Marcos Antônio Birocchi Jr.
Mauricio Pereira Carrari

# INTEGRAÇÃO DOS DOCUMENTOS EM UM CARTÃO A MICROPROCESSADOR UTILIZANDO A TECNOLOGIA JAVACARD

Projeto de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro

São Paulo
2004

Marcos Antônio Birocchi Jr.
Mauricio Pereira Carrari

# INTEGRAÇÃO DOS DOCUMENTOS EM UM CARTÃO A MICROPROCESSADOR UTILIZANDO A TECNOLOGIA JAVACARD

Projeto de Formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro

Área de concentração:
Engenharia de Computação

Orientador:
Prof. Dr. Paulo Cugnasca

São Paulo
2004

# Sumário

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

A popularização da tecnologia de cartões a microprocessador e a crescente demanda por segurança na identificação de pessoas fez com que alguns países desenvolvessem projetos de substituição dos documentos de identidade de um cidadão em papel por documentos em formato eletrônico. Enquanto os documentos de identidade se tornam eletrônicos, os demais documentos de uma pessoa continuam sendo em papel, ou ainda tornam-se eletrônicos independentemente uns dos outros. Apesar de se ter os documentos em smartcards, eles residem fisicamente em suportes diferentes. Este projeto de formatura faz uma primeira análise da viabilidade técnica de se juntar todos os documentos de um cidadão sob a forma eletrônica em um único cartão. Este tipo de implementação é particularmente interessante para os services públicos, e objetiva aumentar a segurança e agilidade dos mesmos. Porém seu valor agregado está na praticidade para o cidadão. Propõe-se aqui uma solução baseada na tecnologia Java Card, que permite a interoperabilidade e a coexistência de diversas aplicações em um mesmo cartão. Concentramo-nos principalmente nos aspectos técnicos da implementação deste tipo de aplicação tanto do lado do cartão como do lado da estação de trabalho que deverá gerir o conteúdo do cartão.

## ABSTRACT

The dissemination of the smartcards technology and the growing demand for security in the personal identification domain has pushed some countries to develop projects to replace the usual paper-format citizens identification cards by electronic microprocessor cards. Whereas the ID cards become electronic, the other papers of a citizen remain in the usual form, or they are upgraded to an electronic format independently of the ID card. Although one may have all his/her documents as smartcards, they physically resides on different supports. Our graduation work aims to do a preliminary analysis about the technical viability of gathering all the citizen's papers in electronic format in one single card. This kind of implementation is particularly interesting for public services, and envisages increasing the security and agility of these services. But its added-value regards the convenience for the citizen. We propose here a solution based on the use of the Java Card technology, which allows the interoperability and co-existence of several applications in a same smartcard. We concentrate mainly on the technical aspects of developing such an application both from the card side and the workstation side.

# ESTRUTURA DO DOCUMENTO

O corpo principal deste documento é composto por 5 capítulos:

**Introdução**

Apresentamos nesta parte os objetivos do projeto, o porquê da escolha do tema de pesquisa e definimos o escopo do trabalho efetuado.

**Aspectos conceituais**

Falaremos um pouco sobre a história da tecnologia de cartões a microprocessador e qual foi sua primeira utilização comercial. Traçaremos em seguida um estado da arte desta tecnologia, apontando todos os domínios no qual ela é aplicada hoje em dia em todo mundo. Terminamos descrevendo alguns dos projetos existentes na área de governo eletrônico e que utilizam cartões a microprocessador.

**Arquitetura e funcionamento de um cartão a microprocessador**

Neste tópico mostramos como é por dentro um cartão a microprocessador, falamos das vantagens da tecnologia Java Card e sua importância para a popularização dos *smartcards* e explicamos de maneira objetiva como se dá a comunicação entre um terminal e um cartão.

**Desenvolvimento do projeto**

Falaremos da metodologia de desenvolvimento adotada e mostraremos o trabalho realizado ao longo do ano, desde a sua especificação técnica até à descrição da solução implementada para cada uma das sub-partes do projeto.

**Conclusões**

Fazemos um balanço dos resultados obtidos, dos problemas encontrados e do aprendizado do grupo com este trabalho. Por fim, analizamos brevemente a viabilidade da solução proposta e sugerimos possíveis seqüências para o projeto visando tornar o protótipo desenvolvido em um produto.

# 1. INTRODUÇÃO

## 1.1. Objetivo do projeto

Este projeto tem por objetivo realizar um primeiro estudo para a implementação de identificação eletrônica voltada para os serviços públicos, visando aumentar a segurança e a agilidade de tais serviços. A utilização da tecnologia Java Card estará permitindo a interoperabilidade de diversas aplicações em um único cartão a microprocessador.

O produto final do trabalho serão alguns cartões programados com os *applets*[1] personalizados representando os diferentes documentos de uma pessoa sob formato digital, além é claro de uma aplicação, ao nível da estação de trabalho na qual o leitor de smartcards estará conectado, para efetuar a gravação, leitura e atualização dos dados no cartão.

Em um único cartão poderão estar os dados do RG, CPF, carteira de habilitação, título de leitor, passaporte e certificado de reservista, de acordo com os documentos que uma dada pessoa possui. Além disso o cartão conterá alguns dados médicos que podem ser úteis em caso de emergência, tais como o tipo sangüíneo ou uma alergia a algum medicamento.

## 1.2. Justificativa

Este projeto segue uma tendência que pode-se observar em diversos países, principalmente na Europa, da substituição dos tradicionais documentos em suporte visual por documentos eletrônicos sob a forma de cartões a microprocessador. Projetos

---

[1] *Applet* é um pequeno programa, uma classe em Java que encapsula um conjunto de dados e os métodos que os manipulam.

como este já existem na Bélgica, Itália, Finlândia e Suécia. Diversos exemplos de uso podem ser imaginados para um tal documento:

- automatização da identificação nas eleições, assim como da emissão do comprovante de votação e sua gravação no cartão;
- inserção de "vistos eletrônicos" no passaporte e sua verificação em aeroportos;
- identificação de um motorista infrator pego em flagrante e consulta automática do seu cadastro junto ao Detran;
- identificação por RG na recepção de edifícios comerciais;
- bancos, lojas;
- qualquer outra aplicação que necessite identificação.

A tecnologia Java Card é a nova tendência para o desenvolvimento de aplicações na área de cartões, permitindo uma grande flexibilidade de implementação e a integração de diversas aplicações em um único cartão. Embora ainda seja mais cara do que outras tecnologias de implementação para cartões, a tecnologia Java Card vem sendo cada vez mais utilizada por permitir um ambiente multi-aplicação e interoperável, por ser um padrão aberto e por ser compatível com as normas ISO 7816 e EMV (*Europay, Mastercard, Visa*).

Através deste projeto estaremos nos familiarizando com esta tecnologia, nos aprimorando em programação Java e estudando um campo que será certamente útil em um futuro próximo. É também importante ressaltar que neste projeto estaremos desenvolvendo código tanto em aplicações de baixo nível, entre o cartão e o leitor em um ambiente cuja memória é limitada, quanto em aplicações de alto nível, entre a estação de trabalho e o leitor.

## 1.3. Escopo

O projeto consiste em desenvolver as diferentes *applets* residentes no cartão encapsulando os seguintes dados:

- Carteira de identidade (RG)
- Cadastro de pessoas físicas (CPF)
- Carteira Nacional de Habilitação (CNH)
- Título Eleitoral
- Passaporte
- Certificado de reservista
- Dados médicos de urgência

Será desenvolvida, ainda, uma aplicação para a estação de trabalho, permitindo a criação de um cartão, sua leitura e atualização dos dados.

Como último passo, caso haja tempo para tanto e dependendo da facilidade de implementação, será feita a criptografia dos dados.

Deve estar claro que os seguintes tópicos não estão no escopo do projeto:

- o estudo das questões legais envolvendo a substituição dos documentos por um suporte eletrônico e a emissão deste;
- as questões legais e éticas que envolvem a inserção de dados médicos no cartão;
- as questões envolvendo o reconhecimento legal de uma assinatura eletrônica em um sistema de *Public Key Infrastructure* (PKI);
- os serviços que estariam utilizando os dados do cartão, como por exemplo alguma aplicação de *e-government* com autenticação remota do usuário.


**Figura 1 - Escopo do projeto**

# 2. ASPECTOS CONCEITUAIS

## 2.1. O que é um Smartcard?

Smartcard (ou cartão a memória ou a microprocessador, em português) é um cartão com chip de memória ou microprocessador acoplado e de tamanho idêntico aos cartões de crédito atuais. Este chip armazena dados e programas eletronicamente. Quando associado a um leitor, um smartcard tem o poder de processamento para servir a muitas aplicações diferentes. Como um dispositivo de acesso controlado, um smartcard torna os dados pessoais ou comerciais nele contidos disponíveis apenas para os usuários apropriados. Inventados em 1974, hoje os smart cards já são usados em diversas situações: desde controle de acesso a prédios e até como moeda eletrônica. Smartcards proporcionam portabilidade, segurança e conveniência para seus usuários.

Os smartcards estão divididos em dois tipos: memory cards e microprocessor cards. Um memory card tem apenas a capacidade de armazenar dados, sem nenhum processador, análogo a um disquete. Além disso, ele oferece pouca ou nenhuma segurança quando comparado ao microprocessor card, porque ele não tem nem a inteligência capaz de reconhecer um intruso, nem a capacidade de suportar algoritmos de segurança. Sem um processador não há como usar algoritmos de criptografia ou mecanismos de segurança. Logo, a segurança para um memory card deve ser implementada pelo software aplicativo, e não pelo próprio cartão. Como o nosso trabalho é totalmente baseado em cartões a microprocessador, o termo *smart card* fará daqui para frente referência a tal tipo de cartão

Distingue-se ainda os smartcards com ou sem contato. Smartcards com contato devem ser inseridos em um leitor para que se possa efetuar alguma transação. Eles possuem uma pequena placa metálica na sua frente ao invés de uma tarja magnética na parte de trás, como os cartões de crédito convencionais. Quando esse cartão é inserido num

leitor, essa placa faz o contato elétrico com o mesm, permitindo a troca de dados com o cartão.


**Figura 2 - Smartcard com contato**

Já os smartcards sem contato não precisam ser inseridos num leitor: apenas precisam ser aproximados a uma antena - presente no próprio leitor - para que seja efetuada a transação. A energia para o seu funcionamento também é fornecida pelo leitor no momento da transação. Eles se parecem com um smart card comum, exceto por possuírem uma antena interna. Essa antena, juntamente com o microprocessador, permite que o cartão se comunique com o leitor ou terminal, sem a necessidade do contato entre os dois. Esse tipo de cartão é ideal para transações que devem ser processadas rapidamente, como por exemplo no transporte coletivo.

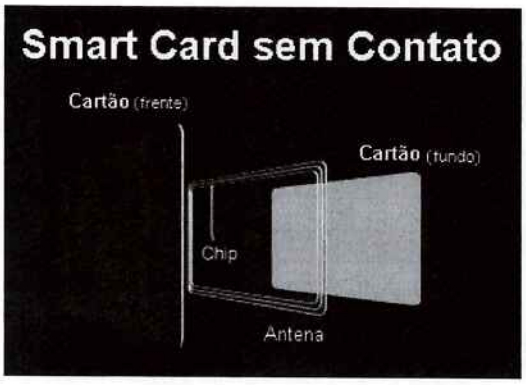



**Figura 3 - Smartcard sem contato**

Há vários tipos de mecanismos de segurança usados nos smartcards. Os mecanismos utilizados pelos memory cards são menos sofisticados do que aqueles utilizados pelos microprocessor cards. O acesso a uma informação contida num smartcard depende dos privilégios de acesso do usuário sobre essa informação. Alguns smartcards não necessitam de senha; qualquer usuário pode acessar uma informação para leitura (por exemplo, o nome e o tipo de sangue de um paciente num Medicard). Outros cartões necessitam de um PIN (Personal Identifier Number), número de identificação pessoal que deve ser digitado para validação e permissão de acesso às informações. Alguns modelos de cartões a microprocessador são equipados com criptocontroladores que comandam todas as funções de criptografia e que são projetados para cálculos extremamente complexos em alta velocidade.

As principais características de um smartcard são:

- As informações podem ser apagadas, alteradas, substituídas ou incluídas;
- Podem ser criadas senhas para acesso a cada informação (PIN), onde as aplicações serão protegidas de maneira independente;
- Muito difícil de anular qualquer informação;
- Difícil de obter dados do cartão, fraudá-los ou modificá-los indevidamente;
- Alto poder de durabilidade;
- A informação e o poder de processamento residem no próprio cartão, não sendo assim necessário o acesso a uma base remota durante uma transação.

## 2.2. *História da tecnologia de smartcards*

A história do smart card é rica e movimentada. Aparentemente, ela começa com um romance de ficção francês, "La nuit des temps", escrito por René Barjavel em meados de 1960. Neste livro, existe um aparelho eletrônico em forma de jóia, capaz de abrir portas, entre outras funções de identificação e autenticação.

A idéia de se incorporar um circuito integrado a um cartão plástico surgiu em 1968 graças aos alemães Jürgen Dethloff e Helmut Grötrupp, que em seguida patentearam a invenção no seu país. De maneira independente, o japonês Kunitaka Arimura registrou

uma patente no japão em 1970. Começam a surgir diversos pedidos de patentes na Europa, Estados Unidos e Japão. No entanto a maioria dos projetos não chega ao final de suas realizações industriais por serem muito complexos para serem implementados levando-se em conta os conhecimentos tecnológicos da época.

Atribui-se freqüentemente a invenção do smart card a Roland Moreno, que registrou 47 patentes relacionadas a smart cards em 11 países entre 1974 e 1979.

O uso comercial da tecnologia só veio no fim dos anos 70 com o Grupo Bull (na época CII-Honeywell-Bull). Mas a idéia que de fato impulsionou o mercado surgiu na França no início dos anos 80. Historicamente todos os aparelhos públicos de telefone funcionavam com moedas, o que vinha trazendo vários inconvenientes. Do lado do usuário, este deveria sempre possuir as moedas dos valores apropriados para poder efetuar uma ligação. Do lado da France Télécom, empresa que administrava o sistema, além de o sistema ser pós-pago, havia também muitas fraudes com o roubo e a depredação de aparelhos públicos. Foi então que surgiu a idéia de se usar smart cards, o que tornaria o sistema anti-fraudes e reduziria os custos de manutenção pelo fato de os aparelhos não abrigarem mais cofres com moedas, reduzindo as depredações. Além disso, o sistema passava ser pré-pago, o que sem dúvida era financeiramente muito interessante.

Criou-se assim em um curto espaço de tempo um mercado enorme para fabricantes de smart cards naquele país, o que explica de certa forma o fato de os grandes grupos do ramo serem todos de origem francesa, como a Bull, a Axalto, a Gemplus e a Oberthur. O sucesso dessa migração do sistema de telefonia público na França foi tão grande que essas empresas expandiram rapidamente seus negócios para diversos países, e juntas dominam hoje mais de 80% do mercado mundial.

## 2.3. História da tecnologia Java Card

A API Java Card foi pensada pela primeira vez em novembro de 1996 por um grupo de engenheiros da Schlumberger em Austin, no Texas. Eles trabalhavam para tornar o desenvolvimento de aplicações para smart cards algo acessível, sem no entanto abrir mão da segurança inerente à tecnologia. Eles perceberam então que a melhor saída era o uso da linguagem Java. Eles propuseram então um primeiro rascunho do que seria a API Java Card. Um tempo depois, Gemplus e Bull se juntaram à Schlumberger para formar o Java Card Forum, um consórcio que visava desenvolver e promover a tecnologia Java Card como um padrão da indústria.

A versão 1.0 do Java Card consistia apenas nas especificações das APIs. Entrou então em cena a Sun Microsystems para desenvolver o Java Card como uma tecnologia de plataforma Java para dispositivos de memória restrita. Em novembro de 1997, a Sun anunciou a versão 2.0 da especificação do Java Card, que tornava-se assim efetivamente orientado a objeto. Além disso, definiu-se mais detalhadamente o ambiente de execução (Java Card Runtime Environment). Havia porém ainda um problema: o formato de applet carregável no cartão não tinha sido especificado, o que na prática inviabilizava a interoperabilidade de applets em Java Card.

O marco inicial do Java Card pode assim ser considerado quando do lançamento da versão 2.1 em março de 1999. Foram especificados a API do Java Card 2.1, o Java Card 2.1 Runtime Environment e o Java Card 2.1 Virtual Machine. A contribuição fundamental dessa versão foi a definição da arquitetura da virtual machine e o formato de applets carregáveis, o que posiibilitava definitivamente uma interoperabilidade real dos applets. Somente nesse momento o Java Card passou a ser efetivamente utilizado na indústria e tornou-se o padrão para o desenvolvimento de applets.

## 2.4. Estado da arte

Smart cards podem ser usados em diversas aplicações nos mais variados setores da economia. Ilustraremos aqui os mais importantes domínios em que smart cards são utilizados, explicando o benefício do uso da tecnologia e citando exemplos reais.

### 2.4.1. Indústria das telecomunicações

O uso de cartões telefônicos pré-pagos oferecem um mecanismo seguro, anti-fraude e de baixo custo de manutenção para o acesso a telefones públicos. No entanto é a telefonia móvel o maior mercado de smartcards no mundo.

Toda a segurança do sistema de celulares GSM (GlobalSystem for Mobile Communications) é baseado em smart cards. Um celular GSM possui um SIM card (Subscriber Identity Module, conhecido usualmente como "chip" de celular) que identifica o usuário e fornece as chaves de criptografia para a transmissão digital de voz. Como a identificação do usuário (da linha, na verdade) é programada dentro do SIM card, a pessoa pode usá-lo em diferentes aparelhos GSM.

Mais do que a simples função de autenticação, os SIM cards têm sido usados pelas operadoras para abrigar serviços de alto valor agregado, como *mobile banking*, *instant messaging*, download de músicas e imagens, entre outros.

Além do sistema GSM, o antigo sistema CDMA começa aos poucos a utilizar SIM cards para a autenticação do usuário na rede. Isso começou graças a um esforço da China Unicom, que elaborou uma especificação similar às normas GSM e pressionou seus fornecedores a adaptar os aparelhos CDMA de maneira a abrigar os SIM cards (chamados de UIM cards na nomenclatura da China Unicom). Um número crescente de operadoras que utilizam o padrão CDMA estão começando a fazer o mesmo para não perder terreno para operadoras de GSM e para não terem que migrar de um sistema para o outro. A primeira operadora de CDMA na América Latina a utilizar UIM cards será a Telefónica Móviles do Peru a partir de 2005.

