

Sumário

1) Visão Geral	3
1.1) Descrição dos componentes	3
1.1.1) Smart POS	3
1.1.2) Vero Pagamentos	4
1.1.3) Vero Store	4
1.2) Requisitos	5
1.3) Ativação do Terminal.....	6
2) Atualização do Vero Pagamentos e Vero Store	8
2.1) Configurando o ADB	8
2.2) Atualizando a Vero Store	8
2.3) Atualizando o Vero Pagamentos	8
3) Instalando seu aplicativo	10
4) Biblioteca Auxiliar Smart POS	11
4.1) Impressora	11
4.2) Comunicação com a Vero Store	12
4.3) Internet Móvel	14
5) Solicitações de Pagamento Externo	16
5.1) Transações Suportadas	16
5.2) Ação da Intent (Action).....	17
5.3) Código de Requisição da Intent (Request code).....	17
5.4) Extras da Intent	18
5.5) Extras da Intent de Retorno	20
5.6) Descrição e Exemplos das Transações	22
5.6.1) Sem valor e sem transação	22
5.6.2) Com valor definido e sem transação	22
1.6.1) Crédito	23
5.6.3) Crédito a Vista	23
5.6.4) Crédito Parcelado pelo Lojista	24
5.6.5) Crédito Parcelado pelo Emissor	24
5.6.6) Crédito Vero Repay	25
5.6.7) Débito	25
5.6.8) Débito a vista	26
5.6.9) Banricompras Débito a vista	26
5.6.10) Banricompras Débito Pré-datado	27

5.6.11)	Banricompras Débito Parcelado	27
5.6.12)	Banricompras Crédito 1 Minuto	28
5.6.13)	Voucher	29
5.6.14)	Cancelamento (estorno)	29
5.6.15)	Vero Wallet	30
5.6.16)	Vero X (Pix)	30
5.6.17)	Crédito digitado	31
5.6.18)	Veropay	31
5.6.19)	Exemplo para captura do retorno da chamada	31
6)	Leitura de Cartão MIFARE	33
7)	Considerações Finais	34

1) Visão Geral

Este documento se propõe a ajudar a quem deseja desenvolver aplicativos para terminais Smart POS da Vero e publicá-los na loja de aplicativos da Vero, a Vero Store. Serão abordadas as questões de atualização dos aplicativos Vero, a instalação de aplicativos do desenvolvedor, o uso da biblioteca disponibilizada junto a esse documento, e a solicitação de requisitar que pagamentos sejam realizados a partir da sua aplicação para a aplicação Vero Pagamentos.

Até o atual momento, a Vero utiliza dois terminais: Gertec GPOS700 e Sunmi P2, porém outros terminais podem vir a serem usados pela Vero, e fica a cargo do desenvolvedor testar o seu aplicativo nos futuros terminais, que podem possuir versões diferentes de Android. A cada terminal novo disponibilizado, será necessário atualizar a biblioteca distribuída junto com esse documento para que funcionalidades específicas do terminal funcionem, como por exemplo a impressora.

1.1) Descrição dos componentes

Serão citados ao longo desse documento três componentes principais, são eles:

1.1.1) Smart POS

Atualmente, os terminais Smart POS da Vero são o GPOS700 (também referido como G700), fabricado pela Gertec, e o P2, fabricado pela Sunmi. Ambos serão citados ao longo do documento como Smart POS. Possui os recursos de hardware para a leitura de cartões e realização de pagamentos seguro. O Sistema Operacional (SO) do G700 é o Android 5.1.1, enquanto que do P2 é Android 7.1.1



Figura 1. Smart POS G700 (esquerda) e P2 (direita)

1.1.2) Vero Pagamentos

A aplicação de pagamentos da Vero, o **Vero Pagamentos** é o **launcher** padrão do terminal, e o responsável por receber as requisições de pagamento realizados pelos aplicativos publicados na Vero Store, ou pelo próprio usuário.

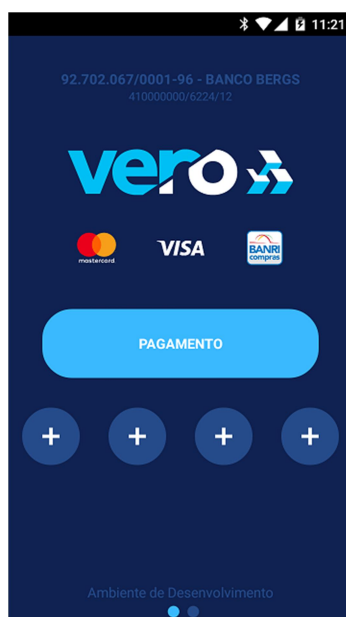


Figura 2. App Vero Pagamentos

1.1.3) Vero Store

A loja de aplicativos da Vero, a **Vero Store** é o aplicativo que disponibiliza a instalação e atualização dos aplicativos publicados na respectiva plataforma Smart POS. Realiza a atualização de componentes do sistema, incluindo atualizações de SO.

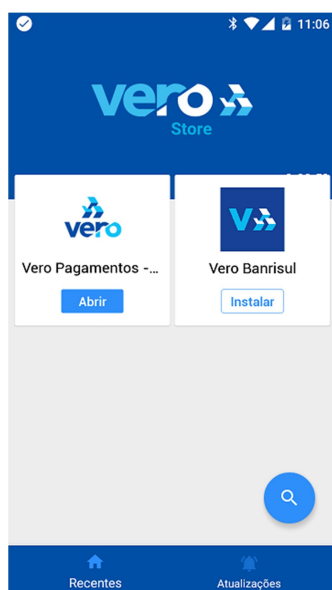


Figura 3. App Vero Store

1.2) Requisitos

Para realizar os passos descritos nesse documento, é necessário que, além de possuir um Smart POS de **desenvolvimento**, é necessário que ele esteja ativado em **ambiente de desenvolvimento**, conectá-lo ao computador via USB e possuir as ferramentas para enviar comandos ADB para o terminal.

Para verificar se o terminal está ativado em ambiente de desenvolvimento, basta procurar na tela inicial do Vero Pagamentos a frase **Ambiente de Desenvolvimento**. A localização dessa frase pode variar entre as versões. Na **figura 2** é possível localizar essa frase na parte inferior da tela. Caso não seja possível localizar essa frase, ou se o terminal não estiver inicializado (exibindo o botão **ATIVE SUA CONTA**), será necessário ativá-lo em ambiente de desenvolvimento. Para realizar esse processo, será necessário possuir a **SENHA TÉCNICA**. Esta é alterada diariamente, por tanto será necessário entrar em contato conosco para obtê-la. O processo de ativação do terminal está descrito no tópico 1.3.

