UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

APLICATIVO MOBILE PARA GESTÃO PESSOAL DE DOAÇÕES PARA INSTITUIÇÕES DE CARIDADE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2020

GLAUBER PROCESKI ANDREOLLI

APLICATIVO MOBILE PARA GESTÃO PESSOAL DE DOAÇÕES PARA INSTITUIÇÕES DE CARIDADE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profa. Mainara Cristina Lorencena

PATO BRANCO
2020

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
UTFPR - CAMPUS PATO BRANCO
DIRETORIA GERAL - CÂMPUS PATO BRANCO
DIR. DE GRAD. E EDUCAÇÃO PROFISSIONAL-PB
DEP. ACADÊMICO DE INFORMÁTICA DAINF-PB

TERMO DE APROVAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

Aplicativo Mobile para Gestão Pessoal de Doações para Instituições de Caridade

POR

Glauber Proceski Andreolli

Este trabalho de conclusão de curso foi apresentado no dia 30 de junho de 2020, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas, pela Universidade Tecnológica Federal do Paraná. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Banca examinadora:

Profa. MSc Mainara Cristina Lorencena	**Prof. MSc Vinicius Pegorini**	**Profa. MSc Emanoeli Madalosso**
Professora orientadora	Professor convidado	Professora convidada

Prof. Dr. Edilson Pontarolo	**Profa. Dra. Mariza Miola Dosciatti**
Coordenador do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas	Responsável pela Atividade de Trabalho de Conclusão de Curso

Documento assinado eletronicamente por **MARIZA MIOLA DOSCIATTI, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/07/2020, às 10:18, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **MAINARA CRISTINA LORENCENA, PROFESSOR MAGISTERIO SUPERIOR-SUBSTITUTO**, em 01/07/2020, às 10:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **VINICIUS PEGORINI, PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 01/07/2020, às 10:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **EMANOELI MADALOSSO, PROFESSOR MAGISTERIO SUPERIOR-SUBSTITUTO**, em 01/07/2020, às 12:04, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **EDILSON PONTAROLO, COORDENADOR(A) DE CURSO/PROGRAMA**, em 01/07/2020, às 15:07, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.utfpr.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1503112** e o código CRC **8097C564**.

RESUMO

É comum que instituições de caridade e assistenciais realizem campanhas públicas para que a população em geral, tanto cidadãos quanto empresas, tomem conhecimento de suas necessidades básicas e possam lhes oferecer doações de diversos itens, como alimentos, produtos de limpeza, produtos de higiene pessoal, roupas, auxílio financeiro, entre outros. Obter informações específicas sobre itens faltantes ou primordiais de uma instituição permite uma melhor organização de campanhas para doação e de alocação dos recursos doados, e, consequentemente, um melhor aproveitamento desses recursos. Assim, é importante que os doadores dispostos a auxiliar essas instituições possam ser informados de sua existência e das suas necessidades, que podem ser emergenciais e que, geralmente, são expressas por meio de campanhas de rua, redes sociais ou outros formatos para angariar os recursos financeiros ou produtos que necessitam. Como uma forma de facilitar e agilizar este processo, a tecnologia da informação pode ser aliada, aproximando instituições que precisam de ajuda com pessoas e outras instituições dispostas a ajudar. Este trabalho apresenta o desenvolvimento de um aplicativo para celular utilizando a plataforma Android com a linguagem de programação Java e o banco de dados Firebase, para cadastro de dados relacionados a instituições de caridade e assistenciais, visando facilitar a criação de campanhas de doação, permitindo o cadastro de instituições, de usuários, o acompanhamento das ações e o recebimento de notificações, entre outras funcionalidades.

Palavras-chave: Doações. Android. Firebase. Java.

ABSTRACT

It is quite common for charity and assistential institutions to run campaigns so that the general population, both citizens and companies, can become acquainted of their basic needs and then take part in with donations containing the most diverse items, such as food, cleaning products, personal care products, clothes, financial aid, among others. Often, getting specific information about missing or key items of an institution allows a better allocation of the donated resources, and, consequently, a better use of them. Thus, it is important that donors willing to assist these organizations can become aware of their existence and what is needed, which can even be emergency, and which are usually expressed through street marketing, social media or other formats used to raise the financial resources or products that are needed most. As a way to smooth and speed up this process, information technology may be combined, bringing together institutions that need help with people and other organizations willing to help. This graduation project presents the development of a mobile application using the Android platform along with Java programming language, and also Firebase, for data registration related to charity and assistential institutions, which makes the creation of your donation campaigns easier, allowing the inclusion of institutions, users, actions monitoring, notifications reception, among other features.

Keywords: Donations. Android. Firebase. Java.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

API	*Application Programming Interface*
DVM	*Dalvik Virtual Machine*
IBGE	Instituto Brasileiro de Geografia e Estatística
GPS	*Global Positioning System*
JNI	*Java Native Interface*
PDA	*Personal Digital Assistant*
RF	Requisito Funcional
RNF	Requisito Não Funcional.

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta a introdução do trabalho que abrange as considerações iniciais, os seus objetivos e a justificativa. O capítulo é finalizado com a apresentação dos capítulos subsequentes que compõem o texto.

1.1 CONSIDERAÇÕES INICIAIS

A informatização é uma realidade consolidada, visível em um aspecto amplo como, podendo auxiliar nos processos que envolvem os setores da produção de matéria-prima (primário), transformação dessa matéria-prima em produtos (setor secundário) e a venda desses produtos, com os serviços (setor terciário), além dos mais diversos segmento de educação, entretenimento e serviços em geral, para citar alguns.

O barateamento de computadores pessoais, *smartphones* e *tablets* e a facilidade ao acesso à Internet no Brasil que, conforme pesquisa divulgada pelo Instituto Brasileiro de Geografia e Estatística (IBGE), 116,1 milhões dos habitantes acessaram a Internet em 2016 (INSTITUTO..., 2018), são fatores que cooperam para o aumento de acesso às Tecnologias de Informação e Comunicação e, consequentemente, de uso dos seus recursos. A disponibilidade de computadores e tecnologias de comunicação global criou o mercado, ou a necessidade, de indústrias destinadas a reunir, processar e transmitir informações (FITZSIMMONS, 2014).

Além do aumento de usuários da Internet e da quantidade de dados gerados e armazenados, a quantidade de pessoas que utilizam *smartphones* para comunicação e entretenimento vem crescendo quase que exponencialmente. Em 2016 atingiu o percentual de 94,6% de usuários de celulares que os utilizaram para acessar a Internet, conforme pesquisa do IBGE (INSTITUTO..., 2018). Considerando essa grande quantidade de pessoas que utilizam dispositivos móveis para acesso à Internet, é possível facilitar vários aspectos do dia a dia de pessoas e empresas fazendo uso desta tecnologia, e, além disso, fomentar ainda mais ações de cunho social.

No caso de instituições de caridade ou assistenciais, aplicativos para dispositivos móveis podem se tornar grandes aliados em ações como campanhas de arrecadação, uma vez que este recurso levaria facilmente até os doadores os itens

de maior necessidade, os lembrariam de campanhas sazonais, os avisaria sobre campanhas emergenciais, e também sobre a existência de todas as instituições existentes em seu município ou em sua região. Além disso, um aplicativo centraliza os dados para que a população tome conhecimento a respeito de pessoas e famílias em condições precárias de sobrevivência, em situação de vulnerabilidade ou que precisem de auxílio em decorrência de eventos específicos, como desastres por fenômenos naturais ou ocasionados por ações humanas.

Este trabalho apresenta o desenvolvimento de um aplicativo com o intuito de estimular doações a pessoas, às instituições de caridade, assistenciais ou que promovam ajuda humanitária em eventos específicos.

O aplicativo desenvolvido tem como principal função avisar e lembrar seus usuários a respeito de campanhas, por meio de notificações relacionadas às instituições para as quais ele definiu previamente que deseja realizar doações. Essas notificações são realizadas no momento em que as entidades pelas quais optou por ajudar cadastrarem estas ações. Assim, o usuário poderá adquirir itens mais pontuais e necessários para alguma das entidades escolhidas para auxiliar.

1.2 OBJETIVOS

A seguir estão o objetivo geral e os objetivos específicos definidos para este trabalho.

1.2.1 Objetivo Geral

Implementar um aplicativo *mobile* para facilitar e fomentar a doação de itens pontuais e de maior necessidade em instituições de caridade e assistenciais, centralizando informações a respeito de campanhas de arrecadação à população em geral.

1.2.2 Objetivos Específicos

- Permitir ao usuário escolher as instituições que ele deseja auxiliar e, assim, receber notificações das campanhas para doação dessas instituições.

- Permitir que o usuário com perfil instituição cadastre campanhas com os produtos necessitados em um determinado período.
- Permitir que o usuário instituição cadastre contas bancárias nas quais poderá receber doações em espécie.
- Permitir ao usuário com perfil administrador aprovar ou não as instituições cadastradas para que então estas possam participar do *app*.

1.3 JUSTIFICATIVA

O uso de um aplicativo para dispositivos móveis que permita selecionar as instituições que o usuário queira ajudar e das quais ele aceita receber notificações sobre necessidade de ajuda, permite que o usuário seja comunicado dessas necessidades com mais conveniência. Essa conveniência é no sentido que ele tem a liberdade de escolher quem quer ajudar e de não ficar recebendo ligações e mensagens em momentos considerados inadequados.

Notificações por mensagens ou mesmo por ligações consideradas inoportunas podem se um fator para a pessoa a não realizar uma doação. Quando a pessoa tem a liberdade de escolher a instituição ela está ciente de quem quer auxiliar e de que será informada das necessidades dessas instituições.

O informe, como notificação referente às campanhas que cada instituição cria, é uma alternativa para agilizar a compra dos produtos pretendidos. Na correria do dia-a-dia, quando as pessoas não têm muito tempo para atividades além das suas rotineiras, esse pode ser um recurso indispensável.

1.4 ESTRUTURA DO TRABALHO

O Capítulo 2 apresenta o referencial teórico sobre aplicativos *mobile*. No Capítulo 3 estão às ferramentas e as tecnologias utilizadas na modelagem do sistema e que serão utilizadas na implementação subsequente do sistema. No Capítulo 4 é apresentado o resultado da realização do trabalho que é a modelagem do sistema com a definição dos requisitos, do banco de dados, do projeto e a implementação do aplicativo *mobile*. Por fim, estão as considerações finais seguidas pelas referências utilizadas na composição do texto.

