

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
CENTRO DE CIÊNCIAS EXATAS, NATURAIS E DA SAÚDE**

**DEPARTAMENTO DE COMPUTAÇÃO  
CIÊNCIA DA COMPUTAÇÃO**

**VALMIR GOMES DE AGUIAR NETO**

**AppElo – APLICATIVO MÓVEL DE INTEGRAÇÃO E ASSISTÊNCIA SOCIAL NO  
ÂMBITO MUNICIPAL**

**ALEGRE  
2022**

VALMIR GOMES DE AGUIAR NETO

**AppElo – APLICATIVO MÓVEL DE INTEGRAÇÃO E ASSISTÊNCIA SOCIAL NO  
ÂMBITO MUNICIPAL**

Trabalho de Conclusão de Curso  
apresentado ao Departamento de  
Computação do Centro de Ciências  
Exatas, Naturais e da Saúde da  
Universidade Federal do Espírito Santo,  
como requisito parcial para obtenção do  
grau de Bacharel em Ciência da  
Computação.

Orientador: Prof.º M.Sc. Giuliano Prado De  
Morais Giglio

**ALEGRE**

**2022**

VALMIR GOMES DE AGUIAR NETO

## **AppElo – APLICATIVO MÓVEL DE INTEGRAÇÃO E ASSISTÊNCIA SOCIAL NO ÂMBITO MUNICIPAL**

Trabalho de Conclusão de Curso  
apresentado ao Departamento de  
Computação do Centro de Ciências  
Exatas, Naturais e da Saúde da  
Universidade Federal do Espírito Santo,  
como requisito parcial para obtenção do  
grau de Bacharel em Ciência da  
Computação.

### **COMISSÃO EXAMINADORA**

---

**Prof. M.Sc. Giuliano Prado de Moraes  
Giglio  
Universidade Federal do Espírito Santo  
Orientador**

---

**Prof<sup>a</sup>. Ma. Valéria Alves da Silva  
Universidade Federal do Espírito Santo**

---

**Prof. Dr. Bruno Vilela Oliveira  
Universidade Federal do Espírito Santo**

## RESUMO

Este trabalho apresenta o processo de desenvolvimento de um aplicativo para gerenciar pedidos de doação para o Centro de Referência de Assistência Social (CRAS), fornecendo informações do que se necessita ser doado, e podendo também a um outro usuário, cadastrar-se e realizar um pedido de doação, além de prover o acesso a campanhas de doação e notícias sobre doações no âmbito municipal. O acesso a doações, em geral, possui dificuldades, sobretudo na aproximação entre doadores e necessitados, tornando difícil tanto receber quanto oferecer ajuda, quanto ajudar. O aplicativo foi desenvolvido para tentar preencher a lacuna existente nestas informações, possibilitando que mais pessoas possam ajudar com doações as famílias em maior vulnerabilidade social. A aplicação foi desenvolvida utilizando a linguagem Kotlin, focada no sistema operacional Android, Interface de Programação de Aplicação (API – do inglês *Application Programming Interface*) criada com o Ktor e utilizando banco de dados PostgreSQL. O aplicativo móvel se propõe a facilitar o acesso às informações sobre pedidos de doação, e permite realizar um gerenciamento dessas doações, bem como possibilita ao usuário realizar ou receber doações de acordo com o seu perfil social, por meio de um cadastro simples, monitorado pelo Centro de Referência de Assistência Social (CRAS), melhorando a assistência social municipal à população mais necessitada, aumentando a acessibilidade e da sociedade à filantropia.

Palavras-chave: Assistência Social; CRAS; Android; Kotlin; PostgreSQL; Ktor; API.

## LISTA DE FIGURAS

Figura 1 – Exemplo de código do Kotlin Coroutines.....	22
Figura 2 – Comandos feitos em linguagem Kotlin utilizando a biblioteca Retrofit para realizar o consumo de uma API.....	22
Figura 3 – Exemplo de código de criação de um módulo utilizando a biblioteca Koin. .....	23
Figura 4 – Exemplo JSON.....	24
Figura 5 – Exemplo de modelagem de dados usando MySQL Workbench .....	25
Figura 6 – Arquitetura do sistema.....	29
Figura 7 – Diagrama do processo Scrum.....	29
Figura 8 – Processo de levantamento e análise de requisitos .....	30
Figura 9 – Diagrama de casos de uso do sistema proposto .....	33
Figura 10 – Diagrama de classe .....	34
Figura 11 – Modelo Conceitual do banco de dados .....	36
Figura 12 – Modelo Lógico.....	38
Figura 13 – Estrutura do projeto backend .....	40
Figura 14 – Propriedades da conexão .....	41
Figura 15 – Exemplo de operação Select feita pelo servidor .....	41
Figura 16 – Exemplo de route .....	42
Figura 17 – Exemplo de service.....	43
Figura 18 – Exemplo de entidade utilizando Exposed.....	43
Figura 19 – Estrutura do frontend.....	44
Figura 20 – Exemplo de requisição GET.....	46
Figura 21 – Mapa do Navigation Component do sistema.....	47
Figura 22 – Exemplo do código utilizado no navigation component.....	48
Figura 23 – Barra de navegação.....	48
Figura 24 – Tela de início com a listagem de pedidos .....	49
Figura 25 – Tela de listagem de campanhas e notícias.....	50
Figura 26 – Tela de usuário não logado .....	50
Figura 27 – Tela de login .....	51

Figura 28 – Tela de usuário com autenticação realizada pelo CRAS.....	51
Figura 29 – Telas de login do sistema .....	52
Figura 30 – Tela cadastro de usuários .....	53
Figura 31 – Tela de aprovação de instituições .....	54
Figura 32 – Tela de informações da instituição .....	55
Figura 33 – Mensagem de sucesso de aprovação de instituição .....	55
Figura 34 – Tela de pedido de doação em aberto. ....	56
Figura 35 – Mensagem de aviso de necessidade de login para realizar sinalização do pedido .....	57
Figura 36 – Mensagem de confirmação de sinalização do pedido.....	57
Figura 37 – Mensagem de conclusão da sinalização do pedido .....	57
Figura 38 – Telas de menu .....	58
Figura 39 – Tela cadastro de necessidade. ....	59
Figura 40 – Mensagem de sucesso exibida na tela de pedido de doação .....	60
Figura 41 – Tela de aprovação de pedidos de doação .....	60
Figura 42 – Tela de aprovação de pedido de doação.....	61
Figura 43 – Mensagem de sucesso na tela de aprovação de pedidos.....	62
Figura 44 – Tela de listagem de pedidos de doações sinalizadas.....	62
Figura 45 – Tela de conclusão de pedido de doação .....	63
Figura 46 – Mensagens de conclusão de pedido de doação .....	63
Figura 47 – Mensagem de conclusão do pedido de doação .....	64
Figura 48 – Tela de cadastro de ponto de coleta.....	64
Figura 49 – Mensagem de sucesso do cadastro de ponto de coleta .....	65
Figura 50 – Tela de cadastro de campanhas e notícias .....	65
Figura 51 – Mensagem de sucesso exibida ao cadastrar campanha ou notícia .....	66
Figura 52 – Tela de aprovação de usuário .....	66
Figura 53 – Mensagem de aprovação de usuário .....	67
Figura 54 – Tela de menu com menu de navegação com erro de seleção .....	74
Figura 55 – Telas de gerenciamento de contas do aplicativo. ....	75
Figura 56 – Menu de navegação inferior do sistema.....	76
Figura 57 – Listagem de pedidos de doação em aberto. ....	76
Figura 58 – Tela de listagem de campanhas e notícias.....	77
Figura 59 – Tela de menu do CRAS.....	78

## LISTA DE TABELAS

Tabela 1 – Usuários do sistema .....	31
Tabela 2 – Descrição dos dados da entidade Pedido.....	37
Tabela 3 – Descrição dos dados da entidade Doacao .....	37
Tabela 4 – Descrição estrutura do backend .....	39
Tabela 5 – Pacotes frontend.....	45
Tabela 6 – Resultados das Inspeções.....	68
Tabela 7 – Resultados dos Testes de Desenvolvimento.....	69
Tabela 8 – Resultados do teste de usabilidade .....	73
Tabela 9 – Descrição de dados da entidade Endereco .....	88
Tabela 10 - Descrição de dados da entidade Cidade .....	88
Tabela 11 - Descrição de dados da entidade PontoColeta .....	88
Tabela 12 - Descrição de dados da entidade Doacao .....	89
Tabela 13 - Descrição de dados da entidade Pedido .....	89
Tabela 14 - Descrição de dados da entidade Usuario .....	89
Tabela 15 - Descrição de dados da entidade Juridico .....	90
Tabela 16 - Descrição de dados da entidade Fisico .....	90
Tabela 17 - Descrição de dados da entidade Necessidade.....	90
Tabela 18 - Descrição de dados da entidade Categoria.....	90
Tabela 19 - Descrição de dados da entidade ResponsavelFamiliar .....	91
Tabela 20 - Descrição de dados da entidade CampanhaNoticia .....	91

## LISTA DE SIGLAS

CRAS	Centro de Referência de Assistência Social
SUAS	Sistema Única da Assistência Social
PNAS	Política Nacional de Assistência Social
LOAS	Lei Orgânica da Assistência Social
ONG	Organização Não Governamental
API	<i>Application Programming Interface</i>
CRUD	<i>Create Read Update Delete</i>
HTTP	<i>Hypertext Transfer Protocol</i>
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
JSON	<i>JavaScript Object Notation</i>
REST	<i>Representational State Transfer</i>
SGBD	Sistema de gerenciamento de banco de dados
UML	<i>Unified Modeling Language</i>
BD	Banco de Dados



## SUMÁRIO

1	INTRODUÇÃO .....	12
1.1	O problema e sua importância .....	13
1.2	Objetivos .....	14
1.2.1	Objetivo geral .....	14
1.2.2	Objetivos específicos .....	14
1.3	Motivação .....	15
2	REVISÃO DA LITERATURA .....	15
2.1	Sistemas de aplicações móveis .....	15
2.2	Aplicativos relacionados .....	16
2.2.1	Ribon .....	16
2.2.2	Joyz .....	18
2.2.3	Doar Fácil .....	19
2.2.4	Análise da proposta com as aplicações pesquisadas .....	19
2.3	Tecnologias Envolvidas .....	20
2.3.1	KOTLIN .....	20
2.3.2	JSON .....	23
2.3.3	PostgreSQL .....	24
2.3.4	KTOR Framework .....	24
2.3.5	MySQL Workbench .....	25
2.3.6	UML .....	25
3	METODOLOGIA .....	26
3.1	Arquitetura do sistema .....	26
3.1.1	API .....	28
3.1.2	Desenho da arquitetura .....	28
3.2	Metodologia de desenvolvimento .....	29

3.3	Análise dos requisitos .....	30
3.3.1	Descrição dos usuários .....	30
3.3.2	Requisitos funcionais.....	31
3.3.3	Casos de uso.....	32
3.3.4	Diagrama de classe .....	34
3.4	Modelagem de banco de dados .....	35
3.4.1	Descrição dos dados .....	37
4	DESENVOLVIMENTO .....	37
4.1	Modelo Lógico.....	38
4.2	BACKEND.....	39
4.2.1	Estrutura do Projeto .....	39
4.2.2	Conexão com o banco de dados .....	40
4.2.3	Endpoints.....	42
4.2.4	Service.....	42
4.2.5	Entidades.....	43
4.3	FRONTEND .....	44
4.3.1	Estrutura do projeto .....	44
4.3.2	Comunicação com o Servidor.....	46
4.3.3	Navegação.....	47
4.4	INTERFACES .....	48
4.4.1	Apresentação.....	49
4.4.2	Realizar login .....	52
4.4.3	Cadastro de Usuário .....	53
4.4.4	Aprovação de instituição.....	54
4.4.5	Sinalizar um pedido de doação.....	56
4.4.6	Realizar pedido de doação .....	58
4.4.7	Aprovação pedidos de doação .....	60

4.4.8 Concluir pedido de doação .....	62
4.4.9 Cadastro de pontos de coleta .....	64
4.4.10 Cadastro de campanhas e notícias.....	65
4.4.11 Aprovação de usuário.....	66
5 GARANTIA DE QUALIDADE E PROCESSOS DE TESTES.....	68
5.1 Teste de Usabilidade .....	72
5.2 Refinamento Visual.....	75
6 CONSIDERAÇÕES FINAIS E CONCLUSÕES .....	79
6.1 Trabalhos futuros .....	80
7 REFERÊNCIAS BIBLIOGRÁFICAS .....	82
APÊNDICE A – DESCRIÇÃO DOS DADOS .....	88
APÊNDICE B – CENÁRIOS DE CASOS DE USO.....	92

## 1 INTRODUÇÃO

O objetivo da assistência social é fornecer proteção social para as pessoas que necessitam. Historicamente, a assistência social foi a primeira política social pública importante que abriu caminho para o estado de bem-estar social. Hoje, essa assistência é de suma importância para a institucionalização dos direitos sociais, pois fornece uma linha base de seguridade social na qual ninguém deveria estar abaixo. (BAHLE; PFEIFER; WENDT, 2010)

Nas últimas décadas no Brasil houve um notável surgimento de formas de assistência social. O Brasil lidera entre os países latino americanos na redução da pobreza e na reversão de longo prazo das tendências crescentes de desigualdade. Um fator que contribui com essa redução da pobreza e desigualdade são os programas de assistência social. (BARRIENTOS, 2013)

Na análise da pobreza há duas dimensões dissociadas: a pobreza de renda e a de capacidades. A dimensão considerada prioritária é a de capacidades individuais para conduzir uma vida onde as necessidades básicas possam ser supridas. (MAURIEL, 2010)

Em meados dos anos 2000 foi desenvolvido o Sistema Único de Assistência Social (SUAS), abrangendo não somente a proteção aos mais pobres, mas também a garantia de direitos para populações em situação de vulnerabilidade por meio de serviços e benefícios. O texto constitucional e sua regulamentação pela Lei Orgânica da Assistência Social (LOAS), em 1993, adensaram a responsabilização do Estado como regulador e provedor das ofertas assistenciais. A configuração da assistência social como sistema descentralizado, previsto na LOAS, ganhou um novo patamar com a aprovação da Política Nacional de Assistência Social (PNAS) em 2004 e a Norma Operacional Básica do Sistema Único da Assistência Social (NOB/SUAS) em 2005. (JACCOUD; BICHIR; MESQUITA, 2017)

Esse serviço é executado, organizado e coordenado nos Centros de Referência de Assistência Social (CRAS), por intermédio do Programa de Atenção Integral à Família (PAIF). (MENEZES et al., 2007). O Centro de Referência de Assistência Social é uma unidade pública estatal descentralizada da política de assistência social, responsável pela organização e oferta de serviços da proteção social básica do Sistema Único de Assistência Social (SUAS) nas áreas de vulnerabilidade e risco

social dos municípios. O CRAS é a unidade em torno da qual se organizam os serviços de proteção básica, do que decorre sua função de gestão local. (ANDRADE; ALMEIDA; LIMA, 2009).

As unidades do CRAS ficam localizadas em regiões mais pobres das cidades e tem o objetivo de identificar as necessidades das famílias e acolher e inserir em atividades coletivas, ou, se necessário encaminhar os integrantes familiares para outros tipos de atendimento. (MENEZES et al., 2007)

O ponto importante é que não apenas recompensas materialistas, como comida ou dinheiro, mas também recompensas sociais imateriais, como aprovação social ou boa reputação, desempenham um papel fundamental na tomada de decisão social. Na literatura da psicologia e da economia, essa teoria forneceu uma resposta possível à questão do porquê os indivíduos ajudam os outros na ausência de um benefício material aparente. (IZUMA; SAITO; SADATO, 2010)

Foi gerado no Brasil um conceito chamado Tecnologia Social.

“Tecnologia Social é entendida como um conjunto de técnicas, metodologias transformadoras, desenvolvidas e/ou aplicadas na interação com a população e apropriadas por ela, que representam soluções para inclusão social e melhoria das condições de vida” (TECNOLOGIA..., 2021).

A relação do homem com a tecnologia é fundamentalmente uma relação de trocas onde o indivíduo busca a tecnologia, em última instância, como um instrumento que agrega conforto e felicidade. Há pouco mais de uma década, a tecnologia dos dispositivos móveis e seus indissociáveis aplicativos provocaram mudanças significativas na sociedade. Os aplicativos sociais, também chamados de aplicativos solidários, são ferramentas da tecnologia social, e surgem com finalidade alternativa ao desenvolvimento tecnológico focado na produção, no consumo, na vigilância e na gestão do indivíduo. (GODINHO et al., 2017).

## **1.1 O problema e sua importância**

Pessoas e institutos desejam doar coisas para organizações carentes. Além disso, muitas pessoas desejam solicitar doações. Porém, não existem muitas fontes disponíveis para que essas pessoas e instituições alcancem seus objetivos. (BELEKAR et al., 2021). Ao procurar por doações, tanto para solicitar como para realizar uma doação, surgem algumas dificuldades em encontrar informações, como

local mais próximo ou quais itens estão sendo doados ou sendo solicitados. À medida que as tecnologias móveis se tornam cada vez mais presentes, os aplicativos móveis de doações representam uma oportunidade como uma forma de atrair usuários e organizações como uma nova e crescente fonte de doações. (CHOI; KIM, 2016).

Atualmente, as complicações enfrentadas ao se tentar encontrar informações acerca de doações e de como realizá-las fazem com que o doador ou necessitado passem por várias páginas *web* ou telefones de contato até encontrar o que precisam.

## **1.2 Objetivos**

Nesta seção são apresentados os objetivos do aplicativo *mobile* a ser desenvolvido.

### **1.2.1 Objetivo geral**

O motivo para desenvolver o aplicativo móvel AppElo é aproximar famílias que necessitam de doações das informações de como receber essas doações e informar doadores onde devem ser feitas as doações. Nos dias de hoje grande parte das pessoas possuem um celular com acesso à internet, permitindo assim o acesso de famílias a serviços de solicitação de doações de forma online.

A partir da dificuldade de acesso as informações de doações, surgiu a necessidade desenvolver um aplicativo que facilitasse esse acesso. O aplicativo será feito para Android com intuito de facilitar interações entre instituições, doadores e necessitados de doação, onde todas informações ficarão disponíveis de uma forma mais intuitiva e facilitada pelo ponto de vista do acesso, permitindo a visualização rápida dessas informações estimulando doações e facilitando a conclusão do objetivo que é a aproximação das doações a pessoas necessitadas.

### **1.2.2 Objetivos específicos**

Além dos objetivos gerais apresentados, pretende-se atingir alguns objetivos específicos como o aumento da quantidade das doações, facilitar a divulgações de

campanhas de doações, sanar dúvidas existentes sobre a realização de doações para pessoas carentes e pesquisar sobre tecnologias diferentes das lecionadas no curso.

### **1.3 Motivação**

Atualmente aplicativos de doações são muito escassos e o acesso a informações sobre doações de objetos são ainda mais complicados, fazendo com que o usuário que procura como realizar uma doação fique navegando por muito tempo até conseguir achar o que procurava, o que pode tornar desanimador o processo de realizar uma doação.

Sites governamentais possuem informações confusas e que dificultam o acesso as mesmas, o que não permite o usuário encontrar mais informações nem de forma geral (informações sobre doações em vários lugares do país) e nem de forma a achar informações sobre locais e necessidades mais próximas a ele.

Observando-se esses fatores se destaca a necessidade de apresentar uma aplicação como uma proposta de melhoria ou alternativa ao cenário atual de informações sobre doações.

## **2 REVISÃO DA LITERATURA**

Nesta seção são apresentados os fundamentos teóricos que embasam o desenvolvimento do aplicativo.

### **2.1 Sistemas de aplicações móveis**

Os primeiros celulares comerciais surgiram na década de 80, porém não eram nada portáteis, possuíam preço elevado e tinham apenas a função de realizar ligações. Com o passar dos anos os celulares além de diminuir em tamanho ganharam novas funções. Hoje os celulares realizam uma variedade imensa de funções e tem preços cada vez mais acessíveis. Os *smartphones*, como são chamados os celulares inteligentes, possuem um sistema operacional para realizar o gerenciamento de todos os componentes do dispositivo. (MENDONÇA; BITTAR; DIAS, 2011)

Os sistemas operacionais são uns dos principais pontos pela ascensão dos dispositivos móveis, pois os aparelhos passaram a ser construídos para gerenciarem recursos de *hardware*, tratando desde o gerenciamento de aplicações simples até aplicações mais complexas, como por exemplo calcular rotas de navegação, usar sensores inteligentes, processamento de imagens, fazendo com que atividades antes impossíveis passassem a ser atividades corriqueiras, quebrando o estereótipo de que celulares eram construídos apenas para comunicação. (ARAÚJO, 2020)

O Android é o sistema operacional mais popular no Brasil e no mundo, sendo utilizado em 2,5 bilhões de dispositivos. (MUXFELDT, 2020). O Android é um Sistema Operacional Móvel *Open Source* desenvolvido inicialmente pela Google e possui uma arquitetura baseada em uma versão do Linux, principal vantagem do Android em relação ao iOS (sistema operacional desenvolvido pela Apple). (MENDONÇA; BITTAR; DIAS, 2011)

## 2.2 Aplicativos relacionados

Para realizar o desenvolvimento deste trabalho foram realizadas pesquisas em portais e aplicativos do Governo Federal e outras instituições sobre doações, com a finalidade de extrair os principais requisitos a serem implementados a aplicação móvel proposta. A partir da pesquisa que foi feita, é possível identificar que não há portais de doações para o serviço público onde o usuário pode se informar da localidade ou o que pode ser doado, assim como para o necessitado, não é possível o acesso às informações, porém, para ONGs e particulares, existem alguns aplicativos. Abaixo, são listados alguns desses portais e aplicativos, cada um com suas características:

### 2.2.1 Ribon

O aplicativo Ribon (RIBON, 2021) possui uma lista vertical de *cards*, que são elementos de design retangulares onde são exibidas informações, podendo ser desde texto a imagens, com causas para o usuário realizar uma doação. Em cada elemento da lista existem fotos sobre a causa, e um botão para realizar uma doação de “Ribons”, que são pontos dados ao usuário ao assinar o aplicativo. Para realizar doações, mesmo com pontos ganhos de forma gratuita, é necessário o *login* que pode ser feito



por meio da conta Google ou da conta Facebook do usuário. Assim que o login é feito e o usuário realiza a doação dos pontos iniciais ganhos de forma gratuita e é mostrada mais uma lista vertical de *cards*, elementos retangulares de design, com opções para doar mais. Com a assinatura mensal o usuário ganha pontos para realizar doações.

