CENTRO UNIVERSITÁRIO DE BRASÍLIA
UNICEUB

CONTROLE DE PONTO UTILIZANDO A TECNOLOGIA JAVA CARD

ALUNO: GIOVANNI FERREIRA DE SOUSA

Orientador: Profª. M.C. Maria Marony Sousa F. Nascimento

Brasília - DF, dezembro de 2009.

GIOVANNI FERREIRA DE SOUSA

CONTROLE DE PONTO UTILIZANDO A TECNOLOGIA JAVA CARD

Trabalho apresentado à Banca examinadora do curso de Engenharia da Computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília

Brasília – DF, 08 de dezembro de 2009

Banca Examinadora

_____

Profª. M.C. Maria Marony Sousa Farias Nascimento
Orientadora

_____

Prof. Francisco Javier de Obaldia Diaz
Examinador

_____

Prof. Gil Renato Ribeiro Gonçalves
Examinador

_____

Prof. João Marcos Souza Costa
Examinador

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, pela saúde e pelas oportunidades que tem me dado, para que conseguisse superar todos os obstáculos que a vida me impôs até hoje, pois sem Ele seria impossível alcançar meus objetivos.

Aos meus pais Maria Edileuza e José Ferreira, pelo carinho, apoio atenção, dedicação e preocupação que sempre tiveram comigo. Pela educação e orientação para seguir sempre o melhor caminho.

A toda a minha família que sempre me apoiou e me incentivou nos momentos mais difíceis. Em especial aos meus avós Selvina Selma e Francisco Bernardino que sempre me apoiaram durante todos esses anos.

Ao Deputado Eunício Oliveira (PMDB-CE), por todo apoio e pelas oportunidades que me deu para a conquista desse objetivo.

A todos meus amigos e colegas que sempre me incentivaram, aos amigos que me apoiaram nos momentos mais críticos do desenvolvimento deste projeto Rodrigo Santos e Julio Lessa.

Aos professores do curso de Engenharia de Computação pela paciência, atenção e por compartilharem todos os seus conhecimentos durante todo o curso.

E por fim a todos que estiveram presentes e que me incentivaram na conclusão deste projeto.

Obrigado a todos.

DEDICATÓRIA

Dedico este trabalho a minha mãe Maria Edileuza,

que foi a minha fonte de incentivo e inspiração.

Ao meu pai José Ferreira,

que foi a fonte de garra, determinação e perseverança.

E ao meu irmão Gustavo, pelo apoio e força.

Pois sem estes pilares, eu não conquistaria esse objetivo.

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação que controla os horários de entrada e saída dos funcionários de forma segura e confiável.

A solução desenvolvida foi implementada utilizando um computador, uma leitora/gravadora de cartões inteligentes e um cartão inteligente de contato JCOP. O sistema de ponto eletrônico solicita ao funcionário a inserção do cartão identificador que contém as informações do mesmo.

Após a inserção do cartão é solicitado ao funcionário que digite a sua senha para confirmar de sua entrada e/ou saída. O sistema possibilita a manutenção do cadastro de funcionários, utilizando a tecnologia Java Card e aplicação cliente que fica em um computador da organização de fácil acesso aos funcionários. Com isso a instituição poderá gerenciar de forma mais eficiente seu capital humano.

Palavras-chave: Smart Card, Java Card, APDU, Applets.

ABSTRACT

This work presents the development of an application that controls the frequency sheets of employees in a safe and reliable manner.

The solution developed was implemented using a computer, a reader / writer smart card and a smart card contact called JCOP. The system requires that the employee inserts the identify card containing his information.

After inserting the card it is requested to enter the password to confirm entry and / or output. The system allows the maintenance of the register of employees, using the Java Card technology and client application that is installed on an accessible computer in the organization to be used by its employees, so that the institution can manage more efficiently its human capital.

Keywords: Smart Card, Java Card, APDU, Applets.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS E ABREVIATURAS

AID	Application ID
APDU	Application Protocol Data Unit
API	Programming Interface
CAD	Card Acceptance Device
CAP	Converted Applet
IDE	Integrated Development Environmnet
IP	Internet Protocol
JCDK	Java Card Development Kit
JCOP	Java Card Open Platform
JCRE	Java Card Runtime Environment
JCVM	Java Card Virtual Machine
JDK	Java Development Kit
JVM	Java Virtual Machine
PIN	Personal Identifier Number
SHA	Secure Hash Algorithm

1 INTRODUÇÃO

Neste capítulo é feita a apresentação deste projeto descrevendo tópicos como motivação, objetivos e estrutura da monografia.

1.1 Motivação

O conceito de cartões inteligentes já é utilizado desde 1968. Porém, sua aplicação prática se iniciou realmente em 1984 através do Serviço Postal de Telecomunicações (PTT) francês, com os cartões de telefone, e se consolidou três anos depois, na Alemanha (CHAVES, 2007).

Na ocasião, um dos grandes fatores que preocupava a indústria financeira era a questão do grau de confiança de suas informações, que se intensificou através do desenvolvimento da criptografia moderna. Conseqüentemente, percebeu-se significativa expansão da aplicação dos cartões inteligentes para outras áreas de atuação, tais como saúde, educação, telecomunicações e transportes. O sucesso dessa tecnologia, comprovado ao longo dos anos, permitiu inclusive que atuasse como componente fundamental para garantir a eficiência e eficácia dos serviços de comércio eletrônico, tão difundidos nos tempos atuais.

Dependendo da área de atuação da tecnologia, como, por exemplo, pontos eletrônicos, o investimento em segurança da informação torna-se essencial, apesar de relativamente oneroso. Portanto, mesmo sendo necessário atualmente, percebe-se grande carência desse tipo de solução computacional.

Existem no mercado diversos tipos de controle de ponto de funcionários. O mais conhecido é a folha de ponto, em que o funcionário deve registrar seus horários e assiná-la diariamente, metodologia essa demasiadamente ineficiente. Portanto, no intuito de suprir a necessidade de gerenciamento seguro e eficiente do capital humano das organizações, este projeto propõe o desenvolvimento de uma aplicação que possibilite controlar eletronicamente os horários dos funcionários. Para tanto, será levada em consideração a utilização dos chamados Smartcards com o uso da tecnologia Java Card, que manterá todas as informações do usuário em um único cartão identificador.

1.2 Objetivos

O objetivo geral do projeto é o desenvolvimento de uma aplicação que controle os horários de entrada e saída dos funcionários de forma segura e confiável. Neste trabalho, é apresentada uma proposta para resolver esse problema. No projeto, é implementado um sistema de ponto eletrônico, que solicita ao funcionário a inserção do cartão identificador contendo suas informações. Após a inserção do cartão, é solicitado a ele a digitação de uma senha individual para confirmação de sua entrada e/ou saída. O sistema possibilitará a manutenção do cadastro de funcionários, utilizando a tecnologia Java Card e uma aplicação cliente que fica em um computador da organização de fácil acesso aos funcionários. Assim, a instituição poderá gerenciar de forma mais eficiente seu capital humano.

Os objetivos específicos do projeto são:

a. Gravar as informações do funcionário no chip do cartão através de leitora/gravadora;

b. Desenvolver uma aplicação que controle os horários de entrada e saída dos funcionários de forma segura e confiável;

c. Armazenar os dados de entrada e/ou saída dos funcionários, os quais tenham sido coletados pela aplicação, permitindo manter um histórico.

1.3 Estrutura da Monografia

Este projeto é constituído de cinco capítulos. Segue uma breve descrição.

Capítulo 1 – **INTRODUÇÃO –** aborda o que será tratado no projeto bem como a motivação e objetivo.

Capítulo 2 - **APRESENTAÇÃO DO PROBLEMA –** aborda o problema identificado e a solução proposta.

Capítulo 3 - REFERENCIAL TECNOLÓGICO - aborda conceitos e definições como, história do smartcard, tipos de smartcard,segurança no smartcard, comunicação smartcard e protocolo apdu. Aborda também, o uso da tecnologia javacard, arquitetura java card, características de segurança, máquina virtual java card (jcvm), java card api e java card applets.

Capítulo 4 - PROTÓTIPO DESENVOLVIDO – descrição detalhada do desenvolvimento do projeto propriamente dito. Citando as ferramentas utilizadas, detalhes de implementação e resultados obtidos.

Capítulo 5 – CONCLUSÃO - Descreve as conclusões obtidas nesse projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

APÊNDICE

2 - APRESENTAÇÃO DO PROBLEMA

Neste capítulo é descrita a apresentação do problema bem como a solução proposta.

O Java Card é capaz de guardar informações, sejam chaves privadas, números de contas, senhas diversas ou informações pessoais e sigilosas, centralizando a identificação do indivíduo em um único dispositivo.

Existe uma grande carência de soluções computacionais, principalmente em aplicações nas quais a segurança é um ponto fundamental, como por exemplo, o ponto eletrônico, fazendo crescer o interesse das organizações em relação à autenticação de seus usuários. Neste projeto, é apresentada uma proposta para resolver este problema, implementado um sistema de ponto eletrônico, que solicita ao funcionário a inserção do cartão identificador que contém as informações do mesmo. Após inserir o cartão a aplicação reconhece os dados do funcionário, diante disso é solicitado ao funcionário a retirada o cartão. Após a retirada do cartão é solicitado ao funcionário que digite a sua senha e em seguida que insira o cartão novamente, a aplicação lê a senha armazenada no chip do cartão, autentica e confirma a entrada e/ou saída do funcionário.

O sistema possibilita a manutenção do cadastro de funcionários, utilizando a tecnologia Java Card e uma aplicação na intranet da organização, com isso a instituição poderá controlar os horários dos funcionários de forma eletrônica e segura com o uso da tecnologia Java Card, que manterá todas as informações do

usuário em um único cartão identificador, gerenciando de forma mais eficiente seu capital humano.

A linguagem de programação JAVA é utilizada para na elaboração do programa que ler as informações do cartão Java Card tais como: nome, cargo, setor e o processamento dos horários de entrada e saída dos funcionários. A solução apresentada é composta de duas aplicações: uma que é responsável pelo gerenciamento dos usuários cadastrados e outra que autentica os usuários e registra os respectivos horários de entrada e saída.

3 - REFERENCIAL TECNOLÓGICO

Neste capítulo, faz-se referência teórica de todos os assuntos abordados no desenvolvimento deste projeto.

3.1. Smart Cards

Smart Card ou cartão inteligente é um cartão de plástico semelhante ao cartão de crédito. Em seu interior existe um chip embutido. A comunicação entre o cartão e as leitoras pode ser por contato ou sem fio. Esse chip embutido funciona basicamente como um computador, possui memória e processador que permitem o armazenamento e o processamento de informação dentro do cartão (CUNHA, 2004).

3.1.1. História do Smart Card

A história dos smart cards teve início em 1968, quando um alemão chamado Jurgen Dethloff decidiu patentear o uso de cartões para carregar chips. Em 1970 um japonês chamado Kunitake Arimura patenteou projeto semelhante, possibilitando a inserção dos smart cards no Japão no mesmo ano. Porém, somente em 1974 ocorreu o primeiro avanço concreto na tecnologia dos smart cards, quando um francês chamado Roland Moreno registrou sua patente (BAHNERT; SAHLBERG, 2006).

