UTILIZAÇÃO DE JAVA CARD COMO PLATAFORMA PARA O DESENVOLVIMENTO DE APLICAÇÕES EM SMART CARD.

Lucas Plis Dolce, Francisco Assis da Silva, Silvio Antonio Carro

Faculdade de Informática (FIPP) – Universidade do Oeste Paulista (UNOESTE) – Presidente Prudente – SP. E-mail: silvio@unoeste.br

RESUMO
Com o crescimento da tecnologia Java Card e o aumento do uso de Smart Cards no mercado, aumentaram-se a demanda pelo desenvolvimento de aplicações que são executadas nos cartões. Essas aplicações necessitam oferecer além de portabilidade, comodidade e segurança, serviços com qualidade e alta disponibilidade para os usuários. Nesse artigo são abordados os conceitos principais da tecnologia Smart Card bem como o uso destas para desenvolver pequenas aplicações usando a linguagem Java Card focando o uso para sistemas de controle nas quais o cartão serviria de repositório de alguns dados e instrumento de consulta de sistemas maiores.
Palavras-chave: cartões inteligentes - Java Card - Applets

USING JAVA CARD AS A PLATFORM FOR DEVELOPING APPLICATIONS IN SMART CARD.

ABSTRACT
With the growth of Java Card technology and the increased use of smart cards in the market, demand for the development of applications that run on the cards has risen. These applications need to offer besides portability, convenience and safety, service quality and high availability for users. This article examines the main concepts of Smart Card technology and use these to develop small applications using the Java Card focusing on the use for control systems where the card would serve as a repository of some data and query tool for larger systems.
Keywords: Smart card - Java Card - Applets

INTRODUÇÃO

O advento da Internet alterou rapidamente a maneira das pessoas se comunicarem umas com as outras, o mundo se tornou conectado e os negócios evoluíram do modelo tradicional frente a frente para o modelo virtual onde através de poucos cliques as pessoas fazem compras sem sair de suas casas ou escritórios. Esse grande crescimento do comércio eletrônico não abriu novos caminhos apenas para os comerciantes, mas também trouxe várias oportunidades para as indústrias alcançarem novos clientes e objetivos. Outro grande responsável pelo crescimento da tecnologia no mercado é a área de telecomunicação móvel, com os chips GSM, no qual a maior parte deles utiliza a tecnologia Java Card. Os Smart Cards oferecem uma maneira segura, confiável, conveniente e eficaz para garantir a segurança de comércio eletrônico e permitir uma ampla gama de novas aplicações [Chen 2000].

Como exemplos de aplicações que usam Java Card e trazem benefícios para a população e para o comércio em geral tem-se o Projeto RIC – Registro de Identidade Civil [Projeto RIC 2009] que tem como objetivo substituir o RG, CPF, Título de Eleitor, CNH, Passaporte e Carteira de Trabalho por um único cartão que contenha todos essas informações como forma de tentar acabar com as fraudes e duplicidades em serviços públicos e facilitar a vida dos cidadãos. Já na área comercial existe o sistema nomeado WhiteCard [Comsoft 2006] que é um sistema que atende a necessidade de controle de informações sobre abastecimentos de frotas em postos de combustíveis, garantindo agilidade no atendimento por não depender de nenhum sistema on-line em funcionamento no momento do abastecimento. Com base na mais avançada tecnologia disponível, o sistema é composto por um ou mais terminais de captura off-line (que são os leitores de cartão), por um ou mais cartões comerciante (usados para armazenar as informações dos abastecimentos), por um ou mais cartões consumidor (pertencentes ao dono da frota) e possui um software de gestão via web que proporciona um relacionamento transparente e confiável entre o posto e o frotista, elimina discordâncias entre relatórios de consumo e possibilita a redução de despesas administrativas com coleta e digitação de dados das bombas de combustível, preparação e emissão de relatórios.

Este trabalho tem como objetivo o desenvolvido de uma aplicação em dois módulos, o primeiro é uma aplicação *host* que tem as funções de acessar, gravar e consultar informações no Smart Card e o segundo são as *applets* Java Card que devem ser instaladas nos Smart Cards.

Foram investigadas algumas tecnologias e recursos como: Java Card, que foi a linguagem de programação utilizada para o desenvolvimento do projeto; o Smart Card responsável por armazenar as aplicações desenvolvidas; o Protocolo de Comunicação APDU responsável pela comunicação entre o cartão e a leitora e o ambiente de desenvolvimento Eclipse com integração da biblioteca JCOP TOOLS.

