UNIVERSIDADE FEDERAL DE JUIZ DE FORA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
ESPECIALIZAÇÃO EM DESENVOLVIMENTO DE SISTEMAS COM TECNOLOGIA JAVA

INTEGRAÇÃO JAVA E NFC: NOVAS APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

TRABALHO DE CONCLUSÃO

BERNARDO JACQUES DELGADO COSTA

Juiz de Fora, Minas Gerais, Brasil
2011

INTEGRAÇÃO JAVA E NFC:
NOVAS APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

por

Bernardo Jacques Delgado Costa

Trabalho de conclusão apresentado ao Departamento de Ciência da Computação da Universidade Federal de Juiz de Fora para obtenção do título de
Especialista em Desenvolvimento de Sistemas com Tecnologia Java.

Juiz de Fora, Minas Gerais, Brasil
2011

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE ESPECIALIZAÇÃO

A comissão examinadora abaixo assinada, aprova o trabalho de conclusão:

INTEGRAÇÃO JAVA E NFC:
NOVAS APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

elaborado por

BERNARDO JACQUES DELGADO COSTA

como requisito parcial para obtenção do título de

ESPECIALISTA EM DESENVOLVIMENTO DE SISTEMAS COM TECNOLOGIA JAVA

_____
WANDER ANTUNES GASPAR VALENTE

_____
TARCÍSIO DE SOUZA LIMA

_____
MARCELO ADRIAN FLORES MANRIQUE

Juiz de Fora, Minas Gerais, Brasil
2011

AGRADECIMENTOS

Aos meus pais, por todo amor e educação dispensados ao longo da vida. Sempre presentes, me incentivando e apoiando minhas decisões.

Ao meu orientador Wander, pela atenção, dedicação e apoio de sempre, agora neste trabalho de pós-graduação.

À Flávia, pelo carinho e apoio incondicional durante todo o processo de pesquisa.

Aos amigos da Lupa, pelo apoio e ensinamentos proporcionados durante esses anos de convivência.

E a todos meus amigos, que influenciaram direta ou indiretamente para a conclusão deste trabalho.

“A mente que se abre a uma nova idéia
jamais volta ao seu tamanho original”
Albert Einstein

RESUMO

A tecnologia *Near Field Communication* (NFC), com seu princípio de comunicação por proximidade, surge como um novo padrão de comunicação, passível de integração com os dispositivos móveis. A linguagem de programação Java, por meio da *Contactless Communication* API, torna possível essa integração. Sendo assim, o contexto deste trabalho consiste em abordar as plataformas Java Micro *Edition* (JME) e Java *Card*, que permitem o desenvolvimento de aplicações NFC. Além disso, também foram pesquisadas as áreas de atuação do NFC e as possibilidades que surgem na utilização desse padrão pelos dispositivos móveis.

Palavras-chave: NFC. Java. Dispositivo móvel. *Contactless Communication* API.

ABSTRACT

The Near Field Communication (NFC), with its proximity communication concept, emerges as a new communication standard, which can be integrated with mobile devices. The Java programming language, using the Contactless Communication API, makes such integration possible. Thus, the context of this work is to present Java Micro Edition and Java Card platforms, which allows the development of NFC applications. In addition, areas of the NFC were also investigated along with the possibilities that arise in using this standard for mobile devices.

Keywords: NFC. Java. Mobile device. *Contactless Communication* API.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

ANATEL	Agência Nacional de Telecomunicações
APDU	Application Protocol Data Unit
API	Application Programming Interface
AWT	Abstract Window Toolkit
CDC	Connected Device Configuration
CLDC	Connected Limited Device Configuration
FP	Foundation Profile
GCF	Generic Connection Framework
ISO	International Organization for Standardization
JCP	Java Community Process
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JEE	Java Enterprise Edition
JME	Java Micro Edition
JSE	Java Standard Edition
JSR	Java Specification Request
JVM	Java Virtual Machine
KVM	Kilobyte Virtual Machine

KB	Kilobyte
Kbps	Kilobit por segundo
MHz	MegaHertz
MIDP	Mobile Information Device Profile
NDEF	NFC Data Exchange Format
NFC	Near Field Communication
NFCIP-1	Near Field Communication Interface and Protocol – 1
NFCIP-2	Near Field Communication Interface and Protocol – 2
PBP	Personal Basis Profile
PDA	Personal Digital Assistants
PP	Personal Profile
RFID	Radio-Frequency Identification
URL	Uniform Resource Locator

SUMÁRIO

INTRODUÇÃO

Em crescente evolução, os dispositivos móveis difundiram-se bastante nos dias atuais. Segundo dados da Agência Nacional de Telecomunicações (ANATEL), o Brasil possui mais de 212,6 milhões de acessos habilitados (ANATEL, 2011). Os *smartphones*, de acordo com estudos do “IDC Latin America Predictions 2011”, estão cada vez mais em foco, devendo superar a venda de computadores *desktop* ainda em 2011 (UOL, 2011).

Em paralelo, mecanismos capazes de promover uma maior interação com esses equipamentos estão sendo desenvolvidos. Hoje em dia os celulares já permitem, por exemplo, acessar a internet através de uma conexão *wi-fi*, tirar fotos ou ainda tocar músicas via *bluetooth*. Seguindo essa evolução tecnológica surgiu, em uma parceria das empresas Philips e Sony, o protocolo *Near Field Communication* (NFC). Com o princípio de comunicação por proximidade, este padrão se torna bem atraente para os dispositivos móveis, ao possibilitar a troca de dados com segurança, uma vez que a comunicação ocorre a curtas distâncias, é difícil que ela seja interceptada por outro equipamento, e em velocidade de até 424 kbps (FERRAZ FILHO, 2010; SIMÕES, 2008).

Na cidade de Tóquio, no Japão, a tecnologia NFC já é utilizada no sistema de metrô, permitindo a compra de passagens com a aproximação do celular às catracas. Em um futuro próximo essa tecnologia poderá vir a substituir os cartões de crédito, ao permitir que os usuários façam pagamento através de aproximação do celular a um leitor compatível (HECKE, 2011).

Para promover essa realidade, a plataforma Java, por meio da *Contactless Communications* API, permite a criação de aplicações capazes de integrar o NFC a esses equipamentos (ORTIZ, 2008).

O curso de especialização em desenvolvimento de sistemas com tecnologia Java agregou conhecimento à produção de sistemas que utilizem esta linguagem de programação. O curso se preocupou em abordar as três principais divisões da linguagem, dentre elas a Java Micro *Edition* (JME), voltada ao desenvolvimento de aplicações para dispositivos móveis. Neste contexto, o presente trabalho tem como objetivo analisar a plataforma Java, com foco na *Contactless Communication* API, e também a tecnologia NFC, no contexto dos dispositivos móveis. O trabalho deve mostrar ainda a integração entre estas tecnologias em situações reais.

1 PLATAFORMAS JAVA MICRO *EDITION* E JAVA *CARD*

Este capítulo apresenta a linguagem Java e suas plataformas, dando ênfase à Java *Micro Edition* (JME) e à Java *Card*, que podem ser utilizadas no desenvolvimento de aplicações NFC.

1.1 LINGUAGEM

Baseada em C++, a linguagem Java começou a ser desenvolvida no início da década de 90, pela empresa Sun Microsystems, sendo formalmente anunciada em 1995 (DEITEL, 2010). Uma de suas características é ser independente de plataforma. Isso é possível uma vez que a linguagem Java, quando compilada, converte o código-fonte em *bytecodes* que são traduzidos, em tempo de execução, por um interpretador conhecido como *Java Virtual Machine* (JVM), que simula um computador abstraindo o sistema operacional e o hardware subjacente. Por ser um conjunto de tarefas, o *bytecode* pode ser executado em qualquer máquina que possua uma JVM, que por sua vez também não depende de uma plataforma específica para execução (DEITEL, 2010). Sendo assim, o sucesso dessa linguagem a partir de então se deve à portabilidade de código, que permite

que um mesmo código-fonte seja utilizado em diferentes plataformas (JOHNSON, 2007).

Para facilitar a utilização em diferentes ambientes, a tecnologia Java foi dividida em três grandes plataformas: Java *Enterprise Edition* (JEE), Java *Standard Edition* (JSE) e Java *Micro Edition* (JME), como mostra a Figura 1. Sendo cada uma dessas plataformas voltada a um grupo específico de *hardware*.

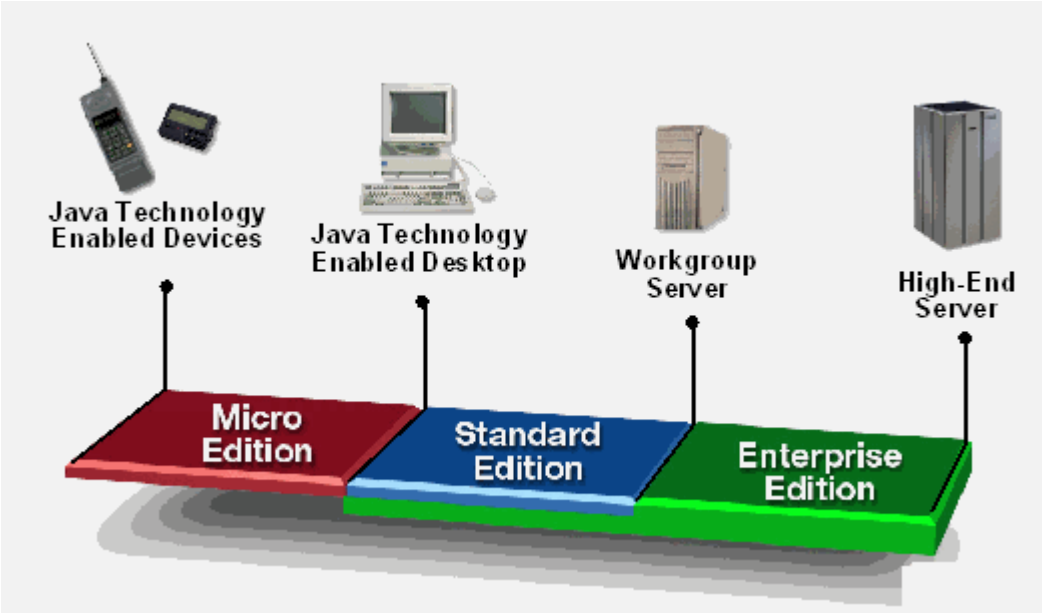

Figura 1 - Plataforma Java e suas subdivisões.
Fonte: UCHÔA, 2010.