Porém, no *app* só existem as causas ali citadas, e não é possível encontrar informações sobre doações físicas para pessoas próximas a você e só é possível a doação de valores em dinheiro. Além disso, os pontos são derivados de doações feitas para o aplicativo Ribon e o aplicativo gera pontos onde estes são frações da doação realizada e os usuários decidem para onde a doação vai.

O aplicativo possui uma barra superior e uma inferior para navegar entre as páginas, onde os elementos são organizados da seguinte forma:

A barra superior possui os ícones de “Dúvidas”, “Notificações” e “Pontos Ribon” onde o ícone de “Dúvidas” leva para a página “Dúvidas Frequentes” onde são mostrados botões para as principais dúvidas e um botão para relatório de doações; o ícone de “Notificações” onde é sinalizado a quantidade de dias seguidos nos quais o usuário realizou uma doação; e por fim, o ícone “Pontos Ribon” no qual mostra a quantidade de pontos Ribon do usuário para a realização de doações e, ao ser selecionado, leva para a página “Ribons recebidos” onde existem mais informações sobre pontos Ribons e como conseguir mais pontos por meio de uma assinatura mensal.

A barra inferior possui os ícones de “Doações”, “Grupos”, “Assinaturas” e “Perfil”. O ícone “Doações” leva para página de “Doações” onde são listadas as causas onde é possível doar; o ícone “Grupos” leva para a página de “Grupos” onde o usuário pode visualizar os grupos onde ele participa e novos grupos onde pode participar para se juntar a outros usuários e bater metas de doação; o ícone “Assinaturas” onde leva para uma página onde são listados os valores de assinatura e respostas as perguntas frequentes sobre os pacotes de assinatura; e o ícone “Perfil” onde leva para página de usuário onde é mostrado o perfil do usuário, a quantidade de pontos Ribon que o usuário possui, causas em que o usuário realizou uma doação e um botão de relatório de doações.

O aplicativo Ribon tem uma abordagem na qual já é definida uma lista de causas que necessitam de doações. É possível comprar pontos dentro do aplicativo para realizar doações, porém, somente para causas já definidas no *app*. Também é possível ganhar pontos realizando missões diárias.

### 2.2.2 Joyz

Ao entrar no aplicativo a primeira vez é mostrada ao usuário uma tela onde ele pode entrar com uma conta já existente, entrar com a conta Facebook e realizar um cadastro. Na tela de cadastro o usuário coloca seus dados para realização do cadastro. No aplicativo é possível comprar pontos “joyz” para realizar doações.

O aplicativo da Joyz (JOYZ, 2021) possui uma barra superior e uma barra inferior onde existem alguns elementos para dividir as seções do *app* e os elementos são organizados da seguinte forma:

A barra superior possui os ícones “Ajuda” e “Pontos Joyz”. O ícone “Ajuda” leva para página de “Ajuda” onde são mostradas as principais dúvidas dos usuários e alguns vídeos sobre as dúvidas; o ícone “Pontos Joyz” mostra a quantidade de pontos joyz que o usuário possui e quando selecionado leva para a página “Loja” onde possui as seções “Ganhar” onde usuário pode ganhar pontos participando de pesquisas, “Comprar” onde o usuário pode realizar a compra de pontos e “Sacar” onde o usuário pode sacar os pontos que ganhou ao postar uma causa pra doação no *app*.

A barra inferior possui os ícones “Home”, “Pesquisa”, “Foto”, “Notificações” e “Perfil”. O ícone “Home” leva para página inicial onde são listadas causas que o usuário segue; o ícone “Pesquisa” leva a uma página com um ícone para realizar a pesquisa e as abas “Usuários” onde são listados os usuários que necessitam, “Tags” onde são listadas as principais *tags* e “Fotos” onde são listadas diversas fotos postadas pelos usuários; o ícone “Foto” onde permite ao usuário tirar uma foto para publicar no aplicativo; o ícone “Notificações” leva para a página de notificações onde possui as abas “Você” que lista as notificações relacionadas ao usuário e “Seguindo” que lista as notificações das causas e usuários que você segue; o ícone “Perfil” leva para a página de perfil do usuário que mostra as informações do usuário e permite editar as informações.

O aplicativo Joyz possibilita um usuário a criar uma causa e anexar imagens a ela. Outros usuários podem seguir essa causa criada, e podem doar pontos, que são comprados no *app*. Só é possível doar pontos, que basicamente, são valores em dinheiro encaminhados como pontos para o usuário necessitado.

### 2.2.3 Doar Fácil

No aplicativo Doar Fácil (DOAR FÁCIL, 2021) apresenta uma página inicial com barra de pesquisa, um *card* com um *slogan* para induzir doações uma lista horizontal com logo e nome de ONGs (Organizações não governamentais) para realizar doações, além de uma barra inferior com duas abas “Início” e “Novidades”.

Na barra inferior ao selecionar “Início” o usuário é levado para página inicial; ao selecionar “Novidades” você é levado para notícias relacionadas as ONGs cadastradas.

Ao selecionar uma ONG o usuário é levado para página da ONG onde é organizada da seguinte forma:

No topo da tela do *app* existe uma imagem da ONG; logo abaixo existe um botão “Quero doar” para realizar uma doação para esta ONG que leva o usuário para a página de doação onde é possível definir o valor da doação, se a doação será única ou mensal e a forma de pagamento; abaixo do botão existe uma descrição sobre a ONG com várias informações e redes sociais; existe uma barra inferior com os ícones “Início” que leva o usuário pra página principal da ONG, “Novidades” que leva para a página de notícias relacionadas a ONG selecionada e “Perfil” que leva para o perfil do usuário.

No aplicativo Doar Fácil são mostradas as ONGs cadastradas definidas pelo próprio *app*. É possível doar valores diretamente através do sistema.

### 2.2.4 Análise da proposta com as aplicações pesquisadas

O *app* proposto tem relações e diferenças com os citados. Alguns dos *apps* proveem pagamento, possibilidade de seguir uma causa específica, possibilidade de seguir outros usuários para acompanhar suas doações, divulgação de fotos por parte do necessitado, atividades que geram “moedas” que podem ser doadas, pesquisa por ONGs específicas.

O aplicativo móvel AppElo trabalha de uma forma diferente, pois é mais ligado as informações que serão passadas, locais para realizar a doação, telefones de contato com o local onde serão realizadas as doações e informações sobre campanhas de doação.

Algumas vantagens da aplicação móvel proposta são o rápido acesso a formas de contato, endereço para realização de doações e o que é necessário ser doado. Em outros *apps* só é possível realizar doações por meio de dinheiro e não é possível a doação de objetos. Porém outros *apps* trazem algumas formas de fidelizar o usuário por meio de atividades para ganhar valores para doação e o acompanhamento de causas, o que também pode induzir o usuário a doar regularmente.

## 2.3 Tecnologias Envolvidas

A aplicação proposta será feita em duas partes: o módulo *front-end* composto de telas e o módulo *back-end* contendo banco de dados e servidor. A parte da aplicação envolve o aplicativo no geral parte visual e algumas computações. A parte de *back-end* terá um servidor responsável por tratar e responder as requisições feitas pela aplicação, e também será responsável pela conexão com o banco de dados. Os dados serão passados para o *app* por meio de uma Interface de Programação de Aplicação (API – do inglês *Application Programming Interface*).

Nesta seção são apresentadas as tecnologias que serão utilizadas no desenvolvimento da aplicação proposta.

### 2.3.1 KOTLIN

Kotlin é uma linguagem de programação moderna, com tipagem estática e compatível com Android, e corrige alguns problemas presentes na linguagem Java, a qual anteriormente era a linguagem mais usada para desenvolver aplicativos Android. Inspirada em Swift, Scala, Groovy, C# e algumas outras linguagens, a linguagem Kotlin foi desenvolvida por profissionais da JetBrains, e foi concebida e analisada por diversos profissionais experientes. Kotlin é descrita como uma linguagem segura, expressiva, concisa, versátil e fácil de usar, além disso, possui interoperabilidade com Java e JavaScript. (MOSKALA; WOJDA, 2017)

Em maio de 2017, foi anunciado pela Google o suporte oficial para linguagem Kotlin na plataforma Android. Após o anúncio, a linguagem cresceu em popularidade, aparecendo em classificações como *Stack Overflow Annual Developer Surveys* 2018 e 2019. Também foi adotada na indústria de software rapidamente por empresas como

Pinterest (2022) e Uber (2022), não apenas em construção de aplicativos Android, mas também para fins internos. (OLIVEIRA; TEIXEIRA; EBERT, 2020)

A interoperabilidade com Java presente no Kotlin permite a capacidade de se usar APIs e bibliotecas Java. Como a linguagem Java já existe há mais de 20 anos, já existem diversas bibliotecas poderosas e extremamente bem adaptadas e essa interoperabilidade existente possibilita o Kotlin utilizá-las sem a necessidade de refazer tudo, além disso, já existem diversas bibliotecas feitas para a linguagem Kotlin. (SPÄTH, 2019)

As Interfaces de Programação de Aplicações, comumente chamadas de API, permitem que o desenvolvedor utilize conjuntos de comandos, funções, protocolos e diversas outras funcionalidades de aplicações já existentes em seu projeto. Isso possibilita agilizar operações comuns entre sistemas, economizando o esforço e a necessidade de se implementar tudo do zero. Além disso, facilita a comunicação, integração e interoperação entre aplicações. E as bibliotecas são coleções de implementações de comportamentos escritos em uma linguagem e importadas no código. Nesse caso, existe uma interface bem definida para cada comportamento invocado. (ZANETTE, 2017)

Epoxy é uma biblioteca *Open Source* feita pela empresa Airbnb que permite criar telas complexas em um *Recycler View* (*container* do *framework* Android que é feito para receber um conjunto de dados e criar uma lista) do Android. Essa biblioteca facilita o trabalho com listas e simplifica a construção de telas com vários tipos de visualizações. Além disso, a biblioteca oferece suporte para salvar o estado de exibição e diferenciar automaticamente as alterações de itens da lista. (HART, 2021)

A programação assíncrona é uma parte importante do cenário de desenvolvimento Android. Ao criar aplicações do lado do servidor, é importante fornecer uma experiência que não seja apenas fluida do ponto de vista do usuário, mas também escalável. O Kotlin resolve esse problema de maneira flexível por meio da biblioteca Kotlin Coroutines. A biblioteca Kotlin Coroutines é uma rica biblioteca para corrotinas desenvolvida pela JetBrains, o que permite realizar a execução de código assíncrono, melhorando a experiência e fluidez do sistema (JETBRAINS, 2022).

Figura 1 – Exemplo de código do Kotlin Coroutines

```

fun getCampaign(idCampaign: Int) {
    viewModelScope.launch { this: CoroutineScope
        requestCampaign(idCampaign)
    }
}

private suspend fun requestCampaign(idCampaign: Int) {
    withContext(Dispatchers.Main) { this: CoroutineScope
        try {
            campaignRepository.getCampaignById(idCampaign).let { it: CampaignModel
                model.value = CampaignModelResponse.Success(it)
            } ^withContext
        } catch (e: Exception) {
            model.value = CampaignModelResponse.Error(e)
            Log.e( tag: "app", msg: "error", e) ^withContext
        }
    }
}

```

Fonte: Do Autor

Retrofit é uma biblioteca de rede HTTP *type-safe* usada tanto no Android como no Java. É uma forma simples e rápida de se consumir API's no Android, economizando muito tempo e código ao fazê-lo. A biblioteca Retrofit vem melhorando muito desde que foi criada, se tornando cada vez mais rápida, oferecendo melhores funcionalidades e tendo sua sintaxe cada vez mais simples. Hoje uma grande parte dos desenvolvedores passaram a usá-la para fazer solicitações de API's. (KAYERE, 2021)

Figura 2 – Comandos feitos em linguagem Kotlin utilizando a biblioteca Retrofit para realizar o consumo de uma API

```

companion object {
    const val BASE_URL = "https://dudley-backbacon-53441.herokuapp.com/"
}

val appModule = module { this: Module
    single { this: Scope
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }
}

```

Fonte: Do Autor

E por último a biblioteca Koin. Koin é uma estrutura de *dependency injection* leve e pragmática utilizada na linguagem Kotlin, que possui a responsabilidade de instanciar diferentes objetos em toda aplicação. Usando *dependency injection* garante uma aplicação com acoplamento fraco, ou seja, uma classe depende de algumas abstrações e não diretamente de outra classe. (AMPIRE, 2019)

Figura 3 – Exemplo de código de criação de um módulo utilizando a biblioteca Koin.

```
val appModule = module { this: Module
    single { this: Scope
        Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }
    if (BuildConfig.DEBUG) {
        single { get<Retrofit>().create<ApiService>() }
    } else {
        single { TestApiService() }.bind<ApiService>()
    }
    single { this: Scope
        AndroidSqliteDriver(
            Database.Schema,
            androidContext(),
            name: "person.db"
        )
    }.bind<SqlDriver>()
    single { Database(get()) }
    single { DataRepository(get(), get(), Dispatchers.IO) }.bind<Repository>()
}
```

Fonte: Do Autor

### 2.3.2 JSON

JSON (*JavaScript Object Notation*) é um formato compacto que é utilizado para troca de dados. É baseado em um subconjunto do *JavaScript Programming Language Standard ECMA-262 3rd Edition* – dezembro de 1999.

JSON é um formato de texto livre de linguagem, porém, usa convenções que são familiares aos programadores das linguagens da família C, como C, C++, C#, Perl, Python, entre outras. Essas propriedades tornam o JSON uma linguagem de troca de dados ideal. (JSON, 2021). O formato JSON pode ser utilizado para enviar dados dos bancos de dados para as aplicações Android por meio de APIs.

Figura 4 – Exemplo JSON

```
{  
  "id" : "valmir.aguiar@example.com",  
  "first_name" : "Valmir",  
  "last_name" : "Aguiar",  
  "avatar" : "https://example.com/img/users/avatar-img.jpg"  
}
```

Fonte: Do Autor

### 2.3.3 PostgreSQL

PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional *open source*. Esse SGBD (Sistema de gerenciamento de banco de dados) é um dos principais usados no mundo por possuir uma arquitetura comprovada, confiabilidade, integridade de dados, um conjunto de recursos robusto e algumas outras coisas, além de uma dedicação da comunidade, pelo fato de ser *open source*, para fornecer soluções inovadoras e de alto desempenho de maneira consistente. (PostgreSQL, 2021)

### 2.3.4 KTOR Framework

Ktor (KTOR, 2021) é um *framework* produzido pela JetBrains utilizando Kotlin que permite criar APIs com protocolo HTTP que servem como estrutura interna para aplicativos, permitindo conectar o aplicativo ao banco de dados, facilitando assim, a manipulação dos dados feita pelo aplicativo. Isso facilita o trabalho e mapeamento dos

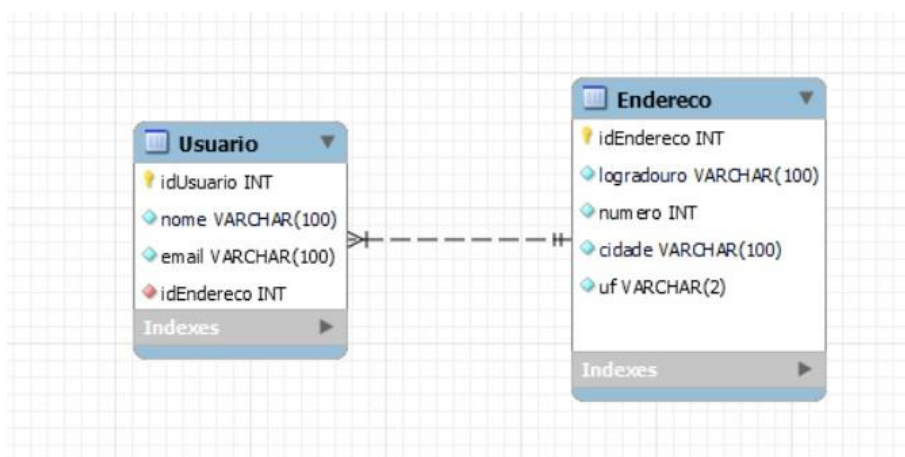


verbos estilo HTTP (GET, PUT, POST e DELETE) para URLs, serialização e desserialização utilizando o JSON no envio e recebimento de objetos.

### 2.3.5 MySQL Workbench

O MySQL Workbench (MYSQL, 2022) é uma ferramenta visual unificada utilizada por arquitetos de banco de dados, desenvolvedores e administradores de banco de dados. Esta ferramenta oferece muitas funções, como por exemplo, modelagem de dados. O MySQL Workbench permite que o utilizador projete, modele, gere e gerencie visualmente bancos de dados. Ele inclui tudo o que um modelador de dados precisa para criar modelos complexos de entidade relacionamento.

Figura 5 – Exemplo de modelagem de dados usando MySQL Workbench



Fonte: Do Autor

### 2.3.6 UML

A UML (*Unified Modeling Language*) é uma linguagem para visualização, especificação, construção e documentação de artefatos de software. Esta linguagem proporciona uma forma padrão para preparação de planos de arquitetura de sistemas, incluindo também aspectos conceituais tais como processos de negócios e funções do sistema. (BOOCH; RUMBAUGH; JACOBSON, 2006)

### 3 METODOLOGIA

A pesquisa utilizada no trabalho será de cunho qualitativo e exploratório. A partir da observação e levantamento dos requisitos gerais para solucionar o problema no acesso as informações sobre doações feitas por meio do serviço público, pretende-se utilizar a metodologia de desenvolvimento de sistemas descrita nas subseções seguintes, para a implementação do referido sistema.

Para a resolução dos problemas citados na seção 1.3, relacionados aos portais de doações do governo, foram levantados os principais requisitos para a criação do sistema, mediante a observação de aplicativos relacionados a doações como Ribon, Joyz e Doar Fácil, além de contemplar necessidades não abordadas nos respectivos *apps*.

O presente trabalho, do ponto de vista da natureza pode ser classificado como uma pesquisa exploratória aplicada. Do ponto de vista técnico este trabalho pode ser classificado como um estudo de caso.

#### 3.1 Arquitetura do sistema

A arquitetura de *software* é a forma como as partes de um sistema são organizadas, incluindo algumas questões, por exemplo: o comportamento de uma estrutura e quais componentes são responsáveis por realizar um específico conjunto de funções; o qual é um modelo repetível sob o qual um *software* pode ser desenvolvido (UNYLEYA, 2022).

No processo de desenvolvimento de um sistema, seu tamanho e complexidade escalam, e conseqüentemente surge a necessidade de se usar a arquitetura de software para manter sua robustez, desempenho, segurança e manutenibilidade (SOMMERVILLE, 2011).

A arquitetura de software aplicada ao desenvolvimento de aplicativos *mobile* tem como benefício principal a praticidade na implementação e a facilidade de uso, o que possibilita uma distribuição mais inteligente das responsabilidades na hora do desenvolvimento além de benefícios como escalabilidade, performance, disponibilidade, segurança e confiabilidade (KAZAP, 2020).

Segundo KAZAP (2020) os principais padrões utilizados na arquitetura *mobile* são:

- Arquitetura MVC (*Model-View-Controller*): o sistema é dividido em camadas, separando funcionalidades relacionadas à interface do usuário da lógica de negócios. A camada *model* é responsável por reter dados e pela manipulação da lógica de negócio, além de definir regras para modificar e operar dados. A camada *view* é relacionada a parte visual da aplicação. Por fim, a camada *controller* é a camada de intermediação entre as camadas *model* e *view*.
- Arquitetura MVP (*Model-View-Presenter*): é uma variação da arquitetura MVC onde o objetivo principal é separar a parte de apresentação das camadas de regras de negócio e dados, permitindo que o modelo se comunique diretamente com a *view*, sem precisar passar pelo controlador. A camada *model* armazena dados e lógica de negócio, comunicando banco de dados e a rede. A camada *view* é uma interface que exibe os dados e transfere a entrada do usuário para a camada *presenter*. A camada *presenter* é responsável por consultar a camada *model* e traduzir as atualizações na camada *view*.
- Arquitetura VIPER (*View-Interactor-Presenter-Entity-Router*): o código é dividido em módulos. O módulo *view* transfere a entrada do usuário para a camada *presenter*. O módulo *interactor* mantém a lógica de negócio. O módulo *presenter* mantém a lógica da *view* e prepara o conteúdo para exibição. O módulo *entity* contém objetos de modelo que são usados pelo *interactor*. E o módulo *router* que é responsável pela lógica de navegação.
- Arquitetura MVVM (*Model-View-ViewModel*): tanto a camada *view* como *model* agem da mesma forma que na arquitetura MVC, porém a camada *viewmodel* é uma espécie de intermediária entre as outras duas camadas. A camada *view* representa a interface do usuário e a camada *viewmodel* é responsável pela lógica de apresentação, capturando dados da camada *model*.