SMART CARD

Parecido com os cartões de crédito de tarja magnética, os Smart Cards possuem capacidade de armazenamento e processamento de informações de maneira segura e confiável. O Smart Card é independente de sistema corporativo e de banco de dados, o seu funcionamento é de forma isolada. Sistemas convencionais têm seu funcionamento prejudicado caso ocorra uma falha na rede de comunicação devido ao não acesso às informações no banco de dados, esses sistemas

podem ser desenvolvidos para funcionar de maneira *off-line* usando Smart Cards para armazenar as informações sem a necessidade de consultar constantemente o banco de dados.

Ao contrário das práticas de tecnologia da informação no âmbito do PC, o desenvolvimento das funcionalidades dos Smart Cards é fortemente influenciado por padrões internacionais como, por exemplo, ISO/IEC. A razão para isso é que a interoperabilidade é um fator muito importante para o Smart Card, portanto todas as suas características físicas e elétricas são guiadas por normas e padrões [Rankl 2007]. A Figura 1 representa um Smart Card.

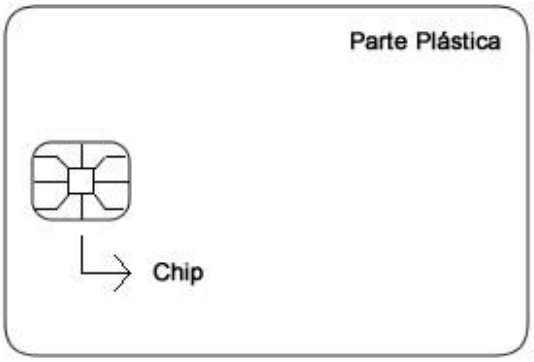

Figura 1. Smart Card

Existem dois tipos de Smart Card (cartão inteligente): com contato e sem contato. Smart Cards com contato necessitam de contato físico para se comunicar e trocar informações com o exterior, são aqueles que necessitam de um leitor de cartão, como exemplo os cartões de crédito atuais que precisam ser inseridos em um leitor no caixa eletrônico para realizar determinado serviço. Já Smart Cards sem contato são aqueles que não precisam de contato físico para se comunicar e trocar informação com o exterior, eles possuem uma antena embutida em seu interior e se comunicam por meio de rádio frequência, como exemplo os cartões de transporte coletivo o qual ao se aproximar de um equipamento transmissor de rádio frequência faz a comunicação e a troca das informações necessárias [Rankl 2007].

Todo Smart Card contém alguns elementos de hardware que são obrigatórios (EEPROM ou Flash, RAM, ROM e CPU) como a Figura 2, mas também podem conter vários outros elementos que são opcionais e servem para determinados tipos de serviços e aplicações como, por exemplo: Timer, USB, Co-processador para trabalhar com criptografia e outros.

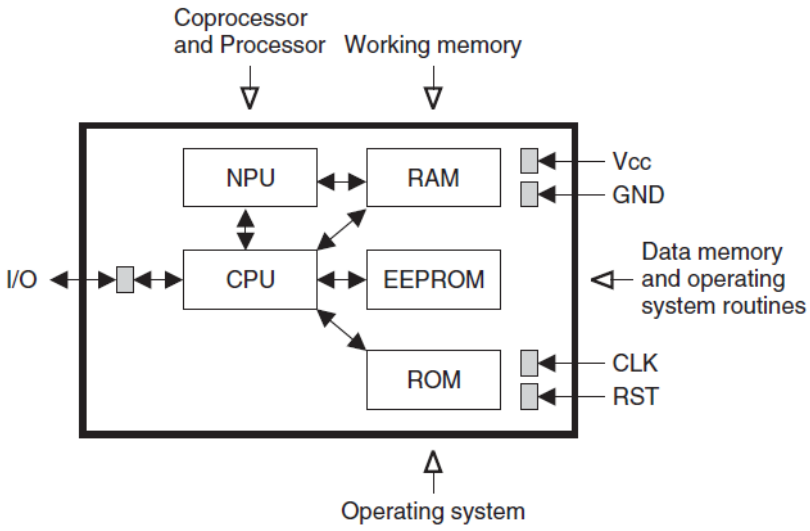

Figura 2. Diagrama de blocos de um microcontrolador de um cartão inteligente com uma interface de contato

No projeto foi utilizado o Smart Card JCOP - SmartMX JCOP21 V2.3.1 on Secure PKI com 36 KB de capacidade de armazenamento, com um Co-Processador auxiliar para trabalhar com funções criptográficas RSA e ECC podendo gerar chaves RSA com comprimento de até 2432 *bits*, habilitado para executar algoritmos como SHA-1, MD5 e CRC e com suporte às interfaces de contato do tipo T=0 e T=1 definidas pela ISO/IEC 7816-3 [Scm 2006].

