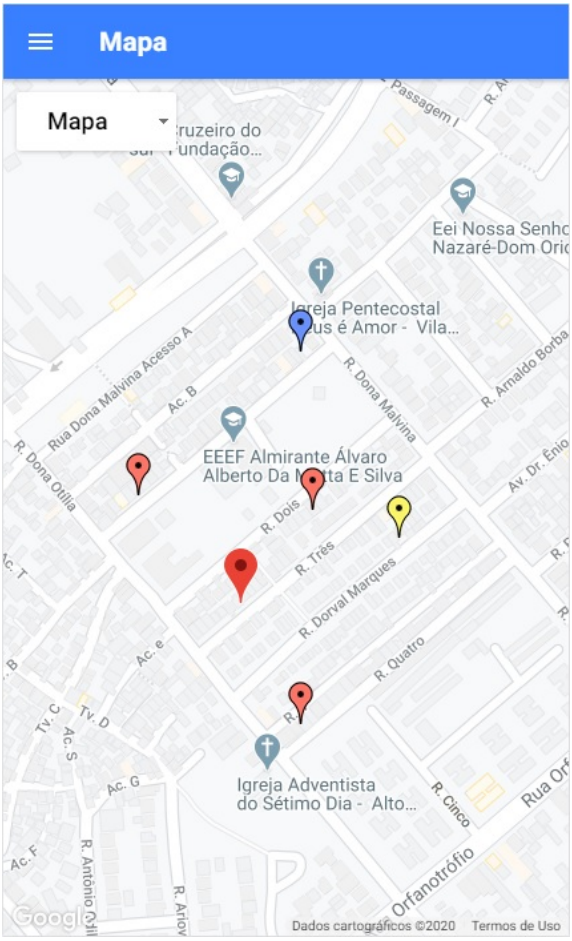
No canto superior esquerdo do mapa, há um botão que permite ao usuário alternar entre visualizar o mapa ou as imagens de satélite do local. As imagens reais obtidas por satélite podem auxiliar na identificação da residência correta e tornar as informações inseridas no aplicativo mais precisas.

Para que o mapa não fique poluído com informações desnecessárias, o aplicativo

automaticamente filtra os marcadores de negócios próprios na área.

Essas configurações estão presentes também na tela de registro de localização de uma família.

Figura 5.9: Tela de visualização geográfica dos cadastros.



Fonte: O Autor

6 EXPERIMENTOS

Este capítulo é dedicado a apresentar os experimentos realizados com usuários. Duas circunstâncias distintas foram importantes para avaliar a eficácia e aceitação do aplicativo Cocar. Primeiro, é descrita a realização presencial de uma capacitação para os usuários finais reais do aplicativo utilizando a primeira versão do mesmo. E por fim, é descrito o experimento realizado após o encerramento da etapa de desenvolvimento, que contou com a participação de usuários reais e participantes voluntários para avaliar o grau de dificuldade percebido na realização de tarefas propostas no aplicativo.

6.1 Capacitação

Com a finalização da versão de lançamento do aplicativo, foi realizado um encontro presencial para capacitar os primeiros usuários, bem como observar quais as reações, dúvidas e sugestões os mesmos teriam. Os primeiros usuários reais do aplicativo Cocar foram escolhidos pelos coordenadores do coletivo União de Vilas. Tratavam-se de quatro moradores locais, que anteriormente já exerciam posições de liderança na comunidade e atuavam como ponte entre as famílias e a União de Vilas. O cadastramento destes usuários no sistema se deu de forma manual pelo autor deste trabalho, utilizando inserções diretamente na base de dados MySQL. Os nomes de usuário e senha foram escolhidos pelos usuários e informados ao autor de maneira privada em conversas particulares com cada um. Foi utilizado um projetor para exibir *slides* com capturas de tela do aplicativo juntamente com instruções de uso para cada funcionalidade criada. Os usuários acompanharam a apresentação com seus *smartphones* em mãos e foram encorajados a seguir as instruções em tempo real para os processos de: *login*, criação de um cadastro, edição de um cadastro, registro de uma entrega de doação, reordenação da lista de cadastros, visualização de relatórios e *log-out*. Nesta etapa, ainda não haviam sido implementadas as funcionalidades relacionadas a localização geográfica das famílias.

Os usuários não encontraram dificuldades em seguir o passo-a-passo da apresentação, que ocorreu sem interrupções significativas. Após o termino da apresentação, uma conversa informal diretamente com os usuários levantou alguns pontos onde foi observado que o aplicativo poderia ser ajustado e melhorado, como por exemplo: possibilitar a escolha de data anterior na tela de registro de entrega de doações.

6.2 Experimento de Usabilidade

Quando foi considerada finalizada a fase de desenvolvimento pertinente a este trabalho, foi realizado um teste de usabilidade. A princípio, os usuários escolhidos para a realização deste teste seriam exclusivamente os mesmos que já possuíam acesso e estavam utilizando o aplicativo com regularidade. Entretanto, este se tratava de um grupo bastante reduzido de pessoas, e como algumas destas não participaram, o experimento foi realizado também com pessoas fora do contexto dos movimentos sociais de forma a obter um volume maior de dados para análise.

