

Universidade de São Paulo
Escola de Artes, Ciências e Humanidades

Bruno Yoshikazu Shimada

**Desenvolvimento de aplicativo para micropagamentos
na plataforma Android**

São Paulo
2017

Bruno Yoshikazu Shimada

Desenvolvimento de aplicativo para micropagamentos na plataforma Android

Relatório parcial apresentado à Escola de Artes,
Ciências e Humanidades, da Universidade de São
Paulo, como parte dos requisitos exigidos na
disciplina ACH 2018 - Projeto Supervisionado ou
de Graduação II, para obtenção do título de
Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Luciano Vieira de Araújo

São Paulo

2017

Nome: SHIMADA, Bruno Yoshikazu

Título: Desenvolvimento de aplicativo para micropagamentos na
plataforma Android

Relatório parcial apresentado à Escola de Artes,
Ciências e Humanidades, da Universidade de São
Paulo, como parte dos requisitos exigidos na
disciplina ACH 2018 - Projeto Supervisionado ou
de Graduação II, para obtenção do título de
Bacharel em Sistemas de Informação.

Aprovado em: ____/____/____

Banca Examinadora

Prof. Dr:_____ Instituição:_____

Julgamento:_____ Assinatura:_____

Prof. Dr:_____ Instituição:_____

Julgamento:_____ Assinatura:_____

Prof. Dr:_____ Instituição:_____

Julgamento:_____ Assinatura:_____

Agradecimientos

section

Resumo

Vivemos em uma época que a cada dia, novas soluções digitais são lançadas para resolver nossos problemas cotidianos de uma maneira simplificada que se incorporam ao nosso cotidiano de uma maneira que após algum tempo não conseguimos imaginar como nós conseguimos viver sem isso por tanto tempo.

Entre as soluções digitais podemos citar por exemplo o ramo econômico e bancário, com desde os básicos aplicativos de bancos para consulta a extratos, pagamentos de boletos, etc, até soluções mais sofisticadas como aplicativos para receber e processar pagamentos por cartão.

Porém um problema corriqueiro que temos no dia-a-dia é o pagamento de pequenas transações monetárias, como tomar um café na padaria, comprar um chiclete na bomboniere, emprestar uma pequena quantia de dinheiro para um conhecido, entre outras que na maioria das vezes não passam de 10R\$ e que geram uma perda de tempo de ter que realizar o pagamento com um cartão, digitando a senha pessoal em uma máquina e torcendo para o serviço da operadora do cartão estar disponível no momento.

Para resolver esse problema entram em cena os serviços de micropagamento, que buscam solucionar o problema de fazer um pagamento ou uma transação de pequeno valor entre duas pessoas, independente de ser pessoa-a-pessoa ou pessoa-a-negócio, de forma rápida, eficaz, segura e descomplicada. O propósito deste trabalho é então desenvolver um aplicativo Android, SO mais comum entre os usuários de celular no Brasil, que gere as transações entre usuários do aplicativo e que faça valer as características descritas anteriormente.

Palavras Chaves

- Micropagamentos
- *Android*
- *Fintech*

Abstract

We live in time that everyday new digital solutions are released to solve our daily problems in a simplified manner, that merges into our days that after some time we are unable to imagine how we lived so much time without it.

Among these digital solutions we can mention for example the economic/banking areas, that has the basic bank app for checking account information, payment of tickets, etc.. To more sophisticated apps that receive and process payments by card.

However a common problem we have in our daily lives is the payment of small monetary transactions, such as having a coffee at a bakery, buying gum from a local store, lending money to an acquaintance, among others that in most times dont surpass 10 BRL and generate a loss of time for having to make the payment with a card, typing our personal password on a machine and hoping the service of the card operator will be available at the moment.

To solve this problem the micropayments services come into play, seeking to solve the problem of making the payment or transaction of a small amount of money between two persons, regardless of being a person-to-person or person-to-business, in a quick, efficient, safe and uncomplicated way. The purpose of this work is to develop an Android app, the most common OS among mobile users in Brazil, that manage transaction between users of the app and enforce the characteristic described before.

Keywords

- Micropayments
- Android
- Fintech

Lista de Figuras

1	Ícone usado para simbolizar a ação de pagamento	6
2	Interface de cadastro de usuário'	10
3	Demonstração do uso de <i>hint</i> e preenchimento de campo com atributo <i>textPassword</i>	11
4	Interface principal do aplicativo	12
5	Demonstração do uso do <i>Spinner</i>	13
6	Interface do recebente aguardando contato por <i>Bluetooth</i>	14
7	Interfaces de confirmação de pagamento	15
8	Interfaces de carregamento de valores	16
9	Alerta de erro	17
A.1	sample	22

Sumário

1	Introdução	1
2	Objetivos	3
2.1	Objetivo Geral	3
2.2	Objetivos Específicos	3
3	Revisão Bibliográfica	4
4	Metodologia	7
5	Resultados	9
5.1	Desenvolvimento das interfaces	9
5.2	Desenvolvimento das interfaces do aplicativo	17
6	Considerações Finais	18
	Referências Bibliográficas	20
	Apêndice A Fluxo do aplicativo	22

1 Introdução

A definição de micropagamento é relativamente simples, se usado uma associação das palavras que a compõem, se infere que se trata de transações cujo valor é uma quantia muito pequena. Trazendo para a realidade dele, um micropagamento é uma transação online de uma quantia muito pequena de dinheiro (INVESTOPEDIA). Quão pequeno esse valor é, varia entre as diferentes empresas no mercado, o *PayPal*, *e-commerce* que faz a transação de valores digitais entre usuários nas duas pontas, considera um micropagamento qualquer transação cujo valor seja menor do que 10USD (PAYPAL).

A evolução do mercado de micropagamentos é creditada a 3 fatores (*HERNANDEZ - VERME, BENAVIDES; 2013*) definidos com base em um relatório de *VASILJEV (2016)* e *BURELLI (2016)*, definem eles como:

- O crescimento da infraestrutura de rede e do e-commerce em geral
- O crescimento das redes sociais, jogos online e negócios de bens digitais
- O aparecimento de novas formas de serviços de pagamento online

Com base nisso podemos ver que as soluções atuais para micropagamento surgiram principalmente da necessidade de pagamento de bens para consumo digital como assinaturas de sites, compras de músicas digitais, o melhor exemplo aqui seria o *iTunes* por exemplo. Dentre as soluções que existem atualmente, a *Investopedia* dá foco em dois modelos:

- A plataforma agindo como carteira digital. Cada usuário cria sua conta na plataforma que irá gerenciar a transação entre dois usuários, a transação é efetuada e a plataforma se encarrega de armazenar esse valor, quando a carteira de qualquer uma dessas partes atinge um limite, a quantia total do dinheiro é liberada e transferida para o usuário.
- A plataforma agindo com créditos. Cada usuário cria sua conta no plataforma, cada um deles compra/recarrega a quantia desejada que deseja ter como crédito, a cada transação entre duas partes a plataforma se encarrega de atualizar os créditos de ambas as partes.