Complementar a essa divisão, existe a plataforma *Java Card*, voltada para o desenvolvimento de pequenas aplicações, principalmente em cartões inteligentes.

1.2 JAVA MICRO *EDITION*

A plataforma Java Micro Edition foi desenvolvida pela Sun Microsystems e lançada em 1999, com o objetivo de permitir o desenvolvimento de aplicativos Java para dispositivos móveis e com recursos computacionais limitados, tais como celulares e PDA's. Devido às limitações de processamento e memória, os recursos da linguagem foram reduzidos nessa plataforma, bem como a Máquina Virtual Java (JOHNSON, 2007).

Visando fornecer os recursos necessários ao desenvolvedor, mas tendo em vista as limitações computacionais, foram criadas as API's, responsáveis por prover recursos específicos e que são utilizadas pelo programador apenas quando necessárias.

Seguindo a ideia da tecnologia Java, a plataforma JME tornou o software escrito independente do tipo de dispositivo móvel, sendo dividida em camadas, como mostra a Figura 2.

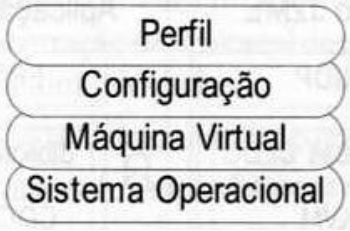

Figura 2 **-** Camadas da arquitetura JME.
Fonte: JOHNSON, 2007, p. 29.

A primeira camada é a *Kilobyte Virtual Machine* (KVM), máquina virtual da plataforma JME, desenvolvida para dispositivos com baixo poder de processamento. Acima desta existe a camada de configuração, responsável por fornecer as classes, bibliotecas e API's necessárias ao dispositivo móvel. E a última camada é a de perfil, que possui API's mais específicas, utilizadas pelo desenvolvedor para adaptar o aplicativo ao dispositivo em questão (JOHNSON, 2007).

1.2.1 **C**onfiguração

A camada de configuração tem o objetivo de fornecer as bibliotecas básicas necessárias à linguagem, estabelecendo um padrão de ambiente de execução entre os dispositivos com características semelhantes.

Em julho de 2011, o JME possui dois tipos de configurações, *Connected Limited Device Configuration* (CLDC) e a *Connected Device Configuration* (CDC), conforme ilustrado na Figura 3.

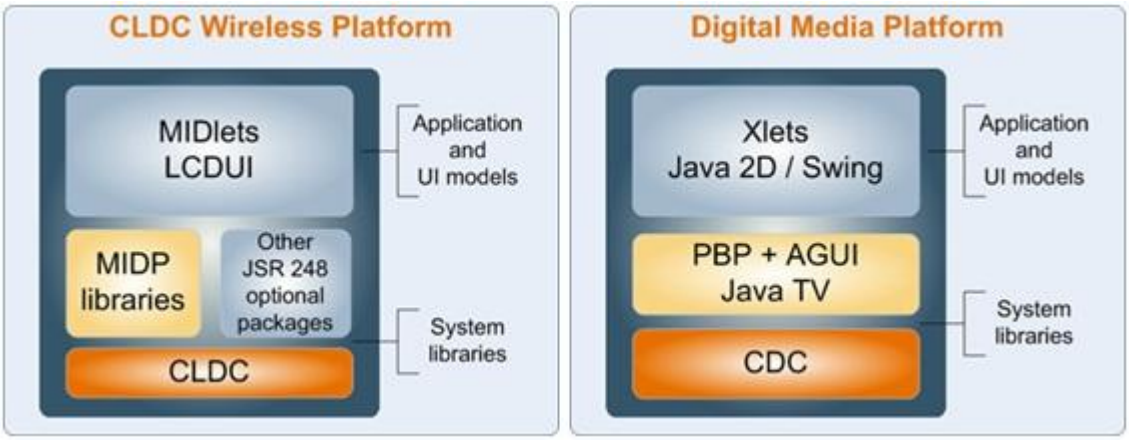

Figura 3 - Configurações da arquitetura JME.
Fonte: ORACLE, 2011.

1.2.1.1 *Connected Limited Device Configuration*

A CLDC, disponível nas versões 1.0 (JSR 30) e 1.1 (JSR 139), é um tipo de configuração da JME voltada para os dispositivos com grandes restrições de recursos computacionais. Suas classes estão contidas em quatro pacotes, como mostra a Tabela 1. Existem ainda pacotes opcionais que podem ser utilizados para prover funções especiais.

Pacote Java	Descrição
java.io	Tratamento de entrada e saída de dados usando *streams* (abstração de um canal de comunicação entre o programa e a fonte de dados).
java.lang	Classes básicas da linguagem Java.
java.util	Classes de utilidades genéricas, como algumas estruturas de dados e classes para manipulação de dados.
java.microedition.io	Exclusivo da JME; inclui as classes de conexões genéricas, que permitem manipular diversos tipos de conexões.

Tabela 1 - Classes da CLDC.
Fonte: JOHNSON, 2007, p. 31.

1.2.1.2 *Connected Device Configuration*

A CDC, disponível nas versões 1.0 (JSR 36) e 1.1.2 (JSR 218), foi desenvolvida para os dispositivos móveis com capacidades computacionais superiores aos dispositivos que utilizam a CLDC. Esse tipo de configuração possui uma máquina virtual própria chamada *CDC HotSpot Implementation* e suporta os pacotes listados na Tabela 2.

Pacotes Java			
java.io	java.lang	java.lang.ref	java.lang.reflect
java.math	java.net	java.security	java.security.cert
java.text	java.util	java.util.jar	java.util.zip
javax.microedition.io			

Tabela 2 - Pacotes suportados pela CDC 1.1.2
Fonte: RISCHPATER, 2008, p. 23.

1.2.2 **P**erfil

O perfil é um conjunto de API's específicas, como elementos de interface gráfica, persistência de dados e meios de comunicação, relacionadas ao tipo de dispositivo, associando-se assim a apenas uma configuração.

As API's incluídas em um perfil tornam a aplicação desenvolvida comum entre o conjunto de dispositivos que possuam o mesmo perfil, permitindo assim a portabilidade do software desenvolvido (JOHNSON, 2007).

Existem perfis para os dois tipos de configuração (CDC e CLDC), entre eles:

- *Mobile Information Device Profile* (MIDP): Disponível nas versões 1.0 (JSR 37), 2.0 (JSR 118), 2.1 (JSR 118) e 3.0 (JSR 271). É o perfil mais conhecido e voltado para a configuração CLDC (RISCHPATER, 2008). Está presente nos dispositivos de pequena capacidade computacional, com um mínimo de 512 KB de memória (volátil e não volátil), conectividade a algum tipo de rede sem fio e suporte a interface gráfica, como os celulares comuns (JOHNSON, 2007). Seu objetivo é aumentar a quantidade de recursos disponíveis, como gerência de sons, transações seguras, interface com o usuário, entre outros, aos dispositivos com esta configuração.
- *Foundation Profile* (FP): é o nível de perfil mais básico da configuração CDC, uma vez que está voltada para *hardwares* embutidos. Fornece

uma implementação de rede, porém não possui suporte à interface gráfica.

- *Personal Basis Profile* (PBP): está voltado para os dispositivos de configuração CDC que suportam um nível básico de apresentação gráfica, ou seja, que não possuem compatibilidade total com o *Abstract Window Toolkit* (AWT), como é o caso do menu de computadores de bordo e outros dispositivos eletrônicos.
- *Personal Profile* (PP): é o perfil de configuração CDC mais completo, que dá suporte a interface gráfica, *applets* e inclui biblioteca AWT completa (JOHNSON, 2007). É utilizado por dispositivos como os PDA's.

1.2.3 **P**acotes opcionais

Os pacotes opcionais oferecem aos dispositivos, API's padrões para tecnologias de mercado, como *web services*, gráficos 3D, NFC, entre outros. São opcionais já que nem todos os dispositivos têm recursos de *hardware* para implementar esses recursos.

Os fabricantes de dispositivos também desenvolvem API's próprias, utilizadas para prover recursos extras, porém restringindo-se aos seus dispositivos. Essas API's diferem dos pacotes opcionais por não fazerem parte de uma especificação da *Java Community Process* (JCP).

1.2.4 **L**imitações no uso das bibliotecas

As restrições de cada perfil de configuração mostradas na subseção 1.2.2 ocorrem dentro dos pacotes suportados pela tecnologia JME, cabendo ao desenvolvedor adaptar a aplicação para utilizar os pacotes suportados pelo perfil escolhido (JOHNSON, 2007).