Para o experimento foi utilizada uma metodologia baseada no “System Usability Scale” (SUS) (BROOKE, 1996), criado pelo britânico John Brooke em 1986. Este método consiste em utilizar questionários com perguntas de escolha única com 5 opções de respostas, que representam uma escala iniciando em "Discordo fortemente"até "Concordo fortemente", com a resposta mediana significando neutralidade.

6.2.1 Protocolo de Testes

Foi elaborado um formulário (vide Apêndice A) através do *Google Forms*, ferramenta que permite fácil elaboração e acesso de formulários. Este formulário contava com instruções para a realização de atividades antes de todas as perguntas, que indagavam acerca da dificuldade percebida pelos usuários na execução destas atividades.

As instruções do formulário não contavam com instruções de utilização do aplicativo, pois ele foi inicialmente desenvolvido apenas pensando nos usuários que já haviam passado por uma capacitação. Para que não houvesse necessidade da criação de um novo formulário para o segundo grupo de pessoas, estas receberam instruções gerais de uso do aplicativo antes do começo do experimento. Foi realizada uma demonstração, ao vivo, para este grupo, de como utilizar os recursos do aplicativo, bem como fornecido um login e senha criado exclusivamente para a utilização durante o experimento.

Os usuários acessaram o formulário do experimento através de um link enviado de forma pessoal pelo autor. Para cada usuário, foram realizadas duas etapas:

- **Formulário de pré-teste:** são feitas perguntas para caracterização do usuário como "idade", "nível de escolaridade", "grau de experiência com aplicativos móveis"e "grau de experiência com Internet";

- **Formulário de teste:** o usuário é orientado a executar atividades e questionado sobre a usabilidade da funcionalidade que acabou de utilizar. Esta metodologia foi escolhida por possibilitar um experimento completamente remoto de fácil entendimento, que não obrigasse o usuário a fazer todas as tarefas antes do questionário, pois acreditou-se que o usuário poderia se confundir e se cansar mais desta forma.

  As tarefas solicitadas foram:

  - 1. Utilizar nome de usuário e senha previamente cadastrados para realizar o log-in no sistema;
  - 2. Abrir um novo cadastro de família;
  - 3. Salvar uma localização geográfica para esta família, utilizando a ferramenta de mapa implementada;
  - 4. Cadastrar um morador da família com sintomas de COVID-19;
  - 5. Cadastrar um morador da família com doença(s) pré-existente(s);
  - 6. Finalizar e salvar o cadastro com todas as informações básicas obrigatórias;
  - 7. Encontrar o cadastro recém salvo utilizando os métodos de reordenação oferecidos;
  - 8. Registrar uma doação para a família;
  - 9. Excluir o cadastro da família;
  - 10. Realizar o log-out do sistema.

6.2.2 Perfil dos participantes

Os resultados obtidos foram separados em dois grupos: o montante total com as respostas enviadas por todos os participantes, e o sub-grupo contendo apenas os usuários reais cadastrados no sistema. Uma análise foi realizada sobre o resultado geral, mas através da análise dos gráficos, é possível perceber que os resultados obtidos apenas pelo sub-grupo dos usuários reais segue a mesma tendência e pouco difere da soma.

Todos os participantes assinalaram estarem de acordo com a participação do experimento e estarem cientes do uso dos resultados de forma anônima neste trabalho.

Participaram ao todo 18 pessoas, com idades variando entre 26 e 67 anos, sendo 43 a média das idades de todos os participantes. Destes, 5 eram usuários reais do aplicativo

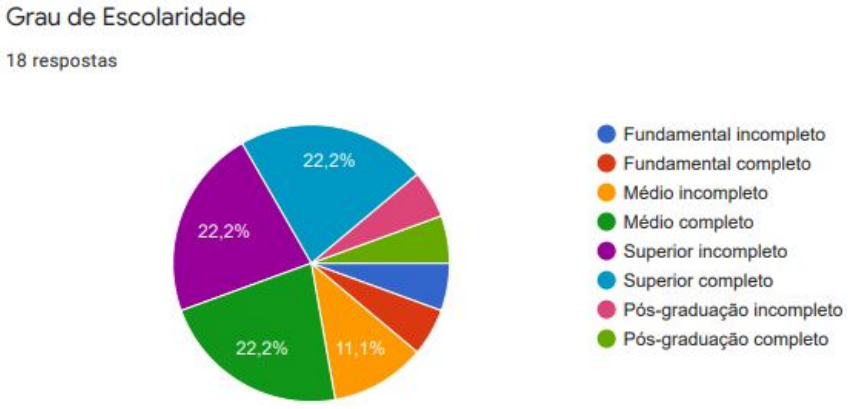
membros dos movimentos sociais, com idades entre 41 e 59, a média deste sub-grupo estabelecendo-se um mais elevada, em 48,8 anos.

