UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

ANDRÉ MARQUES DA SILVA
ÉDER JOSÉ PELEGRINI
FLÁVIO URSCHEI
RODRIGO RAGE FERRO

Projeto SysSec
Controle de Acesso baseado em biometria digital, smartcard e vídeo digital

São Paulo
2005

ANDRÉ MARQUES DA SILVA
ÉDER JOSÉ PELEGRINI
FLÁVIO URSCHEI
RODRIGO RAGE FERRO

Projeto SysSec
Controle de acesso baseado em biometria digital, smartcard e vídeo digital

Monografia apresentado à disciplina
PCS2502 – Projeto de Formatura II
da Escola Politécnica da Universidade de São Paulo

Área de Concentração:
Engenharia Elétrica
Ênfase em Computação e Sistemas Digitais

Orientador:
Prof. Dr. André Riyuiti Hirakawa

São Paulo
2005

FICHA CATALOGRÁFICA

Ferro, Rodrigo Rage.
Pelegrini, Éder José.
Silva, André Marques da.
Urschei, Flávio.

Projeto SysSec - Controle de Acesso baseado em Biometria Digital, Smartcard e Vídeo Digital; orientador: Prof. Dr. André Riyuiti Hirakawa; São Paulo, 2005. 154p.

Monografia apresentada à disciplina PCS2502 – Projeto de Formatura II, da Escola Politécnica da Universidade de São Paulo.

1. Biometria Digital 2. Smartcard 3. Vídeo Digital 4. Controle de Acesso

EPÍGRAFE

"A diferença entre o sonho e a realidade é a quantidade certa de tempo e trabalho".

William Douglas

AGRADECIMENTOS

Primeiramente a Deus, por nos dar força e coragem para enfrentar os cinco anos de Escola Politécnica da Universidade de São Paulo, permitindo hoje finalizar mais uma etapa de nossas vidas.

Ao amigo e orientador Prof. André Riyuiti Hirakawa, pela idéia do projeto, pelo apoio, pela paciência, pelos conselhos nas horas mais complicadas quando parecia que o sistema não sairia do papel. Somos bastante gratos ao professor que com sua tranqüilidade nos transmitiu confiança e mostrou que nada é impossível quando existe esforço e dedicação.

À Empresa Philips pela palestra sobre o cartão MIFARE e pelo leitor de *smartcard* fornecidos.

À FUSP (Fundação de Apoio à Universidade de São Paulo) pela verba para compra do leitor biométrico.

Ao LSA (Laboratório de Sistemas Abertos) que nos deu a infraestrutura necessária para que conseguíssemos realizar esse projeto com sucesso.

Às nossas famílias, pela compreensão e apoio que nos deram durante essa jornada do conhecimento.

Temos certeza que agradecemos a uma pequena parte das pessoas a quem devemos muito. Pedimos desculpas àquelas que omitimos.

RESUMO

Ferro, Rodrigo Rage; Pelegrini, Éder José; Silva, André Marques da; Urschei, Flávio. **Projeto SysSec - Controle de acesso baseado em biometria digital, *smartcard* e vídeo digital.** 2005. 154p. Monografia (Projeto de Formatura), Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.

O número alarmante de fraudes e furtos que afligem a sociedade brasileira requer medidas drásticas para cercear as ações dos delinqüentes. Na tentativa de saciar esse anseio por medidas de segurança, métodos para controlar acesso ganham cada vez mais notoriedade nos meios acadêmico e comercial. Aplicações envolvendo segurança são cada vez mais freqüentes, não só por questões de violência, mas também por razões corporativas (espionagem industrial). Dentre as diversas aplicações de segurança disponíveis encontram-se aquelas cujo objetivo é controlar o acesso a locais onde há a necessidade inerente de proteção (laboratórios de pesquisa, instituições financeiras, entre outros).

Há diversos meios de se restringir o acesso a um determinado local, podendo ser usados cartões de acesso, senhas ou identificação por alguma característica única do indivíduo (biometria). Para todos esses métodos de identificação há vantagens e desvantagens, sendo alguns mais eficientes que outros. Analisando-se as diferentes técnicas existentes, verifica-se que a eficiência no controle de acesso não pode ser atingida apenas com o uso de uma técnica de identificação e, sim, por uma combinação delas. *Smartcard* ou mesmo biometria não são impossíveis de serem burlados, mas a integração de ambos diminui a probabilidade de que isso ocorra. Esse é o principal objetivo desse projeto: desenvolver um protótipo que integre *smartcard*, biometria por meio de impressões digitais e sincronização de vídeo digital de forma que o nível de acesso seja restringido e as pessoas que acessem um determinado local sejam identificadas.

Palavras-Chave: Biometria Digital, *Smartcard*, Vídeo Digital, Controle de Acesso.

ABSTRACT

Ferro, Rodrigo Rage; Pelegrini, Éder José; Silva, André Marques da; Urschei, Flávio – **Project SysSec – Access control based on fingerprint biometry, smartcard and digital video**. 2005. 154p. Monograph (Graduate Project), Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.

Due to a great deal of fraud and theft which afflicts Brazilian society, extreme measures are required to restrict any transgressor's action. To satisfy this need for security, access control systems are gaining more and more notoriety in academic and commercial environments. Applications involving security are increasingly found, not only regarding violence matters, but also due to corporate reasons (industrial espionage). Amongst several available security applications there are a few that aim to control the access to places where there is an inherent need for protection (research laboratories, financial institutions, among others).
There are several means of controlling the access in a specific place, such as: access cards, passwords and identification based on some individual feature (biometry). To all of these identification methods there are advantages and disadvantages, some of them being more efficient than others. By analyzing different existing techniques, it is verified that the efficiency of access control is not able to be achieved only by using one technique, but with the combination of many. Smartcard or biometry are not immune from deception, however the integration of both decreases the likelihood that problems will arise happens. This is the main purpose of this project: to develop a prototype that joins smartcard and biometry by means of fingerprint and video synchronization so that the level of access is restricted and people who have access to a specific place are identified.

Keywords: Fingerprint Biometry, Smartcard, Digital Video, Access Control.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES E SIGLAS

ASCII	-	*American Standard Code for Information Interchange*
API	-	*Application Program Interface*
ATR	-	*Answer to Request*
ATS	-	*Answer to Select*
BD	-	Banco de Dados
BIOS	-	*Basic Input Output System*
BPS	-	*Bits* Por Segundos
DMA	-	*Direct Memory Access*
CD	-	*Compact Disk*
CLR	-	*Common Language Runtime*
CMYK	-	*Cyan, Magenta, Yellow, Black*
CRC	-	*Cyclical Redundancy Check*
DCT	-	*Discrete Cosine Transform*
DLL	-	*Dynamic Linked Library*
DPI	-	*Dots Per Inch*
ECP	-	*Extended Capability Port*
EEPROM	-	*Electrically Erasable Programmable Read Only Memory*
EPC	-	*Event-Driven Process Chain*
EPP	-	*Enhanced Parallel Port*
FAR	-	*False Acceptance Rate*
FBI	-	*Federal Bureau of Investigation*
FIR	-	*Fingerprint Identification Record*
FRR	-	*False Reject Rate*
FUSP	-	Fundação de Apoio à Universidade de São Paulo
GSM	-	*Global System for Mobile Communications*
HIS	-	*Hue, Intensity, Saturation*
IDE	-	*Integrated Development Enviroment*
IDEF	-	*Integrated Definition Methods*
IP	-	*Internet Protocol*
ISO	-	*International Organization for Standardization*
J2SE	-	Java 2 *Standard Edition*
JMF	-	Java *Media Framework*
LED	-	*Light Emitting Diodes*

LPT	-	*Line Print Terminal*
LSA	-	Laboratório de Sistemas Abertos
KB	-	*Kilo Byte*
MB	-	*Mega Byte*
MF-RD	-	*MiFare ReaDer*
MFC M200	-	MIFARE *Micro Module*; nome formal: MMM
MPEG	-	*Moving Picture Experts Group*
MS-SQL	-	*MicroSoftware Structured Query Language*
PC	-	*Personal Computer*
PCB	-	*Printed Circuit Board*
PDF	-	*Portable Document Format*
PMBoK	-	*Project Management Body of Knowledge*
PMI	-	*Project Management Institute*
RF	-	*Radio Frequency*
RGB	-	*Red, Green, Blue*
RLE	-	*Run Length Encoding*
SADT	-	*Structured Analysis and Design Technique*
SDK	-	*Software Development Kit*
SGBD	-	Sistema Gerenciador de Banco de Dados
SPP	-	*Standard Parallel Port*
SQL	-	*Structured Query Language*
TCP	-	*Transmission Control Protocol*
UML	-	*Unified Modeling Language*
USB	-	*Universal Serial Bus*
VLSI	-	*Very Large Scale Integration*
VFW	-	*Video For Windows*
WBS	-	*Work Breakdown Structure*
WDM	-	*Windows Driver Model*
WOM	-	*Write Only Memory*
YUV	-	*"Y" Luminance, "U" axis (blue), "V" axis (red)*

SUMÁRIO

1 Introdução

1.1 Objetivos do Projeto

O projeto SysSec tem por objetivo projetar e implementar um protótipo de sistema de controle de acesso a partir da combinação de duas técnicas de identificação: cartões de acesso inteligentes (*smartcards*) e identificação por biometria de impressões digitais. Aliado a essas técnicas, será implementado um sistema de vídeo de *log* cujo objetivo é prover o histórico de acesso, permitindo a posterior identificação das pessoas que acessaram o local, cujo acesso está sendo controlado.

Resumidamente, o sistema desempenhará o seguinte funcionamento:

O usuário da sala/laboratório aproxima seu cartão (*smartcard*) do dispositivo de leitura disparando o primeiro nível de identificação, no qual o cartão é validado no sistema. Em caso de identificação positiva do cartão, o sistema requisita que o usuário forneça suas digitais (segundo nível de identificação), as quais são digitalizadas por meio de um leitor. O sistema valida as impressões digitais e as compara com as previamente cadastradas, liberando ou não o acesso. Se o acesso for permitido, o sistema inicia a gravação do vídeo de *log* (por meio de uma câmera digital), o qual será armazenado. Em seguida será acionada a fechadura eletrônica.

Além disso, o sistema possuirá outras funcionalidades que fornecerão subsídios ao processo de identificação descrito, tais como: o gerenciamento dos usuários (dados cadastrais, cartão de acesso e impressões digitais) do sistema, consulta e exibição dos vídeos de *log* e controle do processo de identificação.

1.2 Motivação

A necessidade de aplicações envolvendo segurança é cada vez mais freqüente, não só por questões de violência, mas também por razões coorporativas (espionagem industrial). Dentre as diversas aplicações de segurança disponíveis encontram-se aquelas cujo objetivo é controlar o acesso a locais onde há a necessidade inerente de proteção (por exemplo, laboratórios de pesquisa e instituições financeiras). Há diversos meios de se restringir o acesso a um determinado local, podendo ser usados cartões de acesso, senhas ou identificação por alguma característica única do individuo (biometria). Para todos esses métodos de identificação há vantagens e desvantagens, sendo alguns mais eficientes que outros.

Analisando-se as diferentes técnicas existentes, verifica-se que a eficiência no controle de acesso não pode ser atingida apenas com o uso de uma técnica de identificação e, sim, por uma combinação delas.

Sendo assim, a realização deste projeto foi impulsionada pelas vantagens proporcionadas ao se combinar diferentes técnicas de identificação. No caso desse projeto, biometria por impressões digitais, cartões de acesso inteligentes e histórico de acesso por meio de vídeo de *log*.

A identificação biométrica por impressões digitais possui a vantagem de ser uma técnica segura que elimina a necessidade de memorização de *logins* e senhas, pelo fato de estar baseada em uma característica presente em todas as pessoas (Universalidade – todas as pessoas possuem impressões digitais), ser única para cada indivíduo (Singularidade – as impressões digitais são únicas) e dificilmente sofrem alterações com o passar do tempo (Permanência – as impressões digitais raramente se alteram), sendo difícil violar sistemas que utilizem essa técnica. Além disso, as impressões digitais podem ser facilmente mensuradas e, em geral, essa forma de identificação é bem vista pelos usuários.

O controle de acesso por meio de cartões inteligentes (*smartcards*) é uma técnica segura de identificação, pois possui características como: dificuldade de se alterar informações contidas nos cartões (alto nível de proteção nas informações contidas no cartão), possibilidade de agregar informações nos cartões (um cartão pode ser a chave de acesso para diversos sistemas de segurança), também dispensa o uso de senhas e possui alto poder de durabilidade. Além disso, pode-se agregar essa técnica de identificação à técnica biométrica armazenando-se as informações representativas das impressões digitais no próprio cartão, diminuindo-se assim o risco das informações biométricas serem burladas (segurança da informação do cartão). Há também a vantagem de se acelerar o processo de identificação.

A técnica de armazenamento do histórico de acesso por meio de vídeo de *log* não adiciona segurança no processo de identificação propriamente dito, não obstante possibilita futuras identificações dos usuários. Esse recurso não só intimida o infrator, como também pode servir como subsídio para a implementação de um outro sistema de segurança envolvendo reconhecimento de face por meio da análise dos vídeos.

Aliado às vantagens supracitadas, outro fator motivante do projeto é a expansão do mercado de segurança, que reforça a viabilidade econômica e a aplicabilidade prática. Os seguintes dados demonstram essa idéia:

- O mercado de biometria (*hardware*, *software* e serviços) registrou crescimento mundial de 85,3% em relação a 2002, alcançando volume de cerca de US$ 830 milhões [1];
- Segundo o Caderno de Negócios do jornal "O Estado de São Paulo" estima-se que, em 2005, houve um crescimento de até 60% no faturamento das empresas que vendem equipamentos empregando a tecnologia de reconhecimento biométrico no Brasil [2];
- De acordo com o IBG (*International Biometric Group*), as tecnologias baseadas em impressão digital, tidas atualmente como as de maior eficácia, projetam uma taxa alta de crescimento. Só em 2003 houve uma previsão de que 52% do mercado total (mundial) seria ocupado por tais tecnologias [3];
- Segundo [4], a divulgação crescente dos tipos de tecnologias biométricas e a preocupação cada vez maior com segurança são os principais catalisadores deste mercado. Entretanto, ainda existe desconfiança em relação à utilização de identificação biométrica;
- Segundo [5], dentre as diferentes técnicas de reconhecimento biométrico utilizadas pelo mercado, destaca-se o reconhecimento por impressão digital como sendo o segmento mais difundido (figura 1).

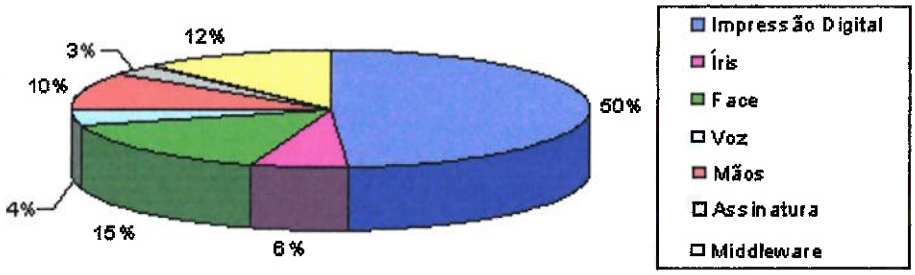

Figura 1 - Mercado Biométrico X Sistemas Biométricos (extraído de [5]).

1.3 Estrutura do Documento

O corpo principal deste documento é composto por 5 capítulos, 1 anexo e 2 apêndices:

1. Introdução: este capítulo apresenta uma breve descrição dos objetivos do projeto, assim como a motivação que norteou seu desenvolvimento;

2. Conceitos e Tecnologias/Produtos: este capítulo encerra os principais conceitos e tecnologias abordados no projeto. São abordados aspectos conceituais sobre biometria digital, vídeo digital, compressão de vídeo, *smartcards,* arquitetura cliente-servidor, arquitetura de software, IDEF0, threads e interface paralela. As tecnologias abordadas são: Java, JMF, C#.NET, MIFARE, leitor MF-RD260, CREATIVE *Webcam* NX PRO e *Scanner* Biométrico *FingKey Hamster* (HFDU01) e SDK (*software development kit*) NITGEN;

3. Planejamento e Metodologia: este capítulo aborda as técnicas de gestão de projetos e metodologias de desenvolvimento de software utilizadas no projeto;

4. Especificações do projeto: este capítulo apresenta as especificações do projeto. Encerra uma descrição funcional do projeto, dos diferentes modelos utilizados, da arquitetura de *software* e da arquitetura de *hardware*;

5. Projeto e Implementação: este capítulo apresenta todos os passos e detalhes de implementação. Encerra uma descrição da implementação do banco de dados, do módulo de vídeo, do módulo de biometria, do módulo do cartão, do módulo de gerência e dos passos de integração e testes;

6. Considerações Finais: esse capítulo encerra as considerações finais do trabalho. Encerra aspectos como: problemas encontrados, resultados obtidos, considerações sobre o aprendizado com o projeto, validade técnica da solução proposta e possíveis aperfeiçoamentos do projeto;

Anexo 1: apresenta a especificação de casos de uso do sistema;

Apêndice 1: apresenta o manual do usuário para a utilização do sistema;

Apêndice 2: apresenta a história da dactiloscopia;

Apêndice 3: apresenta o material eletrônico.

2 Conceitos e Tecnologias/Produtos

2.1 Conceitos

Esta seção apresenta uma revisão bibliográfica dos principais conceitos utilizados durante o desenvolvimento do projeto.

2.1.1 Biometria – Impressão Digital

A crescente demanda por sistemas de segurança e a necessidade de melhorar os sistemas já existentes fazem com que métodos para controlar acesso a locais restritos ganhem cada vez mais notoriedade nos meios acadêmico e comercial. A biometria surge como uma opção para resolver esse problema.

A cada ano mais empresas estão adotando sistemas de segurança com biometria. A grande vantagem dos sistemas biométricos sobre os demais (por exemplo, baixo índice de fraudes e a facilidade de utilização [6]) e a crescente necessidade por mais segurança tornarão esses tipos de sistema padrões de mercado no futuro. Aplicações envolvendo biometria já são bastante comuns atualmente: segurança em transações financeiras, em sistemas civis (por exemplo, em carteira de identidade); em sistemas criminais para verificar quem realizou um disparo de arma de fogo e, principalmente, em sistemas de controle de acesso.

Segundo dicionário *online Michaelis* [7], biometria é a "ciência da aplicação de métodos de estatística quantitativa a fatos biológicos; análise matemática de dados biológicos", em outras palavras, métodos que identificam um indivíduo baseado em suas características físicas únicas (por exemplo: impressão digital, identificação de íris, reconhecimento de voz, geometria da mão, a projeção facial e até a caligrafia).

Para que algumas características físicas sejam potencialmente utilizáveis em sistemas biométricos devem ser obedecidos alguns princípios [8]:

- **Universalidade:** cada pessoa é classificada por meio de um conjunto de características;
- **Unicidade (variabilidade):** cada pessoa tem suas características individuais (únicas). Por exemplo: as impressões digitais variam de uma pessoa para outra, ou de um dedo

para outro. Portanto, uma impressão digital somente é igual a ela mesma, não podendo ter duas pessoas com a mesma impressão digital;

- **Permanência (imutabilidade):** as características não podem se alterar no decorrer do tempo (haver modificação);
- **Critério quantitativo:** características podem ser medidas quantitativamente, apresentando um valor que pode ser usado para diferenciar duas pessoas.

Apesar de nestes anos recentes ter havido um considerável progresso em tecnologias biométricas baseadas no reconhecimento de íris e reconhecimento facial, biometria baseada em impressões digitais ainda continua sendo a tecnologia biométrica mais difundida em projetos biométricos por todo o mundo. A individualidade da impressão digital é amplamente reconhecida e vem sendo utilizada há bastante tempo (vide seção "História da dactiloscopia").

História da dactiloscopia

A dactiloscopia é um método de identificação humana baseada em impressões digitais. Esta técnica é bastante antiga, mas apesar disso ainda é prática, segura e econômica [9]. Um histórico da evolução (no Brasil e no mundo) desse método e de alguns processos de identificação segue no apêndice. Esta seção foi baseada no trabalho de [9] e [10].

Características Únicas

A impressão digital é uma característica física única de cada ser humano (não é a mesma nem para gêmeos univitelinos), oferecendo um método claro e não ambíguo para identificá-lo. Ela é constituída por uma superfície que apresenta franjas (linhas datilares) denominadas *ridges* localizadas nas pontas dos dedos.

As franjas, em seu trajeto de um lado ao outro da impressão digital, podem interromper-se ou bifurcar-se formando pontos característicos: fins-de-linha (*ending ridges*) e bifurcações (*bifurcation ridges*). Esses pontos característicos são denominados minúcias (*minutiaes*) e a relação espacial entre elas (ocorrência e distribuição planar) caracteriza, de forma única, um indivíduo. Além das minúcias, muitos sistemas utilizam outros parâmetros encontrados na impressão digital para identificação como: núcleos e deltas, maiores detalhes em [9].

Em sistemas automáticos de identificação de impressões digitais (AFIS - *Automated Fingerprint Identification System*), geralmente, informações como posição (coordenadas), direção (orientação das *ridges*) e tipo das minúcias (crista final ou bifurcação) são guardadas

em forma numérica no banco de dados para a representação da impressão (imagem capturada). A verificação, então, é feita comparando-se os valores obtidos a partir de uma impressão digital capturada com os dados armazenados no banco de dados (armazenados no momento do cadastro) [9].

Procedimentos de aquisição de impressões digitais

Para adquirir imagens de impressões digitais existem dois procedimentos: a impressão tintada em papel (método *ink and paper*, onde o dedo sujo de tinta é rolado de um lado ao outro em um papel), bastante usado pelas autoridades no país em sistemas civis ou criminais e outro procedimento é o uso de um sistema eletrônico de geração de dados (aquisição de impressão digital por meio de leitores eletrônicos biométricos). No primeiro caso, a verificação manual de impressão digital apresenta alto custo, além de consumir bastante tempo, justificando o uso do segundo procedimento e de sistemas de identificação de impressão digital automatizados.

Leitor ótico biométrico

Visando automatizar o processo de captura de impressões digitais, estão sendo utilizados cada vez mais leitores biométricos. Eles juntamente com *softwares* especializados são responsáveis por transformar os aspectos físicos da impressão digital em um *template*, ou seja, um conjunto de características extraídas da imagem (por exemplo, extração de minúcias). Existem atualmente vários tipos de leitor: leitores capacitivos, os leitores óticos, entre outros.

No caso do leitor biométrico ótico (utilizado neste projeto), a imagem do dedo é capturada por meio do seguinte processo: a parte do leitor onde o dedo fica em contato corresponde a uma das faces de um prisma a luz. Por meio de reflexões, a imagem então é projetada pela outra face do prisma e capturada por meio de uma câmera localizada na terceira face do prisma. A câmera consegue capturar a imagem porque as partes do dedo que estão em contato com o vidro causam uma alta diferença de contraste que é detectada pela câmera. Em geral, usa-se um sistema de lentes para corrigir a imagem (distorção). Normalmente, os leitores óticos se conectam ao computador por meio de uma interface USB (como no caso deste projeto), ou *firewire* [6].

Funcionamento de um sistema AFIS

Com a implantação de um sistema AFIS (sistema automático de identificação de impressão digital), o processo de identificação biométrica é agilizado, pois não requer o trabalho manual de um especialista.

Em geral, o sistema AFIS possui estágios como: captura, extração de minúcias, comparação e verificação/identificação do indivíduo, conforme apresentado na figura 2 [9].

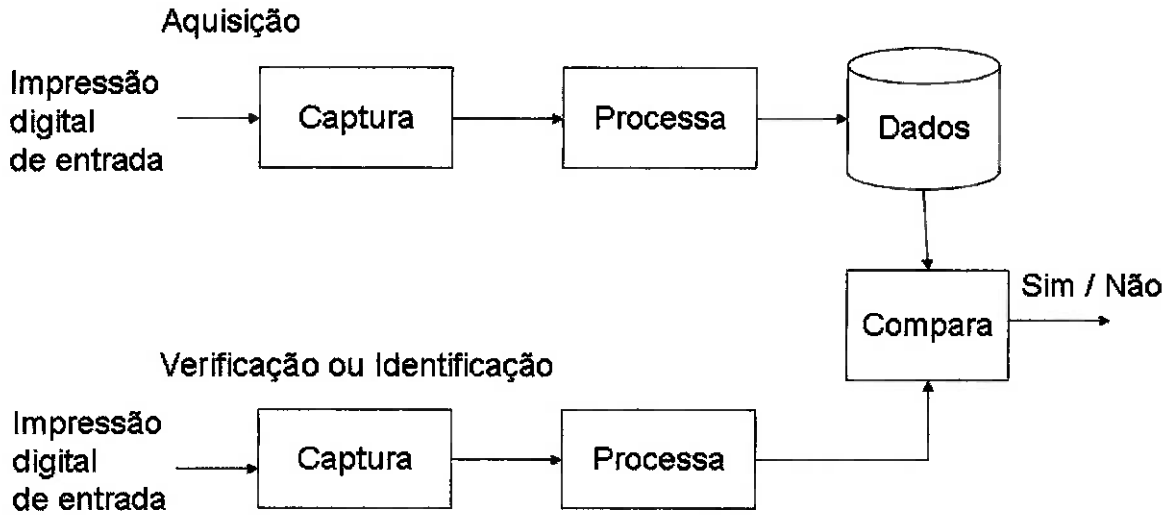

Figura 2 - Estágios de um AFIS – obtida de [9]

Antes de ocorrer a identificação/verificação, para o indivíduo ter acesso permitido a um determinado local, ele deve estar cadastrado no sistema biométrico. Para isso o sistema deve adquirir (capturar) a impressão digital do dedo do usuário. O sistema, então, cria um *template* com informações biométricas que serão usadas como referência para o processo de identificação/verificação. Em geral, é criado o *template* de forma a evitar que seja gravada no banco de dados uma imagem (otimizando espaço). Normalmente, no cadastramento são capturadas várias amostras de impressão digital, sendo que a melhor delas será o *template* a ser cadastrado e utilizado como referência.

Quando o usuário solicita a verificação/autenticação, sua característica física, isto é, a impressão digital é capturada pelo sensor. A informação é, então, comparada com o modelo biométrico armazenado no cadastramento. Se a comparação é tida como verdadeira, então o indivíduo será reconhecido pelo sistema e terá acesso permitido.

A comparação supracitada é sobre minúcias, cujo processo de extração (os passos mais importantes) está descrito na seção seguinte. A comparação de minúcias está mostrada na figura 3.

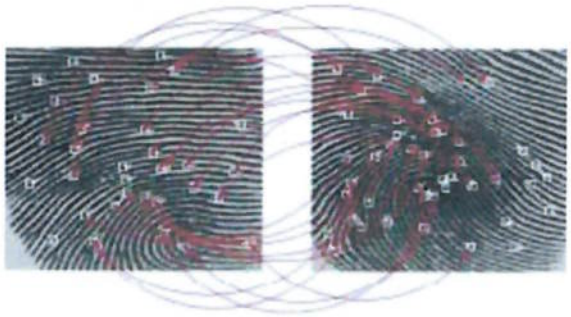

Figura 3 - Comparação de minúcias (cristas finais e cristas bifurcadas), extraída de [9].

Obtenção das Informações Biométricas (extração de minúcias)

Na maioria dos sistemas AFIS (sistemas automáticos de identificação de impressões digitais), alguns passos, geralmente, ocorrem nos algoritmos para extração de informações biométricas (extração de minúcias) de uma imagem de impressão digital (a ser capturada pelo leitor). Passos como: captura da imagem, melhoramento da imagem (uso de filtros), esqueletização (afinamento das franjas – *thinning*) e extração de minúcias. Não se pode fazer maiores comentários sobre os métodos/técnicas que foram utilizados(as) em cada passo no *software* fornecido para este projeto, pois essas informações não foram disponibilizadas pelo fabricante.

- **Captura da imagem:** nesse passo, a imagem é capturada e convertida para o formato digital e enviada para o *software*, que realizará as demais etapas de processamento de imagens (para extração de informações biométricas). A figura 4 mostra um exemplo de uma imagem capturada.

Figura 4 - Imagem capturada, extraída de [6]

- **Melhoramento das imagens:** em geral, diversos algoritmos de processamento de imagens são aplicados com a finalidade de realçar as papilas na imagem. Esses algoritmos tentam diminuir o nível de ruído nas imagens (filtro passa-baixa) e realçar os contornos/fronteiras da imagem (filtro passa-alta). A figura 5 mostra uma imagem melhorada, após passar pelos filtros.

Figura 5 - Imagem melhorada, extraída de [6]

- **Esqueletização da imagem:** neste passo, as papilas são afinadas até ter um *pixel* de comprimento. O objetivo é facilitar a extração das minúcias. É comum utilizar a transformada *hit-or-miss* para realizar o afinamento (*thinning*) [9].

Figura 6 - Imagem esqueletizada, extraída de [6]

- **Extração de minúcias:** neste passo, os *pixels* da imagem são analisados por meio da busca de minúcias. Em alguns sistemas, aplicam-se também algoritmos complexos para obtenção de núcleo e delta.

Figura 7 - Imagem com as minúcias, extraída de [6]

- **Inserção ou busca no banco de dados:** as minúcias são extraídas e catalogadas de acordo com relações espaciais/matemáticas existentes entre elas. Os valores calculados a partir dessas relações são armazenados no banco de dados e são utilizados para realizar comparações (visando, por exemplo, a identificação/verificação biométrica de um indivíduo).

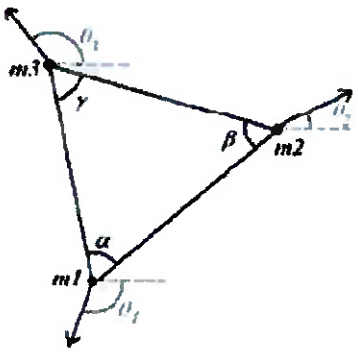

Figura 8- Relações matemáticas, extraída de [6]

FAR versus FRR

Para sistemas biométricos comerciais, o desempenho no processo de verificação deve ser considerado. Ele é em geral, medido por duas variáveis: a taxa de falsa aceitação – FAR (*False Acceptance Rate*) e a taxa de falsa rejeição – FRR (*False Reject Rate*). Devido às variações/erros no processo de identificação/verificação por problemas na captura, ruído nas imagens, fatores externos como sujeira no dedo, excesso ou pouca força aplicada pelo dedo no sensor, entre outros, existe uma probabilidade de um impostor ser aceito na verificação biométrica (FAR, variando de 0 a 100%) ou um indivíduo cadastrado no sistema não ser aceito (FRR, variando de 0 a 100%). Para saber os valores dessas variáveis, basta tomar o conjunto total de amostras que foram submetidas ao processo de identificação / verificação e saber quantos indivíduos cadastrados foram rejeitados e quantos impostores foram aceitos.

FAR e FRR são dependentes uma da outra, isto é, normalmente, quando há uma pequena FRR, ocorre uma alta FAR e vice-versa [9]. Geralmente, pode-se configurar os sistemas biométricos para serem mais (baixa taxa FAR e alta FRR) ou menos sensíveis (alta taxa FAR e baixa FRR).

Esses valores estão relacionados com o processo de comparação dos dados biométricos, isto é, minúcias. Caso se deseje um sistema mais sensível, o número de minúcias que devem combinar (das extraídas de uma imagem capturada versus as cadastradas no banco de dados para um determinado dedo) deve ser maior (aumento do nível de segurança) e vice-versa vale para o caso de sistemas menos sensíveis (detecção fraca).

No caso do sistema biométrico utilizado nesse projeto, o fabricante informa que o algoritmo usado possui FAR de 0,0% e FRR de 0,0% [11].

2.1.2 Vídeo Digital

Um vídeo é constituído por uma seqüência de imagens exibidas em uma taxa suficientemente alta (acima de 24 imagens por segundos) de modo a criar a sensação de continuidade (movimento) [12]. Sendo assim, um vídeo é simplesmente um conjunto de imagens (domínio espacial) exibidas ao longo do tempo (domínio temporal).

O domínio espacial do vídeo é caracterizado pela imagem digital. Uma imagem digital é uma função bidimensional f(x,y), em que:

- x representa a dimensão horizontal;
- y representa a dimensão vertical;
- f(x,y) representa a cor na coordenada espacial (x,y).

Para que se gere uma imagem digital, deve-se realizar uma amostragem espacial ao longo das dimensões horizontal e vertical, assim como na amplitude z igual a f(x,y). Em geral, o processo de amostragem é uniforme ao longo do domínio espacial e resulta numa matriz de amostras, onde cada elemento da matriz é chamado de *pixel* (*picture elements*). A matriz de amostras (ou matriz de *pixels*) é frequentemente usada para a representação de uma imagem.

A qualidade de uma imagem digital está relacionada com a quantidade de *pixels* utilizada para a representação de uma imagem (resolução da imagem) e com a quantidade de informação que um *pixel* pode agregar (profundidade da imagem). A forma de representação de um *pixel* está relacionada com a base de cores utilizada. As bases de cores comumente

encontrados na literatura são: o RGB (existem variações que usam 8, 16, 24 e 32 *bits*), CMY, HSI e YUV [12].

A figura 9 permite verificar como a variação da resolução espacial interfere na qualidade da imagem. Para tanto, foi fixada uma profundidade (256 níveis de cinza – imagem monocromática) e foi variada a resolução espacial da imagem. Note a degradação (representada pela visualização "quadriculada") sofrida pelas imagens devido à perda de resolução espacial (observe que o *pixel* começa a ficar perceptível).

Figura 9 - Imagens da Lenna de 256 níveis de cinza. (a) 256 x 256 pixels, (b) 128 x 128pixels, (c) 64 x 64 pixels. Extraída de [13].

A figura 10 permite verificar como a variação de profundidade interfere na qualidade da imagem. Para tanto, foi fixada a mesma resolução espacial (256 x 256 *pixels*) e foi variada a profundidade da imagem. Note que os detalhes da imagem original transformam-se gradativamente em regiões homogêneas de níveis de cinza.

Figura 10 - Imagens da Lenna de 256 x 256 pixels. (a) 256 níveis de cinza, (b) 16 níveis de cinza, (c) 2 níveis de cinza (imagem binária). Extraída de [13].

O domínio temporal é caracterizado pela seqüência de imagens ao longo do tempo. O número de imagens (ou quadros) exibidas por segundo (denominado taxa de quadros) é outro parâmetro que caracteriza a qualidade de um vídeo.

A gravação do vídeo de *log* no momento do acesso permite manter um histórico de controle que possibilita a identificação de falhas de segurança (como o acesso de duas ou mais pessoas a partir da identificação de uma única pessoa). Para atingir esta funcionalidade, o vídeo gravado deve ter qualidade suficiente em relação à profundidade, resolução dos quadros e da taxa de quadros.

Deve-se haver um compromisso entre o tamanho do arquivo de vídeo e a qualidade do mesmo. Um vídeo gravado na resolução de 352 x 288 pixels, com a base de cores RGB24 (3 *bytes* por *pixels*) a 30 quadros por segundo, ou seja, com qualidade de *Webcam* (considerado baixa), por dez segundos resulta em um arquivo com de vídeo de, aproximadamente, 87 MB.

2.1.3 Compressão de Vídeo

Considerando o formato de 30 quadros por segundo na resolução de 352 x 288 *pixels* no formato RGB24, descritos na seção anterior, obtêm-se os seguintes dados: um quadro desse tipo de vídeo, sem compressão, ocupa um espaço de 0,3 MB (352 x 288 x 3 *bytes*). Para 10 segundos de gravação (ou 300 quadros) o volume de disco ocupado por um vídeo chega a aproximadamente a 87 MB. Considerando a finalidade do vídeo (sistema de vídeo de *log*), o armazenamento de arquivos de vídeos com dezenas de mega *bytes* é inadequado ao sistema, em razão do alto custo da infra-estrutura necessária para armazená-los. Desse modo foi utilizada a técnica de compressão de vídeo para reduzir o tamanho do arquivo de vídeo.

Os compressores de vídeo realizam a redução do tamanho do vídeo baseando-se na remoção de informações redundantes ou de menor importância (compressão com perdas) e na compactação dos dados (compressão sem perdas). Para tanto, os compressores beneficiam-se das redundâncias no domínio temporal (correlação de valores do *pixel* entre quadros consecutivos), das redundâncias no domínio espacial (correlação de valores do *pixel* dentro de um mesmo quadro) e das características da percepção humana (a maior parte dos detalhes de uma imagem complexa em movimento não é perceptível aos seres humanos) [14].

Existem diversos tipos de compressores de vídeo no mercado. Ao escolher um, deve-se levar em consideração a qualidade do vídeo comprimido resultante, o índice de compressão e a velocidade de compressão (como neste projeto a duração do vídeo é curta, o requisito de velocidade de compressão será ignorado). Alguns compressores, como o H.261, realizam

grandes reduções no arquivo, em contrapartida, a qualidade do vídeo é degradada. Outros compressores, como os baseados em MPEG-4 permitem *bit-rate* (taxa de *bits* por unidade de tempo) variado, possibilitando taxas de reduções diversas, ficando a cargo do usuário o compromisso entre o índice de redução e a qualidade final do vídeo (a qualidade da compressão é determinada, dentre outros coisas, pela taxa de *bits* por unidade de tempo).

