SMART INTERFACE: FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DE APLICAÇÕES JAVA CARD

Gleison Tavares DIOLINO (1); Leonardo Ataíde MINORA (2); Fellipe Araújo ALEIXO (3);
(1) CEFET-RN, Av. Sen. Salgado Filho, 1559, Tirol, Natal-RN – CEP: 59015-000, telefone: (84) 4005 2637,
e-mail: gleison.tavares@yahoo.com
(2) CEFET-RN, e-mail: minora@cefetrn.br
(3) CEFET-RN, e-mail:fellipe@cefetrn.br

RESUMO

Com a evolução da tecnologia de microprocessadores, hoje é possível colocar um minúsculo *chip* em um cartão de plástico, nas dimensões de um cartão de crédito convencional. *Chip* este com capacidade de armazenamento de informação e processamento. Esse tipo de cartão é chamado de *Smart Card*. É possível desenvolver aplicações para esses cartões utilizando um subconjunto da plataforma Java – *Java Card*. O subconjunto da plataforma Java para o desenvolvimento de aplicações próprias para executar em *Smart Cards* é bastante limitado, em virtude das limitações do hardware envolvido. O desenvolvimento deste tipo de aplicação implica invariavelmente em se trabalhar no nível de *bytes*, pois toda a comunicação com o cartão acontece através de APDUs (*Applicaion Protocol Data Unit*) montadas como uma sequência de *bytes*. Para auxiliar nesse trabalho, foi desenvolvido o SMART SHELL: ferramenta, modo texto, que interpreta comandos para a interação com a aplicação gerente do cartão – *card manager*. Os objetivos deste trabalho são: aperfeiçoar as funcionalidades do SMART SHELL e desenvolver uma interface gráfica para o mesmo, denominada de SMART INTERFACE. A aplicação em questão visa oferecer ao desenvolvedor de aplicações *Java Card* um mecanismo simples e ágil para a realização de atividades comuns de manipulação de um cartão, tais como: (1) realizar o processo de autenticação para operar com o *card manager*, (2) listar as aplicações – *applets Java Card* – instalados em um cartão, (3) preparar uma aplicação para ser instalada em um cartão, (4) instalação de uma aplicação no cartão, (5) remoção de uma aplicação instalada no cartão, entre outras. Para a realização do trabalho adotou-se a seguinte metodologia: levantamento bibliográfico sobre o tema; definição dos aperfeiçoamentos necessários ao SMART SHELL; implementação dos aperfeiçoamentos propostos; e implementação da interface gráfica para o mesmo denominada de SMART INTERFACE.

Palavras-chave: *Smart Card*, *Java Card*, Cartão inteligente.

1. INTRODUÇÃO

Com a consolidação dos *smart cards,* cartões inteligentes, como dispositivos seguros e confiáveis, o número de aplicações dessa tecnologia vem aumentando a cada dia, assim como o número de empresas que utilizam esses cartões em diversas áreas como: telefonia móvel, indústria bancária, transporte coletivo, segurança, dentre outras (CHEN, 2004).

A maioria dessas aplicações para cartões inteligentes é desenvolvidas utilizando um subconjunto da plataforma Java, Java Card. Esse subconjunto para o desenvolvimento de aplicações próprias para executar em *Smart Cards* é bastante limitado, em virtude das limitações do hardware envolvido. Dessa forma, o desenvolvimento desse tipo de aplicação implica invariavelmente em se trabalhar no nível de *bytes*, pois toda a comunicação com o cartão acontece através de pacotes APDUs (*Applicaion Protocol Data Unit*) montados como uma seqüência de *bytes.* Isso torna operações fundamentais como autenticação, instalação, seleção e remoção de aplicações do cartão bastante complexas.

Tendo isso em vista, notou-se a necessidade do desenvolvimento de uma ferramenta livre que auxiliasse o desenvolvedor de aplicações Java Card nessas tarefas. Assim, foi desenvolvido o SMART SHELL*,* ferramenta em modo texto que interpreta comandos para a interação com a aplicação gerente do cartão, *card manager*. Utilizando o SMART SHELL*,* o desenvolvedor pode abstrair as cadeias de bytes e executar comandos em alto nível como AUTH, para se autenticar no cartão*,* INSTALL e DELETE, para instalar e remover aplicações do cartão, dentre outros.

