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:_____

Agradecimentos

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 solu'ções 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 disponvel 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 gerencie as transaes 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

Sumário

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 e 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 currncia que substitua o dinheiro padrão em pequenas comunidades.

Imagine por exemplo o ambiente de um condomínio privado que oferça 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, micropaymentes 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 sitema de uma maneira que permita que ela seja escalável é um requisito necessário para poder lidar com um pontencial aumento do nmero 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 a a partir de que ponto é necessário saber a identididade 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 nennhuma 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 deprivação 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 anncios 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 técnincas 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 importncia no projeto.

A página ***Material Icons*** [8] é uma espécie de reposítorio 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 repósitorio 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 feito 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 configurãções 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* dinmicas 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 estrutra da plataforma de serviços e revisão das interfaces, o nmero 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 *Relative Layout* 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 usário (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 trs tipos de componentes: *EditText*, *Button* e *ImageView*, os trs componentes não apresentaram grande dificuldade em assimilar o comportamento e o que deveria ser feito para aplicar na interface.

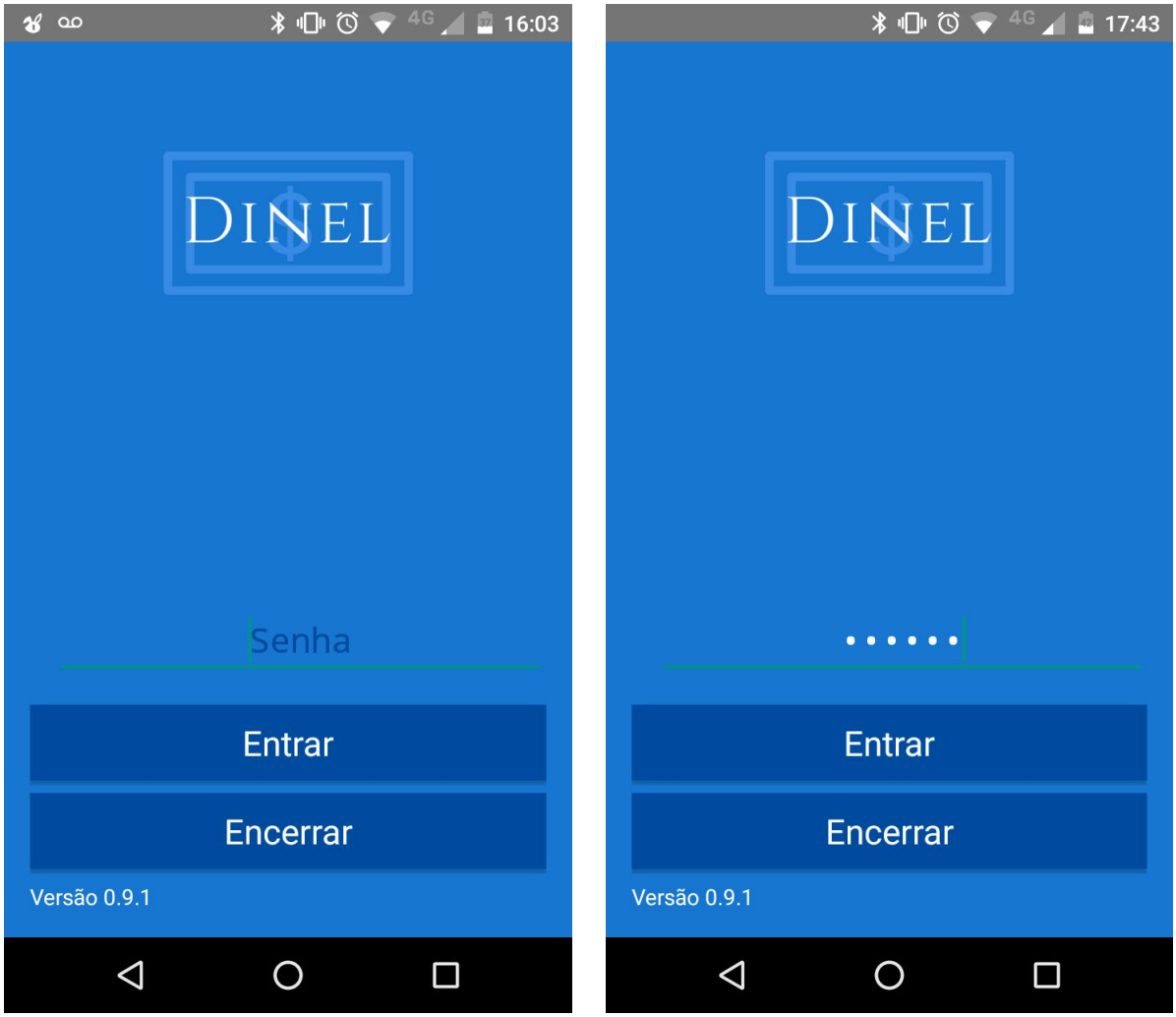
Figura 2: Interface de cadastro de usuário'