Em um Smart Card pode ser instalado qualquer tipo de aplicação de pequeno porte que seja possível de ser executado com as limitações do hardware existentes e que necessitem de portabilidade, comodidade e segurança como, por exemplo, aplicações para TV a cabo ou satélite, telecomunicações em geral, identificação, autenticação, moeda eletrônica, controle de acessos, programas de fidelidade, controle em universidades e controle em hospitais [Chen 2000].

JAVA CARD

A idéia da tecnologia Java Card é colocar um software Java em um sistema de cartão inteligente, preservando o espaço suficiente para as aplicações, no entanto, o hardware de um microcontrolador de um cartão inteligente tem poder de processamento significativamente menor do que o hardware de um PC, então apenas um subconjunto da linguagem Java é definida pelas especificações da linguagem Java Card. Para desenvolver aplicações Java Card é aconselhado levar em conta todos os aspectos da especificação desde o início do processo de desenvolvimento, pois caso contrário será bastante difícil aumentar a velocidade ou diminuir o tamanho de um programa caso necessário [Rankl 2007].

O código do programa é armazenado na memória não-volátil do cartão inteligente, memória EEPROM ou *flash*.

Devido a limitação de hardware existente em um Smart Card a especificação Java Card suporta algumas funcionalidades da linguagem Java enquanto outras não são suportadas. A Tabela 1 relaciona algumas das funcionalidades suportadas e não suportadas.

Tabela 1. Funcionalidades suportadas e não suportadas

Funcionalidades suportadas	**Funcionalidades não suportadas**
Tipos primitivos: boolean, byte, short	Tipos primitivos: long, double, float
Suporte ao tipo int é opcional	Caracteres e string
Arrays unidimensionais	Arrays multidimensionais
Pacotes, classes, interfaces e exceções	Threads
Herança	Clone de objetos

A máquina virtual Java Card presente no cartão é dividida em duas partes: uma que é executada *off-card* e outra que é executada on-card. A parte *on-card* da máquina virtual é responsável por executar os arquivos CAP (*Converted Applet*) independente do hardware envolvido (arquivos CAP possuem uma representação binária do aplicativo desenvolvido) e a parte *off-card* é responsável por realizar a conversão dos arquivos de classe em arquivos CAP. Essa divisão é necessária devido as limitações de recursos nos dispositivos envolvidos [Chen 2000]. A Figura 3 mostra os componentes da máquina virtual.

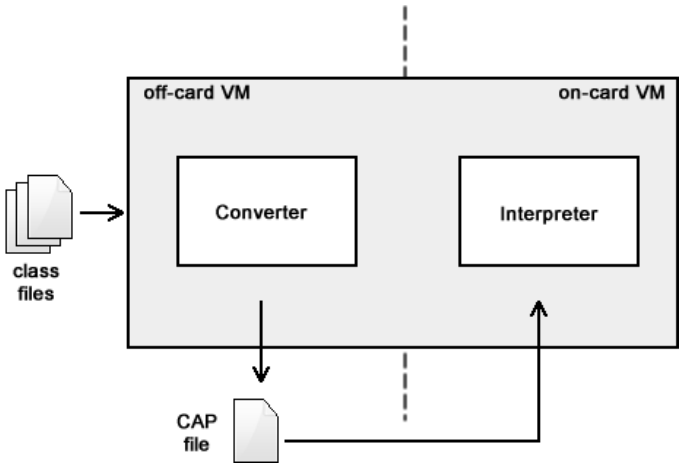

Figura 3. JCVM (Máquina Virtual Java Card)

O JCRE (*Java Card Runtime Environment*) é o ambiente de execução no cartão inteligente, é responsável por executar os aplicativos, gerenciar os recursos do cartão, comunicação e segurança. É inicializado apenas uma vez e isso ocorre durante o período de fabricação do cartão no qual são inicializados a máquina virtual e os objetos que prestam serviços para o ambiente de execução. Quando a energia é removida, o JCRE é suspenso e mantém seu estado, objetos e dados em memória persistente. Quando o cartão é re-energizado o JCRE é reiniciado e carrega os dados que estão na memória persistente retomando o seu funcionamento no estado exato em que tinha sido deixado, a partir desse momento o JCRE fica aguardando algum comando enviado pelo aplicativo *host*. O JCRE se mantém em estado de execução durante uma sessão entre o cartão e o leitor (uma sessão é o tempo em que um cartão permanece conectado e energizado pelo leitor) [Chen 2000].