2 SISTEMA OPERACIONAL ANDROID

Os aplicativos móveis (costumeiramente conhecidos como Apps) são produtos desenvolvidos para serem executados especificamente em dispositivos eletrônicos móveis, como os *Personal Digital Assistants* (PDA) que representam *palmtops*, *tablets*, leitores de MP3, telefones celulares e *smartphones* mais modernos e com grande capacidade de armazenamento e de processamento (SILVA; PIRES; CARVALHO NETO, 2015).

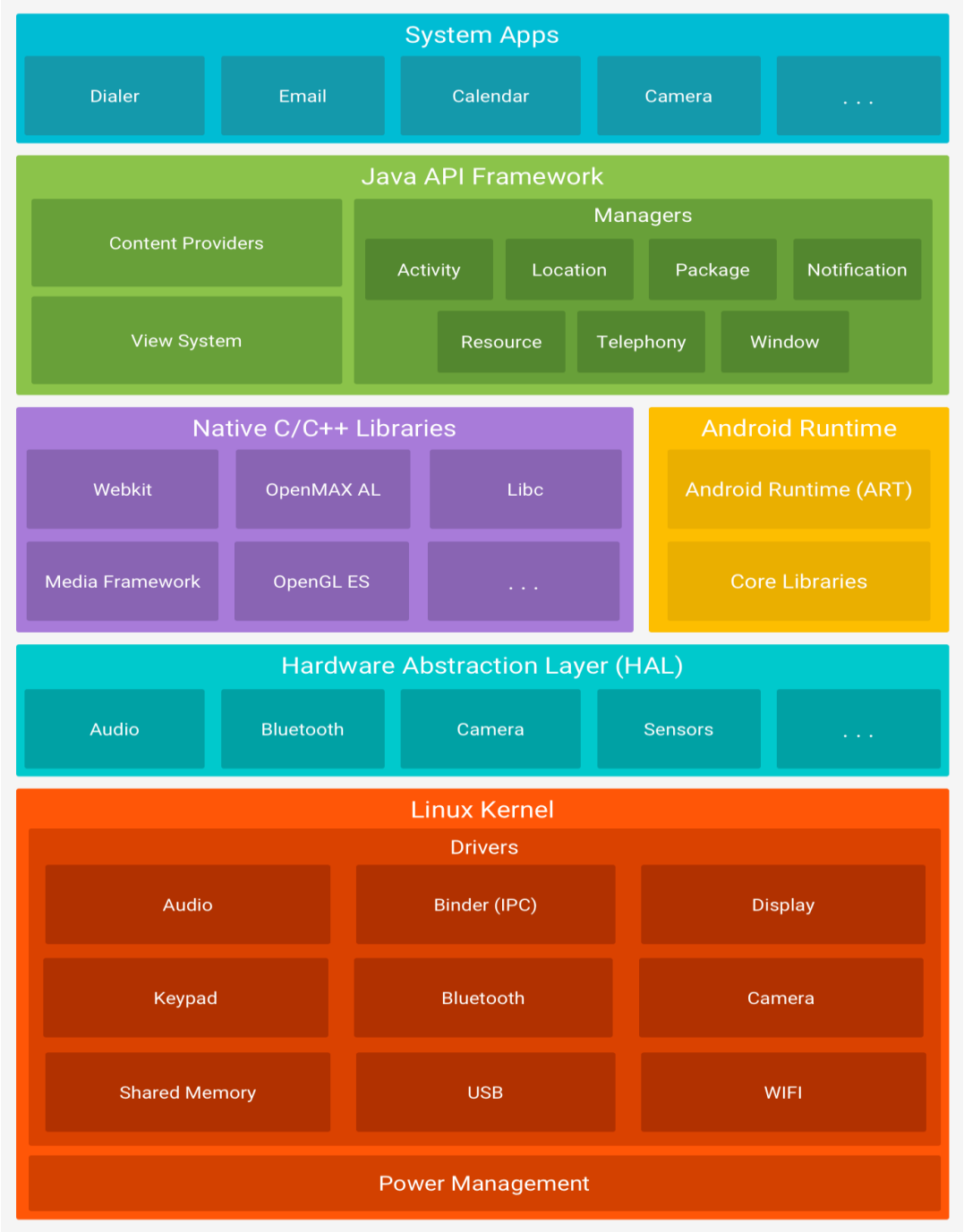
Dispositivos móveis são produzidos por diferentes fabricantes, o que inclui uma variedade de plataformas de desenvolvimento, sistemas operacionais móveis, software e hardware. São exemplos de arquiteturas desses dispositivos: Iphone, Black Berry, Android e Windows Phone.

A empresa Android Inc. foi fundada em 2003 com a ideia ser um sistema operacional inteligente para câmeras digitais (KLEINA, 2017). Foi em 2005, que a Google adquiriu a empresa Android Inc. E em 2014, começou a iniciativa Android One, uma versão básica do sistema para celulares de baixo custo. O sistema operacional Android foi apresentado em 2008 (PRIMORAC; RUSSO, 2012). Android é considerada a plataforma mais popular para *smartphone* (XU; ZHANG; ZHU, 2013).

O Android é um sistema operacional (plataforma) projetado para dispositivos móveis como *smartphones* e *tablets*. Contudo, atualmente, ele pode ser encontrado em outros dispositivos como câmeras digitais, televisores e videogames. O Android tem como base a arquitetura do Linux e, assim, seguindo suas premissas de distribuição, o seu código fonte pode ser modificado, mas o Google requer que a versão modificada execute todos os aplicativos disponíveis na Google Play, que é a loja virtual do Android (OLIVEIRA *et al.*, 2013).

O sistema operacional Android utiliza um conjunto, uma pilha, de software que inclui sistema operacional, *middleware*, interface com o usuário e aplicações (BRAY, 2010). Os aplicativos Android atendem a esse paradigma hierárquico no desenvolvimento de aplicações com a linguagem Java: a camada de mais baixo nível utiliza C e C++ e sobre ela está o código Java que faz chamadas às bibliotecas C e C++ por meio de *Java Native Interface* (JNI) (LI; WANG, 2014). A Figura 1 apresenta a estrutura, organização da arquitetura, do sistema operacional Android.

Figura 1 – Estrutura do sistema operacional Android



Fonte: Android Developers (2018, p.s.n.).

A arquitetura do sistema operacional Android, como apresentada na Figura 1, é dividida em seis camadas que possuem funcionalidades e comportamentos específicos. Essas camadas são descritas a seguir (GUANA *et al.*, 2012, PRIMORAC; RUSSO, 2012):

a) *System apps* - camada que contém as aplicações visíveis para o usuário. São as aplicações que executam sobre o sistema operacional e que estendem as

funcionalidades desse sistema, como clientes de *e-mail*, navegadores, jogos, agenda e todas as aplicações utilizadas pelo usuário. As aplicações são desenvolvidas com a linguagem Java.

b) Java API *framework* - camada de *framework* da aplicação. É um conjunto extensível de componentes de software (*Application Programming Interface* - API) usados pelas aplicações no sistema operacional. Essas APIs contêm ferramentas para a criação de interfaces e ferramentas de sistema, como as *intents* utilizadas para iniciar outros aplicativos ou atividades como abertura de arquivos.

c) *Native C/C++ Libraries* – são bibliotecas C/C++ utilizadas por componentes Android. Essas bibliotecas são usadas pelo *framework* da aplicação para gerenciar a renderização da tela, prover as funcionalidades de segurança de dispositivos e a persistência de dados das aplicações, entre outras funcionalidades.

d) Android *runtime* – cada aplicação no Android é executada no sistema operacional em um processo independente e para cada processo é criada uma instância da máquina virtual Dalvik. A camada de *runtime* é composta pela *Dalvik Virtual Machine* (DVM) e as bibliotecas principais (*core libraries*) que especificam o ambiente de execução da aplicação no sistema operacional. A DVM é a parte principal do ambiente de execução que é usada para iniciar as bibliotecas *core* desenvolvidas em Java.

e) *Hardware Abstraction Layer* – conectam o hardware e o software. Essa conexão ocorre entre o *framework Android* e o sistema operacional Linux. Essa camada abstrata permite que a aplicação ou *framework* Android se comunique com *drives* de dispositivos de específicos.

f) *Linux Kernel* – é a camada responsável pelo controle de processos, gerência de memória, *threads*, protocolos de rede, agendamento de tarefas, controles de segurança e gerenciamento de arquivos, entre outros.

2.1 FIREBASE

O Firebase é uma plataforma mantida pela Google, a qual disponibiliza várias ferramentas como Realtime Database, um banco de dados em tempo real e o Authentication, no qual pode ser utilizado para realizar autenticação de usuários. Estas duas ferramentas conversam entre si, são integradas, um usuário autenticado pode gravar e ler dados do Realtime Database. A forma como o Firebase armazena

os dados é em JSON desta forma o Firebase Realtime Database é um banco “noSQL”. Como o próprio nome informa o Firebase é um banco de dados em tempo real no qual a sincronização das informações ocorre instantaneamente, sempre que os dados são alterados os usuários conectados são atualizados sem necessidade de realizar requisições. Nos casos em que o usuário está “offline” os dados da aplicação ficam salvas localmente, na memória do *gadget*, e assim que o aparelho tem uma nova conexão disponível os dados são sincronizados.

3 MATERIAIS E MÉTODO

A seguir estão os materiais e o método utilizados para a modelagem e a implementação do sistema obtido como resultado deste trabalho.

3.1 MATERIAIS

As ferramentas e as tecnologias que serão utilizadas para a modelagem e posteriormente para o desenvolvimento do aplicativo são as listadas no Quadro 1.

Quadro 1 – Ferramentas e tecnologias

Ferramenta / Tecnologia	Versão	Finalidade
Visual Paradigm Online Express Edition		Modelagem de casos de uso Modelagem do banco de dados
Balsamiq Mockups	3.5	Modelagem das telas do aplicativo
Android Studio	3.6.2	Ambiente para desenvolvimento em linguagem Java para a aplicação *mobile*
Java	1.8	Linguagem de programação para a aplicação *mobile*
Firebase	19.2.0	Banco de dados da aplicação *mobile*

Fonte: Autoria própria.

