Instituto Superior de Engenharia do Porto

Accommodation Management System using Mobile Devices

Diogo Emanuel Nóbrega Garcia

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Arquiteturas, Sistemas e Redes

Orientador: Prof. Doutor Lino Manuel Baptista Figueiredo

Co-orientador: Eng.º Ricardo Manuel Soares Anacleto

Júri:

Presidente:

Doutor Luís Miguel Moreira Lino Ferreira

Vogais:

Doutor Paulo Manuel Baltarejo de Sousa

Doutor Lino Manuel Baptista Figueiredo

Eng.º Ricardo Manuel Soares Anacleto

Porto, Outubro 2014

Dedico este trabalho à minha família, amigos e todos os que estiveram presentes na minha vida académica.

Resumo

Os dispositivos móveis são pessoais, intransmissíveis e cada vez mais utilizados, tornando-se assim numa boa ferramenta para a realização de um conjunto de serviços na indústria hoteleira. Entre esses serviços que necessitam da identificação pessoal, encontram-se a possibilidade do cliente reservar um quarto ou utilizar o serviço de quartos.

Atualmente é muito utilizado, nos locais de alojamento, um *smart card* que possibilite ao cliente ter acesso a alguns dos serviços disponíveis. O objetivo deste documento é apresentar uma alternativa ao sistema de cartões, utilizando para o efeito, dispositivos móveis.

De modo a garantir a segurança e uma utilização semelhante ao sistema de cartões existentes foi utilizada a tecnologia NFC (*Near Field Communication*) que, ao permitir o modo de emulação de cartão, facilita a transação do sistema de *smart card* existente, para o da utilização de dispositivos móveis na realização das mesmas funções. Mais concretamente, será abordada a utilização de *smartphones* para o processo de abertura de portas. Para que exista uma melhor compreensão e para que haja um conhecimento das suas capacidades e limites foram estudados casos de uso da tecnologia NFC.

Este documento apresenta ainda os processos de desenvolvimento de uma aplicação nativa para o sistema operativo Android, cujo objetivo é proporcionar ao cliente de um local de alojamento um novo modo de acesso ao quarto, utilizando a tecnologia NFC. Para além desta funcionalidade a aplicação permite ainda ao utilizador fazer reservas, fazer o *check-in*, fazer o *check-out* entre outras. Posteriormente serão apresentadas as conclusões e possíveis trabalhos futuros.

Palavras-chave: *Near Field Communication*, Android, *Smart Card*, Emulação de Cartão, Hotelaria

Abstract

Mobile devices are personal and intransmissible objects. Those characteristics make them able to perform tasks that need the confirmation of the person's identity, for example in the hotel industry when a client is booking a room or requesting room services.

Nowadays, in the hotel industry, the use of smart cards to open the room door or even for the client to be able to use some of the hotel services is a common practice. This document intends to present an alternative to the smart card system, using smartphones instead.
In order to develop a secure system and with a similar use of the smart card system the NFC (Near Field Communication) technology was used. This technology has a card emulation feature, which allows an easy transaction from the previous system to the new one.

With this document we intend to present a new approach based on mobile devices and NFC technology to access an hotel room replacing the traditional smart card. In order to understand the capabilities and limits of the NFC technology use cases of applications that already use this technology were studied.

This document also describes the process of development of a native Android application whose main objective is to make this new method of door access using NFC technology available. Besides this feature the application also allows the user to make reservations, do the check-in, the check-out and also other features.

In the latest section of the document the conclusions and future works are presented.

Keywords: Near Field Communication, Android, Smart Card, Card Emulation, Hotel industry

Agradecimentos

Agradeço em primeiro lugar à minha família por estar sempre presente e me ter incentivado desde sempre a lutar pelos meus objetivos. Agradeço também a todos que me apoiaram durante o desenvolvimento deste projeto e a todos que estiveram presentes e contribuíram para o meu ensino. Em especial gostaria de agradecer ao meu amigo André Almeida por ter disponibilizado o seu *smartphone* para que pudesse desenvolver e testar este projeto.

Um agradecimento especial ao Professor Lino Figueiredo e ao Eng.º Ricardo Anacleto por estarem sempre disponíveis e por me orientarem durante o desenvolvimento deste projeto.

A todos, o meu muito obrigado!

Índice

Lista de Figuras

Lista de Tabelas

Lista de Código

Acrónimos e Símbolos

Lista de Acrónimos

AID	*Application ID*
APDU	*Application Protocol Data Unit*
API	*Application Programming Interface*
BSSID	*Basic Service Set Identifier*
CPU	*Central Processing Unit*
ECMA	*European Computer Manufacturers Association*
ECDH	*Elliptic curve Diffie–Hellman*
ES	Elemento Seguro
IEC	*International Electrotechnical Commission*
ISO	*International Organization for Standardization*
Kbps	Quilobit por Segundo
NDEF	*NFC Data Exchange Format*
NFC	*Near Field Communication*
P2P	Ponto-a-Ponto
PIX	*Proprietary application Identifier eXtension*
RID	*Registered application provider Identifier*
RIM	*Research In Motion*
RFID	*Radio Frequency Identification*
SCH	*Secure Channel Service*
SDK	*Software Development Kit*
SIM	*Subscriber Identity Module*
SO	Sistema Operativo
SSE	*Shared Secret Service*

Soft-SE *Software Secure Element*

XML *eXtensible Markup Language*

1 Introdução

Neste capítulo é apresentada uma introdução ao documento. Começa por ser feito um enquadramento à utilização de *smartphones* equipados com a tecnologia NFC (*Near Field Communication*), seguidos pela descrição dos objetivos propostos e pela exposição do problema. Por fim são ainda descritas as motivações para ao desenvolvimento deste projeto.

1.1 Enquadramento

Com a evolução da tecnologia torna-se cada vez mais evidente que os dispositivos móveis fazem parte do quotidiano do ser Humano. Entre estes dispositivos encontram-se os *smartphones* que têm vindo gradualmente a substituir os telemóveis mais tradicionais. No quarto trimestre de 2013 a venda de *smartphones* alcançou 57,6 % das vendas de dispositivos móveis, ultrapassando pela primeira vez desde a sua criação as vendas dos telemóveis (Gartner, 2014a). Com este crescimento surgem novas oportunidades para disponibilizar serviços, que até então eram impossíveis de existir em telemóveis.

Os telemóveis, e mais recentemente os *smartphones,* têm caraterísticas de serem dispositivos pessoais e intransmissíveis, tornando-se assim possíveis ferramentas para poder identificar uma pessoa. Esta caraterística permite que um *smartphone* possa ser utilizado para funções que necessitem da identificação pessoal, tais como, utilizar o dispositivo para fazer pagamentos de compras ou serviços, ou ainda garantir o acesso a um edifício (Ok et al, 2010).

Com o objetivo de tornar a vida de um utilizador mais conveniente foi desenvolvida a tecnologia NFC (*Near Field Communication*). Esta tecnologia, incorporada num dispositivo móvel, permite que dois dispositivos possam interagir, a uma curta distância, de modo a trocarem informações ou emular um *smart card*. Este sistema de emulação torna assim possível que, um dispositivo móvel possa emular um cartão de pagamento para posteriormente ser utilizado a partir do *smartphone*. Tem ainda a possibilidade de ser utilizado noutras funções, como por exemplo, emular *smart cards* cujo objetivo é ter acesso a um determinado edifício.

A emulação de cartão permite que num mesmo dispositivo sejam armazenados mais do que um cartão, tornando-se para o utilizador uma tecnologia bastante conveniente. Através desta tecnologia, o utilizador não necessita de se preocupar com o facto de se ter esquecido de um determinado cartão, ou de ter de despender de tempo à procura do cartão correto, desde que não se esqueça do seu dispositivo móvel com a tecnologia NFC este faz a escolha do cartão correto a utilizar.

1.2 Objetivos

O principal objetivo deste projeto consiste, em desenvolver um sistema que permita efetuar a gestão de diversos serviços disponíveis num local de alojamento. Para que o utilizador possa interagir com este sistema, será desenvolvida uma aplicação móvel que, entre outras tecnologias, utilizará a tecnologia NFC.

Mais especificamente, o projeto que aqui se apresenta, tem o seguinte conjunto de objetivos:

- Criar uma aplicação para o sistema operativo Android que incorpora alguns dos serviços disponibilizados pelo local de alojamento;
- Interação da aplicação com os dispositivos existentes no local de alojamento tais como acesso ao quarto ou controlo de luzes;
- Interação com outros serviços disponíveis pela unidade de alojamento, tais como, fazer a reserva de um quarto;
- Desenvolver uma aplicação para testar os serviços que utilizam NFC.

1.3 Motivação

A motivação para desenvolver este projeto passa pela possibilidade de trabalhar com uma tecnologia ainda pouco conhecida, mas com bastante potencialidade. A utilização mais conhecida do NFC, no quotidiano do ser Humano, é o modo de pagamento de compras ou serviços (Ok et al, 2010). Embora este seja o modo mais conhecido, não é ainda utilizado pela maioria das pessoas, pois existe alguma incerteza como o processo de pagamento se desenrola e se é totalmente seguro.

Para além deste modo de funcionamento onde são emulados os cartões para proceder posteriormente ao pagamento de serviços, o NFC permite ainda que sejam emulados qualquer *smart card* existente atualmente no mercado. Deste modo um dispositivo móvel com esta tecnologia pode emular um *smart card* de acesso a um edifício.

Este e os outros modos de operação do NFC, modo de leitura/escrita e P2P (Ponto a Ponto), permitem ao programador liberdade para criar um conjunto de aplicações que ainda não se encontram disponíveis no mercado mas que eventualmente, no futuro, irão fazer parte do quotidiano do ser Humano.

A motivação para desenvolver este projeto passa ainda pela procura de novos modos de utilização da tecnologia NFC bem como tentar desmistificar a sua utilização, aumentando assim o seu uso.

1.4 Exposição do Projeto

Como referido na secção dos objetivos, o principal objetivo deste projeto consiste no desenvolvimento de um sistema, que permita efetuar a gestão de diversos serviços disponíveis num local de alojamento. A Figura 1 apresenta a arquitetura do sistema que existirá num local de alojamento e consiste num diagrama onde é possível observar a interação entre os diferentes componentes desse sistema. É ainda possível observar que a aplicação móvel que irá ser desenvolvida comunicará com os sistemas externos disponibilizados pelo local de alojamento e com terminais existentes no quarto. Estes serviços externos, bem como os serviços internos compõem um *backoffice* que permitirá manter o sistema do local de alojamento funcional. A receção permite disponibilizar uma alternativa à utilização da aplicação móvel. Desse sistema somente a aplicação Android e alguns dos serviços externos que sejam utilizados pela mesma aplicação, serão descritos neste documento.

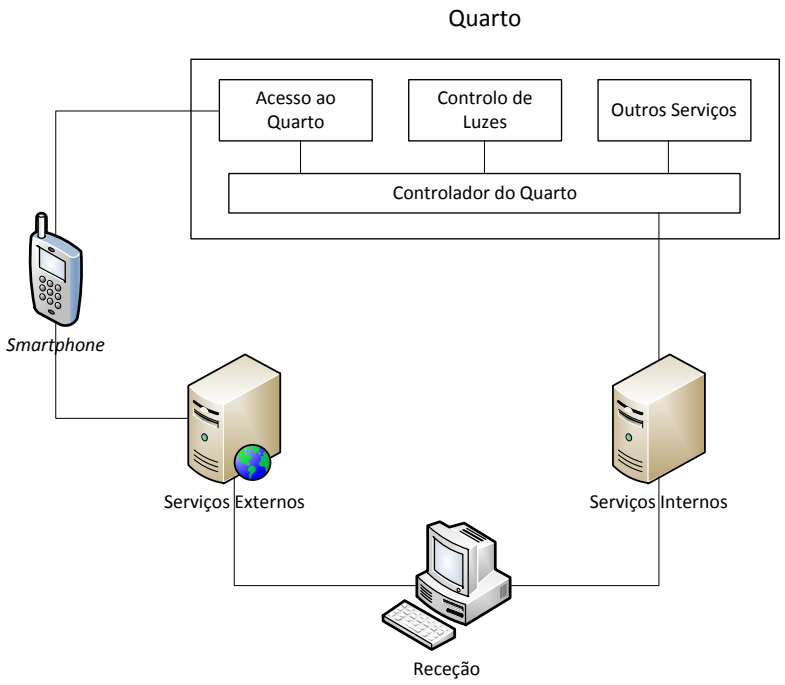


Figura 1 - Arquitetura do Sistema

Através do seu *smartphone* e da aplicação Android, um cliente poderá aceder aos serviços externos de modo a fazer a reserva de um quarto ou consultar as reservas já efetuadas. O cliente poderá também utilizar a aplicação para fazer o *check-in* por *Wi-Fi* ou então colocar o *smartphone* em contacto com um terminal de *check-in* para, através da tecnologia NFC, realizar este processo de forma automática. Após realizado este processo de *check-in*, o cliente

conseguirá abrir a porta do respetivo quarto, utilizando o seu *smartphone*. No fim da estadia, o cliente poderá também utilizar a aplicação para realizar o processo de *check-out,* podendo, tal como no processo de *check-in,* realizar o processo por *Wi-Fi* ou por NFC.

1.5 Organização do Documento

Este documento está dividido em seis capítulos. Estes capítulos permitem ao leitor uma absorção de conhecimento gradual, de modo a facilitar a compreensão dos temas abordados.

Assim, no primeiro capítulo encontra-se uma introdução à utilização de *smartphones* equipados com a tecnologia NFC, os objetivos propostos, uma exposição do problema e ainda as motivações para o desenvolvimento do projeto.

O segundo capítulo apresenta o Estado da Arte, isto é, apresenta uma introdução teórica dos temas abordados. Neste capítulo é descrita a tecnologia principal utilizada neste projeto, a tecnologia NFC, abordando assim os seus modos de uso, a segurança e os riscos da sua utilização. Este capítulo é fundamental para que sejam transmitidos os conhecimentos necessários para melhor interpretar o capítulo referente ao desenvolvimento do sistema.

O terceiro capítulo exibe casos de estudo de soluções já existentes. Com este propósito pretende-se elucidar o leitor para o que já se encontra desenvolvido e disponível no mercado.

No quarto capítulo são descritas as outras tecnologias utilizadas para desenvolver o projeto, abordando assim a tecnologia Android, os serviços *web* e o processo para garantir uma maior segurança dos dados. É ainda apresentado um estudo da utilização do modo HCE (*Host-based Card Emulation*) disponibilizado pela versão 4.4 do sistema operativo Android. Desse estudo fazem parte duas aplicações, uma utiliza o modo HCE para emular um cartão, a outra faz a leitura dos dados existentes nesse cartão.

No quinto capítulo é descrita a implementação do projeto. Nele estão contidos todas as etapas para a criação da aplicação móvel e respetivos serviços que serão disponibilizados pelo local de alojamento. É também descrita a aplicação para testar a componente NFC da aplicação móvel. Este capítulo contém ainda os resultados a um inquérito realizado. Esse inquérito tem como objetivo provar que a solução desenvolvida se encontra funcional e cumpre os objetivos propostos.

No sexto e último capítulo, apresentam-se as conclusões e trabalhos futuros. Neste capítulo ainda é realizada uma síntese sobre o trabalho realizado bem como as dificuldades encontradas no desenvolvimento do projeto.

2 Estado da Arte

Neste capítulo será apresentado o enquadramento teórico do trabalho. Serão abordados os principais conceitos, no sentido de aprofundar os conhecimentos do leitor relativamente à tecnologia NFC. O capítulo começa por uma breve descrição da evolução dos *smartphones*, seguido da descrição da tecnologia NFC e dos seus modos de utilização, terminando com as ameaças ao NFC e métodos de segurança implementados pela tecnologia NFC.

2.1 Evolução dos Smartphones

Nos últimos anos temos assistido a uma evolução extraordinária no mundo dos dispositivos móveis, tendo estes evoluído de simples telemóveis para os atuais *smartphones*. Este desenvolvimento começou com o aparecimento do primeiro dispositivo móvel em 1994, o IBM Simon (IBM corp, 1994). Este aparelho foi o primeiro dispositivo a possuir simultaneamente as funcionalidades de um telefone portátil e PDA (*Personal Digital Assistant*) sendo assim capaz, de efetuar e receber chamadas, bem como receber e enviar e-mails. Para além destas funcionalidades, o IBM Simon possuía também um número interessante de aplicações, incluindo: calendário, calculadora, agenda, entre outras.

Em 2003 a Microsoft lançou, um dos primeiros SO (Sistema Operativos) para dispositivos móveis, o Windows Mobile. Para além das normais funcionalidades de um telemóvel, possuía também a capacidade de interagir com outros produtos Microsoft lançados até à data, facilitando assim, a sua integração com produtos instalados em computadores pessoais ou profissionais. Em 2010 o Windows Mobile foi descontinuado, dando origem ao atual Windows Phone.

O SO Android começou por ser desenvolvido pela empresa Android Inc em 2003, sendo esta adquirida em 2005 pelo Google. Somente em 2007 é que o SO Android foi apresentado ao público em conjunto com a criação do Open Handset Alliance. Nesse mesmo ano de 2007 a Apple apresenta o seu primeiro *smartphone*, o iPhone, sendo posteriormente precedida pela HTC que, em 2008 lançou o primeiro *smartphone* com SO Android, o HTC Dream. No ano de

2010 a Google lança o seu primeiro *smartphone* equipado com NFC, o Nexus S (Clark, 2010a). Embora tenha sido o primeiro *smartphone* a utilizar a tecnologia NFC, não foi pioneiro, dado que à data do seu lançamento já existiam vários telemóveis equipados com esta tecnologia. O primeiro telemóvel com a tecnologia NFC foi desenvolvido no ano de 2006 pela Nokia através do modelo 6131 (Fraser, 2011).

De acordo com (Gartner,2014b) em 2013 foram vendidos aproximadamente 900 milhões de dispositivos com a tecnologia Android e em 2015 são esperados que sejam vendidos 1.3 milhões.

Relativamente à tecnologia NFC já existe um grande número de dispositivos com o SO Android equipados com esta tecnologia. Segundo (IHS Inc., 2014) foram vendidos em 2013 aproximadamente 275 milhões de dispositivos com esta tecnologia, prevendo que no ano de 2015 sejam vendidos aproximadamente 575 milhões. A partir destes dados é possível concluir que, em 2013, os dispositivos Android com tecnologia NFC correspondem a 30.6% sendo que em 2015 este número sobe consideravelmente para aproximadamente 51%.

2.2 Near Field Communication

O NFC (Curran, et al.,2012) é uma tecnologia sem contacto, que permite que dois dispositivos comuniquem entre si, utilizando ondas eletromagnéticas. O NFC é baseado na tecnologia RFID (*Radio-Frequency Identification*) desenvolvida por Charles Walton e encontra-se atualmente normalizado através da norma ISO/IEC 18092 (ISO/IEC, 2013). Esta norma é também conhecida por *NFC Interface and Protocol* (NFCIP-1).

O NFC permite a comunicação entre dois dispositivos até uma distância máxima de 4 cm e com uma velocidade máxima de transmissão de 424 Kbps. Todas estas especificações bem como as boas práticas para a utilização de NFC, encontram-se detalhadamente explicadas no NFC Forum[1], cujo objetivo é, impulsionar a utilização do NFC através do desenvolvimento de especificações, assegurar a interoperabilidade entre os dispositivos e serviços, e ainda educar o mercado sobre as potencialidades da utilização da tecnologia NFC.

O NFC possui três modos de operabilidade:

- Modo de leitura e escrita: permite que os dispositivos possam aceder a *smart cards* e *tags* NFC;
- Modo P2P (Ponto-a-Ponto): Este modo permite que dois dispositivos comunicquem entre si, sendo possível transferir documentos ou mesmo a troca de contactos virtuais;
- Modo de emulação de cartão: Este modo permite a um dispositivo emular um *smart card* e assim ser capaz de comunicar com leitores existentes de RFID.

[1] O NFC Forum Inc. é uma instituição sem fim lucrativos criada em 2004 pelas seguintes empresas: Nokia, Philips e Sony. O NFC Forum pode consultado em: http://nfc-forum.org/

2.2.1 Modo de Leitura/ Escrita

No modo de leitura e escrita um dispositivo móvel equipado com NFC tem a possibilidade de iniciar uma comunicação sem fios, de modo a ler ou alterar dados contidos em *tags* NFC, desde que estas sigam as normas existentes pelo NFC Forum (NFC Forum, 2009a). Uma *tag* é um dipositivo passivo, isto é, responde a um sinal enviado por um elemento ativo. Neste caso um dispositivo móvel com NFC é o elemento ativo, produzindo o sinal que será utilizado pela *tag*.

2.2.1.1 *Tags*

Foram definidas quatro *tags* pelo NFC Forum (NFC Forum, 2009a) designadas entre Tipo 1 e Tipo 4, existindo entre elas variações de formatos e capacidade. De seguida serão apresentadas as diferentes tipos de *tags* existentes:

- ***Tag* Tipo 1:** A *tag* do Tipo 1 é baseada na norma ISO/IEC 14443 (ISO/IEC,2011) Tipo A. Este tipo de *tag* pode ser, tanto de escrita como de leitura. Os dados contidos nestas *tags* podem ser alterados pelos utilizadores ou ainda serem reconfiguradas para passarem a ser somente de leitura. A memória disponível é de 96 bytes, o que é suficiente para armazenar um URL. A memória pode ser expandida até 2 kbytes e tem uma velocidade de comunicação de 106kbps. Estas características fazem com que este tipo de *tag* seja a melhor em termos de custo/eficiência;
- ***Tag* Tipo 2:** A *tag* de Tipo 2 é semelhante à *tag* de Tipo 1 tendo a particularidade de a memória inicial ser de apenas 48bytes ao invés dos 96 bytes iniciais de *tag* de Tipo 1;
- ***Tag* Tipo 3:** A tag de Tipo 3 é baseada na tecnologia FeliCa[2] e as *tags* são pré-configuradas pelo fabricante para poderem ser de leitura/escrita ou somente de leitura. Este tipo de *tags* tem uma memória de 2kb e a velocidade de comunicação é de 212kbps. Este tipo de *tag* é o mais indicado para aplicações complexas, tendo um valor mais dispendioso do que as *tags* anteriores;
- **Tag Tipo 4:** A *tag* de Tipo 4 é compatível tanto com as normas ISO/IEC 14443 Tipo A e Tipo B. Estas *tags* são pré-configuradas durante a fase de fabrico e podem ser de escrita ou somente de leitura. A memória é variável podendo ir até aos 32kb e a velocidade de comunicação varia entre os 106kbps e os 424kbps.

2.2.1.2 Pilha de protocolos

É possível visualizar na Figura 2 que a camada mais baixa da pilha de protocolos é a camada analógica. Esta camada está relacionada com as caraterísticas de RF (Rádio Frequência) existentes nos dispositivos NFC e determina ainda os alcances operacionais dos dispositivos.

A camada do protocolo digital refere-se à construção dos blocos de comunicação bem como os aspetos digitais contidos nas normas ISO/IEC 18092 e ISO/IEC 14443.

[2] Sistema de *smart card* utilizando RFID desenvolvido pela Sony.

A camada referente às operações com *tags* contém as instruções e os comandos que têm de ser utilizados pelos dispositivos de modo a poderem comunicar com as diferentes *tags* existentes.