A Figura 6.1 apresenta o grau de escolaridade dos participantes, com o resultado tendo participações em todas as faixas, e a maior representação sendo principalmente em formados no ensino médio, superior, e pessoas com graduação incompleta. Dos 5 participantes que são usuários reais, 2 possuem nível superior incompleto, 1 possui médio completo, 1 possui fundamental incompleto e 1 possui pós-graduação incompleta. Não foi feita separação entre grande grupo e usuários reais neste gráfico por acreditar-se que os resultados estão bem distribuídos e não são motivos de influência nos resultados da usabilidade.

Foi questionado também o grau de experiência que o participante possui com serviços de internet. Como é possível observar na 6.2, a maioria dos participantes, tanto no grande grupo como apenas os usuários reais, sentiram-se confortáveis em apontar 4 como seu nível de experiência.

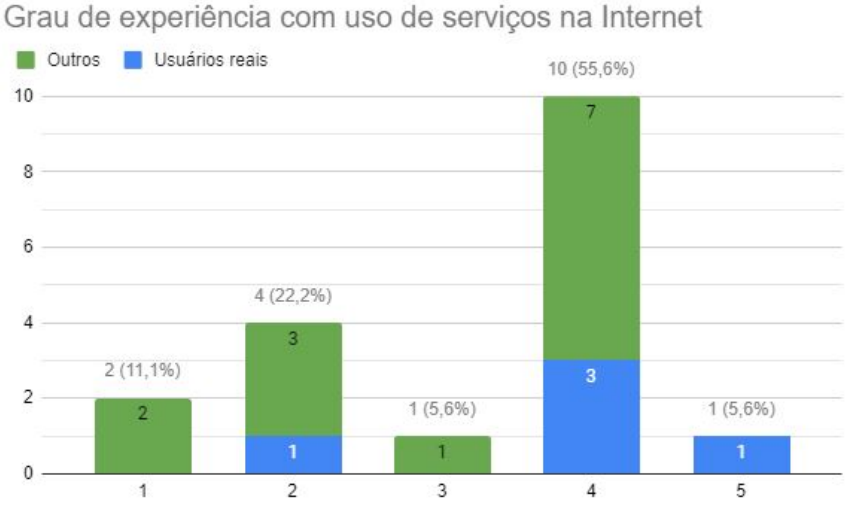
O grau de experiência dos participantes com aplicativos para celular ficou com uma média maior, como mostra a 6.3. É possível inferir pela comparar destes dois últimos gráficos, que o uso de aplicativos para dispositivos móveis foi considerado algo mais comum na vida dos participantes. É possível que a pergunta sobre a experiência com serviços de internet tenham sido interpretadas como o acesso à internet a partir de um computador pessoal, criando então essa diferença nos resultados.

Figura 6.1: Grau de escolaridade dos participantes.



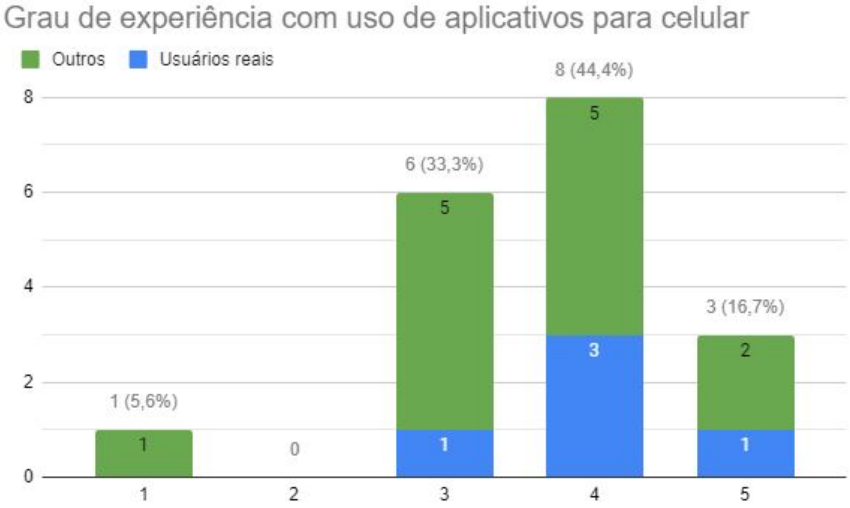
Fonte: Autor

Figura 6.2: Grau de experiência com internet dos participantes.