O componente *EditText*, simplificadamente, permite que o usuário insira caractéres 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 caractéres 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

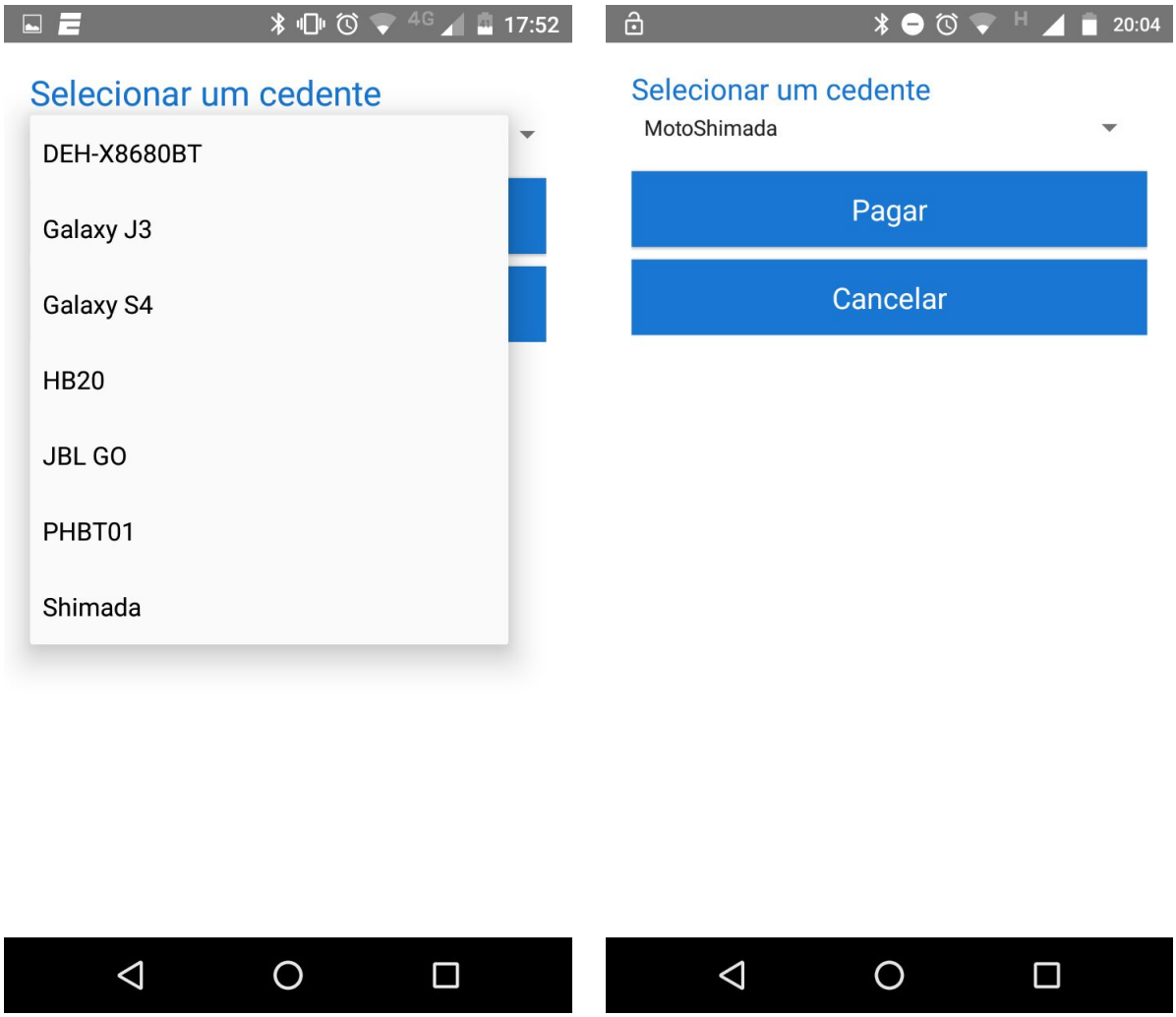
Figura 3: Demonstração do uso de *hint* e preenchimento de campo com atributo *textPassword*

