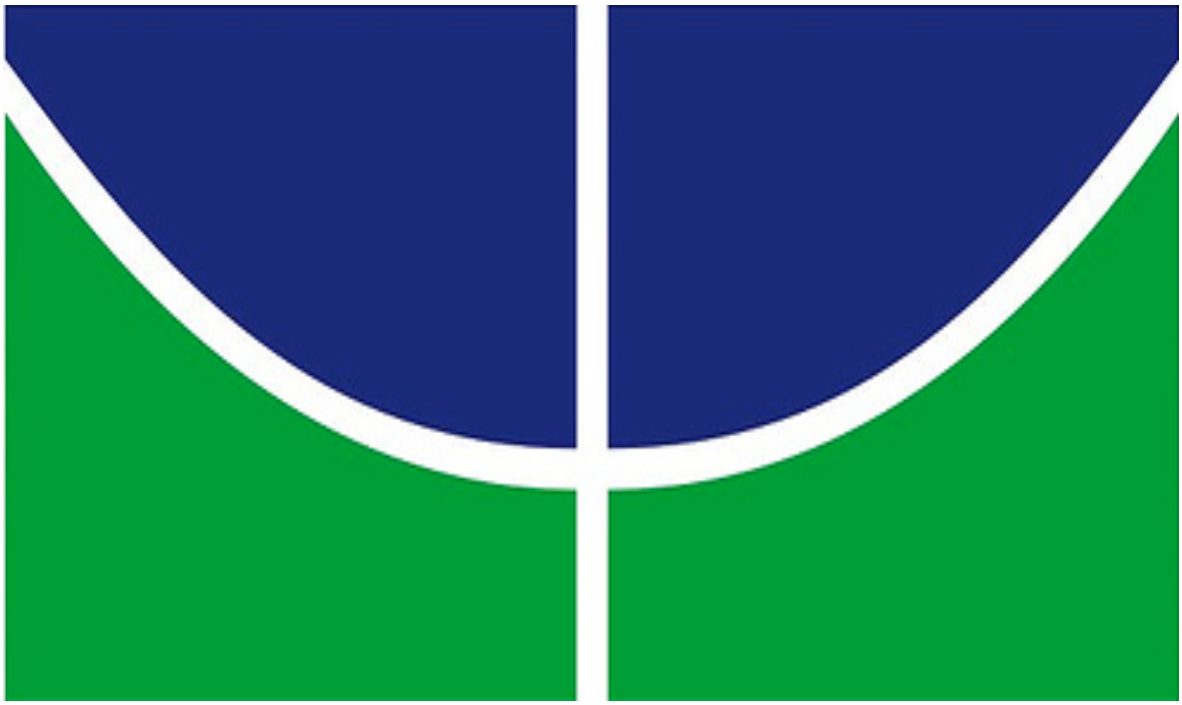

FRAMEWORK UTILIZANDO CÓDIGO 2D PARA PAGAMENTOS COM BASE EM DISPOSITIVOS MÓVEIS

MAURICIO PEREIRA BORGES JÚNIOR

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FRAMEWORK UTILIZANDO CÓDIGO 2D PARA PAGAMENTOS COM BASE EM DISPOSITIVOS MÓVEIS

MAURÍCIO PEREIRA BORGES JÚNIOR

Orientador: Dr. PAULO ROBERTO DE LIRA GONDIM

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO PPGENE.DM - 520/2013
BRASÍLIA/DF,

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FRAMEWORK UTILIZANDO CÓDIGO 2D PARA PAGAMENTOS COM BASE EM DISPOSITIVOS MÓVEIS

MAURÍCIO PEREIRA BORGES JÚNIOR
ORIENTADOR: Dr. PAULO ROBERTO DE LIRA GONDIM

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.
APROVADA POR:

_____

Paulo Roberto de Lira Gondim, Dr., ENE/UNB
(Orientador)

_____

Joel José Puga Coelho Rodrigues, PhD, UBI-PT

_____

Cláudio de Castro Monteiro, D.C., IFTO

BRASILIA/DF

FICHA CATALOGRÁFICA

MAURÍCIO PEREIRA BORGES JÚNIOR

***Framework* Utilizando Código 2D para Pagamentos com Base em Dispositivos Móveis, 2013.**

(UnB, Mestre, Engenharia Elétrica, 2013

Dissertação de Mestrado – Universidade de Brasília.

Faculdade de Tecnologia

Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

BORGES JR, M. P. (2013). *Framework* Utilizando Código 2D para Pagamentos com Base em Dispositivos Móveis. Dissertação de Mestrado em Engenharia Elétrica. Publicação 520/2013, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF.

CESSÃO DE DIREITOS

AUTOR: Maurício Pereira Borges Júnior

TITULO: *Framework* Utilizando Código 2D para Pagamentos com Base em Dispositivos Móveis"

GRAU/ANO: Mestre/2013.

_____

Maurício Pereira Borges Júnior

Universidade de Brasília – UNB

Campus Darcy Ribeiro

Faculdade de Tecnologia – FT

Departamento de Engenharia Elétrica

Brasília – DF

CEP: 70910-900

Agradecimentos

Agradeço, primeiramente, ao meu Deus por me dar a oportunidade de chegar ao final do mestrado. Também agradeço à minha esposa, Sílvia Lôbo Borges por compreender e me ajudar em momentos difíceis. Não posso deixar de agradecer aos meus amigos da UnB, que conviveram comigo durante grande tempo de estudo, bem como à igreja - que orou e agradeceu a Deus em uma só voz por ter conseguido entrar e, para finalizar essa caminhada; ao meu Professor Paulo Roberto de Lira Gondim, pela paciência, compreensão e direcionamento nos estudos.

Não posso deixar de agradecer aos meus pais, por me darem o apoio e ensino para poder chegar até aqui. Aos meus tios, que participaram em grande parte do meu crescimento e amadurecimento; e à minha avó, Durvalina Silva, que, mesmo de longe preocupa-se comigo.

Dedico a

Deus, meus pais, esposa, irmão, sobrinho e amigos pelo apoio, compreensão e suporte.

Resumo

Esta dissertação descreve um *framework* utilizando código 2D para pagamentos com base em dispositivos móveis (por exemplo, terminal celular, *smartphone* e *tablet*).

O trabalho inicialmente apresenta conceitos de códigos 2D, comparações entre os diversos tipos e funcionalidades de pagamentos móveis, tais como serviço de confirmação por SMS e email.

O *framework* proposto utiliza serviços de diferentes provedores, nas áreas de telecomunicações, logística, pagamento, segurança, autoridade certificadora e outros, permitindo popularizar o pagamento móvel no Brasil, ainda pouco explorado.

A implementação inclui a construção de uma aplicação *Web* móvel desenvolvida em C#, HTML5 e JavaScript, apresentada como prova de conceito. Tal aplicação utiliza o .NET *framework* Microsoft para construção do código, jQuery para *interface* gráfica do usuário e *Web Services* para comunicação entre plataformas. Usamos também o protocolo SSL(*Security Socker Layer*) a fim de proporcionar segurança, junto do padrão avançado de criptografia (AES) *Advanced Encryption Standard* e o protocolo HTTP para compor o *m-Payment*.

O *framework* possibilita também o comércio eletrônico utilizando o Sistema Brasileiro de TV Digital, possibilitando ao usuário pagar usando a sua carteira virtual (celular, *smartphone* ou *tablet*) em frente ao aparelho de TV.

Dessa forma tem-se a convergência de modelos de comércio virtual, capaz de realizar transações comerciais independentes de dispositivos. O uso de *Web Services* possibilitou a comunicação e integração com outras plataformas como a de TV Digital, por meio do protocolo SOAP.

Palavras-chave: Código 2D, Mobilidade, Carteira Virtual, Segurança, Criptografia, TV Digital.

ABSTRACT

This paper describes a framework for 2D code using payments with your mobile device (mobile terminal, smartphone and tablet).

The work initially presents concepts of 2D code, making comparisons between different types and functionalities of mobile payments, such as confirmation service by SMS or email.

The proposed framework utilizes services from different providers in the areas of telecommunications, logistics, payment, security, certificate authority and others, allowing popularize mobile payment service in Brazil, still unexplored. We also use SSL to provide security, with the Advanced Encryption Standard (AES) *Advanced Encryption Standard* and the HTTP protocol to compose the m-Payment.

This implementation of the framework includes building a mobile web application developed in C#, JavaScript and HTML5, presented as proof of concept. This application uses the. NET Microsoft framework for building code, jQuery for graphical user interface and Web Services for communication between platforms.

The framework also permit to make commerce transactions using Brazilian Digital TV System, allowing the user to be able to pay using your virtual wallet (mobile, smartphone or tablet) in front of the TV set.

Thus there has been a convergence of virtual commerce models, able to perform commercial transactions independently from devices. The use of Web Services enabled communication and integration to other platforms such as Digital TV, using the SOAP protocol.

SUMÁRIO

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Termos

AES	*Advanced Encryption Standard*
AIM	*The Association for Automatic Identification and Mobility*
ATM	*Automated Teller Machine*
CA	*Certification Authority*
CASE	*Computer Aided Software Engineering*
C#	Linguagem de Programação Orientada a Objetos – CSharp
CCD	*Charge-Coupled Device* Estático
CDMA	*Code Division Multiple Access*
CMOS	*Complementary Metal-Oxide* Semiconductor
COAF	Conselho de Controle de Atividades Financeiras
DES	*Data Encryption Standard*
DSP	Digital *Signal Processor*
ECI	*Extended Channel Interpretation*
EDI	*Eletronic Data Interchange*
GPRS	*General Packet Radio Services*
GSM	*Global System for Mobile*
HSM	*Hardware Security Module*
HTML	*HyperText Markup Language*
HTML5	*HyperText Markup Language version* 5
HTTP	*Hypertext Transfer Protocol*
HTTPS	*Hypertext Transfer Protocol Secure*
IBGE	Instituto Brasileiro de Geografia e Estatística
ISO	*International Organization for Standardization*
JIS	*Japanese Industrial Standards*
J2ME	*Java Platform Micro Edition*
LCD	*Liquid Crystal Display*
MCAT	*Mobile CreditCard with Analytical Transaction*
MCDM	Modelo de Plataforma de Pagamento
MD5	*Message Digest Algorithm 5*
MIN	*Merchant Identification Number*
MSISDN	*Mobile Subscriber Integrated Services Digital Network Number*
MTD	*Mobile Telephony Device*
MVC	*Model View Controller*
NFC	*Near Field Communication*
OTA	*Over The Air*
OO	Orientação a Objetos
OS	*Operation System*
PIN	*Personal Identification Number*
POS	*Point-of-Sale*
P2P	*Peer to Peer*
QR Code	*Quick Response Code*
RFID	*Radio Frequency Identification*
RIM	*Research in Motion*
RPC	*Remote Procedure Calls*
SET	*Secure Electronic Transaction*
SIM	*Subscriber Identity Module*
SHA-1	*Secure Hash Algorithm 1*
SMS	*Short Message Service*

SOA	*Service Oriented Architectures*
SOAP	*Simple Object Access Protocol*
SSL	*Security Socker Layer*
STK	*(Subscriber Identity Module) Tool Kit*
TTP	*Trusted Third Party*
UDDI	*Universal Description, Discovery and Integration*
UIN	*User Identification Number*
UML	*Unified Modeling Language*
UMTS	*Universal Mobile Telecommunications System*
UnB	Universidade de Brasília
USSD	*Unstructured Supplementary Service Data*
URL	*Uniform Resource Locator*
VCC	Cartões de Crédito Virtual
WAP	*Wireless Application Protocol*
WIB	*Wireless Internet Browser*
WCSS	*Wireless Application Protocol Cascading Style Sheet*
WLAN	*Wireless Local Area Network*
WSDL	*Web Services Description Language*
WWW	*World Wide Web*
W3C	*World Wide Web Consortium*
XHTML	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*

Capítulo 1

1. Introdução

Desde os tempos primitivos o ser humano utiliza o comércio para trocar produtos ou serviços, como na época do "escambo". Com o passar do tempo, o comércio por meio de troca indireta (usando dinheiro) foi introduzido em toda parte do mundo, crescendo e sendo implementado na forma de supermercados, lojas, empresas prestadoras de serviços, e centros comerciais como os shoppings.

Com o avanço e a popularização da Internet, o comércio começou a se estender para o mundo virtual, utilizando a *WWW* (*World Wide Web*). As mesmas lojas físicas começaram a fazer os seus próprios sites para vender produtos pela Internet, caracterizando o *e-Commerce* ou comércio pela Internet.

A tecnologia continuou evoluindo, a rede de comunicação móvel passou por diferentes gerações (1G, 2G, 3G), permitindo o aumento de recursos colocados à disposição dos usuários.

Com o aumento da popularidade da Internet no dispositivo móvel usando rede 3G e/ou Wi-Fi (*Wireless Fidelity*), surgiu a possibilidade de fazer pagamento eletrônico usando terminais que hoje cabem na palma da mão, nomeado de *m-Payment* ou comércio móvel.

Nesta dissertação, considera-se um *framework* como sendo um conjunto de classes que incorpora um projeto abstrato de soluções para uma família de problemas relacionados [Johnson et al., 2011], o qual pode ser denominado como arcabouço, estrutura e esqueleto.

Tendo em vista toda a evolução do comércio, crescimento da Internet e tecnologia de redes de comunicação, a presente dissertação descreve um *framework* para provimento de pagamentos com base em dispositivos móveis, possibilitando também a comunicação entre diversas plataformas e tecnologias incluindo a TV Digital (*T-Commerce*) em junção do *m-Payment* e se tornando o *u-Commerce – Universal Commerce*.

Dessa forma a proposta aqui apresentada serve como base para realização de pagamentos, visando a popularização de serviços de comércio móvel no Brasil que ainda não é muito explorado.

O *framework* proposto utiliza tecnologia HTML5 *web-browser*, QR-Code (*Quick Response Code*) para leitura de códigos, provedor de serviço de pagamento, protocolo SSL e *Web Services* para comunicar e disponibilizar funcionalidades a outros

aplicativos. Dessa forma, é caracterizado como baseado em *browser*, facilitando o uso e acesso a serviços da Internet.

O trabalho realizado envolve também a utilização do *Framework* .NET Microsoft, que permite o uso de temas e a adaptação automática da *interface* de diferentes tamanhos ou até mesmo em aparelhos de TV Digital, integrando e dando continuidade a dissertação anteriormente defendida na Universidade de Brasília [Filho 2011].

1.1 Objetivos

1.1.1 Geral

Propor e desenvolver um *framework* para provimento de pagamentos pelo dispositivo móvel, que permita a integração com Internet e TV Digital.

1.1.2 Específicos

A) pesquisar sobre os aplicativos e ferramentas existentes de pagamento móvel e fazer comparações;

B) elicitar requisitos funcionais e não funcionais a serem atendidos por um *framework*, a ser utilizado para comércio eletrônico via dispositivo móvel;

C) propor e implementar um *framework* que atenda aos requisitos citados, incluindo emprego de padrões de serviços *Web*, *Web Services* e de TV Digital;

D) estender o *framework* de pagamentos, permitindo o tratamento centralizado de características visuais, dando suporte a múltiplos dispositivos com diferentes resoluções de tela, como TV's e dispositivos móveis.

1.2 Justificativa

A justificativa apareceu diante de uma necessidade tecnológica para tentar solucionar problemas de pagamento móvel, levando em conta segurança de dados, a rapidez e a facilidade de integração em outros ambientes.

Com o *framework* desenvolvido, torna-se possível pagar usando o dispositivo móvel em sites de comércio eletrônico e sistemas de TV Digital, com base na utilização de equipamento de leitor de código 2D.

Como o Brasil possui diversos casos de sucesso no mercado de comércio eletrônico em larga escala, a disponibilização de aplicações para pagamento móvel é uma tendência natural, podendo aumentar consideravelmente as vendas das empresas do

ramo, além de ser uma nova plataforma de *m-Payment* para os usuários com equipamentos que cabem na palma da mão.

Com base em pesquisa do IBGE 2010 (Instituto Brasileiro de Geografia e Estatística), a região com maior percentual de pessoas com celular (64,3%) é o Distrito Federal [IBGE 2010]. Na opinião de Jorge Marinho (presidente da *m-Pay*), o Brasil tem potencial para oferecer opções de *m-Payment* porque a imensa base de usuários de telefones celulares é superior à de usuários do sistema bancário [Dantas 2010].

O problema ocorre devido à falta de estrutura confiável, simples e rápida para o mecanismo de pagamentos genéricos utilizando dispositivos móveis. Hoje a população não possui um separador móvel de recursos para compras e recursos diversos, o *framework* traz além de tudo, um gerenciador *on-line* de gastos.

Nessa percepção e com o objetivo de atingir classe social, propomos um *framework* focado em pagamentos. A combinação de QR Code (*Quick Response Code*), *Web*, acessibilidade e portabilidade de dispositivo móvel garante mais confiança e satisfação [Gebrekristos et al., 2008]. Como será observado ao longo deste estudo, o número crescente de aparelhos com câmera digital pode facilitar a popularização no uso do pagamento móvel no Brasil usando código 2D [Brambilla et al., 2010].

1.3 Metodologia

Para o desenvolvimento da presente dissertação foi feito levantamento bibliográfico sobre as tecnologias, linguagens, ferramentas e trabalhos relacionados para nortear a implementação do *framework* de pagamento.

Utilizamos tecnologias de comunicações móveis (como o SMS) e realizamos análise e desenvolvimento seguindo o paradigma de orientação a objetos, com a UML (*Unified Modeling Language*) para modelagem e o Visual Studio para desenvolvimento.

Utilizamos os protocolos *SOAP*, *HTTP* e *SSL* para integração e segurança; escolhidos por terem o padrão regulamentado pelo *W3C* (*World Wide Web Consortium*); o padrão *MVC (Model View Controller)* foi utilizado com objetivo de reutilizar classes e componentes, uma vez que o *framework* de pagamentos necessita de comunicação com servidores *Web* por meio de conexão à Internet.

No processo de desenvolvimento em cascata, foram seguidas as seguintes etapas:

A) especificação de requisitos - foram definidos os requisitos funcionais e não funcionais, que são apresentados ao longo do trabalho;

B) projeto - foi adotado o paradigma de Orientação a Objetos (OO), utilizando-se a *Unified Modeling Language* (*UML*) para modelagem, com o auxílio de ferramenta CASE (*Computer-Aided Software Engineering*);

C) implementação - a implementação seguiu o paradigma de orientação a objetos, conforme definido na fase do projeto;

D) testes - foram feitos testes de interoperabilidade, integração e pagamentos para verificar se os componentes do *framework* estavam se comunicando conforme o esperado.

Os requisitos do *framework* foram levantados tomando por base as funcionalidades existentes na grande maioria dos sistemas de comércio eletrônico disponíveis na Internet e difundidos no Brasil.

1.4 Principais Contribuições

As principais contribuições desta dissertação foram:

A) Realização de levantamentos sobre vários tipos de pagamentos móveis e ferramentas existentes no mercado;

B) Criação de um *framework* para pagamentos móveis usando código 2D, a fim de interagir com outros tipos de aplicativos/sistemas como o da TV Digital, facilitando e simplificando a implementação de funcionalidades;

C) Implementação de código 2D de forma criptografada usando esteganografia na *URL (Uniform Resource Locator)*, podendo ser visualizado em qualquer tipo de dispositivo móvel e TV Digital;

D) Integração à Internet utilizando padrões de serviços *Web Services*.

1.5 Organização do Trabalho

O Capítulo 2 apresenta a revisão bibliográfica das tecnologias empregadas e relacionadas ao desenvolvimento do trabalho.

O Capítulo 3 trata aspectos do *QR Code* (código 2D), normas e noções básicas.

O Capítulo 4 apresenta o desenvolvimento, validação e métricas do *framework* para provimento de pagamentos.

O Capítulo 5 apresenta as conclusões e trabalhos futuros.

Por fim, os apêndices mostram parte dos códigos fonte utilizados para desenvolvimento do *framework*.

Capítulo 2

2. Revisão Bibliográfica

Este capítulo fornece noções básicas sobre comércio eletrônico, sistemas de *m-payment*, plataformas de redes sem fio para *m-payment*, *web services*, segurança e, por fim, analisa alguns trabalhos relacionados.

2.1 Comércio Eletrônico

No início da Internet (*World Wide Web*) (comumente denominada *Web*), o conteúdo era estático permitindo pouca ou nenhuma interação com o usuário. Com a crescente demanda de troca de informações entre empresas, o advento de padrões abertos de comunicação e a necessidade de alcançar novos clientes e mercados; surgiu um novo gênero de *Web* sites na década de 90 voltadas para comércio eletrônico [Chu et al., 2007].

Tais *Web* sites possibilitaram a realização de negócios entre compradores, vendedores e seus parceiros. Desta forma surgiu o *e-Commerce*, baseado na possibilidade de realização de transações eletrônicas. Segundo os autores de [Veijalainen et al., 2006]: uma transação eletrônica é uma venda ou compra de produtos ou serviços, entre empresas, familiares, indivíduos, governos e outras organizações públicas ou privadas, conduzidas por redes mediadas por computador.

A troca de informações entre essas empresas era feita por meio de *Eletronic Data Interchange* (EDI) (troca eletrônica de informações), no entanto, requeria realização de acordos entre as organizações [Chu et al., 2007], o que poderia dificultar tais parcerias. O advento de padrões abertos como o XML permitiu a evolução de tais processos de intercâmbio de dados e integração de informações entre empresas.

Atualmente existem diversas empresas consolidadas na área de comércio eletrônico. Algumas nasceram na era digital e vendem exclusivamente pela Internet; outras evoluíram para essa nova era, alcançando novos clientes, mercados nacionais e internacionais.

Com a recente popularização de dispositivos móveis e da Internet sem fio de grande abrangência (por exemplo, as tecnologias de comunicação móvel de 3a geração como o *Universal Mobile Telecommunications System* - UMTS), surgiram novas oportunidades para as lojas de comércio eletrônico.

Estas entram em uma nova era, com novas perspectivas de captação de clientes e mercados. Com isto surgem novas tendências como o Comércio Móvel (*M-Commerce*) onde as transações são feitas utilizando dispositivos e redes de acesso móveis, tais como *Wireless Local Area Networks* (WLAN's), redes de telecomunicações 2G ou 3G, conexões *Bluetooth* ou infravermelho [Veijalainen et al., 2006].

Tais dispositivos móveis permitem que os usuários possam realizar compras em qualquer lugar que eles estejam, até mesmo em suas horas livres, durante o trajeto para o trabalho, ou no horário de almoço, por exemplo.

Outro ponto importante em relação à evolução das tecnologias móveis é o crescimento do número de usuários nos últimos anos, tornando assim, cada vez maior, a aplicabilidade de soluções que se baseiam no emprego de tais tecnologias.

2.2 Classificação de Sistemas de *m-Payment*

Nesta seção, classificamos os sistemas de pagamento móvel baseados nos artigos dos autores citados a seguir.

2.2.1 Classificação proposta em [Mader 2011]

De acordo com [Mader 2011] o *m-Payment* pode ser dividido em duas categorias:

A) *m-Payment* baseado em *browser*: envolve aplicações que utilizam o navegador do dispositivo móvel para acesso a um *Web* site otimizado, muitas vezes, um site de *e-Commerce* o qual será utilizado para procurar, pesquisar, selecionar, salvar, comprar e pagar algum produto ou serviço. O *browser* móvel tem se tornado uma ferramenta muito poderosa e ao mesmo tempo de simples manuseio.

B) *m-Payment* baseado em aplicação local: envolve aplicações ou aplicativos que precisam ser baixados, transferidos e executados nativamente. O aplicativo cliente geralmente pode navegar, pesquisar, buscar, selecionar, salvar, comprar e pagar a partir do ambiente criado pelo aplicativo.

m-Payment baseado em browser

A Internet móvel é a forma como os consumidores interagem com muitas oportunidades de comércio sobre os seus dispositivos móveis. Planos ilimitados de dados estão disponíveis a partir de certas operadoras sem fio, de modo que o custo para os consumidores para o uso da internet móvel pode ser baixo, e o uso da *Web* no móvel está se tornando cada vez mais comum.

Para os proprietários de smartphones, usar a *Web* móvel é quase o mesmo que usar um navegador da *Web* baseado em PC. Mas o usuário típico de hoje é, provavelmente frustrado, pois o acesso à *Web* móvel atualmente sofre de problemas de interoperabilidade e usabilidade.

Esses problemas decorrem da fragmentação de plataformas de dispositivos móveis, sistemas operacionais móveis e da variedade de navegadores da *Web* encontrados em dispositivos diferentes.

Desafios são encontrados em diferentes resoluções de tela, tamanhos de tela, velocidade de entrega de conteúdo, e a grande variedade de métodos de entrada do usuário.

Hoje é difícil para o varejista projetar e gerenciar uma variedade de modelos *Web* móveis que funcionam bem em todos os navegadores e em todas as plataformas e telefones. Em alguns casos, pode ser tão difícil suportar todos os navegadores quanto suportar aplicações nativas para diferentes sistemas operacionais móveis.

Existem padrões com os quais a maioria ou todos os telefones móveis cumprem (especificamente, XHTML e WCSS) que são muito úteis para a implementação de diferentes plataformas. Além disso, alguns vendedores oferecem tecnologia capaz de otimizar ainda mais a experiência de navegação.

As figuras 1 e 2 mostram duas abordagens populares para permitir o *m-Payment*. A figura 1 ilustra uma abordagem que aproveita serviços comuns de comércio eletrônico, otimizados para determinado dispositivo do consumidor móvel. Esta abordagem é considerada a melhor prática [Mader 2011].

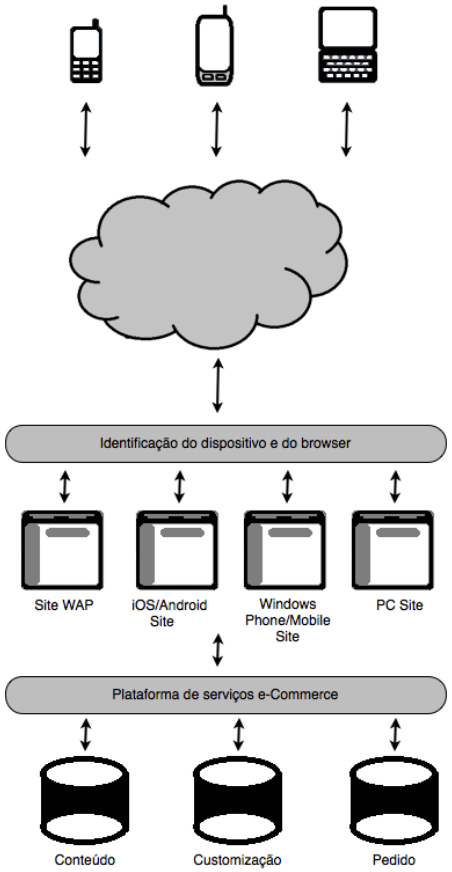

Figura 1 - Melhores práticas Web Móvel - Arquitetura alto nível
Fonte: [Mader 2011]

A figura 2 ilustra uma abordagem em que o conteúdo específico é fornecido para cada tipo de dispositivo móvel. Enquanto esta abordagem é eficaz, não é eficiente e pode levar a inconsistência. Portanto, essa abordagem não é recomendada como um projeto de longo prazo.

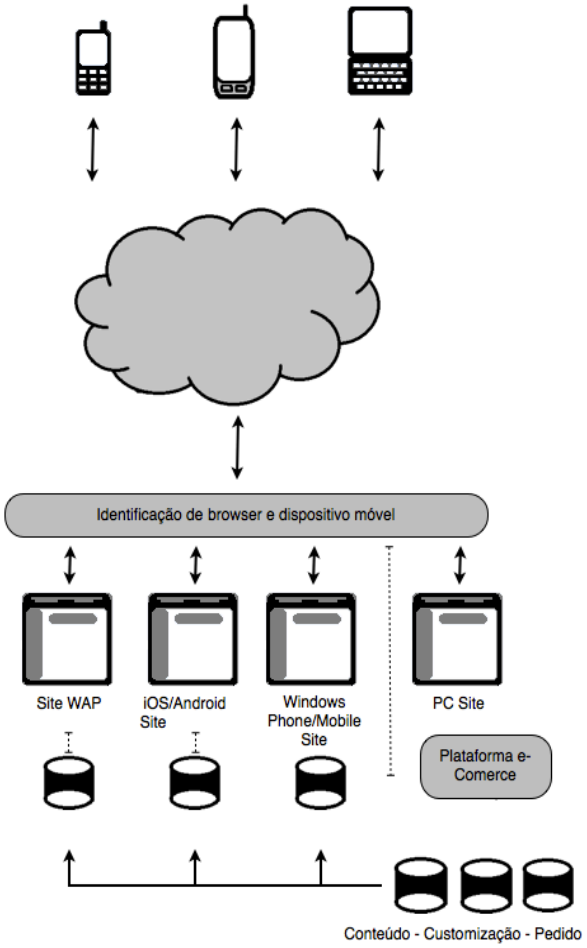

Figura 2 - Típico Web móvel - Arquitetura alto nível
Fonte: [Mader 2011]

O dispositivo móvel acessa uma URL através da Internet que identifica o tipo do dispositivo e redireciona para a melhor *interface* específica criada, a fim de renderizada a forma mais agradável possível. Cada conteúdo é customizado para que o pedido de produto ou serviço seja feito sem apresentação de problemas ao usuário, conforme mostrado na figura 2.

m-Payment baseado em aplicação local

Grande parte do entusiasmo atualmente centra sobre a utilização de aplicações *m-Payment*. Essas aplicações têm vantagens e desvantagens em comparação com os móveis baseados na *Web*.

As aplicações de *m-Payment* oferecem alguns benefícios importantes, mais do que o baseado na *Web*:

• Experiência do cliente e desempenho melhorados
• Recursos adicionais de compras
• Capacidade de alavancar as capacidades-chave dos dispositivos
• Navegação offline

Grande parte do conteúdo necessário para criar a experiência do cliente pode ser pré-carregado com a aplicação e até mesmo atualizada periodicamente (por exemplo, quando a presença de uma conexão Wi-Fi é detectada).

A interação cliente-vendedor pode ser enriquecida, e o cliente pode ter uma experiência mais envolvente, sem esperar o *download* de páginas de atualização e conteúdo.

Aplicações locais permitem que o varejista use conteúdo rico, como o vídeo, grandes imagens com *zoom*, ou catálogos virtuais. Algumas características comerciais adicionais disponíveis através de aplicativos podem incluir elementos como auto completar nas buscas, classificação e redes sociais.

Estas características levam ao emprego de largura de banda intensiva e podem ser de difícil implementação na *Web* móvel.

Como os aplicativos devem aproveitar as capacidades do dispositivo, os varejistas podem usá-los para fornecer ao cliente a localização de conteúdo, localizadores de lojas, ofertas promocionais, integrações, mapa e notificações que informam o cliente de mudanças e ofertas.

Câmeras em dispositivos móveis podem desempenhar um papel fundamental em aplicações móveis, leitura de códigos de barra e QR *Code*, tirar fotos de produtos para compartilhar socialmente e mais.

As aplicações podem incluir recursos como integração com uma lista de contatos ou um livro de endereços, dando aos consumidores acesso fácil a informações de endereço durante as compras.

Navegação offline é fundamental para os usuários de dispositivos que não têm acesso celular, como o *iPod Touch*, ou usuários que queiram interagir enquanto não estiverem em uma rede.

Seleção de plataforma

A escolha de plataforma depende do mercado de varejo e de dados demográficos dos clientes, e oferece suporte a múltiplas plataformas pode ser caro e necessitar de apoio extra. Atualmente, a escolha de uma plataforma pode envolver aspectos tais como:

• iOS, da Apple está agora em versão 6.0, e a versão 6.1 estará disponível em um futuro próximo.

• Google Android OS tem menos de 3 anos de idade, portanto requer maturação.

• Várias versões da *Research In Motion* (RIM) OS estão em distribuição para diversos dispositivos.

• Symbian, enquanto liga o maior número de dispositivos no mundo, não tem um kit de desenvolvimento de *software* eficaz e pouco mercado para distribuição.

Considerações futuras

Uma distinção entre a aplicação *Web* móvel e aplicação local é que em breve os browsers terão acesso direto ao *hardware* do dispositivo usando padrões HTML5 e OMTP. São padrões adotados que favorece o desempenho de aplicativos baseados em navegador.

Armazenamento persistente por meio de especificações do HTML5 e o acesso a funções de *interface* podem reduzir ainda mais a necessidade para o desenvolvimento focado em cada plataforma, as aplicações locais.

Essas mudanças vão melhorar a capacidade dos varejistas, e vão apoiar o desenvolvimento do *software* baseado em navegador *Web*.

2.2.2 Classificação proposta em [Gao, 2009]

De acordo com [Gao, 2009], os sistemas de pagamento móvel podem ser de dois tipos:

a) *M-Payment* baseado em conta
b) *M-Payment* baseado em carteiras móveis

2.2.2.1 Sistema de *m-Payment* baseado em conta

De acordo com [Gao, 2009] cada cliente é associado a uma conta específica mantida por uma terceira parte como um banco (ou uma empresa de telecomunicações). Nas operações pré-pagas, essa conta será diretamente vinculada à conta poupança do consumidor. O consumidor mantém saldo positivo da conta que é debitado quando uma operação de pré-pagamento é processada. Se as operações pós-pagas são suportadas, os encargos de uma transação são acumulados na conta do consumidor. As contas do consumidor são, então, periodicamente faturadas e pagas debitando do saldo da conta pela TTP (*Trusted Third Party*).

A) *M-Payment* baseado em telefone móvel

Permite ao cliente comprar e pagar por produtos ou serviços através de telefones celulares. Aqui, cada telefone celular é usado como ferramenta de pagamento pessoal em ligação com vendas à distância utilizando SMS. O sistema de pagamento móvel usando cartão de crédito tem uma vantagem sobre o pagamento tradicional, devido à placa do terminal que pode ser diminuída ou substituída pelo dispositivo móvel e o pagamento pode ocorrer em qualquer lugar [Lin et al., 2000] [Huang et al., 2002].

M-Payment* baseado em *Smart Payment

É usado o cartão inteligente, circuito integrado que contém microprocessador e memória, juntamente com o sistema operacional. Esses cartões inteligentes podem ser utilizados para identificação eletrônica, assinatura eletrônica, criptografia, pagamento e armazenamento de dados [Gao et al., 2006a] [Gao et al., 2006b]. Existe um problema nesse tipo de *m-Payment*, pois em situação de perda do aparelho os dados permanecem armazenados, dificultando os aspectos de segurança a ele [Zhang et al., 2004] [Saxena et al., 2005].

B) *M-Payment* baseado em cartão de crédito

Este tipo de sistema de pagamento móvel permite ao cliente fazer pagamentos em dispositivos móveis usando seus cartões de crédito. Estes sistemas são desenvolvidos com base na atual infraestrutura de cartão de crédito, adicionando a capacidade de pagamento sem fio para os consumidores em dispositivos móveis [Yang Li et al., 2012]. Existe o protocolo seguro chamado SET, desenvolvido pela Visa e MasterCard para a transferência segura de transações com cartões de crédito. A nossa proposta tem por objetivo utilizar esse protocolo para pagamento seguro. Um exemplo de sistema desse tipo é apresentado em [Fourati et al., 2002].

***M-Payment* baseado em *Mobile POS* (*Point-of-Sale*)**

O sistema de pagamento *Point-of-Sale* permite ao cliente comprar produtos em lojas de varejo usando dispositivo móvel. Existem dois tipos populares de sistemas móveis:

a) realização de pagamentos automatizados;
b) participação de pagamento no ponto de venda.

a) Realização de pagamentos automatizados:

É frequentemente usado em máquinas de ATM, máquinas de vendas automáticas, parquímetros, coletores de impostos, e máquinas de bilhetes que permitem aos usuários comprarem bens moveis, lanches, bilhetes de cinema e estacionamentos.

b) Participação de pagamento no ponto de venda:

Esse sistema é útil para balcões de lojas e táxis. [Gao et al., 2005] apresentam um sistema de pagamento móvel P2P que permite os usuários móveis utilizarem seus dispositivos como ponto de venda, além de poder se comunicar entre si usando operações seguras de pagamento [Gao et al., 2006a] [Gao et al., 2006b].

2.2.2.2. Carteiras móveis ou *m-Wallet*

Carteiras móveis são do tipo mais popular da opção pagamento via celular. Elas permitem que o usuário armazene informações de faturamento e compras. Os principais tipos de carteiras móveis no mercado são: [Gao et al., 2006a] [Gao et al., 2006b].

a) Carteira de clientes

As carteiras dos clientes são armazenadas no dispositivo do usuário na forma de cartão SIM *Toolkit Application*. A carteira baseada em *hardware* tem alguns problemas, como a difícil atualização de informações potencialmente confidenciais do usuário e a informação financeira que fica comprometida se o dispositivo for perdido ou roubado.

b) Carteira de hospedagem

Carteiras de hospedagem referem-se a carteiras digitais hospedadas em um servidor. Isso dá ao prestador de serviços um controle muito maior sobre a funcionalidade que ele oferece à segurança dos dados e transações. A carteira de hospedagem pode ser auto-hospedada ou hospedada por terceiros. Além disso, o servidor de hospedagem *m-Wallet* utiliza a tecnologia SET, citada anteriormente.

A nossa proposta tem o objetivo de agregar o melhor das tecnologias citadas, adicionando simplicidade e confiabilidade do usuário.

2.3 Plataformas de Rede Sem Fio para *M-Payment*

A tecnologia móvel provê várias possibilidades para a implementação do *m-Payment*. Primeiro, essencialmente, um telefone GSM pode enviar e receber informações por meio de serviços de dados móveis com base em oito possibilidades de serviços - SMS, USSD, WAP/GPRS, aplicativo baseado em telefone, aplicativo baseado em SIM, telefone dual chip, telefone dual slot e SRCN (Rede de Comunicação de curto alcance). Iremos aqui apresentar as cinco primeiras, indicando ao leitor interessado a consulta a [Tehrani et al., 2010].

2.3.1 *Short Message Service* (SMS) Puro

Este é um serviço de mensagem de texto habilitado para curtas mensagens (140-160 caracteres) que podem ser transmitidas pelo dispositivo móvel. Mensagens curtas são armazenadas e enviadas pela central de SMS. O SMS possui um canal de acesso no dispositivo, diferente do canal de voz (Valcourt; Robert; Beaulieu, 2005). Pode ser usado para prover informações sobre o status da conta (método de informação) como banco ou para transmitir instruções de pagamento.

Este canal não é muito seguro porque não utiliza criptografia referente ao recebimento ou envio de dados [Dai et al., 2011]. A figura 3 apresenta uma plataforma para *M-Payment* utilizando SMS. Detalhes a respeito das funcionalidades dos componentes de uma rede GSM (como MSC, HLR, AuC) podem ser encontrados em [Kurose 2010] e [Tehrani et al., 2010].