Fonte: Autor

Figura 6.3: Grau de experiência com aplicativos dos participantes



Fonte: Autor

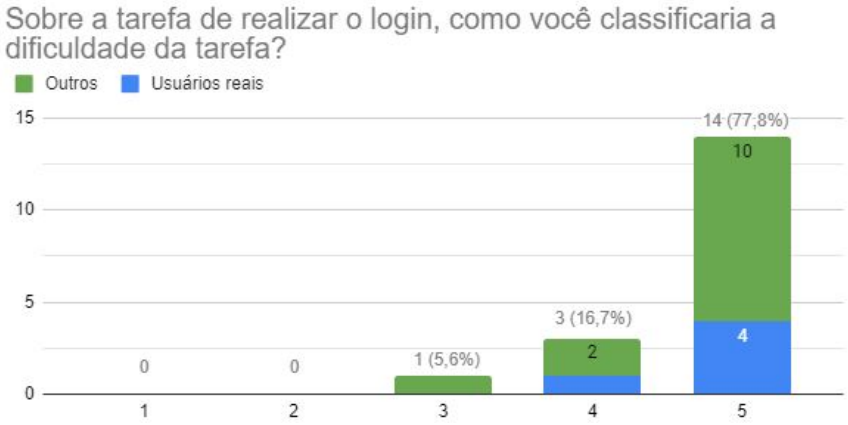
6.2.3 Usabilidade

Nesta seção, serão analisados os resultados obtidos da porção do formulário que tratava da usabilidade do aplicativo Cocar. Para cada uma das tarefas solicitadas, uma questão de escala linear era proposta perguntando o nível de dificuldade encontrado pelo participante ao realizá-la. Em todas as questões dessa seção do formulário, o valor 1 corresponde a "Muito difícil" e 5, "Muito fácil". Para realizar as análises, além das notas recebidas, também é utilizada a experiência do autor com a reação dos participantes ao aplicativo no momento da capacitação, tanto dos usuários reais como aquela realizada

com o restante dos participantes.

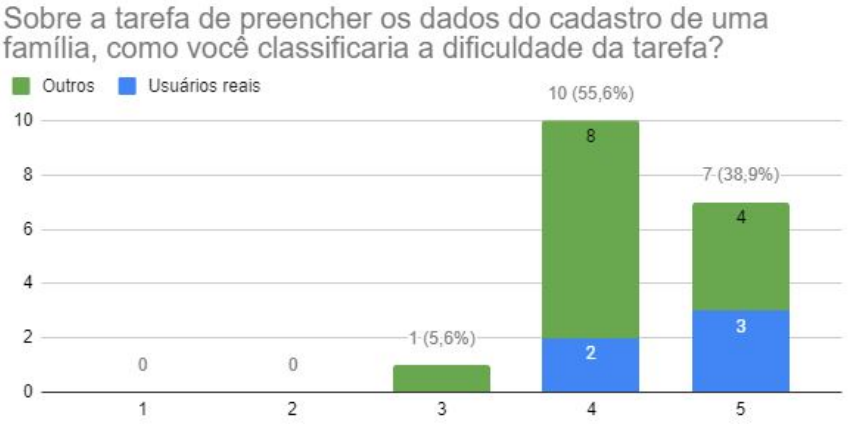
A tarefa de login foi considerada simples, como podemos observar na Figura 6.4. A maioria dos participantes atribuiu a nota 5, com apenas um caso onde foi encontrada maior dificuldade e foi utilizada uma nota média. Esta dificuldade, acredita-se, foi por uma questão de falta de conexão com a internet experienciada pelo participante no momento deste teste.

Figura 6.4: Dificuldade dos participantes na realização da tarefa de login.



Fonte: Autor

Figura 6.5: Dificuldade dos participantes na realização da tarefa de preencher o cadastro de família.

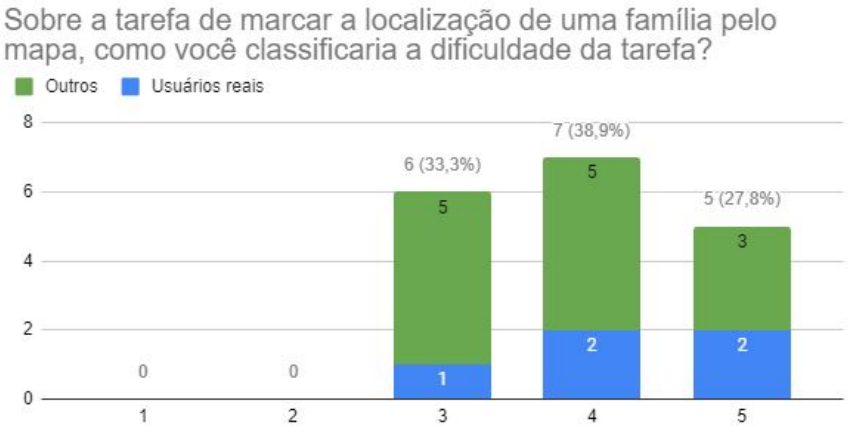


Fonte: Autor

Pode-se dizer que a tarefa em que os participantes apresentaram a maior dificuldade foi a que solicitava a marcação de uma localização utilizando o mapa. Um terço dos participantes marcou a nota 3 por terem dificuldades com o mapa, que apresentou lentidão para carregar em alguns dispositivos e teve que ser reaberto em algumas situações para dar prosseguimento da tarefa. O experimento possibilitou a percepção desta falha no aplicativo, que deverá ser corrigida no futuro. No entanto, a maneira com que a ferra-

menta foi implementada parece ter sido satisfatória para os participantes, que entenderam rapidamente o modo de operação do mapa para seleção de um local.