Contudo, o SMART SHELL necessita de alguns aperfeiçoamentos, sendo a implementação de uma interface gráfica o principal. Este é o objetivo deste trabalho, aperfeiçoar as funcionalidades do SMART SHELL e desenvolver uma interface gráfica para ele, denominada de SMART INTERFACE. A aplicação em questão visa oferecer ao desenvolvedor de aplicações *Java Card* um mecanismo simples e ágil para a realização de atividades comuns de manipulação tais como: (1) realizar o processo de autenticação para operar com o *card manager*, (2) listar as aplicações – *applets Java Card* – instalados em um cartão, (3) selecionar essas aplicações, (4) executar seus comandos, (5) instalar uma aplicação no cartão, (6) remover uma aplicação instalada no cartão, entre outras.

A metodologia seguida para a realização do trabalho foi baseada nos seguintes passos: levantamento bibliográfico sobre o tema; definição dos aperfeiçoamentos necessários ao SMART SHELL; implementação dos aperfeiçoamentos propostos e desenvolvimento da sua interface gráfica, denominada de SMART INTERFACE. O trabalho tem como base trabalhos e pesquisas anteriores.

2. SMART CARDS

Os *smart cards,* cartões inteligentes, são cartões de plástico semelhantes aos cartões de crédito convencionais, que armazenam e processam informações, possuindo poder de processamento embutido. Eles contêm um microprocessador ou um *chip* de memória, sendo que para Chen (CHEN, 2004), os cartões com apenas chip de memória podem não ser considerados como cartões inteligentes, pois não realizam nenhum tipo de processamento, realizando apenas funções pré-definidas.

As principais aplicações dos *smart cards* são cartões de telefone pré-pagos, sistemas GSM (*Global Sisitem for Mobile Communication*), cartões de crédito ou débito seguros, tarifa de transporte coletivo, acesso a ambientes e assinaturas digitais. O *smart card* é a ferramenta ideal para tais aplicações, pois proporciona diversos benefícios como: poder computacional embutido, praticidade, flexibilidade de uso e segurança. Um cartão inteligente é extremamente seguro, pois a obtenção de informações necessita da posse física do cartão e de um conhecimento profundo do seu *hardware* e s*oftware* (VIANA, 2007).

Os *smart cards* são padronizados para a indústria pela ISO (*Internation Organization Standardization*) 7816 e suas sete partes, que definem características físicas, dimensões e localização de contatos, sinais eletrônicos e protocolos de transmissão, entre outros. Chen (CHEN, 2004) também cita as seguintes especificações para *smart cards:* GSM (*Global Sisitem for Mobile Communication*), EMV (Europay , MasterCard e Visa), *Open Platform, Open Card Framework e PC/SC,* além do Java *Card.*

3. A TECNOLOGIA JAVA CARD

A tecnologia *Java Card* é uma adaptação da plataforma Java para ser utilizada em *smart cards* e outros dispositivos cujos ambientes são altamente especializados, e cuja memória e restrições de processamento

são mais severas do que as dos dispositivos J2ME[1] (CHEN, 2004). A tecnologia Java *Card* possui sua própria máquina virtual, API (*Application Programming Interface*) e especificação de *Run Time*, e está atualmente na versão 2.2.2 (MEIRELES; VIANA; VIDAL, 2006).

3.1. Applets Java *Card* e Aplicação *Host*

Os *applets* Java *Card* são classes Java convencionais que estendem a classe javacard.framework.Applet, seguindo as especificações da tecnologia Java *Card* (CHEN, 2004).
As classes e os *applets* são empacotados em um único arquivo denominado CAP (*Converted Applet*), conceito semelhante ao arquivo de extensão JAR. É esse arquivo que é instalado no cartão.

As aplicações *host,* aplicações externas ao cartão que se comunicam com ele, podem ser desenvolvidas em várias linguagens de programação. Para uma aplicação desenvolvida em Java existe o *OpenCard Framework* (OCF), que é um conjunto de bibliotecas que provêem comunicação entre diferentes softwares e hardwares; escrito em J2SE[2], esse *framework* funciona em qualquer sistema que contenha uma máquina virtual Java (JVM) compatível.

4. COMUNICAÇÃO EM UM CARTÃO INTELIGENTE

4.1. Dispositivo Receptor e Aplicação *Host*

Um cartão inteligente é inserido em um dispositivo receptor de cartão (CAD – *card acceptance device*), que pode se conectar a outro computador. Dispositivos receptores de cartões podem ser classificados em dois tipos: leitores e terminais. Um leitor fica conectado em uma porta serial, paralela ou USB de um computador, através da qual o cartão se comunica, já os terminais são computadores que têm um leitor de cartão integrado como um de seus componentes (VIANA, 2007).