A camada de aplicações NDEF (*NFC Data Exchange Format*) contém as aplicações que são baseadas nas especificações NDEF, como por exemplo, *smart posters*[3].

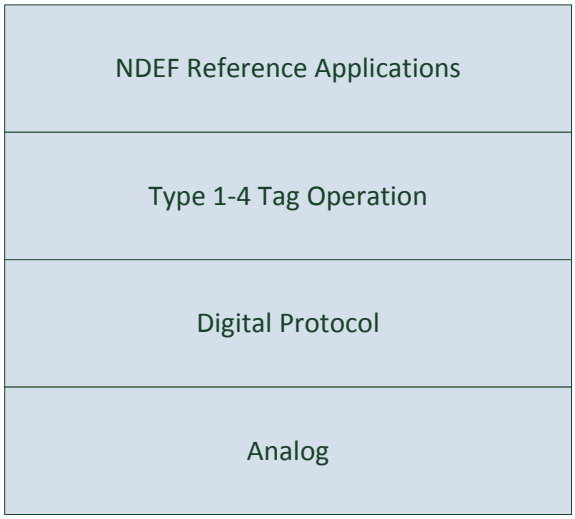


Figura 2 - Pilha de protocolos do modo leitura/escrita de uma *tag* NFC

2.2.2 Modo Ponto-a-Ponto (P2P)

No modo ponto a ponto dois dispositivos móveis equipados com NFC estabelecem uma ligação bidirecional com o objetivo de transferir dados de um dispositivo para o outro. Este modo de operação encontra-se normalizado pela norma ISO/IEC 18092.

2.2.2.1 Pilha de protocolos

É possível visualizar na Figura 3 a pilha de protocolos referente ao modo P2P. Nesta, para além das camadas Analógica e Protocolo Digital, que foram explicadas anteriormente, encontra-se a camada do *Logical Link Control Protocol* (LLCP). Este protocolo é o que suporta toda a comunicação P2P entre dois dispositivos e é essencial para que exista uma comunicação bidirecional. LLCP é um protocolo baseado na norma IEEE 802.2 desenhada para suportar transferências de ficheiros pequenos ou protocolos de rede tais como OBEX e TCP/IP.

Na camada seguinte encontra-se o *Protocol Bindings,* cuja função é permitir a interoperabilidade entre os diferentes protocolos, e o *NDEF Exchange Protocol,* cuja função é permitir que sejam trocadas mensagens do tipo NDEF.

[3] *Smart posters* são normalmente imagens que contém uma tag NFC

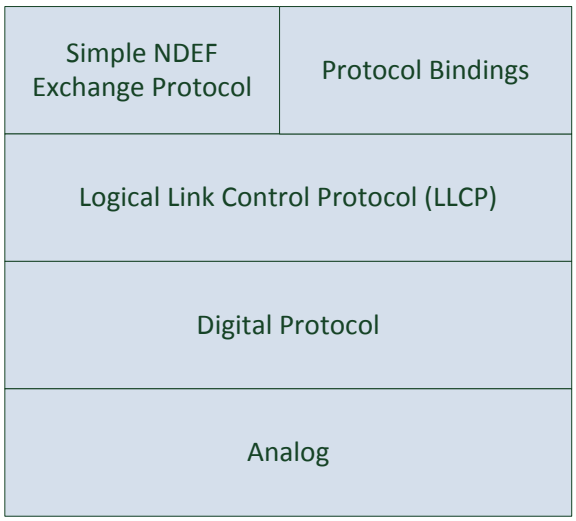


Figura 3 - Pilha de protocolos do modo P2P

2.2.3 Emulação de Cartão

No modo de emulação de cartão, um dispositivo NFC age como um *smart card*. Neste processo o dispositivo emula a norma ISO/IEC 14443-4 (ISO/IEC, 2008) de modo a poder usufruir de todas as potencialidades que a norma dos *smart cards* fornece. Dos modos de funcionamento do NFC este é o único que pode utilizar um Elemento Seguro (ES), dado que, existem funcionalidades que necessitam de um local seguro para armazenar informação. Por exemplo, para utilizar o NFC com o objetivo de emular um cartão de multibanco é necessário que determinados dados não possam estar disponíveis para serem consultados. O ES garante ainda, que determinadas operações sejam efetuadas com segurança.

2.2.3.1 Elemento Seguro

De acordo com (Roland, 2012) o ES tem como objetivo garantir às aplicações um ambiente seguro para guardar e executar dados críticos. Desta forma o ES permite delimitar um certo espaço de memória para cada aplicação de modo a armazenar dados críticos, funções de encriptação / decriptação e ainda assinar o pacote de dados.

O ES consiste num *microchip* que pode estar incorporado no mecanismo de NFC do dispositivo móvel, ou estar incorporado no *Subscriber Identity Module* (SIM), sendo que deste modo se encontra controlado pelas operadoras de telecomunicações, ou ainda estar incorporado num cartão *Secure Digital* (SD).

2.2.3.2 Pilha de Protocolos

É possível visualizar na Figura 4 as camadas que constituem a pilha de protocolos do modo de emulação de cartão. A camada Analógica e a camada do Protocolo Digital são as mesmas utilizadas pelos *smart cards,* de modo a que possa existir uma absoluta compatibilidade com este tipo de cartões. A camada das Aplicações é referente a aplicações

proprietárias, tais como aplicações de pagamento, baseadas nas normas ISO/IEC 14443 Tipo A, Tipo B e FeliCa.

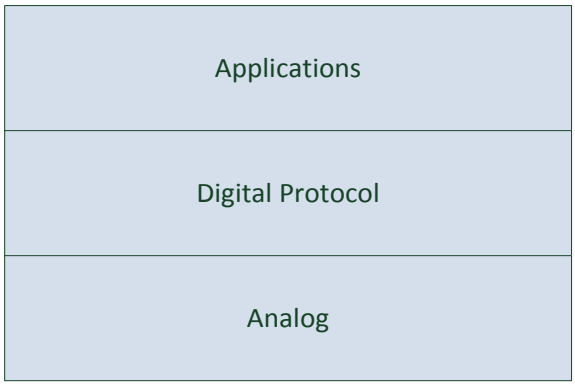


Figura 4 - Pilha de Protocolos do modo Emulação de Cartão

2.2.3.3 Emulação de Cartão com ES por *Hardware*

A Figura 5 apresenta os vários caminhos que os dados podem seguir num dispositivo com NFC. O Controlador NFC é o componente principal da funcionalidade NFC e o ES é um *microchip* cuja finalidade é garantir a segurança de dados privados, tais como informações de cartões de crédito ou números de identificação pessoais. O *Central Processing Unit* (CPU) do dispositivo é o principal processador do mesmo.

Figura 5 - Caminhos da informação num dispositivo com NFC

Todos os modos de operabilidade da tecnologia NFC disponíveis interagem primeiro com o controlador NFC do dispositivo (1). Posteriormente, caso não seja necessário nenhuma

informação armazenada pelo ES para comprovar a identidade do utilizador, o controlador NFC interage com o CPU do dispositivo de modo a executar o pretendido (2). Caso seja necessário aceder à informação armazenada pelo ES, o Controlador NFC interage com o mesmo de modo a obter esses dados (3).

2.2.3.4 Emulação de Cartão com ES por *Software*

O *Software Secure Element* (*Soft-SE*) foi introduzido nos dispositivos móveis pela *Research in Motion* (RIM) para a plataforma BlackBerry. Atualmente o *Soft-SE* também se encontra disponível para outras plataformas como o SO Android. O *Soft-SE* permite assim que deixe de ser necessário existir um *hardware* específico para realizar as funções do ES, passando essa função a ser executada pelo *software* do dispositivo. Esta abordagem vem retirar às empresas de telecomunicações o controlo que estas têm sobre as transações efetuadas, sobre NFC, dado que são estas que fornecem o método mais utilizado para fazer de ES, os cartões SIM.

Figura 6 - Caminhos da informação num dispositivo com NFC através de *Soft-SE*

Como se pode observar na Figura 6 a interação com o ES passa a ser realizada por *software* instalado no dispositivo, não sendo então necessária utilização de *hardware* específico para o efeito.

2.3 Ameaças à utilização de NFC

Existem vários problemas já identificados (Haselsteiner et al., 2006), (Mulliner, 2009), (Francis et al., 2013) relativamente a ameaças utilizando NFC. Alguns desses problemas são: a

escuta de comunicações, negação de serviço, inserção de dados, *Man-in-the-Middle* e *Relay Attack.*

2.3.1 Escuta das Comunicações

Também conhecido por *sniffing* este é o tipo de ataque mais comum, associado a comunicações sem fios. Os dispositivos ao comunicarem através de RF, permitem que um atacante utilize uma antena para escutar as comunicações. Para interpretar os dados que recebe, o atacante pode simplesmente consultar a documentação disponível ou através de experiencias próprias. Existe também uma diferença entre escutar um dispositivo em modo ativo ou em modo passivo. Em modo ativo, um dispositivo cria o seu próprio sinal, enquanto que em modo passivo, o dispositivo utiliza o sinal criado por outro dispositivo. Como o modo de transmissão de dados é diferente em ambos os casos, torna-se assim mais difícil escutar dispositivos que enviem dados em modo passivo.

2.3.2 Negação de Serviço

O objetivo deste tipo de ataque é impossibilitar um dispositivo de receber os dados enviados por outro dispositivo. Para realizar o ataque, o atacante tem de transmitir frequências dentro do espetro correto, de modo a sobrecarregar de dados o dispositivo que atua como recetor.

2.3.3 Inserção de Dados

Neste ataque, o atacante tenta inserir dados na comunicação entre dois dispositivos. Este tipo de ataque necessita que as comunicações tenham um tempo de resposta bastante demorado de modo a que, esse intervalo, seja utilizado para introduzir os dados por parte do atacante. O ataque só terá sucesso caso o ataque seja feito antes da resposta por parte do dispositivo original. No caso das duas mensagens se sobreporem os dados ficam corrompidos.

2.3.4 Man-in-the-Middle

Neste tipo de ataque, os dispositivos intervenientes na comunicação pensam que estão a comunicar entre eles, contudo estão a comunicar através de um terceiro elemento. Este terceiro elemento, o atacante, recebe as mensagens enviadas por um dispositivo e de seguida reenvia-as para o outro dispositivo. Como o NFC é uma tecnologia de curta distância este tipo de ataque não costuma ser muito utilizado.

2.3.5 Relay Attack

Este ataque (Francis et al., 2013) consiste num dispositivo fazer a leitura de um *smart card* e enviar essa leitura para um segundo dispositivo para, ser posteriormente usado num leitor RFID. A Figura 7 ilustra o ataque para que se obtenha uma melhor compreensão.

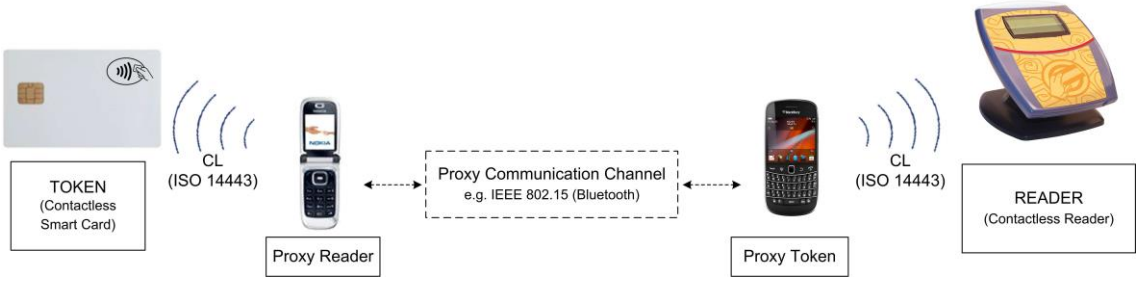


Figura 7 – Exemplo prático de um *Relay Attack* (Francis et al., 2013)

Com exceção ao *Relay Attack,* a todos os outros se coloca a questão da distância a que o atacante tem de estar de modo a poder realizar os referidos ataques. A resposta para esta questão é que não há uma distância exata pois existem alguns factores: físicos (tipo de antena utilizada, força do sinal emitido) ou ambientais (ruido, existência de paredes) que são necessários ter em consideração.

2.4 Protocolos de Segurança do NFC

Os protocolos de segurança somente são utilizados nas operações P2P e encontram-se normalizados pela norma ECMA 385 (ECMA, 2013), o NFC-SEC, e pela norma ECMA 386 (ECMA, 2010), o NFC-SEC-01. Normalmente o uso de mecanismos de segurança faz diminuir o desempenho mas no caso do NFC-SEC existe uma boa combinação entre as duas vertentes.

O protocolo NFCIP-1 não possui qualquer tipo de segurança, tendo sido criado posteriormente para o efeito o NFC-SEC. Após a sua criação o NFC-SEC, passou a ser executado em cima do NFCIP-1 e consiste num conjunto de protocolos capazes de dar a uma aplicação funções de encriptação, colmatando as falhas do NFCIP-1.

O NFC-SEC fornece proteção contra ataques de escuta e de alteração dos dados, enquanto o NFC-SEC-01 acrescenta mecanismos de encriptação de dados tais como *Elliptic curve Diffie–Hellman* (ECDH) e o *Advanced Encryption Standard* (AES). O protocolo ECDH é utilizado para a partilha de chaves, enquanto o algoritmo AES é utilizado para encriptar os dados e manter a sua integridade.

Para garantir a segurança, o NFC-SEC, disponibiliza os seguintes serviços: o *Shared Secret* (SSE) e o *Secure Channel* (SCH). O SSE consiste em partilhar uma chave entre os dois pares da comunicação NFC de modo a que estes encriptem os seus próprios dados. Já o serviço SCH

consiste em criar um canal de comunicações seguro onde permita que todas as comunicações realizadas pelos intervenientes sejam seguras.

A Figura 8 apresenta os procedimentos que o protocolo NFC-SEC tem de realizar. O protocolo começa por realizar um acordo com ambos os intervenientes na comunicação. Assim ambas as partes devem estabelecer uma chave secreta utilizando o ACT_REQ (*Activation Request PDU*), para requisitarem um serviço, e o ACT_RES (*Activation Response PDU*), para aceitar o pedido do serviço. De seguida surge a confirmação da chave, onde ambas as partes devem verificar a chave secreta através do VRY_REQ (*Verification Request PDU*), para verificar o pedido da chave secreta do emissor, e do VRY_RES (*Verification Response PDU*), para verificar o pedido da chave secreta do recetor. Posteriormente deve ser escolhido o tipo de serviço que será utilizado, o SSE ou o SCH. Se for optado pelo SSE as trocas de dados entre os dispositivos pode começar, caso seja escolhido o SCH necessita ainda de ser efetuada mais uma validação de segurança, a segurança do PDU (*Protocol Data Unit*). Esta segurança consiste em que ambos os intervenientes protejam a troca de dados utilizando ENC (*Encrypted Packet PDU*). A segurança do PDU permite assim garantir a integridade, a confidencialidade e a origem dos dados.

Após as transferências dos dados estarem concluídas, o protocolo necessita de ser terminado.

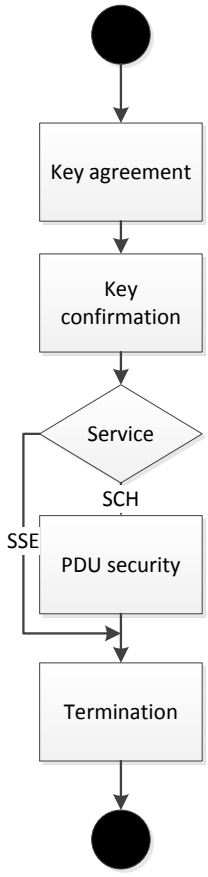


Figura 8 – Passos do protocolo NFC-SEC (ECMA, 2013)

3 Utilização do NFC

Um dispositivo equipado com NFC apresenta uma nova variedade de utilizações que até então não se encontravam disponíveis. Nas próximas secções serão apresentadas algumas dessas utilizações, bem como casos de estudo para cada uma delas.

3.1 Pagamentos

Provavelmente o modo de utilização do NFC mais conhecido é o da utilização de um dispositivo móvel para realizar pagamentos. Por exemplo, um cliente desloca-se a um supermercado e perante os modos de pagamento atuais tem de pagar com cartão multibanco ou com dinheiro. Se optar por pagar com dinheiro implica ter de levar sempre consigo o valor necessário para as referidas compras. Em muitos casos, são feitas grandes quantidades de compras, o que implica, que tenha de ser transportada uma avultada quantia de dinheiro, o que em muitas situações não é seguro. Comparando com o pagamento em dinheiro, o processo de pagamento por cartão multibanco é mais rápido e mais eficiente. No entanto, no caso de o cliente possuir vários cartões multibanco terá de escolher o mais conveniente e de seguida colocá-lo pela orientação correta e introduzir o PIN (*Personal Identification Number*) para concluir a compra.

Uma aplicação de pagamento que utilize a tecnologia NFC torna o processo de pagamento mais rápido e eficiente. Por exemplo, um utilizador simplesmente necessita de passar o seu telemóvel com NFC num dispositivo que faça de leitor e deverá ser inserido um código PIN de modo a tornar o método de pagamento mais seguro. Existe também a vantagem de o mesmo telemóvel poder possuir vários cartões multibanco possibilitando desta forma ao utilizador selecionar o que melhor se adequa ao pagamento em questão. É possível ainda que o telemóvel seja utilizado para armazenar cupões promocionais, acabando deste modo, com a utilização de papel ou de outro dispositivo que contenha essas informações.

A aplicação (Google Wallet, 2011*)* foi a primeiro grande impulsionadora da utilização de um *smartphone* com NFC para realizar pagamentos. Disponível desde 2011, e somente nos Estados Unidos da América, o Google Wallet é uma aplicação gratuita que tem como objetivo armazenar os cartões multibanco de um utilizador para posteriormente os utilizar tanto em terminais de pagamento como em pagamentos *on-line*. Esta função de pagamento em terminais é conhecida por *Tap and Pay* e só funciona em dispositivos que possuam um cartão SIM proveniente dos Estados Unidos da América. Apesar de ser uma aplicação que utiliza NFC nem todos os dispositivos com NFC existentes no mercado a podem utilizar, isto é, atualmente somente dispositivos equipados com a versão de *software* 4.4 (*KitKat*) podem utilizar a aplicação.

Para além das funções de pagamento, o Google Wallet permite ainda o armazenamento de *vouchers* ou outros cupões promocionais, de modo a que possam ser utilizados nos respetivos estabelecimentos. Este armazenamento de vouchers facilitam o utilizador, na medida em que muitas vezes os *vouchers* são perdidos ou esquecidos no ato de pagamento.

Embora seja uma aplicação desenvolvida pela Google, funciona também em dispositivos que não utilizem o sistema operativo Android, encontrando-se deste modo disponível para ser utilizado em *iPhone/iPad* com *software* superior à versão 6. Apesar de a aplicação estar disponível para *iPhone/iPad*, o modo de funcionamento é diferente pois atualmente estes dispositivos não possuem NFC.

O modo de pagamento utilizando o Google Wallet pode ser visualizado na Figura 9 e consiste na interação entre o *smartphone* e o terminal de pagamento. No caso de possuir vários cartões armazenados no *smartphone* deve ser escolhido o cartão adequado antes de proceder ao pagamento. O mesmo processo pode ser feito de modo a utilizar *vouchers* adquiridos antecipadamente.

Figura 9 – Pagamento com Google Wallet (Haselton, 2014)

Em 2012 surge o principal concorrente do Google Wallet, o Isis Mobile Wallet (GSM Association, 2013). Esta aplicação é em tudo semelhante ao Google Wallet, isto é, somente está disponível nos Estados Unidos e disponibiliza as mesmas funcionalidades.

Na China existe já em funcionamento um sistema de pagamento diferente dos anteriores, denominado SIMpass (Watchdata,2011). Este sistema permite a que telemóveis sem tecnologia NFC possam realizar pagamentos utilizando esta tecnologia. Para tal é necessário adquirir uma peça de *hardware*, o SIMpass. Este consiste numa antena NFC externa e de um cartão SIM especial que, como pode ser visualizado na Figura 10 – SIMpass (Watchdata, 2013), se encontram interligados. O cartão SIM é colocado no compartimento habitual e a antena NFC é colocada entre a bateria e a tampa do telemóvel. Após incorporar o SIMpass no telemóvel este fica automaticamente capaz de realizar pagamentos por NFC nos terminais de pagamento.

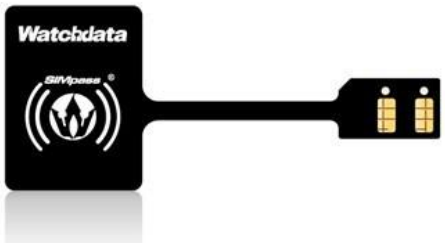


Figura 10 – SIMpass (Watchdata, 2013)

3.2 Ticketing

Mobile Ticketing (GSM Association, 2011) é outra possibilidade para o uso de NFC e consiste na compra de bilhetes ou assinaturas de títulos de transporte. Este modo torna mais conveniente a compra de bilhetes por parte do utilizador, pois evita as filas de espera sendo possível, por exemplo, adquirir os bilhetes *on-line* em vez de num posto de venda. Um dispositivo com uma aplicação de *Ticketing* possibilita a consulta dos bilhetes, sabendo assim o tipo de título de transporte e se estão ou não carregados, sem ser necessário recorrer a outro tipo de máquina. Outra vantagem deste modo passa pela robustez de um smartphone comparado com um bilhete tradicional de papel ou mesmo um *smart card*, sendo mais provável perder-se um bilhete deste tipo do que o smartphone.

Esta utilização do NFC é bastante vantajosa para as empresas de transportes, pois permite reduzir os custos do fabrico de bilhetes, assim como da manutenção das máquinas e dos funcionários das bilheteiras. Do ponto de vista ambiental este processo é também muito vantajoso pois elimina-se a utilização do papel para o fabrico dos bilhetes.

Foi criado na Alemanha em 2008 o projeto *Touch&Travel* (NFC Forum Inc., 2011) o qual permitia ao cliente de serviços de transporte a utilização de um dispositivo móvel para a compra e validação de títulos de transporte.

Para utilizar o sistema, o utilizador tinha de passar o seu telemóvel num *Touchpoint* que se encontrava na estação ou paragem, como é possível visualizar na Figura 11. Este *Touchpoint* contém uma *tag* NFC que possui informação sobre a estação. Após esta interação o telemóvel

envia para o serviço de *back end* a informação do local onde foi utilizado e o *back end* retorna um bilhete válido para a viagem. Este bilhete fica guardado no cartão SIM e só pode ser acedido por um agente autorizado de modo a que, durante a viagem, possa validar o bilhete. Quando chega ao destino o utilizador necessita novamente de passar o telemóvel no *Touchpoint* do local de destino de modo a que seja calculado o custo da viagem.

A aceitação dos utilizadores ao projeto foi de tal modo positiva que o projeto ainda se encontra ativo. Neste momento para além da aplicação inicial, existem novas aplicações para *smartphones* com mais funcionalidades do que somente a aquisição de títulos de transporte.