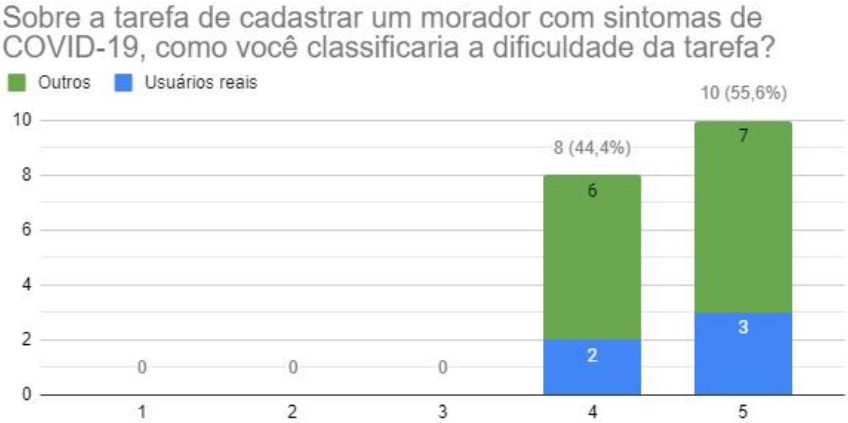
Figura 6.6: Dificuldade dos participantes na realização da tarefa de marcar localização da família.



Fonte: Autor

As tarefas de cadastrar moradores específicos, com sintomas de COVID-19 ou com doenças pré-existentes tiveram notas semelhantes, como já era esperado pela maneira similar de execução. Os resultados podem ser observados nas Figuras 6.7 e 6.8, respectivamente.

Figura 6.7: Dificuldade dos participantes na realização da tarefa de registrar um morador com sintomas de COVID-19.

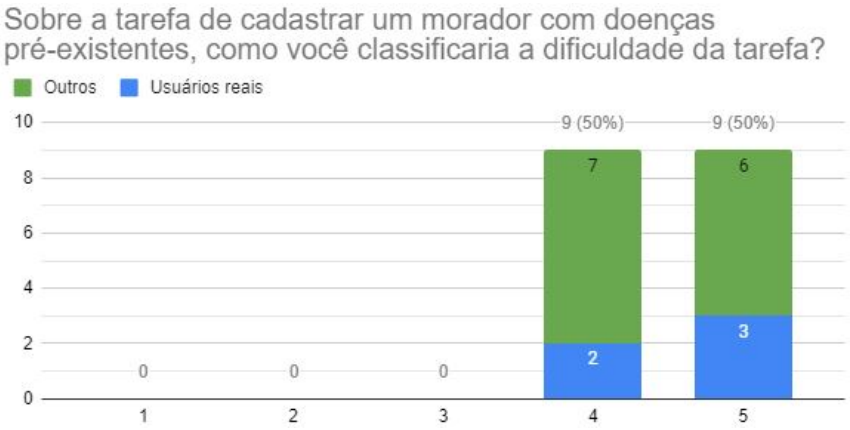


Fonte: Autor

As tarefas consideradas mais triviais pelos participantes foram as de salvar um cadastro já preenchido e de excluir um cadastro de família. Como pode ser observado nas Figuras 6.9 e 6.10, a maioria dos participantes sentiu-se confortável em atribuir a nota máxima de facilidade para estas atividades.

A Figura 6.11 ilustra a dificuldade dos participantes em encontrarem um cadastro na lista geral de famílias cadastradas. Assim, como acontece com a tarefa de log-out (Fi-

Figura 6.8: Dificuldade dos participantes na realização da tarefa de registrar um morador com doenças pré-existentes.



Fonte: Autor

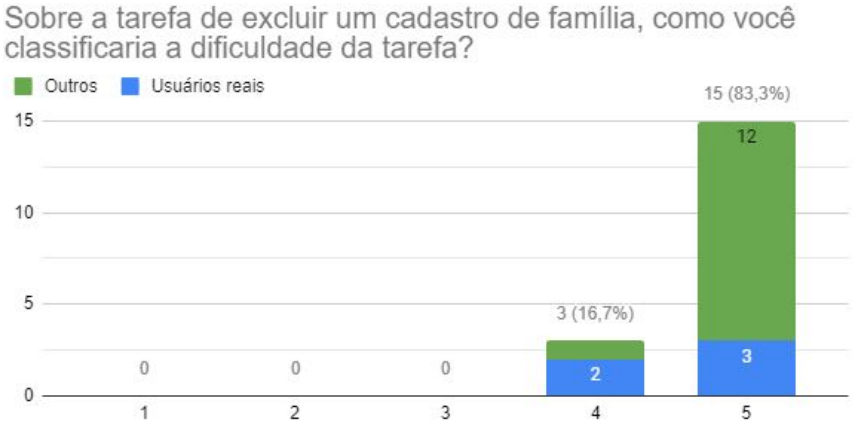
Figura 6.9: Dificuldade dos participantes na realização da tarefa de salvar o cadastro de uma nova família.

