INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, CÂMPUS PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

ALEX FINATTO DONASSOLO

ESTUDO DE CASO DA TECNOLOGIA NEAR FIEL COMMUNICATION NA PLATAFORMA ANDROID

Orientador
José Antônio Oliveira Figueiredo

PASSO FUNDO, 2014

ALEX FINATTO DONASSOLO

ESTUDO DE CASO DA TECNOLOGIA NEAR FIEL COMMUNICATION NA PLATAFORMA ANDROID

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador : José Antônio Oliveira Figueiredo

PASSO FUNDO, 2014

ALEX FINATTO DONASSOLO

ESTUDO DE CASO DA TECNOLOGIA NEAR FIEL COMMUNICATION NA PLATAFORMA ANDROID

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Prof. Esp. José Antônio Oliveira Figueiredo

_____

Prof. Msc. Evandro Miguel Kuszera

_____

Prof. Esp. Jorge Luis Boeira Bavaresco

_____

Prof. Dr. Alexandre Tagliari Lazzaretti

Coordenação do Curso

PASSO FUNDO, 2014

DEDICATÓRIA

Aos meus pais, aos meu irmãos e aos meus amigos
pela compreensão, apoio e o estímulo
em todos os momentos.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por todas as oportunidades que recebi ao longo de minha vida e também por me dar a capacidade de aproveitar cada uma delas, assim como a superar todos os desafios aprendendo cada vez mais com cada um deles. Agradecimentos também a toda equipe docente e administrativa do IFSUL, em especial do meu orientador, professor José Figueiredo, de quem recebi ótimas orientações e sugestões, e a todos aqueles a qual recebi a honra de ter seus ensinamentos. Agradeço também aos meus pais, irmãos por todo incentivo ao longo do curso. Também demonstro gratidão a todos os amigos e companheiros de percurso ou não, os quais vencemos cada desafio e, mesmo não conseguido concluir o curso juntos, a amizade concretizada permanecerá. E por fim, aos meus colegas de trabalho, pelo crescimento profissional e por entenderem as dificuldades passadas ao longo desse processo.

“Se você pensa que pode
ou se pensa que não pode,
de qualquer forma você está certo.”
Henry Ford

RESUMO

Este trabalho apresenta um estudo sobre a computação móvel, pervasiva e ubíqua, apresentando seus conceitos, características, diferenças e semelhanças. Também são estudadas as tecnologias RFID e NFC focando em seus aspectos teóricos como conceitos, padrões, arquitetura e em aspectos práticos, aplicando-a sobre o sistema operacional para dispositivos móveis Android. Para isso foi feito o desenvolvimento de duas aplicações que rodam em celulares com Android e são capazes de ler e escrever em *tags* por meio do NFC.

Palavras-chave: NFC; android; RDF; computação ubíqua.

ABSTRACT

This works presents a study on the mobile, pervasive and ubiquitous computing, showing up its concepts, characteristics, differences and similarities. There are also a study about RFID and NFC technologies focusing on its theoretical aspects as concepts, standards, architecture and practical aspects by applying it on the Android operating system for mobile devices. This was done by developing two applications that run on Android phones and are able to read and write NFC tags through.
Key words: NFC; android; RFID; ubiquitous computing.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AAR	*Android Application Record*
AID	*Application Identifier*
ADT	*Android Developer Tools*
API	*Application Programming Interface*
ASK	*Amplitude-Shift Keying*
BPSK	Binary Phase-Shift Keying
GPS	*Global Positioning System*
IANA	*Internet Assigned Numbers Authority*
IEC	*International Electrotechnical Commission*
IDE	*Integrated development environment*
ISO	*International Organization for Standardization*
HCE	*Host Card Emulation*
HF	*High Frequency*
LF	*Low Frequency*
LLCP	*Logical Link Control Protocol*
MAC	*Media Access Control*
MIME	*Multipurpose Internet Mail Extensions*
MP	Medida Provisória
NDEF	NFC *Data Exchange Format*
NFC	*Near Field Communication*
NFCIP	*Near Field Communication Interface and Protocol*
P2P	P*eer-to-pee*r
R/W	*Reader*/*Writer*
RF	*Radio Frequency* (Radio Frequência)
RFID	Ra*dio Frequency Identification*
RTD	*Record Type Data*
TNF	*Type Name Field*
SE	*Secure Element*
SDK	*Software Development Kit*
SIM	*Subscriber Identity Module*
SNEP	*Simple* NDEF *Exchange Protocol*
UHF	*Ultra High Frequency*

USB	*Universal Serial Bus*
URI	*Uniform Resource Indentifier*

SUMÁRIO

1 INTRODUÇÃO

Que computadores e sistemas computacionais estão cada vez mais presentes em nossa vida diária ninguém mais tem dúvida. Acontece que o futuro promete, além da presença desses equipamentos, outro nível de interação, que pode acontecer de forma percebida ou não, mas que, de uma forma ou de outra, sabe-se que está lá. Dispositivos cada vez menores, mais rápidos e poderosos tomam lugares e formas muitas vezes nem imaginadas, enquanto nossos conhecidos celulares vão adquirindo características e funcionalidades que antes só apareciam em dispositivos isolados.

No decorrer desse trabalho, será possível ver breves conceitos desses dispositivos, os quais fazem parte dos ambientes computacionais móvel, pervasivo ou ubíquo. Conceitos não muito conhecidos, mas que apresentam uma aplicabilidade extensa e expansível. Além disso, efetuou-se um estudo sobre uma tecnologia que, apesar de já fazer parte da vida de alguns de forma desapercebida, está em constante desenvolvimento. Junto com ela criam-se novas possibilidades e aplicações. Aplicações essas que tornam ambientes futurísticos, onde um sensor avisa um sistema geral para aumentar ou diminuir a temperatura do ambiente por meio do controle de um sistema de refrigeração, ou então para diminuir o volume do rádio ao perceber uma chamada telefônica por meio do controle de sistemas de som, cada vez mais presentes ou pelo menos, reais.

Essa tecnologia é a *Near Field Communication*, conforme Nfc Forum (f, 2014), tecnologia capaz de transmitir dados entre dispositivos próximos através de frequência de rádio, baseada na *Radio Frequency Identification* (WANT, 2006) e, ao longo desse trabalho, será feita sua introdução, conceituação e apresentação de seu funcionamento, de suas características e possibilidades. Para isso, será demonstrado o desenvolvimento de um trabalho prático em que a tecnolocia NFC é aplicada no sistema operacional Android, por Android (2014), mostrando algumas de suas funcionalidades, técnicas.

1.1 MOTIVAÇÃO

Verificando o crescimento da quantidade de dispositivos que apresentam a tecnologia *Near Field Communication*, assim como o número de aplicações já desenvolvidas para uso

dessa tecnologia, surge a necessidade de efetuar um estudo sobre a sua aplicabilidade e uma análise de seu funcionamento.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Estudar características, funcionalidades e modos de utilização da tecnologia *Near Field Communication* na plataforma Android.

1.2.2 Objetivos específicos

- Efetuar uma análise da base aplicável da tecnologia *Near Field Communication* (NFC), a computação ubíqua.
- Estudar a tecnologia usada como base no desenvolvimento do NFC.
- Estudar as características, padrões, arquitetura e componentes da tecnologia NFC.
- Destacar alguns exemplos de aplicação em que a tecnologia já vem sendo aplicada.
- Desenvolver uma aplicação para sistema operacional Android e entender e explicar as formas de utilização da tecnologia NFC para Android.

2 REFERENCIAL TEÓRICO

A Era da Informação, surgiu para demonstrar a necessidade do ser humano de compartilhar e transmitir dados principalmente por meio de dispositivos computacionais. Além dessa necessidade de compartilhamento, aumentaram também as possibilidades e meios para que aconteça essa troca, como resultado dos avanços da tecnologia. Atualmente, existem diversas tecnologias aplicáveis nos mais diferentes contextos para a troca de informações. Os contextos estudados nos próximos capítulos são da computação móvel, pervasiva e ubíqua. As tecnologias abordadas para que se faça essa transmissão de dados são a *Radio Frequency Identification* e a *Near Field Communication*.

2.1 Comunicação Móvel

Desde a travessia continental pelo Oceano Atlântico dos primeiros sinais de rádio, no ano de 1901, o sistemas de comunicação sem fio vem se desenvolvendo e se adaptando cada vez mais. Precedidos pelo telégrafo, tais sinais permitiram o surgimento do telefone inventado por Alexandre Graham Bell, como um meio de comunicação complementar. Os computadores, junto com a tecnologia digital, entram nessa história compondo uma nova geração nos sistemas de comunicação. O uso de fios, aliado a um elevado custo para acessos remotos, predominavam, tornando o uso de sistemas sem fios cada vez mais atraentes e necessários. Nessa situação, o principal impedimento ainda permanecia sendo a dependência de redes fixas.

Conforme Mateus e Loureiro (2005, p. 4),

> Enquanto a tecnologia sem fio se expande rapidamente para as redes de acesso, com baixo custo independente da distância da rede pública, as redes fixas, pelo uso da fibra ótica e os satélites, se complementam nas comunicações de longa distância. Essas são as alternativas tecnológicas atuais e de futuro, mesmo que de difícil previsão.

Sistemas de comunicação móvel são baseados, principalmente, na transmissão e na emissão de ondas de sinais de rádio frequência. Sendo assim, será feita uma análise de algumas formas de transmissão de dados por meio de rádio frequência bem como os sistemas de computação e comunicação que usam esses meios de transmissão.

2.2 Computação Móvel

O conceito básico, no que se refere à computação móvel, diz respeito à elevada capacidade de movimentação física, levando consigo os serviços computacionais. Assim, o "computador torna-se um dispositivo sempre presente que expande a capacidade de um usuário utilizar os serviços que um computador oferece, independentemente de sua localização." (ARAÚJO, 2004 p. 49). Tais características, aliadas com a possibilidade de acesso a serviços computacionais, fazem da computação uma atividade que independe da posição física do usuário, sendo a mobilidade sua característica fundamental.

Dentro do conceito de computação móvel, é importante destacar que o sistema computacional continua o mesmo, enquanto ocorre a variação da posição física do usuário. Sistemas e dispositivos ditos móveis não possuem a capacidade de obter informação sobre o contexto atual de sua aplicação e se adaptar automaticamente. Isso obriga os usuários a realizarem o controle e a configuração da aplicação, de acordo com a maneira que se movimentam, dificultando a aceitação e o uso por parte dos usuários.

2.2.1 Características básicas

Mobilidade, variação nas condições de comunicação e gerenciamento de energia são os 3 principais fatores que caracterizam a computação móvel, e devem ser estudados de forma particular, embora atuem de forma conjunta, conforme Mateus e Loureiro (2005, pág. 45). Esses fatores são frequentemente considerados como problemas na hora da pesquisa sobre esse assunto, porém são a base da computação móvel. Esses fatores são descritos na sequência.

2.2.1.1 Mobilidade

O mais importante aspecto em relação à mobilidade é a localização do dispositivo móvel em relação a um ponto de acesso para uma rede fixa, que pode mudar de acordo com a movimentação do dispositivo. "Como consequência da mobilidade, temos problemas

relacionados com gerência de localização, projeto de protocolos e algoritmos, heterogeneidade, segurança, dentre outros" (MATEUS e LOUREIRO, 2005, p. 46).

Assim como qualquer atividade, existe um custo[1] na comunicação do dispositivo móvel, e, junto a esse custo, existe o custo da pesquisa que o dispositivo realiza para encontrar um ponto de acesso à rede, como dizem Mateus e Loureiro (2005) ao explicarem também que precisam ser desenvolvidos algoritmos e estruturas de dados de forma eficiente, assim como os planos de execução de consultas devem ser projetados para consultar a localização dos elementos móveis com o menor custo possível.

Essa é, provavelmente, uma das grandes dificuldades no uso de dispositivos de computação móvel, já que a mobilidade não é caracterizada como uma simples capacidade de mudança de localização física do dispositivo, mas também com a capacidade de, mesmo com essa mudança, o dispositivo continuar conectado a uma rede, recebendo e enviando dados de forma constante. Afinal, a capacidade de troca de posição física se aplica a qualquer dispositivo, até mesmo a um servidor, o que muda é a facilidade e a praticidade na troca e nas condições de uso.

Mateus e Loureiro ainda falam que

> No projeto de protocolos e algoritmos distribuídos para ambientes móveis a configuração do sistema não é estática e, por essa razão, a topologia, que pode representar a comunicação entre as entidades comunicantes ou uma dependência de serviço ou uma outra relação, passa a ser dinâmica. Nesse contexto, o centro de atividades das aplicações e servidores, a carga do sistema e a noção de localidade mudam ao longo do tempo. Esses fatores não podem ser desprezados e, na verdade, um dos grandes desafios da computação móvel é projetar novas aplicações e algoritmos que levem em consideração essas características do ambiente. (2005, p. 46)

Essa mudança nas características do ambiente pode fazer com que a comunicação entre os elementos computacionais não seja garantida a todo tempo, e mesmo quando está disponível, ainda depende de variáveis como a confiabilidade. De forma geral, em ambientes externos a velocidade de comunicação é mais baixa do que em ambientes internos, onde há uma maior confiabilidade e ainda uma disponibilidade de conexão com uma rede fixa. Outro ponto a ser observado é a quantidade de elementos presentes no ambiente, que pode ser alterada, fazendo com que ocorra um estreitamento na largura da banda, assim como também podem ser alterados os serviços de rede disponíveis para um elemento.

[1] Custo: gasto de energia e tempo durante a execução de qualquer processo.

Outros fatores importantes a respeito da mobilidade são a segurança e a autenticação. Em uma conexão de rede sem fio, a tarefa de interceptação de mensagens é mais facilmente realizável, fazendo com que seja necessário, em alguns casos, o uso de criptografia. Além disso, a possibilidade de localização de um dispositivo móvel em um ambiente pode não ser de interesse quando o usuário deseja sigilo de localização, por exemplo.

2.2.1.2 Variação nas condições de comunicação

Mateus e Loureiro (2005) comentam que as redes de comunicação sem fio apresentam um custo maior de implementação, oferecem uma largura de banda mais baixa e são muito menos confiáveis a respeito da estabilidade da conexão do que as redes com fio, que cada vez mais aumentam a capacidade de transmissão e largura de banda.

Enquanto alguns pontos ainda são válidos em relação ao uso de uma rede de transmissão de dados sem fio, no entanto, hoje, pode-se dizer que as redes sem fio são uma maneira de baratear os gastos quando a largura de banda não é algo essencial, e sim a facilidade de conexão é o desejável naquele momento ou lugar.

No entanto, a falta de confiabilidade faz com que a taxa de bits perdidos seja consideravelmente maior em uma rede sem fio, e isso implica em um reenvio de dados por meio de uma conexão mais lenta. Outro fato interessante é o de que quanto mais dispositivos estiverem presentes (e conectados a uma rede) em um ambiente, menor será a largura de banda disponível para cada dispositivo.

Além disso, podem ocorrer, até com certa frequência, a chamada desconexão que, de acordo com Mateus e Loureiro (2005), pode ser voluntária ou involuntária. A desconexão voluntária se caracteriza por ser feita por meio de uma ação do usuário ou dispositivo e tem inúmeras razões, como, por exemplo, a redução de custo de conexão ou redução do consumo de energia e largura de banda. Enquanto isso, as desconexões involuntárias podem ocorrer quanto há falta de sinal em um determinado momento ou lugar. Em casos de desconexões com uma curta duração, cabe ao hardware e ao software fazerem essa cobertura para que o evento possa passar de forma quase imperceptível ao usuário. Além disso, tanto aplicativos quanto dispositivos devem ser desenvolvidos de forma que permitam o seu uso no modo desconectado.

Ainda, de acordo com Mateus e Loureiro,

> Outro aspecto importante relacionado com a comunicação sem fios são as características do computador móvel. Uma unidade móvel deve ser leve, pequena e fácil de carregar. Estas características em conjunto com o custo e tecnologias existentes fazem com que um computador móvel atual tenha menos recursos que computadores fixos, incluindo memória, velocidade de processador, tamanho de tela, dispositivos periféricos, memória secundária e inexistência de problemas relacionados com consumo de energia. Além disso, computadores móveis são mais fáceis de serem danificados, roubados ou perdidos. (2005, p. 48)

2.2.1.3 Gerenciamento de energia

A ideia é simples: como computadores móveis não estão ligados a um ponto fixo de energia durante seu funcionamento, eles dependem exclusivamente de uma bateria para realizarem suas tarefas e funções. Assim, Mateus e Loureiro (2005, p. 48) complementam dizendo que, para infelicidade dos usuários, o desenvolvimento de baterias cada vez menores e mais duráveis não acompanhou o desenvolvimento de dispositivos móveis menores, mais poderosos e com maiores capacidades de processamento e armazenamento. Logo, a tarefa de gerenciamento de energia deve ser tratada tanto pelo hardware quanto pelo software.

Em um processo de comunicação sem fio existem algumas peculiaridades que precisam ser levadas em consideração. A primeira delas é que o recurso usado para esta comunicação é limitado à capacidade de carga da bateria, devendo ser minimizado sempre que possível. A segunda é que um sinal deve ser transmitido com o uso de uma potência correta para que seja possível sua recepção e para que não se misture com outros sinais, evitando assim o ruído na comunicação.

Mateus e Loureiro (2005, p. 49) finalizam afirmando que “o grande desafio é projetar todo o software de um computador móvel considerando o consumo de energia. Por exemplo, tarefas do sistema operacional como escalonamento de processador e outros dispositivos, protocolos de comunicação e, principalmente, aplicações”.

2.3 Computação Pervasiva

De acordo com Araújo (2003, p. 50), “O conceito de computação pervasiva implica que o computador está embarcado no ambiente de forma invisível para o usuário”. Dessa forma, o dispositivo é capaz de obter informações sobre o ambiente no qual ele está contido, e, além disso, é capaz de usá-la de forma dinâmica. Isso significa que o computador é capaz de realizar controle, configuração e ajuste da aplicação para um modo que melhor atenda às

necessidades do usuário. Além disso, exige a capacidade de detecção de outros dispositivos que também façam parte daquele mesmo ambiente.

Nesse aspecto de uso, pode-se verificar que o conceito de computação pervasiva se aplica menos a elementos ou dispositivos computacionais, e mais aos próprios sistemas e aplicações que fazem uso desses elementos para se autogerenciar e controlar. Sistemas que estão presentes, ao mesmo tempo, em diversos elementos computacionais recebem a denominação de sistemas distribuídos, devido à distribuição que pode ocorrer em seu processamento ou no armazenamento de dados.

Dessa forma, Tanenbaum e Steen (2007) classificam sistemas distribuídos pervasivos como algo que faz parte de nosso ambiente, de forma embarcada. Além disso, uma característica interessante é a ausência geral de controle administrativo humano, fazendo com que os dispositivos precisem descobrir automaticamente seu ambiente e se adaptar o melhor que puderem.