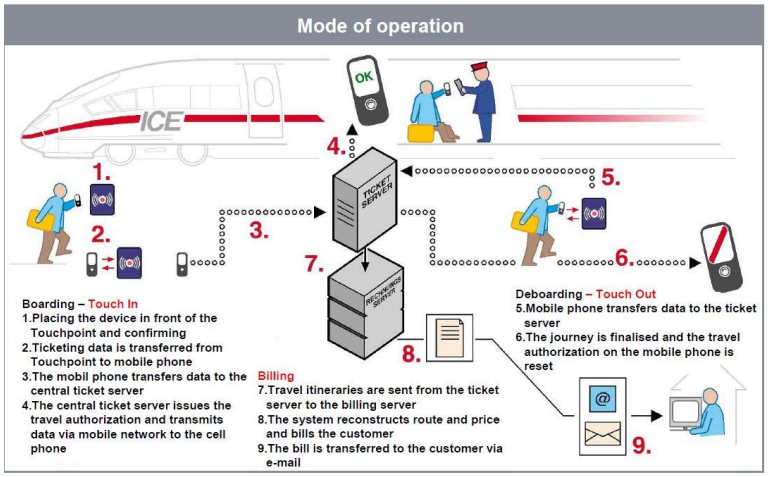


Figura 11 – Modo de operação Touch&Travel (NFC Forum Inc., 2011)

Outro exemplo da utilização de NFC para *Ticketing* é o caso do *Transport for London* (TfL) (Blaban, 20012), que em 2007 desenvolveu um projeto piloto para a utilização de NFC para o pagamento de bilhetes nos transportes públicos de Londres. Embora tenha sido um projeto com uma taxa de aprovação 92 %, o sistema não foi implementado fora do período de testes. Uma das razões que originaram essa decisão foi o facto de a partir de 2008 o ES ter de ser incorporado nos cartões SIM. Esta mudança levou a que as transações ultrapassassem os 500 milissegundos tornando o processo de validação do bilhete demasiado lento para ser utilizado.

3.3 Controlo de Acesso

O NFC pode também ser utilizado de modo a controlar o acesso a salas ou edifícios. Partindo do mesmo princípio da utilização de NFC para efetuar pagamentos, também os *smart cards* utilizados para abrir portas podem ser armazenados no dispositivo. O facto de vários cartões poderem ser armazenados dentro do mesmo dispositivo dão a este sistema, um dos principais pontos fortes para a sua utilização. De realçar que o NFC sendo uma tecnologia sem contato e de ser compatível com os sistemas RFID já existente, melhora a transição entre o sistema de *smart cards* existente e esta nova abordagem para o controlo de acesso a salas ou edifícios. O NFC pode ainda servir para garantir o controlo de acesso a viaturas ou acesso a cofres, por exemplo, em hotéis. Este sistema permite que no caso de ser necessário garantir

um novo acesso a uma pessoa não seja necessária nenhuma chave física, isto é, não é necessário nem copiar uma chave nem criar um novo *smart card*, podendo simplesmente o acesso ser atribuído através de uma aplicação existente no smartphone.

O primeiro hotel a utilizar NFC como método de acesso foi em 2010 o *Clarion Hotel* em Estocolmo, Suécia. De modo a testar este novo método de acesso foi criado um projeto teste (Assa Abloy, 2011) em que alguns dos clientes do hotel receberiam um telemóvel com NFC e com uma aplicação própria para o efeito. Esta aplicação permitir-lhes-ia não só, aceder ao quarto, como realizar o processo de *check-in* e *check-out* sem ter de se deslocarem à receção do hotel.

Assim após fazerem a reserva os clientes recebiam no dia selecionado um SMS que continha uma ligação para uma aplicação *on-line* onde seria realizado o *check-in*. Este processo de *check-in* sendo bem-sucedido, o cliente receberia, no seu telemóvel, o número do quarto bem como o acesso ao mesmo. O processo de atribuição de uma chave pode ser visualizado na Figura 12, onde após o *check-in* ser realizado e de o hotel atribuir um quarto ao cliente, uma chave é enviada do hotel para um servidor externo (ambiente seguro), que irá encriptar a chave e posteriormente enviar para o cartão SIM do cliente.

Chegando ao hotel o cliente não necessitava de fazer o *check-in* na receção podendo imediatamente deslocar-se para o seu quarto. De modo a abrir a porta do quarto simplesmente seria necessário colocar o telemóvel em contato com o leitor existente na porta e caso o acesso fosse válido a porta abrir-se-ia. Para realizar o processo de *check-out* era necessário colocar o telemóvel em contato com uma *tag* NFC existente no hotel e confirmar o processo.

As conclusões deste projeto foram bastante positivas e encorajadoras para a utilização desta tecnologia. A maioria dos participantes fizeram referência ao facto de pouparem tempo no processo de *check-in* e referiram ainda que caso o seu telemóvel suportasse NFC, utilizá-lo-iam para este tipo de acesso.

Outro sistema desenvolvido foi o Key2Share (Dmitrienko, 2013) e possui um modo de operação semelhante ao anterior. No entanto este sistema foi desenhado de modo a suportar outros casos de uso relacionados com a autorização de acesso, como receber uma chave eletrónica de um automóvel. Como é possível observar na Figura 13, uma empresa envia para o seu colaborador uma chave eletrónica de modo a que este se registe no servidor da empresa. De seguida o colaborador cria o seu registo, utilizando a respetiva chave, e recebe da parte do servidor uma chave definitiva que lhe garante acesso ao edifício. Após este processo o novo colaborador fica com acesso ao edifício utilizando o seu *smartphone*.

Outro sistema existente é o NFCPorter (IMA s.r.o, 2013) e já possui uma aplicação disponível no mercado. No entanto este sistema possui uma diferença significativa comparado com os sistemas anteriores pois não opera sobre leitores NFC/RFID já implementados. Para este sistema ser utilizado é necessário que sejam adquiridos leitores NFC próprios criados pela empresa que disponibiliza o NFCPorter. Esta necessidade de adquirir novo *hardware* é uma

grande desvantagem perante os outros sistemas que funcionam sobre os leitores NFC/RFID existentes.

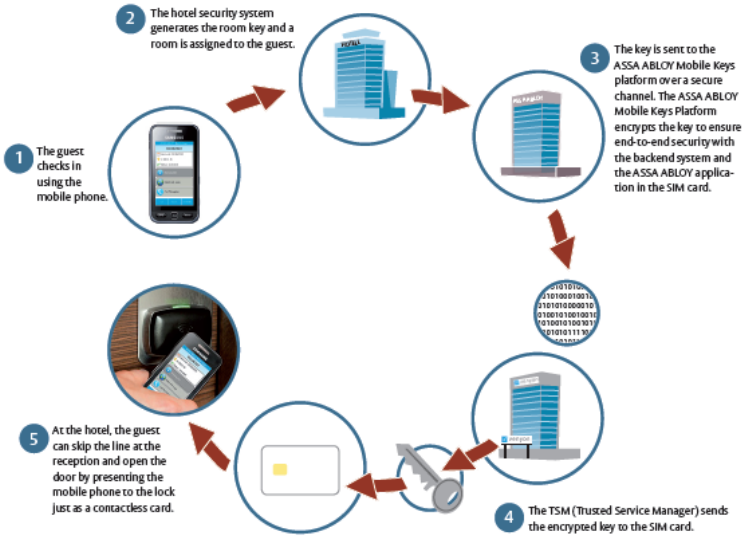


Figura 12 – Processo de atribuição de chave do quarto de hotel (Assa Abloy, 2011)

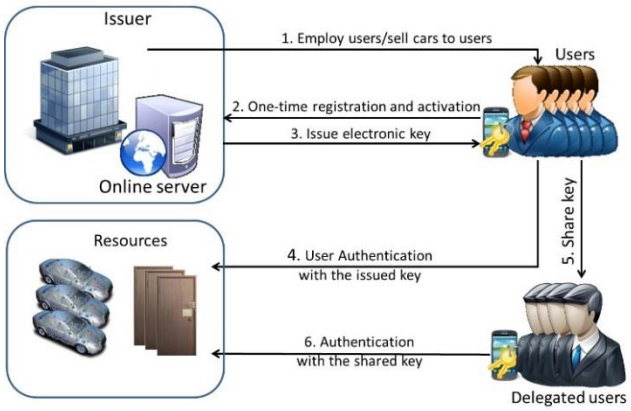


Figura 13 – Processo de atribuição de chave eletrónica (Fraunhofer, 2013)

3.4 Saúde

A tecnologia NFC pode ainda ser utilizada na área da saúde, mais concretamente nos hospitais de modo a facilitar a identificação de um paciente ou para reduzir as taxas de erro existentes na utilização de medicamentos. Entre esses erros encontram-se a incorreta dosagem de medicação, administrar medicamentos fora da hora prevista ou administrar um medicamento errado. Tirando partido do modo de leitura/escrita é possível utilizar *tags* NFC para identificar o paciente, de modo a que, em caso de emergência, seja possível com maior rapidez aceder às suas informações pessoais, como o tipo de sangue ou alergias. As *tags* NFC

podem também ser inserida nos recipientes que contêm os medicamentos para cada paciente, contendo as dosagens corretas, o meio e a hora a que devem ser administrados, de modo a reduzir os erros que ocorrem durante a sua utilização.

Surgiu em 2008 no Kuopio University Hospital um projeto que propunha a utilização da tecnologia NFC de modo a reduzir os erros associados à administração de medicamentos durante a presença do paciente no hospital (Lahtela, et al.,2008).

A Figura 14 apresenta o funcionamento deste sistema. Neste, um funcionário da enfermaria começa por inserir no sistema de processamento de ordens uma lista de medicamentos que necessita. De seguida esta lista é enviada para a farmácia do hospital de modo a que o farmacêutico insira os pedidos no distribuidor automático de medicação. Este distribuidor informa o sistema de informação que por sua vez envia para o leitor NFC a informação correta sobre o medicamente e sobre o paciente que dele necessita. Esta informação é posteriormente escrita numa *tag* NFC que será introduzida no recipiente que contém o medicamento.

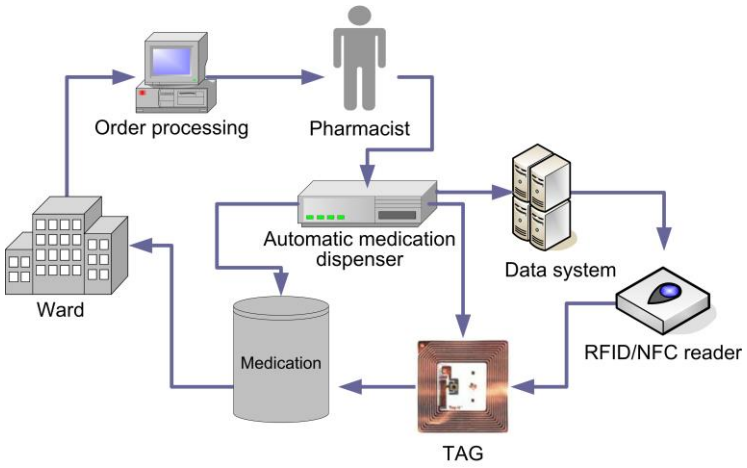


Figura 14 – Distribuição da medicação (Lahtela, et al.,2008)

Quando o medicamento chega à enfermaria a enfermeira necessitará de um aparelho com tecnologia NFC para ler as informações contidas na *tag* do medicamento, como se pode observar na Figura 15.

Num sistema onde, para além dos medicamentos, o paciente também possui uma *tag* com as suas informações pessoais (nome, tipo de sangue, alergias) os erros associados à administração de medicamentos tornar-se-ão menores, melhorando assim o sistema de saúde.

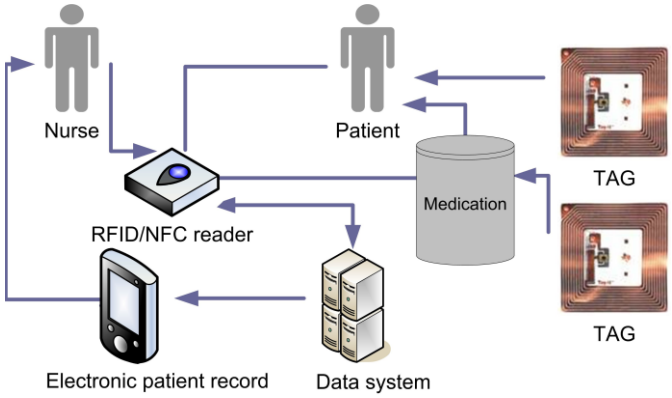


Figura 15 – Utilização de NFC em medicação (Lahtela, et al.,2008)

3.5 Cultura

Como referido anteriormente pode ser utilizada uma aplicação de *Ticketing* de modo a comprar e utilizar bilhetes. No caso de um evento cultural podem ser adquiridos por exemplo bilhetes para uma peça de teatro, um filme ou mesmo para entrada num museu. Atualmente já existem alguns museus que utilizam uma aplicação baseada em NFC para que os seus visitantes possam, não só adquirir o ingresso, como interagir com a exposição. Esta interação consiste em utilizar o *smartphone* para consultar informações sobre uma obra de arte, como por exemplo o seu autor, ano ou informação sobre o período a que pertence.

Desde 2011 o *Museum of London* disponibiliza aos seus visitantes uma aplicação que permite a quem possuir um *smartphone* com NFC, utilizar uma variedade de serviços. Entre esses serviços, encontra-se um que permite aos visitantes adquirirem mais informações sobre as obras que estão a ver, necessitando somente de passar o *smartphone* pelo *smart poster* da respetiva obra como se pode visualizar pela Figura 16.

Figura 16 – *Smart poster* (Ceipior, et al.,2013)

Desde 2010 existe em França o *Smart Muse Mobile NFC Guide* (Clark, 2010b). Um projeto semelhante ao do *Museum of London* no entanto tem a particularidade de não se restringir somente a museus. Neste caso a aplicação pode ser utilizada também para consultar informações espalhadas pelo centro histórico de Nice. Outros projetos semelhantes são: o *Wolfsoniana Museum* (Ceipior, et al.,2013), o *Ancient Italian Renaissance Villas* (Angelaccio, et al., 2012) e o *Please Touch the Exhibits!* (Blöckner, et al., 2009).

4 Tecnologias em ambientes Móveis

Neste capítulo apresentam-se as tecnologias utilizadas para desenvolver o sistema proposto. Deste modo é feita uma descrição do SO Android, da utilização de serviços e dos métodos utilizados de modo a tornar, mais seguros, determinados dados guardados pelo sistema. Neste capítulo, e com o objetivo de compreender melhor o modo de funcionamento da tecnologia NFC em Android, foram também estudados dois exemplos de aplicações disponibilizadas pela plataforma de desenvolvimento Android. Uma destas aplicações demonstra como emular um cartão NFC utilizando a funcionalidade de HCE (*Host-based Card Emulation*), disponibilizada pela versão Android 4.4. A outra aplicação demonstra como implementar um leitor de cartões NFC, neste caso, cartões que não contenham NDEF ou dados *Android Beam*. Ambas as aplicações foram necessárias para analisar e compreender o funcionamento da tecnologia NFC, bem como compreender as trocas de mensagens entre as duas aplicações.

4.1 Android

O sistema operativo Android foi desenvolvido por um conjunto de empresas, o Open Handset Alliance, e tornou-se no sistema operativo móvel com maior taxa de crescimento, com cerca de 1 milhão de novas ativações todos os dias.

De acordo com (Saha, 2008) o SO Android consiste numa pilha de *softwares* para dispositivos móveis que inclui um sistema operativo, um *middleware* e um conjunto de aplicações-chave, tais como calendário, *browser*, contactos entre outras. As aplicações normalmente são escritas usando a linguagem de programação Java e executam sobre uma máquina virtual desenvolvida para o sistema operativo Android, o *Dalvik*. Para desenvolver aplicações para esta plataforma é necessário o *Software Development Kit* (SDK) e conhecer a API (*Application Programming Interface*). O SDK consiste num conjunto de ferramentas de desenvolvimento contendo documentação, tutoriais, um emulador entre outas funcionalidades.

A pilha de *software* Android pode ser dividida em cinco camadas: núcleo e ferramentas de baixo nível, bibliotecas nativas, *Android Runtime*, *Application Framework* e a um nível mais elevado, a camada das aplicações.

A camada referente ao núcleo baseia-se no *kernel Linux* e funciona como camada abstrata entre o hardware e a pilha de softwares da plataforma. As bibliotecas nativas encontram-se escritas em C/C++ e disponibilizam recursos aos restantes componentes da pilha. A camada *Android RunTime* é responsável pelo processo de execução de aplicações desenvolvidas na plataforma Android, enquanto a camada *Application Framework* compreende um conjunto de serviços e sistemas usados no processo de desenvolvimento. A última camada, *Applications*, inclui um conjunto de aplicações básicas, como calendário, mapas, navegadores, entre outras.

4.2 SOA (Service Oriented Architecture)

SOA estabelece uma arquitetura de *software* que promove a cooperação entre os diversos sistemas de modo mais eficiente e ágil. É assim um modelo agnóstico de qualquer plataforma tecnológica, isto é, deve ser possível utilizar esta arquitetura independentemente da tecnologia. Como se trata de uma arquitetura orientada a serviços, pretende que cada serviço seja independente, no entanto, capaz de colaborar com outro serviço. Esta colaboração torna os serviços capazes de executarem tarefas mais complexas.

De acordo com (W3C Working Group, 2004), uma arquitetura SOA tem as seguintes caraterísticas:

- Visão Lógica: Um serviço é uma abstração lógica da aplicação, isto é, abstrai processos de negócio, bases de dados, entre outros;
- Orientado a Mensagens: Um serviço é definido em termos das mensagens que troca entre os seus intervenientes e não entre as propriedades dos mesmos. A estrutura interna dos intervenientes, tais como linguagem de implementação, estrutura dos processos e base de dados, são deliberadamente abstraídos;
- Orientado à descrição: Um serviço é descrito através de meta dados de modo a poderem ser expostos publicamente. A semântica do serviço deve estar documentada, direta ou indiretamente, pela sua descrição;
- Granularidade: Um serviço usa um número pequeno de operações com um grande número de mensagens complexas;
- Orientado à rede: Um serviço tende a ser orientado para utilização numa rede, embora não seja um requisito obrigatório;
- Neutro de plataformas: As mensagens são enviadas num modo padronizado e independente da plataforma e entregues através de uma interface. O XML é um dos formatos que garante esta neutralidade.

É possível constatar que este tipo de arquitetura promove a distribuição dos sistemas, garantindo no entanto a interoperabilidade entre sistemas heterogéneos. Esta distribuição dos sistemas traz alguns desafios, tais como:

- Problemas relacionados com a fiabilidade e latência associados ao meio de transporte;
- A falta de memória partilhada entre os intervenientes do serviço;
- Problemas associados com falhas parciais do sistema;
- Desafios relativamente ao acesso concorrente a recursos remotos;
- A fragilidade do sistema distribuído se forem feitas alterações que tornem incompatível um serviço.

Atualmente a utilização que mais facilmente se associa à SOA são os serviços *web*. Embora estejam associados, um serviço *web* não transforma um sistema distribuído automaticamente numa arquitetura SOA. Embora existam outras formas de implementar uma arquitetura SOA, no desenvolvimento deste projeto, somente serão apenas utilizados serviços *web* como meio de comunicação entre os diferentes sistemas.

Segundo (Thomas Erl, 2007) a primeira geração de plataformas para o desenvolvimento de serviços *web* é constituída por: *Web Service Description Language* (WSDL), *XML Schema Defenition Language* (XSD), *Simple Object Access Protocol* (SOAP), *Universal Description, Discovery, and Integration* (UDDI) e o *Web Services Interoperability Basic Profile* (WS-I BP). Embora ainda sejam utilizadas, estas plataformas pecam pela falta de qualidade de serviço necessárias para serem utilizadas num mundo tecnológico cada vez mais exigente. Entre essas falhas encontram-se a insegurança no envio de mensagens e da fiabilidade de entrega das mesmas.

Para colmatar essas falhas surge uma segunda geração de tecnologias, constituída por extensões que podem ser adicionadas aos serviços *web* existentes. Estas tecnologias são também conhecidas por WS-* e entre elas encontram-se, por exemplo, o WS-Policy e o WS-Security.

Os tipos de serviços *web* mais utilizados são:

- SOAP, que consiste num protocolo de trocas de mensagens baseado em XML e que é possível de utilizar sobre outros protocolos. Entre esses protocolos encontram-se HTTP e o SMTP. SOAP utiliza uma elevada descrição nos seus serviços, devido à utilização de XML nas suas mensagens. Embora facilite a sua interpretação e correta utilização, acrescenta um *overhead* às mensagens que por vezes torna esses serviços lentos e dispendiosos;

- *Representational State Transfer* (REST), são um conjunto de boas práticas que definem como o protocolo HTTP e respetivos URIs devem ser utilizados. Os serviços são disponibilizados pelos URIs e podem ser utilizados a partir dos comandos POST, PUT, GET, DELETE que definem qual a ação que o serviço vai utilizar.

Entre estes dois sistemas, cada um deles tem as suas vantagens e desvantagens. Para este projeto vão ser utilizados serviços *web* baseados em REST, pois como se trata de transmitir dados entre uma aplicação móvel e um serviço de *backoffice* é imperativo que as mensagens não sejam muito "pesadas".

4.3 Segurança de dados

A segurança dos dados dos utilizadores é um dos fatores de maior importância em qualquer aplicação. Deste modo e com o objetivo de proteger esses dados foi analisada a utilização de funções de Hash.

As funções de Hash têm como objetivo mapear dados de comprimento variável em dados de comprimento fixo, originando um conjunto de bits também denominado de Hash. As funções de Hash são unilaterais o que significa que é impossível descobrir o conteúdo original a partir do Hash gerado. Deste modo as funções de Hash tornam-se assim bons mecanismos para proteger as palavras-chave dos utilizadores pois é necessário proteger os dados dos utilizadores de modo a que seja impossível descodificar o conteúdo e ao mesmo tempo ser possível comprar a palavra-chave para fins de autenticação. Embora as funções de Hash possibilitem toda esta segurança existem algumas maneiras de conseguir descodificar um Hash. O método mais simples de descodificar um *Hash* é tentar adivinhar a palavra-chave existindo para esse efeito dois métodos distintos, ataques por dicionário e ataque de força bruta.

Um ataque por dicionário consiste em utilizar um ficheiro (dicionário) que contém palavras-chave mais comuns bem como palavras ou frases que normalmente são utilizadas como palavra-chave. O objetivo é converter essas palavras-chave em *Hashs* utilizando uma função de Hash. Após converter a palavra-chave esta é comparada com o *Hash* que queremos descodificar. Se existir uma correspondência entre os dois *Hashs* ficamos a saber a palavra que originou o *Hash*.

Um ataque de força bruta consiste em tentar todas a possibilidades de carateres até um determinado comprimento. Embora um ataque de força bruta encontre sempre a palavra-chave, na maioria dos casos é um processo que consome muito tempo e que necessita de um maior poder de computação, o que o torna um processo pouco eficiente para a descodificação de *Hashs*. Não existe um método para impedir os ataques por dicionário ou por força bruta, existem, apenas maneiras de os tornar menos eficientes de modo a dissuadir o atacante.

Um método mais eficiente de descodificar *Hashs* do mesmo tipo consiste em utilizar tabelas de pesquisa, também conhecidas por *Lookup Tables*. Essas tabelas contêm as palavras-chave existente num dicionário, seguidas dos *Hashs* pré-computados dessas mesmas palavras. Deste modo e ao contrário dos ataques por dicionário, as palavras já foram previamente transformadas em *Hash*, reduzindo assim o tempo e o poder de computação necessário. Existe ainda uma versão melhorada de *Lookup Tables* denominada de *Rainbow tables*.