1.3 JAVA *CARD*

Java *Card* é uma plataforma complementar às três principais divisões, que tem por objetivo permitir o desenvolvimento de pequenas aplicações, denominadas *applets*, para *smart cards* ou demais dispositivos com memória muito limitada, empregando a tecnologia Java.

Esta plataforma pode ser subdividida em três partes, que serão apresentadas a seguir.

.

1.3.1 **J**ava *Card Virtual Machine*

Assim como na plataforma JME, a Java *Card* também possui uma máquina virtual própria. A Java *Card Virtual Machine* (JCVM), diferente das demais máquinas virtuais Java, é dividida em duas partes: *off-card* (fora do cartão) e *on-card* (dentro do cartão), conforme mostra a Figura 4 (CHEN, 2000).

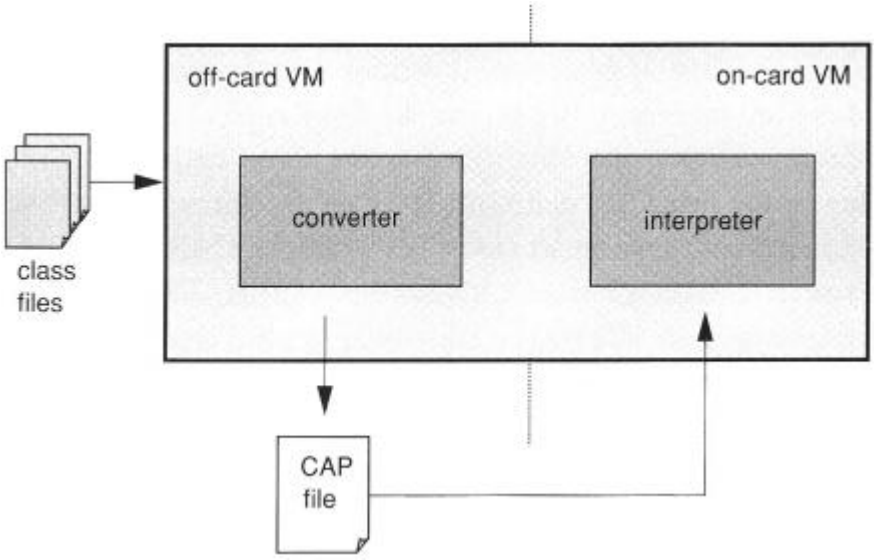

Figura 4 - Máquina Virtual Java Card.
Fonte: CHEN, 2000, p. 31.

A parte *off-card* roda em um computador, ou estação de trabalho, onde a aplicação está sendo desenvolvida, e tem como principal função gerar o *bytecode* que será interpretado pela parte *on-card*.

1.3.2 **J**ava *Card Runtime Environment*

O Java *Card Runtime Environment* (JCRE) é o mecanismo principal da tecnologia *Java Card*. Ele é o responsável por separar as *applets e* o *hardware* e sistema nativo do *smart card*, conforme ilustra a Figura 5, permitindo assim a portabilidade das aplicações (CHEN, 2000).

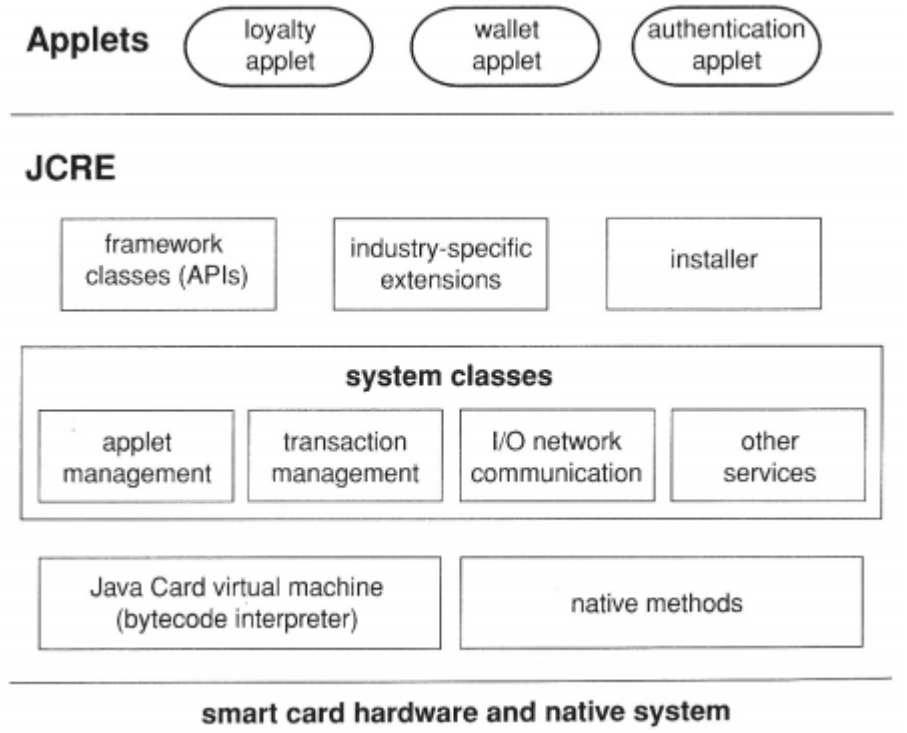

Figura 5 - Arquitetura interna de um *smart card.*
Fonte: CHEN, 2000, p. 36.

Por ser uma composição de todos os componentes de sistema Java *Card* que são executados dentro do cartão, inclusive a JCVM, o JCRE age como um

sistema operacional, promovendo todo o funcionamento da tecnologia no interior do *smart card* (CHEN, 2000).

1.3.3 **J**ava *Card* API

A API para Java *Card* contém as definições de classes, em conformidade com o *International Organization for Standardization* (ISO), para o desenvolvimento de aplicações que utilizem a plataforma.

1.4 CONCLUSÃO

Uma das principais características da linguagem Java é a JVM, responsável por proporcionar a portabilidade do código (JOHNSON, 2007). As plataformas JME e *Java Card*, apesar de voltadas para o desenvolvimento em dispositivos que apresentam recursos limitados, disponibilizam uma gama de funcionalidades que possibilitam ao programador desenvolver *softwares* de acordo com os requisitos computacionais do equipamento em questão.

2 NEAR FIELD COMMUNICATION

A tecnologia NFC obtém destaque no contexto dos dispositivos móveis devido ao seu princípio de comunicação. Este capítulo tem por objetivo abordar o padrão e apresentar a tecnologia e suas funcionalidades.

2.1 ESPECIFICAÇÃO

NFC é um padrão de comunicação de curto alcance, desenvolvido em uma parceria das empresas Philips e Sony e definido pelo *NFC Forum*. NFC utiliza tecnologia de ondas de rádio, na faixa de frequência de 13,56 MHz, para permitir conexões sem fios entre dispositivos. As conexões normalmente atingem a distância de 0 a 20 centímetros e a velocidade de transferência de dados varia de 106 kbps a 424 kbps (ATOJI, 2010; NFC FORUM, 2011a).

2.2 PROTOCOLOS NFC

O NFC pode implementar dois protocolos de comunicação próprios: NFCIP-1 e NFCIP-2, ilustrados na Figura 6 (SIMÕES, 2008).

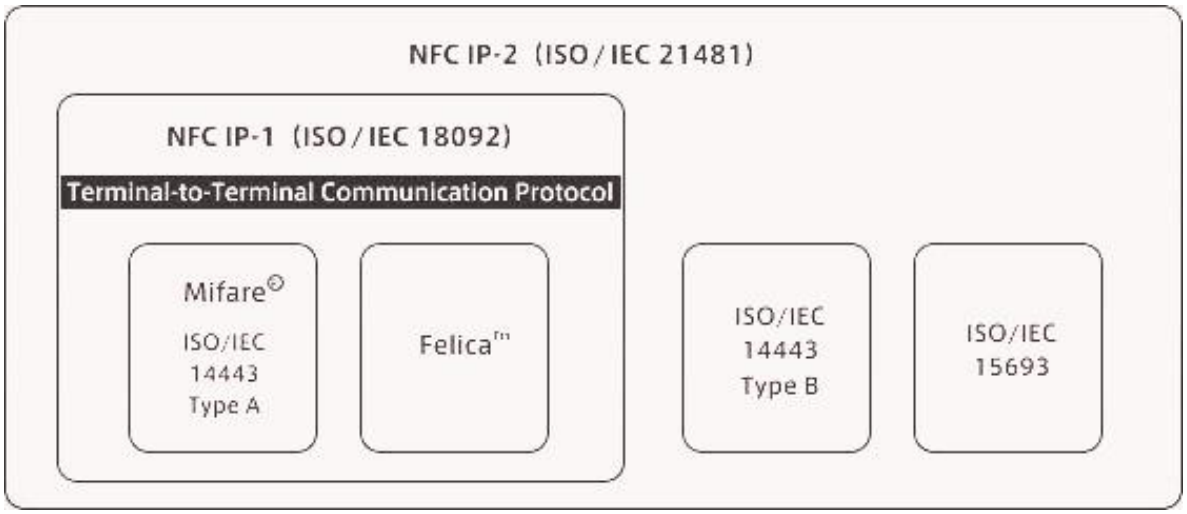

Figura 6 - Protocolos NFCIP-1 e NFCIP-2.
Fonte: PORTAL NFC, 2011.