Para a compressão do vídeo no sistema SysSec, foi escolhido o compressor Xvid [15], especificamente a variante Koepi [16]. Este *codec* (codificador/decodificador) é amplamente utilizado por ser gratuito (tanto o codificador quanto o decodificador) e por fornecer um alto índice de compressão mantendo uma boa qualidade visual com relação ao vídeo original.

O *codec* Xvid implementa parte das especificações do MPEG-4. Utiliza o mesmo princípio de redução para vídeo usado no MPEG-1 [12][17], que consiste nos seguintes passos de compressão:

- Compensação de Movimento: responsável pela compressão do vídeo ao longo do tempo, por meio do uso de quadros preditivos (estes quadros contêm apenas a informação da diferença do quadro atual com o anterior, ou com o posterior, ou com os dois).
- DCT (Transformada do Cosseno Discreto): responsável, juntamente com a quantização (próximo passo) pela codificação intra-quadro (interna ao quadro). Converte um bloco de 8x8 *pixels* em um bloco de 8x8 coeficientes através da utilização da transformada DCT.
- Quantização: matriz de pesos aplicada sobre o bloco de 8x8 coeficientes da DCT, a qual elimina ou atenua os coeficientes de alta freqüência do bloco. A eliminação ou atenuação destes coeficientes são responsáveis pela redução do tamanho do vídeo.
- Compressão sem perdas: aplicação de algoritmos de compressão sem perdas. Os principais algoritmos dessa etapa são o *Run-Lenght Enconding* (RLE – que consiste na substituição de cadeias longas de valores repetidos por seu valor repetido e o número de ocorrências deste valor) e o algoritmo de Huffman (substituição de padrões recorrentes por um código mais curto – símbolos mais freqüentes são representados com menores números de *bits*).

Outras vantagens deste *codec* são: possibilidade de configurar parâmetros de codificação (exemplos: codificação em um ou dois passos, relação compressão-qualidade, escolha da matriz de quantização), possibilidade de operar sobre vídeos de diversas resoluções espaciais,

compatibilidade com o *codec* Divx (amplamente utilizado), e suporte ao formato AVI/Xvid por parte do *Windows Media Player* e por parte de alguns DVDs *Players* comerciais.

Utilizando esse compressor, um vídeo de 10 segundos, que antes ocupava 87 MB passa a ocupar um espaço da ordem de 250 KB (aproximadamente 0,28% do tamanho original) mantendo qualidade visual em relação ao vídeo original. Valores na ordem de centenas de kilo *bytes* são aceitáveis para os vídeos gerados para este projeto.

2.1.4 Smartcards

Smartcards ou cartões inteligentes são cartões que possuem um *chip* de memória e/ou microprocessador acoplado. As dimensões da maior parte desses cartões disponíveis são idênticas as de um cartão de crédito convencional.

Os cartões que possuem apenas o *chip* de memória, também chamados de *memory cards* (cartões de memória), não possuem poder significativo de processamento, isso porque em geral há apenas uma unidade aritmética para realização de cálculos simples como incremento e decremento de valores. Desse modo, tais cartões servem basicamente para armazenar dados.

Os cartões que possuem microprocessador, também chamados de *microprocessor cards* possuem poder de processamento, podendo armazenar dados e processá-los por meio de programas também armazenados na memória. Em geral, tais cartões são utilizados em aplicações mais complexas, como para cartões de débito inteligentes, em que a segurança é um requisito crítico. Isso ocorre, pois, nos cartões de memória convencionais, não é possível usar algoritmos de criptografia em software (não há um microprocessador), desse modo a segurança para esses tipos de cartões (*memory cards*) deve ser implementada pelo software aplicativo [18].

Há também uma distinção entre os *smartcards* com contato e sem contato. Os cartões com contato possuem um contato metálico no cartão, nesse contato há terminais metálicos ligados ao circuito interno. Desse modo, a leitura desses cartões é realizada por um leitor que permita a inserção do cartão e, conseqüentemente, a conexão dos contatos [49].

Os cartões sem contato não possuem contatos metálicos externos ao cartão, sendo semelhantes a cartões plásticos inteiriços. Tais cartões possuem uma antena interna responsável por captar a energia fornecida pelo leitor, o qual também possui uma antena interna. A antena do leitor fornece a energia necessária ao cartão por meio de indução eletromagnética (essa é a razão pela qual as antenas possuem o formato de espira). Esse tipo

de cartão é extensamente utilizado em transações que necessitem ser efetivadas rapidamente, como, por exemplo, no transporte coletivo.

Um dos pontos-chave quando se utilizam *smartcards* é a questão de segurança. Há vários mecanismos de segurança, sendo que os mecanismos utilizados pelos cartões de memória são menos sofisticados do que aqueles utilizados pelos cartões microprocessados. Os níveis de sofisticação para esses tipos de cartões são os mais diversos. Alguns, por exemplo, possuem criptocontroladores que comandam todas as funções de criptografia, sendo projetados para cálculos extremamente complexos em alta velocidade. Há também tipos de cartões que necessitam de um PIN (*Personal Identifier Number*), o qual precisa ser digitado para validação e permissão de acesso [50]. O cartão que foi utilizado no projeto possui um outro esquema de segurança específico que será explicitado em uma seção posterior [18].

Para esse projeto foram utilizados cartões de memória (*memory cards*). Apesar de existirem diferenças de um cartão para outro, pode-se enumerar as principais características dos *smartcards* (tanto *memory cards* quanto *microprocessor cards*) como sendo [18]:

- Os dados armazenados podem ser modificados (incluídos, alterados, excluídos) por meio de aplicações específicas;
- Difícil acessar e corromper informações;
- Os dados ficam gravados permanentemente, a menos que sejam alterados por meio de aplicações.

História da tecnologia de *smartcards*

A idéia de incorporação de 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. Independentemente, o japonês Kunitaka Arimura registrou uma patente no Japão em 1970. A partir daí começaram a surgir diversos tipos de pedidos de patentes na Europa, EUA e Japão. Como os conhecimentos tecnológicos da época eram insuficientes para permitir a realização industrial de muitos projetos, muitos acabaram sendo abandonados. Apesar das idéias iniciais já citadas, é freqüente encontrar na literatura que Roland Moreno é o inventor do *smartcard*, visto que registrou 47 patentes relacionadas a *smartcards* em 11 países entre 1974 e 1979 [49] [51].

Os cartões somente passaram a ser usados comercialmente no final dos anos 70 com o Grupo Bull. No entanto, o fato que alavancou o mercado surgiu na França, no início dos anos 80. Nessa década, foi verificado que a substituição das moedas nos telefones públicos

convencionais por *smartcards* trazia tanto benefícios para os usuários, que não necessitariam mais possuir moedas nos valores apropriados para efetuarem uma ligação, quanto para a empresa (France Télecom) que administrava o sistema, a qual não teria mais problemas com fraudes causadas pela depredação e roubo de aparelhos públicos. Além disso, o sistema passaria a ser pré-pago, o que certamente era mais interessante.

A partir dessa atitude na França, criou-se um amplo mercado para fabricantes de *smartcards* naquele país, o que é refletido ainda hoje em grandes grupos franceses do ramo de fabricação de *smartcards*.

Exemplos de domínios de aplicação para *smartcards*

Atualmente, há diversas aplicações para *smartcards* [52]. A seguir serão citadas algumas consideradas mais relevantes e serão explicados os benefícios proporcionados por elas.

Uso na indústria de telecomunicações

Conforme já fora supracitado, o uso de cartões telefônicos pré-pagos oferece um mecanismo seguro, anti-fraude e de baixo custo de manutenção para o acesso a telefones públicos. Não obstante, o maior mercado de *smartcards* do mundo é a telefonia móvel [18].

Toda segurança do sistema de celulares GSM (*Global System for Mobile Communications*), inclusive no que tange a evitar clonagem, é baseada em *smartcards*. Os *smartcards* utilizados nos celulares GSM são to tipo SIM *card* (*Subscriber Identity Module*), o qual é usualmente referenciado como "*chip*" pelos usuários.

Uso para efetivação de transações bancária e como moeda eletrônica

O uso de *smartcards* em detrimento aos de tarja magnética para a efetivação de transações bancárias proporciona a vantagem de não serem facilmente clonados, uma vez que o *chip* proporciona um nível de segurança maior aos cartões. No Brasil, os *smartcards* estão começando a serem amplamente utilizados, não só em substituição aos cartões magnéticos, mas também como moeda eletrônica. Os vales-refeição, por exemplo os encontrados em [19], têm sido amplamente substituídos por *smartcards*, os quais são carregados mensalmente.

Uso no transporte público

No Brasil, assim como em diversas partes do mundo, estão sendo adotados *smartcards* no transporte coletivo. Em São Paulo, por exemplo, o uso de *smartcards* no transporte coletivo ("BILHETE ÚNICO", [20]), proporcionou a integração entre conduções, proporcionando economia para aqueles que utilizam o sistema.

Acesso a ambientes restritos

Muitos locais que possuem bens e/ou informações de valor tais como instituições financeiras, laboratórios de pesquisa e empresas (prédios comerciais) têm adotado o uso de *smartcards* para restringir o acesso a determinados lugares a apenas pessoas autorizadas. Algumas instituições têm adotado o uso de tais cartões combinados com outras formas de identificação, de modo a reduzir a possibilidade de fraudes. O projeto descrito nessa monografia abordará uma possível combinação.

Dificuldades no desenvolvimento de aplicações para *smartcards*

Quando a tecnologia de *smartcards* começou a ser utilizada, desenvolver aplicações era um processo longo e complexo, pois havia enormes diferenças de funcionamento interno dos cartões entre um fabricante e outro, além de não haver interfaces de alto nível para *smartcards*. Desse modo, os desenvolvedores estavam sujeitos a lidar com protocolos de comunicação de baixo nível, gerência de memória e outros detalhes específicos dos dispositivos que eram utilizados, fazendo com que o processo de desenvolvimento tornasse dificultoso, exigindo um conhecimento muito específico e, conseqüentemente, um processo de alto custo. Havia também problemas de portabilidade de aplicações, uma vez que as plataformas de desenvolvimento eram proprietárias [18].

Outro ponto crítico no desenvolvimento estava ligado a questões de segurança, pois como os desenvolvedores lidavam com aspectos de baixo nível, como gerência de memória, a segurança acabava ficando fortemente dependente da habilidade de programar.

Atualmente existem tecnologias que fornecem subsídios para suplantar os problemas citados. Uma das tecnologias existentes é a plataforma *JavaCard* que oferece segurança e

portabilidade entre aplicações, além de facilitar o desenvolvimento. Há também plataformas proprietárias para desenvolvimento como a para cartões MIFARE® da empresa *Philips,* que foi utilizada no desenvolvimento desse projeto.

2.1.5 Arquitetura Cliente-Servidor

"A arquitetura cliente-servidor é um modelo para a interação de processos em execução" [21].

Para esta arquitetura existem duas entidades lógicas distintas: o cliente – processo que requisita serviços, e o servidor – processo que atende a requisição do cliente, realizando o processamento e enviando os resultados ao cliente.

Em grande parte dos casos, o cliente e o servidor estão localizados em máquinas distintas, entretanto, conceitualmente nada impede que ambos estejam na mesma máquina [21].

Nesta arquitetura, em geral o processo servidor executa em uma máquina com alta capacidade de processamento, uma vez que o servidor pode ter que atender diversos clientes ao mesmo tempo, ou atender requisições onerosas em termos de processamento.

Atributo	Cliente	Servidor
Modo de Operação	Ativo – sempre gera requisição de serviços	Reativo – sempre responde a requisição de um serviço
Execução	Possui início e fim	Fica em execução permanente
Objetivo principal	Atender as necessidades do usuário	Prover serviços aos clientes
Transparência	Oculta rede e servidores	Oculta detalhes da implementação de serviços
Inclui	Comunicação com diferentes servidores	Comunicação com diferentes clientes
Exclui	Comunicação entre clientes	Comunicação entre servidores

Figura 11 - Características Cliente-Servidor, extraído de [21]

Para estabelecer a comunicação entre as entidades lógicas (cliente e o servidor) é necessário estabelecer um protocolo de comunicação e interagir com a camada de transporte do modelo de rede [22].

A arquitetura cliente-servidor foi utilizada no módulo de vídeo do sistema SysSec. A escolha desta arquitetura pode ser justificada pelos seguintes fatos:

1. O processamento de vídeo digital demanda capacidade de processamento, principalmente, para a compressão do vídeo. Para não sobrecarregar a máquina que estará executando o SysSec, o processamento do vídeo (captura e compressão) foi deslocado desta máquina (cliente) para um outra máquina (servidor de vídeo).

2. Parte do processo do vídeo de *log* está implementado com a tecnologia C# e parte com a tecnologia Java. A utilização desta arquitetura permite a interoperabilidade dos módulos. O processo em C# ficou responsável apenas pela requisição da gravação do vídeo de *log* e o recebimento desta. O processo em Java realiza todo o processamento para a gravação e compressão do vídeo.

Para implementação da arquitetura cliente-servidor foi criado um protocolo de comunicação baseado em mensagens e na utilização de *sockets* TCP (a rede entre o cliente e o servidor deve utilizar o protocolo TCP/IP) para a interação com a camada de transporte.

O *socket* é uma interface entre a camada de aplicação e a camada de transporte do modelo OSI, a qual está baseada no modelo cliente-servidor. O termo *socket* também é utilizado para designar a API de uma linguagem de programação que permite acesso aos serviços da camada de transporte. Ao utilizar as APIs *socket*, as operações de envio e recebimento de mensagens tornam-se semelhantes às operações de entrada e saída de arquivo.

A implementação da arquitetura cliente-servidor usando *sockets* está representada na figura 12.

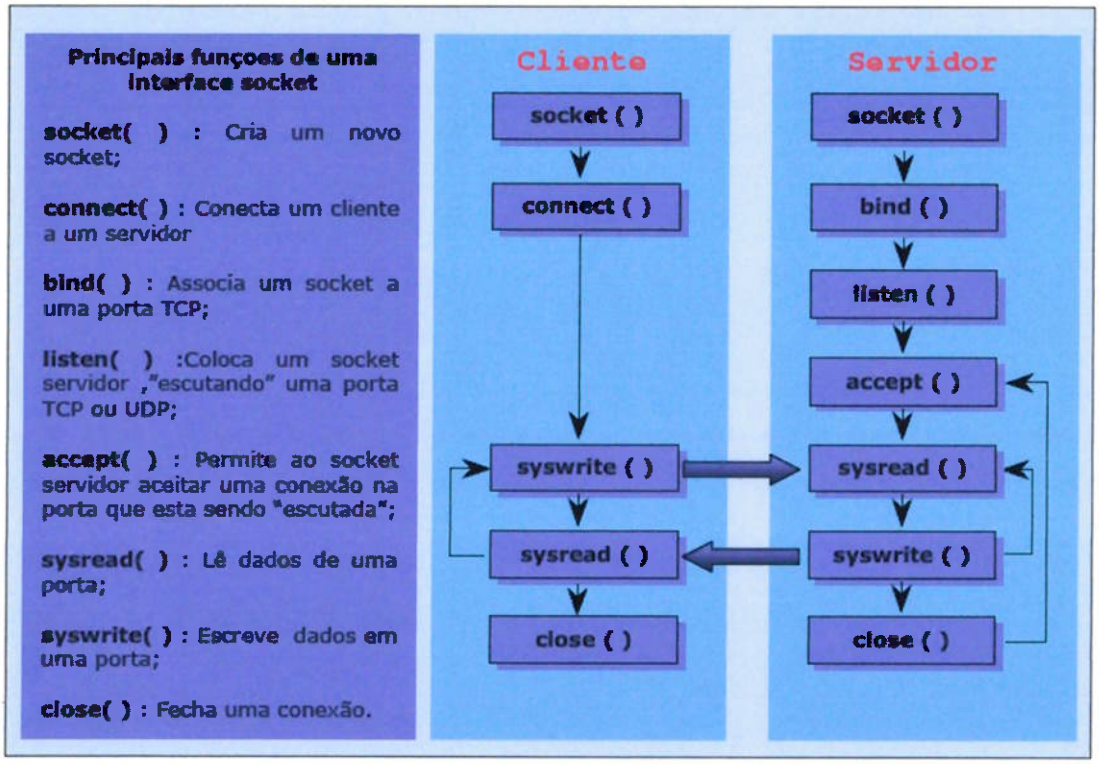

Figura 12 - Funções *Socket* TCP/IP para a arquitetura cliente servidor. Extraído de [21]

2.1.6 Arquitetura de *Software*

Cada vez mais o software constitui uma parte significativa dos sistemas atuais, seja na área de gerência ou mesmo em sistemas de automação. Por isso métodos têm sido propostos a fim de tornar o processo de desenvolvimento de software mais eficiente e a custos de manutenção menores. Além disso, sistemas cada vez maiores requerem o uso de disciplina, de modo a se obter resultados de menor custo e de melhor qualidade. Neste contexto, arquitetura de software tem aparecido como uma solução de modo a lidar com tais problemas.

Segundo [23]: "A arquitetura não é software operacional. Ao contrário é a representação que permite ao engenheiro de software (1) analisar a efetividade do projeto em satisfazer seus requisitos declarados, (2) considerar alternativas arquiteturais num estágio em que fazer modificações de projeto é ainda relativamente fácil e (3) reduzir os riscos associados com a construção de software".

Na literatura especializada, considera-se, geralmente, o projeto de arquiteturas convencionais como sendo constituídas de projeto de dados (envolve o armazenamento e estruturação da informação) e do projeto arquitetural propriamente dito.

Existem vários estilos de arquitetura [48]: arquiteturas centradas em dados (foco central no repositório de dados em torno do qual estão agrupados os *softwares*-clientes); arquiteturas de fluxos de dados (em que componentes computacionais ou manipulativos (filtros) transformam sucessivamente dados de entrada em dados de saída); arquiteturas de chamada e retorno (arquiteturas que podem ser expressas através de diagramas hierárquicos e envolvem os sistemas que possuem programa principal e subrotinas); arquiteturas orientadas a objetos e arquiteturas em camadas (caracterizadas por uma estrutura em que camadas mais próximas da máquina (instruções de máquina) são envolvidas por camadas mais abstratas. A camada superior pode usar os serviços das camadas inferiores, mas não vê as camadas que estão acima dela, apenas provê serviços para a mesma).

Modelo de três camadas

Na década de 90, este modelo tornou-se bastante popular e utilizado em engenharia de software, principalmente devido ao aumento da importância do papel das interfaces homem-máquina [48]. Nesse período, devido ao grande desenvolvimento para *web*, passou-se a dedicar uma camada especialmente para interface, evitando problemas entre o desenvolvedor

e o *web designer*. A recomendação básica foi separar de forma rígida onde termina a interface e onde começa a camada de negócios.

A arquitetura do modelo de três camadas divide o *software* em camadas correspondentes a serviços de usuário (interface homem-máquina), serviços de negócios (onde são implementadas as regras de negócios) e serviços de dados (responsável pela interação entre o sistema e o mecanismo de acesso à base de dados, contendo os objetos persistentes e os mecanismos para armazenamento dos mesmos).

A arquitetura de três camadas ainda continua sendo difundida porque separa a regra de negócio (cálculos, validações e processos) do banco de dados e do *Front End* (telas do usuário), permitindo a escolha do banco de dados no desenvolvimento, como por exemplo MS SQL (utilizado neste projeto), e evita que pessoas que desconheçam o processo de negócio e desejem melhorar a interface modifiquem a camada de negócios, gerando problemas para o desenvolvedor.

2.1.7 IDEF0

Para modelar processos de negócio existem várias ferramentas disponíveis, como por exemplo SADT (*Structured Analysis and Design Technique*) (Diagrama de Atividades), IDEF (*Integrated Definition Methods*), ou mesmo EPC (*Event-Driven Process Chain*) [24].

O IDEF (*Integrated Definition Methods*) é um grupo de métodos de modelagem bastante conhecidos e utilizados. Esta notação foi desenvolvida na década de 70 pela força aérea americana e, atualmente, está sendo utilizada em sistemas baseados em conhecimento. Originalmente criada para ambiente de manufatura, ao longo do tempo foi adaptada para uso em desenvolvimento de software (modelagem). É constituída de 3 tipos de representações gráficas: IDEF0, IDEF1X e IDEF2.

IDEF0 é uma notação gráfica especificada em *Draft Federal Information Processing Standards Publication* 183 (FIPS 183) e utilizada para modelagem funcional fornecendo uma visão detalhada do processo e de suas atividades componentes. O seu principal objetivo é mapear o fluxo de informações existentes entre funções componentes do processo. Esta notação gráfica é adequada para a representação estática de processos, modelando o sistema em *top-down*. Cada função ou atividade componente do processo pode ser detalhada por meio de sucessivas explosões da mesma.

IDEF0 possui quatro elementos básicos para representação: função/atividade, entradas (*inputs*), saídas (*outputs*), informações de controle e informações de recursos. A figura 13 mostra a representação básica de seus elementos constituintes.

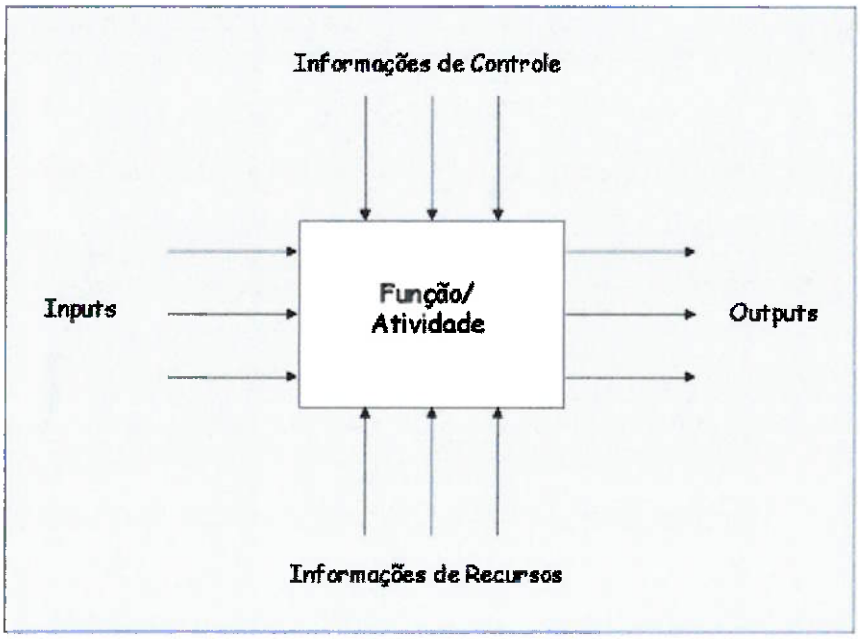

Figura 13 - A visão básica dos elementos do IDEF0, extraído de [24].

As flechas indicam:

3. **Entradas (*inputs*):** entradas necessárias para executar uma função;
4. **Recursos:** infra-estrutura utilizada;
5. **Controles:** controles necessários para execução da função;
6. **Saídas (*outputs*):** produtos (resultados) gerados pela função.

2.1.8 Threads

Threads são subprocessos de um processo principal que executam em paralelo com este. Em um ambiente multi-processado (plataforma com vários processadores), as várias *threads* podem ser escalonadas para diferentes processadores e neste momento tem-se uma aplicação paralela, com as várias *threads* executando ao mesmo tempo. Em um ambiente mono-processado, mas multiprogramado, existe uma entidade responsável por escalonar o processador para as várias *threads* do processo, de maneira a simular o paralelismo. No momento que uma *thread* tem a seu "quantum" de tempo de execução terminado, salva seus dados de controle de execução e uma outra *thread* "carrega" seus dados de controle de modo a começar a executar.

Threads dentro de um mesmo processo compartilham o mesmo espaço de endereçamento (principal diferença entre *threads* e processos). Cada *thread* possui sua pilha de execução (espaço usado para controle de chamadas a funções e valores de variáveis), e também possui registradores associados a ela, sendo o contador de instruções o registrador mais conhecido. O conjunto das variáveis de controle de uma *thread* armazenadas em registradores é conhecido como contexto [46] [47].

Threads são suportadas por muitas linguagens de programação, sistemas operacionais e outros ambientes de software.

A linguagem C# permite a criação de aplicações "*threaded*" através da declaração de um objeto do tipo *Thread*, o qual é associado a uma função específica. Quando a função termina sua execução, a *thread* é destruída.

Abaixo um exemplo de como criar uma *thread* e inicializá-la:

```
Thread t = new Thread(new ThreadStart(foo.A));
t.Start();
```

A seguir têm-se algumas funções básicas e uma breve descrição para o uso das *threads* em C# [43]:

1. `Start()`: inicia a execução de uma *thread.*
2. `Suspend()`: suspende a *thread*, caso já tenha sido suspensa, nada ocorrerá.
3. `Resume()` : reativa uma *thread* que foi suspensa, ocorre uma exceção, caso a *thread* não esteja suspensa.
4. `Interrupt()`: caso seja necessário interromper uma *thread* em particular, este método é utilizado. Se uma *thread* "t" está bloqueada esperando por um objeto e outra *thread* chama `t.interrupt()`, então "t" irá continuar a execução prendendo novamente o objeto, e então será lançada a exceção *ThreadInterruptedException.* Alternativamente, se "t" não esperar por um objeto, então de fato esta interrupção que foi chamada é gravada e a *thread* irá lançar a exceção *ThreadInterruptedException* na próxima vez que entrar no modo *sleep.*
5. `Sleep(int x)`: suspende a *thread* por uma fração de tempo específica (em milissegundos).

6. `Abort()`: Finaliza uma *thread*. Uma vez que a *thread* tenha sido finalizada, não pode ser reiniciada novamente utilizando-se a função `Start()`.

Em Java, ao contrário de outras linguagens de programação que utilizam o conceito de *threads* através de bibliotecas específicas, este conceito é incorporado dentro da própria linguagem, e por isso não é necessário efetuar o *link* entre seu programa *multi-thread* e nenhuma outra biblioteca externa.

Existem basicamente duas formas de se criar uma *thread* em Java: herdando da classe `java.lang.Thread` ou implementando a interface `java.lang.Runnable` (a classe `java.lang.Thread` implementa esta interface).

A fila de *threads* prontas e o conjunto de *threads* em espera são únicos por objeto Java, ou seja, cada objeto Java possui um monitor implicitamente associado a ele e a esses dois conjuntos de *threads*. É a Máquina Virtual Java que gerencia e controla ambos os conjuntos de cada objeto.

Há um outro método chamado `notifyAll()` que ao invés de acordar uma *thread*, acorda todas que estão a espera e as coloca na fila de *threads* prontas. A ordem das *threads* é aleatória e depende da Máquina Virtual Java. Maiores informações sobre *thread* em Java vide [39].

2.1.9 Interface Paralela

Cada vez mais os computadores pessoais (PC's) estão deixando de serem usados apenas para funções básicas como por exemplo editor de texto ou mesmo para realizar cálculos em planilhas [25], estão sendo utilizados para controlar equipamentos ou processos através de suas interfaces-padrão (portas), seja pela serial ou pela paralela. Por meio de comandos simples, pode-se acessar e controlar equipamentos conectados ao PC e, conseqüentemente, proporcionar um certo nível de automação aos processos.

A interface paralela surgiu em 1981, quando a IBM resolveu incluí-la no PC como alternativa as lentas portas seriais que eram usadas para se conectar as impressoras da época. Desse período até os dias atuais, por ser uma forma relativamente simples de se usar, pela não necessidade de recursos adicionais para sua interconexão e apresentar uma boa performance [25], a interface paralela deixou de ser apenas utilizada para o fim ao qual foi inicialmente

projetada, passou também a ser usada como interface de rede local e como interface com equipamentos externos e que se deseja controlar.

Atualmente, a interface paralela possui três modos de operações: SPP (*Standard Parallel Port* - em geral, *bits* de dados unidirecionais); EPP (*Enhanced Parallel Port* - *bits* de dados bidirecionais) e ECP (*Extended Capability Port* - *bits* de dados bidirecionais e este modo utiliza DMA (acesso direto à memória)) . Estes modos são configurados pelo BIOS Setup.

A porta paralela (utilizando conector DB-25) é composta por 25 pinos: 17 para linhas de sinal e 8 para linhas de terra. Essa pinagem do conector DB25 (para linhas de sinal) é dividida em três grupos: pinos de dados (8 vias: D0 a D7 – representa 8 *bits* (1 *byte*) e pode ser usado para controlar dispositivos externos); pinos de controle (4 vias- utilizadas para o controle de impressoras) e pinos de estados (5 vias – permite que o PC verifique o estado da impressora e o estado da interface paralela).

Os 17 sinais descritos anteriormente são determinados por registradores da interface paralela. Estes registradores são mapeados em memória em blocos contíguos de três registradores e são acessados a partir do endereço base da interface paralela em que estão mapeados (por exemplo, LPT1 – 378h e LPT2 –278h). Assim para se enviar ou receber dados via paralela é necessário saber o endereço base.

2.2 Tecnologia/Produto

Esta seção descreve as tecnologias e produtos usados para a elaboração do projeto.

2.2.1 JAVA 2 *Standard Edition* (J2SE)

A tecnologia Java é uma linguagem de programação, e também uma plataforma mantida pela Sun Microsystems Inc [26].

A plataforma Java (figura 14) é o ambiente de hardware e de software que o programa Java utiliza para sua execução. Possui dois componentes, a máquina virtual Java (Java VM) e o Java *Application Programming Interface* (Java API) [26].

A máquina virtual Java executa sobre um sistema operacional (plataforma baseada em hardware) e é responsável pela interpretação do programa Java. O outro elemento da plataforma é o Java *Application Programming Interface* (Java API), que é uma coleção de componentes de software prontos, que podem ser usados pelos programadores.

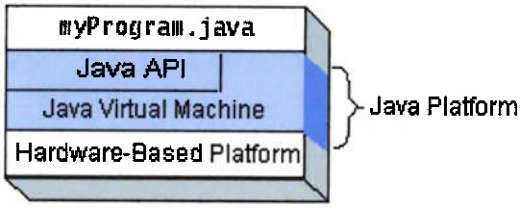

Figura 14 - Ilustração Plataforma Java, extraído de [26]

A linguagem Java é uma linguagem de programação de alto nível, voltada à filosofia de orientação ao objeto. Como principais características, temos:

- Altamente portável;
- Interpretada;
- Sintaxe simples;
- *Multithreading*;
- Mecanismos de segurança bastante eficazes;
- *Garbage collection* automático;
- Suporte amplo de desenvolvimento;
- Suporte para aplicações em rede.

Neste projeto foi utilizado o Java SE (*Standard Edition*) versão 1.4.2, que representa o núcleo da tecnologia Java para Desktop. A arquitetura da versão 1.4 pode ser visualizada na figura 15

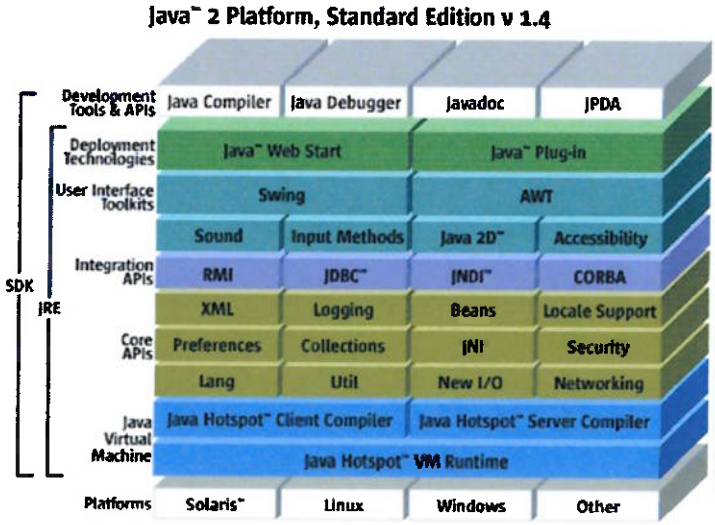

Figura 15 - Arquitetura Java 2 SE 1.4 (extraído de [26])

A tecnologia Java é gratuita e pode ser encontrada no site da Sun [27], juntamente com sua documentação e exemplos para o usuário.

2.2.2 Java *Media Framework*

O Java *Media Framework* (JMF) é uma API Java fornecida pela *Sun Microsystems Inc*, que estende a plataforma Java 2. Esta API especifica uma arquitetura simples e unificada para sincronizar e controlar áudio, vídeo e outras "mídias baseadas em tempo" em aplicações ou *applets* JAVA [28].

Originalmente, o JMF (JMF 1.x) especificava apenas a reprodução dos dados baseados em tempo. A partir da segunda versão (JMF 2.x), foi especificada tanto a reprodução quanto a captura, transmissão (através do suporte ao *Real Time Transport Protocol* – RTP) e codificações dos dados.

Assim sendo, o JMF fornece suporte a recursos de áudio e vídeo digitais permitindo a reprodução, captura, transmissão e codificações destes, de maneira a facilitar a programação e utilização de recursos multimídias na tecnologia Java. Para o tratamento destes tipos de dados, o JMF versão 2 (arquitetura disponível na figura 16) utiliza as seguintes abordagens:

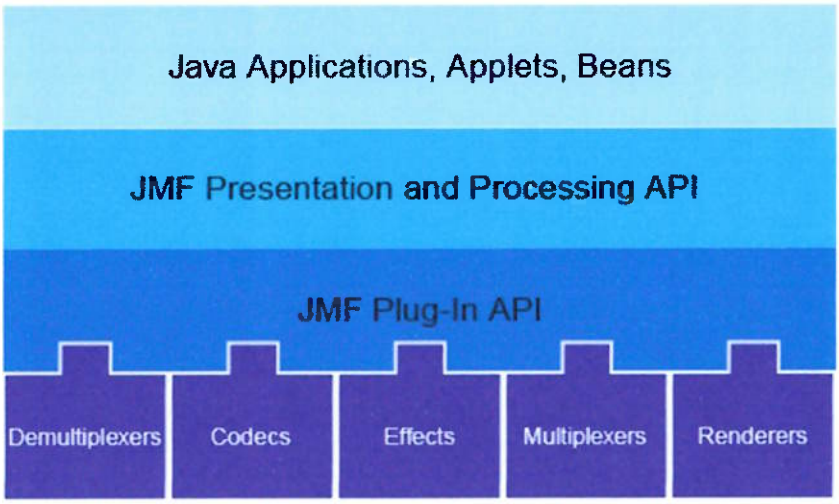

Figura 16 - Arquitetura alto nível JMF, extraído de [29]

- *Presentation*: referente à exibição da mídia através de algum dispositivo de saída (como monitores, caixas de som) sem que o usuário tenha controle de como é feito o processamento, ou como são renderizados (o termo renderização refere-se à formatação dos dados para sua "exibição" pelo dispositivo de saída).

- *Processing*: refere-se ao processamento da mídia (como aplicações de filtros, conversão de formatos). Estende as funcionalidades oferecidas pela abordagem *presentation* e permite que o usuário tenha controle de como é feito o processamento e renderização da mídia.

Para a apresentação de dados como vídeo e áudio é utilizada a interface *Player*. Um exemplo da utilização desta interface é a visualização do vídeo capturado por uma *Webcam*. Ao se utilizar um *Player*, os dados são renderizados e enviados ao dispositivo de saída, sendo omitidos detalhes de processamento e renderização (abordagem *presentation*). Por causa dessa omissão é considerada uma interface simples de se utilizar.

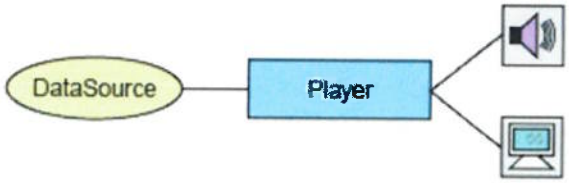

Figura 17 – Modelo esquemático do *Player*, extraído de [29]

A modelo do *Player* (figura 17) permite observar um detalhe de implementação importante: ao invés de existir o dispositivo de captura ou arquivo de mídia, existe um *data source* na entrada. O *data source* (ou *dataSource*) encapsula o *stream* de mídia. Este encapsulamento permite que o JMF realize a transferência deste tipo de dado.