O grande problema na área de micropagamentos são as taxas cobradas pelas operadoras de cartão de crédito, meio que a maioria dos modelos existentes como demonstrado acima usa como forma preferencial de pagamento o cartão crédito, porém é sabido que para cada transação são cobradas taxas em cima do valor pago, é um caso raro mas podem existir situações que as taxas podem, porque não, superar o valor da transação, o que para um cliente que seja dono de um negócio, torna-se algo insustentável.

Uma plataforma de micropagamentos também pode se aplicar ao mundo offline como uma forma de currencia que substitua o dinheiro padrão em pequenas comunidades.

Imagine por exemplo o ambiente de um condomínio privado que ofereça serviços internos para os condminos, a forma natural que se imagina que as transações vão ocorrer nesse meio é por forma de dinheiro em espécie, o que levanta alguns riscos para essa população, sabendo que nessa ambiente circula dinheiro vivo, e que a segurança é alguns níveis mais baixa que a comum, alguma pessoa mal intencionada pode pensar em levar vantagem e cometer um assalto contra os prestadores desse serviço visando roubar o dinheiro que fica acumulado.

Com o uso de uma plataforma de micropagamentos os condminos poderiam fazer a carga de uma quantia de dinheiro digital em um aplicativo, os prestadores de serviço aceitarem essa forma de pagamento, e a pessoa fazer a transação usando esse dinheiro digital, isso faz com que não circule dinheiro vivo no ambiente deles, o dinheiro está circulando, porém ele está externo ao ambiente, ele não pode ser acessado de uma maneira física naquele local o que pode ajudar a inibir pessoas que pensem em roubar o condomínio já que não haveria dinheiro físico no local.

Essa mesma abordagem pode ser aplicada para por exemplo, o restaurante universitário da nossa faculdade. Ao inve de pagar a carga do cartão com o dinheiro físico, os alunos poderiam fazer isso com um aplicativo, efetuando a carga de créditos com um cartão de crédito por exemplo e pagando na entrada com o seu celular e a funcionária do restaurante com o celular da empresa responsável recebendo os pagamentos.

2 Objetivos

2.1 Objetivo Geral

O objetivo do trabalho foi desenvolver um aplicativo *Android* que conseguisse efetuar micropagamentos entre *smarthphones*.

2.2 Objetivos Específicos

Os objetivos específicos do trabalho foram:

- Aprendizagem do conceito de micropagamentos
- Estudo da bibliografia de artigos de micropagamentos.
- Desenvolver a *UI* do aplicativo
- Integrar a interface com base em uma plataforma de serviços e infraestrutura existente.

3 Revisão Bibliográfica

Ao longo do desenvolvimento do aplicativo, em diversos momentos apareceram dúvidas que não conseguiam ser resolvidas por força bruta, como por exemplo, como definir a posição fixa de um botão em relação a outros no mesmo layout? Dúvidas desde as mais simples, que surgem principalmente nos momentos iniciais de aprendizado de alguma linguagem ou *framework* novo, até as mais complexas, quando já se tem uma base sólida de conhecimento sobre o tópico, são comuns no meio da tecnologia, fato que pode ser comprovado fazendo uma simples busca no *Google*, nas páginas de resultado é comum encontrar pessoas com a mesma dúvida, em variados fóruns, blogs e sites diversos. O mais interessante nesse ponto é a também a variedade de abordagens diferentes que são possíveis de encontrar para uma mesma dúvida, fazendo com que tenhamos que filtrar dentre elas a que melhor se aplica ao contexto do nosso problema.

Abaixo estão listadas as fontes consultadas ao longo do trabalho que ajudaram em diversos momentos do desenvolvimento.

Para entendimento do conceito de micropagamentos foram usadas duas fontes principais.

A página do termo [1] no site da *Investopedia* que deu uma visão geral sobre o assunto, enfocando primeiro em um resumo simplificado para depois se aprofundar um pouco mais nele. O interessante desse site é que a partir do termo "*micropayment*", ele busca termos relacionados buscando fazer uma trilha de informações para abranger o assunto.

Outra fonte consultada foi o artigo "***Virtual currencies, micropayments and the payments systems: a challenge to fiat money and monetary policy?***" [2] que discorre não só sobre o conceito de micropagamentos como também aborda o tema das moedas virtuais e relaciona ambos. O artigo foca mais na parte de como os micropagamentos são mais relevantes no mundo online na compra de itens que eles classificam como microprodutos, fazem uma comparação das transações online que envolvem produtos físicos e digitais. Esse artigo considero como o principal usado já que ele aborda de uma maneira mais aprofundada o ambiente que os micropagamentos se inserem.

O artigo "***Micropayments: A Viable Business Model?***" [3] apresenta um ponto interessante nas questões sobre os desafios técnicos que uma aplicação de micropagamentos teria que se preocupar em atender para desenvolver uma aplicação segura. Os autores listam 5 pontos que consideram serem desafios que precisam ser vencidos para uma implementação bem-sucedida, segurança, escalabilidade, confiabilidade, interoperabilidade e anonimidade. O interessante desse artigo é ele fazer o relacionamento do ecossistema de micropagamentos com os componentes e requisitos desejáveis para desenvolvimento de um aplicativo.

Segurança é um requisito necessário para praticamente qualquer aplicativo, porém

quando tal aplicação envolve dinheiro e transações ele se torna um problema de alta prioridade de ser resolvido, ter um aplicativo inseguro que apresente muitas falhas, e entre elas as que sujeitam o aplicativo a sofrer ataques que comprometam as informações e dados de usuários, podem fazer a confiança nele cair, o que reduz a base de usuários. Os autores nesse item que durante o desenvolvimento é preciso tomar atenção com autenticação, autorização, integridade de informação e confidencialidade.

Arquitetar o sistema de uma maneira que permita que ela seja escalável é um requisito necessário para poder lidar com um pontencial aumento do número de acessos a aplicação. O desenho da solução deve ser de tal forma que seja possível acoplar mais servidores a medida que foram necessários para poder lidar com o tráfego de informações.