Fonte: Autor

gura 6.13), os participantes acharam a atividade “fácil” e “muito fácil”, e não encontraram maiores dificuldades.

Na Figura 6.12 é possível observar que a tarefa de registrar uma doação para uma família também apresentou algumas dificuldades para uma pequena parcela dos participantes. Embora não tenham sido relatados problemas de natureza técnica, como no caso da tarefa de marcação da localização, alguns participantes não acharam a abordagem a mais apropriada.

Figura 6.10: Dificuldade dos participantes na realização da tarefa de excluir o cadastro de uma família.

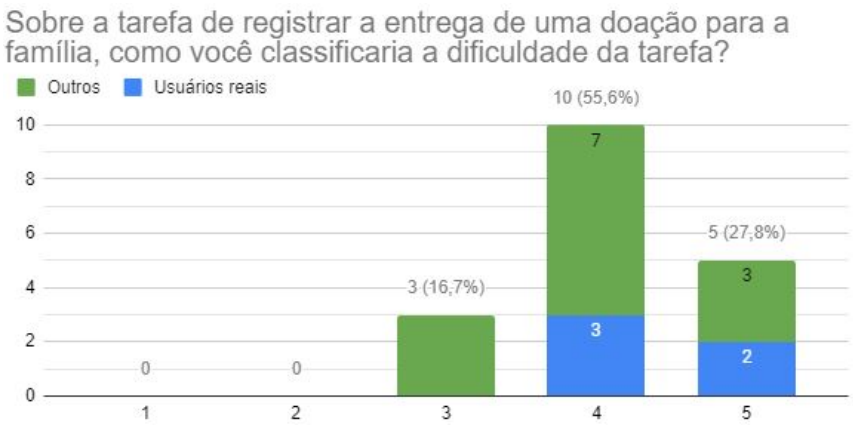


Fonte: Autor

Figura 6.11: Dificuldade dos participantes na realização da tarefa de encontrar uma família na lista de cadastros.

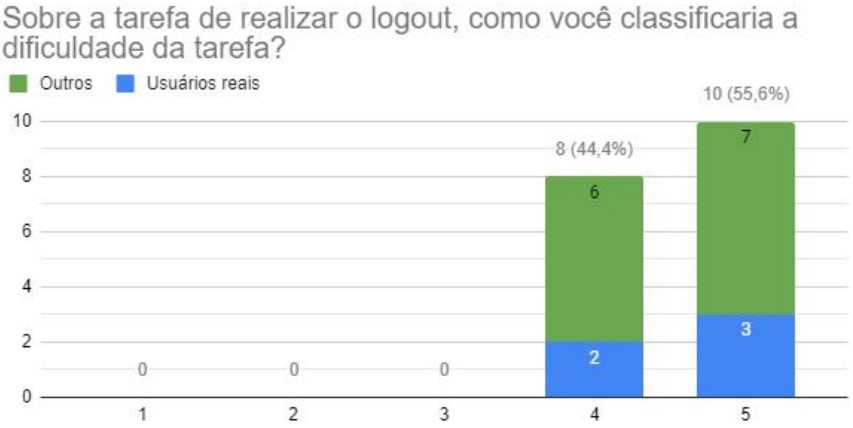
Fonte: Autor

Figura 6.12: Dificuldade dos participantes na realização da tarefa de registrar uma doação para uma família.



Fonte: Autor

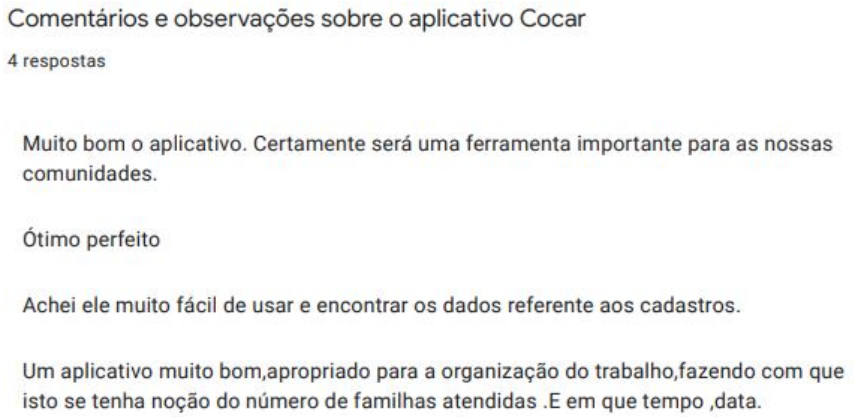
Figura 6.13: Dificuldade dos participantes na realização da tarefa de realizar o log-out do aplicativo.