Como já foi dito, as aplicações que se comunicam com o cartão, não importando se residem no computador conectado a um leitor ou em um terminal, são chamadas de aplicações *host* (VIANA, 2007).

4.2. Modelo de Comunicação de um Cartão Inteligente

De maneira similar à troca de mensagens que ocorre entre dois computadores conectados em rede, através do protocolo TCP/IP, os cartões inteligentes conversam com computadores usando pacotes de dados específicos, chamados APDUs (*application protocol data unit* – unidade de dados do protocolo de aplicação). Um APDU contém uma mensagem de comando ou de resposta (CHEN, 2004).

4.3. Segurança na Comunicação

Fundamentalmente, um *smart card* deve possuir integridade de informação, ou seja, a informação recebida deve ser completa e autêntica, e segurança de transação, ou seja, a informação trafegada não pode ser interceptada por usuários não-autorizados (OpenCard Consortium, 2006).

Um *smart card* que esteja em concordância com a *OpenPlatform* deve possuir um componente especial chamado de *Card Manager*. O *Card Manager* é uma aplicação, *applet,* que vem previamente instalado no cartão e atua como seu administrador central, realizando múltiplas responsabilidades. É ele que realiza toda a interação com o desenvolvedor de aplicações, recebendo as instruções, comandos APDU, para efetuar diversas operações como: autenticação, listagem dos *applets* instalados em um cartão, seleção dessas aplicações, instalação de um *applet* no cartão, remoção de um *applet* instalado no cartão, dentre outras (OpenCard Consortium, 2006).

Internamente, o *Card Manager* contém um conjunto de chaves, sendo cada um desses composto por três chaves: KAUTH/ENC, KMAC, KKEK, suportando de 0 a 127 conjuntos de chaves (MEIRELES; VIANA; VIDAL, 2006).

A autenticação mútua é um dos procedimentos para se garantir a segurança no cartão. Ela inicia um canal seguro e garante ao cartão e à aplicação que ambos estão "conversando" com uma entidade autenticada.

---

1 J2ME (Java 2 Platform Micro Edition) é uma edição da tecnologia Java para dispositivos móveis.

2 J2SE, Java 2 Plataform Standard Edition, é a edição padrão da tecnologia Java.

Ambas as entidades têm a uma mesma informação secreta (chaves) e se houver alguma falha, todo o processo é reiniciado (CHEN, 2004).

4.4. Comandos da APDU

O protocolo APDU, especificado pela ISO 7816-4, é um protocolo de nível de aplicação entre o cartão e a aplicação *host*. Mensagens APDU, de acordo com o ISO 7816-4, compreendem duas estruturas: uma usada pela aplicação *host* para enviar comandos para o cartão, e outra usada pelo cartão para enviar as respostas de volta a aplicação *host*. A primeira estrutura é conhecia como APDU de comando (C-APDU) e a última como APDU de resposta (R-APDU). Para todo APDU de comando existe sempre um APDU de resposta correspondente (VIANA, 2007).

4.5. APDUs de Comando e a Instalação de um Applet

A estrutura de um APDU de comando é ilustrada na tabela 1.

Tabela 1: Estrutura de um APDU de comando

Cabeçalho Obrigatório				Corpo Opcional		
CLA	INS	P1	P2	LC	Dados	LE

O cabeçalho de um APDU de comando consiste em 4 *bytes*: CLA (Classe de Instrução), INS (código da instrução), P1 e P2 (parâmetros 1 e 2). O *byte* da classe identifica a categoria dos APDUs de comando e de resposta. O *byte* de instrução especifica a instrução do comando. Os dois *bytes* de parâmetros P1 e P2 são usados para informar mais alguma qualificação à instrução (VIANA, 2007).

A seção após o cabeçalho no APDU de comando é um corpo opcional que varia em tamanho. O campo LC no corpo especifica o tamanho do campo de dados (em *bytes*). O campo de dados contém os dados que são enviados para o cartão executar a instrução especificada no cabeçalho do APDU. O último *byte* no corpo do APDU de comando é o campo LE, que especifica o número de *bytes* esperados pela aplicação *host* na resposta do cartão (CHEN, 2004).