As *Lookup Tables* e as *Rainbow Tables* funcionam porque as palavras-chave são todas geradas da mesma maneira, isto é, uma palavra utilizando o mesmo algoritmo de *Hash* gera sempre o mesmo *Hash*. Deste modo dois utilizadores com a mesma palavra-chave vão ter o mesmo *Hash* o que reduz ao atacante o tempo necessário para as descodificar.

Como o objetivo de introduzir um componente de modo a transformar palavras-chaves idênticas em *Hashs* diferentes surge o *salt*. O *salt* consiste em introduzir um conjunto de carateres aleatórios à palavra-chave do utilizador de modo a que esta, utilizando um algoritmo de *Hash*, produza um *Hash* diferente. Este componente aleatório faz com que as *Lookup Tables* e as *Rainbow Tables* deixem de poder ser utilizadas, pois torna-se impossível pré-computar os *Hashs* tornando estes métodos ineficazes. Um dos erros mais comuns é utilizar o mesmo *salt* para todas as palavras-chave o que irá originar, como referido anteriormente, que palavras-chaves idênticas vão possuir o mesmo *Hash*. De modo a criar um *salt* com maior segurança pode ser utilizado um *salt* do tamanho do próprio *Hash*, isto é, se por exemplo a função de *Hash* que é utilizada é *SHA256* (256 bits) convêm que o *salt* tenha no mínimo também 256 bits.

É necessário ter ainda em consideração que algoritmo usar para gerar aleatoriamente o *salt*. Entre as várias linguagens de programação existem diferentes algoritmos para o efeito, no entanto, existe um que garante maior segurança chamado CSPRNG (*Cryptographically Secure Pseudo-Random Number Generator*). O facto de ser desenhado para ser criptograficamente seguro oferece uma maior garantia de gerar aleatoriamente e de modo imprevisível números, tornando-se numa mais-valia para a segurança das palavras-chave.

Embora a utilização de um *salt* previna os ataques referidos anteriormente, não previne ataques de força bruta de modo a decifrar uma palavra-chave individualmente. Para tornar este tipo de ataques menos eficientes pode utilizar-se uma técnica denominada de *key stretching*. A ideia por detrás desta técnica consiste em tornar os algoritmos de *Hash* mais lentos de modo a que mesmo com um computador com bom poder de processamento não tenha a performance desejada. No entanto, é necessário ter em consideração que este processo tem de permanecer rápido o suficiente para que o utilizador normal não note o atraso. Para este projeto será utilizado o algoritmo PBKDF2 (*Password-Based Key Derivation Function 2*).

4.4 Estudo do modo HCE em Android

A arquitetura HCE existente no SO Android é baseada em serviços. Um serviço é um componente que permite executar operações em *background*, sem que seja necessário existir uma interface visível para o utilizador. Deste modo uma aplicação pode iniciar um serviço e este continuará a ser executado mesmo que seja iniciada outra aplicação. Uma das particularidades do serviço HCE é que este só está disponível caso o ecrã do *smartphone* se encontre ligado, no caso de ser desligado, o SO Android desativa o controlador NFC, o que consequentemente indisponibiliza os serviços que utilizam NFC.

Como é possível existir mais do que um serviço a ser executado em simultâneo é necessário saber qual dos serviços é o indicado. Deste modo é utilizado o AID (*Application ID*), que permite associar um identificador a um serviço criado por uma aplicação. Quando se desenvolve uma aplicação HCE para um leitor NFC já existente deve ser utilizado o AID do leitor, que normalmente é público e bem conhecido, como por exemplo o AID dos leitores de pagamento Visa. No caso de ser desenvolvido um leitor próprio para a aplicação HCE, este deve possuir o seu próprio AID. Este registo necessita de ser definido segundo a norma ISO/IEC 7816-5 (ISO/IEC, 2004) de modo a garantir que não existem colisões com outras aplicações.

Deste modo, e segundo a norma ISO/IEC 7816-5, um AID é constituído por um *Registered application provider IDentifier* (RID) e caso necessário por um *Proprietary application Identifier eXtension* (PIX).

O RID é um identificador composto por 5 bytes que identifica o fornecedor da aplicação, apenas sendo permitido existir um RID por cada fornecedor. O PIX é composto no máximo 11 bytes e é utilizado para diferenciar aplicações de um mesmo fornecedor, não sendo no entanto obrigatório para a construção do AID. Como exemplo, foi analisado o cartão Visa que tem o RID A000000003 para registar o fornecedor, no entanto como o cartão Visa pode ser associado a diferentes tipos de produtos (*V PAY, Visa Electron, etc)* necessita de utilizar o PIX para diferenciar esses produtos. No caso do *Visa Electron* o PIX é o 2010, criando assim o AID para este tipo de catões com o valor de A0000000032010.

Os primeiros 4 bits do RID tem como objetivo especificar a categoria de registo, por exemplo no caso anterior, cartão Visa, os primeiros 4 bits são referentes à letra 'A', esta significa que o RID tem um registo Internacional. As outras categorias de registo podem ser visualizadas na Tabela 1.

Tabela 1 – Valores categorias de Registo

Código do Registo	Categoria de registo
'0'-'9'	Definido na norma ISO/IEC 7812
'A'	Registo Internacional
'B'	Reservado para ISO
'C'	Reservado para ISO
'D'	Registo Nacional
'E'	Reservado para ISO
'F'	Proprietário não registado

Um serviço HCE pode necessitar de ter associados vários AIDs, por exemplo um serviço de cartão de multibanco pode necessitar de comunicar com leitores de diferentes fabricantes. Para resolver esta particularidade o sistema de HCE permite associar vários AIDs a um serviço, através da utilização de grupos. Estes grupos garantem que todos os AIDs que o constituem são vistos como um só pelo SO, sendo todos os AIDs redirecionados para o mesmo serviço. Atualmente os grupos de AIDs podem ser associados a categorias, o que permite ao SO Android agrupar os serviços por categorias e por conseguinte permitir ao utilizador predefinir que

categorias utilizar. Esta camada de abstração permite ao utilizador selecionar com maior facilidade o tipo de serviço que quer utilizar, em vez de lhe serem apresentados um conjunto de AIDs sem qualquer significado.

O Android 4.4 suporta apenas duas categorias, uma categoria que cobre o conjunto das aplicações de pagamentos (*category_payment*) e uma categoria que engloba todas as outras aplicações (*category_other*). Relativamente à categoria de pagamentos, somente pode estar ativo um grupo de AIDs, sendo que desta forma convém que a aplicação que o utilizador usa, possua a maioria dos protocolos dos cartões de pagamento.

4.4.1 Implementação do Serviço

Neste subcapítulo será descrito o processo da criação de um serviço HCE, utilizando o exemplo disponibilizado pela plataforma de desenvolvimento Android.

A aplicação HCE analisada consiste em utilizar o *smartphone* como um cartão de fidelidade. Para tal, o utilizador apenas necessita de introduzir o número do cartão, como se pode visualizar na Figura 17, sendo este número posteriormente lido pelo leitor NFC que será posteriormente descrito.

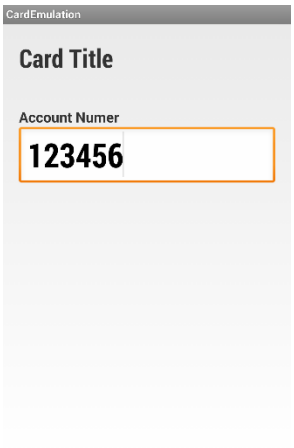


Figura 17 – Aplicação HCE

Para tal, é necessário criar o serviço que permitirá interagir com o leitor NFC. O Android 4.4 possui uma classe que permite implementar, facilmente, o serviço de HCE. Essa classe tem o nome de *HostAdpuService* e para desenvolver um serviço de HCE apenas é necessário estender essa classe. A classe *HostAdpuService* declara dois métodos abstratos que necessitam de ser implementados, o *processComandApdu* e o *onDeactivated*.

O método *processComandApdu* é invocado sempre que o leitor NFC envia um APDU (*Application Protocol Data Unit*) para o serviço. Os APDUs são os pacotes utilizados para a troca de dados entre o serviço de HCE e o leitor NFC. A troca desses dados é realizada num canal *half-*

duplex, isto é, o leitor NFC envia um comando APDU e fica em espera até que o serviço HCE envie uma resposta.

De acordo com a norma ISO/IEC 7816-5 existem dois tipos de APDUs: *Command* APDU e *Response* APDU. Os *Commnad* APDUs são enviados pelo leitor para o cartão e contém obrigatoriamente um cabeçalho de 4 bytes, podendo ser opcionalmente enviados dados com um máximo de 255 bytes. Os Response APDUs são enviados do cartão para o leitor e contém obrigatoriamente dois *status* bytes, os dados também são opcionais e podem conter no máximo 64kb.

O *Command* APDU consiste numa estrutura idêntica à apresentada na Tabela 2.

Tabela 2 – Estrutura Command APDU

Body	**Trailer**
CLA INS P1 P2	Lc Data Le

- CLA - consiste na classe da instrução;
- INS - consiste no código da instrução;
- P1-P2 - consistem nos parâmetros da instrução;
- Lc - consiste no número de bytes presente nos dados da instrução;
- Data - consiste no conjunto de bytes de dados existentes na instrução;
- Le - consiste no número máximo de bytes esperado pela resposta à instrução.

Como é possível observar na Tabela 2 o cabeçalho APDU é constituído por CLA,INS,P1,P2 e para poder comunicar com o serviço HCE é necessário que o cabeçalho seja sempre o seguinte:

- CLA = 00;
- INS = A4;
- P1 = 04;
- P2 = 00.

O Response APDU consiste na estrutura da Tabela 3.

Tabela 3 – Estrutura Response APDU

Body	**Trailer**
Data	SW1 SW2

- Data consiste em dados que podem ser enviados na resposta;
- SW1-SW2 consistem no *status word* resultante do comando APDU e são obrigatórios em cada resposta do serviço HCE. Por convenção o *status word* é em hexadecimal, sendo que, por exemplo o resultado OK é indicado por 9000 (SW1 = 90 e SW2 = 00).

Como referido anteriormente quem inicia a comunicação é o leitor NFC, que quando deteta a presença de uma *tag*, envia deste modo um APDU com o AID do serviço para o qual quer comunicar. Este APDU é posteriormente processado pelo SO Android, sendo redirecionado para serviço correto caso este esteja disponível. Em modo HCE o *smartphone* é considerado como sendo uma *tag*, permitindo assim que o leitor NFC leia o dispositivo como uma outra qualquer *tag*, como por exemplo um *smartcard*.

Caso o tempo de computação do serviço HCE seja elevado, a resposta do método *processComandApdu* pode ser enviada de imediato, podendo posteriormente ser utilizado o método *sendResponseApdu* para enviar a resposta para o leitor NFC. No caso da comunicação NFC ser interrompida ou no caso de o leitor NFC enviar um pedido APDU com um AID diferente, e em que o SO decide utilizar outro serviço, o método onDeactivated é invocado sendo enviado como parâmetro a razão da interrupção.

O Código 1 apresenta o método *processCommandApdu* implementado pela aplicação HCE.

```
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
      if (Arrays.equals(SELECT_APDU, commandApdu)) {
            String account = AccountStorage.GetAccount(this);
            byte[] accountBytes = account.getBytes();
            return ConcatArrays(accountBytes, SELECT_OK_SW);
       }else {
            return UNKNOWN_CMD_SW;
       }
   }
```

Código 1 - Método *processCommandApdu*

Como é possível visualizar, são comparados dois *arrays* de bytes de modo a verificar, das opções existentes, qual delas irá processar o APDU recebido. Neste caso, só existem duas opções, ou o APDU coincide com a variável SELECT_APDU e são enviados para o leitor NFC as informações do cartão ou então retorna para o leitor NFC somente o *status word* que indica que não conseguiu processar o APDU recebido. A variável *account* contém o número do cartão introduzido pelo utilizador, sendo posteriormente criado um *array* de bytes com essa variável, a variável *accountBytes*. Por fim, para enviar os dados para o leitor NFC é necessário introduzir o *status word,* representado pela variável SELECT_OK_SW. Assim, é essencial concatenar o *array* com os dados e o *array* com o *status word*, utilizando para o efeito o método *ConcatArrays*.

4.4.2 Declaração do serviço no Android Manifest

Como todos os serviços, este também tem de estar declarado no *manifest* da aplicação. Para o serviço ser do tipo HCE é necessário que contenha a permissão *BIND_NFC_SERVICE*. Esta permissão é um requisito do sistema e garante que somente o SO Android se pode conectar ao serviço. O atributo *exported*, permite especificar se o serviço pode ser invocado por outras

aplicações. No caso desta, aplicação este atributo tem de ser verdadeiro, pois é necessário que o leitor NFC possa invocar o serviço HCE.

O serviço necessita ainda de filtrar o tipo de objetos para os quais pode ser utilizado, assim contém um *intent-filter* para filtrar objetos (Intent) do tipo *android.nfc.cardemulation.action.HOST_APDU_SERVICE.*

A declaração do serviço contém ainda a utilização de um ficheiro do tipo XML que inclui uma lista dos AIDs associados a esta aplicação. Para além destas permissões específicas do serviço, é necessário ainda que exista no *manifest* a indicação de qua a aplicação precisa de utilizar a tecnologia NFC. A declaração do serviço HCE utilizado pela aplicação pode ser visualizada no Código 2**Erro! A origem da referência não foi encontrada.**.

```
<service android:name=".CardService"
         android:exported="true"
         android:permission="android.permission.BIND_NFC_SERVICE">
  <intent-filter>
  <action android:name="android.nfc.cardemulation.action.HOST_APDU_SERVICE"/>
   </intent-filter>
  <meta-data android:name="android.nfc.cardemulation.host_apdu_service"
                       android:resource="@xml/aid_list"/>
</service>
```

Código 2 - Declaração do serviço HCE no *Android manifest*

4.5 Estudo de Aplicação Leitor NFC em Android

A aplicação leitor NFC tem como objetivo ler o cartão emulado pela aplicação HCE e a sua interface pode ser visualizada pela Figura 18.

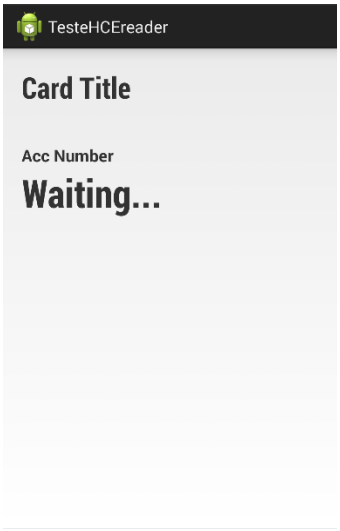


Figura 18 – Aplicação leitor NFC

Para o efeito, a aplicação utiliza uma classe que implementa a interface *NfcAdapter.ReaderCallback*. Esta interface consiste num *callback* que é invocado quando uma *tag* é descoberta, enquanto a *Activity* que está em primeiro plano se encontra em modo de leitura. Este modo de leitura, implementado através da utilização do método *enableReaderMode*, limita a utilização do controlador NFC para que seja somente utilizado para modo de leitura/escrita, desativando o modo P2P (*Android Beam*) e o modo de emulação de cartão.

Para utilizar a interface referida anteriormente é necessário implementar o método *onTagDiscovered*. Este método é invocado sempre que uma *tag* é descoberta e a sua implementação é possível visualizar no Código 3.

```
public void onTagDiscovered(Tag tag) {
     IsoDep isoDep = IsoDep.get(tag);
     if (isoDep != null) {
         try {
             isoDep.connect();
             byte[] command = BuildSelectApdu(LOYALTY_CARD_AID);
             byte[] result = isoDep.transceive(command);
             int resultLength = result.length;
             byte[] statusWord = {result[resultLength-2],
result[resultLength-1]};
             byte[] payload = Arrays.copyOf(result, resultLength-2);
             if (Arrays.equals(SELECT_OK_SW, statusWord)) {
                 String accountNumber = new String(payload, "UTF-8");
                 mAccountCallback.get().onAccountReceived(accountNumber);
             }
         } catch (IOException e) {
                Log.e(TAG, "Error communicating with card: " + e.toString());
         }
     }
 }
```

Código 3 - Método onTagDiscovered

Como o modo HCE implementa o protocolo *IsoDep* (ISO/IEC 14443), e de modo a comunicar com o *smartphone* que contém esse serviço, é necessário que a *tag* descoberta seja processada utilizando a classe *IsoDep*. Assim é criado um objeto do tipo *IsoDep* com a *tag* descoberta, sendo estabelecida a ligação entre o leitor e o serviço HCE através do método *connect* do objeto *IsoDep*.

De seguida é criado um *array* de bytes que contém o APDU que vai ser enviado para o serviço HCE. Este APDU contém o AID da aplicação para o qual o leitor NFC quer iniciar a comunicação. Após a criação do comando APDU é utilizado o método *transceive* do objeto *IsoDep* para enviar o APDU para a aplicação HCE.

A variável *result* contém a resposta do serviço HCE ao APDU enviado e é a partir dessa variável que se irá obter o *status word* de modo a saber se o APDU enviado foi bem processado.

O *status word* obtido é comparado com *SELECT_OK_SW,* que é uma variável composta por um *array* de bytes com o código OK, e caso sejam iguais os dados são convertidos de *array* de bytes para uma variável do tipo *string*. Esta variável, com o nome *accountNumber*, vai deste modo conter o número do cartão emulado pela aplicação HCE. O método *onAccountReceived* tem como objetivo enviar para outra classe o número do cartão, para que este possa ser exibido ao utilizador. Na Figura 19 é possível observar o valor do cartão lido pelo leitor NFC.

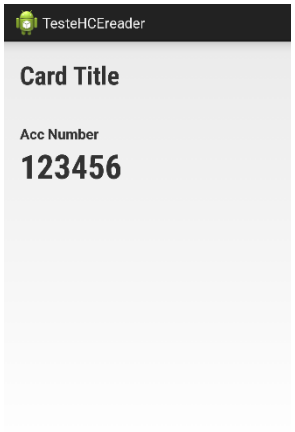


Figura 19 – Aplicação leitor NFC após ler catão

Para uma melhor compreensão da troca de dados e o funcionamento entre o leitor NFC e a aplicação HCE é apresentado na Figura 20 um fluxograma das duas aplicações. De relembrar que o *smartphone* em modo HCE é visto pelo leitor NFC como sendo uma *tag.*

4.5.1 Filtros

Quando um dispositivo em modo de leitura lê uma *tag* e não consegue mapear os dados obtidos tenta iniciar uma aplicação que contenha o filtro *android.nfc.action.TECH_DISCOVERED*. Este filtro permite ainda que lhe sejam agregados os tipos de *tags* que a aplicação pode ler, por exemplo é possível que a aplicação seja somente utilizada para ler *tags* do tipo A (NfcA). Para tal é necessário criar um ficheiro XML que contém o tipo de *tag* que a aplicação vai ler e associá-lo à aplicação.

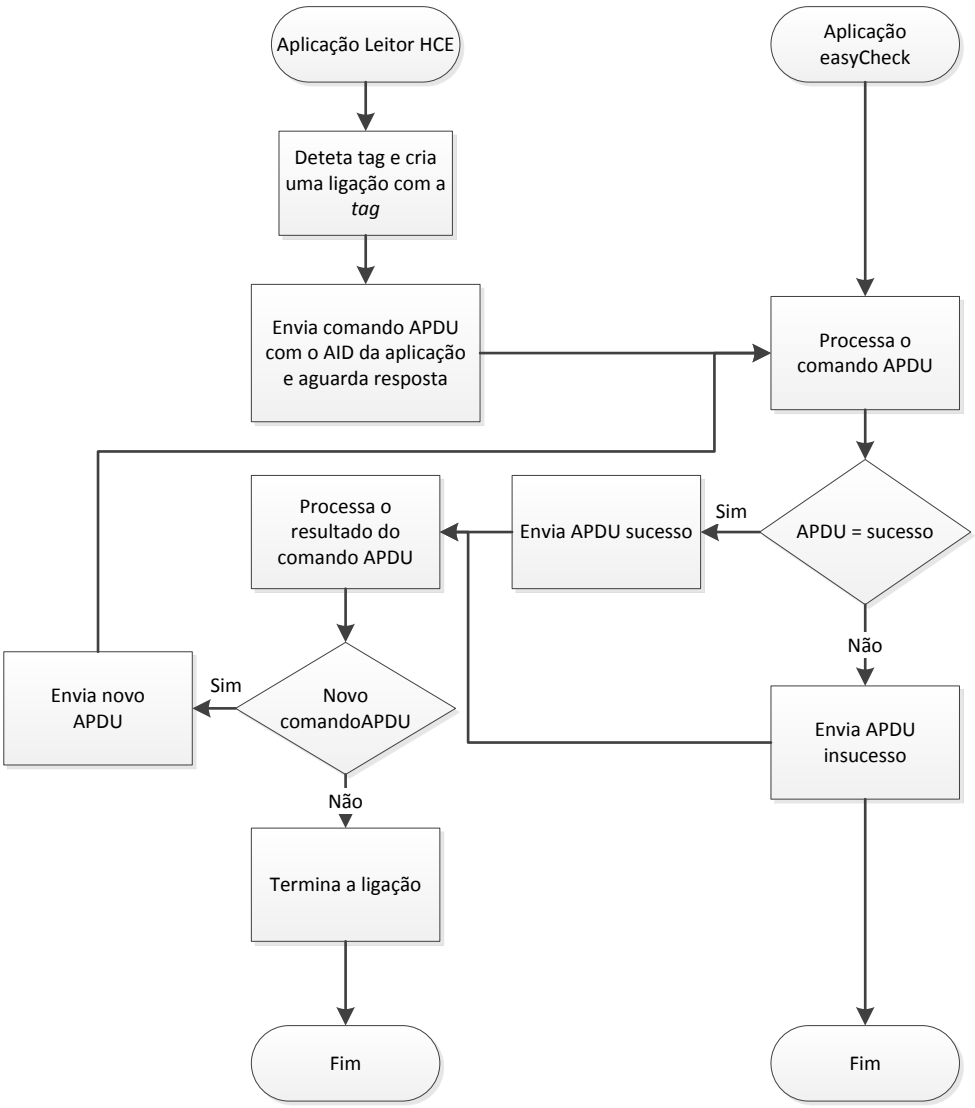


Figura 20 – Funcionamento da tecnologia NFC em modo de HCE

5 Implementação

Neste capítulo será descrito o processo de desenvolvimento da aplicação Android proposta nos objetivos, bem como o suporte necessário para que essa aplicação possa funcionar corretamente. Após desenvolvida esta aplicação foi necessário testar a sua componente NFC. Devido à impossibilidade de testar as funcionalidades NFC num terminal NFC foi desenvolvida uma outra aplicação para testar essas mesmas funcionalidades. Deste modo, esta segunda aplicação, irá servir para testar os processos de *check-in*, *check-out* e de acesso ao quarto. Ambas as aplicações encontram-se ligadas a um *backoffice* que faz a gestão do sistema. A arquitetura do sistema pode ser observada na Figura 21.