Levantado também a questão de interoperabilidade, permitir que diferentes aplicações de micropagamentos conversem entre si, este talvez seja o ponto de maior dificuldade de se atingir. Aplicações diferentes possuem diferentes meios de tratar a informação que chega, sai, e processada nele.

A questão da anonimidade é tratado como um ponto complicado de se definir, até que ponto a anonimidade deve existir e a partir de que ponto é necessário saber a identidade dos usuários, quanto um usuário pode e deve saber do outro, questões que valem para os dois atores envolvidos em um micropagamento. O artigo não levanta nenhuma resposta definitiva sobre o assunto, ele apresenta bons argumentos a favor e contra a quantidade de anonimidade no sistema.

Um dos exemplos é explorado é o registro de histórico de transações, para um usuário que faz o pagamento é uma informação til de ser acessada, mas que precisa de uma pequena fração de privação de anonimidade. Mas ao mesmo tempo é informação que pode ser usada por terceiros para descobrir padrões de consumo e o que foi consumido, e que pode ser usado contra o usuário. Um *e-commerce* por exemplo, com base nessas informações pode manipular os preços e anúncios que tem potencial de ser interesse de tal usuário.

No começo do projeto meu conhecimento de programação para *Android* era nulo, durante a preparação da máquina encontrei no mesmo site que disponibiliza a *IDE* para desenvolvimento, uma sub-seção intitulada *Training*, debaixo da seção *Develop* [4]. Nela encontrei uma série de lições básicas para quem está começando a desenvolver para *Android*. Foi útil para ter uma noção básica de onde começar e os termos específicos usados.

Após a criação de um aplicativo *Hello World* básico seguindo o guia [5] do site da *Android*, fui atrás de mais algum material para criar um aplicativo e aprofundar um pouco mais o conhecimento na plataforma. O livro [6] utilizado nesse momento foi **"Android 5 Programming by Example"**, o livro [6] se aprofunda um pouco mais nos exercícios ajudando e servindo de guia para criar um aplicativo mais robusto do que o visto no

guia do site da *Android*. O livro [6] foi usado de guia no segundo passo do projeto onde criei um aplicativo *mock* que simulava localmente uma transação nos moldes de um micropagamento de um aparelho para um servidor, para testar os conhecimentos e novas técnicas adquiridas. Apesar do livro [6] focar mais na versão *Lollipop*, lançada em 2015 mesmo ano da publicação do livro, grande parte dos componentes não mudou muito para a versão mais atual.

Durante a elaboração do rascunho das interfaces do aplicativo uma página [7] que foi muita usada como fonte de consulta e guia é a *Material Design*. Nela foi possível encontrar orientações e guias de como desenhar a interface para se encaixar no padrão do *Material Design*. A seguir são descritas as ferramentas e guias disponíveis que se destacam pelo seu uso e importância no projeto.

A página ***Material Icons*** [8] é uma espécie de repositório da *Google* com vários ícones disponíveis para uso. Todos são liberados para uso sobre a licença *Apache License Version 2.0*. Os ícones são desenhados de maneira simples e de fácil interpretação, o design deles é de uma maneira que quando se olha para o ícone é fácil deduzir qual seu significado. Por exemplo o ícone da figura 1 que foi usado para simbolizar a ação de pagamento:



Figura 1: Ícone usado para simbolizar a ação de pagamento

Além da página com o repositório dos ícones, o design e o conceito dos itens é explicado no artigo ***Style - Icons*** [9]

A página ***Material Colors*** [10] possui uma ferramenta [11] que ajuda na escolha da paleta de cores que vão compor o aplicativo, ele oferece também um guia visual de como ficam as cores nas interfaces do sistema. Outro artigo relacionado é ***Style - Colors*** [10] onde é explicado as boas práticas e guias para selecionar as cores dos componentes da interface do aplicativo.

4 Metodologia

Para o projeto eram necessários conhecimentos em dois tópicos, micropagamentos e *Android*. Inicialmente foi feito um estudo da bibliografia de micropagamentos, descrita no capítulo 3, para entender melhor o conceito dele como um todo, suas características, limites, o panorama atual e o mercado. Tendo aprendido esse conceito foi feito um estudo buscando aprender os conceitos da programação para *Android*, posteriormente foram desenvolvidos alguns aplicativos de teste para aplicar o conhecimento adquirido.

Adquiridos os fundamentos necessários sobre os dois assuntos dei andamento no trabalho para definir o escopo dele, foram definidas quantas e quais interfaces seriam necessárias para atingir o objetivo proposto, desenvolver a interface de um aplicativo *Android* com base numa plataforma de serviços existentes buscando fazer a mesma ser simples, no sentido de ter uma fácil utilização, buscando obedecer e seguir os guias definidos pelo *Android* como boas práticas e que atendem e se adequam a filosofia do *Material Design*. Foram avaliados nesse momento as estruturas e componentes que seriam usados, como por exemplo botões, layouts, esquema de cores, etc.

Definida as interfaces passei a avaliar a estrutura de serviços com as quais as interfaces seriam integradas, foi o momento de aprender como funcionava a plataforma, como ela se comunicava internamente entre suas classes, quais eram as entradas e saídas de cada momento. Com isso foi refinado a etapa anterior de definição das interfaces para se adequar a plataforma, foram revistas as interfaces imaginadas, adicionando e retirando alguns componentes visuais, e criadas novas interfaces.

Com o escopo definido foi feita uma avaliação de esforço para definir o cronograma que seria seguido e dado início ao desenvolvimento das interfaces do aplicativo.

Para o desenvolvimento das interfaces do aplicativo, e controle das ações foram utilizadas duas linguagens no trabalho:

- ***XML***: todas as interfaces do *Android* são escritas usando a linguagem de marcação *XML*, o *Android* usa um conjunto de elementos e atributos próprios na marcação como por exemplo o elemento `<RelativeLayout>` e o atributo `android:layout_width`. *XML* também usado para definir *Strings*, o tema base, e qualquer outro elemento visual que possa ser reaproveitado, como por exemplo um *layout* customizado para um botão. Também é usado para definir o manifesto do aplicativo, onde são declaradas configurações específicas do aplicativo, como por exemplo permissão para uso de *bluetooth*, *internet*, etc.
- ***Java***: A parte do *back-end* que controla a aplicação é toda codificada em *Java*. O *Android* usa *Java* para por exemplo codificar um *listener* que capta quando o usuário clica em algum botão por exemplo, para fazer a troca de interfaces, controle

de sessão, carregar *Strings* dinâmicas na interface, tratamento de erros e exceções, etc.