### 2.4.2. Sistema bancário

O uso de smart cards no lugar dos cartões a tarja magnética é um avanço no combate às fraudes de cartões. Contrariamente aos cartões magnéticos, os dados contidos em um chip não são facilmente copiados ou indevidamente utilizados. Na França os cartões bancários são smart cards há vários anos [20], e essa migração tem sido feita em vários outros países europeus. Alguns bancos no Brasil estão fazendo testes com essa nova tecnologia, mas o primeiro país latino-americano a efetivamente migrar todo o sistemo é o México. A expectativa da indústria é de que com a migração das fraudes do México para outros países do continente, os demais países sejam obrigados a migrar seus sistemas bancários nos próximos anos.

### 2.4.3. Moeda eletrônica

O projeto mais notável existente é o francês Moneo [19]. Este tipo de aplicação, conhecido como e-wallet ou e-purse, consiste em um cartão que é carregado com pequenas quantias de dinheiro para ser usado em compras do dia-a-dia, eliminando assim inconvenientes como falta de troco, ter que carregar muitas moedas, etc. Comparativamente a um cartão de crédito, o dinheiro eletrônico não precisa de autenticação remota, barateando o processo e eliminando assim o valor mínimo da transação normalmente exigido para a utilização do cartão.

No Brasil, lançou-se a pouco tempo o Smart VR [21], um smart card que substitui os tradionais vale-refeição. Embora seu escopo não seja genérico como o Moneo, trata-se de uma aplicação concreta de dinheiro eletrônico.

### 2.4.4. Programas de fidelidade

Os programas de fidelidade podem ser encaixados no âmbito do marketing das empresas. Os cartões são concebidos para armazenar pontos ou soma de descontos para que o cliente possa ter vantagens sendo fiel a um determinado estabelecimento ou rede de estabelecimentos. Programas de milhagem de companhias aéreas elocadoras de automóveis também se encaixam nesse perfil.

Um exemplo de programa de fidelidade no Brasil que utiliza smart cards é o Smart Club [22]. Ele transforma os gastos com cartão de crédito em pontos que podem ser trocados por produtos em lojas conveniadoas e até mesmo em cinemas.

### 2.4.5. Transporte público

A eficiência dos cartões a microprocessador também foi comprovada no setor de transporte público. As vantagens são a facilidade de manipulação pela eliminação de moedas e bilhetes, sem contar a agilidade proporcionada aos usuários e o aspecto segurança. Hoje em dia, utiliza-se smart cards em transporte urbano em algumas cidades do Brasil, como por exemplo em Campinas e mais recentemente em São Paulo.

O uso dos cartões facilita a integração dos diversos sistemas (metrô, ônibus, trens de subúrbio) e permite facilmente uma tarifação diferenciada por trecho percorrido utilizando um só cartão ao invés de vários bilhetes.

### 2.4.6. Acesso corporativo

A maioria dos prédios comerciais novos já utiliza smart cards como meio de dar acesso somente às pessoas autorizadas. Quase todas as Universidades nos países desenvolvidos utilzam a tecnologia para dar níveis de acesso às suas instalações a alunos, professores e funcionários.

Atualmente a aplicação de acesso corporativo que mais vem sendo implantada é o chamado VPN, *Virtual Private Network*. Cada usuário de uma rede corporativa se autentica nela através de um smart card. Assim um funcionário não precisa estar nos locais da sua empresa para ter acesso à rede, podendo trabalhar em casa ou acessar todas as informações que necessita durante suas viagens de maneira segura.

### 2.4.7. e-Health

Na área da saúde, os smart cards são uma ferramenta que reduz a complexidade da manipulação das informações relacionadas ao tipo de cobertura a que um paciente tem direito. O cartão tem duas funções básicas: a automatização do cálculo do pagamento a ser efetuado para um profissional da saúde e o arquivo do histórico médico do paciente.

A França foi o primeiro país a adotar o smart card no seu sistema público de saúde [25]. Antigamente, o paciente e o médico preenchiam um formulário que era enviado por correio a um centro de processamento de dados do Seguro Saúde. O paciente pagava o médico e recebia um reembolso após o processamento do seu formulário, o que podia demorar algumas semanas. No novo sistema, um formulário eletrônico é gerado e autenticado pelos cartões do paciente e do médico. O paciente não deve mais pagar o médico no ato, o pagamento sendo agora efetuado diretamente ao médico em no máximo 5 dias. Além disso, eliminou-se as fraudes e os erros existentes no sistema antigo e economizou-se gastos no processamento das informações.

Vários países europeus adotaram o sistema francês. Até 2005 quase todos os países da União Européia terão sistemas automatizados com smart cards. Embora o procedimento em cada país tenha sido automatizado, quando um cidadão europeu necessita de cuidados médicos em um outro Estado-Membro, o sistema de formulários e reembolso ainda é utilizado.

Existem dois importantes projetos na União Européia que visam acabar com os formulários necessários entre os diferentes países. Um é o projeto Netcards , cujo objetivo é viabilizar o uso dos cartões nacionais em todos os países da União de maneira transparente para o paciente. Este projeto visa dar todo o suporte técnico a um projeto da Comunidade Européia, o *e*-EHIC – electronic European Health Insurance Card – que visa implantar até 2009 cartões europeus de seguro saúde em todos os países membros [24].

### 2.4.8. e-Government

O uso de smart cards no setor público também é bastante interessante e promissor. Após o atentado de 11 de setembro de 2001, os EUA decidiram de aumentar a segurança, principalmente quanto à entrada de estrangeiros no país e uma das possíveis medidas era aplicar a tecnologia de smart card para os passaportes americanos. Outros exemplos podem ser encontrados na Bélgica, Itália, países Escandinavos e até mesmo no Brasil, onde documentos eletrônicos já começam a serem emitidos para determinadas aplicações.

No caso do Brasil, a CertSign [26] já desenvolve cartões como a Identidade Digital ou o e-CPF, que permitem que o cidadão brasileiro utilize os serviços digitais implementados pelo Governo através da Internet. Além do acesso privilegiado aos serviços governamentais, o cidadão pode assinar documentos eletronicamente, preservando o sigilo de informações pessoais e garantindo uma navegação mais segura pela Internet. No entanto, estes cartões não substituem os documentos atuais, ao contrário do que ocorre nos demais países citados.

Detalharemos a seguir alguns dos projetos mais importantes existentes na área de governo eletrônico em todo o mundo.

## *2.5. Projetos existentes na área de Governo Eletrônico*

### 2.5.1. Bélgica

Embora não tenha sido o primeiro país a apresentar uma identidade eletrônica (perde para Finlândia e Itália), a Bélgica é o primeiro país a adotá-la para toda a sua população. A decisão partiu do conselho de Ministros belga em julho de 2001. Em uma primeiro fase, 70 000 cartões foram distribuídos para 10 municípios. Após esse projeto piloto, até o fim de 2006 onze milhões de cartões serão distribuídos a toda a população do país.

Hoje o cartão conta com um sistema de segurança PKI (Public Key Infrastructure), e guarda e atualiza informações em seu chip. Em um próxima versão, a identidade belga contará com dados biométricos.

O documento eletrônico contém uma foto, o número de registro nacional único para cada cidadão, uma assinatura e uma série de dados de identificação de base do titular tanto sob forma eletrônica como sob forma visual impressos no corpo do cartão. O cartão possui também as chaves computacionais que permitem que o cidadão se identifique e se autentique remotamente, gerando uma assinatura eletrônica com validade jurídica.

Alguns serviços já estão disponíveis para os cidadãos que possuem a identidade eletrônica, como declaração de imposto de renda, declaração de tempo de trabalho, votação eletrônica em plebiscitos locais, obtenção de placa de imatriculação de veículos, emissão de certidão de nascimento, entre outros.

Técnicamente, a solução adotada se baseia em: uma rede federal de alta velocidade segura (*Federal Metropolitan Area Network*) para a troca de dados entre os serviços públicos federais; um "universal messaging engine" (UME), que é um middleware que permite a troca de informação entre os sistemas de informação heterogêneos da administração federal, das autoridades locais e os portais web; portais que integram o cidadão com os serviços públicos; e um número de identificação único para cada cidadão, que é gerado a partir de uma modificação na legislação que define o número de registro nacional.

### 2.5.2. Finlândia

A finlândia foi o primeiro país a adotar uma identidade eletrônica para seus cidadãos. No entanto, o documento eletrônico não substitui o tradicional, mas serve apenas como uma opção ao cidadão que quer ter acesso a serviços públicos on-line. Diferentemente dos demais países que possuem projetos de identidade eletrônica, o governo finlandês

disponibilizou toda a especificação técnica da solução de maneira que qualquer um possa desenvolver aplicações para o cartão. Em resumo, o cidadão tem não somente acesso a serviços públicos seguros, mas também a serviços desenvolvidos por terceiros, como bancos, agências de viagens, e qualquer setor que possa fazer uso de autenticação legal do usuário. As especificações podem ser encontradas em [31].


**Figura 4 - Identidade eletrônica finlandesa**

### 2.5.3. Itália

Foi o segundo país a apresentar uma identidade eletrônica, depois da Finlândia. No país escandinavo, no entanto, o documento eletrônico não substitui o tradicional, serve apenas como uma opção ao cidadão que quer ter acesso a serviços públicos on-line. O projeto italiano é o primeiro a substituir totalmente o documento tradicional.

O cartão italiano é híbrido. Ele possui tanto um microprocessador, com um motor criptográfico para identificação e autenticação, quanto uma tarja magnética que guarda dados grandes (1,8 Mbytes de capacidade) e serve para uma identificação no ato pela polícia. Além disso, o cartão tem impresso a banda ICAO, que permite o uso do documento como passaporte.


**Figura 5 - Carteira de Identidade Eletrônica italiana**

Em versões futuras, o cartão conterá a impressão digital do portador, assimcomo dados médicos caso este autorize.

O cartão italiana é totalmente compatível com as normas ISO 7816 e ISO 11964 (cartão a microprocessador e a tarja magnética, respectivamente), além de seguir as normas PKCS para o processo de autenticação. Maiores detalhes técnicos do projeto podem ser encontrados em [30].

### 2.5.4. Outros países

Praticamente todos os países europeus já têm projetos de substituição da identidade convencional por uma eletrônica. Isso é devido a algumas diretivas da Comissão Européia sobre governo eletrônico e uniformização dos serviços nos seus Estados membros, que passam por um processo de identificação e autenticação inter-Estados.

Porém alguns países ainda estão em fase do processo de mudança das leis para reconhecimento da identidade eletrônica, e outros ainda nem isso. Os mais avançados são a França, a Alemanha e a Suíça, cujos governos já deram sinal verde para os

respectivos congressos. Mas do ponto de vista técnico esses países aguardam ainda os resultados dos projetos piloto belga e italiano antes de lançar protótipos.

Fora da Europa, Hong Kong já possui identidade eletrônica, e os Estados Unidos terminaram em 2004 o processo de licitação para a emissão de passaporte eletrônico

# 3. ARQUITETURA E FUNCIONAMENTO DE UM CARTÃO A MICROPROCESSADOR

## *3.1. Processador e sistema de memória*

Smart cards contêm uma CPU e três tipos de memória: memória persistente e imutável (ROM), memória persistente e mutável (EEPROM), e memória não-persistente e mutável (RAM):

- **CPU – Unidade de Processamento Central**: serve como uma via de comunicação inteligente, fazendo a interface entre o cartão e seu leitor. Os chips são geralmente 805 / 8051 / H8 / RISC, de 8, 16 ou 32 bits, funcionando a 3,57 ou 5 Mhz, com alimentação de 3 ou 5 volts.
- **Read-Only Memory – ROM**: é usada para registrar os programas fixos do cartão, tais como o sistema operacional, dados e programas do usuário. Ela só pode ser gravada na hora da fabricação, em um processo chamado *masking*.
- **Electrical Eraseable Programmable Read-Only Memory – EEPROM**: similar à ROM, os dados ali gravados são preservados quando o cartão é desenergizado. A diferença é que o conteúdo da EEPROM pode ser alterado durante a vida útil do cartão. Aplicações podem ser gravadas na EEPROM após a cartão ser fabricado. A EEPROM pode ser apagada por processos elétricos e re-escrita até 10.000 vezes, e é ela que limita fisicamente a vida útil de um smart card.
- **Random Access Memory – RAM**: é o espaço de trabalho temporário para estocar e modificar dados. É uma memória não-persistente, ou seja, os dados são perdidos quando o cartão é desenergizado.

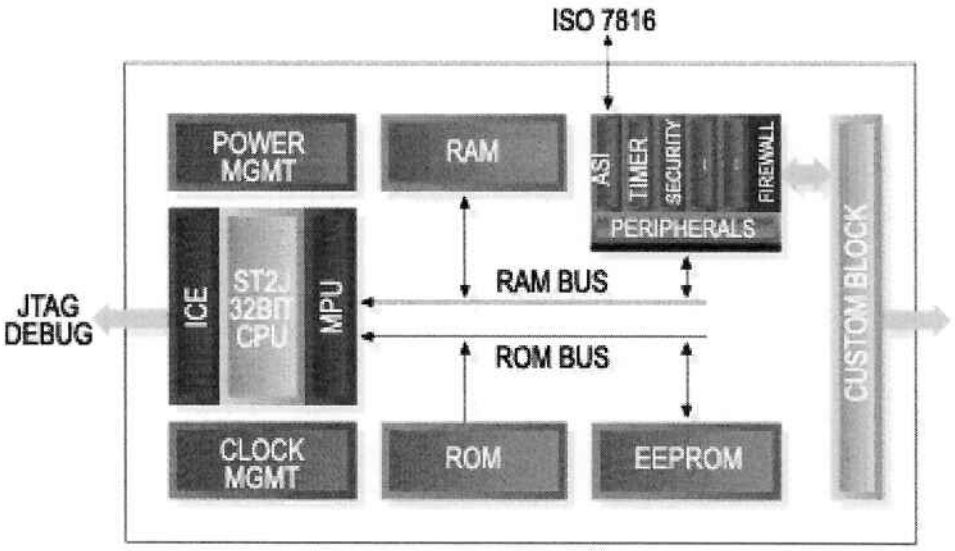



**Figura 6 - Arquitetura física de um smart card**

Os cartões hoje no mercado possuem usualmente 32 ou 64 kilobytes de memória persistente, embora já existam cartões de 128 e até de 256 kilobytes (esses últimos ainda não comercializados).

## 3.2. *Arquitetura do sistema do cartão – Java Card Runtime Environment*

Como ilustrado na figura a seguir, o sistema é composto por três camadas principais: o hardware do smart card e o sistema nativo, o Java Card Runtime Environment (JCRE) e os applets.

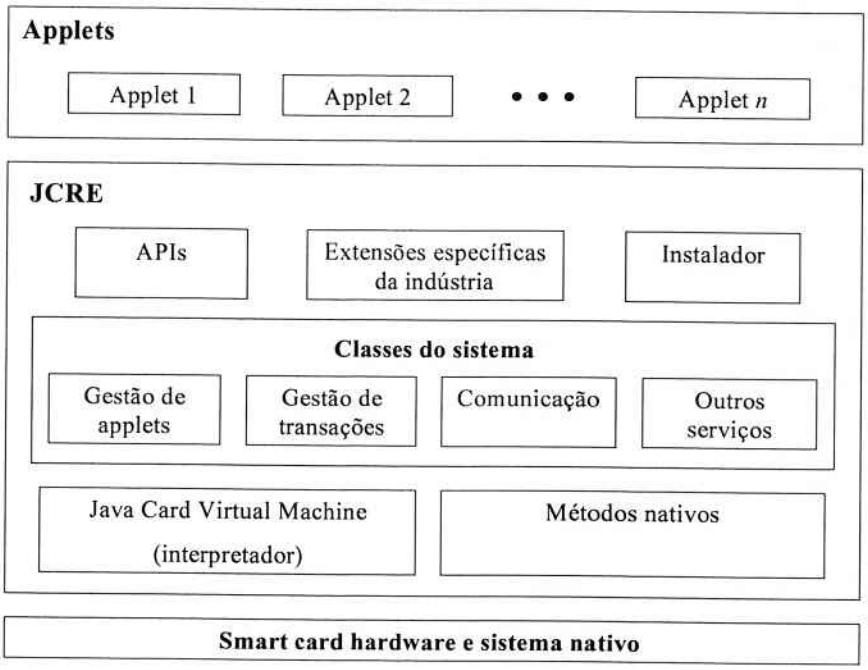



**Figura 7 - Arquitetura lógica de um smart card**

A camada inferior do JCRE é composta da Java Card Virtual Machine, que executa os bytecodes, controla a alocação de memória, entre outros, e de métodos nativos, os quais são responsáveis por manusear os protocolos de comunicação de baixo nível, pela gestão de memória e outros.

As classes do sistema funcionam como o núcleo de um sistema operacional. Elas controlam as transações, a comunicação com as applets e com as aplicações fora do cartão, a criação e seleção de applets, etc. Para tanto, as classes do sistema fazem chamadas dos métodos nativos.

Na camada superior do JCRE encontram-se as APIs do Java Card, bem como possíveis APIs de extensão definidas pela indústria, como por exemplo API específicas para

aplicações de GSM ou bancárias. Um instalador possibilita o carregamento de applets no cartão após a sua emissão.

Os applets se interfaceiam com o JCRE através das APIs do Java Card e das APIs de extensão.

## 3.3. Dificuldades no desenvolvimento de aplicações para Smart Cards

Tradicionalmente, o desenvolvimento de aplicações para smart cards era um processo longo e muito complexo. Isso era devido às enormes diferenças de funcionamento interno dos cartões entre um fabricante e outro, e à ausência de interfaces de alto nível para smart cards. Dessa forma, os desenvolvedores estavam sujeitos a lidar com protocolos de comunicação de baixo nível, gestão de memória e outros detalhes inerentes ao equipamento utilizado, o que tornava a tarefa extremamente árdua e exigia um conhecimento muito específico.

Além disso, tais aplicações eram desenvolvidas em plataformas proprietárias, impossibilitando a existência de aplicações de diferentes fornecedores em um mesmo cartão.

### 3.3.1. Aspectos de segurança

Os fatores citados acima são sem dúvida os maiores inibidores da popularização da tecnologia. No entanto, um problema crucial reside no âmbito da segurança das aplicações. Pelo fato de os programadores terem que lidar com aspectos de baixo nível tais como a gestão da memória, a segurança dos dados passa a depender fortemente da sua habilidade em programar. Ainda, e mais grave de tudo, não é possível garantir a segurança de uma aplicação face à um desenvolvedor malicioso e mal-intencionado, que pode deixar uma "porta aberta" que só ele conheça.