Fonte: Autor

Por fim, a Figura 6.14 contém as observações feitas pelos participantes sobre o aplicativo Cocar, escritas no campo de texto ao final do formulário destinado a este fim. Todos os comentários são de participantes integrantes do grupo de usuários reais, que de fato utilizaram o aplicativo no cadastramento de famílias da região da Vila Cruzeiro e da Glória. O fato de apenas este grupo ter escrito comentários é provavelmente explicado pelo fato de que o restante dos participantes teve oportunidade de fazer observações pessoalmente, no dia do experimento.

Figura 6.14: Comentários dos participantes sobre o aplicativo Cocar.



Fonte: Autor

6.3 Análise Geral dos Resultados e Limitações

Como foi objetivado durante toda a fase de desenvolvimento, os resultados mostram que o aplicativo resultante tornou-se intuitivo e de fácil manuseio. O aplicativo foi

muito bem recebido pelos usuários reais, que em diversas oportunidades salientaram a utilidade observada para a realização do trabalho de cadastramento de famílias. Foi chamada a atenção para a facilidade de se encontrar uma cadastro na lista de famílias existentes. Acredita-se que esta avaliação é creditada tanto aos diversos modos de reordenamento da lista quanto ao simples fato de o aplicativo reunir todos os dados em uma base de dados centralizada.

O aplicativo ainda possui limitações que dificultam a utilização de algumas funcionalidades e tornam outras levemente confusas. Um dos pontos mais importantes é a dificuldade de alguns usuários com o carregamento correto dos mapas, que foi observado como problemático nos experimentos. Foi observada também a necessidade de exibir mensagens de erros mais claramente para os usuários, especialmente para distinguir casos onde o erro resulta de uma interrupção na conexão com a Internet do aparelho utilizado.

7 CONCLUSÃO

Este trabalho teve como proposta desenvolver uma aplicação móvel capaz de auxiliar movimentos sociais no gerenciamento de informações referentes a famílias de regiões periféricas de Porto Alegre.

Para o processo de desenvolvimento foi adotada uma combinação de diferentes metodologias ágeis de desenvolvimento de software que possibilitassem a flexibilidade necessária para o projeto.

Foram realizadas cerca de quinze reuniões *on-line*, contando com professores de disciplinas varias da Universidade, além dos representantes de quatro movimentos sociais atuantes na região da Grande Cruzeiro e Glória.

A versão final disponibilizada, produzida pelos esforços descritos neste trabalho, conseguiu contemplar os requisitos estabelecidos na fase final de encontros virtuais e encontra-se até o momento disponível para a utilização dos usuários registrados. Conforme os resultados do experimento de usabilidade realizado no aplicativo por participantes voluntários e usuários reais, pode-se dizer que as abordagens escolhidas para a resolução dos problemas apresentados pelos requerimentos foi satisfatória.

Como trabalhos futuros, existem diversos nichos em que o aplicativo pode atuar para ajudar ainda mais os coletivos, além de pontos de melhoria para funções já implementadas neste trabalho. A área de georreferenciamento é talvez umas das que tem um maior potencial de expansão, como a implementação das funcionalidades de exibição da demarcação de territórios e também de pontos considerados relevantes na região, como postos de saúde, escolas e centros de lazer.

Embora não tenha sido um requisito inicial para este trabalho, uma ferramenta visual de gerenciamento dos usuários, comunidades e coletivos envolvidos no projeto também será de grande valor ao projeto, principalmente considerando a forte possibilidade de o mesmo vir a tomar maiores proporções. No mesmo sentido, processos automatizados de redefinição de senha e a atribuição de um e-mail pessoal para cada usuário devem trazer facilidades na realização destas tarefas.

REFERÊNCIAS

BECK, K. **Extreme programming explained**: embrace change. [S.l.]: Addison-Wesley, 1999.

BECK, K, et al. The agile manifesto. 2001. Available from Internet: <https://agilemanifesto.org/>.

BROOKE, J. Sus - a quick and dirty usability scale. 1996. Available from Internet: <https://www.taylorfrancis.com/books/e/9780429157011/chapters/10.1201/9781498710411-35>.

FIELDING, R. Architectural styles and the design of network-based software architectures. 2000. Available from Internet: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

Google Team. Google cloud maps plataform. 2020. Available from Internet: <https://cloud.google.com/maps-platform/maps?hl=pt>.

LIKER, J. **The Toyota Way**: 14 management principles from the world's greatest manufacturer. [S.l.]: McGraw Hill, 2004.

PALMER, S. R.; FELSING, M. **A Practical Guide to Feature-Driven Development**. [S.l.]: Pearson Education, 2001.

SCHWABER, K. Scrum development process. **Business Object Design and Implementation**, Springer, London, 1997. Available from Internet: <https://doi.org/10.1007/978-1-4471-0947-1_11>.

STAPLETON, J. **DSDM**: A framework for business centered development. [S.l.]: Addison-Wesley, 1997.

APÊNDICE A — FORMULÁRIO DE TESTE

Formulário Aplicativo Cocar

Este teste tem como objetivo avaliar a experiência de utilização do aplicativo Cocar.