Quanto a *hardware* foram usados um *notebook* próprio rodando um SO *Debian*, e dois aparelhos celulares próprios com SO *Android* para simular o comportamento de dois usuários. Devido a diferença de SO de ambos, foi interessante observar como a interface se comportava em cada um deles. A título de curiosidade seguem as especificações:

- *Notebook*
 - SO: *Ubuntu 16.04 LTS - 64-bit*
 - Memória RAM: 8GB
 - Processador: *Intel®Core™ i7-5500U CPU @ 2.40GHz x 4*
 - Processador gráfico: *Intel®HD Graphics 5500 (Broadwell GT2)*
- Celular *Motorola Moto G 3ª Geração*
 - SO: *Android 5.1 Lollipop*
 - Memória RAM: 1GB
 - Processador: *Qualcomm Snapdragon 410*
- Celular *Samsung S3 Mini*
 - SO: *Android 4.1 Jellybean*
 - Memória RAM: 1GB
 - Processador: *Dual-core, 1000 MHz*

5 Resultados

5.1 Desenvolvimento das interfaces

Foram definidas inicialmente para o trabalho o desenho de 4 interfaces, *login*, uma tela principal com as ações permitidas ao usuário, a tela de efetuar o micropagamento e a tela de confirmação do pagamento, após estudo da estrutura da plataforma de serviços e revisão das interfaces, o número de interfaces que seriam necessárias ficou em 13. Para cada uma delas será descrita o que foi feito e os componentes usados.

Para o desenvolvimento das interfaces foi levantado duas abordagens diferentes, o uso de *RelativeLayout* ou o uso do *ConstraintLayout*, ambas alternativas foram testadas para mensurar as possibilidades e dificuldades que cada uma apresentava, por fim foi escolhido seguir o desenvolvimento usando o primeiro. A vantagem no uso do *RelativeLayout* é poder declarar a posição dos componentes de acordo com seus parentes ou seus irmãos, ele dá um maior controle sobre os itens no *layout*. Por exemplo, temos uma *view* que define uma área retangular na interface, e dentro dessa área precisam estar 3 outros componentes, vamos dizer botões, e esses 3 componentes precisam estar alinhados direita da área. Com o *RelativeLayout* declaramos a área, o primeiro botão alinhado a direita da área, e cada um dos botões subsequentes um abaixo do outro, com a *ConstraintLayout* o mesmo processo precisa que algumas propriedades a mais sejam usadas para obter o mesmo resultado.

A primeira interface a ser feita foi a de cadastro de usuário (imagem 2). No fluxo de execução normal, essa interface é a primeira, porém ela deve ocorrer apenas uma vez no momento que o usuário vai se cadastrar, se o usuário já fez o cadastro essa tela não volta a ser exibida sendo exibida no seu lugar a interface de *login* (imagem 3a).

Para a interface de cadastro (imagem 2) foram necessários avaliar e estudar a implementação de três tipos de componentes: *EditText*, *Button* e *ImageView*, os três componentes não apresentaram grande dificuldade em assimilar o comportamento e o que deveria ser feito para aplicar na interface.



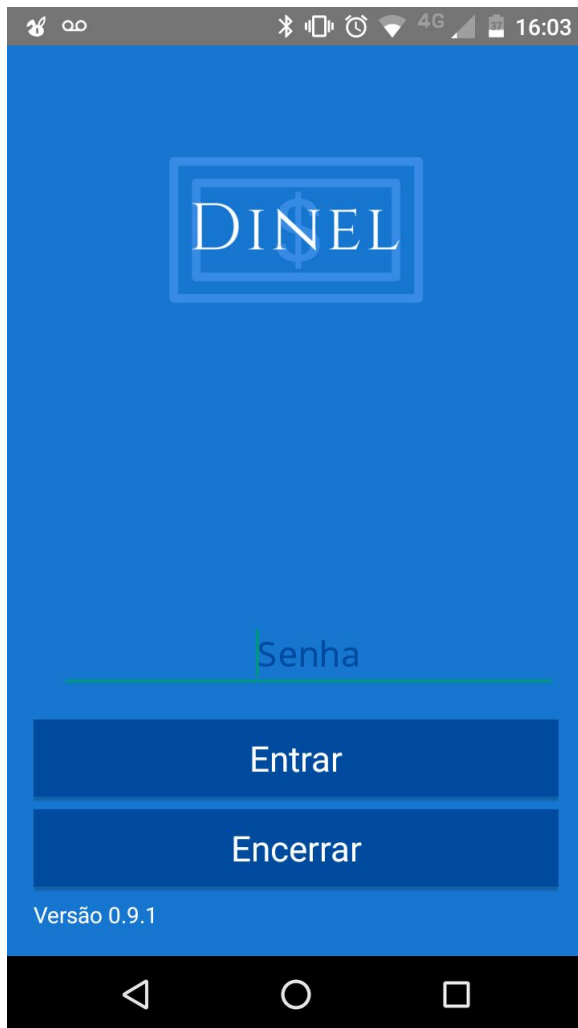
Figura 2: Interface de cadastro de usuário'

O componente *EditText*, simplificadaamente, permite que o usuário insira caracteres na interface para serem interpretados pela aplicação. Na interface eram necessários que o usuário entrasse com duas *strings*, uma para que ele definisse qual o nome de usuário e outra para a senha. A de entrada do nome de usuário foi mais direta a aplicação, a entrada de *strings* é um componente padrão que não precisou de muita configuração, a de senha era necessário aplicar uma máscara que ocultasse os caracteres a medida que eles fossem digitados, por questão de segurança e privacidade, o efeito desejado foi facilmente obtido apenas atribuindo o valor *textPassword* na propriedade *InputType* (exemplo na imagem 3b), essa propriedade é a que controla, nesse caso, se deve ser aplicado uma máscara de senha, ou como será visto mais a frente, se o teclado de entrada deve exibir apenas números ou números e letras.