### 3.3.2. Java Card *versus* C

Aplicações para smart cards têm sido desenvolvidas principalmente em linguagem C/C++, o que dificulta o tratamento dos problemas de segurança mencionados e a rigor não permite a coexistência de aplicações no mesmo cartão.

A tecnologia Java Card oferece uma maneira de suplantar esses problemas, uma vez que define uma plataforma segura, portável e multi-aplicações, além de tornar a programação mais fácil e acessível. As vantagens do Java Card são:

- **facilidade no desenvolvimento de aplicações**: os programadores trabalham com interfaces de alto nível que encapsulam a complexidade e os detalhes do sistema do cartão; um programador Java pode rapidamente estar apto a desenvolver applets em Java Card
- **segurança**: diversos mecanismos são implementados. Além dos mecanismos nativos da linguagem Java, diversos outros são implementados especificamente para smart cards afim de evitar acesso indevido aos applets e à memória do cartão. Discutiremos a segurança da plataforma Java Card no tópico a seguir;
- **indepêndencia com relação ao cartão utilizado**: um applet Java Card (a partir da versão 2.1.X) pode a princípio rodar em qualquer cartão e com qualquer processador (8, 16 ou 32 bits).
- **capacidade de estocar e gerir múltiplas aplicações**: um cartão pode hospedar diversas aplicações, e mesmo applets novos ou atualizados podem ser carregados no cartão após a sua emissão (contanto que não haja mecanismos lógicos de travamento definitivo do cartão, como encontramos em alguns cartões no mercado). Um mecanismo de firewall impede que um applet enxergue ou acesse outro, a não ser que isso seja explicitamente definido;
- **compatibilidade com as normas existentes**: a tecnologia Java Card é baseada na norma ISO 7816, podendo assim suportar diferentes sistemas e aplicações compatíveis com tal norma.

### 3.3.3. Segurança da plataforma Java Card

Além dos mecanismos próprios da linguagem Java (linguagem fortemente tipada, verificação forçada do tamanho dos vetores, níveis de acesso a classes, métodos e campos estritamente controlados, etc), diversos outros são implementados especificamente para smart cards:

- **Modelo de objetos transientes e persistentes**: objetos são gravados a princípio na memória persistente, mas por motivos de segurança e performance a plataforma permite que dados temporários sejam gravados em objetos transientes na memória RAM.
- **Atomicidade de operações e transações**: para garantir a integridade dos dados e impedir a corrupção da memória do cartão (e conseqüentemente a sua perda), três dispositivos são definidos. Primeiro, a atualização de um campo de um objeto persistente ou uma classe é garantida ser atômica. Segundo, o mesmo ocorre para a atualização de blocos de elementos em um vetor. Terceiro, a plataforma Java Card suporta um modelo de transação no qual um applet pode atualizar atomicamente diferentes campos em diferentes objetos persistentes. Dessa forma, ou todos são atualizados corretamente ou todos eles são restaurados com seus valores anteriores.
- **Applet firewall**: mecanismo que reforça o isolamento de um applet. Assim, um applet só poderá ter acesso a um outro se estiverem no mesmo pacote (*package*), ou através de mecanismos bem definidos e seguros de compartilhamento de objetos.
- **Compartilhamento de objetos (*object sharing*)**: applets em diferentes contextos podem compartilhar objetos que são instâncias de uma classe que implemente uma interface compartilhável (*shareable interface*). Tais objetos são chamados de objetos de interface compartilhável (*shareable interface objects*).

## 3.4. Comunicação cartão – terminal

### 3.4.1. Modelo de comunicação

A comunicação entre um cartão e um terminal (um computador) é de tipo half-duplex, ou seja, os dados podem ser enviados do terminal para o cartão e vice-versa, mas não ambos ao mesmo tempo.

Os pacotes trocados entre o terminal e o cartão são chamados APDUs (*application protocol data units*). Uma APDU pode conter um comando (do terminal para o cartão) ou uma mensagem de resposta (do cartão para o terminal).

O modelo de comunicação é mestre-escravo, sendo sempre o cartão o elemento passivo (o escravo). O cartão espera um *comando APDU* do terminal. Ele então processa este comando e retorna uma *resposta APDU*. A troca de comandos e respostas APDU é feita sempre de modo alternado.

### 3.4.2. Protocolo APDU

Definido pela norma ISO 7816-4, é um protocolo de camada de aplicação. As estruturas dos comandos e respostas APDU são mostrados nas tabelas a seguir.

| **Comando APDU** | | | | | | |
|---|---|---|---|---|---|---|
| **Cabeçalho (obrigatório)** | | | | **Corpo (opcional)** | | |
| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |

**Tabela 1 - Estrutura do comando APDU**

| **Resposta APDU** | | |
|---|---|---|
| **Corpo (opcional)** | **Trailer (obrigatório)** | |
| Campo de dados | SW1 | SW2 |

**Tabela 2 - Estrutura da resposta APDU**

O cabeçalho do comando APDU é composto de 4 bytes: CLA (classe da instrução), identifica a categoria do par comando/resposta APDU; INS (código de instrução), especifica a instrução do comando; P1 e P2 (parâmetros 1 e 2), são os parâmetros da instrução passada (devem ser 0 se não houver parâmetros).

O corpo do comando tem comprimento variável. O byte Lc define o comprimento em bytes do campo de dados. Este campo contém os dados eventualmente necessários ao processamento do comando pelo cartão. O byte Le especifica o número de bytes esperado pelo terminal na resposta APDU ao comando.

A resposta APDU consiste em um corpo opcional e um trailer obrigatório. O corpo contém um campo de dados cujo comprimento é definido pelo campo Le do comando APDU. O trailer é composto por 2 bytes SW1 e SW2, que em conjunto são chamados de *status word*, e denota o estado de processamento do cartão após a execução de um comando APDU. Por exemplo, uma *status word* "0x9000" (SW1 = 0x90, SW2 = 0x00) significa que o comando oi executado com sucesso. Algumas status words são definidas pela ISO 7816, mas outras podem ser definidas para uma aplicação específica.

### 3.4.3. Protocolo TPDU

As APDUs são transmitidas pelo protocolo da camada de transporte, definidos pela ISO 7816-3. Os dados trocados entre o terminal e o cartão utilizando o protocolo de transporte são chamados de *transmission protocol data units*, ou TPDUs. São usados hoje em dia dois protocolos de transporte: o T=0, que é orientado a byte, e o t=1, que é orientado a blocos.

Como o TPDU é totalmente transparente para um desenvolvedor de aplicações utilizando smartcards, não o detalharemos aqui. Detalhes podem ser encontrados na ISO 7816-3.

# 4. DESENVOLVIMENTO DO PROJETO

## *4.1. Metodologia*

Pelo fato de a equipe ser constituída por apenas dois membros, o que facilita a coordenação e a comunicação, adotou-se uma metodologia tradicional de especificação, desenvolvimento e testes. Desde a especificação, o projeto foi dividido em duas partes: uma relacionada à aplicação para PC e outra relacionada ao smart card, cada membro sendo responsável por uma delas. Isso permitiu um desenvolvimento em paralelo durante todo o projeto, o que exigiu sem dúvida uma intensa comunicação para que as duas partes andassem na mesma direção.

Adotou-se uma política de realimentação constante entre os membros do grupo, onde cada avanço feito no projeto (desenvolvimento, estudo e documentação) era imediatamente comunicado e todos os produtos passados para o outro membro do grupo. Garantiu-se assim a consistência do trabalho e a constante remotivação do grupo ao longo de todo o ano.

## *4.2. Especificação técnico-funcional*

### 4.2.1. Especificação dos dados do cartão

Os dados encapsulados no cartão serão aqueles encontrados nos seguintes documentos oficiais:

- Carteira de identidade (RG)
- Cadastro de pessoas físicas (CPF)
- Carteira Nacional de Habilitação (CNH)
- Título Eleitoral
- Passaporte
- Certificado de Reservista
- Dados médicos de urgência

| **Dados comuns a todos os documentos** | | |
|---|---|---|
| **Código** | **Campo** | **Descrição** |
| 0xA0 | Nome | Até 60 caracteres. Não editável. |
| 0xA1 | Data de nascimento | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. Não editável. |
| 0xA2 | Filiação paterna | Até 60 caracteres. Não editável. |
| 0xA3 | Filiação materna | Até 60 caracteres. Não editável. |
| 0xB0 | Foto (opcional) | - |
| 0xA4 | Assinatura eletrônica pessoal (opcional) | Não editável. |
| ***Carteira de Identidade*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | Registro Geral | 9 dígitos (começando com zeros a esquerda se necessário) seguidos de um caractere alfanumérico |
| 0x01 | Data de expedição | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x02 | Naturalidade | Até 21 caracteres, sendo os dois últimos a sigla do estado. |
| 0x03 | Documento de origem | Até 84 caracteres |
| 0x04 | CPF | 11 dígitos (começando com zeros a esquerda se necessário) |
| ***Cadastro de Pessoas Físicas*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | N° de inscrição | 11 dígitos (começando com zeros a esquerda se necessário) |
| 0x01 | Data de emissão | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| ***Carteira Nacional de Habilitação*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | Documento de identidade | 8 dígitos (começando com zeros a esquerda se necessário) |
| 0x01 | Órgão emissor do documento de identidade | 5 caracteres |
| 0x02 | Categoria de habilitação | 1 caractere |
| 0x03 | Data de validade | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x04 | CPF | 11 dígitos (começando com zeros a esquerda se necessário) |

| | | |
|---|---|---|
| 0x05 | N° de registro | 11 dígitos (começando com zeros a esquerda se necessário) |
| 0x06 | Data de emissão | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x07 | Data da 1ª habilitação | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x08 | Observação 1 | Até 42 caracteres |
| 0x09 | Observação 2 | Até 42 caracteres |
| 0x0A | Estado de emissão | 2 caracteres |
| ***Título Eleitoral*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | N° de inscrição | 13 dígitos (começando com zeros a esquerda se necessário) |
| 0x01 | Zona | 3 dígitos (começando com zeros a esquerda se necessário) |
| 0x02 | Seção | 4 dígitos (começando com zeros a esquerda se necessário) |
| 0x03 | Município | 22 caracteres |
| 0x04 | UF | 2 caracteres |
| 0x05 | Data de emissão | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| ***Passaporte*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | Número | 8 caracteres, onde os dois primeiros são letras e os demais são dígitos |
| 0x01 | Sexo | 1 caractere (M para masculino ou F para feminino) |
| 0x02 | Local de nascimento | Até 36 caracteres |
| 0x03 | Repartição expedidora | Até 90 caracteres |
| 0x04 | Data de validade | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x05 | Data de emissão | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| ***Certificado de Reservista*** | | |
| **Código** | **Campo** | **Descrição** |
| 0x00 | CSM | 2 dígitos |
| 0x01 | RA | 12 dígitos |
| 0x02 | Naturalidade | Até 36 caracteres |
| 0x03 | Data da dispensa do serviço militar inicial | 8 dígitos no formato DDMMAAAA, onde D é o dia, M o mês e A o ano. |
| 0x04 | Motivo | Até 56 caracteres |
| ***Dados médicos de urgência*** | | |

| Código | Campo | Descrição |
|---|---|---|
| 0x00 | Tipo Sanguíneo | Até 3 caracteres |
| 0x01 | Doador de órgãos | 1 caractere (S para doador e N para não doador) |
| 0x02 | Doença de base 1 | Até 26 caracteres |
| 0x03 | Doença de base 2 | Até 26 caracteres |
| 0x04 | Doença de base 3 | Até 26 caracteres |
| 0x05 | Alergia a medicamento 1 | Até 26 caracteres |
| 0x06 | Alergia a medicamento 2 | Até 26 caracteres |
| 0x07 | Alergia a medicamento 3 | Até 26 caracteres |
| 0x08 | Medicamentos de uso contínuo 1 | Até 26 caracteres |
| 0x09 | Medicamentos de uso contínuo 2 | Até 26 caracteres |
| 0x0A | Medicamentos de uso contínuo 3 | Até 26 caracteres |

**Tabela 3 - Descrição dos dados encapsulados no cartão**

### 4.2.2. Funções das *Applets*

Os 7 conjuntos de dados serão encapsulados em 7 *applets* diferentes. Os dados comuns podem ser acessados por todas estas *applets*.

As funções desempenhadas pela Identidade Eletrônica para cada um dos conjuntos de dados descritos em 4.2.1 são:

- **Verificação do código PIN (exceto dados médicos de urgência)**: o código PIN deve ter sido verificado antes de qualquer instrução poder ser executada. Caso o código PIN não tenha sido digitado, as instruções devem retornar um erro.
- **Leitura dos dados**: retorna os dados do documento eletrônico em questão.
- **Gravação dos dados**: esta instrução é usada quando da criação de um novo cartão ou de um novo documento eletrônico (uma nova *applet*). Ela deve verificar se os campos (variáveis) onde serão escritos os dados têm valor `NULL`. Se este não for o caso, nada deve ser gravado e um erro deve ser retornado.

- **Atualização de dados**: grava um dado em uma variável por cima do valor existente anteriormente.

### 4.2.3. Funções da aplicação na Estação de Trabalho

A aplicação na estação de trabalho deve permitir:

- a **criação** de um documento eletrônico gravando os dados no cartão;
- a **leitura** de um documento eletrônico;
- a **atualização** dos dados de um documento;
- a **remoção** de um documento

### 4.2.4. AIDs – Application Identifier

Em Java Card, cada applet é identificada por um AID, assim como cada package. Um AID é constituído por 5 bytes que identificam a companhia que desenvolveu a aplicação (RID – *Registered Application Provider*), seguidos de 0 a 11 bytes que identificam a aplicação em si (PIX – *Proprietary application identifier extension*). O RID é atribuído pela ISO, então para este projeto usa-se um RID fictício.

A tabela a seguir resume os diferentes AIDs utilizados.

| **Package AID** | | |
|---|---|---|
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03 | 2 bytes |
| **Applet RG AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x01 | 3 bytes |
| **Applet CPF AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |

| PIX | 0x00, 0x03, 0x02 | 3 bytes |
|---|---|---|
| **Applet CNH AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x03 | 3 bytes |
| **Applet TituloEleitoral AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x04 | 3 bytes |
| **Applet Passaporte AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x05 | 3 bytes |
| **Applet CertificadoReservista AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x06 | 3 bytes |
| **Applet DadosMedicosUrgencia AID** | | |
| **Campo** | **Valor** | **Comprimento** |
| RID | 0xFF, 0x12, 0x34, 0x56, 0x78 | 5 bytes |
| PIX | 0x00, 0x03, 0x07 | 3 bytes |

**Tabela 4 - Definição dos AIDs utilizados pelos applets**

### 4.2.5. Comandos aceitos pelos applets

A tabela a seguir descreve o formato dos comandos e respostas APDU que podem ser enviados para os applets. Cada applet suporta um comando `SELECT`, que o seleciona para execução, e um conjunto de comandos associados às funções desempenhadas pelo applet.

#### 4.2.5.1. Applet RG

| **Comando SELECT** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x01 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet CPF. | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | **Status word** | **Significado da status word** | | | | |
| nenhum | 0x9000 | Processamento bem-sucedido | | | | |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. | | | | |
| | | | | | | |
| **Comando VERIFY** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |
| Campo de dados contém o código PIN. | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | **Status word** | **Significado da status word** | | | | |
| nenhum | 0x9000 | Processamento bem-sucedido | | | | |
| | 0x6399 | PIN bloqueado | | | | |
| | 0x6300 | Falha na verificação do PIN | | | | |
| | | | | | | |
| **Comando READ** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |
| | | | | | | |
| ***Resposta APDU*** | | | | | | |

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |

| **Comando WRITE** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |
| Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão | | | | | | |

| ***Resposta APDU*** | | |
|---|---|---|
| **Dados opcionais** | **Status word** | **Significado da status word** |
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 5 - Definição dos comandos APDU da applet RG**

#### 4.2.5.2. Applet CPF

| **Comando SELECT** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x02 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet CPF. | | | | | | |
| ***Resposta APDU*** | | | | | | |

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. |

**Comando VERIFY**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |

Campo de dados contém o código PIN.

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6399 | PIN bloqueado |
| | 0x6300 | Falha na verificação do PIN |

**Comando READ**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |

**Comando WRITE**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |

| | | |
|---|---|---|
| Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão | | |
| ***Resposta APDU*** | | |
| **Dados opcionais** | **Status word** | **Significado da status word** |
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 6 - Definição dos comandos APDU da applet CPF**

### 4.2.5.3. Applet CNH

| | | | | | | |
|---|---|---|---|---|---|---|
| **Comando SELECT** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x03 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet CNH. | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | **Status word** | **Significado da status word** | | | | |
| nenhum | 0x9000 | Processamento bem-sucedido | | | | |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. | | | | |
| | | | | | | |
| **Comando VERIFY** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |
| Campo de dados contém o código PIN. | | | | | | |
| ***Resposta APDU*** | | | | | | |

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6399 | PIN bloqueado |
| | 0x6300 | Falha na verificação do PIN |

**Comando READ**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |

**Comando WRITE**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |

Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 7 - Definição dos comandos APDU da applet CNH**

### 4.2.5.4. Applet TituloEleitoral

**Comando SELECT**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x04 | N/A |

O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.
O campo de dados contém o AID da applet TituloEleitoral.

***Resposta APDU***

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. |

**Comando VERIFY**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |

Campo de dados contém o código PIN.

***Resposta APDU***

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6399 | PIN bloqueado |
| | 0x6300 | Falha na verificação do PIN |

**Comando READ**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |

***Resposta APDU***

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | | |
| **Comando WRITE** | | |
| *Comando APDU* | | |

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |

| |
|---|
| Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão |
| |
| *Resposta APDU* |

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 8 - Definição dos comandos APDU da applet TituloEleitoral**

### 4.2.5.5. Applet Passaporte

| **Comando SELECT** | | | | | | |
|---|---|---|---|---|---|---|
| *Comando APDU* | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x05 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet Passaporte. | | | | | | |
| *Resposta APDU* | | | | | | |

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. |

**Comando VERIFY**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |

Campo de dados contém o código PIN.

