

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	4
2	Interfaces do aplicativo do tutorial	8
3	Interfaces do aplicativo <i>mock</i>	10
4	Exemplo de uso do <i>Fragment</i> do <i>DatePicker</i>	12
5	Confirmação dos dados de uma transação	14
A.1	sample	20

Listings

1	<i>Fragment</i> do calendário	11
2	<i>SeekBar</i> do valor	13
3	<i>Exemplo de uso de um RelativeLayout</i>	15

Sumário

1	Introdução	1
2	Objetivos	2
2.1	Objetivo Geral	2
2.2	Objetivos Específicos	2
3	Revisão Bibliográfica	3
4	Metodologia	5
5	Resultados	7
5.1	Aprendizado em desenvolvimento para <i>Android</i>	7
5.2	Desenvolvimento das interfaces do aplicativo	14
6	Considerações Finais	17
	Referências Bibliográficas	19
	Apêndice A Fluxo do aplicativo	20

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.

2 Objetivos

2.1 Objetivo Geral

O objetivo do trabalho foi desenvolver um aplicativo *Android* que conseguisse efetuar transações monetárias entre duas pessoas usando o celular, com uma interface simples de usar.

2.2 Objetivos Específicos

Os objetivos específicos do trabalho foram:

- Aprendizagem em programação para *Android*
- Desenvolver a *UI* do aplicativo com base em uma plataforma de serviços e infraestrutura existente
- Aprendizagem do conceito de micropagamentos

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.
- A segunda fonte 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.

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* [3]. Nela encontrei uma série de lições básicas para quem está começando a desenvolver para *An-*

droid. 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 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 [4] utilizado nesse momento foi "***Android 5 Programming by Example***", o livro [4] 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 [4] 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 [4] 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 [5] 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*. Dentre as ferramentas e guias disponíveis os que se destacam pelo seu uso e importância no projeto foram as seguintes:

- ***Material Icons*** [6] - 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*** [7]

- ***Material Colors*** [8] - Uma ferramenta [9] 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*** [8] onde é explicado as boas práticas e guias para selecionar as cores dos componentes da interface do aplicativo.

4 Metodologia

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*.

As linguagens utilizadas no projeto foram:

- ***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*

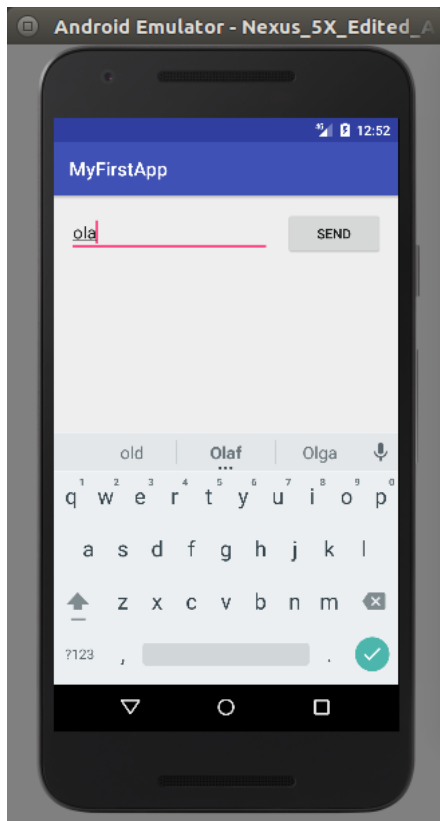
5 Resultados

Para melhor entendimento dos resultados, os mesmos vão ser divididos em duas seções. A primeira com os resultados do aprendizado em desenvolvimento para *Android*, onde vão ser apresentados as dificuldades iniciais, dúvidas que surgiram e o aplicativo *mock* desenvolvido que simulava uma transação de micropagamento. Na segunda seção será descrito o desenvolvimento em si do aplicativo final, usando já a plataforma de serviços.

5.1 Aprendizado em desenvolvimento para *Android*

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 experiências passadas de estudo onde a página de alguma linguagem, *framework*, etc, em grande partes dos casos possuía 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 [10] que explica como criar um aplicativo básico (imagem 2) para testar as configurações locais. O resultado é um aplicativo com duas *Activities* (interfaces), a primeira (imagem 2a) possui uma *EditText* para escrever uma mensagem e um botão que chama a próxima *Activity*, a segunda (imagem 2b) *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 2: 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 tráfego 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**” [4]. 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.

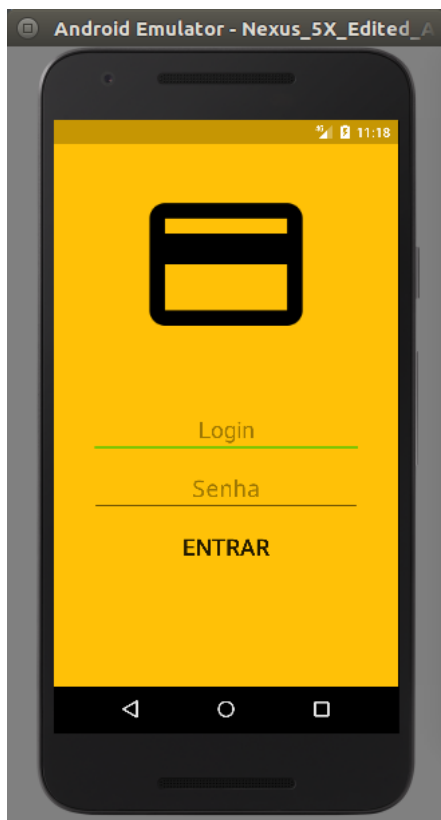
Para o aplicativo tentei imaginar qual o mínimo de interfaces que precisaria para simular o uso do aplicativo por um usuário que queria executar uma transação, e cheguei as seguintes interfaces:

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 caracteres aparecem como asterisco a medida que são digitados. A interface por fim está ilustrada na imagem 3a.