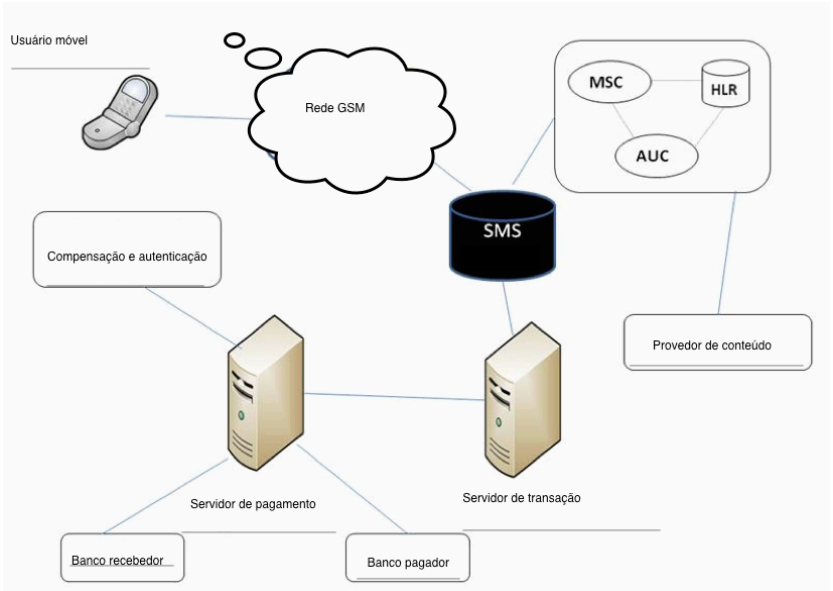

Figura 3 – Plataforma de *m-payment* baseado em SMS
Fonte: [Tehrani et al., 2010]

2.3.2 *Unstructured Supplementary Services Data* (USSD)

Unstructured Suplementary Service Data (USSD) é uma tecnologia única do GSM. Tem a capacidade de construir, dentro do GSM, um padrão para suporte transmitindo informação com o canal de sinalização da rede. O USSD provê uma sessão base de comunicação permitindo uma variedade de aplicativos, mantendo ativa uma transação enquanto o serviço SMS é utilizado para envio. O tempo de resposta para aplicações interativas é mais curto em relação ao SMS. A figura 4 apresenta uma plataforma de *M-Payment* usando USSD.

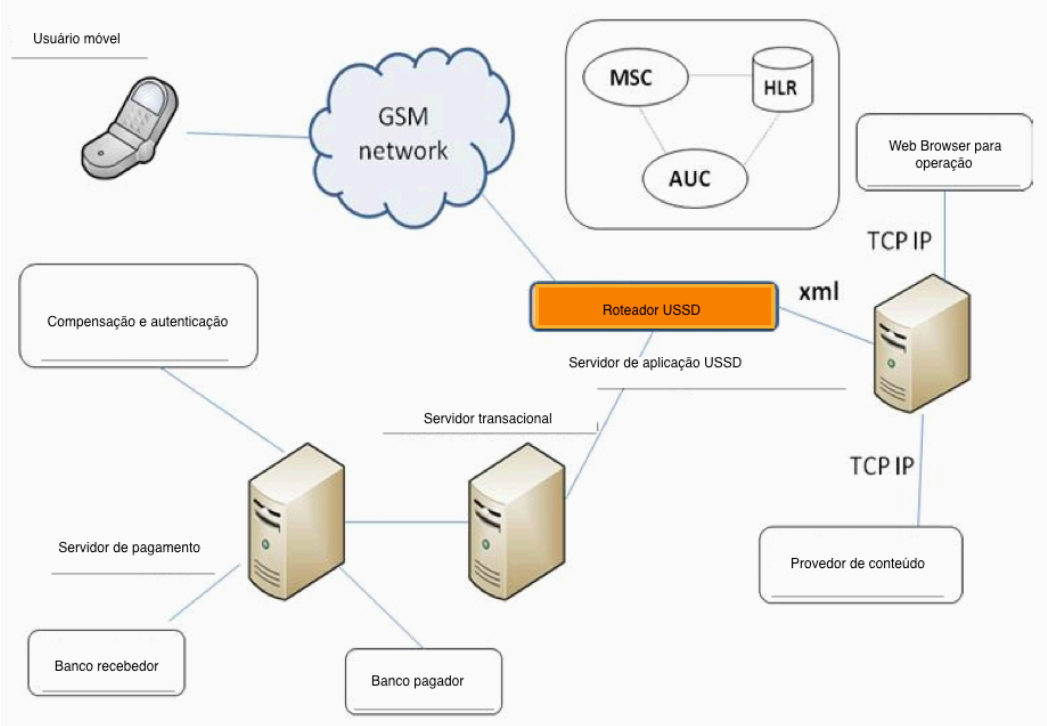

Figura 4 – Plataforma de *m-payment* baseado em USSD
Fonte: [Tehrani et al., 2010]

2.3.3 Wireless Application Protocol (WAP) / General Packet Radio Service (GPRS)

O GPRS é um serviço de dados móvel disponibilizado para usuários GSM. O GPRS provê pacotes de dados para a rede GSM e permite acesso ao WAP, bem como *Mulltimedia Messaging Service* (MMS) para serviços de comunicação de Internet, assim como e-mail e *Web* site móvel. Muitos desenvolvedores estão utilizando a Internet para o uso do pagamento móvel usando 3G, 4G ou Wi-Fi. A figura 5 apresenta uma plataforma de *M-Payment* usando WAP/GPRS.

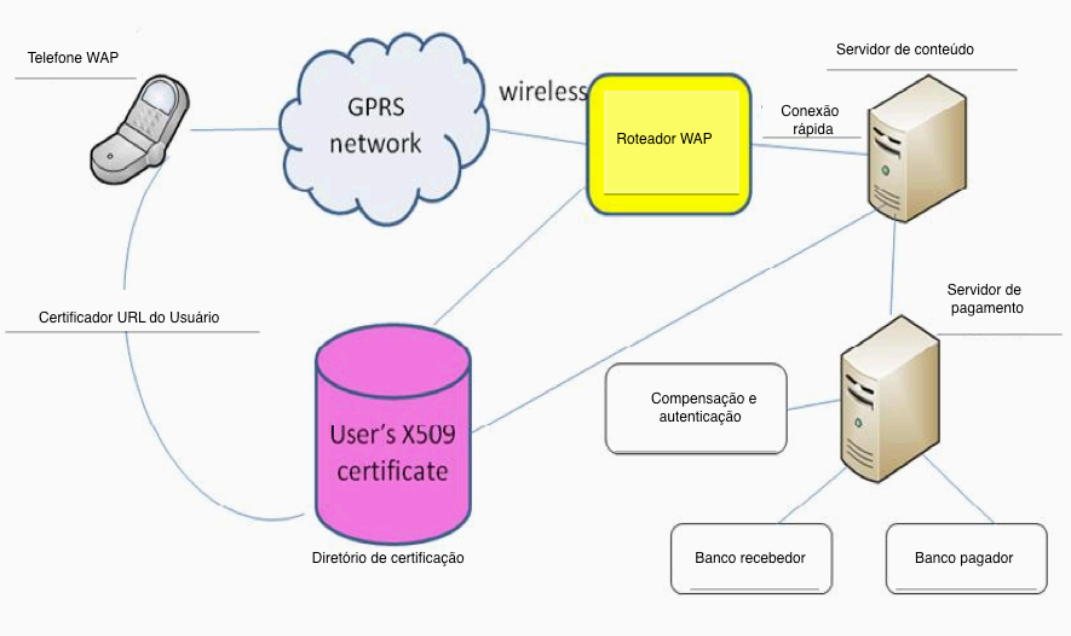

Figura 5 – Plataforma de *m-payment* baseado em GRPS/WAP
Fonte: [Tehrani et al., 2010]

2.3.4 Plataforma de aplicação baseada no telefone móvel

Nesta plataforma o *software* de pagamento é instalado no dispositivo móvel e a operação de pagamento é feita através do *software*. Os canais de comunicação (SMS, USSD e WAP) são necessários para a transferência de informações de pagamento. Dependendo de qual canal é utilizado, os serviços de segurança, custo e acessibilidade serão diferentes.

J2ME e BREW são ferramentas para o desenvolvimento de *software* de pagamento, de acordo com a rede GSM ou CDMA respectivamente. As desvantagens desta plataforma incluem: aplicabilidade apenas com Java habilitado no telefone, atualização manual, duplicidade na instalação e em caso de troca de telefone versões diferentes serão necessárias. No entanto, as vantagens da utilização de J2ME são segurança fim a fim de criptografia de conteúdo, e melhoria do uso da largura de banda de rede.

Alguns dos serviços móveis, que podem ser fornecidos por esta plataforma incluem a operação financeira e reservas (restaurante, bilhete, etc.), dentre outros.

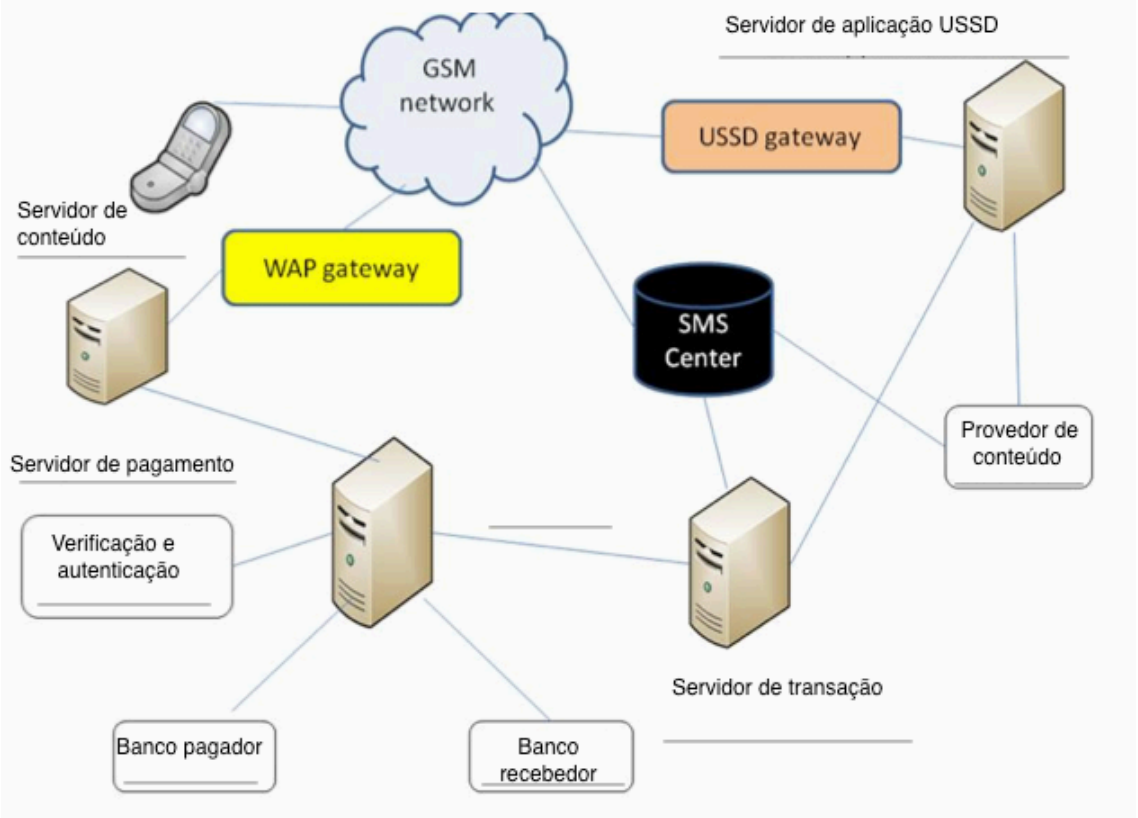

Figura 6 – Plataforma de *m-payment* baseado em telefone móvel
Fonte: [Tehrani et al., 2010]

2.3.5 Plataforma de aplicação baseado em *SIM*

Esta plataforma é baseada em aplicativo instalado no SIM. O usuário recebe o *software* de pagamento e outros serviços diretamente através do servidor OTA (*Over the Air*). Quando o *software* é instalado com sucesso, o usuário pode enviar um pedido de serviços suportados. Este pedido é processado no servidor OTA e gravado no servidor de transação.

Esta plataforma permite ao usuário criptografar mensagens. O servidor OTA as decifra pelo HSM (*Hardware Security Module*) incluindo chaves de criptografia.

A ferramenta para o desenvolvimento deste tipo de aplicação é chamado de SIM *Toolkit*, permite que a aplicação forneça serviços e valores. É constituída por um conjunto de instruções programadas no SIM, que especificam como interagir com o lado de fora da rede. Um cartão SIM especial chamado WIB-cartão é um produto adequado para o operador móvel que deseja implantar uma ampla variedade de serviços agregados a aplicativos.

O cartão possui muitos recursos úteis, tais como a segurança dos dados, o aumento da velocidade e outros mecanismos. O menu STK e a programação WIB podem ser atualizados rapidamente através de OTA e SMS. Alguns dos serviços móveis, que podem ser fornecidos por esta plataforma incluem a operação financeira, eletrônica e física, reserva (restaurante, bilhete, etc.)

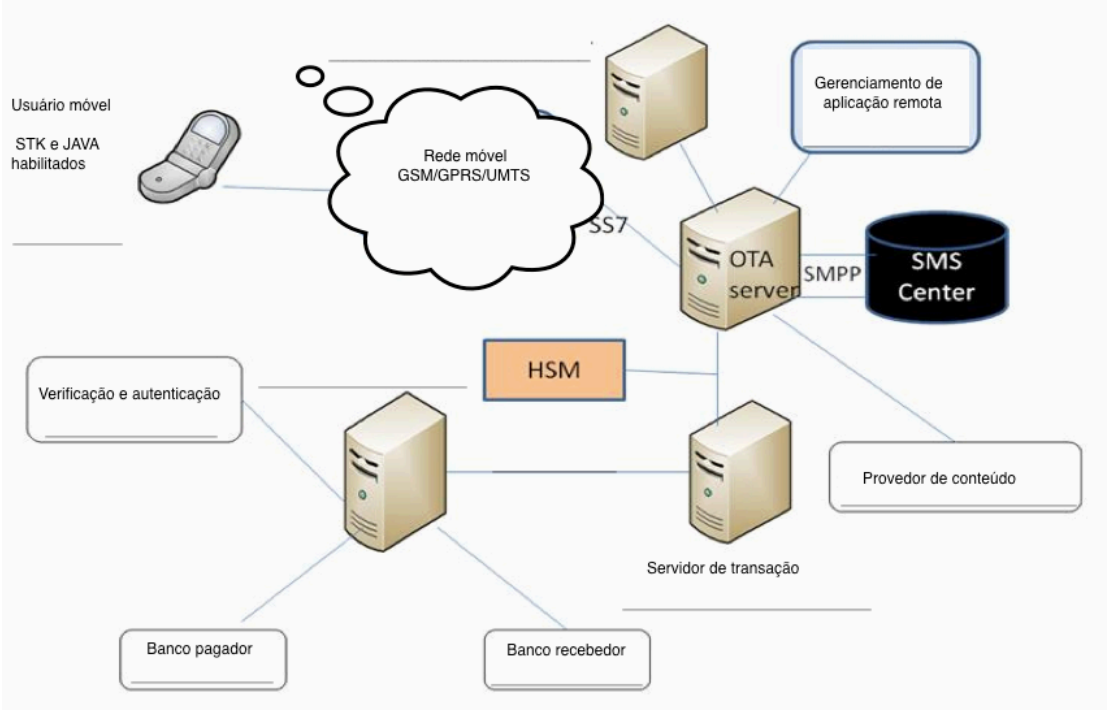

Figura 7 – Plataforma de *m-payment* baseada em SIM
Fonte: [Tehrani et al., 2010]

2.4 Tecnologias usadas para pagamento móvel

Nesta seção, falamos das tecnologias usadas para pagamento móvel e comparamos umas com as outras.

NFC versus Código de Barras 2D

A tecnologia NFC - *Near Field Communication* foi concebida através da combinação das tecnologias de identificação sem contato; baseia-se na conectividade sem fio que, possibilita a comunicação de curto alcance (até 15cm) entre dispositivos eletrônicos [Barbosa 2008], conforme apresentado na figura 8. O NFC é uma extensão da norma ISO/IEC 14443 mais as especificações técnicas do fórum NFC [NFC Forum 2012], fundado pela Nokia, Sony e Philips em 2004, dando inicio à criação de uma nova norma ISO/IEC 18092 [*Information technology - Telecommunications and information exchange between systems - Near Field Communication - Interface and Protocol* (NFCIP-1)].

A norma [ISO/IEC 14443], em desenvolvimento pelo subcomitê 17 da ISO, fala também de outra tecnologia por aproximação chamada *QR Code* (*Quick Response*) ou código 2D. Essa norma define os tipos de cartões por proximidade e os protocolos de transmissão associados. A norma [ISO/IEC 14443] é composta de quatro partes que descrevem dois tipos de cartões (tipo A e tipo B). Todos os tipos de cartões se comunicam através de rádio em 13,56 MHz, usando o mesmo protocolo de transmissão.

A parte um trata de características físicas de proximidade, a parte dois trata do sistema de codificação, a parte três, dos procedimentos de inicialização dos protocolos e a parte quatro, da transmissão. A parte da transmissão mostra como encadear os blocos de dados, tempo de espera e ativação.

Não utilizamos neste presente trabalho a norma [ISO/IEC 14443], devido os poucos aparelhos encontrados no Brasil com a tecnologia e a restrição dos aparelhos existentes em vários sistemas operacionais. Como a idéia principal era desenvolver um *framework* que funcionasse em qualquer plataforma ou dispositivo móvel independente do *hardware*, vimos que ficaríamos restritos a alguns dispositivos.

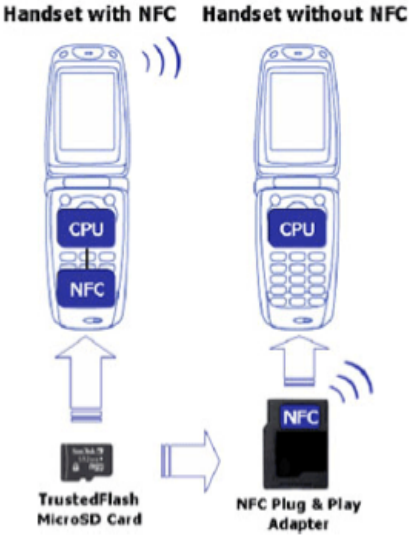

Figura 8 - NFC e placa mãe do aparelho - Fonte: [Barbosa 2008]

A tecnologia NFC é considerada de última geração e o código 2D ainda é considerado uma opção interessante devido a sua simplicidade no conceito, várias possibilidades de uso e baixo custo tanto para o usuário como para o empreendedor. Estudos anteriores provam que o código 2D ainda é bem mais barato do que o NFC [Rouillard et al., 2008], por exemplo: Se alguém quiser saber se é alérgico a determinado produto, basta posicionar o dispositivo no código 2D e aparecerá automaticamente na tela, [Rouillard et al., 2008] dependendo da base de dados existente sobre o paciente.

Para que a tecnologia por aproximação NFC funcione perfeitamente, é necessário ter um chip instalado junto ao celular comunicando-se com a CPU do aparelho de forma dinâmica [Barbosa 2008]. As duas tecnologias são classificadas como de aproximação, porém são tecnologias diferentes. Usando o NFC, a comunicação vai de 106 até 848 Kbps enquanto o código 2D é de 384 Kbps.

O custo do NFC ainda é grande para ser colocado em todos os aparelhos e precisa de uma modificação no sistema operacional. Existe também um problema de segurança ainda não tratado de forma séria, pois, ao perder o celular, todos os créditos serão perdidos. Em outros artigos, autores indicam o bloqueio do aparelho enviado pela operadora, evitando assim o uso indevido. Mesmo assim, o valor em créditos dentro do chip NFC será perdido.

O Código 2D não precisa de mudança de *hardware* no aparelho como o NFC, porém precisa de alguns requisitos, como câmera integrada no aparelho móvel e um leitor de Código de Barras 2D. A vantagem é que a maioria dos aparelhos tem câmera e leitor padrão instalados ou disponível para download. O aplicativo para leitura pode ser obtido gratuitamente na Internet através de download ou envio de SMS.

No Código 2D, não há custo de *hardware* para ser adicionado. De acordo com a nossa proposta, o valor em crédito adicionado, mesmo se o aparelho for extraviado ou roubado, não é perdido. A leitura do Código de Barras 2D depende do manuseio do usuário, abrindo o aplicativo instalado e alinhando a câmera para leitura corretamente. Devido ao baixo custo, a tendência dessa tecnologia é de se tornar popular.

NFC versus WAP

Pode-se dizer que a principal diferença entre NFC e WAP é a facilidade e rapidez no processo, isto é, enquanto o NFC usa a técnologia de aproximação com outros dispositivos, no WAP acontecem vários processos e solicitações de conferências para que a transação seja efetuada. Por utilizar a Internet, a tecnologia WAP é mais lenta e os aparelhos que as utilizam são mais simples e vulneráveis a ataques de *hackers* [Barbosa 2008].

O WAP tem como principal atrativo a acessibilidade devido à compatibilidade que se tem entre as marcas antigas de aparelhos. O WAP possui também problemas de segurança, muitos *hackers* já relataram a possibilidade de invasão e captura de dados no acesso ao WAP [Neto et al., 2010].

Código de Barras 2D versus SMS

O Código de Barras 2D tem como principal característica a simplicidade na transmissão de dados. Basta um celular com câmera e um aplicativo de decodificação para que o usuário esteja apto a fazer pagamentos.

No SMS, o usuário deverá ter um cadastro pré-aprovado na operação, ter créditos e, possuir obviamente, um aparelho celular. O SMS é a forma mais utilizada no Brasil dentre as tecnologias estudadas neste trabalho, mas existe motivação sobre a popularização do código de barras 2D, por ser de fácil utilização e ter baixo custo de implantação por parte do empresário [Barbosa 2008].

O SMS é utilizado por diversas empresas em Brasília, dentre as quais a “Dular”. A tecnologia é usada na compra de presentes de casamentos ou produtos de casa. No momento em que a pessoa faz uma compra para presentear, os créditos são direcionados para a pessoa pré-cadastrada na loja.

Se o usuário cadastrado quiser trocar o presente por algum outro modelo, basta pegar os pontos adicionados e utilizar como créditos. O atendente pega um celular e uma mensagem é enviada para o usuário cadastrado e na resposta dessa mensagem, o crédito é debitado automaticamente. Essa operação necessita de pelo menos dois aparelhos para envio e recebimento. Não é necessário estar no local para confirmar a operação.

Código de Barras 2D versus WAP

O Código de Barras 2D difere bastante do WAP no quesito tempo de acesso. Enquanto o Código de Barras 2D transfere dados a 384 Kbps, o WAP chega geralmente a 19,2 Kbps. O Código de Barras 2D é considerado seguro e atualmente não existem relatos de que houve invasões em transações deste tipo. Mesmo quanto ao WAP, apesar

de também ser considerado seguro, existem notícias de que *hackers* já burlaram esse tipo de serviço.

Após a transação WAP, o cliente recebe uma fatura em casa como a de cartão de crédito. No Código de Barras 2D, o pagamento é realizado com a confirmação do usuário, debitada no crédito preexistente no aplicativo [Barbosa 2008].

SMS versus WAP

Ambos funcionam para *Mobile Payment*, pois não há necessidade de estar presente na hora da compra. O SMS tem ampla vantagem em relação ao tempo de acesso, pois pode ser até 10 vezes mais rápido que o WAP. Nas duas tecnologias, não é cobrado qualquer tipo de taxa pelo uso do serviço, pois está contido no plano mensal com a operadora.

2.5 Web Services

Nesta seção falaremos sobre conceitos básicos, arquitetura, protocolo de comunicação SOAP, troca de mensagens, WSDL e UDDI.

2.5.1 Conceitos Básicos

Não muito tempo atrás, os sistemas eram desenvolvidos e precisavam ser instalados localmente em cada máquina da empresa. A atualização de cada sistema era demorada e trabalhosa para o suporte.

Com o avanço da *Web* e suas linguagens de programação, tornou-se possível desenvolver ou converter informações em páginas ou aplicações ricas em termos de recursos e de fácil atualização. Os computadores clientes começaram a ter acesso aos servidores remotos, até mesmo fora de suas próprias empresas. Se as informações das empresas fossem convertidas em HTML, qualquer pessoa poderia acessá-las através de Internet.

Até aqui, era considerado implicitamente que o *software* do lado do cliente baseado na *Web* consistia em um browser que age como *interface* para o usuário. Essa premissa deixou de ser universalmente válida. Para contornar este problema de usar apenas o browser, a fim de acessar informações, noções de *Web Services* foi proposta.

Usando a *interface* de *Web Services*, empresas começaram a tornar acessíveis suas informações a outros programas definindos. Essa *interface* define os dados disponíveis e como eles podem ser acessados. [Summerville 2011] considera que *Web Services* são representações padronizadas de alguns recursos computacionais e de informações que podem ser usadas por outros programas.

Considere muitas máquinas e seus respectivos programas acessando um serviço *Web* de forma centralizada ou semicentralizada, representada pela figura 9.

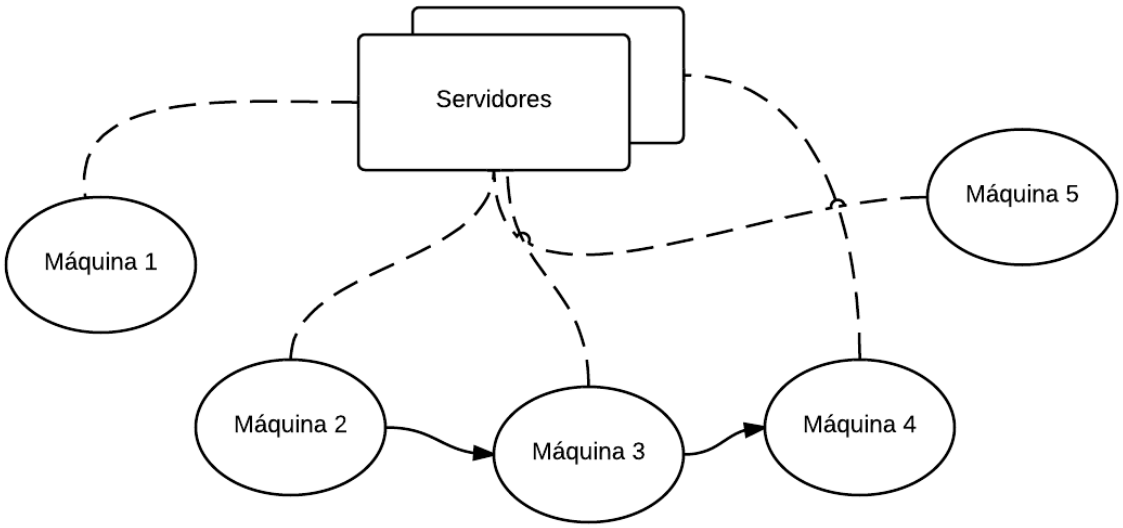

Figura 9 - Serviço Web Semicentralizado
Fonte: [Tanenbaum et al., 2007]

É importante que o serviço seja fornecido independente da aplicação que o usa. Os provedores podem desenvolver serviços especializados e oferecê-los a usuários de diferentes empresas [Curbera et al., 2010]. Existem vários modelos de serviços: JINI, *Web Services e Grid Services*. Conceitualmente, todos estes funcionam de acordo com o modelo mostrado na figura 10.

Um provedor de serviços oferece a definição de sua *interface* e a implementação da funcionalidade. O solicitante do serviço vincula ele a sua aplicação. Isso significa que a aplicação vinculada inclui códigos para chamar o serviço e processar os resultados da chamada [Vilas 1999].

Para assegurar que os Web Services sejam acessados por usuários externos, o provedor de serviços faz uma entrada no registro e inclui informações sobre o serviço e o que ele faz.

Os três padrões fundamentais para comunicação entre Web Services são:

1- SOAP (*Simple Object Access Protocol*): Este protocolo define uma organização para troca estruturada de dados entre Web Services;

2- WSDL (*Web Services Description Language*): Este protocolo define como as *interfaces* dos Web Services podem ser representadas;

3- UDDI (*Universal Description, Discovery and Integration*): Este é um padrão de descobrimento que define como as informações do serviço, usadas pelos solicitantes para descobrir serviços, podem ser organizadas.

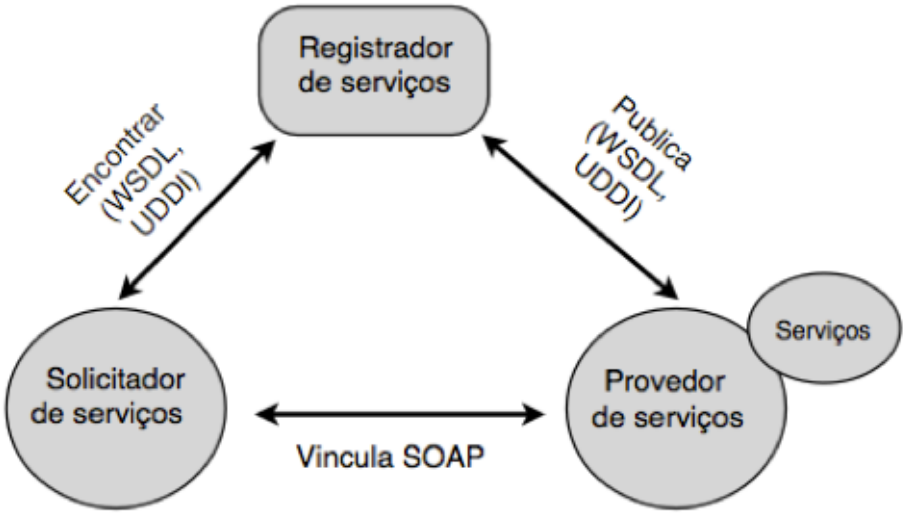

Figura 10 - Modelo conceitual de sistemas orientados a serviços (adaptado)
Fonte: [Sommerville, 2011]

Todos esses padrões são baseados em XML - uma linguagem de marcação legível por humanos e máquinas. Alguns sistemas serão somente construídos com a utilização de *Web Services* e outros os integrarão com componentes desenvolvidos localmente.

Os provedores de serviços projetam, implementam e especificam serviços em uma linguagem chamada WSDL, além de publicar informações sobre eles no registro de acesso geral usando UDDI (padrão de publicação) [Lausen et al., 2007].

Os solicitantes de serviços (chamados também de clientes), que desejam fazer uso de um serviço, buscam o registro UDDI para descobrir a especificação solicitada. A partir daí, suas aplicações podem ser vinculadas, usando geralmente um protocolo chamado SOAP.

Arquitetura Orientada a Serviços

Por natureza, as arquiteturas orientadas a serviços SOA (Service Oriented Architectures) são um caminho para o desenvolvimento de sistemas distribuídos ou sistemas de serviços dedicados [Curbera et al., 2010].

Os serviços podem ser executados em computadores distribuídos geograficamente. Os protocolos padronizados, citados anteriormente, foram projetados para apoiar troca de serviços de comunicação e de informações.

A arquitetura orientada a serviços é atualmente reconhecida como desenvolvimento significativo, especialmente para sistemas de aplicações de negócios. Ela permite flexibilidade, isto é, fornecido localmente ou terceirizados para provedores externos.

O SOA permite que plataformas e tecnologias de implementação diferentes sejam usadas em diferentes partes das empresas para se interoperarem.

Talvez, a razão principal para o sucesso das arquiteturas orientadas a serviços seja, o processo ativo de padronização, trabalhando lado a lado com os desenvolvimentos técnicos.

Arquiteturas orientadas a serviços não sofrem com incompatibilidade, o que acontece normalmente com inovações tecnológicas. A promessa do SOA é facilitar a integração de forma automática.

2.5.2 Protocolo de Comunicação SOAP

Enquanto o protocolo HTTP é padrão de comunicação para sistemas distribuídos baseados na Web, o protocolo simples de acesso a objeto (Simple Object Access Protocol - SOAP) é padrão de comunicação para serviços Web [Tanenbaum 2007].

O SOAP tornou o HTTP ainda mais importante do que já era: a maioria das comunicações são implementadas por meio do HTTP ou HTTPS. Em si, o SOAP não é um protocolo difícil. Sua principal finalidade é fornecer um meio relativamente simples de comunicação.

Em outras palavras, o protocolo foi projetado tendo como premissa que duas partes se comuniquem sem muito conhecimento uma da outra.

De modo geral, uma mensagem SOAP consiste em duas partes que são colocadas juntas, dentro do que é denominado envelope SOAP. O corpo contém a mensagem, ao passo que o cabeçalho é opcional e contém informações relevantes para os nós entre o remetente e o receptor.

Tudo que está dentro do envelope é expresso em XML, isto é, o cabeçalho e o corpo. Por estranho que pareça, um envelope SOAP não contém o endereço do receptor. O SOAP utiliza vinculações como protocolos de transferência. Atualmente existem dois tipos de vinculações, uma para HTTP e uma para SMTP (o protocolo de transferência de correio da Internet) [Curbera et al., 2010] [Sommerville 2011].

A) HTTP: quando a mensagem SOAP está vinculada ao HTTP, o receptor será especificado na forma de uma URL.

B) SMTP: quando a mensagem SOAP está vinculada ao SMTP, o receptor será especificado na forma de um endereço de e-mail.

Esses dois tipos de vinculações indicam também dois estilos diferentes; a troca em estilo colonial e a troca em estilo RPC (*Remote Procedure Calls*).

A) Troca em estilo colonial: nessa troca, duas partes trocam documentos estruturados. Por exemplo, tal documento pode ter um pedido de compra completo e a resposta será um documento com um número da confirmação.

B) Troca em estilo RPC: essa troca segue mais de perto o comportamento de requisição-resposta tradicional, quando invoca o serviço Web. Por exemplo, a mensagem SOAP identifica o procedimento a ser chamado e fornece uma lista de valores com parâmetros de entrada.

Uma troca de estilo RPC normalmente é suportada pelo HTTP, ao passo que uma mensagem é vinculada ao SMTP ou ao HTTP. Contudo, na prática, grande parte das mensagens SOAP são enviadas por HTTP, como visto na listagem 1.

Listagem 1 - Enviado SOAP montado

```
<soap:envelope xmlns:soap=” http://schemas.xmlsoap.org/soap/envelope/”>
        <soap:header>
                <conteudo>valor</conteudo>
        </soap:header>
        <soap:body>
                <conteudo>valor</conteudo>
        </soap:body>
</soap:envelope>
```

2.5.2.1 Troca de Mensagens SOAP

Um exemplo para ilustrar a troca de mensagens seria a seguinte situação: uma pessoa faz um pedido de uma refeição no restautante. Enquanto conversa com o garçom, são criadas várias interações síncronas que definem o pedido.

O garçom anota o pedido de uma ou mais pessoas e então passa a mensagem para a cozinha, que preparará a refeição.

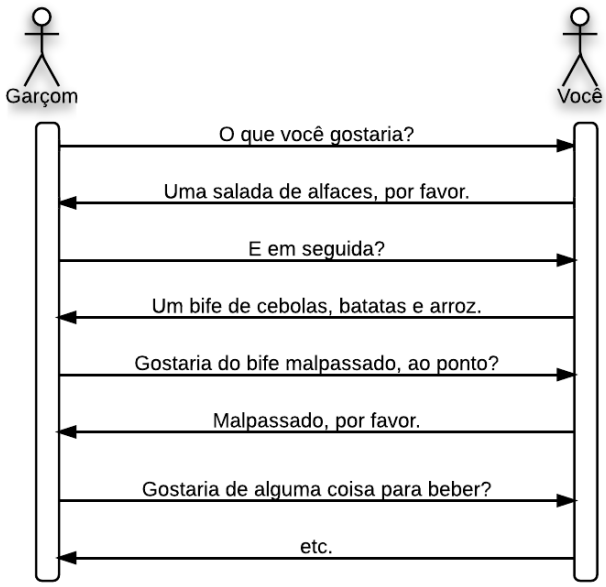

Figura 11 - Interações - Fonte: [Sommerville 2011]

O restaurante pode criar Web Services para integrar o seu sistema com o dispositivo móvel do cliente. Dessa forma, o dispositivo móvel pode enviar um envelope SOAP para o Web Services disponibilizado pelo próprio restaurante, contendo os dados do pedido com o número de mesa.

A listagem 2 mostra um exemplo do envelope SOAP empacotado numa mensagem HTTP.

Listagem 2 - Envelope SOAP sendo transportado

```
<soap:envelope xmlns:soap=" http://schemas.xmlsoap.org/soap/envelope/">
    <soap:header>
        <Restaurante>Real Food</Restaurante>
    </soap:header>
    <soap:body>
        <rf:Pedido xmlns:rf"http://realfood.com/ws/schema">
            <rf:entrada>Sala de alface</rf:entrada>
            <rf:almoco>Bife acebolado</rf:almoco>
            <rf:almocotipo>Mal passado</rf:almocotipo>
            <rf:bebida>...</rf:bebida>
        <rf:Pedido>
    </soap:body>
</soap:envelope>
```

2.5.2.2 Chamadas de Procedimento Remoto Usando SOAP

Muitos sistemas distribuídos eram baseados em troca explícita de mensagens entre processos, e a chamada de procedimento remoto não era nada transparente. Até

que o artigo de Birrell e Nelson (1984) propôs um modo completamente diferente de manipular a comunicação [Tanenbaum 2007].

A proposta dos autores era básica: o RPC precisava fazer uma chamada de procedimento remoto que parecesse uma chamada local, de forma transparente e simples, mesmo que essa chamada fosse em máquinas diferentes ou com endereços diferentes.

Dessa forma, para fazer uma chamada de procedimento remoto usando o SOAP é necessário encapsular e depois transportar, ao longo de uma estrutura de rede.

O serviço criado acessa um objeto localizado em outro ponto da rede, através de uma chamada local. Cada chamada ou requisição exige uma resposta.

Antes de serem enviadas pela rede, as chamadas de RPC (emitidas pela aplicação cliente) são encapsuladas segundo o padrão SOAP. O serviço remoto, ao receber a mensagem, faz o processo contrário: desencapsula e extrai as chamadas do método [Baccaro 2010].

A aplicação servidora processa esta chamada e envia uma resposta ao cliente. A resposta é também encapsulada e enviada pela rede. A máquina cliente desencapsula a mensagem e repassa para aplicação cliente, que pode ser uma aplicação local ou uma aplicação em *browser* [Baccaro 2010].

Para encapsular e desencapsular a mensagem é necessário identificar alguns tipos fortes de parâmetros enviados e recebidos, por exemplo: string e int. Continuando o exemplo do restaurante, o método GetPedido foi enviado passando alguns parâmetros, listagem 3.