O sistema de software proposto utilizará o a arquitetura MVVM em seu desenvolvimento, o qual aumenta a facilidade de manutenção, acoplamento flexível, facilidade de alterações e simplificação de código, o que torna o sistema robusto, confiável e escalável.

Nas subseções seguintes são apresentados conceitos relacionados a arquitetura do sistema que será desenvolvido.

### 3.1.1 API

API (*Application Programming Interface*) é um conjunto de definições e protocolos para criar e integrar aplicações. Sua solução ou serviço podem se comunicar com outros produtos e serviços sem a necessidade de saber como foram implementados. Isso pode simplificar o desenvolvimento de aplicações, gerando economia tanto de tempo quanto de dinheiro (REDHAT, 2017).

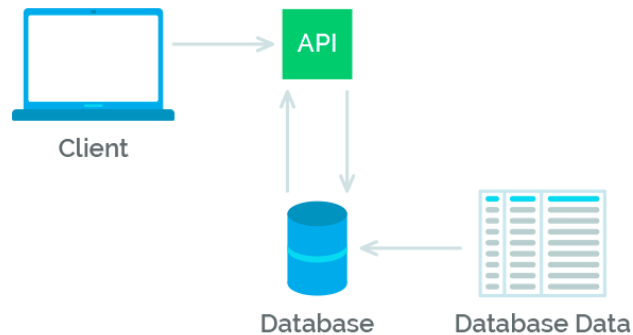
As APIs simplificam a forma como os desenvolvedores integram componentes de aplicação com uma arquitetura pré-existente. Elas funcionam como se fossem contratos, com uma documentação que representa um acordo entre as partes interessadas, além de possibilitar o compartilhamento de dados com clientes e usuários externos (REDHAT, 2017).

### 3.1.2 Desenho da arquitetura

A arquitetura do sistema seguirá o conceito que é geralmente utilizando em sistemas de software quando desenvolvidos com o *back-end* e *front-end* separados. O *back-end* será a API HTTP desenvolvida utilizando o Ktor, no qual se comunica com o banco de dados, e o *front-end* será a aplicação mobile que será desenvolvida utilizando o Kotlin.

Conforme a figura 5, o *back-end* é representado pela API que se comunica com o banco de dados. A aplicação se comunica com a API por meio de requisições HTTP, onde a API recebe as requisições, as processa e o servidor se comunicará com o banco de dados e retornará uma resposta para aplicação.

Figura 6 – Arquitetura do sistema



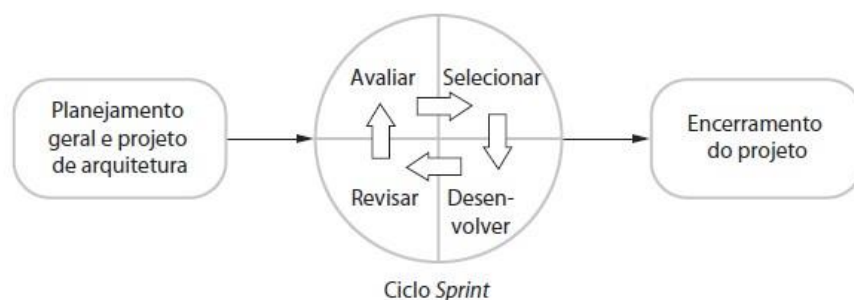
Fonte: Pires (2018)

### 3.2 Metodologia de desenvolvimento

No contexto de desenvolvimento de software, métodos de desenvolvimento ágeis foram originalmente concebidos por pequenas equipes de desenvolvedores de software. Como a aceitação a esses métodos cresceu, eles foram apresentados a uma infinidade de configurações diferentes para se adaptar a diversos contextos. Um dos métodos ágeis mais conhecidos é o Scrum (HRON; OBWEGESER, 2018).

No Scrum existem três fases. A primeira fase é a fase de planejamento geral, onde se estabelecem os objetivos gerais do projeto e arquitetura do sistema. Em seguida, ocorrem intervalos de tempo para desenvolvimento de *releases* do sistema. Na sua última fase onde se encerra o projeto, completa-se a documentação do sistema e avalia-se as lições aprendidas com o projeto (SOMMERVILLE, 2011).

Figura 7 – Diagrama do processo Scrum



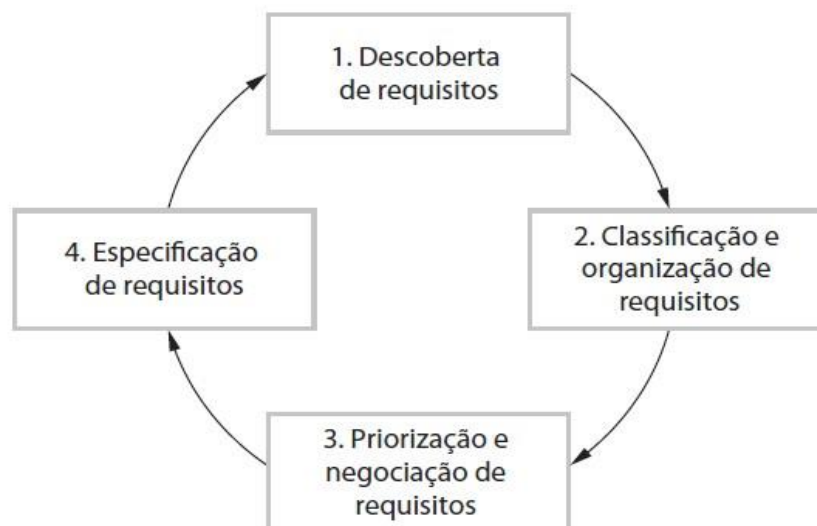
Fonte: Sommerville (2011)

### 3.3 Análise dos requisitos

Segundo Sommerville (2011), após a verificação inicial de viabilidade, o próximo estágio do processo de engenharia de requisitos é o levantamento e análise de requisitos. As atividades desta parte do processo são:

- Descoberta de requisitos, onde ocorre a interação com as partes interessadas para descobrir seus requisitos.
- Classificação e organização de requisitos, onde são tomadas as coleções de requisitos não estruturados, agrupados e organizados em grupos coerentes.
- Priorização e negociação de requisitos, no qual busca-se encontrar e resolver conflitos por meio da negociação de requisitos com as partes envolvidas.
- Especificação de requisitos, os quais são documentados e inseridos no próximo ciclo.

Figura 8 – Processo de levantamento e análise de requisitos



Fonte: Sommerville (2011)

#### 3.3.1 Descrição dos usuários

Do ponto de vista da modelagem, os usuários do sistema são vistos como atores. Um ator é a representação de um usuário que interage e exerce algum papel

em relação ao sistema. Cada ator está ligado a determinadas funcionalidades e permissões no sistema.

Na aplicação proposta, teremos um ator Coordenador, que exercerá o papel de coordenador do local onde serão realizadas as doações, além do Coordenador temos o ator Funcionário que representa o papel de funcionário do local onde são realizadas as doações e o ator Público que representa qualquer pessoa que queira utilizar a aplicação.

Tabela 1 – Usuários do sistema

CRAS	Representa o Centro de Referência de Assistência Social
Usuário	Representam os usuários que doam e pedem doações
Instituição	Representa as instituições que divulgam doações
Visitante	Representa o usuário que não realizou autenticação no sistema

Fonte: Do Autor

### 3.3.2 Requisitos funcionais

Requisitos funcionais dão declarações de serviços que o sistema deve oferecer, reagir e se comportar diante determinadas situações, e em alguns casos, também pode explicitar o que o sistema não deve fazer (SOMMERVILLE, 2011).

A seguir estão descritos os requisitos funcionais do sistema.

RF01 – Autenticação de usuário do tipo Usuário – O sistema deverá permitir um usuário realizar autenticação para acessar funções extras do aplicativo.

RF02 – Autenticação de usuário do tipo Instituição – O sistema deverá permitir um usuário do tipo Instituição realizar autenticação para acessar funções extras do aplicativo.

RF03 – Autenticação do CRAS – O sistema deverá permitir o CRAS realizar autenticação para acessar funções extras do aplicativo.

RF04 – Cadastro de Doador/Necessitado – O visitante, o CRAS, o usuário ou instituição poderá cadastrar um usuário do tipo doador/necessitado.

RF05 – Cadastro de Instituição – O visitante poderá se cadastrar como um usuário do tipo instituição.

RF06 – Criar de Pedido de Doação – O CRAS, o usuário e a instituição poderão cadastrar um pedido de doação.

RF07 – Autorização de Pedido de Doação – O CRAS poderá autorizar um pedido de doação.

RF08 – Aprovação de usuário – O CRAS poderá aprovar um usuário.

RF09 – Aprovação de instituição – O CRAS poderá aprovar uma instituição.

RF10 – Sinalizar doação – O usuário poderá sinalizar uma doação.

RF11 – Concluir pedido de doação – O CRAS poderá concluir um pedido de doação.

RF12 – Visualizar pedidos de doação – O visitante, o CRAS, o usuário e a instituição poderão visualizar pedidos de doação em aberto e pedidos de doação concluídos.

RF13 – Visualizar campanhas e notícias – O visitante, o CRAS, o usuário e a instituição poderão visualizar as campanhas e notícias cadastradas.

RF14 – Cadastro de campanhas e notícias – O CRAS e a instituição poderão cadastrar uma nova campanha ou notícia.

RF15 – Visualizar pedidos de doação feitas pelo usuário – O usuário e a instituição poderão visualizar os pedidos de doação submetidos por eles que estão aprovados ou reprovados.

RF16 – Visualizar Pontos de coleta – Ao visualizar um pedido, o visitante, o CRAS, o usuário e a instituição poderão visualizar os pontos de coleta cadastrados na instituição onde deverá ser realizada a doação.

RF17 – Cadastrar pontos de coleta – O CRAS e a instituição poderão cadastrar pontos de coleta.

RF18 – Visualizar pedidos de doação concluídos – O CRAS poderá visualizar os pedidos de doação concluídos.

RF19 – Visualizar pedidos de doação sinalizados – O CRAS poderá visualizar os pedidos de doação sinalizados.

### **3.3.3 Casos de uso**

Os casos de uso identificam os atores envolvidos em uma interação e nomeia essa interação. O conjunto de casos de uso representa todas as interações do sistema, sendo interações individuais entre o sistema e seus usuários ou até outros sistemas (SOMMERVILLE, 2011).

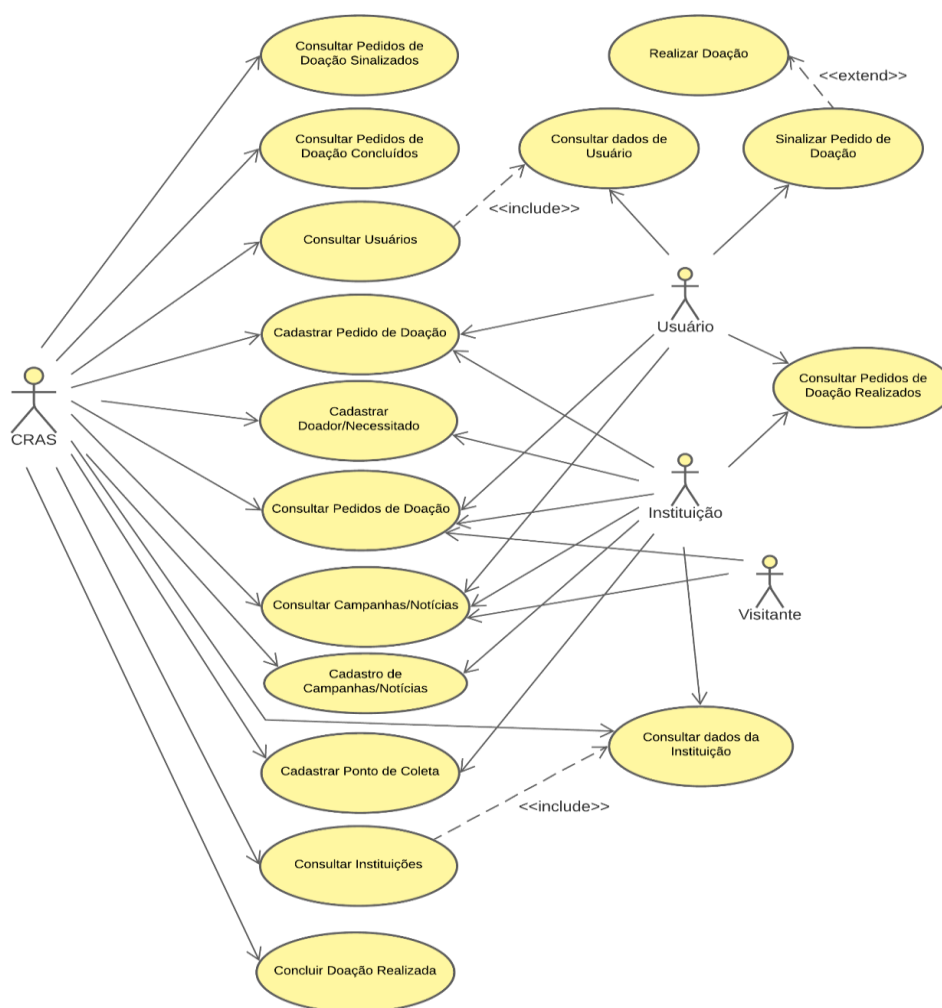


A figura 8 ilustra o diagrama de casos de uso do sistema proposto. O CRAS faz a publicação de assistidos, a instituição faz a publicação das necessidades de doação, e tanto o CRAS como as Instituições podem confirmar o recebimento de doações.

O Público confirma ajuda, para assim, realizar a doação, enquanto a doação não for confirmada pelo CRAS ou pela Instituição o pedido fica em aberto. O assistido ao receber a doação O CRAS ou a Instituição responsável confirma o recebido da doação, notificando então sobre a doação entregue.

As especificações e detalhamento dos casos de uso serão feitas após a validação dos requisitos e a simulação preliminar do funcionamento do *app* junto ao CRAS, o qual não houve possibilidade de ser feito a tempo da qualificação desta proposta, logo, será realizada em reunião marcada com a instituição, de acordo com a disponibilidade da mesma.

Figura 9 – Diagrama de casos de uso do sistema proposto





a objeto e modela os dados usando objetos e seus relacionamentos (SOMMERVILLE, 2011).

### **3.4 Modelagem de banco de dados**

Basicamente o projeto de banco de dados especifica a estrutura e o comportamento do banco. O ponto de partida se dá a partir dos requisitos de informação e as regras de negócio referentes ao domínio do problema, utilizando-se de ferramentas de projeto ou modelagem, para procurar atender uma série de critérios de qualidade.

Segundo Heuser (2009), o projeto de um BD (banco de dados) se dá em duas fases:

- Modelagem conceitual, onde se constrói um modelo conceitual na forma de um diagrama entidade-relacionamento.
- Projeto lógico, onde objetiva-se transformar o modelo conceitual em modelo lógico.

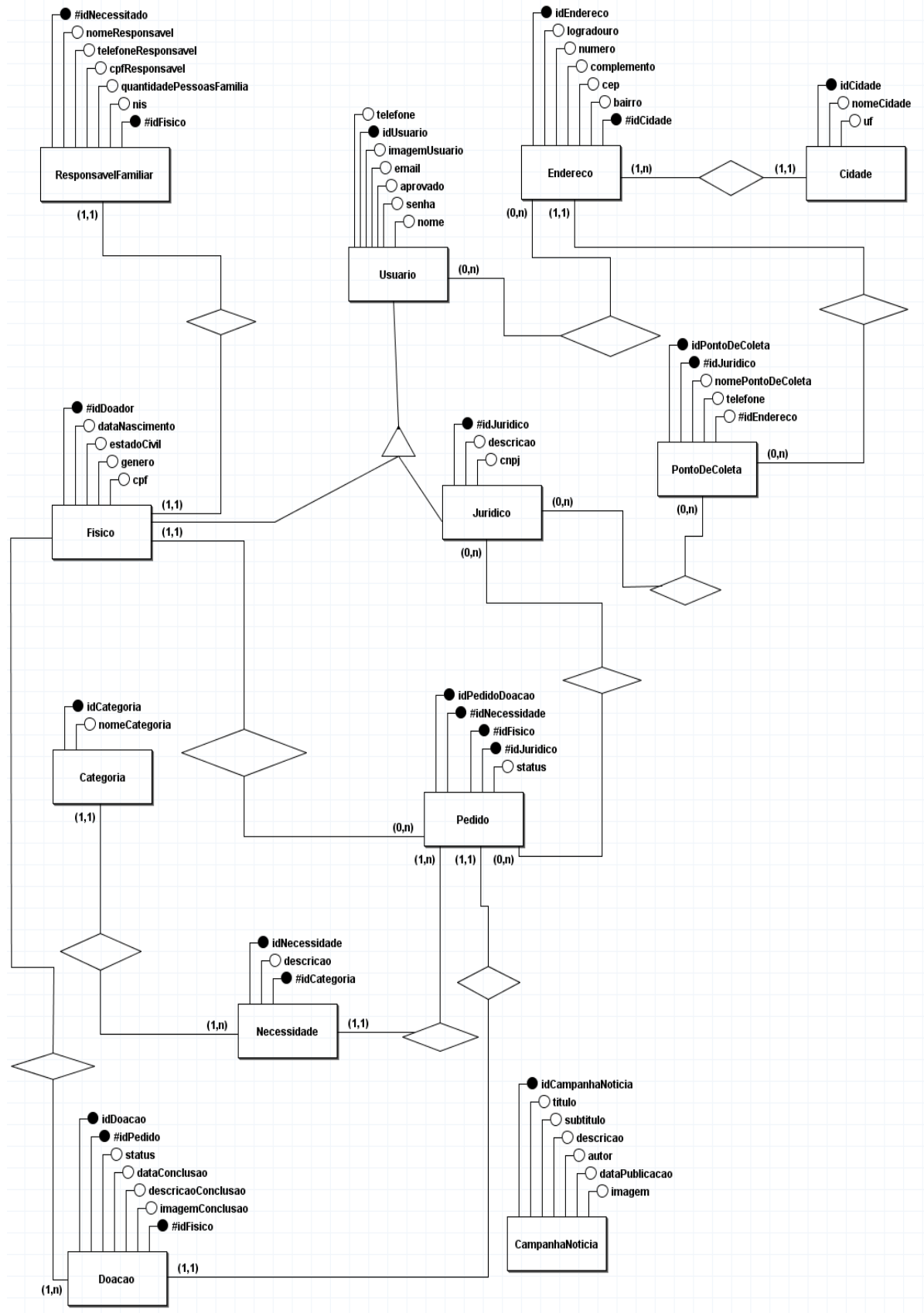
Os principais conceitos do diagrama entidade-relacionamento são as entidades, relacionamento, atributos, generalização/especialização e entidades associativas (HEUSER, 2009).

A figura 11 mostra o diagrama entidade-relacionamento do sistema proposto. O modelo conceitual proposto representa um banco de dados que visa atender um sistema de locais de doação, logo vários locais utilizariam o mesmo banco de dados.

Na Figura 11 é possível visualizar as entidades Físico e Jurídico que representam usuários comuns e instituições. Pode-se ver que ao haver um pedido, dados como informações do necessitado e da instituição onde foi cadastrado o pedido ficam na entidade Pedido.

Ao haver uma sinalização de doação, os dados do usuário que sinalizou o pedido são armazenados na entidade Doacao (Figura 11), na qual também possui um campo de status para identificar se a doação foi concluída ou não. Na entidade Pedido também há um campo de status, que significa se o pedido de doação foi aprovado ou não pelo CRAS.

Figura 11 – Modelo Conceitual do banco de dados



Fonte: Do Autor

### 3.4.1 Descrição dos dados

Para deixar mais claro as características dos dados armazenados, se faz necessário a descrição dos dados. Os dados serão descritos em tabelas, onde cada tabela descreverá os dados de uma entidade, a primeira coluna será o nome do atributo, a segunda coluna o tipo de dado do atributo, a terceira coluna indica se o atributo é chave primária ou chave estrangeira e por último uma coluna descrevendo o atributo.

A tabela 2 exemplifica a descrição dos dados de cada entidade do modelo conceitual proposto.