O funcionamento da interface *Player* é baseado em seis estados (apresentado na figura 18). Segue abaixo a descrição de cada estado, baseado em [29]:

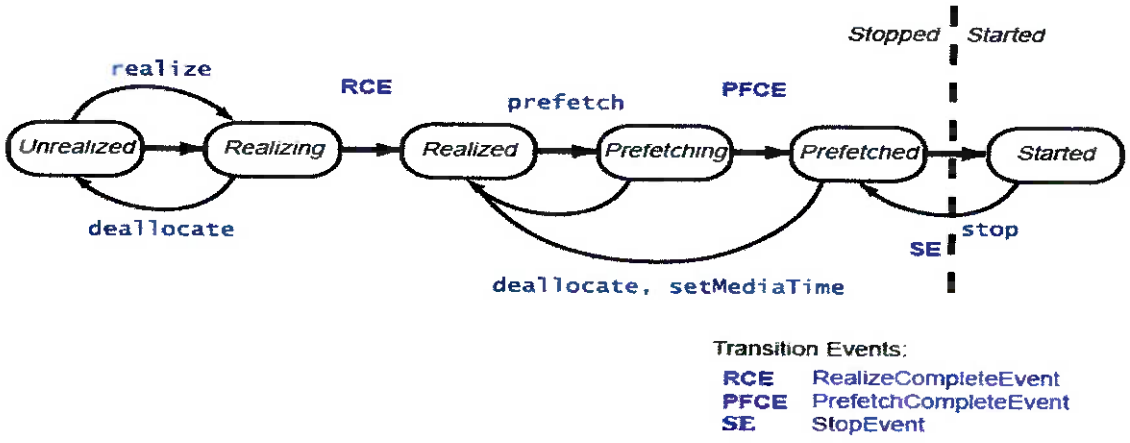

Figura 18 - Estados do *Player*, extraído de [29]

- *Unrealized*: representa a interface instanciada, mas sem conhecimento algum sobre a mídia a ser associada a esta interface;
- *Realizing*: representa a interface determinando os recursos necessários para seu funcionamento;
- *Realized*: representa a interface com os conhecimentos dos recursos necessários e do tipo de mídia a ser exibida. Neste estado, por se conhecer o tipo de mídia, a interface já pode fornecer os componentes de controle e componentes visuais;
- *Prefetching*: representa a interface preparando para a "exibição" da mídia;
- *Prefetched*: representa a interface pronta para executar;
- *Started*: representa a interface executando.

Para o processamento de dados como vídeo e áudio é utilizada a interface *Processor*. Essa interface, assim como a interface *Player*, permite a "exibição" da mídia. Entretanto, o controle de que tipo de processamento é feito com a entrada não é omitido, tornando-se uma interface complicada e "poderosa" quando comparado com a interface *Player*.

Basicamente, seu funcionamento consiste no tratamento do *stream* de entrada associado ao *dataSource* de entrada, seguido do processamento do *stream* definido pelo programador e, por fim, da renderização (quando enviado para um dispositivo de saída) ou tratamento do *stream* de saída (associando a um novo *dataSource*).

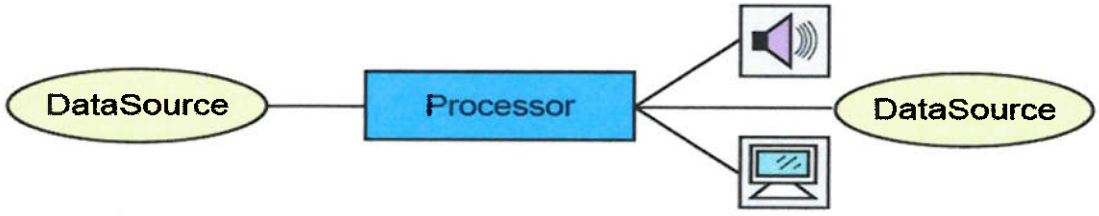

Figura 19 - Modelo interface *Processor*, extraído de [29]

O modelo da interface *Processor* (figura 19) permite associar os dados processados (saída) a um outro *dataSource*. Tal característica possibilita a associação de diversos *Processors*, ou a associação de um *Processor* com um *Player*, ou a gravação dos dados processados em um arquivo por meio de um *dataSink* (o *dataSink* é responsável por associar um *dataSource* a um arquivo em disco). É por meio desta interface que é possível gravar um vídeo capturado por uma *Webcam* num arquivo.

O funcionamento da interface *Processor* também é baseado em estados (figura 20), possuindo dois estados a mais em relação à interface *Player* (os demais estados são análogos aos descritos na interface *Player*). Os novos estados estão descritos abaixo, baseados em [29]:

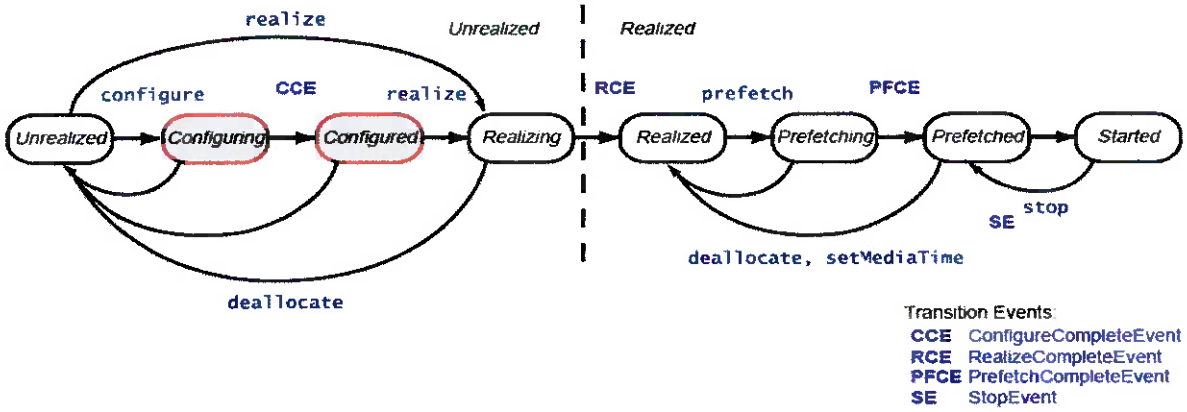

Figura 20 - Estados do *Processor*, extraído de [29]

- *Configuring*: estado em que a interface conecta-se ao *dataSource* de entrada e acessa a informação do formato do *stream* de entrada;
- *Configured*: estado em que a interface está conectada ao *dataSource* de entrada e conhece o formato do *stream* de entrada.

Para implementar tais interfaces, a API JMF possui diversas funcionalidades como modelo para a representação do tempo (medido em nanosegundos), modelo dos dados (baseado no uso de *dataSources* para a transferências dos *streams* e dos formatos de dado para a representação), modelo de eventos, *managers* (que são responsáveis por instanciar as interfaces *Player* e *Processor*, por exemplo), dentre outras que podem são descritas em [29].

Além da biblioteca para Java, o JMF possui um programa (denominado JMF *Registry*) que permite o cadastro de dispositivos de captura, *plugins* e *packages* de maneira a estender as funcionalidades do JMF.

No projeto SysSec, esta API será utilizada para a gravação em arquivo dos vídeos capturados pela *Webcam*. Para tanto, será utilizado a versão 2.1.1e do JMF, otimizada para *Windows* (JMF *Performance Pack for Windows*). Estes *packs* executam de maneira otimizada na plataforma específica (no caso *Windows*), mas em contrapartida reduzem a portabilidade do programa final. O programa JMF *Registry* também foi utilizado pelo projeto para registrar a *Webcam*. Para utilizar qualquer dispositivo de captura suportado pelo JMF, deve-se antes registrá-lo por meio do botão *Commit* do programa JMF Registry, menu Capture Devices

(figura 21). Ao cadastrar um dispositivo, o programa fornece a localização do dispositivo e os formatos aceitos por este dispositivo (dados úteis para a programação usando a API JMF).

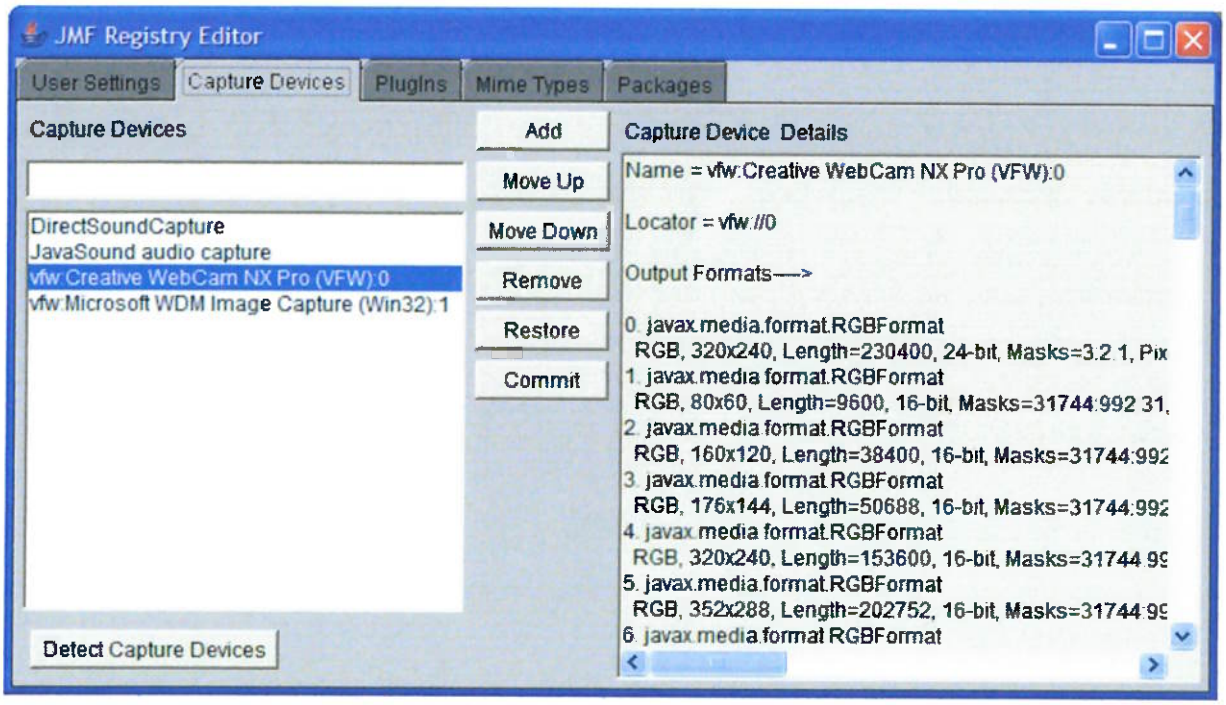

Figura 21 – JMF *Registry – capture device*

2.2.3 Linguagem C# e Plataforma .NET

A plataforma .NET, desenvolvida pela Microsoft [30], está baseada no .NET *Framework* (figura 22) e nas linguagens compatíveis com a plataforma. Como descrito em [31], o .NET *framework* é um ambiente de execução e desenvolvimento que permite diferentes linguagens e bibliotecas trabalharem juntas.

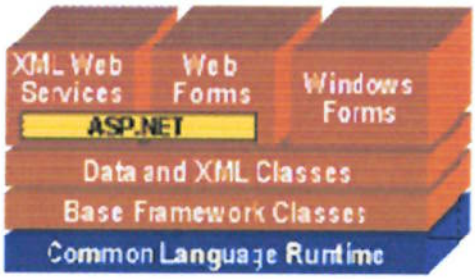

Figura 22 - .NET Framework, extraído de [31].

A linguagem C# (leia-se c sharp) é uma linguagem de programação que é executada sobre o .NET *Framework*. Evolução do Microsoft C e Microsoft C++, é uma linguagem simples com tipos seguros e orientada a objeto. Por evoluir do C, acabou herdando algumas

características e estruturas típicas de linguagem estruturadas, como a existência de ponteiro (que pode ser declarado em blocos de código inseguro).

2.2.4 Linguagem C

A linguagem C foi criada por Dennis M. Ritchie e Ken Thompson no laboratório Bell em 1972. Foi baseada na linguagem B, que era uma evolução da antiga linguagem BCPL. A definição formal da linguagem está contida no livro *The C Programming Language,* escrito por Brian W. Kernighan e Dennis M. Ritchie [44].

A linguagem C é uma linguagem de programação estruturada bastante popular devido a sua versatilidade e poder. Um programa em C consiste, basicamente, de uma ou mais funções. Estas funções podem estar dentro de um mesmo código-fonte ou em códigos-fontes separados (estes códigos que possuem apenas funções de apoio para outros programas são denominados bibliotecas) tornando a linguagem modular. Algumas vantagens da linguagem C são [44][45]:

- Possuir tanto características de alto nível quanto de baixo nível. Por possuir características de baixo nível, C é frequentemente usada para a elaboração de programas em *hardwares* dedicados que não possuam sistema operacional.
- Os códigos em C são, em sua maioria, compilados em código de máquina nativos. Por gerar códigos na linguagem de máquina, programas em C são tipicamente mais rápidos e não necessitam de interpretadores (poupando recursos), entretanto não possui características de controle (como *Garbage collection*).
- Possibilidade de utilizar módulos de software escritos diretamente em linguagem nativa.

Para este projeto, devido a restrições tecnológicas, será utilizado C compilado para 16 *bits* (o sistema operacional *Windows* tem a capacidade de emular programas MS-DOS 16 *bits*).

2.2.5 *Smartcard Contactless* Mifare® 1k

Tendo em vista que o cartão a ser utilizado no projeto é um cartão de memória, nessa seção será apresentada a arquitetura e os principais aspectos relacionados ao cartão de

memória utilizado no projeto. Cumpre ressaltar que as informações dessa seção foram extraídas da documentação técnica disponível em [32][33][34].

Diagrama de blocos da arquitetura interna

Nesse projeto foi utilizado o cartão de memória MIFARE MF1ICS50. A arquitetura interna desse cartão pode ser vista na figura 23.

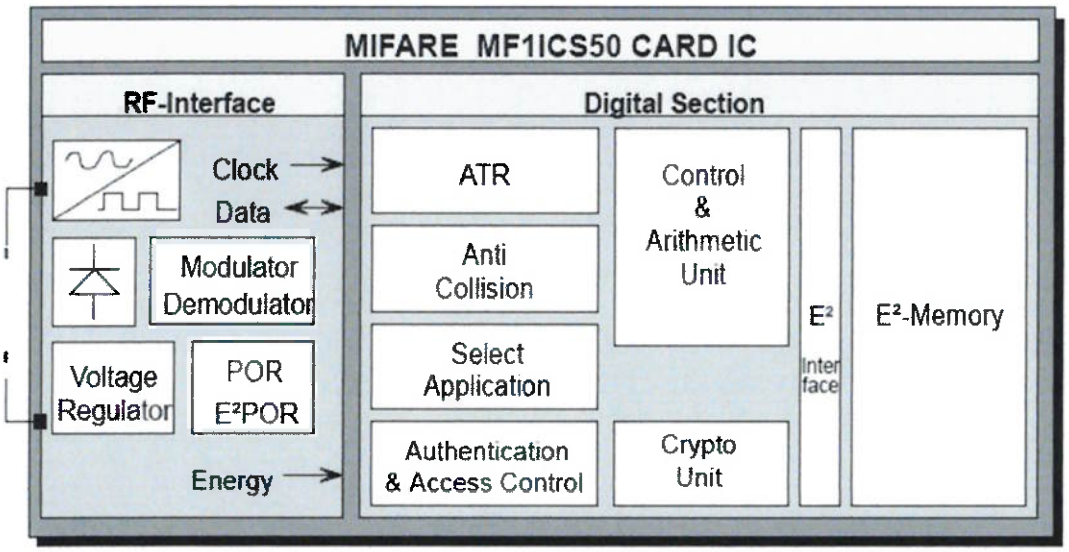

Figura 23 - Diagrama de blocos do cartão MIFARE MF1ICS50. Extraído de [32]

Os blocos que constituem a arquitetura interna do cartão podem ser descritos como segue:

- Memória EEPROM: memória de escrita e leitura para dados organizada em setores e blocos;
- Interface de acesso para memória: interface responsável pelo controle de acesso à memória;
- Unidade de criptografia: esta unidade criptografa os dados segundo a chave de autenticação (após o cartão ser autenticado, os dados trocados entre o leitor e o cartão são criptografados);
- Unidade de controle e aritmética: responsável pela realização de cálculos simples (incremento, decremento);
- ATR (*answer to request*): unidade responsável pelo reconhecimento de *requests* recebidos do leitor;

Esquema de comunicação entre o Leitor/Cartão

Sempre que ocorre a comunicação entre o cartão e o leitor, os seguintes passos devem ser seguidos (figura 24):

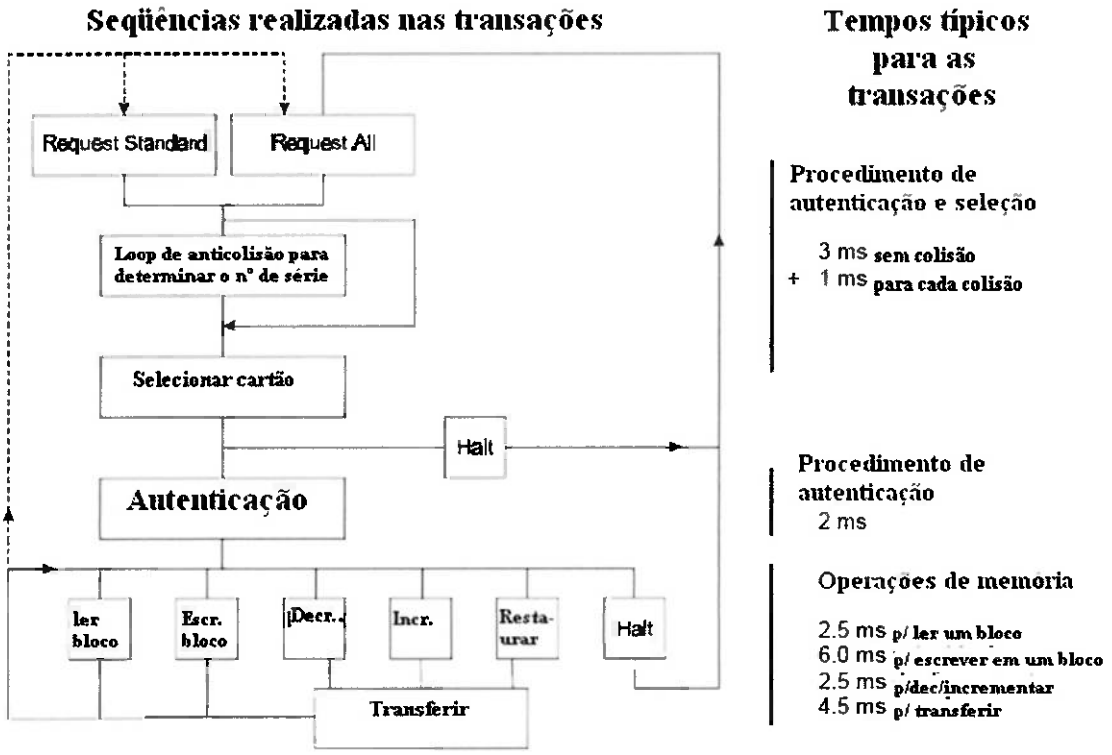

Figura 24 - Seqüência de passos existentes no momento que uma transação entre o leitor e o cartão é realizada. Extraído de [32]

O primeiro passo da seqüência consiste em enviar requisições (*request*) para todos os cartões que estiverem no campo de alcance da antena do leitor. Uma vez que algum(ns) cartões tenham sido reconhecidos, a comunicação continua com o protocolo adequado. Após o envio das requisições, inicia-se o processo (*looping*) de anticolisão que é responsável por escolher um cartão dentre os vários que estiverem no campo de operação do leitor, ou melhor, é responsável por ler o número de série de um cartão específico. Nessa etapa, os cartões que não foram selecionados, entram em modo de espera (*halt*) e ficam aguardando outras requisições. Em seguida é realizada a operação de seleção de cartão (comando *select*) que tem por finalidade selecionar um cartão individualmente para posterior autenticação e operações relacionadas à memória (leitura, escrita, incremento de valor, decremento de valor). Nessa etapa, o cartão retorna o ATS (*Answer to Select Code*) que determina o tipo individual do cartão selecionado. Uma vez identificado e selecionado um cartão, inicia-se a autenticação,

cuja finalidade é fazer com que todos os dados que trafeguem entre o leitor e o cartão a partir da autenticação sejam criptografados segundo uma chave que somente o cartão e o leitor conhecem. Por fim os dados podem ser lidos (*read*), escritos (*write*), decrementados (*decrement*) e incrementados (*increment*). Ao final da transação os dados são transferidos e os cartões voltam para o modo de espera (*halt*).

Integridade dos dados

Alguns mecanismos de proteção são implementados para garantir a integridade dos dados que trafegam durante a comunicação entre o dispositivo de leitura/escrita e o cartão:

- Mecanismo de anticolisão cuja finalidade é selecionar um entre diversos cartões que estejam no campo de alcance do leitor;
- CRC de 16 *bits* por bloco;
- Existência de um *bit* de paridade para cada *byte* de um determinado bloco (16 *bits* de paridade por bloco);
- Existência de um contador de *bits* para checagem;
- Existência de um *bit* de codificação que permite distinguir entre nível "1", "0" ou sem informação;
- Monitoração do canal de comunicação (análise do *stream* de *bits* e seqüência de protocolo).

Segurança

Para prover um nível adequado de segurança, é utilizada a autenticação em três passos (de acordo com a norma ISO 9798-2), a criptografia dos dados baseada no algoritmo "*stream cipher*" com gerador aleatório, número de série único e chaves de autenticação de 48 *bits* (6 *bytes*). As chaves do cartão são protegidas contra leitura, mas podem ser alteradas, fazendo com que o conhecimento das mesmas seja restrito. Assim, pode-se dividir a memória do cartão em partes de acordo com diferentes aplicações, sendo que cada aplicação acessará o cartão com as chaves especificadas (proteção de memória entre aplicações).

Organização de memória e condições de acesso

O MF1ICS50 possui uma capacidade de memória interna de 1K*byte* (EEPROM) organizados em 16 setores com 4 blocos de 16 *bytes* cada (16 x 4 x 16 = 1024 *bytes* = 1K*byte*). A Figura 25 mostra a organização de memória interna do cartão.

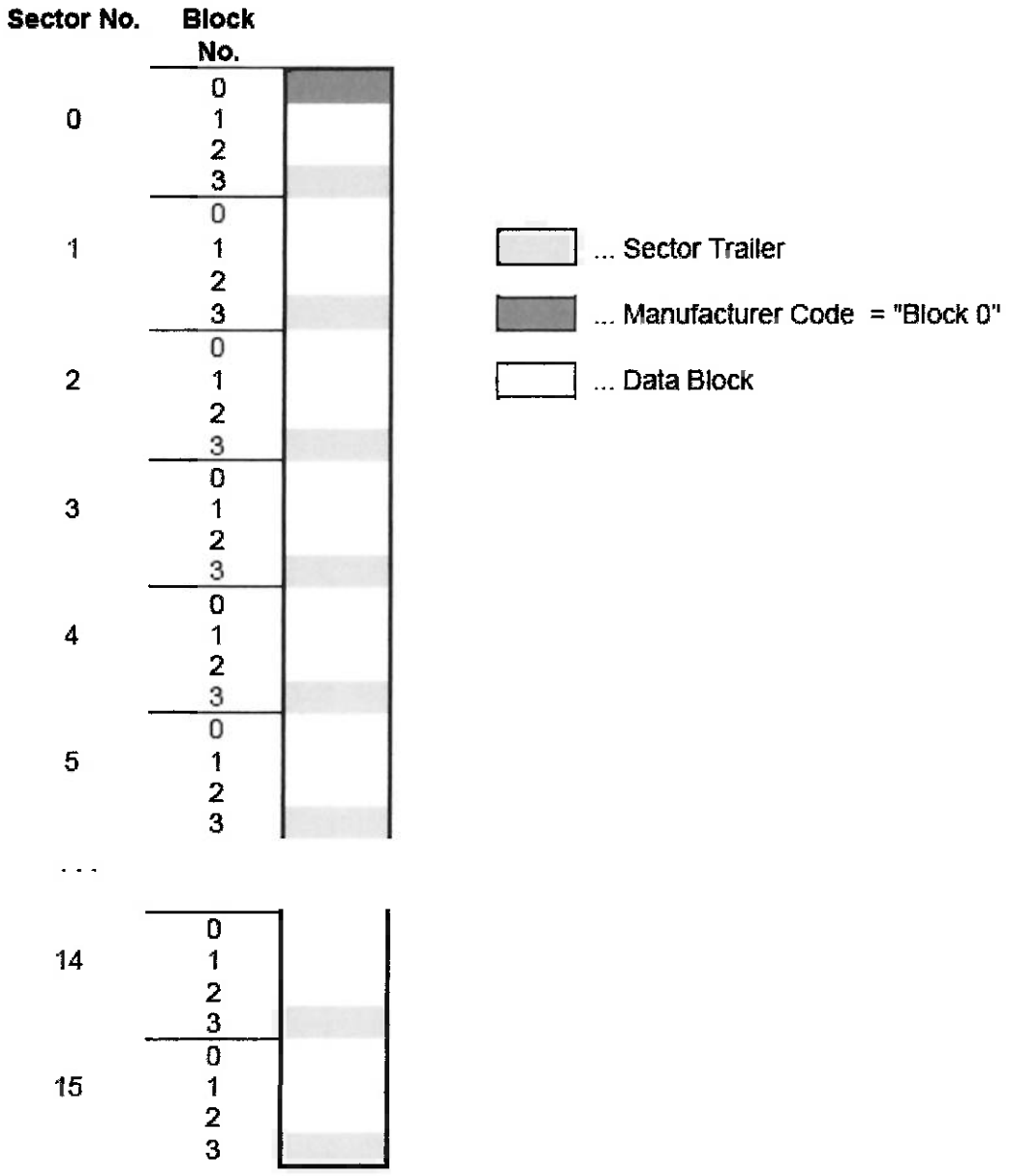

Figura 25 - Organização de memória do cartão MF1ICS50. Extraído de [32]

Conforme pode-se observar na figura 25, há três tipos de blocos:

- Bloco 0 do setor 0: é um bloco somente leitura que contém o número serial do cartão e informações do fabricante.
- Bloco 3 de cada setor (bloco de *trailer*): o bloco de trailer é um bloco de escrita/leitura que contém as chaves de autenticação e as condições de acesso para os outros blocos de um determinado setor.
- Bloco de dados: corresponde a um bloco de leitura/escrita onde se pode armazenar 16 *bytes* de dados.

2.2.6 Leitor MD-RD260

O leitor/gravador de cartões utilizado nesse projeto foi o *MIFARE® Serial Reader (Short Range)*. Este dispositivo está baseado no *MFCM200 - MIFARE® Micro Module (MMM)* e consegue ler cartões *MIFARE® 1 S50 card ICs*, ou seja, os tipos usados nesse projeto.

Segue um diagrama de blocos do dispositivo:

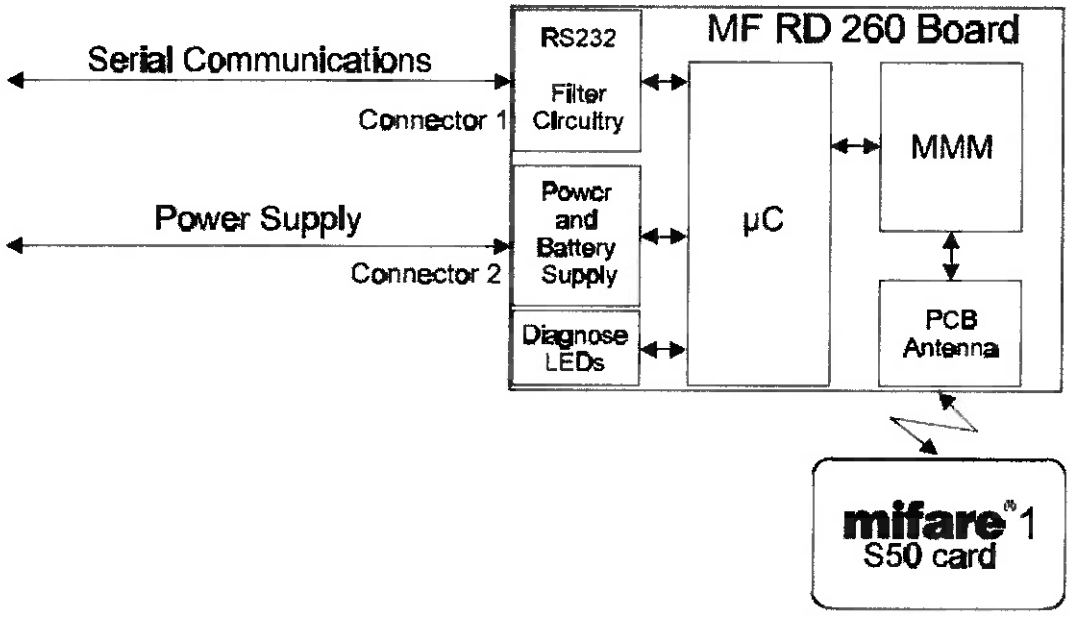

Figura 26 – Diagrama de blocos do leitor MF-RD260 extraído de [33]

A Figura 26 mostra os componentes do dispositivo de leitura/gravação. As funções de cada componente serão explicitadas a seguir:

- *Power Supply*: cabo de tensão de alimentação (6 a 8V – tensão contínua);
- *Serial Communications*: cabo serial para comunicação com o PC;

- RS232 *Filter Circuitry*: filtro para os sinais da interface serial;
- *Power and Battery Supply*: bateria utilizada para prevenir que chaves secretas gravadas na WOM (*Write Only Memory*) do MFCM200 sejam perdidas quando dispositivo de leitura (MF-RD260) não estiver sendo alimentado.
- *Diagnose* LEDs: LEDs de monitoração. Há dois: um verde e outro amarelo;
- MF-RD260 *Board*: placa de circuito impresso que acomoda os componentes do leitor;
- μC: microcontrolador responsável por traduzir o protocolo serial (RS232) para a interface paralela existente no MMM (*MFCM200 MIFARE® Micro Module)*
- MMM: *MFCM200 MIFARE® Micro Module* inclui uma unidade de rádio-freqüência (RF-*unit*) e o leitor *VLSI MIFARE® Reader ASIC*. A unidade de rádio-freqüência comunica-se com a antena por meio de rádio freqüência e com o *MIFARE® Reader ASIC* por meio do barramento (interface digital). O MMM realiza todas as funções analógicas de modo a acessar os cartões. Estas funções incluem modulação e demodulação e geração do sinal de rádio freqüência. A comunicação do MMM com o μC é realizada por meio de uma interface paralela (barramento paralelo);
- PCB *Antenna*: corresponde a uma antena no formato de espiral (trilhas em espiral na placa de circuito impresso).

A freqüência de operação do leitor é 13.56MHz. Cada leitor possui um número serial que pode ser lido por *software* e pode ser usado para implementar maior segurança em sistemas que utilizem múltiplos leitores. A taxa de comunicação entre o cartão e o leitor é de 106 Kbps. A memória do MFCM200 é *read only* para que não seja possível outros leitores capturarem a chave de criptografia armazenada no leitor no momento de uma transação de leitura/escrita (após a autenticação do cartão).

2.2.7 Creative *WebCam* NX PRO

No projeto SysSec os vídeos de *log* serão gravados por meio dessa *Webcam*. A escolha de uma *Webcam* para a gravação dos vídeos se deve ao baixo custo.

Principais características da WebCam NX PRO [35]:

- Sensor CMOS (640x480 colorido);
- Captura de Vídeo no formato de 24 *bits*, com as seguintes características:

  - 30 *frames* por segundo, na resolução 352x288 (*Standard System*);
  - 15 *frames* por segundo, na resolução 640x480 (*Standard System*);
  - Formatos I420 e RGB24;
  - Grava em Formato AVI.
- Conexão USB;
- Lentes de Alta Qualidade;
- Foco Manual;
- Campo visual de 52 graus;
- Compatível com *Video For Windows*;
- WDM *MiniDriver* para *DirectShow*;
- Suporte ao *Microsoft Still Image*.

É importante destacar o fato de possuir compatibilidade com *Video For Windows* (VFW). Esta característica, segundo a documentação do JMF [36], é necessária para que este possa acessar a WebCam.

3 Planejamento e Metodologia

3.1 Gestão do Projeto

Um projeto, segundo o *Project Management Institute* (PMI 2000) [37], é definido como "um empreendimento temporário feito para criar um produto ou serviço único". A *International Organization for Standardization* (ISO) [38] define na norma ISO 10006 de 1997 como "um processo único, consistindo de um grupo de atividades coordenadas e controladas com datas para início e término, empreendido para alcance de um objetivo conforme requisitos específicos, incluindo limitações de tempo, custo e recursos".

Para o planejamento do projeto SysSec foram utilizadas práticas de gestão de projetos visando atingir as metas definidas, dentro do prazo de tempo e dos recursos disponíveis (características destacadas pela definição do termo projeto).

No início do planejamento foi elaborado o *Work Breakdown Structure* (WBS), exibido na figura 27. Esta ferramenta de gestão de escopo decompõe o trabalho do projeto em entregas parciais e entrega final. As entregas estabelecidas na WBS permitem identificar as atividades necessárias para completar o projeto. A lista de atividades obtida para o Projeto SysSec encontra-se abaixo:

1. Pesquisa das tecnologias utilizadas (*smartcard*, biometria digital e vídeo digital);
2. Avaliação da infra-estrutura existente (leitor de *smartcard*, *scanner* de impressão digital e *Webcam*);
3. Elaboração da arquitetura final do sistema;
4. Definição do ambiente de desenvolvimento;
5. Implementação do módulo de vídeo digital;
6. Implementação do módulo de identificação biométrica;
7. Implementação do módulo de cadastramento de biometria;
8. Implementação do módulo de acesso ao cartão;
9. Implementação do módulo de cadastro do cartão;
10. Implementação do módulo de identificação por meio do cartão;
11. Implementação do módulo de gerência;
12. Integração dos módulos;
13. Testes do sistema;

14. Relatório de especificações parcial do sistema;
15. Relatório de especificações final do sistema;
16. Relatório parcial do andamento do projeto;
17. Elaboração da monografia final;
18. Elaboração do Pôster;
19. Resumo, apresentação e prestação de contas;
20. Encerramento do projeto.

Uma vez definido o planejamento do escopo, foi desenvolvido o planejamento do tempo. Na primeira etapa do planejamento do tempo foi elaborada a rede de precedências (figura 28), ferramenta de gestão de tempo que estabelece relações de precedências entre as atividades listadas. Na segunda etapa, foi elaborado o cronograma do projeto, baseado na rede de precedência e na definição do tempo de duração de cada atividade do projeto.

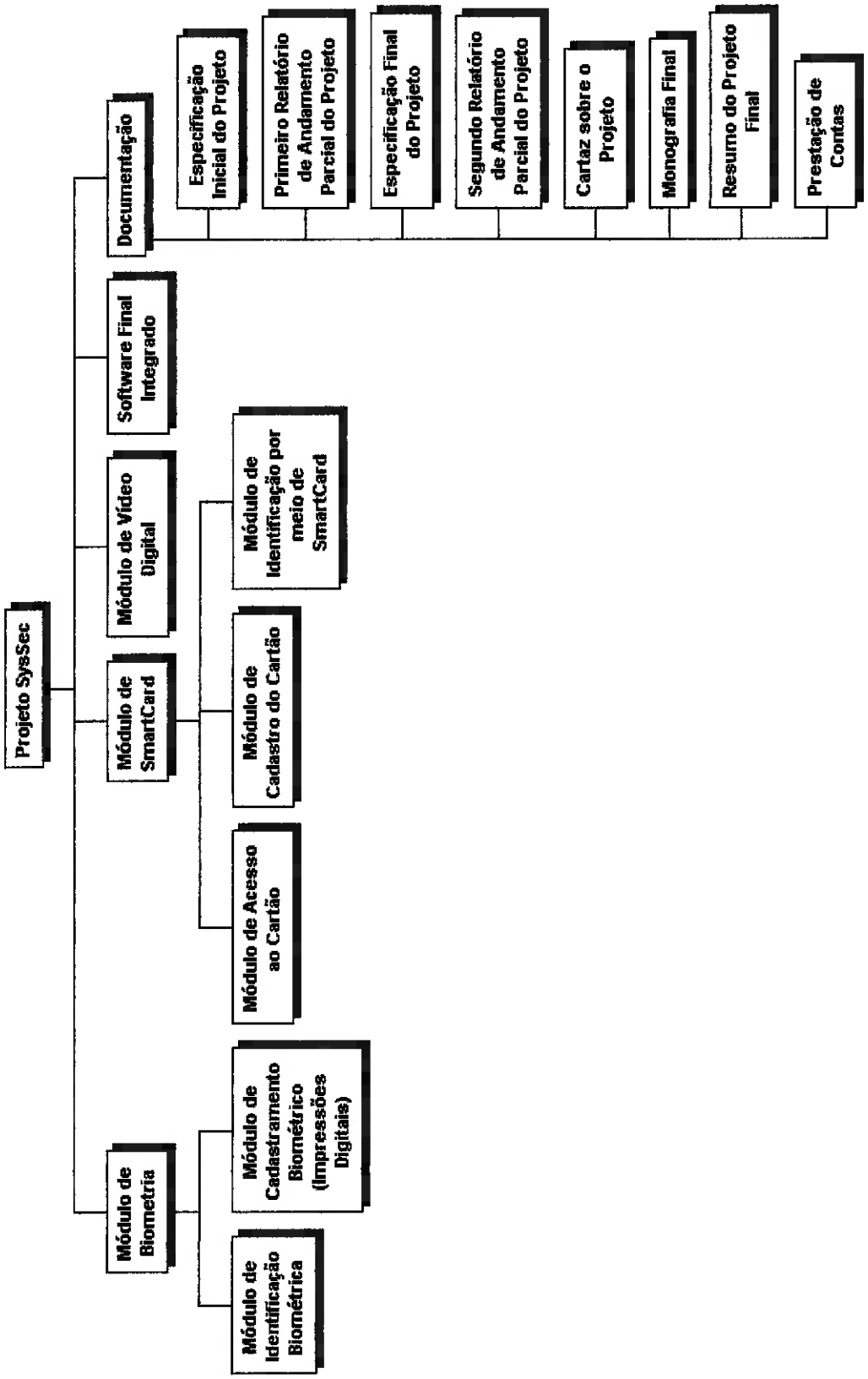

Figura 27 – WBS do Projeto SysSec

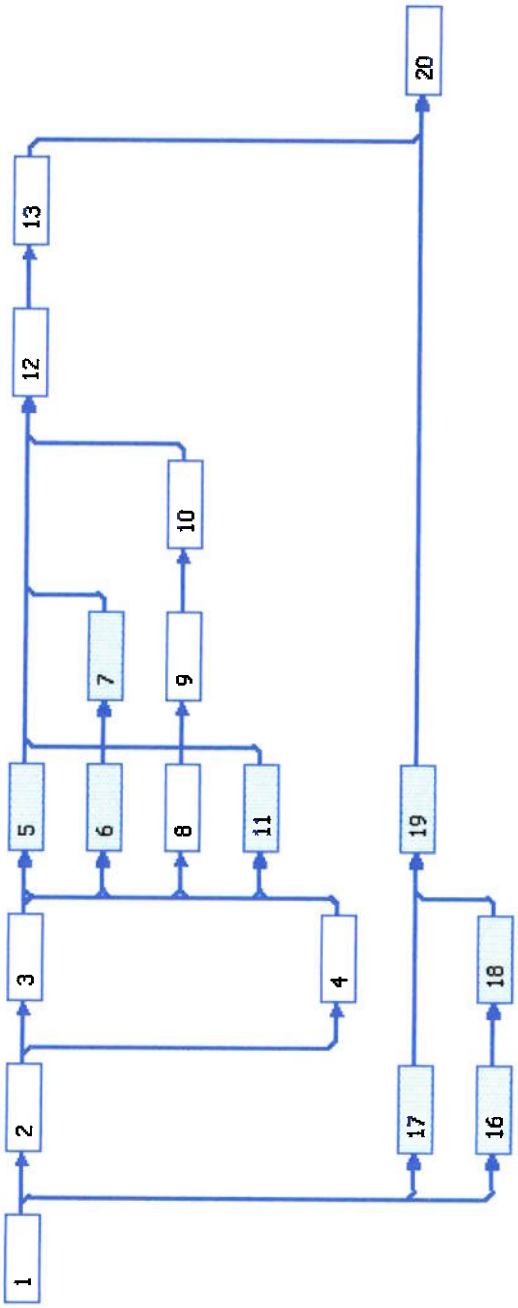

Figura 28 - Rede de Precedência

3.2 Desenvolvimento de *Software*

Para o desenvolvimento dos módulos de *software* do sistema SysSec, foi utilizada a tecnologia de desenvolvimento UML. Os modelos da UML utilizados foram: descrição de casos de uso, diagrama de casos de uso, diagrama de classes, diagrama de estados (para modelagem da navegabilidade) e diagrama de pacotes.