Além disso, Grimm et al. (apud TANENBAUM e STEEN, 2007) fala sobre três requisitos para caracterizar um sistema:

1. **Adoção de mudanças contextuais**: o dispositivo deve estar ciente da possível alteração de seu ambiente de trabalho durante sua atividade, por exemplo, se conectar a outra rede quando ocorrer a perda da conexão inicial devido à movimentação da unidade.
2. **Incentivo de composição ad hoc**: cada usuário pode fazer um uso diferente para o dispositivo.
3. **Reconhecimento de compartilhamento como padrão**: Os dispositivos fazem acesso ao sistema de forma conjunta, com necessidades de leitura, armazenamento, gerenciamento e compartilhamento de informações.

Aliando algumas características da pervasividade, com características da mobilidade, percebe-se que deve ocorrer a adaptação do dispositivo ao meio em que está inserido, e consequentemente, tornando-se dependente do mesmo.

Das diversas possibilidades de uso de sistemas distribuídos pervasivos, pode-se citar duas usadas por Tanenbaum e Steen (2007): um sistema eletrônico para tratamento de saúde do paciente, em que são dispersos sensores em pontos-chave do corpo do paciente, junto a um sistema de transmissão sem fio que, quando ocorrer alguma mudança ou a cada intervalo de tempo determinado, envia os dados por meio de uma rede sem fio externa para uma unidade de armazenamento e, em caso de algum problema de saúde, faz a comunicação de emergência ao médico.

Existem alguns pontos que precisam ser observados nesse tipo de sistema como: onde serão armazenados os dados colhidos pelos sensores? Como fazer com que não ocorra a perda de dados cruciais? Qual a infraestrutura necessária para realizar o envio de sinais de alerta?

Além de um sistema de tratamento médico, Tanenbaum e Steen (2007) falam das redes de sensores. Além da elevada pervasividade, uma rede de sensores é dotada de inúmeros sensores capazes de gerar processamento de dados na própria rede, além de proverem informações, caracterizando-as como banco de dados distribuído. Esse processamento de dados feito pelos sensores deve ser feito da forma mais eficiente possível, por causa da limitação de recursos e energia.

2.4 Computação ubíqua

A computação ubíqua faz uso dos benefícios da computação móvel e da pervasiva, possuindo a capacidade de unir a mobilidade da computação móvel com a funcionalidade e o embarcamento da computação pervasiva. Araújo (2004, p. 50) define que, dessa forma, "qualquer dispositivo computacional, enquanto em movimento conosco, pode construir, dinamicamente, modelos computacionais dos ambientes nos quais nos movemos e configurar seus serviços dependendo da necessidade".

Segundo Araújo (2004), o grau de embarcamento indica a capacidade da computação de detectar, explorar e construir dinamicamente modelos em seu ambiente de forma embutida e transparente para o usuário.

Entende-se então, dessa forma, que nem todo dispositivo embutido é considerado móvel, assim como um dispositivo móvel não precisa estar embarcado no ambiente. Logo, a computação ubíqua agrega as características principais da computação móvel e pervasiva. Assim, um dispositivo ubíquo deve apresentar tanto a mobilidade, quanto certo grau de embarcamento.

2.4.1 Características básicas

De acordo com Araujo (2004), a computação ubíqua apresenta alguns princípios básicos para seu entendimento, conforme explicados a seguir.

2.4.1.1 Diversidade

No universo da computação existem diversos tipos de dispositivos. O mais conhecido deles é o computador pessoal, que atende muitos requisitos de praticamente todos os usuários. Em compensação, a computação ubíqua também é caracterizada pela diversidade de dispositivos, porém esses dispositivos apresentam uma nova ideia, que é a de realizar funções específicas para usuários particulares.

2.4.1.2 Descentralização

Na computação pervasiva todas as tarefas, funções e responsabilidades são distribuídas entre os vários pequenos dispositivos. Isso exige certo nível de cooperação entre eles para que a característica de ambiente inteligente seja refletida nas aplicações ubíquas. Nesse caso, é formada uma rede dinâmica entre dispositivos e entre dispositivo e servidor do ambiente, o que caracteriza um sistema distribuído, de acordo com Araújo (2004).

2.4.1.3 Conectividade

A ideia da computação ubíqua é uma conectividade sem limites de forma que dispositivos e aplicações se movam junto com o usuário através das diversas redes disponíveis, sendo redes sem fio de grande, médio e curto alcance. Isso exige uma padronização para atingir a conectividade e a interoperabilidade necessária.

2.4.1.4 Dispositivos

Como visto anteriormente, existe uma enorme diversidade de dispositivos que pertencem ou podem pertencer à computação ubíqua, sendo essa diversidade um de seus princípios básicos. Alguns exemplos desses dispositivos são listados a seguir:

a) Controles Inteligentes

São controles muito pequenos que podem ser conectados a diversos aparelhos como lâmpadas, interruptores, sensores, entre outros. Atuam de forma

embarcada nesses dispositivos sem necessitarem de interação humana, podendo ser controlados local ou remotamente. De acordo com Araújo (2004), esses controles podem ser usados no controle da segurança residencial ou de comodidades do usuário.

Os chamados *smart cards*[2] são um tipo específico de controle inteligente. Possuem um alto potencial de uso e um elevado grau de segurança. O melhor exemplo de uso, hoje em dia, é o dos cartões SIM, usados em aparelhos de telefonia móvel. Em relação ao seu tamanho, um *smart card* possui um chip de no máximo 25 mm², contendo um microprocessador, memória e um sistema operacional para realizar suas funções.

b) Utensílios inteligentes

Esse tipo de dispositivo, na verdade, possui uma complexidade maior do que um controle inteligente, pois pode contar com vários controles inteligentes dentro do mesmo dispositivo. O que torna o dispositivo inteligente é a própria presença de um ou mais controles.

Um dos ambientes em que se percebe o elevado crescimento no uso da tecnologia é, com certeza, o dos automóveis. Sistemas de computador de bordo, GPS, além de sensores e controladores internos. E seu uso pode se estender do interior do veículo para, por exemplo, informar a montadora sobre dados de temperatura do veículo e problemas elétricos, ou até buscar informações na rede sobre acidentes e desvios, atualizando o GPS para uma melhor rota.

c) Dispositivos de acesso à informação

São os dispositivos que oferecem um acesso aos sistemas de informação, permitindo uma intensa interação com o usuário, que pode ser feita por meio de uma tela sensível ao toque, um teclado, comandos de voz, entre outros.

O uso desse tipo de dispositivo está em alta, assim como as tecnologias neles usadas. Mesmo com um espaço físico limitado, suas capacidades aumentam a cada desenvolvimento, não deixando a desejar em nenhum aspecto mesmo quando comparados aos tradicionais computadores pessoais.

[2] Smart Card, tradução do inglês de cartões inteligentes

2.4.2 Hardware de sistemas ubíquos

O que torna um dispositivo ubíquo, além de suas capacidades mencionadas anteriormente, como conectividade e mobilidade, são algumas características físicas que podem variar entre os mesmos, mostrando a diversidade e heterogeneidade. Mesmo com essas diferenças, alguns elementos são fundamentais e merecem destaque no estudo desses dispositivos.

O desenvolvimento de diversas tecnologias presentes nos dispositivos, como o processador, está em constante crescimento. Infelizmente, a evolução das baterias não tem acompanhado o mesmo ritmo e, por consequência, está limitando a capacidade de diversos dispositivos. Como diz Araújo (2004, p. 14), "qualquer avanço é rapidamente superado pelo aumento de consumo de energia dos processadores mais rápidos".

Além da bateria que, por necessidade, está em constante evolução, outros componentes como processador, memória, tela, teclado, entre outros apresentam um processo evolutivo considerável, embora estejam ainda em um nível mais baixo quando comparados com computadores pessoais ou notebooks.

2.5 RFID

Radio Frequency Identification, cuja tradução pode ser Identificação de Frequência de Rádio, é uma tecnologia desenvolvida há cerca de uma década que permite a transmissão de dados por meio de ondas de rádio por distâncias que variam de alguns centímetros a distâncias superiores a cem metros, variando conforme a frequência da transmissão.

Como diz Want (2006), a tecnologia RFID permite identificação a distância, assim como o código de barras, porém faz isso sem a necessidade de visão do objeto a ser identificado. Além disso, *tags*[3] RFID oferecem suporte ao armazenamento e transmissão de diversos identificadores como fabricante, tipo de produto, ou até mesmo características presentes no meio, como temperatura.

Conforme EBV Elektronik (2010), algumas outras vantagens do uso de RFID em relação ao código de barra ou à fita magnética são:

- Capacidade de leitura e escrita;
- Diferentes capacidades de armazenamento e tecnologias disponíveis;

[3] *Tags* ou etiquetas denominações usadas para *smart cards.*

- Disponibilidade de aspectos de segurança;
- Capacidade de leitura múltipla;
- Funcionamento em ambientes agressivos.

2.5.1 Tags

Tags RFID são os dispositivos capazes de armazenar e transmitir os dados para o dispositivo leitor e existem em basicamente dois tipos, conforme Want (2006).

2.5.1.1 *Tags* Ativas

Tags são consideradas ativas quando tem uma fonte de energia própria disponível, normalmente uma bateria. Conforme EBV Elektronik (2010), isso permite que esse tipo de *tag* faça transmissões e possam ser lidas e escritas a distâncias maiores. Por esse motivo, esse tipo de dispositivo só pode ser obtido a um custo mais elevado, além de serem consideravelmente maiores fisicamente. Nesse sentido, Want (2006) diz que a aplicabilidade de *tags* ativas se torna muitas vezes inviável devido ao custo de uma bateria e ao tempo de vida da mesma.

2.5.1.2 *Tags* Passivas

Diferentemente das *tags* ativas, as *tags* passivas obtêm a energia necessária para a transmissão dos dados do campo magnético criado pelo dispositivo que fará a leitura/escrita. Assim, conforme EBV Elektronik (2010), não requisitando uma fonte própria de energia faz com que esse tipo de *tag* seja muito menor e mais barato.

Want (2006) explica que a composição de uma *tag* passiva se resume a três elementos:

- Uma antena para captar o sinal e fazer a transmissão.
- Um chip semicondutor para armazenamento dos dados.
- Uma proteção para evitar o contato dos outros elementos com o ambiente externo.

2.5.2 Frequência

Dispositivos RFID trabalham em determinadas faixas de frequência, descritas a seguir:

- Low Frequency (LF): < 135 kHz
- High Frequency (HF): 13.56 MHz
- Ultra High Frequency (UHC): 850...960 MHz

Uma rápida comparação entre cada uma das faixas e suas características básicas pode ser encontrada na tabela 1:

Tabela 1 Comparativo Entre Faixas de Frequência RFID

Frequência	LF	HF	UHF
Distância	0,5 até 1m	< 1m	> 3m
Custo	Maior custo	Regular	Menor custo
Penetração de materiais	Excelente	Média	Fraca
Taxa de transmissão	Baixa	Boa	Excelente

Fonte: Adaptado de EBV Elektronik, 2010

2.5.2.1 Transmissão

Como visto anteriormente, a tecnologia RFID provém recursos para a transmissão de dados a longas e curtas distâncias. De acordo com Want (2006), essa diferença ocorre basicamente nos modos de transferência de energia utilizados que são: indução magnética e captura de ondas eletromagnéticas.

Para esse trabalho será feito um estudo para a comunicação a curta distância.

2.5.2.2 Transmissão de campo próximo

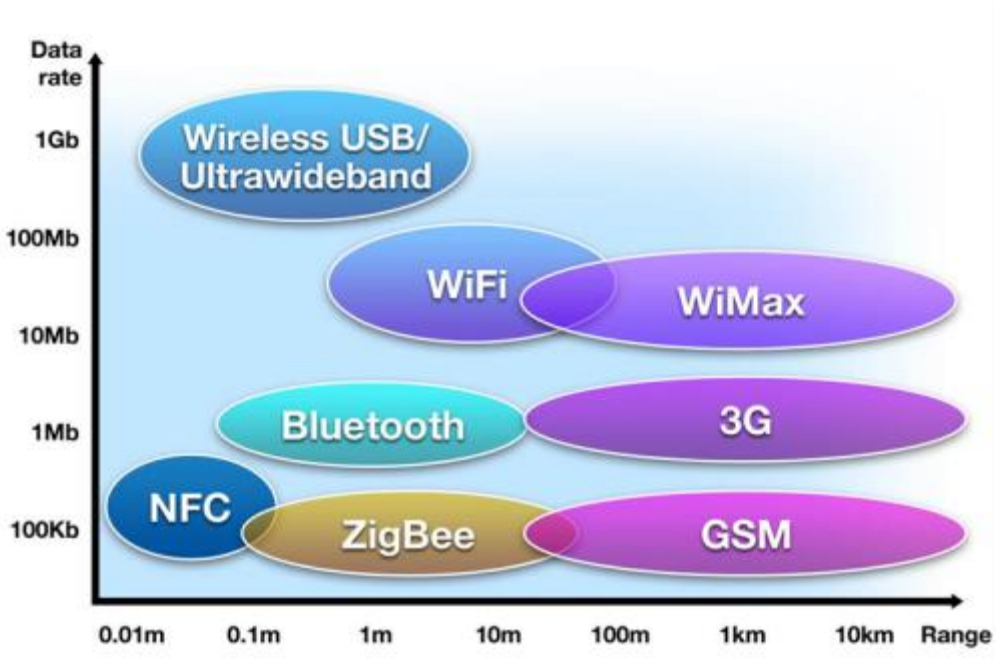
Conforme adaptado de Want (2006, p. 3), "Indução magnética é a base para o acoplamento entre o dispositivo leitor e *tag* para transmissão de campo próximo." O autor também explica que há *tags* que usam esse tipo de transmissão enviando os dados ao leitor usando modulação de carga. Isso acontece porque, após gerar o campo de rádio frequência, o dispositivo leitor irá detectar as alterações presentes na corrente gerada pelo seu campo magnético em forma de modulação e, de acordo com a carga dessa modulação, é gerada a resposta ao dispositivo leitor pela *tag*.

2.6 NFC

Near Field Communication, de sigla NFC – a qual pode ser traduzida como Comunicação de campo próximo –, é uma tecnologia usada para comunicação entre dois dispositivos próximos, sem que haja um contato físico necessário entre eles, e é esse o motivo da expressão inglesa largamente usada para designar esse tipo de comunicação: *contactless*[4]. O NFC é baseado na tecnologia RFID e tem sua padronização de acordo com a ISO/IEC 18092. Para comunicação, o uso do NFC é limitado a uma distância de no máximo 20 centímetros entre os dispositivos, variando conforme o modo de utilização que será estudado no capítulo 2.6.2.

O objetivo do NFC é facilitar transações, troca de conteúdo digital e conexão entre dispositivos com apenas um movimento. Essa tecnologia trabalha na frequência de 13,56 MHz e foi desenvolvida em conjunto pelas empresas NXP Semicondutores (formada pela Philips Semicondutores) e Sony Corporation. (NFC-forum.org, apud CURRAN, MILLAR e GARVEY, 2012).

Figura 1 Comparação com outras tecnologias de transmissão de dados sem fio



Fonte: NFC FORUM (a), 2013

[4] Contactless: do inglês, sem contato. Que não precisa de um contato físico.

Uma simples comparação entre o NFC e outros meios de transmissão de dados sem fio é mostrada na Figura 1, onde se pode observar aonde está situada cada tecnologia de acordo a distância de comunicação e a taxa de transmissão.

O site oficial referente a NFC (http://www.nfc-world.com), que possui diversas informações sobre a tecnologia, informa que ela foi desenvolvida para ser a segunda geração da tecnologia RFID, e é esperada a sua implantação em diversos dispositivos, conforme Levandoski et al. (2011).

A tecnologia NFC é um padrão aprovado, significando que, na expectativa de seus criadores, será amplamente usada e reconhecida ao redor do mundo com o objetivo de ser uma tecnologia multiuso.

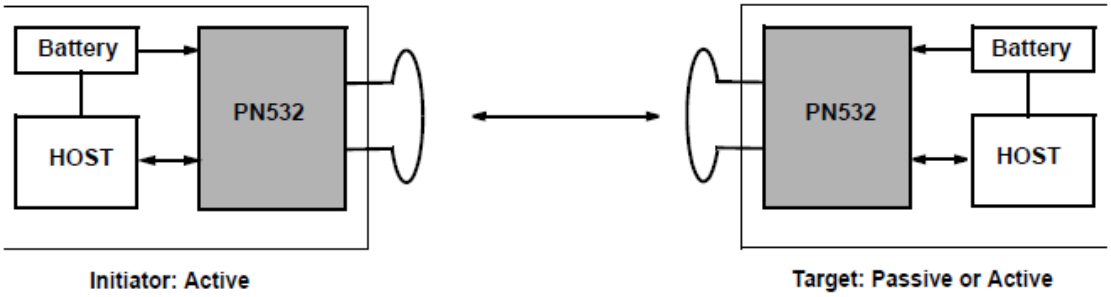
Ferraz Filho (2010, p. 56) diz que o

> NFC é uma tecnologia aberta, e seu padrão se chama NFCIP-1 (Near Field Communication Interface and Protocol 1). Esse padrão especifica os esquemas de modulação, codificações de bit, taxas de transmissão e formato de quadro para a interface aérea, assim como os mecanismos de inicialização e controle de colisão.

2.6.1 Modos de transmissão

Para sua utilização, o NFC necessita de dois dispositivos que, apesar de poderem apresentar características iguais ou semelhantes, têm funções bem diferenciadas no momento da comunicação. De acordo com Levandoski et al. (2011, p. 4), “Na comunicação via NFC sempre estão envolvidos dois elementos, um iniciador e um destino, o iniciador ativamente gera um campo de RF (campo magnético) que pode alimentar um alvo passivo”. Dessa forma, o iniciador controla a troca de informações de forma que a requisição do iniciador é respondida pelo alvo destino. Além dos dispositivos, o protocolo do NFC permite a distinção de duas formas de comunicação, ativa e passiva, que dependem diretamente da configuração, tipo e disponibilidade dos dispositivos envolvidos. Um dispositivo ativo é considerado aquele alimentado por uma fonte constante de energia, enquanto um dispositivo passivo não dispõe de nenhuma fonte de energia. A figura 2 mostra uma representação dos dois dispositivos.

Figura 2 Elementos de uma comunicação NFC



Fonte: NXP Semicondutores, 2012

Conforme NFC Forum, (apud LEVANDOSKI et al., 2011 , p. 3),

> O NFC é compatível com o *Smart Card* "*FelicaTM*" desenvolvido pela Sony e utilizado em países, como Japão e Ásia, e também com o "*Mifare*", *Smart Card* desenvolvido pela Philips. O NFC pode se comunicar com estes dispositivos, mas por poderem se comunicar entre eles (NFC com NFC) espera-se maior flexibilidade nas comunicações utilizada pelos *smart cards*.

2.6.1.1 Transmissão Ativa

A forma de comunicação ativa ocorre quando tanto o dispositivo iniciador quanto o dispositivo destino, ou alvo, são caracterizados como ativos. Assim, conforme Levandoski et al. (2011), neste modo de operação ambos os equipamentos geram um campo de rádio frequência para efetuar a transmissão. A transmissão ativa é necessária para os casos em que se deseja uma interação entre os aparelhos, na forma de requisições e respostas. Conforme Curran, Millar e Garvey (2012), essa forma de transmissão é do tipo *half-duplex*, ou seja, os dados são transmitidos tanto do iniciador quanto do alvo, cada um no seu tempo, e nenhum dos dispositivos desativa seu campo de rádio frequência até que a transmissão seja finalizada.