**Resposta APDU**

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6399 | PIN bloqueado |
| | 0x6300 | Falha na verificação do PIN |

**Comando READ**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |

***Resposta APDU***

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |

**Comando WRITE**

***Comando APDU***

| CLA | INS | P1 | P2 | Lc | Campo de dados | Le |
|---|---|---|---|---|---|---|
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |

| Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão | | |
|---|---|---|
| | | |
| ***Resposta APDU*** | | |
| **Dados opcionais** | **Status word** | **Significado da status word** |
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 9 - Definição dos comandos APDU da applet Passaporte**

### 4.2.5.6. Applet CertificadoReservista

| **Comando SELECT** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x06 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet CertificadoReservista. | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | | **Status word** | | **Significado da status word** | | |
| nenhum | | 0x9000 | | Processamento bem-sucedido | | |
| | | 0x6999 | | Falha na seleção: a applet não existe ou não pode ser selecionada. | | |
| | | | | | | |
| **Comando VERIFY** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |
| Campo de dados contém o código PIN. | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | | **Status word** | | **Significado da status word** | | |
| nenhum | | 0x9000 | | Processamento bem-sucedido | | |
| | | 0x6399 | | PIN bloqueado | | |
| | | 0x6300 | | Falha na verificação do PIN | | |
| | | | | | | |

| **Comando READ** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |
| | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | | **Status word** | | **Significado da status word** | | |
| Dados lidos | | 0x9000 | | Processamento bem-sucedido | | |
| | | 0x6301 | | PIN requerido | | |
| | | | | | | |
| **Comando WRITE** | | | | | | |
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |
| Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão | | | | | | |
| | | | | | | |
| ***Resposta APDU*** | | | | | | |
| **Dados opcionais** | | **Status word** | | **Significado da status word** | | |
| nenhum | | 0x9000 | | Processamento bem-sucedido | | |
| | | 0x6301 | | PIN requerido | | |
| | | 0x6A80 | | Código do dado ou comprimento incorretos | | |

**Tabela 10 - Definição dos comandos APDU da applet CertificadoReservista**

### 4.2.5.7. Applet DadosMedicosUrgencia

| **Comando SELECT** | | | | | | |
|---|---|---|---|---|---|---|
| ***Comando APDU*** | | | | | | |
| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
| 0x00 | 0xA4 | 0x04 | 0x00 | 0x08 | 0xFF, 0x12, 0x34, 0x56, 0x78, 0x00, 0x03, 0x07 | N/A |
| O cabeçalho (CLA, INS, P1 e P2) deve ser codificado dessa maneira, de forma que o JCRE possa identificá-lo como um comando APDU SELECT.<br>O campo de dados contém o AID da applet DadosMedicosUrgencia. | | | | | | |
| ***Resposta APDU*** | | | | | | |

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6999 | Falha na seleção: a applet não existe ou não pode ser selecionada. |

**Comando VERIFY**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x20 | 0x00 | 0x00 | 4 | Código PIN | N/A |

Campo de dados contém o código PIN.

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6399 | PIN bloqueado |
| | 0x6300 | Falha na verificação do PIN |

**Comando READ**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x30 | 0x00 | 0x00 | 1 | Código do dado | Tamanho dos dados |

***Resposta APDU***

| **Dados opcionais** | **Status word** | **Significado da status word** |
|---|---|---|
| Dados lidos | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |

**Comando WRITE**

***Comando APDU***

| **CLA** | **INS** | **P1** | **P2** | **Lc** | **Campo de dados** | **Le** |
|---|---|---|---|---|---|---|
| 0xB0 | 0x40 | Código do dado | 0x00 | Comprimento do campo de dados | string de dados | N/A |

Código do dado a ser escrito ou alterado de acordo com a Tabela 3 - Descrição dos dados encapsulados no cartão

***Resposta APDU***

| Dados opcionais | Status word | Significado da status word |
|---|---|---|
| nenhum | 0x9000 | Processamento bem-sucedido |
| | 0x6301 | PIN requerido |
| | 0x6A80 | Código do dado ou comprimento incorretos |

**Tabela 11 - Definição dos comandos APDU da applet DadosMedicosUrgencia**

### 4.2.6. Aplicação da estação de trabalho

#### 4.2.6.1. Interface gráfica

A interface homem-máquina será feita através de janelas do Windows. As estações de trabalhos terão acesso somente aos documentos necessários a sua aplicação. Somente um documento por vez pode ser manipulado.

#### 4.2.6.2. Criar um documento

A criação do documento só será autorizada **após verificação de que tal documento ainda não existe no cartão**. Então se instancia um objeto relativo à classe do documento correspondente. Em seguida, comandos WRITE serão enviados contendo os dados do documento. Um documento será considerado pronto para uso quando todos os seus campos de dados estiverem preenchidos.

O código PIN do portador será definido quando da criação do documento.

#### 4.2.6.3. Ler um documento

Uma autenticação do portador do cartão através do código PIN deverá preceder a leitura do cartão, salvo para a consulta dos dados médicos de urgência, que poderão ser vistos em qualquer tipo estação de trabalho e sem a necessidade da autenticação do portador.

A leitura do documento consiste na busca dos valores estocados nos membros do objeto instanciado do documento correspondente. O comando utilizado será o READ que retorna todos os dados sob a forma de uma cadeia de caracteres.

#### 4.2.6.4. Atualizar dados de um documento

Não é necessária a verificação do código PIN do portador para esta função. A aplicação deve primeiramente verificar a presença do documento no cartão. A atualização dos dados será feita através da atribuição de novos valores dos membros do objeto relativo ao documento correspondente. O comando de atualização será também o WRITE que será enviado ao cartão, contendo os novos dados.

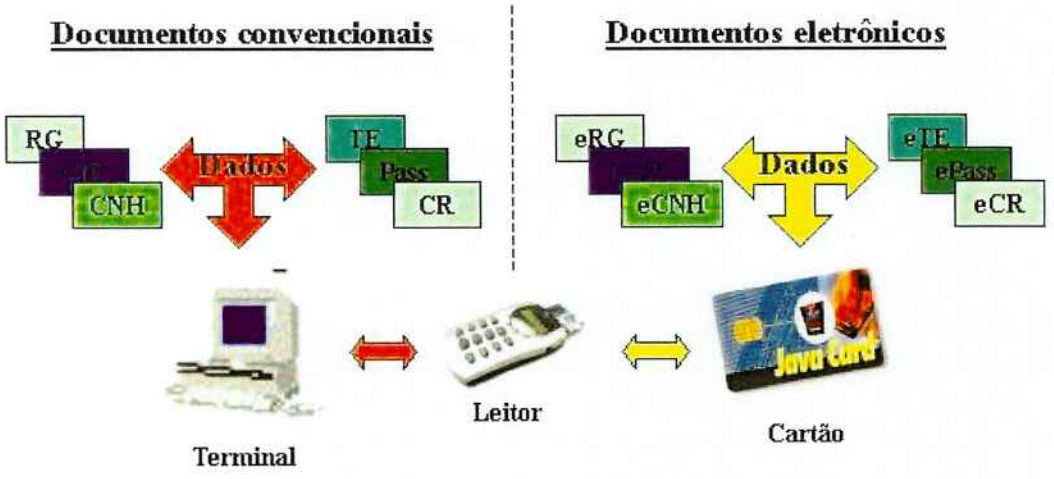
#### 4.2.6.5. Remover um documento

A verificação de acesso de cada terminal será feita pelo cartão buscando as permissões presentes no terminal. Não é necessária a verificação do código PIN para esta função.

## *4.3. Descrição da solução*

### 4.3.1. Arquitetura

A solução é baseada em uma aplicação de terminal que se comunica com um leitor de smartcard, que por sua vez faz a comunicação com o cartão.




### 4.3.2. Cartão

Os documentos modelados possuem um conjunto de dados comuns a todos eles: nome, data de nascimento, filiação paterna e materna. Para garantir a consistência desses

dados e economizar espaço no cartão, esses dados foram separados em um applet. Este applet implementa uma *Shareable Interface* (ver 3.3.3) de forma que ele possa ser "enxergado" e compartilhado pelos demais applets.

Além disso, nos dados compartilhados armazenamos também o código PIN do cartão e os métodos a ele relacionados. Dessa forma todos os applets utilizam o mesmo código e compartilham um método único de verificação, garantido consistência e eliminando redundância na implementação. Na prática, isso significa que uma vez que o PIN é verificado para, digamos, o RG, ele não precisa ser verificado novamente para os outros documentos, a menos é claro que o cartão seja resetado.

A arquitetura do cartão fica então como mostrado na figura a seguir. As partes descritas como não implementadas no projeto são na realidade implementações já presentes nos cartões comerciais.

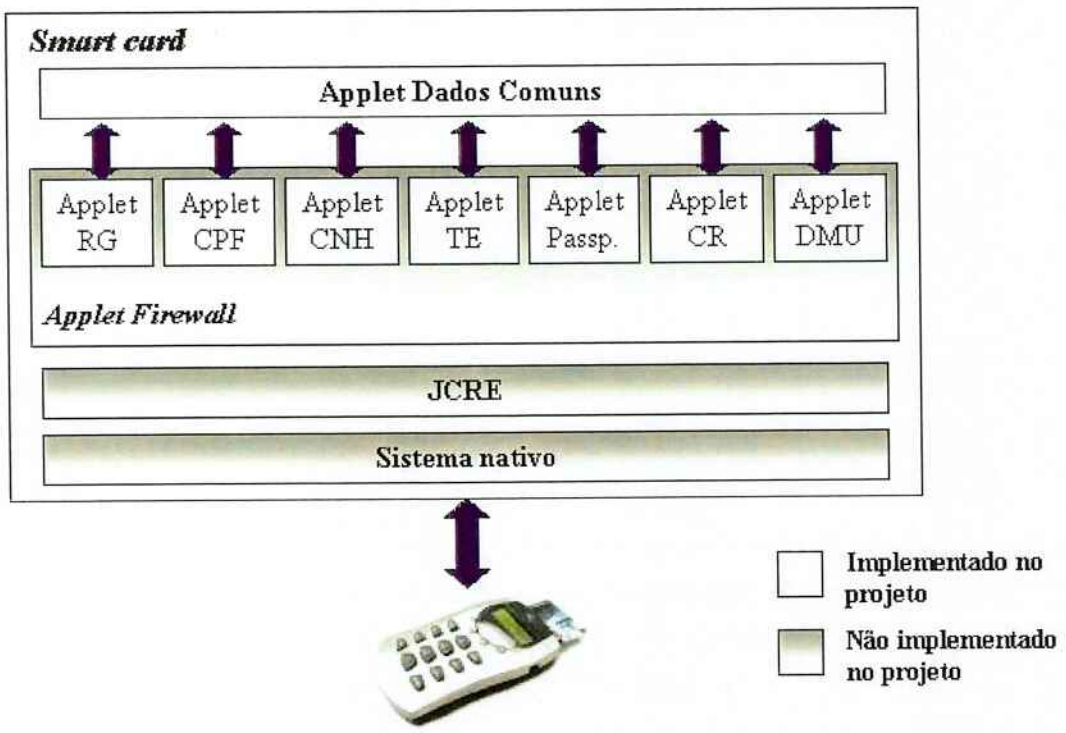



**Figura 8 - Arquitetura do cartão**

Um mecanismo de firewall próprio do Java Card impede que os applets se comuniquem e compartilhem dados. Apenas o applet dos dados comuns, por implementar uma Shareable Interface, tem um nível de segurança diferente.

Criamos assim primeiramente uma interface CommonDataInterface que implementa a interface javacard.framework.Shareable, segundo o diagrama UML a seguir.

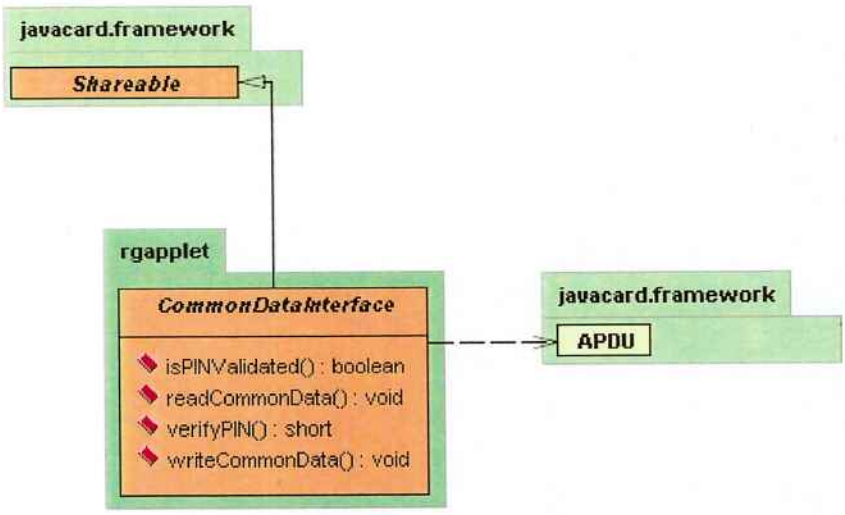



**Figura 9 - Diagrama UML da interface CommonDataInterface**

O Applet dos dados comuns implementa então a CommonDataInterface:

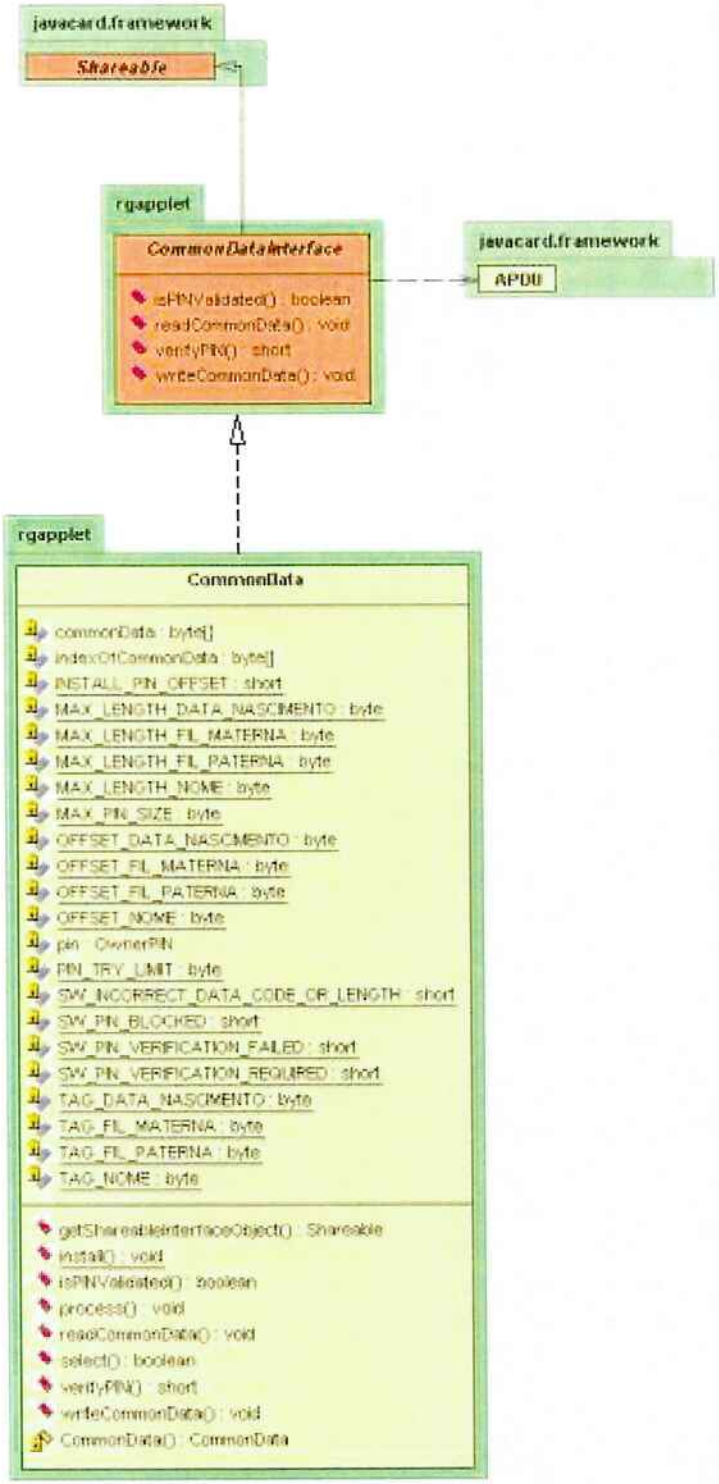



**Figura 10 - Diagrama UML do applet CommonData**

Os diagramas UML dos demais applets são semelhantes. Todos os applets herdam de javacard.framework.Applet, que é uma exigência do Java Card, por isso esta relação foi omitida nos diagramas a seguir.