Abstraindo a complexidade dessa estrutura, é preciso compreender os comandos necessários à instalação de um *applet* no *smart card*. Entre todos esses comandos, é necessário inicialmente o entendimento de dois: INTIALIZE UPADATE, que abre um canal seguro para a comunicação entre o *applet* selecionado e o terminal do cliente, e EXTERNAL AUTHENTICATE, que autentica o terminal externo e define o modo de segurança do canal seguro, que varia de *applet* para *applet.* Antes de executar o EXTERNAL AUTHENTICATE, um comando INITIALIZE UPDATE deve ter sido executado com sucesso.

Outros comandos relacionados à instalação de um *applet* são os comandos INSTALL e LOAD. O primeiro é responsável por iniciar e finalizar o processo de instalação da aplicação, enquanto o segundo envia *byte*-a-*byte* o arquivo CAP que contém o *applet* a ser instalado. Geralmente, entre o INSTALL inicial e o final, são executados mais de um comando LOAD.

Além dos citados, também existem os seguintes comandos: DELETE, GET DATA, GET STATUS, PIN CHANGE UNBLOCK, PUT DATA, PUT KEY, SELECT, SET STATUS (MEIRELES; VIANA; VIDAL, 2006).

5. SMART SHELL

O SMART SHELL é uma ferramenta idealizada para auxiliar o desenvolvedor de aplicações Java *Card* nas diversas operações de manipulação do cartão inteligente. Para desenvolvê-lo foram utilizadas as bibliotecas do OCF (*OpenCard Framework).*

Como pode ser percebido, montar um comando APU é uma tarefa bastante complexa, porém com a utilização do SMART SHELL, ela torna-se bastante simples, pois ele interpreta comandos em alto nível, abstraindo as cadeias de *bytes* para o desenvolvedor de aplicações Java *Card.* Os comandos interpretados pelo SMART SHELL são: AUTH, DELETE, EXEC, EXECFILE, INSTALL, LIST e SELECT, além de HELP e EXIT.

O comando AUTH realiza a autenticação no cartão, recebendo a senha de autenticação como parâmetro. Para realizar a autenticação através do comando AUTH, o SMART SHELL envia os APDUs correspondentes aos comandos INTIALIZE UPADATE e EXTERNAL AUTHENTICATE para o *card manager*. O comando AUTH deve ser executado da seguinte maneira: AUTH "senha"

O comando DELETE exclui um *applet* ou arquivo carregado no cartão, recebendo como parâmetro o ID, identificador em hexadecimal, do componente a ser excluído. Para excluir um componente do cartão, O SMART SHELL envia o APDU correspondente ao comando DELETE ao *card manager*. O comando DELETE é executado da seguinte maneira: DELETE "id do componente(ex: A0:00:00:00:62:03:01)".

O comando EXEC executa um comando APDU, que deve ser processado pelo *applet* que estiver selecionado no instante do comando. Ele recebe como parâmetro o comando a ser executado em hexadecimal (ex: A0:00:00:00:62:03:01) e deve ser executado da seguinte maneira: EXEC "comando APDU".

O comando EXECFILE executa cada linha de um arquivo como uma série de comandos do SMART SHELL, recebendo o caminho do arquivo como parâmetro. Esse comando deve ser executado da seguinte maneira: EXECFILE "caminho do arquivo (ex: C:\apdus.txt)" .

O comando INSTALL instala um *applet* ou carrega um arquivo no cartão. Se o arquivo contém mais de um *applet,* cada um é instalado. Ele recebe como parâmetro o caminho do arquivo CAP (ver *Applets* Java *Card* e Aplicação *Host)*. Para realizar a instalação de uma aplicação no cartão, o SMART SHELL, ao receber o comando INSTALL, envia um ou diversos APDUs correspondentes aos comandos INSTALL e LOAD ao *card manager*. Esse comando deve ser executado da seguinte maneira: INSTALL "caminho do arquivo CAP (ex: C:\arquivo.cap)" .

O comando LIST lista informações sobre o *card manager*, applets ou arquivos instalados no cartão, recebendo dois parâmetros. O primeiro parâmetro especifica o componente a ser listado. Os valores válidos são "CARDMANAGER", "APPLETS" ou "LOADFILES". O segundo parâmetro é opcional e especifica um ID para restringir a pesquisa. Para listar os componentes do cartão, O SMART SHELL envia o APDU correspondente ao comando GET STATUS para o *card manager*.