Listagem 3 - Enviado o Pedido pelo método GetPedido

```
<soap:envelope xmlns:soap=” http://schemas.xmlsoap.org/soap/envelope/”>
    <soap:body>
        <a: GetPedido xmlns:a”http://realfood.com/ws/schema”
        S O A P:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <a:mesanr xsi:type=”xsd:int”>33</a:mesanr>
            <a:entrada xsi:type=”xsd:string”>Sala de alface</a:entrada>
            <a:almoco xsi:type=”xsd:string”>Bife acebolado</a:almoco>
            <a:almocotipo xsi:type=”xsd:string”>Mal
passado</a:almocotipo>
            <a:bebida xsi:type=”xsd:string”>Agua</a:bebida>
```

```
            <rf:Pedido>
        </soap:body>
</soap:envelope>
```

A listagem 4 mostra a resposta de requisição SOAP empacotada. O mesmo nome do método é gerado com a palavra response na frente.

De acordo com autor [Filho 2011], as implementações de SOAP existem para diversas linguagens. Essas linguagens abstraem as mensagens SOAP, assumindo a conformidade do protocolo. Dessa maneira a interoperabilidade torna-se total entre sistemas heterogêneos, mas na prática não é sempre verdade para sistemas que utilizam a arquitetura CORBA.

Listagem 4 - Resposta empacotada GetPedidoResponse

```
<soap:envelope xmlns:soap=” http://schemas.xmlsoap.org/soap/envelope/”>
        <soap:body>
                <a: GetPedidoResponse xmlns:a”http://realfood.com/ws/schema”
                S O A P:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                        <a:mesanr xsi:type=”xsd:int”>33</a:mesanr>
                        <a:status xsi:type=”xsd:string”>Pedido sendo
preparado</a:status>
                <rf:Pedido>
        </soap:body>
</soap:envelope>
```

2.5.3 Informações Concretas de Binding

Até agora, todos os elementos discutidos descrevem as funcionalidades do serviço a nível de aplicativo. Para completar a interação do serviço do cliente, será necessário obter outras três informações [Alonso et al., 2010]:

- O que o protocolo de comunicação usa (como SOAP sobre HTTP),
- Como realizar interações em cada serviço sobre este protocolo, e
- Onde terminar a comunicação (o endereço de rede).

O elemento de ligação WSDL fornece apenas as partes “o que” e o “como”, incluindo o protocolo de comunicação e especificação no formato de dados portType. Outra parte da ligação diz “onde” a interação ocorre. Pode ser visto na listagem 5.

Listagem 5 - Fazendo o binding para acessar o serviço

```
<binding name="GetPedidoServiceSoapBinding"
type="tns:GetPedidoServicePortType">
        <soap:binding transport= "http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetPedido"> <soap:operation style="rpc"
soapAction="http://realfood.com/ws/getpedido"/>
        <input>
                <soap:body use="encoded"
namespace="http://realfood.com/ws/getpedido" encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
        <output>
                <soap:body use="encoded"
namespace="http://realfood.com/ws/getpedido" encodingStyle=
"http://schemas.xmlsoap.org/soap/encoding/"/>
        </output>
    </operation>
    <operation name="Pedido">
        <soap:operation style="document"
soapAction="http://realfood.com/ws/getpedido"/>
        <input>
        <soap:body use="literal"/>
        </input>
        </operation>
    </binding>

    <service name="pedidoservice">
        <port name="getpedidoservicePort"
binding="tns:GetPedidoServiceSoapBinding"> <soap:address
location="http://realfood.com/ws/getpedido"/>
        </port>
    </service>
```

Usando o WSDL

Uma vez identificados os serviços e projetadas as *interfaces*, o estágio final é a implementação do serviço. Essa implementação pode envolver a programação dos serviços usando uma linguagem padronizada de programação, como Java ou C#. Atualmente, ambas atualmente incluem bibliotecas como apoio extensivo para o desenvolvimento de serviços [Sommerville 2011].

Para usuários e desenvolvedores, o WSDL fornece uma descrição formal da interação do serviço. Durante o desenvolvimento, os programadores usam WSDL como entrada para um gerador de proxy que produz o código do cliente de acordo com as necessidades [Sommerville 2011].

O WSDL pode também ser utilizado como entrada para um proxy de invocação dinâmica, o qual pode, então, gerar os pedidos de serviços corretos no tempo de execução.

O resultado, em ambos os casos, tem por objetivo aliviar o usuário/desenvolvedor da necessidade de lembrar ou entender todos os detalhes de acesso ao serviço. Por exemplo, os usuários que vão fazer o pedido no restaurante só precisam obter a descrição do WSDL e usá-lo como entrada para as ferramentas e infra-estrutura.

Assim, as mensagens SOAP são trocadas de forma correta e em tempo de execução.

2.5.4 Descoberta de Serviços com UDDI

A Microsoft [Microsoft 2008] informa que o Universal, Description, Discovery, e integrações (UDDI) é uma especificação industrial para publicar e localizar informações sobre serviços na Web.

Os autores [Curbera et al., 2010] citam o UDDI (Universal Description Discovery) e suas especificações de integração como uma unificação que tem por finalidade encontrar serviços centralizados.

A interação pode ser criada dentro de empresas ou entre parceiros comerciais. Com o UDDI Services, os desenvolvedores podem interagir com Web Services usando as próprias ferramentas de desenvolvimento de *software*.

De acordo com a [Microsoft 2008], deriva-se do UDDI Services o UDDI Component Web Services que tem por finalidade fornecer uma *interface* Web para o usuário administrador. Com essa *interface*, o administrador pode coordenar tarefas, publicar, consultar e fornecer *interface* de programação de aplicativos (API).

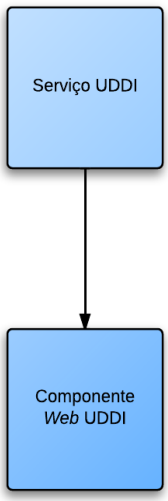

Figura 12 - Derivação UDDI Componet - Fonte: [Microsoft 2008]

2.6 Segurança

2.6.1 Padrão para criptografia de dados (DES) e Padrão Avançado de criptografia (AES)

O (DES) *data encryption standard* - é um padrão para criptografia de dados com chaves simétricas desenvolvido em 1977 e atualizado em 1993, pelo U. S. *National Bureau of Standards* destinado para o uso comercial [Kurose 2010].

O DES codifica texto aberto em porções de 64 bits usando uma chave de 64 bits. Na verdade, oito desses 64 bits da chave são bits de paridade ímpar (há um bit de paridade para cada um dos 8 bits), de modo que a chave DES tem efetivamente 56 bits de comprimento [Wagner and Schneier 2000].

O DES consiste em dois estágios de permutação, (o primeiro e o último passo do algoritmo), nos quais todos os 64 bits são permutados, havendo 16 rodadas idênticas de operação entre eles. A operação de cada rodada é idêntica e toma a saída de dados da rodada anterior como entrada. A Figura 13 mostra o funcionamento da operação básica.

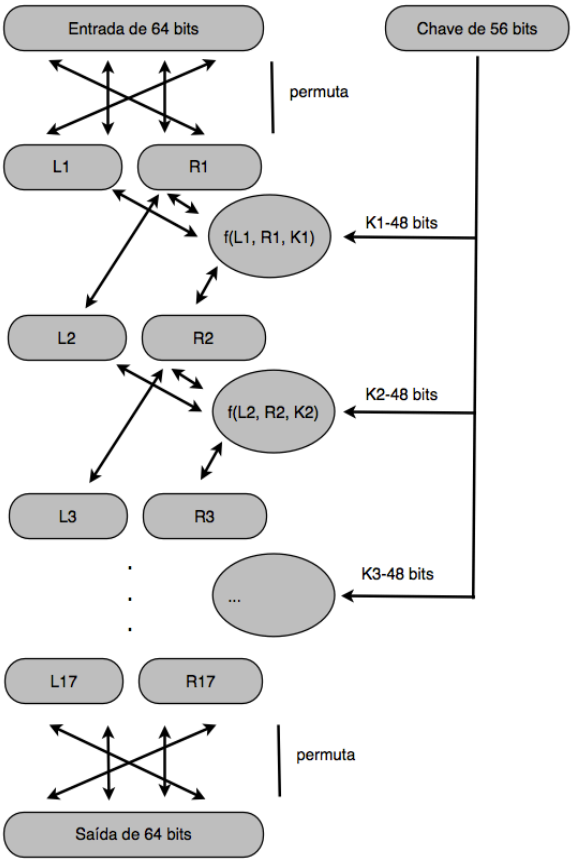

Figura 13 - Operação básica do DES - Fonte: [Kurose 2010]

Até que ponto o DES funciona? Até que ponto ele é seguro? Ninguém pode ter certeza [Kaufman, 1995]. Em 1997, uma empresa de segurança de redes, a RSA *Data Security Inc*. lançou um desafio (DES Challenge) para quebrar (decodificar) uma frase curta que tinha sido criptografada usando o DES de 56 bits.

A frase: *Strong cryptography makes the world a safer place* (a boa criptografia faz do mundo um lugar mais seguro) foi descodificada em menos de quatro meses por uma equipe que usou voluntários por toda a Internet a fim de explorar sistematicamente o espaço de chaves. Cerca de 18 quatrilhões de chaves [RSA Challenge 2002].

Devemos observar também que, até aqui, consideramos apenas a criptografia de uma quantidade de 64 bits. Quando são criptografadas mensagens mais longas, o que é tipicamente o caso, o DES é em geral usado junto com a técnica conhecida como encadeamento de blocos de cifras [Wagner et al., 1996].

Se o DES de 56 bits for considerado muito inseguro, pode-se simplesmente executar o algoritmo de 56 bits várias vezes, tomando a saída de 64 bits de uma iteração do DES como entrada para a iteração DES seguinte e usando uma chave criptográfica diferente para cada rodada.

Por exemplo, o DES triplo (3DES) que é um padrão proposto pelo governo norte-americano [NIST, 1999b] para substituir o DES, o qual está sendo desativado e é permitido apenas em sistemas legados.

Em uma configuração, o (3DES) executa primeiro o algoritmo de criptografia DES sobre os dados utilizando uma primeira chave de 56 bits; em seguida, executa o algoritmo de decriptação do DES sobre a saída da primeira rodada de criptografia usando uma segunda chave e, finalmente, executa o algoritmo de criptografia DES sobre a saída da segunda rodada usando uma terceira chave.

O 3DES foi proposto como o padrão criptografado para o PPP (protocolo ponto-a-ponto) [RFC 2420] para camada de enlace. Uma discussão detalhada sobre os comprimentos das chaves e sobre o tempo e o orçamento estimados necessários para quebrar o 3DES pode ser encontrada em [Blaze 1996].

Em novembro de 2001, o NIST anunciou o sucessor do DES: o Padrão Avançado de Criptografia (Advanced Encryption Standard - AES) [NIST 2001; Danielyan 2001], também conhecido como algoritmo de Rijndael [Daemen, 2000].

O AES é um algoritmo de chave simétrica que processa dados em blocos de 128 bits e pode funcionar com chaves de 128, 192 e 256 bits de comprimento. O NIST estima que uma máquina que conseguisse quebrar o DES de 56 bits em 1 segundo, levaria aproximadamente 149 trilhões de anos para quebrar uma chave AES de 128 bits [Kurose 2010].

Com ajuda do *framework* .NET da Microsoft, o *framework* para pagamento, na parte de criptografia foi desenvolvido utilizando o algoritmo de criptografia AES.

2.6.2 Segurança SSL

A segurança da estrutura é feita em vários momentos específicos. Sempre embutida e desde a criação do QR Code até o momento final da transação.

O QR Code é gerado para acessar uma URL criptografada e com SSL.

SSL é um protocolo desenvolvido pela Netscape que permite a comunicação de um navegador *Web* e um servidor com segurança, e que permite que o navegador *Web* se autentique no servidor *Web* [Wagner et al., 1996].

O protocolo SSL exige que o servidor *Web* tenha um certificado digital instalado nele para que a conexão SSL seja feita. O SSL trabalha usando chave pública para encriptar dados que são transferidos através da ligação SSL.

Todos os navegadores, hoje em dia, suportam SSL, e isso é visto como segurança na intercepção de dados. Muitos sites usam o protocolo para obter confidencialidade de informações de usuários, como números de cartão de crédito e outros. Por convenção, as URLs que requerem uma conexão SSL, começam com "https:" em vez de "http:" [Zheng et al., 2008].

Além da conexão SSL, o *framework* utiliza a criptografia para encriptar os dados do banco de dados.

2.6.3 Segurança permissional

A estrutura de controle de permissão do usuário precisa ser feita de forma temporal, isto é, os privilégios do usuário são fundamentalmente determinados pelas tarefas que deverão ser executadas. Por exemplo: se a um usuário é atribuída a tarefa de escrever um relatório que analisa o desempenho de vendas da empresa, o acesso precisa ser dado apenas para isso. Sem esse privilégio é impossível concluir e acessar o relatório com dados reais.

Tarefas, muitas vezes, servem para lidar com situações que surgem de forma dinâmica, os usuários são convidados a executar uma variedade de tarefas diferentes e, portanto, necessitam de permissões diferentes. Como tal, qualquer atribuição estática de privilégios ou funções para os usuários, tende-se a dar mais privilégios do que o estritamente necessário, na maioria do tempo em vez disso, seria prudente dar apenas os privilégios necessários para a execução da tarefa atual, ou seja, temporal. Não é fácil manter controle temporal sobre cada indivíduo [Alpern et al., 2011].

O conceito de tarefas em controle de acesso trata especificamente de controle de privilégios aos usuários. Em particular, uma tarefa é frequentemente associada a um período de tempo para sua realização (caso contrário, há pouca motivação para um utilizador trabalhar). Por exemplo, um relatório pode ser concluído até o final do mês.

A propriedade temporal de uma tarefa permite-nos, naturalmente, restringir não apenas o conjunto de privilégios concedidos, mas também quando e por quanto tempo eles estarão disponíveis. Mesmo com segurança e privilégios, atividades prejudiciais não são definidas por qualquer ação, mas sim por uma sequência de ações que, tomadas em conjunto, podem causar danos [Alpern et al., 2011].

Além da permissão e privilégio, é necessário saber o que cada indivíduo fez dentro da estrutura, ou seja, a monitoração de ações em tempo real. Neste caso seria, um administrador monitorando as ações de um usuário com permissão e privilégio.

2.6.4 Terceira Parte Confiável

A fim de manter e transmitir ao consumidor informações importantes de pagamento de maneira segura, tais como transação segura, confirmação de dados e etc., foram propostos modelos baseados em uma terceira parte confiável, usando unidade certificadora. Sua principal ideia é de que a unidade certificadora atue como intermediário entre o consumidor e o comerciante, autenticando o dispositivo móvel e o servidor.

Consumidor: o iniciador de uma transação usa o túnel seguro SSL e depois entra com seu usuário e senha previamente cadastrado. As informações fornecidas são verdadeiras e confiáveis, caso contrário o usuário não consegue entrar.

Fornecedor: o fornecedor de bens e serviços recebe um pagamento por parte do provedor de serviços de pagamento. O comerciante deve garantir a informação confiável em tempo e manter alguns dados secretos dos consumidores quanto à privacidade de dados.

TP - Terceira Parte: a terceira parte mantém o perfil dos consumidores, registro histórico, filiação, negócio e etc. As credenciais entre o fornecedor e o consumidor estão no túnel SSL e os registros estão guardados de forma segura dentro do provedor de serviços de pagamento [Ally et al., 2010].

A CA (*Certification Authority*) garante que o túnel de conexão entre o aparelho e a aplicação seja criptografada com, no mínimo 512 bits no HTTPS.

Em resumo, essa estrutura serve para garantir a segurança na comunicação para pagamento móvel além da autenticação dos dados do usuário. O modelo utiliza ainda o gerenciamento e *Single Sign-On* [Ally et al., 2010].

2.6.5 Esteganografia

É uma técnica muito antiga que, em grego, significa "escrita escondida", e baseia-se no estudo de técnicas para ocultar a existência de uma mensagem dentro da outra. Em síntese, é um ramo particular da criptografia que consiste em fazer com que a forma escrita seja camuflada em outra, a fim de mascarar o verdadeiro sentido [Julio et al., 2007].

A esteganografia inclui vasto conjunto de métodos para comunicações secretas desenvolvidas ao longo da história. Dentre tais métodos, destacam-se: micropontos, arranjo de caracteres, assinaturas digitais, canais escondidos, comunicações por espalhamento de espectro e outros mais.

Funcionamento

A técnica de esteganografia pode ser empregada em meios digitais ou não: textos, imagens, áudios, vídeos e espalhamento de espectro. Para mostrar um exemplo específico com textos, segue a frase:

Unificação de Navios e Barcos

Lendo apenas as iniciais de cada palavra acima, teremos: UNB

Esse exemplo é tão trivial que fica na fronteira entre a esteganografia e a criptografia.

Para [Chung et al., 2009], foram usadas na esteganografia várias técnicas de construção de esquemas. Os dados dos textos utilizados para a geração da imagem não têm qualquer distorção, ou seja, qualquer pessoa que ver a imagem consegue facilmente ler o texto usando programa leitor de código 2D [QuickMark 2010].

Existem dois tipos de esteganografia:

A) Lossless: possui o texto aberto dentro ou fora da imagem sem qualquer distorção dos dados.

B) Lossy: possui o texto dentro ou fora da imagem com distorção dos dados. Depois da leitura da imagem, é necessário encontrar o texto distorcido.

2.6.6 Segurança no *QR Code*

A segurança no QR Code é um ponto essencial para realização de compras pelo dispositivo móvel.

Consumidores confiantes que usam essa tecnologia, dão sempre permissão para baixar aplicativos. O aplicativo passa a ter todo acesso e você pode ter problemas de segurança e privacidade [Shannon 2012].

Pesquisa feita pelo Forrester Research, consumidores de códigos 2D que usam smartphones, saltou de 5 por cento para 15 por cento há um ano, principalmente entre os modelos Android e iPhone, os smartphones mais utilizados.

Especialistas dizem que o aumento do consumo de código 2D poderia causar problemas de segurança, privacidade ou até desativação de usuários. O código 2D pode vincular mensagens de texto mal-intencionadas ou *Web* sites maliciosos, disse time Armstrong, um pesquisador de malware no internacional anti-virus Kaspersky Lab [Shannon 2012]. Você pode digitlizar um código 2D e ser induzido a fazer download de um aplicativo, mas você não sabe a origem da aplicação, disse Armstrong [Shannon 2012].

Figura 14 - Vírus no QR Code - Fonte: [Kaspersky - www.kaspersky.com, 2012] [Business Insider 2012]

Esses casos maléficos ainda são relativamente raros, mas ocorrem, disse Armstrong. Por exemplo, a empresa Kaspersky Lab ficou sabendo de um esquema fraudulento que emana de um site russo, oferecendo um chat cliente móvel.

Ao digitalizar o código 2D, os usuários eram redirecionados ao site russo solicitando a baixa do cliente para os telefones. Antes mesmo de baixar o aplicativo, o site pediu permissão para acessar o dispositivo do usuário. O aplicativo então envia secretamente mensagens de texto e, passado um tempo, o usuário pode ser até desativado [Shannon 2012].

Outro fator que os consumidores devem estar cientes são os dois tipos de código 2D: diretos e indiretos. Um código 2D direto contém todas as informações do produto que você precisa saber dentro de si, mas o código indireto requer um aplicativo para chegar a um servidor online afim de procurar informações necessárias.

Neste último caso, o código 2D trabalha em harmonia com um aplicativo, que por vezes têm de ser comprado ou baixado em um site diferente e não as lojas apropriadas, diz Andrew Kinnear, estrategista digital sênior da Aimia Inc., empresa de marketing em Toronto.

Códigos indiretos normalmente exigem que o usuário use aplicativos proprietários, uma vez que eles estão indo para decodificar o código 2D em *Web* sites suspeitos.

Devido a este fato, há muita oportunidade para o fabricante de aplicativo colocar nos sistemas operacionais, sistemas de medição e monitoramento, disse Kinnear [Shannon 2012].

Nem tudo é inofensivo ao lado do código direto. Códigos diretos podem usar URLs encurtadas para levar o usuário a um site desconhecido, confundindo-o como se fosse o verdadeiro destino mas que na verdade é um malware que, possivelmente será instalado em seu dispositivo móvel. Armadilhas podem surgir e para segurança é bom observar como o mundo do código 2D se desenvolve, como aconteceu no início da Internet usando navegadores.

Armstrong diz: “Sendo este um novo território, desconfie de tudo. Se você estiver andando pela cidade e ver um código 2D em um escritório imobiliário local, há uma pequena chance de ser um código infectado, mas se estiver visitando um site russo, você tem uma chance muito maior de ser infectado.”

Kinnear repetiu o conselho: “Semelhantemente a segurança de seu PC, os usuários devem sempre saber o que está sendo instalado e quando. Deve-se ainda determinar fontes confiáveis para que possam usar sem causar problemas”, disse ele [Shannon 2012].

Outra dica dada pelo especialista: “É bom conhecer o seu dispositivo, conhecer o código 2D, saber como ele interage com URLs e como funcionam as permissões concedidas no sistema operacional móvel. Baixar qualquer aplicativo em sites sem o uso de certificados digitais como o https pode te trazer problemas.”

Figura 15 - Alerta de vírus no QR Code - [Fonte: Kaspersky - www.kaspersky.com 2012] [Business Insider 2012]

As grandes empresas de *software* como Apple, Google e Microsoft lançaram a sua própria loja de aplicativos afim de centralizar, testar e verificar qualquer tipo de aplicativo feito por desenvolvedores em todo mundo. Os aplicativos são testados para que não haja nenhum tipo de violação de privacidade, cópia de dados de forma escondida e uso de cartão de crédito indevido.

Até o momento, a empresa mais rígida referente aos testes aplicados foi a Apple, eles passam 15 dias testando e tudo é verificado nos mínimos detalhes. Na loja do Google já foram encontrados aplicativos que na verdade eram vírus e por isso o único smartphone hoje que precisa de anti-vírus é o Android. Como última dica, procure baixar aplicativos diretamente das lojas e não de sites desconhecidos ou os que prometem burlar as lojas das empresas.

Para minimizar os problemas de segurança do código 2D, o *framework* desenvolvido não requer instalação de aplicativo após a digitalização. Sempre é solicitado autenticação de usuário e utilizamos protocolo SSL. O único aplicativo que utilizamos é o leitor de código 2D que em vários modelos são instalados de fábrica.

2.6.7 Função *hash* de criptografia

Como mostrado na figura 16, a função *hash* recebe uma entrada e computa uma cadeia de tamanho fixo conhecido como *hash*. Tem termos de processamento, é impraticável encontrar duas mensagens diferentes *x* e *y* tal que $H(x) = H(y)$.

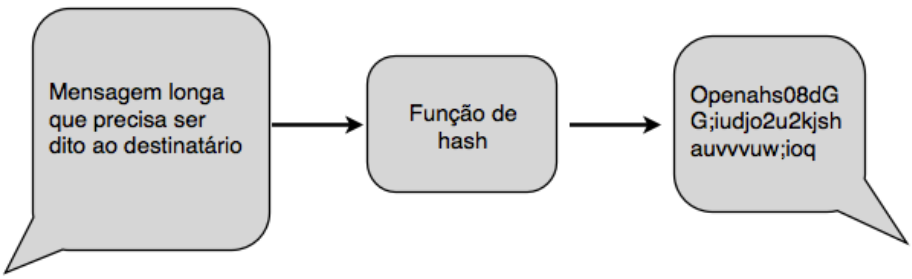

Figura 16 - Função *hash* de criptografia

Informalmente, significa que, em termos de processamento, é impraticável que um invasor substitua uma mensagem por outra que está protegida pela função *hash*; ou seja, se a mensagem e o *hash* da mensagem forem criados pelo emissor, então um invasor não pode forjar os conteúdos de outra mensagem, que possui o mesmo valor *hash* que a mensagem original.

É bom nos convencermos de que uma simples soma de verificação, como a da *Internet*, daria um péssimo algoritmo de resumo da mensagem. Para efeito de segurança, precisaremos de uma função de *hash* muito mais poderosa do que uma soma de verificação.

O algoritmo de *hash* MD5 de *Ron Rivest* [RFC 1321] é amplamente usado hoje. Ele processa um resumo de mensagem de 128 *bits* por meio de um processo de quatro etapas, constituindo de uma etapa de enchimento (adição de um seguido de zeros suficientes, de modo que o comprimento da mensagem satisfaça determinadas condições).

Existe a etapa de anexação (anexação de uma representação de 64 *bits* do comprimento da mensagem antes do enchimento), outra etapa de inicialização de um acumulador e um etapa final iterativa, na qual os blocos de 16 palavras da mensagem são processados (misturados) em quatro rodadas de processamento [RFC 1321].

O segundo principal algoritmo de *hash* em uso atualmente é o SHA-1 (*secure hash algoritmo* – algoritmo de *hash* seguro) [Kurose 2010]. Esse algoritmo se baseia em princípios similares aos usados nos algoritmos MD4 e MD5.

O uso do SHA-1, um padrão federal norte-americano, é exigido sempre que aplicações de âmbito federal precisarem de um algoritmo de resumo de mensagem seguro. Ele produz um resumo de mensagem de 160 *bits*. O resultado mais longo torna o SHA-1 mais seguro [Kurose 2010].

Para criação do SHA-1 conta-se com a ajuda do *Framework* .NET Microsoft, usando a linguagem C# [Listagem 7].

Listagem 7 – Método usando SHA1

```
1.      using System.Security.Cryptography;
2.      using System.Text;
3.
4.          private static string GetSHA1(string strPlain)
5.          {
6.              UnicodeEncoding UE = new UnicodeEncoding();
7.              byte[] HashValue, MessageBytes = UE.GetBytes(strPlain);
8.              SHA1Managed SHhash = new SHA1Managed();
9.              string strHex = "";
10.
11.             HashValue = SHhash.ComputeHash(MessageBytes);
12.             foreach (byte b in HashValue)
13.             {
14.                 strHex += String.Format("{0:x2}", b);
```

```
15.             }
16.             return strHex;
17.         }
```

2.7 Trabalhos Relacionados

Nesta seção falamos dos trabalhos relacionados, destacando cada autor e fazendo pequenas comparações ao nosso trabalho proposto.

Plataformas e sistemas para pagamento móvel [Tehrani et al., 2010]

Em [Tehrani et al., 2010], os autores apresentam um *review* descrevendo as vantagens e desvantagens de cada plataforma, além de oferecer um *ranking* das tecnologias consideradas. O modelo MCDM (modelo de plataforma de pagamento e critério de avaliação) foi usado para comparar diferentes tipos de plataformas e pontos de negócios.

O resultado indica que o aplicativo baseado em cartão SIM em conjunto com SMS oferece a melhor solução hoje, quando comparada com outras alternativas.

A aceitação do sistema de pagamento móvel depende principalmente da plataforma. A plataforma de pagamento móvel VISA, por exemplo, define de forma abrangente um conjunto de ferramentas tecnológicas, padrão de segurança e modelos de negócios que habilitam operadores móveis oferecendo serviços móveis (www.visa.ca).

Foram discutidos os fatos que afetam a usabilidade e satisfação dos consumidores referente a serviços móveis, além dos requisitos de segurança e soluções para comércio móvel baseado em WAP.

O WAP usa a linguagem WML para comunicação entre a rede WAP e o conteúdo da Internet, considerado antigo nos tempos atuais. O WAP converte o WML para HTML, permitindo a entrega baseada no conteúdo e na capacidade do móvel. Alguns serviços móveis podem prover operações financeiras armazenando os dados na *Web*.

A plataforma inclui algumas desvantagens: aplicabilidade somente no Java habilitado no móvel, instalação e atualização manual, duplicidade de instalação no caso da mudança no móvel ou diferentes versões requeridas pela variedade dos dispositivos.

De acordo com [Tehrani et al., 2010], a plataforma baseada em aplicação SIM é uma aplicação instalada dentro do cartão SIM. Os usuários recebem o aplicativo de pagamento direto do servidor. Quando o aplicativo não é instalado com sucesso, o usuário pode enviar requisição para o operador de suporte.

A requisição é processada no servidor e é gravada como transação. A plataforma habilita o usuário para criptografar mensagens, o servidor é responsável pela decriptografia usando HSM, [Tehrani et al., 2010].

A ferramenta para desenvolver esse tipo de aplicação é chamada de SAT (*SIM Application Toolkit*) habilitado no SIM para prover valores. É necessário ter um cartão

especial chamado WIB-Card, um produto para operação móvel que adiciona os valores aos serviços e aplicações.

A plataforma é baseada em dois chips, considerando que alguns dos móveis estão habilitados para guardar dois chips, um para ligações e outro para pagamento. O segundo cartão pode ser personalizado. Existe um produto piloto desenvolvido pelo grupo EMPS com três companhias: VISA, Banco Noreda e Nokia.

O usuário pode iniciar a transação usando o PIN *number* e depois concluir com o outro chip de pagamento. Alguns serviços podem incluir operação financeira e eletrônica. O método de pagamento é uma suíte de pagamentos adaptado com o modelo de negócio central. Alguns serviços móveis podem ser providos pela plataforma incluindo pagamento eletrônico, físico e reservado.

A segurança apresentou problemas no algoritmo de criptografia e implementação. Neste *paper*, várias plataformas foram descritas. Para cada plataforma envolveu arquitetura, regra, transação e segurança. O resultado de sugestão do usuário foi o aplicação baseada em SIM com o SMS binário. É necessário SLA proposto pelo serviço de pagamento móvel.

***A 2D barcode-based mobile payment system* [Gao et al., 2009]**

Jerry Gao utiliza em seu artigo o código 2D do tipo DataMatrix, comprovado que é mais lento para leitura dos dados e leitura de símbolos japoneses [Gao et al., 2009]. O processo para criptografia e decriptografia do código 2D possui falha de segurança, isso porque os dados são colocados no aplicativo sem qualquer criptografia, gerando a possibilidade de intercepção de dados antes de passar pelo processo de criptografia das informações.

Existem dois cenários para negócios de pagamento. O primeiro cenário pode ser usado em parques, taxis, aeroportos e estradas. O segundo cenário possibilita a compra e venda com serviços de pagamento identificados usando código 2D. A arquitetura possui banco de dados, servidor de pagamento, middleware e servidor de Internet. Foi utilizado J2ME desenvolvido com a ferramenta Netbeans, MIDP 2.1, JSON, API de criptografia de dados, JSP (Java Server Pages) e servidor Tomcat.

É necessário registrar o usuário junto do aplicativo instalado localmente e, no momento do registro, um certificado é gerado guardando o ID/PIN do usuário. É necessário instalar o aplicativo cliente no dispositivo móvel para funcionamento. Para isso é preciso ter instalado o *framework* J2ME. Nem todos os dispositivos móveis possuem o *framework* Java ou pode ser instalado.

***Virtual Credit Cards on Mobile for m-Commerce Payment* [Waraporn et al., 2009]**

Waraporn utilizou em seu artigo tecnologia antiga para desenvolvimento e comunicação WAP. Utilizou SMS para venda segura de comunicação via Bluetooth, lembrando que o raio de comunicação não pode ser muito longo referente à máquina principal. Antes de qualquer transação, é necessário emparelhar o dispositivo com a máquina.

A transação precisa ser avaliada e aprovada pelo usuário usando SMS do próprio aparelho. O aplicativo desenvolvido funciona apenas para o sistema operacional Android e é necessário instalar localmente.

O banco de dados e o aplicativo são instalados localmente no dispositivo móvel, que pode ser facilmente interceptado, caracterizando assim falta de segurança.

Foi usado o Android 1.0 SDK, banco de dados SQLite, arquivo XML para prover dados do design e comunicação Bluetooth. Foi utilizado componente intitulado WCCR *Wireless Credit Card Reader* instalado na máquina caixa do lojista, responsável pela comunicação via *Bluetooth* entre o dispositivo e a máquina caixa, para depois efetuar a comunicação com o POS (*Point of Sale*), isto é, operadora de cartão de crédito.

Quanto à segurança, a criptografia no número de cartão de crédito é feita com chave pública, mas não são todas informações enviadas para o WCCR, caracterizando uma segurança relativa. As informações criptografadas entre o device e o WCCR são montadas usando chave privada com a combinação do PIN do telefone e o número.

Podem ser armazenados vários números de cartões de crédito dentro do dispositivo móvel, de modo a dinamizar a quantidade de cartões por usuário nos dias de hoje. O custo é alto para o lojista que precisa comprar a máquina WCCR e o POS.

Funções do Cartão de Crédito Virtual

Três principais funções são inclusas no aplicativo *Virtual Credit Card*: sincronização e pagamento, gerenciamento de conta e relatório de reporte da conta.

A função de Gerenciamento de Conta permite que o proprietário do cartão adicione um novo cartão de crédito em conta, inserindo as informações do mesmo, tais como número de cartão de crédito, código de segurança e data de validade. Permite também que o proprietário defina uma lista de promoção da conta, de modo que o titular do cartão possa ser notificado de um negócio especial quando escolher o cartão de pagamento.

Quando o titular do cartão selecionar um cartão de crédito, as informações da conta serão exibidas para edição em "Editar Conta". A função de relatório exibe determinado período de tempo. Podem ser mostradas todas as contas no banco de dados móvel com a despesa total e créditos disponíveis. O VCC fornece uma função opcional ao titular do cartão a fim de alterar o número do celular e a senha.

Para agilizar o uso, no menu "opções", a funcionalidade permite que o titular defina o cartão de crédito padrão como a primeira escolha entre eles [Waraporn et al., 2009].

***An integrated mobile phone payment system based on 3G network* [Dai et al., 2011]**

[Dai et al., 2011] falam sobre a integração do dispositivo móvel com o sistema de pagamento baseado na rede 3G. Em seu paper os autores afirmam que o padrão de comércio móvel precisa de profundas mudanças, assim como o canal de pagamento.

Depois de estudos sobre os vários métodos de pagamento, esse paper introduz um novo modelo combinado com o IC Chip, celular e Internet.

Os autores deram ênfase a vasta gama de implementação nas gerações 3G e 4G, usando características do comércio móvel; isto é; rapidez e mobilidade na palma da mão.

Os autores defenderam a ideia de que o pagamento móvel é uma transação financeira que pode ser feita por qualquer dispositivo móvel, como defendemos também no nosso trabalho.

Os autores consideram dois tipos de pagamento móvel na China, um chamado *Fee Account Mobile Payment* e o outro *Bank Card Mobile Payment*. Quando a compra é realizada, o primeiro cobra uma taxa na conta do celular e o segundo debita o valor no cartão de crédito dependendo do seu limite.

O artigo cita outros tipos de pagamentos e canais móveis existentes como o código 2D, NFC e RFID, SMS e WAP citados também neste trabalho. O segundo tipo de pagamento móvel adotou a tecnologia Java Applet, mas nem todos os dispositivos móveis podem instalar a máquina virtual Java, como o iPhone.

Os autores utilizam o SOA (*Service Oriented Architecture*) em sistemas embarcados considerando quatro sistemas: plataforma de pagamento, sistema de consumo, sistema de gestão comercial e sistema de gestão móvel na operação.

Os autores consideram o uso de um cartão de pagamento IC *Card Payment* que é um chip no celular e pode ser aceito no futuro. O Chip é combinado com a tecnologia NFC e comunicação pela Internet usando 3G a fim de facilitar ao comerciante o envio de promoções.

Passando para a parte de segurança, as chaves de criptografia podem ser gravadas dentro do cartão IC, que se comunica com o NFC mantendo a segurança. A chave pública é usada pela infraestrutura e a chave privada pode ser guardada dentro do cartão.

Sistema de pagamento móvel baseado em código 2D [Abajo et al., 2011]

O artigo de [Abajo et al., 2011] apresenta uma comparação entre os mais frequentes serviços de comércio móvel, de acordo com as características de *Killer Apps*.

Um estudo é feito para avaliar o impacto que os serviços *m-Payment* podem ter agora e no futuro dentro da perspectiva da economia. A principal contribuição do artigo é descrever as características do sistema, usável para leigo e com vantagens sobre os outros sistemas de pagamento móvel.

Outra contribuição que resulta desse trabalho é a análise de resultados sobre estudantes mediante as respostas dadas, foram vistas vantagens comparadas entre outros sistemas de pagamentos.

Os autores afirmaram que o modelo das aplicações *m-Commerce* estão baseadas no *e-Commerce* e que é melhor fazer aplicações focadas para o móvel.

A particularidade dos serviços de *m-Payment* é que requerem, entre outras coisas, gestão de localização, entrega em tempo real, qualidade de serviço QoS, suporte para as transações, segurança e confiabilidade da rede sem fio. Essa é uma visão global que deve ser seguida desde o início do desenvolvimento.

Foram citadas algumas aplicações *m-Payment* em desenvolvimento, mas muitos padrões são proprietários: P2P Payd, SET, MobiCash, Voice Cheque, Baseados em biometria, MPCS, VCC, Square, EMPS, SimPay, SEMPOS, Poste Mobile, Paypal Mobile, FINEID, Dina Card, i-Mode, Vodafone m-Pay, Paybox, Phonebrain, e m-Pay.

Os *middlewares* utilizados em dispositivos móveis se encarregam de transportar os conteúdos da Internet para as estações móveis com segurança e o bloco encarregado de processar informação é requerida por diferentes aplicações *m-Payment* compostos pelos servidores *Web*.

Os autores falaram que o *m-Wallet* é o tipo mais frequente de pagamento móvel usado porque os dados pagos podem ser vistos a qualquer momento com apenas um clique, a qualquer hora e qualquer local [Hennessy 2012].

O *m-Wallet* permite que os usuários móveis efetuem transações baseadas nas contas vinculadas de pagamento, conforme mostrado na figura 17. O usuário registrado usa o seu número PIN para registrar no sistema de pagamento mediante o envio da solicitação de login no servidor de autenticação.

O servidor analisa, processa a autenticação do cliente e envia a resposta do login mediante um servidor de IDs proporcionando assim uma sessão segura, usando também uma chave pública para comunicação. O cliente autentica no servidor de pagamento móvel com a chave pública e recebe do servidor um certificado de IDs.

O cliente recebe o código 2D relacionado com o produto que lhe interessa. O usuário pode usar a câmera do próprio dispositivo para capturar a imagem relacionada ao produto.