Em 1984 o Serviço Postal de Telecomunicações francês obteve sucesso em sua primeira tentativa de utilização dos cartões de telefone e, três anos mais tarde, devido ao sucesso dessa tecnologia, a Alemanha adotou seu uso.

A partir de então, percebeu-se os principais benefícios oferecidos pelos smart cards, a saber: armazenamento, processamento e uso de informações com confiança, segurança e alto grau de flexibilidade. Tais vantagens permitiram sua expansão a nível mundial.

3.1.2. Tipos de Smart Card

Os smart cards podem ser classificados de acordo com dois critérios:

- Meio de comunicação;
- Método de transferência de dados para a leitora.

Com base nesses dois critérios suas duas principais classes são os cartões de contato (contact cards) e os cartões sem contato (contactless cards).

3.1.3. Cartões de Contato (Contact Cards)

Como o próprio nome já diz, esse tipo de cartão precisa ser inserido na leitora para que se tenha acesso aos dados e para que os dados sejam gravados. Esse cartão possui um chip que se comunica com a leitora por meio de sinais elétricos e não possui bateria, pois utiliza a energia fornecida pela própria leitora (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2009).

As principais vantagens dos Cartões de Contato são: o baixo custo e a capacidade de armazenar e processar dados. Já as principais desvantagens são: o desgaste dos contatos devido ao tempo de uso, a fragilidade dos cartões e sua vulnerabilidade devido às descargas eletrostáticas.

A Figura 3.1 a seguir apresenta um exemplo de cartão de contato:

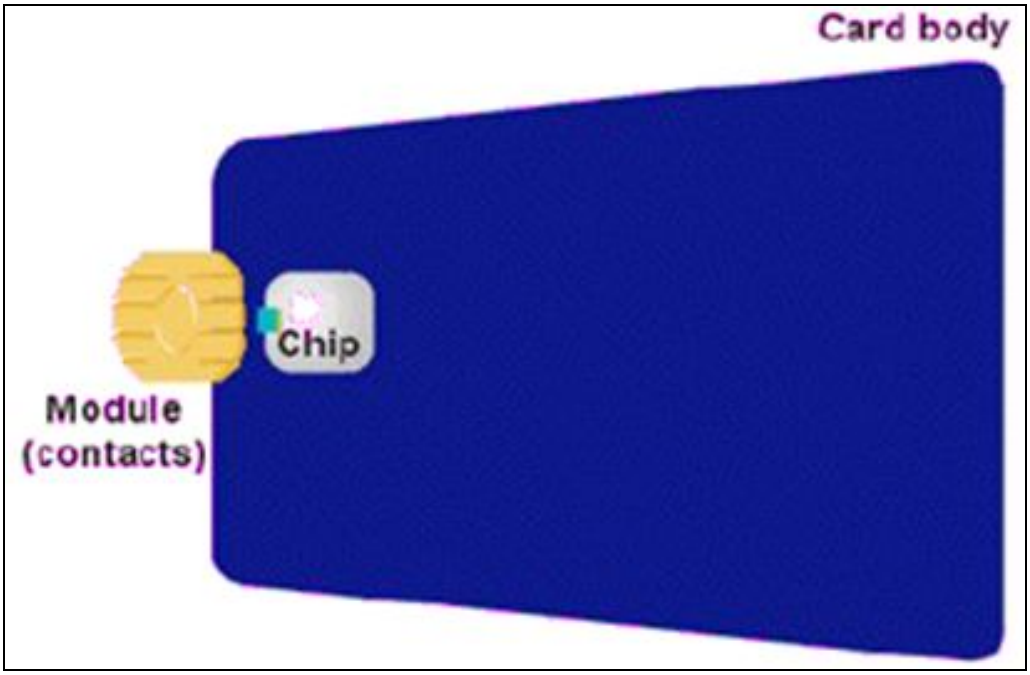

Figura 3-1: Exemplo de Cartão de Contato

3.1.4. Cartões sem Contato (Contactless Cards)

Diferentemente dos cartões de contato, esse tipo de cartão não necessita de contato físico com a leitora, ou seja, a conexão é feita através de ondas eletromagnéticas, sendo necessário apenas aproximá-lo de uma antena. Também não possuem bateria, já que a energia é fornecida pela leitora no momento da

transação. São utilizados nos casos em que a transação precisa ser rápida, simples e segura (MATOS, 2003).

Entre as vantagens estão: maior vida útil pelo fato de não terem contato físico e menor necessidade de manutenção. Dentre as desvantagens destacam-se: o fato de não armazenarem dados, o alto custo se comparados com os cartões de contato e o fato de as transações poderem ser feitas sem o conhecimento do usuário.

Na Figura 3.2 é apresentado um exemplo de cartão sem contato.

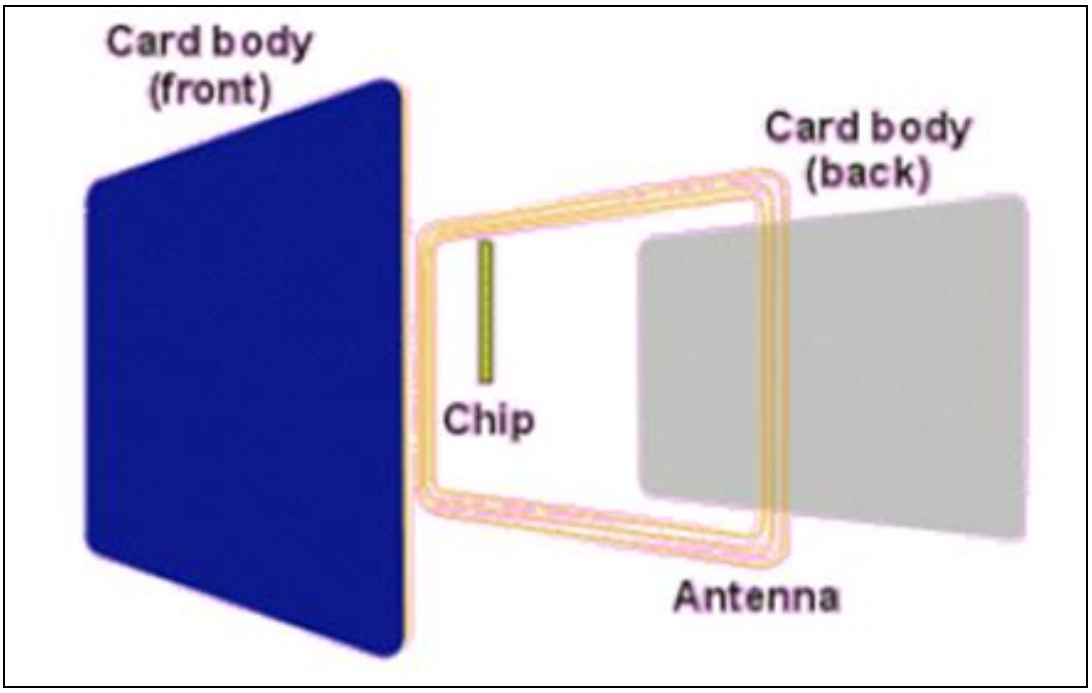

Figura 3-2: Exemplo de Cartão sem Contato

3.1.5. Segurança no Smart Card

O rápido aumento de fraudes com cartões criou a necessidade de um nível de segurança oferecido pelos smart cards em que o chip do cartão fornece ao usuário segurança e privacidade de dados com relação ao seu gerenciamento de armazenagem e distribuição. Isso dificultaria sua cópia e alteração, como ocorre com cartões de tarja magnética que podem ser facilmente clonados (LORENZONI, 2006).

No controle de segurança oferecido pelos smart cards, o acesso aos dados contidos no chip depende do controle de privilégios que se pretende obter. Portanto, é possível que alguns smart cards não necessitem de senha, possibilitando que qualquer usuário tenha acesso à leitura. Visando obter segurança considerável, este projeto utilizará como meio de proteção o uso de senhas. Por isso levará em consideração o uso do PIN (*Personal Identifier Number*), número de identificação pessoal, conhecido somente pelo proprietário do cartão e que deverá ser digitado para a validação das informações nele contidas.

3.1.6. Comunicação no Smart Card

O leitor de cartões também conhecido como CAD (*Card Acceptance Device*) é o dispositivo que envia e recebe os comandos do cartão. O CAD pode ser classificado em leitoras e terminais. As leitoras podem ser ligadas a um computador utilizando a porta serial ou USB. Os terminais são computadores nos quais o leitor dos smart cards é um de seus componentes (LORENZONI, 2006).

As aplicações que se comunicam com o cartão, independentemente de serem leitoras ou terminais, são chamadas de aplicações **host**, ou seja, é ela a responsável pela comunicação do sistema final com o leitor no qual o smart card será inserido, bem como a comunicação com os applets do cartão. É um programa que pode estar tanto em um PC, como em um terminal de pagamento entre outros dispositivos.

A comunicação entre a aplicação host e o smart card ocorre através de comandos e de um protocolo de comunicação denominado de APDU (*Application Protocol Data Unit*). A aplicação host negocia a comunicação com os applets contidos no cartão, através de um CAD. Após a comunicação estabelecida, o CAD providencia a troca dos comandos APDU entre a aplicação host e o smart card, sendo que, para cada comando APDU executado, uma resposta é recebida com o status da execução do comando (LORENZONI, 2006).

Na Figura 3.3 é apresentado o Modelo de Comunicação do Smart Card.