O aplicativo foi desenvolvido como trabalho de conclusão do curso de bacharelado em Ciência da Computação pela Universidade Federal do Rio Grande do Sul. O aplicativo Cocar visa oferecer uma ferramenta de gerenciamento de dados de famílias em situação de risco. O público alvo deste aplicativo são movimentos sociais que trabalham com distribuição de doações às famílias necessitadas.

Procedimentos:

Inicialmente, o usuário responde um questionário pré-teste para coletar informações de caracterização.
Depois, o usuário passa por uma pequena lista de atividades, cada uma seguida imediatamente de uma pergunta sobre a dificuldade da atividade pedida.

O tempo total do experimento é de aproximadamente 15 minutos.

Os dados obtidos ao longo do experimento serão utilizados apenas neste estudo e de forma totalmente anônima.
*Obrigatório

1. Caso você esteja de acordo com este termo, marque a opção abaixo. *

Marque todas que se aplicam.

☐ Aceito participar deste teste. Declaro que fui devidamente informado sobre os objetivos da pesquisa e os procedimentos envolvidos nos testes. Foi-me garantido o sigilo de minhas informações.

Formulário Pré-Teste

2. Idade *

_____

3. Grau de Escolaridade *

Marcar apenas uma oval.

- ◯ Fundamental incompleto
- ◯ Fundamental completo
- ◯ Médio incompleto
- ◯ Médio completo
- ◯ Superior incompleto
- ◯ Superior completo
- ◯ Pós-graduação incompleto
- ◯ Pós-graduação completo

4. Grau de experiência com uso de serviços na Internet

Marcar apenas uma oval.

	1	2	3	4	5	
Pouca	◯	◯	◯	◯	◯	Muita

5. Grau de experiência com uso de aplicativos para celular

Marcar apenas uma oval.

	1	2	3	4	5	
Pouca	◯	◯	◯	◯	◯	Muita

Tarefa 1: Realizar Login

Abra o aplicativo.

Utilizando seu nome de usuário e senha, realize o login no aplicativo (se seu usuário já estiver salvo, por faça o log-out pelo menu e faça o login novamente).

6. Sobre a tarefa de realizar o login, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Cadastrar uma nova família

Você deve estar agora na tela da lista de cadastros das famílias.

Nos próximos passos, vamos criar um novo cadastro aqui para testar a usabilidade. Não se preocupe, você excluirá este cadastro mais adiante neste experimento e não serão deixados dados residuais na lista.

Tarefa 2: Criar novo cadastro

Toque no botão de criar nova família e insira os dados básicos obrigatórios para completar o cadastro. Por favor, ainda não envie o cadastro.

7. Sobre a tarefa de preencher os dados do cadastro de uma família, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 3: Salvar localização

Utilizando o botão ao lado do campo endereço, salve uma localização geográfica no mapa para esta família.

8. Sobre a tarefa de marcar a localização de uma família pelo mapa, como você classificaria a dificuldade da tarefa?

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 4: Morador com sintomas de COVID-19

Cadastre um membro da família com sintomas de COVID-19. Preencha os campos de nome, idade e selecione os sintomas, e a seguir salve o registro desta pessoa para voltar à tela da família.

9. Sobre a tarefa de cadastrar um morador com sintomas de COVID-19, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 5: Morador com doenças pré-existentes

Cadastre um membro da família com doença(s) pré-existente(s). Preencha os campos de nome, idade e quais são as doenças, e a seguir salve o registro desta pessoa para voltar à tela da família.

10. Sobre a tarefa de cadastrar um morador com doenças pré-existentes, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 6: Salvar o cadastro

Salve o cadastro. Se um dado obrigatório estiver faltando, o sistema deve lhe informar exibindo uma mensagem. Se for o caso, por favor preencha o campo indicado na mensagem e tente novamente.

11. Sobre a tarefa de salvar o cadastro, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 3: Registrar uma entrega

Na lista de famílias, encontre a família que você acabou de cadastrar. Quando encontrar, entre neste cadastro e registre uma doação de Produtos Alimentícios para esta família.

12. Sobre a tarefa de encontrar um cadastro já existente, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

13. Sobre a tarefa de registrar a entrega de uma doação para a família, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 4: Excluir um cadastro

Ainda dentro do cadastro, utilize o botão apropriado para excluir esta família da lista de registros.

14. Sobre a tarefa de excluir um cadastro de família, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Tarefa 5: Realizar Logout

Através do menu principal, toque na opção de logout e confirme. Você será redirecionado ao login novamente.

15. Sobre a tarefa de realizar o logout, como você classificaria a dificuldade da tarefa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Muito Difícil	◯	◯	◯	◯	◯	Muito Fácil

Comentários

Utilize essa seção para deixar comentários ou observações acerca do uso do funcionamento do aplicativo Cocar. Esta é uma pergunta opcional.

16. Comentários e observações sobre o aplicativo Cocar