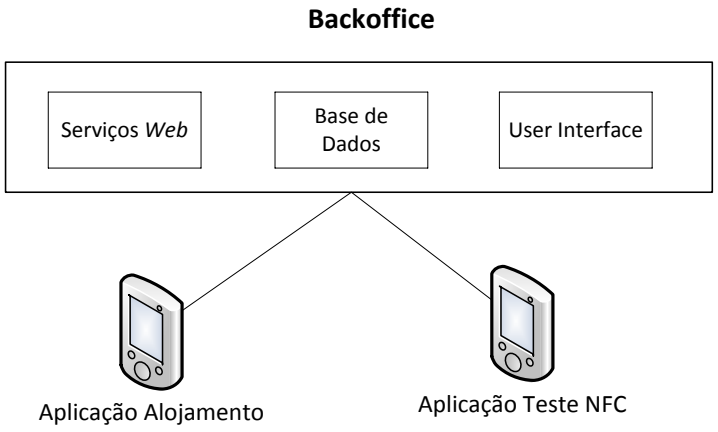


Figura 21 – Arquitetura do sistema

5.1 Aplicação easyCheck

Esta aplicação easyCheck é a aplicação que se encontra proposta nos objetivos deste projeto e que pretende disponibilizar para o utilizador algumas das funcionalidades existentes num local de alojamento. Entre essas funcionalidades encontram-se a possibilidade de um utilizador poder fazer reservas, consultar as suas reservas, aceder ao quarto reservado entre outras. De seguida serão expostas com maior detalhe todas as suas funcionalidades sendo possível visualizar na Figura 22 o diagrama de fluxo da aplicação.

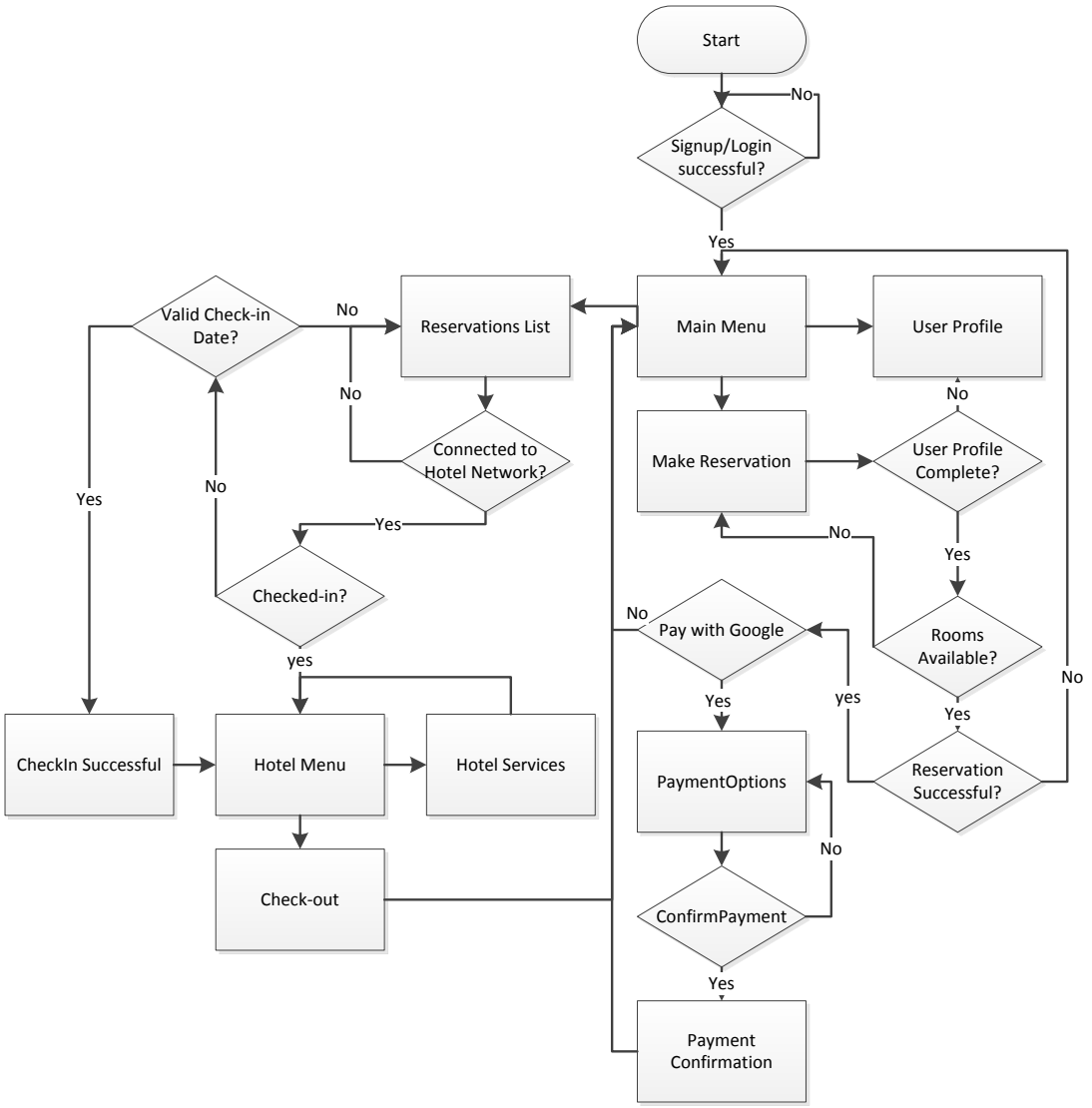


Figura 22 – Diagrama de fluxo da aplicação easyCheck

A aplicação começa inicialmente por apresentar um formulário para que o utilizador se possa registar. Este processo de registo é essencial para posteriormente identificar as reservas feitas por um utilizador. No caso de o utilizador já se encontrar registado, somente necessita introduzir o seu *email* e respetiva palavra-chave de modo a ter acesso à aplicação. De notar que o mesmo formulário é utilizado tanto para fazer o registo como para autenticação. A decisão

de juntar estas duas funções numa só, parte do princípio de que uma aplicação móvel deve ser de fácil e rápida utilização.

Após entrar na aplicação com sucesso, as informações relativamente à autenticação ficam guardadas na aplicação e o formulário de registo/autenticação deixa de aparecer quando a aplicação é aberta. O modo de armazenamento dos dados de autenticação foi implementado utilizando a classe *SharedPreferences* e encontram-se descritas no capítulo 5.1.4. O formulário de registo/autenticação pode ser observado na Figura 23.

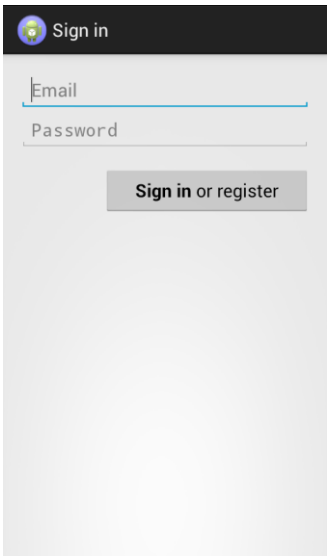


Figura 23 – Registo/Autenticação

Após realizar com sucesso o processo de registo / autenticação é apresentado o menu principal da aplicação. Neste menu é possível navegar para as principais funcionalidades da aplicação, tais como: fazer reservas, consultar as reservas previamente marcadas, e ainda alterar as informações pessoais do utilizador. Neste menu principal, pressionado o botão de menu do *smartphone*, é apresentada a opção para o utilizador fazer *logout* da aplicação. Ao realizar esta funcionalidade o utilizador limpa as suas informações de autenticação e é apresentado novamente o formulário de registo/autenticação. O menu principal da aplicação pode ser observado na Figura 24.

Após aceder ao menu principal o utilizador deve completar o seu perfil. Para o efeito necessita de selecionar a opção “User Profile”, sendo apresentado um formulário para o utilizador preencher. Este processo é fundamental para o utilizador obter um perfil válido, ficando assim apto a fazer reservas. O formulário com o perfil do utilizador pode ser observado na Figura 25.

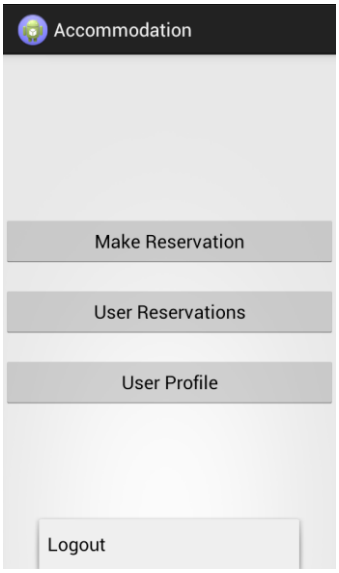


Figura 24 – Menu Principal

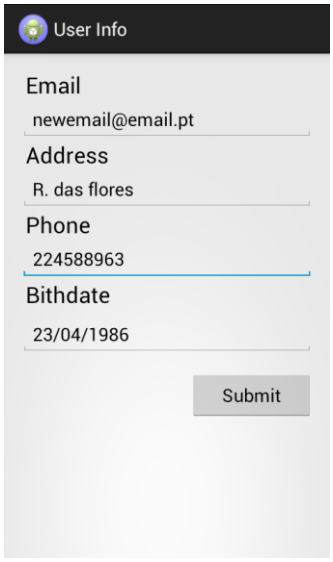
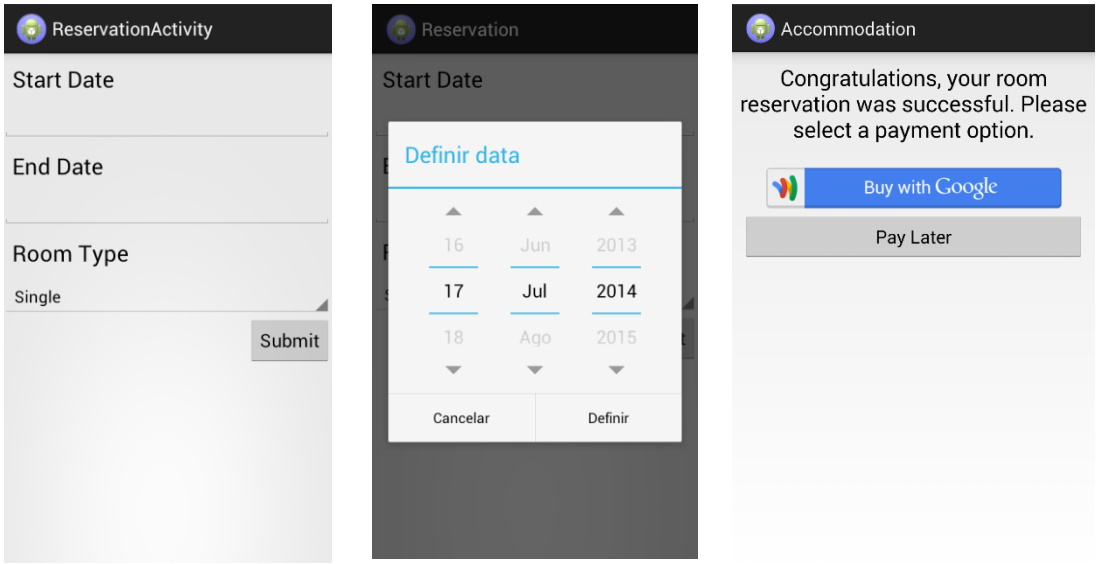


Figura 25 – Perfil Utilizador

Uma das funcionalidades fundamentais da aplicação é a possibilidade do utilizador poder fazer reservas. Para tal o utilizador necessita de selecionar a opção “Make Reservation”, disponível no menu principal. De seguida é apresentado um formulário como é possível observar na Figura 26 a). Neste formulário pode definir-se: a data do início e data do fim da estadia, bem como o tipo de quarto pretendido. De modo a impedir que sejam feitas reservas para dias anteriores ao dia atual são feitas validações no ato de submeter a reserva. Quando se pressiona o campo para introduzir uma data é apresentado um calendário para facilitar ao utilizador a introdução de uma data. Este calendário pode ser visualizado na Figura 26 b).

No ato de submeter a reserva, e no caso de o utilizador não possuir as suas informações pessoais completas, a aplicação apresenta o formulário existente no “User Profile” para que o

utilizador possa completar as suas informações pessoais. Somente com um perfil válido é que o utilizador poderá fazer uma reserva. No fim de concluir o processo de reserva é apresentado ao utilizador a possibilidade de fazer o pagamento da reserva, como se pode observar na Figura 26 c). O utilizador pode então optar por fazer o pagamento utilizando a sua conta Google Wallet, pressionado o botão "Buy with Google". No caso de não possuir uma conta Google Wallet ou não queira optar por este modo de pagamento pode selecionar a opção "Pay Later", de modo a realizar o pagamento da reserva presencialmente no local de alojamento. O processo de pagamento utilizando o Google Wallet será descrito na secção 5.1.5.



a) b) c)

Figura 26 a) Formulário Criar Reserva, b) Calendário, c) Pagamento

Outra das funcionalidades é a possibilidade de consultar as reservas previamente feitas. Para tal o utilizador necessita de selecionar a opção "User Reservations" presente no menu principal.

Seguidamente é apresentada uma lista com as reservas do utilizador, como se pode observar na Figura 27, e é a partir desta lista de reservas que o utilizador procede ao *check-in*. De salientar que o utilizador só pode fazer *check-in* de reservas para o próprio dia.

A aplicação disponibiliza dois processos para o utilizador realizar o *check-in*, um consiste em fazer o *check-in* utilizando para isso a rede Wi-Fi do local de alojamento, o outro processo consiste em utilizar a tecnologia NFC.

No método de *check-in* por Wi-Fi, e para garantir uma maior segurança, somente é possível fazer o processo de *check-in* quando o *smartphone* estiver conectado ao Wi-Fi do local de alojamento. Caso não esteja conectado ao Wi-Fi do local de alojamento é lhe apresentada uma mensagem informativa, como se pode visualizar na Figura 27. De modo a verificar se o utilizador está conectado a uma rede Wi-Fi válida, a aplicação requisita ao *backoffice* uma lista com os identificadores dos pontos de acesso, procedendo depois a verificar se o utilizador se encontra ligado a algum deles. O método de *check-in* utilizando a tecnologia NFC consiste em

colocar o *smartphone* em contacto com um terminal existente, por exemplo, na receção do local de alojamento. Ambos os métodos de *check-in* estão disponíveis quando é apresentada a lista com as reservas do utilizador, sendo que, para utilizar o *check-in* por Wi-Fi o utilizador necessita de pressionar a reserva indicada, enquanto que para utilizar o *check-in* por NFC, o utilizador tem de pressionar a reserva durante mais alguns segundos. O processo de *check-in* por NFC será explicada na secção 5.1.1 deste relatório. Como medida de segurança associa-se o identificador do *smartphone* durante o processo de *check-in,* impossibilitando que após este processo seja utilizado outro dispositivo para aceder ao quarto. O *check-in* por NFC, para além do identificador do *smartphone* associa o identificador da *tag* criando assim mais uma camada de segurança.

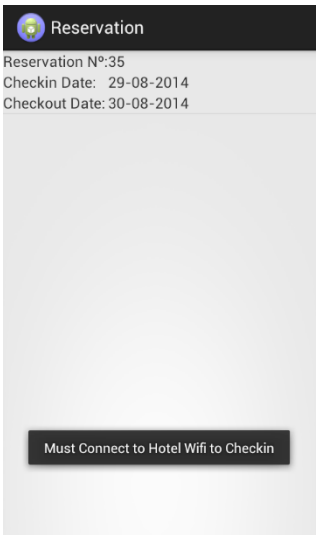


Figura 27 – Lista de Reservas

Após o sucesso no processo de *check-in* a aplicação apresenta o menu do local de alojamento. Nesse menu o utilizador pode consultar o número do seu quarto, aceder aos serviços internos disponibilizados pelo local de alojamento e proceder ao *check-out*, como é observável na Figura 28. O processo de *check-out*, tal como o processo de *check-in,* é possível de ser realizado de dois modos, por Wi-Fi e por NFC. A diferença entre o processo de *check-in* e processo de *check-out* consiste na utilização de serviços *web* diferentes e ainda o terminal NFC utilizado, tem de ser diferente. O processo de *check-out* por NFC será posteriormente explicado na secção 5.2.3 deste relatório.

Os processos de *check-in* e *check-out* por NFC foram implementados utilizando a vertente de HCE. Para além dos processos anteriores foi também utilizada esta tecnologia para implementar o acesso ao quarto, encontrando-se este processo descrito na secção 5.2.4 deste documento. A implementação dos processos que utilizam a vertente HCE necessita que estes possuam um AID, deste modo foram utilizados os AIDs existentes na

Tabela 4.

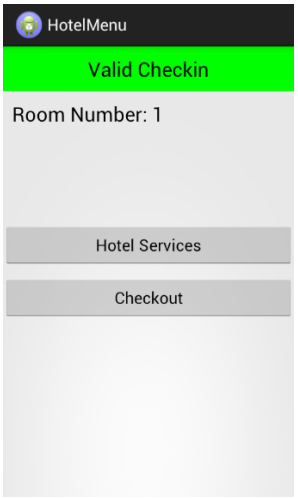


Figura 28 – Menu do local de alojamento

Tabela 4 – AIDs serviços HCE

Serviço	AID
Acesso ao quarto	F2222222210
Check-in	F2222222220
Check-out	F2222222230

5.1.1 Check-in

O processo de *check-in*, como referido anteriormente, pode ser realizado de duas formas distintas. A primeira consiste em estar conectado à rede Wi-Fi do local de alojamento e fazer o *check-in* carregando na respetiva opção na aplicação, tendo este processo como principal vantagem não precisar de interagir com mais nenhum periférico.

A segunda consiste em colocar o *smartphone* em contacto com um terminal de *check-in*. Por exemplo, pode existir um terminal de *check-in* na receção de modo a que, para além do *check-in* e caso necessário, proceder a alguma alteração na reserva.

De modo a implementar este modo de *check-in* foi desenvolvido um serviço HCE, que como referido anteriormente, necessita de estender a interface *HostApduService* e implementar os seus respetivos métodos. Este serviço tem de ter obrigatoriamente associado um AID, sendo utilizado então o AID F2222222210, como é possível verificar pela

Tabela 4.

Relativamente ao método *processCommandApdu*, o primeiro procedimento quando o serviço recebe o comando APDU é verificar se o APDU que recebeu tem o AID do serviço. No caso de serem iguais são enviados para o leitor NFC as informações necessárias para realizar o *check-in*, neste caso: o identificador da reserva, o identificador do *smartphone* e o identificador

da *tag*. Estes dados são posteriormente concatenados juntamente com o *status word*, que indica ao leitor NFC que o APDU enviado, foi processado com sucesso.

Após estes dados serem enviados e devidamente processados pelo leitor NFC, este envia para a aplicação um comando APDU com o número do quarto e a sua respetiva chave. Este APDU é então processado pela aplicação easyCheck sendo a chave do quarto armazenada nas preferências da aplicação. O armazenamento da chave recebida vem da necessidade de ser necessário utilizar o *smartphone* para ter o acesso ao quarto. Após armazenar a chave é iniciada a *Activity AccommodationMenu*. A implementação do método *processCommandApdu* pode ser observada no Anexo A e na Figura 29, pode ser visualizado um diagrama de sequência do processo de *check-in,* para uma melhor compreensão do mesmo.

Ambos os processos utilizam o serviço *web* com o URI /Checkin, no primeiro método o serviço *web* é invocado pela aplicação, enquanto que no segundo método, é invocado pelo leitor NFC. Este serviço *web* encontra-se descrito mais detalhadamente na secção 5.3.2 deste documento.

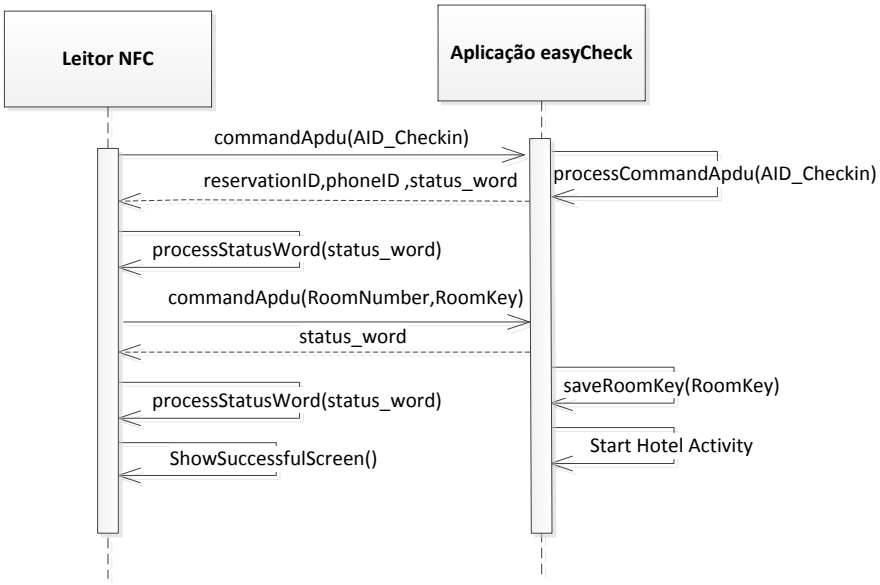


Figura 29 – Diagrama de sequência do processo de *check-in*

5.1.2 Check-out

Tal como no processo de *check-in*, o processo de *check-out* pode ser feito de dois modos distintos, por Wi-Fi ou através da interação do *smartphone* com um terminal de *check-out*. Este terminal pode estar colocado, por exemplo, junto das portas de acesso do local de alojamento, relembrando o cliente para que não se esqueça de fazer o *check-out*.

De modo a implementar este segundo método de *check-out*, e tal como para o processo de *check-in*, foi criado um serviço HCE. Este serviço tem de ter obrigatoriamente associado um AID, sendo utilizado então o AID F2222222220, como é possível verificar pela

Tabela 4. A implementação do método *processCommandApdu* pode ser observada no Anexo A. O primeiro procedimento quando o serviço recebe o comando APDU é verificar se o APDU que recebeu tem o AID do serviço. No caso de serem iguais são enviadas para o leitor NFC as informações necessárias para realizar o *check-out*, neste caso o identificador da reserva. Este identificador é posteriormente concatenado juntamente com o *status word*, que indica ao leitor NFC que o APDU enviado foi processado com sucesso. De seguida o leitor NFC procede ao *check-out*, e no caso de este ser bem sucedido envia para a aplicação um APDU que indica o sucesso da operação. O método *processCommandApdu* processa esse APDU e a aplicação sai do menu do local de alojamento para o menu Principal. Na Figura 30 pode ser visualizado um diagrama de sequência do processo de *check-out*. De realçar que quem inicia a comunicação é o leitor NFC.

Ambos os processos utilizam o serviço *web* com o URI /Checkout, no primeiro método o serviço *web* é invocado pela aplicação, enquanto que no segundo método, é invocado pelo leitor NFC. Este serviço *web* encontra-se descrito mais detalhadamente na secção 5.3.2 deste documento.