Tabela 2 – Descrição dos dados da entidade Pedido.

pedido			
Atributo	Tipo de dado	(PK/FK)	Descrição
idPedido	inteiro	PK	Identificador
imagemFamilia	texto		Imagem da família
idNecessidade	inteiro	FK	Identificador da necessidade
idJuridico	inteiro	FK	Identificador da unidade
idFisico	inteiro	FK	Identificador do necessitado
status	caractere		Status atual do pedido

Fonte: Do Autor

Tabela 3 – Descrição dos dados da entidade Doacao

doacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
idDoacao	inteiro	PK	Identificador
idPedido	inteiro	FK	Identificador do pedido sinalizado
status	caractere		Status da doação
dataConclusao	data		Data da conclusão
descricaoConclusao	texto		Descrição da conclusão
imagemConclusao	texto		Imagem da conclusão

Fonte: Do Autor

No apêndice A, são apresentadas as descrições das demais entidades do modelo conceitual proposto.

## 4 DESENVOLVIMENTO

Como resultado da modelagem e dos estudos sobre Android e da linguagem



sinalizado para doação então temos na tabela Doacao onde possui o identificador do pedido, status de conclusão, e a data da conclusão.

Para o desenho do modelo lógico foi utilizado a ferramenta MySQL Workbench, por apresentar visual fácil e intuitivo, fácil manuseio e por ser uma ferramenta poderosa e amplamente utilizada (MYSQL, 2022).

## 4.2 BACKEND

Seguindo a arquitetura definida na Figura 5 anteriormente apresentada, o servidor *backend* é responsável por atender às requisições do *frontend*, conter a lógica de negócio e realizar a comunicação com o banco de dados. O *backend* disponibiliza uma API onde os consumidores podem fazer requisições, dessa forma o *backend* fica totalmente desacoplado do *frontend*, tendo a possibilidade do *backend* atender a diferentes *frontends*, simultaneamente, caso haja necessidade, ou de ser atendido em sua forma de API, podendo também ser usado por outros sistemas.

### 4.2.1 Estrutura do Projeto

A Figura 13 mostra a estrutura de pacotes da aplicação Kotlin desenvolvida e a Tabela 4 descreve a função dos principais pacotes.

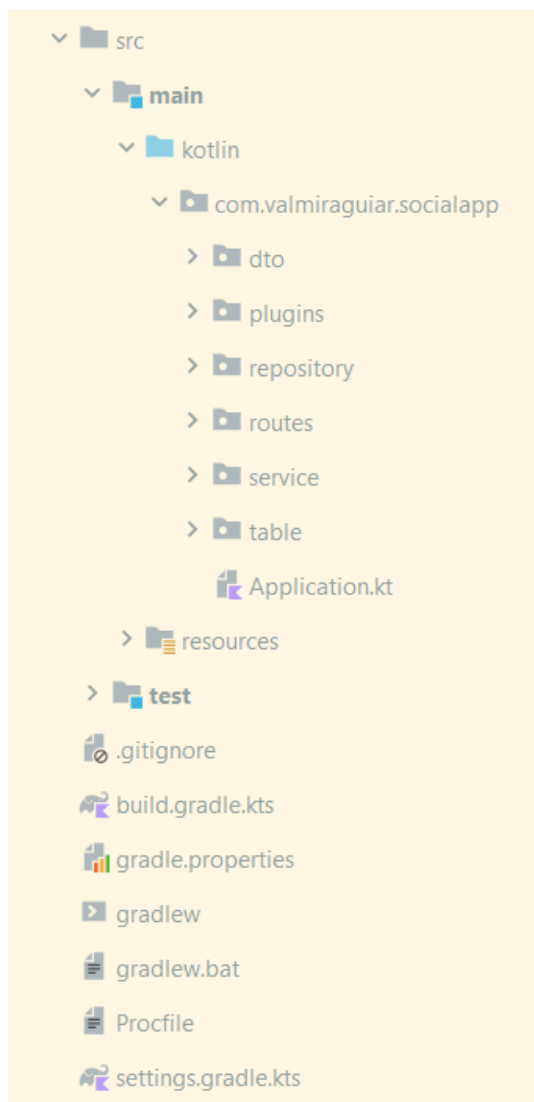
Tabela 4 – Descrição estrutura do *backend*

Pacote	Função
dto	Contém os objetos usados nas transferências de dados.
plugins	Contém objetos usados para configurações de funcionalidades das listagens do sistema.
repository	Contém as interfaces de comunicação com o banco de dados.
routes	Contém as entidades responsáveis pelas requisições HTTP.
service	Contém regras de negócio.

table	Contém entidades de tabelas que referenciam as tabelas do banco de dados.
-------	---

Fonte: Do Autor

Figura 13 – Estrutura do projeto *backend*



Fonte: Do Autor

#### 4.2.2 Conexão com o banco de dados

A conexão do servidor Kotlin com o banco de dados é realizada por meio da biblioteca Hikari e do driver do SGBD PostgreSQL. Os dados de conexão são informados no arquivo “DatabaseFactory.kt” como mostra a Figura 14, tornando fácil a alteração de dados relacionados a conexão com o banco.



Figura 14 – Propriedades da conexão

```

fun hikari(): HikariDataSource {
    val config = HikariConfig()
    config.driverClassName = System.getenv( name: "JDBC_DRIVER")
    config.maximumPoolSize = 3
    config.isAutoCommit = false
    config.transactionIsolation = "TRANSACTION_REPEATABLE_READ"

    val uri = URI(System.getenv( name: "DATABASE_URL"))
    val username = uri.userInfo.split( ...delimiters: ":" ).toTypedArray()[0]
    val password = uri.userInfo.split( ...delimiters: ":" ).toTypedArray()[1]

    config.jdbcUrl =
        "jdbc:postgresql://" + uri.host + ":" + uri.port + uri.path + "?sslmode=require" + "&user=$username&password=$password"

    config.validate()

    return HikariDataSource(config)
}

suspend fun <T> dbQuery(block: () -> T): T =
    withContext(Dispatchers.IO) { this: CoroutineScope
        transaction { block() }
    }

```

Fonte: Do Autor

Por meio de herança da classe Table presente no Ktor é possível fazer referência a tabela do banco, conseguindo assim, resgatar os dados desejados.

Figura 15 – Exemplo de operação Select feita pelo servidor

```

override suspend fun getAll(): List<DonationRequestDTO> {
    return DatabaseFactory.dbQuery {
        DonationRequestTable.select { this: SqlExpressionBuilder
            DonationRequestTable.aprovado.eq( t: true ) and
            DonationRequestTable.sinalizado.eq( t: false )
        }.mapNotNull { it: ResultRow
            DonationRequestTable.toDonationRequest(it)
        }
    }
}

```

Fonte: Do Autor

### 4.2.3 Endpoints

Para atender as requisições, o servidor foi desenvolvido utilizando o estilo arquitetural REST, onde o mesmo se torna a API com *endpoints* a serem consumidos por outras aplicações.

O acesso ao servidor é disponibilizado por meio de *routes*, que são classes que expõem ações e informações para consumidores da API, todas essas informações trocadas são feitas utilizando o formato JSON.

Um *endpoint* é simplificadaamente um recurso da aplicação exposto para que possa ser consumido por clientes externos.

Figura 16 – Exemplo de route

```
get( path: "/instituicao/all") { this: PipelineContext<Unit, ApplicationCall>
  try {
    val instituicoes = db.getAll()

    instituicoes.let { it: List<InstituicaoDTO>
      call.respond(status = HttpStatusCode.OK, it)
    }
  } catch (e: Throwable) {
    application.log.error("Failed to get all institutions", e)
    call.respond(HttpStatusCode.BadRequest, message: "Problems to get institutions")
  }
}
```

Fonte: Do Autor

### 4.2.4 Service

Os *services* têm a responsabilidade de fazer toda a regra de negócio da aplicação, além de ser responsável por se comunicar com as camadas internas da aplicação. Neste sistema o *service* envia dados para a camada *Repository*, onde é feita a persistência dos dados.

Figura 17 – Exemplo de service

```

class EnderecoService (
    val enderecoRepository: EnderecoRepository
) {

    fun updateById(id: Int, endereco: EnderecoDTO) {
        enderecoRepository.updateById(id, endereco)
    }

}

```

Fonte: Do Autor

#### 4.2.5 Entidades

As entidades são objetos que representam uma abstração do domínio do problema, contendo atributos e métodos que condizem com as necessidades do sistema. O controle e mapeamento das entidades é feito pelo *framework Exposed*. O *Exposed* possui uma variedade de mecanismos de banco de dados para ajudar na criação de *apps* (EXPOSED, 2022).

Figura 18 – Exemplo de entidade utilizando Exposed

```

object PedidoDoacao : Table( name: "pedido_doacao" ) {
    val idPedidoDoacao = integer( name: "id_pedido_doacao" )
    val idNecessidade = integer( name: "id_necessidade" )
    val dataPedidoDoacao = date( name: "data_pedido_doacao" )
    val idNecessitado = integer( name: "id_necessitado" )
    val aprovado = bool( name: "aprovado" )
    val reprovado = bool( name: "reprovado" )
    val sinalizado = bool( name: "sinalizado" )
    val concluido = bool( name: "concluido" )
    val idUnidade = integer( name: "id_unidade" )
    val idDoador = integer( name: "id_doador" )
}

```

Fonte: Do Autor

### 4.3 FRONTEND

O *frontend* da aplicação foi desenvolvido utilizando linguagem Kotlin e o *framework* Android. Atualmente o Android utiliza-se de arquivos .xml para desenvolver os layouts. Possuindo vários componentes como botões, *radio buttons*, menus, barras de navegação e muitos outros elementos.

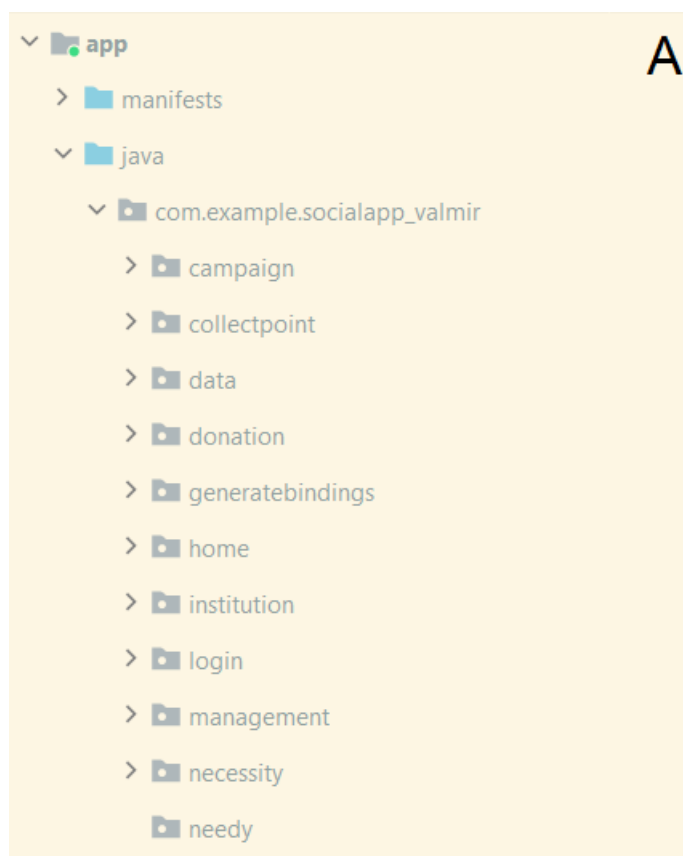
No Android é definida uma atividade principal (*MainActivity*) e a navegação é feita por meio de fragmentos (*fragments*), facilitando o fluxo entre as telas com um mapa de navegação.

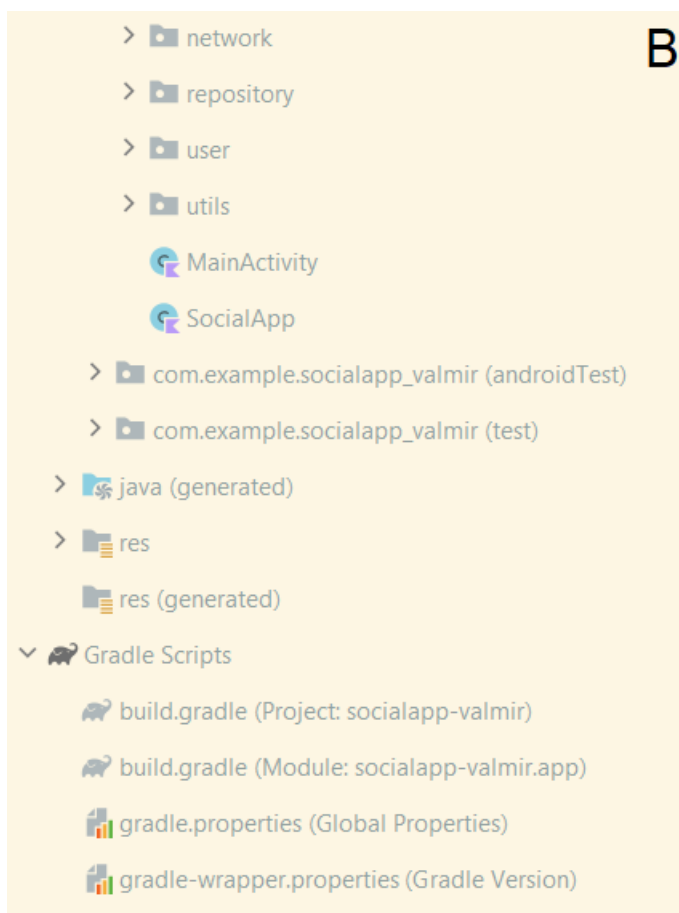
#### 4.3.1 Estrutura do projeto

A Figura 19 mostra a estrutura da aplicação Android desenvolvida e a Tabela 5 descreve os pacotes.

Figura 19 – Estrutura do frontend

A) Primeira parte da tela dos pacotes do sistema. B) Segunda parte da tela dos pacotes do sistema





Fonte: Do Autor

Tabela 5 – Pacotes frontend.

Pacote	Função
campaign	Contém as classes de View e ViewModel das campanhas.
collectpoint	Contém as classes de View e ViewModel dos pontos de coleta.
data	Contém as entidades utilizadas nos componentes para respostas.
donation	Contém as classes de View e ViewModel dos pedidos de doação.
generatebindings	Contém arquivos de configuração da biblioteca Epoxy.
home	Contém as classes de View e ViewModel da home.

institution	Contém as classes de View e ViewModel da instituição.
login	Contém as classes de View e ViewModel do login.
management	Contém as classes de View e ViewModel do menu.
necessity	Contém as classes de View e ViewModel da necessidade.
network	Contém arquivos de comunicação com a API.
repository	Contém arquivos de manipulação de models.
user	Contém as classes de View e ViewModel da home.
utils	Contém classes com utilidades específicas.

Fonte: Do Autor

#### 4.3.2 Comunicação com o Servidor

A comunicação do frontend com o servidor é realizada por meio de chamadas HTTP usando a biblioteca Retrofit. Como mostra a Figura 2, o Retrofit é configurado informando o endereço para onde serão realizadas as requisições.

A Figura 20 mostra um exemplo de requisição GET onde é passado o caminho do endpoint e o parâmetro id. Assim que o servidor responde, o Retrofit fornece a resposta onde podem ser extraídos os dados necessários.

Figura 20 – Exemplo de requisição GET

```
@GET("/donation_request/concluido/all/{id}")
suspend fun getAllDonationRequestConcluidoById(@Path("id") id: Int): List<DonationRequestDto>
```

Fonte: Do Autor

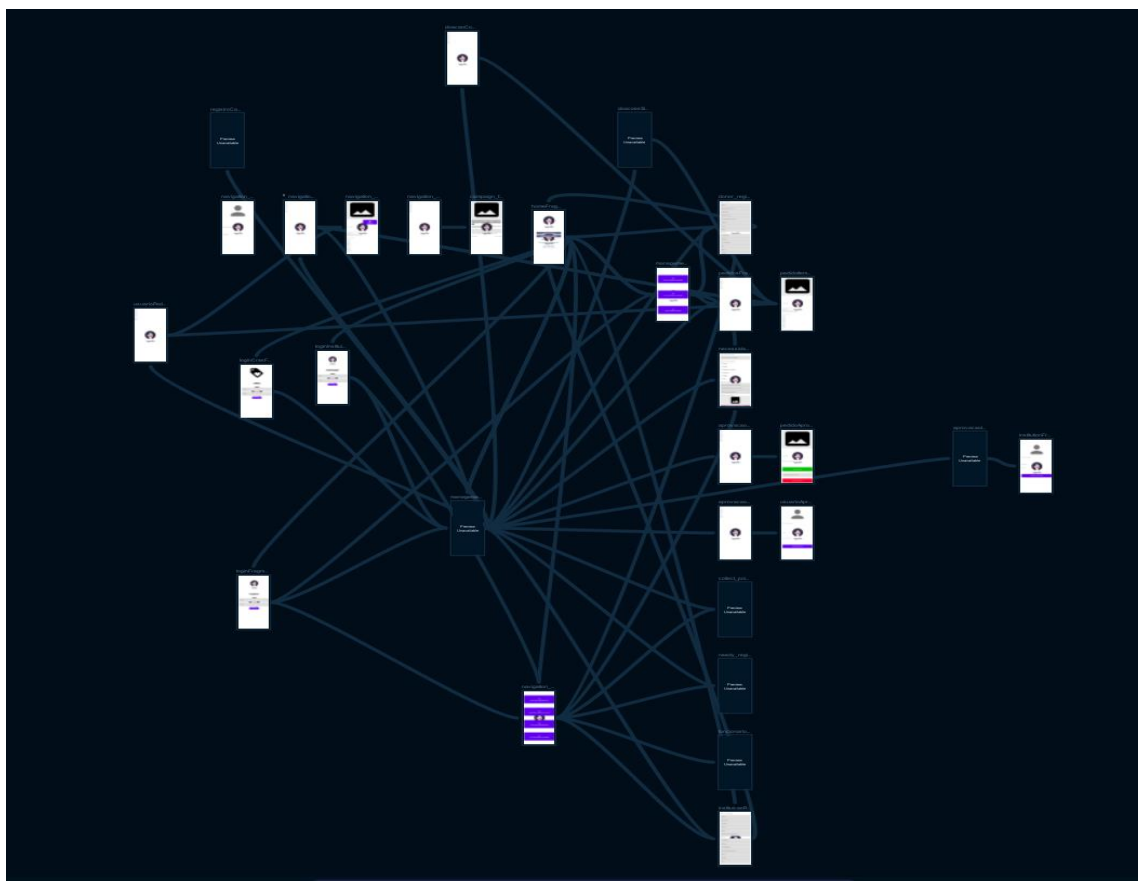
### 4.3.3 Navegação

A interface principal dá acesso ao usuário a listagens de pedidos, campanhas, informações do usuário e um menu baseado nas permissões que ele possui, podendo ser visitante, cras, usuário ou instituição.

Para controlar a navegação entre as páginas é utilizado um *component* do framework Android chamado de *Navigation Component*. A aplicação possui um mapa de navegação criado com as rotas, e a partir de ações do usuário algum fluxo é seguido. Na Figura 21 é mostrado o fluxo de telas da navegação do sistema, e na Figura 22 é exemplificado o código utilizado no *Navigation Component* do *app*.

A principal tela da aplicação é a tela de menu, de onde saem a maior quantidade de fluxos. Da tela de menu é possível ir para telas de cadastro de ponto de coleta, de necessidades, de necessitados e instituições, de campanhas e notícias. Também leva a listagens importantes como lista de aprovação de pedidos, de aprovação de usuários e instituições e de conclusão de pedidos de doação.

Figura 21 – Mapa do Navigation Component do sistema



Fonte: Do Autor

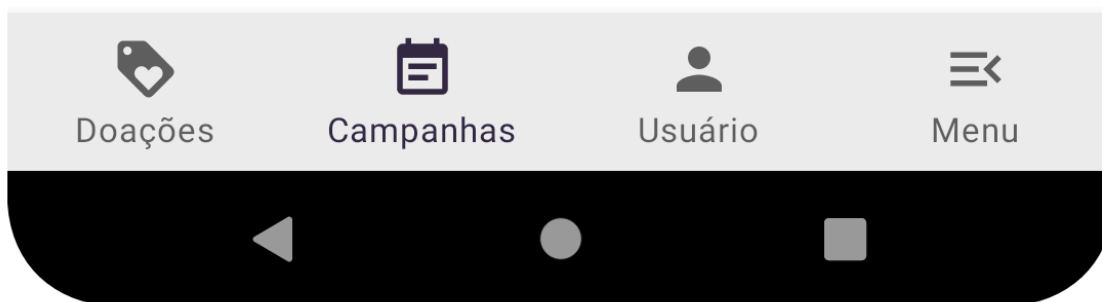
Figura 22 – Exemplo do código utilizado no *navigation component*

```
<fragment
    android:id="@+id/navigation_list_donation_request"
    android:name="com.example.socialapp_valmir.donation.DonationRequestListFragment"
    android:label="Doações"
    tools:layout="@layout/fragment_list_donation_request">
    <action
        android:id="@+id/action_navigation_list_publicacao_to_navigation_publicacao"
        app:destination="@id/navigation_donation_request" />
    <action
        android:id="@+id/action_navigation_list_donation_request_to_pedidoItemFragment"
        app:destination="@id/pedidoItemFragment" />
    <action
        android:id="@+id/action_navigation_list_donation_request_to_managementCrasFragment"
        app:destination="@id/managementCrasFragment" />
    <action
        android:id="@+id/action_navigation_list_donation_request_to_pedidoItemFragment2"
        app:destination="@id/pedidoItemFragment" />
</fragment>
```

Fonte: Do Autor

Para cada usuário é mostrado um menu baseado nas suas permissões. Caso for visitante é mostrado a tela de login.