2.6.1.2 Transmissão Passiva

O modo de transmissão passiva se caracteriza pela ausência de um dos pontos geradores do campo eletromagnético. Apenas o iniciador, ao iniciar a transmissão, gera o

campo que, ao atingir o alvo, fornece a energia necessária para o funcionamento deste, e assim poder efetuar a transmissão de dados. É essa capacidade que torna possível o uso do NFC em objetos passivos, que não dispõem de recursos energéticos. Ainda de acordo com Curran, Millar e Garvey (2012), o alvo responde ao chamado modulando o campo de rádio frequência criado pelo iniciador, que detecta essa modulação e processa a transferência de dados. De acordo com as especificações da NFC, as taxas de transmissão são 106, 212 ou 424 Kbit/s.

2.6.2 Modos de utilização

O NFC pode ser aplicado em três modos de operação distintos, detalhados a seguir.

2.6.2.1 Modo Leitor/Gravador

Do inglês *reader*/*writer mode* (R/W), neste modo, o dispositivo pode ler e alterar os dados em um NFC compatível de forma passiva. Essa alteração é feita sem interação do usuário e conforme os dados que estão inseridos na etiqueta NFC. Etiquetas, ou *tags*, como também são chamadas, são dispositivos que trabalham de modo passivo e podem ser usados para comunicar com os ativos. Nesse modo de uso, há a presença do dispositivo ativo e da *tag*. O ativo faz a leitura dos dados no formato NDEF, ou NFC *Data-exchange format*, e RTD, ou *Record-Type Definition*. Essas duas especificações padronizam a forma como o dispositivo NFC faz o processo de leitura e escrita.

Conforme Suparta (2012), o dispositivo ativo pode acessar os dados de um objeto RFID, como, por exemplo, um pôster contendo uma *tag* NFC, permitindo ao usuário fazer o download do trailer do filme do pôster, por meio de uma URL (*Uniform Resource Locator*) contida na *tag*. A codificação do formato da mensagem usada pelo NFC é transmitida pela NDEF. Isso permite a divisão das múltiplas mensagens NDEF em partes, estágios, atividades, tarefas e passos. O RTD define como é feita a gravação das mensagens. Cada gravação tem um tipo que indica o que a mesma contém e são expansíveis, sendo definidas pelas especificações do fórum NFC ou por algum outro órgão.

Existem quatro tipos diferentes de etiquetas que, de forma padrão, todo dispositivo NFC deve oferecer suporte a leitura das mesmas. Essas etiquetas são definidas na norma ISO/IEC 14443 como PICC, sigla de *Proximity Cards*, como são conhecidos os cartões de transferência de dados de forma *contacless*. O dispositivo que faz a leitura, no caso, o que

possui o NFC habilitado, é definido também na ISO/IEC 14443 como um PCD, de *Proximity coupling device*. A tabela 2 mostra um comparativo entre as 4 *tags* padronizadas na ISO 14443 e o tipo clássico Mifare. Conforme Igoe, Coleman e Jepson (2014, pág. 20), nem todos os dispositivos que oferecem suporte a NFC conseguem trabalhar com todos os tipos de *tags* existentes.

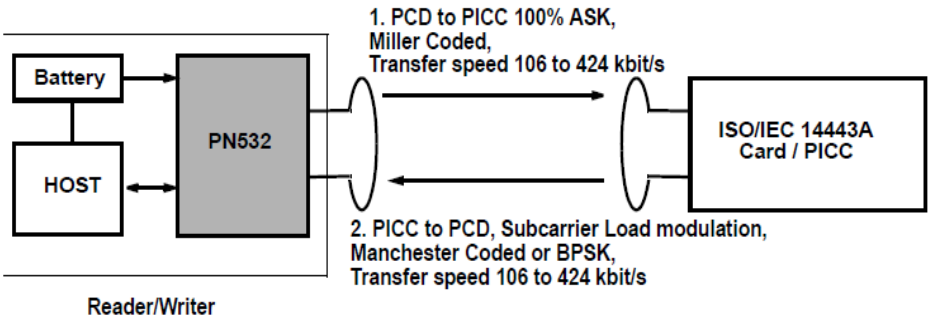
Tabela 2 Comparativo entre os tipos de *tags*

Tag	Padrão	Armazenamento	Transmissão	Características	Exemplo
Tipo 1	ISO-14443-A	96B até 2kB	106 kbp/s	Sem suporte anti-colisão	Innovision Topaz
Tipo 2	ISO-14443A	96B até 2kB	106 kbp/s	Anti-colisão de dados	NXP Mifare Ultralight
Tipo 3	JIS	até 1MB por serviço	212, 424 kbp/s	Anti-colisão de dados	Sony Felica
Tipo 4	ISO-14443A	até 32 kB por serviço	106, 212, 424 kbps	Anti-colisão de dados	NXP DESFire
MIFARE	ISO-14443A	192, 767, 3584 bytes	106 kbp/s	Anti-colisão de dados	NXP MIFARE Classic
JIS: Sigla de Padrão Industrial Japonês					

Fonte: Autor

Existem algumas diferenças na transmissão de dados para cada situação que vão além das características de cada cartão específico, como mostrado na Figura 3, onde pode-se observar variações na codificação, modulação e velocidade de transferência de dados.

Figura 3 Diagrama de comunicação no modo Leitor/Gravador



Fonte: NXP Semicondutores, 2012

2.6.2.2 Modo Emulação de cartão

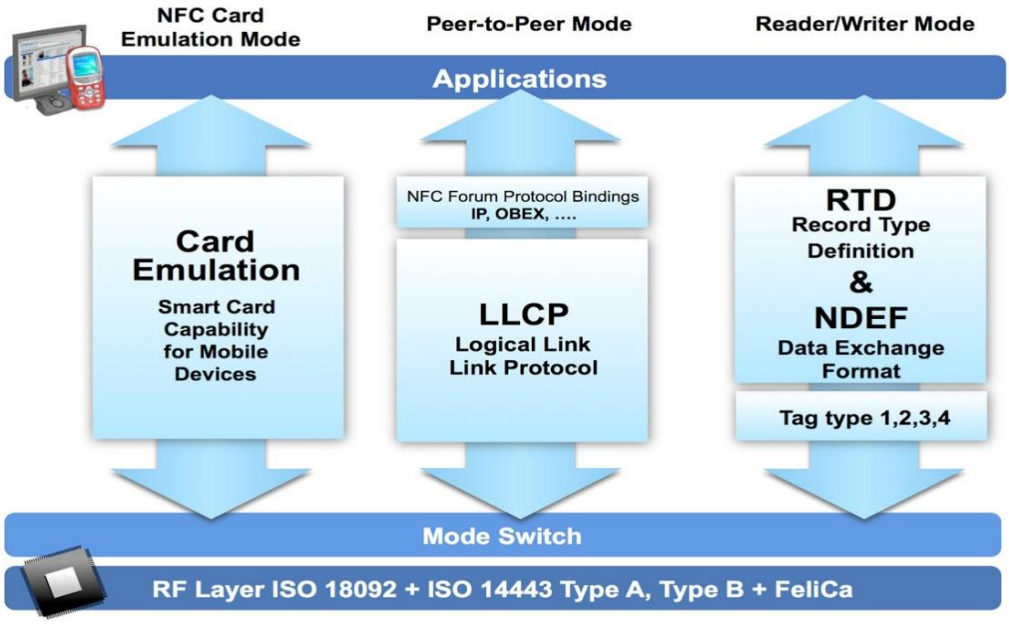
Usando o NFC na forma Cartão de Emulação, ou NCE, sigla de NFC Card Emulation, os dispositivos NFC podem agir como um *Smart Card*, contendo um chip seguro, também conhecido como Elemento Seguro ou *Secure Element* (SE). O SE é conectado ao controlador de NFC. Dessa forma, um leitor externo pode se comunicar com o dispositivo para troca de dados, fazendo com que ele se comporte de forma passiva, semelhante a uma etiqueta NFC, não gerando o campo de rádio frequência. Esse modo de utilização é usado principalmente para pagamento móvel.

2.6.2.3 Modo Ponto-a-ponto

No modo P2P, do inglês *Peer-to-peer*, ocorre uma conexão bidirecional entre dois dispositivos NFC, o que permite uma troca de contatos, informações e sincronização para *Bluetooh*, conforme informa Suparta (2012). O autor ainda informa que operações P2P podem fazer com que ocorra troca de dados entre um computador pessoal e um celular, caso ambos tenham NFC habilitado.

A Figura 4 demonstra os modos de utilização do NFC e as tecnologias usadas para cada modo.

Figura 4 Componentes e tipos de transmissão do NFC

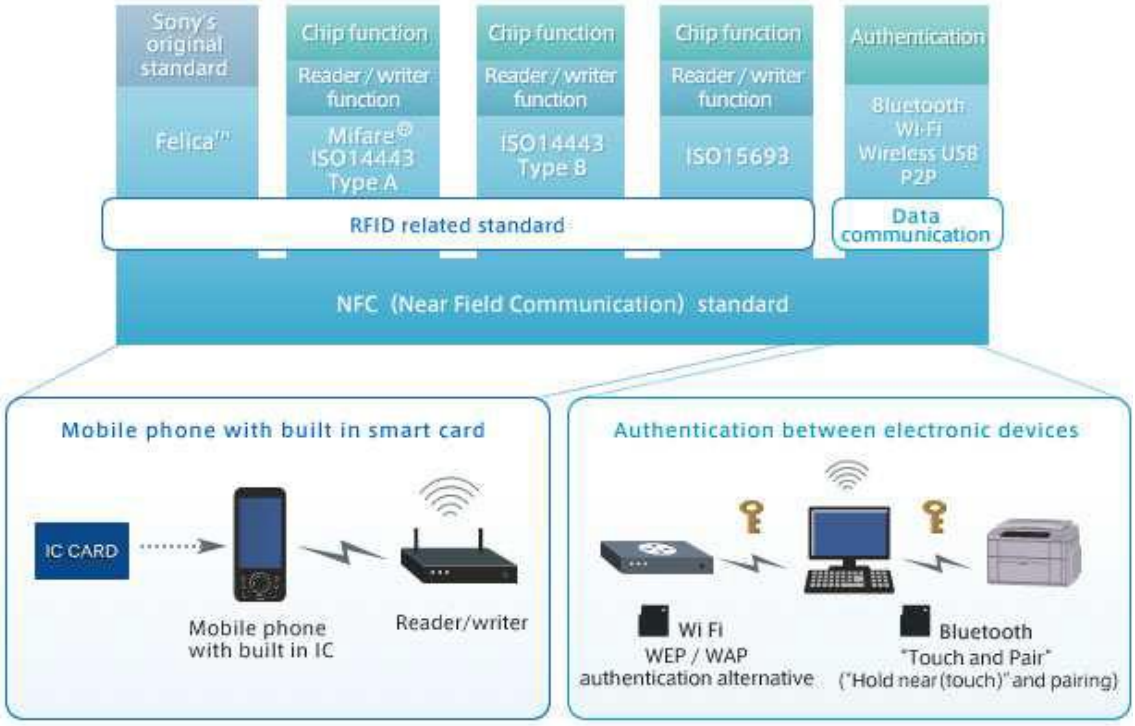


Fonte: NFC Fórum (b), 2013

2.6.3 Características

O padrão ISO/IEC 18092 especifica uma interface e um protocolo de comunicação sem fio entre um par de dispositivos próximos com uma taxa de transferência de 106, 212 e 424 kbp/s (*International Standart Organization* – ISO, apud KEVIN, CURRAN e GARVEY, 2012).

Figura 5 Padrões de Compatibilidade da tecnologia NFC



Fonte: Levandoski et. al. , 2011

A figura 5 mostra como funcionam os padrões para compatibilidade da tecnologia NFC, de acordo com os modelos de operação mencionados anteriormente.

Conforme Want (2006, p. 6), "o padrão NFC tem como objetivo tornar mais simples os processos de envio de endereços MAC (Media Access Control) e chaves de criptografia por distâncias limitadas a no máximo 20 cm por meio de ondas de rádio entre dispositivos NFC, aumentando a segurança física dos usuários." O autor também explica a capacidade de leitura dos, já então populares no Japão, cartões Felica e Mifare, característica que cria muitas possibilidades para a tecnologia.

Além disso, o NFC permite o modo de comunicação P2P, o que permite uma autenticação entre dispositivos.

Como visto até esse ponto, a tecnologia NFC pode ser usada de diversas maneiras e de acordo com alguns modos de operação e é isso que faz dela uma tecnologia multiuso. Hoje em dia, a necessidade de tecnologia aumenta cada vez mais e é nesse ponto que o NFC entra em cena. Sua capacidade de estar presente em vários dispositivos fez dele uma excelente opção para utilização nas aplicações que envolvem uma troca rápida de dados.

A partir do momento em que foi criado, o NFC já possuía um propósito inicial e que já está sendo aplicado em alguns países, como o Japão. Esse propósito era o uso de um dispositivo móvel, em especial um smartphone[5], para facilitar a compra de produtos e serviços, a chamada carteira eletrônica. Por meio do smartphone com a tecnologia NFC habilitada, é gerada a capacidade de transferência de informações por meio do toque ou aproximação com outro dispositivo que também possua a tecnologia.

2.6.3.1 PN532

Hoje em dia existem alguns chips com a função de comunicação NFC. O mais popular deles é o PN532, da fabricante NXP Semicondutores. Todas as suas características de transmissão de dados, modos de utilização, codificação e modulação estão descritas no manual de especificações técnicas disponível no site da fabricante. Ele oferece suporte a todos os modos de utilização estudados anteriormente.

2.6.3.2 NFC e RFID

Want (2006) explica que, no ano de 2002, a empresa Philips já explorava um padrão aberto, o que resultou no *Near-Field Communication Forum*. Esse fórum propõe integrar informações sobre a transmissão entre dispositivos em um campo próximo e usar essas informações para aproximar o que for compatível com os já existentes dispositivos passivos RFID. Esse novo padrão NFC fornecia meios nos quais dispositivos móveis poderiam se comunicar a curtas distâncias, sem depender de sistemas de detecção e descobertas, que são exigidos tecnologias como Bluetooh e Wi-Fi.

Dessa forma, o padrão NFC compartilha as mesmas características de leitura e escrita de cartões descritas na ISO 15693, que descreve características de cartões de proximidade,

[5] Smartphone. Nome comercial para celular com a possibilidade de instalação de aplicativos e alguns recursos adicionais.

porém com um alcance de leitura superior aos da ISO 14443. Além disso, o padrão NFC trabalha na frequência fixa de 13,56 MHz, faixa de alta frequência para o RFID.

Sendo a tecnologia NFC baseada na RFID, porém com uma frequência fixa e menor alcance de leitura e escrita, não existem *tags* NFC ativas, pois a presença de fontes de energia em *tags* é necessária quando se deseja um grande alcance para comunicação e transmissão de dados.

Conforme Joan (2009), ondas de rádio frequência são usadas para transmitir dados por longas distâncias, principalmente quando energizadas. Esse tipo de alcance é desejável quando o objetivo é, por exemplo, rastreamento de animais. Porém, esse alcance deixa de ser desejado quando o situação exige uma certa privacidade como, por exemplo, carteira digital e identificação digital. É para essas situações que o NFC é usado.

2.6.3.3 NFC e Bluetooth

A diferença entre essas duas tecnologias pode ser explicada pela simples comparação entre suas características principais e seus objetivos. Tanto o alcance quando a velocidade de transmissão são largamente maiores quando usando o Bluetooth. Em consequência dessa maior capacidade de transmissão, o consumo de energia é também maior durante seu uso (JOAN, 2009).

Quando se deseja utilizar o bluetooth é necessário realizar a sincronização entre os dispositivos, isso é, fazer com que se enxerguem e se conectem um ao outro, antes de iniciar a transmissão. No entanto, para o NFC, é necessário apenas que esse meio de transmissão esteja ativado nos dois dispositivos, sendo que a transmissão e a detecção ocorrem de forma automática. Com essa característica, é possível, inclusive, realizar uma sincronização mais rápida entre dispositivos na hora de utilizar o Bluetooth utilizando o NFC.

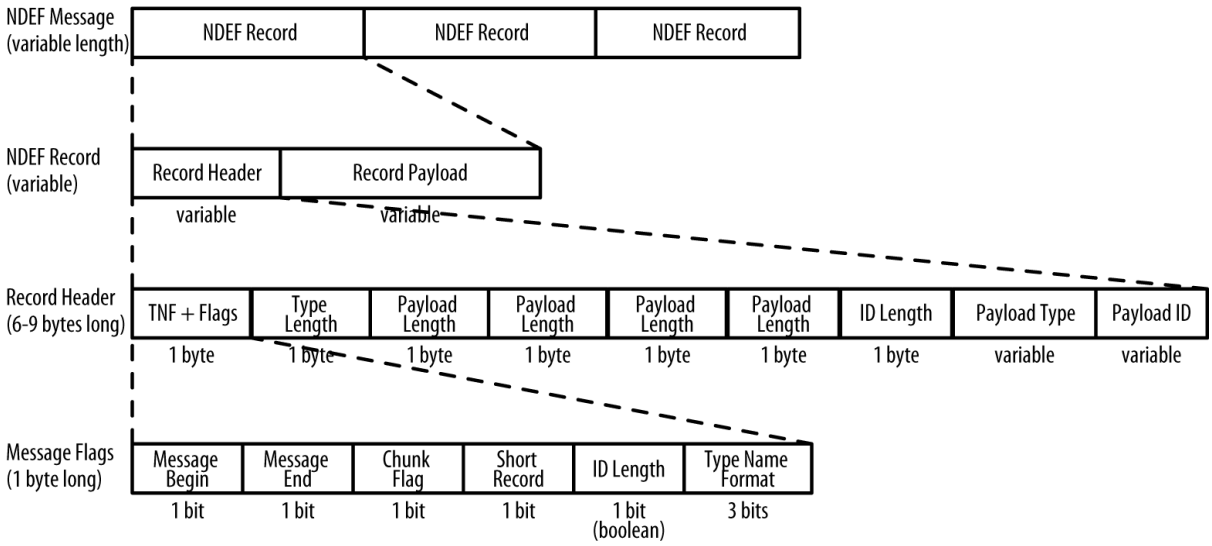
2.6.4 Padrões de transferência de dados

2.6.4.1 NDEF

O NDEF foi criado para que seja possível a transferência de dados entre *tags* e dispositivos NFC. Conforme adaptado de Nokia Developer (2011), “NDEF é um formato binário leve e compacto para armazenar os variados tipos de dados suportados pelo NFC.” Dessa forma, um dispositivo NFC pode fazer a leitura de qualquer *tag*, pois o padrão NDEF faz a transparência da leitura para as aplicações. Ainda de acordo com Nokia Developer (2011), o formato NDEF consiste em uma sequência de *records*, sendo que cada uma

armazena um *payload.* Cada *payload* contém um tipo de dado específico, junto ao valor representado no formato daquele tipo de dado. O tipo de dado contido no *record* e o tamanho dela são indicados dentro de um *header* que está junto ao *payload*, dentro do *record*, conforme mostrado na figura 6.