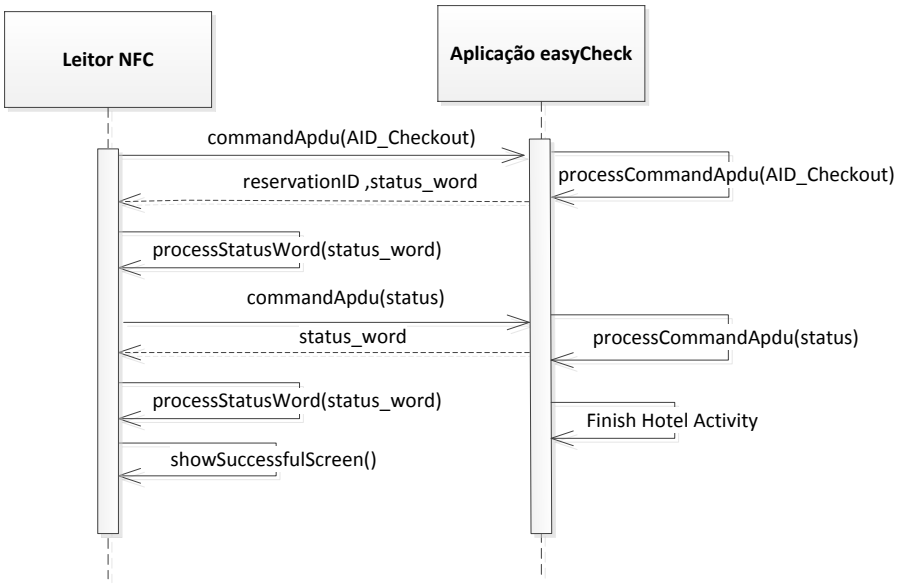


Figura 30 - Diagrama de sequência do processo de *check-out*

5.1.3 Acesso ao quarto

O processo de acesso ao quarto consiste em colocar o *smartphone* em contacto com o terminal que se encontra na porta do quarto. Ao contrário do processo de *check-in* e de *check-*

out não é possível ter acesso ao quarto através de uma funcionalidade da aplicação, isto é, somente quando o *smarthphone* é detetado pelo leitor NFC, é que é possível ter acesso ao quarto.

De modo a implementar esta funcionalidade, e tal como para os processos anteriores, foi criado um serviço HCE. Este serviço tem de ter obrigatoriamente associado um AID, sendo utilizado então o AID F2222222230, como é possível verificar pela

Tabela 4. A implementação do método *processCommandApdu* pode ser observada no Anexo A. O primeiro passo quando o serviço recebe o comando APDU, é verificar se o APDU que recebeu tem o AID do serviço. No caso de serem iguais são enviados para o leitor NFC as informações necessárias para realizar o acesso ao quarto, neste caso as chaves armazenadas no *smartphone*, o identificador do *smartphone* e o identificador da *tag*. Estes dados são posteriormente concatenados juntamente com o *status word*, que indica ao leitor NFC que o APDU enviado foi processado com sucesso. O leitor NFC não necessita de enviar mais nenhum APDU, no caso de ter sido enviada uma chave que permite o acesso ao quarto pretendido, o leitor NFC apresenta uma cor verde, caso contrário, vermelha. Na Figura 31 pode ser visualizado um diagrama de sequência do processo de acesso ao quarto.

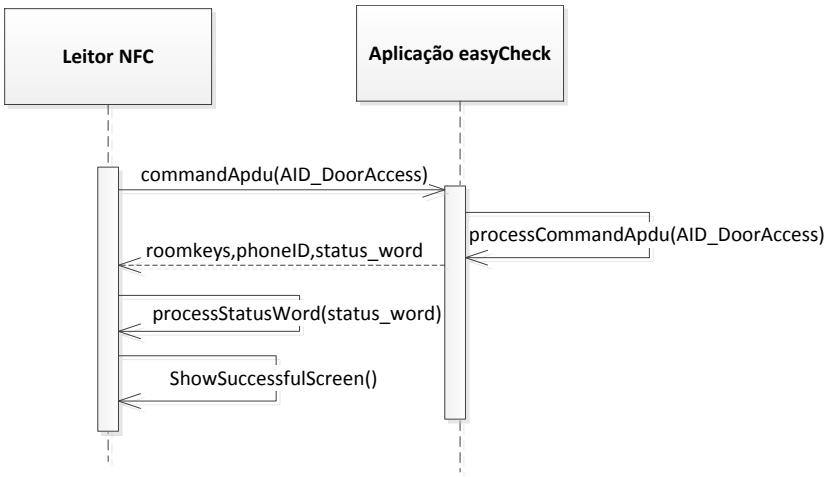


Figura 31 - Diagrama de sequência do processo de acesso ao quarto

5.1.4 Armazenamento Interno de Dados

Com o objetivo de armazenar de um modo permanente determinados dados fundamentais para o funcionamento da aplicação, foi utilizada a classe *SharedPreferences*. Esta classe consiste num conjunto de métodos que permitem armazenar e retirar dados no formato de chave-valor, sendo que os dados armazenados têm de estar num formato primitivo, isto é, *strings*, *floats*, *booleans*, etc. Relativamente à segurança dos dados armazenados através desta classe, é necessário referir, que cria um ficheiro e guarda-o no diretório da aplicação. Embora esteja guardado no sistema de armazenamento do *smartphone*, não pode ser lido por outra aplicação, apenas pode ser lido/escrito pela aplicação com o UID (*Unique user ID*) que o criou.

No entanto, como o SO Android é baseado no sistema UNIX, um utilizador com *root access* tem acesso a todos os ficheiros existente no sistema podendo deste modo aceder a este ficheiro. Para salvaguardar os utilizadores desta exposição aos seus dados pessoais, a aplicação encripta os dados mais sensíveis, neste caso a palavra-chave, com o algoritmo SHA-256, dificultando assim o trabalho ao atacante.

Entre os dados guardados encontram-se:

- *Email* e palavra-chave para poder ser efetuado o processo de *login* com maior rapidez;
- Identificador do utilizador, userID, porque é necessário em vários serviços *web* e desta forma não é necessário estar sempre a sobrecarregar o sistema com o pedido do userID;
- Chaves dos quartos dos quais o utilizador já fez o *check-in*, pois são necessárias para o processo de acesso ao quarto.

5.1.5 Google Wallet - Instant Buy

De modo a possibilitar ao utilizador um modo de pagamento mais rápido e cómodo foi integrada na aplicação a vertente *Instant Buy* (Google Wallet, 2013) disponibilizada pelo Google Wallet. Esta permite assim o pagamento de produtos sem que seja necessário grande esforço por parte do utilizador, utilizando para isso a sua conta Google Wallet. A vertente *Instant Buy* foi selecionada de entre os modos de pagamento disponibilizados pelo Google Wallet pois permite que seja a empresa a ficar responsável por processar o pagamento, isto é, permite que toda a aplicação possa ser integrada num local de alojamento, inclusive utilizar já o sistema de pagamento existente. A não utilização de outros sistemas de pagamento retiram a complexidade à aplicação facilitando a sua integração num local de alojamento. Nesta vertente o Google Wallet não partilha a totalidade das informações do cartão de crédito do utilizador com a aplicação que o integra, em vez disso cria um Google Wallet *Virtual Onetime Card* com as informações do cartão que o utilizador selecionou para fazer o pagamento. Esta funcionalidade é apresentada ao utilizador na aplicação através da opção "Buy with Google" como é apresentado pela Figura 26 c), e quando o utilizador seleciona esse modo de pagamento, são apresentadas um conjunto de informações relativamente à sua conta Google Wallet, como é possível observar na Figura 32**Erro! A origem da referência não foi encontrada.**. Entre essas informações encontram-se o cartão predefinido para realizar pagamentos e a morada de faturação. Neste menu é possível selecionar outro cartão ou ainda outra morada de faturação, sendo possível também acrescentar informações sobre um novo cartão ou uma nova morada de faturação. Para além destas funcionalidades é ainda possível autorizar o Google Wallet para que futuros pagamentos feitos através desta aplicação sejam realizados, utilizando estas definições, deixando assim de aparecer este menu.

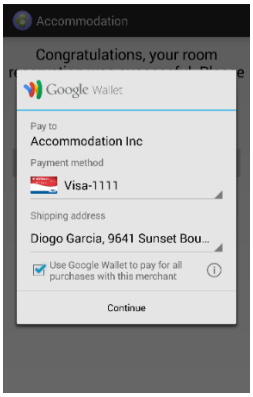


Figura 32 – Opções Pagamento

Após selecionar corretamente o cartão e a morada de faturação é apresentada uma confirmação do pedido de pagamento, isto é, a aplicação apresenta com detalhe o que irá ser debitado. Neste menu é apresentado ainda o cartão que o utilizador selecionou para realizar o pagamento bem como a morada de faturação, podendo estes dados ser alterados. Este menu de confirmação do pedido de pagamento é apresentado na Figura 33.

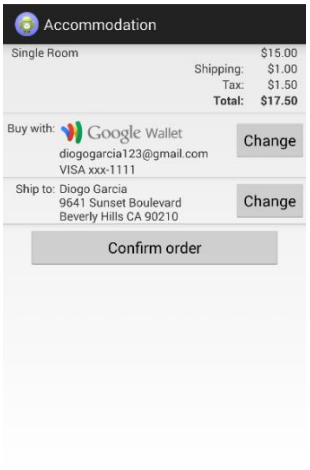


Figura 33 – Confirmação pedido pagamento

Após confirmar o pedido de pagamento a aplicação apresenta um menu que informa o utilizador que o pagamento foi feito com sucesso, como é possível observar na Figura 34.

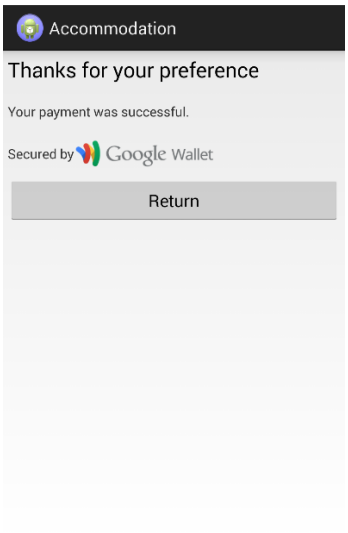


Figura 34 – Confirmação Pagamento

De modo a compreender as trocas de mensagens entre o serviço do Google Wallet e a aplicação é apresentado o digrama da Figura 35. Nesse diagrama é possível visualizar que a aplicação começa por apresentar o menu para o pagamento (1) Figura 26 c). Quando o utilizador carrega do botão "*Buy with Google*" a aplicação (2) cria um objeto do tipo *GoogleApiClient* de modo a criar uma ligação com o Google Wallet e envia um objeto do tipo *Masked wallet request* através dessa ligação. Este objeto *Masked wallet request* consiste em requisitar ao Google Wallet parcialmente o cartão de crédito e ainda caso seja pedida a morada associada ao cartão para posteriormente ser exibida pela aplicação. (3) O Google Wallet envia um objeto *Masked wallet response* com as informações requisitadas no *Masked wallet request.* (4) A aplicação apresenta o menu para o utilizador confirmar o pagamento. No caso de serem feitas alterações ao cartão de pagamento ou morada de faturação a aplicação envia novamente um *Masked wallet request* de modo a obter um objeto *Masked wallet* atualizado. (5) Quando o utilizador confirma o pagamento a aplicação criar novamente um objeto do tipo GoogleApiClient de modo poder enviar para o Google Wallet um objeto do tipo *Full wallet request*. Este objeto *Full wallet request* tem como objetivo pedir ao Google Wallet que lhe envie as informações necessárias para que a aplicação possa concluir o pagamento. A este pedido o *Google Wallet* envia um objeto Full wallet response que contém um cartão virtual de uso único de modo a que este possa ser usado pela aplicação para fazer o pagamento. (6) O *backoffice* da aplicação processa o pedido de pagamento e envia para a aplicação o estado do processo (sucesso ou insucesso). (7) A aplicação cria um objeto do tipo *notifyTransactionStatus* para informar o Google Wallet do estado do pagamento e é apresentado o menu da Figura 34.

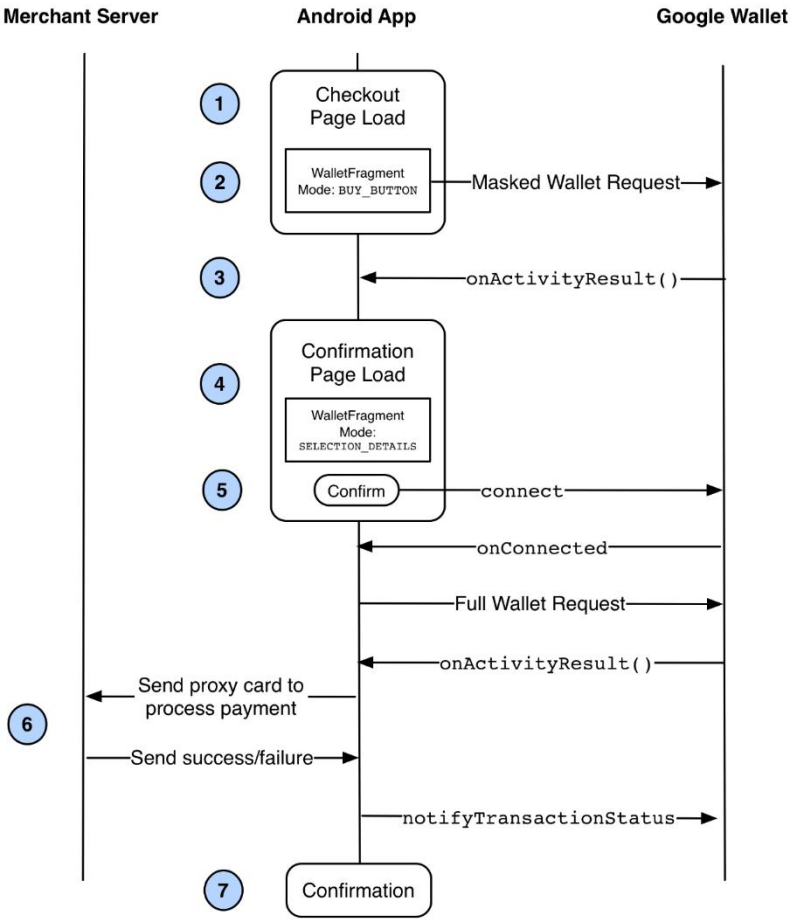


Figura 35 – Diagrama do funcionamento da vertente Instatbuy do Google Wallet

5.2 Aplicação Teste NFC

Esta aplicação surge da necessidade em testar a componente NFC do projeto. Na ausência de um terminal que fizesse a leitura de NFC, foi usado outro *smartphone* com NFC de modo a poder simular esse terminal.

Assim com o objetivo de testar todos os cenários de utilização do NFC, e em vez de serem criadas aplicações diferentes para cada um deles, foi criada apenas uma aplicação que junta todos esses cenários. Como é possível observar na Figura 36 a aplicação começa por apresentar um menu, onde é possível selecionar que vertente da aplicação que queremos testar.

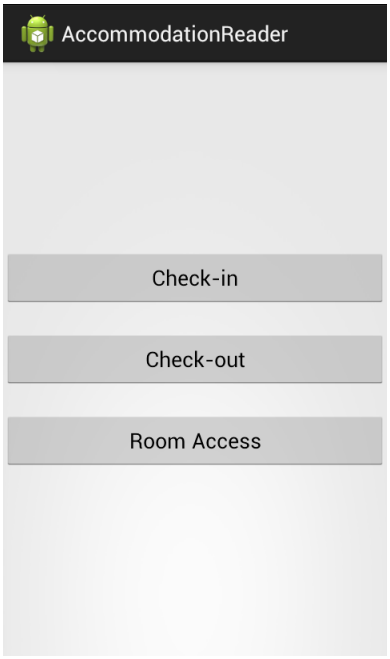


Figura 36 - Menu Principal

Como a aplicação tem como objetivo servir de leitor NFC, é fundamental desativar o *Android Beam*. Este é o responsável pela rápida utilização do NFC utilizando o modo P2P e tem de ser desativada, pois o SO Android faz com que seja sempre utilizada mal exista a proximidade necessária entre dois dispositivos. Para o efeito, surgiu na API 19 um método destinado para desativar o *Android Beam* e ativar o modo de leitura, o método *enableReaderMode*, que se encontra disponível através do objeto *NfcAdapter*. No Código 4 é possível observar o método criado para mais facilmente ativar o modo de leitura. Este método é utilizado no método *onResume* da *Activity* que necessita de ativar o modo de leitura. O atributo *DoorReader.this* é o contexto da própria *Activity,* o mDoorReader é uma instância da classe criada para fazer a leitura das *tags* NFC, *READER_FLAGS* consiste num conjunto de *flags* que vão ser permitidas fazer a sua leitura, como por exemplo *FLAG_READER_NFC_A*.

```
private void enableReaderMode() {
NfcAdapter nfc = NfcAdapter.getDefaultAdapter(getApplicationContext());
if (nfc != null) {
nfc.enableReaderMode(DoorReader.this, mDoorReader, READER_FLAGS, null);
      }
}
```

Código 4 - Método enableReaderMode

Do mesmo modo que é desativado o modo P2P quando é necessário fazer a leitura, convém que este seja de novo ativado quando esta leitura deixar de ser necessária. Assim, é também utilizado um método do objeto *NfcAdapter*, o *disableReaderMode*. Tal como para o método que ativa a leitura, e para mais facilmente utilizar o *disableReaderMode*, foi criado um

método que é utilizado quando a aplicação se encontra em pause, isto é, no método *onPause*. O método *disableReaderMode* necessita apenas do contexto da própria *Activity* para desativar o modo de leitura.

5.2.1 Leitor

A classe responsável pela leitura das *tags* implementa uma interface chamada *NfcAdapter.ReaderCallback.* Esta interface permite que, quando uma *Activity* se encontra em primeiro plano e está a funcionar em modo de leitura, invocar o método *onTagDiscovered,* que será responsável pelo processamento da *tag* detetada.

O método *onTagDiscovered* é semelhante para cada uma das diferentes funcionalidades, sendo que a única diferença é a construção do APDU. Como referido o primeiro APDU enviado tem de conter o AID do serviço que o irá processar, assim cada uma das funcionalidades tem de enviar um APDU com o AID do serviço para quem se destina. Tal como na aplicação estudada, quando uma *tag* é detetada é criado um objeto *IsoDep* com essa *tag,* de modo a que seja possível trocar dados entre o leitor NFC e a *tag*. Após criar a ligação entre os dois dispositivos, é enviado um APDU para a aplicação com o AID do serviço e é recebida uma resposta por parte da *tag.* No caso de a *tag* possuir o serviço, são então enviados dados da *tag* para o leitor, sendo enviado o *status word* no fim da transmissão dos dados. Este *status word* é processado e, no caso de ser positivo, os dados recebidos são enviados para a *Activity* que instanciou a classe leitor, através do método *onDataReceived*. A implementação do método *onTagDiscovered* pode ser observada no Anexo A e as interações entre a classe leitor e cada uma das *Activities* será apresentada na descrição das mesmas, isto é, entre o leitor e a *Activity* encarregue do processo de testar o processo de *check-in* na seção 5.2.2, do leitor e a Activity encarregue do processo de testar o *check-out* na seção 5.2.3 e entre o leitor e a Activity encarregue pelo processo de testar o acesso ao quarto na secção 5.2.4.

5.2.2 Check-in

A *Activity* encarregue pelo processo de *check-in* começa por instanciar a classe leitor, utilizando para isso o seu contexto e o AID do serviço de *check-in* (F2222222210). A partir deste momento fica apta a interagir com a aplicação de alojamento quando esta se encontra à distância correta. Como referido anteriormente quem inicia a troca de mensagens é o leitor, que neste caso, por ter sido instanciado pelo processo de *check-in* irá enviar para a aplicação de alojamento o AID de *check-in*. A aplicação de alojamento processa o AID e envia para o leitor os dados respetivos, que posteriormente serão enviados para a Activity de *check-in.* De modo a que seja possível transferir os dados da classe leitor para a Activity de *check-in* é necessário implementar um interface criada pela classe leitor. Esta interface é constituída por um método, o *onDataReceived*, e a sua implementação para o processo de *check-in* pode ser visualizada no Anexo A.

Os dados recebidos através do método *onDataReceived* são processados, sendo posteriormente criado um objeto do tipo *JSONObject*, de modo a que possam ser enviados para o serviço *web* que esta encarregue do processo de *check-in*.

Após o serviço *web* processar o pedido de *check-in,* envia uma resposta, e no caso do processo de *check-in* ter sido executado com sucesso é necessário enviar para a *tag* os dados recebidos, isto é, a informação com o número do quarto e a chave respetiva. Assim, é novamente enviado um APDU com esta informação e no caso de a *tag* processar os dados com sucesso é enviado um *status word* que informa o ocorrido. Para este caso de sucesso a aplicação apresenta no ecrã a cor verde, no caso da resposta recebida pelo serviço *web* indicar que o processo de *check-in* não foi efetuado, é apresentada a cor vermelha. Para que todo o processo de *check-in* seja melhor compreendido é apresentado na Figura 37 um diagrama de sequência do processo.

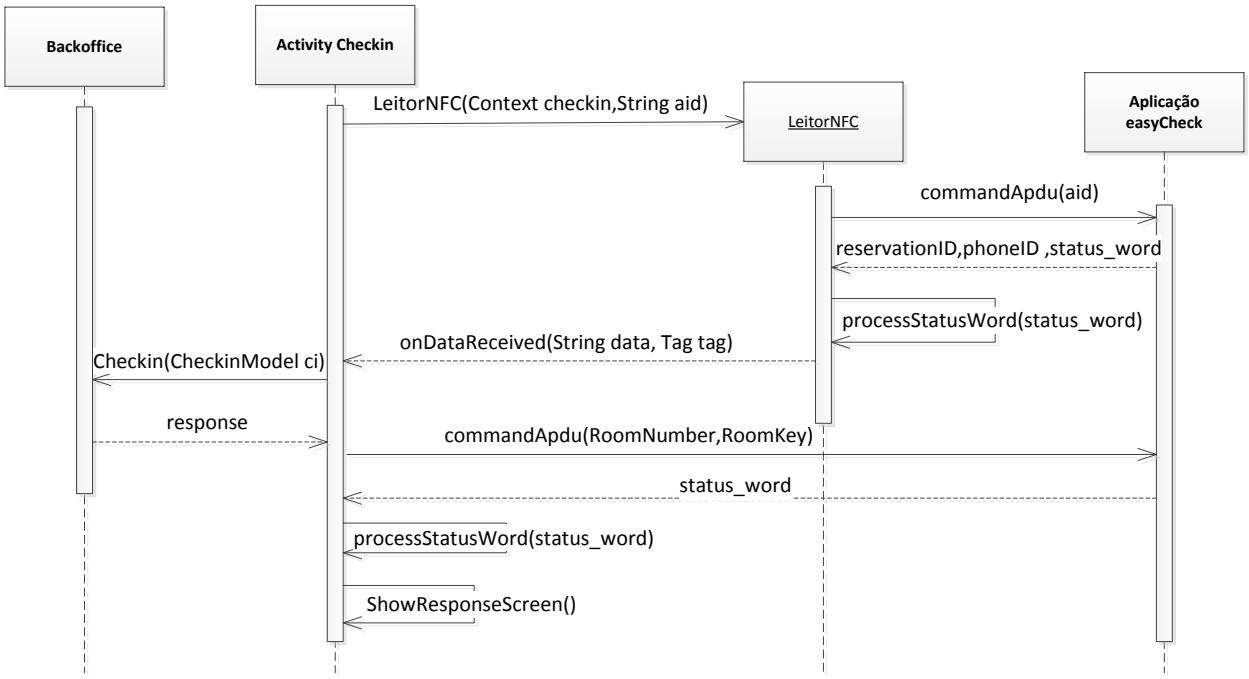


Figura 37 – Diagrama de sequência *check-in*

5.2.3 Check-out

A *Activity* encarregue do processo de *check-out*, tal como a *Activity* de *check-in,* começa por instanciar a classe leitor, utilizando para isso o seu contexto e o AID do serviço de *check-out* (F2222222220). Como referido anteriormente quem inicia a troca de mensagens é o leitor, que neste caso, por ter sido instanciado pelo processo de *check-out* irá enviar para a aplicação de

alojamento o AID de *check-out*. A aplicação de alojamento processa o AID e envia para o leitor os dados respetivos, que posteriormente serão enviados para a Activity de *check-out.*

Tal como para o *check-in* o método *onDataReceived* tem de ser implementado pela Activity de *check-out.* Esta implementação é no entanto semelhante ao da *Activity* de *check-in*, sendo a única diferença o serviço *web* que é utilizado e os dados que são enviados. No caso do processo de *check-out* é utilizado o serviço *web* com o URI /Checkout e é enviado o identificador da reserva no pedido. No caso do processo de check-out ser bem sucedido o ecrã apresenta a cor verde, e envia para a aplicação um APDU que indica o sucesso da operação. Caso não tenha sucesso apresenta a cor vermelha. Para que todo o processo de *check-out* seja melhor compreendido é apresentado na Figura 38 um diagrama de sequência do processo.