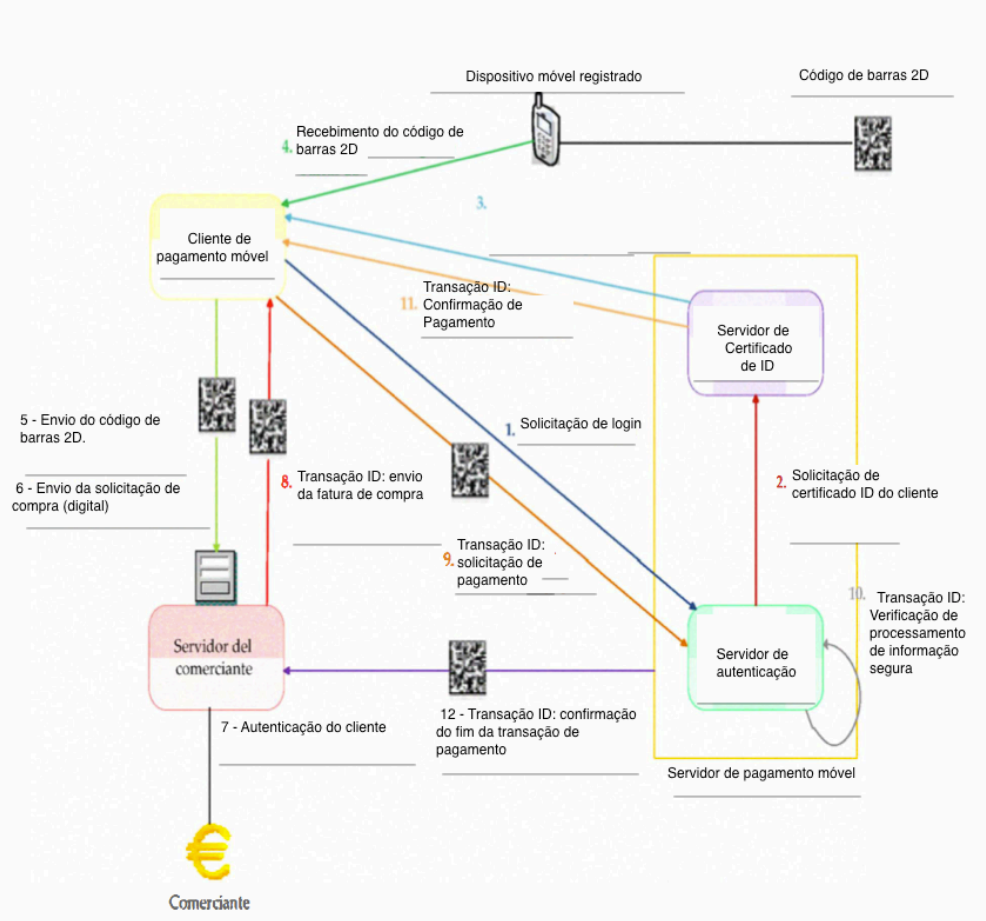

Figura 17 – *m-Wallets* com uso de código 2D - Fonte: [Abajo et al., 2011]

Um segundo caso, o usuário final pode receber um anúncio no dispositivo móvel do comerciante. O usuário pode clicar sobre o código 2D que o envia para o site de comércio eletrônico utilizando o redirecionamento pelo código 2D. Assim o usuário pode solicitar a compra do produto.

O servidor autentica o usuário usando sessão e chave pública. Enquanto isso, a aplicação assinada é recebida pelo comerciante, validada pela chave privada. O servidor do comerciante gera uma fatura de compra assinada e o envia ao cliente em forma de transação.

O cliente envia uma solicitação de pagamento com a mesma transação ID e através da chave privada do cliente, o pagamento é confirmado. Uma vez o servidor estabelecendo conexão segura entre o servidor de pagamento móvel e o cliente móvel, o servidor inclui o certificado no cliente móvel, sessão ID, chave pública e chave privada.

Após a confirmação de pagamento, o servidor envia um código 2D na tela do cliente móvel. Não foi falado de envio de SMS ou e-mail.

O artigo informa também sobre a pesquisa feita entre os alunos de engenharia e técnicos de informática. Os autores concluíram de acordo com as respostas dos alunos que, a aplicação oferece grande interatividade e navegabilidade. Os usuários podem recuperar facilmente a informação do produto graças ao código 2D. O aplicativo suporta de forma simples a verificação do produto.

Sistema e método para pagamento móvel - Patente US2010/0133335 A1 - 03/jun/2010

A patente publicada nos Estados Unidos em 03 de junho de 2010 fala sobre um sistema e método para pagamento através do móvel. É usado uma *tag* de identificação sem fio e transação financeira pelo POS (*Point of Sale*), usando um número de identificação pessoal para completar a transação.

A patente informa o uso de USSD - *Unstructured Supplementary Service Data* suportado pelo MTD - *Mobile Telephony Device* e os passos necessários:

1- Registrar o comerciante;
2- Registrar o usuário;
3- Entrar para comprar;
4- Ler a identificação do usuário;
5- Enviar a transação pela TTP (*Personal Identification Number*) de autenticação da loja;
6- Conferência de valor;
7- Envio de mensagem;
8- Entrar com o PIN do usuário;
9- Envio de confirmação de resposta;
10- Autenticação do PIN;
11- Completar a transação;
12- Enviar confirmação para o POS e para o MTD.

A patente destaca o problema de usar cheque ou dinheiro porque pode ser roubado e qualquer um pode usá-los facilmente. Mesmo que o cartão de crédito use a segurança como o PIN, se houver perda, o número pode ser usado em qualquer transação.

A patente defende a ideia de usar o comércio eletrônico pelo celular devido à performance rápida e simples para os dois lados, do consumidor e do lojista que pode atrair mais consumidores.

O próximo passo lógico do comércio eletrônico pessoal é a autorização de compras de produtos ou serviços utilizando o MTD, sem esquecer do nível de segurança mesmo que o MTD for perdido ou roubado. Nesse caso, a patente informa apenas a desabilitação do celular e mais nada.

A patente destaca os três principais métodos que podem ser utilizados com o MTD: QR *Code*, SMS (*Short Message Service*) e RFID com o POS.

A tecnologia por proximidade permite comunicação de rádio entre o RFID com o produto de venda POS device. O lojista inicia a transação ou serviço, o consumidor traz o celular, aproxima ao POS device e digita a informação financeira. Esse número pode ser o cartão de crédito, número da conta bancária ou o *m-Wallet*.

Uma vez permitindo, o consumidor entra com o seu PIN - *Personal Identification Number* para completar a transação.

A patente informa que o pagamento móvel provê ao usuário uma tag de identificação para pagamentos e outras transações similares pelo MTD e POS. A identificação é provida pela identificação pessoal numérica para completar a transação usando a infraestrutura de dados USSD.

O MTD lê a tag por rádio frequência RFID e permite uma única identificação de leitura no sistema de POS. É necessário ter um celular e um POS com o MIN - *Merchante Identification Number*.

O sistema faz a comunicação *wireless* para o código de UIN - *User Identification Number* com o objetivo de associar o usuário a um POS usando o UIN do MTD quando se aproximar. O POS usa o UIN para mandar de volta os dados.

O MTD recebe do sistema POS uma mensagem e espera a resposta do MTD. O método de pagamento precisa do registro do lojista pelo MIN vinculado a uma conta financeira. O registro do usuário UIN usa comunicação *wireless* e o sistema cria e grava os dados no repositório local. O valor da conta do usuário também é gravado no repositório local do lojista.

O *Finantial Account Belonging to the User* verifica se existe crédito suficiente para pagamento e comunica com o MTD solicitando o PIN e no caso de autenticação com sucesso, os créditos são transferidos para a loja que envia a mensagem ao MTD.

Os dados a serem lidos são criptografados e não podem ser interceptados quando transitados. O MTD não depende do GSM ou TDMA e o sistema de POS tem suporte a TTP usando HTTP.

Cada repositório de dados pode prover uma operação de rede financeira ou similiar. Os dados retornados são guardados com uma ou mais chaves. É necessário que um aplicativo seja instalado no computador do lojista. Dependendo do MTD, a comunicação pode ter interferência.

A patente não informou sobre o uso do aplicativo em vários sistemas operacionais, aparelhos ou testes realizados.

2D Barcodes For Mobile Phones *[Kato et al., 2005]*

Como apontado por H. Kato e Tan KT (2005), códigos 2D foram concebidos para o transporte de dados. Os telefones móveis evoluíram a partir de chamadas de voz com mobilidade para plataforma de celular móvel multimídia [Kato et al., 2005]

A estrutura de código bidimensional pode ser usada para apoiar pré-venda, compra e atividades de pós-venda em operações. Por exemplo, o código 2D pode ser usado como anúncio, cupom, ou promoção de materiais adaptados e decodificados pelo usuário com o dispositivo móvel. Além disso, permite que dispositivos móveis se tornem pontos de venda com base na leitura de código 2D e operações de pagamento.

Após operação de pagamento, códigos 2D podem ser usados pelos clientes como recibo e comprovante de compra para ganhar acesso a bens e serviços comprados com seu dispositivo móvel.

Pode ser usado como ferramenta eficaz para evitar fraudes em produtos de luxo, tais como quadros pintados.

A tabela 1 compara os pontos abordados entre o nosso trabalho e os trabalhos relacionados.

Critérios \ Proposta	Proposta (Waraporn)	Proposta (Abajo)	Proposta (Gao)	Proposta (Kato)	Patente	Nossa Proposta
Tipos de Sistema *m-Payment*						
Classificação de Gao	Cartão de crédito	Cartão de crédito	Cartão de crédito	Não disponível	Cartão de crédito	Cartão de crédito
Classificação de Mader	Aplicação Local	Aplicação *browser*	Aplicação *browser*	Aplicação *browser*	Aplicação Local e *browser*	Aplicação *browser*
Requisitos funcionais e não funcionais	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Disponível
Adoção de limites para valores pagos	Não	Não	Não	Não	Não	Sim
Adoção de MVC	Informação não disponível	Informação não disponível	Informação não disponível	Informação não disponível	Informação não disponível	Sim
Linguagem de desenvolvimento de *software*	Java	Informação não disponível	Java	Informação não disponível	Informação não disponível	C#
Adoção de SMS	Sim	Sim	Não	Sim	Sim	Sim
Integração usando *Web Services*	Sim	Não	Não	Não	Não	Sim
Plataforma						
Celular	Disponível	Disponível	Disponível	Disponível	Disponível	Disponível
Desktop	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível	Não disponível
Receptor de TVD	Não	Não	Não	Não	Não	Sim
Utiliza código 2D						
Tipo	Não usa	DataMatrix	DataMatrix	DataMatrix	QR Code *	QR Code
Utiliza esteganografia	Não	Não	Não	Não	Não	Sim
Usa certificado digital	Sim	Sim	Sim	Não	Sim	Sim
Necessita comunicação por *bluetooth*	Sim	Não	Sim	Não	Não	Não
Utilização de criptografia	Sim	Sim	Sim	Sim	Sim	Sim
Tipo	Informação não disponível	Informação não disponível	ECC (Elliptic Curve Crypt)	Informação não disponível	Informação não disponível	AES
Requisição de login / autenticação	Sim	Sim	Sim	Não	Sim	Sim
Criação de código 2D seguro	Não	Sim	Sim	Sim	Sim	Sim

Tabela 1 – Comparação entre o nosso trabalho e os trabalhos relacionados.

* Pode usar NFC ou RFID.

Capítulo 3

3. QR Code - Código 2D

Este capítulo revisa os conceitos básicos de códigos de barras 2D e afins referente a aplicações móveis, além de relatar sobre as normas.

3.1 Normas JIS X 0510 e ISO/IEC 18004

Para o QR Code se tornar popular, foi necessário abrir a especificação de forma livre. Existem outras especificações fechadas e protegidas por patentes, não populares. A norma é de propriedade da empresa Denso Wave, mas livre para uso, isto é, não se paga nada para usar.

Em 1994 a empresa Denso Wave apresentou o seu padrão que foi aprovado pela AIM Internation (*Automatic Identification Manufactures International*) padrão ISS - QR Code. Em Março de 1998 foi aprovado pela JEIDA (Indústria e Associação Japonesa de Desenvolvimento Eletrônico) o padrão JEIDA-55. Em Janeiro de 1999, o (JIS) (Japanese Industrial Standards) (padrão industrial japonês) aprovou a especificação que virou um padrão aprovado internacionalmente em Junho de 2000 na ISO/IEC 18004 [Denso 2012].

A presente dissertação utilizou o padrão internacional ISO/IEC 18004 *Information technology - Automatic identification and data capture techniques - Bar code symbology* - QR Code para criar o código 2D de acordo com as especificações, acrescentando a criptografia das informações.

Precisávamos saber sobre a estrutura do símbolo descrito na [norma ISO/IEC 18004, pg.12] e detalhada a seguir.

Cada símbolo do código 2D deve ser construído usando módulos nominalmente quadrados e fixados numa matriz quadrada regular. É constituído por uma região codificada e por funções padronizadas,ou seja, localizador, padrões de separação, padrões de tempo, e alinhamento [ISO/IEC 18004]. O símbolo deve ser cercado em todos os quatro lados por uma zona com borda. A figura 18 ilustra a estrutura do código 2D versão 7.

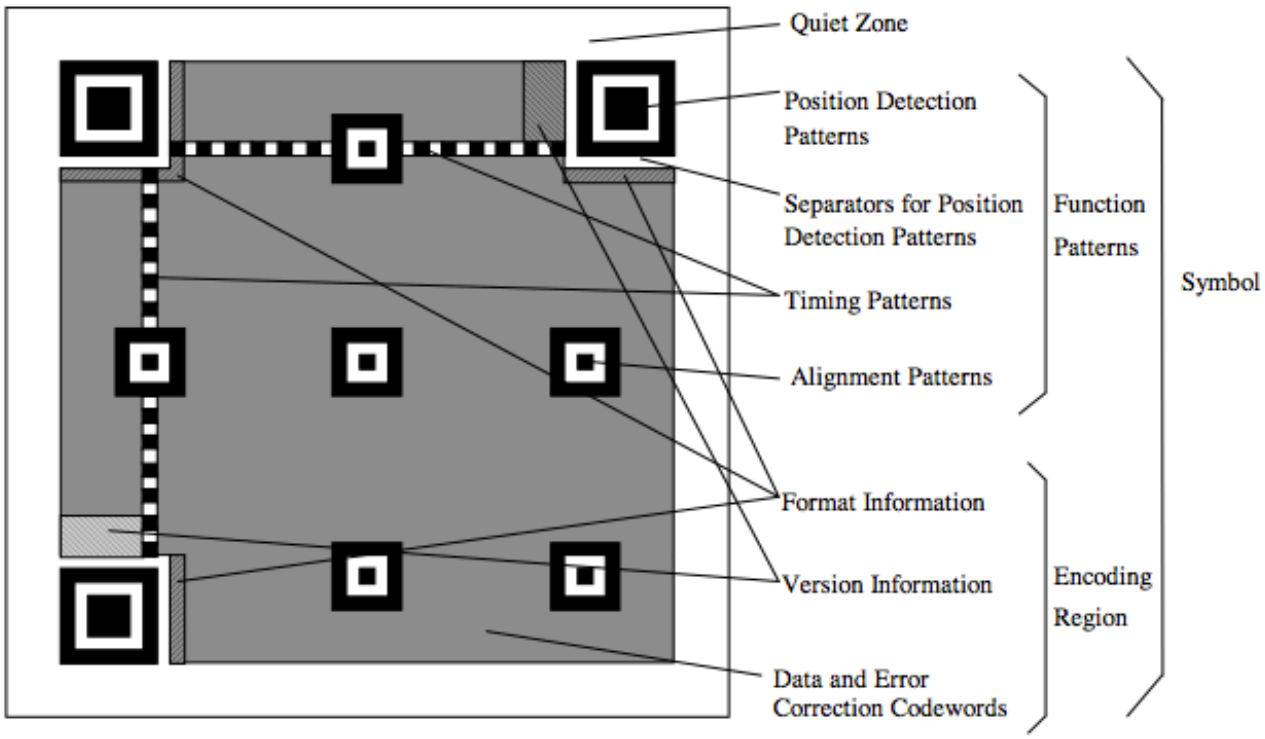

Figura 18 - Posicionamento ISO/IEC - Fonte: [ISO/IEC 18004]

Há quarenta tamanhos de símbolos do código 2D referidas como versão 1, versão 2 ... versão 40. A versão 1 possui medidas de 21 módulos por 21 módulos, a versão 2 possui medidas de 25 módulos por 25 módulos e assim por diante aumentando em passos de 4 módulos de cada lado até a versão 40, que mede 177 módulos por 177 módulos.

A) Localizador

O localizador padrão é composto por três padrões de posições idênticas na detecção, localizadas na parte superior direita, esquerda superior e cantos inferior esquerdo do símbolo.

Cada padrão de detecção pode ser visto como três quadrados concêntricos sobrepostos e é construído na cor escura de 7x7 módulos, cor clara de 5x5 módulos e outras escuras de 3x3 módulos, ilustrado na figura 18.

O símbolo é preferencialmente codificado de modo que os padrões semelhantes têm uma baixa probabilidade de serem encontrados em qualquer parte do símbolo o que permite a identificação rápida do código 2D.

B) Separador

Um separador do módulo de largura é colocado entre cada padrão de detecção na posição e na região de codificação.

C) Padrão de Tempo

Os padrões de temporização horizontal e vertical consistem de uma linha no módulo de largura e uma coluna alternando entre os módulos escuros e claros.

O padrão de sincronismo horizontal atravessa a linha 6 do símbolo entre os separadores para os padrões de detecção na posição superior, o padrão de sincronismo vertical é bem semelhante, desce na coluna 6 do símbolo, entre os separadores, para a esquerda.

Os padrões permitem que o símbolo, densidade e versão sejam determinados para fornecer posições de coordenadas do módulo.

D) Região de codificação

Esta região deve conter os caracteres e símbolos que representam os dados e palavras do código de correção de erro que representam as informações de versão e informações de formato. As listagens 14 e 15 do apêndice fazem o tratamento do código de erro.

E) Zona calma

Esta é uma ampla região que deve ser livre de todas as outras marcas e em torno do símbolo e em todos os quatro lados.

Processo de conversão dos dados de entrada

Tivemos que aprender sobre o funcionamento e *input* dos dados. A norma ISO/IEC 18004, pg.20 nos mostrou como converter, conforme abaixo:

A) Passo 1 - Analisando os dados

A analise do fluxo de dados serve para identificar a variedade de diferentes caracteres para a codificação.

O código 2D suporta o recurso de interpretação extensiva, permitindo que os dados de diferentes conjuntos de caracteres possam ser codificados.

O código 2D inclui vários modos que permite diferentes sub-conjuntos de caracteres a serem convertidos em símbolos de formas eficientes. É aconselhável alternar entre os modos, conforme necessário, a fim de alcançar a conversão mais eficiente de dados para uma cadeia de binário.

B) Passo 2 - Codificação de dados

A codificação de dados converte os caracteres para um fluxo de bits de acordo com as regras (informadas posteriormente neste documento), inserindo indicadores de modo que, em caso de necessidade, altere os modos no início de cada segmento para um novo modo.

É necessário dividir o fluxo de bits resultando em 8-bits e adicionar caracteres de preenchimento conforme o necessário para preencher o número de palavras conforme a versão.

C) Passo 3 - Módulo de matriz

Coloque os módulos da palavra em código de matriz juntamente com os padrões: localizador, separador, padrão de tempo e alinhamento.

D) Passo 4 - Máscara

Aplicar os padrões de máscaras para a região de codificação do símbolo. Avaliar os resultados, selecionar o padrão que otimiza o equilíbrio do módulo claro / escuro e minimiza a ocorrência de padrões indesejáveis, tais como, o aparecimento de uma tarja preta no meio do bloco ou a troca de cor do bloco, ao invés de ser preta ficou branca.

E) Formato e informações sobre a versão

Gerar o formato e (quando aplicável), gerar as informações sobre a versão e preencher o símbolo.

Fluxo de bits (exemplo):

1) Valor de entrada: 01234567

2) Dividir em grupos de três: 012 345 67

3) Converter cada grupo para a forma binária equivalente: 012 -- 0000001100; 345 -- 0101011001; 67 -- 1000011.

4) Colocar os binários em sequência: 0000001100 0101011001 1000011.

5) Converter a quantidade de caracteres em binário: 8 caracteres -- 0000001000 (indicador).

6) Adicionar o modo indicador 0001 e acrescentar o próprio indicador aos dados binários: 0001 0000001000 0000001100 0101011001 1000011. O modo indicador 0001 é adotado porque existe a necessidade de conectar ao módulo citado na opção 5. No caso dos quatro números finais terminados em 0000, então se utiliza o módulo 0001.

Formato e posição da informação

A [norma ISO/IEC 18004, pg.60] informa sobre o formato e posição da informação dentro do código 2D, conforme figura 19.

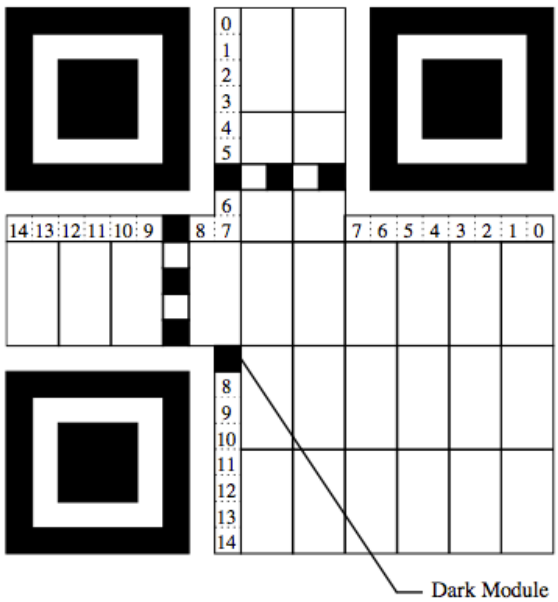

Figura 19 - Formato e Posicionamento da Informação - Fonte: [ISO/IEC 18004 2000]

Depois de criar o *QR Code*, precisávamos de informações sobre a leitura. Vários testes foram feitos, seguindo o padrão. A listagem 9 no apêndice A é responsável por fazer o cálculo da parte preta do *QR Code*.

Com a [norma ISO/IEC 18004, pg.68] aprende-se sobre a detecção e alinhamento padrão de cada informação.

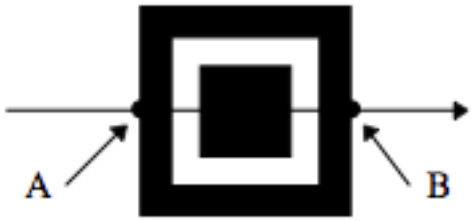

Figura 20 - Um ponto de detecção - Fonte: [ISO/IEC 18004 2000]

Quando a área do ponto é detectada, as posições dos pontos (primeiro e último ponto) A e B, estão nas bordas externas do padrão, conforme a figura 20.

Deve-se escanear os pixels adjacentes da imagem até que, todas as linhas que atravessam a caixa central do padrão de detecção no eixo x da imagem, forem identificadas. Deve-se fazer a mesma coisa para o eixo y após passar pelo eixo x.

É necessário ainda localizar o centro do padrão, construindo uma linha através dos pontos médios entre A e B, nas linhas que atravessam a caixa central de detecção no eixo x.

O mesmo procedimento deve ser feito para localizar o centro do padrão no eixo y. O centro do padrão situa-se na interseção destas duas linhas.

Como escolhemos o código 2D para a utilização, não precisamos estudar profundamente os outros, mas existem muitos mais tipos de código 2D citados posteriormente.

3.2 Diversidade de códigos 2D

Existem variedades de códigos 2D, alguns com padrões abertos e outros fechados. Os mais famosos e mais rápidos são: QR Code e DataMatrix.

Existem outros modelos como: Shotcode caracterizado como Código Circular, Beetagg caracterizado como Código Bidimensional com logomarca e Microsoft Tag caracterizado por ser um código com cores [Rouillard et al., 2008].

Figura 21 - Código 2D, modelo DataMatrix - Fonte: [Rouillard et al., 2008]

Figura 22 - Código 2D, modelo ShotCode - Fonte: [Rouillard et al., 2008]

Figura 23 - Código 2D, modelo Beetagg - Fonte: [Rouillard et al., 2008]

Existe um problema em determinado aspecto caracterizado pela *interface* de pequeno porte. Claro que é muito atraente ter pequenos dispositivos móveis, mas possuem campos limitados de acordo com a pesquisa de Interação Humano-Computador e Engenharia de Usabilidade [Campos 2007].

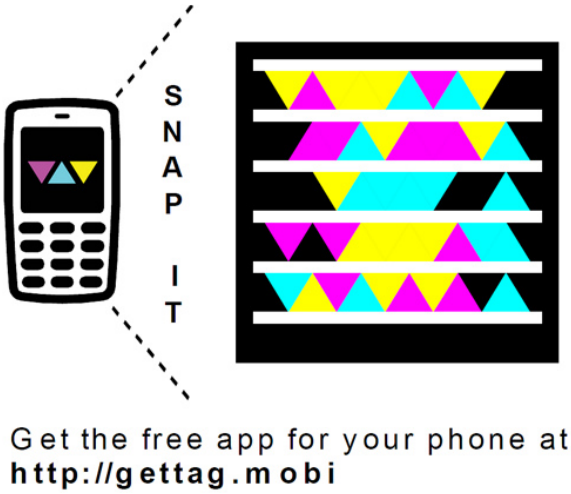

Figura 24 - Código 2D, modelo com cores Microsoft Tag - Fonte: [Getz 2009]

Talvez a maior barreira fosse digitar grandes dados no dispositivo, URL grande da Internet WWW (*World Wide Web*) utilizando teclas numerais, isto é, seria impraticável fazer esse tipo de operação. O objetivo do código 2D, dentre outros, é economizar na digitação para apenas um ou dois cliques. Os QR Codes foram denominados códigos bidimensionais, com possibilidade de transferir dados do objeto físico para o telemóvel [Acohido 2010].

Como mencionado, para o funcionamento do *framework*, é necessário, apenas, um leitor de código 2D [Tukenmez 2010] que, pode ser adquirido gratuitamente na Internet pelo dispositivo.

Além de grandes universidades, o novo código de barras 2D tem atraído a atenção de grandes empresas, como a Microsoft, que lançou o Microsoft Tag silenciosamente em janeiro de 2010.

O código 2D da Microsoft inclui coloração diferente do padrão preto e branco. Aron Getz diz que a Microsoft está planejando fazer mais testes [Getz 2009]. Para a utilização do código, é necessário um *software* leitor específico instalado no dispositivo, ou seja, não pode ser um leitor comum, precisa ser o que a Microsoft criou, figura 24.

Hoje, é mais fácil para o código 2D habilitar aparelhos móveis e tornar um ponto de venda de dispositivo do que apenas a leitura de propagandas para redirecionamentos. A operação de pagamento é um mercado promissor no Brasil.

Após operação de pagamento, o código 2D pode ser usado por clientes como recibo ou comprovante de pagamento. Dessa maneira ele pode ter acesso ao produto ou serviço usando o móvel [Gao et al., 2007].

Segue algumas noções básicas do código de barras 2D ou QR Code.

3.3 Noções Básicas do Código de Barras 2D

De acordo com [Palmer et al., 1995], códigos de barras tradicionais armazenam dados sob a forma de linhas paralelas em larguras diferentes, e eles são conhecidos como códigos de barras 1D (ou barcodes linear).

Código de barras linear refere-se à forma de codificação de números e letras em uma sequência de barras de largura e espaços a serem lidos, recuperados, processados e validados utilizando um computador. Os códigos de barras lineares têm sido usados há 30 anos em transportes ferroviários, para o acompanhamento das mercadorias, produtos de supermercado e muito mais.

Hoje, como representação de leitura óptica de informações em um formato visual, códigos de barras lineares são usados em quase toda parte, tais como fabricação, transporte, serviços postais, saúde, varejo e setor automotivo.

Os códigos de barras podem ser facilmente armazenados, transferidos, processados e validados em formato digital. A identificação do código de barras fornece uma maneira simples e barata de codificação de informações de texto, lido de forma simples usando leitores. Assim, os dados podem ser inseridos rapidamente sem teclado [Gao et al., 2009].

Uma vez que as formas anteriores de códigos de barras 1D não foram capazes de suprir nova demanda com mais informações, os códigos de barras 2D foram inventados para atender às necessidades de codificação de dados alfanuméricos, incluindo letras, números e sinais de pontuação. Existem dois tipos de códigos de barra 2D principais:

A) Chamado código 2D empilhado, como PDF417
B) Matriz de códigos de barras 2D, como Data Matrix e QR Code.

Algumas normas comuns de código de barras 2D estão listadas a seguir [Gao et al., 2009], na tabela 2.

	QR Code	PDF417	DataMatrix	Maxi Code
Developer (country)	DENSO (Japan)	Symbol Technologies (USA)	RVSI Acuity CiMatrix (USA)	UPS (USA)
Numeric	7,089	2,710	3,116	138
Alphanumeric	4,296	1,850	2,355	93
Binary	2,953	1,018	1,556	
Kanji	1,817	554	778	
Major Features	Large capacity Small printout size High speed scan	Large capacity	Small printout size	High speed scan
Standards	AIM International JIS ISO	AIM International ISO	AIM International ISO	AIM International ISO

Tabela 2 - Tabela códigos 2D - Fonte: [Gao et al., 2009]

3.4 QR Code

QR significa *Quick Response*, pois o criador permitiu que o seu conteúdo fosse decodificado com alta velocidade [Digital 2009] [Computing CMIS 2008].

O QR Code é um código 2D introduzido pela companhia japonesa Denso Wave, em 1994. Este tipo de código de barras foi inicialmente utilizado para controle de estoque de peças de veículos e, agora, é usado em uma variedade de indústrias e serviços [Gabriel 2009]. A figura 25 apresenta diferenças em relação ao código de Barras (unidimensional).

Figura 25 - Código 2D e Código de Barras - Fonte: [Gabriel 2009]

O código 2D tem um processo de reconhecimento com 5 etapas: [Reilly et al., 2007]:

A) detecção de bordas;
B) forma da detecção;
C) identificação da barra de controle de código;
D) identificação de orientação do código de barras, dimensões, pouca densidade utilizando a barra de controle; e
E) cálculo do valor do código de barras.

Além do processo de reconhecimento, existem padrões para impressão de tamanho pequeno e com alta velocidade de digitalização. O QR Code é composto dos seguintes padrões: *finder pattern*, calendário, informações sobre o formato de alinhamento e dados da célula, [Sun et al., 2007] conforme figura 26.

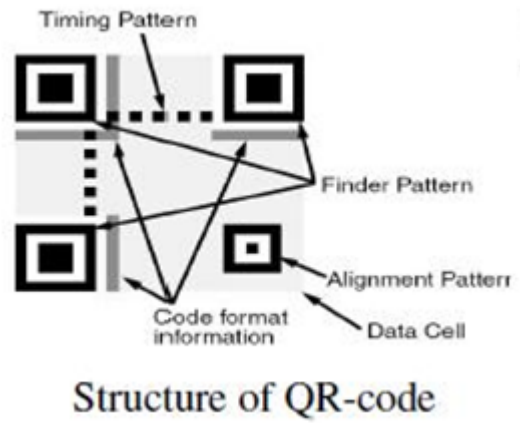

Figura 26 - Estrutura do QR Code - Fonte: [Brambilla et al., 2010]

O código bidimensional também é adequado para ser exibido no meio eletrônico, tais como monitores de computador, telas de TV, parede, papel e até mesmo em pequenas telas de aparelhos portáteis [Brambilla et al., 2010].

O uso do código 2D no meio da comunidade começou, primeiramente, com os alunos dos campos universitários, colocando uma maneira nova de acessar as ementas dos cursos pelo celular e sem digitar nada em tela, apenas digitalizando o código 2D.

Figura 27 - QR Code em hambúrguer - Fonte: [Pinto 2010]

A Apple anunciou o novo i-Pod em outdoors com QR Code, assim como o QR Code foi utilizado em uma campanha publicitária da Nike, permitindo acesso direto ao site dedicado exclusivo para dispositivo móvel. QR Code, agora, aparece em revistas, anúncios, produtos de entretenimento, camisetas, passaportes, cartões de visita e no metrô.

Figura 28 - Cartão utilizando QR Code - Fonte: [Pinto 2010]

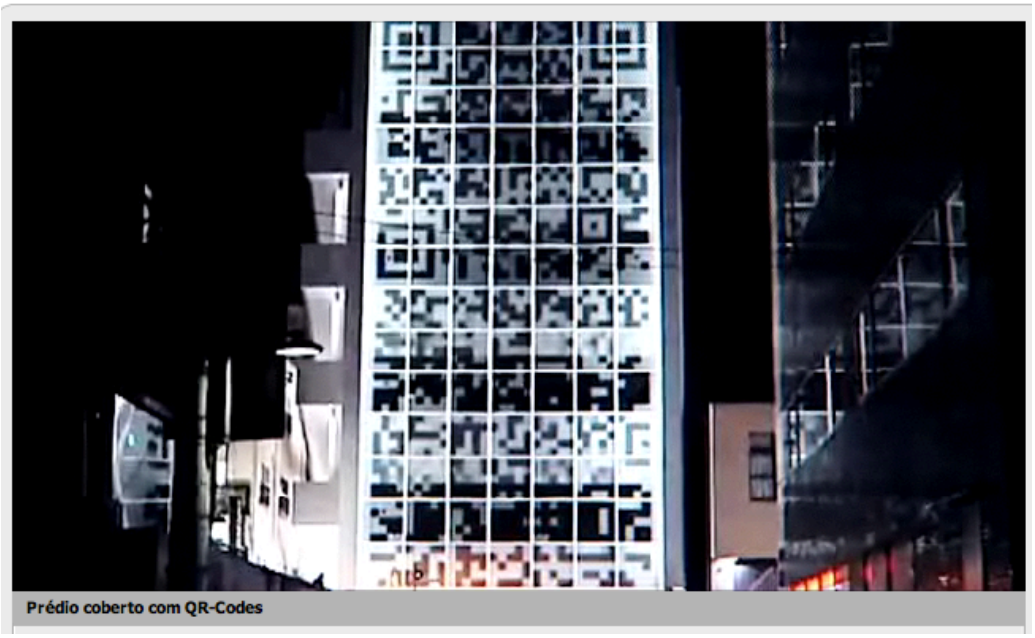

Figura 29 - Prédio com tecnologia QR Code - Fonte: [Olhar Digital 2010]

Aplicativo para deficientes visuais está sendo projetada para mapear todos os locais por QR Code impresso. De acordo com informações [CMIS School 2008], o aplicativo baseado em código 2D ajuda deficientes visuais e cegos a identificarem objetos no ambiente introduzido.

Baseado na ideia do QR Code colado a um objeto, o dispositivo móvel equipado com *software* leitor é posicionado para leitura e decodifica dos dados. O navegador é acionado e direcionado para buscar arquivo de áudio. A proposta deverá ser útil na interação em tempo real de diferentes ambientes e para melhorar a ilustração, nesse sentido, dois cenários estão sendo estudados para apresentação [CMIS School 2008].

A primeira empresa que utilizou o QR Code foi a Mercedez Bens que o fez para identificar partes do carro, motor e salvar informações sobre o uso.


Figura 30 - Montadora de carro e QR Code - Fonte: [Ramsden et al., 2009]

DB BAHN

Bitte auf A4 ausdrucken

Muster-Ticket

ICE Fahrkarte

Gültigkeit: Hinfahrt ab 06.03.2008, Rückfahrt ab 07.03.2008
Gilt 1 Tag bis 10:00 Uhr des Folgetages.

Reisender-Spezial (Hin- und Rückfahrt)
Klasse: 2
Ermäßigung: 1
Hinfahrt: Frankfurt(Main) → Stuttgart, mit ICE
Rückfahrt: Stuttgart → Frankfurt(Main), mit ICE
Über: H: F-Hbf 9:05 ICE571 R: S-Hbf 16:51 ICE1090
DB: Kein Umtausch, keine Erstattung des Aktionspreises.

Barcode bitte nicht knicken!

Hinfahrt:
Zertifikat: 204B 90UU KQ8
Gültig ab: 06.03.2008

Zangenabdruck

Rückfahrt:
Zertifikat: 20W8 V42V ZR1
Gültig ab: 07.03.2008

Zangenabdruck

Herr Heinz Mustermann
Ausweis: BahnCard 6065
Auftrag (NVG): 927420200
Position: 01/01
Belegnummer: 427530

Zahlungspositionen und Preis

Kreditkartenzahlung		Positionen			
Betrag	EUR 76,00	Fahrkarte Hin- und Rückfahrt	1	EUR	68,00
Datum	09.01.2008	Reservierung Hinfahrt	1	EUR	4,00
Transaktions-Nr	2210	Reservierung Rückfahrt	1	EUR	4,00
VU-Nr	9505602250	Summe		EUR	76,00
Gen-Nr	882210	Enthaltene MwSt. (D) 19%		EUR	12,14

Ihre Kreditkarte wurde mit dem oben genannten Betrag belastet. Die Buchung Ihres Online-Tickets erfolgte am 09.01.2008. DB Fernverkehr AG/DB Regio AG, Stephensonstr. 1, 60326 Frankfurt, Steuernummer: 345 231 28662.

Ihre Reiseverbindung und Reservierung Hinfahrt am 06.03.2008

Halt	Datum	Zeit	Gleis	Fahrt	Reservierung
Frankfurt/Main Hbf	06.03.	ab 09:05	5	ICE 571	1 Sitzplatz, Wg. 5, Pl. 55, 1 Fenster, Großraum, Nichtraucher
Stuttgart Hbf	06.03.	an 10:33	5		

Ihre Reiseverbindung und Reservierung Rückfahrt am 07.03.2008

Halt	Datum	Zeit	Gleis	Fahrt	Reservierung
Stuttgart Hbf	07.03.	ab 16:51	16	ICE 1090	1 Sitzplatz, Wg. 2, Pl. 55, 1 Fenster, Großraum, Nichtraucher
Frankfurt/Main Hbf	07.03.	an 19:08	9		

Hinweise:
- Die Fahrkarte muss ausgedruckt vorliegen und gilt nur zusammen mit der beim Kauf angegebenen eigenen gültigen Identifizierungskarte.
- Bei Normalpreisen auch in anderen Zügen als in der Reiseverbindung angegeben innerhalb der Geltungsdauer gültig (ggf. Aufpreis für anderen Weg erforderlich).
- Bei Fragen, Erstattungs- oder Rücknahmewunsch wenden Sie sich bitte an Ihre Agentur. Keine Erstattung/Rücknahme in DB Reisezentren oder im Internet.
- Das Online-Ticket gilt nur für den unter "Fahrkarte" angegebenen Reiseabschnitt. Die Übersicht "Ihre Reiseverbindung" enthält zu Ihrer Information ggf. zusätzliche Teilstücke (z.B. Bus zum Zielort), für die vor Ort ein Ticket erworben werden muss.
- Wenn Ihr Ticket die City-Option beinhaltet, gilt diese nur am Ankunftstag der Hinfahrt bzw. am Abfahrtstag der Rückfahrt (Reisetage wie unter "Ihre Reiseverbindung" angegeben). Die Hinfahrt muss durch Zangenabdruck entwertet sein.
- Es gelten die Beförderungsbedingungen der DB AG sowie innerhalb von Verkehrsverbunden / anfgemeinschaften deren jeweilige Bestimmungen.

Mehr Information gibt es unter www.bahn.de/onlineticket. Wir danken Ihnen für Ihre Buchung und wünschen Ihnen eine angenehme Reise!