O Smart POS de desenvolvimento, por utilizar o sistema operacional Android, pode ser usado para desenvolver o seu aplicativo da mesma forma que um smartphone. As opções de desenvolvedor e a depuração USB estão habilitadas por padrão.

As ferramentas para enviar comandos ADB podem ser encontradas na internet, e também estão inclusas no Android Studio, caso você o use como ferramenta de desenvolvimento de aplicativos. Também é disponibilizada uma cópia dessas ferramentas dentro do arquivo compactado que inclui esse documento, na pasta **Atualização Apps Vero Desenvolvimento/platform-tools**.

1.3) Ativação do Terminal

Como citado anteriormente, caso o seu terminal não esteja ativado em ambiente de desenvolvimento, siga as instruções desse tópico. Caso já esteja ativado em ambiente de desenvolvimento, pule para o próximo tópico que aborda a atualização das aplicações Vero no terminal.

Caso o seu terminal já esteja ativado, porém em ambiente de produção, será necessário navegar até as configurações técnicas do terminal e selecionar a opção **Inicialização**, conforme as figuras abaixo:

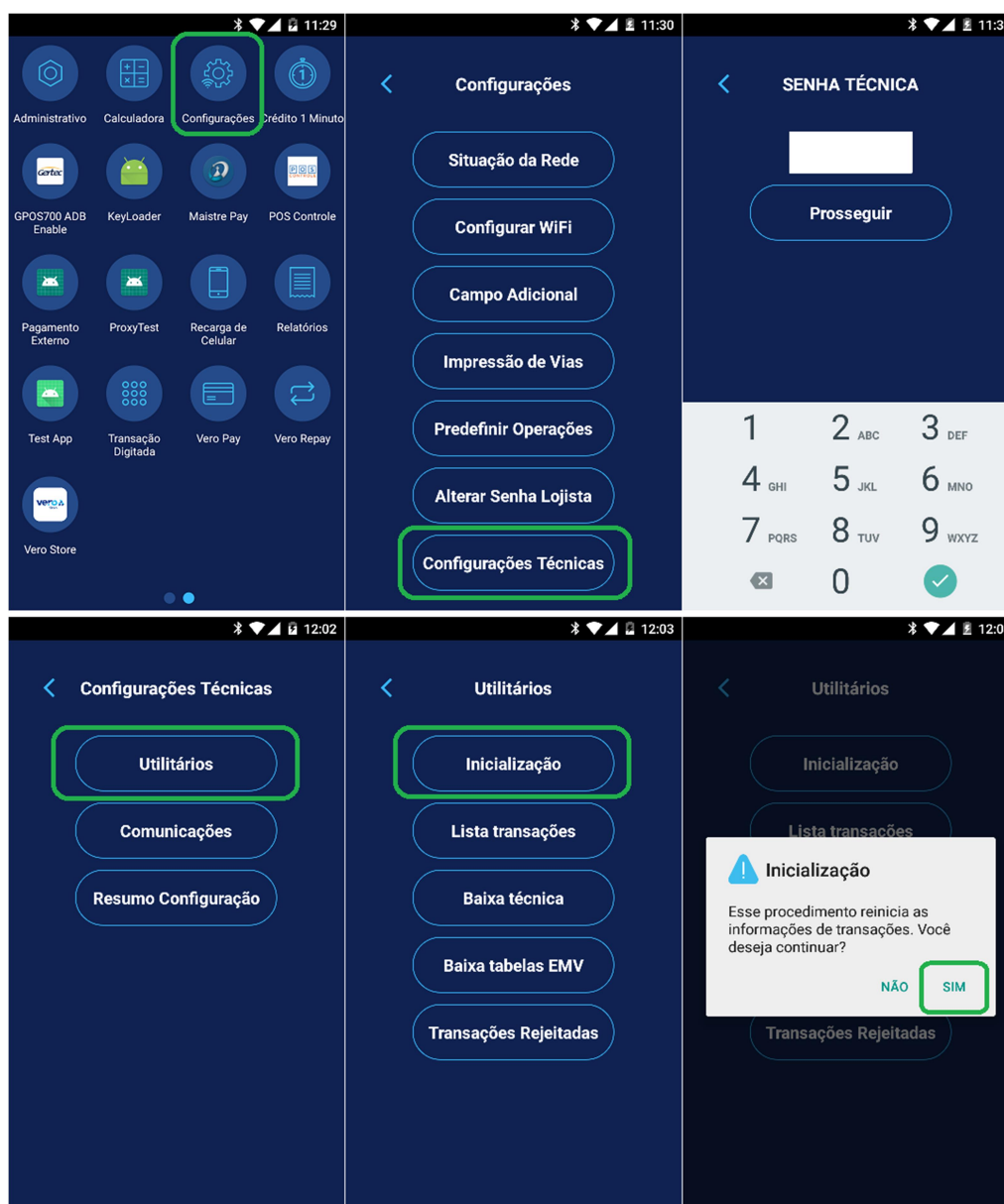


Figura 4. Inicialização de um terminal caso o mesmo já esteja inicializado

Depois, clique no botão de voltar que irá aparecer a tela inicial conforme abaixo. Com o terminal não inicializado, clique no logo da Vero diversas vezes até que seja solicitado uma senha, e prossiga conforme as figuras abaixo:

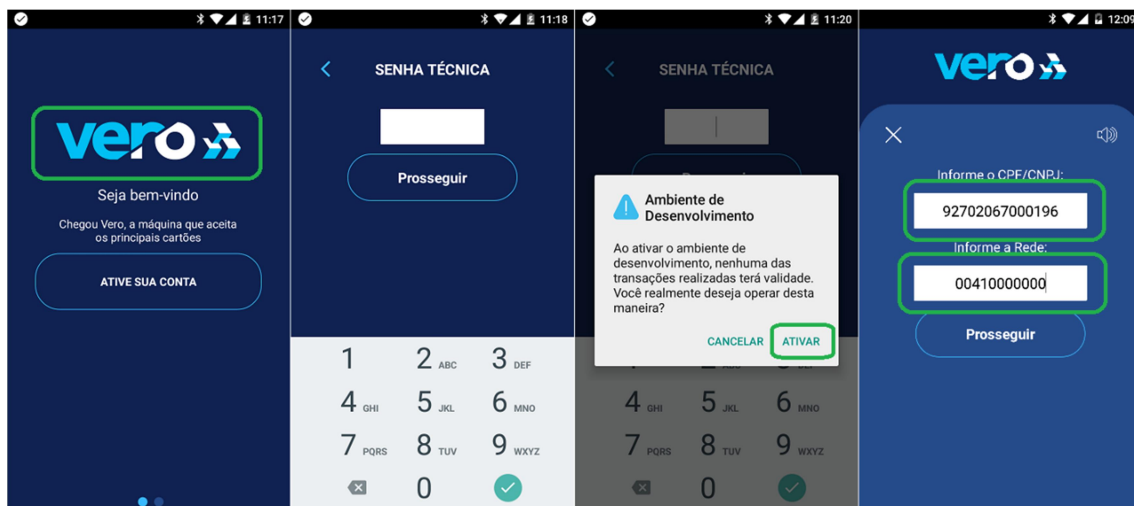


Figura 5. Inicialização do terminal

O terminal então irá realizar o download das configurações e por fim exibirá uma mensagem de **Leitor de cartões pronto para uso** e você verá na tela inicial a mensagem de **Ambiente de Desenvolvimento**.

Caso a versão do app Vero Pagamentos instalada no terminal seja antiga, pode ser que a organização das telas seja diferente. Caso deseje, pode primeiro realizar a atualização da aplicação Vero Pagamentos conforme descrito no tópico a seguir, e então retornar para realizar as instruções deste tópico.

O próximo tópico abordará a atualização das aplicações Vero.

2) Atualização do Vero Pagamentos e Vero Store

A aplicação Vero Pagamentos e Vero Store trabalham em conjunto. É possível que o Smart POS de desenvolvimento venha com versões antigas dessas duas aplicações, nas quais os recursos de instalação e atualização de aplicativos funcionavam apenas nos terminais de **produção**, devido a questões de assinatura que são diferentes entre os terminais de desenvolvimento e produção. Porém, nas versões mais recentes, a atualização também funciona nos terminais de desenvolvimento, então o primeiro passo é atualizar as duas aplicações caso a atualização automática não funcione, para que essa funcionalidade passe a funcionar e o seu terminal passe a receber as atualizações mais recentes dos aplicativos Vero e de SO do terminal.

Primeiro vamos descrever como usar as ferramentas de envio de comandos ADB. Caso você já saiba como fazer isso, pode passar para o tópico seguinte.

2.1) Configurando o ADB

Para realizar a configuração do ADB, é necessário abrir o **prompt de comando** e navegar para a pasta **Atualização Apps Vero Desenvolvimento**, onde há a pasta **platform-tools** e os arquivos **APK** de atualização de ambos os aplicativos Vero Pagamentos e Vero Store. Isso pode ser feito entrando nesta pasta pelo navegador de arquivos, clicando na barra de endereços, digitando **cmd** e apertando **enter**. Em seguida, copie e cole o comando:

```
SET PATH=platform-tools
```

2.2) Atualizando a Vero Store


É necessário atualizar primeiro a Vero Store, usando o comando:

```
adb install -r VeroStore-1.00.63-debug-desenv.apk
```

Será exibido a mensagem de **success**, então avance para a atualização do Vero Pagamentos.

2.3) Atualizando o Vero Pagamentos

Esta atualização deve ser feita após a atualização da Vero Store, pois ao ser realizada, a aplicação automaticamente fará uma operação de fechamento de caixa, seguido de uma operação de baixa das tabelas EMV, finalizando com



uma configuração da Vero Store em ambiente de desenvolvimento, que habilitará a Vero Store a instalar e atualizar aplicações e o SO do terminal de desenvolvimento. Para atualizar o Vero Pagamentos, digite um dos comandos abaixo, para a device plataforma do seu Smart POS

adb install -r VeroPagamentos-8.05.06-G700-release-desenv.apk

adb install -r VeroPagamentos-8.05.14-P2-release-desenv.apk

Ao aparecer a mensagem de sucesso, abra a aplicação Vero Pagamentos apertando o botão **Home** abaixo do display do terminal (botão do meio), e o processo descrito anteriormente irá ser realizado. É necessário que o terminal haja conexão com internet para que seja concluído com sucesso. Automaticamente a Vero Store começará a baixar atualizações pendentes, incluindo de SO. Ao realizar atualizações de SO, o terminal irá fechar o Vero Pagamentos e irá reiniciar o terminal, e ao terminar de reiniciar, irá exibir uma tela onde irá concluir a atualização, e após um minuto irá reabrir automaticamente o Vero Pagamentos. É importante que o terminal esteja com bateria suficiente, e que não seja usado durante todo esse processo.

Com o Vero Pagamentos, a Vero Store e o SO atualizados para as versões mais recentes, você possuirá o ambiente mais próximo possível ao ambiente encontrado em campo, permitindo até mesmo que você teste a atualização do seu aplicativo pela Vero Store. Para isso, é importante que o APK que você suba no site da Vero Store sempre esteja assinado com o certificado do terminal de desenvolvimento, mesmo que o destino do seu APK seja os terminais de produção, para os quais o seu APK será reassinado com o certificado de produção. Esse assunto é abordado no próximo tópico.

3) Instalando seu aplicativo

Para instalar sua aplicação no terminal de desenvolvimento, é necessário assinar a sua aplicação com o certificado de desenvolvimento, caso contrário, ao tentar instalar o seu APK no terminal, irá ocorrer o erro:

INSTALL_PARSE_FAILED_INCONSISTENT_CERTIFICATES

O G700 de desenvolvimento apenas permite a instalação de aplicativos assinados com o certificado que está incluso no arquivo compactado que contém esse documento, na pasta **Instalação de Apps no GPOS700**. Caso você use o Android Studio, pode adicionar a configuração da assinatura no arquivo **app/build.gradle** da seguinte maneira:

```
android {  
    ...  
    signingConfigs {  
        gertec {  
            storeFile file("../Development_3IA.jks")  
            storePassword '3IADevelopmentKeyStore'  
            keyAlias 'gpos700Enhanced'  
            keyPassword '3IA2018'  
        }  
    }  
    ...  
    buildTypes {  
        all {  
            ...  
            signingConfig signingConfigs.gertec  
            ...  
        }  
    }  
    ...  
}
```

Após realizado a configuração para a assinatura, quando o seu APK for gerado, você pode instalar tanto pelo Android Studio quanto via ADB. O comando para instalar via ADB é:

```
adb install -r nome_do_seu_pacote.apk
```

4) Biblioteca Auxiliar Smart POS

A biblioteca auxiliar provê funcionalidades específicas do terminal, sendo elas:

- Impressora do terminal;
- Comunicação com a Vero Store;
- Acesso à internet via redes móveis utilizando o chip privado fornecido pelo adquirente.