2.2.1 ***N****ear Field Communication Interface and Protocol – 1*

O *Near Field Communication Interface and Protocol – 1* (NFCIP-1) contém especificações para o processo de comunicação, bem como a definição dos modos de comunicação que um dispositivo NFC pode assumir: ativo e passivo (FERRAZ FILHO, 2010).

No modo ativo, os dois dispositivos envolvidos na comunicação geram um campo de rádio frequência para transmissão de dados. Já no modo passivo, apenas um dos equipamentos envolvidos, o iniciador, gera um campo de rádio freqüência capaz de alimentar eletricamente o segundo equipamento, o alvo, permitindo a comunicação (LEVANDOSKI *et al.*, 2011).

2.2.2 ***N****ear Field Communication Interface and Protocol – 2*

O *Near Field Communication Interface and Protocol – 2* (NFCIP-2) apresenta os mecanismos que permitem a detecção e posterior seleção de um dos modos de operação disponíveis (FERRAZ FILHO, 2010).

2.3 MODOS DE OPERAÇÃO

A tecnologia NFC permite três modos de comunicação distintos, ilustrados na Figura 7 e descritos a seguir.

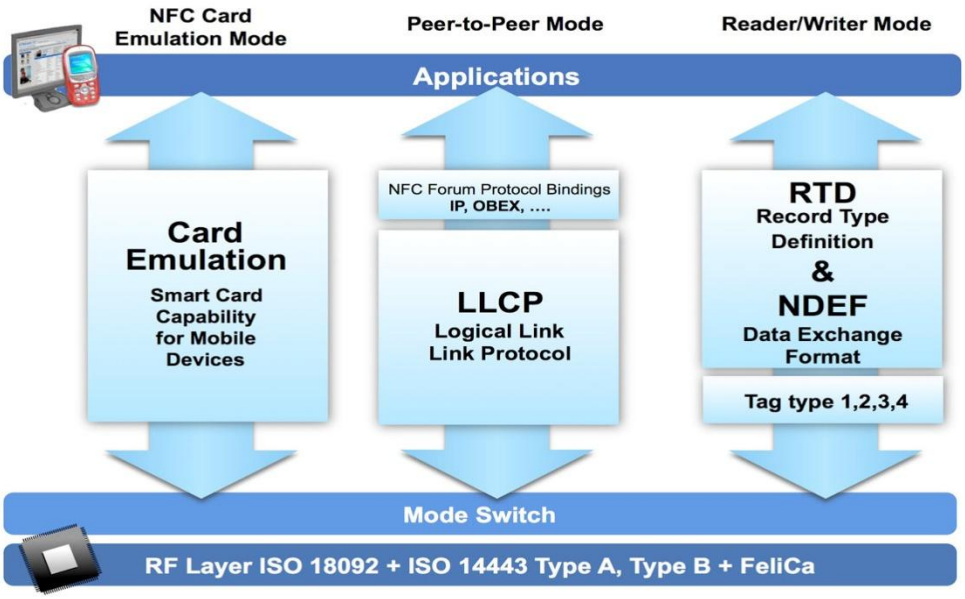

Figura 7 - Modos de operação NFC
Fonte: NFC Forum, 2011b.

2.3.1 ***P****eer-to-peer*

Este modo, padronizado no ISO 18092, permite que uma conexão bidirecional, ponto a ponto, seja estabelecida entre o dispositivo iniciador e alvo para troca de dados.

2.3.2 **L**eitor / Gravador

Padronizado no ISO 15693, possibilita a um dispositivo NFC ler e gravar dados em uma etiqueta NFC passiva. A etiqueta NFC pode conter diversos tipos de informações, como, por exemplo, uma URL de um site, fazendo com que o dispositivo iniciador promova uma conexão *wi-fi* para o endereço lido.

2.3.3 **E**mulador de cartão

Neste modo, padronizado no ISO 14443, o dispositivo NFC é capaz de operar como uma etiqueta NFC passiva, permitindo sua interpretação por qualquer leitor. Diversos tipos de cartões podem ser emulados nesse modo, permitindo a substituição de muitos cartões diferentes por apenas um dispositivo.

2.4 PRINCÍPIO DE COMUNICAÇÃO

A tecnologia NFC utiliza a indução magnética para permitir a comunicação entre dois dispositivos próximos.

No modo leitor/gravador, apresentado na subseção 2.3.2, um leitor NFC, que é ativo, uma vez que é capaz de produzir sua própria energia, gera uma corrente alternada que ao passar por um indutor interno, cria um campo magnético ao seu redor. Ao aproximar uma etiqueta passiva (incapaz de gerar energia) desse campo gerado pelo leitor, uma tensão alternada aparecerá na etiqueta, podendo assim alimentar seus componentes eletrônicos internos, permitindo assim a comunicação através de modulação de sinais. Esse processo pode ser visualizado na Figura 8.

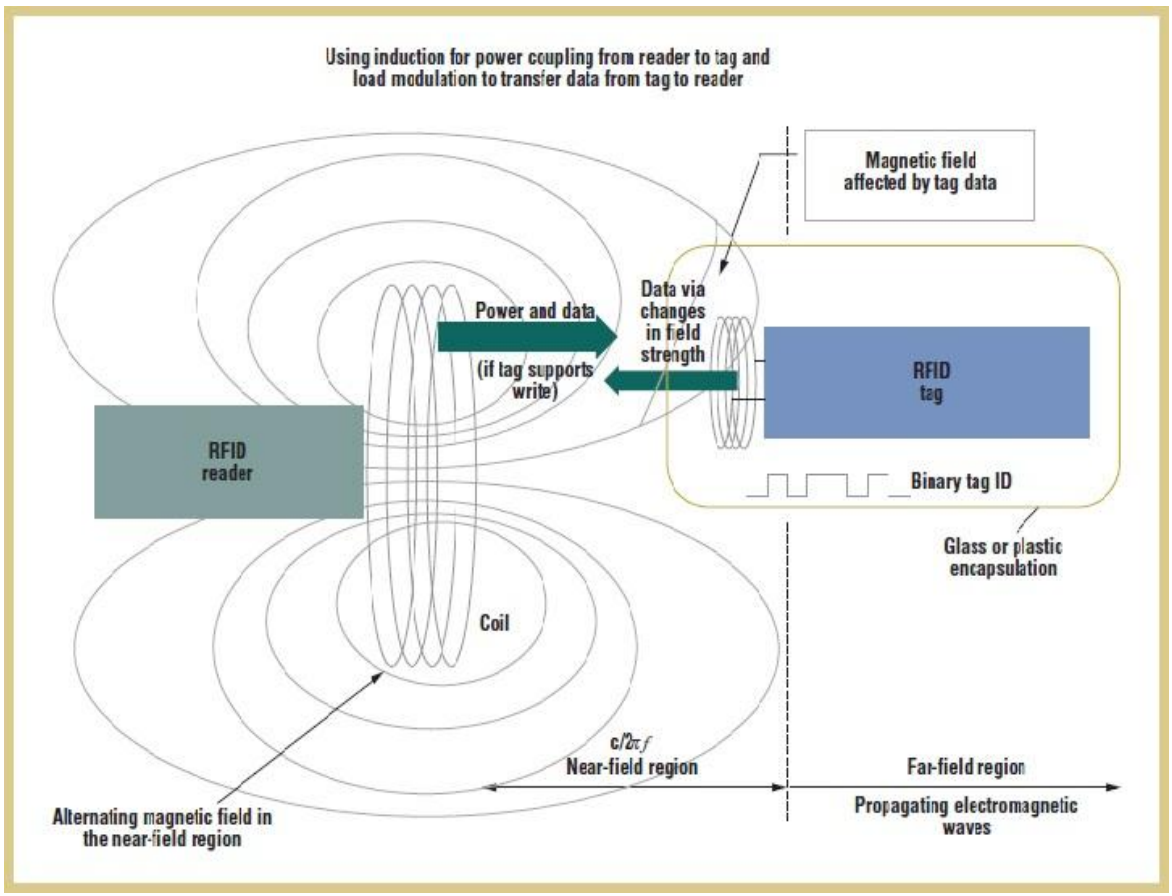

Figura 8 - Comunicação entre um leitor e uma *tag* passiva.
Fonte: NEAR COMMUNICATIONS, [201?].

Já no modo *peer-to-peer*, apresentado na subseção 2.3.1, onde ambos os dispositivos envolvidos na comunicação são ativos, a comunicação ocorre da mesma forma, porém o dispositivo alvo não necessita do campo magnético do dispositivo iniciador para alimentar seu circuito interno.

2.5 *CONTACTLESS COMMUNICATION* API

Descrita no JCP pela JSR 257, a *Contactless Communication* é uma API opcional da plataforma JME e tem por objetivo permitir a comunicação com dispositivos que necessitam de proximidade física, sem contato, para troca de informações (MOURÃO JÚNIOR, 2008).

Disponível na versão 1.0, essa API dá suporte a quatro tipos de dispositivos (MOURÃO JÚNIOR, 2008):

- *Smart cards* e *tags* RFID que utilizam o formato de dados *NFC Data Exchange Format* (NDEF) para comunicação;
- *Tags* RFID que não utilizam o formato de dados NDEF para comunicação;
- *Smart cards* que utilizam mensagens *Application Protocol Data Unit* (APDU) para comunicação;
- Etiquetas de código de barras.

Além de permitir que o dispositivo móvel atue no modo emulador de cartão, apresentado na subseção 2.3.3, conforme observado na Figura 9.