Também foram elaborados modelos IDEF0 para descrição do sistema em nível de processo, diagrama entidade-relacionamento para a modelagem dos dados, diagrama da arquitetura de software e diagrama da arquitetura de hardware.

Considerando manutenções, expansões futuras e visando estabelecer uma organização na codificação do *software*, o mesmo foi desenvolvido segundo o modelo de três camadas.

4 Especificações do Projeto

4.1 Funcionalidade

Para descrever as funcionalidades do sistema foram utilizadas duas técnicas de modelagem. A primeira abordagem fornece uma visão *top-down* através da ferramenta IDEF0. A segunda abordagem fornece uma visão *bottom-up* através da descrição detalhada dos casos de uso (anexo 1).

O sistema SysSec está dividido funcionalmente em dois blocos: módulo de controle de acesso e módulo de gerenciamento do sistema. Inicialmente, para modelar as funcionalidades foi utilizada a ferramenta de modelagem IDEF0.

4.1.1 Módulo de controle de acesso

O módulo de controle acesso é responsável por autenticar o usuário no sistema, liberando ou não o acesso. O diagrama de nível zero está representado na figura 29:

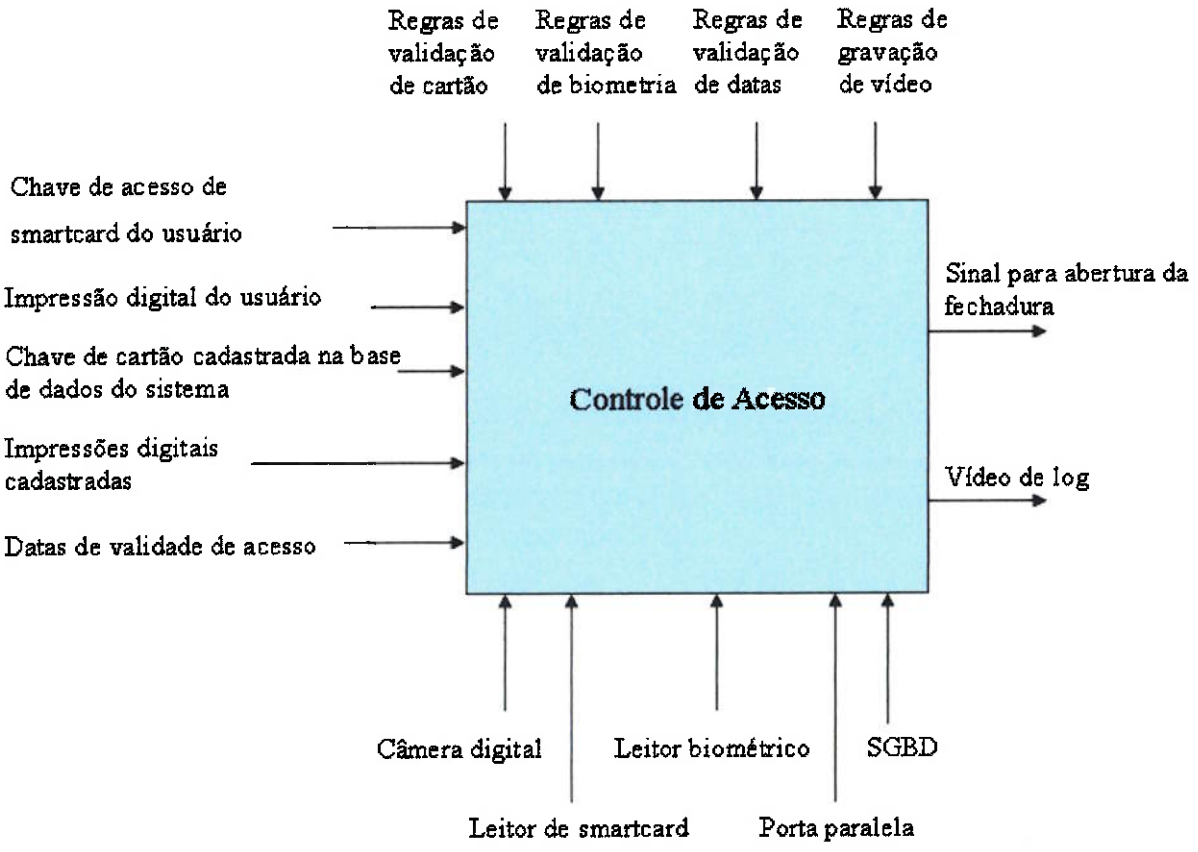

Figura 29 – IDEF0 Nível Zero - Controle de Acesso

Entradas externas ao sistema:

- **Chave de acesso de *smartcard* do usuário:** consiste de uma chave de 16 *bytes* gerada randomicamente e armazenada em um *smartcard*. Esta é obtida por meio do leitor de *smartcard* durante o processo de validação.
- **Impressão digital do usuário:** informação de impressão digital do usuário que pretende acessar o local. Esta é obtida por meio do leitor biométrico durante o processo de validação.
- **Chave de cartão cadastrada na base de dados do sistema:** informação que é utilizada para validação do cartão no sistema. É realizada uma comparação entre a chave gravada no cartão e a chave cadastrada na base de dados para um determinado usuário.
- **Impressões digitais cadastradas:** informação que é utilizada para a validação da impressão digital do usuário.
- **Datas de validade de acesso:** consiste das datas de início e expiração de autorização de acesso.

Saídas:

- **Sinal para abertura da fechadura:** sinal que é enviado para a fechadura de modo a controlar a abertura ou não da mesma.
- **Vídeo de *log*:** arquivo de vídeo gerado durante a autenticação do usuário.

Recursos:

- **Câmera digital:** câmera responsável pela gravação do vídeo.
- **Leitor biométrico:** dispositivo responsável pela digitalização das impressões digitais do usuário.
- **Leitor de *smartcard*:** dispositivo responsável pela leitura/gravação em *smartcards*.
- **SGBD:** corresponde ao software gerenciador de banco de dados responsável pela manipulação, persistência e integridade dos dados.
- **Porta paralela:** meio pelo qual o sinal para abertura da fechadura é propagado.

Controle:

- **Regras de validação do cartão:** consiste em verificar se a chave lida no cartão de acesso do usuário consta no sistema.
- **Regras de validação de biometria:** consiste em verificar se as impressões digitais obtidas pelo leitor conferem com as previamente cadastradas.
- **Regras de validação de datas:** consiste em verificar se o acesso está ocorrendo dentro do período de validade de autorização do usuário, ou seja, se a data atual está contida no intervalo definido pela data de início e data de fim de autorização.
- **Regras de gravação de vídeo:** consiste na regulamentação da gravação de vídeo (tempo de registro, regras para a criação do nome do vídeo, entre outras).

De modo a especificar melhor as funcionalidades do sistema, fornecendo assim uma visão mais detalhada do mesmo, o diagrama de nível zero foi explodido gerando o nível um. O diagrama de nível um deste módulo está representado na figura 30.

A explosão do módulo de controle de acesso gerou os seguintes módulos:

- **Autenticação por *smartcard*:** Neste módulo, ocorre a verificação da validade do cartão, ou seja, se a chave contida no mesmo consta no sistema (foi previamente cadastrada). Se o cartão for reconhecido pelo sistema, inicia-se a verificação de período de vigência de autorização, ou seja, verifica-se se a data atual está contida no intervalo definido pela data de início e data de fim de autorização. Em ambas as verificações, de maneira independente, o acesso poderá ser impedido. O *status* de autenticação do *smartcard* (identificação positiva ou negativa) é enviado para o módulo de autenticação de biometria digital.
- **Autenticação por biometria digital:** A partir do *status* de autenticação de *smartcard* recebido é disparada a autenticação por biometria digital. Nesse módulo, as impressões digitais do usuário são digitalizadas por meio do leitor biométrico e são comparadas com aquelas previamente cadastradas na base de dados, de modo a validar o usuário. O *status* de autenticação biométrica (se a impressão digital confere ou não com as do usuário do *smartcard* autenticado no módulo anterior) é enviado para o módulo de gravação de vídeo de *log*.

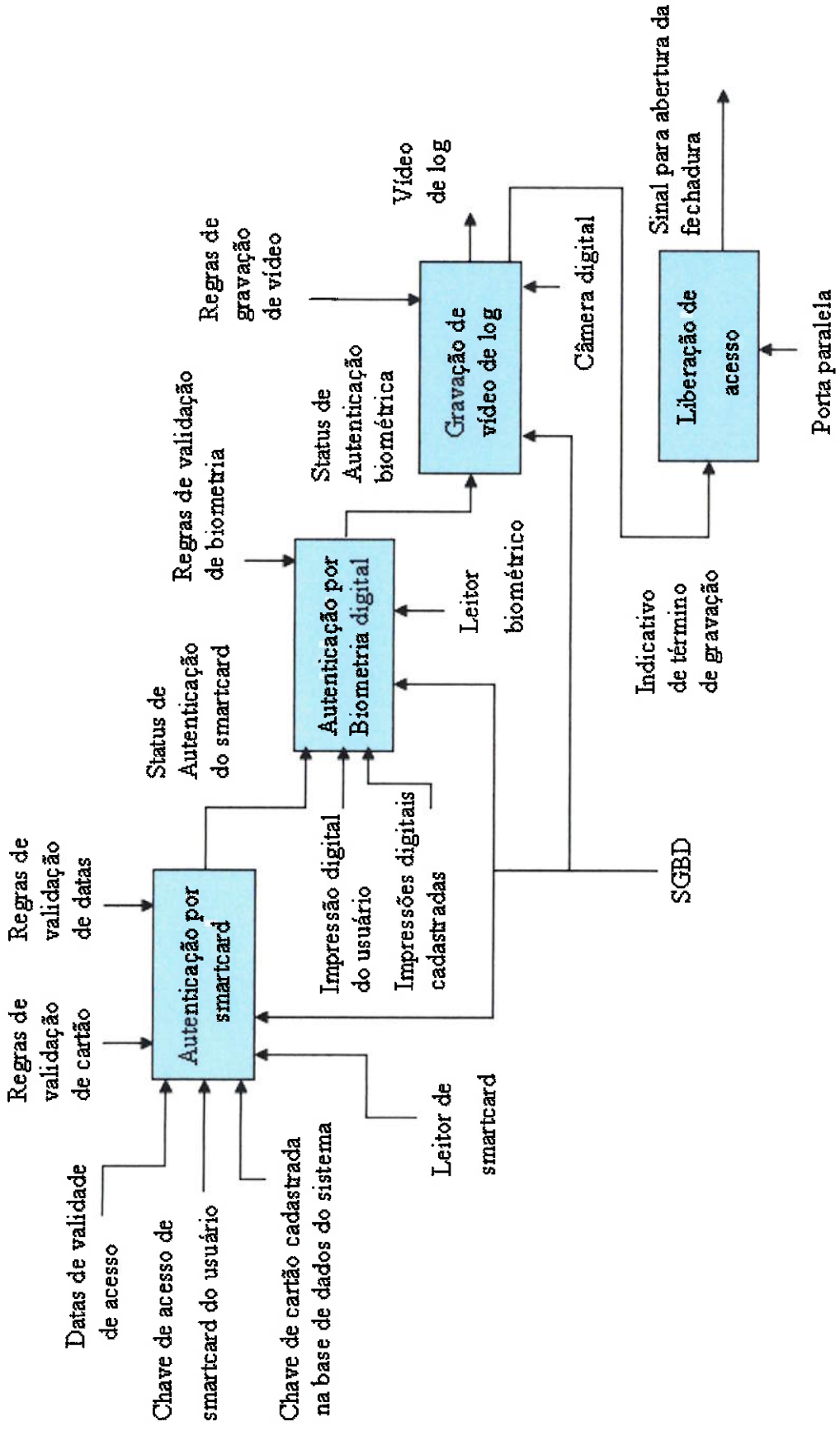

Figura 30 - Diagrama Nível 1 - Controle de Acesso

- **Gravação de vídeo de *log*:** A partir do *status* de autenticação biométrica recebido, é disparada a gravação do vídeo de *log*. Nesse módulo, caso a impressão digital confira, a câmera é ativada por um período de 3 segundos de maneira a registrar as imagens do usuário no momento do acesso. Tanto o vídeo gerado quanto informações de identificação dos mesmos são armazenados. Ao término desse módulo é enviada uma informação indicativa para o módulo de liberação de acesso.
- **Liberação de acesso:** A partir da informação indicativa de término de gravação, é disparado o módulo de liberação de acesso. Esse módulo é responsável pelo controle da fechadura eletrônica, ativando-a ou não por meio de um sinal de controle enviado pela porta paralela (saída desse módulo).

4.1.2 Módulo de gerenciamento do sistema

O módulo de gerenciamento do sistema é responsável por prover funcionalidades administrativas do sistema como início/parada do processo de identificação, gerência de vídeos, gerência de usuários, autenticação no sistema, cadastramento de cartões e cadastramento de impressões digitais. O diagrama nível zero deste módulo está representado na figura 31.

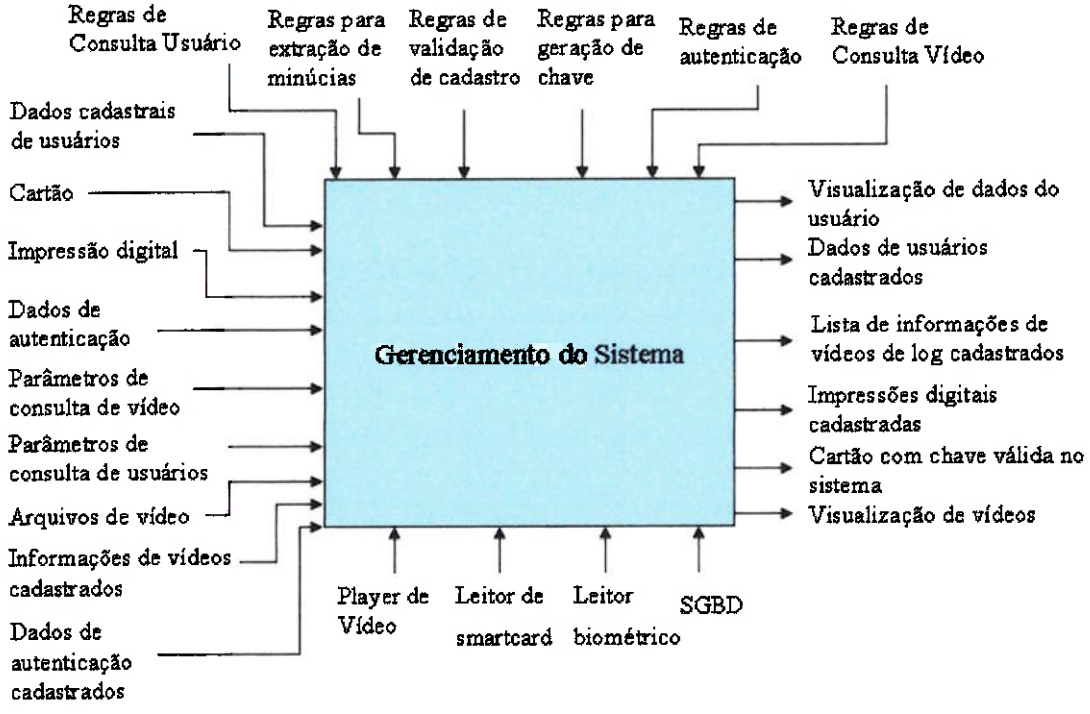

Figura 31 - IDEF0 Nível Zero Gerenciamento do Sistema

Entradas externas ao sistema:

- **Dados cadastrais de usuários:** informações cadastrais dos usuários que terão acesso ao local protegido pelo sistema.
- **Cartão:** *smartcard* a ser cadastrado no sistema, ou seja, o cartão em que será gravada a chave de acesso que o identifique no sistema.
- **Impressão digital:** impressões digitais de um usuário a ser cadastrado no sistema.
- **Dados de autenticação:** *login* e senha de um administrador do sistema para sua autenticação no módulo de gerenciamento.
- **Parâmetros de consulta de vídeo:** representa os parâmetros de busca que deverão ser informados pelo administrador ao serem consultados vídeos. Nesse sistema, os parâmetros são: data em que o vídeo foi gravado e/ou parte do nome do usuário a qual pertence o vídeo.
- **Parâmetros de consulta de usuários:** representa os parâmetros de busca que deverão ser informados pelo administrador ao serem consultados usuários. Nesse sistema o parâmetro deverá ser parte do nome (ou nome completo) de um usuário.
- **Arquivos de vídeo:** representa um arquivo de vídeo de *log* armazenado.
- **Informações de vídeos cadastrados:** representa as informações dos arquivos de vídeo. São elas: identificação de vídeo, data e hora da criação, caminho (endereço absoluto do arquivo de vídeo) e nome do arquivo de vídeo.
- **Dados de autenticação cadastrados:** representa as informações de *login* e senha contidas no sistema.

Saídas:

- **Visualização de dados do usuário:** representa os dados cadastrais de usuários sendo exibidos.
- **Dados de usuários cadastrados:** representa a persistência dos dados cadastrais (após a transação de gerenciamento de dados cadastrais de usuário)
- **Impressões digitais cadastradas:** representa a persistência dos dados biométricos de um usuário.
- **Cartão com chave válida no sistema:** representa o cartão com uma chave (gerada randomicamente) gravada em sua memória interna. Esta chave também estará armazenada na base de dados do sistema (cartão cadastrado no sistema).

- **Lista de informações de vídeos de *log* cadastrados:** representa o resultado da busca de vídeos, ou seja, uma lista contendo informações que identificam os vídeos (nome do usuário a qual pertence o vídeo, data e hora da criação do vídeo e opção de exibir o vídeo).
- **Visualização de vídeos:** representa a exibição de um determinado arquivo de vídeo selecionado através da lista de informações de vídeos de *log*.

Recursos:

- **Leitor biométrico:** dispositivo responsável pela digitalização das impressões digitais do usuário.
- **Leitor de *smartcard*:** dispositivo responsável pela leitura/gravação em *smartcards*.
- **SGBD:** corresponde ao software gerenciador de banco de dados responsável pela manipulação, persistência e integridade dos dados.
- ***Player* de vídeo:** *software* responsável pela exibição de vídeos.

Controle:

- **Regras de consulta usuários:** representa as regras usadas para realizar a busca pelos usuários cadastrados.
- **Regras para extração de minúcias:** representa as regras usadas na implementação do algoritmo de reconhecimento biométrico.
- **Regras de validação de cadastro:** regras para a consistência dos dados do cadastro. Exemplo: senha com no mínimo oito caracteres.
- **Regras para geração de chave:** representa as regras utilizadas na geração da chave randômica de 16 *bytes*.
- **Regra de autenticação:** representa as regras que regem o processo de autenticação do administrador no sistema.
- **Regras de consulta vídeo:** representa as regras utilizadas para buscar os vídeos cadastrados no sistema.

De modo a especificar melhor as funcionalidades do sistema, fornecendo assim uma visão mais detalhada do mesmo, o diagrama de nível zero foi explodido gerando o nível um. O diagrama de nível um está representado nas figuras 32 e 33:

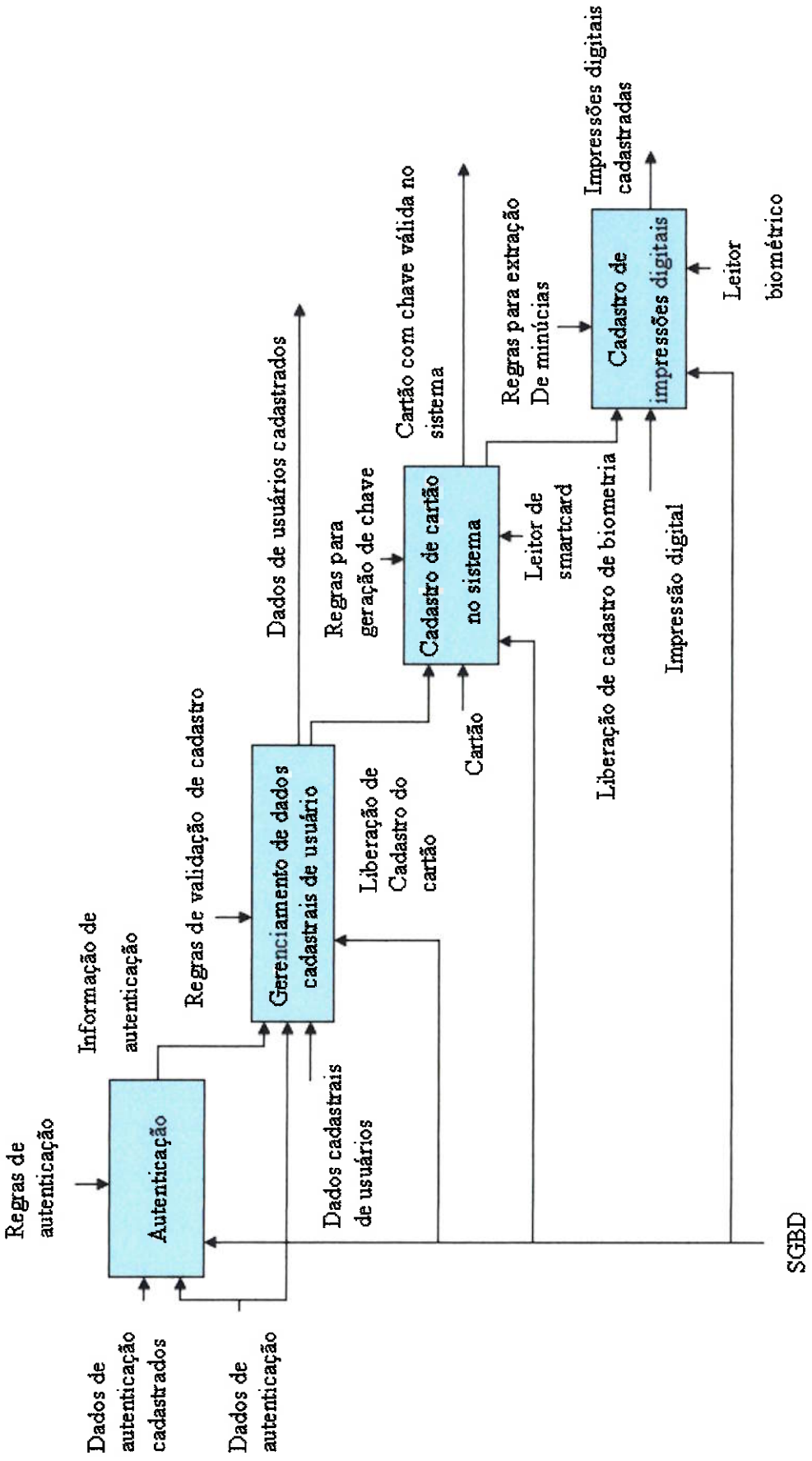

Figura 32 - IDEF0 Nível 1 - Gerenciamento do Sistema - Parte 1

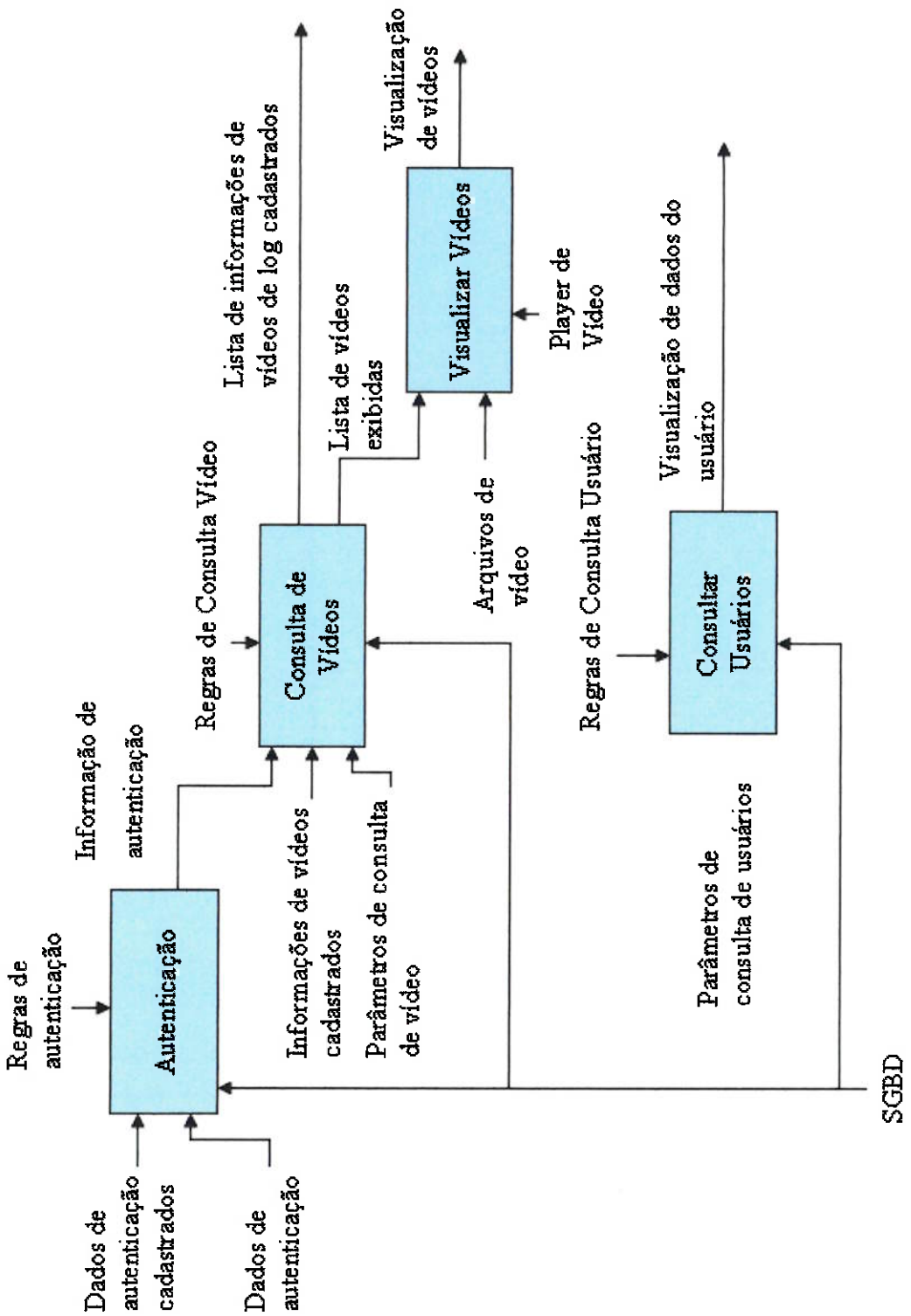

Figura 33 - IDEF0 Nível 1 - Gerenciamento do Sistema - Parte 2

A explosão do módulo de gerenciamento do sistema gerou os seguintes módulos:

- **Autenticação:** Esse módulo realiza a autenticação de um administrador no sistema a partir dos dados de autenticação (*login* e senha) fornecidos. A regra para autenticação consiste simplesmente na comparação entre os parâmetros de *login* e senha informados com os armazenados no sistema. A saída desse módulo consiste na informação de sucesso na autenticação (positiva/negativa).
- **Gerenciamento de dados cadastrais de usuário:** Esse módulo gerencia os dados cadastrais do usuário, ou seja, implementa funcionalidades como inclusão, alteração, remoção de usuários no sistema. Para que seja disparado esse módulo, é necessário que a autenticação tenha sido bem sucedida (informação positiva de sucesso na autenticação). Uma vez executado esse módulo (cadastrado o usuário), torna-se liberada a opção de cadastramento de cartão do usuário no sistema.
- **Cadastro de cartão no sistema:** Esse módulo gerencia cadastro de cartões no sistema, ou seja, a geração da chave randômica de 16 *bytes* e sua gravação na memória do cartão. Uma vez executado esse módulo (cadastrado o cartão no sistema), torna-se liberada a opção de cadastramento de impressões digitais do usuário.
- **Cadastro de impressões digitais:** Esse módulo é responsável pelo gerenciamento das impressões digitais cadastradas para um determinado usuário. Suas funções principais são: cadastrar uma ou mais impressões digitais (até 10 dedos) e remover uma impressão digital previamente cadastrada.
- **Consulta de vídeos:** Esse módulo é responsável pela exibição de uma lista com informações de vídeo para um ou mais usuários. Para que seja realizada a consulta é necessário que a autenticação tenha sido bem sucedida (informação positiva de sucesso na autenticação) e os parâmetros de consulta (parte do nome do usuário a qual pertence o vídeo e/ou data/hora de criação do vídeo) sejam informados.
- **Visualizar vídeos:** Esse módulo é responsável pela exibição de vídeo por meio do *player* de vídeo. Para que seja disparado esse módulo é necessário que antes sejam consultados os vídeos, pois é na lista de vídeos exibidos que irá aparecer um link para o arquivo de vídeo. A saída desse módulo é o *player* de vídeo exibindo o vídeo selecionado.
- **Consultar usuários:** Esse módulo é responsável pela visualização de dados de usuários. Para que seja realizada a consulta, é necessário que a autenticação tenha sido bem sucedida (informação positiva de sucesso na autenticação) e o parâmetro de

consulta (nome completo ou parte do nome do usuário(s) a serem consultados) sejam informados.

4.2 Modelagem

Esta seção apresenta os principais diagramas UML, que foram elaborados para o desenvolvimento do projeto de software.

4.2.1 Diagrama de Casos de Usos

As descrições dos casos de uso se encontram no anexo 1.

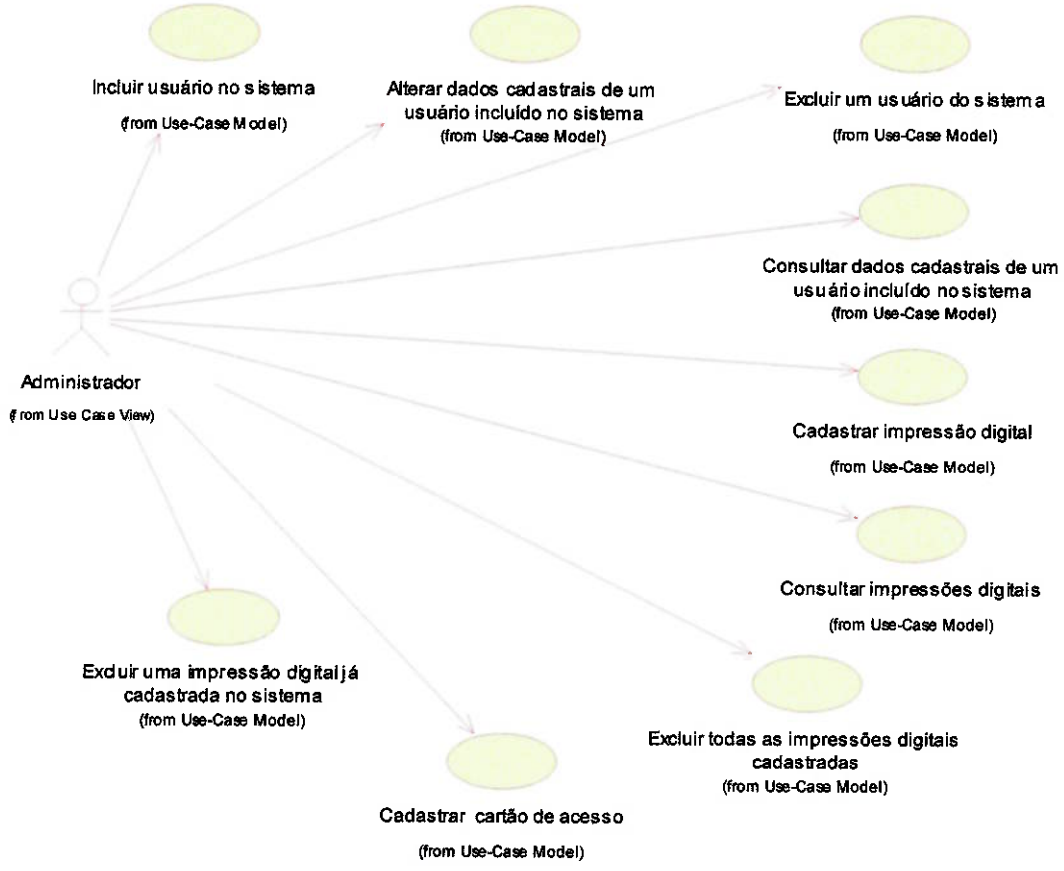

Figura 34 - Diagrama de Casos de Uso - Parte 1

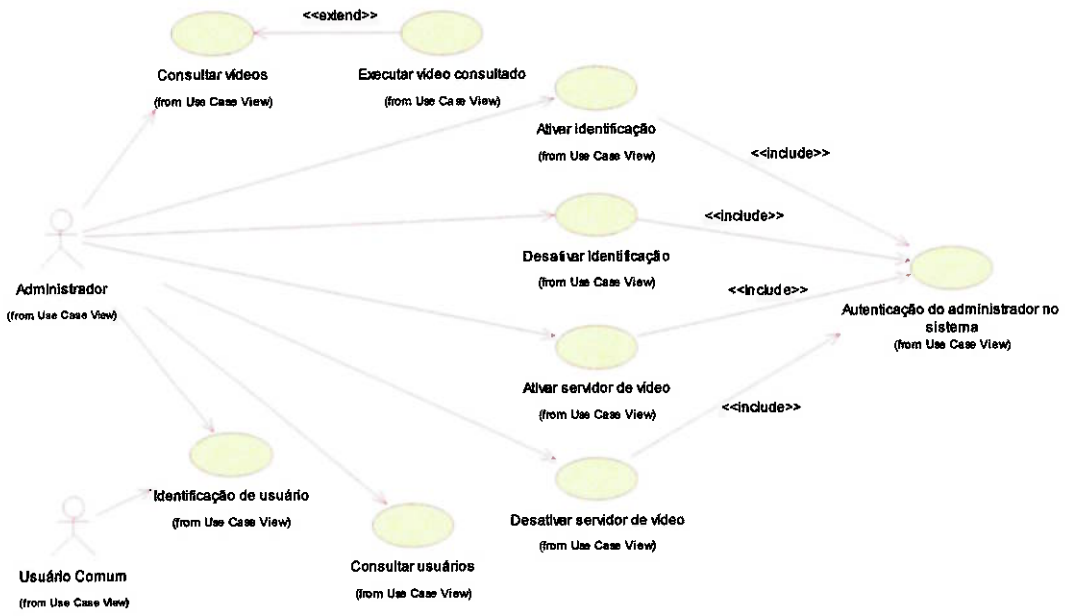

Figura 35 - Diagrama de Casos de Uso - Parte 2

4.2.2 Diagrama de Classes

A seguir são apresentados os diagramas de classe do sistema.