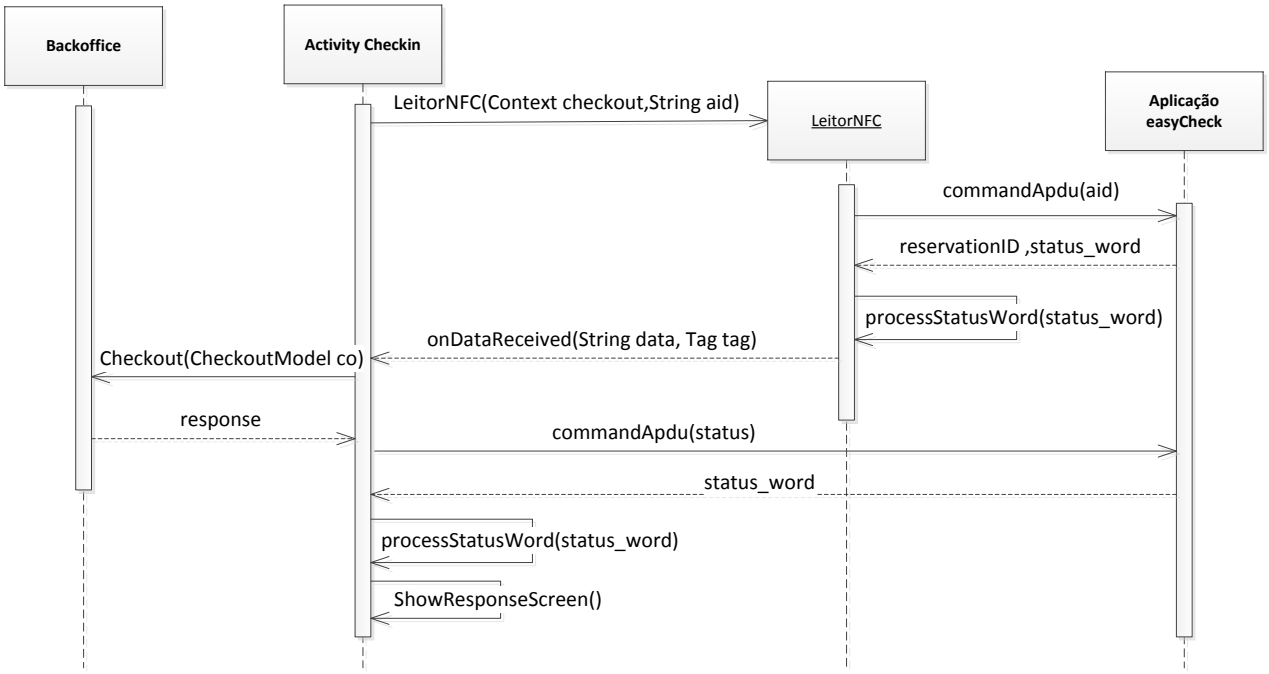


Figura 38 - Diagrama de sequência *check-out*

5.2.4 Acesso ao quarto

A *Activity* encarregue do processo de acesso ao quarto, tal como as *Activities* anteriores, começa por instanciar a classe leitor, utilizando para isso o seu contexto e o AID do serviço de acesso ao quarto (F2222222230).

Tal como para as Activities anteriores, o método *onDataReceived* tem de ser implementado pela Activity de acesso ao quarto. Esta implementação é no entanto idêntica às anteriores, sendo a única diferença o serviço *web* que é utilizado e os dados que são enviados.

Neste processo de acesso ao quarto é utilizado o serviço *web* com o URI /DoorAccess e os dados enviados no pedido são as chaves dos quartos e o identificador do *smartphone*.

No caso do acesso ao quarto ser bem-sucedido a cor muda para verde, no caso de não ser bem-sucedida muda para vermelho. Para que todo este processo de acesso ao quarto seja melhor compreendido é apresentado na Figura 39 um diagrama de sequência do processo.

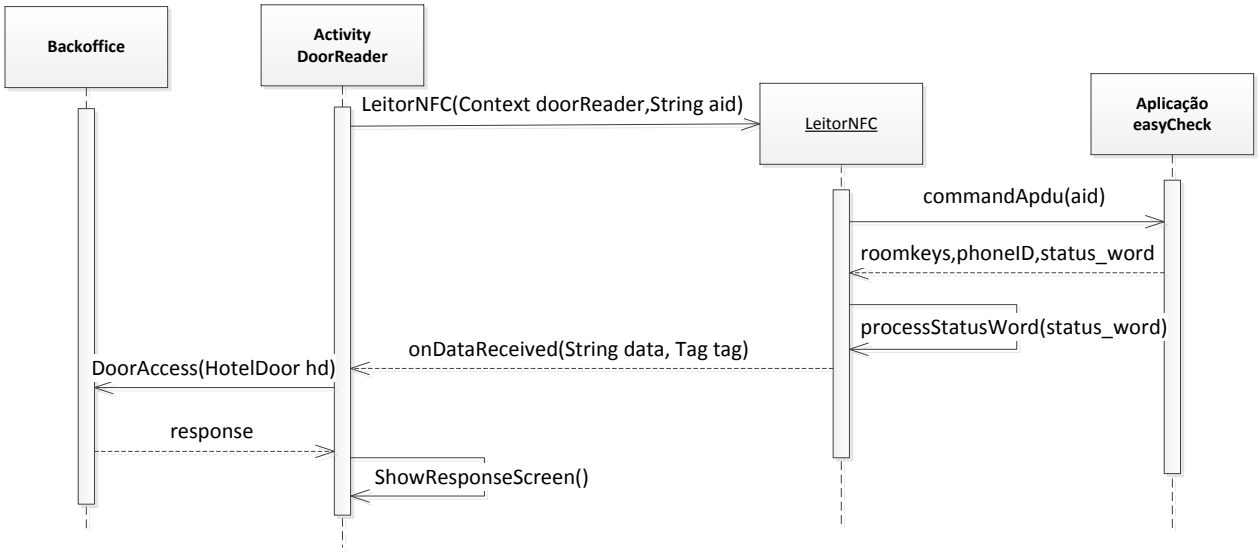


Figura 39 – Diagrama sequência de acesso ao quarto

5.3 Backoffice

Para suportar as aplicações desenvolvidas e para garantir uma maior segurança dos sistemas foi desenvolvido um *backoffice*. Este tem como objetivo disponibilizar serviços para as aplicações e ainda armazenar os dados relativamente a esses serviços. Entre os serviços disponibilizados encontram-se: fazer registar-se, fazer uma reserva, fazer *check-in,* fazer *check-out*, entre outros. O processo de armazenamento dos dados pode ser consultado na secção 5.3.1 e os serviços disponíveis podem ser consultados na secção 5.3.2.

5.3.1 Base de Dados

Com a necessidade de armazenar permanentemente os dados referentes à aplicação easyCheck, foi criada uma base de dados relacional SQL. Esta base de dados é constituída por apenas cinco tabelas: a tabela Users, a tabela Reservations, a tabela Rooms, a tabela RoomTypes e a table RoomPrice.

A tabela Users tem como objetivo guardar as informações pessoais dos utilizadores da aplicação. Entre os atributos habituais, como nome, palavra-chave, morada, entre outros, encontra-se o atributo “ProfileComplete”. Este atributo tem como objetivo saber se o utilizador possui um perfil válido, para assim poder fazer reservas.

A tabela Reservations é a tabela onde são guardadas as reservas dos utilizadores. Para além da data de início e de fim da estadia, contém ainda atributos que possibilitam saber se o utilizador já fez *check-in* ou *check-out* na unidade de alojamento. Possui também um atributo para associar o identificador do *smartphone* a uma determinada reserva (*Phone_Id*), deste modo, garante que somente o utilizador que fez a reserva pode ter acesso ao quarto. Existe ainda o atributo *Tag_Id* que permite criar mais uma camada de segurança no caso de o utilizador fazer o *check-in* utilizando a vertente NFC. Como referido, um dispositivo em modo de HCE é visto pelos leitores como sendo um *tag*, assim é guardado o identificador da *tag* do dispositivo para garantir a identidade do smartphone.

A tabela Rooms é constituída por atributos referentes ao quarto do local de alojamento, contendo o número do quarto e o identificador do tipo de quarto. O Atributo *RoomType_Id* está associado à tabela RoomType, que como o nome indica, tem como objetivo guardar os tipos de quartos existente. Para além da descrição do tipo de quarto possui ainda o atributo *RoomPrice_Id*, associado à tabela RoomPrice, que como o nome indica, contém os preços dos quartos. Esta tabela foi criada, pois existe a possibilidade de existirem preços diferentes para um quarto, por exemplo dependendo da altura do ano. Desta forma, podem ser criados preços diferentes, podendo ser colocada uma descrição para facilitar a atribuição dos preços.

A Figura 40 contém todos os atributos contidos nas tabelas descritas, bem como as relações entre elas.

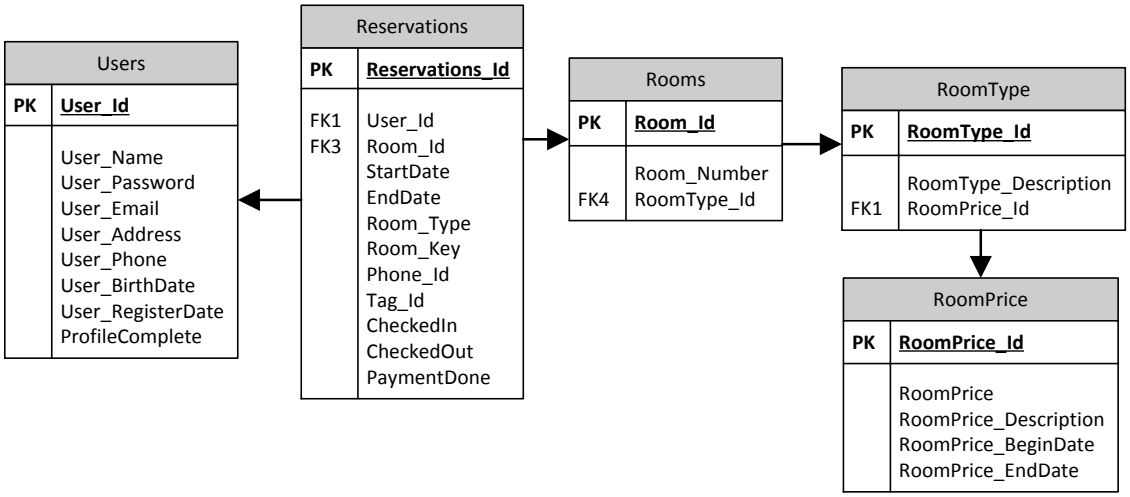


Figura 40 – Base de dados

5.3.2 Serviços Web

Os serviços *web* surgiram com a necessidade da aplicação móvel comunicar com o *backoffice* que suporta a aplicação. Deste modo foram criados serviços *web* REST, com o objetivo de tornar o processo de comunicação mais rápido.

Os serviços *web* disponíveis são apresentados pela Tabela 5.

Tabela 5 – Serviços *Web*

Método	URI	Dados Pedido	Dados Resposta
POST	/Register_Login	UserModel	int
GET	/User/{UserId}	String	UserModel
PUT	/User/{UserId}	UserModel, String	Boolean
POST	/Reservation	ReservationModel	ReservationResponse
GET	/Reservation/{UserId}	String	Reservations
POST	/Checkin	CheckinModel	ReservationModel
POST	/Checkout	CheckoutModel	ReservationModel
GET	/AccessPoints	(Não necessita)	AccessPoints
POST	/DoorAccess	HotelDoor	Boolean

Sempre que necessário foram criadas estruturas para os dados para garantir uma maior segurança aos mesmo. De seguida é apresentada uma descrição com maior detalhe cada um dos serviços *web*:

5.3.2.1 Serviço registo/login

Este serviço tem como URI o endereço /Register_Login e é composto por um método POST que foi desenvolvido com o objetivo de fazer o registo de novos utilizadores e ainda fazer a sua autenticação. O serviço necessita que seja enviado no pedido a estrutura *UserModel* (Código 5**Erro! A origem da referência não foi encontrada.**), uma estrutura composta pelos dados pessoais do utilizador. Neste serviço não é necessário enviar todas as informações existentes na estrutura *UserModel*, sendo somente necessário enviar o *email* e a palavra-chave. Quando o serviço recebe esses dados verifica se o *email* já se encontra na base de dados de utilizadores, caso já exista verifica se a palavra-chave coincide com a palavra-chave previamente guardada. No caso de não existir, o serviço assume que se trata de um novo utilizador, procedendo assim à criação de um novo utilizador. A resposta deste pedido consiste no envio do identificador do utilizador.

```
[DataContract]
public class UserModel
{
    [DataMember]
    public int UserId { get; set; }
    [DataMember]
    public string Name { get; set; }
    [DataMember]
    public string Password { get; set; }
    [DataMember]
    public string Email { get; set; }
    [DataMember]
    public string Address { get; set; }
    [DataMember]
    public int Phone { get; set; }
    [DataMember]
    public String BirthDate { get; set; }
    [DataMember]
    public DateTime RegisterDate { get; set; }
    [DataMember]
    public Boolean ProfileComplete { get; set; }
}
```

Código 5 - Estrutura *UserModel*

5.3.2.2 Serviço devolver ou alterar as informações do utilizador

Este serviço tem como URI o endereço /User/{UserId} e é composto por um método PUT e por um método GET. O método PUT foi desenvolvido com a finalidade de editar as informações pessoais relativamente a um utilizador. Necessita que seja enviado no pedido a estrutura de dados *UserModel* e o identificador do utilizador. Como resposta recebe um booleano indicando, verdadeiro, se tiver sido editado com sucesso ou falso, caso contrário.

O método GET foi desenvolvido com a finalidade de fornecer as informações pessoais de um utilizador. Necessita que seja enviado no pedido o identificador do utilizador e como resposta recebe os dados do utilizador sob a forma da estrutura *UserModel*.

5.3.2.3 Serviço criar reservas

Este serviço tem como URI o endereço /Reservation e é constituído por um método POST que foi desenvolvido com o objetivo de fazer as reservas para os quartos do local de alojamento. No pedido é necessário enviar a estrutura de dados *ReservationModel* (Código 6) e como resposta recebe a estrutura de dados, *ReservationResponse* (Código 7). O serviço começa por verificar se para as datas e para o tipo de quarto que o utilizador selecionou existe algum quarto disponível. Caso exista, procede à reserva do quarto e envia no atributo *Response* existente no *ReservationResponse* a indicação de sucesso, 1, caso não exista envia a indicação da falha do processo de reserva, -1.

```
[DataContract]
public class ReservationModel
    {
        [DataMember]
        public int ReservationId { get; set; }
        [DataMember]
        public int UserId { get; set; }
        [DataMember]
        public string StartDate { get; set; }
        [DataMember]
        public string EndDate { get; set; }
        [DataMember]
        public int RoomKey { get; set; }
        [DataMember]
        public int RoomNumber { get; set; }
        [DataMember]
        public string RoomType { get; set; }
        [DataMember]
        public string PhoneId { get; set; }
        [DataMember]
        public string TagId { get; set; }
        [DataMember]
        public Boolean CheckedIn { get; set; }
        [DataMember]
        public Boolean CheckedOut { get; set; }
    }
```

Código 6 - Estrutura *ReservationModel*

```
[DataContract]
public class ReservationResponse
 {
     [DataMember]
     public int Response  { get; set; }
 }
```

Código 7 - Estrutura *ReservationResponse*

5.3.2.4 Serviço devolver reservas

Este serviço tem como URI o endereço /Reservation/{UserId} e é constituído por um método GET que foi desenvolvido com o objetivo de retornar as reservas de um utilizador que ainda se encontram válidas, isto é, reservas que possuem uma data de fim superior ao dia atual. Para o pedido é enviado o identificador do utilizador e como resposta é enviada uma estrutura com a lista de reservas, a estrutura *Reservations* (Código 8).

```
[DataContract]
public class Reservations
{
    [DataMember]
    public List<List<String>> ReservationsList { get; set; }
}
```

Código 8 - Estrutura *Reservations*

5.3.2.5 Serviço check-in

Este serviço tem como URI o endereço /Checkin e é constituído por um método POST que foi desenvolvido com o objetivo de fazer o processo de *check-in* e de retornar as informações de um quarto. No pedido é utilizada a estrutura *CheckinModel* (Código 9) e como resposta é enviado a estrutura *ReservationModel*. O serviço começa por verificar se já tinha sido feito o *check-in* da reserva. No caso de já ter sido feito retorna as informações do quarto, caso contrário o serviço verifica se a data de início de reserva é a mesma do dia atual. Se forem idênticas atualiza a reserva e retorna as informações do quarto, caso contrario o *check-in* não é permitido. Durante o processo de atualização da reserva é associado o identificador do *smartphone* à reserva e ainda gerada uma chave aleatória para a porta do quarto. A indicação do processo de *check-in* ter sido feito com sucesso está associada ao atributo *CheckedIn* da estrutura *ReservationModel*. Se este atributo for verdadeiro, significa que o processo de *check-in* foi efetuado com sucesso e falso caso contrário.

```
[DataContract]
public class CheckinModel
{
    [DataMember]
    public int RsvID { get; set; }
    [DataMember]
    public String PhoneID { get; set; }
    [DataMember]
    public String TagID { get; set; }

}
```

Código 9 - Estrutura *CheckinModel*

5.3.2.6 Serviço check-out

Este serviço tem como URI o endereço /Checkin e é constituído um método POST que foi desenvolvido com o objetivo de fazer o processo de *check-out*. No pedido é utilizada a estrutura *CheckoutModel* e como resposta é enviado a estrutura *ReservationModel*. O serviço recebe através do *CheckoutModel* (Código 10) o identificador da reserva que posteriormente é utilizado para atualizar a reserva, indicando na resposta do serviço e através do atributo *CheckedOut* o estado do processo. Se foi feito com sucesso o atributo *CheckedOut* será verdade, caso contrário será falso.

```
[DataContract]
public class CheckoutModel
{
    [DataMember]
    public int RsvID { get; set; }
}
```

Código 10 - Estrutura *CheckoutModel*

5.3.2.7 Serviço accessPoints

Este serviço tem como URI o endereço /AccessPoints e é constituído por um método GET que foi desenvolvido com o objetivo de retornar os *Basic Service Set IDentification* (BSSID) dos pontos de acesso aos quais o utilizador se pode ligar de modo a fazer o *check-in*. Não é enviada nenhuma informação no pedido e como resposta recebe a estrutura *AccessPoints* (Código 11). Esta estrutura contém uma lista com os BSSIDs dos pontos de acesso autorizados para o utilizador fazer o *check-in*.

```
[DataContract]
public class AccessPoints
{
    [DataMember]
    public List<String> AccessPointsList { get; set; }
}
```

Código 11 - Estrutura *AccessPoints*

5.3.2.8 Serviço acesso ao quarto

Este serviço tem como URI o endereço /DoorAccess e é constituído por um método POST que foi desenvolvido com o objetivo de verificar se uma determinada chave abre uma porta de um quarto. No pedido é enviado a estrutura *HotelDoor* (Código 12), que contém o número da porta ao qual se quer aceder, uma lista com as chaves que o utilizador possui e o identificador do *smartphone*. O identificador do *smartphone* existe para garantir que somente o *smartphone* que fez a reserva pode aceder ao quarto, criando uma camada extra de segurança.

```
[DataContract]
public class HotelDoor
{
    [DataMember]
    public int DoorID { get; set; }
    [DataMember]
    public List<String> DoorKey { get; set; }
    [DataMember]
    public String PhoneID { get; set; }
}
```

Código 12 - Estrutura *HotelDoor*

5.3.3 Interface

Com o desenvolver do projeto surgiu a necessidade de poder verificar se os dados que estavam a ser introduzidos na base de dados estavam corretos. Desta forma, e sem ser necessário abrir cada uma das tabelas diretamente na base de dados, surge uma interface para o utilizador que poderá facilmente consultar todos os dados armazenados pelo sistema. Este interface do utilizador poderá perfeitamente estar disponibilizada no computador da receção podendo, para além de consultar os dados, retificar informações dos clientes.

Esta aplicação foi desenvolvida utilizando *Windows Forms* e na Figura 41 é possível observar a tabela referente aos utilizadores da aplicação móvel. Neste menu, "Users", são apresentados dados relevantes para poderem ser interpretados pelo utilizador interface do *backoffice*. Para além da consulta é possível ainda editar os dados do utilizador, no caso de ser necessário alterar alguma informação. Para cada uma das outras opções são apresentadas as suas respetivas informações, isto é, as informações existentes na Figura 40.

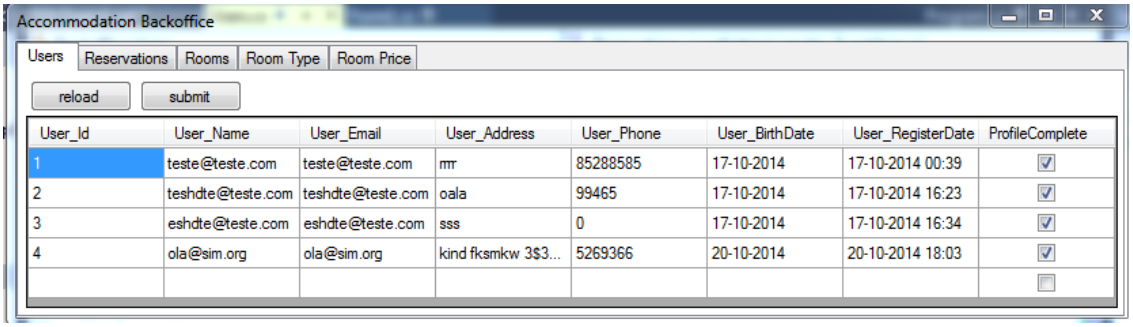


Figura 41 – Interface do Utilizador Backoffice

5.4 Questionário sobre a aplicação easyCheck

Após o desenvolvimento do sistema era necessário testá-lo de modo a descobrir potenciais falhas e procurar melhorá-lo. Deste modo a aplicação de alojamento foi disponibilizada a algumas pessoas para ser testada, e por conseguinte todo o restante sistema. No fim de realizarem o teste da aplicação foi requisitado que respondessem a um pequeno questionário de modo a avaliar o desempenho da mesma e das suas tecnologias. Foram oito o número de pessoas que responderam a este questionário, tendo estas uma faixa etária compreendida entre os 22 e os 26 anos. Os inquiridos possuíam uma educação superior e conhecimentos tecnológicos avançados. De seguida serão apresentadas as conclusões retiradas desse questionário que pode ser consultado no Anexo B.