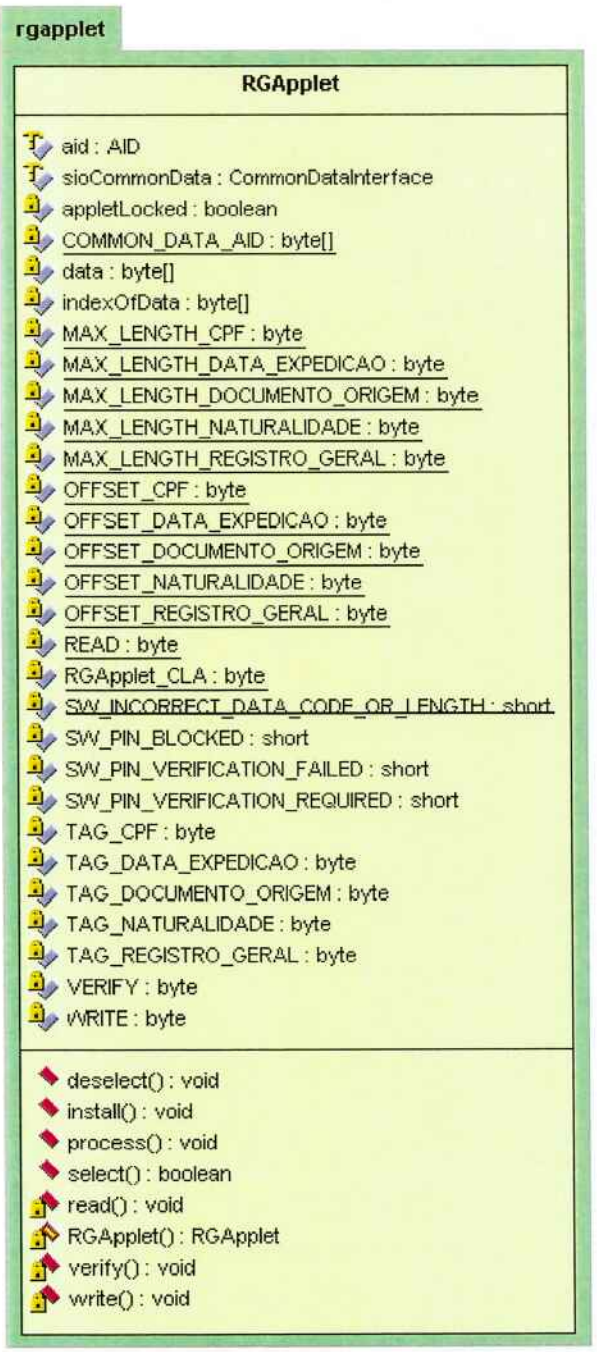



**Figura 11 - Diagrama UML do applet RG**

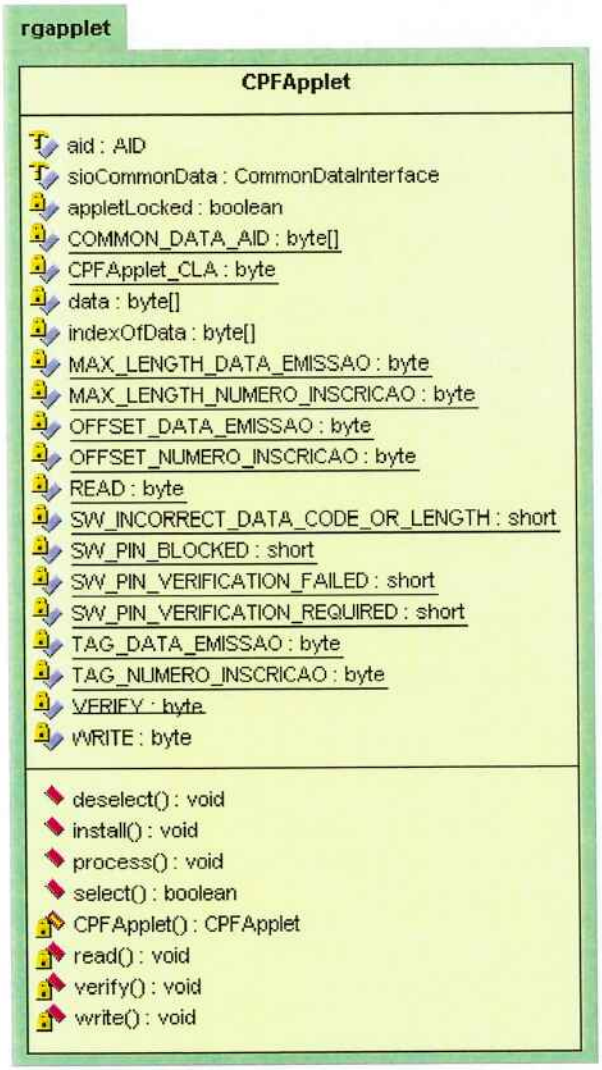



**Figura 12 - Diagrama UML do applet CPF**

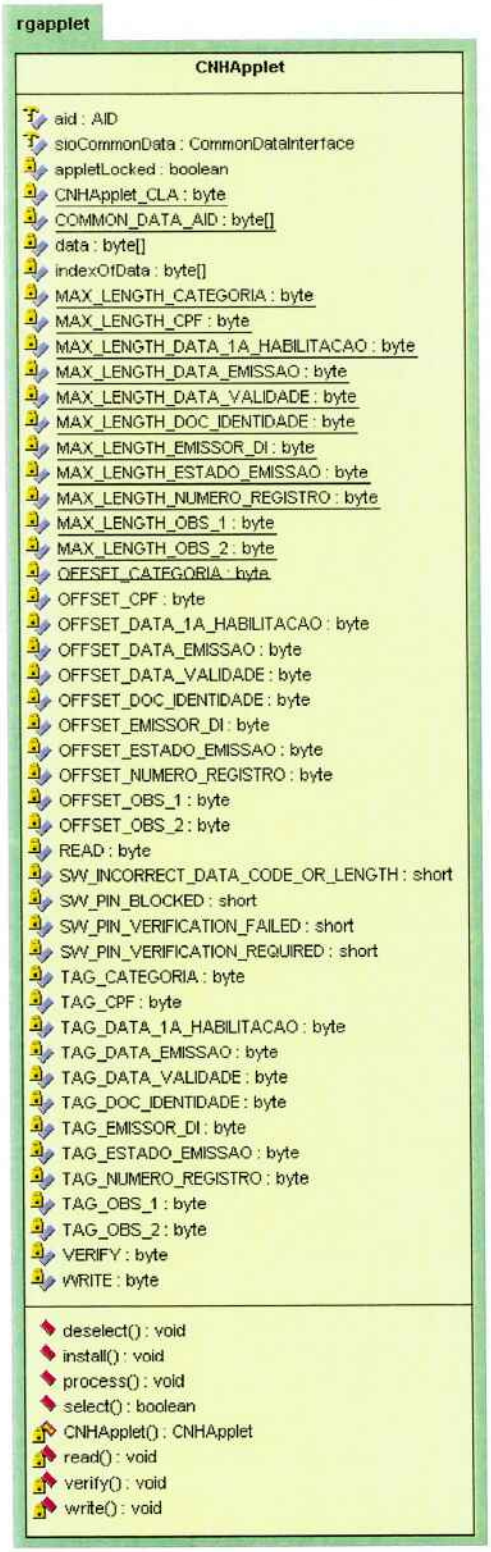



**Figura 13 - Diagrama UML do applet CNH**


**Figura 14 - Diagrama UML do applet TituloEleitor**


**Figura 15 - Diagrama UML do applet Passaporte**


**Figura 16 - Diagrama UML do applet CertificadoReservista**

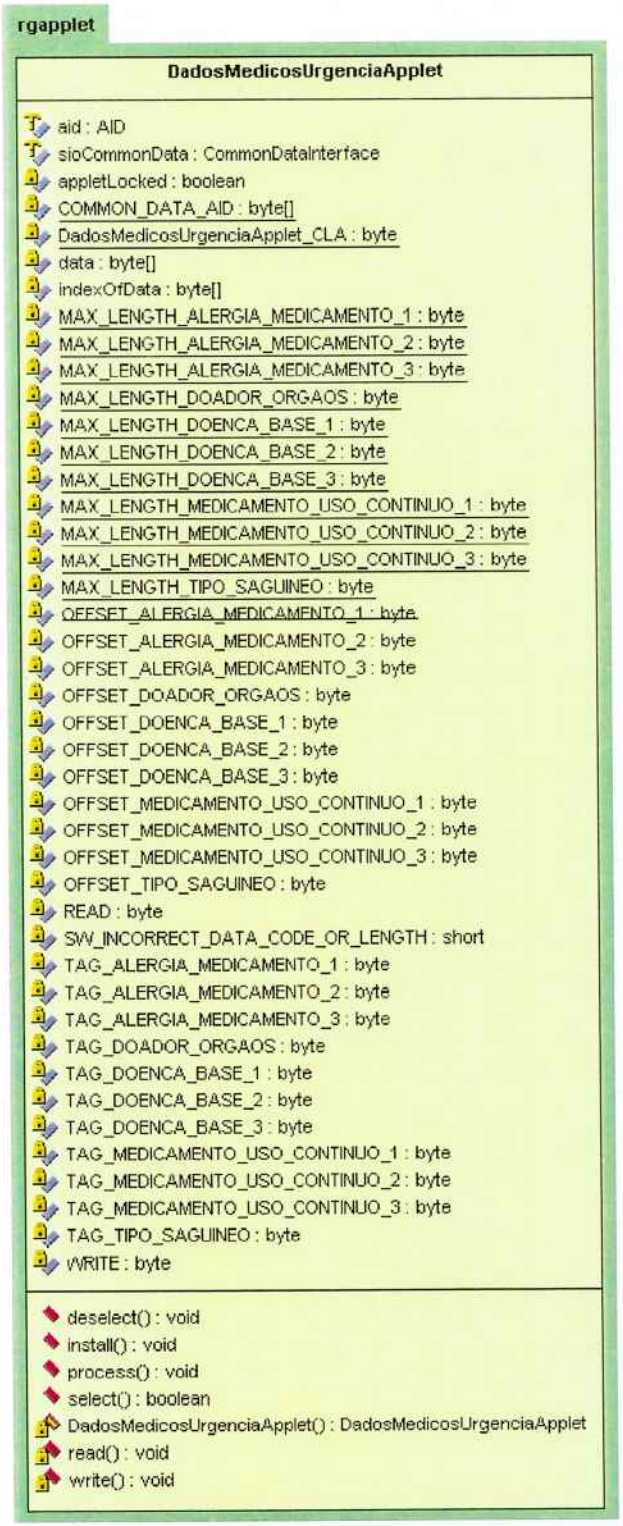



**Figura 17 - Diagrama UML do applet DadosMedicosUrgencia**

Podemos observar que todos os applets possuem os métodos `select()`, `process()` e `install()`. Esses métodos são obrigatórios em qualquer applet Java. O método `install()` é chamado uma única vez pelo Java Card Runtime Environment para criar uma instância do applet e registrá-lo. O método `select()` é chamado pelo JCRE para "informar" o applet de que ele foi selecionado. O método `process()` é o ponto de entrada do applet. Quando uma APDU é enviada ao cartão e o applet está selecionado, o JCRE redireciona esta APDU para o applet, que vai processá-la dentro do `process()`. O trecho de código abaixo mostra o método `process()` do applet RG e ilustra o processamento da APDU recebida pelo applet.

```
public class RGApplet extends Applet {
  //...
  // CLA byte corresponding to this applet
  private static final byte RGApplet_CLA = (byte)0xB0;

  // INS values in the command APDUs
  private static final byte VERIFY = (byte)0x20;
  private static final byte READ = (byte)0x30;
  private static final byte WRITE = (byte)0x40;

  //...

  /**
   * process APDUs
   */
  public void process(APDU apdu) {

    byte[] buffer = apdu.getBuffer();

    // if the command is SELECT, we shall not do anything
    if (selectingApplet())
      return;

    // verify the CLA code
    if (buffer[ISO7816.OFFSET_CLA] != RGApplet_CLA)
      ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);

    switch (buffer[ISO7816.OFFSET_INS]) {
      case VERIFY:       // verify PIN code
        verify(apdu);
        return;

      case READ:         // read data from the applet
        read(apdu);
```

```
        return;

      case WRITE:        // write data
        write(apdu);
        return;

      default:           // else, throw exception
        ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);
    }
  }
  //...
} // end of class RGApplet
```

Observe que neste nível verificamos apenas o campo CLA da APDU e o campo INS para saber de qual comando se trata (ver 4.2). Nas funções `read()`, `write()` e `verify()` verificamos os demais parâmetros do comando e efetuamos as operações em si.

Como foi falado anteriormente, o código PIN e alguns dados comuns a todos os applets são administrados em um só applet, chamado CommonData. Para que os métodos do CommonData possam ser chamados de fora, o applet que faz a chamada deve conhecer o AID do CommonData e "capturar" a sua interface compartilhada. Isso pode ser feito quando o applet é selecionado por exemplo.

```
public class RGApplet extends Applet {
  //...
  // array storing the CommonData applet AID
  private static final byte[] COMMON_DATA_AID =
      {(byte) 0xFF, (byte) 0x12, (byte) 0x34, (byte) 0x56,
      (byte) 0x78, (byte) 0x00, (byte) 0x03, (byte) 0x08};

  // variables that'll store a reference to the CommonData applet
  AID aid;
  CommonDataInterface sioCommonData;
  //...
  public boolean select() {

      //  obtain the CommonData applet AID
      aid = JCSystem.lookupAID(COMMON_DATA_AID, (short) 0,
                        (byte) COMMON_DATA_AID.length);
```

```
        // request its shareable interface object (sio), so that we can
        // use its shared methods
        sioCommonData = (CommonDataInterface) JCSystem
                    .getAppletShareableInterfaceObject(aid, (byte) 0);

        return true;
        }
    //...
} // end of class RGApplet
```

Para chamar um método do applet CommonData a partir do RG, digamos o verifyPIN(), basta chamá-lo como se fosse um método estático do Java tradicional:

```
public class RGApplet extends Applet {
    //...
    private void verify(APDU apdu) {

        short result;

        // call the method
        result = sioCommonData.verifyPIN(apdu);

        //...
    }
    //...
} // end of class RGApplet
```

### 4.3.3. Comunicação leitor-terminal

A comunicação entre o leitor e o terminal se faz através de uma DLL (Dynamic Link Library) fornecida pelo fabricante do leitor. Essa DLL pode ser carregada por qualquer aplicação, que então poderá chamar suas funções para enviar comandos a um cartão inserido no leitor. No entanto, a DLL fornecida não é compatível com Java Native Interface (JNI), que é a maneira como uma aplicação Java pode chamar funções implementadas em uma outra linguagem através de uma DLL. Para que a aplicação em Java interprete corretamente as funções da DLL, estas devem possuir uma "assinatura" (o nome da função e seus argumentos) específica para JNI.

A solução foi então criar uma DLL compatível com JNI que chamasse a DLL original do leitor, inserindo assim uma camada intermediária na comunicação leitor-terminal, como ilustra o diagrama a seguir.

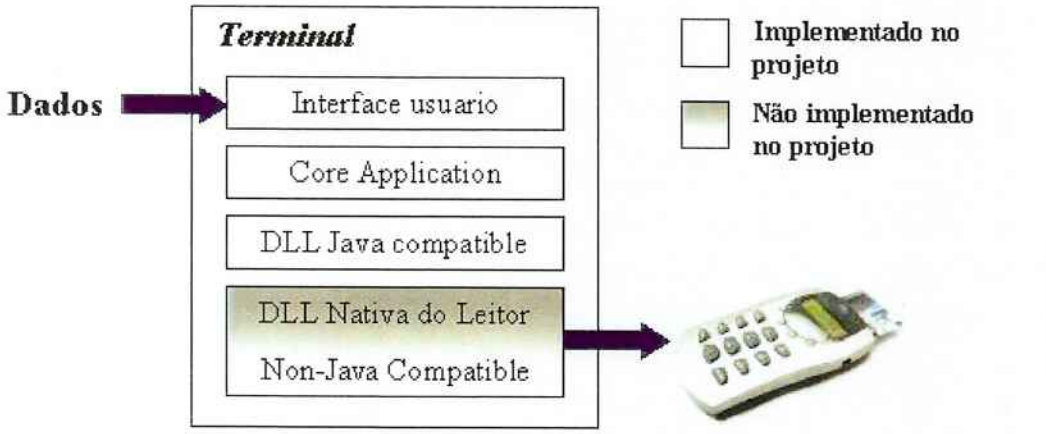



**Figura 18 - Duas camadas na comunicação leitor-terminal**

A técnica adotada consiste em mapear cada função da DLL original com uma função na DLL Java. A aplicação do terminal chama uma função da DLL Java. Esta função traduz os objetos Java passados como argumento para o formato dos argumentos da função correspondente na DLL original (em C++), chama esta função, e traduz as saídas de volta nos objetos Java para que eles possam ser recuperados pela aplicação do terminal.

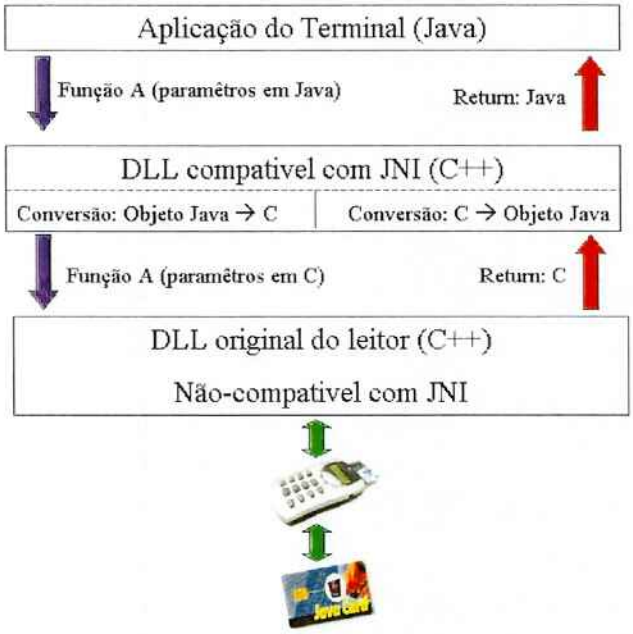



**Figura 19 - Fluxo de dados entre a aplicação e o leitor**

Esta técnica é conhecida como *one-to-one mapping*, e é a técnica de implementação mais direta. Existem outras técnicas que utilizam fortemente ponteiros, mas seu uso só é vantajoso no caso de se ter que implementar um número grande de funções. No nosso caso, apenas algumas funções da DLL original tiveram que ser implementadas, o que nos fez optar pelo one-to-one mapping. A tabela a seguir resume as funções implementadas e o que elas fazem.

| javaSlbReader.dll - **Função** | **Descrição** |
|---|---|
| Java_smartcardterminal_CardManager_APIAllocate | Aloca um leitor em uma dada porta |
| Java_smartcardterminal_CardManager_APIFree | Libera o leitor e a porta correspondente |
| Java_smartcardterminal_CardManager_APIPowerUp | Liga o leitor |
| Java_smartcardterminal_CardManager_APIPowerDown | Desliga o leitor |
| Java_smartcardterminal_CardManager_APIReset | Envia um cold reset ao cartão |
| Java_smartcardterminal_CardManager_APISendIsoInT0 | Envia um comando com dados para o cartão |

| | |
|---|---|
| Java_smartcardterminal_CardManager_APISendIsoOutT0 | Envia um comando que pega dados do cartão |
| Java_smartcardterminal_CardManager_APIAllocateComManager | Aloca o gerenciador de portas do computador |
| Java_smartcardterminal_CardManager_APIFreeComManager | Libera o gerenciador de portas do computador |
| Java_smartcardterminal_CardManager_APIGetNbrReaders | Retorna o número de leitores conectados ao computador |
| Java_smartcardterminal_CardManager_APIGetAPIGetReaderName | Retorna o nome do leitor |
| Java_smartcardterminal_CardManager_APIGetNbrFreePorts | Retorna o número de portas disponíveis no computador |

**Tabela 12 - Funções da DLL do leitor de smart cards**

### 4.3.4. Terminal

A aplicação desenvolvida para o terminal tem como objetivo a gerência dos dados inseridos nos cartões de identidade eletrônica. Esta aplicação é destinada a órgãos do governo que serão os responsáveis pela inserção, atualização e remoção dos documentos dentro de cada cartão, eles também poderão certamente utilizar o serviço de consulta dos dados. Além da gerência dos documentos, estes órgãos serão responsáveis pelo tratamento da senha do cartão (o PIN, Personal Identification Number). Os serviços disponíveis para o PIN é a sua inserção e a sua atualização, no caso em que a senha foi bloqueada.

O PIN tem uma grande importância na segurança dos dados do portador da identidade. Todos os documento são passiveis de consulta, atualização, inserção e remoção mediante a verificação do PIN. A única exceção é a consulta aos dados médicos de urgência. Este não necessita da verificação do portador para ser lido, pois em caso de urgência quando o portador estiver inconsciente, este deverá ser habilitado para leitura sem o conhecimento da senha.