3.2 MÉTODO

A definição inicial dos requisitos foi informal e teve como base o que poderia ser necessário para que o usuário do aplicativo pudesse ter acesso aos dados de Instituições do gênero assistenciais que ele pretende apoiar. Em seguida, aplicativos *mobile* foram avaliados visando identificar funcionalidades neles existentes e que poderiam ser utilizadas no aplicativo a ser desenvolvido. Dentre essas funcionalidades foi identificado que o uso de notificações como forma de lembrete ao usuário das doações para as entidades que ele escolher, tendo assim uma assertividade maior e potencializando a possibilidade de doações.

A partir de uma definição básica dos requisitos do ponto de vista de negócio (seleção de instituições para as quais o usuário pretende doar e de configurações para recebimento de notificações em relação a essas doações) foram definidos os requisitos do sistema. Os requisitos foram organizados em funcionais e não

funcionais e elaborados na forma de casos de uso. Um diagrama de entidades e relacionamentos do banco de dados também foi elaborado. E foi, ainda, desenvolvido o protótipo das telas do aplicativo utilizando a ferramenta Balsamiq Mockups. A elaboração do protótipo das telas auxiliou a definir os campos dos formulários, a forma de organização desses campos da tela.

A partir dos requisitos foram identificadas as tabelas do banco de dados e os seus campos. Os cadastros do sistema serão todos realizados Firebase.

Em seguida, as telas do aplicativo foram desenvolvidas no Android Studio, seguindo o padrão dos protótipos elaborados no Balsamiq e visando escolher os elementos de interface mais adequados para atender os requisitos. E posteriormente foi realizada a codificação do aplicativo.

Os testes foram informais, ou seja, realizados pelo próprio autor do trabalho, mas sem um plano de testes específico e estiveram centrados na identificação de problemas de código e na verificação do atendimento aos requisitos definidos para o aplicativo.

4 RESULTADOS

Este capítulo apresenta os resultados da realização deste trabalho. A modelagem apresenta os requisitos definidos e a sua organização em casos de uso que são expandidos, o diagrama do banco de dados e o protótipo das telas, com a implementação das telas em Android.

4.1 ESCOPO DO APLICATIVO

O aplicativo desenvolvido é para a plataforma *mobile* Android que realiza consulta em um banco de dados “noSQL” Firebase. O aplicativo possui uma tela inicial na qual o usuário informará seu e-mail e senha que ficará persistido no aplicativo móvel do usuário. Posteriormente informará os dados cadastrais. O nome será utilizado para o usuário ser identificado nas notificações. Uma vez acessado o aplicativo, o usuário poderá acessar uma tela na qual ele selecionará uma ou mais instituições (previamente cadastradas no banco de dados) que pretender ajudar com doações.

O usuário com perfil Instituição poderá cadastrar campanhas de doções, que poderão ser permanentes ou apenas em um determinado período.

O usuário Instituição poderá cadastrar as contas correntes que dispõe para receber doações em dinheiro, através de transações financeiras. A Instituição ainda poderá cadastrar os produtos os quais deseja receber em suas campanhas.

O usuário com perfil Administrador poderá aprovar ou reprovar as instituições que se cadastrem no aplicativo antes destas serem mostradas aos usuários apoiadores.

O usuário apoiador será notificado das campanhas cadastradas pelas instituições as quais escolheu apoiar. O usuário apoiador poderá cadastrar doações realizadas, informando valor e se a doação foi em produtos.

4.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta os requisitos funcionais definidos para o sistema. Neste quadro RF significa Requisito Funcional.

Quadro 2 – Requisitos funcionais

Identificação	Requisito	Descrição
Aplicativo móvel		
RF01	Cadastrar usuário	Cadastro do usuário armazenando informando dados como nome, CPF, endereço e telefone. O nome será utilizado nas notificações e em telas e relatórios.
RF02	Consultar e escolher instituições	A partir de uma lista cadastrada no servidor, o usuário poderá selecionar as instituições para as quais ele pretende doar.
RF03	Cadastrar doações realizadas	O usuário poderá cadastrar as doações realizadas informando a instituição, data e valor, ou caso tenha sido efetuado através de produtos apenas informar que a ajuda foi realizada com produtos.
RF04	Receber notificações de campanhas realizadas pelas instituições	O usuário deverá ser notificado das campanhas cadastradas pelas instituições as quais este usuário escolheu apoiar.
RF05	Manter instituições	Registro de instituições, com dados cadastrais como razão social, nome fantasia, endereço e outros. Também é possível listar, excluir e alterar dados de instituições.
RF06	Manter produtos	Manter cadastro de produtos. As instituições utilizam o cadastro de produtos para definir o que eles precisam naquele momento e que desejam seja doado. Esses produtos também serão utilizados nas notificações, baseados nas necessidades de cada instituição para um determinado período.
RF07	Manter contas bancárias	Uma instituição poderá ter mais de uma conta bancária para receber doações. Esse cadastro permitirá cadastrar contas vinculadas as instituição. Esses registros serão exibidos nas notificações e na tela de visualização das instituições.
RF08	Analisar Insituiçoes	Usuário com perfil administrador deverá analisar as instituições antes de estas estarem disponíveis para os apoiadores escolherem como instituições favoritas.
RF09	Controle de acessso de usuários	Cada perfil de usuário, apoiador, adminitrador, instituição poderá ter acesso somente às telas as quais tiver permissão.

Fonte: Autoria própria.

Os requisitos não funcionais identificados para o sistema estão definidos no Quadro 3. Neste quadro RNF significa Requisito Não Funcional.

Quadro 3 – Requisitos não funcionais

Identificação	Requisito	Descrição
RNF01	Acesso ao sistema	Para o primeiro acesso ao sistema será obrigatório o preenchimento e-mail e senha, seguido de um cadastro com informações básicas como CPF, endereço telefone.
RNF02	Interface do aplicativo	O aplicativo não terá interface responsiva, a interface será somente na posição *portrait.*

RNF03	Validar as instituições selecionadas pelo usuário para envio de notificações	O aplicativo deverá validar instituições selecionadas pelo usuário apoiador para realizar as notificações.

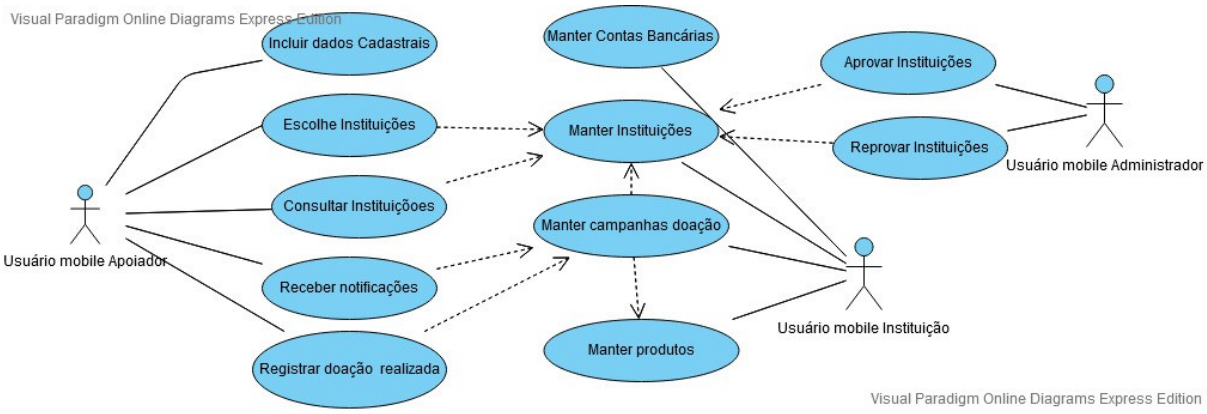
Fonte: Autoria própria.

A Figura 2 apresenta o diagrama de casos de uso definido a partir dos requisitos. Foram definidos três atores: usuário *mobile* Instituição, Apoiador e Administrador. O primeiro ator representa a aplicação que executa no dispositivo móvel do usuário (cliente) e tem como base os cadastros realizados por meio do segundo ator, Instituição. O usuário utiliza esses cadastros para escolher as instituições para as quais ele quer realizar doações. O usuário também faz um cadastro, com dados básicos como nome, CPF, endereço e telefone. O nome do usuário, os dados de configurações, as instituições para as quais o usuário quer doar e dados de doações realizadas são os únicos que ficam armazenados no banco de dados Firebase.

O usuário Instituição realiza os cadastros das instituições que recebem doações, contas bancárias das instituições para recebimento de doação em espécie (valor monetário), produtos que podem ser doados, campanhas de doação sendo realizadas pelas instituições e outros cadastros básicos.

O usuário Administrador analisará as instituições cadastradas antes que estas possam ser selecionadas pelos usuários apoiadores, a fim de evitar golpes.

Figura 2 – Diagrama de casos de uso



Fonte: Autoria própria.

Os quadros 4 a 7 apresentam a descrição das operações de cadastro dos casos de uso definidos como “manter” na Figura 2. Esses casos de uso são: manter contas bancárias, manter produtos, manter campanhas de doação e manter instituições.

No Quadro 4 está a expansão da operação “incluir” dos casos de uso “manter” e do caso de uso “incluir nome”. No primeiro acesso no aplicativo *mobile,* o usuário deverá informar o seu nome que será utilizado nas notificações e nos relatórios gerados.

Quadro 4 – Operação “incluir” dos casos de uso de cadastro

Caso de uso: Operação incluir. Refere-se à operação de inclusão de todos os casos de usos identificados como “manter”. **Descrição:** Inclusão dos dados cadastrais no sistema. **Evento Iniciador:** Ator solicita inclusão de um registro no sistema. **Atores:** Apoiador e instituição. **Pré-condição:** Não há. **Sequência de Eventos:** 1. Ator acessa a tela para realizar o cadastro inserindo as informações necessárias. 2. O sistema insere as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Registro inserido no banco de dados. **Extensões:** Campos obrigatórios faltantes e campos no formato incorreto.	
Nome do fluxo alternativo	Descrição

(extensão)	
1. Campos obrigatórios não informados.	1.1. O usuário deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema verifica que não foram informados todos os campos obrigatórios e exibe mensagem sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2. Campos informados em formato incorreto.	2.1. O usuário informa dados em formato incorreto e clica em salvar. 2.2. O sistema valida e verifica que os dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados já informados.