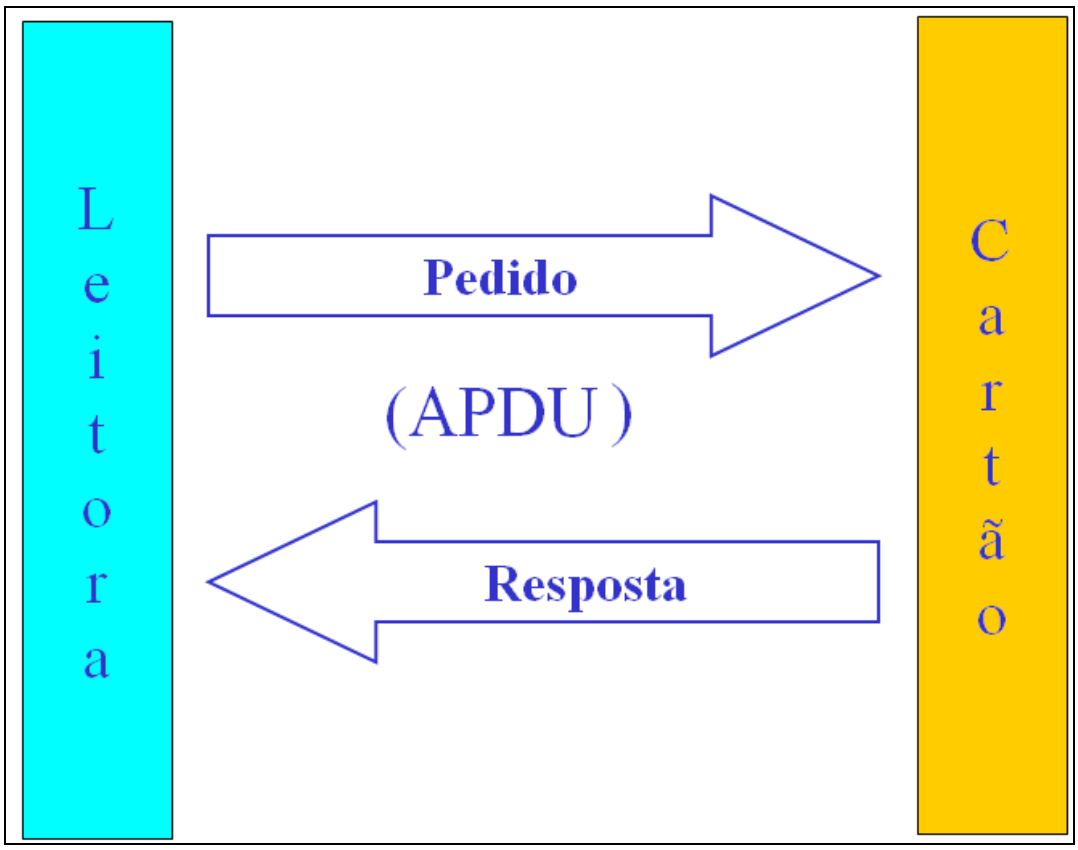

Figura 3-3: Modelo de Comunicação do Smart Card

3.1.7. Protocolo APDU

O protocolo APDU (*Application Protocol Data Unit*), é utilizado na comunicação entre a aplicação host e o cartão. Conforme especificado pela norma ISO-7816-4, o modelo de comunicação utilizado por este protocolo é o half-duplex, ou seja, os dados podem ser transmitidos em ambas direções, porém em um dado momento só podem transmitir dados em uma única direção.

Na Figura 3.4 é descrito o protocolo APDU (CALEGARI, 2002, p. 34-35).

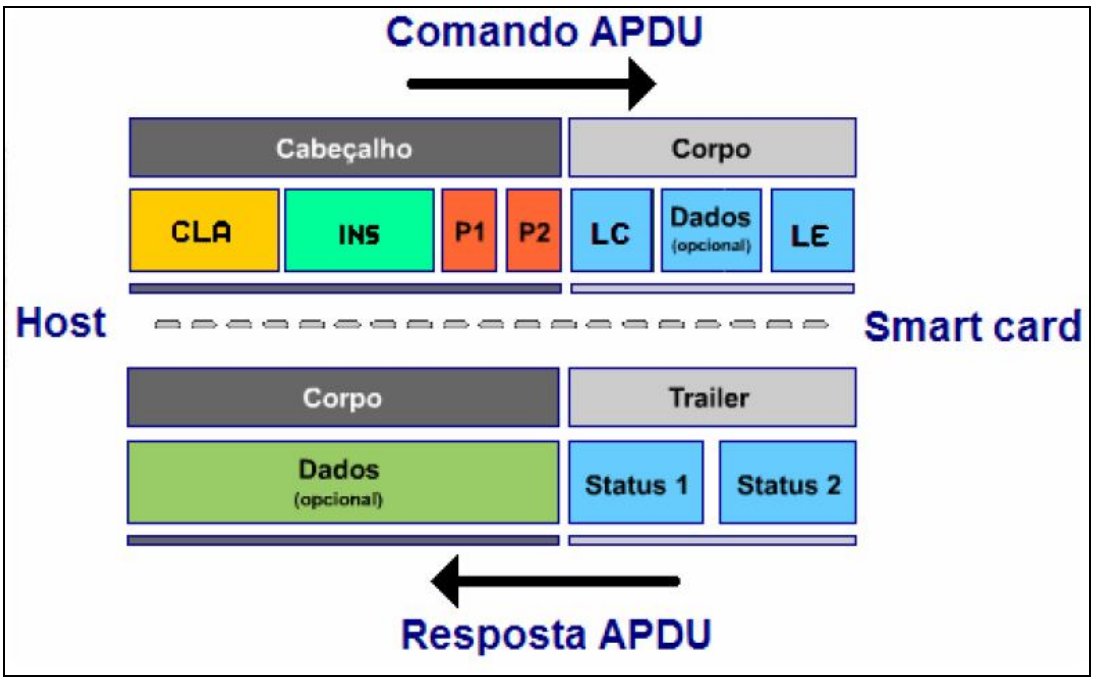

Figura 3-4: Protocolo APDU

Conforme a Figura 3.4, o cabeçalho do comando APDU é dividido em quatro bytes: CLA (classe da instrução), INS (código da instrução), P1 e P2 (parâmetros 1 e 2).

O byte reservado à classe indica a estrutura e o formato para uma categoria de comando e resposta APDU. O byte reservado à instrução especifica a instrução do comando. Os dois bytes reservados aos parâmetros P1 e P2 oferecem mais informação sobre a instrução.

A seção denominada corpo é opcional e de tamanho variável. O campo LC especifica o tamanho do comprimento do campo Dados. O campo Dados contém

os dados a serem enviados ao smart card. Já o campo LE especifica a quantidade máxima de bytes esperados como resposta.

A resposta APDU enviada pelo smart card é mais simples e composta por um Corpo opcional e um Trailer obrigatório. No campo Dados do corpo da resposta são enviados os dados do cartão. Os campos Status1 e Status 2, que compõem o campo Trailer, formam a palavra de status (*status word*). Essa *status word* armazena o resultado das operações executadas no cartão (LORENZONI, 2006).

3.2. **Java Card**

Java card é a tecnologia que permite aos smart cards e outros dispositivos executarem pequenas aplicações chamadas applets, utilizando e suportando somente alguns recursos da linguagem Java. O principal objetivo dessa tecnologia é possibilitar que um software Java instalado no cartão reserve espaço para execução de outras aplicações.

Através da tecnologia Java Card é possível que desenvolvedores construam, testem e distribuam aplicações e serviços de modo seguro e rápido, acelerando o processo de desenvolvimento, diminuindo os custos e aumentando a produção de diversos produtos.

A segurança no cartão Java Card foi aprimorada significativamente, de forma que todas as instruções possam ser verificadas previamente à sua execução. Isso previne a execução de funções ilegais e, eventualmente, o fornecimento de dados de outras aplicações.

O Java Card pode ser continuamente atualizado com novas aplicações, não sendo necessária a aquisição de um novo cartão.

Os benefícios que essa tecnologia traz são (MOSTARDINHA; SILVA, 2005):

- **Interoperabilidade** - applets desenvolvidos com a tecnologia Java Card podem ser executados em qualquer dispositivo ou smart card que siga os padrões dessa tecnologia, independente de fornecedor e plataforma de hardware;

- **Segurança** - a tecnologia Java Card leva em consideração a segurança inerente da linguagem de programação Java que provê um ambiente de execução seguro. Projetada através de um processo aberto, assegurado pela indústria como solução capaz e segura;

- **Muli-aplicação -** permite a coexistência de múltiplas aplicações de forma segura em um único smart card;

- **Dinamismo -** novas aplicações podem ser instaladas e atualizadas, de modo seguro, mesmo após o lançamento do smart card. Evita-se a troca dos cartões a cada atualização de software;

- Compatibilidade - a API Java Card é compatível com padrões internacionais para smart cards como ISO7816 e EMV.

Uma aplicação Java Card completa consiste em: uma aplicação host (off-card), um dispositivo de interface (leitor smart cards) e uma aplicação no cartão (applet).

3.2.1. Arquitetura Java Card

Por possuir CPU, memória ROM, uma memória EEPROM e um co-processador criptográfico, a arquitetura Java Card é muito semelhante à arquitetura Smart Card.

Sua arquitetura consiste em:

- Applets;
- JavaCard API;
- JavaCard Virtual Machine (JCVM) / JavaCard Runtime Environment (JCRE);
- Sistema operacional do cartão.

Na Figura 3.5 é apresentada a arquitetura Java Card (ARIZA, 2004).

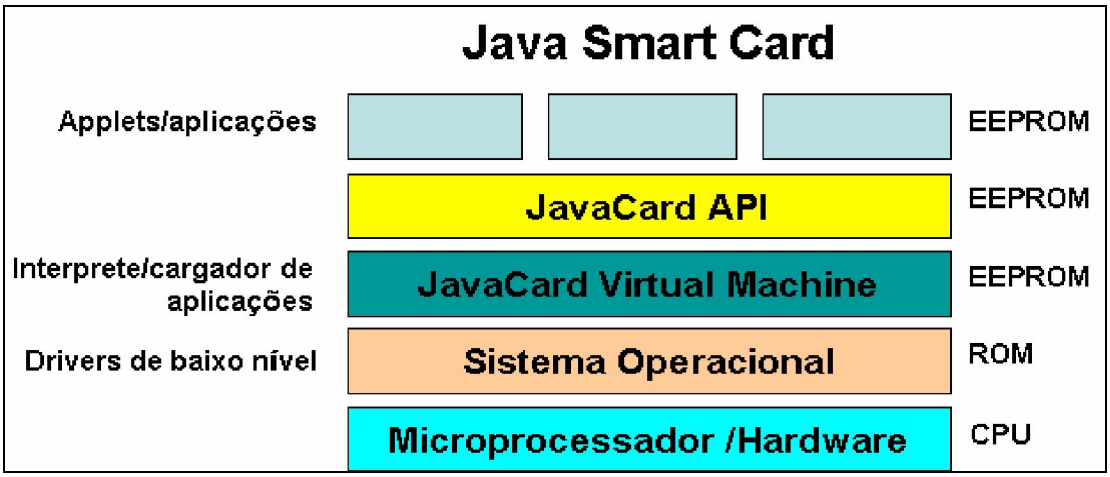

Figura 3-5: Arquitetura Java Card

O sistema operacional é uma máquina virtual Java (JVM), que é responsável pela execução dos programas escritos em Java. A máquina virtual Java

Card (JCVM) é uma versão da linguagem Java, na qual são executados serviços tais como: a comunicação da leitora com o Smart Card ou criar dados na memória. Isso é feito por meio de bibliotecas em programas Java que são chamadas de classes. A JCVM é composta de duas partes distintas, uma externa ao cartão e outra interna. O JavaCard Runtime Environment (JCRE) é constituído por uma máquina virtual Java Card (JCVM), uma API Java Card e extensões específicas. Sua função é definir o ciclo de vida de um applet em uma máquina virtual Java Card (JCVM). Já as applets são as aplicações contidas dentro do cartão.

A questão mais significativa do Java Card é a distinção entre o sistema e as aplicações, além de definição de um ambiente de apoio à memória do Smart Card, à comunicação e à segurança.

3.2.2. Características de Segurança Java Card

Um dos fatores mais importantes da tecnologia Java Card é a segurança que ela oferece às aplicações, evitando o acesso indevido de aplicações ilegais à integridade e à segurança dos dados contidos no cartão.

Existem várias técnicas de segurança a serem consideradas. Neste projeto, será utilizada a criptografia.