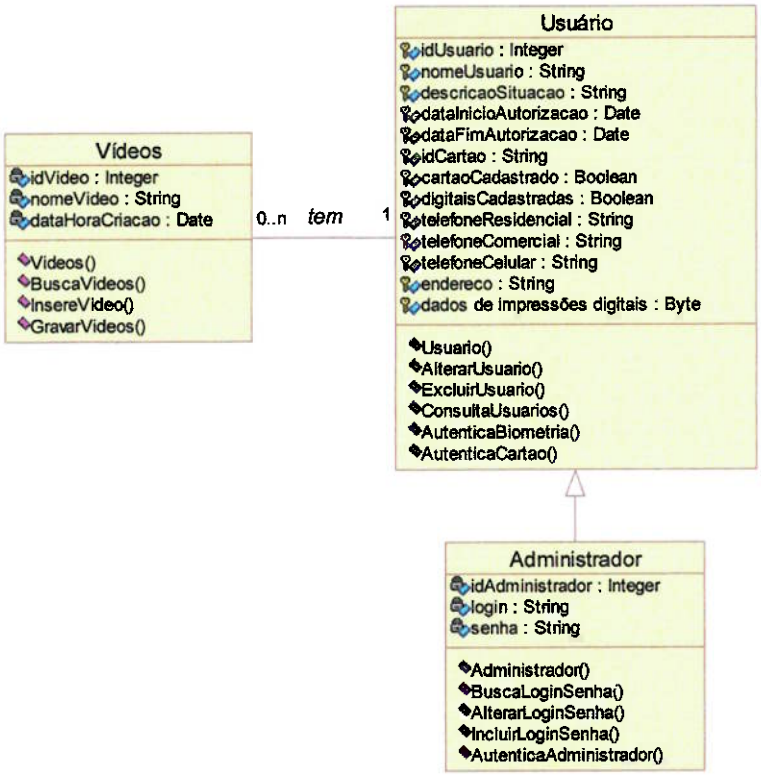

Figura 36 - Diagrama de classes alto nível

No início da modelagem, foi feito um primeiro diagrama de classes em alto nível (poucos detalhes - figura 36). Conforme o andamento do projeto os modelos tornaram-se mais detalhados, sendo necessário um novo diagrama de classes, mais próximo do nível de implementação (figura 37).

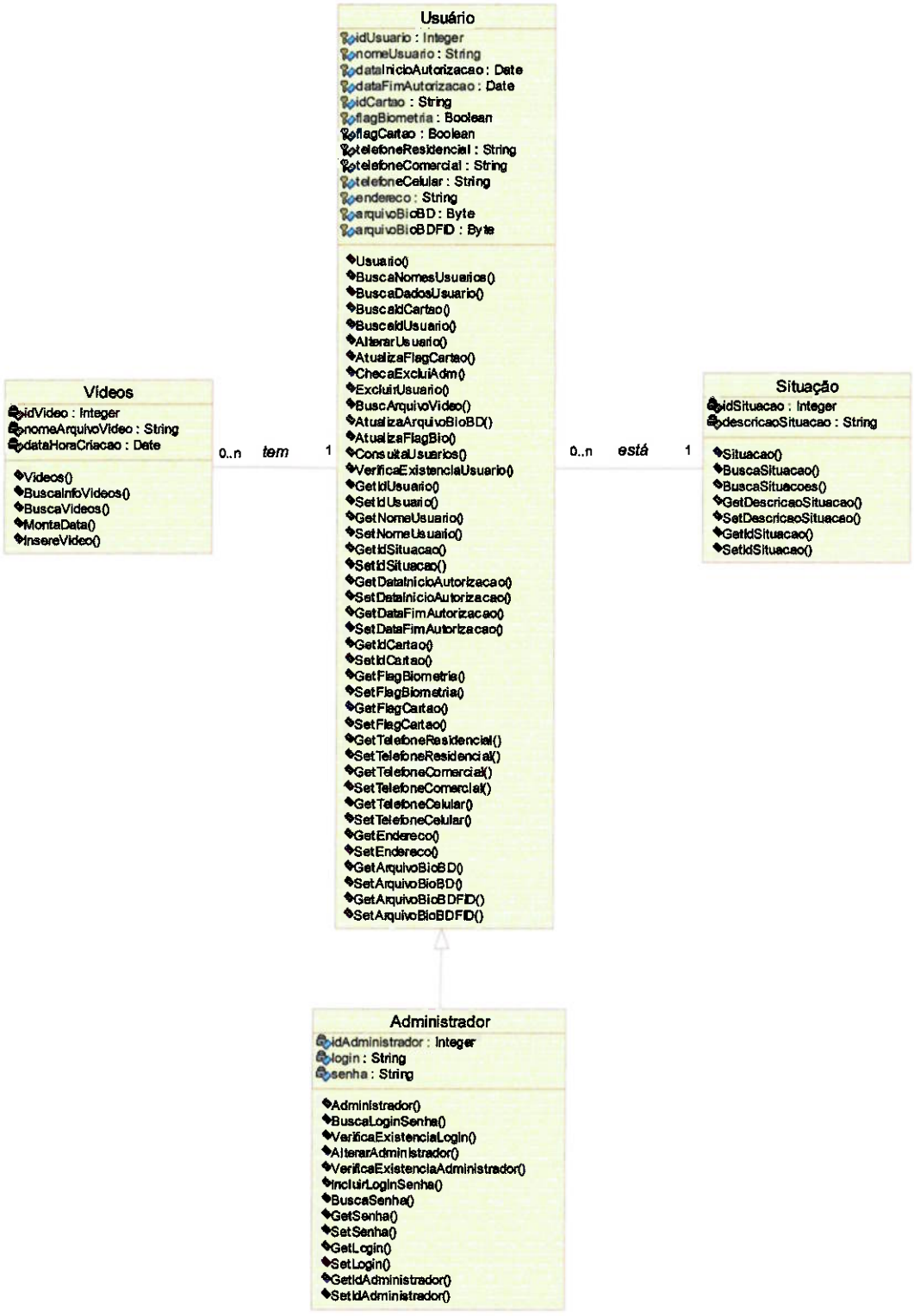

Figura 37 - Diagrama de Classes – Nível Implementação

4.2.3 Diagrama de Entidade-Relacionamento

A seguir, é apresentado o diagrama Entidade-Relacionamento, elaborado para a modelagem do banco de dados do projeto.

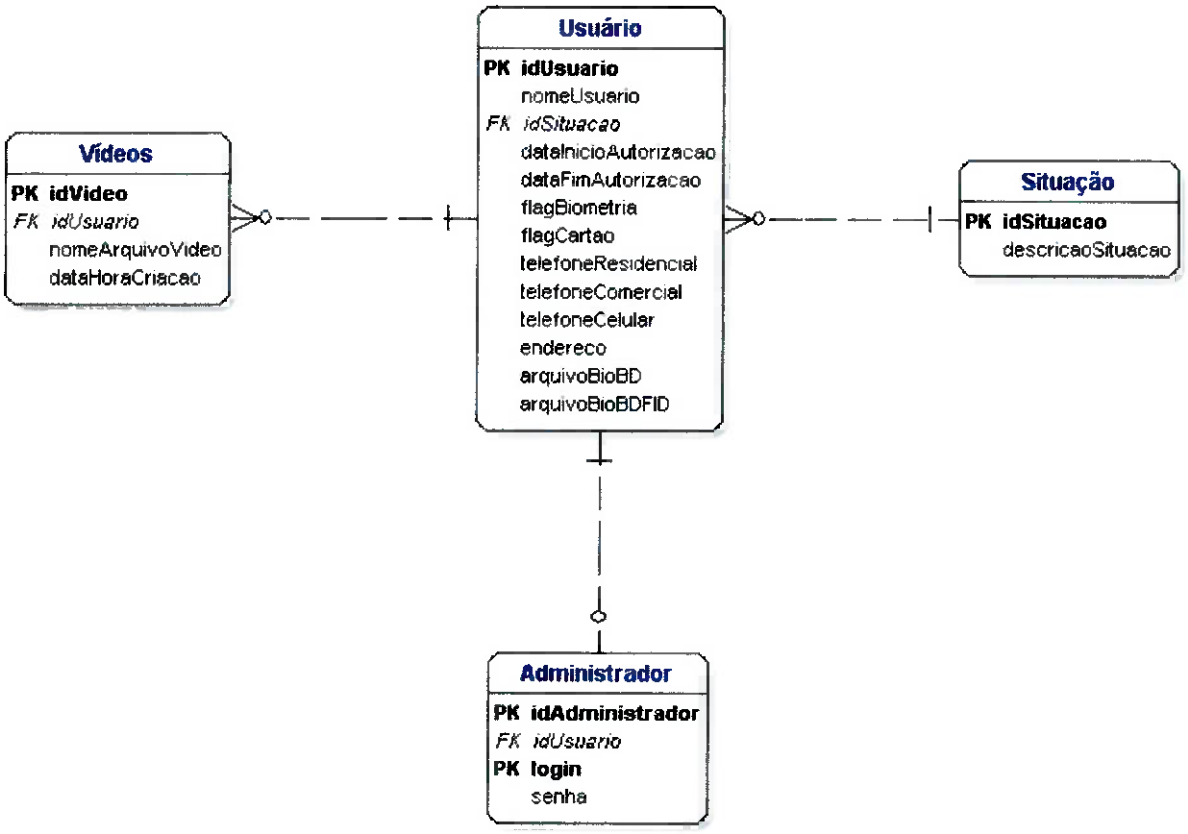

Figura 38 - Diagrama Entidade-Relacionamento

4.2.4 Diagrama de Navegação

Para o projeto das interfaces do sistema foi feito um protótipo em papel das telas, e em seguida, fez-se um diagrama de estados representando a navegabilidade do sistema. A partir dos protótipos de interfaces em papel e dos diagramas de estado foram projetadas as interfaces definitivas do sistema (em C#).

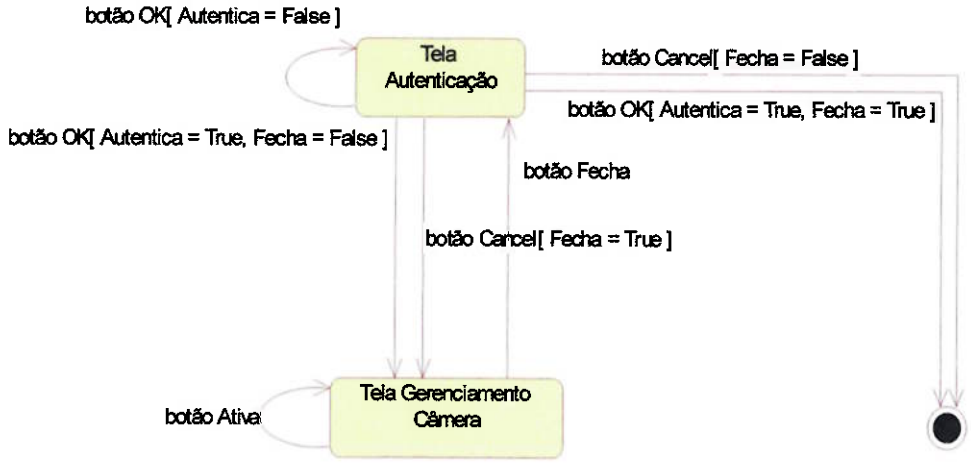

Figura 39 - Diagrama navegação - parte servidor de vídeo

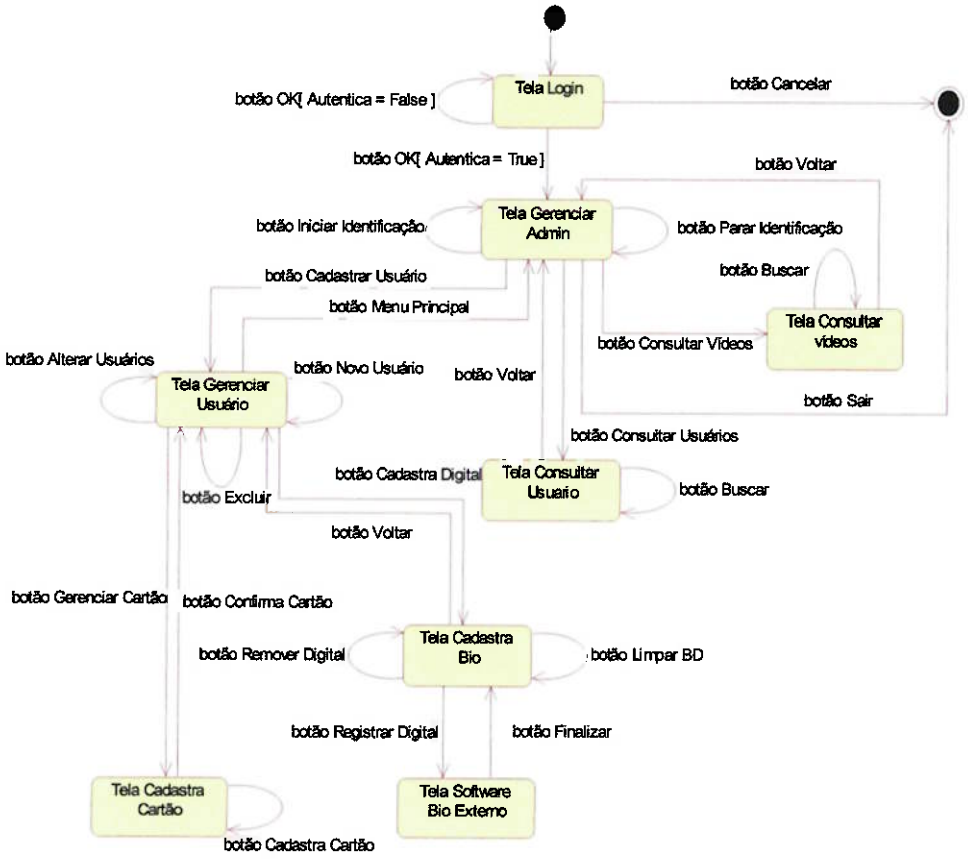

Figura 40 - Diagrama navegação - parte gerência

4.3 Arquitetura *Software*

O *software* foi desenvolvido a partir do modelo de arquitetura de três camadas (interface homem-máquina, controle e acesso ao BD) descrita na seção de conceitos. Os módulos Programa Cartão (módulos de leitura/escrita no cartão), Servidor Vídeo (módulo responsável

por receber requisições de gravação, capturar e salvar os vídeos) e Teste Vídeo (módulo que permite testes na câmera) estão separados das três camadas por razões tecnológicas (linguagens diferentes).

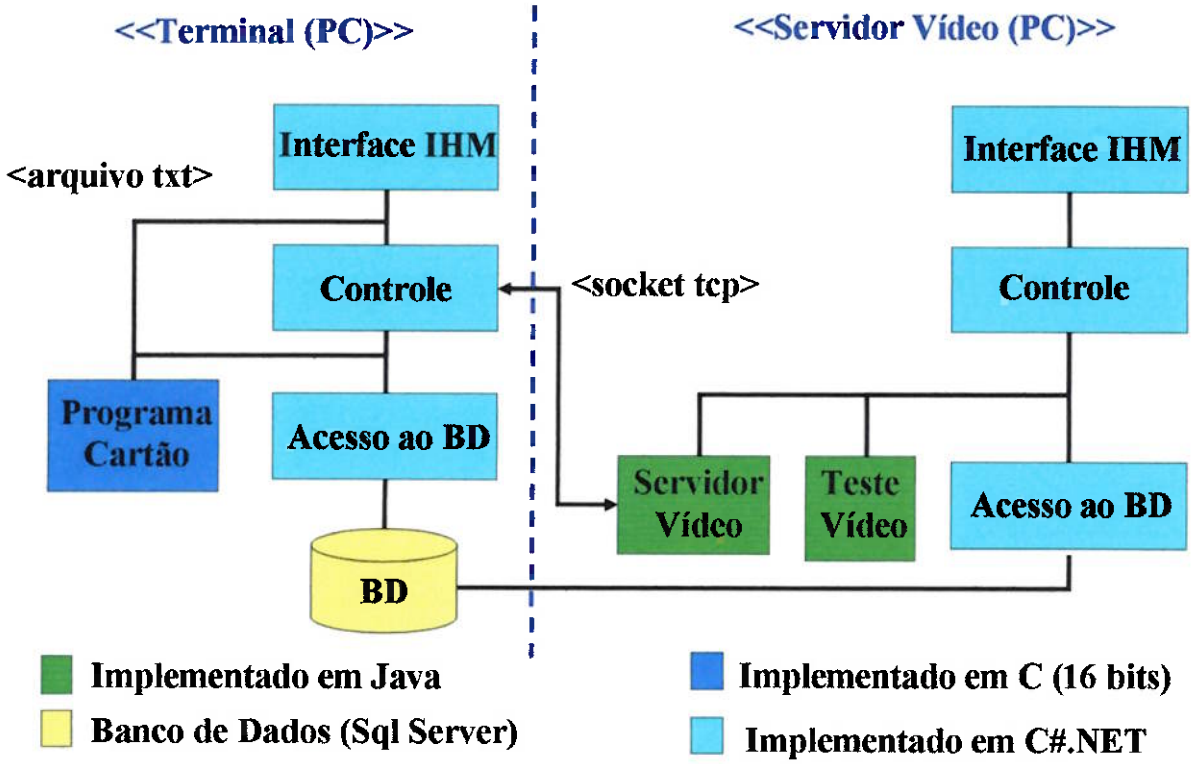

Figura 41 – Arquitetura de *Software*

A figura 41 representa além da arquitetura de *software*, o acoplamento dos módulos no *hardware*.

4.4 Arquitetura *Hardware*

A arquitetura de *hardware* apresenta a infra-estrutura física adotada para o funcionamento da solução. A figura 42 ilustra a arquitetura de *hardware* do sistema.

Figura 42 - Arquitetura de *Hardware*

Os componentes da arquitetura são:

- **Terminal** – terminal de acesso responsável pelo processo de identificação. Para este projeto foi utilizado um PC, de maneira a simular um terminal. O módulo de gerência/identificação e o banco de dados se encontram no terminal pelos seguintes motivos:
  - Existência de apenas um leitor de cartão de acesso e de um leitor biométrico para a realização deste projeto. Se no terminal ficasse apenas a funcionalidade de identificação, seria necessário um leitor de cartão para identificação e outro para o cadastro. A mesma idéia é válida para o leitor biométrico;
  - Não seria possível executar o processo de identificação caso ocorresse uma falha na rede (neste caso o banco de dados estaria indisponível);
  - Descentralização do sistema de segurança. Considerando que este sistema pudesse controlar o acesso a diversos locais, essa descentralização permitiria que cada terminal controlasse o acesso a um local independentemente dos outros, ou seja, em caso de falha apenas o local protegido pelo terminal que falhou seria afetado.

- **Fechadura eletrônica** – dispositivo que será acionado pelo sistema de modo a liberar o acesso;
- ***Scanner* biométrico** – dispositivo utilizado para a leitura das impressões digitais do usuário. Neste projeto, foi utilizado o *scanner* biométrico *Fingkey Hamster* III;
- **Leitor de *smarcards*** – dispositivo responsável pela leitura e gravação dos cartões de acesso. Neste projeto, foi utilizado o leitor MF-RD 260;
- **Servidor de vídeo** – é responsável pelo processamento do vídeo (captura e compressão). Neste projeto, foi utilizado um PC para implementar este servidor;
- **Câmera digital** – dispositivo de captura de vídeo. Neste projeto, foi utilizada a *Webcam* NX PRO.

5 Projeto e Implementação

A implementação do projeto esta descrita nos itens abaixo, baseada nos principais módulos desenvolvidos.

5.1 Banco de Dados

Para a implementação do banco de dados do sistema foi utilizado o MS SQL *Server* 2000. Foi utilizado esse SGBD pela sua robustez, segurança e, principalmente, pela sua fácil integração com a plataforma .NET (*Visual Studio* C#). A partir do diagrama E-R descrito anteriormente, foram implementadas as tabelas no MS SQL *Server* 2000. Segue abaixo um diagrama obtido a partir do SQL *Server Enterprise Manager*.

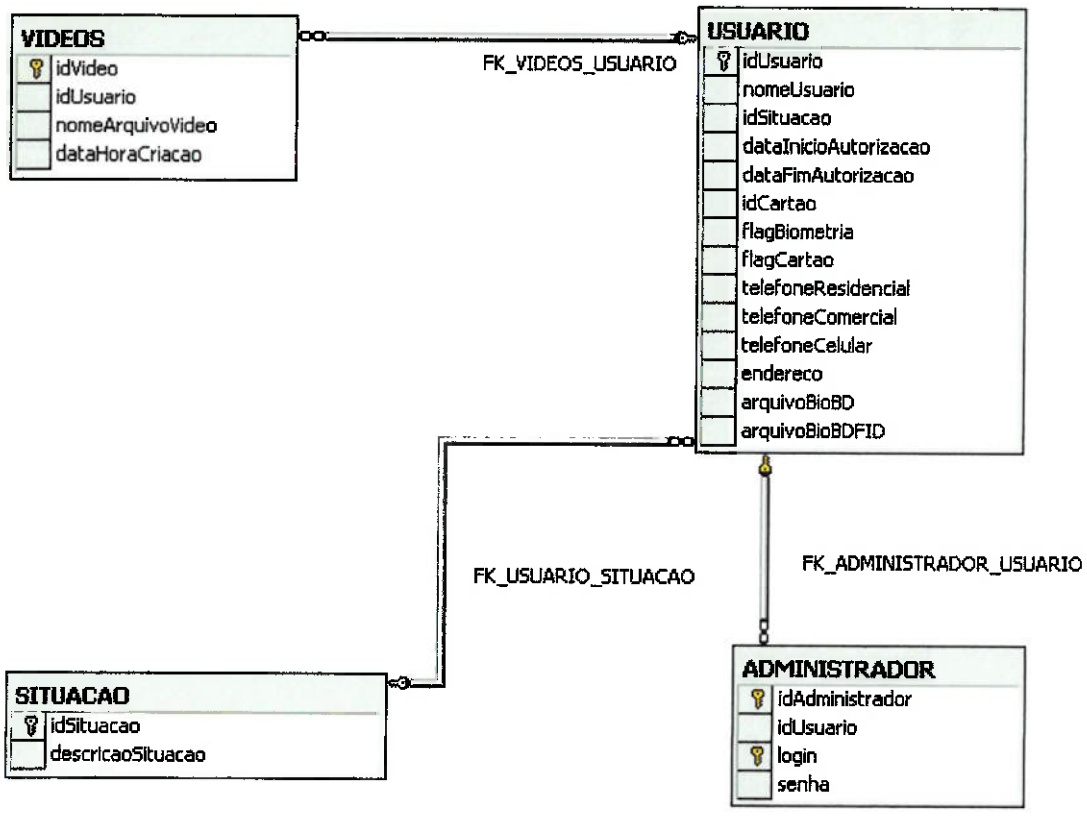

Figura 43 - Diagrama das tabelas do banco de dados extraído a partir do *Enterprise Manager* do MS SQL *Server* 2000

Com o objetivo de melhor descrever as tabelas implementadas segue a figura abaixo:

Tabela Administrador

Column Name	Data Type	Length	Allow Nulls
idAdministrador	int	4	
idUsuario	int	4	
login	varchar	30	
senha	varchar	20	

Tabela Situação

Column Name	Data Type	Length	Allow Nulls
idSituacao	int	4	
descricaoSituacao	char	40	

Tabela Videos

Column Name	Data Type	Length	Allow Nulls
idVideo	int	4	
idUsuario	int	4	
nomeArquivoVideo	varchar	250	
dataHoraCriacao	datetime	8	

Tabela Usuário

Column Name	Data Type	Length	Allow Nulls
idUsuario	int	4	
nomeUsuario	char	50	
idSituacao	int	4	
dataInicioAutorizacao	datetime	8	
dataFimAutorizacao	datetime	8	
idCartao	char	16	
flagBiometria	int	4	
flagCartao	int	4	
telefoneResidencial	varchar	10	✓
telefoneComercial	varchar	10	✓
telefoneCelular	varchar	10	✓
endereco	varchar	100	✓
arquivoBioBD	image	16	✓
arquivoBioBDFID	image	16	✓

Figura 44 - Tabelas do sistema extraídas a partir do *Enterprise Manager* do MS SQL *Server* 2000

É interessante ressaltar que os atributos `flagCartao` e `flagBiometria` da tabela Usuário são tipos booleanos que foram implementados como inteiros pois não há o tipo de dado bool (aceitando valores *true* e *false*) no MS SQL *Server* 2000. Assim, houve a necessidade de converter os valores desses flags no C# antes de serem gravados no banco de dados. Para isso foi utilizado o seguinte exemplo de código:

```
if (flagBiometria == true)
     flagBiometriaAux = 1;
else
     flagBiometriaAux = 0; //esse flagBiometriaAux será salvo
                           //no BD
```

Foram utilizados dois atributos (`arquivoBioBD` e `arquivoBioBDFID`) do tipo image na tabela Usuário para armazenar o arquivo de dados de impressões digitais (criado pela biblioteca de software do SDK - *Software Development Kit* - biométrico). São armazenados dois arquivos para cada usuário. Isso ocorre porque o SDK armazena as informações biométricas em dois arquivos distintos, um com informações de controle e o outro com as informações biométricas das impressões digitais propriamente ditas.

A tabela Situação foi criada por motivos tecnológicos de forma a normalizar as tabelas do sistema evitando repetição de informação.

5.2 Módulo de Vídeo

Como já foi mencionado em seções anteriores, os vídeos gravados devem ter uma qualidade aceitável de modo a possibilitar a identificação das pessoas. Considerando tal requisito e a partir da análise das diferentes qualidades de vídeo possíveis de serem geradas pela câmera, verificou-se que, para atender aos requisitos do projeto, os vídeos gravados deveriam ter as seguintes características: 30 quadros por segundo (máximo suportado pela *Webcam*), na resolução de 352 x 288 *pixels* (resolução máxima para capturar a 30 quadros por segundo), no formato RGB24 (24 *bits* ou 3 *bytes* – máxima profundidade suportada pela câmera no modelo de cor RGB).

Definidos os parâmetros de qualidade do vídeo (resolução, profundidade e a taxa de quadros) e cadastrado a *Webcam* no JMF *Registry* (veja seção sobre JMF), a implementação do módulo de vídeo foi dividida em cinco etapas distintas:

- Primeira etapa: desenvolvimento de um algoritmo para acesso a *Webcam* e visualização da mídia capturada;
- Segunda etapa: elaboração de um algoritmo para capturar e gravar o vídeo da *Webcam* por um período de tempo;
- Terceira etapa: aperfeiçoamento do algoritmo de gravação, através da inserção de rotinas para a compressão do arquivo de vídeo;
- Quarta etapa: elaboração do cliente e do servidor de vídeo;
- Quinta etapa: desenvolvimento de uma interface de controle para o servidor de vídeo;

5.2.1 Primeira etapa – Acesso a *Webcam* e a visualização do conteúdo

Para o desenvolvimento desta etapa foi utilizada a tecnologia Java, especificamente, o Java 2 SE versão 1.4.2. Esta decisão foi tomada baseada no fato da API Java *Media Framework* (JMF) fornecer uma infra-estrutura para o acesso a dispositivos de captura de mídias (como uma *Webcam*) e sua manipulação e também tendo em vista a complexidade envolvida na manipulação de vídeos e o prazo de tempo restrito para este projeto, não seria possível desenvolver rotinas que fornecessem as mesmas funcionalidades do JMF.

O acesso a *Webcam* e a visualização do vídeo foram feitos através da interface `Player` (`javax.media`). A interface `Player` permite a leitura de um *streaming* de entrada

(baseado em mídias) fornecido por um *dataSource* (associado a um dispositivo ou arquivo), realizando a renderização e o controle baseado no tempo do *streaming*. Esta interface oculta detalhes de acesso ao dispositivo e de manipulação de *streaming*.

Para instanciar o *player*, foi chamado o método `createPlayer` da classe `Manager` (`javax.media`), que utiliza como atributo a URL do dispositivo (que pode ser obtida no JMF *Registry*).

Após o tratamento dos estados iniciais do *player* foi utilizado o método `start` da interface `Player` de modo a começar a exibição. Para terminar a execução, utilizou-se o método `close` desta classe.

Para a exibição e controle do vídeo exibido foram utilizadas duas classes:

- `Component visualMedia`: responsável pelo componente visual gráfico para a exibição do vídeo. O componente `visualMedia` pode ser extraído do player através do método `getVisualComponente` da interface `Player`.
- `Component mediaControl`: componente que permite o controle do vídeo a ser exibido (executar, pausar). O componente `mediaControl` pode ser extraído do *player* através do método `getControlPanelComponent` da interface `Player`.

O resultado desta fase foi um programa que permite a visualização do vídeo capturado pela *Webcam*, baseado no código disponível em [39]. Este código, denominado `TesteCam`, é utilizado no programa final para realizar testes com a câmera, assim como permitir ao usuário ajustar a posição e o foco da câmera.

5.2.2 Segunda etapa – Gravação do vídeo

Desenvolvido um primeiro *player*, o próximo passo foi o desenvolvimento da rotina de gravação de vídeo (não comprimido).

O primeiro passo consistiu na seleção do dispositivo de captura desejado (*Webcam*). Os dispositivos de captura cadastrados no JMF *Registry* podem ser obtidos através do método `getDeviceList` da classe `CaptureDeviceManager` (`javax.media`), responsável por prover a lista de acesso a todos os dispositivos de captura cadastrados no sistema. A partir da lista de dispositivos, foi selecionado o dispositivo desejado (no caso desta implementação a seleção foi feita pelo índice do dispositivo, cujo valor é dois). A seleção de um índice da lista

de dispositivos retorna uma instância da classe `CaptureDeviceInfo` (`javax.media`), que representa o dispositivo.

Selecionado o dispositivo, o próximo passo foi associar um *dataSource* a este dispositivo, de maneira a acessar os dados capturados. A classe `dataSource` (`javax.media.protocol`) é responsável pela abstração dos protocolos das mídias e permite a transferência de dados entre duas localidades. Para realizar esta tarefa deve-se instanciar um *dataSource* e associá-lo ao dispositivo através do método `createDataSource` da classe `Manager` (`javax.media`). Este método possui como argumento uma instância da classe `MediaLocator` (`javax.media`), que descreve a localização do conteúdo de mídia (ou seja do dispositivo). Este argumento pode ser obtido através do método `getLocator` da classe `CaptureDeviceInfo`.

O próximo passo consistiu na seleção do formato de captura (formato do vídeo) realizado pelo dispositivo, uma vez que este suporta diversos formatos de gravação (pode-se, por exemplo, selecionar a resolução do quadro, o número de quadro por segundos). Utiliza-se a classe `Format` (`javax.media`) para armazenar o formato. Para instanciar esta classe utiliza-se o construtor da classe `VideoFormat` (`javax.media.format`), sendo passados os seguintes parâmetros como argumentos:

- *Encoding*: `VideoFormat.RGB` – codificação RGB24;
- *Size*: `new Dimension(320, 240)` – tamanho de vídeo de 320 x 240 *pixels*;
- *maxDataLenght*: 230400 – tamanho máximo do dado permitido pela API JMF;
- *dataType*: `Format.byteArray`;
- *frameRate*: `30f` – 30 quadros por segundo (o f representa *frame*);

Escolhida a configuração do dispositivo de captura, o próximo passo foi a definição do formato do descritor do arquivo de vídeo. Utilizou-se a classe `FileTypeDescriptor` (`javax.media.protocol`) para armazenar o descritor do arquivo. A escolha do descritor é feita durante a construção da classe, através de métodos estáticos desta classe os quais contêm a definição de cada formato de descritor. Foi utilizado o formato `FileTypeDescriptor.MSVIDEO` que representa conteúdos do tipo AVI.

Definidas as entradas, configurou-se o processo de captura. Para realizar a captura propriamente dita foi utilizada a interface `Processor` (`javax.media`). Para criar uma instância da interface `Processor` procedeu-se do seguinte modo:

- Instanciou-se a classe `ProcessorModel` (`javax.media`), que fornece as informações básicas para a criação do processo. Para criar esta instância do `ProcessorModel` foi necessário passar o *dataSource* associado ao dispositivo, o formato de saída (classe `Format`) e descritor do arquivo de vídeo (classe `FileTypeDescriptor`) criados nos passos anteriores.
- Foi passada a instância da classe `ProcessorModel` para o método `createRealizedProcessor` da classe `Manager` (`javax.media`), responsável pela construção da interface `Processor`.

Com o *Processor* de captura instanciado, o último passo consistiu na gravação do vídeo em um arquivo. Para realizar este passo precedeu-se da seguinte forma:

- Abriu-se o arquivo onde o vídeo será gravado (com a extensão .AVI) usando a classe `File` (`java.io`). Associou-se o arquivo de vídeo a uma instância da classe `MediaLocator` (`javax.media`). Esta associação pôde ser feita através do construtor desta classe e por meio da URL do arquivo.
- Associou-se o *Processor* a um *dataSource* de saída. Para tanto foi necessário instanciar um *dataSource* de saída e associá-lo ao *dataSource* obtido do método `getDataOutput` da classe `Processor`, que fornece o *dataSource* resultante do processo de captura.
- Gravou-se os dados do *dataSource* de saída no arquivo de vídeo. O responsável por armazenar o *dataSource* num arquivo é a interface `dataSink` (`javax.media`). No projeto foi utilizado uma instância da interface `dataSink` e adicionado a esta um `DataSinkListener` (interface que permite o tratamento de eventos, como fim de *streaming*) através da função `addDataSinkListener` da interface `dataSink`. Após instanciar o `dataSink`, foi utilizado o método `open` seguido do `start` da interface `dataSink` para iniciar a gravação no arquivo. O `dataSink` pôde ser fechado pelo método `close` durante o disparo do evento fim de *streaming*.

Para iniciar a captura foi necessário utilizar o método `start` da interfase *Processor*. Para finalizá-la foi utilizada a função `stop` e `close` da interface *Processor*.

Para esperar o tempo de 3s de gravação, a Thread principal foi parada pelo método estático `sleep(3100)` da classe `Thread`. Como o *Processor* executa em outra Thread, o processo de gravação não será interropido. Após o comando de `sleep`, a gravação é encerrada.

Ao término desta etapa, foi desenvolvida uma classe denominada `CapturePlayer`, baseada no código disponível em [39], que realiza o processo de captura de vídeo através do método `roda (String nome)`, onde o `string nome` representa o nome do arquivo de vídeo que será capturado.

5.2.3 Terceira Etapa – Rotinas de Compressão

O tamanho do arquivo do vídeo gravado na segunda etapa é inadequado para o armazenamento em um sistema de vídeo de *log*, sendo assim, foi necessário comprimir o vídeo.

Para realizar esta compressão foi utilizado um programa externo, gratuito, denominado VDub (Virtual Dub). Esta decisão está baseada nos seguintes fatores: a API Java *Media Framework* não possui suporte ao compressor Xvid e implementar tal suporte seria inviável devido a limitação de tempo do projeto.

O acesso ao programa VDub por meio da linguagem Java foi feito utilizando a interface baseada em linha de comando (*shell*) através do executável "vdub.exe". O programa Java fornece as linhas de comando necessárias para a execução da atividade desejada (no caso realizar a compressão de vídeo).

Foi utilizada a seguinte linha de comando para execução da compressão: "vdub.exe /i xvid.vcf /p videodescomprimido.avi videocomprimido.avi /r", onde:

- /i xvid.vcf – este argumento carrega o script "xvid.vcf" (arquivo de configuração dos parâmetros para a tarefa de compressão);
- /p videodescomprimido.avi videocomprimido.avi - este argumento especifica o arquivo de origem (videodescomprimido.avi) e o arquivo de destino videocomprimido.avi;
- /r – argumento que inicia a execução da tarefa definida pelo script.

Para acessar o VDub a partir da linguagem Java, foi utilizada a classe `Runtime` (`java.lang`). O método `Runtime.getRuntime().exec` ativa um processo externo ao programa, retornando um dado do tipo `Process` (`java.lang`) que permite o controle deste processo (tais como: parar o processo, receber a variável de retorno). Este método apresenta diversos tipos de argumentos (sobrecarga de método), entretanto foi utilizado o que recebe apenas um único string.

O string atribuído ao método ativa o processo *command* do *Windows* (linha de comando), seguido de duas instruções: o acesso ao diretório do VDub e o comando para ativar a operação de compressão explicado anteriormente.

O retorno do método `Runtime.getRuntime().exec` foi atribuído a uma variável do tipo *Process*. Em seguida a chamada do processo externo, foi utilizado o método `waitfor` da classe `Process`, que aguarda o término da execução do processo (caso não se use esse comando, a execução do processo é feita em paralelo a execução do programa Java).

O código de compressão resultou em uma função denominada `compactar()`, utilizada logo após o término da gravação do vídeo.