Na sequência são mostradas as etapas passo a passo desde a criação de uma *applet* Java Card até a sua instalação e execução no Smart Card.

1 – Definir as APDUs de comando e resposta: através desse protocolo de comunicação será possível inserir, consultar e excluir dados do cartão.

2 – Criar as *applets*: escrever o código Java Card da aplicação.

3 – Gerar os arquivos .class: através do ambiente de desenvolvimento, realiza a compilação da aplicação e automaticamente são criados os arquivos .class.

4 – Definir o AID: é um *array de bytes* em hexadecimal que permite que a aplicação seja selecionada, toda vez que for necessário utilizar uma aplicação a primeira coisa a ser feita é selecionar a aplicação através desse AID para depois realizar a troca de informações com o cartão.

5 – Gerar o arquivo .cap: todas os arquivos .class devem ser convertidos em um único arquivo .cap como se fosse um pacote no qual todas as classes foram convertidas em um único arquivo.

6 – Instalar a *applet* no cartão: através do ambiente de desenvolvimento é possível instala a aplicação no Smart Card.

7 – Envio e recebimento de comandos APDUs: após as etapas anteriores a aplicação já está pronta para ser utilizada, para se comunicar com a aplicação Java Card deve-se enviar comandos APDUs definidos na primeira etapa.

A Figura 4 ilustra todos os passos citados anteriormente para o desenvolvimento de uma aplicação Java Card.

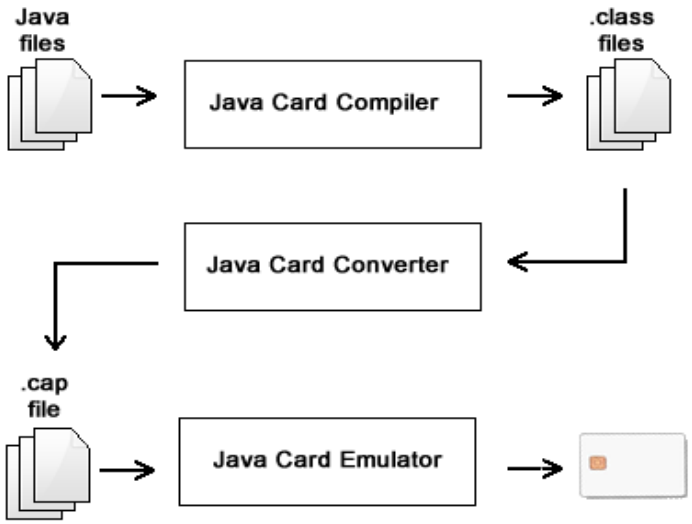

Figura 4. Etapas para o desenvolvimento de uma aplicação Java Card

PROTOCOLO DE COMUNICAÇÃO

APDU (*Application Protocol Data Unit*) é um protocolo em nível de aplicação e foi utilizado nesse projeto para realizar a comunicação e a troca de informações entre o Smart Card e a aplicação *host*.

Ele trabalha através de mensagens que podem ser divididas em duas estruturas: a primeira que é utilizada pela aplicação host para enviar comandos para o cartão e a segunda que é utilizada pelo cartão para enviar respostas para a aplicação *host*. Toda mensagem de comando APDU é combinada com uma mensagem de resposta APDU [Chen 2000].

As Figuras 5 e 6 representam o formato de um comando e de uma resposta APDU respectivamente.

Cabeçalho obrigatório				Corpo opcional		
CLA	INS	P1	P2	Lc	Data field	Le

Figura 5. Formato de um comando APDU

CLA: identifica a categoria dos comandos e respostas APDU.

INS: especifica qual o comando.

P1 e P**2**: usados quando necessário para fornecer mais informações para a instrução.

Lc: especifica o tamanho em *bytes* do campo **Data field**.