Para utilizar a biblioteca, é necessário incluir nas dependências do seu projeto o arquivo **posmp_api-1.00.87.aar**. Esse arquivo está localizado na pasta **Biblioteca Auxiliar para Smart POS**. Segue exemplo usando **Gradle** no Android Studio. Dentro do arquivo **app/build.gradle**:

```
...
dependencies {
    ...
    implementation(name: 'posmp_api-1.00.87', ext: 'aar')
    ...
}
```

4.1) Impressora

A impressora do terminal é capaz de imprimir texto simples ou imagens no formato **bitmap**. A utilização é independente do modelo do terminal, e conforme sejam adicionados outros terminais, a biblioteca será atualizada visando manter a transparência para o desenvolvedor. Segue exemplo de uso:

```
...
import br.com.execucao.posmp_api.printer.Printer;
...
Printer printer = Printer.getInstance(context);
...
printer.print(bitmap);
printer.eject();
printer.close();
...
```

É possível também enviar funções de call-back para serem chamadas quando a impressão for concluída, ambas as variantes são **não-blocantes**. Segue exemplo:

```
printer.print(toPrint, new
br.com.execucao.posmp_api.printer.PrinterListener() {
    @Override
    public void onFinish() {
        // impressão ocorreu com sucesso
    }

    @Override
    public void onError(int error) {
        // houve um erro na impressão:

        // br.com.execucao.posmp_api.printer.Printer.ERRO_SEM_PAPEL
        // br.com.execucao.posmp_api.printer.Printer.ERRO_DESCONHECIDO
    }

    @Override
    public IBinder asBinder() {
        return null;
    }
});
```

4.2) Comunicação com a Vero Store

A comunicação com a loja tem o propósito de informar o estado de ativação do seu aplicativo para o adquirente realizar a cobrança de valores pelo uso do seu aplicativo e também receber informações sobre o terminal e lojista. Os possíveis estados que sua aplicação pode informar são:

Tipo	Descrição	Valor Inteiro	Valor String
enum	Aplicação Inativa	1	INA
enum	Aplicação de Demonstração	2	DEM
enum	Aplicação Ativa	3	ATV

Para maiores informações sobre as modalidades de cobrança que serão efetuadas ao lojista, entre em contato com o adquirente.

Exemplo de uso apenas enviando a informação de ativação:

```
import br.com.execucao.posmp_api.socket.SituationsEnum;
import br.com.execucao.posmp_api.socket.SocketInformation;
...
SocketInformation.sendInformation(SituationsEnum.ATV, context);
...
```

Existe também a possibilidade de receber informações sobre o terminal e o lojista, as quais estão disponíveis a partir da versão 1.00.41 do aplicativo Vero Store, que já está em produção. Exemplo de uso enviando a informação de ativação e recebendo os dados sobre o terminal e lojista:

```
import br.com.execucao.posmp_api.socket.SituationsEnum;
import br.com.execucao.posmp_api.socket.SocketInformation;
...
JSONObject json = null;
...
new Thread(new Runnable() {
    public void run() {
        json =
        SocketInformation.sendAndReceiveInformation(MenuVero.this,
        SituationsEnum.ATV);
    }
}).start();
...
```

Como é necessário aguardar o recebimento da resposta com os dados, a função **sendAndReceiveInformation** não pode ser executada na thread da interface do usuário (UI) pois é bloqueante. Esta função retorna um JSON que pode conter as seguintes informações:

Tipo	Nome	Valor exemplo	Descrição
String	"SerialNumber"	"5201051902000038"	O número de série do terminal.
String	"Rede"	"004100000000"	O número da rede a qual o lojista está cadastrado no adquirente.
String	"Estabelecimento"	"0000000000006225"	O número do estabelecimento cadastrado no adquirente.
String	"Terminal"	"50001531"	O número do terminal cadastrado no adquirente.

4.3) Internet Móvel

A Vero utiliza em seus terminais chips de telefonia móvel **privados**, ou seja, não utilizam a infraestrutura convencional, adicionando assim um nível de segurança extra. Todas as conexões estabelecidas utilizando a internet móvel desses chips passam por um proxy privado, e apenas endereços pré-aprovados são permitidos. No momento tais restrições não estão sendo aplicadas por estarem em testes, porém no futuro isso pode mudar, portanto é de responsabilidade do desenvolvedor informar a quais domínios eles desejam se conectar. Para fazer isso, é necessário criar, na pasta **assets** do seu projeto, um arquivo de nome **hosts.xml** com a lista de domínios aos quais sua aplicação conecta, no formato abaixo:

```
<resources>
  <string-array name="hosts">
    <item>www.google.com.br</item>
    <item>www.stackoverflow.com</item>
  </string-array>
</resources>
```

Dessa maneira, quando você realizar o upload da sua aplicação para o site da Vero Store, iremos buscar esse arquivo a fim de liberar esses domínios, caso sejam aprovados pelo adquirente.

O uso da função disponível na biblioteca substitui o uso da função **URLConnection()**, já que é testado internamente se o terminal está conectado na Wi-Fi ou em redes móveis, então independente da conexão, será retornado uma conexão funcional. Segue exemplo:

```
import br.com.execucao.posmp_api.connection.Connectivity;
...
Connectivity connectivity;
...
connectivity = new Connectivity(context);

try {
    // exemplo de uso 1

    HttpURLConnection connection = (HttpURLConnection)
connectivity.getConnection(new URL(address1));
    // exemplo de uso 2

    connection = (HttpURLConnection)
connectivity.getConnection(address2);
    // exemplo de uso 3

    HTTPSURLConnection connectionHTTPS = (HTTPSURLConnection)
connection;

} catch (IOException e) {}
```

É possível também configurar o proxy globalmente dentro do aplicativo, da maneira abaixo:

```
import br.com.execucao.posmp_api.connection.Connectivity;
...
Connectivity connectivity;
...
connectivity = new Connectivity(context);
connectivity.setSystemProxy(true);

...
connectivity.refreshInfo(context);
connectivity.setSystemProxy(true);
...
```

Porém, é necessário chamar a função **refreshInfo()** e **setSystemProxy(true)** periodicamente, para caso a interface da internet mude, o proxy seja configurado ou desconfigurado sob demanda.