Figura 23 – Barra de navegação



Fonte: Do Autor

A barra de navegação (Figura 23) é mostrada a todo momento na aplicação, exceto quando é selecionado algum item do menu.

#### 4.4 INTERFACES

Nesta seção serão apresentadas as telas do sistema, suas principais características e a pertinência destas com as funcionalidades previstas em seus



requisitos, mostrando também o funcionamento do aplicativo.

#### 4.4.1 Apresentação

A página inicial do sistema conta com uma listagem de pedidos de doação em aberto e concluídos, com uma aba para visualizar uma listagem de campanhas e notícias, a tela de usuário e a tela de login, as quais podem ser acessadas no menu de rodapé.

A listagem de pedidos de doação em aberto e concluídos pode ser vista na Figura 24, a listagem de campanhas e notícias pode ser vista na Figura 25 e a tela para realizar login ou cadastrar-se pode ser vista na Figura 27.

Figura 24 – Tela de início com a listagem de pedidos



Fonte: Do Autor

Quando não há usuário logado, na tela de informações do usuário é mostrada uma mensagem (Figura 26), e na tela de menu é mostrada uma tela onde pode-se realizar o login ou cadastrar-se como usuário ou instituição, como pode ser visto na Figura 27.

Figura 25 – Tela de listagem de campanhas e notícias



Fonte: Do Autor

Figura 26 – Tela de usuário não logado



Fonte: Do Autor

Figura 27 – Tela de login



Fonte: Do Autor

Figura 28 – Tela de usuário com autenticação realizada pelo CRAS



Fonte: Do Autor

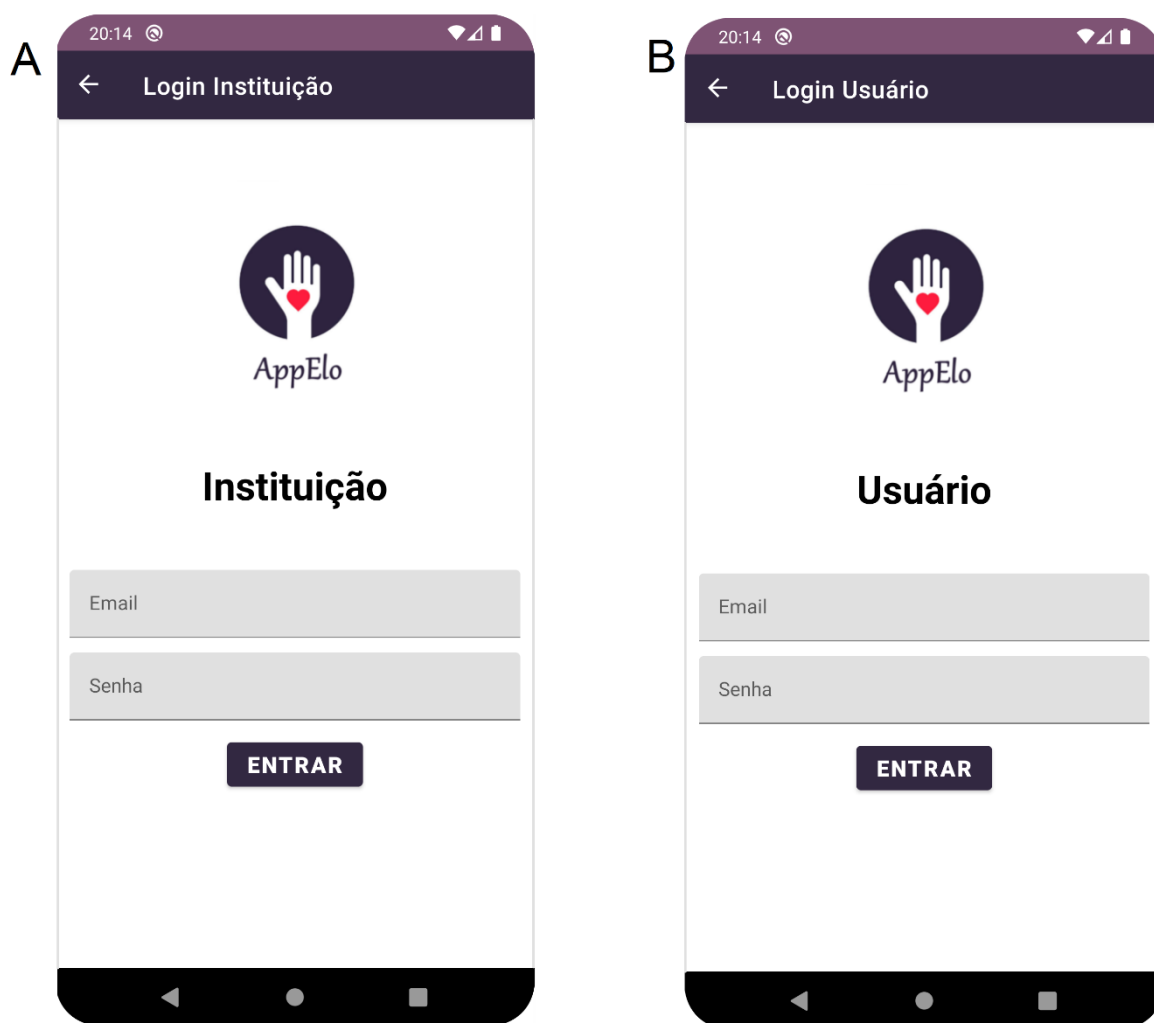
Se algum usuário já estiver com a autenticação feita, a tela de usuário mostra suas informações, como mostra a Figura 28, e a tela de menu fica de acordo com seu tipo de conta, como pode ser visto na Figura 38.

#### 4.4.2 Realizar login

Ao usuário clicar em “Login Usuário”, como pode ser visto na Figura 27, ele é levado para tela de realização de login, que pode ser vista na Figura 29. A tela de login de instituição possui basicamente o mesmo visual, mudando somente o título.

Figura 29 – Telas de login do sistema

A) Tela de Login de Instituição B) Tela de Login de Usuário



Fonte: Do Autor

#### 4.4.3 Cadastro de Usuário

Ao clicar em “cadastrar-se como usuário” o usuário é levado para tela de cadastro de usuário. A tela possui campos para preenchimento com as informações acerca do usuário e da família do mesmo, assim como um campo para seleção de uma imagem de usuário. A tela de cadastro é mostrada na Figura 29.

Figura 30 – Tela cadastro de usuários

A) Parte inicial da tela. B) Parte final da tela

The figure displays two screenshots of a mobile application's user registration screen, labeled A and B.

**Screenshot A: Parte inicial da tela**

The screen shows a header bar with a back arrow and the title "Cadastro de Doadores e Nece...". Below the header, there are two main sections:

- Dados Pessoais:** A vertical list of input fields for "Nome", "Data de Nascimento", "Estado Civil", "Gênero (M ou F)", "CPF (Somente números)", "Telefone", "Email", and "Senha".
- Endereço:** A vertical list of input fields for "Logradouro", "Número", and "Complemento".

**Screenshot B: Parte final da tela**

The screen shows the same header bar. Below the header, there are two main sections:

- Dados do Responsável Familiar:** A vertical list of input fields for "UF", "Nome do Responsável Familiar", "Telefone do Responsável Familiar", "CPF do Responsável Familiar (Somente núm...", "Quantidade de pessoas na família", and "Número do NIS (Somente números)".
- Image Selection:** A large square button with a mountain icon and the text "SELECIONAR IMAGEM".
- Save Button:** A large rectangular button with the text "SALVAR".

Fonte: Do Autor

A tela de cadastro de instituição é basicamente a mesma da tela de cadastro de usuário, mudando somente alguns campos. Na tela de cadastro de instituição possui o campo de cnpj, e não possui os campos de dados pessoais como gênero, estado civil, cpf, e também não possui dados do responsável familiar.

#### 4.4.4 Aprovação de instituição

Ao realizar o cadastro, tanto para o perfil do tipo usuário como do tipo instituição, é necessário esperar a aprovação do CRAS para ter sua conta ativada e poder realizar o login. As telas de listagem de usuário e listagem de instituições para aprovação são praticamente iguais, mudando somente os dados. A listagem de instituições pode ser vista na Figura 31.

Figura 31 – Tela de aprovação de instituições



Fonte: Do Autor

Ao selecionar uma instituição, ou usuário no caso da listagem de usuários, é exibida uma tela com as informações acerca do mesmo, e um botão para confirmar a aprovação, como pode ser visto na Figura 32.

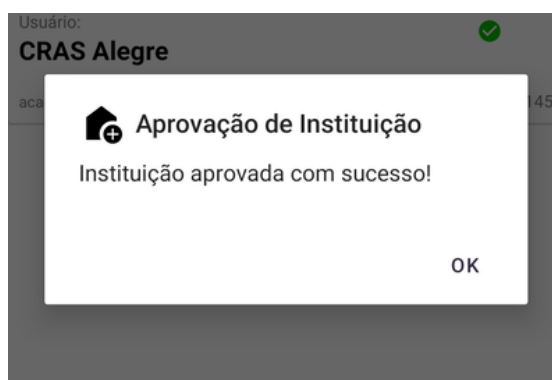
Figura 32 – Tela de informações da instituição



Fonte: Do Autor

Ao selecionar aprovar instituição na tela de informações da instituição (Figura 32) é exibida uma mensagem de sucesso na tela, que pode ser vista na Figura 33.

Figura 33 – Mensagem de sucesso de aprovação de instituição



Fonte: Do Autor

#### 4.4.5 Sinalizar um pedido de doação

Na tela de início está disponível a listagem de pedidos de doação em aberto e concluídos. Nos pedidos em aberto é possível sinalizar uma doação. Para realizar a sinalização é necessário realizar a autenticação no sistema.

Ao selecionar um pedido de doação na listagem é exibida a tela com as informações do pedido como a instituição onde foi cadastrada, dados de contato da instituição, necessidade, dados da família que necessita das doações e os pontos de coleta, como é mostrado na Figura 34.

Figura 34 – Tela de pedido de doação em aberto.



Fonte: Do Autor

Ao sinalizar o pedido de doação, se o usuário ainda não houver realizado o login é mostrada uma mensagem de aviso, como pode ser visto na Figura 35. Ao clicar em realizar login o usuário é redirecionado para tela de login, que pode ser vista na Figura 29.



Figura 35 – Mensagem de aviso de necessidade de login para realizar sinalização do pedido



Fonte: Do Autor

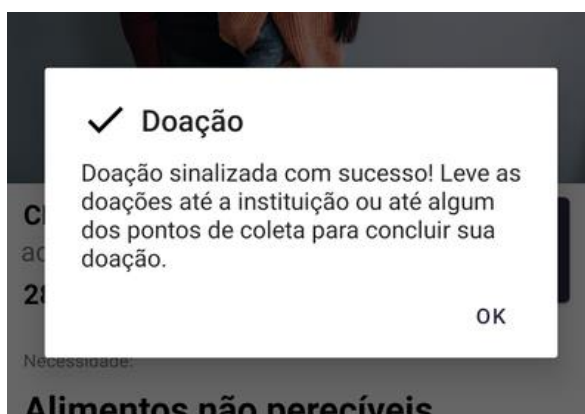
Se o usuário já estiver realizado a autenticação é exibida uma mensagem de confirmação como é mostrado na Figura 36, caso sim, é exibida uma mensagem de conclusão da sinalização, como mostrado na Figura 37.

Figura 36 – Mensagem de confirmação de sinalização do pedido



Fonte: Do Autor

Figura 37 – Mensagem de conclusão da sinalização do pedido



Fonte: Do Autor

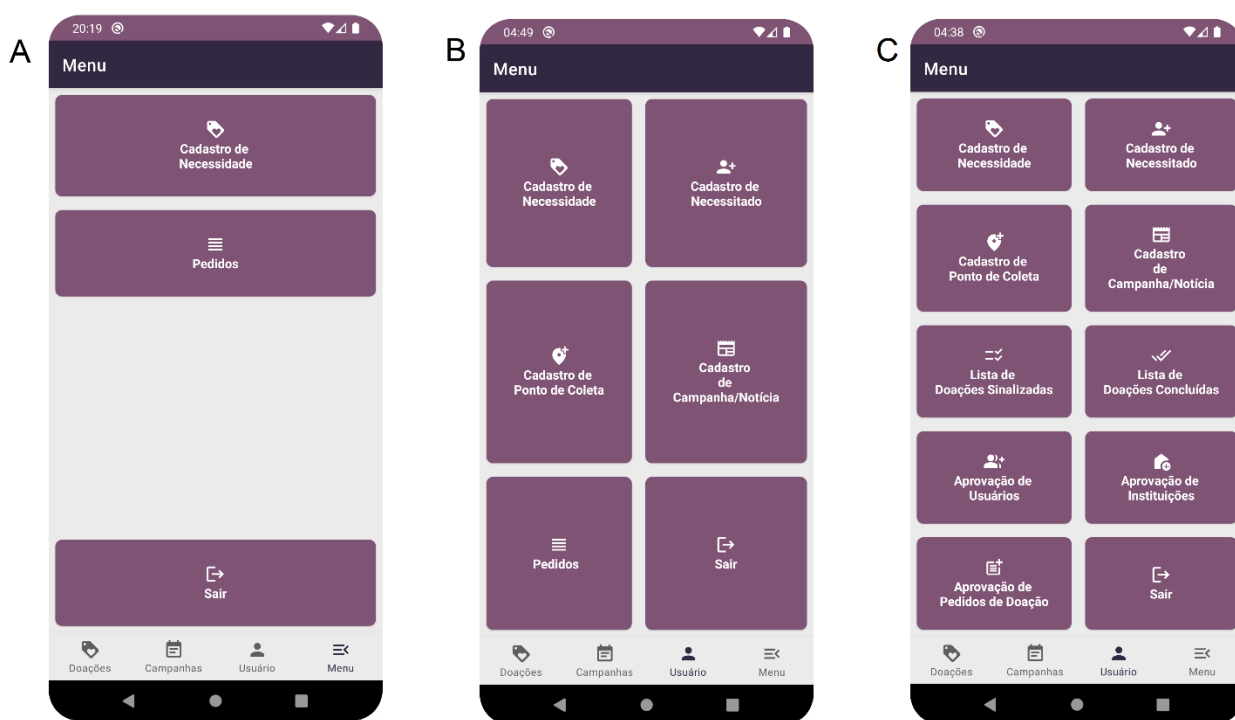
#### 4.4.6 Realizar pedido de doação

A funcionalidade de realizar um pedido de doação está disponível para qualquer usuário logado no sistema e permite a criação de um pedido de doação.

Ao estar logado no sistema é exibido um menu, diferente para cada tipo de usuário como mostrado nas Figuras 38, 39 e 40, e o usuário pode selecionar a opção cadastro de necessidade para realizar o pedido.

Figura 38 – Telas de menu

A) Tela de menu do usuário. B) Tela de menu da Instituição. C) Tela de menu do CRAS.



Fonte: Do Autor

Ao selecionar cadastro de necessidade é exibida uma tela com um formulário para preenchimento e podendo anexar uma imagem, como pode ser visto na Figura 39.

Figura 39 – Tela cadastro de necessidade.

A) Parte inicial da tela. B) Parte final da tela

**A**

04:38

← Cadastro de Necessidade

Dados da Necessidade

Descrição da Necessidade

Selecione uma categoria:

☐ Comida

☐ Roupas

☐ Cobertores e agasalhos

☐ Brinquedos

☐ Dinheiro

☐ Outro

Dados da Família

Nome do responsável da família

Quantidade de pessoas na família

Telefone do responsável

Selecione uma imagem:

**B**

04:38

← Cadastro de Necessidade

☐ Brinquedos

☐ Dinheiro

☐ Outro


Dados da Família

Nome do responsável da família

Quantidade de pessoas na família

Telefone do responsável

Selecione uma imagem:



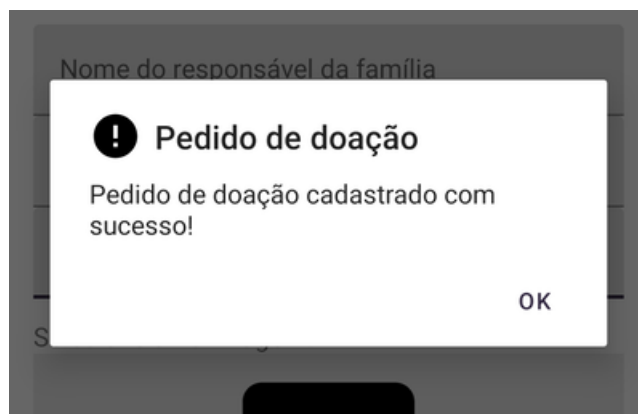
**SELECIONAR IMAGEM**

**SALVAR**

Fonte: Do Autor

Ao clicar em salvar é informado ao usuário por meio de uma mensagem na tela, que seu pedido foi realizado com sucesso, como pode ser visto na Figura 40.

Figura 40 – Mensagem de sucesso exibida na tela de pedido de doação

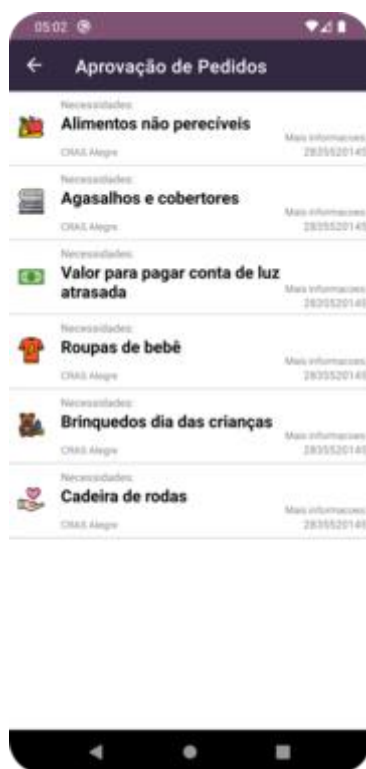


Fonte: Do Autor

#### 4.4.7 Aprovação pedidos de doação

Ao cadastrar um pedido o mesmo vai para aprovação do CRAS. Ao usuário clicar em “Aprovação de pedidos de doação” no menu, como pode ser visto na Figura 38, é exibida uma listagem com pedidos a serem aprovados, como mostrado na Figura 41. Somente o CRAS pode autorizar os pedidos de doação.

Figura 41 – Tela de aprovação de pedidos de doação

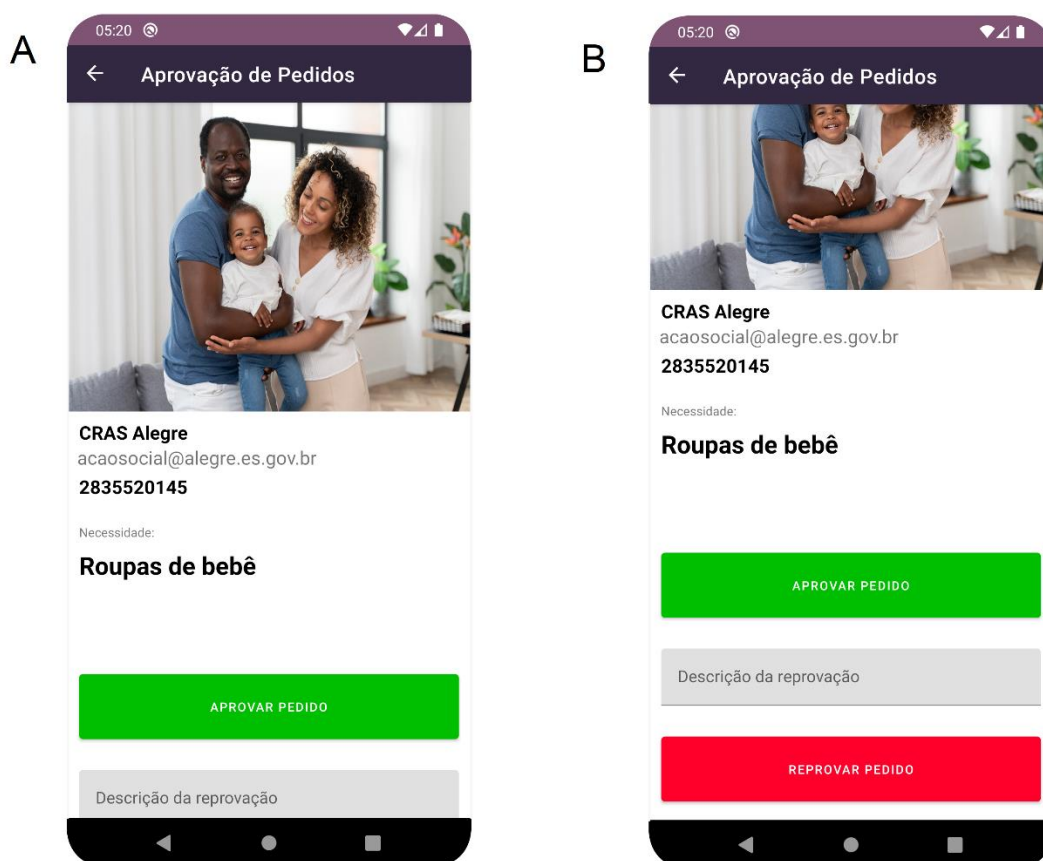


Fonte: Do Autor

Ao selecionar um pedido de doação na listagem, é exibida uma tela com as informações sobre o pedido, um botão para aprovar, um botão para reprovar e um campo de descrição para ser preenchido o motivo da reprovação, caso seja reprovado.