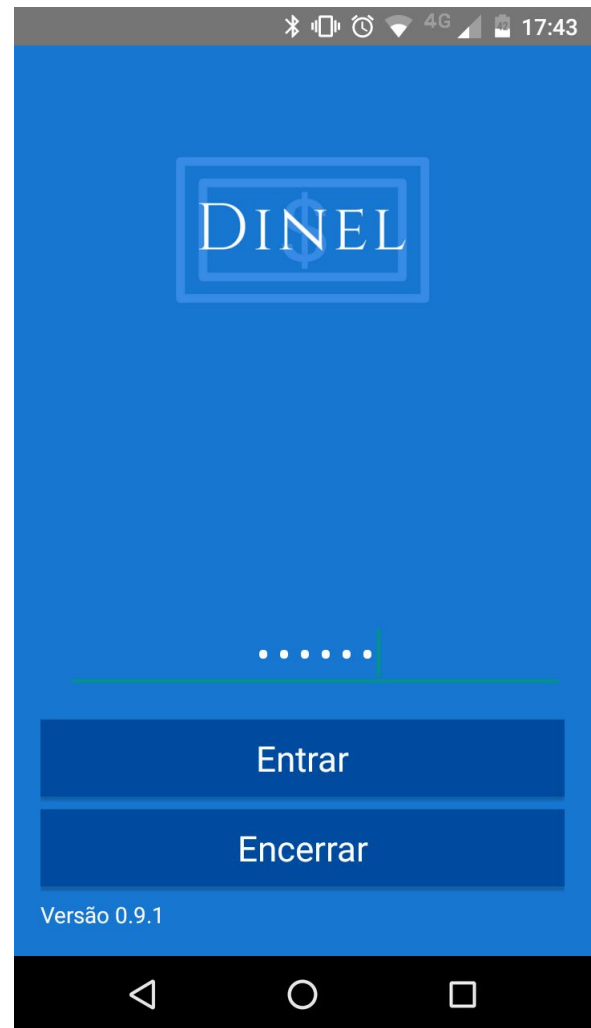
O componente *Button* e o *ImageView* exigiram menos customização para serem aplicados, o botão exigiu apenas que fossem definidos as cores de fundo e de texto que são exibidos nele, a imagem apenas que fosse indicado qual o caminho e identificador da ima-

gem. Um ponto a ressaltar que encontrei dificuldade foi no posicionamento correto da imagem no *layout*, foi preciso voltar a documentação do *Android* para entender a hierarquia de *views* [12] e entender onde a imagem deveria ser posicionada para apresentar o comportamento esperado.

A interface de *login* (imagem 3a), já que ela também é uma das interfaces de entrada do aplicativo, como j explicado essa tela após o usuário se cadastras é a primeira que vai ser carregada, ela não precisou que fosse implementado nenhum componente novo, os usados aqui foram os mesmo da interface de cadastro descritos acima. Uma diferença entre elas é que como a nica entrada de texto necessário é a senha do usuário não foram usados *TextView* para indicar o que é para ser iserido no campo, em seu lugar foi usado a propriedade *hint* que, como o nome indica, exibe uma dica para o usuário do que deve ser colocado no campo, e desaparece quando o usuário clica nele.



(a) Interface de *login* do aplicativo



(b) Campo de senha preenchido

Figura 3: Demonstração do uso de *hint* e preenchimento de campo com atributo *textPassword*

A interface principal do aplicativo é de onde o usuário pode selecionar qual ação ele deseja tomar, se ele deseja fazer um micropagamento, se quer receber, fazer uma carga de créditos ou encerrar a sessão, todas essas ações foram mapeadas para botões dispostos em formatos de blocos. Nessa interface foram adicionados ícones nos botões para simbolizar cada uma das ações, como por um exemplo a imagem de um cartão 1 para indicar que é um micropagamento a ser efetuado. Nenhum componente diferente dos citados até agora foi usado nesse momento. O ponto de atenção nessa tela foi tomar cuidado com o identificador que foi atribuído para cada um dos botões, já que no *back-end* os métodos que disparam cada ação buscam por esse identificador, um nome colocado errado em alguma das pontas pode ocasionar que a ação desejada não seja disparada ou que outra seja ativada no seu lugar. Dito isso a interface pode ser vista na imagem 4.



Figura 4: Interface principal do aplicativo

A principal ação do aplicativo é a de efetuar um micropagamento, o fluxo de pagamento envolve dois usuários e 4 interfaces no caso de sucesso, a de seleção de cedente, confirmação de transação do lado do pagante, aguardando conexão e confirmação de transação do

lado do recebente. Iniciando pelo lado do pagante, como já dito são duas interfaces, a primeira o usuário efetuando o pagamento vai selecionar numa lista, obtida com base nos usuários com quem ele já pareou o aparelho, e confirmar ou cancelar a transação. Na interface de seleção foi usado o componente *Spinner*, ele é um componente que permite selecionar um valor de uma lista de opções possíveis, quando não selecionado ele assume um valor padrão pré-definido, quando selecionado ele exibe em uma lista todos os valores disponíveis, quando um valor é selecionado ele fica exibido onde previamente estava o valor padrão. Este comportamento pode ser observado na imagem 5, est exemplificado o componente selecionado (imagem 5a) e como ele fica aps um valor ter sido selecionado (imagem 5b)