Fonte: Autoria própria.

A descrição da operação “alterar” dos casos de uso “manter” é apresentada no Quadro 5.

Quadro 5 – Operação “alterar” dos casos de uso de cadastro

Caso de uso: Operação alterar. Refere-se à operação de alteração de todos os casos de usos identificados como “manter”. **Descrição:** Alteração dos dados cadastrais no sistema. **Evento Iniciador:** Ator solicita alteração de um registro no sistema. **Atores:** Apoiador e instituição. **Pré-condição:** Registro estar incluso no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para visualização dos dados do registro. 2. O sistema apresenta o registro selecionado para alteração. 3. Ator altera os dados do registro. 4. O sistema altera as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Registro alterado no banco de dados. **Extensões:** Campos obrigatórios faltantes e campos no formato incorreto.	
Nome do fluxo alternativo (extensão)	Descrição
1. Campos obrigatórios não informados.	1.1. O usuário apaga dados obrigatórios e clica em salvar. 1.2. O sistema valida que não foram informados todos os campos obrigatórios e exibe mensagem ao usuário sem salvar o registro. 1.3. O sistema permanece na tela de edição mantendo as alterações realizadas.
2. Campos informados	2.1. O usuário altera dados deixando-os em formato incorreto e

em formato incorreto.	clica em salvar. 2.2. O sistema valida que dados não estão no formato esperado e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de edição mantendo as alterações realizadas.

Fonte: Autoria própria.

O Quadro 6 apresenta a descrição da operação “excluir” dos casos de uso “manter”.

Quadro 6 – Operação “excluir” dos casos de uso de cadastro

Caso de uso: Operação excluir. Refere-se à operação de exclusão de todos os casos de usos identificados como “manter”. **Descrição:** Exclusão dos dados cadastrais no sistema. **Evento Iniciador:** Ator solicita exclusão de um registro no sistema. **Atores:** Apoiador e instituição. **Pré-condição:** Registro estar incluso no sistema. **Sequência de Eventos:** 1. O sistema exclui as informações no banco de dados e informa ao usuário o *status* do procedimento. **Pós-Condição:** Registro excluído no banco de dados. **Extensões:** Registro possui vínculo com outros cadastros.	
Nome do fluxo alternativo (extensão)	Descrição
1. Exclusão de registro com vínculos no sistema	1.1. O usuário clica em excluir um registro que possui vínculos no sistema. 1.2. O sistema verifica que o registro tem vínculos, não o exclui e exibe mensagem de alerta ao usuário.

Fonte: Autoria própria.

No Quadro 7 está a descrição da operação “consultar” referente aos casos de uso “manter”.

Quadro 7 – Operação “consultar” dos casos de uso de cadastro

Caso de uso: Operação consultar. Refere-se à operação de consulta de todos os casos de usos identificados como “manter”. **Descrição:** Consulta dos dados cadastrais dos registros do sistema.

Evento Iniciador: Ator solicita consulta de um registro no sistema. **Atores:** Administrador, instituição e apoiador **Pré-condição:** Registro estar incluso no sistema. **Sequência de Eventos:** 1. Ator acessa a tela para visualização dos dados do registro. 2. O ator indica os filtros desejados para consulta. 3. O sistema apresenta os dados da consulta ao usuário. **Pós-Condição:** Dados da consulta apresentados ao usuário.

Fonte: Autoria própria.

O Quadro 8 apresenta a expansão do caso de uso Cadastrar Campanha. O ator Instituição poderá cadastrar as campanhas de arrecadação de donativos. Estas campanhas poderão ser visualizadas pelos atores Apoiadores. Os atores apoiadores somente visualizarão as campanhas das suas instituições favoritas.

Quadro 8 – Caso de uso manter campanha de doação.

Caso de uso: Operação Cadastrar Campanha. Refere-se à operação de inclusão de registro do caso de uso “Manter campanhas de doação”. **Descrição:** Inclusão de dados cadastrais no sistema, referente a campanha. **Evento Iniciador:** Ator solicita inclusão de um registro no sistema. **Atores:** Instituição. **Pré-condição:** Cadastro de Instituição pré-existente. **Sequência de Eventos:** 1. Ator acessa a tela para realizar o cadastro inserindo as informações necessárias. 2. O sistema insere as informações no banco de dados e informa ao usuário o status do procedimento. **Pós-Condição:** Registro inserido no banco de dados. **Extensões:** Campos obrigatórios faltantes, registros com vínculos a outros cadastros.	
Nome do fluxo alternativo (extensão)	Descrição
1. Campos obrigatórios não informados.	1.1. O usuário deixa de informar dados obrigatórios e clica em salvar. 1.2. O sistema verifica que não foram informados todos os campos obrigatórios e exibe mensagem sem salvar o registro. 1.3. O sistema permanece na tela de inclusão mantendo os dados informados anteriormente.
2. Registros com vínculos a outros cadastros.	2.1. O usuário deixa de selecionar produtos para campanha. 2.2. O sistema valida e verifica que o registro não tem produtos

	vinculados e exibe mensagem ao usuário sem salvar o registro. 2.3. O sistema permanece na tela de inclusão mantendo os dados já informados.

Fonte: Autoria própria.

O Quadro 9 apresenta a expansão do caso de uso escolher instituições. O usuário, a partir de uma lista de instituições cadastradas pelo servidor, seleciona as que ele quer auxiliar, ou seja, contribuir nas campanhas de doação. O usuário receberá notificações das instituições que selecionar para auxiliar.

Quadro 9 – Caso de uso escolher instituições

Caso de uso: Escolher instituições. **Descrição:** O usuário escolhe a partir de uma lista as instituições que ele quer ajudar. As instituições foram cadastradas. **Evento Iniciador:** Ator seleciona lista de instituições. **Atores:** Usuário apoiador **Pré-condição:** Haver instituições cadastradas. **Sequência de Eventos:** 1. Ator acessa a listagem de instituições. As instituições que ele já escolheu auxiliar são apresentadas. 2. O seleciona as instituições que quer auxiliar. 3. O sistema marca essas instituições e passará a emitir notificações relacionadas a ela, de acordo com as campanhas cadastradas. **Pós-Condição:** Instituições selecionadas, marcadas, como escolhidas para apoiar. **Extensões:** Usuário quer desmarcar uma instituição escolhida para auxiliar.	
Nome do fluxo alternativo (extensão)	Descrição
1. Desmarcar instituição	1.1. O usuário clica no componente referente a excluir a seleção de uma instituição anteriormente marcada. 1.2. O sistema excluir a instituição de selecionada. Assim, o usuário não mais receberá notificações relacionadas à referida instituição.

Fonte: Autoria própria.

O caso de uso aprovar instituições é expandido no Quadro 10.

Quadro 10 – Caso de uso aprovar instituições

Caso de uso: Aprovar Instituições **Descrição:** O usuário administrador irá analisar os dados cadastrais das instituições e com base

nestas informações irá aprovar a instituição para que ela possa ser uma instituição que irá ser mostrada para os apoiadores.

Evento Iniciador:

Ator seleciona a tela de instituições não avaliadas

Atores:

Administrador.

Pré-condição:

Aplicativo em execução.

Sequência de Eventos:

1. Ator acessa a tela de instituições não avaliadas e seleciona a instituição a ser aprovada.
2. O sistema salva as a instituição como aprovada no banco de dados
.

Pós-Condição:

Instituição cadastrada

Fonte: Autoria própria.

A expansão do caso de uso reprovar instituições é mostrada no Quadro 11.

Quadro 11 – Caso de uso reprovar instituições

Caso de uso:

Reprovar instituições.

Descrição:

O usuário configura administrador irá analisar os dados cadastrais das instituições e com base nestas informações irá aprovar a instituição para que ela possa ser uma instituição que irá ser mostrada para os apoiadores.

Evento Iniciador:

Produtos serem vinculados com estabelecimentos que os vendam. Essa necessidade pode ser gerada no cadastro de um novo produto, de um novo estabelecimento ou que um estabelecimento passe a vender produtos que não costumava vender.

Atores:

Sistema no servidor.

Pré-condição:

Campanhas ativas com produtos como doação e estabelecimentos cadastrados que vendam esses produtos.

Sequência de Eventos:

1. Ator seleciona produtos.
2. Ator vincula estabelecimento a produto
3. Ator solicita inclusão da informação no banco de dados
4. Sistema salva registro no bando de dados.

Pós-Condição:

Produto vinculado a estabelecimento que o vende.

Fonte: Autoria própria.

A expansão do caso de uso receber notificação é apresentada no Quadro 12.

Quadro 12 – Caso de uso receber notificação

Caso de uso:

Receber notificação.

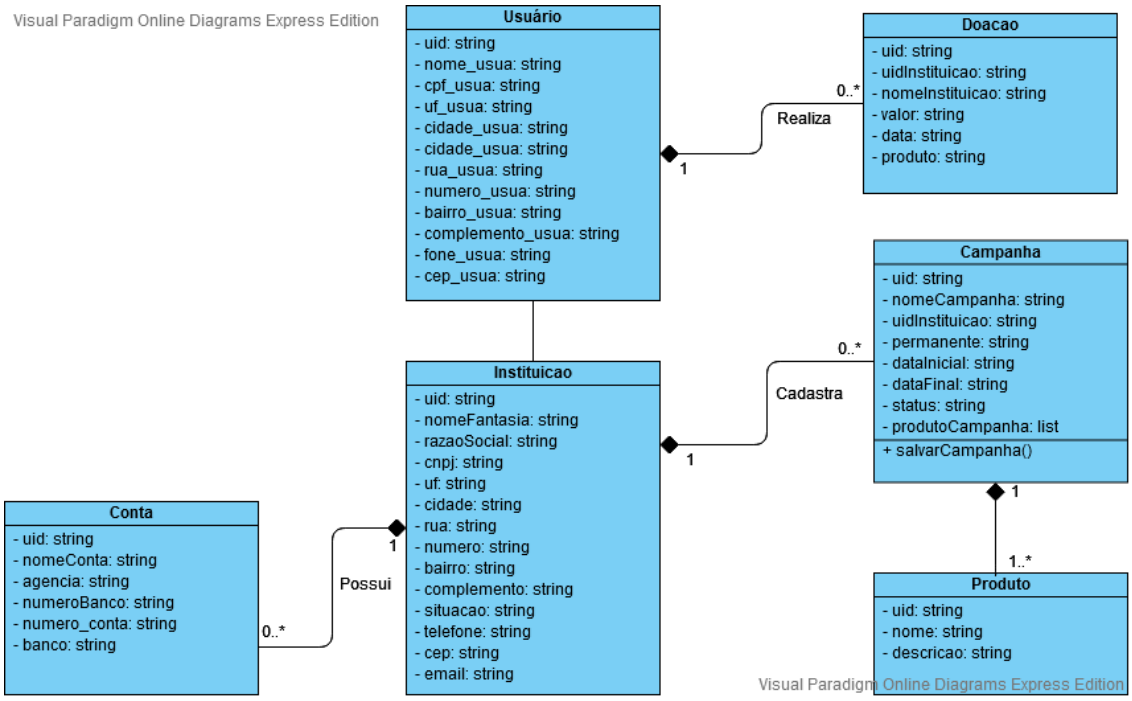
Descrição:

No momento que um usuário Instituição cadastrar uma campanha o usuário Apoiador irá receber uma notificação informando sobre a campanha, produtos que a instituição necessita e período da campanha ou se a mesma é permanente. Essa notificação tem o objetivo de informar ao usuário que aquela determinada instituição está necessitando daqueles produtos específicos.
Evento Iniciador:
Ao confirmar o cadastro de uma campanha.
Atores:
Sistema no aplicativo *mobile*.
Pré-condição:
Cadastro de instituição, cadastro de campanha.
Sequência de Eventos:
1. Usuário Instituição cria um registro de uma campanha.
2. Sistema emite notificação para o usuário.
Pós-Condição:
Notificação enviada.

Fonte: Autoria própria.

A Figura 3 apresenta o diagrama de classes. As classes Usuario, Doacoes, Instituicao, Conta, Campanhas, Produtos. Os dados gerados por meio destas classes ficam armazenadas no banco de dados NoSQL Firebase.

Figura 3 – Diagrama de Classes



Fonte: Autoria própria.

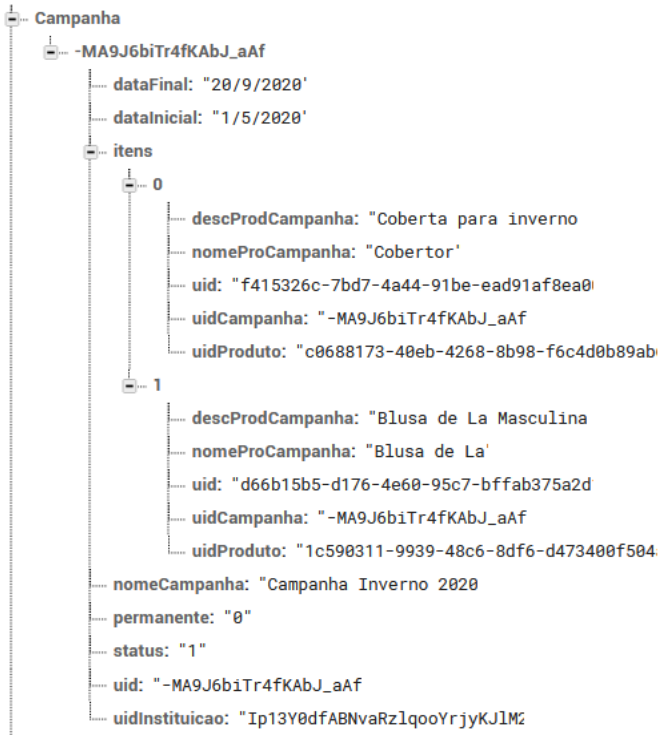
- A classe Instituicao representa as entidades assistenciais, sociais e outras que podem receber doações vinculadas às campanhas.
- A classe Produtos, se referee aos produtos das campanhas, realizadas pelas instituições.
- As contas bancárias das instituições são apresentadas na classe Conta.
- A classe Campanha representa as campanhas cadastradas pelas instituições. Estas definem o que cada instituição necessita naquele momento e permite vincular aos produtos (classe Produto) que são itens de doação solicitados nas campanhas. Esse vínculo permite ao usuário receber notificações por meio das campanhas cadastradas pelo usuário Instituição.
- A classe Doacao representa as doações realizadas pelo usuário. As Doações podem ser realizadas em espécie (dinheiro) ou podem ser anotadas como produtos. No caso o usuário realizar doações em espécie este tem acesso à identificação das contas bancárias das instituições, podendo realizar as doações de forma eletrônica.
- A classe que representa os usuários do sistema, nomeada Usuario, tem os atributos dos usuários, como nome, endereço, CPF. Os dados de *login* e senha são gravados em outra ferramenta, o Firebase Authentication.

As figuras 4 e 5 apresentam a estrutura das tabelas no Firebase.

Figura 4 – Estrutura Firebase

Fonte: Autoria própria.

Figura 5 – Estrutura Firebase - Nodo Campanha



Fonte: Autoria própria.

4.3 APRESENTAÇÃO DO SISTEMA

A Figura 6 apresenta a tela inicial do aplicativo, na qual o usuário se identifica. Essa identificação será utilizada para realizar a autenticação no aplicativo.

Figura 6 – Tela de primeiro acesso ao aplicativo

Fonte: Autoria própria.

A Figura 7 apresenta a tela de cadastro usuário Apoiador.

Figura 7 – Tela de cadastro de usuário

Fonte: Autoria própria.

A Figura 8 mostra a tela em que o usuário determina qual será o seu perfil, se Instituição, ou usuário Apoiador.

Figura 8 – Tela de escolha de perfil

Fonte: Autoria própria.

A Figura 9 apresenta a tela do menu inicial do usuário apoiador.

Figura 9 – Tela de menu inicial de usuário apoiador

Fonte: Autoria própria.

A tela de seleção de Instituições implementada em IDE Android Studio é apresentada na Figura 10. Nessa tela são apresentadas as instituições cadastradas no servidor e o usuário poderá selecionar as que ele pretende auxiliar. O usuário pode selecionar a instituição com o clique longo.

Figura 10 – Tela de lista de Instituições

Fonte: Autoria própria.

A Figura 11 apresenta a tela na qual detalha as informações da instituição. São apresentadas informações como razão social, CNPJ, endereço, nome fantasia e também uma lista com as contas bancárias desta instituição.

Figura 11 – Tela a qual detalha os dados Instituições

Fonte: Autoria própria.

Na Figura 12 é apresentada a tela de cadastro de doações. O cadastro de doações é apenas informativo, não são realizadas doações ou transações financeiras pelo aplicativo. No cadastro da doação, deve-se informar a instituição, data e valor da doação e caso seja selecionada a opção Produtos, representa que a doação foi realizada em produtos. O cadastro de doação tem como objetivo realizar um controle de doações que o usuário realiza.

Figura 12 – Tela de cadastro de doações

Fonte: Autoria própria.

Na Figura 13 pode-se visualizar a tela inicial do usuário Administrador.

Figura 13 – Tela do menu inicial do usuário Administrador

Fonte: Autoria própria.

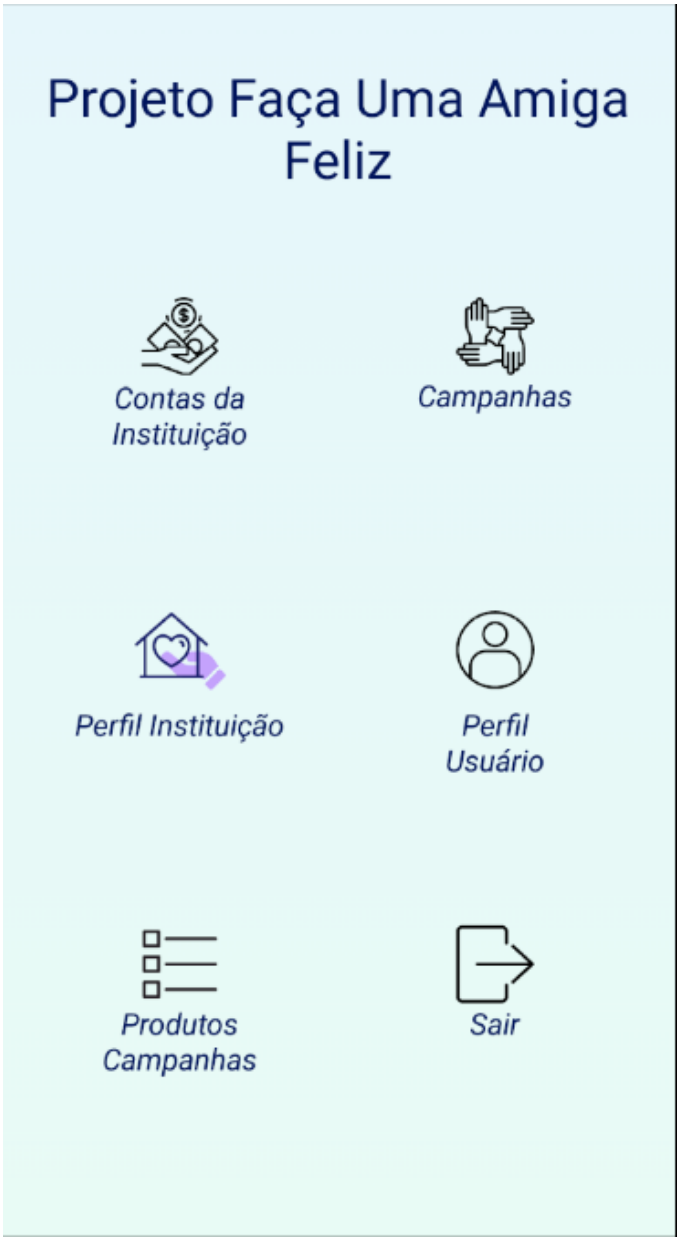
A Figura 14 mostra a lista de instituições as quais o Administrador analisará.

Figura 14 – Tela de lista de instituições a serem analisadas

Fonte: Autoria própria.

Na Figura 15 é mostrado o aplicativo acessado por um perfil Instituição.

Figura 15 – Tela do menu inicial do usuário Instituição



Fonte: Autoria própria.