Figura 31 - Orçamento e controle com QR Code - Fonte: [Ramsden et al., 2009]

Esta tecnologia é usada em projetos inovadores com orientação pessoal ao indivíduo com deficiência cognitiva [Ramsden et al., 2009].

Na Universidade de Bath, QR *Codes* foram projetados para as palestras, anexa aos impressos e às instruções de curso, para produzirem materiais de apoio aos alunos. Assim, os mesmos poderiam fazer download da apresentação, fornecer *feedback* e avaliação (Roper e Ramsden, 2008) [Ramsden et al., 2009].

De acordo com a [Tecnologia 2010], a empresa de telecomunicações Safaricom foi a primeira a lançar o serviço no país vizinho do Quênia, que hoje conta com 8 milhões de usuários. Além de poder fazer transferências para familiares e amigos, as pessoas podem pagar contas de luz e até receber dividendos de companhias.

A figura 32 mostra um cheque sendo usado com a tecnologia QR *Code* a fim de eliminar falsificação de cheques existente no mercado brasileiro. O banco Banestes usa essa tecnologia para eliminar os problemas causados por cheques falsos. Tecnologia desenvolvida usando parte do código C#, mostrado nos apêndices A e B.

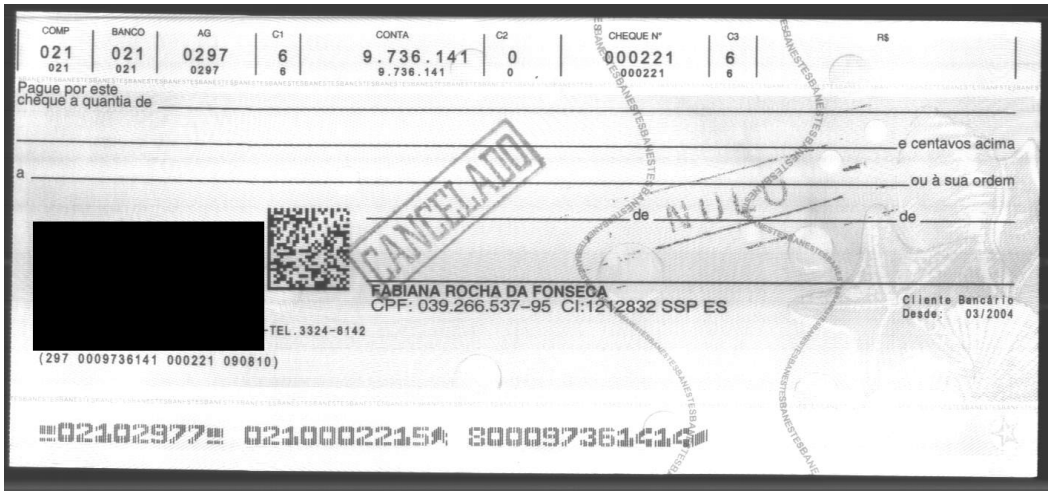

Figura 32 - Cheque com QR Code – Fonte: [ATP S.A. 2012]

O QR *Code* impresso no cheque possui código único para a verificação e conferência. No momento em que o mesmo for passado na leitora (localizada na agência bancária), a verificação sobre a veracidade é feita on-line e em tempo real.

Hoje, muitas pessoas acreditam que o código 2D fornece um meio eficaz para estrutura de aplicação para comércio móvel, devido às seguintes razões [Gao et al., 2007].

• Utilizar códigos de barras digital oferece método simples e barato para apresentar dados de comércio eletrônico em diversas soluções de compra e venda, incluindo a identificação do produto, informações detalhadas, além de propagandas.

• Quanto mais câmeras digitais implantadas em dispositivos móveis, mais utilizado é o código 2D de forma eficaz.

O código de barras fornece um método simples e barato de codificação de informações de texto, lido facilmente por leitores de processos eletrônicos. Os códigos de barras são amplamente utilizados, pois a tecnologia e o processamento fornecem rapidez na entrada de dados sem o uso do teclado.

Em 2002, H. Hahn JK e Joung apresentaram seu algoritmo de implementação no intuito de decodificar barcode 2D PDF-417. Adickes apresentou, então, o protocolo de teste para leitores de mão, usando a simbologia de código de barras PDF417.

Eles compararam o desempenho de cinco leitores de mão em relação a outra tecnologia já existente, incluindo laser linear CCD (*Charge-Coupled Device*), na área de tecnologia CCD e CMOS (*Complementary Metal-Oxide Semiconductor*). O resultado do teste mostra que os leitores de mão, quanto à decodificação, executam as tarefas em menor tempo que outros leitores [Gao et al., 2007].

A [NPC Intelcom 2011] tem se empenhado no desenvolvimento de métodos de código 2D e criação de *software* desde 1999. Criou um kit de desenvolvimento de *software* para apoio de codificação, decodificação e verificação de códigos 2D, tais como Data Matrix, figura 33.

Figura 33 - iReader - Fonte: [NPC Intelcom 2011]

A Xerox é importante no ocultamento de dados usando a tecnologia DataGlyphs e incorporação DigitalData. Suas aplicações incluem gestão de documentos, prevenção da fraude, monitoramento de inventário, cartões de identificação, peças de marcação ou marcação de produtos, [Xerox 2013].

Sony Inc. fez muitas pesquisas e experiências no uso de código 2D, como marca visual.

NTT DoCoMo é uma das principais empresas japonesas de tecnologia sem fio e móvel com aplicações de *software*. Foi responsável pela criação de aplicativos e soluções de código de barras 2D, baseados em dispositivos móveis para consumidores nos mercados de alimentos.

TAL Technologies Inc. (http://www.taltech.com) é uma das principais desenvolvedoras de ferramentas de código 2D, desenvolvem APIs para distribuição utilizando o que existe de mais moderno em leitores.

AirClic (www.airclic.com) dispõe de código de barras para pequenos leitores, pode ser conectado a telefones móveis.

Capítulo 4

4 Desenvolvimento e Validação do *Framework*

Este capítulo tem o objetivo de mostrar aspectos relativos ao desenvolvimento do *framework* e a arquitetura para provimento de pagamento. Assim, o capítulo descreve o .NET *framework* e padrão de desenvolvimento de *software*, bem como apresenta diagrama de sequência, diagrama de classes, fluxograma do processo de compra, comunicação com a TV Digital e infraestrutura necessária.

4.1 Requisitos Funcionais e Não Funcionais

Após detalhamento e estudo dos diversos modelos, tecnologias e implementações existentes de pagamentos móveis, notou-se que poucos estão sendo efetivamente utilizados em larga escala no mundo e muito menos no Brasil. Isso acontece devido a uma série de fatores, incluindo mudança de cultura, falta de motivação na utilização de pequenos teclados e pequenos visores a fim de executar pagamentos, além da preocupação com a segurança (a maior entre elas).

O possível surgimento de aplicativos ou sistemas operacionais dentro dos próximos anos faz com que aumente a desconfiança dos usuários e, a fim de eliminar esse problema, deve-se colocar arquiteturas e modelos padronizados para lidar com pagamentos móveis, já que têm sido proposto pelos diversos fóruns, associações e alianças.

Tentando suavizar o impacto dessas novas possibilidades de pagamentos móveis na cultura dos usuários, principalmente os brasileiros, e com o objetivo de seduzi-los de forma gradual, propõe-se a fácil utilização e entendimento no momento de comprar um produto ou pagar um serviço. O Google *Wallet*, por exemplo, concretizou parcerias em lojas em que um cliente é escolhido para a demonstração do aplicativo e pagamento da conta.

Entre as diversas opções, escolhemos o pagamento móvel para dispositivos pós-pagos ou pré-pagos, não importando o tipo da conta do usuário. O usuário compra créditos com cartão de crédito, débito ou boletos para fazer compras em diversos estabelecimentos, utilizando provedor de serviço de pagamento.

O *framework* possui um mecanismo de configuração que determina um limite para pagamentos, caso essa configuração não seja feita, não haverá limites para realizar pagamentos.

Essa estrutura digital acrescenta grande flexibilidade aos usuários, como adicionar crédito a qualquer hora do dia e em qualquer lugar. Isso representa grande motivação para os usuários começarem a se interessar por esse tipo de serviço. Outra grande vantagem dessa "carteira móvel" é a extensão da simples utilização dos créditos exclusivamente para compra de produtos ou serviços (como pagamentos de contas, loteria, ingressos, transferências entre pessoas, estacionamento, cinema e *fast-foods*).

Os requisitos funcionais da *interface* gráfica do *framework* foram baseados em análise empírica das funcionalidades existentes na maioria dos sistemas analisados e disponíveis em referências consultadas [Waraporn et al., 2009], [Gao et al., 2009], [Tehrani et al., 2010] e [Abajo et al., 2011].

A tabela 3 apresenta os requisitos funcionais e não funcionais considerados.

Requisito	**Funcional**	**Não Funcional**
Exibição do QR Code com os dados criptografados	Sim	Não
Exibição da tela principal para confirmar o pagamento	Sim	Não
Buscar as compras realizadas de produtos ou serviços	Sim	Não
Adicionar produto ou serviço para pagamento	Sim	Não
Cadastramento de usuário	Sim	Não
Login de usuário utilizando email, CPF e senha	Sim	Não
Recuperação de senha por email e SMS	Sim	Não
Exibição de preço do produto ou serviço	Sim	Não
Realização de processo de compra em etapas, permitindo que o usuário volte a uma etapa anterior, antes de finalizar a compra	Sim	Não
Utilização da Internet com SSL como meio seguro	Sim	Não
Envio de SMS	Sim	Não
Verificação do saldo do usuário	Sim	Não
Visualização do código 2D na tela do usuário	Sim	Não
Utilização dos botões para aprovação ou cancelamento de compra	Sim	Não
Processamento de grande parte das regras de negócio no servidor *Web* que integra a arquitetura	Não	Sim
Uso do *framework* .NET como plataforma de desenvolvimento	Não	Sim
Uso do servidor *Web* para processamento e integração dos dados	Não	Sim
Uso de serviço *Web Service* para comunicação com outros sistemas	Não	Sim
Uso de um provedor de pagamento	Não	Sim
Integração com a TV Digital de forma simples	Não	Sim

Tabela 3 - Requisitos funcionais e não funcionais do *framework* de pagamento

4.2 Modelagem

4.2.1 Caso de Uso

Para dar uma visão geral das funcionalidades providas pelo *framework*, tanto para usuários finais, quanto para desenvolvedores, são apresentados os diagramas de casos de uso. A figura 34 apresenta as funcionalidades que o *framework* provê aos usuários finais usando qualquer dispositivo móvel. Tais funcionalidades permitem aos usuários utilizar o próprio aparelho móvel para comprar produtos ou serviços com segurança.

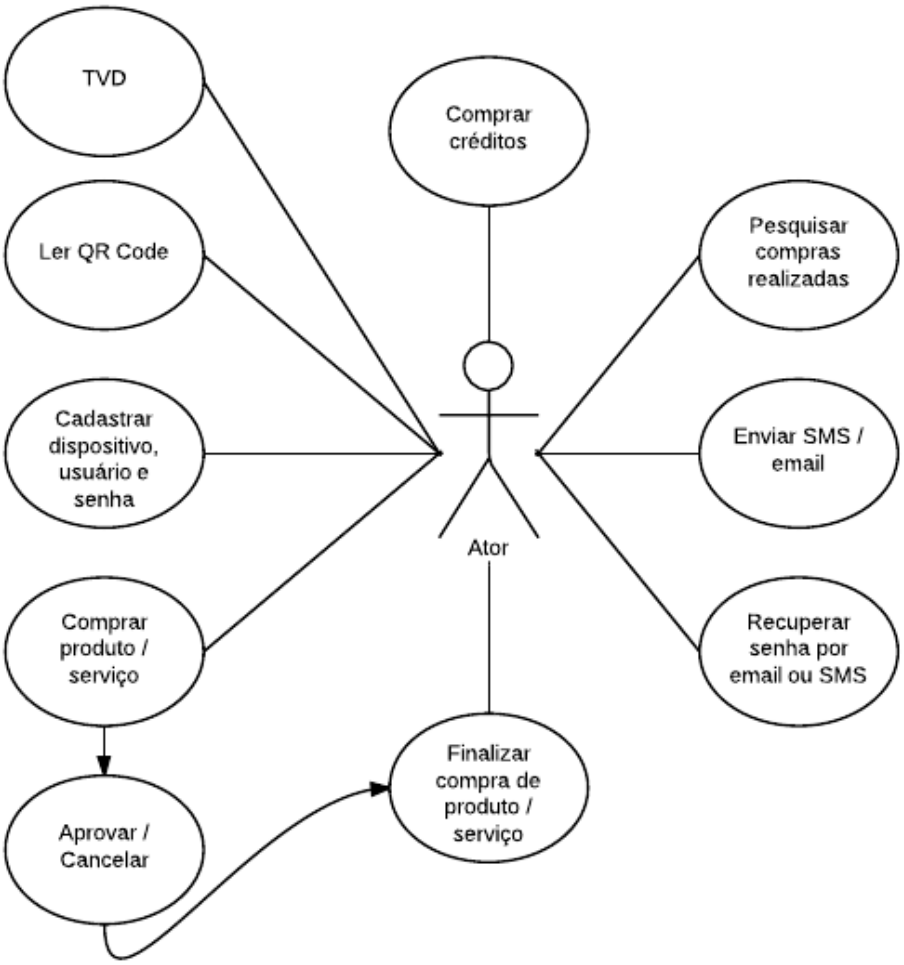

Figura 34 - Diagrama de Casos de Uso das funcionalidades providas aos usuários finais

Fluxo Principal

P1. Este caso de uso começa quando o ator posiciona o seu dispositivo móvel para leitura do código 2D. [A4]

P2. O dispositivo é levado a uma nova tela segura solicitando usuário e senha. [A1] [A4] [E2]

P3. Após a autenticação feita pelo ator, uma nova tela será apresentada informando a compra e o produto lido no código 2D. Pode ser verificado as informações como nome do produto/serviço, valor e foto (se necessário), conforme apresentado na figura 53.

P4. O ator aprova a compra clicando no botão *OK*. [A2]

P5. O *framework* verifica o saldo/cartão do ator e finalizar a compra. [E1]

P6. Finalizando a compra, o ator recebe um email e SMS com as informações, além de gerar na tela do próprio dispositivo, de forma automática, um código 2D de confirmação.

P7. O ator pode pesquisar as compras realizadas.

P8. O ator pode comprar créditos usando cartão de crédito, débito ou boleto. Os créditos serão gastos nas lojas e estabelecimentos parceiros usando o dispositivo móvel.

Fluxo alternativo

A1. Caso o ator não tenha cadastro, o mesmo pode ser cadastrado.

A2. O ator pode clicar no botão cancelar e uma nova tela será apresentada saindo do sistema.

A3. No caso do ator esquecer a senha, ele pode recuperá-la digitando o usuário e solicitando-a. Um email e SMS serão enviados para o seu dispositivo.

A4. O acesso pode ser feito diretamente pela TVD, com esse acesso o QR Code será gerado na tela TV.

Fluxo de exceção

E1. Caso o saldo seja insuficiente, uma nova tela será apresentada ao ator informando que o saldo é insuficiente para realizar a operação.

E2. Caso o ator tente se autenticar usando dados inválidos por 3 vezes, o seu acesso será bloqueado e um email enviado solicitando informações pessoais.

A figura 36 apresenta os casos de uso com funcionalidades providas pelo *framework*, isto é, qualquer outro sistema poderá se comunicar com o *framework* para usar os seguintes casos:

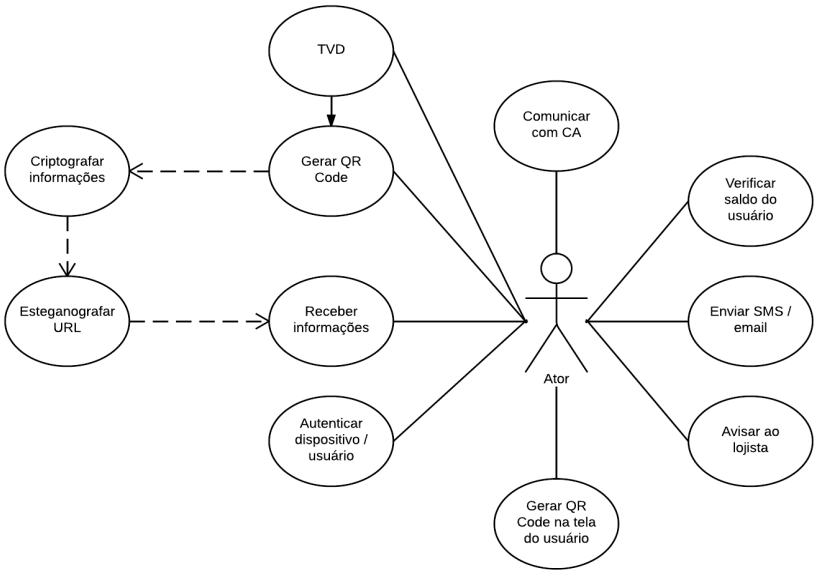

Figura 35 - Diagrama de Casos de Uso das funcionalidades providas pelo *framework*

4.2.2 Diagrama de Sequência

O processo de compra/venda desenvolvido é bastante simples e direto. O diagrama de sequência apresentado na figura 36 mostra como tal processo ocorre.

A definição do trabalho foi decidida pela segurança e facilidade de integração sem precisar de qualquer chip embutido no dispositivo, como falado em capítulos anteriores. A grade de decisão foi dada também pela segurança das informações centralizadas fora do dispositivo móvel. Mesmo se o aparelho for perdido, os dados são guardados no servidor central.

A proposta é diferente de outras abordadas anteriormente. Isso porque a transação é feita por uma estrutura *Web* segura, redirecionada pelo código 2D lido do dispositivo.

A identificação permissiva e autenticação são feitas para continuar o processo de compra, em caso de não identificação o processo é paralisado. O consumidor tem mais segurança, pelo uso do SSL, criptografia de dados, conferência de valores e flexibilidade de dispositivo, pois o *framework* não é instalado no aparelho. Evita, assim, qualquer problema no caso de roubo ou perda. Nenhum aplicativo precisa ser instalado especificamente, apenas um leitor de código 2D, o qual vem instalado de fábrica ou pode ser adquirido gratuitamente.

Como citado anteriormente, toda transação é feita pela Internet, depois de finalizada, é enviada a confirmação via SMS, e gerado em tela, um novo código 2D com um número do protocolo único de confirmação. Além do gasto atual, o usuário pode verificar o relatório on-line de toda a compra, data, loja e produto.

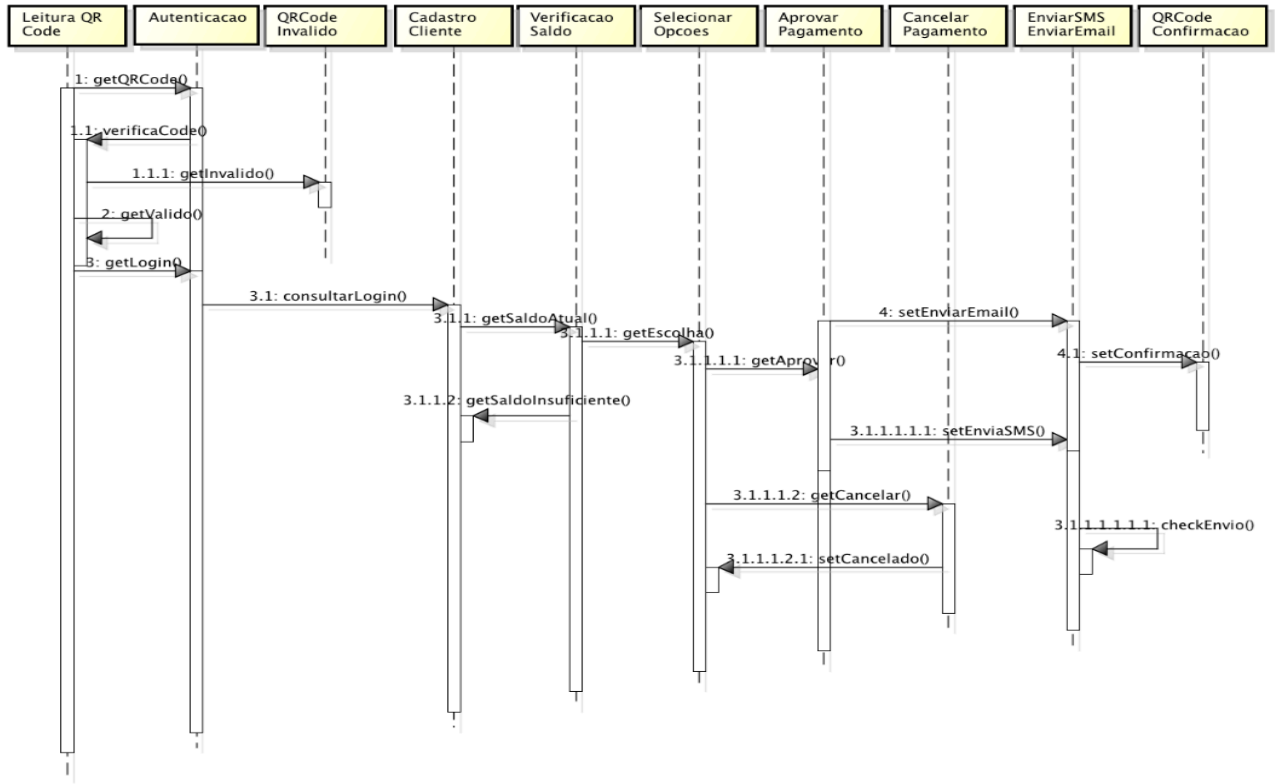

Figura 36 - Diagrama de sequência

Segundo o COAF (Conselho de Controle de Atividades Financeiras), empresa responsável pelo combate à lavagem de dinheiro, grande parte da população que não tem acesso aos serviços bancários, utilizam a ferramenta de pagamento, depósito e transferência entre dispositivos móveis. Temos o objetivo de expandir o nosso *framework* a fim de possibilitar a transferência de valores em trabalhos futuros.

4.2.3 Diagrama de Classes

As figuras 37 e 38 mostram as classes utilizadas para desenvolvimento do *framework* proposto, com métodos privados, públicos, propriedades, eventos e campos utilizando orientação a objetos.

No diagrama 37 as classes *default, SMS e aprovar* têm um papel fundamental para o *framework*, pois são responsáveis pela verificação, envio de sms e aprovação da transação. Cuidam dos eventos ocorridos no *framework* de pagamento, principalmente os eventos de pressionamento de teclas.

O diagrama de classes, figura 38 mostra a parte de acesso a dados, preparado para atender os vários tipos de banco de dados existentes.

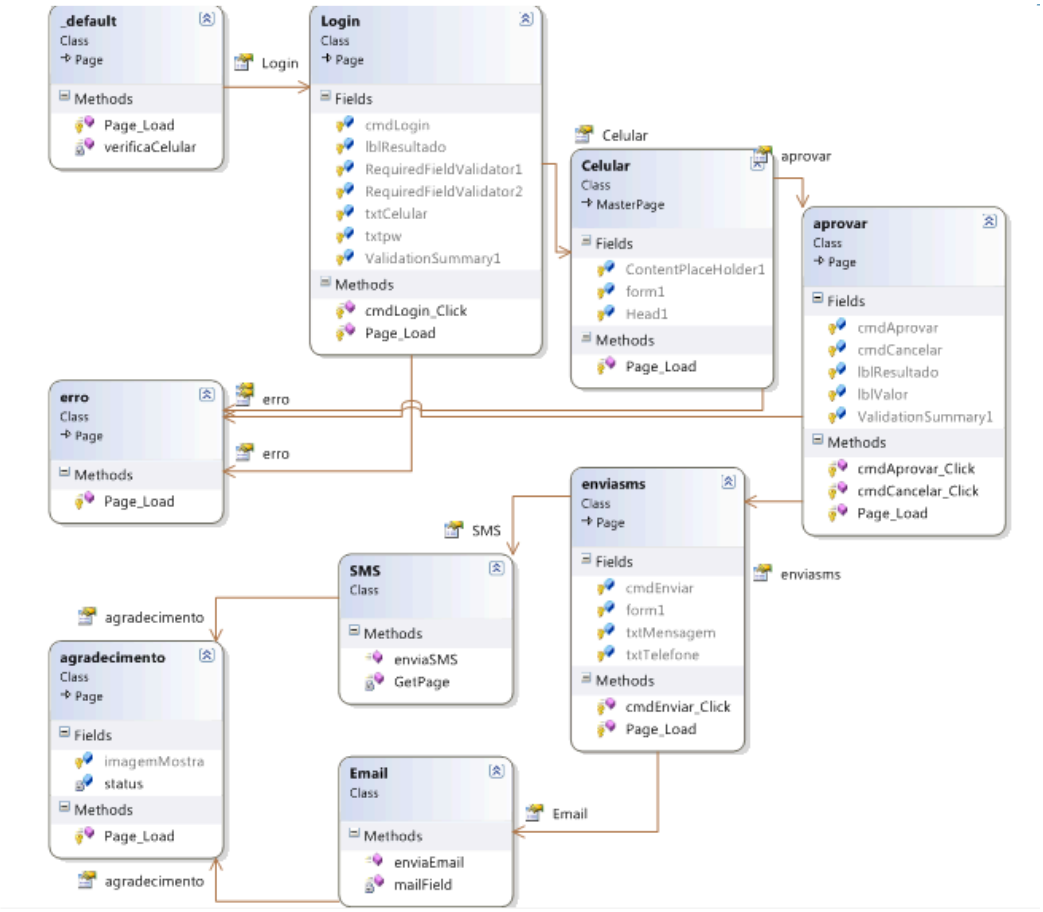

Figura 37 - Diagrama de classes do *framework*

A *interface* IDTec é responsável pelo gerenciamento e escolha da *Helper* específica do tipo do banco de dados. A classe *DataConnection* possui métodos de criptografia e decriptografia da string de conexão para acesso a dados.

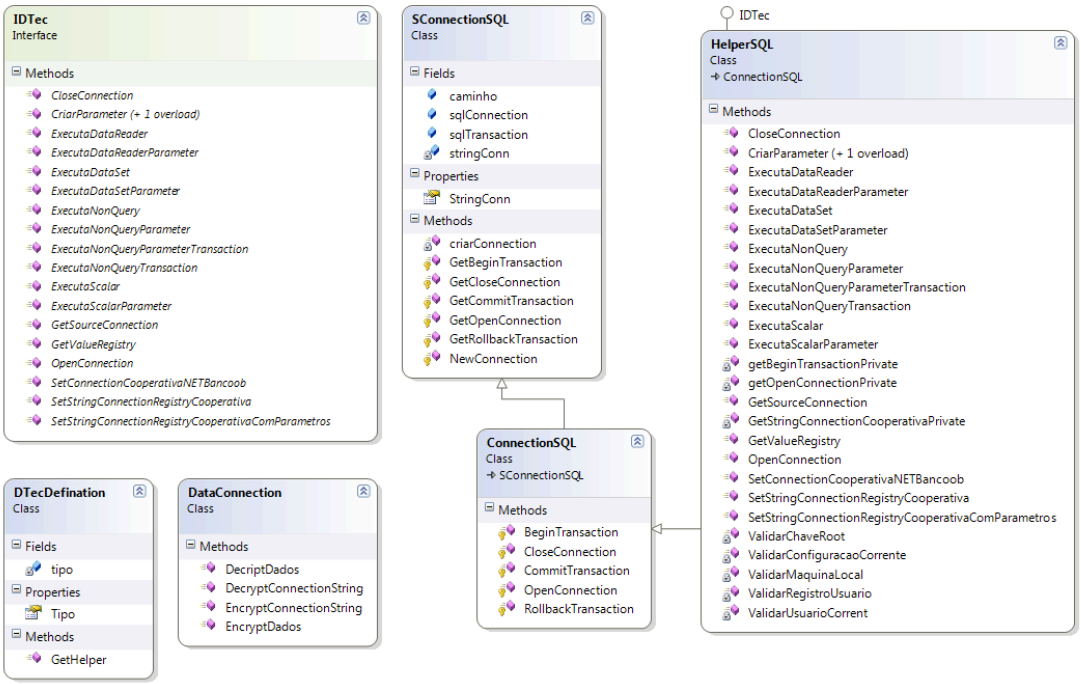

Figura 38 - Diagrama de classes para acesso a dados e criptografia

O diagrama de classes, figura 39 refere-se a parte de geração do código 2D na tela do dispositivo móvel.

A classe considerada principal desse diagrama, figura 40, é a QR Code, porque gera a imagem. O método *calQrcode* é o que efetivamente gera o código e faz cálculos de acordo com os dados informados pelo usuário na geração.

Foi previsto uma classe responsável por dar suporte (*SupportClass*) a classe principal em caso de erro, informações de url e problemas de geração.

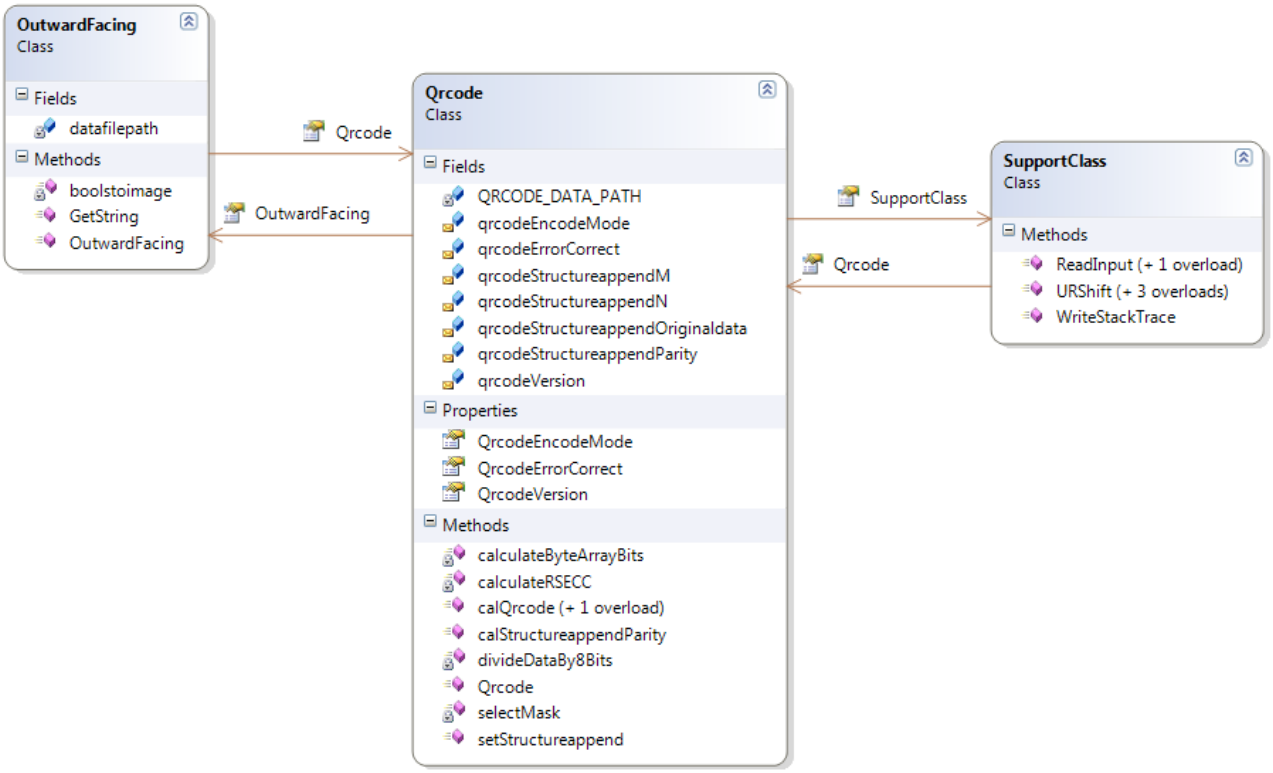

Figura 39 - Diagrama de classes para a geração do código 2D

Esse diagrama refere-se ao apêndice B colocado no final deste trabalho, isto é, o código fonte dos métodos.

4.2.4 Diagrama de Distribuição / Implantação

A figura 40 apresenta um diagrama de distribuição/implantação que mostra como os componentes do *framework* apresentados neste trabalho são distribuídos em diversos *hardwares*, mostrando a arquitetura montada e a relação de dependência entre cada componente. Tal diagrama apresenta todos os componentes de *hardware* e *software* do *framework* proposto, incluindo o uso de outros componentes implementados.

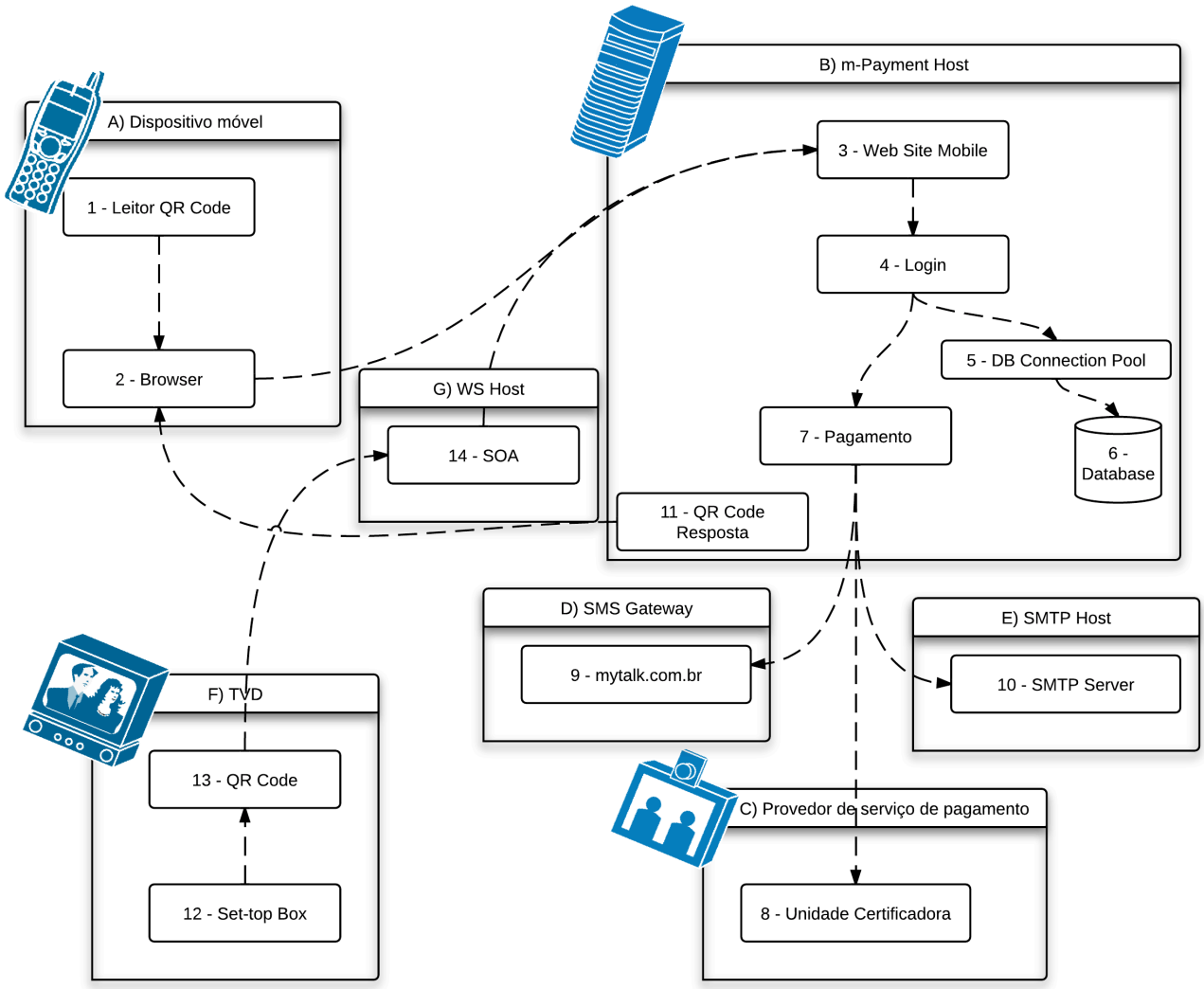

Figura 40 - Diagrama de Distribuição/Implantação

As funcionalidades de cada componente são resumidas a seguir:

A) Dispositivo móvel - dispositivo responsável pela leitura do código QR Code e pagamento.

1 - Leitor QR Code: aplicação necessária para leitura do código 2D, em muitos modelos móveis, a aplicação já vem instalada ou pode ser baixado pelas lojas gratuitamente;

2 - *Browser*: Aplicativo existente nos dispositivos móveis, utilizado para navegação em *Web* sites;

B) *m-Payment Host* - Servidor *Web* que hospeda o *framework* a ser utilizado pelo dispositivo móvel.

3 - *Web* Site Mobile: site móvel que utiliza o *framework* a fim de receber os parâmetros enviados pelo *browser* do dispositivo móvel;

4 - Login: Parte do *Web* site que solicita ao usuário a fim de identificar e autenticar;

5 - DB *Connection Pool*: *pool* de conexões ao banco de dados, implementado utilizando .NET ADO, que permite que conexões ao banco de dados sejam compartilhadas entre diferentes requisições, possibilitando um aumento de performance no acesso aos dados;

6 - *DataBase*: banco de dados SQL Server 2008 que armazena todos os dados do usuário, como cadastro de clientes, produtos e pedidos. Os dados críticos como número de cartão de crédito não são guardados no banco de dados, comunicamos diretamente com a provedora de serviço pagamento;

7 - Pagamento: o pagamento é acionado após a confirmação feita pelo usuário, dessa maneira, a comunicação com o provedor de serviço de pagamento é feita, enviado os dados a unidade certificadora para aprovação.

11 - QR Code resposta: após a confirmação de pagamento, um QR Code resposta é gerado na tela do dispositivo móvel do usuário, com um número de recebido.

C) Provedor de serviço de pagamento - passo essencial de comunicação com a operadora de cartão de crédito.

8 - Unidade Certificadora: provê o certificado SSL e comunica com a operadora de responsável pelo serviço de pagamento.

D) SMS *Gateway* - *Gateway* para envio de mensagens SMS aos clientes:

9 - mytalk.com.br: Host que hospeda o serviço Mytalk SMS, um Gatway para envio de mensagens SMS.

E) SMTP Host - Host que hospeda servidor de e-mail:

10 - SMTP Server: Servidor de e-mail para envio de mensagens eletrônicas aos clientes.

F) TVD - Plataforma de TV Digital

12 - SET-Top Box: Conversor de TV Digital, TV com conversor integrado, notebook ou celular que executa as aplicações de TVD desenvolvidas. A aplicação T-Commerce que se comunica com o QR Code fica dentro do aparelho.

13 - QR Code: Código 2D retornado do *Web Service* (SOA) que aparece no aparelho de televisão.