Figura 6 Mensagem NDEF



Fonte: Igoe, Coleman e Jepson, 2014

O padrão NDEF define duas estruturas:

NDEF Message: Uma mensagem no formato NDEF é um container que serve para armazenar um ou mais *records*.

Ndef Record: Conforme NFC Forum (c), um *record* deve especificar o tipo de dado que nele está gravado por meio de um *Record Type Name*. Esses nomes podem ser especificados em diversos formatos, chamados de TNF - Type Name Field. Os TNF que podem ser usados para declarar um *record* type name são do tipo: MIME - Multipurpose Internet Mail Extensions -, URI's (*Uniform Resource Indentifier*) absolutas , tipos externos do NFC Forum e RTD's.

Igoe, Colema e Jepson (2014, pág. 50) dizem que o TNF diz como deve ser interpretado o tipo do *payload*. Os TNF existentes e seus respectivos valores são:

Nome	Valor	Descrição
Empty	0	*Record* vazio sem tipo e *payload*.
Well-Known	1	Um dos tipos pre-definidos pelo NFC Forum.

MIME media type	2	Um tipo *MIME*.
Absolute URI	3	Uma URI.
External	4	Um valor definido pelo usuário, de acordo com as especificações de um RTD definidas pelo NFC Forum.
Unknown	5	Tipo desconhecido. O tamanho deve ser 0.
Unchanged	6	Não pode ser usado no primeiro *Record*. É usado em caso de *Record's* fragmentados.
Reserved	7	Ainda sem especificação.

2.6.4.2 RTD

Para cada tipo de dado que pode ser armazenado no formato NDEF, é definido um documento RTD. De acordo com Nokia Developer (2011), a tecnologia NFC tem definidos alguns formatos de RTD que são:

- NFC Text RTD
- NFC URI RTD
- NFC SmartPoster RTD
- NFC Signature RTD

O formato de texto é o mais simples, podendo armazenar sequência de caracteres. O formato de URI pode ser usado para armazenar URLs e e-mails, enquanto o formato SmartPoster define como incluir dados como URLs, telefones e outros dados em uma *tag* NFC e como transportar esse dados entre os dispositivos, conforme a Nokia Developer (2011).

Sendo o mais simples de todos os formatos, o Text não possui nenhuma dependência, sendo que seu uso normalmente está associado à descrição de um serviço ou ao conteúdo de uma *tag*. É possível que apareça de forma isolada e, dessa forma, é a aplicação que vai manusear os dados contidos, pois a *tag* não possuirá um objetivo definido (NFC FORUM, 2006).

As tabelas 3, 4 e 5 mostram a composição de uma *record* com dados no formato texto.

Tabela 3 Composição de um Payload no formato texto

Posição	Tamanho	Conteúdo

(byte)	(bytes)	
0	1	Byte para Status
1	<n>	ISO/IANA *Language Code*
n+1	<m>	Texto

Fonte: NFC Forum, 2006

Tabela 4 Composição do Byte para Status

Bit Number	Conteúdo
7	0 para texto no formato UTF-8 1 para texto no formato UTF-16
6	Valor deve ser 0
5..0	Tamanho da Language Code

Fonte: NFC Forum, 2006

Language Code é a codificação no formato ISO/IANA para representar o idioma da mensagem contida no texto. Por exemplo, "pt" para Português e "pt-BR" para Português-Brasil.

Tabela 5 Record com um RTD no formato texto

Posição	Conteúdo	Explicação	Função
0		1 para campo identificador 0 para formato curto	Header
1	0x01	Tamanho do *Record name*	
2	0x10	Tamanho do *Payload*	
3	"T"	Codificação Binária para o formato	
4	0x05	UTF-8 e 5 bytes para a *Language Code*	Payload
5	pt-BR	Código ISO para idioma Português Brasil	
7	"Olá, Mundo"	Texto	Texto

Fonte: NFC Forum, 2006

2.6.4.3 LLCP

Para o uso do NFC no modo P2P um novo e essencial protocolo entra em cena, o LLCP, Logical Link Control Protocol, como descreve o NFC Forum (c) (2013), dizendo que o LLCP define dois tipos de serviço, os orientados a conexão e os não-orientados a conexão organizados de três formas: Serviço só orientado à conexão, serviço não-orientado à conexão e o que usa ambos os modos. O serviço-não orientado à conexão, também chamado de *connectionless*, oferece as condições mínimas de configuração, segurança e controle de fluxo. Enquanto isso, o orientado à conexão incrementa segurança de entrada, controle de fluxo, ordenamento e serviço baseado em múltiplas camadas. O NFC Forum (c) diz ainda que o LLCP oferece suporte a pequenas aplicações com limitados requisitos de envio de dados, sendo, dessa forma, uma base para as funcionalidades oferecidas pelo NFC nesse modo de operação. O protocolo LLCP é usado pelo protocolo SNEP, Simple NDEF Exchange Protocol, com orientação à conexão, permitindo a troca de mensagens quando usado o modo P2P de forma confiável e segura, quando o formato de dados é conhecido, ou seja, já está definido o NDEF.

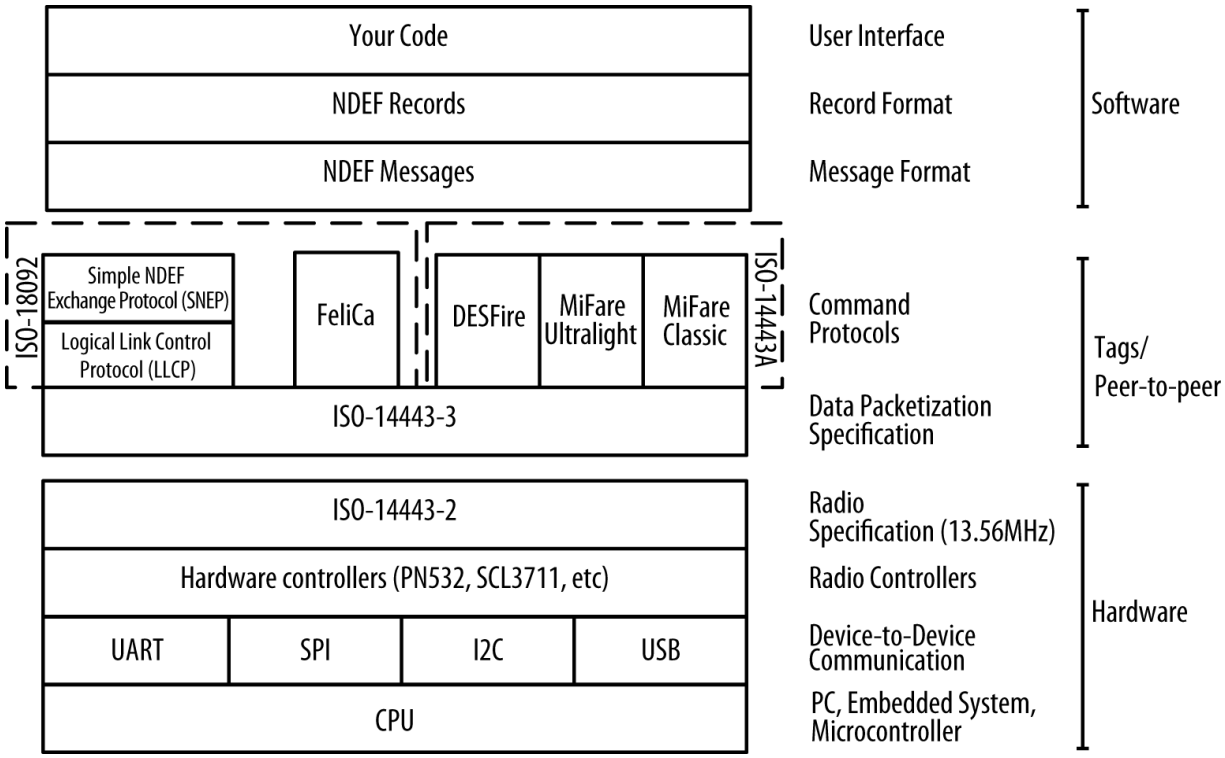
2.6.5 Arquitetura

Conforme descrito por Igoe, Coleman e Jepson (2014, pág. 16), a arquitetura NFC é composta de diversas camadas. Na camada física está presente o hardware que compõem a tecnologia, especificados na ISO-14443-2, que descreve o uso de comunicação de rádio frequência operando em 13,56 MHz. Dentro do dispositivo que oferece suporte NFC está presente o controlados NFC, que se comunica com o processador principal por meio de um ou mais padrões de transmissão de dados em série. Logo acima da camada física estão os protocolos e ISO's que padronizam as formas de comunicação. A ISO-14443A usada para transmissão RFID usada nos modos de leitura e escrita de *tags* e a ISO-18092 que define o modo de troca de dados na comunicação P2P, onde estão presentes os protocolos LLCP e SNEP.

As principais diferença entre o NFC e o RFID é a ISO-18092 para transmissões P2P e o uso do NDEF para padronizar o formato de uma mensagem.

A figura 7 mostra como estão espalhadas as camadas da arquitetura presente em um dispositivo NFC.

Figura 7 Arquitetura NFC



Fonte: Igoe, Coleman e Jepson (2014, pág. 16)

2.6.6 Codificação

Como visto anteriormente, de acordo com o modo de transmissão e de características dos dispositivos existem diferenças entre o modo como são enviados os bits em uma transmissão por meio do NFC. Estas codificações existem para evitar que os sinais transmitidos sejam interpretados de forma errada, ou seja, ao transmitir um bit 0 ou 1 não ocorra um erro de leitura destes dados.

2.6.6.1 Codificação Manchester

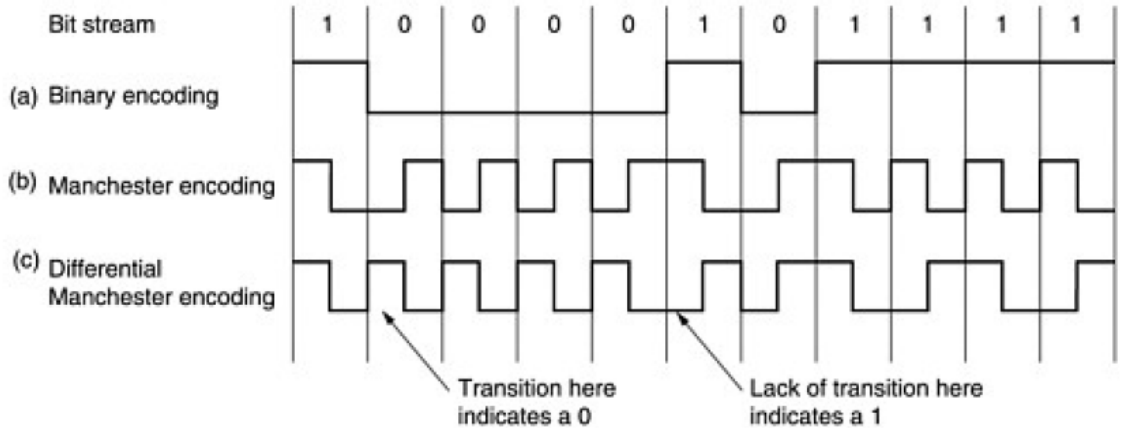
De acordo com Tanembaum (2003, p. 218)

> Na codificação Manchester, cada período de bits é dividido em dois intervalos iguais. Um bit 1 binário é enviado quando a voltagem é definida como alta durante o primeiro intervalo, e como baixa no segundo intervalo. Um bit 0 binário é exatamente o oposto: primeiro baixo, e depois alto. Esse esquema garante

> que cada período de bit terá uma transição na parte intermediária, tornando fácil para o receptor sincronizar-se com o transmissor.

Dessa forma, com a utilização da codificação Manchester, é criada a capacidade do receptor visualizar claramente o início e o fim de cada bit durante a transmissão. Existe também a codificação Manchester Diferencial, baseada na codificação Manchester. A diferença entre elas é que, na codificação Manchester o bit 1 é indicado com uma ausência de transição no início do intervalo enquanto o bit 0 tem a presença da transição, conforme Tanenbaum (2003), mostrado na figura 7.

Figura 8 a) Codificação binária, b) Codificação Manchester e c) Manchester Diferencial



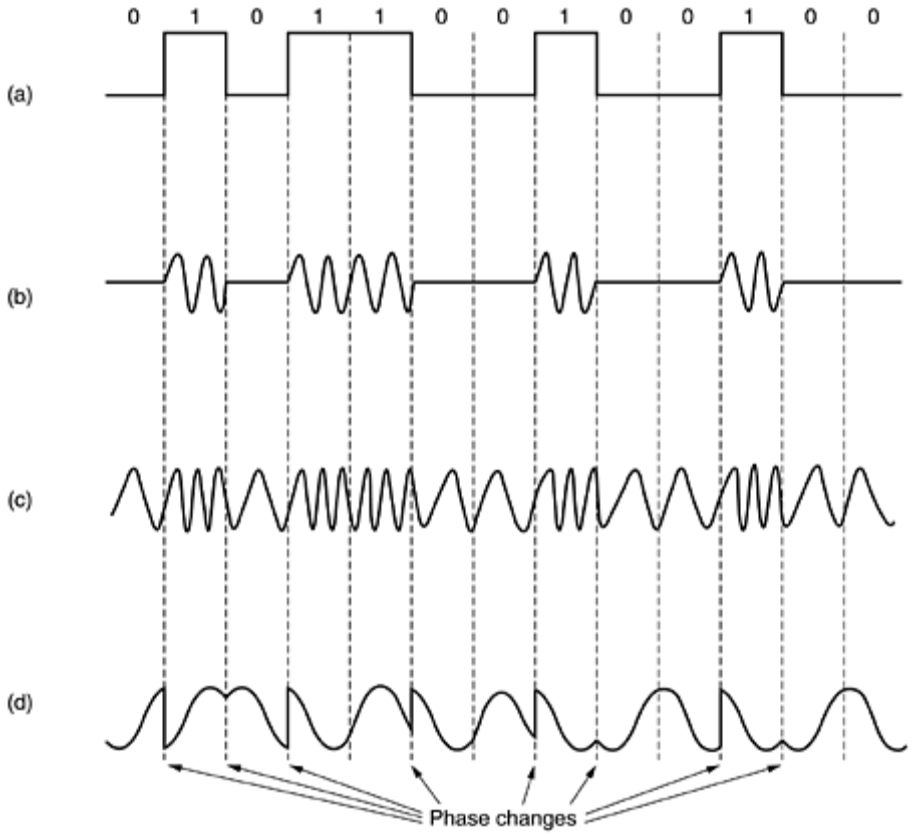
Fonte: Tanenbaum (2003)

A codificação Manchester é usada nos modos de comunicação passivo e ativo e nos modos de utilização de leitura/escrita, simulação de cartão e P2P, a depender do dispositivo receptor.

2.6.7 Modulação

A modulação se refere à transformação do sinal digital em analógico e pode ser feita de vários modos. Tanto a modulação ASK quanto a modulação BPSK estão presentes nos modos de comunicação ativo e passivo. A Figura 8 mostra a diferença de cada tipo de modulação em comparação a um sinal binário.

Figura 9 a) Sinal binário, b) Amplitude, c) Frequência e d) Fase



Fonte: Tanenbaum, 2003

2.6.7.1 Modulação ASK

A Amplitude Shift-Keying ou Modulação por chaveamento de amplitude é usada na comunicação NFC em ambos os modos ativo e passivo e de acordo com Filho (2010, p. 86) "é uma forma de modulação em que um sinal digital é representado por variações na amplitude de uma onda portadora por meio do chaveamento da mesma. Fase e frequência são constantes". Dessa forma a diferença entre o sinal digital 1 e o 0 é a amplitude da onda. Ferraz Filho (2010) ainda explica que a modulação ASK pode ser implementada facilmente e com um baixo custo, muito usada em transmissores por LED onde o sinal de luz é representado pelo nível lógico alto enquanto o nível lógico baixo representa a ausência de luz.

Essa modulação também é conhecida por 2-ASK devido às duas possibilidades: nível lógico alto e nível lógico baixo.

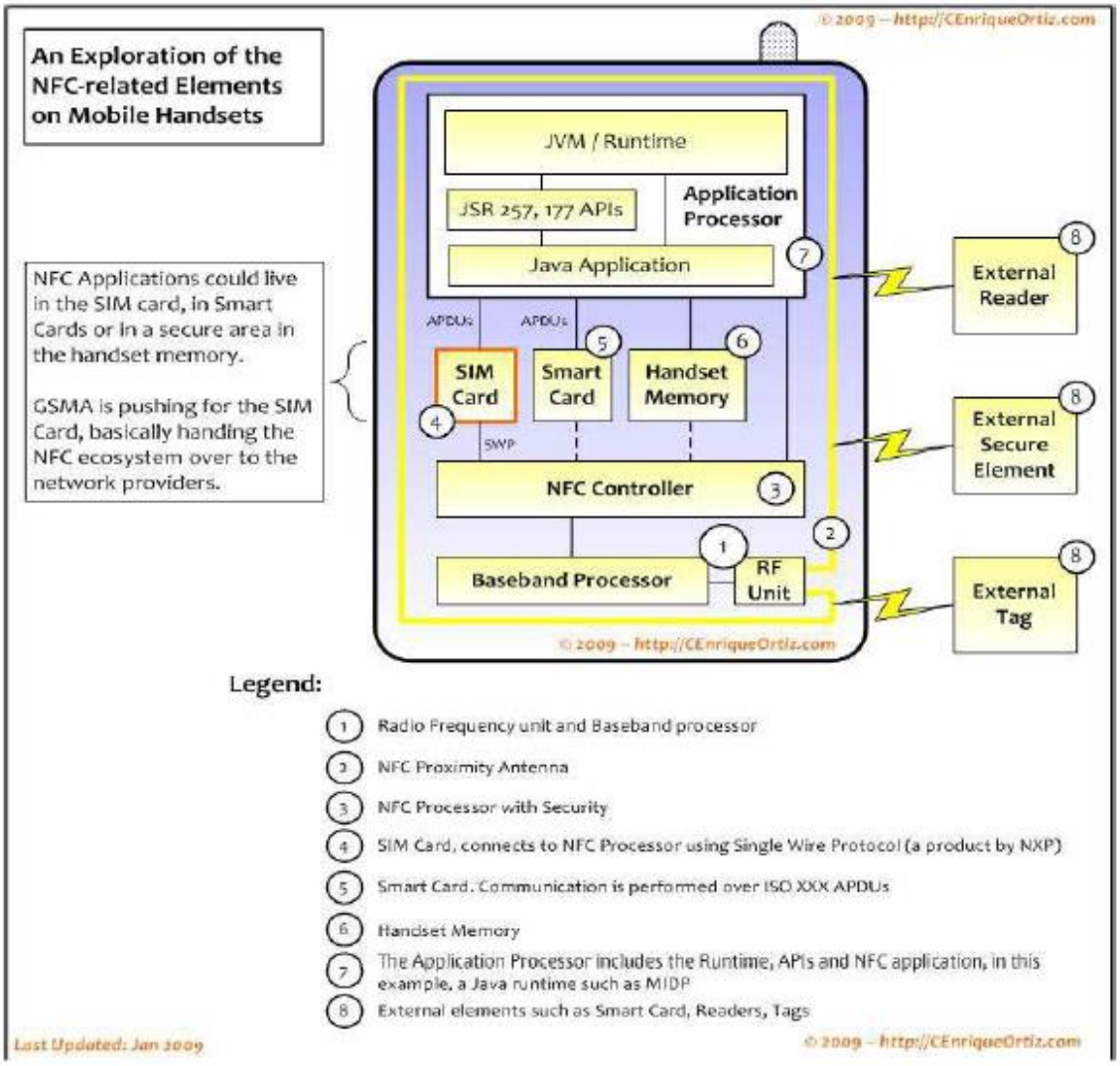
2.6.7.2 Modulação BPSK

A Bit-Phase Shift Keying ou Phase Shift Keying trabalha, de acordo com Ferraz Filho (2010), modulando a onda portadora por meio de chaveamento de fase. Isso é, em cada mudança de sinal digital, há a mudança na fase da onda.