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 nesse 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)



(a) *Spinner* selecionado (b) Valor selecionado no *Spinner*

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 exibida a interface que indica que est aguardando a conexo *Bluetooth* do pagante, exibido o logo do *Bluetooth* junto com uma mensagem informando ao usurio o que o aplicativo est executando, no caso aguardando o sinal do outro *smartphone*, importante oferecer um *feedback* para o usurio para ele entender que a aplicao no est travada ou que parou de funcionar. Esta interface est disponvel na imagem 6.

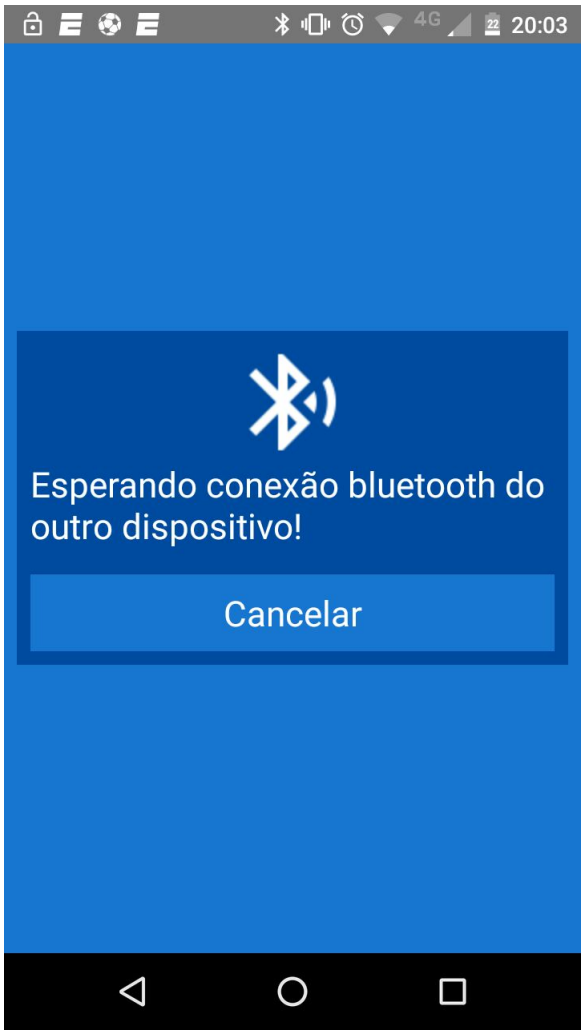
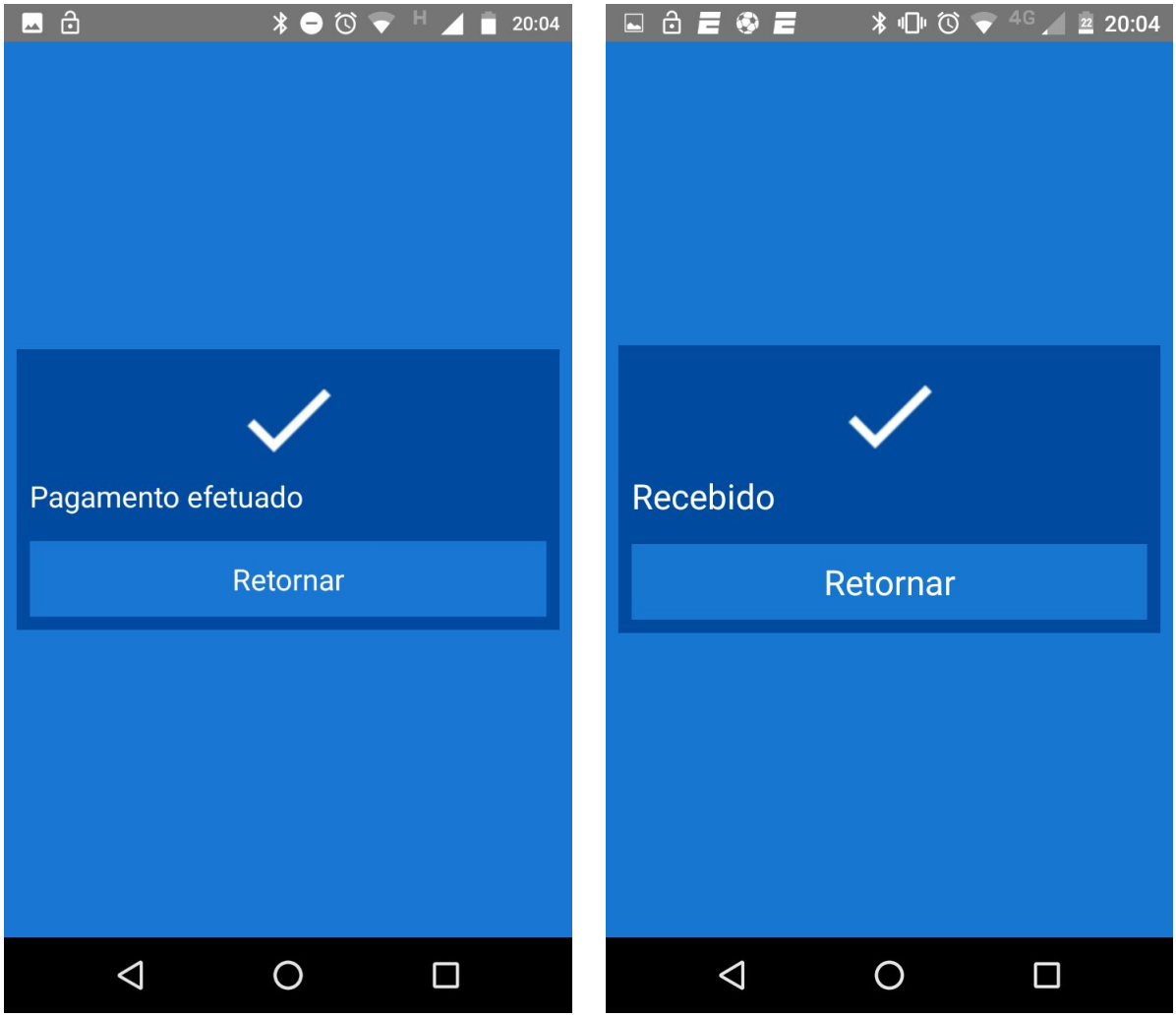