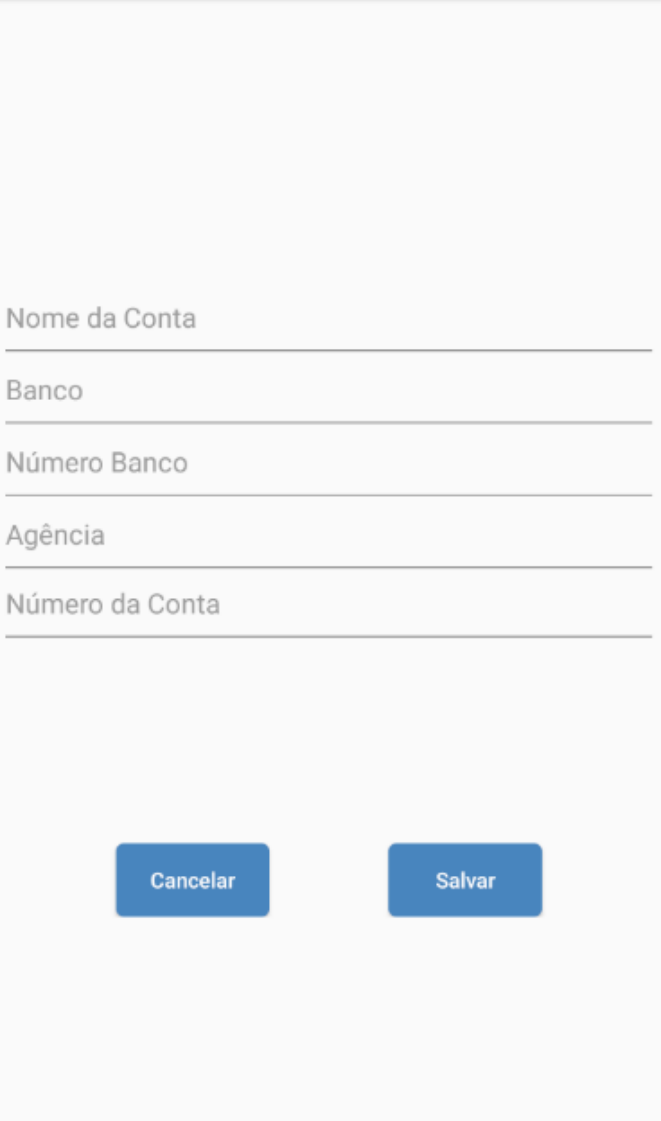
A Figura 16 apresenta a tela de cadastro de produtos, no perfil Instituição. Os produtos cadastrados nessa tela poderão ser utilizados no cadastro de campanhas.

Figura 16 – Tela de cadastro de produtos

Fonte: Autoria própria.

Na Figura 17 é apresentada a tela de cadastro de contas bancárias, as quais a Instituição cadastra e serão apresentadas aos apoiadores para caso de doações em espécie.

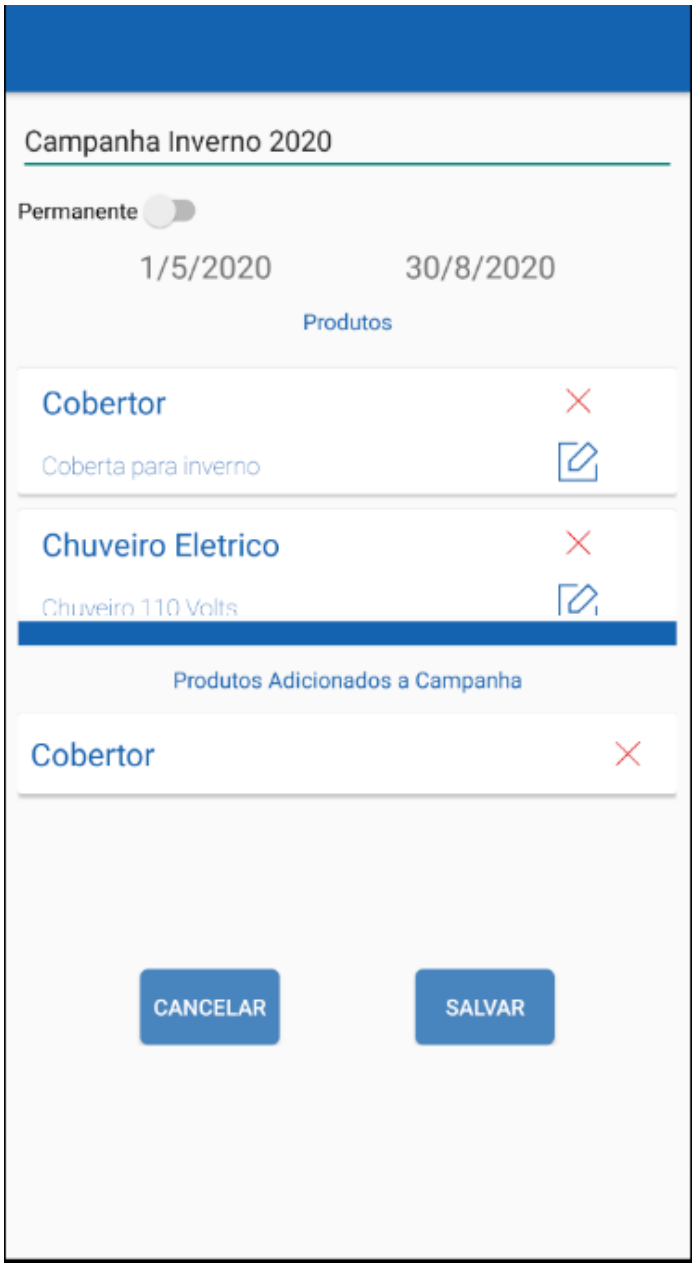
Figura 17 – Tela de cadastro de contas bancárias



Fonte: Autoria própria.

Na Figura 18 é mostrada a tela de cadastro de campanhas, as quais as instituições cadastram e os usuários apoiadores são alertados.

Figura 18 – Tela de cadastro de campanhas de doação



Fonte: Autoria própria.

Na Figura 19 é apresentado a tela com a lista de campanhas, visualizada pelo Apoiador. Nesta lista são mostradas as campanhas cadastradas pelas instituições favoritas deste usuário, as instituições as quais este escolheu para apoiar. A Figura 20 apresenta a lista de produtos da campanha. Esta pode ser visualizada quando o usuário clica em uma das campanhas, apresentada na Figura 19.

Figura 19 – Lista de Campanhas apresentada para o Apoiador

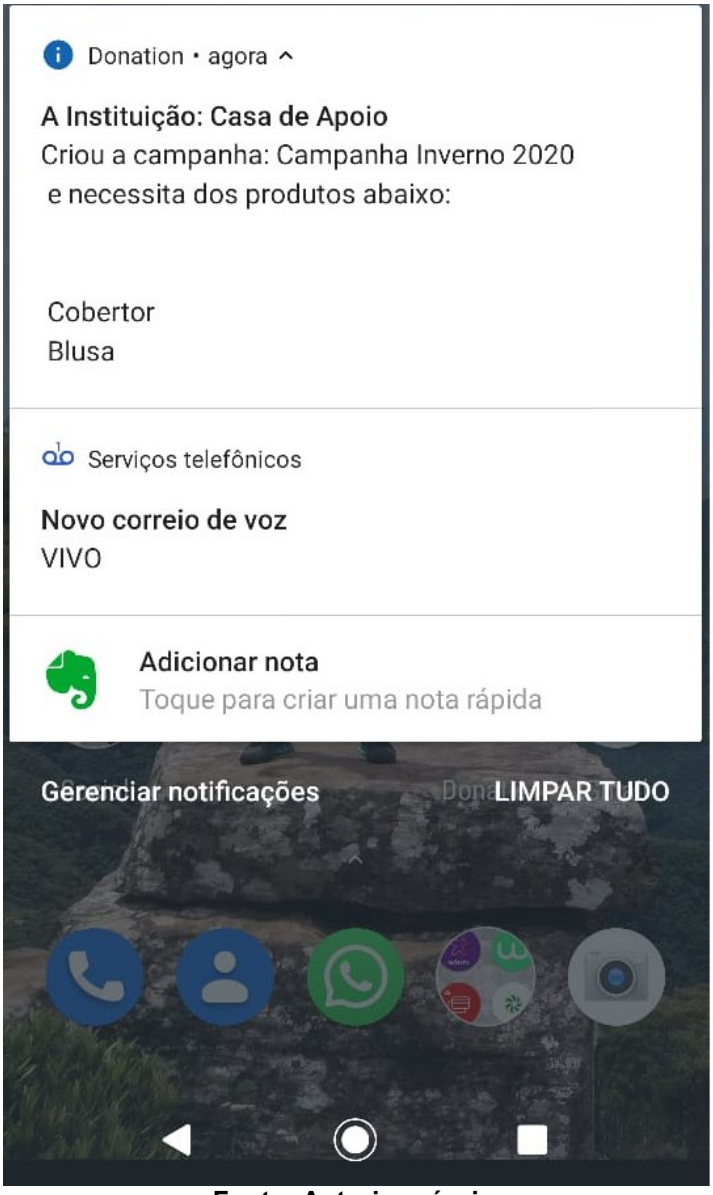
Fonte: Autoria própria.

Figura 20 – Visualização dos produtos da campanha pelo usuário Apoiador

Fonte: Autoria própria.

A Figura 21 apresenta a notificação recebida pelo usuário. A notificação se refere a uma campanha cadastrada por uma das instituições favoritas do usuário. No momento em que a instituição favorita cadastra uma campanha, o usuário recebe a informação da campanha cadastrada, com o nome da instituição e também a lista de produtos que esta instituição necessita para esta campanha específica.