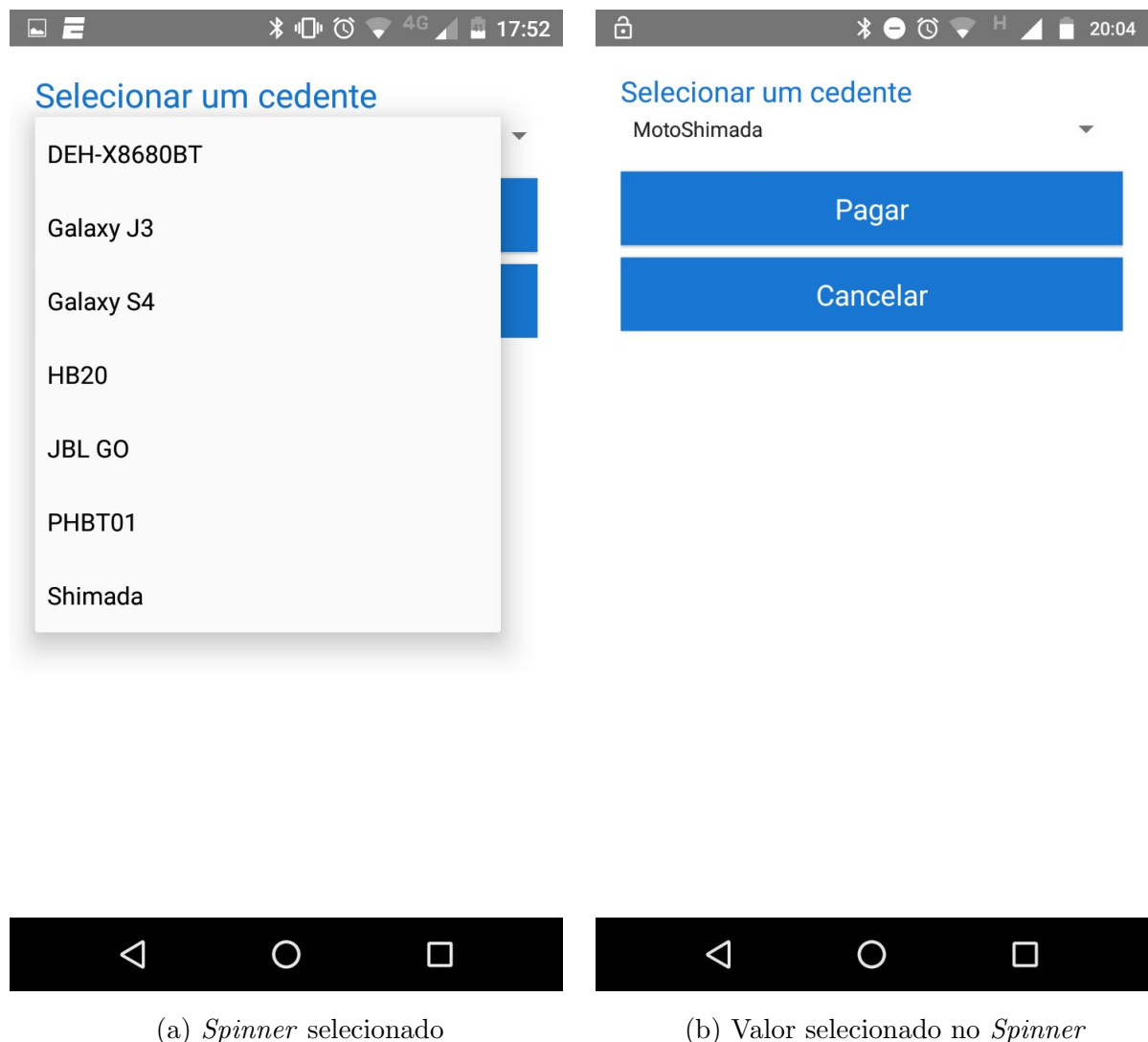


Figura 5: Demonstração do uso do *Spinner*

Na interface é necessário apenas adicionar o elemento `<Spinner>` no código, com suas respectivas propriedades, e na parte do controlador é preciso sobreescrever os métodos que definem o que deve ser feito quando o usuário seleciona um item e o que deve ser feito

se nada for selecionado.

Do lado do recebente é exibida a interface que indica que está aguardando a conexão *Bluetooth* do pagante, exibido o logo do *Bluetooth* junto com uma mensagem informando ao usuário o que o aplicativo está executando, no caso aguardando o sinal do outro *smartphone*, importante oferecer um *feedback* para o usuário para ele entender que a aplicação não está travada ou que parou de funcionar. Esta interface está disponível na imagem 6.

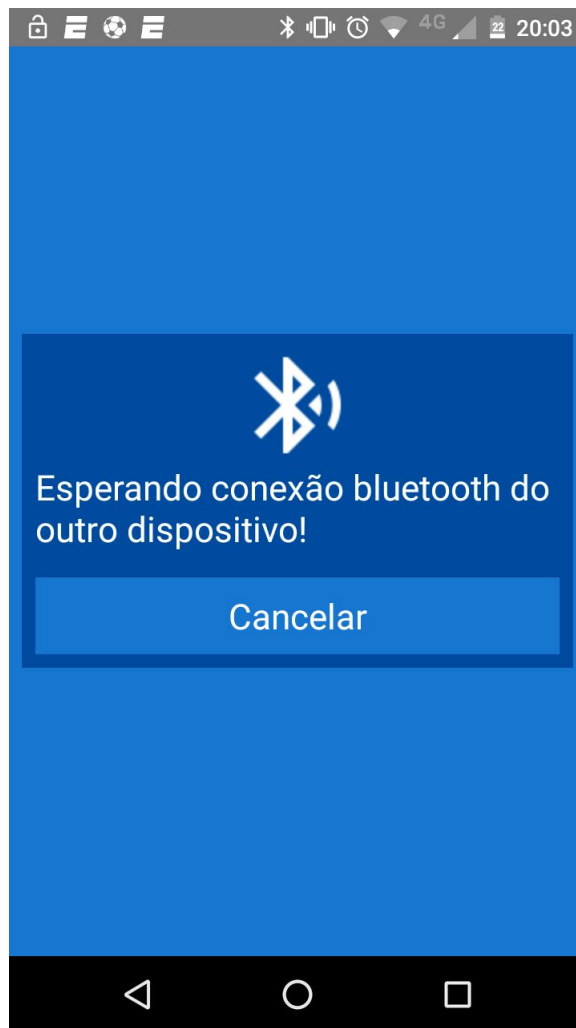
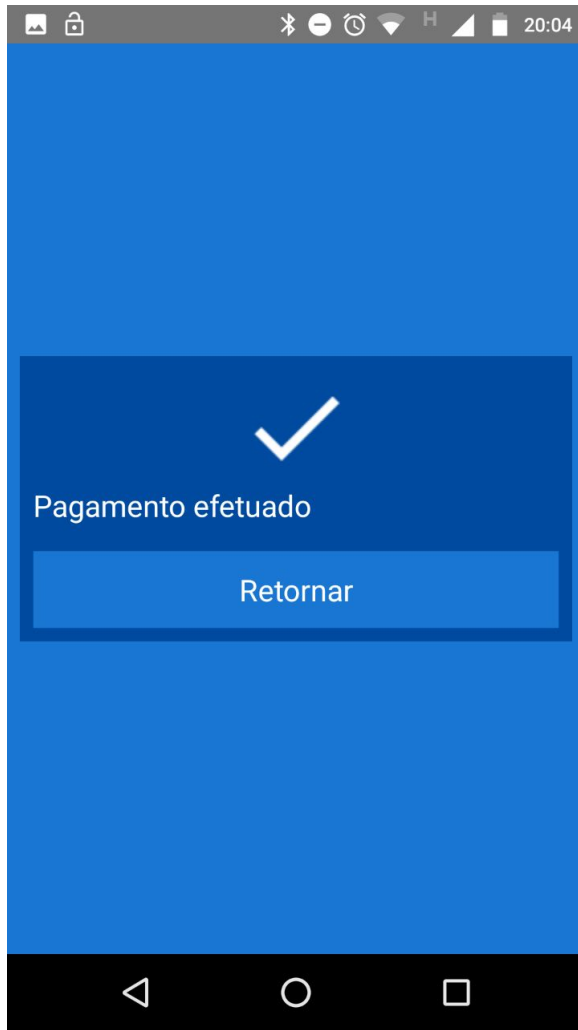
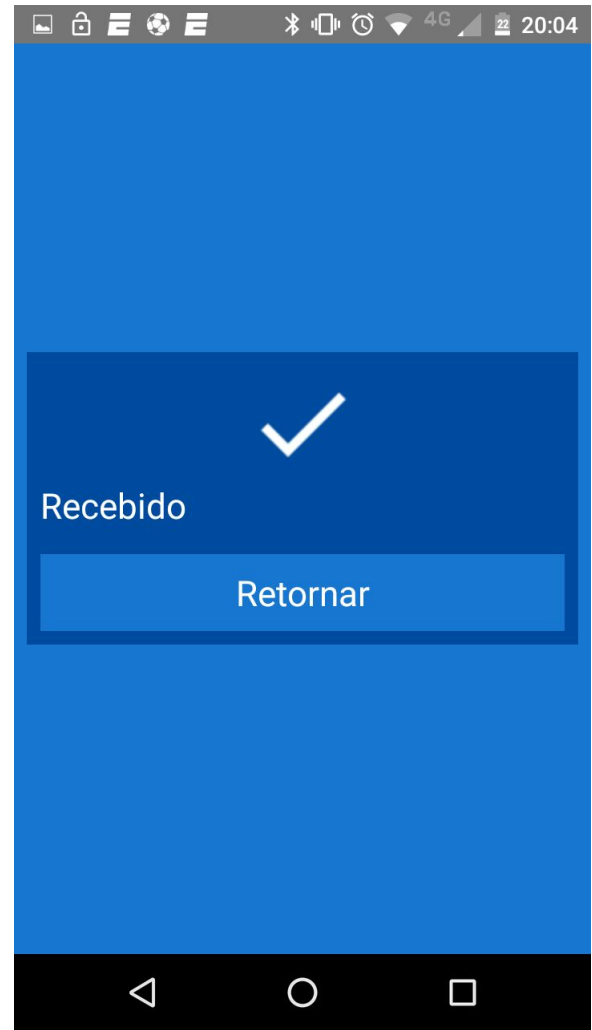