Figura 6: Interface do recebente aguardando contato por *Bluetooth*

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

Um dos objetivos específicos para esse projeto era aprender desenvolvimento para a plataforma *Android*. Até o início do projeto eu nunca havia tido contato com o desenvolvimento para *Android*, então o primeiro passo foi ir atrás de material sobre o mesmo, o ponto inicial para isso assumi que seria o site oficial da plataforma, isso por experincias passadas de estudo onde a página de alguma linguagem, *framework*, etc, em grande partes dos casos possuia uma boa documentação sobre a mesma. A página da plataforma do *Android* felizmente seguia essa linha, e na mesma encontrei as informações necessárias para dar início no aprendizado.

Após a configuração do ambiente, instalando *Java* e a *IDE* apenas, segui o tutorial [5] que explica como criar um aplicativo básico (imagem 8) para testar as configurações locais. O resultado é um aplicativo com duas *Activities* (interfaces), a primeira (imagem 8a) possui uma *EditText* para escrever uma mensagem e um botão que chama a próxima *Activity*, a segunda (imagem 8b) *Activity* apenas exibe a mensagem escrita.

(a) Interface do aplicativo que recebe um texto e envia para a próxima interface

(b) Interface que exibe a mensagem da *Activity* anterior

Figura 8: Interfaces do aplicativo do tutorial

Apesar de ser um exemplo um tanto trivial, ele engloba alguns aspectos importantes do *Android*:

- Criação de interface
- Criação de componentes para a interface
- Uso de *intents* para trafego de dados entre as interfaces
- Procurar, extrair e exibir textos nas interfaces a partir do ID

Após o contato inicial com a plataforma e o desenvolvimento do aplicativo acima, fui em uma busca de mais material e comecei a ler o livro **"*Android* 5 Programming by Example"** [6]. Com o conhecimento sobre micropagamentos, o livro citado e os guias, todos indicados na seção 3, iniciei o desenvolvimento de um aplicativo que simulasse um micropagamento, o aplicativo era totalmente *offline* e apenas fazia um *mock* de serviço de transações.