G) WS Host - Host que hospeda o serviço de criação do código 2D

14 - SOA: *Web Service* que busca do *Web* site o código 2D de determinado produto ou serviço.

4.3 Tecnologias de *Software* Utilizadas para Desenvolvimento do *Software*

4.3.1 .NET *Framework*

O .NET *Framework* é uma nova plataforma que simplifica o desenvolvimento de aplicações para o ambiente altamente distribuído da *Web*, Desktop, *Mobile* e mais [Microsoft 2012]. Os objetivos principais listados são:

A) Fornecer um consistente ambiente de programação orientada a objetos.
B) Fornece um ambiente de execução de código que minimize os conflitos de versionamento e empacotamento/distribuição.
C) Prover um ambiente de execução de código que garanta a execução segura do código, incluindo código criado por terceiros.
D) Prover um ambiente de execução de código que elimine os problemas de desempenho de ambientes interpretados ou de scripts.

Os dois principais componentes do .NET *Framework* são o CLR e o Class Library.

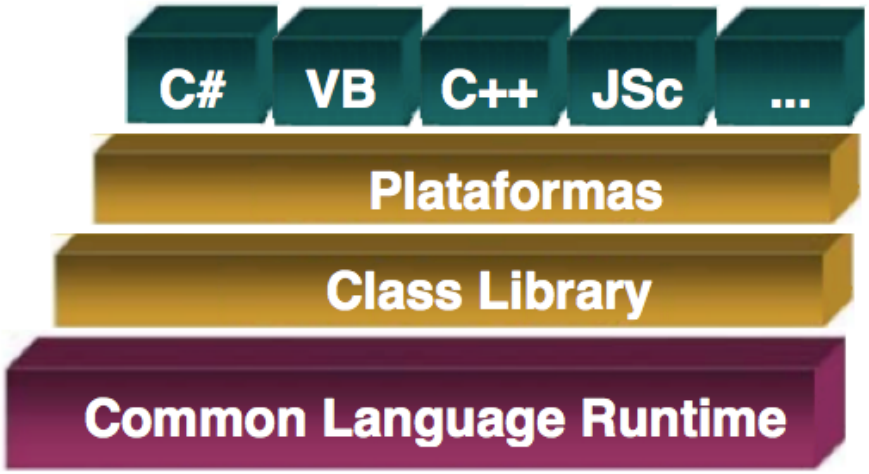

Figura 41 - Representação .NET *Framework* - Fonte: [Microsoft 2012]

CLR

O CLR (*Common Language Runtime*) gerência a memória, as *threads*, a verificação da segurança, a compilação e o código em tempo de execução. Todo o código gerado pelo desenvolvedor, quem executa é o CLR quer usando a linguagem VB, C#, C++, JScript, F#, Cobol.NET e outras [Microsoft 2012].

Todas as linguagens são compiladas para a MSIL (*Microsoft Intermediate Language*) que em seguida é convertido em código nativo durante sua primeira execução, como uma JVM (*Java Virtual Machine*).

Podemos dizer então que o CLR é o coração do .NET *Framework*, e, apesar disto, ele trabalha de forma oculta, sendo que as partes que aparecem do .NET *Framework* são as classes que usamos na nossa aplicação ou *framework*. *Framework* porque podemos fazer outros *framework*'s a partir do .NET *Framework* [Macoratti 2010].

Class Library

A *Class Library* (Livraria de Classes) é uma coleção de classes orientadas a objeto de tipos reutilizáveis e integradas com o CLR. O .NET *Framework* pode ser usado para criar os seguintes tipos de aplicações e serviços:

A) Aplicações do tipo console.
B) Baseadas em scripts.
C) Windows Forms / WPF.
D) ASP.NET (*Web*).
E) Windows Services (Serviços).
F) XML *Web Services*

O .NET *Framework* é então um poderoso ambiente de desenvolvimento que consiste de vários componentes e serviços combinados. É constituído de milhares de classes (umas 6 mil) que provêm toda funcionalidade que antes você encontrava quer no Windows quer no Visual Basic [Microsoft 2012].

Com tantas classes, para encontrar aquela que você precisa pode ser um problema. Para facilitar a vida do desenvolvedor, o .NET *Framework* é organizado de forma hierárquica com nomes (*Namespaces*) que indicam uma determinada ramificação.

Plataformas

Abaixo das linguagens, existem várias plataformas; ou seja; o desenvolvedor pode criar aplicativos para plataformas diferentes como: Mobile, *Web*, Desktop, *Web Service* e Windows Services usando linguagens diferentes dentro de um mesmo projeto. Isso porque o código compilado vira um só, que é a linguagem de máquina.

4.3.2 Padrão MVC

Quando a aplicação ou *framework* começa a crescer, é necessário criar um padrão ou padrões de desenvolvimento com camadas. A complexidade da aplicação criado para o ambiente *Web* veio crescendo de maneira estrondosa, sem padrão de organização, reuso, eficiência e de manutenção.

O MVC (*Model-View-Controller*) veio resolver este problema e foi o mais aceito entre os desenvolvedores [Brambilla et al., 2010]. O padrão do desenvolvimento de *software* MVC organiza o quesito camadas e pode ser usada em qualquer tipo de plataforma: *Web*, móvel, desktop e outros.

O MVC foi criado em 1998 por Glenn Krasner e St. Pope (Barros et al., 2007), e a análise da robustez introduzida por Ivar Jacobson prova que qualquer *software* pode ser facilmente deslocado para a arquitetura MVC [Jacobson 1991]. Com o MVC, o reuso do código torna-se mais fácil e simples, a confiabilidade das camadas funcionam e a camada de dados se comporta melhor para acesso ao banco de dados [Brambilla et al., 2010].

O padrão mantém o estado persistente do negócio e fornece ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas. A ferramenta de desenvolvimento utilizada para apoiar o nosso *framework* de pagamento foi o "Visual Studio" da Microsoft. A empresa Microsoft disponibiliza vários componentes e o *Framework* .NET para o verdadeiro funcionamento do MVC.

Em outro artigo estudado e usando o mesmo *Framework*.NET da Microsoft, o autor informa equivocadamente sobre a *View* ser uma classe da própria *interface* da página *Web* [Zheng et al., 2008]. A classe View representa muito mais do que uma simples classe da *interface* da página *Web*.

A parte de acesso a dados pode ser plugada ou desplugada a qualquer hora, isto é, basta virar a chave, por exemplo: a mudança de conexão com o banco de dados Oracle para o SQL Server fica simples. Além de ser um ótimo padrão, o MVC desassocia a camada de negócios e dados da camada de apresentação. Dessa forma, outro tipo de apresentação pode ser construída utilizando as mesmas classes de negócio e dados [Brambilla et al., 2010].

A *Model* é formada por entidades que representam os dados da aplicação. A *View* tem por objetivo realizar a apresentação destes dados e capturar eventos do usuário, representada pela ação da tela. A entidade *Controller* faz ligação entre a *Model* e a *View*, realizando o tratamento dos eventos e alterando a View para representar a nova forma dos dados retornados.

As linguagens e plataformas mais utilizadas no desenvolvimento de sites, que apoiam o padrão MVC são: Asp.net utilizando C#, VB.NET e J#, Objective-C e Java.

A figura 42 mostra o funcionamento e a facilidade de desacoplar as camadas.

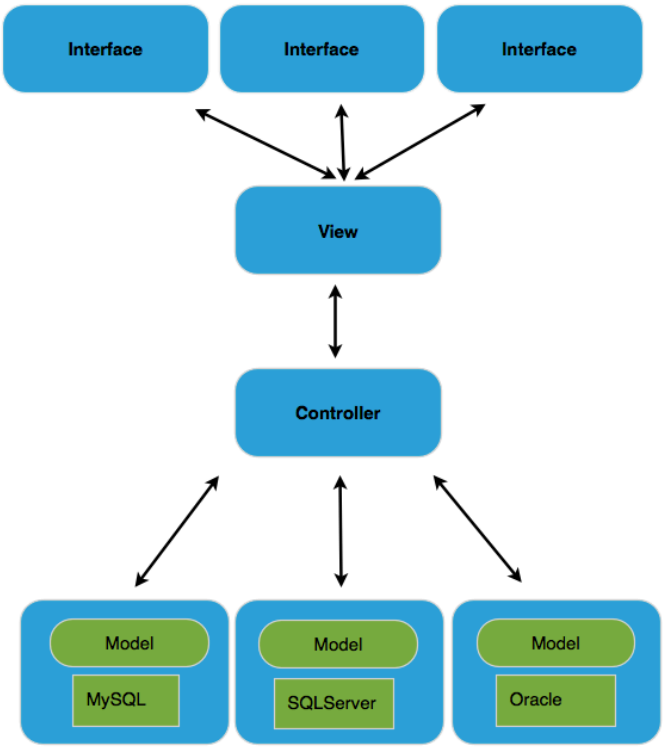

Figura 42 - Padrão MVC - Fonte: [Brambilla et al., 2010].

Em vários artigos analisados, notou-se que a maioria dos autores informam apenas que a aplicação pode ser feita usando Java Sctructs, ou seja, nenhum detalhe é informado no uso de outras linguagens [Selfa et al., 2006].

O MVC demonstra o benefício para aplicativos de interatividade que permitem múltiplas representações de uma mesma informação. Além da interatividade, a informação pode ser mostrada de acordo com o meio de acesso do usuário [Selfa et al., 2006].

A figura 42 mostra que a arquitetura MVC de desenvolvimento de *software* é feita em blocos de fácil desacoplamento. Para desacoplar a parte *View* fica simples quando feito em blocos, ou até então criar uma outra *View* mais específica como para TV Digital.

O *framework* permite integração entre sistemas prontos usando *Web Services*, isto é, em supermercados, lojas, empresas prestadoras de serviços, e centros comerciais os chamados Shoppings.

A figura 49 apresentada a infraestrutura de máquinas/servidores utilizados para o funcionamento do *framework* proposto neste trabalho.

O cliente pode ser acessado através de qualquer dispositivo móvel ou integração através de sistemas prontos. Um *e-Commerce*, computador, TV Digital acessam utilizando *Web Services* como mostrado na figura 50 pelo símbolo ao lado do servidor.

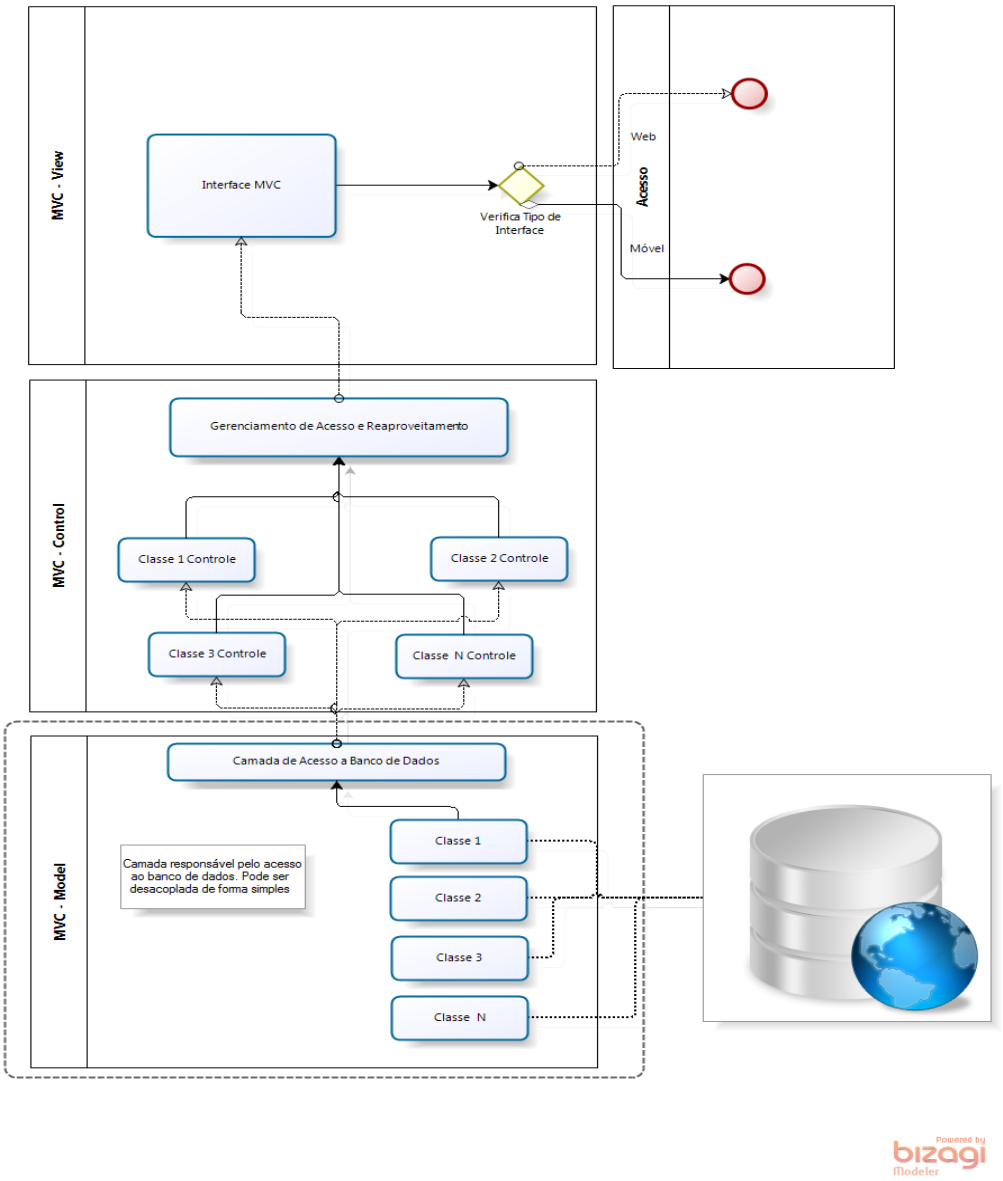

Figura 43 - Arquitetura baseada em MVC

O servidor de e-mail SMTP é responsável pelo envio de e-mail automático após a compra de algum produto/serviço, além de poder mandar algumas promoções específicas aos usuários.

O servidor de SMS é responsável pelo envio de informações específicas para o dispositivo móvel do usuário. Utilizamos uma estrutura criada por mim e disponibilizada no site Mytalk - http://www.mytalk.com.br com integração através de *Web Services* para envio de SMS automático. Esse mesmo site foi utilizado no trabalho da UNB [Filho 2011].

O *DataBase* é um banco de dados local responsável por guardar informações dos usuários como registro de compras, log de acesso e cadastro. Temos ainda um provedor de serviço de pagamento onde fica armazenado saldos, valores e créditos registrados de cada usuário.

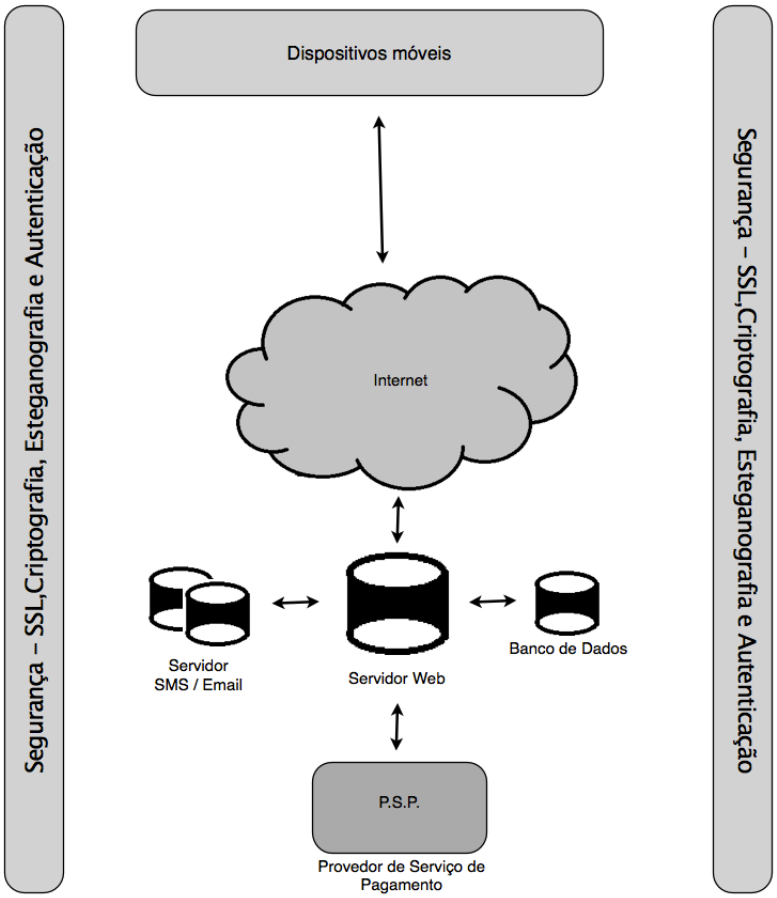

Figura 44 – Arquitetura de pagamento

4.4 Abordagem para Segurança da Informação

4.4.1 Infraestrutura

A figura 45 de infraestrutura mostra que existem vários meios de entrada como TV digital, tablets, lojas de comércio eletrônico e outros. Começando da parte dos clientes, existe uma diferença entre os dispositivos. Aqueles como TV Digital e Computador acessam os dados através do SOAP XML, enquanto os celulares e smartphones acessam direto pela Internet.

Passando pela Internet, o servidor *Web* entra em ação com o objetivo de processar os dados. Note que o provedor de serviços de pagamento está comunicando com o servidor *Web* garantindo ainda a segurança para o usuário.

Toda infra-estrutura está fechada usando SSL (implementação do protocolo HTTP sobre uma camada de segurança, que permite que os dados sejam transmitidos através da conexão criptografada). O *framework* utiliza e solicita ainda a criptografia e autenticação representados na figura 45, verificando a sua autenticidade no servidor e no cliente por meio de certificados digitais.

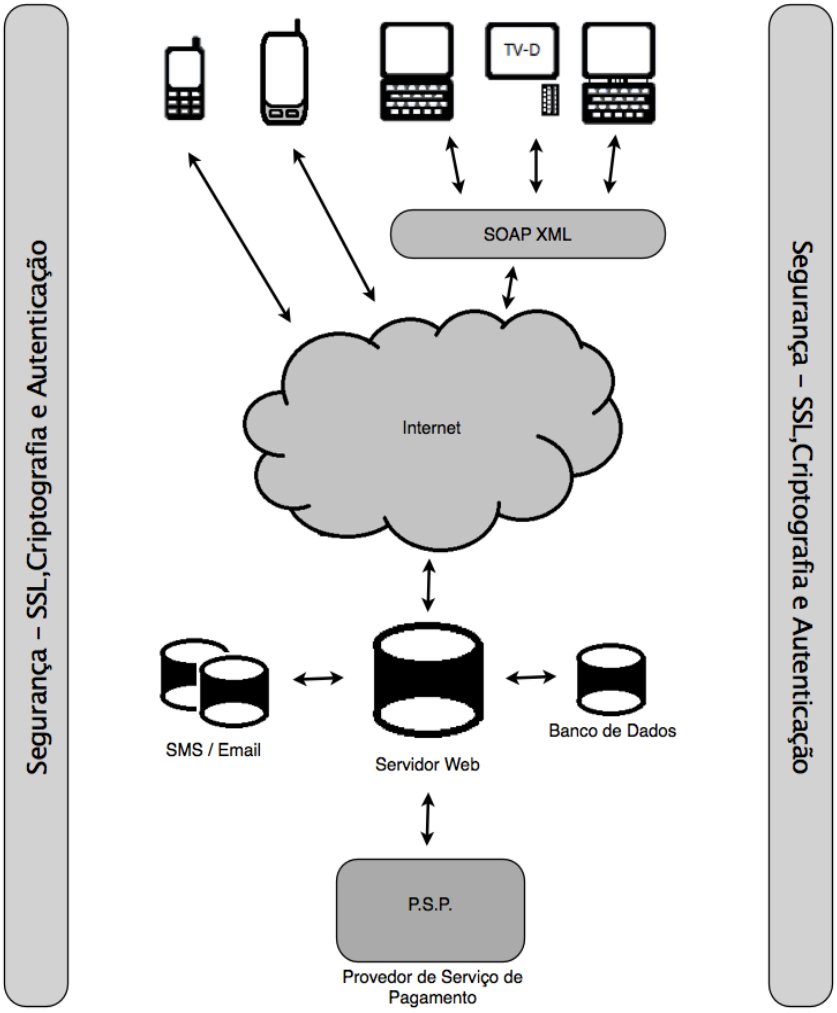

Figura 45 - Infraestrutura de segurança com código 2D

4.4.2 Processo de Esteganografia

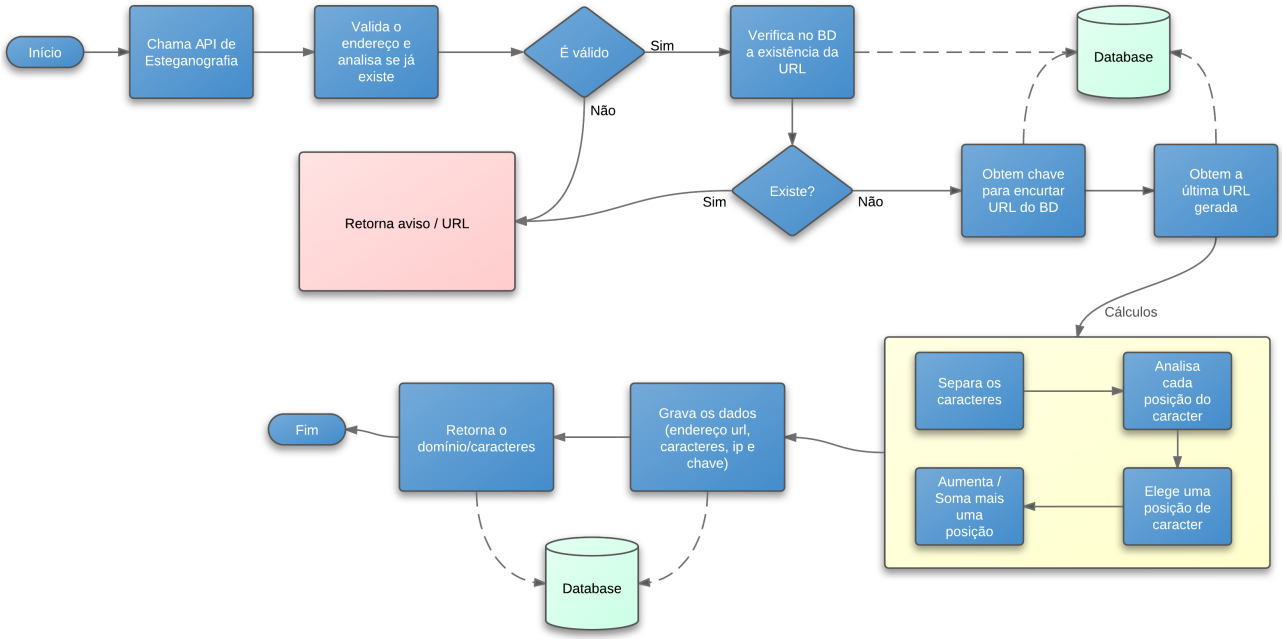

Figura 46 - mostra o fluxograma para a geração da URL esteganografada.

Fluxo de Esteganografia - Geração

Após a criptografia dos dados a API de esteganografia é chamada e os seguintes passos são executados:

Passo 1: A API verifica se o endereço é válido.
Passo 2: Se o endereço for válido, a API verifica se o mesmo endereço existe no banco de dados. Se o endereço não for válido, a API retorna um aviso para o usuário.
Passo 3: A API verifica se o endereço URL existe no banco de dados. Se existir ele retorna a URL gravada.
Passo 4: Se o endereço não existir, a API obtem no banco de dados a chave GUID e a última URL gerada.
Passo 5: A API faz os cálculos para criar a URL. Separa os caracteres da URL a partir da barra, analisa cada posição, elege uma posição aleatória, aumenta ou soma mais uma posição.
Passo 6: A API grava os dados no banco de dados (endereço url, caracteres, ip e chave).
Passo 7: Retorna o domínio e os caracteres.

Fluxo de Esteganografia - Leitura

A figura 47 mostra o fluxograma de como é feito o redirecionamento da URL esteganografada. Esse fluxograma entra em ação após a digitalização do código 2D.

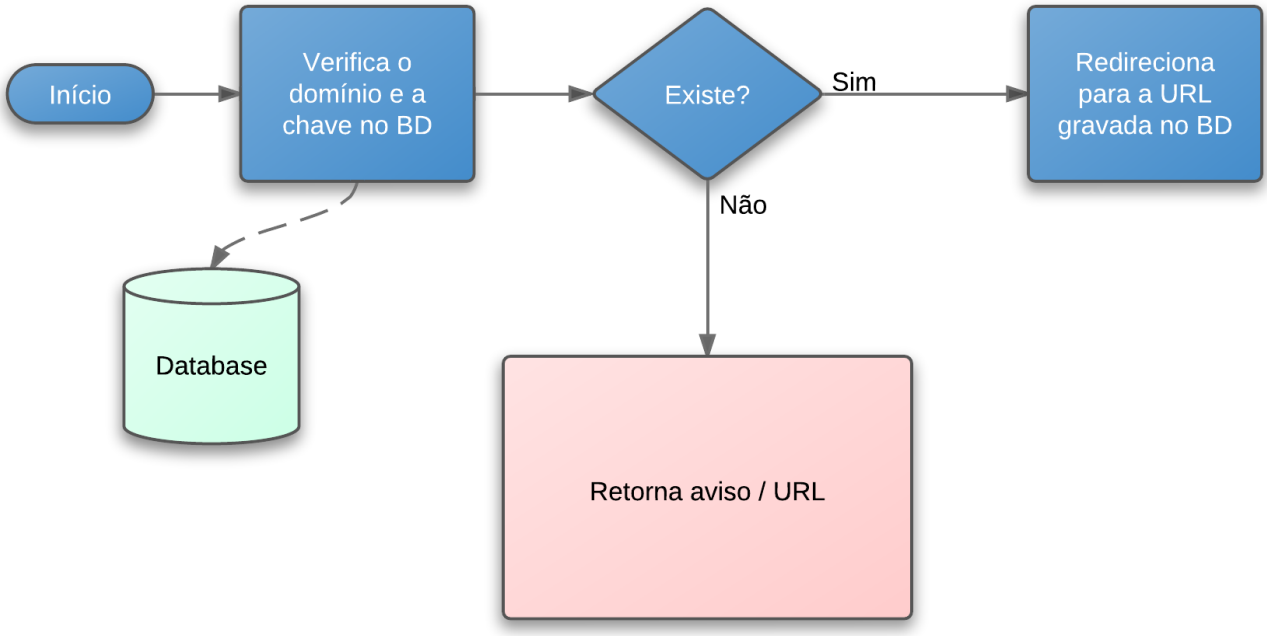

Figura 47 - Fluxo de Esteganografia - Redirecionamento

O fluxograma de redirecionamento é mais simples do que o fluxo da criação. São executados os seguintes passos:

Passo 1: verifica o domínio e a chave recebidos diretamente no banco de dados.
Passo 2: Se existir, redireciona para o link gravado no banco de dados.
Passo 3: Se não existir, retorna um aviso para o usuário.

4.5 Fluxograma do Processo de Compra

4.5.1 Processo de leitura do código 2D

A figura 48 detalha um pouco mais sobre o que existe dentro do código 2D, diferentemente de outras soluções vistas anteriormente, cujo os dados estão abertos e disponíveis a qualquer pessoa. O texto interno possui informações criptografadas que por sua vez envia ao servidor central.

As informações criptografadas são encaminhadas pela rede de dados do dispositivo, (pondendo ser 2G, 3G, Wi-fi ou qualquer outra rede futura que conecte o dispositivo à Internet). O código 2D não pode ser gerado por qualquer site, isso porque a criptografia das informações são verificadas logo no primeiro acesso.

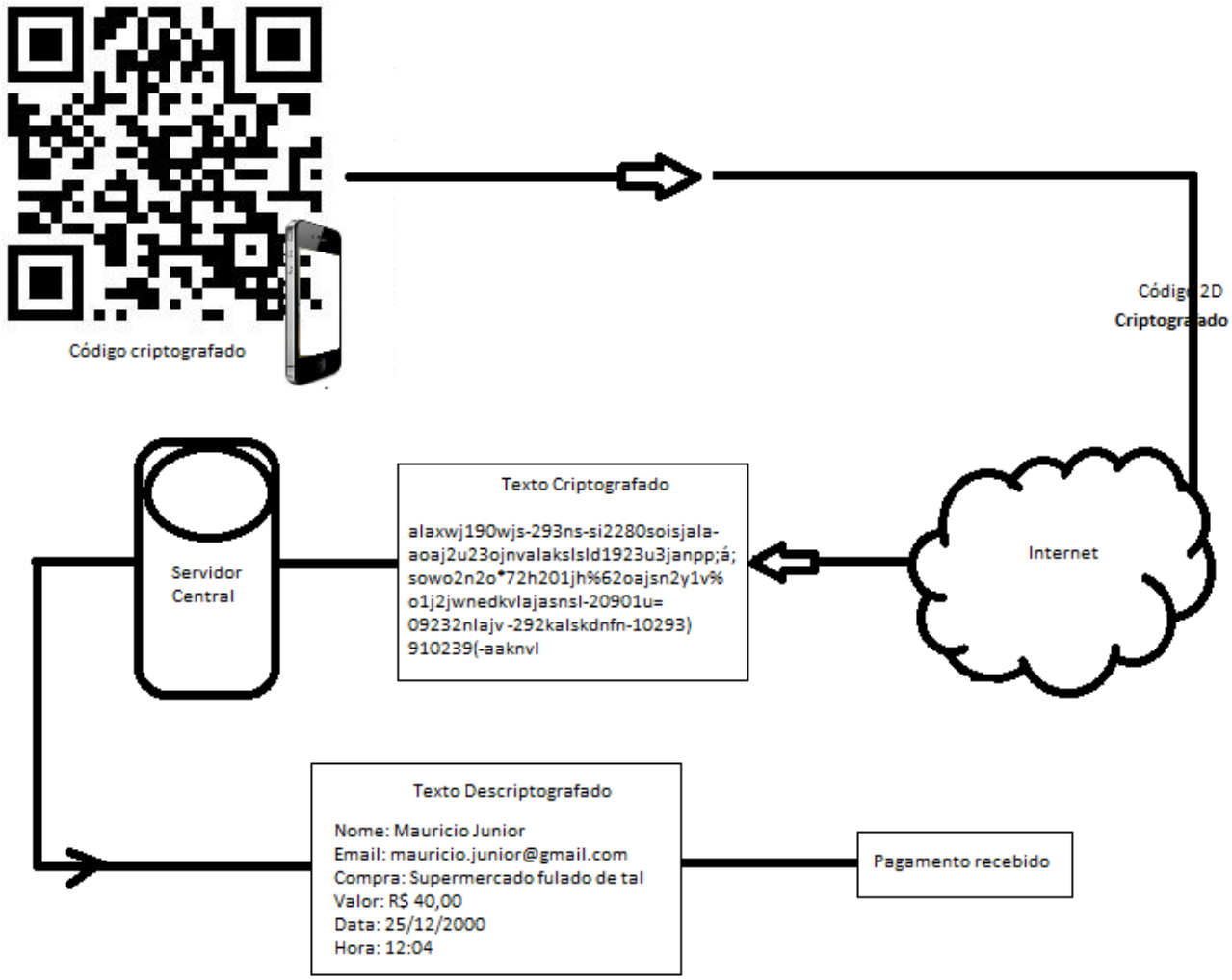

Figura 48 - Processo de leitura do código 2D

O apêndice B mostra a informação sendo decriptografada na tela principal. Em continuação, o *framework* recebe os dados criptografados e processa as informações. O

processo detalha a forma de envio dos dados criptografados, desde o envio pelo usuário móvel até a autorização de compra temporária.

Antes de dar continuidade na operação do usuário, o *framework* verifica o número único do dispositivo móvel (PIN) gravado no momento do cadastro. Se a informação do PIN, autenticação do usuário e senha possuírem integridade positiva, o servidor central autoriza a operação.

Além de toda essa preocupação com a segurança dos dados e operações, preocupamos também com os possíveis ataques chamados de inject sql. Todo *framework* está preparado para não aceitar esse tipo de tentativa.

4.5.2 Processo de compra de produtos ou serviços

A figura 49 mostra o fluxo do processo de uso pelo usuário, começando do lado direito para a esquerda, escaneando o código 2D até o momento da finalização da compra. Todo o fluxo é realizado em segundos pelo dispositivo móvel emulando uma compra real de um livro.

O usuário escanea o código 2D com informações criptogradas usando o dispositivo móvel. Automaticamente o *browser* do dispositivo é redirecionado para o aplicativo *Web* mobile passando informações.

O servidor central recebe os dados e solicita a autenticação do usuário a fim de criar uma sessão temporária. Após a autenticação, o *framework* mostra as informações na tela do dispositivo móvel com os botões "aprovar" e "cancelar" a compra.

Clicando no botão "aprovar", o saldo/cartão de crédito do usuário são verificados através do "provedor de serviço de pagamento"; tendo saldo suficiente, o valor do produto ou serviço é deduzido e a compra realizada.

O *framework* envia o SMS e email para o dispositivo do usuário com valores e no qual confirma a compra. Também aparece na tela do usuário um novo código 2D de confirmação.

Foi realizado testes em uma loja de livros e todo o processo foi realizado em poucos segundos, sendo aceito pelos usuários.

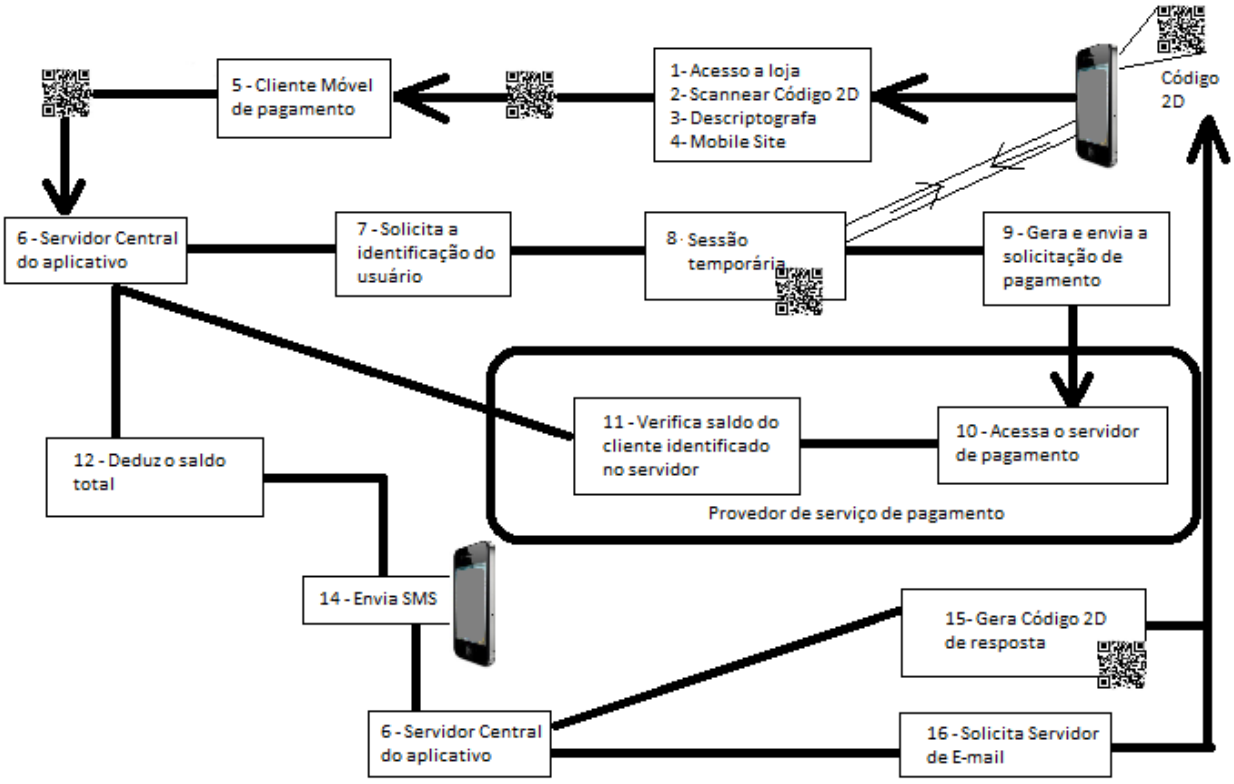

Figura 49 - Fluxograma do processo de compra de produtos/serviços

4.5.3 Processo de compra de créditos

O processo de compra de créditos para uso é bem simples e ao mesmo tempo seguro. O usuário pode utilizar o seu computador, tablet, celular, smartphone e a TV Digital (após integração) para colocar créditos específicos de compra.

Utilizamos de um provedor de serviços de pagamento para que o usuário possa comprar créditos utilizando o código 2D criptografado ou acessando um *Web* site do provedor.

Podem ser adicionados créditos utilizando o cartão de crédito, débito ou boleto. Os valores aprovados até o momento são: R$ 10,00, R$ 25,00, R$ 50,00 e R$ 100,00. A figura 56 mostra como adicionar créditos.

Lembrando que, tirando os celulares e smartphones, todos os outros dispositivos precisam da iteração com o SOAP XML em primeiro lugar.