3.2.3. Criptografia

A criptografia expressa à idéia de confidencialidade. O algoritmo de criptografia utilizado neste projeto é o algoritmo de hash SHA1.

O algoritmo de hash é uma seqüencia de bits gerados por um algoritmo de dispersão, em geral representada em base hexadecimal, que permite a visualização em letras e números (zero a nove e A F). O conceito teórico diz que hash é a transformação de uma grande quantidade de informações em uma pequena quantidade de informações.

Essa seqüencia identifica a informação unicamente. Como por exemplo, a senha do funcionário. Transforma dados de tal forma que o resultado seja exclusivo. Além disso, funções usadas em criptografia garantem que não é possível a partir de um valor de hash retornar à informação original. O tipo de algoritmo de HASH utilizado foi o SHA1.

A SHA (Secure Hash Algorithm) está relacionada com as funções criptográficas. A função mais usada é a SHA-1, é usada numa grande variedade de aplicações. SHA-1 é um one-way hash função. One-way funções são caracterizados por duas propriedades. A primeira é que eles são uma via de único sentido. Isto significa que você pode ter uma mensagem e calcular um valor de hash, mas você não pode ter um valor hash e recriar a mensagem original (TECH FAQ, 2009).

3.2.4. Máquina Virtual Java Card (JCVM)

A tecnologia Java Card tem sua máquina virtual dividida em duas partes. Uma parte externa ao cartão e outra no cartão. A parte externa chamada de conversor, é executada em um PC, é responsável por interpretar bytecodes, gerenciar classes e objetos. Na interpretação dos bytecodes é verificado se não houve a utilização de algum comando que não atenda as especificações Java Card.

Após essa verificação é gerado um arquivo chamado CAP (Converted Applet). O CAP contém as classes carregáveis e executáveis na forma de uma representação binária.

Após o arquivo CAP ser gerado pelo conversor é utilizado um dispositivo chamado CAD (Card Acceptance Device) que diz respeito ao leitor/gravador onde o smart card é inserido, a partir desse momento a transmissão do arquivo CAP do computador pessoal para o smart card é iniciada. Diante disso o applet pode ser instalado e registrado, função esta executada pela segunda parte da JCVM, que também é responsável pela execução das instruções em bytecode dos applets presentes no smart card.

3.2.5. Java card runtime enviroment - JCRE

O JCRE é o responsável por isolar as aplicações dos detalhes específicos de cada cartão, além de efetuar a inicialização da JCVM e o gerenciamento de recursos entre sessões CAD.

O JCRE consiste em uma máquina virtual Java Card (JCVM), uma API Java Card e extensões específicas. O JCRE também controla o ciclo de vida dos applets.

Cada applet no smart card é identificado unicamente, através de um ID da aplicação (Application ID ou AID), e controlado pelo JCRE.

3.2.6. Java Card API (Java Card Application Programming Interface)

As aplicações Java, API's são classes pré-definidas para programar smart cards. É composta de um pacote básico (java-card.framework) e três pacotes estendidos (java-cardx.framework, java-cardx.crypto e java-cardx.crypto.enc). O pacote básico java-card.framework define todas classes essenciais necessárias para a criação de um applet Java Card.

3.2.7. Java Card Applets

Java Cards Applets são programas que residem dentro do cartão e tem a capacidade de suportar multi-aplicações, ou seja, é capaz de ter mais de um applets. Os applets são executados na JCRE. Eles são ativados no momento da inserção do cartão no dispositivo CAD, permanecendo nesse estado até a o cartão ser retirado do mesmo.

Um applet é derivado da classe javacard.framework.Applet. Essa classe é a de todos os applets em um Java Card e ela define as variáveis e os métodos de um applet. Um applet em execução no cartão é uma instância da classe Applet. Assim como qualquer objeto persistente, uma vez criado, um applet vive no cartão para sempre.

4 - PROTÓTIPO DESENVOLVIDO

Este capítulo apresenta o protótipo desenvolvido neste projeto, tecnologia adotada, hardware e software.

Baseado nos conceitos abordados, o projeto propõem uma solução tecnológica utilizando Java Card para Smart Cards. Essa solução é divida em duas partes onde a primeira é a aplicação de cadastro dos dados dos funcionários e a segunda é o desenvolvimento das applets que serão gravadas no cartão, além de demonstração prática da aplicação.

Para demonstrar esta solução, foi observada a maneira de como as organizações gerenciam seus funcionários, e a folha de ponto ainda é utilizada. Isto propicia a utilização da tecnologia Java Card para elevar a agilidade, a segurança e o gerenciamento dos horários de entrada e saída dos funcionários dentro de uma organização.

Ao invés do funcionário ter que assinar a sua folha de ponto diariamente ou assiná-la por completo no final do mês, a aplicação exige que ele registre seu ponto sempre que chegar à organização e ao sair também. Isso garante que ele não fraude o seu gerenciamento se por acaso faltar sem justificativa ou chegar atrasado. As informações são armazenadas de forma segura através de um módulo de cadastro de funcionários e personalização dos Java Cards, pois cada funcionário possui um cartão e senha pessoal.

Portanto, a escolha do uso da tecnologia Java Card para o desenvolvimento desse projeto destaca, além da segurança, a privacidade das informações, praticidade de uso e a flexibilidade de implementação, sem contar a contribuição ambiental pela eliminação do uso de papel, dentre outras vantagens oferecidas pela tecnologia.

4.1 Tecnologia adotada

Para o desenvolvimento deste projeto foi necessário utilizar as seguintes ferramentas:

- Eclipse – IDE para o desenvolvimento da aplicação host e dos applets.

- JDK 6 – Ferramenta para compilar a aplicação host e os applets. Utilizada também para executar a aplicação host.

- JCDK 2.2.1 – Ferramenta de desenvolvimento e testes das applets, pois possui as classes Java Cards.

- GPSHELL – Ferramenta utilizada para instalar os applets no cartão.

- Leitor/gravador de cartões inteligentes da empresa Perto Software. A figura 4.1 apresenta o tipo de leitora/gravadora de cartões inteligentes utilizado.

- Cartões JCOP (Java Card Open Platform) 21.

A figura 4.1 apresenta as ferramentas utilizadas.

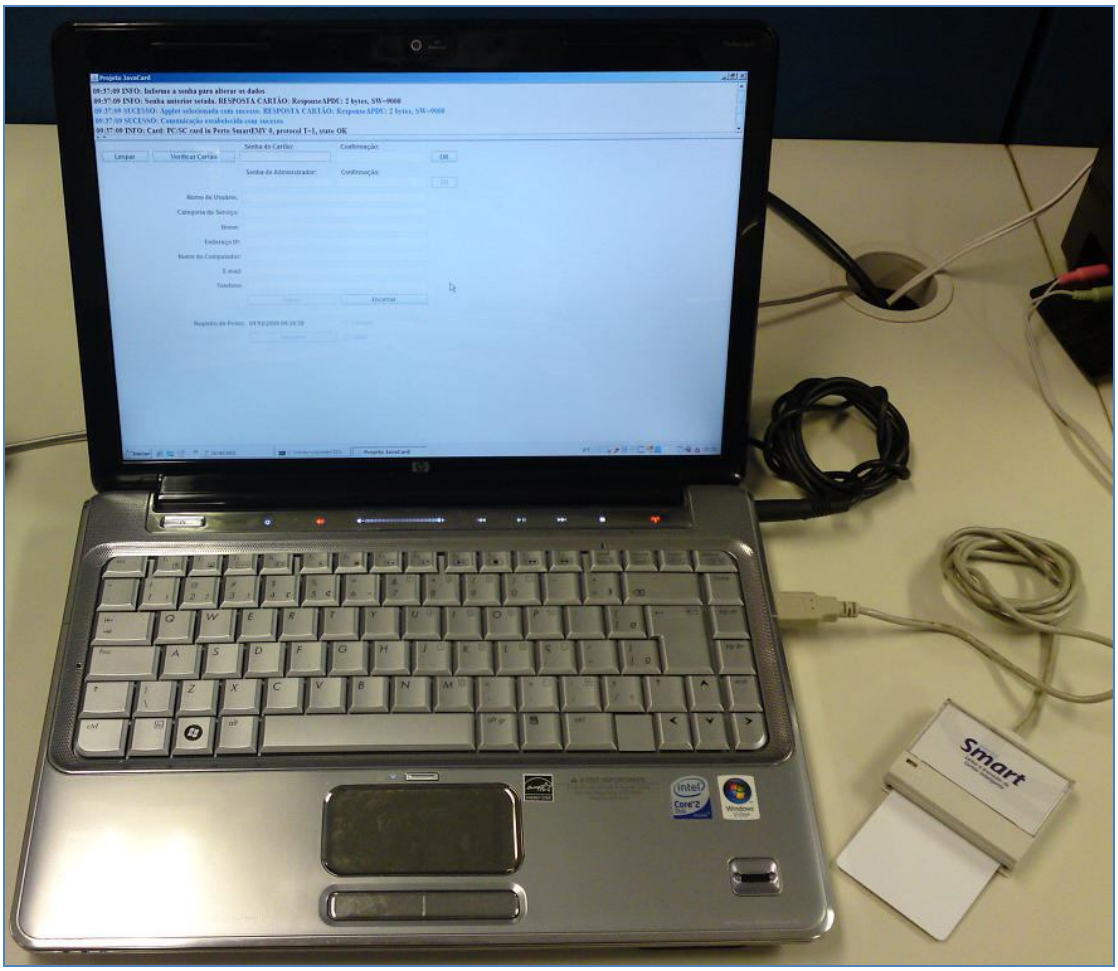

Figura 4-1: Foto das Ferramentas Utilizadas

A partir daí o desenvolvimento e testes foram realizados com as ferramentas citadas acima.

4.1.1 Escopo do Projeto

A Figura 4.2 mostra o escopo do projeto.