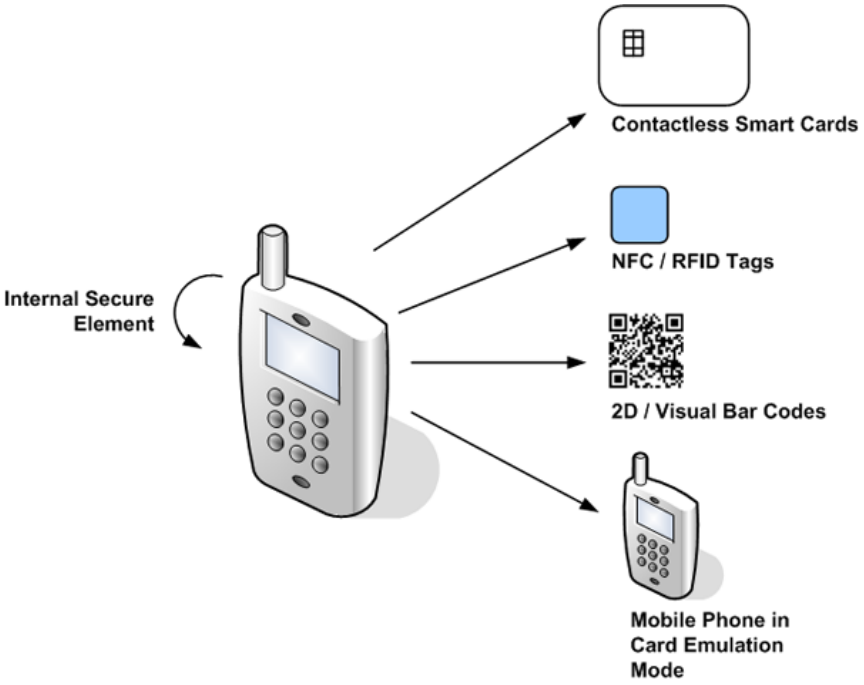

Figura 9 - Suporte oferecido pela *Contactless Communication* API.
Fonte: HOPKINS, 2009.

A JSR 257 possui o pacote Java principal *javax.microedition.contactless* e mais quatro sub-pacotes, conforme apresenta a Tabela 3.

Pacote Java	Interfaces	Classes
javax.microedition.contactless	TagConnection TargetListener TargetProperties TransactionListener	DiscoveryManager TargetType
javax.microedition.contactless.ndef	NDEFRecordListener NDEFTagConnection	NDEFMessage NDEFRecord NDEFRecordType
javax.microedition.contactless.rf	PlainTagConnection	
javax.microedition.contactless.sc	ISO14443Connection	
javax.microedition.contactless.visual	ImageProperties VisualTagConnection	SymbologyManager

Tabela 3 - Pacotes Java da JSR 257.
Fonte: ORTIZ, 2008.

Esta API disponibiliza um mecanismo capaz de procurar por dispositivos compatíveis que estejam dentro da área de atuação, de forma unificada, ou seja, pode-se identificar um dispositivo NDEF da mesma forma que um RFID ou uma etiqueta de código de barras (MOURÃO JÚNIOR, 2008).

Por se tratar de dispositivos propensos a entrar e sair do raio de comunicação com frequência, uma vez que trabalham a curta distância, a API fornece o método *DiscoveryManager.addTargetListener(TargetListener listener, TargetType targetType)*, apresentado na Figura 10 (MOURÃO JÚNIOR, 2008). Este método é responsável por registrar um tipo de dispositivo para que seja identificado pela aplicação quando entrar no campo de atuação.

```
public NFCDiscovery() {
    try {
        //Registra listeners para as tags
        DiscoveryManager.getInstance ().addTargetListener (this, TargetType.NDEF_TAG);
        DiscoveryManager.getInstance ().addTargetListener (this, TargetType.RFID_TAG);
        DiscoveryManager.getInstance ().addTargetListener (this, TargetType.ISO14443_CARD);
      } catch (ContactlessException ce) {
        System.out.println("Error in adding listeners: " + ce.getMessage());
      } catch (Exception ce) {
        System.out.println("Error in adding listeners: " + ce.getMessage());
    }
}
```

Figura 10 - Implementação do registro de dispositivos.
Fonte: Do autor, 2011.

Uma vez registrado, o método *targetDetected(TargetProperties[] properties)*, mostrado na Figura 11, notifica que um dispositivo foi encontrado, e através do objeto *properties*, pode-se obter a URL para conexão com o dispositivo detectado.

```
public void targetDetected(TargetProperties[] targetProprties) {
    System.out.println("Tag NFC detectada!");
    TargetProperties target = null;
    target = targetProprties[0];
    //Imprime URL de conexão
    System.out.println("URL:"+target.getUrl(target.getConnectionNames()[0]));
    ...
}
```

Figura 11 - Implementação do método *targetDetected.*
Fonte: Do autor, 2011.

Com a URL de conexão, devido a utilização do *Generic Connection Framework* (GCF) pela JSR 257, basta invocar o método *Connector.open(String url)*, com a URL obtida como parâmetro, para estabelecer a conexão com o dispositivo, conforme pode ser observado na Figura 12.

```
...
    String URLBlue = target.getUrl(target.getConnectionNames()[0]);
    try {
        //Abre uma conexão utilizando a URL obtida
        this.conn = (StreamConnection)Connector.open(URLBlue, Connector.READ_WRITE);
        this.aplicacaoForm.setConnection(conn);
        NFCBlueMIDlet.getInstance().setCurrent(aplicacaoForm);
    } catch (IOException ex) {
        ex.printStackTrace();
    }
...
```

Figura 12 - Estabelecendo uma conexão com base na URL obtida.
Fonte: Do autor, 2011.

2.6 CONCLUSÃO

Com o suporte da *Contactless Communication* API, a tecnologia NFC pode ser integrada aos dispositivos móveis dando a estes dispositivos uma nova gama de funcionalidades, visto todas as possibilidades promovidas por essa tecnologia e apresentadas nesse capítulo.

3 APLICAÇÕES NFC EM DISPOSITIVOS MÓVEIS

Este capítulo apresenta algumas áreas onde a tecnologia NFC pode atuar e suas vantagens na utilização conjunta com dispositivos móveis.

3.1 ÁREAS DE ATUAÇÃO

Em Londres, na Inglaterra, já são feitos testes para implantação da tecnologia NFC no sistema de metrô, permitindo a compra de passagens com a aproximação do celular às catracas, sistema esse que já está em funcionamento na cidade de Tóquio, no Japão (TEIXEIRA JÚNIOR, 2011; HECKE, 2011). Este é um exemplo das áreas nas quais as aplicações NFC podem atuar, conforme ilustrado na Figura 13.

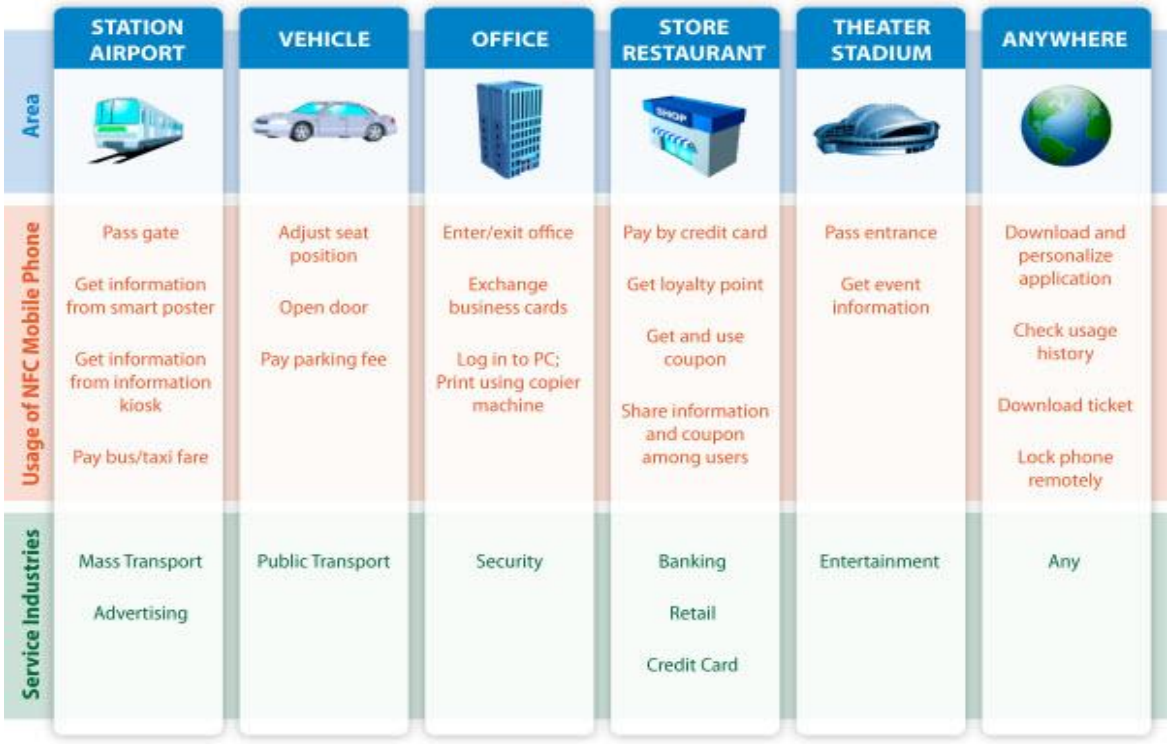

Figura 13 - Áreas em que a tecnologia NFC pode ser aplicada.
Fonte: NFC FORUM, 2011c.