**Figura 20 - Tela de entrada do aplicativo do terminal**

### 4.3.4.1. Gerenciamento do PIN

A inserção do PIN deve ser a primeira operação feita num cartão virgem, ou seja, um cartão que ainda não foi personalizado. Esta operação é obrigatória para que o cartão seja habilitado e aceite a inserção de documentos. Ele deve conter quatro números na sua combinação. Como podemos ver na figura abaixo, nenhum botão na opção de inserção de documentos está habilitado, somente o botão de criação do PIN.


**Figura 21 - Tela de inserção da senha do cartão**

Uma vez o PIN criado, os botões para inserir os documentos serão habilitados e o botão de inserção do PIN desaparecerá.

O PIN deve ser verificado antes de qualquer ação sobre qualquer documento dentro do cartão, com exceção da consulta dos dados médicos de urgência. Uma vez o PIN verificado, a sessão do PIN é aberta e não pede outras verificações do PIN até que esta seja fechada. Esta sessão se fecha quando o aplicativo do terminal é fechado, quando o cartão é retirado ou quando o leitor é desligado. Veja abaixo a tela de verificação do PIN:


**Figura 22 - Tela de verificação da senha do cartão**

No caso em que o portador digitou três vezes a senha do cartão erradamente, esta senha será bloqueada e o cartão não responderá mais a nenhum serviço. O serviço para desbloquear o PIN é usado para esta circunstância. O portador poderá entrar com um novo PIN mesmo que ele não se lembre do PIN anterior.

#### 4.3.4.2. Serviço de inserção de documentos

Após a criação da senha (PIN) para o cartão, todos os documentos estarão habilitados para inserção.

Para todos os documentos existem campos de dados obrigatórios a serem preenchidos e outros não dependendo de sua natureza. Exemplo: o número de registro do CPF é um dado não obrigatório no documento de RG. No caso em que um campo obrigatório não foi preenchido, uma tela indicará o erro no momento da criação do cartão, escolhendo o botão voltar, todos os dados antes preenchidos ainda estarão nos campos de texto, evitando assim que o usuário deva preenchê-los novamente.

Existem documento que possuem informações sobre outros documentos, este documentos são o RG e a CNH. Ambos possuem informações do CPF e a CNH possui informações sobre o RG.

Quando criamos um documento onde estão figuradas informações de outros documentos, primeiramente os documentos referentes a estas informações são procurados, se eles estão presentes no cartão, então seus dados serão colocados no campo de texto referentes a eles, serão desabilitados para escrita e inseridos desta maneira no novo documento, já que eles vêm dos seus documentos originais. Se a pesquisa não encontrar o documento referido no cartão, então o campo de texto estará habilitado para escrita e os dados colocados serão transportados para o cartão no momento da criação documento.

Os dados comuns aos documentos (nome, filiação e data de nascimento), já mencionado anteriormente, também serão tratados como vindos de outros documentos. Estes dados serão lidos e colocados nas caixas de texto que estarão desabilitadas a escrita. A modificação dos dados comuns somente poderá ser feita através do serviço de atualização de documentos. Veja abaixo a tela de inserção da CNH onde o RG já havia sido criado, mas o CPF ainda não:

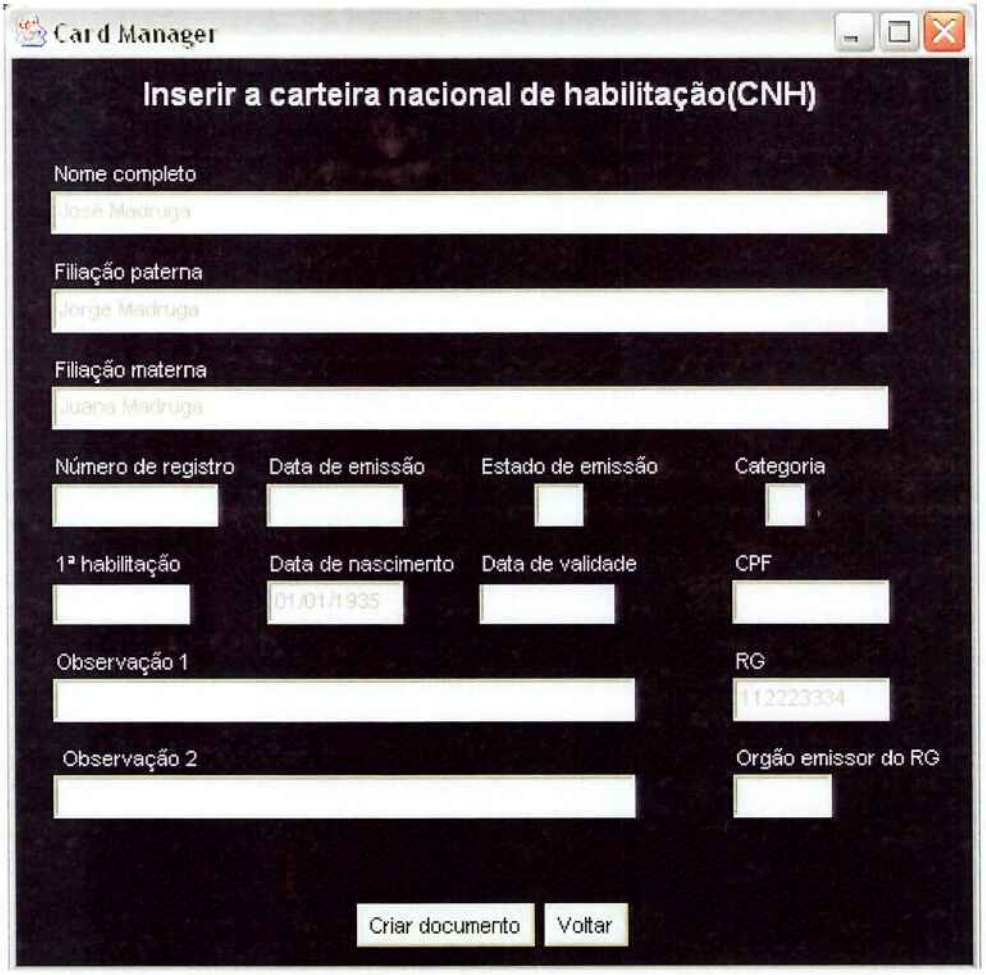



**Figura 23 - Tela de inserção da CNH**

A figura acima exemplifica bem o uso de dados referentes a outros documentos. A tela de inserção mostra que os dados comuns já foram inseridos através de outro documento e que o RG também já está presente, pois os seus dados estão na sua caixa de texto. Como visto anteriormente, as caixas de texto estão desabilitadas a escrita. Já as informações referentes ao CPF não constam na tela acima, isso indica que este documento ainda não foi criado e que poderemos inserir um número provisório para ele na CNH até que o ele seja inserido e passe a ser a fonte desta informação.

### 4.3.4.3. Serviço de atualização de documentos

Este serviço é importante para os casos onde os dados foram inseridos no cartão de forma incorreta ou para caso de alteração de datas de validade, etc. Ele funciona de forma semelhante ao serviço de inserção de documentos. No entanto, ele somente pode ser acessado para os documentos que já foram inseridos no cartão. Este serviço possibilita a atualização de todos os dados contidos no documento, com exceção dos dados provenientes de outros documentos inseridos no cartão. Este funcionalidade pode inclusive alterar os dados comuns aos documentos. Obviamente, as alterações destes dados repercutirão sobre todos os outros, já que todos compartilham estes mesmos dados. Veja na figura abaixo os campos de texto dos dados comuns habilitados para a escrita enquanto o campo de texto do CPF esta desabilitados, pois este já foi criado no cartão.

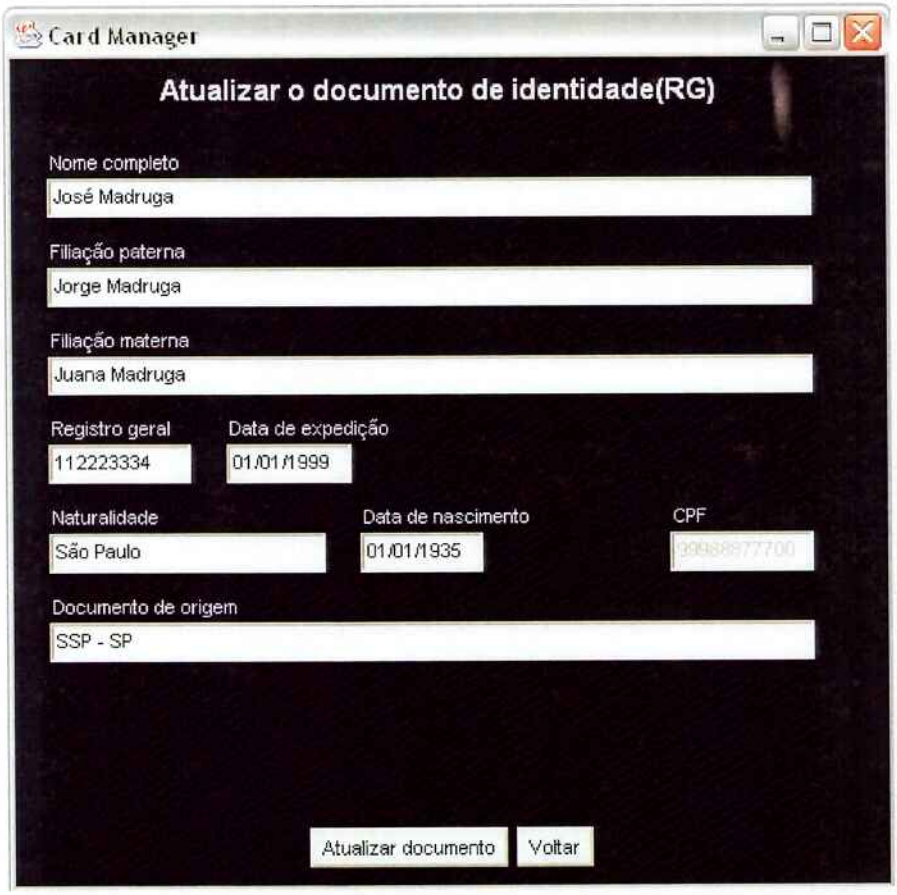



**Figura 24 - Tela de atualização do RG**

#### 4.3.4.4. Serviço de remoção de documentos

Este serviço tem como finalidade remover os documentos do cartão. Isso pode acontecer caso o documento não tenha mais utilidade ao portador. Um exemplo desta funcionalidade seria a remoção da CNH caso o portador seja impossibilitado permanentemente de conduzir um veículo.

Na remoção do documento, o applet relacionado com o documento é desinstalado do cartão. Os dados não poderão ser recuperados posteriormente, com exceção dos dados comuns. Os dados comuns ainda serão utilizados pelo documento restantes no cartão. Mesmo que todos os documentos sejam retirados do cartão, os dados referentes ao applet de dados comum continuarão no cartão. Por isso podemos dizer que o cartão foi personalizado. Esta é uma operação definitiva, logo a reutilização do cartão por vários portadores não pode ser feita.

Antes da remoção completa do documento, uma pergunta de confirmação é feita ao usuário para que este não apague documentos erradamente, veja a figura abaixo:

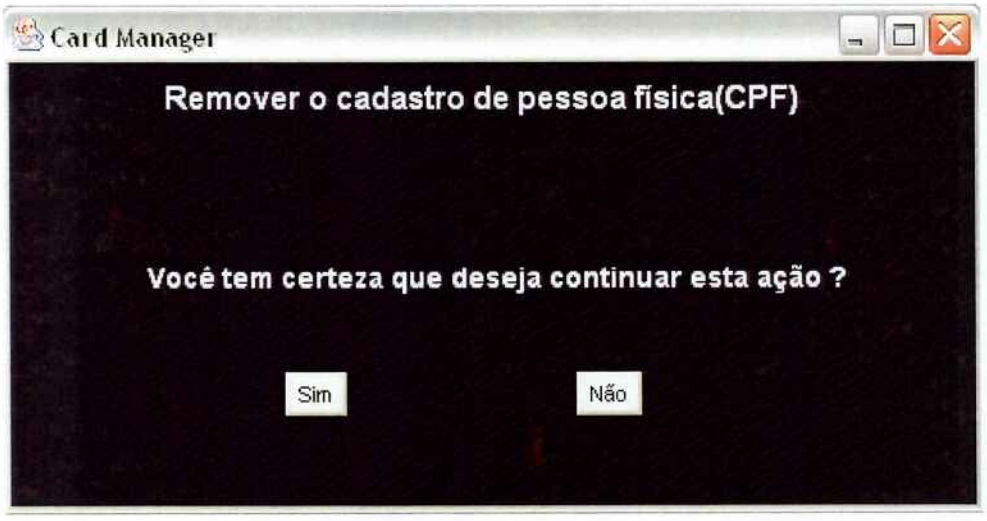



**Figura 25 - Tela de remoção do CPF**

#### 4.3.4.5. Serviço de consulta de documentos

Este serviço é destinado à consulta dos documentos já existentes no cartão pelo usuário. Como dito anteriormente, por questões de urgência médica, os dados referentes aos dados médicos da pessoa podem ser acessados sem o conhecimento da senha do cartão. Os campos de texto dos dados da tela de consulta estão desabilitados a escrita, no entanto eles podem ser selecionados e copiados para alguma outra utilização.

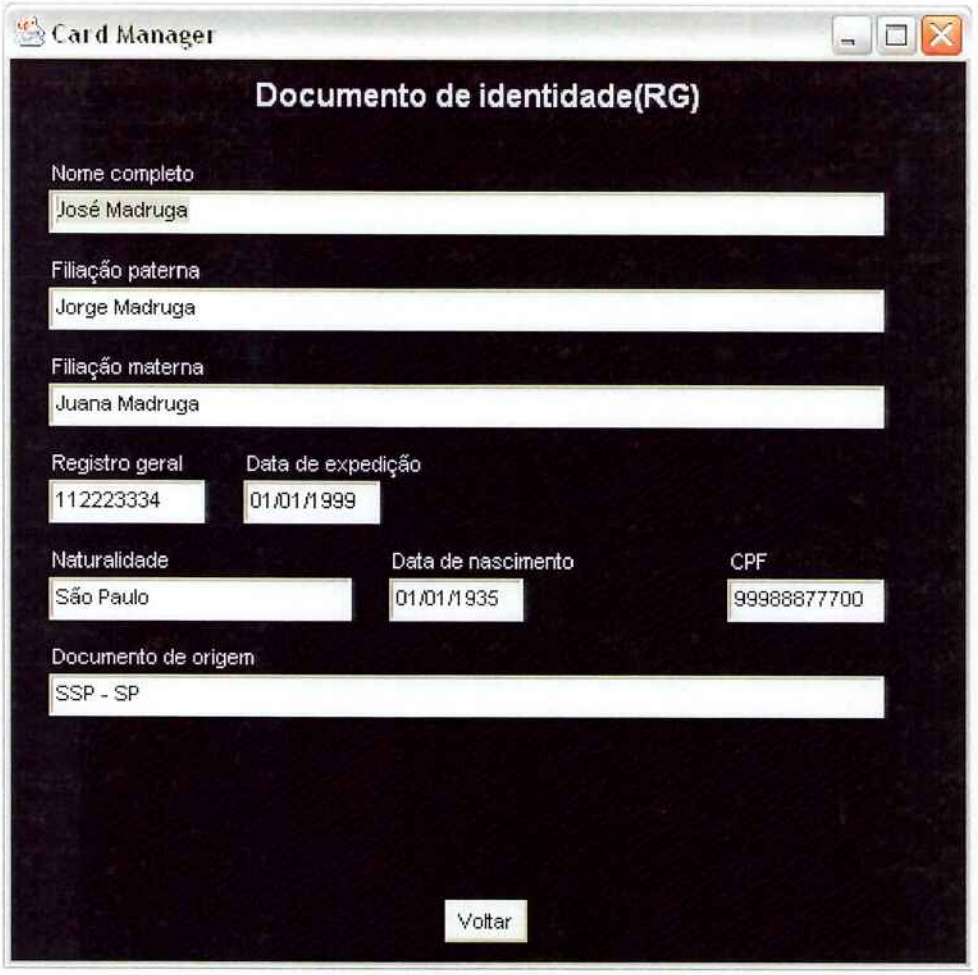



**Figura 26 - Tela de consulta do RG**

#### 4.3.4.6. Detecção de hardware

A falta de um leitor ou do cartão inserido no cartão leva o aplicativo a uma tela de erro, onde ele identifica o hardware que esta faltando e pede a sua instalação ou verificação de funcionamento. Enquanto o problema não for solucionado, a tela de erro continuará mostrando a mesma mensagem, onde o usuário tem a opção de sair do aplicativo ou instalar o hardware e continuar coma sua navegação.

Outro ponto importante na detecção de hardware é a sessão do PIN. Uma vez que PIN foi verificado com sucesso, uma sessão do PIN se abre e isto indica ao aplicativo que não é necessário verificar o PIN para alguma outra operação. No entanto, quando o cartão é retirado e recolocado, ou mesmo quando o cartão é substituído por outro cartão com o aplicativo aberto e a sua sessão aberta, a sessão do cartão deve ser finalizada a fim de evitar que operações sejam feitas num novo cartão, sendo que foi para algum outro que a verificação do PIN havia sido feita.

#### 4.3.4.7. Diagramas de implementação

Existem três classes principais na implementação da aplicação. A principal é a classe *CardManager* (figura 25) onde figuram os métodos e atributos de gerência do cartão. Estes métodos são chamados pela classe *Frame1*, que é a classe que gera a interface do aplicativo, para ligar ações sobre o cartão com botões, caixas de texto e outros objetos da interface. Outra importante classe se chama *APDU* que é utilizada para o gerenciamento das APDU, as mensagem trocadas entre o cartão e o leitor.

Dois tipos abstratos também foram implementados: *CByte* e *CLong*. Eles foram implementados pois as DLLs nativas utilizam esses tipo como parâmetros de entrada e saída. A classe *SizedTextField* é um classe derivada da classe *TextField*. Ela foi implementada para adicionar uma funcionalidade a classe mãe essencial para a nossa aplicação. A funcionalidade implementada limita o número de caracteres que o usuário do aplicativo pode escrever numa caixa de texto. Esta característica é essencial, pois o

cartão trabalha com tamanho máximos de cadeias de caracteres. Se não quisermos perder o que foi escrito a mais, esta limitação deve ser apresentada ao usuário.