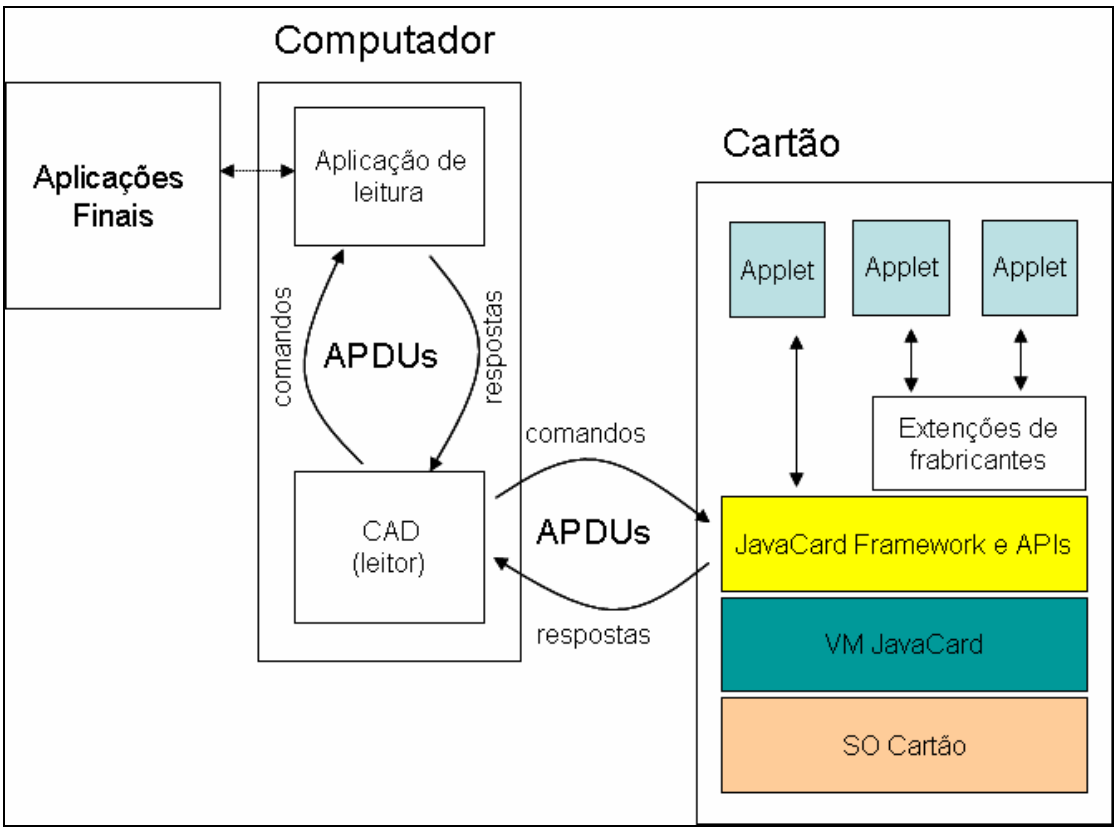

Figura 4-2: Escopo do Projeto

- **Aplicações finais:** São as aplicações que precisam dos cartões, por exemplo o sistema proposto neste projeto;

- **Aplicações de leitura:** São as aplicações que fazem a ponte entre as aplicações finais e o dispositivo de leitura dos cartões e os cartões;

- **CAD (leitor):** O CAD (Card Acceptance Device) é o dispositivo que envia e recebe comandos do cartão, além de fornecer uma fonte de poder para o seu funcionamento;

- **Applets no cartão e ambiente Java**: São os programas que correm no cartão (applets).

4.1.2 Implementação

Ao invés do funcionário ter que preencher a sua folha de ponto com os horários de entrada e saída. Ele utilizará seu cartão conectado a um computador host através de um leitor de smart cards. Para um melhor entendimento do funcionamento da aplicação é descrito todo o processo de desenvolvimento.

Inicialmente deve-se verificar se o serviço chamado Cartão Inteligente está ativo. Ele gerencia o acesso a cartões inteligentes lidos por um computador, caso inserido em um leitor de cartão inteligente conectado ao computador. Se este serviço for interrompido, o computador não poderá ler cartões inteligentes. E se este serviço for desativado, quaisquer serviços que dele depende diretamente não será inicializado.

Para o desenvolvimento da applet que será gravada no cartão, devem-se levar em consideração as ferramentas de desenvolvimento e hardware, uma vez que a tecnologia Java Card impõe suas limitações.

De acordo com o projeto a applet desenvolvida tem a função de retornar as informações necessárias para que o registro de ponto do funcionário seja computado de forma segura e confiável (SOUZA NETO, 2007).

4.1.3 Etapas para a Instalação de um Applet Java Card

- **Escrever o applet Java Card** – O código fonte da applet desenvolvida utilizando o ambiente Eclipse junto com a ferramenta GPSHELL;

- **Compilar o applet** – A compilação é feita utilizando a ferramenta GPSHELL. O código fonte da applet é compilado criando um arquivo do tipo class. No processo de compilação são utilizadas as bibliotecas do JavaCard Framework para gerar o arquivo Class;

- **Converter o arquivo Class** - A conversão do arquivo tipo class é feita utilizando a ferramenta GPSHELL. Converte as classes geradas em um arquivo CAP (Converted APplet), usando a ferramenta de conversão;

- **Instalar a o arquivo CAP no cartão** - A instalação do CAP no cartão. A instalação do arquivo no cartão é feita utilizando a ferramenta GPSHELL que instala a applet no cartão. É invocado o método de install() do applet e é criada uma instancia do applet dentro do cartão.

A figura 4.3 demonstra o esquema de criação de uma applet.

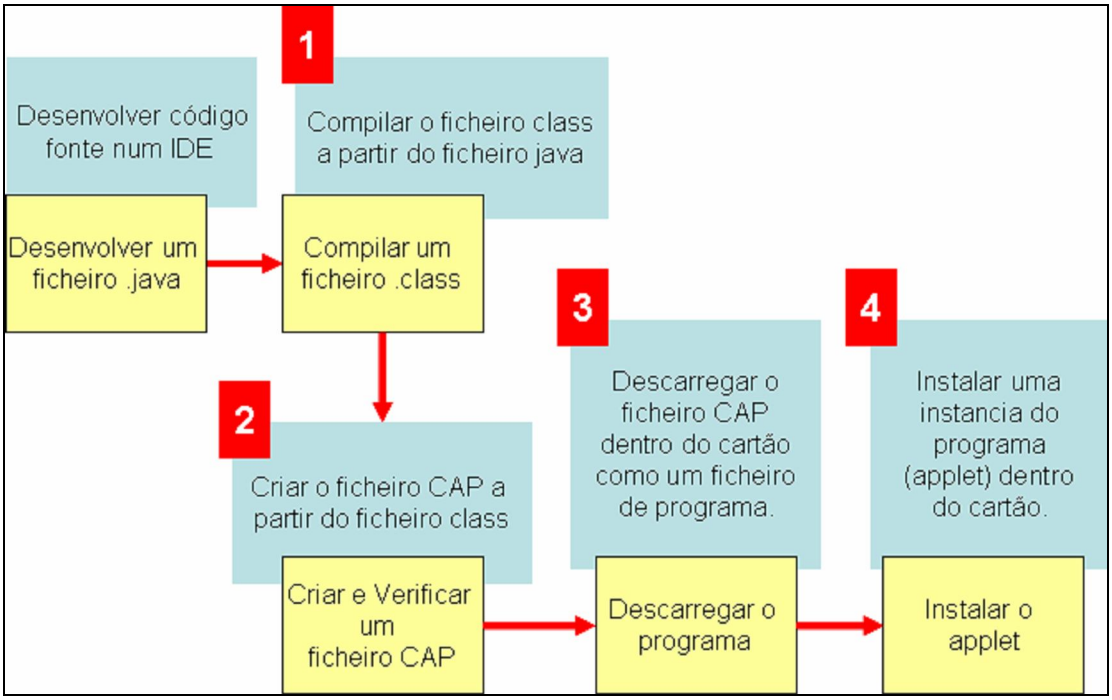

Figura 4-3: Esquema de Criação de uma Applet

4.1.4 Applets Java Card

A aplicação tratada neste projeto é dividida em duas partes: uma que é a aplicação host (cliente), que fica armazenada um computador e a outra que fica no cartão.

A comunicação entre o applet e uma aplicação host dá-se através de troca de pacotes lógicos entre a leitora do cartão e o próprio cartão, essa troca é feita através do protocolo APDU (Application Protocol Data Unit). A leitora fornece energia ao cartão através da porta USB e serve como o meio de comunicação entre a aplicação host e o applet que está no cartão.

As applets são extensões da classe javacard.framework.Applet, com as especificações da tecnologia.

Para o desenvolvimento da applet é necessário executar dois métodos: o método install que instala a applet e o método process que é executado a cada envio de um comando APDU. Depois de executados esses dois métodos a applet fica em estado de unselectd e pronta para execução. Geralmente a applet existe enquanto durar o cartão. A figura 4.4 demonstra o ciclo de vida de uma applet (SOUZA NETO, 2007).

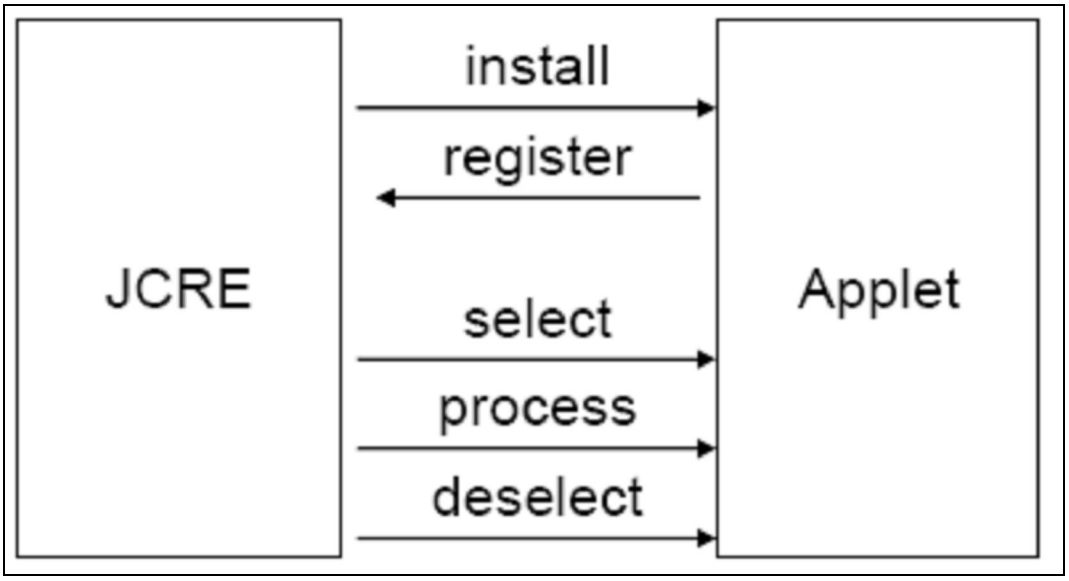

Figura 4-4: Ciclo de Vida de uma Applet

4.1.5 Métodos de uma Applet Java Card

Para que a aplicação funcione a classe javacard.framework.Applet define quatro métodos públicos que são utilizados pelo o JCRE, são eles (SUAVI, 2005):

- ***Método install(byte[], short, byte) –*** Este método é chamado pelo JCRE antes de criar uma instância no applet do cartão. A implementação deste método é responsável pela criação de todos os objetos que o applet necessita para sua execução. E por fim registra o applet com o método register.

- ***Método select() –*** Este método é chamado pelo JCRE quando recebe um comando SELECT APDU. O APDU contém o AID (APPLET IDENTIFIER) do applet que será selecionado. O AID é uma seqüência de entre 5 e 16 bytes que identifica uma aplicação Java Card. Quando o JCRE recebe um comando SELECT APDU, ele verifica se já existe algum applet selecionado, caso exista, ele chama o método deselected, para então chamar o método select da applet cujo AID foi especificado. Feito isso, é chamado o método process do applet selecionado para o seu processamento e devolva à leitora a informação em questão.