Os métodos **getConnection** e **setSystemProxy** não foram implementados de maneira a trabalharem paralelamente, por tanto deve ser escolhido apenas uma das duas abordagens para ser implementado no seu aplicativo.

5) Solicitações de Pagamento Externo

A aplicação de pagamentos Vero Smart aceita a requisição de solicitações de pagamento através de **Intents**. Existem várias transações configuráveis a serem requisitadas, descritas no tópico abaixo.

5.1) Transações Suportadas

- Transação com valor e modalidade a serem definidas;
- Transação com valor definido, e modalidade a ser definida;
- Transação com valor definido de crédito (a definir);
- Transação com valor definido de crédito a vista;
- Transação com valor definido de crédito parcelado, financiado pelo lojista, com número de parcelas definido;
- Transação com valor definido de crédito parcelado, financiado pelo emissor, com número de parcelas definido;
- Transação com valor definido de crédito recorrente Vero Repay, com prazo e melhor dia de pagamento definidos;
- Transação com valor definido de débito (a definir);
- Transação com valor definido de débito a vista;
- Transação com valor definido de débito Banricompras a vista;
- Transação com valor definido de débito Banricompras pré-datado com prazo definido;
- Transação com valor definido de débito Banricompras parcelado com número de parcelas definido;
- Transação com valor definido de débito Banricompras Crédito 1 Minuto com número de parcelas definido;
- Transação com valor definido utilizando voucher;
- Cancelamento (Estorno);
- Transação com valor definido Vero Wallet (Carteira Digital Vero);
- Transação com valor definido Vero X (Pix);
- Transação com valor definido digitada;
- Transação com valor definido VeroPay.

Junto a esse documento, na pasta **Pagamento Externo para Smart POS**, há um aplicativo exemplo que implementa a requisição de todos os tipos de transação suportadas, junto ao seu código fonte. Esse aplicativo simula a implementação que você pode fazer na sua aplicação.

No tópico a seguir será descrito todos os parâmetros a serem definidos na Intent antes de chamá-la.

5.2) Ação da Intent (Action)

A intent que deve ser chamada para iniciar o pagamento externamente deve possuir uma das ações da tabela abaixo.

Tipo	Nome	Valor	Descrição
String	INTENT_PAGAR	"br.com.execucao.PAGAR"	Transação de pagamento.
String	INTENT_ESTORNAR	"br.com.execucao.ESTORNAR"	Transação de estorno.

Declaração:

```
private static final String INTENT_PAGAR = "br.com.execucao.PAGAR";  
private static final String INTENT_ESTORNAR =  
"br.com.execucao.ESTORNAR";
```

5.3) Código de Requisição da Intent (Request code)

O código de solicitação que será retornado na intent de retorno.

Tipo	Nome	Valor	Descrição
Int	REQUISICAO	123	Valor livre para identificação na intent de retorno.

Declaração:

```
private static final int REQUISICAO = 123;
```

5.4) Extras da Intent

São os parâmetros que irão definir a transação.

Tipo	Nome	Valor	Descrição
Int	VALOR_TRANSACAO	Exemplo: 2500	Em centavos. No exemplo R\$ 25.00.
String	TRANSACAO	"CREDITO_GENERICO", "CREDITO", "PARCELADO_LOJISTA", "PARCELADO_EMISSOR", "REPAY", "DEBITO", "DEBITO_VISTA", "BANRICOMPRAS", "PREDATADO", "DEBITO_PARCELADO", "CREDITO1M", "VOUCHER", "WALLET", "PIX", "DIGITADA", "VEROPAY"	Pré-configuração de alguma transação específica.
Int	NUMERO_PARCELAS	Exemplo: 2	Número de Parcelas, caso aplicável.
Int	PRAZO	Exemplo: 30	Em dias. No exemplo 30 dias.
Int	DIA	Exemplo: 12	Dia do mês. No exemplo dia 12.
String	NSU	Exemplo: "12345678"	Número Sequencial Único da transação.

Declaração:

```
private static final String VALOR_TRANSACAO = "VALOR";
private static final String NUMERO_PARCELAS = "NUMPARCELAS";
private static final String PRAZO = "PRAZO";
private static final String DIA = "DIA";
private static final String NSU = "NSU";
private static final String TRANSACAO = "TRANSACAO";
private static final String TRANSACAO_CREDITO = "CREDITO_GENERICO";
private static final String TRANSACAO_CREDITO_VISTA = "CREDITO";
private static final String TRANSACAO_CREDITO_PARCELADO_LOJISTA =
"PARCELADO_LOJISTA";
private static final String TRANSACAO_CREDITO_PARCELADO_EMISSOR =
"PARCELADO_EMISSOR";
private static final String TRANSACAO_CREDITO_REPAY = "REPAY";
private static final String TRANSACAO_DEBITO = "DEBITO";
private static final String TRANSACAO_DEBITO_VISTA = "DEBITO_VISTA";
private static final String TRANSACAO_BANRICOMPRAS_VISTA =
"BANRICOMPRAS";
private static final String TRANSACAO_BANRICOMPRAS_PREDATADO =
"PREDATADO";
private static final String TRANSACAO_BANRICOMPRAS_PARCELADO =
"DEBITO_PARCELADO";
private static final String
TRANSACAO_BANRICOMPRAS_CREDITO_1_MINUTO = "CREDITO1M";
private static final String TRANSACAO_VOUCHER = "VOUCHER";
private static final String TRANSACAO_WALLET = "WALLET";
private static final String TRANSACAO_PIX = "PIX";
private static final String TRANSACAO_DIGITADA = "DIGITADA";
private static final String TRANSACAO_DIGITADA_VEROPAY = "VEROPAY";
```

5.5) Extras da Intent de Retorno

São os parâmetros retornados na intent de retorno, com os dados da transação. Caso não haja valor em AUTORIZACAO, verifique o parâmetro ERRO com o possível erro que ocorreu.