5.2.4 Quarta Etapa – Elaboração Cliente-Servidor

Uma vez que ficou decidido que a linguagem do *software* de integração do sistema C# (devido à facilidade de desenvolvimento de interfaces gráficas e disponibilidade do SDK da biometria digital em C#), foi necessário dividir o módulo de vídeo em um módulo Java e um módulo C#.

Para prover a comunicação e a interoperação entre os dois módulos, assim como transferir o processamento relacionado à captura do vídeo do terminal para o servidor de vídeo foi utilizada a arquitetura cliente-servidor, implementada através de *sockets* no módulo de vídeo.

A parte cliente utilizou a tecnologia C# e realiza as seguintes atividades:

- Requisita conexão ao servidor de vídeo;
- Requisita a gravação do vídeo;

- Recebe o vídeo gravado e o armazena no banco de dados, junto com informações de data de gravação, nome do usuário associado ao vídeo e nome do vídeo (incluindo o caminho – endereço absoluto do arquivo de vídeo);
- Fecha a conexão com o servidor.

Para a implementação do cliente, foram utilizadas as bibliotecas `System.Net.Sockets` e `System.IO`. O cliente se conecta ao servidor através da classe `TcpClient` (`System.Net.Sockets`). Esta classe, através de seu construtor, cria um *socket* do tipo cliente e se conecta ao servidor (pelo nome ou IP do *host* e pela numeração da porta).

Uma vez conectado (criação de uma instância de `TcpClient`) foi possível simplificar o processo de envio e recebimento de mensagens para uma operação de entrada e saída. Para tanto, foi associado o *stream* do *socket* TCP com uma instância da classe `StreamWriter` (`System.IO`), responsável pela escrita de caracteres em um determinado *stream*, e com uma instância da classe `StreamReader` (`System.IO`), responsável pela leitura de caracteres em um determinado *stream*. Esta operação é demonstrada abaixo:

```
StreamWriter sw = new StreamWriter (
                    tcpclientss.GetStream());
StreamReader sr = new StreamReader (
                    tcpclientss.GetStream());
```

Instanciados o `StreamReader` "sr" e o `StreamWriter` "sw", os métodos destas classes (como `WriteLine` – que permite escrever uma linha no *stream* a ser enviado – ou `ReadLine` – que permite ler uma linha do *stream* recebido) podem ser utilizados para enviar e receber mensagens.

Para o recebimento do vídeo (arquivo binário) via *socket* foi utilizada a classe `BinaryReader` (`System.IO`), associada ao *stream* do *socket* TCP de maneira análoga a classe `StreamReader` (esta classe não pôde ser utilizada, visto que trata os dados recebidos como caracteres ASCII). Uma vez recebido o *stream* de vídeo, este é armazenado num arquivo através de uma instância da classe `BinaryWriter` (`System.IO`), responsável pela escrita de dados binários em *stream* (*stream* do arquivo), associada a uma instância da classe `FileStream` (`Sytem.IO`), responsável pelo acesso ao arquivo.

Por fim, uma vez recebido o vídeo, a instância da classe `BinaryWriter` deve ser fechada pelo método `close` desta classe (para efetivar a gravação do arquivo), e o *socket* TCP deve ser fechado pelo método `close` da classe `TcpClient` para desconectar o cliente do servidor.

A implementação do cliente resultou na classe `client.cs` e na função `InsereVideo (int idusuario, string path, DateTime data)` na entidade vídeo desenvolvida para o uso nas rotinas de identificação.

A parte que representa o servidor utilizou a tecnologia Java e realiza as seguintes atividades:

- Espera de conexões do cliente;
- Ao receber conexões, realiza a gravação do vídeo e envia o vídeo ao cliente;
- Retorna ao estado de esperar de conexões por parte do cliente.

Para a implementação do servidor foram utilizadas as bibliotecas `java.io` e `java.net`. No construtor da classe servidora de vídeos, existe um único argumento que representa o valor da porta utilizada pelo servidor. Este argumento é utilizado pelo construtor da classe `ServerSocket` (`java.net`), que representa o *socket* do tipo servidor (por se tratar de uma aplicação servidora, não existe a necessidade do nome ou IP do *host*). O construtor, internamente, realiza as funções de `bind` e `listen`.

Criado o *socket* servidor, este espera a conexão do cliente. Uma vez que o cliente se conecta, é criado um *socket* (classe `Socket` – `java.io`) a partir do método `accept` do socket servidor. Este *socket* será utilizado pelo servidor para a comunicação com o cliente.

Assim como o cliente em C#, é possível utilizar classes e métodos de entrada e saída para enviar ou receber mensagens do cliente. Na implementação do servidor, foram utilizadas as classes `BufferedReader` (`java.io`) para a leitura das mensagens que chegam e a classe `PrintWriter` (`java.io`) para o envio de mensagens ao cliente. Para associar o *stream* da rede com as instâncias destas classes, foram utilizados os seguintes códigos (ss representa a instância de `ServerSocket`):

```
"BufferedReader br = new BufferedReader (
                        new InputStreamReader(
                            ss.getInputStream()))"
```

```
"PrintWriter wr = new PrintWriter (
                    new OutputStreamWriter(
                      ss.getOutputStream()),true)"
```

Após a troca das primeiras mensagens, o servidor executa o método de captura do vídeo. Após executado este método, utiliza-se a classe `RandomAccessFile (java.io)` para abrir o arquivo com o vídeo comprimido, obtendo assim o tamanho do arquivo (método `length` da classe `RandomAcessFile`) e armazenando todo seu conteúdo em um vetor de *bytes* (através do método `readFully(buffer)` da classe `RandomAcessFile`).

O tamanho do arquivo é utilizado para redimensionar o tamanho do buffer de envio do *socket* (através do método `setSendBufferSize(tamanho)` da classe `Socket`). Caso o tamanho do buffer de envio não seja redimensionado, o arquivo poderá não ser enviado (devido o tamanho do buffer a ser enviado ser maior do que a capacidade do buffer do *socket* – é lançada uma exceção). O envio do arquivo é feito usando a classe `PrintStream (java.io)` adequada para transmissão de arquivos binários. Esta classe pode ser instanciada através do seguinte código:

```
PrintStream ps = new PrintStream(
                   socket.getOutputStream(),true).
```

Conhecidos os detalhes de implementação de cada parte foi elaborado o protocolo de comunicação entre o cliente e o servidor. Este protoloco está baseado em trocas de mensagens definidas. A figura 45 representa o protocolo de comunicação (mensagens trocadas entre o cliente e o servidor) e os principais eventos associados a uma requisição de serviço por parte do cliente.

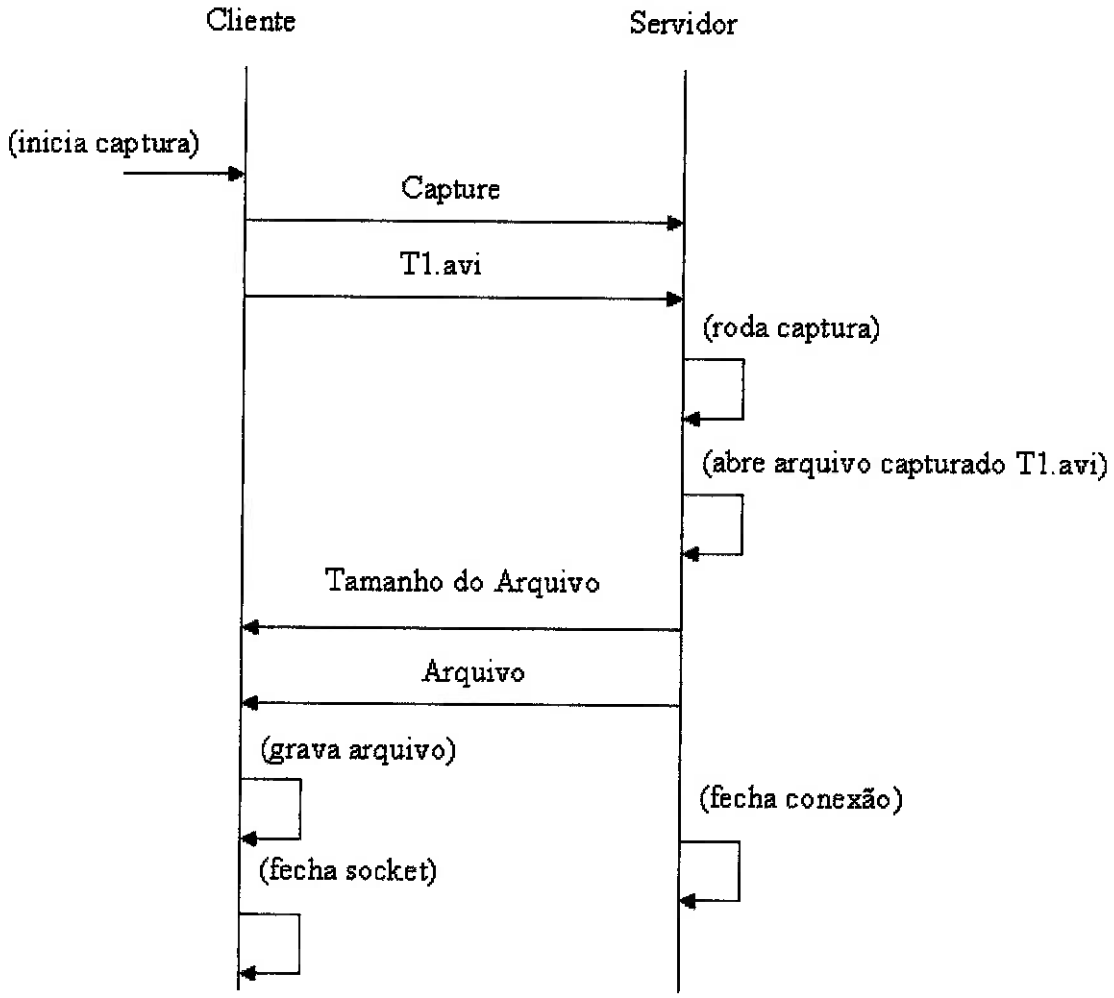

Figura 45 - Protocolo de mensagens e eventos cliente-servidor

5.2.5 Quinta Etapa – Interface de Controle para o Servidor de Vídeo

A interface de controle para o servidor de vídeo tem como objetivo:

- Desenvolver uma interface gráfica para o módulo de vídeo para o servidor de vídeo desenvolvido em Java. Para manter a consistência da interface gráfica do módulo de gerência e do módulo de vídeo, foi desenvolvida uma interface gráfica em C# para o servidor java. Outro fator que motivou a escolha da elaboração da interface gráfica em C# foi a facilidade de programação de interfaces nesta tecnologia. A interface gráfica permite abrir o servidor na porta desejada, executar o programa servidor de vídeo em segundo plano (através do método `Hide` herdado do `Windows.form`), dentre outras funcionalidades.

- Implementar um mecanismo de autenticação que restrinja o acesso aos recursos do servidor java. Com o mecanismo de autenticação, apenas os administradores do sistema podem iniciar ou parar o servidor java.

Cumpre ressaltar que o modelo de integração entre a linguagem Java e C# desenvolvido está baseado na seguinte característica: a única mensagem que o módulo Java (servidor de vídeo) e a interface em C# trocam é a porta em que o servidor de vídeo irá executar. Logo, a integração dos dois módulos pode ser feita utilizando-se o recurso de chamada a processos externos disponível na linguagem C#.

Para chamar um processo externo em C# foi utilizada a classe `Process` (`System.Diagnostics`). Uma vez instanciada, foram configurados os seguintes parâmetros:

- *WorkingDirectory* – diretório em que está localizado o processo a ser executado;
- *FileName* – nome do arquivo a ser executado;
- *Arguments* – argumentos a serem passados;
- *UseShellExecute* – parâmetro para determinar se utiliza ou não o *shell* do sistema operacional para chamar o processo;
- *RedirectStandardOutput* – determina se as saídas do processo devem ser armazenadas.
- *noCreateNoWindow* – determina uma janela será ou não criada para o programa (caso não seja, as mensagens em modo console não podem ser visualizadas).

Configurados os parâmetros, foi utilizado o método `start` da classe Process de modo a iniciar o processo. Para parar o processo, foi utilizado o método `kill` desta mesma classe. Abaixo segue um exemplo da chamada de um processo externo usado no sistema SysSec:

```
proc = new System.Diagnostics.Process();
proc.StartInfo.WorkingDirectory =
          "c:\\JavaApp\\TesteCam\\build\\classes";
proc.StartInfo.FileName = "java";
proc.StartInfo.Arguments = "TesteCam";
proc.StartInfo.UseShellExecute = false;
proc.StartInfo.RedirectStandardOutput = false;
proc.StartInfo.CreateNoWindow = true;
```

```
proc.Start();
```

Com o desenvolvimento deste módulo, foi obtido a parte do sistema SysSec que executa no servidor de vídeo.

5.2.6 Organização das classes no módulo de vídeo

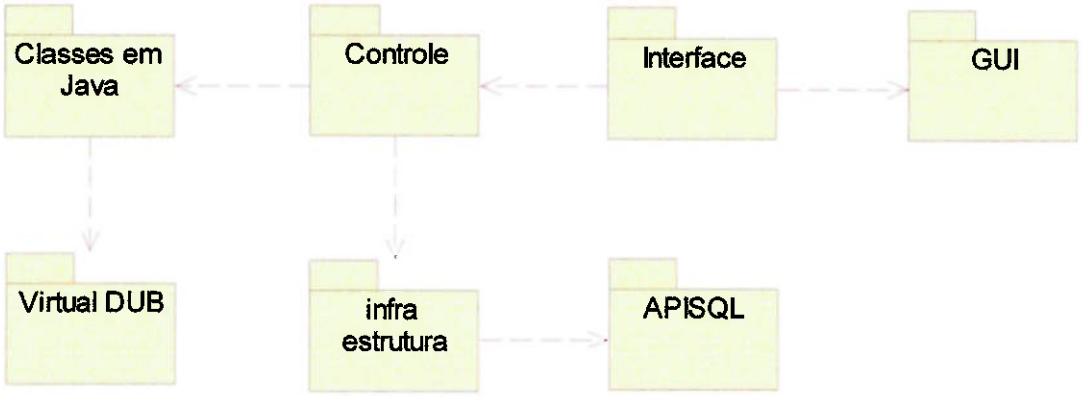

Figura 46 - Pacotes Módulo de vídeos

As classes do módulo de vídeo foram agrupadas em sete pacotes: Controle, Interface, APISQL, GUI, InfraEstrutura, Classes em Java e Virtual DUB. No *package* GUI estão as classes de componentes gráficos (botões, formulários, listbox, grids, entre outros). No *package* API SQL estão as classes para acesso ao sistema gerenciador de banco de dados. No package Interface estão todas as classes representativas dos formulários utilizados na aplicação gerenciadora. São elas: `frmGerCamera`, `frmMsg`, `frmTeste` e `frmSenha`. No *package* Controle estão as classes que implementam a lógica de negócio. São elas: `Autenticacao`, `ServerExec` e `TesteCamExec`. No *package* InfraEstrutura estão todas as classes que acessam o banco de dados. São elas: `AutenticacaoBD` e `Conexão`. No package Classes em Java estão todas as classes responsáveis pela gravação do vídeo. São elas: `TesteCam`, `CapturePlayer` e `JServer`. O *package* Virtual DUB representa o programa Virtual DUB responsável por chamar o compressor de vídeo (XVID) e gerenciar o processo de compressão.

5.3 Módulo de Biometria

Para o desenvolvimento do módulo de autenticação biométrica foram utilizadas bibliotecas pertencentes ao SDK (*Software Development Kit*) fornecido juntamente com o leitor biométrico *Hamster* III. Estas bibliotecas são proprietárias, de forma que o desenvolvedor só tem acesso às API's para utilização de suas funções [40].

A biblioteca utilizada no projeto foi a `NITGEN.SDK.NBioBSP.dll` fornecida para auxiliar o desenvolvimento de uma aplicação usando C# em *windows form*. Esta biblioteca utiliza outras bibliotecas como `NBioBSP.dll` de forma a fornecer ao desenvolvedor interfaces de alto nível, facilitando o uso do leitor biométrico.

Inicialmente, para se ter acesso ao leitor biométrico e todas às funcionalidades desejadas na autenticação (verificação, cadastramento biométrico, entre outras) foi instanciada a classe principal `NBioAPI` da biblioteca. Para utilizar os recursos fornecidos por esta biblioteca, assim como criar um arquivo de dados que contém as informações biométricas foi instanciada a classe `NBioAPI.Nsearch`.

```
// Cria objeto NBioBSP
m_NBioAPI = new NBioAPI();
m_NSearch = new NBioAPI.NSearch(m_NBioAPI);
```

5.3.1 Abertura de conexão com o leitor biométrico

Para usar as funções que controlam o leitor biométrico deve-se primeiro abrir uma conexão com o aparelho, para isso foi utilizada a função `OpenDevice` :

```
m_NBioAPI.OpenDevice(NBioAPI.Type.DEVICE_ID.AUTO);
```

Por meio do parâmetro `NBioAPI.Type.DEVICE_ID.AUTO`, a função buscará o último aparelho usado, caso haja múltiplos aparelhos conectados ao PC, e abrirá uma conexão com ele.

5.3.2 Fechamento de conexão com o leitor biométrico

Para fechar a conexão foi utilizada a função `CloseDevice`:

```
m_NBioAPI.CloseDevice(NBioAPI.Type.DEVICE_ID.AUTO);
```

Por meio do parâmetro `NBioAPI.Type.DEVICE_ID.AUTO`, a função buscará o último aparelho usado, se houver múltiplos aparelhos conectados ao PC, e fechará a conexão com ele.

5.3.3 Extração de informações biométricas

Utiliza-se essa função para obter as informações biométricas da impressão digital capturada (extração de minúcias). Essa função internamente chama uma responsável por acessar e capturar a impressão digital do usuário. As informações extraídas são encapsuladas numa estrutura de dados denominada FIR.

```
// Obtém FIR
m_NBioAPI.Enroll(out hNewFIR, null);
```

No parâmetro `hNewfFIR`, será salvo o FIR com informações da impressão digital capturada. O outro parâmetro possui valor *default null*.

5.3.4 Associação do FIR à instância da classe `NBioAPI.Nsearch` (persiste o FIR internamente, em memória)

Por meio da função `AddFIR`, o arquivo FIR será associado à instância da classe `NBioAPI.Nsearch`, ou seja, registra o FIR no sistema (ele é salvo em memória).

```
m_NSearch.AddFIR(hNewFIR, nUserID, out fpInfo);
```

O parâmetro `hNewfFIR` corresponde ao FIR a ser registrado (associado). O parâmetro `nUserID` corresponde ao ID do usuário, cujo FIR será salvo e o parâmetro `fpInfo` é

utilizado para extrair as informações do FIR para apresentá-las ao usuário (informação de qual usuário (id do usuário), qual dedo foi capturado e quantas amostras daquele dedo foram obtidas).

5.3.5 Remoção das informações biométricas referentes a um dedo cadastrado (do usuário) do sistema

Por meio da função `RemoveData`, as informações biométricas extraídas de uma amostra de um dedo do usuário são removidas internamente do sistema (são removidas do FIR salvado internamente no sistema).

```
m_NSearch.RemoveData(nUserID, nFingerID, nSampleNumber);
```

Os parâmetros da função são: `nUserID` (id do usuário), `nFingerID` (qual dedo, cuja amostra será removida) e `nSampleNumber` (id da amostra, cujas informações biométricas devem ser excluídas).

5.3.6 Remoção de informações biométricas referentes a todos os dedos cadastrados (do usuário) do sistema

Por meio da função `ClearDB`, as informações biométricas referentes a todos dedos cadastrados(do usuário) são removidas do sistema (todas as informações biométricas são removidas do FIR).

```
m_NSearch.ClearDB();
```

5.3.7 Salvar em arquivo de dados o FIR persistido em memória

Por meio da função `SaveDBToFile`, o FIR persistido em memória é salvo em um arquivo de dados (.fdb), cujo *path* é passado como parâmetro para a função.

```
// Carrega os FIRs (dos vários usuários) persistidos em
//memória em um arquivo de dados
m_NSearch.SaveDBToFile(szFileName);
```

O parâmetro `szFileName` é o *path* do arquivo.

5.3.8 Carregar FIR em memória a partir de um arquivo de dados

Por meio da função `LoadDBFromFile`, o FIR do usuário salvo em um arquivo de dados (.fdb), é carregado em memória.

```
m_NSearch.LoadDBFromFile(szFileName);
```

O parâmetro `szFileName` é o *path* do arquivo.

5.3.9 Captura de uma impressão digital

Por meio da função `Capture` uma impressão digital é capturada e as informações biométricas são extraídas. Essas informações são então encapsuladas na estrutura de dados FIR.

```
m_NBioAPI.Capture(out hCapturedFIR,20000,null);
```

O parâmetro `hCapturedFIR` é o FIR onde será postas as informações biométricas extraídas da impressão digital capturada. O parâmetro 20000 corresponde ao tempo de *timeout*, isto é, o tempo que o usuário tem após ser disparado o evento de capturar impressão digital, para pôr o dedo no leitor biométrico. O terceiro parâmetro tem valor *default null*. A diferença entre essa função e a `Enroll` é que na `Capture` apenas uma impressão digital é capturada por dedo, enquanto na `Enroll` são obtidas duas amostras por dedo. Em geral, essa função é executada anteriormente a uma identificação/verificação, para que se capture apenas uma impressão digital (que será identificada/verificada checando no banco de dados de impressões cadastradas).

Identificando se a impressão digital capturada está cadastrada no sistema:

Por meio da função `IdentifyData`, uma impressão digital capturada (capturada por meio da função `Capture`) é comparada com as impressões cadastradas de forma a identificar

o dedo do usuário que está desejando se autenticar no sistema (verifica se combina com alguma impressão digital cadastrada).

```
// Identifica FIR se pertence (está registrado) ao //sistema
m_NSearch.IdentifyData(hCapturedFIR,
NBioAPI.Type.FIR_SECURITY_LEVEL.NORMAL, out fpInfo);
```

O parâmetro `hCapturedFIR` é o FIR da impressão digital capturada (aquela que se quer verificar se combina com alguma cadastrada). O parâmetro `NBioAPI.Type.FIR_SECURITY_LEVEL.NORMAL` é o nível de segurança que se adota no controle de acesso (varia de 1 a 9, sendo o nível "normal" igual a 5). Quanto maior o nível de acesso, mais as informações extraídas da impressão capturada têm que combinar com as das verificadas para que a identificação seja positiva (aumento do nível de segurança). O parâmetro `fpinfo` contém informações trazidas da impressão digital que foi identificada pelo sistema (id do usuário, dedo identificado, por exemplo).

5.4 Módulo de Cartão

Todo desenvolvimento das aplicações responsáveis pelo acesso direto ao leitor de *smartcard* (MF-RD260) acabou sendo restringido pelo fato de que não havia tanto a documentação para as bibliotecas disponíveis para desenvolvimento de aplicações de 32 *bits* para este leitor, quanto o código disponível das mesmas. Assim, foi necessário desenvolver as aplicações de leitura e escrita no cartão em linguagem C para plataforma de 16 *bits* (para estas bibliotecas havia documentação).

Para o desenvolvimento foi utilizado o IDE (*Integrated Development Environment*) Borland C++ 4.5, já que consegue compilar para aplicações legadas de 16 *bits*.

Foram desenvolvidos separadamente dois módulos de acesso ao leitor de *smartcard*: um para leitura e o outro para escrita. A função tanto do programa que realiza a leitura, quanto do que realiza a escrita é ler/gravar uma chave randômica de 16 *bytes* que é gerada na aplicação de 32 *bits* no cartão. Para que a aplicação de 32 *bits* e tais módulos pudessem interoperar, utilizou-se o seguinte: a chave gerada na aplicação de 32 *bits* é passada para o programa de escrita no cartão como parâmetro da função `main` (`argc` e `argv`) do C, e a chave lida do

cartão pelo programa de leitura (16 *bits*) é passada para a aplicação de 32 *bits* por meio de um arquivo texto (`.txt`).

Para a geração da referida chave de 16 *bytes* utilizou-se o seguinte:

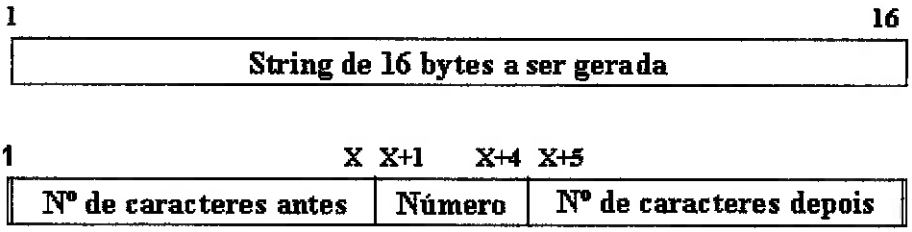

Figura 47 - Estrutura de caracteres na chave randômica do cartão

- **Nº de caracteres antes (X)**: é um número randômico que pode variar de 1 a 12;
- **Nº de caracteres depois**: é um número randômico que pode variar de 1 a (12 – Nº de caracteres antes);
- **Número**: é uma string numérica de 4 dígitos gerados randomicamente podendo variar de 1000 a 9999

Uma vez gerados randomicamente as quantidades de caracteres, foram geradas também, randomicamente, as *strings* propriamente ditas. Após isso, a *string* randômica final foi obtida dividindo-se a *string* randômica inicial em duas partes de 8 *bytes* cada e intercalando-se cada parte. As funções para gerar *strings* randômicas utilizam como semente o horário em que são geradas, tornando-se muito pequena a probabilidade de existirem duas chaves de cartão iguais. No entanto, para garantir que nunca existirão duas chaves iguais, sempre que uma chave é gerada é verificado no banco de dados se já existe uma chave igual, caso exista, uma outra chave é gerada utilizando-se o procedimento acima descrito. Esse processo se repete até o momento em que é gerada uma chave ainda não existente (chave do cartão deve ser primária no banco de dados).

Tanto o módulo de leitura, quanto o módulo de escrita possuem um fluxo básico de operações que devem ser feitas para que seja possível ler ou escrever no cartão. A diferença básica entre os dois módulos está somente no comando de leitura (*read*) ou comando de escrita (*write*). A figuras 48 apresenta um fluxograma de alto nível das operações que são realizadas pelos módulos de leitura/escrita.

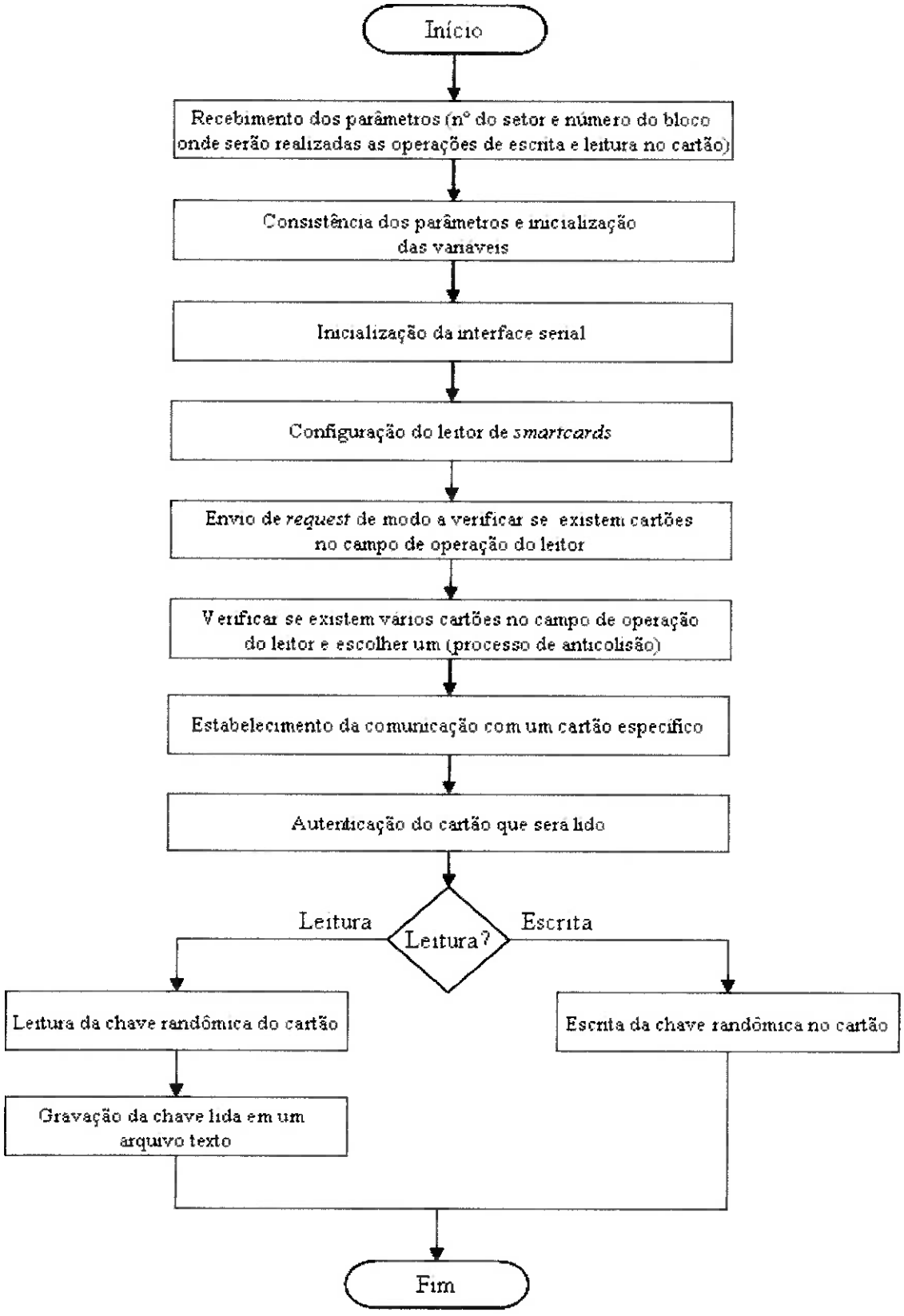

Figura 48 – Fluxograma dos módulos de escrita e leitura no cartão

5.4.1 Inialização da interface serial

A inialização da interface serial é realizada por meio da função `mifs_init (port,baud rate)`. Onde `port` é a porta serial do PC ("COM1", "COM2","COM3", "COM4") e `baud rate` é a taxa de transmissão (varia de 9600 bps a 115200 bps) que o leitor irá se comunicar com o PC pela interface serial. Para que o dispositivo funcione corretamente deve-se configurar a interface serial do PC com a mesma velocidade passada como parâmetro para a função `mifs_init`.

5.4.2 Configuração do leitor de smartcard

A configuração do leitor de smartcard é realizada por meio da função `mifs_config (mode, baud)`. Essa função é responsável por inicializar o sistema (o MF-RD260) com o seu modo de operação, e por configurar a sua taxa de comunicação com o *smartcard*. Na configuração não ocorre nenhuma comunicação com o cartão. O valor *default* do `mode` é 0xC6 e o valor *default* do `baud` é 0x0E.

5.4.3 Envio de *request*

O envio de *request* é realizado por meio da função `mifs_request (mode, tagtype)`. Essa função envia um *request* para o *smartcard* de modo a verificar se o mesmo está no campo de operação da antena do leitor. O `mode` foi configurado com o valor *default* 0 (modo `IDLE`) e o `tagtype` é um ponteiro que indica o *status* da operação informando ao PC se o pedido de *request* foi bem sucedido. A chamada da função `request` é necessária no início de uma seleção de um cartão (antes da chamada da função `select`).

5.4.4 Processo de anticolisão

Se houver um ou mais cartões no campo de operação do leitor, a função `anticollision (bcnt, snr)` irá escolher um dos cartões retornando o número de série do cartão escolhido (40 *bits*). Os parâmetros da função são `bcnt` cujo valor *default* é 0 e `snr` (número de série do cartão).

5.4.5 Estabelecimento da comunicação com um cartão específico

O estabelecimento da comunicação com um cartão específico é realizado pela função `select (snr, size)`, onde `snr` é o número de série retornado pela função `anticollision` e `size` é a resposta que é retornada para o PC (ATS – *answer to select*).

5.4.6 Autenticação

Para se ter acesso aos dados gravados na memória do cartão, o leitor deve conhecer uma das chaves (A ou B) gravadas no bloco de *trailer* do cartão. Somente o leitor e o cartão que está sendo lido em determinado momento tem conhecimento dessa chave, de modo que os dados trocados entre ambos são criptografados (no cartão há uma unidade de criptografia).

5.4.7 Leitura do cartão

A leitura do cartão é realizada pela função `read (address, data)`, onde `address` é o número do bloco (de 0 a 63) e `data` representa o bloco de 16 *bytes* que será lido.

5.4.8 Escrita no cartão

A escrita no cartão é realizada pela função `write(address, data)`, onde `address` é o número do bloco (de 0 a 63), onde os dados serão escritos e `data` representa o bloco de 16 *bytes* que será escrito.

5.4.9 Informações escritas no cartão

Como o cartão utilizado no projeto possui apenas 1K de memória e o algoritmo utilizado precisa de mais de 1K para guardar as informações necessárias ao reconhecimento biométrico (informações das minúcias das impressões digitais), não foi possível utilizar o cartão para guardar tais informações, sendo, portanto, armazenadas no banco de dados. Sendo assim, no

cartão é armazenada apenas uma chave de acesso cujo objetivo é fazer a identificação do usuário (a partir da chave obtém-se o id do usuário) e tornar o processo de identificação biométrica mais rápido, ao passo que não é necessário comparar a informação biométrica com todas as existentes no banco de dados.

5.5 Módulo de Gerência

Após a implementação do módulo de vídeo, do módulo de biometria, do módulo de cartão e da implementação do banco de dados, iniciou-se o desenvolvimento de uma aplicação de gerência com as funcionalidades administrativas do sistema como gerência de vídeos (consultar vídeos), gerência de usuários (incluir, alterar e excluir usuários), autenticação no sistema, cadastramento de cartões (parte da aplicação de gerenciamento que é responsável pela comunicação com a aplicação de escrita no cartão em 16 *bits*), cadastramento de impressões digitais (parte da aplicação de gerência que é responsável pelo interfaceamento com o módulo de biometria) e identificação (processo completo de identificação de um usuário que tenta acessar o local) e abertura da fechadura eletrônica (uso da interface paralela).

A aplicação de gerência foi implementada em C#, seguindo as descrições de casos de uso (vide anexo 1). Esta aplicação foi desenvolvida baseando-se em formulários *Windows* (*Windows Forms*). Essa abordagem foi escolhida por dois motivos: em primeiro lugar havia módulos para plataforma de 16 *bits*, os quais são difíceis de se integrar com aplicações de 32 *bits*, e em segundo lugar, no desenvolvimento do módulo de biometria, havia bibliotecas de desenvolvimento em C# *Windows Form*. Assim, não seria possível desenvolver a aplicação de gerência como uma aplicação *Web* (*Web Form*).

Para a implementação da arquitetura de três camadas do software foram utilizadas DLL's (*Dynamic Link Library* - Bibliotecas de ligação dinâmica). As classes existentes em cada camada foram encapsuladas em sua respectiva DLL. Assim sendo as classes responsáveis pelo acesso ao banco de dados foram implementadas na camada de acesso ao banco de dados (`entidade.dll`) e as classes responsáveis pela lógica de negócio foram implementadas na camada de controle (`controle.dll`). A interface homem-máquina, por se tratar do nível mais alto (nível mais próximo do usuário), foi implementada como uma aplicação *Windows Form*.

5.5.1 Como foi acessada a porta paralela usando a linguagem C#

Sistemas operacionais como Windows NT/200/XP não permitem acesso direto à porta paralela, necessitando um *driver* de sistema. Seguindo a recomendação [41], foi utilizado no projeto a biblioteca `Inpout32.dll` obtida em [42]. Esta biblioteca possui as funções `Out32` e `Inp32`. A função `Out32` é utilizada para escrever um valor em um endereço de entrada e saída e `Inp32` para ler um valor de um endereço de entrada e saída.

Esta biblioteca foi utilizada como código não-gerenciado por não se tratar de uma dll .Net, não sendo portando operado pelo CLR (*Common Language Runtime* - ambiente de execução das aplicações .NET). Foi então necessário o uso do *namespace* `System.Runtime.InteropServices` (*namespace* que permite o uso de bibliotecas não-gerenciadas) e o mapeamento dessas funções em funções correspondentes em C#, conforme o trecho de código abaixo (extraído de [41] e utilizado no projeto para acessar a interface paralela, configurá-la e realizar a conexão (envio de sinal) para a fechadura eletrônica):

```
// Escreve um byte no endereço
[DllImport("Inpout32.dll", EntryPoint="Out32")]
public static extern void Escrever(int endereco, byte valor);

// Lê um byte do endereço - não foi utilizada no projeto,
//pois não houve necessidade da leitura de um dado da
//paralela.
[DllImport("Inpout32.dll", EntryPoint="Inp32")]
public static extern byte Ler(int endereco);
```

5.5.2 Organização das classes no módulo de gerência

As classes utilizadas pelo sistema foram agrupadas em pacotes. Segue um diagrama dos pacotes do sistema.