Figura 6: Interface do recebente aguardando contato por *Bluetooth*

A interface de confirmação do pagamento que existe nos dois lados do pagamento é a mesma, alterando apenas a mensagem que é exibida para o usuário, do lado do pagante é carregada mensagem "**Pagamento efetuado**" (imagem 7a), enquanto que no recebente é exibida a mensagem **Recebido** (imagem 7b). renderizado também um ícone de "finalização" (*done* no repositório de ícones do *Android* [8]), os dois componentes mais um botão para retornar ao menu inicial são exibidos dentro de um *RelativeLayout* que assume o formato de uma caixa



(a) Confirmação do lado do pagante



(b) Confirmação do lado do recebente

Figura 7: Interfaces de confirmação de pagamento

A outra ação desenvolvida foi a de carregamento, envolve duas interfaces (imagem 8, uma principal onde são consultados os valores disponíveis, e é possível selecionar o quanto será carregado no aplicativo 8a, além da tela de confirmação da ação. Na interface de carregamento foi usado uma propriedade diferente no campo *EditText*, como é um campo em que será inserido m valor monetário é esperado que sejam inseridos apenas nmeros no campo, para forçar que apenas nmeros sejam exibidos para o usuário preencher no campo foi usado o valor *"number"* no atributo *InputType*, esse comportamento pode ser verificado na imagem 8b. Na interface existem também mais dois botões que não exigiram nenhuma configuração especial para atingirem o efeito desejado.



(a) Interface de carregamento

(b) Teclado apenas com opções de números

Figura 8: Interfaces de carregamento de valores

Infelizmente até o encerramento do trabalho a plataforma de serviços ainda estava apresentando erro no momento de confirmação do carregamento, por esse motivo não foi possível em tempo de execução obter as imagens de confirmação da aplicação.

Isto porque as mensagens de confirmação e erro, etc, são uma coleção de 3 interfaces que são usadas dinamicamente ao longo do aplicativo, mudando apenas o texto que é carregado dependendo de cada contexto que essas interfaces são carregadas para serem exibidas, um exemplo do uso delas pode ser visto nas imagens 7a, 7b e 6. As mensagens de erro não foram descritas em detalhes por conta da implementação que foi feita, mas como exemplo, temos a mensagem de erro exibida caso o usuário entre com a senha errada, é carregada no *template* um ícone simbolizando que é preciso atenção, e a mensagem explicando o motivo da atenção, no caso a senha que foi colocada de forma incorreta, este exemplo pode ser visto na imagem 9.



Figura 9: Alerta de erro

5.2 Desenvolvimento das interfaces do aplicativo

Finalizada a etapa de aprendizado em *Android*, dei início ao segundo objetivo do trabalho que consistia em desenhar as interfaces do aplicativo de micropagamentos operando sobre uma base de serviços funcional. O primeiro passo foi a familiarização com a plataforma, entender o fluxo de execução, os triggers, os listeners, estudar literalmente a plataforma inteira, nessa parte contei com a colaboração do **Henrique Leme**, desenvolvedor da plataforma, com o auxílio dele essa parte foi facilitada permitindo que a assimilação do conteúdo fosse mais fluída.

Passada essa parte foi definido que o desenvolvimento seria principalmente no diretório *res*, na estrutura de um projeto *Android*, nesse diretório ficam todos os arquivos relacionados a parte visual do aplicativo, as interfaces, os ícones, as imagens, definições de estilos, *Strings*, etc. Também foi preciso em alguns pontos fazer modificações no *MainActivity*, já que algumas interfaces precisaram ser mapeadas e algumas ações foram reescritas para

atender o novo formato no *layout*.

A segunda mudança foi o uso de um arquivo *XML* para armazenar todas as *Strings* usadas no aplicativo. As vantagens dessa abordagem, ao contrário de escrever manualmente todas elas diretamente, é a flexibilidade e a reusabilidade.

Flexibilidade pois tendo o arquivo como base é possível traduzir ele para outros idiomas, como o id das *strings* permanece o mesmo, as mesmas já estão referenciadas no código e vão exibir o valor definido. Isto é til para se localizar o aplicativo para outros países que usam uma linguagem diferente da que se foi desenvolvido o aplicativo inicialmente.