Para o aplicativo tentei imaginar qual o mínimo de interfaces que precisaria para simular o uso do aplicativo por um usuário que queria executar uma transação, e cheguei as seguintes interfaces:

1. Interface de acesso, com espaço para o usuário colocar o seu login, a senha, um botão de confirmação e o logo do aplicativo

2. Interface principal do aplicativo com as ações que o usuário pode fazer mapeadas em botões

3. Interface de pagamento onde o usuário preenche quem é o destinatário, o valor, a data e confirma

4. Interface de confirmação da transação

Essas interfaces foram decididas como as principais para desenvolver e testar o conceito de micropagamento em uma escala absurdamente menor, em um ambiente controlado apenas para validar idéias e testar conceitos novos no desenvolvimento de interfaces para *Android*.

A interface de *login* (item 1) foi a primeira a ser desenvolvida, é a interface mais simples do aplicativo. Os pontos novos do aprendizado nessa interface foram inserir uma imagem na tela e colocar um campo texto do tipo senha, em que os caractéres aparecem como asterisco a medida que são digitados. A interface por fim está ilustrada na imagem 9a.

A interface principal (item 2) do aplicatvo foi a segunda, o ponto novo aplicado nessa interface foi a inclusão de ícones dentro de um *Button*. A interface está ilustrada na imagem 9b.

(a) Interface de *login*

(b) Interface principal do aplicativo com as ações permitidas ao usuário mapeadas como botões

Figura 9: Interfaces do aplicativo *mock*

A interface de pagamentos (item 3) é a que considerei a mais interessante de se fazer, ela possui dois elementos visuais que demandaram grande esforço para entender o funcionamento e a aplicação, um deles é um calendário para selecionar datas e o outro é uma barra deslizante que de acordo com a posição define qual o valor da transação.

Dentre os dois o que gerou maior dificuldade para ser feito foi o calendário, o *Android* fornece um DatePicker padrão, porém ele não se adequava a implementação desejada, o que busquei fazer para o funcionamento dele era que ao clicar em um campo de entrada de texto comum, abrisse um calendário para que o usuário escolhesse a data e ao finalizar fechasse o calendáro novamente.

Para atingir esse efeito foi necessário criar um Fragment que implementasse o DatePicker padrão do *Android*, foi sobreescrito o método de onCreate para que ao criar o Fragment retornasse um DatePicker com a data atual. A implementação do Fragment pode ser vista na *listing* 1.

```
public class DatePickerFragment extends DialogFragment
    implements DatePickerDialog.OnDateSetListener{
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState)
            {
                final Calendar calendar = Calendar.getInstance()
                    ;
                int year = calendar.get(Calendar.YEAR);
                int month = calendar.get(Calendar.MONTH);
                int day = calendar.get(Calendar.DAY_OF_MONTH);
                return new DatePickerDialog(getActivity(), (
                    TransactionActivity)getActivity(), year,month
                    ,day);
        }
```

Listing 1: *Fragment* do calendário

No código de controle da *Activity* foi preciso adicionar um *listener* no campo texto que fica aguardando um 'clique', quando é clicado, chama uma função que instancia o *Fragment* e exibe na interface, ainda nesse controle foi criada uma função que atribui ao campo do calendário a data selecionada, essa função é sobreescrita de uma definida na mesma classe do *Fragment* do *DatePicker* (ver *listing* 1).

Um exemplo de como o *Fragment* é carregado e renderizado na tela pode ser visto na imagem 10. A imagem 10a mostra a interface ao ser carregada, nesse momento o *listener* está aguardando um 'clique' na área de texto ao lado de **Escolha uma data** para carregar o calendrio. A imagem 10b mostra o *Fragment* carregado na interface.

(a) *DatePicker* não selecionado

(b) *Fragment* do *DatePicker* carregado

Figura 10: Exemplo de uso do *Fragment* do *DatePicker*

Para o outro componente, a barra deslizante chamada de *SeekBar*, a implementação exigiu menos codificação, não foi necessário criar classes adicionais como feito no *DatePicker* (*listing* 1), já que o componente padrão atendia as necessidades da interface, o que precisou ser codificado foi implementar um *listener* na barra para detectar movimentos nela e sobrepor 3 métodos padrões que definem as ações de início, fim e mudança de status. A de ínicio não faz nada, apenas é instanciada para definir a barra, a de mudança de status pega o valor de onde a barra parou e atribui à uma varíavel e a final é encarregada de atualizar o campo texto com o valor atual selecionado. A implementação pode ser vista na *listing* 2.

```
SeekBar seekbar = (SeekBar) findViewById(R.id.seekBar);
seekbar.setOnSeekBarChangeListener(new SeekBar.
    OnSeekBarChangeListener(){
        int actual_value = 0;
        @Override
        public void onProgressChanged(SeekBar seekBar, int
            new_value, boolean b) {
                actual_value = new_value;
        }
        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {
        }
        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
                TextView transferValue = (TextView) findViewById
                    (R.id.rvalueTransfer);
                transferValue.setText(Integer.toString(
                    actual_value);
        }
});
```