Tipo	Nome	Descrição
String	SERIAL	Número de série do terminal.
String	BANDEIRA	Visa, Mastercard, Cabal, Sodexo...
String	NUMCARTAO	Número do cartão em forma mascarada.
String	TRANSACAO	Nome da transação (crédito, débito ou voucher).
String	VALOR	Valor da transação.
String	PARCELAS	Número de parcelas da transação.
String	AUTORIZACAO	Número da autorização da transação.
String	NSU	Número Sequencial Único. Utilizado em caso de estorno da transação.
String	CNPJ_ADQUIRENTE	CNPJ da adquirente que executou a transação.
String	NOME_ADQUIRENTE	Nome da adquirente que executou a transação.
String	COMPROVANTE	Comprovante da transação.
String	ERRO	Mensagem de declínio da transação.

Declaração:

```
public static final String SERIAL = "SERIAL";  
public static final String BANDEIRA = "BANDEIRA";  
public static final String NUMCARTAO = "NUMCARTAO";  
public static final String TRANSACAO = "TRANSACAO";  
public static final String VALOR = "VALOR";  
public static final String PARCELAS = "PARCELAS";  
public static final String PRAZO = "PRAZO";  
public static final String DIA = "DIA";  
public static final String AUTORIZACAO = "AUTORIZACAO";  
public static final String NSU = "NSU";  
public static final String CNPJ_ADQUIRENTE = "CNPJ_ADQUIRENTE";  
public static final String NOME_ADQUIRENTE = "NOME_ADQUIRENTE";  
public static final String COMPROVANTE = "COMPROVANTE";
```

5.6) Descrição e Exemplos das Transações

5.6.1) Sem valor e sem transação

Neste modelo, tanto o valor quanto a modalidade (e suas particularidades) serão solicitadas ao usuário, para em seguida solicitar a apresentação do cartão.

```
try {  
    Intent intent = new Intent(INTENT_PAGAR);  
    startActivityForResult(intent, REQUISICAO);  
} catch (Exception e) {  
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");  
}
```

5.6.2) Com valor definido e sem transação

Transação apenas com o valor definido, a modalidade (e suas particularidades) será solicitada ao usuário, para em seguida solicitar a apresentação do cartão.

```
int valor = 2500; // R$ 25,00  
  
try {  
    Intent intent = new Intent(INTENT_PAGAR);  
    intent.putExtra(VALOR_TRANSACAO, valor);  
    startActivityForResult(intent, REQUISICAO);  
} catch (Exception e) {  
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");  
}
```

1.6.1) Crédito

Transação de valor definido de crédito, a modalidade de crédito (e suas particularidades) será solicitada ao usuário, para em seguida solicitar a apresentação do cartão

```
int valor = 3000; // R$ 30,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO, TRANSACAO_CREDITO);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.3) Crédito a Vista

Transação de valor definido de crédito a vista. Será solicitado ao usuário a apresentação do cartão.

```
int valor = 3000; // R$ 30,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO, TRANSACAO_CREDITO_VISTA);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.4) Crédito Parcelado pelo Lojista

Transação de valor definido de crédito parcelado (financiado pelo lojista) com número de parcelas já definido. Será solicitado ao usuário a apresentação do cartão.

```
int valor = 50000; // R$ 500,00
int nParcelas = 4;
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO,
        TRANSACAO_CREDITO_PARCELADO_LOJISTA);
    intent.putExtra(NUMERO_PARCELAS, nParcelas);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.5) Crédito Parcelado pelo Emissor

Transação de valor definido de crédito parcelado (financiado pelo emissor) com número de parcelas já definido. Será solicitado ao usuário a apresentação do cartão.

```
int valor = 50000; // R$ 500,00
int nParcelas = 2;
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO,
        TRANSACAO_CREDITO_PARCELADO_EMISSOR);
    intent.putExtra(NUMERO_PARCELAS, nParcelas);
```



```

startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.6) Crédito Vero Repay

Transação de valor definido de crédito recorrente Vero Repay, com prazo e melhor dia de cobrança já definidos. Será solicitado ao usuário a apresentação do cartão.

```

int valor = 25000; // R$ 250,00
int prazo = 30;
int diaRecorrencia = 12;
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO, TRANSACAO_CREDITO_REPAY);
    intent.putExtra(DIA, diaRecorrencia);
    intent.putExtra(PRAZO, prazo);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.7) Débito

Transação de valor definido de débito. Irá solicitar ao usuário a apresentação do cartão, e dependendo do cartão apresentado, será exibido mais opções de modalidade (caso cartão Banricompras).

```

int valor = 10000; // R$ 100,00
try {

```

```

Intent intent = new Intent(INTENT_PAGAR);
intent.putExtra(VALOR_TRANSACAO, valor);
intent.putExtra(TRANSACAO, TRANSACAO_DEBITO);
startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.8) Débito a vista

Transação de valor definido de débito a vista (não Banricompras). Irá solicitar ao usuário a apresentação do cartão.

```

int valor = 10000; // R$ 100,00
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO, TRANSACAO_DEBITO_VISTA);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.9) Banricompras Débito a vista

Transação de valor definido de débito a vista (não Banricompras). Irá solicitar ao usuário a apresentação do cartão.

```

int valor = 10000; // R$ 100,00
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);

```

```

intent.putExtra(TRANSACAO, TRANSACAO_BANRICOMPRAS_VISTA);
startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.10) Banricompras Débito Pré-datado

Configura uma transação **Banricompras** de débito pré-datado, com o valor e prazo já definidos. Irá solicitar ao usuário a apresentação do cartão.

```

int valor = 5000; // R$ 50,00
int prazo = 30; // 30 dias
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO,
TRANSACAO_BANRICOMPRAS_PREDATADO);
    intent.putExtra(PRAZO, prazo);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.11) Banricompras Débito Parcelado

Configura uma transação **Banricompras** de débito parcelado, com o valor e número de parcelas já definidos. Irá solicitar ao usuário a apresentação do cartão.