A interface principal (item 2) do aplicativo foi a segunda, o ponto novo aplicado nessa interface foi a inclusão de ícones dentro de um *Button*. A interface está ilustrada na imagem 3b.



(a) Interface de *login*



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

Figura 3: Interfaces do aplicativo *mock*

A interface de pagamentos (item 3) é a que considereirei 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 calendário 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);
    }
}

```

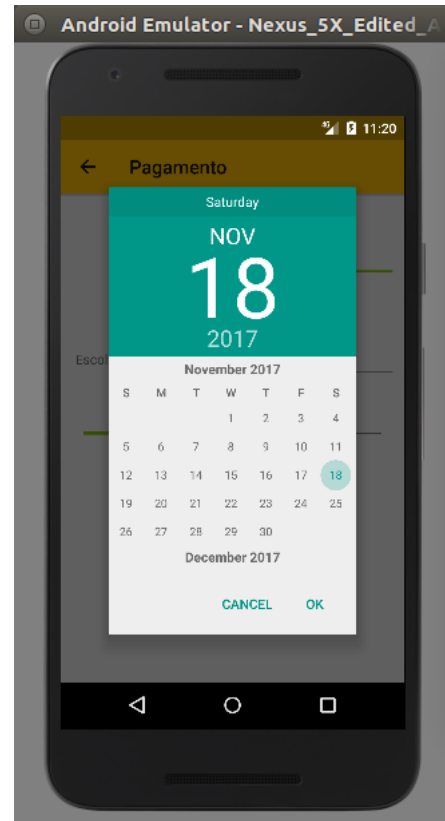
Listing 1: *Fragment* do calendário

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 4. A imagem 4a 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 calendário. A imagem 4b mostra o *Fragment* carregado na interface.



(a) *DatePicker* não selecionado



(b) *Fragment* do *DatePicker* carregado

Figura 4: 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 início 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 variável e a final é encarregada de atualizar o campo texto com o valor atual selecionado. A implementação pode ser vista na *listing 2*.

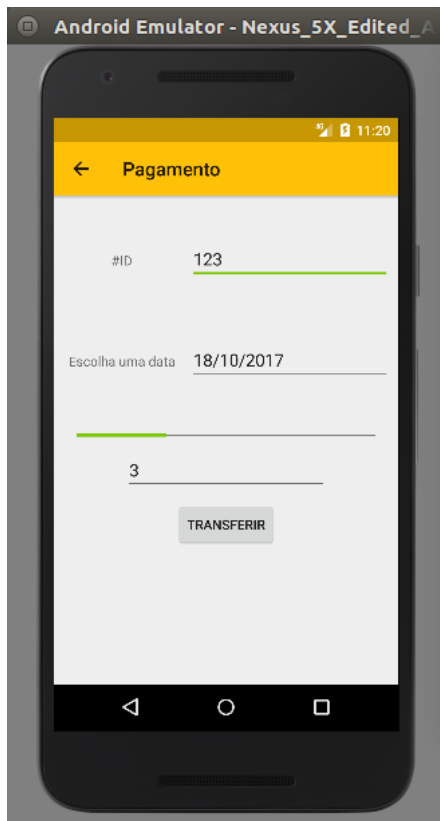
```

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 5b), é uma tela simples que apenas carrega as informações inseridas na interface anterior (ver imagem 4a) 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 5a

Figura 5: Confirmação dos dados de uma transação

Após o desenvolvimento desses dois aplicativos considerei ter adquirido um conhecimento suficiente para prosseguir e dar início 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, além 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 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*.

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 despercebido 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 técnicas não vistos na etapa de aprendizado 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 dispositivos 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 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 dvidas similares ou aproximadamente iguais, e a maioria 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 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 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 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 [11].

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] Android Developers. **Getting Started**. Disponível em: <https://developer.android.com/training/index.html>. Acesso em 21 de ago.2017.
- [4] Kyle MEW. ***Android 5 Programming by Example***. Packt Publishing Ltd., 2015.
- [5] Google. Material Design. Disponível em: <https://material.io/>. Primeiro acesso em 10 de set.2017.
- [6] Google. Material Icons - Material Design. Disponível em: <https://material.io/icons/>. Primeiro acesso em 12 de set.2017.
- [7] Google. Icons - Style - Material Design. Disponível em: <https://material.io/guidelines/style/icons.html#icons-system-icons>. Primeiro acesso em 12 de set.2017.
- [8] Google. Colors - Style - Material Design. Disponível em: <https://material.io/guidelines/style/color.html>. Primeiro acesso em 15 de set.2017.
- [9] Google. Color Tool - Material Design. Disponível em: <https://material.io/color/>. Primeiro acesso em 15 de set.2017.
- [10] Google. Building Your First App. Disponível em: <https://developer.android.com/training/basics/firstapp/index.html>. Primeiro acesso em 21 de ago.2017.
- [11] 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