3.2 INTEGRAÇÃO COM DISPOSITIVOS MÓVEIS

O número de dispositivos móveis em uso no mundo já ultrapassou a marca de 5 bilhões (O GLOBO, 2011). Esses dispositivos apresentam um maior poder computacional em relação aos *smart cards*, além do acesso à rede de telefonia móvel e às tecnologias de comunicação sem fio, como *wi-fi* ou *bluetooth.* Essas características, aliadas a sua grande difusão no mundo, tornam atraente a integração da tecnologia NFC a esses dispositivos (FERRAZ FILHO, 2010).

Capaz de atuar no modo emulador de cartão (apresentado na subseção 2.3.3) com o suporte da *Contactless Communication* API, uma das possibilidades de aplicações que surgem é a utilização do telefone móvel como concentrador das funcionalidades providas por cartões inteligentes de débito e crédito, de transporte e outros. Uma vantagem dessa utilização, além da possibilidade de se ter todos cartões em apenas um aparelho, é a capacidade dos dispositivos móveis de armazenar informações pessoais de seu proprietário, como histórico de compras por exemplo, o que permite que maiores interações entre fornecedor e usuário possam ocorrer (TEIXEIRA JÚNIOR, 2011).

Aproveitando as tecnologias de comunicação disponíveis nos dispositivos móveis, novas possibilidades de aplicações surgem. Uma *tag* NFC pode conter uma URL de conexão *wi-fi* ou *bluetooth*, o que pode acelerar o processo de conexão. Sendo assim, ações de *marketing* podem utilizar *tags* NFC em cartazes de propaganda, permitindo que o usuário, ao aproximar seu telefone móvel, tenha acesso ao site que apresenta as características detalhadas do produto em questão, ou ainda possibilitar ao usuário baixar um aplicativo exclusivo daquele produto, via *bluetooth*, apenas aproximando seu celular da propaganda (FERRAZ FILHO, 2010).

Aplicações de acesso também são possíveis com NFC. Uma fechadura que utilize a tecnologia NFC permitiria a possibilidade de abrir a porta de casa, ou do carro, apenas com a aproximação do celular (RICHTEL E KOPITOFF, 2011).

Porém, o quesito segurança deve ser levado em conta. Aplicações financeiras devem estar protegidas quanto a interceptações na comunicação e ainda a uma possível perda ou roubo do telefone móvel. Já as aplicações que

utilizam URL’s de *tags* para promover uma conexão secundária, de internet por exemplo, devem levar em conta a possibilidade de direcionamento para sites maliciosos.

3.3 CONCLUSÃO

Conforme pode ser visto, a integração entre a tecnologia NFC e os dispositivos móveis oferece um novo horizonte de aplicações que podem ser desenvolvidas. Alguns pontos ainda precisam ser melhorados, como as questões de segurança apontadas na subseção 3.2. Mas essa união é possível, vem sendo aperfeiçoada e já se tornou realidade em algumas partes do mundo, conforme apresentado na subseção 3.1.

4 ESTUDO DE CASO: NFCBLUE

Este capítulo tem a finalidade de apresentar o estudo de caso desenvolvido como complemento ao presente trabalho.

4.1 APLICAÇÃO

Durante o processo de pesquisas foi desenvolvida, em JME, a aplicação NFCBlue, cujo código fonte é apresentado no Anexo A.

O objetivo da aplicação é ler uma *tag* NFC e promover uma conexão secundária *bluetooth* com a URL obtida. Ao executar a aplicação o usuário deve realizar um *login*, para então, se validado, o aplicativo permitir a leitura de uma *tag* NFC que fornecerá uma URL, utilizada para estabelecer uma conexão *bluetooth*.

A aplicação foi dividida em cinco classes denominadas *NFCBlueMIDlet.java*, *LoginForm.java*, *MainMenu.java*, *AplicacaoForm.java* e *NFCDiscovery.java*.

O aplicativo funciona apenas simulado no computador, já que a *Contacteless Communication* API só é suportada por alguns telefones móveis, além de ser necessário um aparelho que integre um leitor NFC para o funcionamento esperado, como os modelos da Nokia linha NFC 6212 e 6131, por exemplo (NOKIA DEVELOPER, [2009?]).

4.2 DIAGRAMAS UML

As Figuras 14, 15 e 16 apresentam, respectivamente, os diagramas de casos de uso, classes e sequência da aplicação desenvolvida.

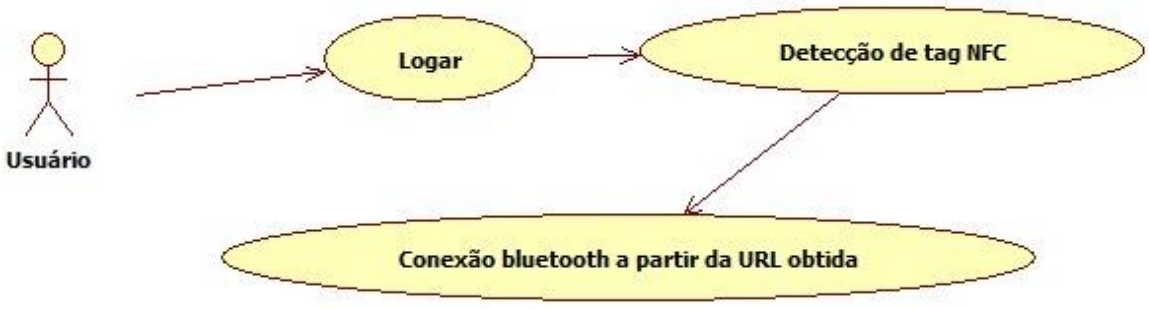

Figura 14 - Diagrama de casos de uso da aplicação NFCBlue.
Fonte: Do autor, 2011.

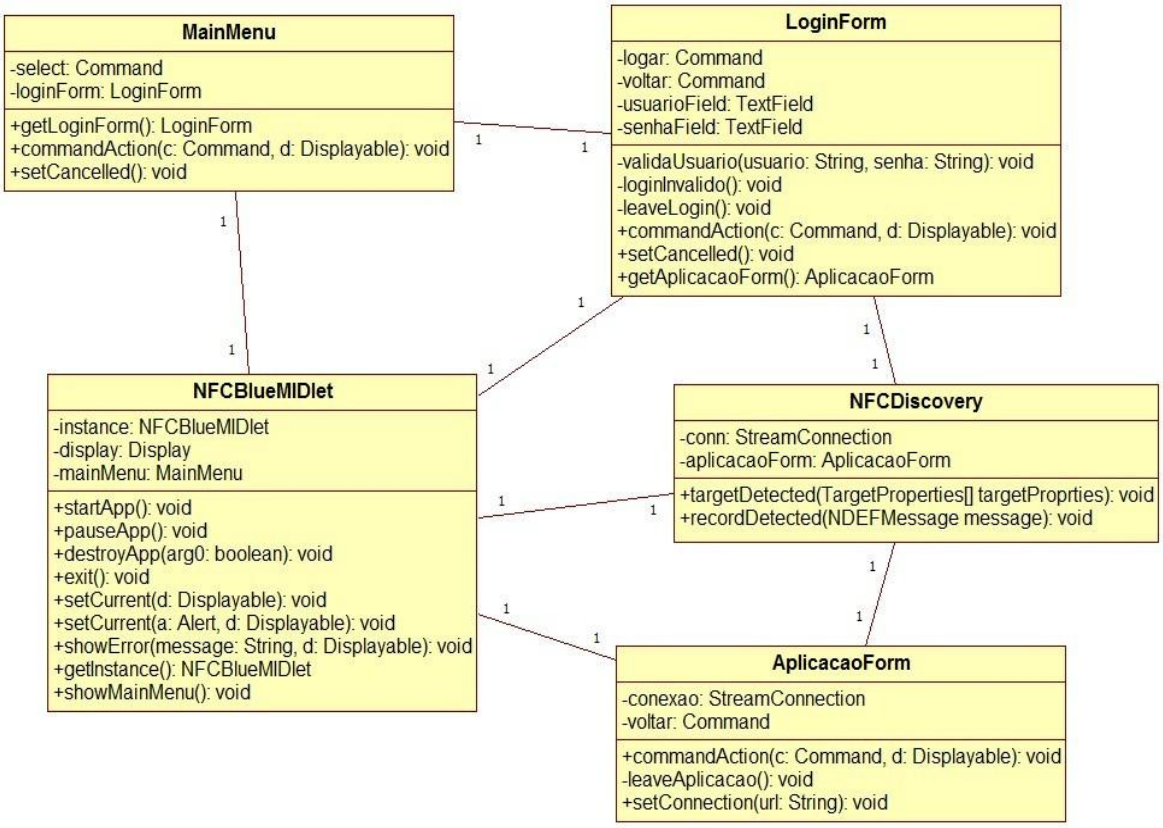

Figura 15 - Diagrama de classes da aplicação NFCBlue.
Fonte: Do autor, 2011.