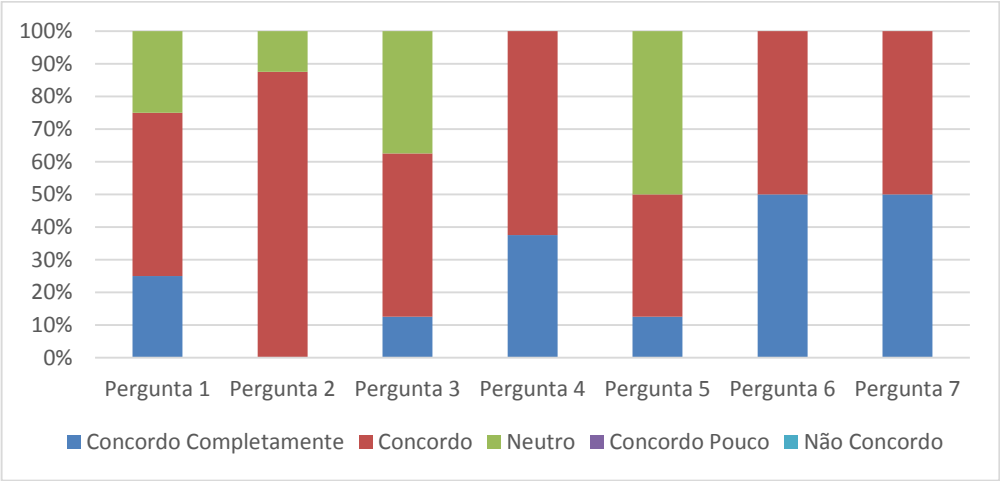


Figura 42 – Respostas às questões do inquérito

Pergunta 1- A aplicação tornaria a minha estadia mais confortável?

Pergunta 2- A aplicação facilita as valências existentes num local de alojamento?

Pergunta 3- A aplicação tem uma interface intuitiva?

Pergunta 4 - A aplicação tem uma velocidade de resposta aceitável?

Pergunta 5 - Sinto que a aplicação é segura

Pergunta 6 - Utilização da tecnologia NFC é intuitiva?

Pergunta 7 - Da próxima vez que comprar um *smartphone* é provável que escolha um que suporte esta tecnologia?

Após a análise das respostas existentes na Figura 42 foi possível concluir que que 75% dos inquiridos considerou que a aplicação desenvolvida tornaria a estadia mais confortável.

Relativamente à interface existente 62% consideram que era intuitiva e 38% tiveram uma opinião neutra. No que diz respeito à velocidade da aplicação, a totalidade dos inquiridos considerou que a aplicação tem uma velocidade aceitável.

Na questão referente à segurança da aplicação 50 % os inquiridos considerou a aplicação segura e os restantes 50% tiveram uma opinião neutra, isso é, nem segura nem insegura.

No que diz respeito à tecnologia NFC a totalidade dos inquiridos considerou que se trata de uma tecnologia intuitiva e que na compra um futuro *smartphone* terão em consideração se este vem equipado com a tecnologia NFC.

Na Figura 42 é também possível constatar que 87% dos inquiridos utilizaria o *smartphone* para realizar o pagamento da estadia em substituição aos sistemas existentes.

Por fim foi colocada uma questão relativamente aos processos de *check-in/check-out* existentes (Wi-Fi ou por NFC). É então possível constatar a partir da análise da Figura 43 que 50% dos inquiridos prefere fazer o *check-in* por NFC e que para os restantes 50% qualquer uma das duas opções disponíveis era aceitável. Desta forma é possível concluir que existe uma preferência pela utilização da tecnologia NFC comparativamente ao Wi-Fi.

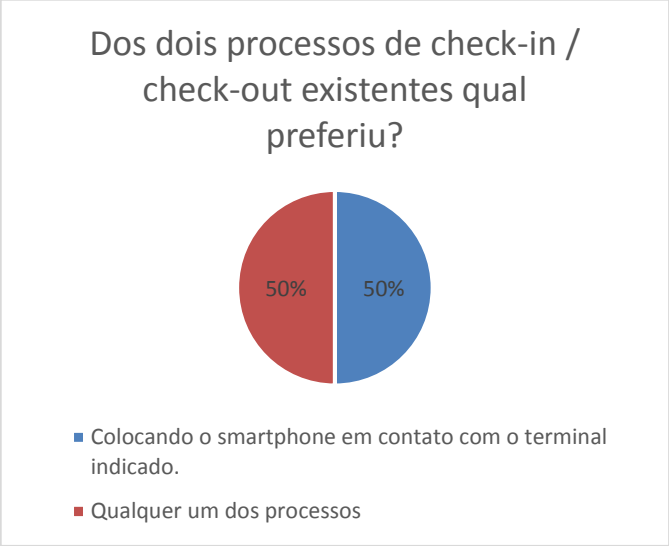


Figura 43 – Questão relativamente aos processos *check-in/check-out*

6 Conclusão

Este é o capítulo final do documento e nele são apresentadas as conclusões sobre o trabalho desenvolvido. São também apresentadas as principais dificuldades encontradas ao longo do desenvolvimento do projeto, bem como as limitações que este apresenta. Por fim, são referidas algumas propostas para possíveis desenvolvimentos futuros.

Com o atual crescimento do mercado dos *smartphones*, e por conseguinte a sua maior utilização, estes tornam-se dispositivos com caraterísticas pessoais. Apoiada por estas caraterísticas pessoais surge então a oportunidade para utilizar os *smartphones* para substituir os atuais *smartcards*. Este trabalho foi desenvolvido com esse objetivo, mais concretamente utilizar os *smartphones* para garantir o acesso a um quarto de um local de alojamento.

Para tal, foi utilizada uma tecnologia que ainda não se encontra muito difundida no mundo tecnológico atual, mas que se mostra bastante promissora, a tecnologia NFC. De modo a utilizar esta tecnologia foi utilizado o SO Android, que a partir da sua versão 4.4, permite o uso do modo HCE. Este modo da tecnologia NFC é o ideal para este projeto pois permite a emulação de cartões, sem que seja necessário a utilização de um ES que garanta a fiabilidade do cartão.

Assim foi criada uma aplicação destinada a, não só permitir ao utilizador fazer reservas, mas também utilizar a mesma aplicação para fazer o processo de *check-in* e por conseguinte aceder ao seu quarto. No fim da sua estadia o utilizador pode também fazer o *check-out* através da aplicação.

Para além da aplicação móvel foi criado um *backoffice*. Este permite que a aplicação seja mais "leve", permitindo ainda que caso sejam necessárias fazer alterações ao modelo desenvolvido não seja necessário alterar a aplicação móvel.

De modo a implementar este sistema e de modo a interligar a aplicação Android com o *backoffice* que faz a sua gestão, foi proposta uma solução que passou pela utilização de serviços *web*, adotando uma arquitetura baseada em SOA. Para o desenvolvimento destes serviços

foram utilizados princípios REST, pois garantem um menor *overhead*. No caso da utilização de tecnologias móveis este é um fator bastante importante, pois um tempo de resposta demorado pode inviabilizar a utilização da aplicação.

Para testar as funcionalidades que utilizam a tecnologia NFC da aplicação de alojamento foi criada uma outra aplicação. Esta aplicação tem como objetivo simular a utilização de leitores NFC que, não estando disponíveis no desenvolvimento deste projeto, são necessários para a utilização de determinadas funcionalidade da aplicação de alojamento.

As principais dificuldades sugiram com o facto da tecnologia NFC, embora não sendo uma tecnologia recente, ainda não se encontrar muito utilizada. Assim, e mais concretamente para o modo de HCE, somente puderam ser utilizadas as páginas da API da Google como suporte à implementação das funcionalidades que utilizam HCE.

Outra dificuldade surge com o facto de serem utilizadas várias tecnologias diferentes. Embora tenham um custo de aprendizagem inicial maior, o uso de todas estas tecnologias, tornam a solução proposta, mais robusta e eficiente.

Os objetivos propostos foram cumpridos, no entanto existem sempre otimizações que podem ser feitas. Podem, por exemplo, ser acrescentados mais mecanismos de segurança, sem esquecer que é fundamental não baixar muito o desempenho da aplicação, pois podem inviabilizar o seu uso. Pode ser desenvolvida uma aplicação *web* que permita ao utilizador fazer ou consultar reservas sem ter de utilizar a aplicação móvel, ou ainda acrescentar novas funcionalidades à aplicação. Entre estas novas funcionalidades poderia constar por exemplo a implementação de um serviço de pagamento utilizando a tecnologia NFC ou ainda implementar mais alguns dos serviços disponíveis no local de alojamento, tais como serviço de lavandaria ou requisitar o serviço de quartos.

Referências

[Angelaccio, et al., 2012] Michele Angelaccio, Alessandra Basili, Berta Buttarazzi, and Walter Liguori. 2012. "*Smart and Mobile Access to Cultural Heritage Resources: a Case Study on Ancient Italian Renaissance Villas*"

[Assa Abloy, 2011] Assa Abloy. 2011. "*Evaluation of the world's first pilot using NFC phones for check-in and hotel room keys*".

[Balaban, 2008] Dan Balaban. 2008. "*London Oyester Card Chief: NFC Not Ready for Fast-Paced Fare Payment*". [Online] Disponível em: http://nfctimes.com/news/london-oyster-card-chief-nfc-not-ready-fast-paced-fare-payment. [Acedido em 03 2014].

[Blöckner, et al.,2009] Magdalena Blöckner, Svetlana Danti, Jennifer Forrai, Gregor Broll, Alexander De Luca., 2009. "*Please Touch the Exhibits! Using NFC-based Interaction for Exploring a Museum*"

[Clark, 2010a] Sara Clak, 2010, "*Google unveils first Android NFC phone — but Nexus S is limited to tag reading only for now*". [Online] Disponível em: http://www.nfcworld.com/2010/12/07/35385/google-unveils-first-android-nfc-phone-but-nexus-s-is-limited-to-tag-reading-only-for-now/. [Acedido em 02 2014].

[Clark, 2010b] Sara Clak, 2010, "Centre Pompidou's Teen Gallery lets young people test NFC". [Online] Disponível em: http://www.nfcworld.com/2010/11/19/35152/centre-pomidou-teen-gallery-nfc/. [Acedido em 04 2014].

[Ceipidor, et al., 2013] U. Biader Ceipidor, C. M. Medaglia, V. Volpi, A. Moroni, S. Sposato, M. Carboni, A. Caridi., 2013. "*NFC technology applied to touristic-cultural field: a case study on an Italian museum*"

[Curran, et al., 2012] Kevin Curran, Amanda Millar, Conor Mc Garvey.2012." Near Field Communication".

[Dmitrienko, 2013] Alexandra Dmitrienko, 2013." *Access Control in Enterprises with Key2Share*". [Online] Disponível em: https://www.sit.fraunhofer.de/en/offers/projekte/key2share/ [Acedido em 04 2014].

[ECMA, 2010] ECMA.2010. "*NFC-SEC-01: NFC-SEC Cryptography Standard using ECDH and AES*"

[ECMA, 2013] ECMA. 2013. "*NFC-SEC:NFCIP-1 Security Services and Protocol*"

[Fraser, 2011] Adam Fraser. 2011. "*Nokia's NFC Phone History*". [Online] Disponível em: http://conversations.nokia.com/2012/04/11/nokias-nfc-phone-history/ [Acedido em 02 2014].

[Fraunhofer, 2013] Fraunhofer Institute for Secure Information Technology, 2013. [Online] Disponível em: http://www.key2share.de/en/concept.html [Acedido em 04 2014].

[Gartner, 2014a] Gartner. 2014. "*Gartner Says Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013*". [Online] Disponível em:

http://www.gartner.com/newsroom/id/2665715 [Acedido em 03 2014].

[Gartner, 2014b] Gartner. 2014. "*Gartner Says Worldwide Traditional PC, Tablet, Ultramobile and Mobile Phone Shipments On Pace to Grow 7.6 Percent in 2014*". [Online] Disponível em: http://www.gartner.com/newsroom/id/2645115 [Acedido em 03 2014].

[Google Wallet, 2011] Google Wallet. 2011. "*Google Wallet - faq*". [Online] Disponível em: http://www.google.com/wallet/faq.html [Acedido em 04 2014].

[Google Wallet, 2013] Google Wallet. 2013. "Google Wallet Instant Buy APIs". [Online] Disponível em: https://developers.google.com/wallet/instant-buy/ [Acedido em 09 2014].

[GSM Association, 2011] GSM Association. 2011. "*M-Ticketing Whitepaper*".

[GSM Association, 2013] GSM Association. 2013. "*Case Study: Isis Mobile Wallet*".

[Haselsteiner et al.,2006] Ernst Haselsteiner and Klemens Breitfuß.2006." *Security in Near Field Communication (NFC) - Strengths and Weaknesses"*.

[Haselton, 2014] Todd Haselton. 2014. "*Google Wallet to Require Android 4.4 For Tap-and-Pay Soon*". [Online] Disponível em: http://www.technobuffalo.com/2014/03/14/google-wallet-to-require-android-4-4-for-tap-and-pay-soon/ [Acedido em 03 2014].

[IBM Corp., 1994] IBM Corp.. 1994." *Simon User Manuals*".

[IHS, 2014] IHS. 2014" *NFC-Enabled Cellphone Shipments to Soar Fourfold in Next Five Years*". [Online] Disponível em: http://press.ihs.com/press-release/design-supply-chain/nfc-enabled-cellphone-shipments-soar-fourfold-next-five-years [Acedido em 03 2014].

[IMA s.r.o, 2013] Institute of microelectronic applications, 2013. "NFCPorter". [Online] Disponível em: http://www.nfcporter.com [Acedido em 03 2014].

[ISO/IEC, 2004] ISO/IEC. 2004. "*Identification cards -- Integrated circuit cards -- Part 5: Registration of application providers*"

[ISO/IEC,2008] ISO/IEC. 2018. "*Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 4: Transmission protocol*"

[ISO/IEC,2011] ISO/IEC. 2011. "*Identification cards -- Contactless integrated circuit cards -- Proximity cards -- Part 3: Initialization and anticollision*"

[ISO/IEC, 2013] ISO/IEC. 2013. "*Information technology -- Telecommunications and information exchange between systems -- Near Field Communication -- Interface and Protocol (NFCIP-1)*"

[Lahtela, et al.,2008] Antti Lahtela, Marko Hassinen, Virpi Jylhä. 2008. "*RFID and NFC in Healthcare: Safety of Hospitals Medication Care*"

[Mulliner, 2009] Collin Mulliner. 2009. "*Vulnerability Analysis and Attacks on NFC-enabled Mobile Phones*".

[NFC Forum Inc., 2009a] NFC Forum Inc.. 2009. "*NFC Forum Type Tags – White Paper"*.

[NFC Forum Inc., 2011] NFC Forum Inc.. 2011. "*NFC in Public Transport"*.

[NFCPorter, 201]

[Ok, et al., 2010] Kerem Ok, Vedat Coskun, Mehmet N. Aydin, and Busra Ozsenizci. 2010. "*Current Benefits and Future Directions of NFC Services*".

[Roland, 2012] Michael Roland. 2012. "*Software Card Emulation in NFC-enabled Mobile Phones: Great Advantage or Security Nightmare?"*.

[Saha, 2008] Amit Kumar Saha. 2008. "*A Developer's First Look At Android*"

[Thomas Erl, 2007] Thomas Erl. 2007. "SOA-*Principles of Service Design*"

[Watchdata, 2011] Watchdata, 2011. "*New Wave In Mobile Payment, Watchdata SIMpass™ Has Done It Again*". [Online] Disponível em: http://www.watchdata.com/press/10212.html [Acedido em 04 2014].

[Watchdata, 2013] Watchdata. 2011. "*Watchdata's SIMpass™ wins"Best Customer Experience"at the 2013 Contactless & Mobile Awards".* [Online] Disponível em: http://www.watchdata.com/press/10266.html [Acedido em 04 2014].

[W3C Working Group, 2004] W3C Working Group, *"Web Services Architecture"* [Online] Disponível em: www.w3.org/TR/ws-arch/

Anexos

Anexo A - Código

```
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {

try {
       card = new String(commandApdu, "UTF-8");
    } catch (UnsupportedEncodingException e) {
       e.printStackTrace();
       }
boolean response = card.contains(":");

if (Arrays.equals(SELECT_APDU, commandApdu)){
sharedpreferences = getSharedPreferences("MyPrefs",Context.MODE_PRIVATE);
resvID = sharedpreferences.getString("resvID", "");
phoneID = sharedpreferences.getString("PhoneID", "");

byte[] reserv = resvID.getBytes();
byte[] phoneid = phoneID.getBytes();
return ConcatArrays(reserv, phoneid, SELECT_OK_SW);
} else if(response){
       String[] str_array = command.split(":");
       String number = str_array[0];
       String roomkey = str_array[1];
       Editor editor = sharedpreferences.edit();
       ArrayList<String> keylist = new ArrayList<String>();
int value = sharedpreferences.getInt("RoomKeyListNumber",0);
                     if (value == 0) {
                            editor.putInt("RoomKeyListNumber", 1);
                            keylist.add(roomkey);
                            saveArray(keylist);
                     } else {
                            keylist = loadArray();
                            if(!keylist.contains(roomkey)){
                                   editor.putInt("RoomKeyListNumber", value+1);
                                   keylist.add(roomkey);
                                   saveArray(keylist);
                            }
                     }
                     editor.putString("ReservationNumber", resvID);
                     editor.commit();

Intent intent = new Intent(CardServiceCheckin.this,HotelMenu.class);
intent.putExtra("ReservationNumber", resvID);
intent.putExtra("RoomNumber", number);
startActivity(intent);
return SELECT_OK_SW;
       }
}
```

Código 13 - processCommandApdu serviço check-in

```
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
if (Arrays.equals(SELECT_APDU, commandApdu)) {
    sharedpreferences =
getSharedPreferences("MyPrefs",Context.MODE_PRIVATE);
    String resvID = sharedpreferences.getString("resvID","");
    byte[] reserv = resvID.getBytes();
    return ConcatArrays(reserv,SELECT_OK_SW);
} else if(Arrays.equals(success, commandApdu)){
 Intent intent = new Intent(CardServiceCheckout.this,MainActivity.class);
 intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
 startActivity(intent);
 return SELECT_OK_SW;
}else{
      return UNKNOWN_CMD_SW;
 }

}
```

Código 14 - processCommandApdu serviço check-out

```
@Override
public byte[] processCommandApdu(byte[] commandApdu, Bundle extras) {
    if (Arrays.equals(SELECT_APDU, commandApdu)) {
    sharedpreferences =
getSharedPreferences("MyPrefs",Context.MODE_PRIVATE);
    ArrayList<String> list = new ArrayList<String>();
    phoneID = sharedpreferences.getString("PhoneID", "");
    String split = "";
    for (int i = 0; i < list.size(); i++) {
             split = list.get(i).toString() + ":";
       }
       byte[] roomk = split.getBytes();
       byte[] phoneid = phoneID.getBytes();
       return ConcatArrays(roomk, phoneid ,SELECT_OK_SW);
    }else{
       return UNKNOWN_CMD_SW;
     }

}
```

Código 15 - processCommandApdu serviço acesso ao quarto

```
public void onTagDiscovered(Tag tag) {
        IsoDep isoDep = IsoDep.get(tag);
        if (isoDep != null) {
            try {
                isoDep.connect();
                byte[] command = BuildSelectApdu(AID);
                byte[] result = isoDep.transceive(command);
                int resultLength = result.length;
                byte[] statusWord = {result[resultLength-2],
result[resultLength-1]};
                byte[] payload = Arrays.copyOf(result, resultLength-2);
                if (Arrays.equals(SELECT_OK_SW, statusWord)) {
                    String card = new String(payload, "UTF-8");
                    mAccountCallback.get().onDataReceived(card);
                }
            } catch (IOException e) {
                Log.e(TAG, "Error communicating with card: " +
e.toString());
            }
        }
    }
```

Código 16 - Método onTagDiscovered

```
public void onDataReceived(final String data, IsoDep tag) {
this.tag=tag;
this.runOnUiThread(new Runnable() {
      @Override
      public void run() {
      String[] str_array = data.split(":");
      JSONObject json = new JSONObject();
      try {
            json.put("RsvID", str_array[0]);
            json.put("PhoneID", str_array[1]);
      } catch (JSONException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
      }
      if(isNetworkAvailable()){
      mAuthTask.execute("POST", getString(R.string.webservicesurl) +
      "/Checkin",json.toString());
      }else{
            Toast.makeText(getApplicationContext(), "No Internet
            Connection", Toast.LENGTH_LONG).show();
            }
      }
      });
}
```

Código 17 - Método onDataReceived na Activity de check-in

Anexo B – Questionário de avaliação da aplicação desenvolvida para o projeto "Accommodation Management System using Mobile Devices"

O projeto "Accommodation Management System using Mobile Devices" foi desenvolvido no âmbito da dissertação para obtenção do Grau de Mestre em Engenharia Informática, na Área de Especialização em Arquiteturas, Sistemas e Redes no Instituto Superior de Engenharia do Porto.

O projeto consistiu em desenvolver uma aplicação móvel cuja finalidade é simular os processos existentes num local de alojamento, tais como fazer reservas, fazer check-in, fazer check-out, entre outros. Este projeto utiliza ainda a tecnologia NFC de modo a que exista uma maior intuição na realização de algumas tarefas. Este questionário tem então como objetivo de avaliar a sua experiência na utilização da aplicação desenvolvida e tem um tempo médio de resposta de 3 minutos.

Obrigado pela colaboração.

1- A aplicação tornaria a minha estadia mais confortável?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

2- A aplicação facilita as valências existentes num local de alojamento? Como por exemplo *check-in*, *check-out*.

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

3- Que importância tem fazer *check-in / check-out* utilizando a aplicação em vez de ter de se dirigir à receção?

Muito Importante	Importante	Pouco Importante	Nada Importante

4- Que importância tem o fato de poder abri a porta do quarto com o *smartphone* em vez de uma chave convencional / *smart card*?

Muito Importante	Importante	Pouco Importante	Nada Importante

5- Dos dois processos de *check-in* / *check-out* existentes qual preferiu?

Utilizando a respetiva opção da aplicação	Colocando o *smartphone* em contacto com o terminal indicado	Qualquer um dos processos

6- A aplicação tem uma interface intuitiva?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

7- A aplicação tem uma velocidade de resposta aceitável?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

8- Sinto que a aplicação é segura?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

9- Da próxima vez que comprar um *smartphone* é provável que escolha um que suporte esta tecnologia?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo

10- Utilizaria o meu *smartphone* para fazer o pagamento da estadia?

Concordo completamente	Concordo	Neutro	Concordo pouco	Não concordo