O Comando SELECT seleciona um applet, que fica ativo e pronto para processar APDUs. Esse comando recebe como parâmetro o ID do applet a ser selecionado. Para selecionar um applet, o SMART SHEL envia o APDU correspondente ao comando SELECT para o *card manager*. O comando SELECT deve ser executado da seguinte maneira: SELECT "id do applet a ser selecionado (ex: A0:00:00:00:62:03:01)".

Por fim, os comandos HELP e EXIT são comandos próprios do SMART SHELL. O comando HELP exibe a ajuda do programa e o EXIT deve ser executado para finalizá-lo e fechar a conexão com o cartão.

6. SMART INTERFACE

O SMART INTERFACE é um software livre, que tem como objetivo o aperfeiçoamento do SMART SHELL, alterando a forma de implementação de alguns comandos dessa ferramenta em modo texto e transformado-a numa ferramenta gráfica.

Inicialmente foram desenvolvidas duas telas para o SMART INTEFACE, uma para autenticação e outra para as demais operações, a tela de gerenciamento.

A tela de autenticação é a tela inicial do programa e é mostrada quando um cartão é detectado. Através dela o desenvolvedor pode autenticar-se e iniciar as operações com o cartão. Essa tela possui o botão "Autenticar", que dispara um evento que executa o comando AUTH do SMART SHELL com algumas adaptações. A tela de autenticação é mostrada a seguir:

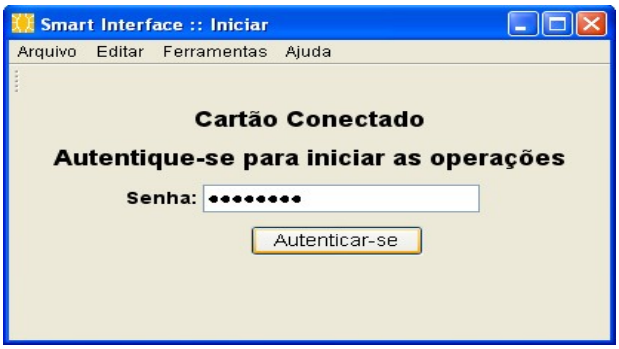

Figura 1 - Smart Interface - Tela de Autenticação

A tela de gerenciamento é mostrada quando o desenvolvedor se autentica. Ela exibe as informações do *card manager*, a lista dos *applets* instalados no cartão e a lista dos arquivos. Essa tela é dividida em dois lados, um lado para operações com *applets* e outro para operações com arquivos.

O lado para operações com *applets* possui uma lista com os *applets* carregados no cartão seguida de dois botões, "Selecionar" e "Excluir", que disparam eventos para executar os comandos SELECT e DELETE, respectivamente. Para esses comandos, é passado como parâmetro o ID do *applet* selecionado na lista. Esse lado possui ainda um campo de texto, onde deve ser digitado um APDU, seguido de um botão "Executar", que dispara um evento para execução do comando EXEC, passando o APDU digitado no campo de texto. Por fim, o lado para operações com applets apresenta um botão procurar que abre uma tela, onde o usuário pode buscar um arquivo CAP para ser instalado no cartão, e o botão "Instalar", que executa o comando INSTALL.

O lado para operações com arquivos possui uma lista com os arquivos carregados e um botão "Excluir Selecionado", que dispara um evento para executar o comando DELETE, passando o ID do arquivo selecionado na lista.

Os comandos EXIT e HELP, são executados pelo item "Sair" do menu "Arquivo" e pelo menu "Ajuda", respectivamente.