Figura 42 – Tela de aprovação de pedido de doação

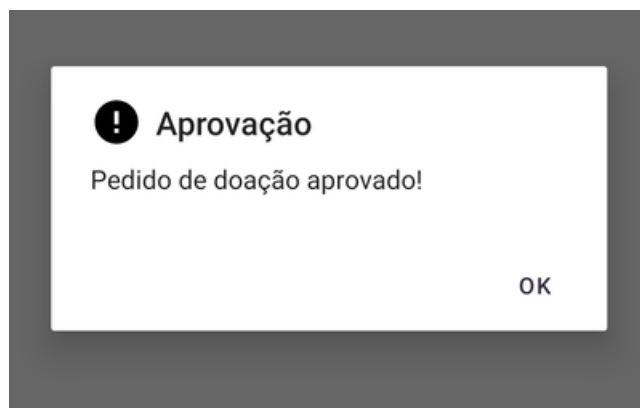
A) Parte inicial da tela. B) Segunda parte da tela



Fonte: Do Autor

Ao selecionar aprovar pedido é exibida uma mensagem de sucesso, como mostrado na Figura 43, e a listagem de pedidos da tela inicial (Figura 24) é atualizada com o pedido aprovado.

Figura 43 – Mensagem de sucesso na tela de aprovação de pedidos



Fonte: Do Autor

#### 4.4.8 Concluir pedido de doação

Ao clicar na opção de “Lista de doações sinalizadas” no menu (Figura 38) o usuário visualiza a listagem de todas as doações que foram sinalizadas, como mostrado na Figura 44.

Figura 44 – Tela de listagem de pedidos de doações sinalizadas



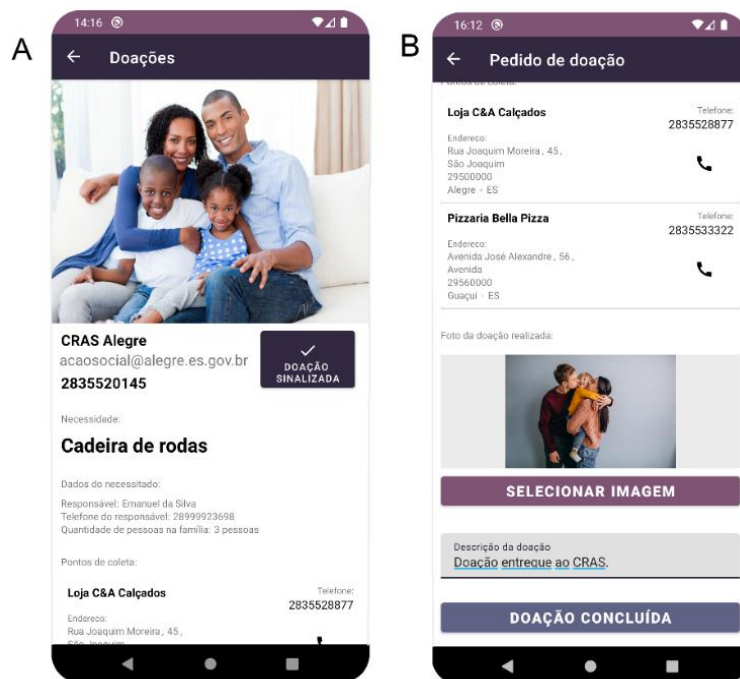
Fonte: Do Autor

Ao selecionar um pedido na listagem o usuário visualiza a informações acerca daquele pedido, e possui um campo de descrição de conclusão do pedido e pode

selecionar uma imagem para a conclusão. A tela de conclusão é mostrada na Figura 45.

Figura 45 – Tela de conclusão de pedido de doação

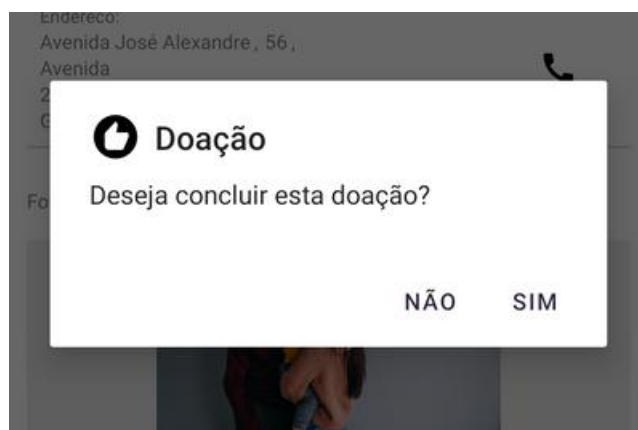
A) Parte inicial da tela. B) Parte final da tela



Fonte: Do Autor

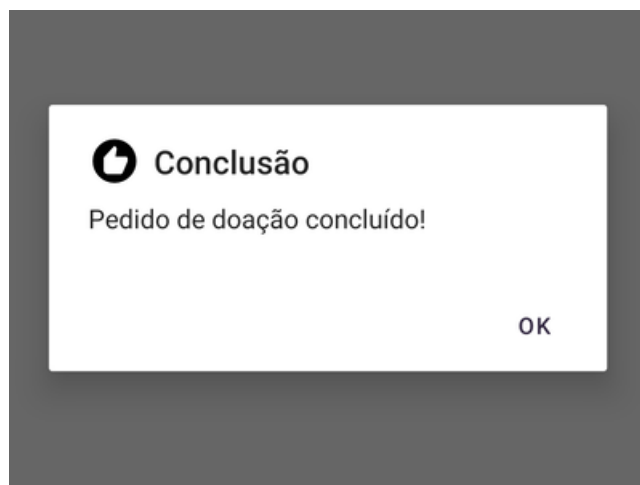
Ao usuário clicar em “doação concluída” é exibida uma mensagem de confirmação (Figura 46) e, ao clicar em sim, a doação é concluída e é exibida uma mensagem de sucesso na tela, como mostrado na Figura 47.

Figura 46 – Mensagens de conclusão de pedido de doação



Fonte: Do Autor

Figura 47 – Mensagem de conclusão do pedido de doação



Fonte: Do Autor

#### 4.4.9 Cadastro de pontos de coleta

O usuário do tipo CRAS ao clicar na opção “Cadastrar ponto de coleta” (Figura 38) é levado a tela de cadastro de ponto de coleta, que pode ser vista na Figura 48. Ao preencher os campos e clicar em salvar é exibida uma mensagem de sucesso, que pode ser vista na Figura 49.

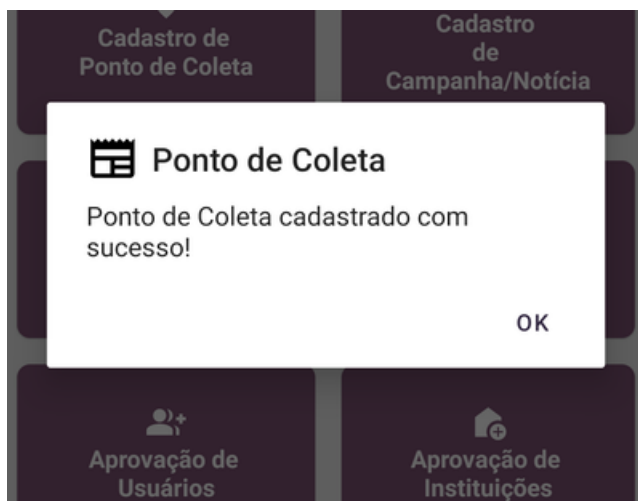
Figura 48 – Tela de cadastro de ponto de coleta

A imagem mostra a interface de uma aplicação móvel para o cadastro de pontos de coleta. No topo, há uma barra de status com o horário 04:39 e ícones de bateria e sinal. Abaixo, há uma barra de navegação com um ícone de seta para trás e o título "Registro de Ponto de Coleta". O formulário é dividido em duas seções: "Dados do Ponto de Coleta" e "Endereço". A primeira seção contém campos para "Nome" e "Telefone". A segunda seção contém campos para "Logradouro", "Número", "Complemento", "CEP", "Bairro", "Cidade" e "UF". No final do formulário, há um botão azul com o texto "SALVAR" em branco. Na base da tela, há uma barra de navegação com ícones de seta para trás, ponto central e seta para frente.

Fonte: Do Autor



Figura 49 – Mensagem de sucesso do cadastro de ponto de coleta



Fonte: Do Autor

#### 4.4.10 Cadastro de campanhas e notícias

Ao clicar na opção “Cadastro de Campanha/Notícia” (Figura 38) o usuário é levado a tela de cadastro de campanhas e notícias, que pode ser vista na Figura 50.

Figura 50 – Tela de cadastro de campanhas e notícias



Fonte: Do Autor

Ao clicar em salvar é exibida uma mensagem de sucesso na tela, como pode ser visto na Figura 51.

Figura 51 – Mensagem de sucesso exibida ao cadastrar campanha ou notícia



Fonte: Do Autor

#### 4.4.11 Aprovação de usuário

A listagem de usuários para aprovação está disponível ao usuário clicar na opção “Aprovação de Usuários” no menu do CRAS (Figura 38). Ao clicar em um usuário da listagem é exibida uma tela com as informações do usuário, como pode ser visto na Figura 51.

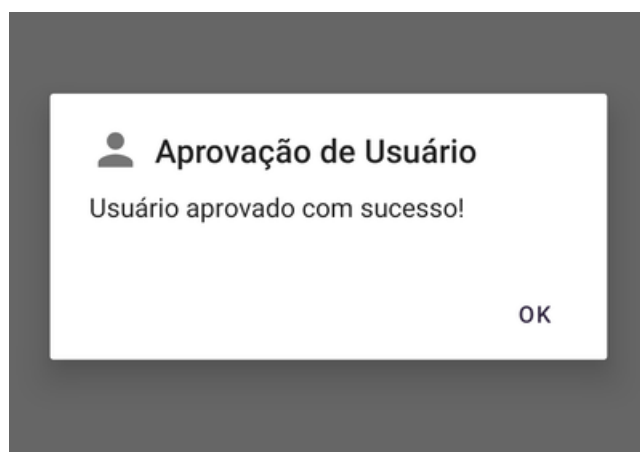
Figura 52 – Tela de aprovação de usuário



Fonte: Do Autor

Ao clicar em aprovar usuário é exibida uma mensagem de sucesso na tela, que pode ser vista na Figura 52.

Figura 53 – Mensagem de aprovação de usuário



Fonte: Do Autor

## 5 GARANTIA DE QUALIDADE E PROCESSOS DE TESTES

As inspeções são uma ideia antiga, porém, vários estudos e experimentos demonstraram que as inspeções são mais eficazes que os testes na descoberta de defeitos. As inspeções centram-se principalmente no código-fonte de um sistema (SOMMERVILLE, 2011).

Baseado nisso, foram realizadas inspeções por todo o código do sistema desenvolvido, com o objetivo de achar inconsistências e melhorar o código para tornar o código mais legível e para facilitar sua manutenção, encontrar erros e falhas, entre outras coisas.

Tabela 6 – Resultados das Inspeções

Defeito	Solução
Ícones de botões e menus não faziam alusão a ação desejada.	Foram alterados os ícones para que o usuário pudesse relacionar o ícone com a ação.
<i>Layout</i> de menu confuso.	Foram alterados o <i>layout</i> e título dos botões do menu e do menu por completo para ficar mais claro cada ação.
Menu inferior sem o título da aba.	Foi inserido título da aba no menu de navegação inferior.
Listagem de pedidos mostrava todos os pedidos, inclusive os sinalizados.	Foi alterada a <i>query</i> para listar somente pedidos em aberto e concluídos.

Fonte: Do Autor

Porém, segundo Sommerville (2011), as inspeções não podem substituir os testes de programa, pois não são boas o suficiente para descobrir defeitos e falhas que ocorrem na interação entre as diversas partes do programa.

Para garantir o funcionamento correto do sistema desenvolvido, foram realizados testes manuais. O teste manual consiste em testar o sistema manualmente por um ser humano, este compara os resultados com suas expectativas e observa-se as discrepâncias (SOMMERVILLE, 2011).

Tabela 7 – Resultados dos Testes de Desenvolvimento

Falhas	Teste Realizado	Motivo (erro)	Solução
Pedido não possuía imagem da família.	Ao selecionar um pedido de doação na listagem de pedidos em aberto.	Não estava sendo feito o armazenamento de imagem.	Foi implementado o armazenamento de imagem.
Ao visualizar o pedido sinalizado não era possível concluir o pedido.	Ao selecionar um pedido sinalizado na lista de pedidos sinalizados.	Não existia como realizar conclusão do pedido.	Foi implementado um botão e campo de descrição para conclusão.
Listagem de campanhas e notícias sem imagem.	Visualizar lista de campanhas e notícias.	Não era possível ver a imagem da campanha ou notícia na listagem.	Foi implementado a imagem na lista.
Ao clicar no ícone de ligação do ponto de coleta não era realizada nenhuma ação.	Na visualização do pedido de doação, onde há a listagem de pontos de coleta.	Ao clicar no ícone de ligação nada acontecia.	Foi adicionado a ação de ligação passando o telefone informado no ponto de coleta.
Ao visualizar listagem de pedidos não era possível realizar uma filtragem.	Listagem de pedidos em aberto e concluídos.	Não existia um menu para filtrar os pedidos.	Foi adicionado um menu <i>hamburger button</i> para filtragem de pedidos.
Ao visualizar a mensagem de necessidade de <i>login</i> para sinalizar	Sinalizar pedido de doação sem estar logado no sistema.	Não havia um botão de redirecionamento para o login.	Foi adicionado na mensagem de necessidade de login um botão

pedido, o usuário não era direcionado a tela de login.			para redirecionamento do usuário.
Telas não possuíam <i>scroll</i> e cortavam informações.	Visualização de qualquer tela com mais conteúdo.	Não havia uma forma de visualizar o resto das informações da tela.	Foi alterado a tela, adicionando um método de rolagem para visualizar totalmente as informações.
Menu de navegação inferior aparente em todas as telas.	Visualização do menu de navegação inferior na parte inferior da tela do sistema.	O menu não ocultava em telas onde não era possível utilizá-lo, ocasionando erros e não funcionando como esperado.	Foi ocultado o menu de navegação inferior em telas onde ele não funcionaria da forma correta e não era necessário.
Botão de voltar na <i>action bar</i> não realizava nenhuma ação.	Visualização de pedidos sinalizados, cadastro de necessidade, cadastro de pedido, e outras telas relacionadas ao menu.	Ao apertar o botão voltar na <i>action bar</i> não realizava nenhuma ação.	Foi atribuído a ação de voltar no botão voltar da <i>action bar</i> .
Mensagens de sucesso ao	Nas telas de aprovar usuário, aprovar pedido,	Ao concluir uma ação nas telas que possuem botões	Foi adicionada uma mensagem de sucesso exibida

realizar alguma ação.	aprovar instituição, cadastro de usuário, cadastro de instituição, entre outras.	para confirmação, não era exibida mensagem de sucesso ao concluir.	na tela ao concluir uma ação com sucesso.
Mensagem de erro ao realizar alguma ação.	Nas telas de aprovar usuário, aprovar pedido, aprovar instituição, cadastro de usuário, cadastro de instituição, entre outras.	Ao concluir uma ação nas telas que possuem botões para confirmação, quando ocorria algum erro não era exibida mensagem de erro.	Foi adicionada uma mensagem de erro exibida na tela ao concluir uma ação com erro.
Não era possível realizar o <i>logout</i> .	Telas de menu.	Não havia uma forma de realizar o <i>logout</i> do sistema.	Foi adicionado um botão "Sair" para o usuário deslogar quando desejado.
Botão de sinalizar doação não mudava ao realizar a sinalização do pedido.	Clicar no botão de sinalização de pedido na tela de visualização de pedido de doação em aberto.	Ao mudar o status para doação sinalizada o botão de sinalizar doação não mudava nada, deixando em dúvida se o status tinha mudado.	Foi adicionado um novo texto e ícone, e o desativado a possibilidade de clique no botão ao sinalizar a doação.
Visualização do pedido não carregava a imagem.	Visualizar qualquer pedido de doação de qualquer listagem.	Ao visualizar o pedido de doação a imagem da família do pedido não era carregada.	Foi corrigido o método de carregamento da imagem da família.

Erro ao validar campo vazio nos formulários.	Qualquer formulário de cadastro no sistema.	Ao clicar em salvar era mostrado erro em todos os campos, não só nos necessários.	Foi corrigido para mostrar erro somente nos campos com erro.
--	---	---	--

Fonte: Do Autor

No sistema desenvolvido foram realizados testes de desenvolvimento. Os testes de desenvolvimento incluem todas as atividades de testes que são realizadas pela equipe de desenvolvimento de um sistema, onde o testador do software geralmente é o programador que desenvolveu, o objetivo desse teste é descobrir *bugs* no sistema e geralmente é intercalado com o processo de depuração de código e sua alteração para correção do problema (SOMMERVILLE, 2011).

Durante o desenvolvimento o teste pode ocorrer em três níveis de granularidade: teste unitário, teste de componentes e teste de sistema. No sistema foi escolhido realizar o teste de sistema, já que consiste em testar o sistema como um todo, da forma que ele será usado pelo usuário final, onde todos os componentes do sistema estão integrados.

## 5.1 Teste de Usabilidade

O teste de usabilidade garante que o sistema esteja funcionando corretamente, testando as funcionalidades, desempenho e verificando vários elementos. Além disso o teste de usabilidade ajuda a descobrir a facilidade de uso de um sistema. Portanto, ajuda a melhorar a experiência do usuário e também ser um teste que simula o uso real do sistema.

O teste de usabilidade foi executado com o objetivo de verificar a experiência do usuário. Dessa forma, neste teste foi checada a organização dos itens disponíveis na interface, observando se o layout é agradável e correto para uso e se os botões se comunicam corretamente entre as diferentes telas do sistema.



Tabela 8 – Resultados do teste de usabilidade

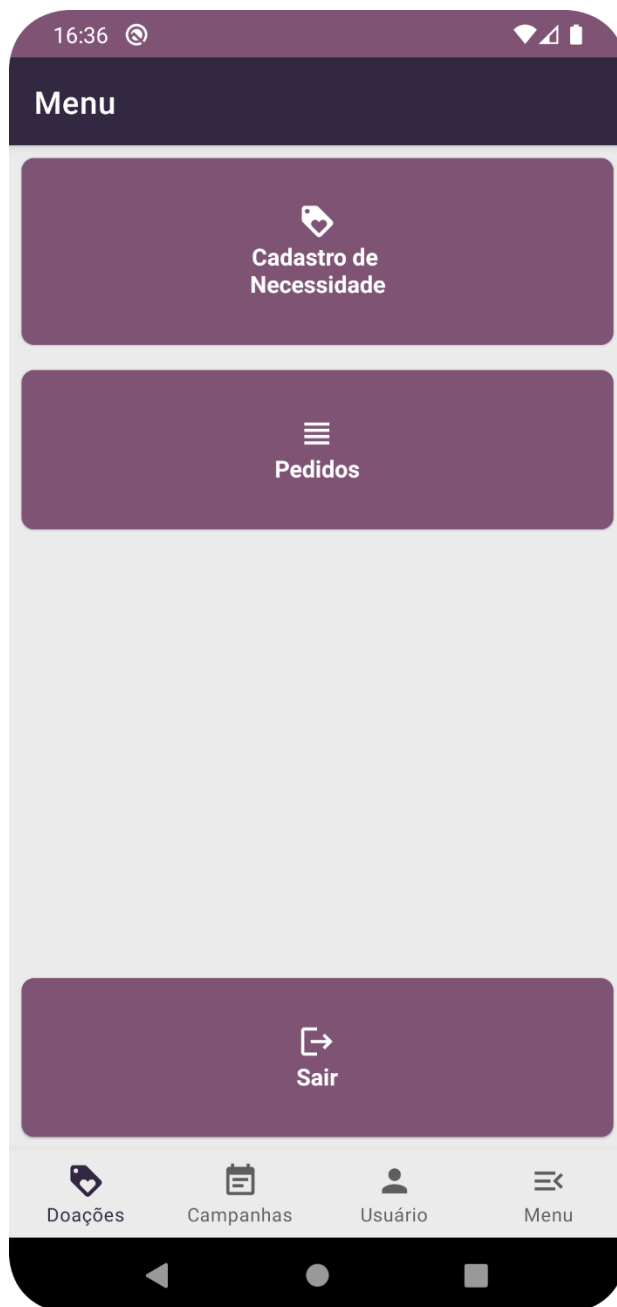
<b>Falhas</b>	<b>Teste realizado</b>	<b>Motivo (erro)</b>	<b>Solução</b>
Formulário de <i>login</i> não alertava sobre e-mail incorreto.	Foi realizado o login no sistema informando e-mail errado.	Não era exibida uma mensagem de erro.	Após verificar que o e-mail era inválido, foi informado um aviso de erro no formulário alertando que o e-mail era inválido.
Cadastros sem validação de campos.	Todos os formulários de cadastro.	Não era exibido o campo que estava com erro.	Foi alterado para que o campo de texto mostrasse se era inválido ou obrigatório caso não fosse preenchido.
O campo telefone nos formulários de cadastro permitiam inserir texto.	Formulários de cadastro que possuem o campo telefone.	O tipo de texto que pode ser inserido.	Foi alterado o tipo de entrada de texto permitido no campo.
Quando logado ao selecionar a aba menu do menu de navegação inferior, o ícone e título do menu não ficam em evidência.	Seleção de aba no menu de navegação inferior.	Desconhecido.	Não foi possível, por meio dos conhecimentos adquiridos e buscados com a ferramenta atual, solucionar este erro.