Reusabilidade porque elas são apenas referenciadas no código, qualquer mudança que for feita em um dos valores é replicada por todo o aplicativo poupando o trabalho de precisar reescrever todas as ocorrências e correr o risco de esquecer alguma, além da possibilidade de usar uma mesma palavra em diversos elementos do aplicativo sem a necessidade de escrever o mesmo repetidas vezes.

6 Considerações Finais

No início do projeto foram definidos dois objetivos para este trabalho, aprender o desenvolvimento para *Android* e aplicar o conhecimento adquirido para produzir as interfaces de um aplicativo. Sendo o primeiro pré-requisito obrigatório para a execução do segundo, é mais prudente discorrer sobre ele de início.

O desenvolvimento para *Android* conta com uma boa base de conhecimento e materiais no próprio site tanto de tópicos básicos, como por exemplo um guia de configuração do ambiente, até mais avançados, como a documentação de cada componente da plataforma, que pode ser considerado suficiente para qualquer um que tenha interesse em aprender a desenvolver aplicativos para *Android* consiga fazer o mesmo. Bem verdade que em alguns assuntos a documentação é um pouco confusa ou peca pela falta de exemplos mais concretos de implementação, mas, como dito na seção 3, fóruns e sites na internet existem em grandes quantidades com outras pessoas com dúvidas similares ou aproximadamente iguais, e a maioria com soluções para tais empecilhos. Um problema recorrente nisso é talvez a falta de uma solução única, para um mesmo problema podem existir várias soluções diferentes, o que exige muita análise e testes para descobrir quais, ou qual, é a mais adequada para o problema encontrado.

Isso era comum de se encontrar tanto para problemas de interfaces, como por exemplo como fazer uma imagem caber dentro de um botão, como para problemas de *back-end*, como por exemplo qual a melhor abordagem para implementar *Fragments* em elementos do *layout*.

Apesar disso considero que a plataforma apresenta uma dificuldade média de aprendizado, alguns elementos são de fácil assimilação ao passo que alguns exigem um pouco mais de estudo e pesquisa para se entender o que precisa ser feito para atingir o resultado esperado. As interfaces são mais fáceis de compreender como se desenvolver do que como programar os controles que gerenciam o funcionamento da aplicação.

Considero que é possível sim aprender sozinho o suficiente para criar um aplicativo simples usando as informações que se encontra nas documentações oficiais e nas literaturas sobre o assunto, mas para projetos de larga escala acredito ser difícil de conseguir o resultado desejado apenas com esses recursos.

Quanto ao segundo objetivo, mantenho o que foi dissertado na sub-seção 5.2 em relação ao desenvolvimento das mesmas, uma das premissas era que as interfaces do aplicativo fossem de fácil usabilidade, com isso em foco, considerei mais prudente não precisar fazer nada que fosse apenas visualmente mais atraente mas de difícil assimilação e foquei em pensar na solução que fosse de mais fácil compreensão qual ação deveria ser tomada em cada uma das interfaces. claro que sem a primeira parte do aprendizado esta etapa teria possuído uma dificuldade extremamente maior de ser realizada, já que ao mesmo tempo seria necessário entender o funcionamento e desenvolver os componentes nas interfaces.

Uma futura possibilidade que poderia ser considerada como uma extensão desse trabalho seria tentar adaptar o aplicativo usando a linguagem *Kotlin*. Recentemente os responsáveis pelo *Android* anunciaram que a plataforma passaria a oferecer suporte para a linguagem [?]. O site da *Wired* publicou um artigo em que diz que a linguagem é a nova tendência no Vale do Silício e que passaremos a ver cada vez mais aplicativos desenvolvidos nela [13].

Para este trabalho em específico o *Kotlin* entrou mais como uma curiosidade, já que o foco dele era desenvolvimento em *Android* puro, porém foi algo que despertou grande curiosidade em explorar essa nova linguagem e medir a diferença do tempo de aprendizagem e implementação de um aplicativo usando cada uma das linguagens.

Referências Bibliográficas

- [1] Investopedia. **Micropayments Definition.** Disponível em: <https://www.investopedia.com/terms/m/micropayment.asp/>. Acesso em 27 de ago.2017.
- [2] Paula L. HERNANDEZ-VERME and Ruy A. Valdes .BENAVIDES. **Virtual currencies, micropayments and the payments systems: a challenge to fiat money and monetary policy?** Disponível em: <http://eujournal.org/index.php/esj/article/download/1264/1273>. Acesso em 9 de ago.2017.
- [3] Stacy KAUFMAN, Abhinav RAMANI, Dave LUCIANO, Long ZOU, and James FOSCO. **Micropayments: A Viable Business Model?** Disponível em: <http://cs.stanford.edu/people/eroberts/cs201/projects/2010-11/MicropaymentsAndTheNet/>. Acesso em 10 de ago.2017.
- [4] Android Developers. **Getting Started.** Disponível em: <https://developer.android.com/training/index.html>. Acesso em 21 de ago.2017.
- [5] Google. Building Your First App. Disponível em: <https://developer.android.com/training/basics/firstapp/index.html>. Primeiro acesso em 21 de ago.2017.
- [6] Kyle MEW. ***Android 5 Programming by Example.*** Packt Publishing Ltd., 2015.
- [7] Google. Material Design. Disponível em: <https://material.io/>. Primeiro acesso em 10 de set.2017.
- [8] Google. Material Icons - Material Design. Disponível em: <https://material.io/icons/>. Primeiro acesso em 12 de set.2017.
- [9] Google. Icons - Style - Material Design. Disponível em: <https://material.io/guidelines/style/icons.html#icons-system-icons>. Primeiro acesso em 12 de set.2017.
- [10] Google. Colors - Style - Material Design. Disponível em: <https://material.io/guidelines/style/color.html>. Primeiro acesso em 15 de set.2017.
- [11] Google. Color Tool - Material Design. Disponível em: <https://material.io/color/>. Primeiro acesso em 15 de set.2017.

- [12] Google. Layouts — Android Developers. Disponível em: <https://developer.android.com/guide/topics/ui/declaring-layout.html>. Primeiro acesso em 21 de ago.2017.
- [13] Klint FINLEY. Kotlin: the Upstart Coding Language Conquering Silicon Valley. Disponível em: <https://www.wired.com/story/kotlin-the-upstart-coding-language-conquering-silicon-valley/>. Primeiro acesso em 4 de nov.2017.

Apêndice A Fluxo do aplicativo

Aqui será descrito o funcionamento do aplicativo



Figura A.1: sample