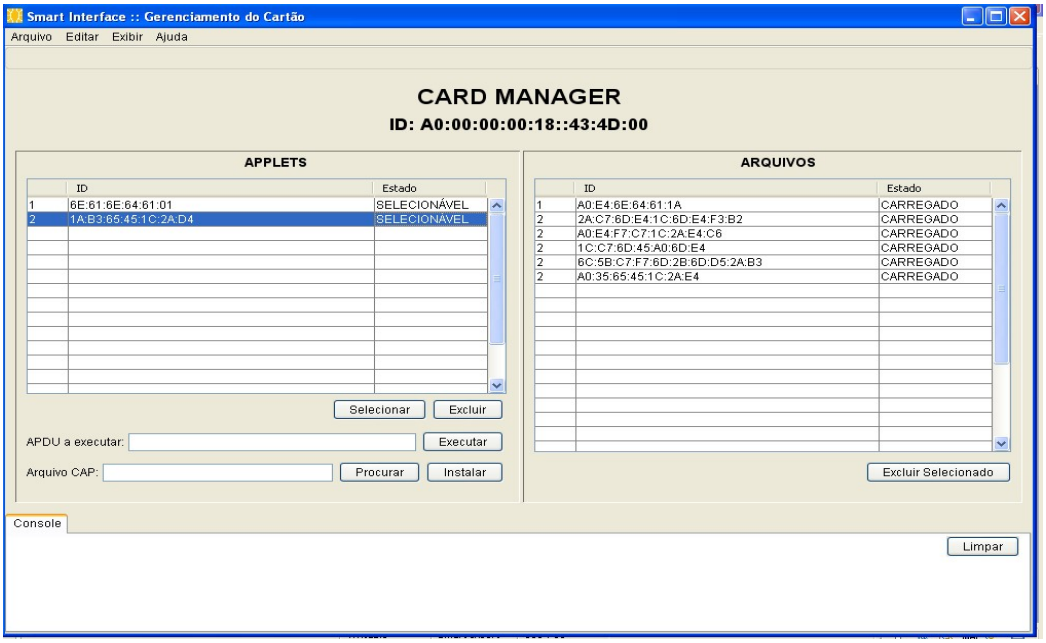

Figura 2 - Smart Interface - Tela de Gerenciamento

O Comando EXECFILE do SMART SHELL não foi implementado no SMART INTERFACE, pois seu objetivo é evitar a digitação de vários comandos por parte do desenvolvedor de aplicações, o que não faz sentido no SMART INTERFACE, pois os comandos são executados simplesmente com cliques em botões.

7. CONCLUSÃO

De acordo com o que foi proposto, o SMART INTERFACE está em fase de conclusão, faltando ainda o fim das discussões acerca das melhorias necessárias ao SMART SHELL, para que sejam implementadas. Contudo, as melhorias já propostas foram concluídas.

O sistema resultante desse trabalho é de grande utilidade para os desenvolvedores de aplicações Java *Card,* facilitando a complexa manipulação de um cartão inteligente e agilizando várias operações, através de uma interface gráfica de uso simples e prático.

Para a conclusão do SMART INTERFACE, resta ainda definir as melhorias necessárias ao SMART SHELL, que ainda não foram implementadas e um aperfeiçoamento do *layout* gráfico da aplicação, além de testes mais rigorosos.

REFERÊNCIAS

CHEN, Zhiqun. **Java Card Technology for Smart Cards***:* Architecture and programmer´s Guid. San Antonio Road – California: Addison-Wesley, 2004. 368 p.

MEIRELES, Paulo. R. M.; SOUZA NETO, Plácido. A. **JML**: Design by Contract em Aplicações JavaCard: Natal-RN: Departamento de Informática e Matemática Aplicada (DIMAp) – Universidade Federal do Rio Grande do Norte (UFRN), 2006.

MEIRELES, Paulo R. M.; VIANA, Crístian D. S.; VIDAL, Jorgiano M. B. Desenvolvendo uma Aplicação Host com O OpenCard Framework 1.2 para um Cartão Inteligente.

VIANA, Crístian Dos Santos. **GP Comm**: Uma Biblioteca Java de Apoio à Comunicação com um Cartão Inteligente Compatível com a GlobalPlatform. Natal-RN: CEFET-RN, 2007. 106 p.

ETSI (European Telecommunications Standards Institute**). World Class Standards.** Disponível em: <http://www.etsi.org>. Acesso em: 10 julho de 2007 -5

GLOBALPLATFORM**. The Standard for Smart Card Infrastructure.** Disponível em: <http://www.globalplatform.org >. Acesso em: 15 de junho de 2007 -4

ISO (International Organization for Standardization**). International Standards for Business, Government and Society**. Disponível em: <http://www.iso.org>. Acesso em: 15 de junho de 2007.

OpenCard Consortium. **OpenCard Framework** .Disponível em: < http://www.opencard.org>. Acesso em: 26 de julho de 2007 -6

PC/SC Workgroup**. PC/SC Workgroup Specifications**. Disponível em: <http://www.pcscworkgroup.com> Acesso em: 13 de julho de 2007 -7