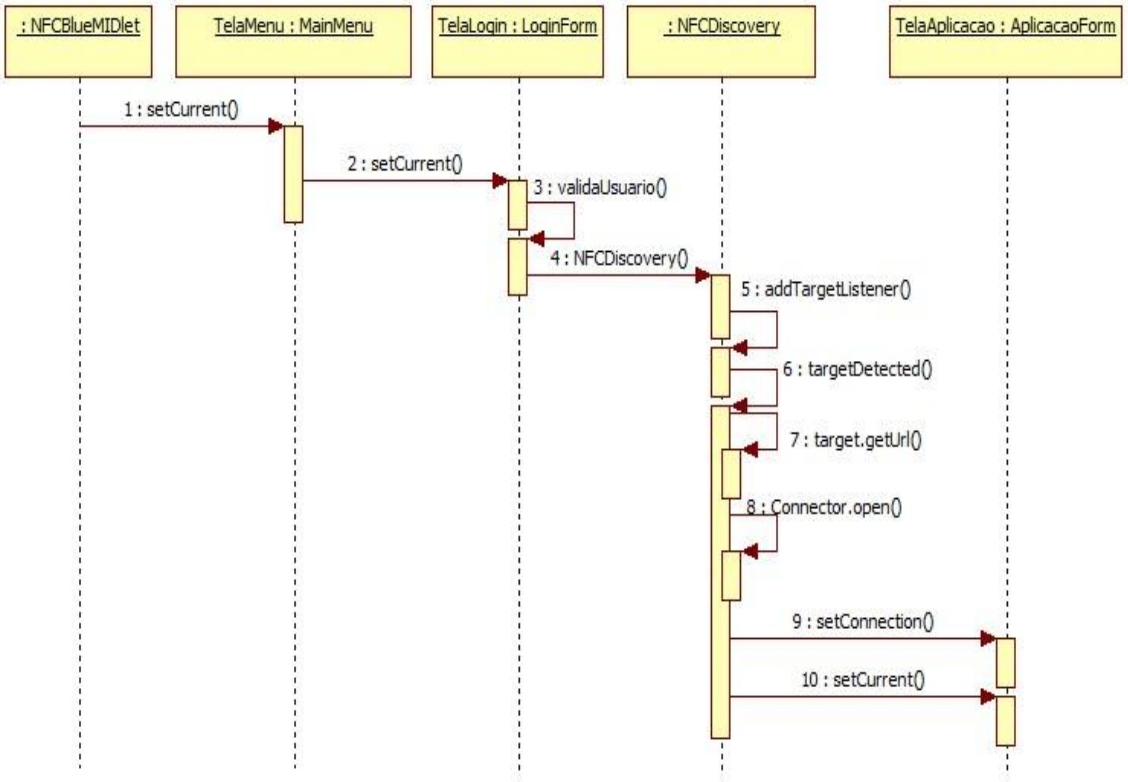

Figura 16 - Diagrama de sequência da aplicação NFCBlue.
Fonte: Do autor, 2011.

4.3 CONCLUSÃO

Apesar de funcionar apenas simulada no computador devido a restrições da API utilizada, a aplicação NFCBlue permitiu aplicar, em um contexto real, a integração entre a plataforma Java e a tecnologia NFC. Durante seu desenvolvimento foi possível aplicar conceitos aprendidos durante as pesquisas e no decorrer do curso de especialização.

CONSIDERAÇÕES FINAIS

A crescente utilização de dispositivos móveis no mundo motiva o desenvolvimento de novas tecnologias capazes de interagir com esses equipamentos. A linguagem de programação Java por sua vez, devido a sua portabilidade, se tornou uma das principais linguagens para o desenvolvimento de aplicações voltadas a dispositivos móveis, através da plataforma JME. Sendo assim, programadores capazes de dominar essa linguagem ganham destaque na área de dispositivos móveis, que conforme apresentado, está em constante evolução.

O presente trabalho teve o objetivo de apresentar uma pesquisa sobre a tecnologia NFC, apresentando o princípio de comunicação desse padrão e as possibilidades que surgem ao integrar essa tecnologia aos dispositivos móveis.

Nesse ponto, a plataforma Java ganha destaque ao promover, por meio da *Contactless Communication* API, o desenvolvimento de aplicações NFC para dispositivos móveis, aumentando assim as possibilidades de utilização desses

equipamentos.

Para alcançar o objetivo proposto, o desenvolvimento da aplicação NFCBlue, com foco na plataforma JME, permitiu aplicar, em uma situação real, a integração da plataforma Java à tecnologia NFC, no contexto dos dispositivos móveis. Ao permitir a leitura de uma *tag* NFC, o aplicativo mostra que esta integração é viável e disponibiliza um novo horizonte de aplicações que podem surgir com base nessas tecnologias.

O desenvolvimento do presente trabalho permitiu uma maior compreensão dos conceitos apresentados durante o curso de especialização a cerca da linguagem Java e suas divisões, e as possibilidades de integração dessa linguagem com outras tecnologias, como o NFC por exemplo.

Para trabalhos futuros, alguns pontos podem ser abordados para tornar a pesquisa apresentada, adequada para uso em aplicações práticas, como por exemplo:

- Aperfeiçoar a aplicação desenvolvida para que se torne prática em situações reais;
- Pesquisar a cerca de mecanismos de segurança que podem ser implementados;
- Utilizar um telefone móvel que integre a tecnologia NFC;
- Desenvolver aplicativos para os sistemas operacionais emergentes em *smartphones* como o Android;

REFERÊNCIAS

ANATEL. *Brasil fecha abril com 212,6 milhões de acessos móveis*. Disponível em: <http://www.anatel.gov.br/Portal/exibirPortalNoticias.do?acao=carregaNoticia&codigo=22734>. Acesso em: 24 jul. 2011.

ATOJI, R. I. *Bluetooth e NFC*: estudo de caso. 58 f. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) – Universidade de São Paulo, São Paulo. 2010.

CHEN, Z. *Java Card Technology for Smart Cards.* Architecture and Programmer's Guide. Massachusetts: Pearson Prentice Hall, 2000. 400 p.

DEITEL, H. M..; DEITEL, P. J. *Java Como Programar.* 8. ed. São Paulo: Pearson Prentice Hall, 2010. 1114 p.

ECMA INTERNATIONAL. *Standard ECMA-352.* Near Field Communication Interface and Protocol-2 (NFCIP-2). Genebra: ECMA, 2003. Disponível em: <http://read.pudn.com/downloads134/doc/comm/572238/NFC%E5%8D%8F%E8%AE%AE%E5%8F%8A%E6%B5%8B%E8%AF%95%E6%96%B9%E6%B3%95/Ecma-352_Near%20Field%20Communication%20Interface%20and%20Protocol%20-2%20(NFCIP-2).pdf>. Acesso em: 15 jul. 2011.

FERRAZ FILHO, O. L. S. *Comunicação NFC (Near Field Communication) entre Dispositivos Ativos*. 100 f. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Universidade Federal de Pernambuco, Recife. 2010.

HECKE, C. *Onde e como a tecnologia NFC está sendo aplicada.* 2011. Disponível em: <http://www.tecmundo.com.br/8173-onde-e-como-a-tecnologia-nfc-esta-sendo-aplicada.htm>. Acesso em: 24 jul. 2011.

HOPKINS, B. *Faster Data Transfer With Bluetooth and Contactless Communication.* 2009. Disponível em: <http://www.oracleimg.com/technetwork/articles/javame/nfc-bluetooth-142337.html>**.** Acesso em: 20 jun. 2011.

JOHNSON, T. M. *Java para Dispositivos Móveis.* Desenvolvendo Aplicações com J2ME. São Paulo: Novatec, 2007. 334 p.

LEVANDOSKI, F.; DIAS, V. F.; CHRIST, V. R.; MARQUES, V. H. *O método de comunicação NFC (Near Field Communication) e sua aplicação no processo de pagamento através de dispositivos móveis.* São Leopoldo: UNISINOS, 2011. Disponível em: <http://www.faustolevandoski.com.br/downloads/Near%20Field%20Communication.pdf>. Acesso em: 09 jul. 2011.

MOURÃO JÚNIOR, E. JSR 257: Contactless Communication API. Trabalhando com RFID em dispositivos móveis. *Java Magazine*, Rio de Janeiro, Ed. 56, Ano 6, p. 62-66, 2008.

NEAR COMMUNICATIONS. *Tecnologia RFID.* [201?]. Disponível em: <https://sites.google.com/site/nearcommunications/tecnologia-rfid>. Acesso em: 20 jul. 2011.

NFC FORUM. *About NFC Technology.* 2011a. Disponível em: <http://www.nfc-forum.org/resources/faqs/>. Acesso em: 24 jul. 2011.

NFC FORUM. *NFC and Interoperability.* 2011b. Disponível em: <http://www.nfc-forum.org/aboutnfc/interop/>. Acesso em: 24 jul. 2011.

NFC FORUM. NFC in action. 2011c. Disponível em: <http://www.nfc-forum.org/aboutnfc/nfc_in_action/>. Acesso em: 23 jul. 2011.

NFC PORTAL. *About NFC*. 2011. Disponível em: <http://www.nfc-world.com/en/about/>. Acesso em: 24 jul. 2011.

NOKIA DEVELOPER. *Introduction to NFC on the Nokia 6212 and Nokia 6131*. [2009?]. Disponível em: <http://www.developer.nokia.com/Community/Wiki/Introduction_to_NFC_on_the_Nokia_6212_and_Nokia_6131>. Acesso em: 27 jul. 2011.

O GLOBO. *Número de internautas já passa de 2 bilhões, afirma ONU.* Rio de Janeiro. 2011. Disponível em: <http://oglobo.globo.com/tecnologia/mat/2011/01/26/numero-de-internautas-ja-passa-de-2-bilhoes-afirma-onu-923610235.asp>. Acesso em: 24 jul. 2011.

ORACLE. *Java ME Technology Overview.* 2011. Disponível em: <http://www.oracle.com/technetwork/java/javame/java-me-overview-402920.html>. Acesso em: 11 jul. 2011.