Data field: contém os dados que são enviados para o cartão executar alguma instrução.

Le: especifica o numero de *bytes* esperado como resposta a esse comando.

Corpo Opcional	Cabeçalho obrigatório	
Data field	SW1	SW2

Figura 6. Formato de uma resposta APDU

Data field: dados enviados pelo cartão como resposta ao comando APDU, com o tamanho igual ao especificado no campo Le do comando APDU (Figura 5) .

SW1 e **SW2**: são chamados de palavras de *status*, retornam códigos hexadecimais, por exemplo, “0x9000” que indicam se o comado foi

executado com sucesso ou não e o que aconteceu caso tenha ocorrido algum problema.

Os possíveis casos de combinação de comando e resposta APDU são mostrados na Figura 7.

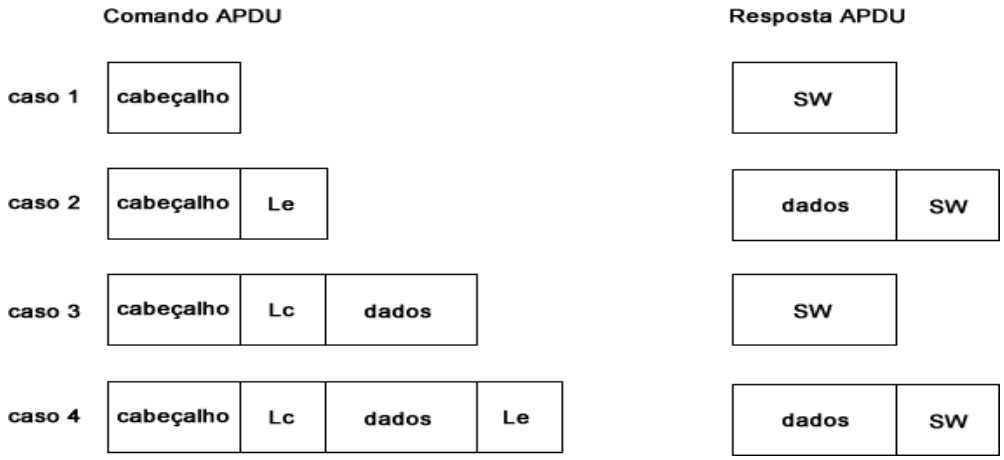

Figura 7. Combinações de comandos e repostas APDUs

AMBIENTE DE DESENVOLVIMENTO

Para desenvolver o módulo *on-card* foi utilizada a versão do *plugin* JCOP Tools 3.1.1b acompanhado das versões JDK 1.5, Eclipse Galileu 3.5.0, Java Card 2.2.1 e Global Platform 2.1.1, as bibliotecas Java Card e Global Platform já acompanham o *plugin* dispensando a sua instalação e também um kit de desenvolvimento com dois cartões JCOP21 36k e uma leitora de cartões ACR38.

O módulo *off-card* foi desenvolvido por meio da IDE NetBeans 6.8 acompanhado da biblioteca Java Card 2.2.1 e da API SmartcardIO.

Com todas essas ferramentas e bibliotecas instaladas tem-se o ambiente gráfico que automatizou o uso das ferramentas disponíveis para o desenvolvimento do projeto.

DESENVOLVIMENTO DE APLICAÇÕES

Para demonstrar o processo de desenvolvimento de uma aplicação Java Card, este artigo traz um exemplo de uma aplicação como trechos de código para enviar e receber comandos APDUs no módulo *off-card* e um trecho da comunicação entre os módulos internos ao cartão.

A aplicação foi dividida em dois módulos: o primeiro é um módulo *on-card*, ou seja, um módulo que trata todas as classes, funções e serviços que foram inseridos no cartão e o segundo é um módulo *off-card*, ou seja, um módulo que trata toda a parte da interface gráfica da aplicação *host* e de como o usuário interage, se comunica e troca informações com o cartão.