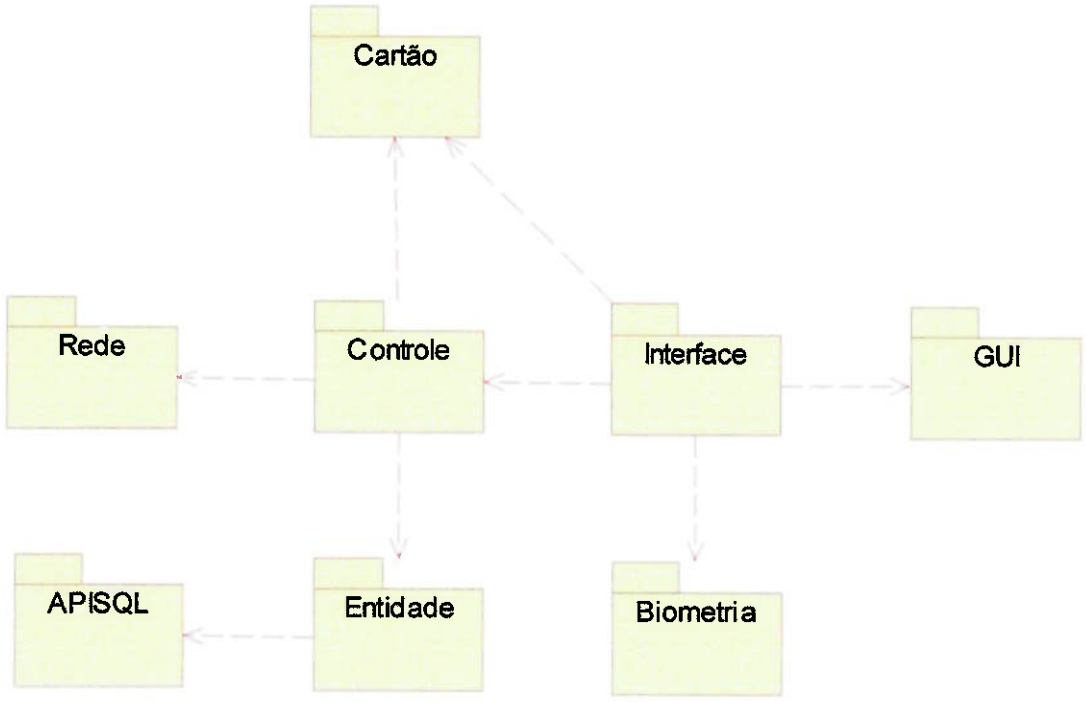

Figura 49 – Pacotes existentes no módulo de gerência

A aplicação de gerência possui oito pacotes: Cartão, Biometria, Rede, Controle, Entidade, API SQL, componentes de interface gráfica (GUI) e Interface (classes representativas da interface homem-máquina). No *package* GUI estão as classes de componentes gráficos (botões, formulários, listbox, grids, entre outros). No *package* API SQL estão as classes do sistema gerenciador de banco de dados. No *package* Interface estão todas as classes representativas dos formulários utilizados na aplicação gerenciadora. São elas: `frmCadastraCartao`, `frmConsultarUsuario`, `frmConsultarVideo`, `frmGerenciaAdmin` e `frmGerUsuarios` e `frmLogin`. No *package* Controle estão as classes que implementam a lógica de negócio. São elas: `Autenticacao`, `Biometria`, `Cartao`, `Client`, `GerUsuarios`, `GerVideos`, `Paralela`. No *package* Entidade estão todas as classes que acessam o banco de dados. São elas: `Administrador`, `Conexão`, `Situação`, `Usuário` e `Vídeos`. No *package* Rede estão todas as classes de acesso a rede (`System.Net.Sockets`). O *package* Cartão representa as aplicações de acesso ao cartão. No *package* biometria estão as classes de acesso ao leitor biométrico, bem como as classes responsáveis pelo cadastramento/validação de impressões digitais.

5.6 Integração e Testes

O sistema foi testado seguindo a seguinte estratégia de testes: primeiro foram realizados testes unitários nas partes dos módulos, em seguida foram realizados testes em cada módulo e por fim foi realizado um teste de integração.

5.6.1 Testes Unitários

Conforme foram sendo desenvolvidas as partes constituintes dos módulos (como métodos de classes e funções), testes unitários foram realizados tendo como referência o funcionamento esperado de cada parte constituinte. Para tanto foram utilizadas as seguintes abordagens:

- Impressão de mensagens de erro ao longo do código;
- Tratamento de exceções através do uso de *try/catch* com impressão de mensagens no *catch*;
- Verificação da ordem correta dos parâmetros nos métodos;
- O banco de dados foi populado manualmente para verificar funcionalidades de consulta;
- Desenvolvimento de protótipos para avaliar a correta execução de uma funcionalidade.

5.6.2 Testes de módulo

Foram feitos tendo por base a descrição de casos de uso referentes ao módulo implementado ou com base na especificação de programa (para o caso das aplicações de 16 *bits* que acessam o cartão). Os erros encontrados eram anotados e ao final do teste eram corrigidos. Após as correções, repetiam-se os testes de modo a verificar se as correções efetivamente solucionaram os problemas encontrados. Testes baseados na abordagem "caixa preta".

5.6.3 Testes de integração

Uma vez que todos os módulos foram testados, procederam-se, para o caso da integração, testes do sistema como um todo. Os testes de integração foram feitos com base nas especificações do projeto, ou seja, com base na visão de negócio (IDEF0 e sua descrição) e com base nos casos de uso descritos (visão de casos de uso).

Os testes de integração apontaram os seguintes problemas:

- Problemas na inicialização/finalização da *thread* responsável pelo processo de identificação de usuários no sistema;
- Problemas com referências da aplicação principal às bibliotecas biométricas (SDK biométrico);
- Problemas na interoperação entre a aplicação principal em 32 bits e as aplicações de leitura/escrita nos cartões (em 16 bits);
    - Arquivo texto com a chave randômica de 16 bytes não estava sendo fechado, comprometendo a correta execução da *thread* de identificação;
    - Os parâmetros (setor/bloco/chave randômica) passados para a aplicação de leitura/escrita nos cartões possuíam espaços em branco adicionais.
- Problemas na interoperação entre o servidor de vídeo e a aplicação principal;
    - Problemas na identificação do *host* (endereço IP) que requisita a gravação do vídeo pelo servidor;
    - Problemas no local (caminho no sistema de arquivos) em que eram armazenados os vídeos.
- Problemas no acionamento da fechadura pela interface paralela. Inicialmente a fechadura eletrônica não estava sendo acionada corretamente pelo correspondente sinal da interface paralela pois:
    - As referências de tensão (terras) entre computador (terminal) e fonte de alimentação da fechadura eletrônica não eram comuns;
    - Havia necessidade da espera de um pequeno intervalo de tempo na aplicação principal após o acionamento da interface paralela.

Todos os casos de uso foram implementados de acordo com a especificação, conseqüentemente, quase todos os testes realizados para os mesmos obtiveram sucesso. Uma

questão a ser enfatizada no plano de testes é o fato de não terem sido feitos testes de usabilidade no sistema. Segue uma tabela com alguns problemas encontrados durante os testes dos casos de uso:

Casos de Uso	Problemas
Caso de uso 1.1: Incluir usuário no sistema	▪ Ao selecionar a opção Novo Administrador não aparecia os campos login/senha/confirmação de senha; ▪ O sistema não exibia os dados que acabaram de ser incluídos após a inclusão.
Caso de uso 1.2: Consultar dados cadastrais de um usuário incluído no sistema	▪ A data de expiração não era exibida; ▪ Quando o administrador digitava parte do nome de um usuário, não apareciam os dados do primeiro usuário que começasse com a parte do nome digitado.
Caso de uso 1.3: Alterar dados cadastrais de um usuário incluído no sistema	▪ O sistema não exibia os dados que acabaram de ser alterados (após a alteração).
Caso de uso 1.4: Excluir um usuário do sistema	▪ O sistema não excluía os arquivos de vídeo de acesso.
Caso de uso 2: Cadastrar cartão de acesso	▪ Após o cadastro do cartão, o sistema não atualizava o flagCartao na interface de gerenciamento de usuários.
Caso de uso 3.1: Consultar impressões digitais	Não foram encontrados problemas.
Caso de uso 3.2: Cadastrar impressão digital	Não foram encontrados problemas.
Caso de uso 3.3: Excluir uma impressão digital já cadastrada no sistema	Não foram encontrados problemas.
Caso de uso 3.4: Excluir todas as impressões digitais cadastradas	Não foram encontrados problemas.
Caso de uso 4: Consultar usuários	▪ Não era exibida uma mensagem, caso nenhum usuário fosse encontrado.
Caso de uso 5: Consultar vídeos	▪ Não era exibida uma mensagem, caso

	nenhum vídeo fosse encontrado.
Caso de uso 6: Executar vídeo consultado	▪ O Player de vídeo não abria, pois o diretório indicado no código era inválido.
Caso de uso 7: Autenticação do administrador no sistema	▪ A opção cancelar não funcionava.
Caso de uso 8: Ativar identificação	▪ O sistema não desbloqueava a opção Parar Identificação.
Caso de uso 9: Desativar identificação	Não foram encontrados problemas.
Caso de uso 10: Ativar servidor de vídeo	Não foram encontrados problemas.
Caso de uso 11: Desativar servidor de vídeo	Não foram encontrados problemas.
Caso de uso 12: identificação de usuário	▪ O sistema validava incorretamente as datas de início e expiração da autorização.

Figura 50 – Tabela com os resultados dos testes dos casos de uso

Os problemas encontrados ao serem realizados os testes foram corrigidos, o que foi constatado com uma nova série de testes.

6 Considerações finais

6.1 Problemas encontrados

O maior problema encontrado durante o desenvolvimento do projeto foi a dificuldade de se encontrar recursos para a realização do mesmo.

Inicialmente, a proposta de projeto era o desenvolvimento de um protótipo de robô para navegação autônoma baseando-se em autômatos adaptativos. Essa primeira proposta foi submetida para a Fapesp com o intuito de se obter verba para sua realização, no entanto, o projeto foi indeferido. Sendo assim, partiu-se para a atual proposta de projeto. Não obstante, surgiram mais problemas com relação a aquisição dos recursos necessários para seu desenvolvimento. Não foi possível desenvolver a aplicação para um terminal de acesso o que levou o grupo a ter que emular o terminal em um PC, fazendo com que o sistema perdesse muito da veracidade.

6.2 Resultados obtidos

O produto final obtido foi um sistema integrado de controle de acesso que envolve *smartcard*, biometria por meio de impressão digital e gravação/gerenciamento de vídeo de *log*. Também foi realizado um sistema de gerência que fornece a infraestrutura necessária a esse sistema de identificação, possibilitando o cadastramento dos usuários no sistema (dados cadastrais, cartão de acesso e impressões digitais) e gerenciamento/consulta de usuários e vídeos de *log*.

6.3 Considerações sobre o aprendizado com o projeto

Foi possível adquirir uma sólida base de conhecimento sobre *smartcards* (aplicações e arquitetura interna), biometria por meio de impressão digital (teoria para reconhecimento biométrico e etapas de processamento de imagens digitais utilizadas para a extração de informações biométricas) e manipulação de vídeos utilizando-se a tecnologia JMF (Java *Media Framework*). Obteve-se maior conhecimento sobre *threads*, arquitetura cliente-servidor (*sockets* TCP), arquitetura de três camadas para desenvolvimento de software, acesso a paralela a partir de C# e integração de aplicações desenvolvidas com diferentes tecnologias.

Obteve-se também maior familiaridade com técnicas de modelagem de processo como IDEF0.

Do ponto de vista de gestão de projetos, pôde-se aplicar a metodologia de gerência disponível no PMBOK (WBS, rede de precedência de eventos) de forma a gerenciar o tempo disponível.

6.4 Viabilidade técnica da solução proposta

Apesar de se ter utilizado diferentes tecnologias e sistemas legados (o leitor de *smartcard* utiliza bibliotecas para plataforma de 16 *bits*), a plataforma C#.NET proporcionou ao projeto uma forma fácil e viável de integração.

Apesar da gravação de vídeo de *log* de dez segundos ocupar cerca de 87 MB, o uso de um compressor permitiu a redução para algumas centenas de KB, permitindo a viabilidade do armazenamento do vídeo, não sendo necessário o uso de imagens estáticas (que poderiam ser tiradas em momentos que não identifiquem o usuário do sistema).

Um empecilho à solução proposta é a não existência de dois leitores de biometria digital, um para cadastro (lado servidor) e o outro para identificação (lado cliente).

A viabilidade técnica da solução também depende de alguns cuidados a serem tomados com relação à infraestrutura do local a ser controlado. Assim, a câmera deve permanecer em um local estratégico para gravação e de difícil acesso aos usuários. Para o caso do leitor de *smartcards* e do leitor biométrico, ambos deveriam estar encapsulados em um mesmo terminal de forma a evitar qualquer tipo de violação física.

6.5 Possíveis aperfeiçoamentos do projeto

Como aperfeiçoamento do projeto, poderia ser implementado um módulo de reconhecimento de face que consiga procurar um determinado usuário no conjunto de arquivos de vídeo de *log*. Além disso, o sistema deveria somente permitir a entrada do usuário, se analisasse o vídeo de *log* e detectasse uma face (suposta do usuário), evitando que usuários mal-intencionados burlem o sistema de gravação.

Para o módulo de identificação por biometria digital, seria interessante que fosse colocado algum dispositivo que detectasse a pressão arterial dos dedos dos usuários, de modo

a evitar que infratores tentem acessar o sistema a partir medidas extremas como dedos amputados e modelos sintéticos de impressão digital.

A usabilidade do sistema poderá ser melhorada de modo a facilitar o aprendizado de seu funcionamento. Para o desenvolvimento de interfaces melhores poderiam ser aplicadas técnicas de *design patterns*.

Um problema que dificilmente seria evitado com a solução proposta seria o caso de vários usuários entrarem no local que está sendo protegido por meio da identificação de apenas um deles. Isso poderia ser resolvido com a existência de duas portas de acesso, sendo que a segunda porta seria aberta apenas após um usuário fechar a primeira. Nessa solução deveria existir algo que identificasse a existência de um único usuário entre as portas.

Os cartões de acesso também poderiam ser substituídos por outros com maior quantidade de memória, de modo a possibilitar armazenar as impressões digitais de cada usuário no próprio cartão de acesso (evitando atraso de tempo pela busca dos dados biométricos no banco de dados). Além disso, a plataforma de desenvolvimento das aplicações de acesso ao cartão (em 16 *bits*) poderia ser migrada para 32 *bits*. Para isso seria necessário adquirir a documentação das bibliotecas para 32 *bits* junto à empresa Philips.

Bibliografia e Referências

Referências bibliográficas

[1] NETMARKT. As tecnologias mais promissoras para 2004. Disponível em: <www.netmarkt.com.br/noticia2004/1324.html>. Acesso em: 15 set. 2005.

[2] ESTADAO-Notícias de Biotecnologia. Biometria deverá crescer até 60% no ano que vem. Disponível em: <http://www.link.estadao.com.br/index.cfm?id_conteudo=1748>. Acesso em: 15 set. 2005.

[3] CSO Brasil. Mercado Bilhométrico. Disponível em: <http://www.isl.com.br/noticias/12052003.asp>. Acesso em: 15 set. 2005.

[4] REGINA,C.; DANIEL, L.; LUIS, R.; ALMEIDA, F.; ALEXANDRE, P. **Estudo de áreas de mercado biometria**. Departamento de Computação da Universidade Federal de Minas Gerais. Disponível em: <http://www2.dcc.ufmg.br/~becker/empreendimentos-2005-2/Aula%204%20-%206%20-%20Ideias%20e%20oportunidades/Biometria.ppt#259,4,Produtos e Serviços >. Acesso em: 15 set. 2005.

[5] PINHEIRO, J. M. S. **Biometria na Segurança de Redes de Computadores**. 11 nov. 2004. Disponível em: <http://www.projetoderedes.com.br/artigos/artigo_biometria_na_seguranca_das_redes.php>. Acesso em: 15 set. 2005.

[6] CANEDO. J.A.F. **Terminal de controle de ponto e acesso usando biometria e integrado a Web**. Projeto final apresentado ao curso de Engenharia de Computação da Escola de Engenharia Elétrica e Computação da Universidade Federal de Goiás. Goiânia, 2003.

[7] MICHAELIS. Dicionário *Online*. Disponível em: <http://www.uol.com.br/michaelis/>. Acesso em: 15 set. 2005.

[8] HONG, L; JAIN, A. K. Integrating Faces and Fingerprints for Personal Identification. IEEE. **Transactions on Pattern Analysis and Machine Intelligence**, vol. 20, n° 12, pp. 1295-1307, 1998.

[9] COSTA, S. M. F; JAVIER, F.; FERNANDEZ, R.. **Classificação e verificação de impressões digitais**. Dissertação (Mestrado). São Paulo, 2001. 123 páginas.

[10] BOMBONATTI, J. **História da Dactiloscopia**. Texto elaborado pelo professor. de papiloscopia da Academia de Polícia Civil do Estado de São Paulo. Disponível em: < http://216.239.51.104/search?q=cache:c4JfeLZXb8YJ:www.aguiarsoftware.com.br/aguiar211.htm+%22hist%C3%B3ria+da+biometria%22&hl=pt-BR>. Acesso em: 15 set. 2005.

[11] SECUGEN. Top 5 Reasons for Choosing SecuGen. SecuGen Biometric Solutions. Disponível em: <http://www.secugen.com/advantage/index.htm>. Acesso em: 15 set. 2005.

[12] POYNTON, C. **Digital Video and HDTV Algorithms and Interfaces**. 1a Edição. Morgan Kaufmann Publishers, 2003. 736p.

[13] FALCÃO, A. X.; LEITE, N.J.. **Curso de tópicos em processamento de imagens.** Instituto de Computação da Universidade Estadual de Campinas. Campinas, 2000. Disponível em: <www.ic.unicamp.br/~siome/teaching/ 2005/mo815-0105/material/lista0.pdf >. Acesso em: 08 ago. 2005.

[14] CASTRO, F. C. C. de; CASTRO, M.C.F. de; **Capítulo V – Introdução ao Sistema MPEG de Codificação de Vídeo**. Departamento de Engenharia Elétrica, Faculdade de Engenharia, PUCRS. 19p.

[15] SITE OFICIAL XVID. Disponível em: <www.xvid.org>. Acesso em: 05 ago. 2005.

[16] *Koepi's Media Development Homepage*. Disponível em: <www.xvid.org>. Acesso em: 05 ago. 2005.

[17] WIKIPEDIA. MPEG. Disponível em: <http://pt.wikipedia.org/wiki/MPEG>. Acesso em: 10 nov. 2005.

[18] BIROCCHI Jr., M. Antônio; CARRARI, M. Pereira. **Integração dos documentos em um cartão a microprocessador utilizando a tecnologia JavaCard.** Projeto de Formatura apresentado à Escola Politécnica da Universidade São Paulo; São Paulo, 2004.

[19] VR ALMENTACAO E SMART. Disponível em: <http://www.vr.com.br>. Acesso em: 09 nov. 2005.

[20] SPTRANS. Bilhete Único. Disponível em: <http://www.sptrans.com.br>. Acesso em: 09 nov. 2005.

[21] POLIPARATODOS. Apostila cliente servidor. Disponível em: <http://poliparatodos .larc.usp.br/modulos/20/index.htm>. Acesso em: 02 nov. 2005.

[22] WIKIPEDIA. Cliente-Servidor. Disponível em: <http://pt.wikipedia.org/wiki/Cliente-servidor>. Acesso em: 30 set. 2005.

[23] PRESSMAN, Roger S. **Engenharia de Software**. 5ª Edição. Mcgraw-Hill Interame, 2002, 872 p.

[24]TRANSMETH.Mapeamento do Processo. Disponível em: <http://www.numa.org.br/ transmeth/ferramentas/ffmapeam.htm>. Acesso em: 17 ago. 2005.

[25] CUGNASCA, C. E.; HIRAKAWA, A. R. **Estudo da Interface Paralela do PC e sua Aplicação no Controle de Display Alfanumérico**. Laboratório de microprocessadores da Escola Politécnica da Universidade de São Paulo. São Paulo, 1998.

[26] SUN. Sun Development Network. About the Java Technology. Disponível em: <http://java.sun.com>. Acesso em: 11 ago. 2005.

[27] SUN. Sun. Java Technology. Disponível em: <http://java.sun.com>. Acesso em: 19 ago. 2005.

[28] SUN. JMF FAQS. Disponível em: <http://java.sun.com/products/java-media/jmf/reference/faqs/index.html#what>. Acesso em: 05 jul. 2005.

[29] SUN. JavaTM Media Framework API Guide. Disponível em: <http://java.sun.com/products/java-media/jmf/2.1.1/specdownload.html>. Acesso em: 22 ago. 2005.

[30] MICROSOFT. Disponível em: <http://www.microsoft.com>. Acesso em: 19 ago. 2005.

[31] MICROSOFT. What is the .NET Framework?. Disponível em: <http://msdn.microsoft. com/netframework/gettingstarted/default.aspx >. Acesso em: 19 ago. 2005.

[32] PHILIPS; **Mifare Standard Card IC MF1ICS50 – Functional Specification**, Julho 1998.

[33] PHILIPS; **Mifare Serial Reader MFRD260 Short Range**, Junho 1997.

[34] PHILIPS; **Mifare Hardware Independent – Low Level Functions**, Maio 1997.

[35] CREATIVE. Creative NP PRO Product Info - Specifications. Disponível em: < http://www.creative.com/products/product.asp?category=218&subcategory=219&product=628&nav=2>. Acesso em: 30 ago. 2005.

[36] SUN. JMF 2.1.1 - Supported Formats. Disponível em: < http://java.sun.com/products/java-media/jmf/2.1.1/formats.html>. Acesso em: 06 set. 2005.

[37] PMI. Project Manager Institute. Disponível em: <http://www.pmi.org>. Acesso em: 10 ago. 2005.

[38] ISO. *International Organization for Standardization.* Disponível em: <http://www.iso.org>, Acesso em: 10 ago. 2005.

[39] DEITEL, H.M., DEITEL, P.J.. **Java Como Programar**. 4ª Edição. Bookman, 2003. 1386p.

[40] NITGEN. Biometric Service Provider SDK - Programmer's Manual SDK version 4.01

[41] CAVALCANTI, E. Controle de dispositivos externos através da porta paralela utilizando C#. Disponível em: <http://www.linhadecodigo.com.br/artigos.asp?id_ac=254>, Acesso em: 11 out. 2005.

[42] LOGIX4U. Inpout32.dll for WIN NT/2000/XP. Disponível em: <http://www.logix4u.net/ inpout32.htm>. Acesso em: 01 nov. 2005.

[43] GEKKO SOFTWARE. Threading in a windows form. Disponível em: <http://www.gekko-software.nl/DotNet/Art05.htm>. Acesso em: 13 out. 2005.

[44] MIZRAHI, V. V.. **Treinamento em linguagem C**. McGraw-Hill, 1990. 241p.

[45] UFMG. Curso de linguagem C UFMG. Disponível em: <http://mico.ead.eee.ufmg.br/~cursoc/c.html>. Acesso em: 20 nov. 2005.

[46] DEITEL, H.M., DEITEL, P.J., CHOFFNES. **Sistemas Operacionais**. 3ª Edição. Person Education, 2005. 784p.

[47] TANENBAUM, A. . **Sistemas Operacionais Modernos**. 2ª Edição. Person Brazil, 2003.

[48] FERREIRA, M. A. G. V. . **Modelo das três camadas**. Apostila da disciplina de Engenharia de Software I do departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo. São Paulo. 2003.

[49] LACCHINI, C. Smart Cards - Uma tecnologia abrindo o seu caminho. Disponível em: < http://www.abinee.org.br/informac/arquivos/185.pdf>. Acesso em: 31 ago. 2005.

[50] SURENDRAW, D. *Smart Card Technology and Security. Department of Computer Science. University of Chicago*. Disponível em: <http://people.cs.uchicago.edu/~dinoj/smartcard/security.html>. Acesso em: 27 ago. 2005.

[51] AME ELETRÔNICA. O que é um smartcard?. Disponível em: <http://www.ame.com.br/SaibaMais.htm>. Acesso em: 15 ago. 2005.

[52] SURENDRAW, D. *Applications of Smart Cards. Department of Computer Science. University of Chicago*. Disponível em: <http://people.cs.uchicago.edu/~dinoj/smartcard/applications.html>. Acesso em: 27 ago. 2005.

Bibliografia

BONATO, VANDERLEI; MOLZ, R. F.; FERRÃO, M. F.; FURTADO, J. C.; **Proposta de um Sistema para Processamento de Impressões Digitais Implementado em Hardware**; Departamento de Informática e Departamento de Física e Química da Universidade Santa Cruz do Sul; Rio Grande do Sul, 2000.

Anexo 1 - Especificação de Casos de Uso

Índice de Casos de Uso

Caso de uso 1: Manipulação dos dados cadastrais do usuário

Caso de uso 1.1: Incluir usuário no sistema

Descrição: Este caso de uso descreve o cadastro de um usuário no sistema SysSec.

Evento iniciador: Seleção da opção de incluir novo usuário (Novo Usuário) na transação de gerenciamento de usuários.

Atores: Administrador.

Pré-condição: Administrador autenticado. O sistema se encontra no estado de gerenciamento de usuários.

Seqüência de eventos:
1. O administrador seleciona a opção de incluir Novo Usuário.
2. O sistema apaga todos os campos permitindo que novos dados cadastrais do usuário sejam digitados.
3. O administrador preenche os dados do novo usuário a ser inserido, completando todos os dados obrigatórios: digita nome, telefone e ddd residencial, telefone e ddd comercial, telefone e ddd celular, endereço de contato, seleciona situação (selecionando entre opções de uma lista: aluno de graduação, aluno de pós-graduação, funcionário, professor, visitante e outros), seleciona data de início da vigência de autorização e data de expiração em um calendário.
4. O administrador não seleciona a opção de Novo Administrador (usuário não constará como administrador).
5. O administrador confirma a inclusão.
6. O sistema exibe uma mensagem pedindo a confirmação da inclusão do novo usuário.
7. O administrador confirma a inclusão.
8. O sistema salva os dados no banco de dados e exibe uma mensagem indicando que a operação foi bem sucedida.

Pós-condição: Um novo usuário foi incluído no sistema. O sistema continua na transação de gerenciamento de usuários e exibe os dados do usuário que acabou de ser incluído.

Extensões:
1. No passo 4, se o administrador seleciona a opção novo administrador, o sistema irá permitir que se incluam dados como login, senha e confirmação de senha para o futuro administrador.
2. No passo 5 e 7, o administrador poderá cancelar a inclusão. Nesse caso o sistema retorna para a situação de pré-condição.
3. Em caso de falta de campos obrigatórios a serem preenchidos o sistema retorna uma mensagem de falta de parâmetros e retorna para o passo 3.
4. No passo 5, se a data de início de vigência de autorização for menor que a data atual ou se a data de expiração for menor que a data de início o sistema exibirá uma mensagem indicando o respectivo erro.

Inclusões: Não há.

Caso de uso 1.2: Consultar dados cadastrais de um usuário incluído no sistema.

Descrição: Consultar os dados de um usuário comum/administrador cadastrado no sistema.

Evento iniciador: Seleção de um nome de um usuário na transação de gerenciamento de usuários.

Atores: administrador.

Pré-condição: O sistema se encontra na transação de gerenciamento de usuários e já houve o cadastramento de um primeiro usuário(administrador mestre).

Seqüência de eventos:
1. O administrador seleciona um nome de um usuário em uma lista que contém o nome de todos os usuários cadastrados no sistema.
2. O sistema pesquisa e exibe os campos referentes ao usuário selecionado que são: nome, telefone e ddd residencial, telefone e ddd comercial, telefone e ddd celular, endereço de contato, situação, data de início da vigência de autorização e data de expiração e esses campos serão exibidos de forma protegida. Além disso será exibida a informação indicando o *status* de cadastramento do cartão de acesso e das impressões digitais(se foi ou não cadastrado pelo menos um dedo).
3. O administrador visualiza os dados da atividade selecionada.

Pós-condição: O sistema apresenta as informações cadastrais do usuário.

Extensões:
1. No passo 2, caso o usuário seja administrador serão exibidos também em forma protegida os campos: login, senha e confirmação de senha.
2. No passo 2, caso seja detectado que o usuário possui cartão cadastrado, a opção de gerenciamento de impressão tornar-se-á disponível.

Inclusões: Não há.

Caso de uso 1.3: Alterar dados cadastrais de um usuário incluído no sistema.

Descrição: Alterar os dados cadastrais de um usuário incluído no sistema (previamente selecionado pelo administrador).

Evento iniciador: O administrador seleciona a opção alterar usuário.

Atores: administrador.

Pré-condição: O sistema se encontra na transação de gerenciamento de usuários e os dados do usuário que se deseja alterar (que foi previamente selecionado – caso de uso consultar dados cadastrais de um usuário incluído no sistema) estão sendo exibidos.

Seqüência de eventos:
1. O administrador seleciona a opção alterar usuário.
2. O sistema desprotege os campos exibidos, permitindo que os dados sejam alterados.
3. O administrador altera os dados conforme desejado.
4. O administrador seleciona a opção de confirmar alteração.
5. O sistema exibe uma mensagem pedindo a confirmação da alteração do usuário.
6. O administrador confirma a alteração.
7. O sistema salva os dados no banco de dados e exibe uma mensagem indicando que a operação foi bem sucedida.

Pós-condição: Os dados do usuário foram alterados, o sistema continua na transação de gerenciamento de usuários, porém exibindo os dados alterados.

Extensões:
1. No passo 2, caso o administrador que esteja logado não seja o administrador mestre, o sistema não permite que os campos de login e senha do administrador selecionado sejam alterados.
2. No passo 2, caso o usuário selecionado seja o administrador mestre, só será permitida qualquer alteração nos dados cadastrais, se o administrador mestre for quem esteja logado.
3. No passo 3, se o administrador seleciona a opção novo administrador, o sistema irá permitir que se incluam dados como login, senha e confirmação de senha para o futuro administrador.
4. No passo 4, se a data de início de vigência de autorização for menor que a data atual ou se a data de expiração for menor que a data de início o sistema exibirá uma mensagem indicando o respectivo erro.
5. No passo 4 e 6, o administrador pode não confirmar a alteração, neste caso o sistema apenas não salva as alterações e retorna para a situação de pré-condição.
6. No passo 4, em caso de campos não preenchidos, o sistema exibe uma mensagem indicando a existência de um campo não preenchido e volta ao passo 3.

Inclusões: Não há.

Caso de uso 1.4: Excluir um usuário do sistema.

Descrição: Excluir os dados do usuário que está sendo visualizado.

Evento iniciador: O administrador seleciona a opção de excluir usuário.

Atores: administrador.

Pré-condição: O sistema se encontra na transação de gerenciamento de usuários e os dados do usuário que se deseja excluir (que foi previamente selecionado – caso de uso consultar dados cadastrais de um usuário incluído no sistema) estão sendo exibidos.

Seqüência de eventos:
1. O administrador seleciona a opção de excluir usuário.
2. O sistema pede a confirmação da exclusão.
3. O administrador confirma a exclusão do usuário.

4. O sistema exclui o usuário do banco de dados juntamente com todos os seus vídeos de acesso.
5. O sistema exibe uma mensagem indicando o sucesso da operação.
6. O sistema exibe dados de outro usuário, o primeiro da tabela em ordem alfabética.

Pós-condição: O usuário selecionado foi excluído, o sistema continua na transação de gerenciamento de usuários exibindo os dados cadastrais do primeiro usuário da tabela em ordem alfabética.

Extensões:

1. No passo 1, se o usuário selecionado for o administrador mestre, o sistema exibe uma mensagem de impossibilidade de exclusão do usuário.
2. No passo 1, se o usuário selecionado for o mesmo que estiver logado será exibida uma mensagem de impossibilidade de exclusão, voltando a situação de pré-condição.
3. No passo 1 , caso o usuário selecionado seja o administrador e o administrador correntemente logado não seja o administrador mestre, o sistema exibe uma mensagem indicativa de impossibilidade de exclusão, voltando a situação de pré-condição.
4. No passo 3, se o administrador não confirmar a exclusão, volta para a situação de pré-condição, voltando a situação de pré-condição.

Inclusões: não há.

Caso de uso 2: Cadastrar cartão de acesso

Descrição: Cadastrar o cartão de acesso para um determinado usuário selecionado pelo administrador.

Evento iniciador: O administrador seleciona a opção cadastramento de cartão na transação de gerenciamento de usuários.

Atores: administrador.

Pré-condição: O sistema se encontra na transação de gerenciamento de usuários e os dados do usuário que se deseja cadastrar o cartão de acesso(que foi previamente selecionado – caso de uso consultar dados cadastrais de um usuário incluído no sistema) estão sendo exibidos.

Seqüência de eventos:

1. O administrador seleciona a opção cadastramento de cartão na transação de gerenciamento de usuários.
2. O sistema exibe uma interface para cadastramento de cartão.
3. O administrador seleciona a opção cadastrar cartão na interface.
4. O sistema exibe uma mensagem pedindo que o administrador insira o cartão no leitor de smartcard.
5. O administrador insere o cartão no leitor de smartcard e seleciona a opção confirmar cadastro.
6. O sistema cadastra o cartão e exibe uma mensagem de sucesso.

Pós-condição: Cartão cadastrado e o sistema se encontra na situação de pré-condição com o *status* de cadastramento do cartão atualizado.

Extensões:
1. O pesquisador poderá cancelar o cadastramento do cartão, nesse caso o sistema retorna para a situação de pré-condição.

Inclusões:
Não há.

Caso de uso 3: Gerenciamento de impressões digitais

Caso de uso 3.1: Consultar impressões digitais

Descrição: Consultar as impressões digitais cadastradas de um determinado usuário selecionado pelo administrador.

Evento iniciador: O administrador seleciona a opção Gerenciar Impressão Digital na transação de gerenciamento de usuários.

Atores: administrador.

Pré-condição: O sistema se encontra na transação de gerenciamento de usuários.

Seqüência de eventos:
1. O administrador seleciona a opção Gerenciar Impressão Digital na transação de gerenciamento de usuários.
2. O sistema exibe a interface de transação de gerenciamento de impressões digitais com a informação de quais são as impressões digitais já cadastradas do usuário.

Pós-condição: O sistema se encontra na transação de gerenciamento de impressões digitais com a informação de quais são as impressões digitais já cadastradas do usuário.

Extensões:
1. No passo 2, se não houver impressões digitais cadastradas, o sistema exibirá uma lista em branco.

Inclusões: Não há

Caso de uso 3.2: Cadastrar impressão digital

Descrição: Este caso de uso descreve o processo de cadastrar impressão(ões) digital(is) de um usuário.

Evento iniciador: O administrador seleciona a opção registrar digital no BD (banco de dados).

Atores: administrador.

Pré-condição: Sistema na transação de gerenciamento de impressão digital.

Seqüência de eventos:
1. O administrador seleciona a opção registrar digital no BD (banco de dados).
2. O sistema exibe uma interface para cadastramento da impressão digital.
3. O administrador solicita que o usuário coloque o dedo no leitor biométrico.
4. O sistema captura, valida a impressão digital e exibe a confirmação do cadastramento para o administrador.
5. O administrador finaliza a operação de cadastramento de impressão digital.
6. O sistema exibe uma lista atualizada com as impressões digitais cadastradas e atualiza o banco de dados (com a nova impressão digital cadastrada e seta o indicador de que houve um cadastramento de impressão digital).

Pós-condição: Impressão digital cadastrada e o sistema se encontra na situação de pré-condição exibindo a lista atualizada com a(s) impressão(ões) digital(is) cadastrada(s).

Extensões:
1. No passo 3, o administrador pode cancelar a operação de cadastramento retornando para a situação de pré-condição.
2. No passo 4, o sistema pode não validar a impressão digital cadastrada e exibe uma mensagem da necessidade de um novo cadastramento, retornando ao passo 3.
3. No passo 5, o administrador poderá cadastrar as impressões digitais de outros dedos do usuário conforme desejado.
4. No passo 6, se o dedo do usuário a ser cadastrado já constar no sistema (incluído no banco de dados), este exibe uma mensagem indicativa de erro.

Inclusões: Não há.

Caso de uso 3.3: Excluir uma impressão digital já cadastrada no sistema

Descrição: Este caso de uso descreve o processo de excluir uma impressão digital já cadastrada no sistema.

Evento iniciador: O administrador seleciona a opção remover digital.

Atores: administrador.

Pré-condição: Sistema na transação de gerenciamento de impressão digital com a impressão digital a ser excluída selecionada.

Seqüência de eventos:
1- O administrador seleciona a opção remover digital.
2- O sistema exclui do banco de dados a impressão digital selecionada, atualiza a lista de impressão(ões) digital(is) cadastrada(s) e exibe uma mensagem indicando o sucesso da exclusão.

Pós-condição: sistema se encontra na situação de pré-condição exibindo a lista atualizada com a(s) impressão(ões) digital(is) cadastrada(s).

Extensões:
1- No passo 1, caso a lista de impressões digitais esteja vazia, o sistema exibirá uma mensagem de erro.

Inclusão:
Não há.