2.6.8 NFC em smartphones

A Figura 10 demonstra alguns componentes presentes em um smartphone com a tecnologia NFC habilitada.

Figura 10 Elementos de um dispositivo móvel com NFC



Fonte: Curran, Millar e Garvey, 2012

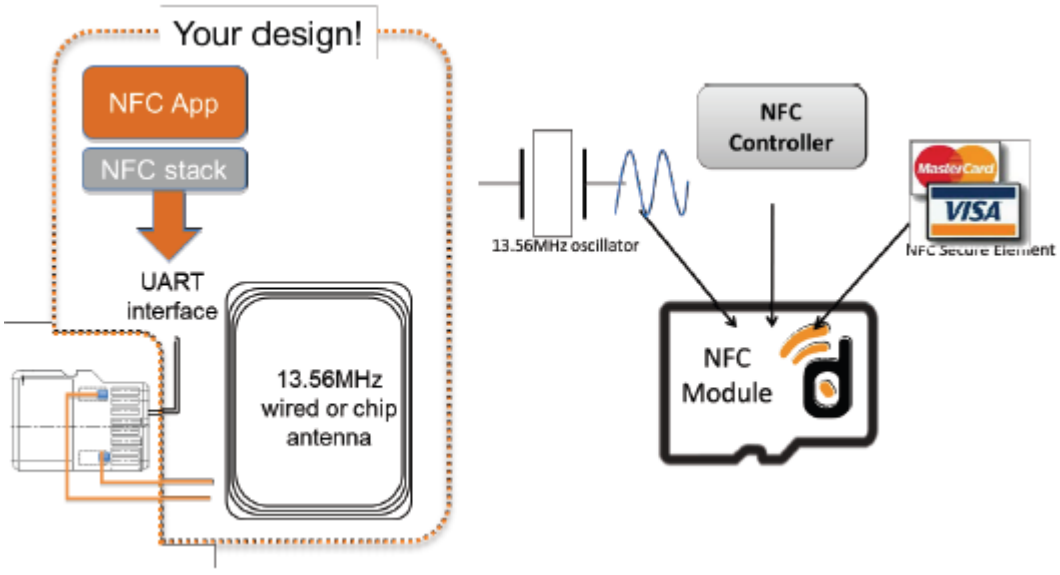
Quanto ao uso do NFC em smartphones, existem algumas opções sobre o local em que o NFC estaria inserido fisicamente como dentro de um cartão SIM[6], embutido em um *smart card*, ou em uma área física da memória do aparelho. No caso do uso do NFC embutido dentro do cartão SIM, o controle passaria para os provedores de telefonia móvel, não mais sendo dos fabricantes dos celulares. (CURRAN, MILLAR e GARVEY, 2012).

De acordo com Levandoski et al. (2011),

> O principal requisito para a utilização em massa da Carteira Eletrônica é a disponibilidade de equipamentos que possuam a tecnologia NFC integrada. Atualmente não existem muitos modelos disponíveis, mas estima-se que grande parte dos aparelhos smartphones produzidos introduzam esta tecnologia. Para os modelos atuais que não possuem tal característica, existe uma solução de kits para adaptação da tecnologia NFC. Estes kits possuem antenas e adaptadores para o cartão SIM ou com cartão MicroSD com antena NFC, como é o caso do modelo In2play, desenvolvido pela empresa Device Fydelity.

Como se pode ver, mesmo que a tecnologia NFC esteja completando uma década de existência e o primeiro celular com a capacidade de fazer leitura de *tags* NFC tenha sido anunciado no ano de 2004, de acordo com Want (2006), ainda existem poucos aparelhos disponíveis para uso dessa tecnologia.

Figura 11 Composição de um módulo MicroNFC



Fonte: Device Fidelity, 2013.

[6] SIM: Sigla de subscriber identity module é um smart card inserido dentro de um cartão de plástico usado nos telefones celulares e smartphones da família GSM.

A empresa DeviceFidelity, Inc desenvolveu um adaptador NFC dentro de um cartão MicroSD chamado de *MicroNFC Module*. Segundo Device FIdelity (2013), esse modulo foi desenhado para facilitar o desenvolvimento de dispositivos NFC e contém todos os componentes necessários para que o dispositivo opere três modos disponíveis. Dentro de um módulo, Como pode ser visto na figura 11, encontram-se:

- Segunda geração de controladores NFC.
- *Secure Element.*
- Circuito para conexão com a antena.

2.6.9 Segurança em NFC

No aspecto segurança, existe a chamada *Secure NFC*, característica do uso da tecnologia em smartphones, que faz uso de um *smart card* junto com o NFC, sendo capaz de armazenar informações pessoais de forma segura. Isso significa que os dados podem ser criptografados e a chave ser armazenada na memória do dispositivo. Esse elemento é chamado de Secure Element (SE) e, segundo o NFC Forum (2009), além de armazenar de forma segura e confiável os dados dentro do dispositivo, pode fazer isso de três formas. O cartão SIM pode funcionar como um SE, ele pode estar dentro do dispositivo ou então embutido em um cartão MicroSD que pode ser inserido ao dispositivo. O SE funciona junto com o controlador NFC e é indispensável no caso de aplicações que trabalham com pagamento e autenticação, entre outros.

Além disso, em alguns casos, o próprio dispositivo NFC oferece o serviço de autenticação. Assim, conforme Curran, Millar e Garvey (2012), esse método de armazenamento seguro vai ser necessário para guardar informações pessoais, chaves de criptografia, dinheiro digital, tornando isso um importante aspecto de um dispositivo NFC. Vale salientar que os aspectos de segurança e ameaças não fazem parte do escopo deste trabalho.

Embora seja considerada uma tecnologia para transmissão segura de dados, ainda existem alguns pontos que devem ser observados e reavaliados em relação ao NFC. A necessidade de os dispositivos estarem bem próximos um do outro é um fator que ajuda muito na segurança durante a transmissão de dados, pois diminui drasticamente as chances de haver uma escuta, entretanto há outros pontos que apresentam certa fragilidade e que, se não tratados, podem ser explorados. De acordo com a ISO, a tecnologia NFC não apresenta uma

criptografia por padrão, cabendo a cada aplicação o desenvolvimento de uma criptografia na hora do envio da informação. Esse desenvolvimento, provavelmente, irá ocorrer com o tempo, mas por ser algo útil, e não por obrigação. A transmissão de dados sem fio pode ser escutada por antenas e, em casos piores, tendo seu dados eliminados ou modificados. Quando ocorre a transmissão de forma passiva é mais difícil que isso ocorra devido ao alcance de captura ser reduzido, mas em uma comunicação ativa, quando os dois dispositivos estão gerando campos de rádio frequência, a possibilidade de escuta é maior. Dispositivos NFC são feitos para serem usados em diferentes ambientes, com diferentes níveis de segurança, cabendo aos desenvolvedores e ao próprio usuário estarem cientes de que proteção é algo que não deve ser deixado de lado e que os perigos não podem ser subestimados.

2.6.10 Onde utilizar NFC

2.6.10.1 Operações de leitura e escrita

As operações de leitura e escrita mais conhecidas são as de leitura de *tags* em geral, principalmente no formato de *smart posters*, os quais geralmente contêm informações rápidas como URLs, e-mails e telefones, que ficam armazenados no dispositivo leitor.

Além disso, utilizando *tags* pré-configuradas existe a possibilidade de um rápido acesso a diversas ferramentas de um smartphone. Por exemplo, uma *tag* posicionada no suporte do smartphone dentro do carro pode fazer com que, ao posicionar o celular próximo a *tag*, inicie o aplicativo de GPS. Ou uma *tag* posicionada ao lado da cabeceira da cama defina as configurações do aparelho para modo silencioso e o alarme para as 7h, por exemplo.

2.6.10.2 Operações de emulação de cartão

Operações básicas de pagamento de passagens ou entradas em ambientes diversos se tornam possíveis por meio da inserção de dados de cartões de crédito dentro do dispositivo. De acordo com Haselsteiner e Breitfub (2007) , quando o usuário quiser pagar algo, ele deverá aproximar o dispositivo de um leitor, que fará a verificação das informações recebidas e processará ou não o pagamento. O pequeno tamanho de uma *tag* faz com que a leitura de um documento e o preenchimento de um cadastro possa se tornar mais fácil se esse documento conter uma *tag* e nela estiverem contidos os dados de identificação do portador.

Com a tecnologia tomando conta do setor automobilístico, até a chave do carro pode ser facilmente substituída por um dispositivo NFC, sendo que este dispositivo pode ser o próprio smartphone.

2.6.10.3 Operações ponto-a-ponto

Podendo se comunicar na forma P2P, não há a necessidade do uso de *tags*. Nessas situações os dois dispositivos devem possuir a capacidade de ler / enviar dados. Haselsteiner e Breitfub (2007) explicam que é possível fazer a sincronização de dois dispositivos como, por exemplo, um notebook e uma câmera digital, para que seja feita a transferência de imagens da câmera para o notebook via Bluetooth, por exemplo. Dessa forma, com a utilização do NFC, aproximando os dispositivos, a conexão Bluetooth seria feita de forma mais rápida. A transmissão da imagem continua sendo feita via Bluetooth, já que o NFC não suporta transferência de imagens por oferecer somente o envio de pequenas quantidades de dados.

2.6.10.4 Google Wallet

A carteira digital da Google, chamada Google *Wallet*, é um serviço da Google que oferece a possibilidade de cadastramento de diversos cartões de crédito e débito de forma digital em sua conta no Gmail, o servidor de e-mails da Google. Além de armazenar, é possível realizar transferências e pagamentos por meio desse serviço com um smartphone com NFC habilitado. Atualmente, a possibilidade de pagamentos por meio de smartphones só está disponível nos Estados Unidos, de acordo com o Google Inc (2013).

2.6.10.5 PagSeguro NFC

Muito semelhante ao Google Wallet, é um serviço da PagSeguro que agora permite, além de compras com dinheiro virtual, o pagamento por meio do aplicativo para smartphones da PagSeguro. De acordo com PagSeguro (2013),

> Comprador e vendedor deverão ter o aplicativo PagSeguro NFC baixado em seus celulares. No momento da compra, o vendedor digitará o valor da transação no celular do estabelecimento e solicitará que o comprador aproxime o aparelho para a validação da compra. Neste momento o comprador escolhe a forma de pagamento.

2.6.10.6 Pagamentos no Brasil

Pagamentos por meio do Google Wallet ainda não estão disponíveis no Brasil e para a utilização do PagSeguro NFC é necessário ir a um dos, até então, poucos estabelecimentos que já usam a aplicação. No entanto, o Governo Brasileiro já está se adiantando e criando medidas para regulamentar o uso de smartphones para pagamento no país.

Conforme o jornal *Zero Hora* (2013), uma medida provisória (MP) recentemente editada pelo governo federal estabelece as bases jurídicas para estimar a implantação do uso do aparelho para esse fim. Na mesma reportagem, o jornal acrescenta ainda que o Banco Central terá 180 dias para estabelecer as normas que deverão ser seguidas pelas operadores de telefonia. Objetiva-se que com a MP seja feito um melhor uso do sistema além de aumentar a concorrência entre as fornecedoras do serviço, que deverão se enquadrar nas normas estabelecidas dentro de um prazo estipulado. Além das operadoras de celulares, administradores de cartões de crédito e empresas que oferecem pagamento pela internet também serão enquadradas na nova legislação. A tecnologia NFC é uma das tecnologias discutidas para esse propósito, sendo que já existem mais de 500 mil máquinas Cielo habilitadas para transações desse tipo.

2.6.10.7 Android Beam

Android Beam é uma função do sistema operacional para dispositivos móveis Android para dispositivos que ofereçam suporte NFC. Ela permite um rápido compartilhamento de informações entre dois dispositivos Android. De acordo com Android Developers (2013), Android Beam permite uma simples troca de dados P2P entre dois dispositivos Android ligados. Para que isso seja feito, alguns requisitos devem ser satisfeitos como:

- A aplicação que irá fazer o envio deve estar em primeiro plano.
- O dispositivo que irá receber os dados não pode estar com a tela travada.

Após isso, quando os dispositivos estiverem próximos o suficiente, aparecerá na tela do "*Touch to Beam*", que significa "Toque para enviar". Nessa etapa, o usuário decidirá se irá ou não proceder o envio.

Páginas da Web, vídeos do Youtube e contatos telefônicos são alguns exemplos de dados que podem ser transferidos com o uso dessa função disponível no sistema operacional Android a partir da versão 4.0, também conhecida como *Ice Cream Sandwich*.

NFC oferece um grande número de possibilidades de uso que podem e já estão sendo exploradas presentes em nosso cotidiano, como pode ser visto na Figura 12.

Figura 12 NFC no uso diário

Fonte: NFC Forum (c), 2013.

2.7 ANDROID e NFC

O sistema operacional Android foi desenvolvido por meio de uma parceira entre a Google e diversas outras empresas participantes da OHA - Open Hanset Allianc - como Motorola, LG, Samsumg, Sony Ericsson, entre outras. Inicialmente um projeto para ser usado em dispositivos celulares, hoje está presente também em tablets, netbooks e até mesmo relógios. Conforme Lecheta (2009, pág. 20), a OHA foi criada com o objetivo de padronizar uma plataforma de código aberto e livre, com o objetivo de atender a demanda de um mercado em intenso crescimento, ainda no ano de 2008. Por meio do Android, a Google entrava no ramo de celulares para brigar por um espaço até então sem dono, porém disputado.

Lecheta (2009, pág. 21) explica que "o objetivo do grupo é definir uma plataforma única e aberta para celulares para deixar os consumidores mais satisfeitos com o produto final. Outro objetivo principal dessa aliança é criar uma plataforma moderna e flexível para o

desenvolvimento de aplicações corporativas". Assim nascia o Android, uma plataforma de desenvolvimento para aplicativos móveis baseada em Linux. Desenvolvido na linguagem de programação Java, o sistema Android é executado sobre uma máquina virtual chamada de *Dalvik Virtual Machine*, atuando sobre Kernel Linux 2.6. O sistema operacional Android é uma das plataformas que permite a utilização integrada com a tecnologia Near Field Communication. Existem outras que também já permitem seu uso como o sistema Windows 8 e as placas Arduino com o chip NFC acoplado.

2.7.1 Versões do Android

Desde seu lançamento em outubro de 2008, quando o HTC G1, primeiro celular a possuir o Android como sistema operacional, foi lançado o Android vem passando por constantes atualizações e aprimoramentos. Nessas novas versões novos recursos são adicionados, antigos erros são corrigidos e algumas características são alteradas, tanto no uso do dispositivo quando no desenvolvimento das aplicações para ele. Atualmente, o Android está na versão 4.4 Kitkat, sendo essa a API - *Application Programming Interface* - Level 19. Conforme Android Developers (a), o API Level é um valor inteiro que identifica (diferencia) as diversas versões da plataforma Android. O desenvolvimento das API's é feito de modo que a API mais nova é compatível com a anterior. A maior parte das mudanças em novas API ocorre de adições ou substituições de funcionalidade, sendo que as partes novas são atualizadas e as partes antigas se tornam obsoletas, mas não são removidas, de modo que antigas aplicações podem ainda fazer uso delas. Em poucos casos há a modificação ou remoção de qualquer parte da API. Isso ocorre principalmente quando é necessário para melhorar o rendimento do sistema ou por questões de segurança.

Na hora de desenvolver uma aplicação para o Android é necessário escolher para qual API se está programando, além de configurar uma API mínima para o uso daquela aplicação.

2.7.1.1 Android 2.3.3 Gingerbread - API Level 10

A tecnologia NFC está presente na plataforma Android desde fevereiro de 2011, conforme explica Android Developer (c). Neste mês foi lançado o Android 2.3.3 Gingerbread, com API *Level* 10 e entre seus principais incrementos estão o suporte à NFC. É claro que,

para sua utilização, o hardware do dispositivo também deve oferecer esse mesmo suporte. Nessa versão já era possível uma comunicação no formato leitura e escrita com diversos padrões de *tags*, como:

- NFC-A (ISO 14443-3A
- NFC-B (ISO 14443-3B)
- NFC-F (JIS 6319-4)
- NFC-V (ISO 15693)
- ISO-DEP (ISO 14443-4)
- MIFARE Classic
- MIFARE Ultralight
- NFC Forum NDEF tags

Android Developer (c) diz ainda que estava inclusa uma limitada comunicação P2P, onde a aplicação em primeiro plano podia gerar mensagens no formato NDEF e enviar para outro dispositivo NFC quando conectados. O processo de detecção de uma *tag* se tornou mais controlado pelas aplicações, permitindo que, por exemplo, aplicações em primeiro plano consigam ter o controle do evento de detecção da *tag* ou ainda que essas mesmas aplicações sejam capazes de esperar uma *tag* com conteúdo específico para ser lido.
Os métodos pelas quais as aplicações fazem isso serão explicados mais adiante, junto com as classes que compõem o pacote para desenvolvimento de uma aplicação que utilize NFC.

2.7.1.2 Android 4.0 Ice Cream Sandwich - API Level 14

Foi em outubro de 2011, com o lançamento da versão de Android 4.0, que passou a estar disponível em dispositivos Android com a tecnologia NFC a funcionalidade do Android Beam. Conforme Android Developer (d), o Android Beam permite o envio de mensagens no formato NDEF para outro dispositivo Android utilizando comunicação P2P. A transferência dos dados é iniciada quando dois dispositivos Android estão a uma distância inferior a 4cm. Normalmente suas partes traseiras devem estar encostadas. A mensagem NDEF pode conter qualquer informação. Normalmente são compartilhados contatos, vídeos do Youtube, páginas abertas em um navegador, entre outros.

Métodos, classes e outros itens envolvidos no uso do Android Beam serão explicados em outro tópico.

2.7.1.3 Android 4.1 Jelly Bean - API Level 16

Em uma atualização do sistema Android para verão 4.1, em junho de 2012, foi adicionada a capacidade de, por meio da tecnologia NFC, dois dispositivos serem pareados pelo Android Beam para realizar uma transferência via Bluetooth. Isso poderia ser necessário quando a quantidade de dados fosse muito grande para uma transferência NFC, que é mais lenta que o Bluetooth. Dessa forma o processo de pareamento dos dispositivos, algo que normalmente leva um certo tempo para o reconhecimento, seria feito de forma rápida, por meio do Android Beam e a transferência dos dados ser feita via Bluetooth, conforme Android Developers (e).