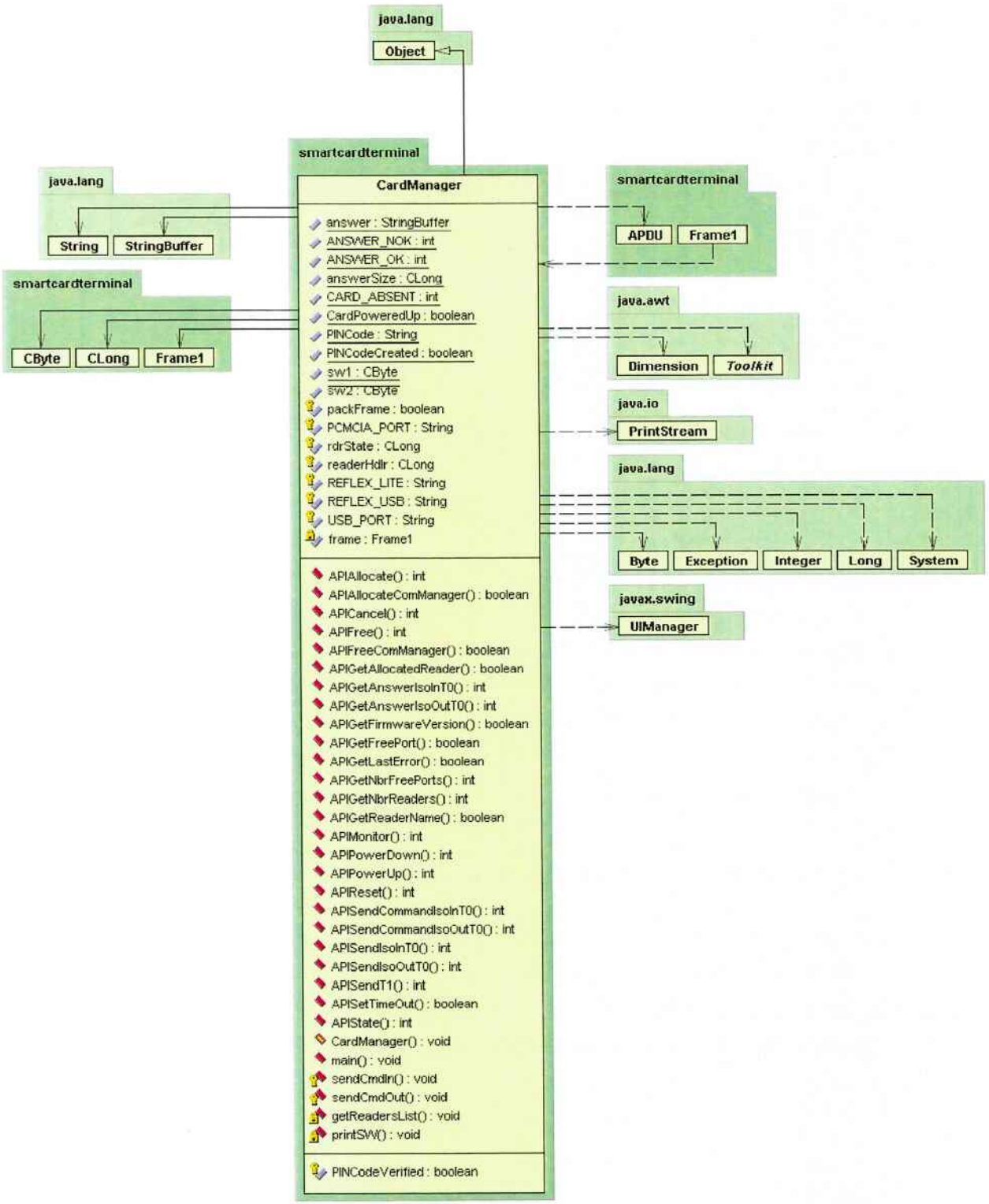



**Figura 27 - Diagrama UML da classe CardManager**

# 5. CONCLUSÕES

## 5.1. Problemas encontrados

No geral, o projeto transcorreu como previsto. O maior fator de risco identificado no início do projeto era a obtenção dos equipamentos necessários, leitor e cartões. Mas esse problema foi rapidamente resolvido. No entanto, dois problemas do ponto de vista técnico surgiram em diferentes momentos, exigindo uma carga extra não prevista no projeto.

O primeiro surgiu no início da implementação dos applets. Como fazer com que os applets compartilhem dados de maneira segura e consistente? Embora se conhecesse a solução do ponto de vista teórico (uso de Shareable Interfaces), sua implementação foi bastante custosa e exigiu um esforço maior que o previsto.

A segunda grande dificuldade apareceu no fim da implementação, e foi o fato de a biblioteca do leitor de smartcards não ser compatível com Java. Evidentemente não haveria tempo para se recomeçar a aplicação do terminal em outra linguagem. Houve assim a necessidade de se encontrar uma solução para fazer a comunicação entre a aplicação do terminal em Java e o leitor. Isso resultou em esforço extra para se estudar e implementar uma solução em Java Native Interface, o que evidentemente pôs em risco o cronograma previsto nos últimos dois meses do projeto.

## 5.2. Resultados obtidos

O produto final do projeto é uma aplicação que administra o conteúdo de um smart card que encapsula 6 documentos oficiais e mais um conjunto de dados médicos de urgência.

Além disso, o presente documento contém um estudo sobre a tecnologia de smart cards, com ênfase na tecnologia Java Card, e também um panorama das aplicações possíveis

destas tecnologias e dos projetos existentes a nível mundial na área de governo eletrônico.

## 5.3. Considerações sobre o aprendizado com o projeto

Foi possível construir uma sólida base de conhecimento sobre smart cards desde a sua arquitetura interna até todas as possíveis aplicações da tecnologia. O estudo efetuado permitiu obter uma visão global dos diversos projetos existentes no mundo utilizando smart cards nas suas diversas aplicações, como governo eletrônico, saúde eletrônica, telefonia, transportes, etc.

Tecnicamente, implementou-se de ponta a ponta um projeto em Java Card, o que envolveu aprendizado tanto do lado da programação como do lado da especificação, já que este não é um projeto que se encaixa nos moldes tradicionais de levantamento de requisitos. Também foram extendidos e consolidados conhecimentos em programação Java sobretudo com relação à elaboração de interfaces gráficas. Ainda do ponto de vista técnico, deve-se também citar o aprendizado sobre o uso de DLLs e sobre Java Native Interface, que é uma ferramenta poderosa para a interoperabilidade de aplicações escritas em diversas linguagens de programação.

Do ponto de vista da gestão de projeto, foi importante administrar um projeto de tão longa duração, com questões difíceis de se lidar como a distribuição do trabalho ao longo do projeto e principalmente a avaliação da carga horária a se gastar em cada atividade sem comprometer o projeto na sua reta final com uma sobrecarga de trabalho.

## 5.4. Viabilidade técnica da solução proposta

A coexistência de diversas aplicações em um mesmo cartão não resulta em problemas de segurança nem de interoperabilidade. O framework Java Card garante que o programador tem o controle da comunicação segura entre os diversos applets. No nosso caso, mostrou-se que é possível que diversos documentos estejam presentes no mesmo cartão compartilhando alguns dados comuns e somente estes dados. Em outras palavras,

o acesso a um documento específico não permite que se burle a segurança para se ter acesso a outro documento.

Para que a solução proposta possa vir a se tornar um produto um dia, é necessário se incluir um módulo de criptografia (que pode ser um módulo em Java Card ou em C, nesse caso presente no sistema operacional do cartão) e se definir melhor o escopo de segurança (quando pode-se atualizar um dado, validação de chaves de acesso, etc.). A coexistência de applets no cartão é perfeitamente viável, tanto que ela é fortemente usada hoje nos chips GSM, onde diversos serviços sob forma de applets co-habitam o cartão sem problemas.

Um empecilho à solução proposta no entanto é o aspecto visual do cartão. No caso de uma identidade eletrônica sozinha no cartão, é possível se replicar os dados de forma visual no corpo do cartão de maneira que se possa utilizar o documento mesmo na ausência de um terminal que leia os dados. No caso de vários documentos, por uma questão elementar de espaço, é possível somente replicar os dados principais de cada documento, não sendo assim viável, na maioria das situações, seu uso caso não se leia os dados eletrônicos.

## 5.5. Possíveis seqüências do projeto

Conforme dito acima, a primeira evolução necessária é a implementação de um **módulo de criptografia** no cartão, possibilitando assim a encriptação dos dados e a geração de **assinaturas eletrônicas**. Ainda com relação à segurança, seria necessário uma definição do **framework de segurança** do cartão: quando e sob quais condições pode-se atualizar (modificar) os dados no cartão, que tipo de chaves e níveis de acesso aos dados pode-se ter em diversos casos.

A seguir, a inclusão de **foto** e de **dados biométricos** (impressão digital e/ou íris) parece ser uma tendência para as versões futuras das identidades eletrônicas (ver 2.5).

O terceiro ponto é sem dúvida a implementação de **serviços de governo eletrônico** que façam uso dos documentos eletrônicos. No Anexo 1 sugerimos algumas possíveis aplicações.

# LISTA DE REFERÊNCIAS

**Livros, artigos e referências Java Card**

[1] CHEN, Z. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide*. San Francisco: Addison-Wesley, 2000, 368p.

[2] RANKL, W.; EFFING, W. *Smart Card Handbook*. New York: John Wiley & Sons, 1997.

[3] Java Card website. Disponível em: <http://java.sun.com/products/javacard/>

[4] Java Card Forum. Disponível em: <http://www.javacardforum.org>

[5] Java World. Portal da linguagem Java, com vários artigos. Disponível em: <http://www.javaworld.com>

[6] DI GIORGIO, R. *Smart Cards: A Primer – Develop on the Java platform of the future*. Java World, dez. 1997. Disponível em: <http://www.javaworld.com>. Acesso em: 17 mar. 2004.

[7] DI GIORGIO, R. *Get a jumpstart on the Java Card: How to utilize Java on your wallet or purse*. Java World, fev. 1998. Disponível em: <http://www.javaworld.com>. Acesso em: 17 mar. 2004.

[8] CHEN, Z.; DI GIORGIO, R. *Understanding Java Card 2.0: Learn the inner workings of the Java Card architecture, API, and runtime environment*. Java World, mar. 1998. Disponível em: <http://www.javaworld.com>. Acesso em: 17 mar. 2004.

**Normas ISO para cartões a microprocessador**

[9] International Organization for Standards. Disponível em: <http://www.iso.ch>

[10] ISO/IEC 7816-1, Identification cards - Integrated circuit(s) cards with contacts - Part 1 : Physical characteristics.

[11] ISO/IEC 7816-2, Identification cards - Integrated circuit(s) cards with contacts - Part 2 : Dimensions and location of the contacts.

[12] ISO/IEC 7816-3, Identification cards - Integrated circuit(s) cards with contacts - Part 3 : Electronic signals and transmission protocols.

[13] ISO/IEC FCD 7816-4: 2003 (Draft) Identification cards - Integrated circuit(s) cards with contacts, Part 4: Interindustry commands for interchange, Working draft dated 2003-01-17, ISO SC17 Document 17N2268T

[14] ISO/IEC FCD 7816-5 : 2003, Identification cards - Integrated circuit(s) cards with contacts - Part 5 : Registration for application providers.

[15] ISO/IEC FCD 7816-6 : 2003 (Draft), Identification cards - Integrated circuit(s) cards with contacts - Part 6 : Interindustry data elements for interchange – FCD dated 2003-01-17, ISO SC17 Document 17N2270T.

**Fabricantes de cartões a microprocessador**

[16] Site da Axalto Inc. Disponível em: <http://www.axalto.com>

[17] Site da Gemplus S.A. Disponível em: <http://www.gemplus.com>

[18] Site da Oberthur Card Systems. Disponível em: <http://www.oberthur.com>

**Projetos envolvendo smart cards**

[19] Site do Moneo, moeda eletrônica. Disponível em: <http://www.moneo.net>

[20] Site da Carte Bleue, cartão bancário a microprocessador francês. Disponível em: <http://www.carte-bleue.com>

[21] VR Alimentação e Smart VR. Disponível em: <http://www.vr.com.br>

[22] Programa Smart Club. Disponível em: <http://www.smartclub.com.br>

[23] Projeto Netcards. Disponível em: <http://www.netcards-project.com>

[24] Site do Cartão Europeu de Seguro Saúde (European Health Insurance Card). Disponível em: <http://europa.eu.int/comm/employment_social/healthcard/index_en.htm>

[25] Site do GIE Sesam-Vitale. Sistema francês de seguro saúde eletrônico. Disponível em: <http://www.sesam-vitale.fr>

[26] Site da Certisign. Disponível em: <http://www.certisign.com.br>

**Projetos na area de governo eletrônico**

[27] Portal Federal da Bélgica (Portail Fédéral de Belgique). Informações sobre a carteira de identidade eletrônica belga. Disponível em: <http://www.belgium.be/eportal>

[28] Agência para o Desenvolvimento da Administração Eletrônica da França (Agence pour le Développement de l'Administration Electronique – ADAE). Disponível em: <http://www.adae.gouv.fr>

[29] Autoridade para a Informatização na Administração Pública da Itália (Autoritá per l'Informatica nella Pubblica Amministrazione - AIPA). Disponível em: <http://www.aipa.it>

[30] GENTILI, M. *Italian Electronic Identity card – principle and architecture.* In: Very Large Data Bases Conference, Roma, 2001. Roma: Autorità per l'Informatica nella Pubblica Ammistrazione.

[31] Centro de Registro da População da Finlândia (Population Register Centre). Informações sobre a carteira de identidade finlandesa e downloads das especificações técnicas. Disponível em: <http://www.fineid.fi>

[32] Hong Kong Smart Identity Card. Disponível em: <http://www.immd.gov.hk/ehtml/hkid_hkid.htm>

# ANEXO 1 – EXEMPLOS DE APLICAÇÕES USANDO IDENTIDADE ELETRÔNICA

## 1. IMPRESSÃO DIGITAL, IMPRESSÃO VOCAL E SENHA

As três maneiras de identificação do portador da identidade eletrônica são feitas através da impressão digital, impressão vocal (reconhecimento por voz) ou senha do cartão, cada uma com finalidades distintas.

A senha do cartão é usada em situações que não requerem alta segurança na autenticação do portador e para utilização do cartão com o consentimento do portador. A senha deve ser armazenada no cartão, pois pode ser usada em situações offline. Exemplo: consulta à SERASA, consulta de dados dos documentos.

A impressão vocal, que utiliza o reconhecimento da voz do portador, é usada em situações que requerem alta segurança na autenticação do portador do documento, mas ainda requer o consentimento do portador, para operações online. Exemplo: Desbloquear a senha do cartão, quando o portador ultrapassou a limite de tentativas da senha.

A impressão digital deve ser usada em situações em que o portador do documento não precisa consentir para que a sua identificação seja feita, mas ele deve ser identificado de qualquer maneira. Neste caso os dados sobre a impressão digital do portador deve estar contido no cartão, pois pode ser usada em operações offline. Exemplo: Policial identificando um criminoso.

Outras maneiras de identificação que poderiam ser implementadas são o reconhecimento facial e o reconhecimento da íris do portador. Estas duas poderiam substituir o reconhecimento por impressão digital. No entanto, deve se tomar cuidado no reconhecimento facial com as imprecisões devidas às mudanças na pessoa, como

uma barba, por exemplo. A vantagem do reconhecimento da íris é a alta confiabilidade da resposta, no entanto requer tecnologias mais sofisticadas.

| Autenticação | Segurança | Consentimento | Offline |
|---|---|---|---|
| Senha | Baixa | Sim | Sim |
| Voz | Alta | Sim | Não |
| Digital | Alta | Não | Sim |
| Facial | Alta | Não | Depende |
| Íris | Alta | Não | Depende |

## 2. VERIFICAÇÃO DE PASSAPORTE

A verificação manual dos passaportes e vistos nas fronteiras é um processo lento e sujeito a falhas. O registro feito manualmente de cada pessoa que entra ou sai de um país causa enormes filas. O uso de um passaporte eletrônico teria assim algumas vantagens com relação ao método tradicional:

a. Agilização da verificação do passaporte e do visto, que seria emitido em formato eletrônico e gravado no cartão, e emissão de carimbo eletrônico na cartão;
b. Redução das fraudes por falsificação de passaporte graças à autenticação do portador através de informações biométricas (impressão digital e íris) gravadas no cartão.
c. Registro das pessoas que entram e saem do país em banco de dados sem a necessidade de digitação manual dos dados;

O grande ganho em segurança seria a identificação biométrica. Informações sobre a impressão digital, a íris ou até mesmo a voz da pessoa podem ser estocadas na cartão e verificadas com as amostras obtidas na hora. Dessa forma a falsificação do passaporte torna-se virtualmente impossível.

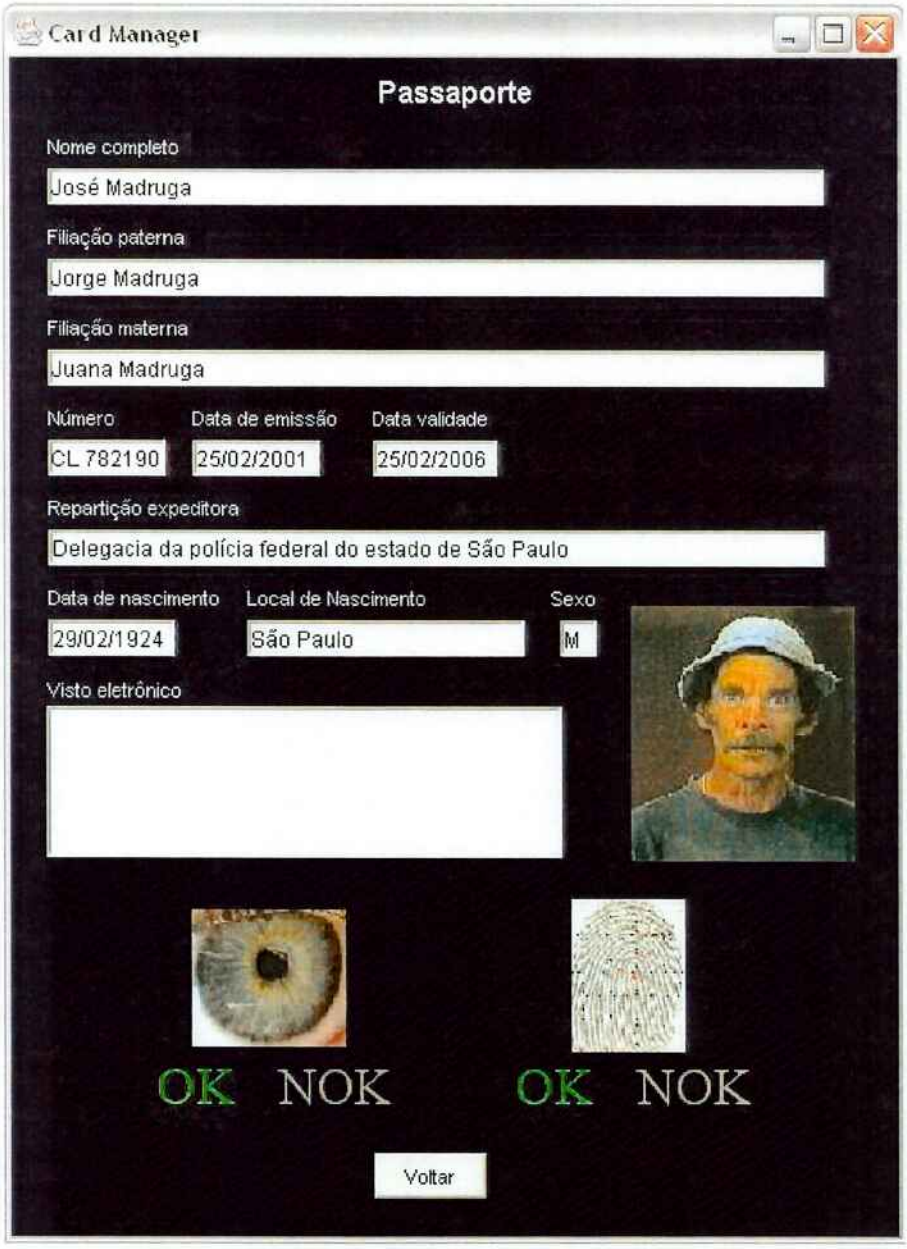



**Figura 28 - Exemplo da aplicação de visualizacao do passaporte com autenticação por impressão digital e íris**

# 3. CONSULTA À SERASA

Uma informação muito importante para lojas e bancos é situação atual das pendências do seu cliente através do CPF. Hoje em dia, as lojas e bancos utilizam o número do

CPF do cliente para consultar o banco de dados da SERASA, onde estão os dados da situação do CPF do cliente atualizados diariamente. Desta maneira, os bancos podem liberar empréstimos a seus clientes e imprimir talões de cheques ou também as lojas podem vender a prazo para estes.