Módulo *on-card*

Seguindo as etapas de como desenvolver uma *applet* Java Card a primeira coisa realizada foi a definição dos comandos e respostas APDUs, pelos quais o módulo *on-card* se comunica com o módulo *off-card*. A tabela 2 mostra todos os comandos APDUs definidos:

Tabela 2. Definição dos comandos APDUs

Variável	Função	Valor
CLA_INFOESTE	Define a classe de comando e resposta para essa aplicação	(byte) 0xB0
INS_SET_CURSOPALESTRA	Realiza a inscrição de cursos e (ou) palestras passados por parâmetro.	(byte) 0x10
INS_DEL_CURSOPALESTRA	Realiza o cancelamento da inscrição dos cursos e (ou) palestras passados por parâmetro.	(byte) 0x20
INS_GET_NTOTALCURSOPALESTRA	Retorna o numero total de inscrições de cursos e palestras.	(byte) 0x30
INS_GET_LISTATODOSCURSOPALESTRA	Retorna o código de todos os cursos e palestras que estão inscritos no cartão.	(byte) 0x40

A *applet* principal reescreve obrigatoriamente alguns métodos devido ao relacionamento de herança que possui com a classe Applet (javacard.framework.Applet) como: *Seletc, deSelect, install e process*.

O método *Select* é responsável por selecionar a aplicação e retorna um valor *true* caso a aplicação seja selecionada. O método *deSelect* pelo contrário retira a seleção da aplicação, ou seja, deixa o cartão liberado para utilizar outra applet.

Já o método *install* é o primeiro a ser acessado dentro da applet toda vez que ela for selecionada esse método pode ser comparado ao método *main* de uma aplicação Java e é ele quem tem a responsabilidade de criar uma nova instância da *applet* executando o seu construtor.

O método *process* é quem recebe os comandos APDUs, verifica se eles são válidos e direciona a aplicação para executar o comando APDU.

Outros dois métodos de extrema importância para a aplicação são os métodos que enviam (Figura 8) e recebem (Figura 9) dados do módulo *off-card*.

```
1  private void enviarDados(APDU apdu, byte[] dados) {
2
3      short dataLen = (short) dados.length;
4      byte[] buffer = apdu.getBuffer();
5      Util.arrayCopyNonAtomic(dados, (short) 0, buffer, (short) 0, dataLen);
6      apdu.setOutgoingAndSend((short) 0, dataLen);
8  }
```

Figura 8. Método que envia dados para a aplicação *host*

```
1   private byte[] receberDados(APDU apdu) {
2
3      byte[] dados = new byte[127];
4      byte[] buffer = apdu.getBuffer();
5      short lengthDados = (short) (buffer[ISO7816.OFFSET_CDATA] & 0x00FF);
6      if (lengthDados == 0)
7            ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);
8      Util.arrayCopy(buffer, (short) ((ISO7816.OFFSET_CDATA) & 0x00FF),
9      dados, (short) 0, (short) 100);
10     return dados;
12  }
```

Figura 9. Método que recebe dados da aplicação *host*

Módulo *off-card*

Para o desenvolvimento desse módulo foi utilizada a API SmartcardIO que dá suporte para trabalhar com a leitora de cartão e com o cartão. A API prevê funções que listam as leitoras disponíveis, as leitoras conectadas e também funções que ajudam a identificar se existem cartões ou não conectados às leitoras.

A Figura 10 mostra a tela principal do módulo *off-card* por onde o usuário realiza a interação com o módulo *on-card*.