2.7.1.4 Android 4.4 Kitkat - API Level 19

Foi somente na última atualização do sistema operacional Android, em outubro de 2013, que se tornou possível o uso de um dispositivo Android no modo emulação de cartão. Conforme Android Developer (f), por meio do HCE - Host Card Emulation - é possível trabalhar com pagamentos, programas de fidelidade, cartões de acesso/passagem além de outros serviços. Uma aplicação pode simular um *smart card* NFC, permitindo aos usuários iniciar transações com outros aplicativos dispensando a presença de um SE.

Ainda conforme Android Developers (f), é possível emular cartões baseados na ISO/IEC 7816, que usam o protocolo ISO/IEC 14443-4 para transmissão de dados. O Android usa AIDs - Application Identifiers, definidos na ISO/IEC 7816-4 para rotear as transações para a aplicação correta.

Cada aplicação deve declarar os AIDs suportados no arquivo androidmanifest.xml, junto com uma categoria que indica o tipo de suporte. No caso em que várias aplicações suportam o mesmo AID na mesma categoria, será mostrada uma caixa de diálogo onde o usuário poderá escolher a aplicação, conforme Android Developer (f).

Android Developer (f) complementa explicando que, para funcionar no modo HCE é necessário um controlador NFC no dispositivo. A maioria dos controladores NFC oferecem suporte a transações HCE e SE.

2.7.2 Classes do Android

2.7.2.1 Classe *Activity*

A classe *Activity*, de tradução Atividade, é uma das principais classes no sistema operacional Android. Normalmente, essa classe apresenta a mesma ideia que uma tela da aplicação. Conforme Lecheta (2008, pág. 93), a classe *Activity* é responsável por tratar os eventos em sua respectiva tela. Toda tela de uma aplicação Android deverá estar associada a uma *Activity*, por isso seus conceitos seguidamente se confundem. Além dos eventos, a passagem de uma tela para outra também é controlada na *Activity*, sendo que essa passagem, enviando ou não parâmetros, pode acontecer de duas formas: a primeira forma simplesmente inicia a segunda *Activity*, que tem uma vida própria pode-se dizer assim, enquando na segunda forma, o modo como a nova *Activity* é iniciada indica que ela deverá retornar algum valor para a *Activity* inicial, podendo ser considerada uma *Activity* filha da mesma.

2.7.2.2 Classe *Intent*

Na plataforma Android, quando você deseja fazer algo como por exemplo, abrir uma nova tela, não é você fará o processo. É o próprio sistema operacional, por meio do envio de uma mensagem de requisição, que fará o processo de abertura de uma nova tela. Essa mensagem é representada pela classe *Intent*, que significa Intenção e que na verdade, é assim mesmo que pode ser entendida. Através da passagem por parâmetro de uma *Intent* é possível iniciar uma nova *Activity*. Além disso, não é necessário que a intenção seja o uso de uma A*ctivity* da própria aplicação, pois a plataforma Android permite uma fácil integração entre as aplicações. É possível que uma aplicação acesse os contatos, câmera, GPS, simplesmente por meio da declaração de uma intenção. Ao iniciar uma *Activity* com essa intenção, o Android fará uma busca em sua base de dados por uma aplicação que seja capaz de responder a essa intenção, que será iniciada logo em seguida. Caso existir mais que uma aplicação capaz de resolver as necessidades daquela intenção, será disponibilizado uma caixa de opção onde o usuário deve escolher qual *Activity* usar. Lecheta (2008, pág. 136) diz "O interessante de uma mensagem enviada por uma *Intent* é que nem sempre ela é recebida pela própria aplicação que a criou, frequentemente sendo utilizada para integrar aplicações".

2.7.2.3 Classe *Intent-Filter*

Conforme descrito na seção sobre *Intent*, o sistema operacional busca aplicações capazes de resolver uma *Intent* quando a mesma é enviada. O *Intent-filter* ou como também é

chamado, filtro de intenções é o meio pelo qual o Android localiza as aplicações que resolvem determinada *Intent*. Esse filtro é declarado no arquivo AndroidManifest.xml, dentro da *Activity* que será iniciada quando a *Intent* for filtrada. Lecheta (2008, pág. 272) diz que "o Android permite definir uma ação que identifica a *Intent*, de forma que quando a mensagem for enviada ao sistema operacional ela seja identificada por essa ação. Somente uma *Activity* que esteja mapeada para aquela ação será executada.".

2.7.2.4 Classe NfcAdapter

A classe NfcAdapter representa o controlador NFC presente no dispositivo. Um objeto dessa classe pode ser obtido pelo método *getDefaultAdapter*(*Context*), retornando nulo caso o dispositivo não ofereça suporte à NFC.

O controlador oferece constantes com a finalidade de comparar com as ações de uma *Intent*, quando, por exemplo, for lida uma *tag*. Estão ainda presentes nessa classe métodos para verificar se o NFC do dispositivo está habilitado, habilitar e desabilitar o envio de NDEF quando a aplicação estiver em primeiro plano, habilitar e desabilitar a leitura de uma *tag* para somente quando a aplicação estiver em primeiro plano, gerar NDEF para enviar por meio do Android Beam, verificar se foi recebida alguma mensagem por meio do Android Beam, entre outros.

2.7.2.5 Classe NdefMessage

A classe NdefMessage representa uma mensagem no formato NDEF. Esse formato para encapsulamento de dados é especificado no NFC Forum. Um objeto do tipo NdefMessage é capaz de armazenar diversos objetos do tipo NdefRecord e é o conteúdo que será gravado em *tags* ou enviado por meio do Android Beam, por exemplo. Seu principal método é o *getRecords()*, cujo retorno é um vetor com todos os *records* que a mensagem contém.

2.7.2.6 Classe NdefRecord

A classe NdefRecord representa um *record*, definido no NFC Forum. Cada *record* contém um payload que é o próprio conteúdo do record e um *header* que contém informações relativas aos tipos, tamanho e identificador do *payload*. Essa classe tem alguns métodos para criar tipos específicos de records como URI's e AAR - Android Application Record. Uma

mensagem com um *record* do tipo AAR irá fazer com que o sistema procure a aplicação definida para ser lançada a partir da leitura da mensagem.

2.7.2.7 Classe *Tag*

Android Developers (i) explica que a classe *Tag* representa o estado de uma *tag* NFC no momento em que é feito a leitura. Cada vez que uma *tag* é descoberta pelo dispositivo, ou seja, é aproximada o suficiente para ser detectada, um novo objeto é criado. A descoberta de uma *tag* pela aplicação ou dispositivo pode ser feita de quatro formas:

- **Foreground activity dispatch**: Ocorre quando uma *Activity* declara o método *NfcAdapter. enableForegroundDispatch()*, fazendo com que essa *Activity* tenha prioridade sobre qualquer *tag* que tenha sido descoberta enquanto ela estiver em primeiro plano da aplicação.
- **NDEF data dispatch**: Ocorre quando, no primeiro *record* da mensagem, o TNF está declarado como um tipo MIME, URI ou então um dos padrões RTD especificados pelo NFC Forum. Dessa forma, uma *Activity* que tenha declarado o filtro ACTION_NDEF_DISCOVERED será escolhida para operar a *tag* descoberta. Se mais de uma aplicação tiver o mesmo filtro será mostrada uma caixa de diálogo para que o usuário faça a escolha da aplicação que será utilizada. Se não houver nenhuma *Activity* registrada para isso a expedição passa para o próximo estágio.
- **Tag Technology dispatch**: Existem diversos tipos de *tags* e, cada *tag*, pode suportar algumas tecnologias. O Android faz a busca por aplicações que consigam trabalhar com uma ou mais das tecnologias presentes na *tag* descoberta. Para isso, essas aplicação deverão ter declarado o filtro ACTION_TECH_DISCOVERED com as respectivas tecnologias.
- **Fall-back dispatch**: Ocorre quando nenhuma *Activity* está registrada para os casos anteriores. Esse tipo de descoberta é usado quando se deseja escrever em uma *tag*. Mesma na operação de escrita é necessário a descoberta de uma *tag* e isso pode ser feito por meio do filtro ACTION_TAG_DISCOVERED.

A classe *Tag* oferece métodos para ser obter seu identificador, se houver. Esse identificador é um número serial de baixo nível usado para identificação e anti-colisão. Algumas *tags* apresentam um identificador estável, outras geram um novo cada vez que são lidas e algumas não possuem nenhum, conforme Android Developers (i).

2.7.2.8 Classe Ndef

A classe Ndef representa o formato NDEF definido pelo NFC Forum e, de acordo com Android Developers (j), fornece os métodos para obter e modificar o conteúdo de um NdefMessage em uma *Tag*. Todas os quatro tipos de *tags* podem, por padrão, trabalhar com o conteúdo no formato Ndef. Ainda de acordo com Android Developers (j), todos os dispositivos Android que tenham a tecnologia NFC habilitada devem trabalhar com todos os métodos Ndef definidos nessa classe. Além das quatro *tags* que o dispositivo deve oferecer suporte, existem outros modelos de *tag* que podem ser utilizados, cada um com sua própria tecnologia.

3 ESTUDO DE CASO

Nesta seção serão detalhados os passos que foram seguidos para o desenvolvimento de uma aplicação simples, sem fins comerciais, para testar algumas funcionalidades da tecnologia NFC na prática. Com a finalidade de dar um propósito para a aplicação, ela foi desenvolvida para trabalhar em um dispensador de senha.

Dispensadores de senha são largamente utilizados em estabelecimentos que necessitam de uma organização no atendimento como bancos, cartórios, hospitais, entre outros. A senha dispensada pelo gerenciador somente é utilizada fisicamente, sendo mostrada ao atendente na hora do atendimento após a mesma tendo sido chamada no painel e logo após descartada. E se essa senha, impressa em uma pequena fita de papel, pudesse ser dispensada e armazenada digitalmente? Para isso, foi desenvolvida uma aplicação capaz de gravar os dados presentes nessa senha, no caso, a numeração e o segmento pertencente, em uma *tag* NFC. Junto com essa primeira aplicação, será usada uma outra capaz de fazer a leitura dos dados contidos nessa *tag*, mostrando-os na tela em seus respectivos campos. Dessa forma, uma aplicação simula o dispensador de senha enquanto a outra aplicação faz o lado do celular do cliente, que consegue ler a senha digitalmente após a mesma ter sido gerada. Após ter feito a leitura, a senha estará visível na aplicação em seu celular, podendo ser mostrada ao atendente.

O modo de operação como leitura e escrita foi escolhido por ser o mais acessível devido a ausência de outro celular que oferecesse suporte NFC no momento. As aplicações criadas são:

GerenciadorDeSenha: faz o papel do dispensador, onde é possível escolher o segmento alvo e ativar o botão para gerar a senha que, automaticamente prepara o dispositivo

para gravar quando detectada uma *tag*. A tela da aplicação, assim como sua representação está presente na Figura 13. Nesta tela é possível observar como será feita a interação com o usuário utilizando um formulário onde deve ser escolhido um segmento. Esse formulário é composto por 4 campos de botões de rádio, sendo que é possível manter somente um campo marcado. Estando um campo marcado, será habilitado ao usuário a opção de gerar uma senha para o segmento escolhido. A senha gerada após o clique do botão "Gerar Senha" é mostrada no campo SENHA sendo composta por três caracteres identificados do campo por exemplo, "SOC" para o segmento social e outros três caracteres numéricos iniciando em "001" para indicar a posição daquela senha gerada para ser utilizada no atendimento. Essa numeração é diferente para cada segmento de forma que cada um tenha um contador específico.

Figura 13 Representação da aplicação Gerenciador de Senha

Fonte: autor

VisualizadorDeSenha: é o aplicativo que será aberto automaticamente quando for detectada uma *tag*. Serve como a senha em formato de papel, mas agora com os dados armazenados digitalmente. Sua função é mostrar na tela a senha armazenada na *tag*, juntamente com as outras informações lidas.

3.1 MODELAGEM

Com o objetivo de proporcionar um melhor entendimento do funcionamento das aplicações, essa seção mostrará por meio de diagramas o projeto de cada uma delas.

3.1.1 Análise de Requisitos

O objetivo de desenvolver duas aplicações que trabalhem com leitura e escrita de tags foi atingido. Para que essas aplicações funcionassem da maneira esperada alguns requisitos funcionais e não funcionais precisaram ser satisfeitos:

Requisitos funcionais - Gerenciador de Senha:

- O usuário deve ser capaz de escolher o segmento.
- O usuário deve ser capaz de gerar uma senha.
- A aplicação avisará o usuário o momento de aproximar a *tag*.
- A aplicação escreverá os dados na *tag* conforme escolha do usuário.
- Informar ao usuário o resultado das ações.

Requisitos não funcionais - Gerenciador de Senha:

- Verificar quanto a possíveis erros na hora de escrever na tag.
- Escrever um record especificando qual aplicação deve ser lançada quando for feita a leitura da *tag*.

Requisitos Funcionais - Visualizador de Senha:

- Mostrar a senha, o horário e o segmento lido ao usuário

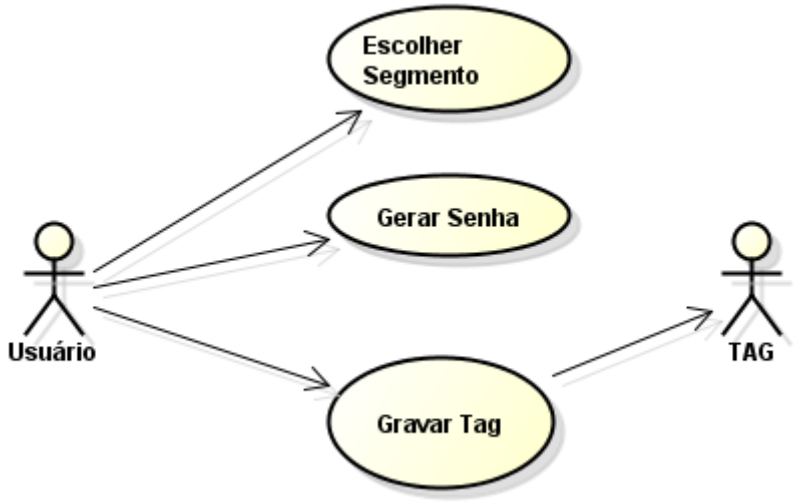
Requisitos Não Funcionais - Visualizador de Senha:

- Aplicação ser executada através da leitura da *tag*, sem ícone executável.

3.1.2 Diagrama de Caso de Uso

O diagrama de caso de Uso da aplicação Gerenciador De Senha pode ser visto na Figura 14. Nele estão presentes as ações que ocorrem entre usuário, sistema e *tag*. Como a aplicação representa o dispensador de senha, em um contexto real ela não estaria instalada no celular do usuário mas em uma máquina na entrada do estabelecimento. Os casos de uso representam as ações que o usuário iria fazer para interagir com o sistema e como o sistema interage com outras entidades externas, no caso a *tag*.

Figura 14 Diagrama de Caso de uso - Gerenciador de Senha



Fonte: autor

Escolher Segmento: Representa a interação do usuário com o formulário na hora de escolher o segmento. O formulário é composto por quatro segmentos representados por botões de rádio: Social, Empresa, Você, Caixa. O usuário deve escolher a senha a ser gerada através do clique em um dos quatro segmentos. Somente um segmento pode estar marcado.

Gerar Senha: Assim que houver um segmento escolhido pelo usuário estará habilitado o botão "Gerar Senha". O usuário pode clicar nesse botão para gerar uma senha para o segmento escolhido. Nesse momento é aberto um diálogo na aplicação com a inscrição "Aproxime de uma tag para escrever". Para isso, o dispositivo é colocado em um estado que, ao aproximar de uma *tag*, essa será escrita com os dados da senha gerada, da hora em que está sendo feita a gravação, do segmento escolhido, de uma URL e dos dados da aplicação que será executada quando for feita a leitura da *tag*. Se o usuário clicar na tela ou se uma *tag* for escrita o diálogo desaparecerá e o dispositivo cancelara o estado de gravação de *tag*. Essa alteração de estado faz uso do filtro de intenções do sistema operacional Android, permitindo que o dispositivo e a aplicação estejam preparados para tratar a detecção de qualquer *tag*, sem importar o seu conteúdo, já que o objetivo da aplicação é somente escrever na *tag*.

Gravar Tag: Enquanto o dispositivo estiver pronto para gravar a *tag* e uma for detectada, será iniciado o processo de gravação. Para isso é feita a criação da mensagem que será gravada, composta pelos seus *records*. Na hora de gravar são realizados testes para verificar se o tamanho da mensagem não excede a capacidade máxima da *tag*, se é possível alterar o conteúdo da *tag*, se a *tag* oferece a gravação no formato Ndef, entre outros. Para cada resultado uma mensagem de saída é mostrada ao usuário.

A Figura 15 mostra o diagrama de caso de uso da aplicação VisualizorDeSenha, que tem um único diagrama de caso de uso.

Figura 15 Diagrama de Caso de Uso - Visualizador de Senha

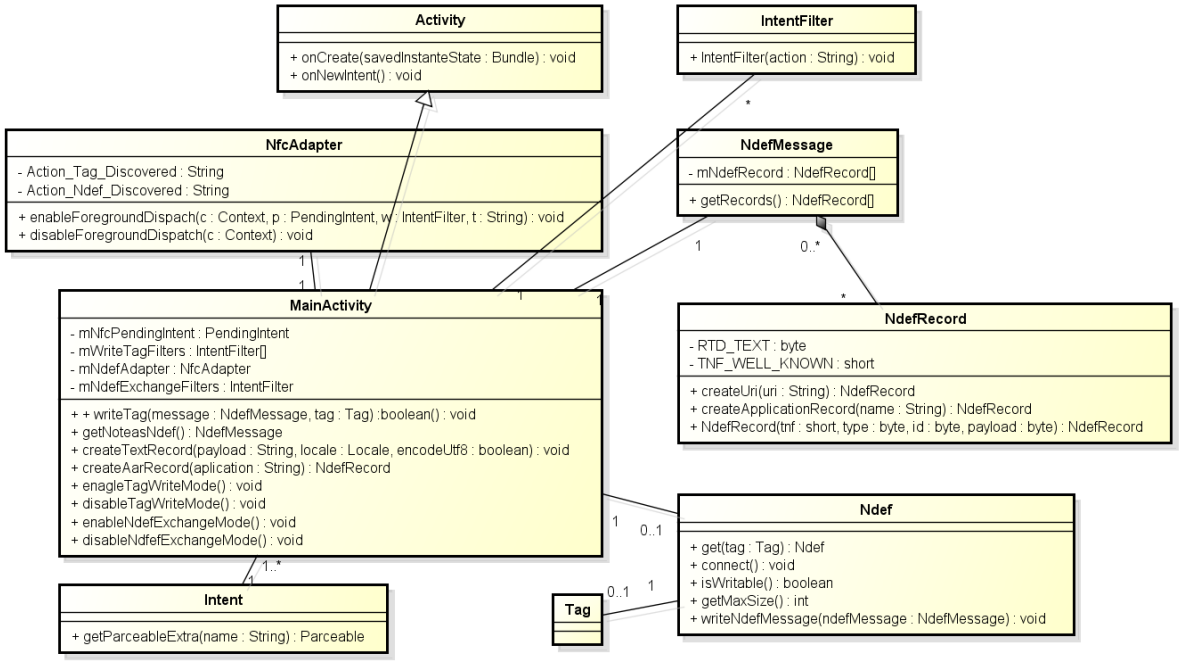
Fonte: autor

Ler Tag: A aplicação Visualizador De Senha é executada após a leitura de uma *tag* que contém um record que especifica como será feito o tratamento da mensagem contida naquela *tag*. Além dele, outros records contendo o segmento, senha, horário em que foi feita a gravação da *tag* e uma URL também são lidos e mostrado na tela dessa aplicação. Além do *record* especificando a aplicação a ser executada, para que o processo de lançamento da aplicação ocorresse foi necessário configurar o filtro de intenções de forma que quando uma *tag* com formato Ndef fosse detectada pelo dispositivo essa aplicação se dispusesse a fazer o tratamento dessa *tag*.