Fonte: Do Autor

Na Figura 53 pode-se ver o erro que não foi possível de se solucionar. Quando um usuário de qualquer tipo realiza o login, ao selecionar outra aba no menu de

navegação inferior e depois selecionar novamente a aba de menu, o ícone e título do menu não ficam evidenciados como selecionado.

Figura 54 – Tela de menu com menu de navegação com erro de seleção



Fonte: Do Autor

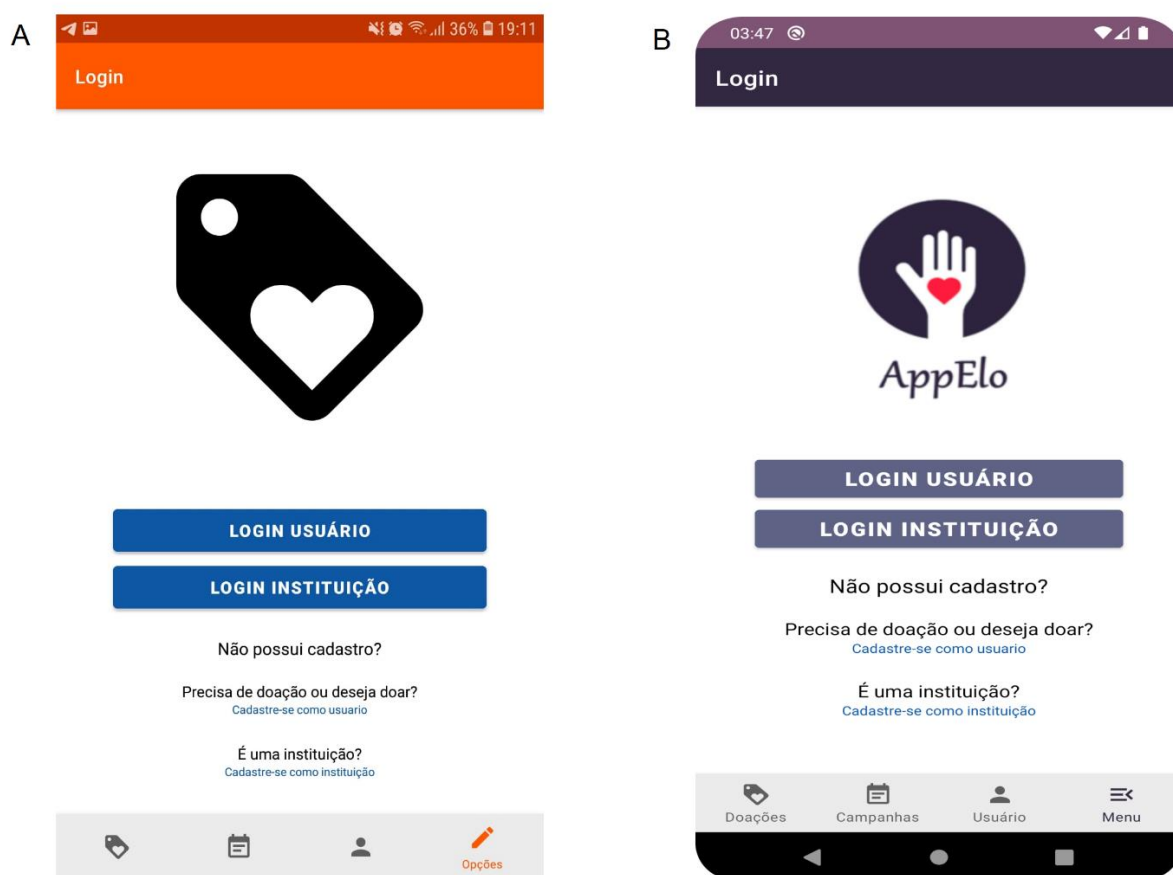
Com o proposto até o momento de desenvolvimento do aplicativo as ocorrências dos testes foram sanadas, com os requisitos iniciais atendidos e a aplicação sendo simples de se usar. Porém ainda se faz necessário testes com os usuários finais para avaliar se irão surgir novos requisitos a serem implementados.

## 5.2 Refinamento Visual

Houve um refinamento visual no aplicativo desenvolvido. Ao chegar em algumas etapas constatou-se que era necessário realizar um redesenho do *layout* do sistema. Isso foi necessário devido alguns ícones e ações do sistema não ficarem claros e concisos, tornando confuso a navegação e usabilidade do mesmo.

Figura 55 – Telas de gerenciamento de contas do aplicativo.

(A) Tela antes do refinamento visual. (B) Tela após o refinamento visual.

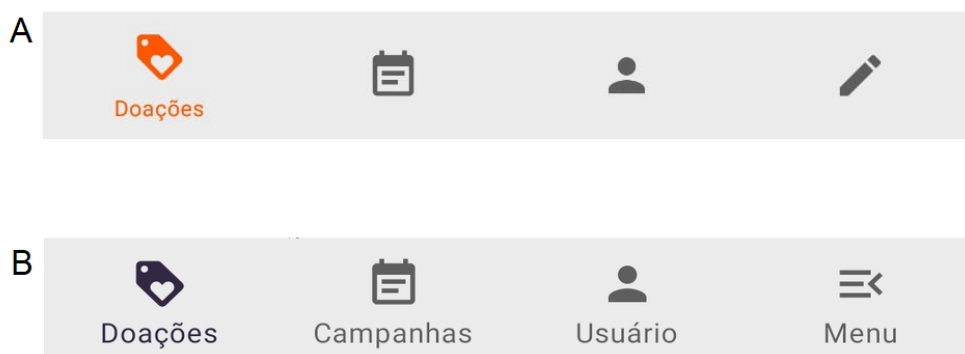


Fonte: Do Autor

Na Figura 55 foi realizada o refinamento da tela de contas, onde é possível realizar o login e a criação de novas contas.

Figura 56 – Menu de navegação inferior do sistema.

(A) Menu antes do refinamento visual. (B) Menu após o refinamento visual

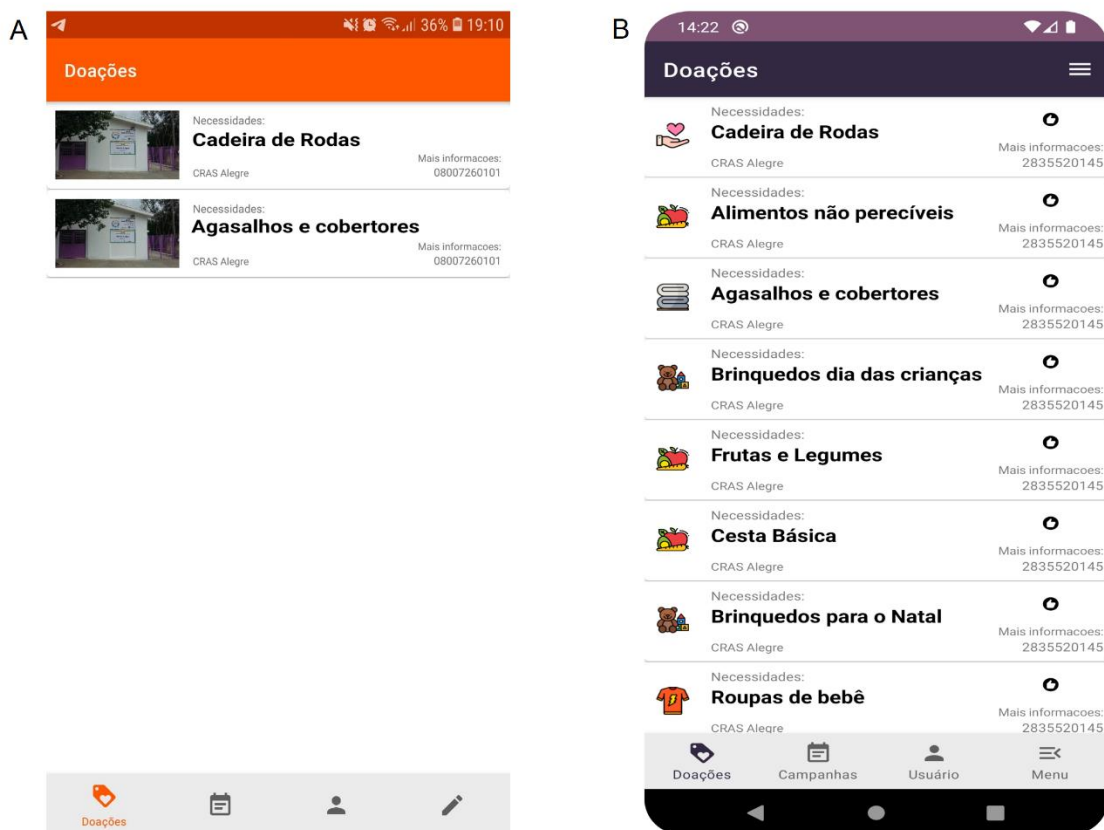


Fonte: Do Autor

No menu de navegação inferior (Figura 56) foram introduzidos os nomes das páginas tornando mais claro a navegação pelo aplicativo.

Figura 57 – Listagem de pedidos de doação em aberto.

(A) Listagem antes do refinamento visual. (B) Listagem após o refinamento visual.



Fonte: Do autor

Na tela de listagem de pedidos de doação em aberto e concluídos (Figura 57) foram introduzidos ícones para cada pedido, possibilitando visualizar a categoria de cada pedido realizado. Também foi implementado um ícone ao invés de imagens, que sinalizam o *status* do pedido, na Figura 57 (B) é possível ver que os *status* dos pedidos estão em aberto.

Nas telas de campanhas e notícias, como pode ser visto na Figura 58, foi feito um recondicionamento nas imagens, colocando um tamanho mais agradável, e também foi introduzido o nome do autor da notícia ou campanha listada (Figura 58 (B)).

Figura 58 – Tela de listagem de campanhas e notícias.

(A) Tela de listagem antes do refinamento visual.

(B) Tela de listagem após o refinamento visual.



Fonte: Do Autor

Todos os menus foram refeitos, adicionando ícones que eram concisos com suas funcionalidades. Além de mais bonito, ficou mais prático saber qual botão realiza cada ação. Além da disposição, tamanho dos botões e cores atribuídas.

Na figura 59 é possível visualizar a tela de menu do CRAS, onde passou de uma listagem de botões que não eram claros e não possuíam ícones com sua alusão correta para uma tela com botões claros e elegantes que além de bonitos tornam a navegação mais clara e rápida.

Figura 59 – Tela de menu do CRAS.

(A) Tela de menu antes do refinamento visual.

(B) Tela de menu após refinamento visual.



Fonte: Do Autor

## 6 CONSIDERAÇÕES FINAIS E CONCLUSÕES

Ao final deste trabalho, foi concluído o desenvolvimento de uma aplicação *mobile* feito para sistemas operacionais Android para cadastro e aprovação de pedidos de doações gerenciados pelo CRAS, com as necessidades e requisitos necessários para atender à necessidade desse gerenciamento, tendo como fonte de requisitos e alvo do trabalho os pedidos de doações gerenciados pelo CRAS.

O sistema desenvolvido permite o CRAS analisar os pedidos de doação e aprova-los ou reprova-los, além de permitir a visualização de campanhas e notícias sobre doações, cadastro de usuários para realizar pedidos de doação, e doadores conseguem se conectar e visualizar o que realmente o que é necessitado.

O sistema torna rápido e prático o acesso as informações acerca do que se necessita, sabendo qual o local de doação, informações sobre a família, e facilitando assim o acesso tanto para se doar algo, quanto para receber uma doação.

A aplicação produzida é uma proposta viável, apresentando um conteúdo mais dinâmico e acessível do que se é possível hoje. Os pedidos em aberto são listados e podem ser vistos facilmente por algum doador em potencial. Além disso, facilita alguém que necessita de algo se cadastrar e realizar um pedido sem precisar se deslocar até o CRAS para fazê-lo, o que traz uma facilidade para ligar pessoas que podem doar a pessoas que precisam dessas doações.

Espera-se que a aplicação desenvolvida seja uma alternativa computacional para o CRAS, já que é uma proposta mais viável para gestão de pedidos de doação que o método atual que é realizado.

A aplicação desenvolvida foi disponibilizada apenas para ambiente de desenvolvimento, sendo assim, não foram realizados testes automatizados, testes unitários, testes de integração, o que tornou inviável o desenvolvimento desses testes, já que teriam de ser criados vários cenários para cada operação disponível no *backend*.

A execução e desenvolvimento do projeto como um todo foi de grande importância, já que ocorreu a possibilidade de participar de todas as etapas de desenvolvimento de um software, além da possibilidade de escolha das ferramentas, técnicas e tecnologias utilizadas.

No desenvolvimento deste sistema, foram utilizadas tecnologias que são amplamente utilizadas no mercado de trabalho atual, a fim de realizar uma pesquisa

mais branda, agregando, além dos ensinamentos lecionados na grade curricular do curso de Ciência da Computação, utilizar-se de tecnologias que melhor se adequam a cada tipo de situação atendendo a necessidades do sistema proposto. Ainda assim, se aplicando e utilizando conceitos ensinados no curso para o desenvolvimento de todo o sistema. Foi utilizado o desenvolvimento nativo para sistema operacional Android feito em Kotlin, entre outras tecnologias e bibliotecas que são amplamente utilizadas no mercado.

Assim, além do desenvolvimento do sistema também houve a busca pelo conhecimento e o aprendizado por parte do autor, que buscou melhor entendimento e desafios com as tecnologias do mercado. Também foi possível para o autor praticar e aperfeiçoar-se no uso de tecnologias as quais já possuía conhecimento. Isso proporcionou um desenvolvimento maior e que também gerou dificuldades, porém, após muito tempo e pesquisa, as dificuldades foram transformadas em conhecimento.

## **6.1 Trabalhos futuros**

Baseado no fato de que no desenvolvimento da aplicação foram utilizadas bibliotecas que facilitam o desenvolvimento e que a forma na qual o sistema foi estruturado, a implementação de novas funcionalidades e possíveis alterações e correções, podem ser feitas de maneira simples.

Esse tipo de aplicação possui um uso muito elevado quando adotado para uso, sendo assim, é natural a necessidade de evolução para atender novas necessidades e demandas do dia a dia do CRAS. Assim, é possível complementar ainda mais o sistema desenvolvido, podendo ser adicionadas novas funcionalidades.

Ainda é necessário realizar a testagem do sistema com os públicos alvos (doadores, necessitados, CRAS e instituições). Logo, fica como sugestão a realização desta testagem, para efetuar o levantamento dos ajustes a serem feitos e de melhorias identificadas, tendo como consequência um sistema com maior nível de qualidade.

A partir do uso do sistema, torna-se necessário o levantamento de novos requisitos, já que por se tratar de um ramo grande, existem muitos requisitos possíveis, principalmente considerando as especificidades de cada pedido de doação, necessitados, doadores e instituições.

Como sugestões também, sugere-se a implementação de funcionalidades importantes como por exemplo, edição de usuários e instituições, edição de pedidos



de doação, edição de campanhas e notícias e correção do erro da seleção da aba no menu.

Sugere-se também, implementar um controle de escalabilidade, teste de integração de módulos na hospedagem, um método de segurança maior para a autenticação, além de recuperação de senha por e-mail e possibilidade de login através de redes sociais ou gerenciadores de e-mail (Google, Facebook, Instagram, etc).

O aplicativo não possui um sistema de notificações, no qual poderá indicar instantaneamente ao usuário se ele foi ou não aprovado no aplicativo, ou mesmo seu pedido, por exemplo, a necessidade que o usuário realizou for ser atendido, o sistema o notifique. Logo, estudos adicionais para esta implementação deverão ser realizados, visto que nessa primeira versão, não foi possível por limitações técnicas e também de tempo.

As doações em dinheiro ou financeiras poderão ser realizadas, por exemplo, através de depósitos feitos por PIX, podendo ser enviado o comprovante para o necessitado para confirmação.

Outra implementação adicional muito útil seria implementar o uso de uma API para busca de endereços utilizando-se do CEP inserido, facilitando o preenchimento dos dados do endereço.

Como o aplicativo móvel proposto foi feito somente com a visão do necessitado, é possível implementar a parte do doador, onde um usuário que possua artigos para doação possa anunciá-los no *app*, possibilitando através de um número de telefone ou local previamente definido no anúncio de doação, estipular um local para retirada da doação.

## 7 REFERÊNCIAS BIBLIOGRÁFICAS

AMPIRE, Eric. **Injeção de dependência com Koin**. 25 ago. 2019. Disponível em: <<https://medium.com/swlh/dependency-injection-with-koin-6b6364dc8dba>>. Acesso em: 2 nov. 2021.

ANDRADE, Priscilla Maia de; ALMEIDA, Aidê Cançado; LIMA, Helena Ferreira de. **Orientações Técnicas - Centro de Referência de Assistência Social - CRAS: Ministério do Desenvolvimento Social e Combate à Fome**. 1. ed. Brasília: 2009. 72 p. ISBN 978-85-60700-29-5.

ARAÚJO, Ivo de Abreu. Análise da ascensão do sistema operacional Android no mercado a partir de suas evoluções. **Revista de Tecnologia da Informação e Comunicação da Faculdade Estácio do Pará**, Belém, ano 5, v. 3, p. 54-67, jul. 2020. Disponível em: <<http://www.revistasfap.com/ojs3/index.php/tic/article/view/328/286>>. Acesso em: 19 set. 2021.

BAHLE, Thomas; PFEIFER, Michaela; WENDT, Claus. **Social Assistance: The Oxford Handbook of the Welfare State**. 2010. DOI 10.1093/oxfordhb/9780199579396.003.0031. Disponível em: <<https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199579396.001.0001/oxfordhb-9780199579396-e-31>>. Acesso em: 18 set. 2021.

BARRIENTOS, Armando. The Rise of Social Assistance in Brazil. p. 888-910, jul. 2013. DOI 10.1111/dech.12043. Disponível em: <[https://www.researchgate.net/publication/264407952\\_The\\_Rise\\_of\\_Social\\_Assistance\\_in\\_Brazil](https://www.researchgate.net/publication/264407952_The_Rise_of_Social_Assistance_in_Brazil)>. Acesso em: 17 set. 2021.

BELEKAR, Shubham *et al.* MOBILE APPLICATION FOR DONATION OF ITEMS. VIVA-TECH INTERNATIONAL JOURNAL FOR RESEARCH AND INNOVATION, v. 1, n. 4, 2021.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML: Guia do Usuário**. 2. ed. rev. e atual. Elsevier, 2006. ISBN 13 978-85-352-1784-1.

CHOI, Boreum; KIM, Mingyung. Donation via Mobile Applications: A Study of the Factors Affecting Mobile Donation Application Use. International Journal of Human-Computer Interaction, p. 1-1, 5 ago. 2016. DOI 10.1080/10447318.2016.1220070. Disponível em: <<https://www.tandfonline.com/doi/full/10.1080/10447318.2016.1220070>>. Acesso em: 19 set. 2021.

DOAR FÁCIL. Doar Fácil. 2021. Disponível em: <<https://www.doarfacil.com.br/>>. Acesso em: 17 out. 2021.

EXPOSED. JetBrains. 2022. Disponível em: <<https://github.com/JetBrains/Exposed>>. Acesso em: 19 jul. 2022.

GODINHO, Sibele Grasiela Guedes *et al.* A doação na perspectiva de aplicativos sociais. Investigação Qualitativa em Engenharia e Tecnologia, v. 4, p. 7-16, 2017.

HART, Eli. **Epoxy for Android**. 2021. Disponível em: <<https://airbnb.io/projects/epoxy/>>. Acesso em: 27 out. 2021.

HEUSER, C. A. **Projeto de banco de dados**. Porto Alegre: Bookman, 2009.

HRON, Michal; OBWEGESER, Nikolaus. Scrum in practice: an overview of Scrum adaptations. **Proceedings of the 51st Hawaii International Conference on System Sciences – 2018**. 2018. ISBN 978-0-9981331-1-9. Disponível em: <<https://scholarspace.manoa.hawaii.edu/bitstream/10125/50568/paper0681.pdf>>. Acesso em: 7 mar. 2022.

IZUMA, Keise; SAITO, Daisuke N.; SADATO, Norihiro. Processing of the Incentive for Social Approval in the Ventral Striatum during Charitable Donation. *Journal of Cognitive Neuroscience*, v. 22, n. 4, p. 621-631, 2010.

JACCOUD, Luciana; BICHIR, Renata; MESQUITA, Ana Cleusa. O SUAS NA PROTEÇÃO SOCIAL BRASILEIRA: Transformações recentes e perspectivas. **Novos Estudos**, São Paulo, v. 36, n. 2, p. 37-53, 1 jul. 2017. DOI <http://dx.doi.org/10.25091/S0101-3300201700020003>. Disponível em: <https://www.scielo.br/j/nec/a/Vkv7r47xGw7Hd6XmZdh7HfL/?format=pdf&lang=pt>. Acesso em: 19 set. 2021.