Figura 21 – Notificação recebida



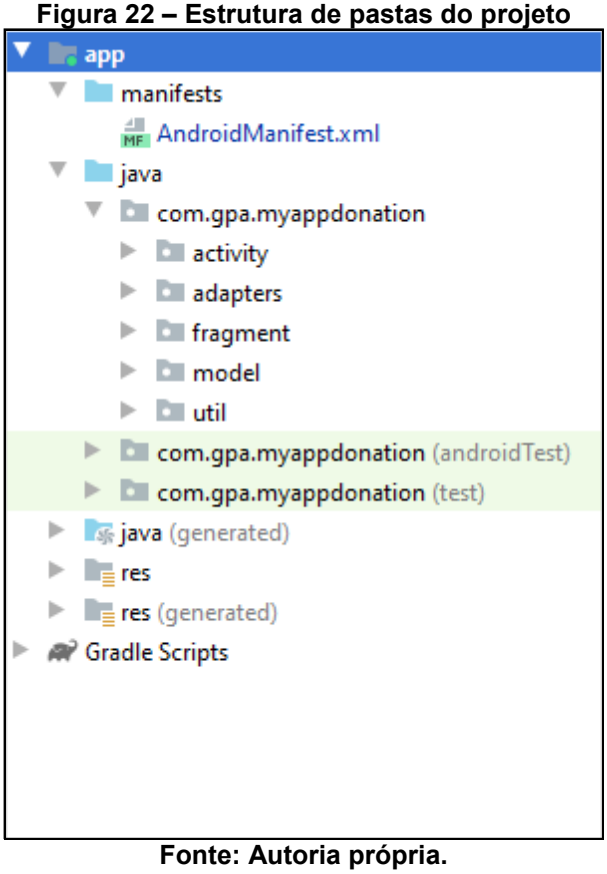
Fonte: Autoria própria.

4.3 IMPLEMENTAÇÃO DO SISTEMA

A codificação do projeto foi realizada em linguagem Java no ambiente de desenvolvimento Android Studio utilizando o banco de dados na nuvem uma noSQL Firebase, que é uma ferramenta disponibilizada pela Google de forma gratuita para projetos que não utilizem muitos recursos (FIREBASE, 2020). O Firebase disponibiliza a ferramenta de Realtime Database, e o serviço de autenticação os quais foram utilizados no projeto. O Relatime Database é um banco de dados que armazena os dados em formato JSON, é um banco de dados noSQL.

A Figura 22 mostra a organização das pastas do projeto, divididas em Activity, Adapters, Fragment, Model e Util.

Figura 22 – Estrutura de pastas do projeto



Fonte: Autoria própria.

Nas listagens de código 1 e 2 são mostrados os códigos do arquivo de configurações do Firebase e a biblioteca implementada.

Listagem de código 1- Arquivo de configuração do Firebase

```
{
  "project_info": {
    "project_number": "558726277980",
    "firebase_url": "https://myappdonation-c51ae.firebaseio.com",
    "project_id": "myappdonation-c51ae",
    "storage_bucket": "myappdonation-c51ae.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:558726277980:android:227ac4e2fbeff00ea1af99",

        "android_client_info": {
          "package_name": "com.gpa.myappdonation"
        }
      },
      "oauth_client": [
        {
          "client_id": "558726277980-
9kq0h381p1ht6uh82i9a38h8h22o0q3f.apps.googleusercontent.com",
          "client_type": 1,
          "android_info": {
            "package_name": "com.gpa.myappdonation",
            "certificate_hash": "e46fb2991d66139080faf03056f1478231ec4f74"
          }
        },
        {
          "client_id": "558726277980-
gr8b2m8v5jdnun1s4bu2t3hqoedoptj5.apps.googleusercontent.com",
          "client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "**************************"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": [
            {
              "client_id": "558726277980-
gr8b2m8v5jdnun1s4bu2t3hqoedoptj5.apps.googleusercontent.com",
              "client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Fonte: Autoria própria.

Listagem de código 2 - Adicionando biblioteca Firebase

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.2"
    defaultConfig {
        applicationId "com.gpa.myappdonation"
        minSdkVersion 25
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {

        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.google.android.material:material:1.0.0'
    implementation 'androidx.annotation:annotation:1.1.0'
    implementation 'androidx.lifecycle:lifecycle-extensions:2.1.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'androidx.navigation:navigation-fragment:2.0.0'
    implementation 'androidx.navigation:navigation-ui:2.0.0'
    testImplementation 'junit:junit:4.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
    implementation 'com.google.firebase:firebase-analytics:17.2.2'
    implementation 'com.google.firebase:firebase-auth:19.2.0'

    implementation 'com.google.code.gson:gson:2.7'
    implementation 'com.google.firebase:firebase-database:19.2.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'com.github.BlacKCaT27:CurrencyEditText:2.0.2'
    implementation 'com.github.santalu:mask-edittext:1.0.7'
    implementation 'com.github.d-max:spots-dialog:1.1@aar'

    //Dependecia para notificações
    implementation 'com.google.firebase:firebase-iid:20.2.0'
    implementation 'com.google.firebase:firebase-messaging:20.2.0'

    //noinspection GradleCompatible
    compile 'com.google.firebase:firebase-messaging:9.4.0'
```

```
    implementation 'com.google.firebase:firebase-core:16.0.8'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
}
```

Fonte: Autoria própria.

Na pasta de 'activity' estão às classes que gerenciam as telas da aplicação, na pasta 'model' estão salvas às classes que representam as entidades do banco de dados, já na pasta '*adapter*' ficam armazendas às classes *adapters* responsáveis por criar a conexão entre uma lista de dados e a visualização na tela. A Listagem 3 mostra o *adapter* responsável pelas listas de instituições.

Listagem de código 3 - Adapter de instituições

```
package com.gpa.myappdonation.adapters;

import android.content.Context;
import android.database.DataSetObserver;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ListAdapter;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.gpa.myappdonation.R;
import com.gpa.myappdonation.model.Instituicao;

import java.util.List;

public class Adapter_instituicoes extends
RecyclerView.Adapter<Adapter_instituicoes.MyViewHolder> {
    private List<Instituicao> instituicoes;
    private Context context;

    public Adapter_instituicoes(List<Instituicao> instituicoes, Context context) {
        this.instituicoes = instituicoes;
        this.context = context;
    }
    @NonNull
    @Override
    public Adapter_instituicoes.MyViewHolder onCreateViewHolder(@NonNull ViewGroup
parent, int viewType) {
        View item =
LayoutInflater.from(parent.getContext()).inflate(R.layout.adapter_instituicoes,par
ent,false);
        return new MyViewHolder(item);
    }
    @Override
    public void onBindViewHolder(@NonNull Adapter_instituicoes.MyViewHolder
holder, int position) {
        Instituicao instituicao = instituicoes.get(position);
```

```
        holder.nomeInst.setText(instituicao.getNomeFantasia());
        holder.cidadeInst.setText(instituicao.getCidade());
        holder.ufInst.setText(instituicao.getUf());
    }

    @Override
    public int getItemCount() {
        return instituicoes.size();
    }
    public class MyViewHolder extends RecyclerView.ViewHolder {
        TextView nomeInst;
        TextView cidadeInst;
        TextView ufInst;

        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            nomeInst = itemView.findViewById(R.id.txtNomeInstituicao);

            cidadeInst = itemView.findViewById(R.id.txtCidadeInstituicao);

            ufInst = itemView.findViewById(R.id.txtUfInstituicao);
        }
    }
}
```

Fonte: Autoria própria.

A classe CampanhaActivity.java da pasta activity é responsável pelo gerenciamento da tela de cadastro de campanhas. Nesta activity estão os blocos de código de salvar editar a campanha. A Listagem de código 4 apresenta o código responsável por salvar a campanha e adicionar a respectiva lista de produtos.

Listagem de código 4 - Classe CampanhaActivity que gerencia cadastro de campanhas

```
recyclerProdutos.addOnItemTouchListener(new RecyclerItemClickListener(
                this,
                recyclerProdutos,
                new RecyclerItemClickListener.OnItemClickListener() {
                    @Override
                    public void onItemClick(View view, int position) {

                    }
                    @Override
                    public void onLongItemClick(View view, final int position) {
                        new AlertDialog.Builder(CampanhaActivity.this).
                                setTitle("Adicionar Produto").
                                setMessage("Deseja adicionar esse produto?").
                                setPositiveButton("Sim",
                        new DialogInterface.OnClickListener() {
                            @Override
                   public void onClick(DialogInterface dialogInterface, int i) {
                     Produto produtoSelecionado = produtos.get(position);


        ProdutosCampanha produtoCampanha = new ProdutosCampanha();
```

```
        produtoCampanha.setUid(UUID.randomUUID().toString());
        produtoCampanha.setUidProduto(produtoSelecionado.getUid());
        produtoCampanha.setNomeProCampanha(produtoSelecionado.getNome());
        produtoCampanha.setDescProdCampanha(produtoSelecionado.getDescricao());

 if (campanhaRecuperada == null) {
       campanhaRecuperada = new Campanha(idInstituicao);
       produtoCampanha.setUidCampanha(campanhaRecuperada.getUid());
  }

 if(adapterProdutosCampanha == null){

       produtosCampanha.add(produtoCampanha);
       campanhaRecuperada.setItens(produtosCampanha);
       produtoCampanha.setUidCampanha(campanhaRecuperada.getUid());

       adapterProdutosCampanha = new
       AdapterProdutosCampanha(produtosCampanha,CampanhaActivity.this);

       recyclerProdutosCampanhaAdd.setLayoutManager(new
       LinearLayoutManager(CampanhaActivity.this));
       recyclerProdutosCampanhaAdd.setHasFixedSize(true);
       recyclerProdutosCampanhaAdd.setAdapter(adapterProdutosCampanha);

} else if (adapterProdutosCampanha != null && extras != null) {

       produtosCampanhaRecuperados.add(produtoCampanha);
       campanhaRecuperada.setItens(produtosCampanhaRecuperados);
       produtoCampanha.setUidCampanha(campanhaRecuperada.getUid());
       adapterProdutosCampanha.notifyDataSetChanged();

} else if (adapterProdutosCampanha != null && extras == null) {

       produtosCampanha.add(produtoCampanha);
       campanhaRecuperada.setItens(produtosCampanha);
       produtoCampanha.setUidCampanha(campanhaRecuperada.getUid());
       adapterProdutosCampanha.notifyDataSetChanged();
      }
     }
  }).setNegativeButton("Não", null).show();
}

    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {

     }
  })
);
```

Fonte: Autoria própria.

No diretório ‘fragment’ estão as classes de *fragments*. No projeto os *fragments* são utilizados nas telas do perfil Administrador. Os *fragments* são as *views* responsáveis por listar as instituições não avaliadas, aprovadas ou

reprovadas. Na Listagem de código 5 está o código do *fragment* das instituições não avaliadas, responsável por aprovar ou reprovar as instituições.