- ***Método process(APDU) -*** Quando o cartão recebe um comando APDU o JCRE chama o método do applet selecionado passando o comando APDU como parâmetro. Feito isso o applet identifica o comando APDU e os parâmetros, caso existam e os processa de acordo com o protocolo que se tenha definido para a interação entre o applet e o host. Se a execução foi concluída com sucesso o applet vai carregar no RESPONSE APDU a

informação de retorno. O JCRE é preenche os demais campos do RESPONSE APDU informando que a execução teve sucesso (0x9000 de acordo com a ISO 7816). Caso o APDU não seja capturado pela applet o JCRE gera um RESPONSE APDU informando uma exceção ou erro (ISOException).

- ***Método deselect()-*** Este método é chamado pelo JCRE para informar ao applet selecionado em questão que vai deixar de estar selecionado. Isso permite ao applet realizar tarefas de limpeza que sejam necessárias para ficar num estado consistente.

O processo completo de verificação da applet dá-se da seguinte forma: A seleção de um applet instalado no cartão ocorre quando a aplicação host requisita ao JCRE para que este selecione o mesmo no cartão, através de um comando APDU denominado select.

O JCRE procura um AID em suas tabelas, correspondente ao applet requisitado.O método select informa ao applet que o mesmo está sendo selecionado.

Após esta operação, o applet encontra-se pronto para receber as requisições aos comandos APDUs da aplicação host. Isto ocorre quando o método process é chamado.

É importante observar que o método select possibilita que o desenvolvedor do applet realize alguma operação de inicialização necessária para que o mesmo torne-se apto a processar os comandos APDU da aplicação host.

As classes e os applets Java Card são empacotados em um único arquivo denominado CAP, semelhante a um arquivo Jar (Java Archive). Esse arquivo é instalado no cartão (SOUZA NETO, 2007, p. 26).

4.2 Desenvolvimento da Aplicação Host

A aplicação host foi desenvolvida em Java. A proposta do projeto é um sistema de cadastro dos dados e senha do funcionário, onde no momento do cadastro o funcionário deve escolher sua senha uma única vez que é pessoal e intransferível.

4.2.1 Módulo de Cadastro

O módulo de cadastro do funcionário é responsável pelas informações do funcionário contidas no cartão, para tanto é necessário para realizar o cadastro que se tenha uma leitora de cartões conectados no computador em que a aplicação host esteja instalada e um cartão com a applet gravada para que a aplicação possa executar a applet.

A figura 4.5 apresenta a tela inicial da aplicação host.

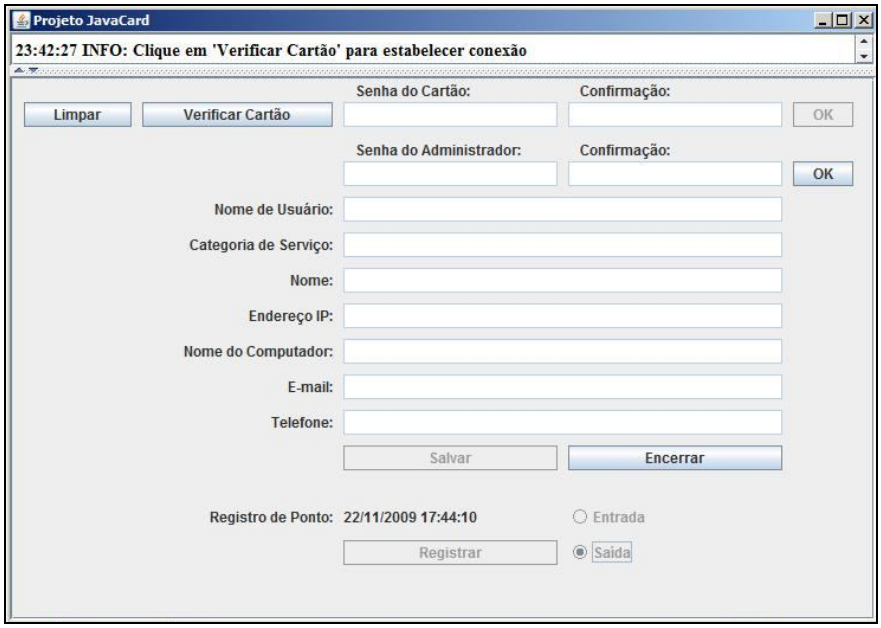

Figura 4-5: Tela Inicial da Aplicação Host

Através desse módulo é possível gerenciar algumas informações relativas do funcionário como:

- Nome de Usuário:
- Categoria de serviço
- Nome
- Endereço IP
- Nome do Computador
- E-mail
- Telefone

Operacionalmente o funcionamento deste módulo é bem simples. No primeiro acesso é solicitado que se clique em Verificar Cartão, onde a aplicação

verifica se há algum cartão inserido na leitora, caso não tenha nenhum cartão a aplicação host retorna uma mensagem de erro informando que não há cartão na leitora.

A figura 4.6 apresenta a tela de erro.

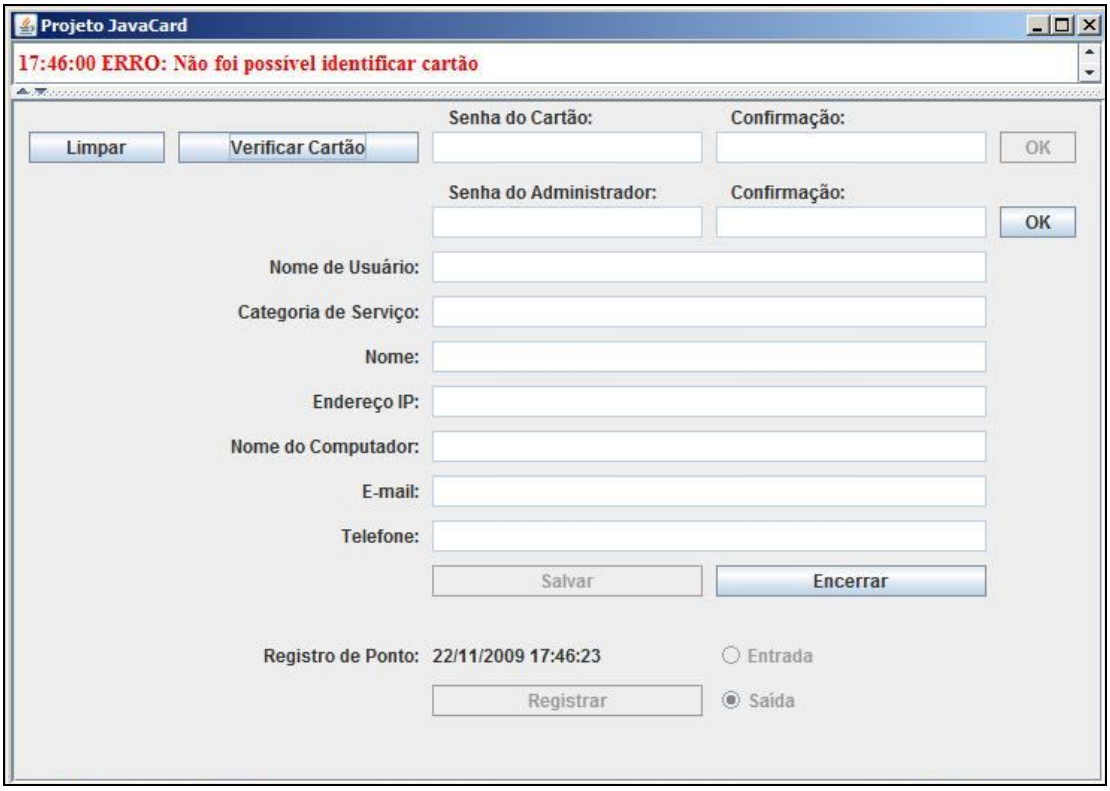

Figura 4-6: Tela de Erro da Aplicação Host

Com o cartão inserido na leitora, a aplicação host envia um comando APDU para verificar se o cartão está inserido e a applet instalada no cartão retorna uma resposta APDU confirmando que o cartão está inserido e que a applet foi selecionada.

A figura 4.7 demonstra a tela de reconhecimento da applet.

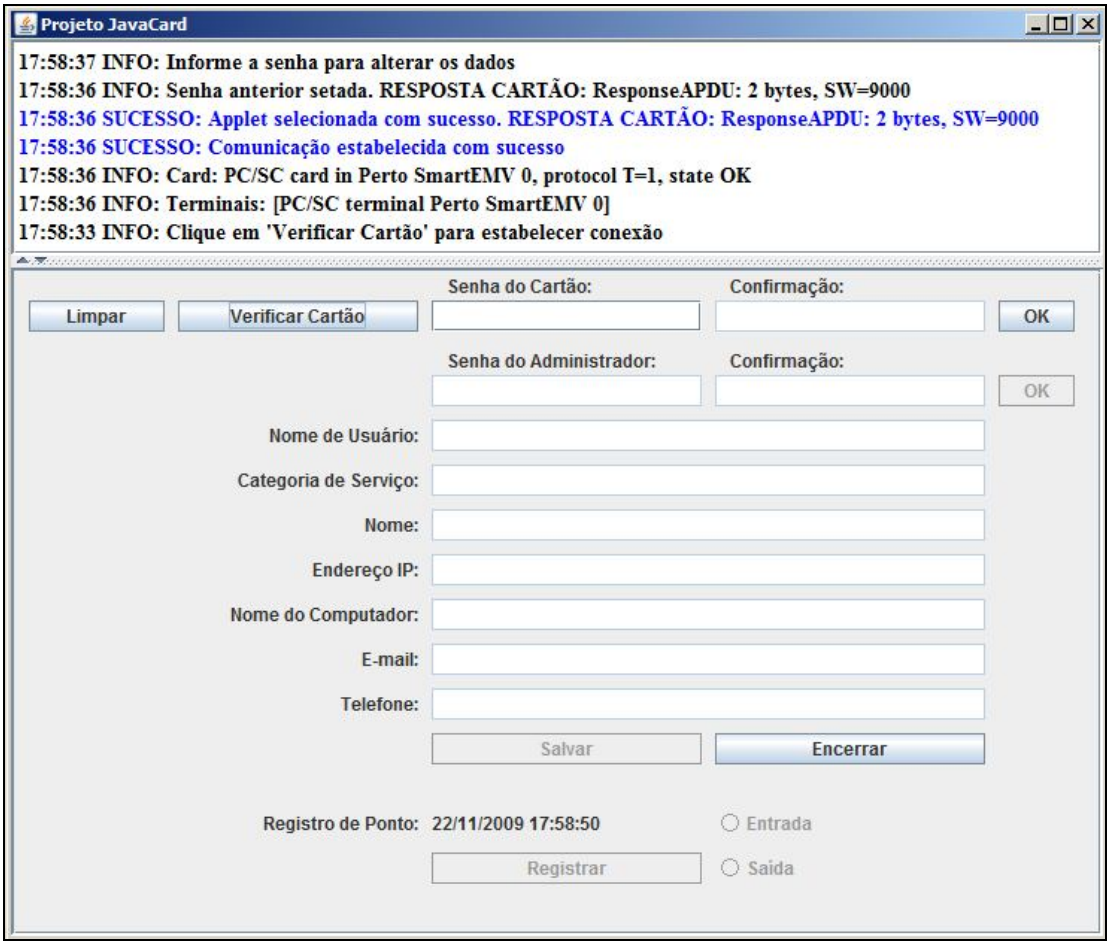

Figura 4-7: Tela de Reconhecimento da Applet

A partir daí dá-se inicio ao cadastro do funcionário. É solicitado ao funcionário que escolha uma senha, o funcionário digita a senha escolhida, confirma a senha e clica no botão “OK” para validar. O responsável pela administração da aplicação digita a senha que é a senha de administração da aplicação. Após isso é feito o cadastro com as informações solicitadas e clica no botão salvar para salvar a informações. Uma vez gravadas as informações do funcionário, só pode ser alterada pelo administrador da aplicação. Essas informações além de serem gravadas no

cartão são gravadas também em um banco de dados. A figura 4.8 apresenta o cadastro de um funcionário.