3.1.3 Diagrama de Classes

A Figura 16 mostra o diagrama de classes da aplicação Gerenciador De Senha, demonstrando as classes utilizadas na programação que trabalham com NFC e o métodos utilizados. Com exceção da classe MainActivity, todas as outras já foram mencionadas durante o referencial teórico. A classe MainActivity representa o fluxo principal de ambas as aplicações e é onde ocorre todo controle de estados e chamada de métodos. Ela trabalha diretamente com a tela da aplicação, sendo que também faz o gerenciamento dos eventos presentes nela, seja o de selecionar o segmento, do clique do botão de gerar senha e colocar o dispositivo em um modo para gravar em uma *tag* ou o evento de detecção de aproximação de uma *tag*. As outras classes são externas a aplicação, sendo utilizadas por esta e tendo seus métodos e atributos utilizados citados no diagrama de classes. Como o Android trabalha com versões, estas classes já estão disponíveis nas versões que oferecem suporte a NFC e não necessitam nenhum pacote extra para serem utilizadas.

Figura 16 Diagrama de Classes - GerenciadorDeSenha

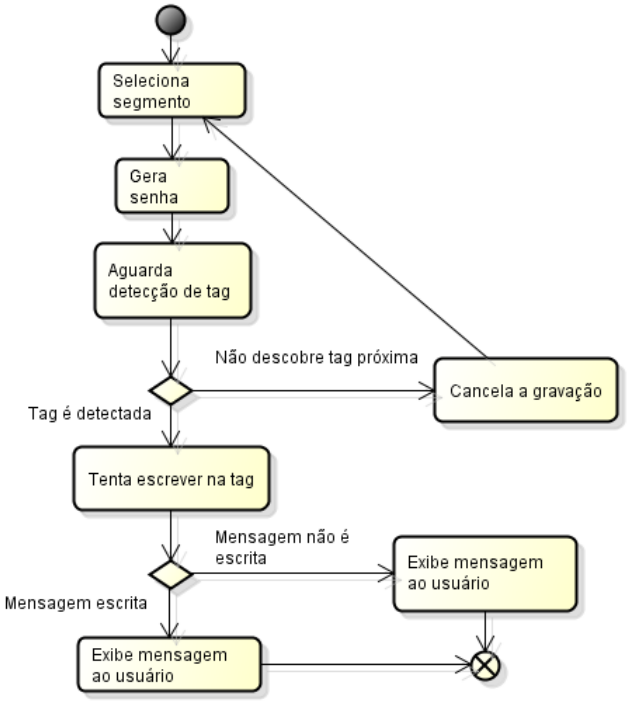


Fonte: autor

3.1.4 Diagrama de Atividades

A Figura 17 destaca as principais etapas no uso da aplicação Gerenciador De Senha. Ao ser iniciada a aplicação, serão exibidas as opções de segmento para serem escolhidas pelo usuário. A partir do momento que o usuário selecionar algum segmento, será possível efetuar o processo de gerar uma senha. Quando a senha for gerada, a aplicação será capaz que fazer a escrita em uma *tag* quando ela for descoberta. Caso nenhuma *tag* seja descoberta ou o usuário cancele o processo, deverá ser gerada uma nova senha para que seja feita a gravação, não sendo possível gravar a mensagem anterior.

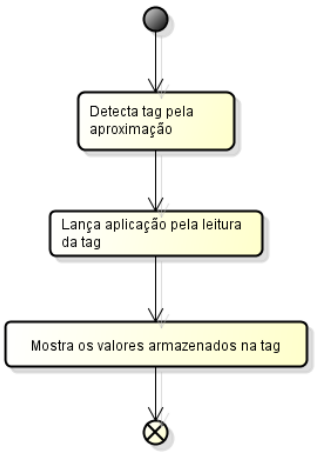
Figura 17 Diagrama de Atividades - Gerenciador De Senha



Fonte: autor

Na Figura 18 está presente o diagrama de atividades da aplicação Visualizador De Senha. Essa aplicação somente é iniciada quando for feita a leitura da uma *tag* com esse propósito. No caso, as *tags* com dados inseridos a partir da aplicação Gerenciador De Senha. Quando o dispositivo detecta essa *tag*, ele abre a aplicação e mostra os valores inseridos, cada um em seu específico campo.

Figura 18 Diagrama de Atividades - Visualizador De Senha



Fonte: autor

3.2 CÓDIGO

Esta seção citará algumas partes dos códigos usados em cada aplicação descrevendo seus objetivos e quais classes fazem uso.

3.2.1 GerenciadorDeSenha

Quando criada a *Activity* são instanciados os atributos mNfcPendingIntent e mWriteTagFilters. O primeiro, quando utilizado, faz com que futuras intenções relacionadas com NFC sejam manuseadas pela própria *Activity*, enquanto o segundo é a declaração do filtro de intenções para a intenção ACTION_TAG_DISCOVERED. Como no processo de escrita não é necessário saber o conteúdo nem mesmo o formato contido em uma *tag* esse filtro é usado para saber quando uma *tag* é descoberta pelo dispositivo. O trecho de código referente a esses atributos está presente no Quadro 1.

Quadro 1 Atributos para filtros de intenções

```
mNfcPendingIntent = PendingIntent.getActivity(this, 0 ,
new Intent(this, getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0 );
IntentFilter tagDetected = new ntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
mWriteTagFilters = new IntentFilter[]{tagDetected};
```

Fonte: autor

Quando o usuário clica no botão para gerar senha, além do próprio processo que gera a senha por meio de um contador, a aplicação e o próprio dispositivo mudam seu estado de funcionamento. Essa aplicação é feita por meio do método enableForegroundDispatch que recebe os já citados atributos como pode ser visto na Quadro 2.

Quadro 2 Método enableTagWriteMode

```
                    private void enableTagWriteMode() {
mNfcAdapter.enableForegroundDispatch(this, mNfcPendingIntent,
        mWriteTagFilters, null);
                                    }
```

Fonte: autor

Ainda no clique do botão, uma caixa de diálogo é aberta indicando ao usuário que uma *tag* deve ser aproximada do dispositivo para que seja feita a escrita, conforme demonstrado no Quadro 3.

Quadro 3 Método para gerar senha

```
public OnClickListener btnGerarSenhaListener = new OnClickListener() {
      enableTagWriteMode();
      new AlertDialog.Builder(MainActivity.this).setTitle("Aproxime de uma tag para
escrever").setOnCancelListener(new DialogInterface.OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog) {
                    enableNdefExchangeMode();
                }
            }).create().show();
}
```

Fonte: autor

O uso desse método permite que a próxima etapa ocorra. O método onNewIntent é da própria classe *Activity* e é chamado quando uma nova intenção for recebida. Após essa intenção ser recebida pela *Activity* é feito o teste para verificar o tipo de intenção. Se for do tipo ACTION_TAG_DISCOVERED a aplicação pega a *tag* que disparou a intenção, conforme visto no Quadro 4.

Quadro 4 Método onNewIntent

```
    protected void onNewIntent(Intent intent) {
        if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
            Tag detectedTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            writeTag(getNoteAsNdef(), detectedTag);
        }
    }
```

Fonte: autor

Quadro 5 Método getNoteAsNdef

```
private NdefMessage getNoteAsNdef() {
        NdefRecord appRecord = createAarRecord("alex.visualizadordesenha");
        NdefRecord dataRecord = createTextRecord(getDataHoraAtual(), Locale.getDefault(), true);

        NdefRecord segmentoRecord = createTextRecord(selected.getText().toString(),
Locale.getDefault(), true);

        String text = txtSenha.getText().toString();
        NdefRecord senhaRecord = createTextRecord(text, Locale.getDefault(), true);
        NdefRecord uriRecord = NdefRecord.createUri("http://www.caixa.gov.br");
        return new NdefMessage(new NdefRecord[] {
                dataRecord,
                segmentoRecord,
                senhaRecord,
                uriRecord,
                appRecord,
        });
}
public NdefRecord createAarRecord(String aplication){
                return NdefRecord.createApplicationRecord(aplication);
}
```

Fonte: autor

No onNewIntent é possível ver a chamada do método writeTag, tendo como parâmetros a *tag* descoberta e o retorno do método getNoteAsNdef. Esse método é o responsável por criar a mensagem que será escrita na *tag*. Essa mensagem é composta por 5 records, como pode ser visto no código presente no Quadro 5.

Para criar um record do tipo RTD Text é usado o método createTextRecord passando por parâmetro o conteúdo do texto, a configuração de localização do dispositivo e a codificação utilizada. Para a criação do record do tipo RTD URI é usado o método ajudante createUri. O método createAarRecord chama um método ajudante que cria um record do tipo AAR. O método createTextRecord está disponível como exemplo de gravação de texto em uma *tag* no Android Developers e realizada o processo necessário para transformação em bytes e organização do record para que fique formato Ndef RTD Text. Esse método está demonstrado no Quadro 6.

Quadro 6 Método createTextRecord

```
public NdefRecord createTextRecord(String payload, Locale locale, boolean encodeInUtf8) {
        byte[] langBytes = locale.getLanguage().getBytes(Charset.forName("US-ASCII"));
        Charset utfEncoding = encodeInUtf8 ? Charset.forName("UTF-8") : Charset.forName("UTF-
16");
        byte[] textBytes = payload.getBytes(utfEncoding);
        int utfBit = encodeInUtf8 ? 0 : (1 << 7);
        char status = (char) (utfBit + langBytes.length);
        byte[] data = new byte[1 + langBytes.length + textBytes.length];
        data[0] = (byte) status;
        System.arraycopy(langBytes, 0, data, 1, langBytes.length);
        System.arraycopy(textBytes, 0, data, 1 + langBytes.length, textBytes.length);
        NdefRecord record = new NdefRecord(NdefRecord.TNF_WELL_KNOWN, NdefRecord.RTD_TEXT, new
byte[0], data);
        return record;
            }
```

Fonte: Android Developers

A mensagem é retornada para o método writeTag. Nesse método é feito o uso da classe Ndef para realizar as operações de consulta e escrita sobre a *tag*, sendo o método writeNdefMessage responsável por isso. Esse é um método da classe Ndef que recebe como parâmetro a mensagem para efetuar a operação de escrita na *tag*. Além disso, são feitos testes para saber se é possível escrever na *tag* devido a ser somente leitura ou seu tamanho insuficiente, como pode ser visto no Quadro 7.

Quadro 7 Método writeTag

```
boolean writeTag(NdefMessage message, Tag tag) {
        int size = message.toByteArray().length;
        try {
            Ndef ndef = Ndef.get(tag);
            if (ndef != null) {
                ndef.connect();

... métodos que verificam o tamanho da tag e se é somente leitura...

                ndef.writeNdefMessage(message);
                toast("TAG escrita com sucesso.");
                return true;
            } else {
                NdefFormatable format = NdefFormatable.get(tag);
... métodos que tentam formatar a escrever a mensagem...
            }
        } catch (Exception e) {toast("Falha ao escrever na tag");}
        return false;
    }
```

Fonte: Android Developers

3.2.2 Visualizador de Senha

Como a mensagem gravada na *tag* continha um record do tipo AAR se, no momento em que a *tag* for lida não tiver nenhuma aplicação que controle a detecção em primeiro plano, o dispositivo irá lançar a aplicação registrada no *record*. Dessa forma é lançada a aplicação VisualizadorDeSenha, que não possui ícone lançador e só pode ser inicializada por meio da leitura desse tipo de mensagem. No arquivo AndroidManifest.xml é adicionado o filtro de intenções do tipo NDEF_DISCOVERED para trabalhar junto com o AAR, conforme visto no Quadro 8.

Quadro 8 Trecho do AndroidManifest.xml

```
<intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
</intent-filter>
```

Fonte: Android Developers

Como a intenção de leitura de *tag* que fará a execução da *Activity*, não é utilizado o método onResume, sendo o tratamento da intenção sendo feito na chamada do método onCreate. Esse tratamento é mostrado no Quadro 9.

Quadro 9 Tratamento da *Intent*

```
if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(intent.getAction())) {
            NdefMessage[] msgs = getNdefMessages(intent);
            lerTag(msgs[0]);
}
```

Fonte: Autor

O método getNdefMessages, visto no Quadro 10, retorna as mensagens que foram lidas na *tag*. Para isso é usado um objeto do tipo Parcelable e passado seu valor para um objeto do tipo NdefMessage. O conteúdo do objeto Parcelable é proveniente da *Intent*, buscado por meio do método getParcelableArrayExtra.

Quadro 10 Método getNdefMessages

```
public NdefMessage[] getNdefMessages(Intent intent) {
        NdefMessage[] msgs = null;
        String action = intent.getAction();
        if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
            Parcelable[] rawMsgs =
intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
            if (rawMsgs != null) {
                Debug("raws != null");
                msgs = new NdefMessage[rawMsgs.length];
                for (int i = 0; i < rawMsgs.length; i++) {
                    msgs[i] = (NdefMessage) rawMsgs[i];
                }
            }
... conteúdo do tipo desconhecido, tag considerada vazia...
        return msgs;
    }
```

Fonte: Android Developers

Esse valor é retornado para o método lerTag, descrito no Quadro 11, que recebe como parâmetro a primeira mensagem. Na aplicação desenvolvida a leitura será feita de somente uma mensagem. O método ler mensagem chama o método readText, passando como parâmetro o *record* a ser lido e atualiza o valor do campo da aplicação com o valor recebido.

Quadro 11 Método lerTag

```
private void lerTag(NdefMessage msg){
        String senha = "";
        String site = "";
        String segmento = "";
        String data = "";
        try {
                data  = readText(msg.getRecords()[0]);
                segmento = readText(msg.getRecords()[1]);
                senha = readText(msg.getRecords()[2]);
                site = readSite(msg.getRecords()[3]);

        } catch (UnsupportedEncodingException e) {
                e.printStackTrace();
        }
        setDataBody(data);
        setSenhaBody(senha);
        setSegmentoBody(segmento);
        setSiteBody(site);
}
```

Fonte: Autor

O método readText serve para ler o conteúdo em formado texto de um *payload*, separando os primeiros caracteres que servem para identificação do idioma e retornando o valor no formato String. O método readSite serve para ler o valor quando está gravado no formato URI. Ambos os métodos estão presentes no Quadro 12.

Quadro 12 Métodos readText e readSite

```
private String readText(NdefRecord record) throws UnsupportedEncodingException {
    byte[] payload = record.getPayload();
    String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8" : "UTF-16";
    int languageCodeLength = payload[0] & 0063;
    return new String(payload, languageCodeLength + 1, payload.length - languageCodeLength - 1,
textEncoding);
}

private String readSite(NdefRecord record) throws UnsupportedEncodingException {
    byte[] payload = record.getPayload();
    return record.toUri().toString();
}
```

Fonte: Autor, adaptado de Android Developers

Outros objetos e métodos foram utilizados no desenvolvimento destes aplicativos, mas não são foram relatados para não estender demasiadamente este relatório. Uma visão completa do aplicativo responsável por gerar a senha pode ser vista nos apêndices 1 e 2.

3.3 TRABALHOS RELACIONADOS

Um aspecto importante a ser considerado é a capacidade do NFC trazer o conceito de computação ubíqua mais próxima de nossa realidade devido a sua capacidade de realizar

interação entre dispositivos de forma simples, fácil e rápida. O trabalho de Danilo Souza Lima (2013) sobre a utilização do NFC aliado a tecnologia Android e Arduino para elaborar um protótipo de sistema de controle de acesso mostra bem esse lado da tecnologia. Em seu desenvolvimento, ele foca o parte bem tecnológica do NFC, além de usar técnicas de criptografia e autenticação na transmissão de dados entre Arduino e Android e, por meio disso, constrói uma aplicação que pode ser utilizada para automação residencial.

Guilherme Borges (2013) também realizou um trabalho tendo como base a computação ubíqua, mas com foco na utilização de microcontroladores, em especial, o Arduino.

4 CONSIDERAÇÕES FINAIS

A comunicação sem utilização de meios físicos não é de agora, mas parece que suas possibilidades ainda estão emergindo e o NFC pode se tornar o agente facilitador desse processo. As grandes empresas aliadas no desenvolvimento da tecnologia Near Field Communication trabalham para que ela se torne o meio padrão para transmissão de dados a curta distância fazendo com que o pagamento por meio de um celular, uma de suas grandes possibilidades, seja feito por meio dessa. Isso ocorre por que, simultaneamente, outras ferramentas estão sendo desenvolvidas com o objetivo de pegar uma fatia desse abrangente mercado. Apesar de não ser considerada uma tecnologia de alto custo, sua presença é encontrada quase que unicamente em dispositivos de preço mais elevado, o que dificulta sua popularização.

A respeito do desenvolvimento desse trabalho, poucas dificuldades foram encontradas. A principal delas foi a ausência de um material de referência na fase inicial do projeto, pois a maioria dos que foram encontrados, além de não estarem em língua materna, não apresentavam um padrão para demonstrar as informações, tendo, inclusive, diferenças em dados coletados de uma referência para outra. Isso fazia com que as referências se complementassem, mas sem demonstrar uma ligação direta entre as mesmas. Outro aspecto observado foi a variação da qualidade e quantidade de referências disponíveis ao longo do projeto, tornando necessárias alterações e adaptações. Devido a rápida evolução tecnológica, afirmações previamente vistas já não seriam mais válidas quando do desenvolvimento de uma nova versão do Android, que incrementaria suas capacidades de utilização da NFC como foi o caso do uso de um dispositivo Android com suporte a NFC no modo emulação de cartão, algo que só se tornou possível com o lançamento da mais recente versão do Android, o Kitkat.

O desenvolvimento do estudo de caso ocorreu sem maiores problemas graças a quantidade de material disponível no Android Developers. Embora sem muitos exemplos e com códigos dispersos, o conteúdo disponível no site dos desenvolvedores da plataforma Android foi de grande ajuda. O fator que dificulta o desenvolvimento de qualquer aplicação que utilize NFC é a impossibilidade de uso em uma máquina virtual pois a mesma não consegue simular o hardware necessário para sua utilização.