A identificação eletrônica usando Smart card seria uma alternativa para esta busca. As lojas e bancos poderiam buscar a situação de inadimplência do cliente e também a sua foto para que seja confirmado que o portador daquele CPF é realmente o cliente a sua frente.

Funcionamento do sistema:

1. Funcionário do banco ou loja abre o aplicativo, que fica a espera da identidade eletrônica do cliente no seu leitor.
2. Cliente põe sua identidade eletrônica no leitor.
3. Aplicativo verifica se o documento foi inserido ou se o seu numero de registro foi inserido no documento do RG ou CNH.
4. Se sim, o cliente deve digitar a senha no teclado numérico.
5. Se não, o busca deve ser feita de maneira convencional, ou seja, documento em papel.
6. Se a senha é correta, o aplicativo busca as informações do CPF no banco de dados da SERASA, busca a foto do portador no banco de dados da polícia federal, mostra se o CPF esta livre de pendências, além da foto do portador.
7. Se a senha é incorreta, o portador tem três chances de acertar a senha antes que esta seja bloqueada.

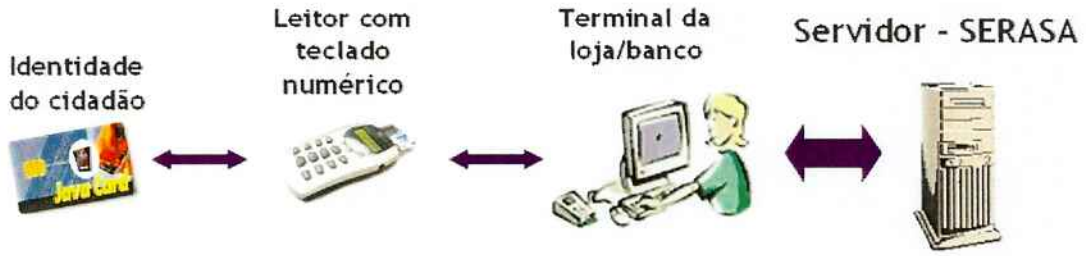



**Figura 29 - Consulta à SERASA**

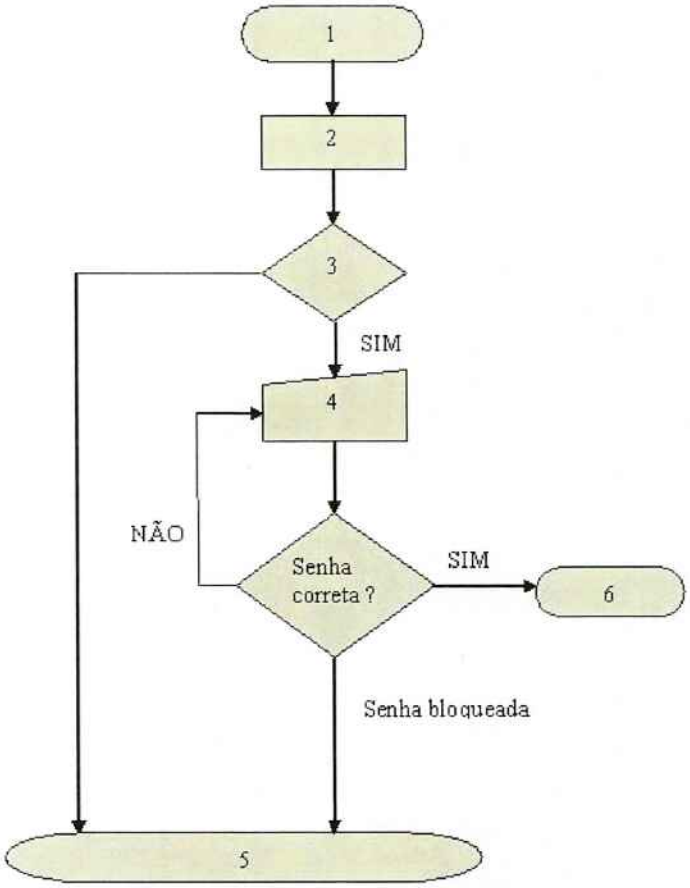



**Figura 30 - Fluxograma de consulta à SERASA**

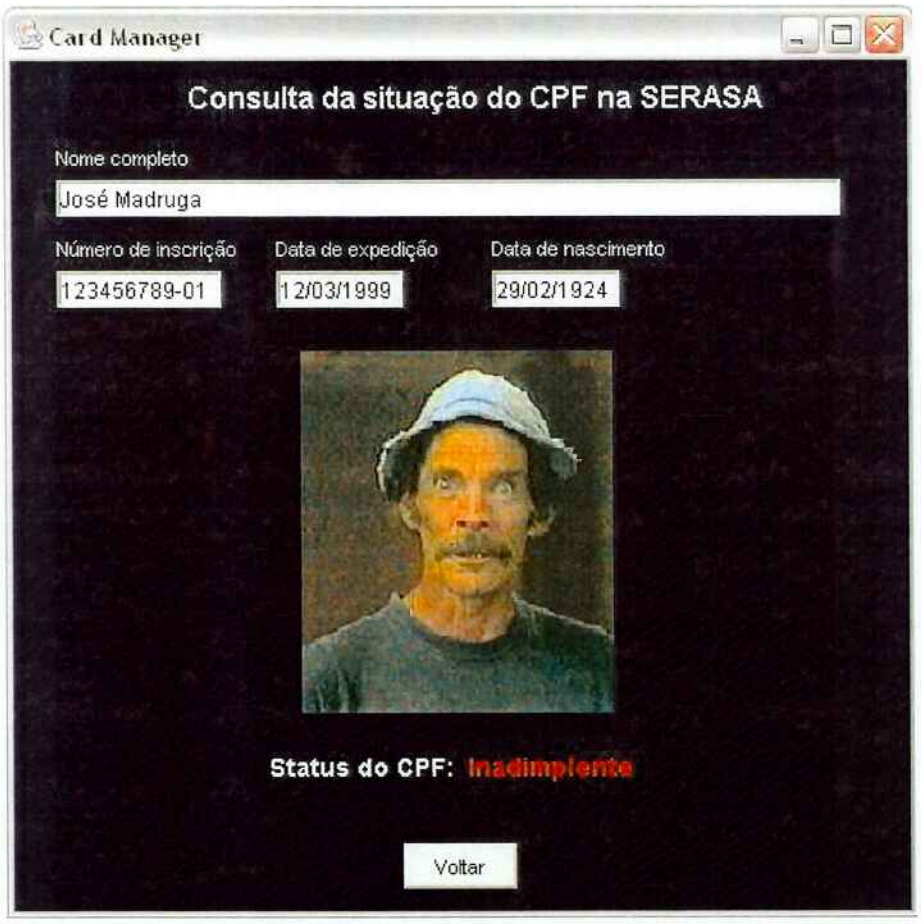



**Figura 31 - Exemplo de aplicação da consulta à Serasa**

# 4. POLICIAMENTO RODOVIÁRIO

Hoje em dia, o policiamento rodoviário esta bastante limitado principalmente pela falta de recursos de informação. No caso em que um motorista é abordado por um policial, este tem a possibilidade de verificar poucas informações sem a ajuda de uma central de polícia. Ele pode verificar as datas de validade dos documentos do motorista e o seu estado de embriaguez. O policial deve contatar uma central de polícia, para que ele possa obter informações de sobre o carro ou o motorista condizente com a sua situação criminal.

A identidade eletrônica utilizada juntamente com outros recursos tecnológicos pode facilitar e agilizar estes processos. Um aplicativo carregado num Palm Top poderia averiguar os dados contidos no cartão de identidade, buscar a foto do motorista, sua situação criminal e o número de pontos de multa na sua carteira de habilitação no banco de dados da polícia. No caso em que o policial aplica uma multa, se a quantidade de pontos na CNH do motorista ultrapassar o valor limite, o policial terá o poder de anular a CNH do motorista além de atualizar o banco de dados de multas, agilizando o processo de entrega de multas.

Funcionamento do sistema:

1. O policial abre o aplicativo de leitura da identidade digital no Palm top.
2. O policial conecta a identidade eletrônica no leitor de smart cards do Palm top.
3. O aplicativo busca as informações do da situação criminal do motorista e a sua foto no banco de dados da polícia federal, busca o número de pontos na CNH no banco de dados do DETRAN e imprime na tela todas as informações acima.
4. No caso de aplicação de multa, a multa seria enviada ao banco de dados do DETRAN e aplicativo alteraria o número de pontos na carteira do motorista e anula-la-ia se cabível.
5. O policial deve tomar as medidas necessárias com base nas informações apresentadas.


**Figura 32 - Consulta dos pontos na carteira de habilitação**

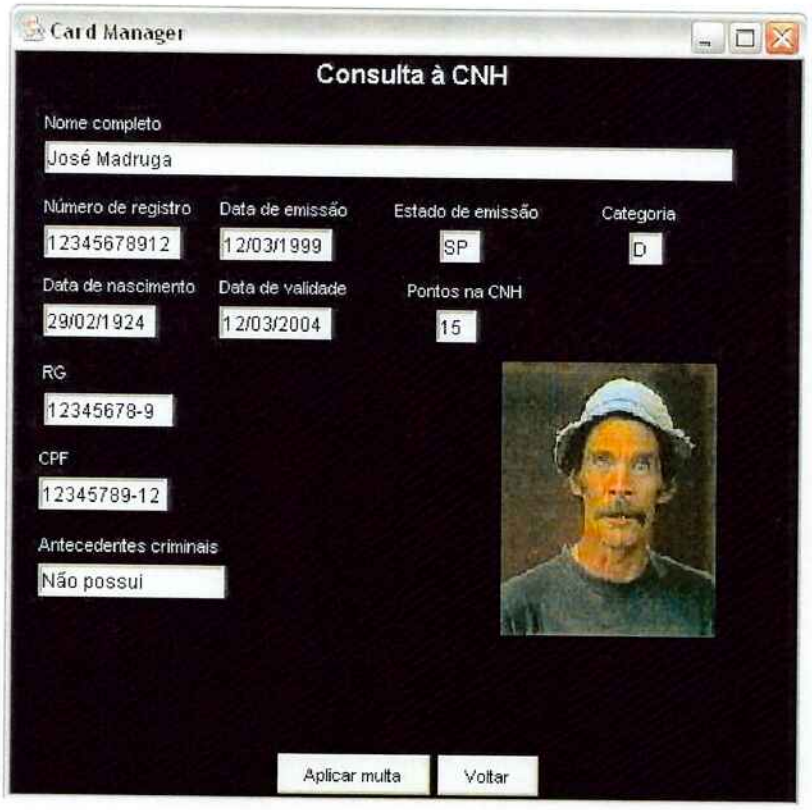



**Figura 33 - Aplicação de consulta à CNH**

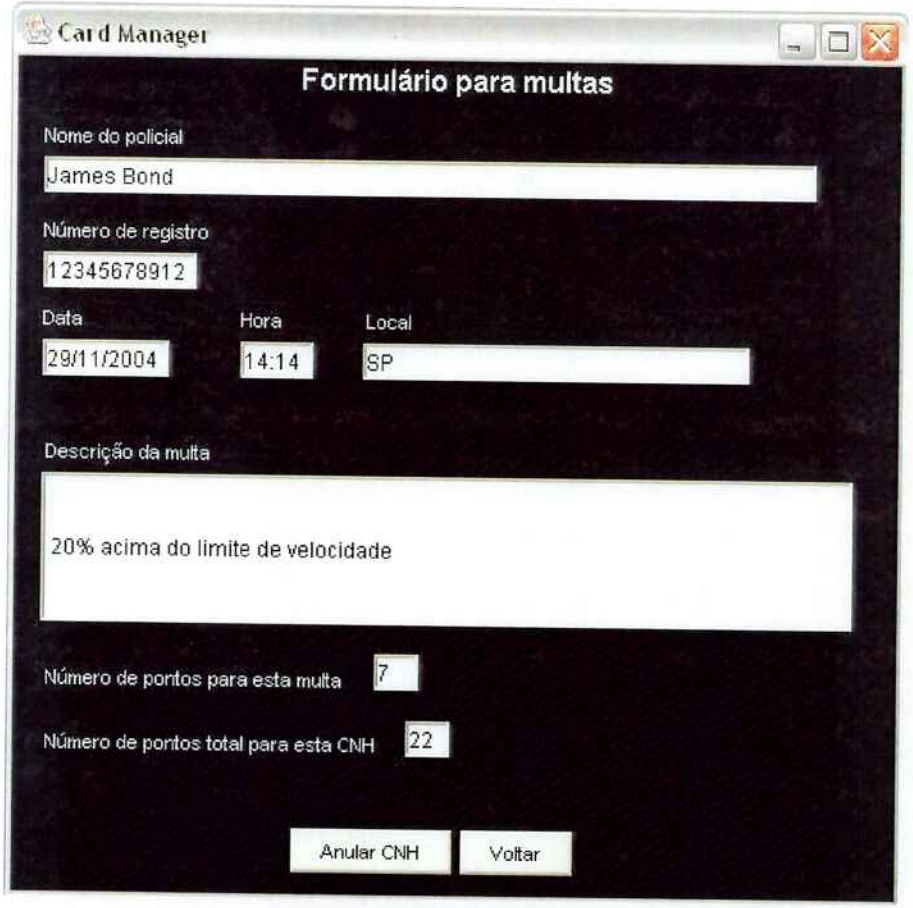



**Figura 34 - Atribuição de multa via eletrônica**

# 5. AUTOMATIZAÇÃO DAS ELEIÇÕES

Hoje, apesar do avanço que representa a urna eletrônica, todo o processo de verificação do título de eleitor, assinatura do registro de votação e emissão do comprovante permanece manual. O uso do título de eleitor eletrônico permite que todo o processo seja automatizado. A figura a seguir ilustra como seria o novo processo.

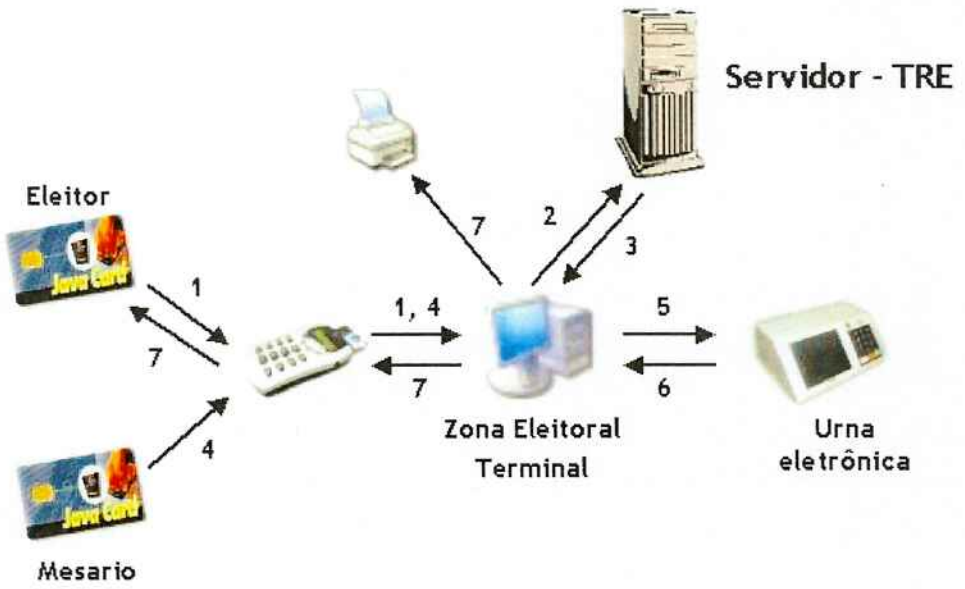



**Figura 35 - Eleições eletrônicas**

1. Eleitor insere seu título na leitora e os dados são transferidos para um terminal na zona eleitoral;
2. O terminal envia os dados para o servidor do Tribunal Regional Eleitoral para autenticação;
3. O TRE confirma que o título é válido;
4. O mesário insere seu cartão na leitora e é autenticado pelo terminal;
5. O terminal então libera a urna eletrônica;
6. Urna envia comando de fim de votação;
7. Terminal emite comprovante de votação, que é gravado no título de eleitor e eventualmente impresso no local.

Elimina-se com esse processo toda a verificação manual dos títulos, o que acarreta uma maior segurança na eleição. Esse novo processo seria um passo intermediário para uma futura eliminação da figura do mesário e até mesmo para a viabilização de votação pela internet, já que toda a segurança necessária à autenticação do título está embutida no cartão. Nesse caso, é necessária a autenticação também do portador para evitar que uma

pessoa vote com o título de outra. Isso poderia ser feito através da inclusão de dados biométricos no cartão, como por exemplo impressão digital e íris.

# ANEXO 2 – CRONOGRAMA