ORTIZ, C. E. *An Introduction to Near-Field Communication and the Contactless Communication API*. 2008. Disponível em: <http://java.sun.com/developer/technicalArticles/javame/nfc/#3>. Acesso em: 24 jul. 2011.

RISCHPATER, R. *Beginning Java ME Platform.* Nova Iorque: Apress, 2008. 569 p.

RICHTEL, M.; KOPYTOFF, V. G. *Tools of Entry, No Need for a Key Chain*. Nova Iorque, 2011. Disponível em: <http://www.nytimes.com/2011/07/04/technology/04key.html?pagewanted=all>. Acesso em: 25 jul. 2011.

SIMÕES, D. B. M. S. *Sistema de Fidelização sobre NFC (Near Field Communication).* 77 f. Trabalho de Conclusão de Curso (Mestrado em Engenharia Informática e de Computadores) – Universidade Técnica de Lisboa, Portugal. 2008.

TEIXEIRA JÚNIOR, S. O celular vai virar carteira. *Exame*, São Paulo, Ed. 991, Ano 45, n. 8, p. 114-118, mai. 2011.

UCHÔA, G. *A Tecnologia Java.* 2010. Disponível em: <http://gustavouchoa.blogspot.com/2010/05/tecnologia-java.html>. Acesso em: 12 jul. 2011.

UOL. *Venda de smartphones deve superar a de computadores no Brasil em 2011, diz IDC*. 2011. Disponível em: <http://tecnologia.uol.com.br/ultimas-noticias/redacao/2011/02/17/venda-de-smartphones-deve-superar-a-de-computadores-no-brasil-em-2011-diz-idc.jhtm>. Acesso em: 24 jul. 2011.

APÊNDICES

APÊNDICE A – CÓDIGO FONTE DA APLICAÇÃO NFCBlue

A.1 NFCBlueMIDlet.java

```
import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * @author Bernardo
 */
public class NFCBlueMIDlet extends MIDlet {
    private static NFCBlueMIDlet instance;
    private Display display;
    private MainMenu mainMenu;

    public NFCBlueMIDlet() {
        display = Display.getDisplay(this);
        instance = this;
    }
```

```
    public void startApp() throws MIDletStateChangeException {
        setCurrent(this.mainMenu = new MainMenu());
    }

    public void pauseApp() {
    }

    public void destroyApp(boolean arg0) throws
    MIDletStateChangeException{
        notifyDestroyed();
    }

     public void exit() {
         try{
            this.destroyApp(true);
         }catch (MIDletStateChangeException e){
            e.printStackTrace();
         }
     }

     public void setCurrent(Displayable d){
         this.display.setCurrent(d);
     }

     public void setCurrent(Alert a ,Displayable d){
         this.display.setCurrent(a, d);
     }

     public void showError(String message, Displayable d){
        Alert alert = new Alert("Erro", message, null, AlertType.ERROR);
        alert.setTimeout(2000);
        display.setCurrent(alert, d);
     }

     public static NFCBlueMIDlet getInstance(){
         return instance;
     }

     public void showMainMenu(){
         this.setCurrent(this.mainMenu);
     }
}
```

A.2 MainMenu.java

```
package TCC;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.List;
/**
 *
 * @author Bernardo
 */
public class MainMenu extends List implements CommandListener{
    private final String SELEC_COMMAND_NAME = "Selec";
    private LoginForm loginForm;
    private boolean cancelled;

    public MainMenu(){
        super ("Inicio", List.IMPLICIT);

        this.append("Login", null);
        this.append("Sair", null);

        this.addCommand(new Command(SELEC_COMMAND_NAME, Command.OK, 1));
        this.setCommandListener(this);
        this.loginForm = new LoginForm();
        this.cancelled = false;
    }

    public void commandAction(Command c, Displayable d) {
        try {
            switch (this.getSelectedIndex()) {
                case 0:

                  NFCBlueMIDlet.getInstance().setCurrent(this.loginForm);
                  break;
                default:
                    NFCBlueMIDlet.getInstance().exit();
                break;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public LoginForm getLoginForm() {
        return loginForm;
```

```
    }

    public void setCancelled() {
        this.cancelled = true;
    }

}
```

A.3 LoginForm.java

```
package TCC;

import java.io.IOException;

import javax.bluetooth.RemoteDevice;

import javax.microedition.io.StreamConnection;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;
import javax.microedition.lcdui.TextField;
import javax.microedition.lcdui.Form;

/**
 *
 * @author Bernardo
 */
public class LoginForm extends Form implements CommandListener {
    private Command logar;
    private Command voltar;
    private TextField usuarioField;
    private TextField senhaField;
    private AplicacaoForm aplicacaoForm;
    private NFCDiscovery discoveryForm;
    private boolean cancelled;
    private StreamConnection conn;
    private RemoteDevice remoteDevice;

    public LoginForm(){
        super("Login");
        this.usuarioField   =   new   TextField("Usuário",   null,   20,
                                                TextField.ANY);
        this.append(this.usuarioField);
        this.senhaField    =    new    TextField("Senha",    null,    20,
                                                TextField.PASSWORD);
```

```
        this.append(this.senhaField);

        this.addCommand(this.logar  =  new  Command("Logar",  Command.OK,
                                                    1));
        this.addCommand(this.voltar = new Command("Voltar", Command.BACK,
                                                   1));
        this.aplicacaoForm = new AplicacaoForm();
        this.setCommandListener(this);
    }
    private  void  validaUsuario(String  usuario,  String  senha)  throws
      IOException{
        if ((usuario.equals("Bernardo")) && (senha.equals("12345"))){
            discoveryForm = new NFCDiscovery();
        } else {
            loginInvalido();
        }
    }

    private void loginInvalido(){
        this.usuarioField.setString("");
        this.senhaField.setString("");
    }

    private void leaveLogin() {
        this.deleteAll();
        this.append(this.usuarioField);
        this.append(this.senhaField);
        NFCBlueMIDlet.getInstance().showMainMenu();
    }

    public void commandAction(Command c, Displayable d) {
        if (c == this.voltar) {
            this.leaveLogin();
        } else if (c == this.logar){
            try {
                validaUsuario(this.usuarioField.getString(),
                              this.senhaField.getString());
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    public AplicacaoForm getAplicacaoForm() {
        return aplicacaoForm;
    }

    public void setCancelled() {
        this.cancelled = true;
    }

}
```

A.4 NFCDiscovery.java

```
package TCC;

import java.io.IOException;

import javax.microedition.contactless.ndef.*;
import javax.microedition.contactless.*;
import javax.microedition.io.Connector;
import javax.microedition.io.StreamConnection;

/**
 *
 * @author Bernardo
 */
public class NFCDiscovery implements TargetListener, NDEFRecordListener {
    private AplicacaoForm aplicacaoForm = new AplicacaoForm();
    private StreamConnection conn;

        public NFCDiscovery() {

            try {
               DiscoveryManager.getInstance  ().addTargetListener  (this,
                                               TargetType.NDEF_TAG);
               DiscoveryManager.getInstance  ().addTargetListener  (this,
                                               TargetType.RFID_TAG);
               DiscoveryManager.getInstance  ().addTargetListener  (this,
                                               TargetType.ISO14443_CARD);
            } catch (ContactlessException ce) {
                System.out.println("Error  in  adding  listeners:  "  +
                                   ce.getMessage());
            } catch (Exception ce) {
                System.out.println("Error  in  adding  listeners:  "  +
                                   ce.getMessage());
            }

        }


        public void targetDetected(TargetProperties[] targetProprties) {
            System.out.println("Tag NFC detectada!");
            TargetProperties target = null;

            target = targetProprties[0];
            //Imprime URL de conexão
```

```
            System.out.println("URL:"+target.getUrl(target.getConnectionN
            ames()[0]));

            String                          URLBlue                         =
                   target.getUrl(target.getConnectionNames()[0]);
            try {
                this.conn   =   (StreamConnection)Connector.open(URLBlue,
                              Connector.READ_WRITE);
                this.aplicacaoForm.setConnection(conn);
                NFCBlueMIDlet.getInstance().setCurrent(aplicacaoForm);
            } catch (IOException ex) {
                ex.printStackTrace();
            }

        }

        public void recordDetected(NDEFMessage message) {
            System.out.println("Record detectado!");
        }

    }
```

A.5 AplicacaoForm.java

```
package TCC;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import javax.microedition.io.StreamConnection;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Displayable;


import javax.microedition.lcdui.Form;
/**
 *
 * @author Bernardo
 */
public class AplicacaoForm extends Form implements CommandListener {
    StreamConnection conexao;
    private Command voltar;
    private OutputStreamWriter writer;
```

```
    private InputStreamReader reader;

    public AplicacaoForm(){
        super("Aplicação Teste");
        this.addCommand(this.voltar = new Command("Voltar", Command.BACK,
                        0));
        this.setCommandListener(this);

    }

    public void commandAction(Command c, Displayable d) {
        if (c == this.voltar){
            this.leaveAplicacao();
        }
    }


    private void leaveAplicacao() {
        this.deleteAll();

        try {
            this.conexao.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
        NFCBlueMIDlet.getInstance().showMainMenu();
    }

    public void setConnection(StreamConnection conn) {
        this.conexao = conn;
        try {
            this.writer                         =                         new
                    OutputStreamWriter(this.conexao.openOutputStream());
            this.reader                         =                         new
                    InputStreamReader(this.conexao.openInputStream());
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```