Após entender a tecnologia de forma geral, não foram sentidas dificuldades na hora de aplicar a mesma sobre a plataforma Android, embora seu uso tenha sido bem específico nas duas aplicações. Esse uso permite o trabalho com o formato NDEF e com os quatro tipos de

tags que oferecem suporte total a esse formato. Mesmo a leitura e a gravação ocorrendo em nível de bits e bytes o NDEF provê uma abstração da camada física facilitando o trabalho e possibilitando uma programação em nível mais elevado. O mesmo não acontece quando o desejo é utilizar o NFC em uma *tag* que não ofereça suporte a NDEF, ou quando o mesmo não será utilizado por alguma razão específica, fato que implicará em leitura e gravação em baixo nível, obrigando o desenvolvedor a conhecer os detalhes da *tag* que será utilizada.

Enfim, por ser uma aplicação bem específica de uso do NFC, a mesma não ficou extensa, sendo necessários poucos ajustes para que funcionasse da maneira desejada: uma aplicação fazendo a gravação dos dados na *tag* enquanto a outra aplicação é lançada somente após a leitura dessa *tag*, não possuindo um ícone para isso no menu de aplicação do dispositivo Android. Concluindo, portanto, que a aplicação funciona e pode servir de base e apoio para futuros trabalhos onde pode ser necessária ou desejada a utilização do NFC. E, a respeito da tecnologia, foi possível verificar seu fácil funcionamento e extensa aplicabilidade, faltando somente o porte de dispositivos capazes que ofereçam seu suporte por parte da população.

4.1 Trabalhos Futuros

Com algumas leves alterações e adequações, o código utilizado pode servir como base para tornar outras aplicações hábeis a ler e escrever em *tags* NFC ou até mesmo a enviar os chamados *beam*, dados enviados de um dispositivo Android para outro por meio do Android Beam, já que há muita semelhança entre os dois modos de utilização no Android. Claro que para isso, além de um conhecimento sobre NFC, também é necessário entender o funcionamento da plataforma Android, que possui alguns detalhes na hora de trabalhar com NFC, conforme demonstrado durante o trabalho. O Android Beam é uma forma de utilização do NFC específica para a plataforma Android, mas é possível realizar a comunicação com outras plataformas como Arduino ou até dispositivos com outros sistemas operacionais como o Windows 8, que também oferece suporte a NFC. O modo emulação de cartão, disponível na última atualização do Android, serve para o chamado "Tocar e Pagar", onde aplicações poderão trabalhar de forma que um outro dispositivo faça a leitura dos dados contidos no cartão emulado por elas. A maior parte das aplicações já existentes que trabalham com leitura e gravação de *tags* serve para inserir na *tag* alguma atividade pré-definida, para que esta seja executada sempre que o dispositivo fizer a leitura dessa *tag*.

REFERÊNCIAS

ISO/IEC 18092 - Telecommunications and information exchange between systems - Near Field Communication- Interface and Protocol. 2 ed. 2013 Disponível em: <http://standards.iso.org/ittf/PubliclyAvailableStandards/c056692_ISO_IEC_18092_2013.zi. Acesso em: 15 mai. 2013.

Android. **Android**. Disponível em: <http://www.android.com/>. Acesso em: 22 jun. 2013.

Android Developers. **NFC Basics**. 2013. Disponível em: <http://developer.android.com/guide/topics/connectivity/nfc/nfc.html >. Acesso em: 22 jun. 2013.

_____. **Uses SDK.** 2014. Disponível em: <http://developer.android.com/guide/topics/manifest/uses-sdk-element.html>. Acesso em: 19 abr. 2014 (a).

_____. **ADT Plugin**. 2014. Disponível em: <http://developer.android.com/tools/sdk/eclipse-adt.html>. Acesso em: 19 abr. 2014 (b).

_____. **Android 2.3.3** APIs. 2014. Disponível em: <https://developer.android.com/about/versions/android-2.3.3.html>. Acesso em: 21 abr. 2014 (c)

_____. **Android 4.0 APIs**. 2014. Disponível em: <https://developer.android.com/about/versions/android-4.0.html>. Acesso em: 21 abr. 2014 (d)

_____. **Android 4.1 APIs**. 2014. Disponível em: <https://developer.android.com/about/versions/android-4.1.html>. Acesso em: 21 abr. 2014. (e)

_____. **Android 4.4 APIs**. 2014. Disponível em: <https://developer.android.com/about/versions/kitkat.html>. Acesso em: 21 abr. 2014 (f)

_____. **NdefMessage**. 2014. Disponível em: <http://developer.android.com/reference/android/nfc/NdefMessage.html>. Acesso em: 21 abr. 2014 (g).

_____. **NdefRecord**. 2014. Disponível em: <http://developer.android.com/reference/android/nfc/NdefRecord.html>. Acesso em: 25 abr. 2014 (h);

_____. **Tag**. 2014. Disponível em: <http://developer.android.com/reference/android/nfc/Tag.html>. Acesso em: 25 abr. 2014 (i);

_____. **Ndef**. 2014. Disponível em: <http://developer.android.com/reference/android/nfc/tech/Ndef.html>. Acesso em: 25 abr. 2014 (j);

ARAÚJO, Regina Bordes de. **Computação Ubíqua**: Princípios, Tecnologias e Desafios. XXI Simpósio Brasileiro de redes de computadores. Departamento de Computação - Universidade Federal de São Carlos (UFSCar). 2004. Disponível em: <http://www.professordiovani.com.br/rw/monografia_araujo.pdf>. Acesso em: 7 abr 2013

CURRAN, Kevin; MILLAR, Amanda, GARVEY, Conor MC. **Near Field Communication**. 2012. Disponível em: <www.iaesjournal.com/online/index.php/IJECE/article/view/234/pdf>. Acesso em: 8 abr. 2013.

DeviceFidelity, Inc. **The MicroNFC Module is a Removable NFC Controller and Secure Element in s Simple Package**. 2013.Disponível em: <http://www.devifi.com/assets/DeviceFidelity_microNFC.pdf>. Acesso em: 05 mai. 2013.

EBV Elektronik. **RFID Guide**. 2010. Disponível em: <http://www.pdfio.com/k-2226017.html>. Acesso em: 14 jun. 2013.

FERRAZ FILHO, Onildo Luciano de Souza. **Comunicação NFC (Near Field Communication) entre Dispositivos Ativos**. 2010. Disponível em <http://www.cin.ufpe.br/~tg/2010-2/olsff.pdf>. Acesso em: 02 jun 2013.

Google Inc. **Buy in Stores**. 2013. Disponível em: <http://www.google.com/wallet/buy-in-store/>. Acesso em: 05 mai 2013.

BORGES, Guilherme Antonio. **Arquitetura Ubiqua para ambientes residenciais**. 2012. Disponível em: <http://inf.passofundo.ifsul.edu.br/graduacao/monografias-defendidas/2012-1/GuilhermeBorges.pdf>. Acesso em: 03 abr. 2014.

IGOE, Tom. COLEMAN, Don. JEPSON, Brian. Beginning NFC - Near Field Communication with Arduino, Android & PhoneGap. Disponível em: <http://filepi.com/i/lEg9uAr>. 1 ed. O'Reilly Media. 2014. Acesso em: 03 mai. 2014.

International Organization Stardart (ISO)/ International Electrotechnical Commission (IEC). **ISO/IEC 14443 - Identification cards - Contactless integrated circuit(s) cards - Proximity cards**. Acesso em 15 mai 2013

LECHETA, Ricardo R. **Google Android - Aprenda a criar aplicações para dispositivos móveis com Android SDK.** 2 ed. São Paulo: Novatec Editora Ltda, 2009.

LIMA, Danilo Souza. **Proposta para controle de acesso físico seguro baseada em Near Field Communication (NFC) e Android**. 2013. Disponível em: < http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-16012014-095759/publico/Lima_Danilo_Souza.pdf>. Acesso em: 11 mar. 2014.

LEVANDOSKI, Fausto; DIAS, Vagner Fleck; CHRIST, Vagner Rafael, MARQUES, Vitor Hugo. **O método de comunicação NFC e sua aplicação no processo de pagamento através de dispositivos móveis**. 2011. Disponível em: <http://www.faustolevandoski.com.br/downloads/Near%20Field%20Communication.pdf>. Acesso em: 11 abr. 2013

MATEUS, Geraldo Robson; LOUREIRO, Antonio Alfredo Ferreira. **Introdução a Computação Móvel**. [S. l: S. n], 2005. Disponível em: <http://homepages.dcc.ufmg.br/~loureiro/cm/docs/cm_livro_1e.pdf>. Acesso em: 13 abr. 2013.

NFC FORUM. **NFC and Contactless Technologies**. Disponível em: <http://www.nfc-forum.org/aboutnfc/nfc_and_contactless/>. Acesso em: 05 mai 2013

_____. **NFC and Interoperability**. Disponível em: <http://www.nfc-forum.org/aboutnfc/interop/>. Acesso em: 05 mai 2013 (a).

_____. **Protocol Technical Specification**. Disponível em: < http://www.nfc-forum.org/aboutnfc/nfc_in_action/>. Acesso em: 05 mai 2013 (b).

_____. **NFC in Action**. Disponível em: <http://www.nfc-forum.org/specs/spec_list/>. Acesso em: 05 mai 2013 (c).

_____. **Text Record Type Definition**. 2006. Disponível em: <http://www.nfc-forum.org/specs/spec_list/>. Acesso em: 05 mai 2013 (d).

_____. **Secure Element Development**. 2009. Disponível em : <http://www.nfc-forum.org/events/oulu_spotlight/2009_09_01_Secure_Element_Programming.pdf>. Acesso em: 22 jun 2013 (e).

_____. **About the Technology**. 2013. Disponível em : <http://nfc-forum.org/what-is-nfc/about-the-technology/>. Acesso em: 15 mai. 2014 (f).

Nokia Developer. **Introduction to NFC**. 2011. Disponível em <http://www.developer.nokia.com/dp?uri=http%3A%2F%2Fsw.nokia.com%2Fid%2Fbdaa4a0f-fcf3-4a4b-b800-c664387d6894%2FIntroduction_to_NFC>. Acesso em: 16 jun. 2013

NXP Semicondutres. **PN532/C1 - Near Field Communication (NFC) controller**. Disponível em: <http://www.nxp.com/documents/short_data_sheet/PN532_C1_SDS.pdf>. 2012. Acesso em: 15 mai. 2013

Open Hardware. **What is an Open Hardware?**. Disponível em: <http://www.openhardware.de/>. Acesso em: 20 jun. 2013.
PagSeguro. **PagSeguro NFC**. 2013. Disponível em: <http://pagseguronfc.uol.com.br/faq.html>. Acesso em: 06 mai. 2013

SUPARTA, Wayan. **Application of Near Field Communication Technology for Mobile Airline Ticketing**. 2012. Disponível em: <http://www.thescipub.com/pdf/10.3844/jcssp.2012.1235.1243> Acesso em: 11 abr. 2013.

TANEMBAUM, Andres S; STEEN, Marten Van. **Sistemas Distribuídos**: Princípios e Paradigmas. 2 ed. São Paulo: Person Prentice Hall, 2007.

TANEMBAUM, Andres S. **Redes de computadores**. 2003. Amsterdan, 4 ed. Editora Campus.

WANT, Roy. **Pervasive Computing**: An Introduction to RFID Technology. 2006. Disponível em: <http://www.faculty.jacobs-university.de/jwallace/xwallace/courses/ap/rfid1.pdf>. Acesso em: 13 abr 2013.

*ZERO HORA***. ZH Classificados - Telefonia e Consumo**: Governo Regulamenta pagamento por celular. p. 2. 26 mai. 2013.

ANEXOS E APÊNDICES

Apêndice 1

```
public class MainActivity extends Activity {
         RadioGroup rdgSegmento; Button btnGerarSenha; int idSelected; RadioButton selected;
         int contCaixa; int contSocial;int contEmpresa;int contVoce;String senha;TextView txtSenha;
         DecimalFormat df;NfcAdapter mNfcAdapter;PendingIntent mNfcPendingIntent;
         IntentFilter[] mWriteTagFilters;IntentFilter[] mNdefExchangeFilters;
         @Override
         protected void onCreate(Bundle savedInstanceState) {
                   super.onCreate(savedInstanceState);
                   setContentView(R.layout.fragment_main);
                   df = new DecimalFormat("000");
                   mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
                   txtSenha = (TextView) findViewById(R.id.txtSenha);
                   btnGerarSenha = (Button) findViewById(R.id.btnGerarSenha);
                   btnGerarSenha.setOnClickListener(btnGerarSenhaListener);
                   rdgSegmento = (RadioGroup) findViewById(R.id.rdgSegmento);
                   rdgSegmento.setOnCheckedChangeListener(rdgSegmentoOnCheckChangeListener);
                   contCaixa = 0; contEmpresa = 0; contSocial = 0; contVoce = 0;
                   mNfcPendingIntent = PendingIntent.getActivity(this, 0 , new Intent(this,
getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0 );
                   IntentFilter ndefDetected = new IntentFilter(NfcAdapter.ACTION_NDEF_DISCOVERED);
                   try {
                            ndefDetected.addDataType("text/plain");
                   } catch (MalformedMimeTypeException e) {}
                   mNdefExchangeFilters = new IntentFilter[]{ndefDetected};
                   IntentFilter tagDetected = new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
                   mWriteTagFilters = new IntentFilter[]{tagDetected};
         }
         public boolean onCreateOptionsMenu(Menu menu) {
                   getMenuInflater().inflate(R.menu.main, menu);return true;
         }
         public boolean onOptionsItemSelected(MenuItem item) {
                   int id = item.getItemId();
                   if (id == R.id.action_settings) {return true;}
                   return super.onOptionsItemSelected(item);
         }
    protected void onNewIntent(Intent intent) {
        if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {
            Tag detectedTag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            writeTag(getNoteAsNdef(), detectedTag);
        }
    }
boolean writeTag(NdefMessage message, Tag tag) {
        int size = message.toByteArray().length;
        try {
            Ndef ndef = Ndef.get(tag);
            if (ndef != null) { ndef.connect();
                if (!ndef.isWritable()) {toast("TAG é somente escrita.");return false;}
                if (ndef.getMaxSize() < size) {
                toast("A capacidade da TAG é " + ndef.getMaxSize() + " bytes, a mensagem tem " + size + "
bytes."); return false;
                }
                ndef.writeNdefMessage(message);
                toast("TAG escrita com sucesso.");
                return true;
            } else {
                NdefFormatable format = NdefFormatable.get(tag);
                if (format != null) {
                    try {
                        format.connect();format.format(message);
                        toast("TAG formatada e escrita"); return true;
                    } catch (IOException e) {
                        toast("Falha ao escrever na tag.");return false;
                    }
                } else {
                    toast("TAG não oferece suporte a NDEF.");return false;
                }
            }
        } catch (Exception e) {toast("Falha ao escrever na tag");}
        return false;
    }
```

```
private NdefMessage getNoteAsNdef() {
         NdefRecord appRecord = createAarRecord("alex.visualizadordesenha");
         NdefRecord dataRecord = createTextRecord(getDataHoraAtual(), Locale.getDefault(), true);
         NdefRecord segmentoRecord = createTextRecord(selected.getText().toString(), Locale.getDefault(), true);
         String text = txtSenha.getText().toString();
        NdefRecord senhaRecord = createTextRecord(text, Locale.getDefault(), true);
        NdefRecord uriRecord = NdefRecord.createUri("http://www.caixa.gov.br");
        return new NdefMessage(new NdefRecord[] {
                  dataRecord, segmentoRecord, senhaRecord, uriRecord, appRecord,
        });
    }
public OnClickListener btnGerarSenhaListener = new OnClickListener() {
         public void onClick(View v) {
                  switch (idSelected){
                           case R.id.rdbCaixa: contCaixa++; senha = "CX"+df.format(contCaixa); break;
                           case R.id.rdbEmpresa: contEmpresa++; senha = "PJ"+df.format(contEmpresa); break;
                           case R.id.rdbVoce: contVoce++; senha = "PF"+df.format(contVoce); break;
                           case R.id.rdbSocial: contSocial++; senha = "SO"+df.format(contSocial);break;
                  }
                  txtSenha.setText(senha); selected = (RadioButton) findViewById(idSelected);
                  if(mNfcAdapter==null){ toast("Sem suporte para NFC");return;}
                  disableNdefExchangeMode(); enableTagWriteMode();
           new AlertDialog.Builder(MainActivity.this).setTitle("Aproxime de uma tag para escrever")
           .setOnCancelListener(new DialogInterface.OnCancelListener() {
               @Override
               public void onCancel(DialogInterface dialog) {
                   disableTagWriteMode(); enableNdefExchangeMode();
               }
           }).create().show();}
};
private OnCheckedChangeListener rdgSegmentoOnCheckChangeListener = new OnCheckedChangeListener() {
         public void onCheckedChanged(RadioGroup group, int checkedId) {
                  btnGerarSenha.setVisibility(View.VISIBLE);idSelected = rdgSegmento.getCheckedRadioButtonId();
                  selected = (RadioButton) findViewById(idSelected);}
};
private void toast(String text) { Toast.makeText(this, text, Toast.LENGTH_SHORT).show();}
public NdefRecord createTextRecord(String payload, Locale locale, boolean encodeInUtf8) {
                byte[] langBytes = locale.getLanguage().getBytes(Charset.forName("US-ASCII"));
                Charset utfEncoding = encodeInUtf8 ? Charset.forName("UTF-8") : Charset.forName("UTF-16");
                byte[] textBytes = payload.getBytes(utfEncoding);
                int utfBit = encodeInUtf8 ? 0 : (1 << 7);
                char status = (char) (utfBit + langBytes.length);
                byte[] data = new byte[1 + langBytes.length + textBytes.length];
                data[0] = (byte) status;
                System.arraycopy(langBytes, 0, data, 1, langBytes.length);
                System.arraycopy(textBytes, 0, data, 1 + langBytes.length, textBytes.length);
                NdefRecord record = new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
                NdefRecord.RTD_TEXT, new byte[0], data);
                return record;
}
private NdefRecord createAarRecord(String aplication){return NdefRecord.createApplicationRecord(aplication);}
private void enableTagWriteMode() {
         IntentFilter tagDetected = new IntentFilter(NfcAdapter.ACTION_TAG_DISCOVERED);
         mWriteTagFilters = new IntentFilter[] { tagDetected};
         mNfcAdapter.enableForegroundDispatch(this, mNfcPendingIntent, mWriteTagFilters, null);
}
private void disableNdefExchangeMode() {mNfcAdapter.disableForegroundDispatch(this);}
private void disableTagWriteMode() {mNfcAdapter.disableForegroundDispatch(this);}
private void enableNdefExchangeMode() {
       mNfcAdapter.enableForegroundDispatch(this, mNfcPendingIntent, mNdefExchangeFilters, null);
}
private String getDataHoraAtual(){
         SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy-HH:mm:ss", Locale.getDefault());
         Date data = new Date();
         Calendar  cal = Calendar.getInstance();
         cal.setTime(data);
         Date data_atual = cal.getTime();
         String data_completa = dateFormat.format(data_atual);
         return data_completa;
         }
}
```