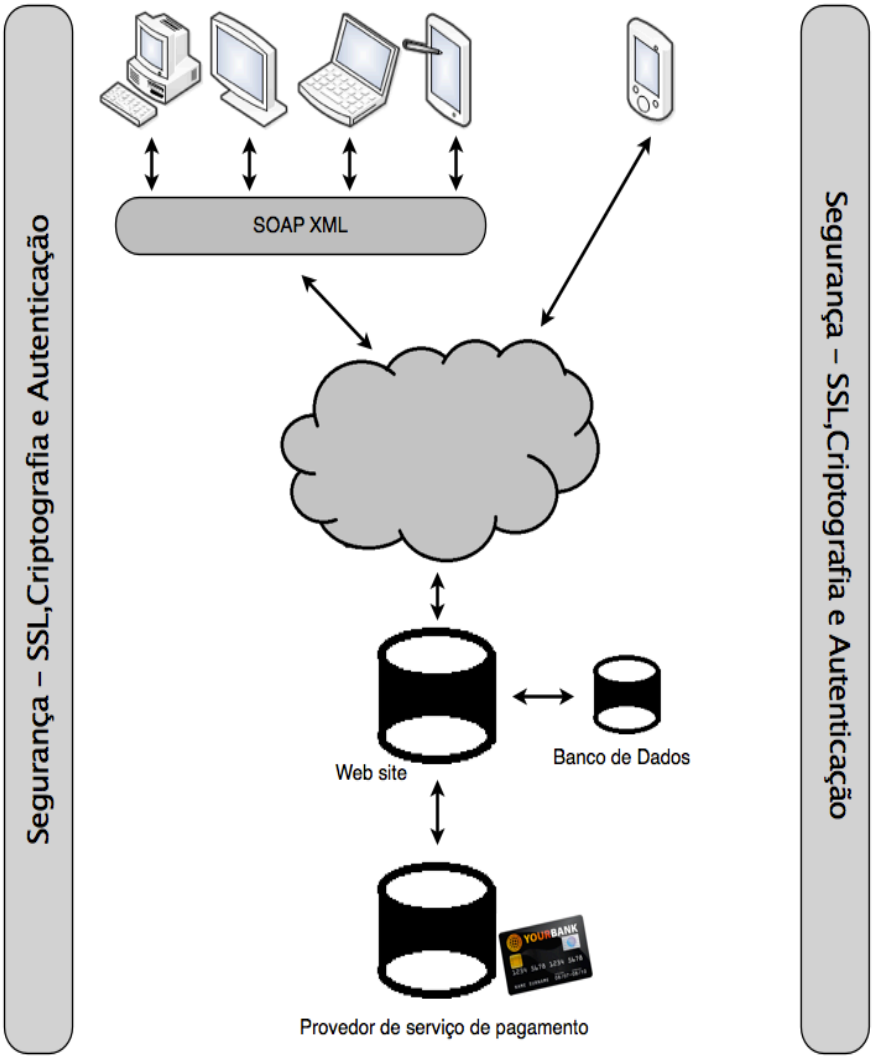

Figura 50 - Fluxograma do processo de compra de créditos

4.6 Comunicação entre a TV Digital e o *Framework*

Tivemos acesso a trabalho realizado pelo Manoel Campos da Silva Filho e seu orientador, que trata de comércio pela TV Digital (*T-Commerce*) usando a tecnologia brasileira Ginga-NCL e a linguagem de programação Lua [Filho 2011].

Todo o processamento ou envio de informações fica dentro do Lua onde o responsável pela exibição é o Ginga-NCL. No trabalho proposto, foi criado uma arquitetura SOA de comunicação entre dispositivos, inclusive foi usado o aplicativo feito por mim para envio de SMS pela TV localizado no site www.mytalk.com.br.

A presente dissertação (de pagamentos) possui um *framework* que disponibiliza *Web Services* para que seja realizado também o pagamento pela TV Digital.

A figura 51 mostra a comunicação entre a TV Digital e o *framework* proposto na presente dissertação. Note que não é muito diferente da proposta original, mas a solicitação é enviada pela própria TV para aparecer o código 2D ou a solicitação de pagamento [Filho 2011].

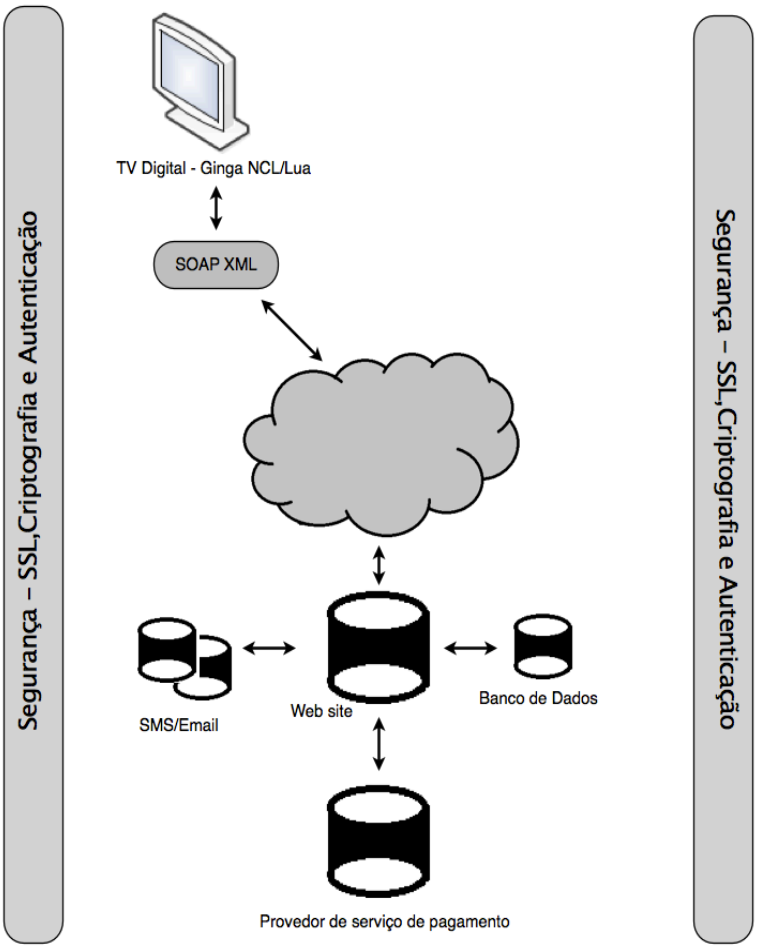

Figura 51 - Arquitetura de comunicação TV Digital e o *framework* de pagamento

O *framework* possui o mesmo funcionamento de envio de SMS pela TV Digital feito por [Campo 2011], mas com algumas adaptações. Ao invés da TVD esperar uma *string* informando que a mensagem foi enviada, o *framework* retorna uma imagem *QR Code* para ser mostrada na tela do aparelho de TV Digital.

Segurança

Todo o ambiente de comunicação, começando da ponta até o momento de pagamento é seguro; isso porque o acesso já começa com o SSL, criptografia e autenticação de dados. É necessário informar usuário e senha no momento de chamar o *Web Service*.

Adaptação

Duas adaptações mínimas precisam serem feitas no código Lua da TVD para o devido funcionamento e comunicação:

1- Chamar a *URL* do *Web Service.*
2- Passar usuário e senha para autenticação.
3- Alterar o código Lua da aplicação de TV Digital para exibição de imagem usando comandos HTML.

4.7 Apresentação de Telas do *Framework*

O padrão de código 2D utilizado no presente trabalho foi o *QR Code,* desenvolvido pela empresa DENSO no Japão. Escolhemos este padrão por alguns motivos:

a) Maior capacidade numérica;
b) Maior capacidade alfa-numérica;
c) Maior capacidade binária;
d) Maior capacidade na utilização de símbolos Kanji;
e) Padrão internacional, ISO (*International Organization for Standardization*), JIS (*Japanese Industrial Standards*) e AIM (*The Association for Automatic Identification and Mobility*).

Figura 52 - *Framework* - QR Code

A figura 53 mostra três plataformas (iOS, Windows Phone e Android) utilizando o *Framework* que, depois de digitalizar o QR Code, por segurança é necessário entrar com o login e senha. Imagine que o código está em um produto na prateleira, ou em um serviço na Internet, ou na sua TV Digital ou na entrada do cinema.

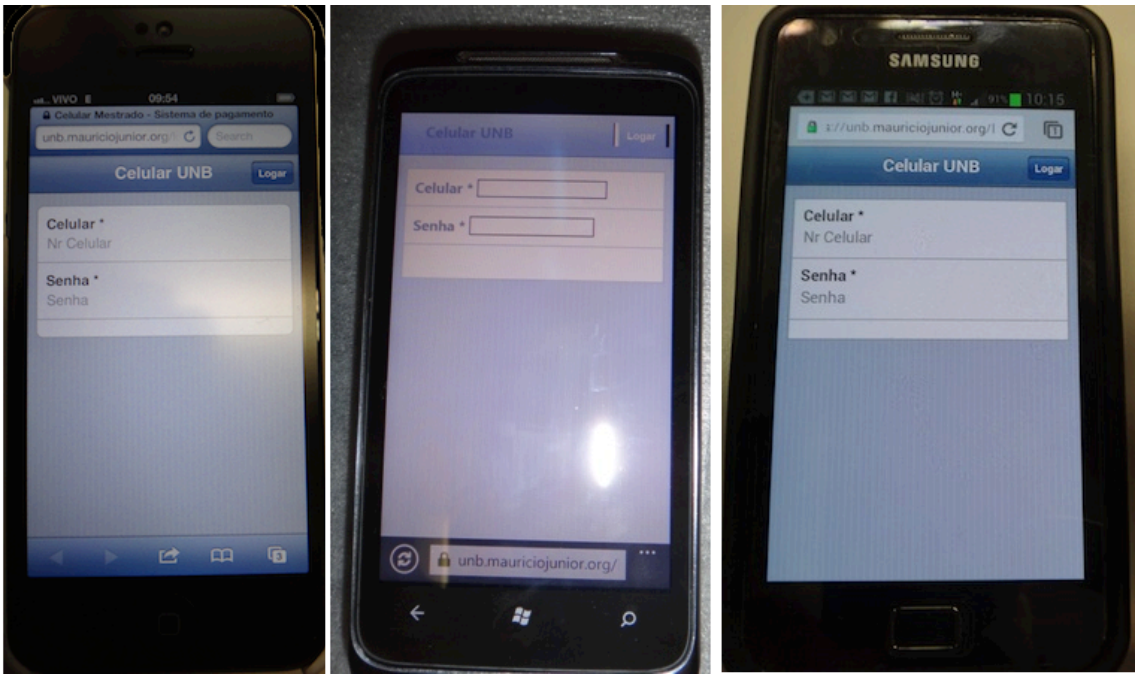

Figura 53 - *Framework* - Passo 1

Note que o *framework* de pagamento está utilizando o protocolo informado em capítulos anteriores.

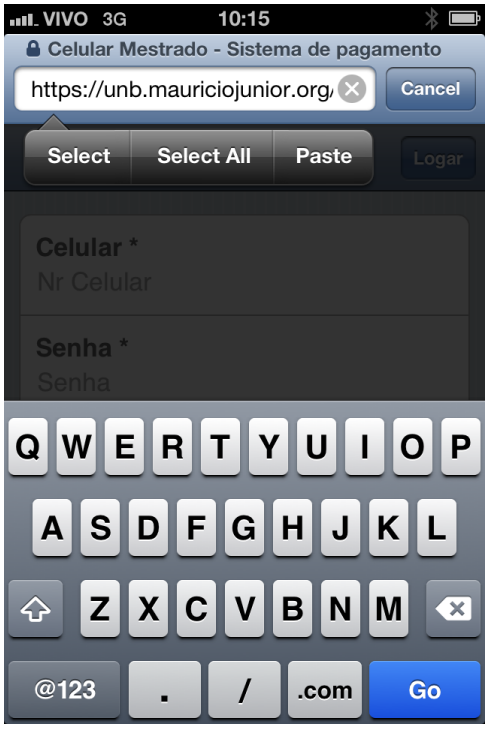

Figura 54 - *Framework* - Uso do protocolo SSL

Após logar com seus dados, você *aprova* ou não a compra. É necessário sempre identificar o valor do produto ou serviço antes de aprovar. A figura 55 mostra o *Framework* sendo utilizado nas três plataformas mais utilizadas nos dias de hoje.

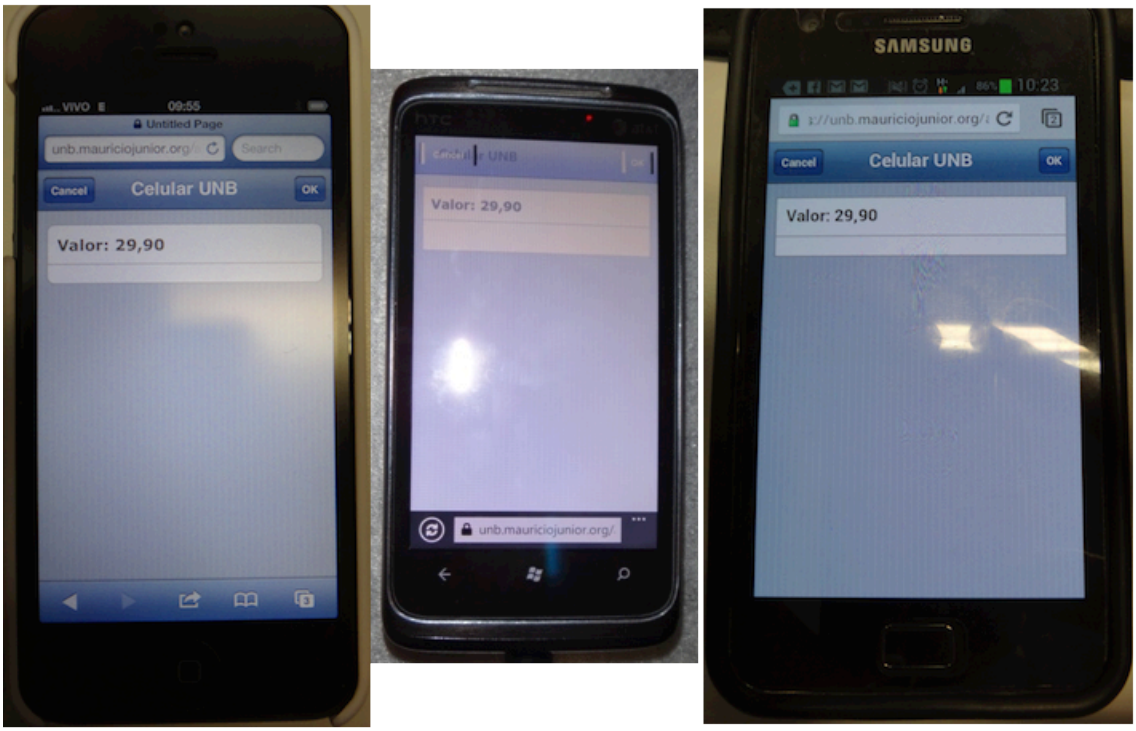

Figura 55 - *Framework* - Passo 2

Para aprovar é necessário clicar no botão *OK*, após o clique, o valor é deduzido do seu saldo no provedor de serviços de pagamento de forma automática, gerando um novo QR Code de identificação e mandando SMS de confirmação, figura 56.

Figura 56 - *Framework* - Passo 3

Em poucos segundos , o SMS é enviado com o valor de compra e o sucesso da transação. O SMS é enviado a qualquer operadora, ou seja, sem distinção de empresa ou operadora. Veja a figura 57.

Figura 57 - *Framework* - Passo 4

Caso o usuário queira vincular seus pagamentos ao seu cartão de crédito de forma automática, basta inserir os dados do cartão, conforme mostrado na figura 58.

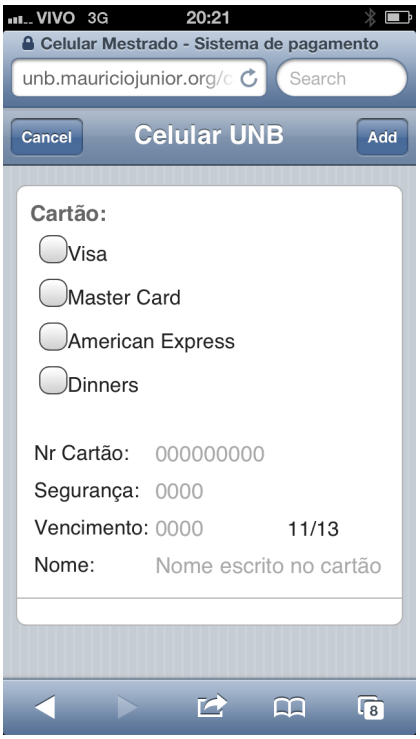

Figura 58 - *Framework* - Cadastro de cartão de crédito

A figura 59 mostra a integração pelo aplicativo de TV Digital. Basta posicionar o dispositivo móvel usando aplicativo de leitura [3G Vision 2010] para comprar o produto distinto.

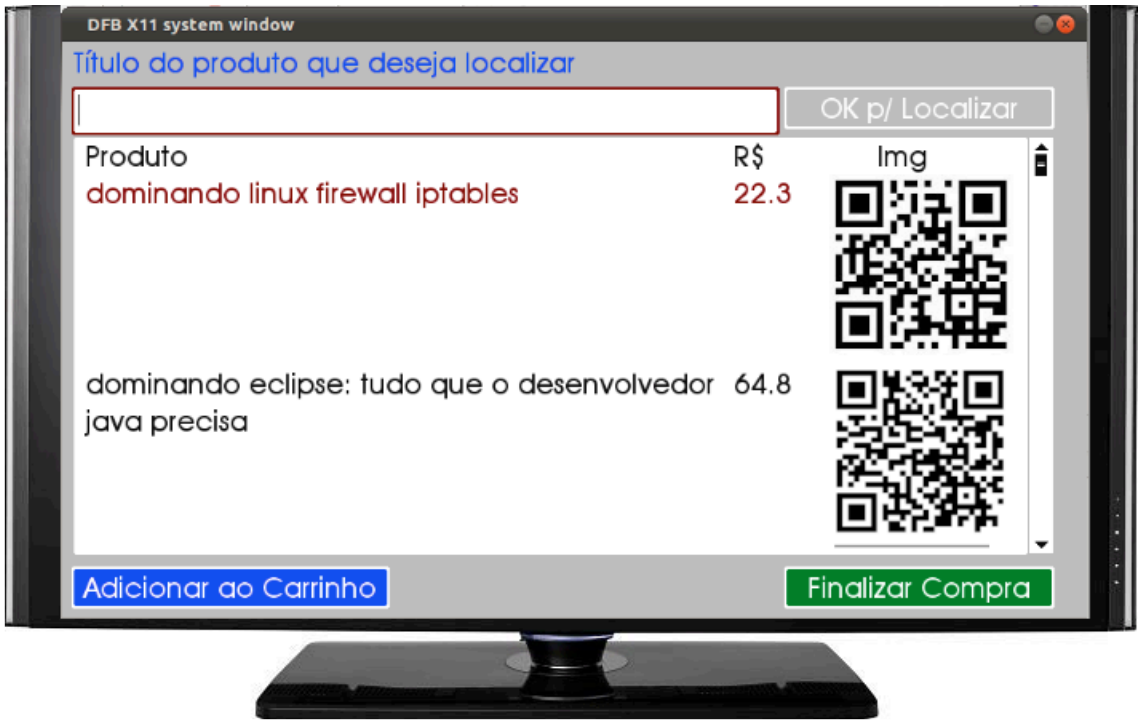

Figura 59 - Integração TV Digital

Caso o usuário queira comprar o produto utilizando a própria TVD, basta adicionar o produto ao carrinho e finalizar a compra. Para finalizar, é necessário logar com usuário e senha, figura 60.

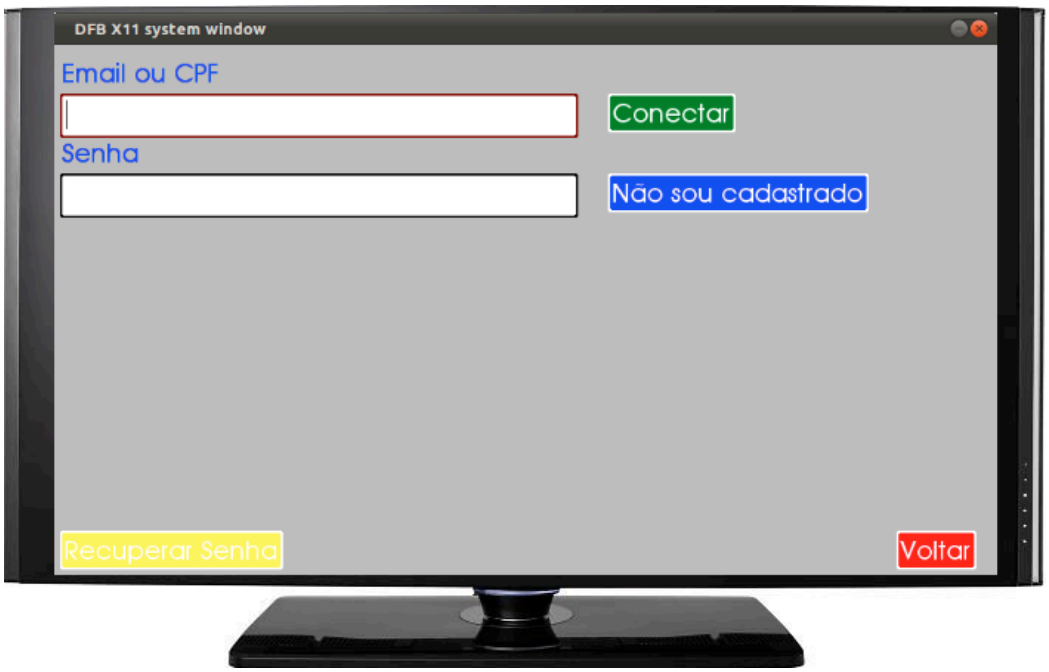

Figura 60 - Solicitação de usuário e senha.

4.8 Métrica

A métrica utilizada para avaliação considera a quantidade de memória utilizada no dispositivo móvel, tomando por base, para fins de comparação, dispositivos baseados em Android (utilizando tecnologia Java) e o nosso *framework* (que pode ser utilizado em qualquer dispositivo).

Usamos os seguintes dispositivos móveis e respectivos sistemas operacionais: o iPhone (com iOS 6.1.3), Samsung Galaxy S2 e Galaxy S3 (com Android 4.1) e Windows Phone - HTC com WP 7.8. É requerido para todos os dispositivos móveis a conexão à Internet através da rede 2G, 3G ou Wi-Fi.

Criamos um aplicativo em Java com as mesmas funcionalidades de *m-Payment*, a fim de comparar a quantidade de memória utilizada. Utilizamos a JVM (*Java Virtual Machine*) versão 7.17, junto da ferramenta Eclipse (*Android Developer Tools*). É necessário instalar o aplicativo no dispositivo, onde desenvolvemos uma camada com o objetivo de se conectar à nossa API e completar a operação. Essa casca feita com Java mostrou que o consumo de memória ficou maior do que se utilizasse apenas o nosso *framework*. Dessa maneira, percebemos que o emprego de Java tornou o *framework* mais pesado em termos de consumo de memória. Da figura 42, a implementação realizada engloba os módulos *Interface*, *View* e *Controller*.

A figura 61 mostra o desempenho com base nas tecnologias utilizadas, Java e HTML5 (utilizada pelo nosso trabalho). O gráfico mostra que a tecnologia Java utiliza mais memória do que a tecnologia HTML5.

Nos testes realizados utilizando os aparelhos Samsung Galaxy S2 e S3, a máquina virtual Java precisou de 20 MB para iniciar o aplicativo e depois de iniciado, em momentos esporádicos o consumo aumentou e diminuiu em torno de 1 MB. Comparado com o uso do nosso *framework*, o consumo ficou mais baixo pelo fato de que não usamos JVM, usamos apenas o *browser* que vem por padrão instalado no sistema operacional. O Android possibilita a verificação no consumo de memória de cada aplicativo utilizado no aparelho, e esse foi o sistema que utilizamos.

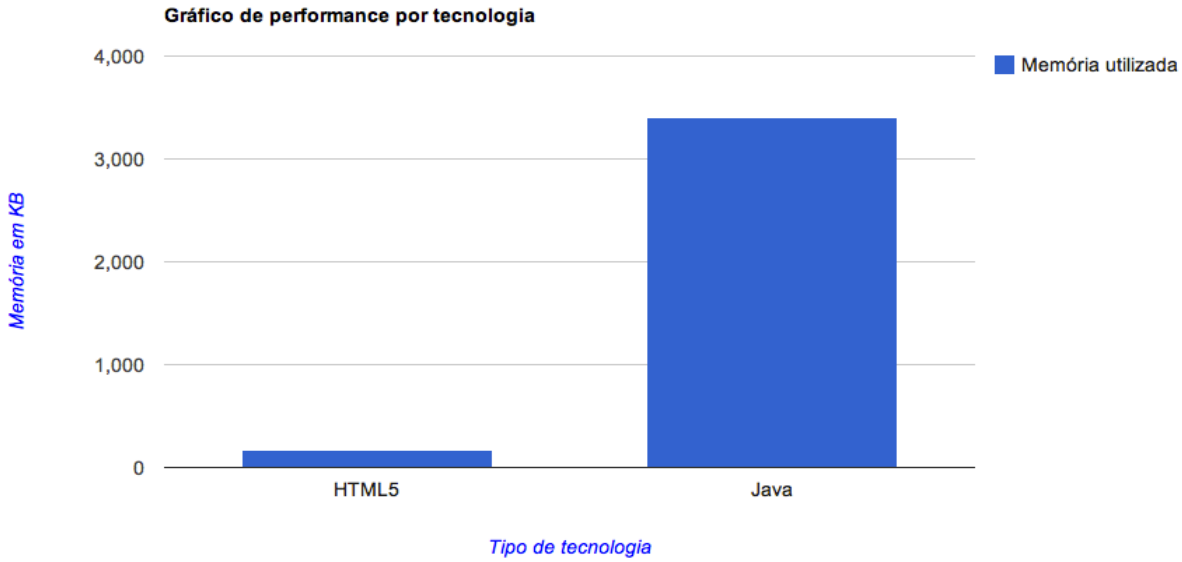

Figura 61 - Comparativo entre Tecnologias

A figura 62 compara o nosso *framework* utilizado em diversas plataformas com a tecnologia usada por diversos autores relacionados anteriormente. Em todas as plataformas, o nosso *framework* continou utilizando a mesma quantidade mínima de memória; enquanto que o Java começou com mais de 21MB de memória no Android. Nota-se que o uso de menos memória deixa o nosso *framework* mais rápido no momento da utilização.

No dispositivo com Android, a JVM precisou de 21 MB de memória enquanto que no dispositivo Symbian o consumo ficou um pouco mais de 13 MB. No Windows Phone o consumo ficou em 12 MB de memória.

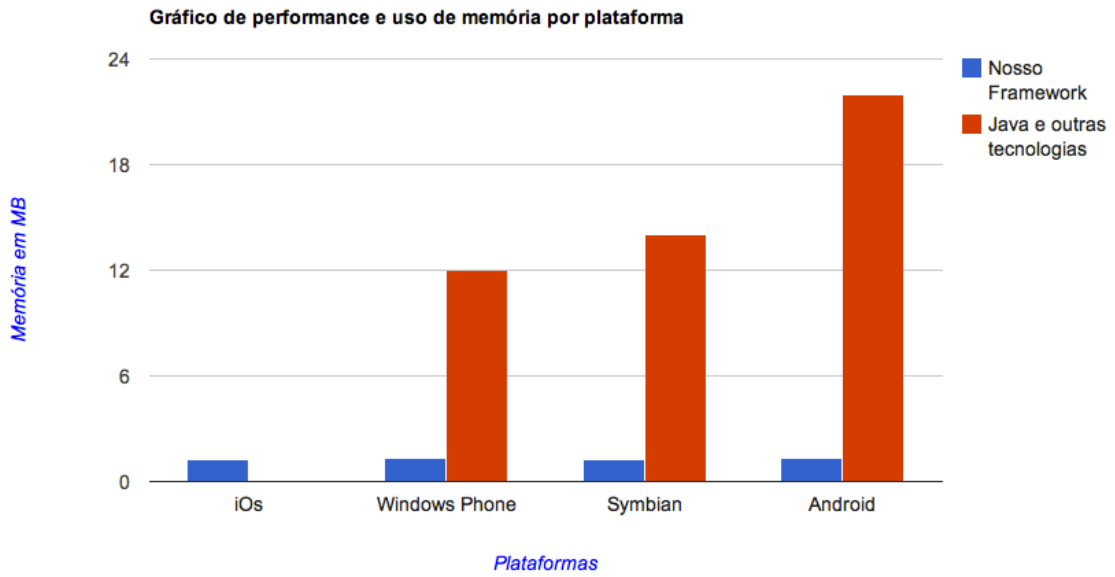

Figura 62 - Comparativo entre Plataformas

A figura 63 faz uma comparação referente aos testes realizados gravando a quantidade de memória consumida.A primeira fase de testes é realizada de 1 a 30 vezes, onde o nosso *framework* se manteve com o mínimo de memória possível, enquanto que no Java iniciou utilizando acima de 20 MB de memória. A segunda fase de testes contou de 31 a 60 testes onde houve um aumento mínimo de memória no nosso *framework* e um aumento considerado usando Java. Os próximos testes realizados, o nosso *framework*se manteve utilizando a mesma quantidade de memória enquanto o Java subiu mais um pouco até atingir os 30 MB.Note que o nosso *framework* não passou de 3 MB em todos os 120 testes realizados.

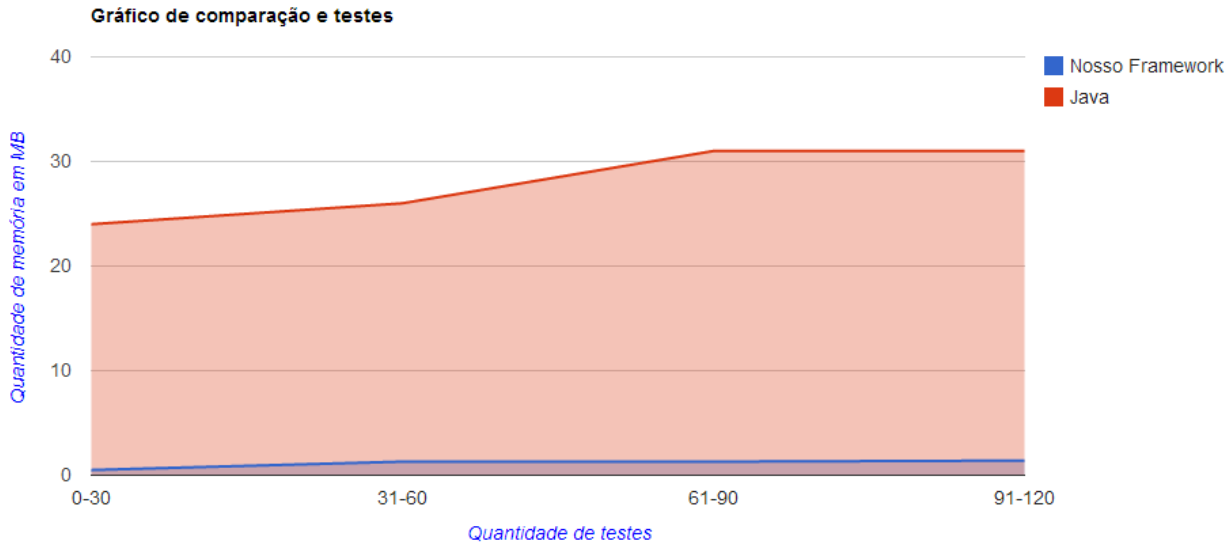

Figura 63 - Comparação e testes

Dessa forma, no tocante a desempenho e uso de memória, o nosso *framework* pode ser considerado melhor do que os outros mostrados pelos autores referenciados. O nosso *framework* pode ser utilizado em qualquer plataforma móvel e não precisa ser instalado localmente, enquanto que os outros precisam de instalação local pois utilizam Java e consomem mais memória. Lembramos que não são todas as plataformas móveisque aceitam a instalação do Java, e as que aceitam a manutenção seria mais trabalhosa.

Capítulo 5

5. Conclusão e Trabalhos Futuros

5.1 Conclusão

Com o aumento rápido do número de usuários usando tecnologia sem fio, soluções de *m-Commerce* e *m-Payment*, têm ganhado mercado. No entanto, uma das questões mais críticas na construção do *m-Commerce* ou *m-Payment* é a entrada de dados [Kato et al., 2005].

Usando dispositivos móveis com câmera, os consumidores podem facilmente inserir um código de barras 2D de um produto por varredura na loja, além de encontrar informações mais detalhadas, incluindo produtores, data de colheita, data de envio e produtos químicos agrícolas. Ademais, códigos 2D são úteis no pós-venda, incluindo o rastreamento de produtos, transporte e entrega [Kato et al., 2005].

A cada dia, mais produtos e serviços são identificados por meio de códigos de barras 2D dentro do comércio, portanto há uma clara necessidade de construir novos sistemas de pagamento móvel, a fim de suportar transações seguras.

Vista essa necessidade, descrevemos um *framework* de pagamento, operacional e de fácil interação.

Vimos que a segurança é essencial ao usuário, e além dela, usamos diferentes provedores, que ajudam a popularizar o serviço de pagamento móvel.

Fizemos várias pesquisas e analisamos artigos sobre *m-Payment* o qual possui vantagens e desvantagens citadas anteriormente.

O próprio *framework* gera o código 2D criptografado, o que facilita a compra de um produto por parte do usuário utilizando a sua carteira móvel.

Padrões foram utilizados com características citadas em seções anteriores, não importando o sistema operacional e *hardware*; isso ajuda a difundir o *framework* desenvolvido. Dessa forma, o *framework* pode ser usado em qualquer dispositivo móvel que tenha acesso a Internet e câmera.

A compra de créditos pode ser feita usando o próprio cartão de crédito e débito através do provedor de serviço de pagamento.

O *framework* resultou em um aplicativo capaz de realizar pagamentos pelo dispositivo móvel, receber créditos e fazer interações com outros tipos de plataformas como receptores de TV Digital.

As várias possibilidades de interação entre plataformas de comércio virtual integradas por meio do *framework*, permite caracterizar o *u-Commerce*, capaz de realizar transações comerciais em operações independentemente de dispositivos.

Assim, considera-se que os objetivos propostos nesta dissertação foram alcançados, apresentando um aplicativo funcional, desde o ambiente de desenvolvimento, até a realização da transação.

5.2 Trabalhos Futuros

Como trabalhos futuros, pretende-se concluir a implementação de relatórios gerenciais para o cliente, resultando assim um melhor controle das compras e gastos. Pretende-se também concluir a comunicação com o provedor de serviço de pagamento com a empresa CIELO aceitando todos os cartões de crédito possíveis.

Pretende-se incluir o tratamento de exceções para permitir que o *framework* possa emitir mensagens amigáveis ao usuário quando uma requisição ou operação falhar.

Pretende-se estender as funcionalidade do *framework* para ser usado em redes sociais como o Facebook, Orkut e outras de sucesso, possibilitando assim o pagamento de produtos e serviços pela rede social usando o próprio dispositivo móvel.

Pretende-se implantar o *framework* em loja de produtos e serviços como beta tester a fim de buscar entre os usuários uma melhor experiência no uso da tecnologia móvel com base em código 2D, a partir de registro que vem sendo realizado por meio do CDT/UnB.

REFERÊNCIAS BIBLIOGRÁFICAS

[Acohido 2010] Acohido, B. New 2D barcodes puts info at the tip of your camera phone. Site Web - http://www.usatoday.com/money/industries/technology/2009-05-19-2d-barcodes-camera-phones_n.htm. Acessado em 10/12/2010.

[Abajo et al., 2011] Abajo, B., Lopez-Coronado, M., Fernandez, F., Salcines, E., and de Castro Lozano, C. (2011). Mobile payment system based on 2D barcodes. In Information Systems and Technologies (CISTI), 2011 6th Iberian Conference on, pages 1–6. IEEE.

[Ally et al., 2010] Ally, M. and Toleman, M. (2004). Towards a theoretical framework of determinants for the adoption and diffusion of buyer authenticated credit card payment programs: the online merchant's perspective. In Proceedings of the IEEE International Conference on E-Commerce Technology, pages 349-352. IEEE Computer Society.

[Alonso et al., 2010] Alonso, G. and Casati, F. 2005. Web Services and service-oriented architectures. In Proceedings of the International Conference on Data Engineering. Volume 21, pages 1-47. IEEE Computer Society Press.

[Alpern et al., 2011] Alpern, B. and Schneider, F. (1987). Recognizing safety and liveness. In Distributed computing, volume 2, pages 117–126. Springer.

[Arendarenko 2009] Arendarenko, E. (2009). A study of comparing RFID and 2D barcode tag technologies for pervasive mobile applications. In Department of Computer Science and Statistics. University of Joensuu.

[ATP S.A. 2012] ATP S.A. Site Web – http://www.atp.com.br. Acessado em 10/02/2012.

[Baccaro 2010] Baccaro, Marco. SOAP e RPC. http://marcobaccaro.wordpress.com/2010/06/29/soap-e-rpc/. Acessado em 02/12/2012.

[Barbosa 2008] Barbosa, G. E. (2008). Mobile payment - estudo comparativo entre tecnologias de transações eletrônicas via dispositivos móveis. Monografia. Universidade Católica do Salvador - Departamento de Informática.

[Brambilla et al., 2010] Brambilla, M. and Origgi, A. (2008). MVC-Webflow: An AJAX Tool for Online Modeling of MVC-2 Web Applications. In Internation Conference on Web Engineering - ICWE 2008, pages 344–349.

[Business Insider 2012] Ray R. QR Code Security Best Practices. Site Web - http://www.businessinsider.com/qr-code-security-best-practices-2012-5. Acessado em 20/05/2012.

[Campos 2007] Campos, J.C. (2007). Uma abordagem formal à Engenharia da Usabilidade. In Universidade do Minho Campus de Gualtar, Portugal, pages 1-12. ACM Press.

[Chu et al., 2007] Chu, S. C. et al., Evolution of e-commerce Web sites: A conceptual framework and a longitudinal study. Jornal Information & Management, Elsevier. Volume. 44, pages 154–164.

[Chung et al., 2009] Chung, C. and Chen (2009). Image Hidden Technique Using QR-Barcode. In 2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pages 522–525. IEEE.

[Computing CMIS 2008] Computing, CMIS School (2008). Interactive everywhere – Quick Response Code, pages 1-8. Mathematical and Information Sciences. University of Brighton, UK.

[Curbera et al., 2010] Curbera, F. E. A. (2002). Unraveling the web services web: an introduction to SOAP, WS, and UDDI, pages 1089–7801. IEEE Internet computing.

[Dai et al., 2011] Dai, W., Cai, X., Wu, H., Zhao, W., and Li, X. (2011). An integrated mobile phone payment system based on 3G network. Journal of Networks, 6(9):1329–1336.

[Dantas 2011] Dantas, A. M-payment: Celular Vira Carteira Eletrônica na Hora de Pagar as Compras. Site Web - http://oglobo.globo.com/tecnologia/mat/2008/01/23/m-payment_celular_vira_carteira_eletronica_na_hora_de_pagar_as_compras-328159483.asp. Acessado em 20/01/2013.

[Denso 2012] Denso 2012, W. QR Code standardization. Site Web - http://www.denso-wave.com/qrcode/qrstandard-e.html. Acessado em 20/02/2012.

[Digital 2009] Olhar Digital. (2009). A magia chamada QR Code. Site Web - http://olhardigital.uol.com.br/produtos/central_de_videos/a-magia-chamada-QR-Code. Acessado em 20/01/2013.

[Dockter 2010] Dockter, P. (2010). Mobile tagging: Bridging the gap-mobile portland (2009). Site Web - http://www.slideshare.net/mobileportland/mobile-tagging-1066220. Acessado em 10/03/2011.

[Filho 2011] Filho, M. C. S. (2011). Arquitetura Orientada a Serviços para Comércio Eletrônico no Sistema Brasileiro de TV Digital. In Dissertação de Mestrado. Universidade de Brasília.

[Fourati et al., 2002] Fourati, A., Ayed, B., Kamoun, F., and Benzekri,A. (2002) A set based approach to secure the payment in mobile commerce. In Local Computer Networks, 2002. Proceedings. LCN 2002. 27th Annual IEEE Conference on, pages 136–137. IEEE.