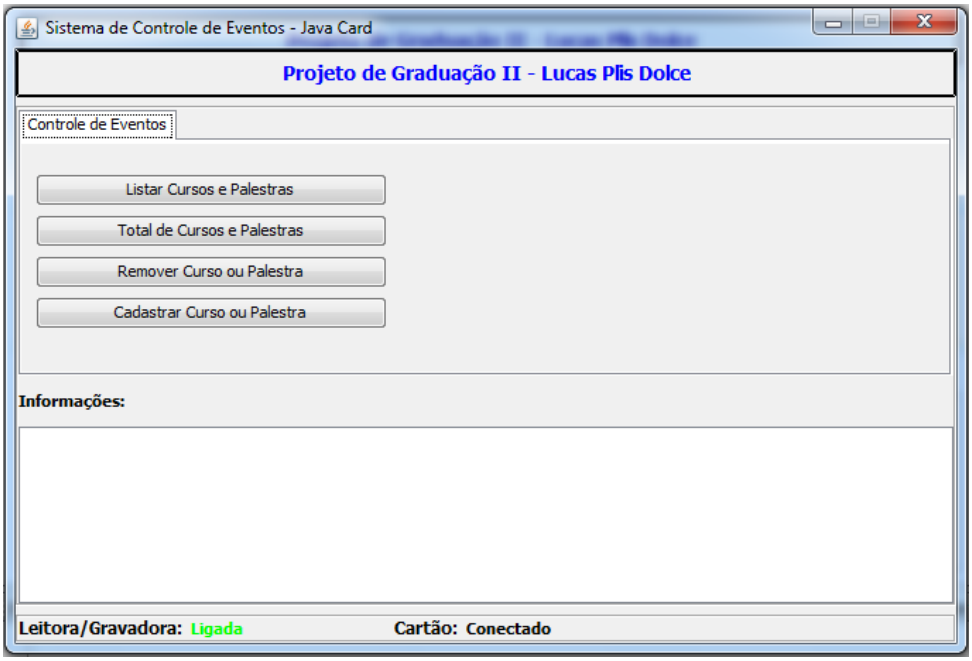

Figura 10. Interface gráfica do módulo *off-card*

O módulo *off-card* faz toda a troca de informações com o módulo *on-card* através de envios e recebimentos de comandos APDUs, a Figura 11 mostra um trecho de código que realiza essa operação.

```
1  if (c.getTerminal().isCardPresent()) {
2          c.iniciaComunicacao();
3
4          byte[] select = {(byte) 0x00, (byte) 0xA4, (byte) 0x04, (byte) 0x00, (byte) 0x08, (byte) 0x43, (byte) 0x6f,
5          (byte) 0x6e, (byte) 0x74, (byte) 0x72, (byte) 0x6f, (byte) 0x6c, (byte) 0x61};
6          byte[] apdu = {(byte) 0xB0, (byte) 0x30, (byte) 0x00, (byte) 0x00, (byte) 0x00};
7
8          ResponseAPDU r_select = c.getCanal().transmit(new CommandAPDU(select));
9          ResponseAPDU r = c.getCanal().transmit(new CommandAPDU(apdu));
10
11          Informacao.append(r + "\n\n");
12          JOptionPane.showMessageDialog(null, Host.paraStringHexa(r.getData()));
13 } else {
14          JOptionPane.showMessageDialog(null, "Cartão não conectado!");
15 }
```

Figura 11. Acessando o módulo on-card por meio da aplicação host

No trecho de código da Figura 11 pode-se observar a criação dos comandos APDUs (linhas 4 e 8), o comando select (linhas 4 e 5) é montado agregando o AID da aplicação (43 6f 6e 74 72 6f 6c 61) em hexadecimal. Na linha 6 a aplicação é selecionada e fica pronta para uso, após isso é enviado (linha 9) um comando APDU montado na linha 8 na classe de comandos e respostas APDU “0xB0” e a instrução “0x30”. Feito isso a aplicação on-card retorna o número total de cursos e palestras inscritas no cartão e uma palavra de *status* (SW) com valor igual a “9000” que significa que o comando foi executado com sucesso.

CONSIDERAÇÕES FINAIS

Este trabalho foi desenvolvido com o intuito de demonstrar que os Smart Cards estão evoluindo tecnologicamente, e uso destes está cada vez mais amplo, não se concentrando apenas nas suas características e funcionalidades originais. Cada vez mais estão sendo agregadas novas funcionalidades, podendo ser utilizadas não apenas para efetuar transações bancárias, mas para uma grande quantidade de aplicações e soluções.

Não é tratado nesse artigo a parte de segurança, que envolve os cartões e o acesso às informações armazenadas nele, pois o principal objetivo do projeto foi utilizar-se das tecnologias citadas nesse artigo para desenvolver pequenas aplicações de controle que venham servir de base para desenvolver outros tipos de aplicações futuramente.

REFERÊNCIAS

CHEN, Z. (2000) “Java Card Technology for Smart Cards: Architecture and Programmer’s Guide”, Edited by Addison Wesley, U.S.A.

COMSOFT. (2006). Disponível em <http://www.comsoft.com.br/?q=produtos/whitecard>. Acessado em: 12 dez 2010.

Projeto RIC. (2009) Disponível em: <http://jus.uol.com.br/revista/texto/17931/a-nova-carteira-de-identidade-e-o-projeto-ric>. Acessado em: 10 dez. 2010.

RANKL, W. (2007) “Smart Card Applications: Design models for using and programming in smart cards”, Edited by Kenneth Cox, John Wiley & Sons ltd., England.

Scm. (2006). Disponível em: <http://www.scmmicro.ru/upload/catalog/items/docs/JCOP21_SPI.pdf> Acessado em: 10 dez 2010