Listagem de código 5 - Código fragment

```
private void reprovarInstituicao(final RecyclerView.ViewHolder viewHolder) {
    new AlertDialog.Builder(getActivity())
            .setTitle("Reprovar Instituição")
            .setMessage("Deseja realmente reprovar essa Instituição")
            .setPositiveButton("Sim", new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    Instituicao instituicao =
instituicoes.get(viewHolder.getAdapterPosition());
                    String idInst = instituicao.getUid();

ConfiguracaoFirebase.getFirebase().child("Instituicao").child(idInst).child("situa
cao").setValue("3");
                }
            })
            .setNegativeButton("Não", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    adapterInst.notifyDataSetChanged();
                }
            }).show();
}
private void atualizaInstituicoes() {
    recyclerViewInstNaoAvaliadas.addOnItemTouchListener(
            new RecyclerItemClickListener(
                    getContext(),
                    recyclerViewInstNaoAvaliadas,
                    new RecyclerItemClickListener.OnItemClickListener() {



                        @Override
                        public void onItemClick(View view, int position) {
                            exibeDadosInstituicao(position);
                        }

                        @Override
                        public void onLongItemClick(View view, int position) {
                            aprovarInstituicao(position);
                        }

                        @Override
                        public void onItemClick(AdapterView<?> adapterView, View
                        view, int i, long l) {  }
                    }
   ));
}

private void aprovarInstituicao(final int position) {

    new AlertDialog.Builder(getActivity())
```

```
            .setTitle("Aprovar Instituição")
            .setMessage("Deseja realmente aprovar essa Instituição")
            .setPositiveButton("Sim", new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    Instituicao instituicao = instituicoes.get(position);
                    String idInst = instituicao.getUid();

ConfiguracaoFirebase.getFirebase().child("Instituicao").child(idInst).child("situa
cao").setValue("2");
                    adapterInst.notifyDataSetChanged();
                }
            })
            .setNegativeButton("Não", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialogInterface, int i) {
                    adapterInst.notifyDataSetChanged();
                }
            }).show();
}
```

Fonte: Autoria própria.

Para o retorno dos dados do Firebase Realtime Database, são necessários *listeners*, funções que ficam “ouvindo” as alterações realizadas no banco de dados. Na listagem de código abaixo temos a codificação de uma função *listener* ligada a uma referência do banco de dados, a qual retorna dados da tabela instituições.

Listagem de código 6 - Código de listener vinculado à referência.

```
private void recuperaInstituicoes() {

    instituicaoRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            instituicoes.clear();
            for (DataSnapshot ds : dataSnapshot.getChildren()) {

                instituicoes.add(ds.getValue(Instituicao.class));
                adapterInst = new Adapter_instituicoes(instituicoes,
ListaInstituicaoActivity.this);
                recyclerInstituicoes.setLayoutManager(new
LinearLayoutManager(ListaInstituicaoActivity.this));
                recyclerInstituicoes.setHasFixedSize(true);
                recyclerInstituicoes.setAdapter(adapterInst);
            }
            Collections.reverse(instituicoes);
            adapterInst.notifyDataSetChanged();

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
```

```
        }
    });
}
```

Fonte: Autoria própria.

Os *listeners* podem também ser vinculados a *querys* do Firebase, para retornarem dados específicos, como apenas um usuário, ou uma instituição ou uma lista de instituições com uma determinada característica. E no *listener* da Listagem de código 7 a query a retorna uma lista de instituições relacionadas somente ao usuário autenticado.

Listagem de código 7 - Código de listener vinculado a query.

```
private void recuperaInstituicoes() {

    queryInstituicaoUsuarioRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            instituicoes.clear();
            for (DataSnapshot ds : dataSnapshot.getChildren()) {

                Instituicao inst = ds.getValue(Instituicao.class);

                if (inst.getSituacao().equals("2")){

                instituicoes.add(ds.getValue(Instituicao.class));
                adapterMyInst = new AdapterMyInst(instituicoes,
MinhasInstituicoesActivity.this);
                recyclerMinhasInstituicoes.setLayoutManager(new
LinearLayoutManager(MinhasInstituicoesActivity.this));
                recyclerMinhasInstituicoes.setHasFixedSize(true);
                recyclerMinhasInstituicoes.setAdapter(adapterMyInst);

                }
            }
            Collections.reverse(instituicoes);
            if (instituicoes.size() > 0) {

                adapterMyInst.notifyDataSetChanged();
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            System.out.println("Erro ao ler Instituições: " +
databaseError.getCode());

        }
    });
}
```

Fonte: Autoria própria.

5 CONCLUSÃO

Este documento apresentou o desenvolvimento de um aplicativo para celular utilizando a plataforma Android com a linguagem de programação Java e o banco de dados Firebase, para cadastro de dados relacionados a instituições de caridade e assistenciais, visando facilitar a criação de campanhas de doação, permitindo o cadastro de instituições, de usuários, o acompanhamento das ações e o recebimento de notificações, entre outras funcionalidades.

O principal objetivo da criação de uma ferramenta para essa área é aproximar instituições ou pessoas que necessitam de auxílio, que muitas vezes é emergencial, de pessoas ou empresas dispostas a realizar doações, e, além disso, promover um melhor aproveitamento momentâneo de recursos, destacando itens de maior necessidade que podem ser doados assertivamente.

Ao cadastrar-se no aplicativo desenvolvido, um cidadão, ou uma empresa, pode optar por realizar doações voluntárias a entidades, e estas, por sua vez, podem, de uma maneira facilitada, divulgar suas ações de campanha, ou seja, ambos os lados são favorecidos, tendo como resultado o fomento das chamadas “boas ações” feitas pela sociedade em geral.

Em relação ao desenvolvimento do aplicativo, os requisitos levantados foram atendidos, permitindo ao usuário escolher as instituições que ele deseja auxiliar e, assim, receber notificações das campanhas para doação dessas instituições; permitindo que instituições cadastrem campanhas com os produtos necessários em um determinado período; deixando que o usuário do tipo “instituição” cadastre contas bancárias nas quais poderá receber doações em espécie, e também ao usuário “administrador” aprovar ou não as instituições cadastradas, garantindo a existência e idoneidade de cada uma delas, e também a veracidade das campanhas abertas, para que então o aplicativo se torne totalmente confiável.

Foram realizados testes com pessoas que poderiam ser consideradas usuários comuns para identificação de possíveis problemas de usabilidade e funcionamento. Nestas situações a maioria das pessoas dispostas a testar a aplicação, apreciou a mesma e disseram estar abertas ao uso deste aplicativo, pois como já explicitado, traz conveniência, comodidade para e o alerta através das notificações. Mostraram também necessidades de melhorias, como nas telas de listas de instituições, apontando a falta de um filtro para realizar a busca por

entidades específicas ou por cidades. Na tela de cadastro de campanhas a melhoria do UX, pois na lista de produtos disponíveis para adicionar a campanha, a lista mostra apenas dois ou três produtos, sendo necessária realizar o "*scroll"* para visualizar os demais itens. Nesta tela ainda, também se faz necessário um filtro para a busca de um produto específico.

Pode-se citar como trabalho futuro a implementação de uma integração entre o aplicativo e o módulo GPS do aparelho móvel, para que seja possível identificar a localização de instituições e notificar seus usuários quando o mesmo estiver próximo a elas, ou quando próximo de algum estabelecimento no qual ele poderá adquirir produtos pertinentes a alguma campanha aberta. Também é possível realizar a integração do aplicativo com meios de pagamento via Banco Digital, para facilitar ainda mais a arrecadação em dinheiro.

REFERÊNCIAS

ANDROID DEVELOPERS. Disponível em: http://developer.android.com. Acesso em: 25 ago. 2018.

BRAY, Tim. **What Android is**. 2010. Disponível em: http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is. Acesso em: 9 set. 2018.

FIREBASE. Firebase. Disponível em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: 17 jun. 2020.

FITZSIMMONS, James; FITZSIMMONS, Mona. **Administração de serviços**: operações, estratégia e tecnologia da informação. São Paulo: AMGH Editora Ltda., 2014.

GUANA, Victor; ROCHA, Fabio; HINDLE, Abram; STROUL, Eleni. **Do the stars align? Multidimensional analysis of Android's layered architecture**. MSR 2012, p. 124-127, 2012.

INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Nove entre dez usuários de Internet no país utilizam aplicativos de mensagens**. Agência IBGE Notícias. 2018. Disponível em: https://agenciadenoticias.ibge.gov.br/agencia-noticias/2012-agencia-de-noticias/noticias/20077-nove-entre-dez-usuarios-de-internet-no-pais-utilizam-aplicativos-de-mensagens.html. Acesso em: 27 ago. 2018.

KLEINA, Nilton. **A história do Android, o robô que domina o mercado mobile**. 2017. Disponível em: https://www.tecmundo.com.br/ciencia/120933-historia-android-robo-domina-o-mercado-mobile-video.htm. Acesso em: 11 set. 2018.

LI, Yang; WANG, Xinning. **Design of adaptive media transmission based on Android platform**. In: 2014 IEEE International Conference on Consumer Electronics, China, p. 1-4, 2014.

OLIVEIRA, André Lucio de; VIANNA, Leonardo Soares; NASCIMENTO, Bruno Rebello do; VITA NETO, Miguel Laroca; SANTOS, Monique Gomes de Araújo. **Um estudo sobre o Sistema Operacional Android**. 2013. Disponível em: http://www.revista.universo.edu.br/index.php?journal=1reta2&page=article&op=view&path%5B%5D=1182&path%5B%5D=886. Acesso em: 11 set. 2018.

PRIMORAC, Sanja; RUSSO, Mladen. **Android application for sending SMS messages with speech recognition interface**. In: 5th International Convention MIPRO, p. 1763- 1767, 2012.

SILVA, Leandro Luquetti B. da; PIRES, Daniel Facciolo; CARVALHO NETO, Silvio. **Desenvolvimento de aplicações para dispositivos móveis**: tipos e exemplo de aplicação na plataforma iOS. II Workshop de Iniciação Científica em Sistemas de Informação, Goiânia - GO, 2015. Disponível em: www.lbd.dcc.ufmg.br/colecoes/wicsi/2015/004.pdf. Acesso em: 10 set. 2018.

XU, Wei; ZHANG, Fangfang; ZHU, Sencun. **Permlyzer: analyzing permission usage in android applications**. In: 24th International Symposium on Software Reliability Engineering (ISSRE). IEEE, p. 400-410, 2013.