JETBRAINS. **Kotlin Coroutines**. 2022. Disponível em: <https://kotlinlang.org/docs/coroutines-guide.html#table-of-contents>. Acesso em: 19 jul. 2022.

JOYZ. **Joyz Doação – A rede do bem**. 2021. Disponível em: <https://joyz.com.br/>. Acesso em: 17 out. 2021.

JSON. **Introducing JSON**. 2020. Disponível em: <https://www.json.org/>. Acesso em: 22 nov. 2020.

KAYERE, Peter. **Simple GET request using Retrofit in Android**. 5 jan. 2021. Disponível em: <https://www.section.io/engineering-education/making-api-requests-using-retrofit-android/#:~:text=Retrofit%20is%20a%20type%2Dsafe,Retrofit%20to%20make%20API%20requests>. Acesso em: 14 jul. 2022.

KAZAP. **Arquitetura Mobile**: Como escolher a arquitetura mobile da melhor forma. 2020. Disponível em: <https://blog.kazap.com.br/arquitetura-mobile/#:~:text=A%20arquitetura%20mobile%20nada%20mais,de%20disponibilidade%20e%20alta%20performance>. Acesso em: 08 mar. 2022.

KTOR. JetBrains. 2021. Disponível em: <https://ktor.io/>. Acesso em: 28 out. 2021.

MAURIEL, Ana Paula Ornellas. Pobreza, seguridade e assistência social: desafios da política social brasileira. *Revista Katál*, Florianópolis, v. 13, n. 2, p. 173-180, jul./dez 2010.

MENDONÇA, Vinícius Rafael Lobo de; BITTAR, Thiago Jabur; DIAS, Márcio de Souza. Um estudo dos Sistemas Operacionais Android e iOS para o desenvolvimento de aplicativos. **ENACOMP2011**, 2011. Disponível em: <[https://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011\\_submission\\_54.pdf](https://www.enacomp.com.br/2011/anais/trabalhos-aprovados/pdf/enacomp2011_submission_54.pdf)>. Acesso em: 19 set. 2021.

MENEZES, Aline *et al*, (coord.). **CRAS, um lugar de (re)fazer histórias**. 2. ed. rev. Brasília: 2007. ISBN 1982-4734. Disponível em: <[https://www.mds.gov.br/webarquivos/publicacao/assistencia\\_social/Revista/Cras\\_Umlugar\\_fazer\\_historias.pdf](https://www.mds.gov.br/webarquivos/publicacao/assistencia_social/Revista/Cras_Umlugar_fazer_historias.pdf)>. Acesso em: 19 set. 2021.

MICROSOFT. **Vinculação de dados e MVVM**. 2022. Disponível em: <[https://docs.microsoft.com/pt-br/windows/uwp/data-binding/data-binding-and-mvvm#:~:text=O%20MVVM%20\(Model%2DView%2D,%C3%A9%20da%20interface%20do%20usu%C3%A1rio.>](https://docs.microsoft.com/pt-br/windows/uwp/data-binding/data-binding-and-mvvm#:~:text=O%20MVVM%20(Model%2DView%2D,%C3%A9%20da%20interface%20do%20usu%C3%A1rio.>)>. Acesso em: 08 mar. 2022.

MOSKALA, Marcin; WOJDA, Igor. **Android Development with Kotlin**. Birmingham - Mumbai: Packt, 2017. ISBN 978-1-78712-368-7.

MUXFELDT, Pedro. **Sistemas operacionais para celulares e dispositivos móveis**. 10 dez. 2020. Disponível em: <<https://br.ccm.net/faq/11106-sistemas-operacionais-para-celulares-e-dispositivos-moveis>>. Acesso em: 19 set. 2021.

MYSQL. MySQL Workbench. 2022. Disponível em: <<https://www.mysql.com/products/workbench/>>. Acesso em: 22 fev. 2022.

OLIVEIRA, Victor; TEIXEIRA, Leopoldo; EBERT, Felipe. On the Adoption of Kotlin on Android Development: A Triangulation Study. **2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)**, p. 206-216, 2020. DOI 10.1109/SANER48275.2020.9054859. Disponível em:

<<https://ieeexplore.ieee.org/abstract/document/9054859/citations#citations>>. Acesso em: 27 out. 2021.

PIRES, Alifyz F. **Consumindo API REST com Retrofit + Kotlin no Android**. 16 dez. 2018. Disponível em: <<https://medium.com/@alifyzpires/consumindo-api-rest-com-retrofit-kotlin-no-android-abba52820cc>>. Acesso em: 6 mar. 2022.

PINTEREST. 2022. Disponível em <<https://br.pinterest.com/>>. Acesso em: 19 out. 2022.

POSTGRESQL. **About**. 2020. Disponível em: <<https://www.postgresql.org/about/>>. Acesso em: 02 nov. 2021.

REDHAT. **O que é uma API?**. 2017. Disponível em: <<https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces>>. Acesso em: 04 mar. 2022.

RIBON. Ribon - doação para a caridade sem gastar dinheiro. 2021. Disponível em: <<https://ribon.io/>>. Acesso em: 16 out. 2021.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson, 2011.

SPÄTH, Peter. **Learn Kotlin for Android Development: The Next Generation Language for Modern Android Apps Programming**. 1. ed. Berkeley, CA: Apress, 2019. 508 p. ISBN 978-1-4842-4466-1.

Tecnologia Social: O que é. Disponível em: <[https://antigo.mctic.gov.br/mctic/opencms/ciencia/politica\\_nacional/\\_social/Tecnologia\\_Social.html](https://antigo.mctic.gov.br/mctic/opencms/ciencia/politica_nacional/_social/Tecnologia_Social.html)>. Acesso em: 19 set. 2021.

UBER. 2022. Disponível em: <<https://www.uber.com/br/pt-br/>>. Acesso em: 19 out. 2022.

UNYLEYA. **Arquitetura de software**: entenda por que ela é tão importante. Disponível em: [https://blog.unyleya.edu.br/bitbyte/arquitetura-de-software/#O\\_que\\_e\\_arquitetura\\_de\\_software](https://blog.unyleya.edu.br/bitbyte/arquitetura-de-software/#O_que_e_arquitetura_de_software)>. Acesso em: 07 mar. 2022.

ZANETTE, Alysson. **Framework x Biblioteca x API. Entenda as diferenças!**. 2017. Disponível em: <https://becode.com.br/framework-biblioteca-api-entenda-as-diferencas/>>. Acesso em: 28 out. 2021.

## APÊNDICE A – DESCRIÇÃO DOS DADOS

Tabela 9 – Descrição de dados da entidade Endereco

endereco			
Atributo	Tipo de dado	(PK/FK)	Descrição
idEndereco	inteiro	PK	Identificador
logradouro	texto		Logradouro do endereço
numero	inteiro		Número do endereço
complemento	texto		Complemento do endereço
cep	texto		Cep do endereço
bairro	texto		Bairro do endereco
idCidade	inteiro	FK	Identificador da cidade

Fonte: Do Autor

Tabela 10 - Descrição de dados da entidade Cidade

cidade			
Atributo	Tipo de dado	(PK/FK)	Descrição
idCidade	inteiro	PK	Identificador
nomeCidade	texto		Nome da cidade
uf	texto		UF da cidade

Fonte: Do Autor

Tabela 11 - Descrição de dados da entidade PontoColeta

pontocoleta			
Atributo	Tipo de dado	(PK/FK)	Descrição
idPontoColeta	inteiro	PK	Identificador
nomePontoColeta	texto		Nome do ponto de coleta
telefone	texto		Telefone do ponto de coleta
idEndereco	inteiro	FK	Identificador do endereço
idJuridico	inteiro	FK	Identificador da unidade

Fonte: Do Autor



Tabela 12 - Descrição de dados da entidade Doacao

doacao			
Atributo	Tipo de dado	(PK/FK)	Descrição
idDoacao	inteiro	PK	Identificador
idPedido	inteiro	FK	Identificador do pedido sinalizado
status	caractere		Status da doação
dataConclusao	data		Data da conclusão
descricaoConclusao	texto		Descrição da conclusão
imagemConclusao	texto		Imagem da conclusão

Fonte: Do Autor

Tabela 13 - Descrição de dados da entidade Pedido

pedido			
Atributo	Tipo de dado	(PK/FK)	Descrição
idPedido	inteiro	PK	Identificador
imagemFamilia	texto		Imagem da família
idNecessidade	inteiro	FK	Identificador da necessidade
idJuridico	inteiro	FK	Identificador da unidade
idFisico	inteiro	FK	Identificador do necessitado
status	caractere		Status atual do pedido

Fonte: Do Autor

Tabela 14 - Descrição de dados da entidade Usuario

usuario			
Atributo	Tipo de dado	(PK/FK)	Descrição
idUsuario	inteiro	PK	Identificador
email	texto		Email do usuário
senha	texto		Senha do usuário
nome	texto		Nome do usuário
telefone	texto		Telefone do usuário
aprovado	booleano		Status do usuário
imagemUsuario	texto		Imagem do Usuário
idEndereco	inteiro	FK	Identificador do endereço

Fonte: Do Autor

Tabela 15 - Descrição de dados da entidade Juridico

juridico			
Atributo	Tipo de dado	(PK/FK)	Descrição
idJuridico	inteiro	PK	Identificador
descricao	texto		Descrição da instituição
cnpj	texto		Cnpj da instituição

Fonte: Do Autor

Tabela 16 - Descrição de dados da entidade Fisico

fisico			
Atributo	Tipo de dado	(PK/FK)	Descrição
idFisico	inteiro	PK	Identificador
dataNascimento	data		Data de nascimento do usuário
estadoCivil	texto		Estado civil do usuário
genero	texto		Gênero do usuário
cpf	texto		CPF do usuário

Fonte: Do Autor

Tabela 17 - Descrição de dados da entidade Necessidade

necessidade			
Atributo	Tipo de dado	(PK/FK)	Descrição
idNecessidade	inteiro	PK	Identificador
descricao	texto		Descrição da necessidade
idCategoria	inteiro	FK	Identificador da categoria

Fonte: Do Autor

Tabela 18 - Descrição de dados da entidade Categoria

categoria			
Atributo	Tipo de dado	(PK/FK)	Descrição
idCategoria	inteiro	PK	Identificador
nomeCategoria	texto		Nome da categoria

Fonte: Do Autor

Tabela 19 - Descrição de dados da entidade ResponsavelFamiliar

responsavelfamiliar			
Atributo	Tipo de dado	(PK/FK)	Descrição
idResponsavelFamiliar	inteiro	PK	Identificador
nomeResponsavel	texto		Nome do responsável
telefoneResponsavel	texto		Telefone do responsável
cpfResponsavel	texto		Cpf do responsável
quantidadePessoasFamili	inteiro		Quantidade de pessoas na família
nis	texto		Nis do responsável
idFisico	inteiro	FK	Identificador usuário

Fonte: Do Autor

Tabela 20 - Descrição de dados da entidade CampanhaNoticia

campanhanoticia			
Atributo	Tipo de dado	(PK/FK)	Descrição
idCampanhaNoticia	inteiro	PK	Identificador
titulo	texto		Titulo da campanha/notícia
subtitulo	texto		Subtitulo da campanha/notícia
descricao	texto		Descricao da campanha/notícia
autor	texto		Autor da campanha/notícia
dataPublicacao	data		Data da campanha/notícia
imagem	texto		Imagem da campanha/notícia

Fonte: Do Autor

## APENDICE B – CENÁRIOS DE CASOS DE USO

UC1 – Consultar Pedidos de Doação Sinalizados

Ator(es): CRAS

Pré-condições

FA1 – Pelo menos um pedido de doação cadastrado

Fluxo Principal

1. [RS] O sistema exibe uma tela com os pedidos de doação sinalizados  
[F1]

Fluxo Alternativo

FA1 – Visualizar pedido de doação

1. [EV] O ator seleciona um pedido da lista.
2. [RS] O sistema exibe uma tela com as informações sobre o pedido de doação.
3. [EV] O ator informa uma descrição para a conclusão do pedido de doação.
4. [EV] O ator insere uma foto para a conclusão do pedido de doação.
5. [RS] O sistema exibe a mensagem de sucesso.
6. [RS] O sistema atualiza a lista de pedidos.

Pós-condições

FA1 – Um pedido de doação terá seu registro removido da listagem.

## UC2 – Consultar Pedidos de Doação Concluídos

Ator(es): CRAS

Pré-condições: FA1 – Pelo menos um pedido de doação cadastrado

Fluxo Principal

1. [RS] O sistema exibe uma tela com os pedidos de doação concluídos

Fluxo Alternativo

FA1 – Visualizar pedido de doação

1. [EV] O ator seleciona um pedido da lista.
2. [RS] O sistema exibe uma tela com as informações sobre o pedido de doação.

Pós-condições

FA1 – Não há.

## UC3 – Consultar usuários

Ator(es): CRAS

Pré-condições: FA1 – Pelo menos um usuário cadastrado.

Fluxo Principal

1. [RS] O sistema exibe uma tela com os usuários cadastrados aprovados e aguardando aprovação.

[FA1]

Fluxo Alternativo

FA1 – Visualizar informações do usuário

1. [EV] O ator seleciona um usuário da lista.
2. [RS] O sistema exibe uma tela com as informações do usuário.

Pós-condições

FA1 – Não há

#### UC4 – Cadastrar pedido de doação

Ator(es): CRAS, Doador/Necessitado, Instituição

Pré-condições: Realizar autenticação no sistema

##### Fluxo Principal

1. [EV] O ator seleciona a opção cadastro de necessidade.
2. [RS] O sistema carrega a tela de cadastro de necessidade.
3. [EV] O ator informa a descrição da necessidade.
4. [EV] O ator seleciona uma categoria para o pedido de doação.
5. [EV] O ator informa o nome do responsável da família.
6. [EV] O ator informa a quantidade de pessoas na família.
7. [EV] O ator informa o telefone do responsável da família.
8. [EV] O ator pode selecionar uma imagem para o pedido de doação.
9. [EV] O ator salva o pedido de doação.
- 10.[RS] O sistema exibe a mensagem de sucesso.
- 11.[RS] O sistema atualiza a lista de aprovação do CRAS.

Pós-condições: Um pedido é inserido na lista pedidos para aprovação do CRAS.

#### UC5 – Cadastrar necessitado

Ator(es): CRAS, Instituição

Pré-condições: Não há

##### Fluxo Principal

1. [EV] O ator seleciona a opção de cadastrar necessitado.
2. [RS] O sistema carrega a tela de cadastro de necessitado.
3. [EV] O ator informa o nome do necessitado.
4. [EV] O ator informa a data de nascimento do necessitado.
5. [EV] O ator informa o estado civil do necessitado.
6. [EV] O ator informa o gênero do necessitado.
7. [EV] O ator informa o cpf do necessitado.
8. [EV] O ator informa o telefone do necessitado.
9. [EV] O ator informa o email do necessitado.
- 10.[EV] O ator informa a senha do necessitado.

- 11.[EV] O ator informa o logradouro do necessitado.
- 12.[EV] O ator informa o número da casa do necessitado.
- 13.[EV] O ator pode informar o complemento do endereço do necessitado.
- 14.[EV] O ator informa o cep do necessitado.
- 15.[EV] O ator informa o bairro do necessitado.
- 16.[EV] O ator informa a cidade do necessitado.
- 17.[EV] O ator informa a unidade federativa do necessitado.
- 18.[EV] O ator informa o nome do responsável familiar.
- 19.[EV] O ator informa o telefone do responsável familiar.
- 20.[EV] O ator informa o cpf do responsável familiar.
- 21.[EV] O ator informa a quantidade de pessoas na família.
- 22.[EV] O ator informa o número do NIS do necessitado.
- 23.[EV] O ator pode selecionar uma imagem para o necessitado.
- 24.[EV] O ator seleciona salvar o necessitado.
- 25.[RS] O sistema exibe a mensagem de sucesso.

Pós-condições: O necessitado vai para lista de aprovação do CRAS.

UC6 – Consultar pedidos de doação em aberto e concluídos

Ator(es): CRAS, Usuário, Instituição, Visitante

Pré-condições: Não há

Fluxo Principal

1. [RS] O sistema exibe uma tela com os pedidos de doação cadastrados que ainda não foram sinalizados e concluídos.

Pós-Condições: Não há

#### UC7 – Consultar campanhas e notícias

Ator(es): CRAS, Usuário, Instituição, Visitante

Pré-condições: Não há

Fluxo Principal

1. [RS] O sistema exibe uma tela com as campanhas e notícias cadastradas.

Pós-condições: Não há

#### UC8 – Cadastrar campanhas e notícias

Ator(es): CRAS, Instituição

Pré-condições: Realizar autenticação no sistema

Fluxo Principal

1. [EV] O ator seleciona a opção cadastrar campanha/notícia.
2. [RS] O sistema carrega a tela de cadastro de notícia/campanha.
3. [EV] O ator informa o título da campanha/notícia.
4. [EV] O ator informa o subtítulo da campanha/notícia.
5. [EV] O ator informa a descrição da campanha/notícia.
6. [EV] O ator informa o autor da campanha/notícia.
7. [EV] O ator seleciona uma imagem para campanha/notícia.
8. [EV] O ator seleciona salvar a campanha/notícia.
9. [RS] O sistema exibe a mensagem de sucesso.
10. [RS] O sistema atualiza a listagem de campanhas e notícias.

Pós-condições: Não há.

#### UC9 – Cadastrar ponto de coleta

Ator(es): CRAS, Instituição

Pré-condições: Não há

Fluxo Principal

1. [EV] O ator seleciona a opção de cadastro de ponto de coleta.
2. [RS] O sistema carrega a tela de cadastro de ponto de coleta.



3. [EV] O ator informa o nome do ponto de coleta.
4. [EV] O ator informa o telefone do ponto de coleta.
5. [EV] O ator informa o logradouro do ponto de coleta.
6. [EV] O ator informa o número do endereço do ponto de coleta.
7. [EV] O ator pode informar o complemento do endereço do ponto de coleta.
8. [EV] O ator informa o cep do ponto de coleta.
9. [EV] O ator informa o bairro do ponto de coleta.
- 10.[EV] O ator informa a cidade do ponto de coleta.
- 11.[EV] O ator informa a unidade federativa do ponto de coleta.
- 12.[EV] O ator seleciona salvar ponto de coleta.
- 13.[RS] O sistema exibe a mensagem de sucesso.

Pós-condições: Não há

#### UC10 – Consultar instituições

Ator(es): CRAS

Pré-condições: FA1 – Pelo menos uma instituição cadastrada.

Fluxo principal

1. [RS] O sistema exibe uma tela com as instituições cadastradas aprovadas, e aguardando aprovação.  
[FA1]

Fluxo Alternativo

FA1 – Visualizar informações da instituição

1. [EV] O ator seleciona uma instituição na listagem.
2. [RS] O sistema exibe uma tela com as informações detalhadas da instituição.
3. [EV] O ator seleciona aprovar a instituição.
4. [RS] O sistema exibe a mensagem de sucesso.
5. [RS] A lista de instituições é atualizada

Pós-condições:

FA1 – O status da instituição é atualizado e a listagem é atualizada.

UC11 – Concluir doação realizada

Ator(es): CRAS

Pré-condições: FA1 – Pelo menos uma doação sinalizada.

Fluxo Principal

1. [RS] O sistema exibe uma tela com os pedidos de doação sinalizados.

Fluxo Alternativo

FA1 – Visualizar informações do pedido de doação

1. [EV] O ator seleciona um pedido na listagem.
2. [RS] O sistema exibe uma tela com as informações detalhadas da instituição.
3. [EV] O ator seleciona uma imagem para a conclusão do pedido de doação.
4. [EV] O ator informa a descrição da conclusão do pedido de doação.
5. [EV] O ator seleciona salvar a conclusão do pedido de doação.

Pós-condições: FA1 – O status do pedido de doação é atualizado e a listagem é atualizada.

UC12 – Consultar dados da instituição

Ator(es): CRAS, Instituição

Pré-condições: Não há

Fluxo Principal

1. [RS] O sistema exibe uma tela com as informações da instituição.

Pós-condições: Não há.

### UC13 – Consultar pedidos de doação realizados

Ator(es): Usuário, Instituição

Pré-condições: FA1 – Pelo menos um pedido de doação cadastrado pelo ator.

#### Fluxo Principal

1. [RS] O sistema exibe uma tela com a lista de pedidos de doação cadastrados pelo ator.

[FA1]

#### Fluxo Alternativo

1. [EV] O ator seleciona um pedido de doação na listagem.
2. [RS] O sistema exibe uma tela com as informações detalhadas do pedido.

Pós-condições: Não há

### UC14 – Consultar dados de usuário

Ator(es): Usuário

Pré-condições: Não há

#### Fluxo Principal

1. [RS] O sistema exibe uma tela com as informações detalhadas do usuário.

Pós-condições: Não há.

UC15 – Sinalizar pedido de doação

Ator(es): CRAS, Usuário, Instituição

Pré-condições:

Realizar autenticação no sistema

FA1 – Pedido ainda não sinalizado e não concluído.

Fluxo Principal

1. [RS] O sistema exibe uma tela com as informações detalhadas do pedido.

[FA1]

Fluxo Alternativo

FA1 – Sinalizar doação

1. [EV] O ator seleciona a sinalização do pedido de doação.

2. [RS] O sistema exibe a mensagem de sucesso.

3. [RS] A listagem é atualizada.

Pós-condições: A listagem é atualizada.