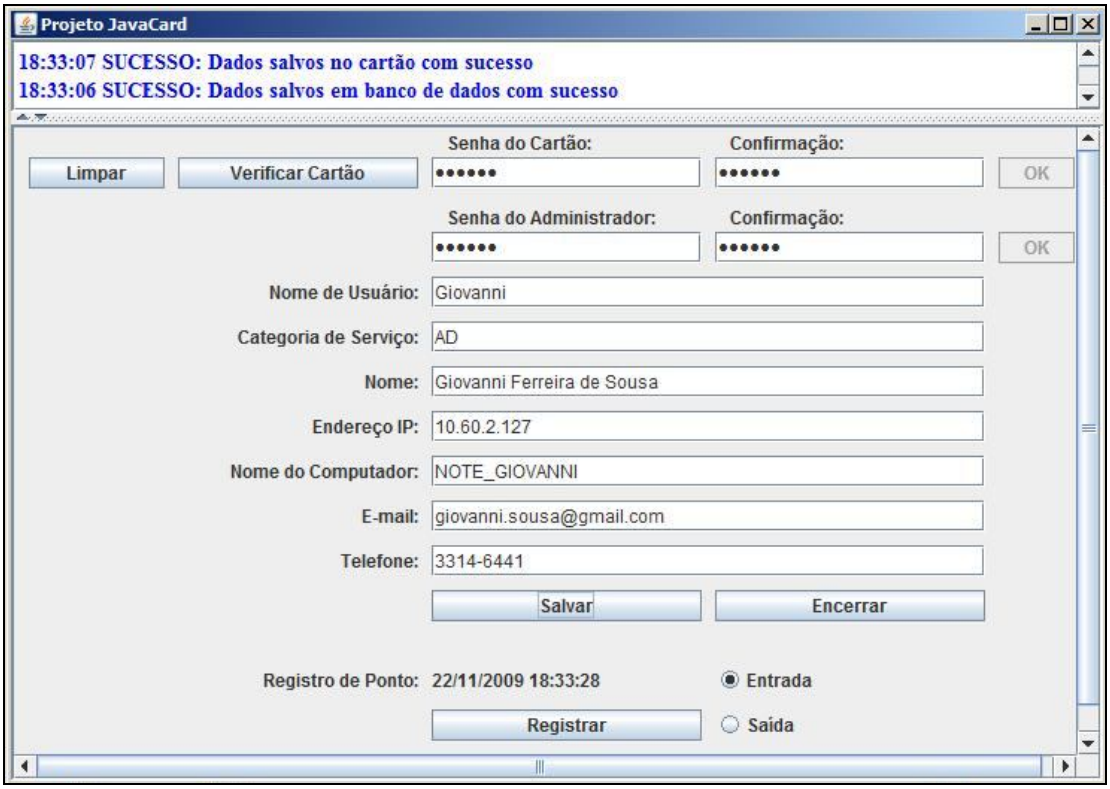

Figura 4-8: Cadastro de um Funcionário

É importante lembrar que para gravar a senha digitada pelo funcionário foi utilizado um algoritmo de hash. A fim de evitar que a senha do funcionário seja utilizada de forma indevida.

Após o funcionário ter sido cadastrado ele receberá o seu cartão e estará apto a registrar seu ponto diariamente. No momento em que ele inserir o seu cartão na leitora e clicar no botão “verificar cartão”, é solicitado que ele digite a senha e

clique em “OK”, nesse momento a aplicação host reconhece todos os seus dados. A figura 4.8 apresenta o reconhecimento dos dados.

Figura 4-9: Reconhecimento dos Dados

Após o reconhecimento dos dados o funcionário escolhe a opção desejada e clica no botão “registrar ponto” e a aplicação registra a data e o horário da ação. As figura 4.9 mostra o registro de entrada e as figura 4.10 mostra o registro de saída.

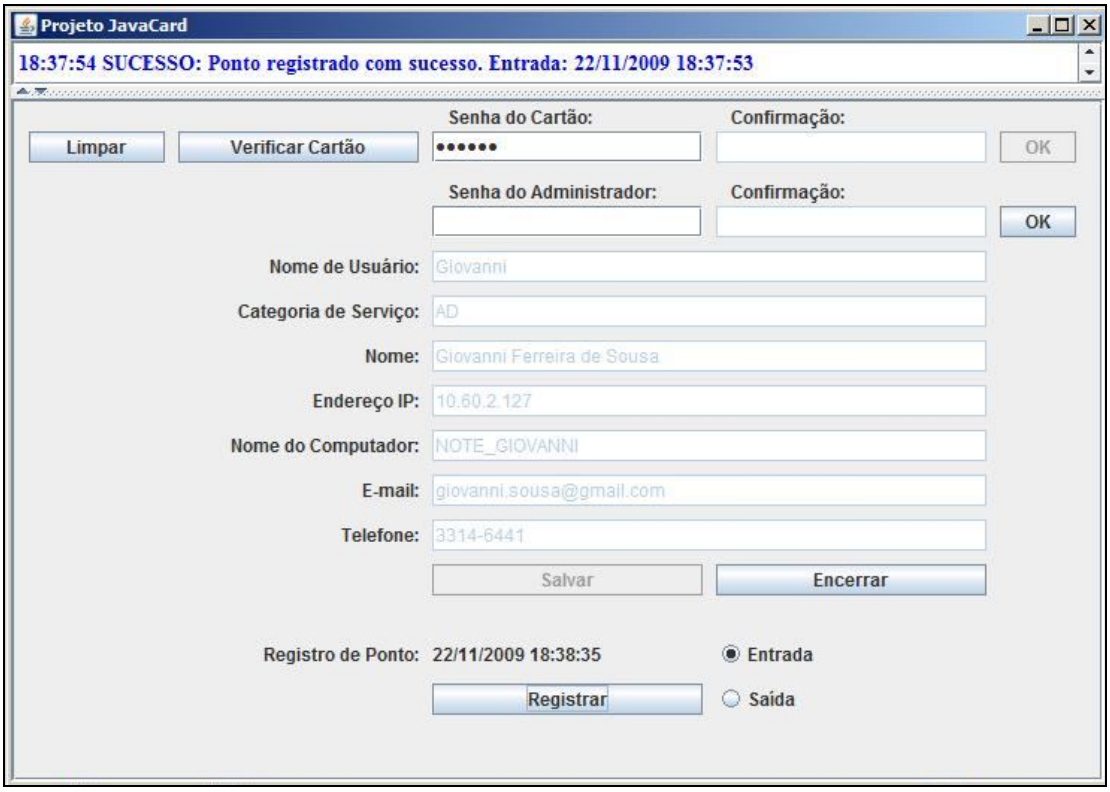

Figura 4-10: Registro de Entrada

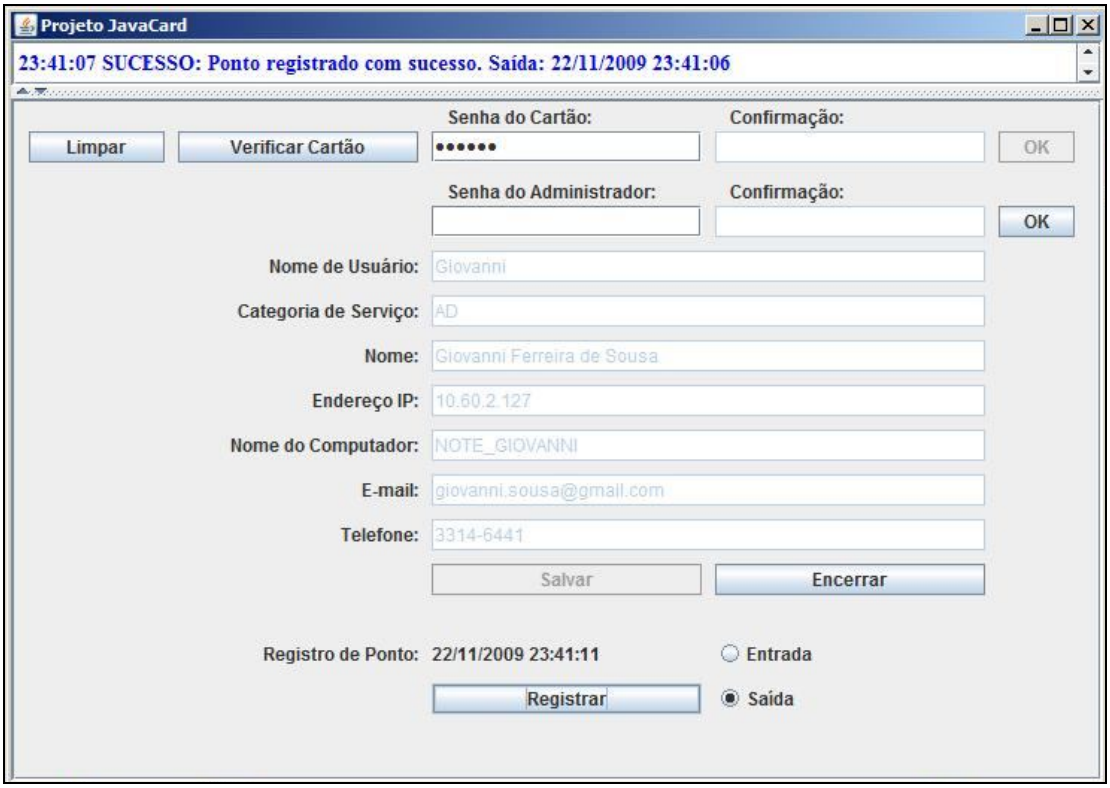

Figura 4-11: Registro de Saída

Antes de enviar as informações dispostas na aplicação host elas são dispostas em comandos APDU junto com seus métodos. Para cada comando enviado para o cartão uma resposta de status é recebida. A aplicação host recebe estes comandos e verifica se houve algum erro. Se não houver nenhum erro significa que a conversão e transmissão das informações foram aceitas pelo cartão. Após essa verificação é exibida ao funcionário uma mensagem que tudo ocorreu perfeitamente, e se ocorrer algum erro é exibida a mensagem de erro.