```

int valor = 15000; // R$ 150,00
int parcelas = 2; // duas parcelas
try {

```

```

Intent intent = new Intent(INTENT_PAGAR);
intent.putExtra(VALOR_TRANSACAO, valor);
intent.putExtra(TRANSACAO,
TRANSACAO_BANRICOMPRAS_PARCELADO);
intent.putExtra(NUMERO_PARCELAS, parcelas);
startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.12) Banricompras Crédito 1 Minuto

Configura uma transação **Banricompras** de Crédito 1 Minuto, com o valor e número de parcelas já definidos. Irá solicitar ao usuário a apresentação do cartão.

```

int valor = 35000; // R$ 350,00
int parcelas = 6; // seis parcelas
try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO,
TRANSACAO_BANRICOMPRAS_CREDITO_1_MINUTO);
    intent.putExtra(NUMERO_PARCELAS, parcelas);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}

```

5.6.13) Voucher

Configura uma transação de débito, com valor definido. Irá solicitar ao usuário a apresentação do cartão, seguido das particularidades da transação caso aplicável.

```
int valor = 1000; // R$ 10,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(VALOR_TRANSACAO, valor);
    intent.putExtra(TRANSACAO, TRANSACAO_VOUCHER);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.14) Cancelamento (estorno)

Irá configurar uma transação de cancelamento de alguma transação previamente aprovada. Irá solicitar ao usuário para apresentar o cartão usado na transação original.

```
int nsuEstornar = "12345678"; // Valor obtido no retorno da requisição da
transação aprovada.

try {
    Intent intent = new Intent(INTENT_ESTORNAR);
    intent.putExtra(NSU, nsuEstornar);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.15) Vero Wallet

Configura uma transação para ser paga através da carteira digital da Vero, usando cartões convencionais de crédito, cartões Banricompras e Banricard, de valor já definido. Irá exibir um QR CODE para o usuário escanear e realizar o pagamento.

```
int valor = 2700; // R$ 27,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(TRANSACAO, TRANSACAO_WALLET);
    intent.putExtra(VALOR_TRANSACAO, valor);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.16) Vero X (Pix)

Configura uma transação PIX com valor definido. Irá exibir um QR CODE para o usuário realizar o pagamento.

```
int valor = 2800; // R$ 28,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(TRANSACAO, TRANSACAO_PIX);
    intent.putExtra(VALOR_TRANSACAO, valor);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.17) Crédito digitado

Configura uma transação de valor definido com cartão digitado. Irá solicitar a modalidade de crédito e a entrada dos dados do cartão..

```
int valor = 2800; // R$ 28,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(TRANSACAO, TRANSACAO_DIGITADA);
    intent.putExtra(VALOR_TRANSACAO, valor);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.18) Veropay

Configura uma transação de débito Banricompras usando cartão digitado de criação única Veropay. Irá solicitar a entrada dos dados do cartão Veropay, seguido da modalidade banricompras.

```
int valor = 2800; // R$ 28,00

try {
    Intent intent = new Intent(INTENT_PAGAR);
    intent.putExtra(TRANSACAO, TRANSACAO_DIGITADA_VEROPAY);
    intent.putExtra(VALOR_TRANSACAO, valor);
    startActivityForResult(intent, REQUISICAO);
} catch (Exception e) {
    Log.e(TAG, "Não localizamos aplicativo de pagamento neste terminal.");
}
```

5.6.19) Exemplo para captura do retorno da chamada

Ao chegar ao fim do processo de pagamento, a aplicação Vero Pagamentos irá retornar uma intent com os detalhes da transação, veja no tópico abaixo um exemplo.

@Override

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == REQUISICAO) {
        if (resultCode == RESULT_OK) {
            if (data == null) {
                Log.e (TAG, "Transação rejeitada");
                return;
            }
            else
            {
                String autorizacao = data.getStringExtra("AUTORIZACAO");
                String bandeira = data.getStringExtra("BANDEIRA");
                String comprovante = data.getStringExtra("COMPROVANTE");
                String adquirente = data.getStringExtra("CNPJ_ADQUIRENTE");
                String nsu = data.getStringExtra("NSU");
                String valorTransacionado = data.getStringExtra("VALOR");
                if (this.autorizacao != null){
                    Log.d("Autorização = ", autorizacao);
                    Log.d("NSU = ", nsu);
                    Log.d("Comprovante = ", comprovante);
                }
                else if (data.getStringExtra("ERRO") != null){
                    Log.e("ERRO = ", data.getStringExtra("ERRO"));
                }
            }
        } else if (resultCode == RESULT_CANCELED){
            Log.e("Retorno = ", "Transação cancelada"); }
    }
}
```


6) Leitura de Cartão MIFARE

Atualmente apenas disponível na plataforma G700, é possível solicitar a aplicação de pagamentos para que leia a tecnologia MIFARE. Na pasta **Pagamento Externo para Smart POS** também há um APK da versão **beta** do aplicativo Vero Pagamentos que suporta a leitura de cartão MIFARE, que ainda não estão em produção, para você poder testar a implementação dessas requisições, caso deseje usá-las quando forem para produção.

O projeto exemplo da mesma pasta também tem um exemplo de implementação da chamada da intent para leitura de cartão MIFARE, e o apk do projeto exemplo já contém o botão **LER MIFARE** que realiza a chamada. Veja abaixo a descrição de cada campo da intent.

Segue a lista de extras de solicitação da intent.

EXTRA "**TIPO_CARTAO**": Qualquer valor diferente de "A" irá considerar o cartão MIFARE do tipo B.

EXTRA "**ENDERECOS**": a lista de endereços com as quais será interagido.

EXTRA "**CHAVES_AUTENTICACAO**": a lista de chaves de autenticação para os endereços do extra anterior, na mesma ordem.

EXTRA "**DADOS_ESCRITA**": a lista de valores a serem escritas nos endereços listados. Valor NULL significa que o valor nesse endereço será lido e retornado na intente de retorno.

Segue a liste de extras da intent de retorno.

EXTRA "**RESULTADO**": o resultado da operação. Qualquer valor diferente de "OK" significa que um erro ocorreu.

EXTRA "**CARTAO_UID**": o UID do cartão aproximado.

EXTRA "**BYTES_LIDO**": a lista de bytes lidos, na mesma ordem dos endereços passados na intent de requisição. Valor NULL indica que foi realizado uma escrita nesse endereço.