[Gabriel 2009] Gabriel, M. (2009). QR Code: o que é isso? Site Web - http://colunistas.ig.com.br/marcelogodoy/2009/03/09/qr-code-o-que-e-isso. Acessado em 10/03/2011.

[Gao et al., 2005] Gao,J., Cai,J., Patel,K., and Shim,S. (2005) A wireless payment system. In Embedded Software and Systems, 2005. Second International Conference on, pages 1-8. IEEE.

[Gao et al., 2006a] Gao, J., Cai, J., Li, M., and Venkateshi, S. (2006). Wireless Payment – Opportunities, Challenges, and Solutions, Published by High Technology Letters, Vol. 12, 2006.

[Gao et al., 2006b] Gao, J. and Ebrary, I. (2006). Engineering Wireless-based Software Systems and Applications. Edição 1, Editora Artech House.

[Gao et al., 2007] Gao, J., Prakash, L., and Jagatesan, R. (2007). Understanding 2D barcode technology and applications in m-commerce-design and implementation of a 2D barcode processing solution. In Computer Software and Applications Conference, 2007. COMP- SAC 2007. 31st Annual International, volume 2, pages 49–56. IEEE.

[Gao et al., 2009] Gao, J., Kulkarni, V., Ranavat, H., Chang, L., and Mei, H. (2009). A 2D barcode-based mobile payment system. Third International Conference on Multimedia and Ubiquitous Engineering, pages 320–329. IEEE.

[Gebrekristos et al., 2008] Gebrekristos, M., Aljadaan, A., and Bihani, K. (2008). QR Codes for the chronically homeless. In CHI'08 extended abstracts on Human factors in computing systems, pages 3879–3884. ACM.

[Getz 2009] Getz, A. (2009). Microsoft tag take off. Site Web - http: //blogs.technet.com/b/microsoft_blog/archive/2010/05/27/beyond-beta-microsoft-tag-takes-off.aspx. Acessado em 05/06/2011.

[Hennessy 2012] Denis Hennessy (2012). Mobile Wallets 'Better Value' Than eWallets. Site Web: http://www.themultichannelretailer.com/item.php?news_id=1640. Acessado em 12/10/2012.

[Huang et al., 2002] Huang, Z. and Chen, K. (2002). Electronic payment in mobile environment. In Proceedings of the 13th International Workshop on Database and Expert Systems Applications, pages 413–417. IEEE Computer Society.

[IBGE 2010] IBGE (2010). IBGE - Instituto Brasileiro de Geografia e Estatística. Site Web - http://www.ibge.gov.br/home. Acessado em 10/03/2011.

[NPC Intelcom 2011] NPC Intelcom, iReader Scanner (2011). Site Web: http://www.intelcom.ru/2d/english. Acessado em 20/04/2011.

[ISO/IEC18004] ISO/IEC18004. Information technology - automatic identification and data capture techniques — bar code symbology — QR Code. In Automatic identification and data capture techniques. Internaction Standard.

[Jacobson 1991] Jacobson, I. (1991). Object oriented software engineering. ACM - Association for Computing Machinery, New York, NY, USA.

[Johnson et al., 2011] Johnson, R.; Foote, B. (2011). Designing reusable classes. Site Web: http://www.laputan.org/drc.html. Acessado em 10/12/2011.

[Johnston et al., 1998] Johnston, R. and Yap, A. (1998). Electronic data interchange using two dimensional bar code, volume 31, pages 83–91. Proceedings of the Hawaii International Conference on System Sciences.

[Julio et al., 2007] Julio, E., Brazil, W., and Neves, C. (2007). Esteganografia e suas aplicações. Universidade Federal do Rio de Janeiro.(Org.). Livro Texto dos Minicursos-SBSEG 2007, pages 54–102.

[Kato et al., 2005] H. Kato and K.T. Tan. 2D barcodes for mobile phones. In Mobile Technol- ogy, Applications and Systems, 2005 2nd International Conference, pages 113–116. IEEE Press, 2005.

[Kopecky 2008] Kopecky, S. (2008). Semantic web service offer discovery for e-commerce. In Proceedings of the 10th international conference on Electronic commerce, pages 1-6. ACM.

[Kurose 2010] Kurose, K. (2010). Ross, Redes de Computadores e a Internet, 5ª edição. Editora Pearson.

[Lausen et al., 2007] Lausen H., Haselwanter H., (2007). Finding Web Services. Volume 2007, pages 1-7. 1st European Semantic Technology Conference.

[Lin et al., 2000] Lin, Y., Chang, M., and Rao, H. (2000). Mobile prepaid phone services. Personal Communications, IEEE, pages 6–14.

[Mader 2011] Mader, Richard. A Comprehensive Guide for Navigating the Mobile Landscape. In Mobile Retailing Blueprint, pages 67-119, Washington, DC.

[Macoratti 2010] Macoratti. .NET Framework Site Web - http://www.macoratti.net. Acessado em 02/02/2012.

[Microsoft 2008] Microsoft. UDDI Services, http://technet.microsoft.com/en-us/library/ccc10047-f34e-4862-84cd-23dad191a06c. Acessado em 02/12/2012.

[Microsoft 2012] Microsoft. .NET Framework. Site Web - http://msdn.microsoft.com/en-us/netframework/aa496123. Acessado em 10/02/2012.

[Monteiro 2013] Monteiro, M., David and Rodrigues, J., P., C., Joel and Lloret, Jaime, and Sendra, Sandra, A Hybrid NFC-Bluetooth Secure Protocol for Credit Transfer Among Mobile Phones, in *Security and Communication Networks*, Wiley, ISSN: 1939-0114, Online ISSN: 1939-0122 (in press).

[Neto et al., 2010] Neto, B., E., Geraldo e Campos S., A., Andréa. Mobile Payment – Estudo comparativo entre tecnologias de transações eletrônicas via dispositivos móveis. Universidade Católica do Salvador, Departamento de Informática, monografia.

[NFC Forum 2012] NFC Forum (2012). Site Web - http://www.nfc-forum.org/specs/spec_list. Acessado em 05/05/2012.

[Oasis 2011] Oasis Standard (2011). Web Services Business Process Execution Language Version 2.0. Site Web - http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf. Acessado em 02/03/2011.
[Palmer et al., 1995] Palmer, R., Burke, H., Craig Harmon, K., and Adams, R. (1995). The bar code book: reading, printing and specification of bar code symbols. TDA Publications.

[Pinto 2010] Pinto, J. (2010). Using qr codes to grow your business (2010). Site Web - http://www.slideshare.net/jasonpinto62/using-qr-codes-to-grow-your-business.
Acessado em 10/03/2011.

[QuickMark 2010] QuickMark (2010). Quickmark mobile barcode. Site Web - http://www.quickmark.cn. Acessado em 10/03/2011.

[Yang Li et al., 2012] Yang Li, Yun Wang, (2012). Secure Electronic Transaction (SET Protocol). Site Web - http://people.dsv.su.se/~matei/courses/IK2001_SJE/li-wang_SET.pdf. Acessado em 20/12/2012.

[Ramsden et al., 2009] Ramsden, A. and Jordan, L. (2009). Are students ready for QR Codes? Findings from a student survey at the University of Bath. Site Web – http://opus.bath.ac.uk/12782/. Acessado em: 21/01/2013.

[Reilly et al., 2007] Reilly, D., Chen, H., and Smolyn, G. (2007). Toward fluid, mobile and ubiquitous interaction with paper using recursive 2D barcodes. Jornal Pervasive Mobile Interaction Devices 2007 (PerMID 2007), pages 1-4. Toronto, Canada.

[Rouillard et al., 2008] Rouillard, J. and Laboratoire, L. (2008). Contextual QR Codes. In Proceedings of the 2008 The Third International Multi-Conference on Computing in the Global Information Technology (ICCGI 2008), pages 50–55. IEEE Computer Society Washington, DC, USA.

[Tanenbaum et al., 2007] Tanenbaum, Andrew S., Steen, Maarten Van. Sistemas Distribuídos - princípaios e paradigmas. 2 edição, reimpressão 2007. Ed. Pearson

[Saxena et al., 2005] Saxena, A., Das, M., and Gupta, A. (2005). MMPS: a versatile mobile-to-mobile payment system. International Conference on ICMB Mobile Business, pages 400-405. Sydney, Australia. IEEE Computer Society.

[Schneider 2000] Schneider, F. (2000). Enforceable security policies. Information and System Security (TISSEC). Volume 3, pages 30–50. ACM New York, NY, USA.

[Selfa et al., 2006] Selfa, D., Carrillo, M., and Del Rocio Boone, M. (2006). A Database and Web Application based on MVC architecture. In Electronics, Communications and Computers, 2006, Mexico, pages 1-6. IEEE.

[Shannon 2012] Shannon, M. How QR Codes hide privacy, security risks. http://www.msnbc.msn.com/id/45729377/ns/technology_and_science-security/t/how-qr-codes-hide-privacy-security-risks/UCAiP_ZlSd1. Acessado em 08/08/2012.

[Sommerville 2011] Sommerville, I. (2011). Engenharia de Software - 9a edição 2011. Editora Pearson.

[Sun et al., 2007] Sun, A. and Sun, Y. and Liu, C. The QR-code reorganization in illegible snapshots taken by mobile phones. Computational Science and its Applications, 2007, pages 532-538. IEEE.

[Tecnologia 2010] Tecnologia, T. (2010). Transferências bancárias pelo celular salvam vidas na Somália. http://tecnologia.terra.com.br/interna/0,,OI4299821-EI4796,00.html. Acessado em 10/03/2011.

[Tehrani et al., 2010] Tehrani, M., Amidian, A., Muhammadi, J., and Rabiee, H. (2010). A survey of system platforms for mobile payment. In the 4th International Conference on Management of e-Commerce and e-Government, pages 376-385.

[Treiber 2007] Treiber M., D. S. (2007). Active Web Service registries. pages 66–71, volume 11. IEEE Internet Computing.

[Tukenmez 2010] Tukenmez, T. (2010). Mobile tagging. Site Web - http://www.slideshare.net/tugca/mobile-tagging-1. Acessado em 10/03/2011.

[Vilas 1999] Vilas, A. (1999). Providing Web Services over DVB-H: Mobile virtual Web Services. In IEEE Transactions on Consumer Electronics, volume 55, pages 644-652. IEEE Internet Computing.

[W3C 2004] W3C. SOAP specification. Site Web - http://www.w3.org/TR/soap. Acessado em 10/03/2011.

[Wagner et al., 1996] Wagner, D. and Schneier, B. Analysis of the SSL 3.0 protocol. In Proceedings of the 2nd USENIX Workshop on Electronic Commerce (EC-96), pages 29–40.

[Waraporn et al., 2009] Waraporn, N., Sithiyavanich, M., Jiarawattanasawat, H., and Pakchai, N. (2009). Virtual Credit Cards on Mobile for M-Commerce Payment. In Proceedings of the 2009 IEEE International Conference on e-Business Engineering, pages 241–246. IEEE Computer Society.

[Xerox 2013] Xerox. SMART send, Guia do usuário. Site Web - http://download.support.xerox.com/pub/docs/FFSMARTSEND/userdocs/any-os/pt_BR/SMARTsend3UserGuide.pdf. Acessado em 01/01/2013.

[Xin 2009] Xin, C. (2009). M-Commerce Development and Challenges Facing. In Proceedings of the 2009 IITA International Conference on Services Science, Management and Engineering, pages 229–232. IEEE Computer Society.

[Zhang et al., 2004] Zhang, Q., Moita, J., Mayes, K., and Markantonakis, K. (2004). The Secure and Multiple Payment System Based on the Mobile Phone Platform, In Smart Card Centre, Information Security Group (ISG), pages 1-15. Royal Holloway, University of London.

[Zheng et al., 2008] Zheng, F., Hu, H., and L, J. (2008). Applying a Component Behavior Model to MVC Pattern. In Proceedings of the 2008 The 9th International Conference for Young Computer Scientists, pages 1106–1111. IEEE Computer Society.

[3G Vision 2010] 3G Vision (2010). QR Code reader. Site Web - http://i-nigma.com/i-nigmahp.html. Acessado em 20/01/2013.

Apêndice A - Código para Rodar em Dispositivo Móvel

O código a seguir faz parte do conjunto de componentes de *software* voltados para Web que chama os métodos de geração do código 2D que serão vistos no apêndice B.

Segue a listagem 8 que gera a confirmação da operação na tela do usuário, ou seja, no dispositivo móvel. No término da compra de produto(s) ou de serviço(s), o *framework* de pagamento chama a classe referenciada. Este é o primeiro método chamado depois da confirmação do pagamento de forma dinâmica e segura. Dentro deste método Main existem outros submétodos chamados.

Note que o método de decriptografia é chamado passando um parâmetro.

Listagem 8- Código principal para a geração do QR Code de resposta

```
1.public void Main(String args[])
2.      {
3.         Bitmap bmp = null;
4.
5.                  string                  datafilepath                  =
System.Configuration.ConfigurationManager.AppSettings["datafilepath"];
6.                  OutwardFacing of = new OutwardFacing(datafilepath);
7.
8.                  try
9.                  {
10.                         Response.ContentType = "image/jpeg";
11.
12.                         switch (cript.Descriptografar(Session["tipo"].ToString()))
13.                         {
14.                                 case "URL":
15.                                         bmp                                     =
of.GetString(cript.Descriptografar(Session["txtUrl"].ToString()));
16.                                         break;
17.                                 case "Text":
18.                                         bmp                                     =
of.GetString(cript.Descriptografar(Session["t"].ToString()));
19.                                         break;
20.                                 case "Phone":
21.                                         bmp      =      of.GetString("TEL:"      +
cript.Descriptografar(Session["txtPhone"].ToString()));
22.                                         break;
23.                                 case "SMS":
24.                                         bmp    =    of.GetString("SMSTO:+55"    +
cript.Descriptografar(Session["txtSMS"].ToString()          +          "          "          +
Session["txtMessage"].ToString()));
25.                                         break;
26.                                 default:
```

```
27.                                         bmp = of.GetString("Olá, você está usando
o serviço aspneti.com");
28.                                         break;
29.                                 }
30.                         }
31.                         catch (Exception ex)
32.                         {
33.                                 bmp = of.GetString("");
34.                         }
35.                         finally
36.                         {
37.                                 bmp.Save(Response.OutputStream,
System.Drawing.Imaging.ImageFormat.Jpeg);
38.                                 bmp.Dispose();
39.                         }
    }
```

Na listagem 9, é mostrado o código feito para dispositivo móvel, lembrando que funciona também para outros dispositivos, como o *browser* de um computador normal como: Safari, Firefox, Internet Explorer e Google Chrome, sem qualquer restrição. O código referenciado na listagem 9 representa a primeira tela que por sua vez solicita ao usuário a identificação usando login e senha.

Listagem 9 - Tela de login do usuário

```
1.<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2.
3.<html xmlns="http://www.w3.org/1999/xhtml" >
4.<head runat="server">
5.   <title>SisPag - Login</title>
6.   <style type="text/css">
7.       .style1
8.       {
9.           width: 67px;
10.       }
11.  </style>
12.</head>
13.<body>
14.   <form id="form1" runat="server">
15.   <div>
16.       <img src="../images/banner_mauriciojunior.gif"
17.           style="width: 225px; height: 35px;" />
18.
19.       <div>Celular:</div>
20.       <div>
21.           <asp:TextBox ID="txtCelular" runat="server" Height="46px"
22.               style="margin-left: 0px" Width="93px" TabIndex="1"></asp:TextBox>
23.       </div>
24.
25.       <div>Senha:</div>
```

```
26.       <div>
27.          <asp:TextBox id="TextBox1" runat="server" Height="43px" Width="93px"
28.                TextMode="Password"></asp:TextBox>
29.       </div>
30.
31.       <div>
32.          <asp:ImageButton ID="ImageButton1" runat="server"
33.                ImageUrl="~/c/image/login-novo.jpg" onclick="cmdLogin_Click"
34.                Width="116px" Height="44px" />
35.       </div>
36.
37.       <div>
38.           <asp:Label ID="Label1" runat="server"></asp:Label>
39.       </div>
40.
41.    </div>
42.    </form>
43.</body>
44.</html>
```

O código referenciado na listagem 10, utiliza o *framework* de pagamentos para receber os dados recebidos do *QR Code* lido pelo móvel e, após a verificação, chama outra *interface* para aprovação e verificação.

Listagem 10 - Tela de verificação de usuário, usando *framework* de pagamento

```
1.using System;
2.using System.Collections;
3.using System.Configuration;
4.using System.Data;
5.using System.Linq;
6.using System.Web;
7.using System.Web.Security;
8.using System.Web.UI;
9.using System.Web.UI.WebControls;
10.using System.Web.UI.WebControls.WebParts;
11.using System.Web.UI.HtmlControls;
12.using System.Xml.Linq;
13.using System.Text;
14.using mauriciojunior.org.code;
15.using CriptQuery;
16.using Cript.Dados;
17.
18.namespace mauriciojunior.org.c
19.{
20.    public partial class _default : System.Web.UI.Page
21.    {
22.        protected void Page_Load(object sender, EventArgs e)
23.        {
24.            Criptografia cript = new Criptografia();
```

```
25.
26.         if (!Page.IsPostBack)
27.         {
28.             if (Request["v"] != null || Session["v"] != null)
29.             {
30.                 if (Request["v"] != null)
31.                     Session["v"] = cript.DecriptDados(Request["v"].ToString());
32.                 else
33.                     Session["v"] = Session["v"].ToString();
34.             }
35.             else
36.                 Response.Redirect("erro.aspx");
37.
38.             if (Session["celular"] == null)
39.                 Response.Redirect("login.aspx");
40.             else
41.                 Response.Redirect("aprovar.aspx");
42.
43.             //verificaCelular();
44.         }
45.     }
46.
47.     private void verificaCelular()
48.     {
49.         string numeroCelular="";
50.         string browser="";
51.
52.         if (Request.Headers["msisdn"] != null)
53.         {
54.             //log.Info("O celular numero: " + Request.Headers["msisdn"] + " esta
tentando logar com o login: " + Request.Form["txtLogin"]);
55.             numeroCelular = Request.Headers["msisdn"];
56.         }
57.         else if (Request.Headers["HTTP_MSISDN"] != null)
58.         {
59.             //log.Info("O celular numero: " + Request.Headers["HTTP_MSISDN"] + "
esta tentando logar com o login: " + Request.Form["txtLogin"]);
60.             numeroCelular = Request.Headers["HTTP_MSISDN"];
61.         }
62.         else
63.         {
64.             //log.Info("O browser: " + Request.Headers["User-Agent"] + " esta
tentando logar com o login: " + Request.Form["txtLogin"]);
65.             browser = Request.Headers["User-Agent"];
66.         }
67.         Response.Write(numeroCelular);
68.         Response.Write(browser);
69.     }
70. }
71.}
```

O código seguinte corresponde à aprovação do valor do produto ou serviço adquirido. Esta primeira parte foi feita com HTML (HyperText Markup Language) e tags em HTML e C# Linguagem de Programação Orientada a Objetos - CSharp.

Listagem 11 - Tela de aprovação html code

```
1.<%@ Page Language="C#" MasterPageFile="~/c/Celular.Master"
AutoEventWireup="true" CodeBehind="aprovar.aspx.cs"
Inherits="mauriciojunior.org.c.aprovar" Title="Untitled Page" %>
2.
3.<asp:Content ID="Content1" ContentPlaceHolderIDa="ContentPlaceHolder1"
runat="server">
4.
5.    <font face="verdana">Valor: <asp:Label ID="lblValor" Font-Bold="true"
runat="server"></asp:Label></font>
6.<br /><br />
7.<asp:ImageButton ID="cmdAprovar" runat="server"
8.        ImageUrl="~/c/image/aprovar-novo.JPG" onclick="cmdAprovar_Click"
9.    Height="44px" Width="124px" />
10.<br /><br />
11.<asp:ImageButton ID="cmdCancelar" runat="server"
12.        ImageUrl="~/c/image/cancelar-novo.JPG" onclick="cmdCancelar_Click"
13.    Height="42px" Width="128px" />
```

A tela a seguir mostra a codificação feita para aprovação do usuário, ou seja, depois de clicar no botão aprovar, é executado o código. Lembrando que todo o código mostrado funciona tanto para dispositivos móveis quanto em computadores. No final, é enviado um SMS de confirmação e gerado novo QR Code dinâmico para comprovação do estabelecimento.

Listagem 12 - Tela de aprovação C# Code

```
1.using System;
2.using System.Collections;
3.using System.Configuration;
4.using System.Data;
5.using System.Linq;
6.using System.Web;
7.using System.Web.Security;
8.using System.Web.UI;
9.using System.Web.UI.WebControls;
10.using System.Web.UI.WebControls.WebParts;
11.using System.Web.UI.HtmlControls;
12.using System.Xml.Linq;
13.using mauriciojunior.org.code;
14.using System.Text;
15.
16.namespace mauriciojunior.org.c
17.{
18.    public partial class aprovar : System.Web.UI.Page
```

```
19.   {
20.       protected void Page_Load(object sender, EventArgs e)
21.       {
22.           if (!Page.IsPostBack)
23.           {
24.               int total = Session["v"].ToString().Length;
25.               string decimento = Session["v"].ToString().PadRight(2);
26.               string a = Convert.ToString(Session["v"].ToString().Substring(0, total -
2));
27.               string b = Convert.ToString(Session["v"].ToString().Substring(a.Length));
28.
29.               Session["v"] = a + "," + b;
30.               lblValor.Text = Session["v"].ToString();
31.           }
32.       }
33.
34.       protected void cmdCancelar_Click(object sender, ImageClickEventArgs e)
35.       {
36.           Session.Abandon();
37.           Response.Redirect("login.aspx");
38.       }
39.
40.       protected void cmdAprovar_Click(object sender, ImageClickEventArgs e)
41.       {
42.           SMS sm = new SMS();
43.           sm.enviaSMS(Session["celular"].ToString(), "Obrigado. Transacao realizada
com sucesso. Valor: R\$ " + Session["v"].ToString());
44.
45.           Response.Redirect("agradecimento.aspx");
46.       }
47.   }
48.}
```

Finalizando todo o processo de compra, uma nova tela de agradecimento e confirmação da operação é mostrada gerando código 2D, conforme listagem A6.

Listagem 13 - Tela de confirmação final

```
1.<%@ Page Language="C#" MasterPageFile="~/c/Celular.Master"
AutoEventWireup="true" CodeBehind="agradecimento.aspx.cs"
Inherits="mauriciojunior.org.c.agradecimento" Title="Untitled Page" %>
2.
3.<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1"
runat="server">
4.
5.    <font face="verdana" size="2">Obrigado, você receberá uma mensagem SMS
confirmando a operação.</font>
6.    <br />
7.    <asp:Image ID="imagemMostra" runat="server"
ImageUrl="http://qrcode.aspneti.com/imagemFlush.aspx?tipo=Text&t=funcionou"></a
sp:Image>
8.</asp:Content>
```

Apêndice B - Gerando código de barras 2D

A seguir, o primeiro método da listagem 14 é responsável por pegar a string de dados vinda de qualquer plataforma, por exemplo: *Web*, celular, computador, e-Reader e outros. O primeiro passo do método é transformar os dados em Byte e depois contar os caracteres armazenando em uma variável conforme a norma mesmo informa [ISO/IEC 18004]. Outro método é chamado no final para dar sequência a codificação.

Listagem 14 - Exemplo do primeiro método que gera o código 2D.

```
1.      public virtual bool[][] calQrcode(string msg)
2.      {
3.         ASCIIEncoding ascii = new ASCIIEncoding();
4.
5.         byte[] encoded = ascii.GetBytes(msg);
6.         sbyte[] sencoded = new sbyte[encoded.Length];
7.         for (int i = 0; i < encoded.Length; i++)
8.            sencoded[i] = (sbyte)encoded[i];
9.         return calQrcode(sencoded);
10.     }
```

A listagem 15 chamado pelo primeiro método calQrcode mostra que, é feito a verificação dos dados recebidos para o processamento de acordo com cada regra informada na norma para o modo "numérico", "alfanumérico" e "byte".

Listagem 15 - Exemplo do segundo método que calcula toda parte da imagem gerada.

```
1.      public virtual bool[][] calQrcode(sbyte[] qrcodeData)
2.      {
3.        ///process mode encode
4.
5.
6.         switch (qrcodeEncodeMode)
7.         {
8.           /* ---- alphanumeric mode ---  */
9.             case 'A':
10.
11.                codewordNumPlus = new int[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };
12.
13.                dataValue[dataCounter] = 2;
14.                dataCounter++;
15.                dataValue[dataCounter] = dataLength;
16.                dataBits[dataCounter] = 9;
17.                codewordNumCounterValue = dataCounter;
18.
19.                dataCounter++;
20.                for (int i = 0; i < dataLength; i++)
21.                {
22.                        ///começa a fazer o processamento
23.                }
24.                dataCounter++;
```

```
25.                 break;
26.
27.             /* ---- numeric mode ---- */
28.
29.             case 'N':
30.
31.                 codewordNumPlus = new int[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4 };
32.
33.                 dataValue[dataCounter] = 1;
34.                 dataCounter++;
35.                 dataValue[dataCounter] = dataLength;
36.
37.                 dataBits[dataCounter] = 10; /* #version 1-9*/
38.                 codewordNumCounterValue = dataCounter;
39.
40.                 dataCounter++;
41.                 for (int i = 0; i < dataLength; i++)
42.                 {
43.                         ///começa a fazer o processamento
44.                 }
45.                 dataCounter++;
46.                 break;
47.
48.             /* ---- 8bit byte ---- */
49.
50.             default:
51.
52.                 codewordNumPlus = new int[] { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8 };
53.                 dataValue[dataCounter] = 4;
54.                 dataCounter++;
55.                 dataValue[dataCounter] = dataLength;
56.                 dataBits[dataCounter] = 8; /* #version 1-9 */
57.                 codewordNumCounterValue = dataCounter;
58.
59.                 dataCounter++;
60.
61.                 for (int i = 0; i < dataLength; i++)
62.                 {
63.                    dataValue[i + dataCounter] = (qrcodeData[i] & 0xFF);
64.                    dataBits[i + dataCounter] = 8;
65.                 }
66.                 dataCounter += dataLength;
67.
68.                 break;
69.         }
70.     }
```

A próxima parte do código, listagem 16, mostra que a contagem dos dados é feita para verificar a correção de erro no código 2D conforme a norma recomenda. A correção de erro gera uma série de palavras de códigos de correção de dados perdidos. Eles são mostrados na figura 64 de níveis de correção [ISO/IEC 18004].

Error Correction Level	Recovery Capacity % (approx.)
L	7
M	15
Q	25
H	30

Figura 64 - Níveis de correção de erro - Fonte: [ISO/IEC 18004]

Listagem 16- Contando e verificando os códigos de correção de erro

```
1.      int totalDataBits = 0;
2.      for (int i = 0; i < dataCounter; i++)
3.      {
4.         totalDataBits += dataBits[i];
5.      }
6.      int ec;
7.      switch (qrcodeErrorCorrect)
8.      {
9.         case 'L':
10.            ec = 1;
11.            break;
12.
13.         case 'Q':
14.            ec = 3;
15.            break;
16.
17.         case 'H':
18.            ec = 2;
19.            break;
20.
21.         default:
22.            ec = 0;
23.            break;
24.
25.      }
```

Existem dois tipos de símbolos de caracter, os regulares e os irregulares, dentro do código 2D. Eles dependem da posição do símbolo relativo, outros símbolos de caracteres e funções padrões. Essa posição pode ser simplificada com 7 bits de acordo com a figura 65 e pode ser de cima para baixo, de baixo para cima, direita para esquerda e esquerda para direita [ISO/IEC 18004].

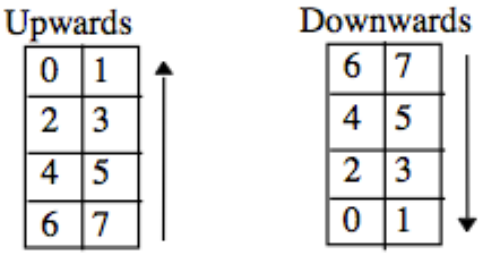

Figura 65 - Símbolo de caracter regular - Fonte: [ISO/IEC 18004]

Para saber a posição de cada caracter, precisamos saber os blocos e a posição representada pelas coordenadas X e Y da matriz. O código a seguir mostra parte do código referente a matriz, blocos e padrões de máscara falado na norma [ISO/IEC 18004]. Mas antes do código fonte, veja a figura 66 responsável por mostrar as posições dos blocos.

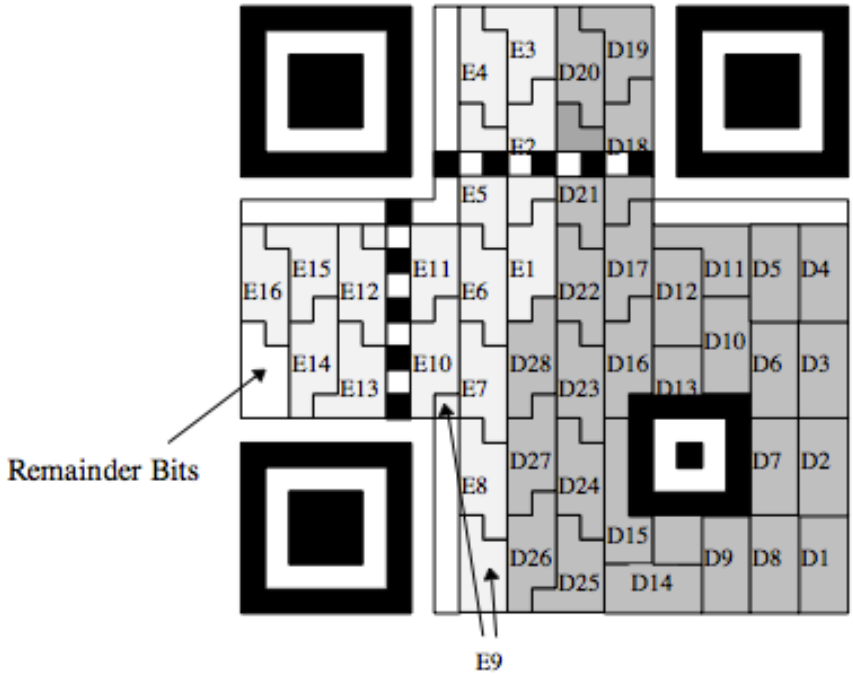

Figura 66 - Posição de blocos e símbolos - Fonte: [ISO/IEC 18004]

Listagem 17- Contando e verificando os códigos de erro de correção

```
1.  int byte_num = matrixRemainBit[qrcodeVersion] + (maxCodewords << 3);
2.
3.  sbyte[] matrixX = new sbyte[byte_num];
4.  sbyte[] matrixY = new sbyte[byte_num];
5.  sbyte[] maskArray = new sbyte[byte_num];
6.  sbyte[] formatInformationX2 = new sbyte[15];
7.  sbyte[] formatInformationY2 = new sbyte[15];
8.  sbyte[] rsEccCodewords = new sbyte[1];
9.  sbyte[] rsBlockOrderTemp = new sbyte[128];
10.
11.   try
```

```
12.         {
13.             System.IO.Stream fis = new System.IO.FileStream(filename,
FileMode.Open, FileAccess.Read);
14.             System.IO.BufferedStream bis = new System.IO.BufferedStream(fis);
15.             SupportClass.ReadInput(bis, matrixX, 0, matrixX.Length);
16.             SupportClass.ReadInput(bis, matrixY, 0, matrixY.Length);
17.             SupportClass.ReadInput(bis, maskArray, 0, maskArray.Length);
18.             SupportClass.ReadInput(bis, formatInformationX2, 0,
formatInformationX2.Length);
19.             SupportClass.ReadInput(bis, formatInformationY2, 0,
formatInformationY2.Length);
20.             SupportClass.ReadInput(bis, rsEccCodewords, 0,
rsEccCodewords.Length);
21.             SupportClass.ReadInput(bis, rsBlockOrderTemp, 0,
rsBlockOrderTemp.Length);
22.             bis.Close();
23.             fis.Close();
24.         }
25.         catch (System.Exception e)
26.         {
27.             SupportClass.WriteStackTrace(e, Console.Error);
28.         }
29.
30.         sbyte rsBlockOrderLength = 1;
31.         for (sbyte i = 1; i < 128; i++)
32.         {
33.             if (rsBlockOrderTemp[i] == 0)
34.             {
35.                 rsBlockOrderLength = i;
36.                 break;
37.             }
38.         }
39.         sbyte[] rsBlockOrder = new sbyte[rsBlockOrderLength];
40.         Array.Copy(rsBlockOrderTemp, 0, rsBlockOrder, 0,
(byte)rsBlockOrderLength);
41.
42.
43.         sbyte[] formatInformationX1 = new sbyte[] { 0, 1, 2, 3, 4, 5, 7, 8, 8, 8, 8, 8,
8, 8, 8 };
44.         sbyte[] formatInformationY1 = new sbyte[] { 8, 8, 8, 8, 8, 8, 8, 8, 7, 5, 4, 3,
2, 1, 0 };
45.
46.         int maxDataCodewords = maxDataBits >> 3;
47.
48.
49.         /* --- attach data */
50.         for (int i = 0; i < maxCodewords; i++)
51.         {
52.
53.             sbyte codeword_i = codewords[i];
```

```
54.             for (int j = 7; j >= 0; j--)
55.             {
56.
57.                 int codewordBitsNumber = (i * 8) + j;
58.
59.                 matrixContent[matrixX[codewordBitsNumber] & 0xFF][matrixY[codewordBitsNumber] & 0xFF] = (sbyte)((255 * (codeword_i & 1)) ^ maskArray[codewordBitsNumber]);
60.
61.                 codeword_i = (sbyte)(SupportClass.URShift((codeword_i & 0xFF), 1));
62.             }
63.         }
64.
65.         for (int matrixRemain = matrixRemainBit[qrcodeVersion]; matrixRemain > 0; matrixRemain--)
66.         {
67.             int remainBitTemp = matrixRemain + (maxCodewords * 8) - 1;
68.             matrixContent[matrixX[remainBitTemp] & 0xFF][matrixY[remainBitTemp] & 0xFF] = (sbyte)(255 ^ maskArray[remainBitTemp]);
69.         }
70.
71.         /* --- mask select --- */
72.         sbyte maskNumber = selectMask(matrixContent, matrixRemainBit[qrcodeVersion] + maxCodewords * 8);
73.         sbyte maskContent = (sbyte)(1 << maskNumber);
74.
75.         /* --- format information --- */
76.
77.         sbyte formatInformationValue = (sbyte)(ec << 3 | maskNumber);
78.
79.         for (int i = 0; i < 15; i++)
80.         {
81.
82.             sbyte content = (sbyte)System.SByte.Parse(formatInformationArray[formatInformationValue].Substring(i, (i + 1) - (i)));
83.
84.             matrixContent[formatInformationX1[i] & 0xFF][formatInformationY1[i] & 0xFF] = (sbyte)(content * 255);
85.             matrixContent[formatInformationX2[i] & 0xFF][formatInformationY2[i] & 0xFF] = (sbyte)(content * 255);
86.         }
87.
88.         bool[][] out_Renamed = new bool[modules1Side][];
89.         for (int i3 = 0; i3 < modules1Side; i3++)
90.         {
91.             out_Renamed[i3] = new bool[modules1Side];
92.         }
93.
```

```
94.          int c = 0;
95.          for (int i = 0; i < modules1Side; i++)
96.          {
97.             for (int j = 0; j < modules1Side; j++)
98.             {
99.
100.                if ((matrixContent[j][i] & maskContent) != 0 || frameData[c] ==
(char)49)
101.                {
102.                   out_Renamed[j][i] = true;
103.                }
104.                else
105.                {
106.                   out_Renamed[j][i] = false;
107.                }
108.                c++;
109.             }
110.             c++;
111.          }
112.
113.          return out_Renamed;
114.       }
```

O código fonte a seguir mostra como é feito o cálculo para verificar a parte preta ou branca da imagem de código 2D de acordo com a norma [ISO/IEC 18004]. A norma fala sobre a versão, posições de erro, máscaras padrões e formado da informação figura 67.

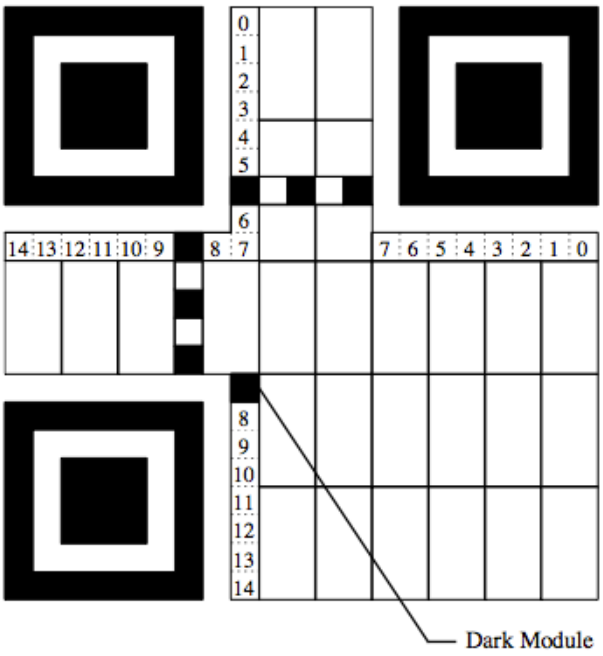

Figura 67 - Blocos pretos e brancos - Fonte: [ISO/IEC 18004]

Listagem 18 - Cálculo para montar a parte preta e branca do código 2D

```
1.      private Bitmap boolstoimage(bool[][] matrix, int pixelmultiplier)
2.      {
3.          int borderoffset = (int) (matrix.Length * pixelmultiplier)/4;
4.          Bitmap bmp = new Bitmap((matrix.Length * pixelmultiplier) + (borderoffset *
2), (matrix.Length * pixelmultiplier) + (borderoffset * 2));
5.          Brush black = new SolidBrush(Color.Black);
6.          Brush white = new SolidBrush(Color.White);
7.          Pen grey = new Pen(Color.Gray);
8.
9.          Graphics g = Graphics.FromImage(bmp);
10.           Rectangle bord = new Rectangle(0,0,bmp.Width-1,bmp.Height-1);
11.           g.FillRectangle(white, bord);
12.           g.DrawRectangle(grey, bord);
13.
14.           for (int i = 0; i < matrix.Length; i++)
15.           {
16.              for (int j = 0; j < matrix.Length; j++)
17.              {
18.                 Rectangle rect = new Rectangle(borderoffset + (j * pixelmultiplier),
borderoffset+(i * pixelmultiplier), pixelmultiplier,
19.                                                 pixelmultiplier);
20.                 g.FillRectangle(matrix[j][i] ? black : white, rect);
21.              }
22.           }
23.           return bmp;
24.       }
```