5 - CONCLUSÕES

Com a crescente preocupação das organizações em gerenciarem o seu capital humano, torna-se necessário o desenvolvimento de aplicações em ambientes seguros, onde a privacidade e a confidencialidade são fatores primordiais em qualquer organização.

Dessa forma este projeto teve como finalidade o desenvolvimento de uma aplicação capaz de controlar eletronicamente de forma segura, rápida e eficiente os horários de entrada e/ou saída dos funcionários de uma organização.

Sendo assim conclui-se que todos os objetivos específicos todos foram alcançados.

REFERÊNCIAS BIBLIOGRÁFICAS

ARIZA, C. **JavaCards:** Prática de Tecnologias de Informação. 2004. Disponível em:<http://www.dca.fee.unicamp.br/projects/regra_c/docs/cartao_unicamp/MSIPTI1ArizaJavaCards2004v02.pdf>. Acesso em: 20 nov. 2009.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT - CB21). et al. **Smart Card Memory Card**. Disponível em: <http://www2.dem.inpe.br/ijar/smartcard.pdf>. Acesso em: 19 nov. 2009.

BAHNERT, G.; SAHLBERG, J. R. **Smart Cards**. 2006. Disponível em: <http://www.inf.pucrs.br/~eduardob/disciplinas/ProgPerif/sem06.2/trabalhos/Seminarios/resumos/SmartCards.pdf>. Acesso em: 19 nov. 2009.

CALEGARI, D. T. **Uma Implementação de Criptografia de Curvas Elípticas no Java Card.** 2002. 61 f. Dissertação (Mestrado em Ciência da Computação), Instituto de Computação, Universidade Estadual de Campinas, Campinas, 2002.

CHAVES, F. C. **Tecnologia Java Card**. 2007. Disponível em: <http://www.javanoroeste.com.br/novo/artigos/javacard.html>. Acesso em: 19 nov. 2009.

CUNHA, R. M. **Smart Card e e-TAG**. 2004. Disponível em: <http://www.gta.ufrj.br/grad/04_2/smartcard/>. Acesso em: 19 nov. 2009.

FONTES, M. F.; DUARTE, O. C. M. B. **Smart Cards para o Controle de Acesso**. 2005. Disponível em: <http://inf.upf.tche.br/~52731/CD/Smart_Card_Trab.pdf>. Acesso em: 19 nov. 2009.

GONÇALVES, E. **Dominando Eclipse:** Tudo que o Desenvolvedor Java Precisa para Criar Aplicativos para Desktop. Rio de Janeiro: Ciência Moderna, 2006. 326 p.

HSW INTERNATCIONAL, INC. **O que é um Smart Card**. 2009. Disponível em: <http://informatica.hsw.uol.com.br/questao332.htm>. Acesso em: 19 nov. 2009.

LORENZONI, A. F. **Smart Cards – Java Card.** 2006. 67 f. Trabalho de Conclusão de Curso (Ciência da Computação), Instituto de Ciências Exatas e Tecnológicas, Centro Universitário Feevale, Novo Hamburgo, 2006.

MATOS, C. L. **Smart Card**. 2003. Disponível em: <http://www.gta.ufrj.br/grad/01_2/smartcard/smartcard.html>. Acesso em: 19 nov. 2009.

MOSTARDINHA, F. A. M.; SILVA, S. **Sistema de Informação Clínica e Genética Suportada num Cartão Inteligente (Java Card).** 2005. 86 f. Relatório Final de Projecto, Departamento de Electrónica e Telecomunicações, Universidade de Aveiro, Aveiro, 2005.

SOUZA NETO, P. A. **JDML – Java Card Modeling Language: Definição e Implementação.** 2007. 68 f. Dissertação (Mestrado em Sistemas e Computação), Departamento de Informática e Matemática Aplicada, Universidade Federal do Rio Grande do Norte, Natal, 2007.

SUAVI, C. G. **Documentos e Dinheiro Eletrônico com Smart Cards utilizando Tecnologia Java Card.** 2005. 53 f. Trabalho de Conclusão de Curso (Ciências da Computação), Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2005.

TECH FAQ. What is SHA-1? Disponível em: <http://www.tech-faq.com/what-is-sha-1.shtml>. Acesso em: 23 nov. 2009.

APÊNDICE

Transcreve-se abaixo o código de desenvolvimento da applet que é gravada no cartão:

```
package javacardpac;


import javacard.framework.APDU;
import javacard.framework.Applet;
import javacard.framework.ISO7816;
import javacard.framework.ISOException;
import javacard.framework.JCSystem;
import javacard.framework.OwnerPIN;
import javacard.framework.Util;


public class AppletSave extends Applet {


  // CONSTANTES APDU
  private final static byte APPLET_CLA = (byte)0x33;
  private final static byte VERIFY_INS = (byte)0x20;
  private final static byte VERIFY_INI_INS = (byte)0x21;
  private final static byte READ_BINARY_INS = (byte) 0xB0;
  private final static byte WRITE_BINARY_INS = (byte) 0xD0;
  private final static byte UPDATE_PIN_INS = (byte) 0x60;
```

```
        private final static byte PIN_TRY_LIMIT = (byte) 0x09; // maximo de tentativas
        private final static byte MAX_PIN_SIZE = (byte) 20;// tamanho maximo (SHA1)
        private final static byte MIN_PIN_SIZE = (byte) 20;// tamanho minimo (SHA1)
        private final static short SW_VERIFICATION_FAILED = (short) 0x6300;
        private final static short SW_PIN_VERIFICATION_REQUIRED = (short) 0x6301;
        private final static short SW_PIN_TO_LONG = (short) 0x6E86;
        private final static short SW_PIN_TO_SHORT = (short) 0x6E87;
        private final static short SW_MAX_PIN_TRY_LIMIT = (short) 0x6E89;
        private final static short SW_PIN_BLANK = (short) 0x6E88;


        // variáveis de escopo
        private byte[] dados = null;
        private OwnerPIN ownerPin;
        private boolean iniciouPIN = false;


        // construtor
        public AppletSave() {
            ownerPin = new OwnerPIN(PIN_TRY_LIMIT, MAX_PIN_SIZE);
            register();  //  Register this applet instance with the JCRE.
        }
```

```
        // métodos padrão da Applet
        public static void install(byte[] bArray, short bOffset, byte bLength) { //
chamado 1 vez, ao instalar a applet no card
                new AppletSave();
        }
        public boolean select() { //  Perform any applet-specific session initialization.
           return true;
        }
        public void deselect() {// sair
           ownerPin.reset();
        }



        // verifica se existe senha gravada
        private void verifyIni(APDU apdu) {
                if(!iniciouPIN) {
                        ISOException.throwIt(SW_PIN_BLANK);
                        return;
                }
        }

        // verificação do PIN
        private void verify(APDU apdu) {
                if(!iniciouPIN) {
```

```
                ISOException.throwIt(SW_PIN_BLANK);
                return;
            }
            if(ownerPin.getTriesRemaining()==0){
                ISOException.throwIt(SW_MAX_PIN_TRY_LIMIT);
                return;
            }
            byte[] buffer = apdu.getBuffer();
            // receive the PIN data for validation.
            byte byteRead = (byte)(apdu.setIncomingAndReceive());
            // check pin
            // the PIN data is read into the APDU buffer
            // starting at the offset ISO7816.OFFSET_CDATA
            // the PIN data length = byteRead
            if (ownerPin.check(buffer, ISO7816.OFFSET_CDATA,byteRead) ==
false) ISOException.throwIt(SW_VERIFICATION_FAILED);
        }

        // alteração do PIN
        private void updatePin(APDU apdu) {
            if (iniciouPIN && !ownerPin.isValidated())
ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);

            byte[] buffer = apdu.getBuffer();
```

```
// get the number of bytes in the
// data field of the command APDU -- OFFSET_LC = positon 4
byte numBytes = buffer[ISO7816.OFFSET_LC];

// recieve data
// data are read into the apdu buffer
// at the offset ISO7816.OFFSET_CDATA
byte byteRead = (byte)(apdu.setIncomingAndReceive());

// error if the number of data bytes
// read does not match the number in the Lc byte
if (byteRead != numBytes)
ISOException.throwIt(ISO7816.SW_WRONG_LENGTH);

if ( numBytes > MAX_PIN_SIZE )
ISOException.throwIt(SW_PIN_TO_LONG);

if ( numBytes < MIN_PIN_SIZE )
ISOException.throwIt(SW_PIN_TO_SHORT);

short offset_cdata = 05;
ownerPin.update(buffer, offset_cdata, numBytes);
ownerPin.resetAndUnblock();
```

```
            iniciouPIN = true;
        }

        // leitura das informações gravadas
        private void readBinary(APDU apdu) {
            if                                          (!ownerPin.isValidated())
ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);
            // inform the JCRE that the applet has data to return
            apdu.setOutgoing();// short le = apdu.setOutgoing();

            if(dados==null) dados = new byte[0];

            byte tamDados = (byte)dados.length;

            // set the actual number of the outgoing data bytes
            apdu.setOutgoingLength(tamDados);

            // send the 2-byte balance at the offset
            // 0 in the apdu buffer
            apdu.sendBytesLong(dados,(short)0, tamDados);
        }

        // escrita das informações
        private void writeBinary(APDU apdu) {
```

```
            if (!ownerPin.isValidated())
ISOException.throwIt(SW_PIN_VERIFICATION_REQUIRED);
            try {
                byte[] buffer = apdu.getBuffer();
                apdu.setIncomingAndReceive();
                dados = new byte[buffer[ISO7816.OFFSET_LC]];
                Util.arrayCopy(buffer, ISO7816.OFFSET_CDATA, dados,
(short)0, buffer[ISO7816.OFFSET_LC]);
                JCSystem.beginTransaction();
                JCSystem.commitTransaction();
            } catch (ISOException e) {
                JCSystem.abortTransaction();
                ISOException.throwIt(e.getReason());
            } catch (Exception e1) {
                JCSystem.abortTransaction();
                ISOException.throwIt(ISO7816.SW_UNKNOWN);
            }

        }

        // método executado ao receber os comandos APDU
        public void process(APDU apdu) throws ISOException {

           // Get the incoming APDU buffer.
```

```
        byte[] buffer = apdu.getBuffer();

        //  If INS is Select, return - no need to process select
        if ((buffer[ISO7816.OFFSET_CLA] == 0) && (buffer[ISO7816.OFFSET_INS]
== (byte)(0xA4)) ) return;

        //  If unrecognized class, return "unsupported class."
        if (buffer[ISO7816.OFFSET_CLA] != APPLET_CLA)
ISOException.throwIt(ISO7816.SW_CLA_NOT_SUPPORTED);//


        //  Process (application-specific) APDU commands aimed at
        switch (buffer[ISO7816.OFFSET_INS]) {
           case VERIFY_INS: verify(apdu);break;
           case VERIFY_INI_INS: verifyIni(apdu);break;
           case READ_BINARY_INS: readBinary(apdu);break;
           case WRITE_BINARY_INS: writeBinary(apdu);break;
           case UPDATE_PIN_INS: updatePin(apdu);break;
           default:
ISOException.throwIt(ISO7816.SW_INS_NOT_SUPPORTED);break;
        }
      }

    }
```