Listing 2: *SeekBar* do valor

A ltima interface, é a de confirmação do pagamento (item 11b), é uma tela simples que apenas carrega as informações inseridas na interface anterior (ver imagem 10a) para confirmação e execução, lembrando que execução no contexto atual é uma simulação, do pagamento. Essa interface não teve nenhum desenvolvimento expressivo já que ela usa elementos, nesse ponto do aprendizado, básicos, como campos textos carregando informações vindos da leitura da *Intent* que transportou os dados de uma *Activity* para outra.

(a) Interface com todas as informações preenchidas

(b) Interface de confirmação com os dados preenchidos na 11a

Figura 11: Confirmação dos dados de uma transação

Após o desenvolvimento desses dois aplicativos considerei ter adquirido um conhecimento suficiente para prosseguir e dar incio ao desenvolvimento da interface do aplicativo alvo deste trabalho. É claro que passei por um número pequeno de componentes que o *Android* possui, existem diversos outros *Fragments* que executam diferentes tarefas de diferentes formas visuais, alm de muitos outros componentes que até o momento da finalização desse aplicativo não vi como necessários para este desenvolvimento em específico.

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 contedo 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*.

Uma mudança substancial no desenvolvimento dessas interfaces foi o uso do *Relative Layout* ao invés do *Constraint Layout* que foi usado nos dois aplicativos descritos na sub-seção 5.1. Durante a execução dos dois aplicativos anteriores esse detalhe passou desapercebido durante o desenvolvimento, o que ocasionou um certo volume de trabalho para estabelecer a posição de cada componente na tela. A vantagem no uso do *Relative Layout* 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 *Relative Layout* declaramos a área, o primeiro botão alinhado a direita da área, e cada um dos botões subsequentes um abaixo do outro, o código para esta ilustrado na *listing* 3.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:paddingLeft="16dp"
        android:paddingRight="16dp">
        <Button
                android:id = "@+id/box1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:text="button1" />
        <Button
                android:id = "@+id/box2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_below="@id/box1"
                android:layout_alignParentLeft="true"
                android:text="button2" />
        <Button
```

```
        android:id = "@+id/box3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/box2"
        android:layout_alignParentLeft="true"
        android:text="button3" />
</RelativeLayout>
```

Listing 3: *Exemplo de uso de um RelativeLayout*

A exceção do aprendizado do *Relative Layout* não houve grandes dificuldades no desenvolvimento das interfaces visto que a maior parte deles foi escrita usando botões e campos textos para carregar as informações. De componentes e técnincas não vistos na etapa de arendizado foram utilizados o *Spinner*, e a declaração de todas as *Strings* em um arquivo *XML* à parte.

O *Spinner* 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. No aplicativo essa lista é obtida de outros dispostivos que foram pareados por bluetooth com o do usuário. 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.

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 ocorrncias 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 dvidas similares ou aproximadamente iguais, e a maoria com soluções para tais empecilhos. Um problema recorrente nisso é talvez a falta de uma solução nica, para um mesmo problema podem existir n 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 progrramar 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 documentaçoes 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 usablidade, com isso em foco, considerei mais prudente não precisar fazer nada que fosse apenas visualmente mais atraente mas de díficil asssimição e foquei em pensar na solução que fosse de mais fácil compreensão qual ação deveria ser tomada em cada umas das interfaces. claro que sem a primeira parte do aprendizado esta etapa teria possuído uma dificuldade exrememamente 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 tendncia 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, micropaymentes 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 Desgin. Disponível em: https://material.io/. Primeiro acesso em 10 de set.2017.

[8] Google. Material Icons - Material Desgin. Disponível em: https://material.io/icons/. Primeiro acesso em 12 de set.2017.

[9] Google. Icons - Style - Material Desgin. Disponível em: https://material.io/guidelines/style/icons.html#icons-system-icons. Primeiro acesso em 12 de set.2017.

[10] Google. Colors - Style - Material Desgin. Disponível em: https://material.io/guidelines/style/color.html. Primeiro acesso em 15 de set.2017.

[11] Google. Color Tool - Material Desgin. 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