7) Considerações Finais

Com o terminal corretamente inicializado em ambiente de desenvolvimento, você pode testar a realização de pagamentos com cartões de crédito/débito das bandeiras MasterCard e Visa, que estas operações não terão validade. Segue abaixo o fluxo completo de pagamentos utilizando um cartão de crédito MasterCard:

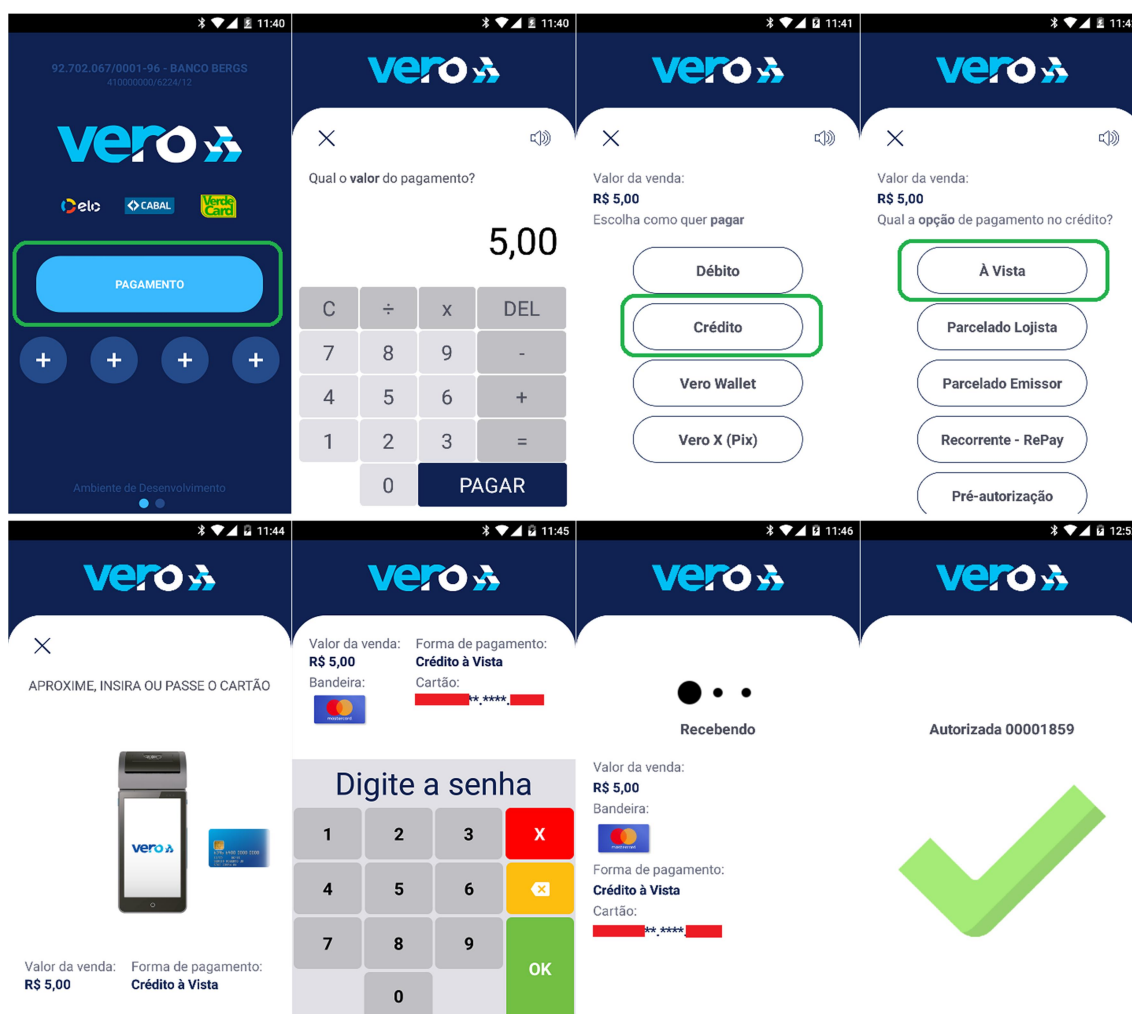


Figura 6. Fluxo completo de transação

Caso ocorra um erro logo em seguida ao entrar na tela de senha, o seu terminal não possui as chaves criptográficas de testes para capturar a senha. Para resolver isso no G700, será necessário carregá-las instalando o app que está na pasta **Inserção das chaves**, usando as ferramentas ADB conforme descrito no tópico 2. Após instalar o aplicativo **app-KeyLoader_v1.2.apk**, abra ele selecionando o app **KeyLoader** na bandeja de aplicativos da Vero, e siga as instruções das figuras abaixo.

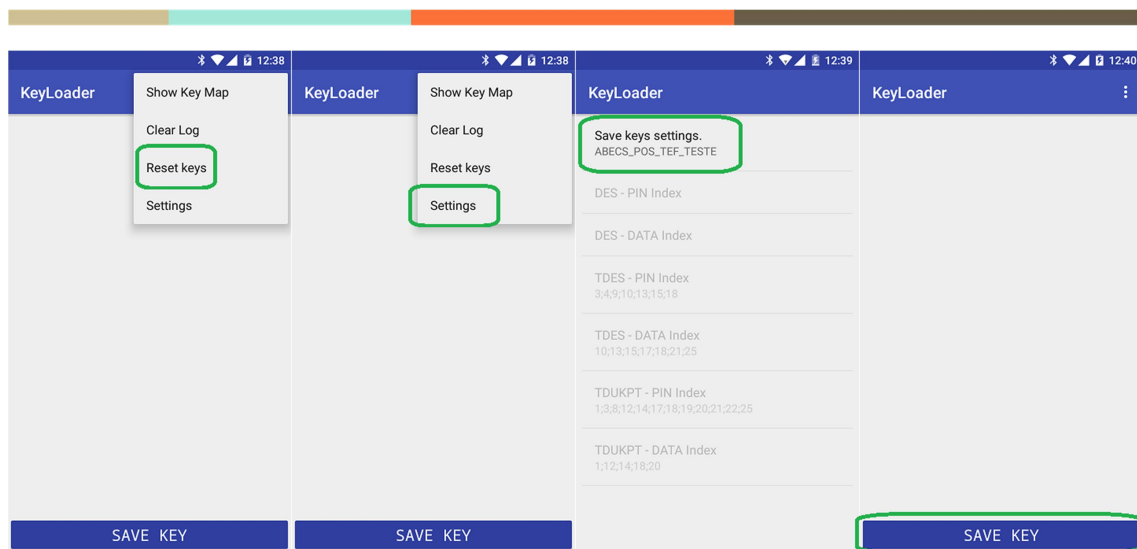


Figura 7. Inserção das chaves com o KeyLoader

Após realizado esse processo, agora você será capaz de realizar transações com o seu terminal de desenvolvimento.