Caso de uso 3.4: Excluir todas as impressões digitais cadastradas

Descrição: Este caso de uso descreve o processo de excluir todas as impressões digitais cadastradas para uma determinado usuário no sistema.

Evento iniciador: O administrador seleciona a opção limpar BD(banco de dados)

Atores: administrador.

Pré-condição: Sistema na transação de gerenciamento de impressão digital .

Seqüência de eventos:
1- O administrador seleciona a opção limpar BD.
2- O sistema exclui do banco de dados todas as impressões digitais cadastradas do usuário, exibindo uma lista em branco de impressões digitais.

Pós-condição: sistema se encontra na situação de pré-condição exibindo a lista em branco de impressões digitais.

Extensões:
1- No passo 1, caso a lista de impressões digitais esteja vazia, o sistema exibirá uma mensagem de erro.

Inclusão:
Não há.

Caso de uso 4: Consultar usuários

Descrição: Este caso de uso descreve o processo de consultar dados cadastrais de usuários cadastrados no sistema.

Evento iniciador: O administrador seleciona a opção consultar usuário.

Atores: administrador.

Pré-condição: administrador autenticado no sistema.

Seqüência de eventos:

1- O administrador seleciona a opção consultar usuário.
2- O sistema exibe uma interface para consulta de usuário.
3- O administrador digita parte do nome do usuário a ser buscado.
4- O administrador dispara a busca.
5- O sistema retorna uma lista de usuários com seus respectivos dados cadastrais (nome do usuário, data de início da vigência de autorização e data de expiração, *status* indicativo da existência de impressão cadastrada, *status* indicativo da existência de cartão cadastrado).

Pós-condição: sistema exibindo uma lista de usuários encontrados.

Extensões:
1- No passo 4, caso o administrador não tenha informado a chave de busca, o sistema exibe uma mensagem indicativa do problema.
2- No passo 5, caso o sistema não encontre nenhum usuário, uma mensagem indicativa será exibida.

Inclusão: Não há

Caso de uso 5: Consultar vídeos

Descrição: Este caso de uso descreve o processo de consultar vídeos de *log* de acesso.

Evento iniciador: O administrador seleciona a opção consultar vídeos.

Atores: administrador.

Pré-condição: administrador autenticado no sistema.

Seqüência de eventos:

1- O administrador seleciona a opção consultar vídeos.
2- O sistema exibe uma interface para consulta de vídeos com os parâmetros de consulta (parte do nome do usuário, cujo vídeo de *log* deseja-se exibir e data em que o vídeo foi gravado).
3- O administrador preenche os parâmetros de consulta que julgar relevantes.
4- O administrador dispara a busca.
5- O sistema retorna uma lista de vídeos de *log* contendo data/hora de entrada do usuário em que foi disparada a gravação do vídeo, nome do usuário, uma opção para executar vídeo (caso de uso executar vídeo consultado).

Pós-condição: sistema exibindo uma lista de vídeos de *log* contendo data/hora de entrada do usuário em que foi disparada a gravação do vídeo, nome do usuário, uma opção para executar vídeo (caso de uso executar vídeo consultado).

Extensões:
1- No passo 4, caso o administrador não tenha informado qualquer uma das duas chaves de busca, o sistema exibe uma mensagem indicativa do problema.

2- No passo 5, caso o sistema não encontre nenhum vídeo, uma mensagem indicativa será exibida.
3- No passo 5, extend:caso de uso executar vídeo consultado.

Inclusão: Não há

Caso de uso 6: Executar vídeo consultado

Descrição: Este caso de uso descreve o processo de execução/visualização de vídeo consultado.

Evento iniciador: Administrador seleciona a opção de play para execução do vídeo de *log* desejado.

Atores: administrador

Pré-condição: Sistema exibindo uma lista de vídeos de *log* contendo data/hora de entrada do usuário em que foi disparada a gravação do vídeo, nome do usuário, uma opção para executar vídeo.

Seqüência de eventos:
1- Administrador seleciona a opção de play para execução do vídeo de *log* desejado.
2- Sistema exibe o vídeo desejado.

Pós-condição: sistema exibindo um vídeo.

Extensões:
1- No passo 1, se o administrador disparar mais de uma vez a execução de um determinado vídeo, o mesmo será reexibido.

Inclusão: Não há

Caso de uso 7: Autenticação do administrador no sistema

Descrição: Este caso de uso descreve o processo de autenticação do administrador junto ao sistema.

Pré-condição: necessidade de autenticação prévia do administrador junto ao sistema para execução de algum módulo (este caso de uso é um include).

Seqüência de eventos:

1- Administrador digita login e senha e submete ao sistema.
2- Sistema verifica o login e senha junto a sua base de dados.
3- Sistema autentica o administrador e fornece ao módulo chamador o respectivo login.

Pós-condição: administrador autenticado.

Extensões:
1- No passo 1, caso o administrador cancele a autenticação, o sistema volta ao estado do módulo chamador imediatamente anterior à chamada da autenticação..
2- No passo 2, caso o login ou senha sejam inválidos, o sistema exibe mensagem de acesso proibido, voltando ao passo 1. Isso prossegue indefinidamente até o administrador ser autenticado ou cancelar a operação de autenticação.

Inclusão: Não há.

Caso de uso 8: Ativar identificação

Descrição: Este caso de uso descreve a ativação do processo de identificação de usuário.

Evento iniciador: O administrador seleciona a opção ativar identificação na transação de gerência de identificação.

Atores: administrador.

Pré-condição: sistema se encontra na transação de gerência de identificação.

Seqüência de eventos:

1- O administrador seleciona a opção ativar identificação na transação de gerência de identificação.
2- O sistema executa o caso de uso: autenticação do administrador no sistema.
3- O sistema dispara o processo de identificação e desbloqueia opção parar identificação na transação de gerência de identificação.

Pós-condição: sistema pronto para identificar usuários cadastrados (dispositivo de leitura de smartcard ativado).

Extensões:
1- Não há.

Inclusão: caso de uso autenticação do administrador no sistema.

Caso de uso 9: Desativar identificação

Descrição: Este caso de uso descreve a desativação do processo de identificação de usuário.

Evento iniciador: O administrador seleciona a opção desativar identificação na transação de gerência de identificação.

Atores: administrador.

Pré-condição: sistema se encontra na transação de gerência de identificação.

Seqüência de eventos:

1- O administrador seleciona a opção desativar identificação na transação de gerência de identificação.
2- O sistema executa o caso de uso: autenticação do administrador no sistema.
3- O sistema pára o processo de identificação e desbloqueia opção ativar identificação na transação de gerência de identificação.

Pós-condição: sistema não pode identificar usuários cadastrados (dispositivo de leitura de smartcard desativado).

Extensões:
1- Não há.

Inclusão: caso de uso autenticação do administrador no sistema.

Caso de uso 10: Ativar servidor de vídeo

Descrição: Este caso de uso descreve a ativação do servidor de vídeo.

Evento iniciador: O administrador seleciona a opção ativar na transação de gerenciamento de câmera(vídeo digital).

Atores: administrador.

Pré-condição: sistema se encontra na transação de gerenciamento de câmera.

Seqüência de eventos:

1- O administrador digita o número da porta em que o servidor deve operar.
2- O administrador seleciona a opção ativar na transação de gerenciamento de câmera (vídeo digital).
3- O sistema executa o caso de uso: autenticação do administrador no sistema.
4- O sistema ativa o servidor, deixando esse em estado de listening (esperando alguma requisição para a gravação do vídeo de *log*) e desbloqueia a opção desativar na transação de gerenciamento de câmera (vídeo digital).

Pós-condição: servidor de vídeo ativado e em estado de listening (esperando alguma requisição para a gravação do vídeo de *log*).

Extensões:
1- No passo 2, caso o número da porta digitada não esteja entre 1440 e 65550, o sistema indica uma mensagem indicativa de erro(deve-se alterar o valor da porta), retornando ao passo 1.
2- No passo 2, caso o servidor já esteja executando, o sistema substituirá a porta em que o servidor está executando (obedecendo o critério de número de porta entre 1440 e 65550) , indo para o passo 4.

Inclusão: caso de uso autenticação do administrador no sistema.

Caso de uso 11: Desativar servidor de vídeo

Descrição: Este caso de uso descreve a desativação do servidor de vídeo.

Evento iniciador: O administrador seleciona a opção desativar na transação de gerenciamento de câmera (vídeo digital).

Atores: administrador.

Pré-condição: sistema se encontra na transação de gerenciamento de câmera.

Seqüência de eventos:

1- O administrador seleciona a opção desativar na transação de gerenciamento de câmera(vídeo digital).
2- O sistema executa o caso de uso: autenticação do administrador no sistema.
3- O sistema desativa o servidor e exibe uma mensagem de servidor inativo.

Pós-condição: servidor de vídeo desativado.

Extensões:
1- Não há.

Inclusão: caso de uso autenticação do administrador no sistema.

Caso de uso 12: Identificação de usuário

Descrição: Este caso de uso descreve o processo de identificação do usuário junto ao sistema.

Evento iniciador: O usuário aproxima o cartão do leitor de smartcard.

Atores: usuário comum/administrador(serão ambos tratados como usuário na seqüência de eventos).

Pré-condição: sistema pronto para identificar usuários cadastrados (dispositivo de leitura de smartcard ativado).

Seqüência de eventos:

1- O usuário aproxima o cartão do leitor de smartcard.
2- O sistema verifica se o cartão está cadastrado e se a data atual está contida no intervalo definido pela data de início e data de fim de autorização do sistema.
3- O sistema requisita que o usuário coloque seu dedo no leitor biométrico.
4- O usuário coloca o dedo no leitor biométrico.
5- O sistema captura a impressão digital e verifica se ela combina com as impressões digitais cadastradas do usuário, cujo smartcard foi autenticado.
6- O sistema inicia a gravação do vídeo de *log* (no servidor de vídeo).

7- O sistema dispara a abertura da fechadura eletrônica.

Pós-condição: acesso liberado.

Extensões:

1- No passo 2, caso o cartão não esteja cadastrado ou a data atual não esteja dentro do período de vigência para o usuário, uma mensagem de erro é enviada e o acesso não é permitido.

2- No passo 5, caso a impressão digital do usuário não combine com a previamente cadastrada, e o número de tentativas mal sucedidas for menor ou igual a 3, o sistema retorna para o passo 3.

3- No passo 5, caso o número de tentativas mal sucedidas exceda 3 tentativas, o sistema bloqueará o cartão do usuário em questão.

Inclusão: não há.

Apêndice 1 - Manual do Usuário

Índice do Manual do Usuário

Este manual tem por objetivo explicar os principais recursos, funções e comandos de gerenciamento do sistema de segurança usando biometria e *smartcard* - SysSec.

Requisitos do Sistema

Para que seja possível executar o sistema é preciso ter alguns programas ou arquivos instalados nos terminais, são eles:

- *Windows Media Player* 9 ou 10;
- SQL *Server* 2000;
- Máquina virtual JAVA versão 1.4.2;
- .NET *Framework* 1.1;
- SDK do leitor biométrico;
- Software de instalação do leitor biométrico;
- Rede TCP/IP entre terminal e servidor de vídeo;
- VirtualDub;
- Xvid.

Para instalar o SysSec, deve-se:

- Transferir o diretório do software responsável pela gerência para um diretório de sua escolha. Neste diretório deve-se criar uma pasta denominado Vídeo;
- Transferir os arquivos referentes ao subsistema vídeo de *log* para uma pasta no servidor de vídeo.

Manual de Operação – Servidor de Vídeo

Este módulo é de uso exclusivo de administradores do sistema.

Instalação

1) Antes de instalar o subsistema servidor de vídeo, certifique-se de que:

- A *Webcam* esteja conectada ao computador em que o programa de vídeo irá executar;
- O *driver* da *Webcam* esteja instalado;
- A máquina virtual Java, versão 1.4.2, esteja instalada na máquina (no servidor de vídeo);
- .NET *Framework* instalado no equipamento (tanto no cliente quanto no servidor);
- A API Java Media *Framework*, versão 2.1.1 e (Windows *Performance Pack*) esteja instalado na máquina;
- A *Webcam* deve estar registrada no programa JMF *Registry*, na terceira posição da lista (conforme figura abaixo). Utilize o botão **Commit** para registrar a *Webcam* e o botão **move up** ou **move down** para mover na posição da lista.

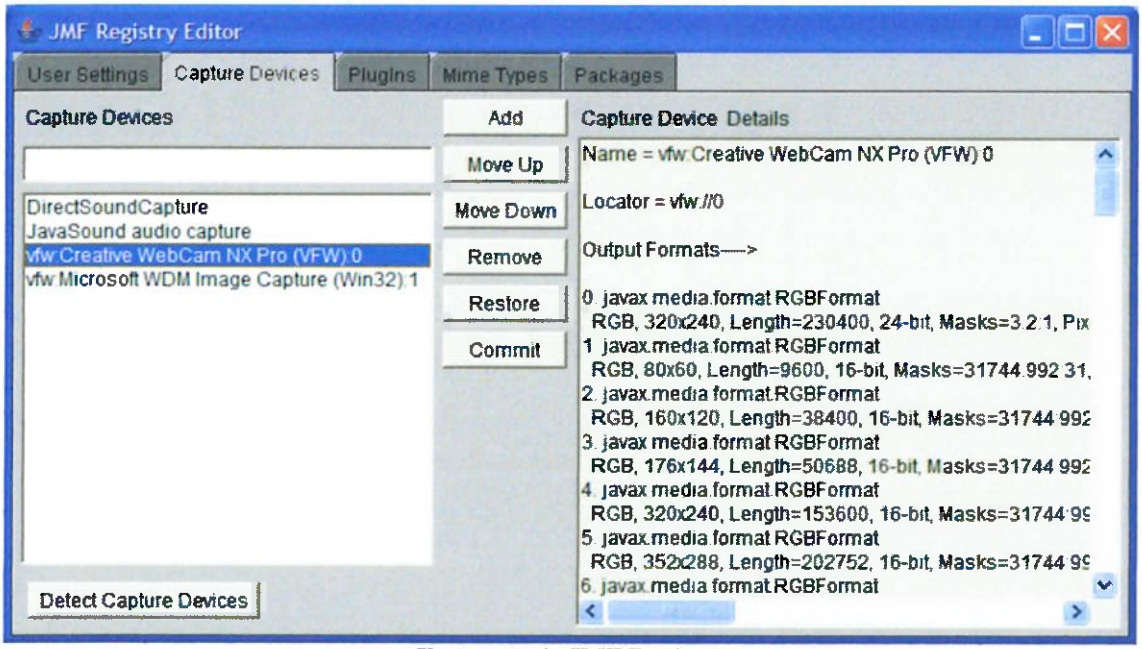

Ilustração 1- JMF Registry

2) Transferir os arquivos referentes ao subsistema vídeo de *log* para uma pasta no servidor de vídeo.

Inicialização do Sistema SysSec – Servidor de Vídeo

Para iniciar o programa servidor de vídeo, execute o arquivo CamGen.exe. Para ter acesso ao sistema, o administrador deverá se autenticar no sistema (ilustração 2). Para tanto, o administrador deve:

- Preencher o **nome de usuário** e **senha**. Se não tiver nenhum administrador registrado no sistema, pode-se usar o administrador mestre para esta tarefa;
- Clicar no botão **OK** para confirmar a autenticação, ou no botão **Cancelar** (neste caso o programa será fechado).

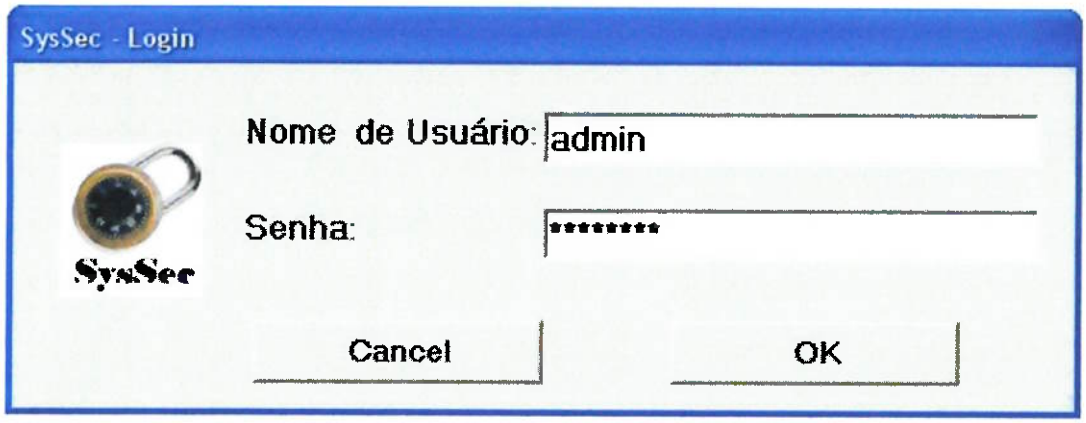

Ilustração 2 - Autenticação SysSec

Se a autenticação for aceita, aparecerá a tela de gerenciamento da câmera. Nessa tela podemos ativar ou desativar o *status* do servidor do vídeo, ocultar esta janela, fazer testes em relação à câmera.

Ilustração 3 - Tela Gerenciamento Câmera

Ativando o servidor de vídeo

Para ativar o servidor de vídeo, é preciso:

- Escolher uma porta entre 1440 até 65550 (campo **Porta Servidor**);
- Clicar no botão **Ativa**.

Após o administrador autenticar-se no sistema, o servidor de vídeo será aberto escutando na porta selecionada, o *status* do servidor de vídeo mudará para **Servidor Ativo** e o botão **Fechar** vai ficar habilitado, conforme pode ser visto na ilustração 4, exceto em caso de valores inválidos de porta.

Ilustração 4 - Servidor Ativo

Desativando o servidor de vídeo

Para desativar o servidor de vídeo, é preciso clicar no botão **Fecha**. Após a autenticação do administrador, o servidor de vídeo será fechado, o *status* do servidor muda para Servidor Inativo e o botão **Fecha** fica desabilitado.

Deslogar e Logar usuário

Se o usuário desejar, ele pode deslogar do sistema (opção **Arquivo→Deslogar**). Esta opção bloqueia os campos contidos na área **Ativar Servidor**. A tela de gerenciamento câmera fica como na ilustração abaixo.

Ilustração 5 - Usuário Deslogado

Para voltar ao estado anterior, o usuário deve utilizar a opção logar (**Arquivo→Logar**).

Rodando em Segundo Plano

Se o usuário desejar, pode-se ocultar a tela gerenciamento de câmera. Para ativar esta opção basta clicar no botão **Arquivo** e selecionar **Rodar em segundo plano**, a janela irá se ocultar e um ícone do aplicativo ficará na barra de tarefas.

Para abrir a janela novamente, é só clicar no ícone SysSec com o **botão direito do mouse**. Aparecerá um *menu* com as opções Ativar Servidor (11000), Desativar Servidor, Restaurar e Fechar. Selecione a opção **restaurar**. Por motivos de segurança, a autenticação é requerida sempre que seja necessário restaurar a janela, ou usar o botão Fecha, ou algum outro item do *menu* associado ao ícone do aplicativo na barra de tarefas.

Teste da Câmera de Vídeo

Este teste tem a função de testar o funcionamento da câmera de vídeo, podendo verificar o seu posicionamento e a qualidade da imagem oferecida. Para entrar no modo de teste, clique no botão Câmera, e selecione a opção Teste (**Câmera→Teste**). Será exibida uma mensagem de aviso, mostrada na ilustração abaixo.

Ilustração 6 - Mensagem de aviso de teste da câmera

Para realizar efetivamente o teste, basta clicar no botão **Executar**. Ao apertar o botão, aparecerá uma janela mostrando o que a câmera está filmando (ilustração abaixo). Atenção: neste momento o servidor de vídeo ficará desabilitado.

Para voltar à tela anterior e finalizar o teste, clique no botão **Fechar** (o servidor de vídeo desabilitado pelo teste voltará a executar automaticamente neste momento).

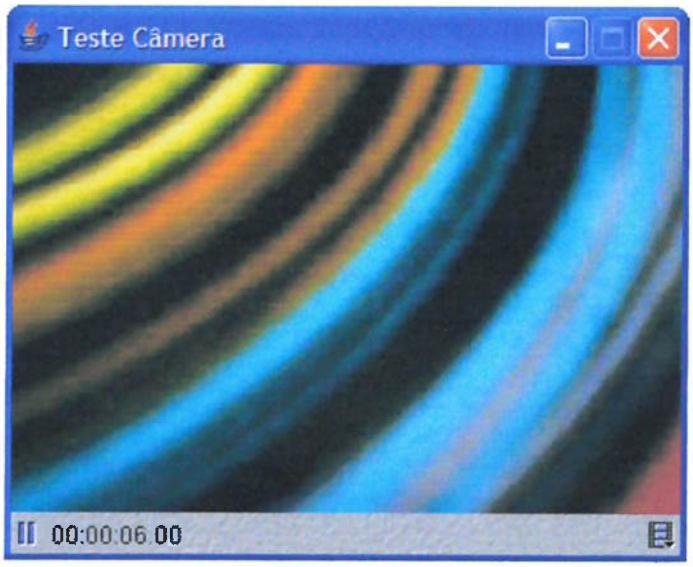

Ilustração 7 - Teste câmera

Fechar o Aplicativo CamGen

Para fechar o aplicativo CamGen, o usuário pode selecionar a opção fechar associada ao ícone na barra de tarefa (**botão direito do mouse no ícone → fechar**) ou usar a opção fechar no menu Arquivo (**Arquivo→Fechar**). Por questões de segurança, o administrador deverá se autenticar para fechar o aplicativo.

Manual de Operação – Gerência Administrador

Este módulo é de uso exclusivo de administradores do sistema.

Instalação

1) Antes de instalar o subsistema de gerência, certifique-se que:

- O leitor *smartcard* está conectado ao PC;
- O leitor biométrico está conectado ao PC;
- O *driver* do leitor biométrico está instalado no PC;
- O SDK biométrico está instalado no PC;
- .NET *Framework* instalado no PC.
- *Windows Media Player 9 ou 10* instalado no PC;

2) Transferir os arquivos referentes ao subsistema vídeo de *log* para uma pasta no servidor de vídeo.

Inicialização do Sistema SysSec – Terminal do Administrador

Para iniciar o programa servidor de vídeo, execute o arquivo CONTROLE_ACESSO.exe. Para ter acesso ao terminal do administrador, o administrador deverá se autenticar no sistema (ilustração 8). Para autenticar-se no sistema, o administrador deve:

- Preenchimento do **nome de usuário** e **senha**. Se não tiver nenhum administrador registrado no sistema, pode-se usar o administrador mestre para esta tarefa.
- Clicar no botão OK para confirmar a autenticação, ou no botão Cancelar (neste caso o programa será fechado).

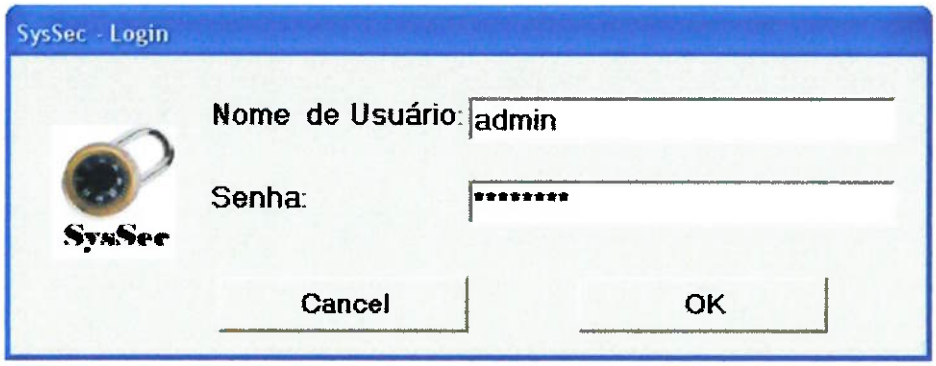

Ilustração 8 - Autenticação

Se a autenticação for aceita, aparecerá a tela de Gerência Administrador (ilustração 9).

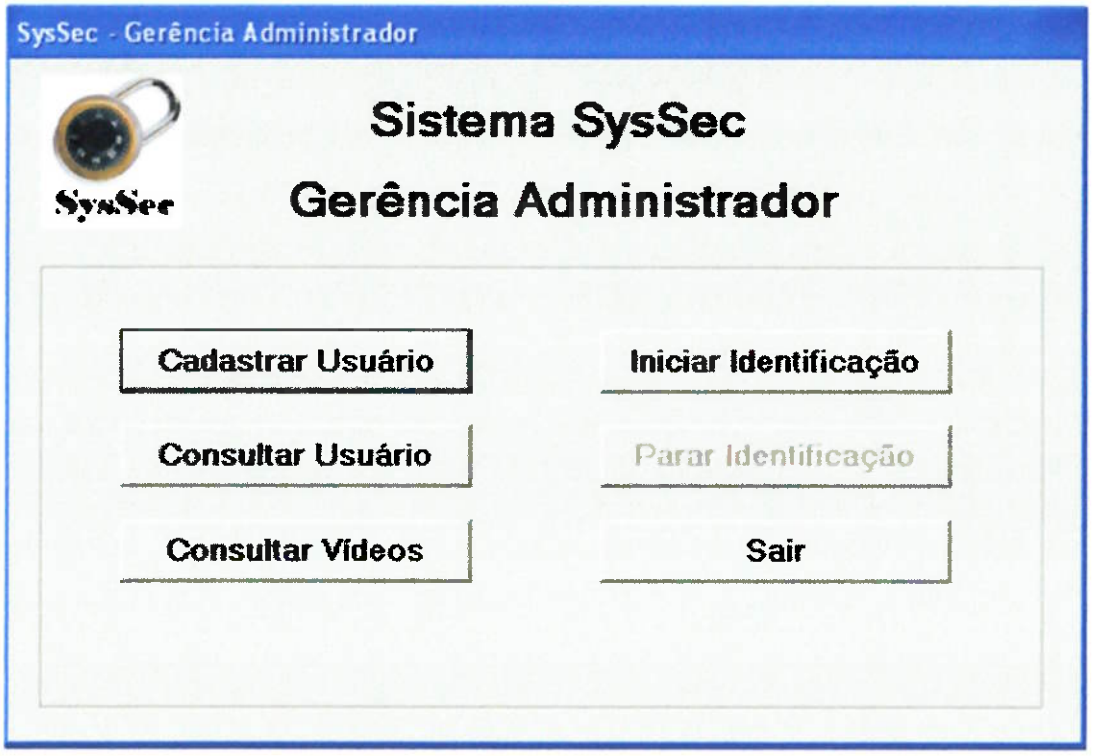

Ilustração 9 - Gerência Administrador

A partir desta tela, podemos ativar as seguintes funções:

- Iniciar e Parar Identificação (apenas para teste, essas funcionalidades aparecem também na tela Gerência de Identificação);
- Cadastrar Usuários (cadastrar, remover, alterar dados dos usuários);
- Consultar Usuário;
- Consultar Vídeos;
- Sair.

Gerência de Usuário

Para realizar a gerência de usuários, deve-se apertar o botão **Cadastrar Usuário** na tela Gerência Administrador (ilustração abaixo).

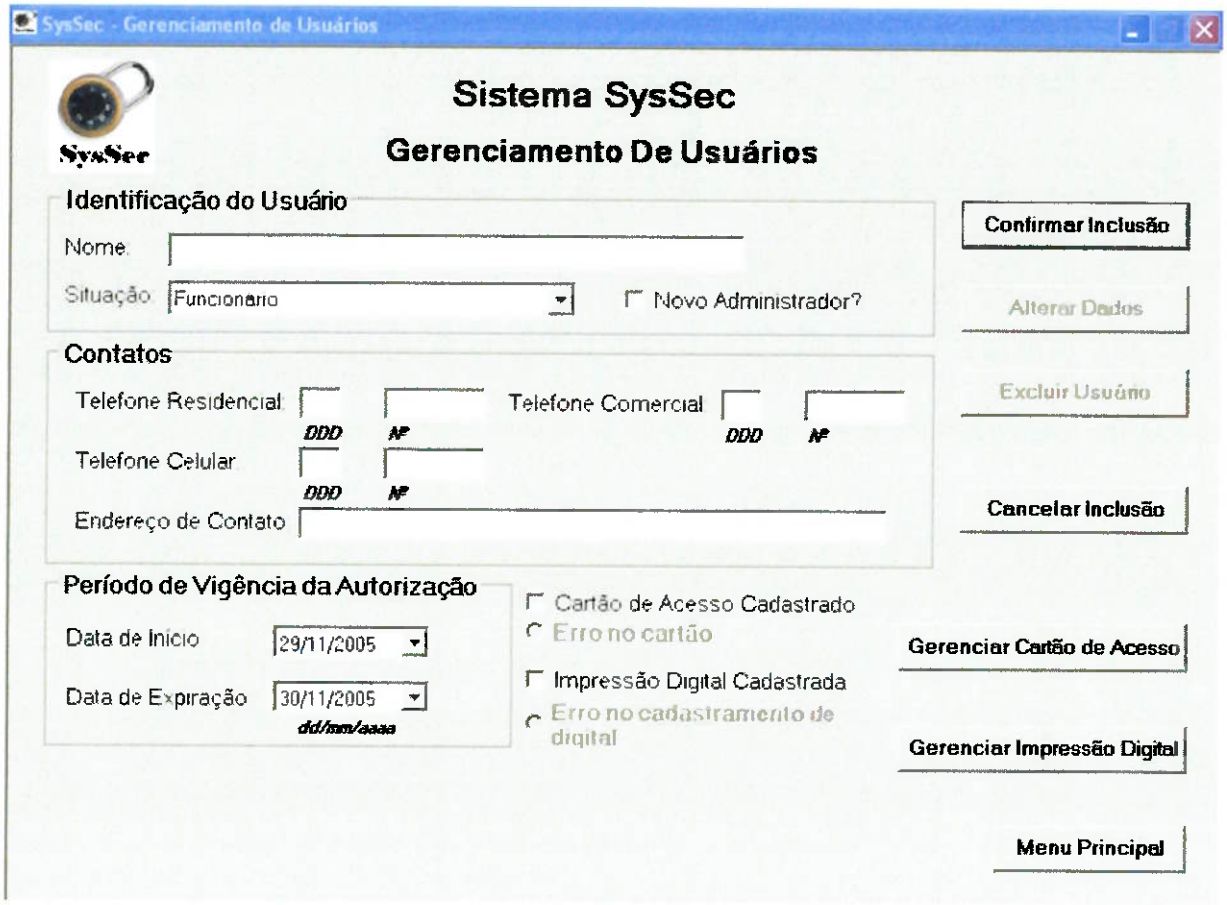

Ilustração 10 - Gerenciamento de Usuários

Cadastrar Usuário

Para cadastrar um usuário no sistema, o administrador deve preencher os seguintes campos:

- **Nome** – nome do usuário;
- **Situação** – escolher o *status* do usuário;
- **Telefone Residencial** – telefone residencial do usuário;
- **Telefone Comercial** – telefone comercial do usuário;
- **Telefone Celular** – telefone celular do usuário;
- **Endereço de Contato** – endereço para contato com o usuário;

- **Data de Início** – data do início da permissão de acesso do usuário ao local restrito;
- **Data de Expiração** – data do fim da permissão de acesso do usuário ao local restrito.

Preenchido os dados, deve-se confirmar o cadastramento por meio do botão **Confirmar Inclusão**. Após a confirmação, pode-se cadastrar o cartão de acesso.

Cadastrar Administrador

Para cadastrar um administrador, além dos dados do usuário, deve-se:

- Marcar a opção **Novo Administrador?** (os campos **login**, **senha** e **confirmação** irão aparecer)
- **Login:** Preencher com o login do administrador;
- **Senha:** Preencher com a senha do administrador;
- **Confirmação:** Preencher com a senha do administrador para confirmação da senha;

Preenchido os dados, deve-se confirmar o cadastramento por meio do botão Confirmar Inclusão. Após a confirmação, pode-se cadastrar o cartão de acesso.

Gerenciamento de Cartão de Acesso

Uma vez que o usuário ou o administrador tenha sido cadastrado no sistema, o botão **Gerenciar Cartão de Acesso** (ilustração 10) será desbloqueado. O cadastramento do cartão poderá ser feito quantas vezes forem necessárias, contato que o usuário já esteja cadastrado.

Para realizar o cadastro do cartão, deve-se prosseguir da seguinte maneira:

- O administrador (que está realizando o cadastro) irá clicar em **Gerenciar Cartão de Acesso**, o sistema será redirecionado para uma tela com um botão chamado **Cadastrar Cartão** (ilustração abaixo).

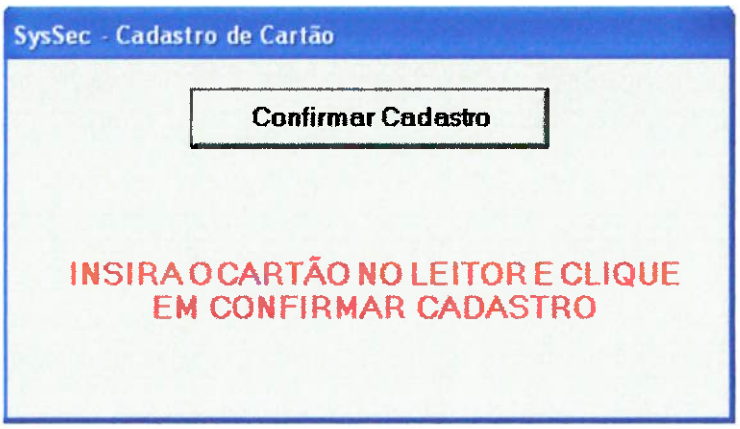

Ilustração 11 - Tela Cadastro de Cartão

- Deve-se aproximar o cartão a ser cadastrado do leitor de cartão.
- Com o cartão próximo do leitor, aperte o botão **Confirmar Cadastro**.
- O cartão deve ficar próximo do leitor até aparecer a mensagem abaixo.

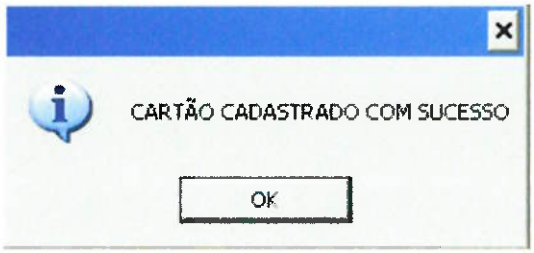

Ilustração 12 - Tele de confirmação de cadastro do cartão

Neste momento, o cartão de acesso foi cadastro. A partir desse momento, o cadastramento de impressão digital será permitido.

Gerenciamento de Impressão Digital

Esta funcionalidade gerencia as impressões digitais de um usuário específico, cadastrando e removendo as digitais (é permitido cadastrar no máximo 10 impressões digitais por usuário). Para poder gerenciar a impressão digital de um usuário, o cartão de acesso do usuário já deve ter sido cadastrado.

Para realizar o gerenciamento da impressão digital, o administrador deverá clicar no botão **Gerenciar Impressão Digital**. Ao clicar no botão, o sistema será redirecionado para a tela de gerenciamento de impressão digital.

Esta tela permite realizar três operações:

- Cadastrar uma nova impressão digital (limite de 10) – **Registrar Digital no BD**;
- Remover uma impressão digital entre as várias cadastradas – **Remover Digital**;
- Remover todas as digitais cadastradas – **Limpar BD**.

Ilustração 13 - Tela de gerenciamento de impressão digital

Para registrar a impressão digital do novo usuário, clique em **Registrar Digital no BD**. Aparecerá uma tela de registro.

Para continuar aperte o botão **NEXT**.

A próxima tela pergunta qual dedo o novo usuário se cadastrará. O administrador deve perguntar para a pessoa que está tendo sua impressão digital cadastrada, qual é o dedo que ela deseja cadastrar e clicar no círculo azul do dedo correspondente ao que será colocado no leitor.

Será pedido que o usuário coloque o dedo no leitor e tire somente quando a digital ficar com cor azul como na figura acima. Terminado este processo, é preciso fazer uma segunda leitura, do mesmo modo que a primeira leitura. Terminada a segunda leitura da digital,

aparecerá a tela de escolha de dedos, sendo possível registrar outros dedos do usuário. Para terminar o cadastro de digitais, clique no botão **NEXT**.

Os dedos que foram registrados estão indicados com um círculo rosa, como no exemplo acima em que foi cadastrado somente o dedo indicador da mão direita. Para finalizar, clique no botão **FINISH**. O sistema voltará para a tela de gerenciamento de impressão digital com as digitais já cadastradas no sistema.

Para remover alguma digital de um usuário, basta selecionar a digital a ser removida e clicar no botão **Remover Digital**, logo após aparecerá uma janela dizendo que a remoção foi efetuada com sucesso.

Para remover todas as digitais cadastradas do usuário selecionado, deve-se apertar o botão **Limpar BD**.

Exclusão de Usuários

A remoção de usuários faz o cancelamento no sistema da conta de um usuário específico, apagando todos os seus dados cadastrais, assim como os vídeos de *log*, identificação do cartão de acesso e da impressão digital referente ao usuário a ser excluído. Para efetuar a operação de remoção, deve-se proceder da seguinte maneira:

- Selecionar o usuário a ser excluído entre os nomes que aparece na tela de gerenciamento de usuários (ilustração abaixo).

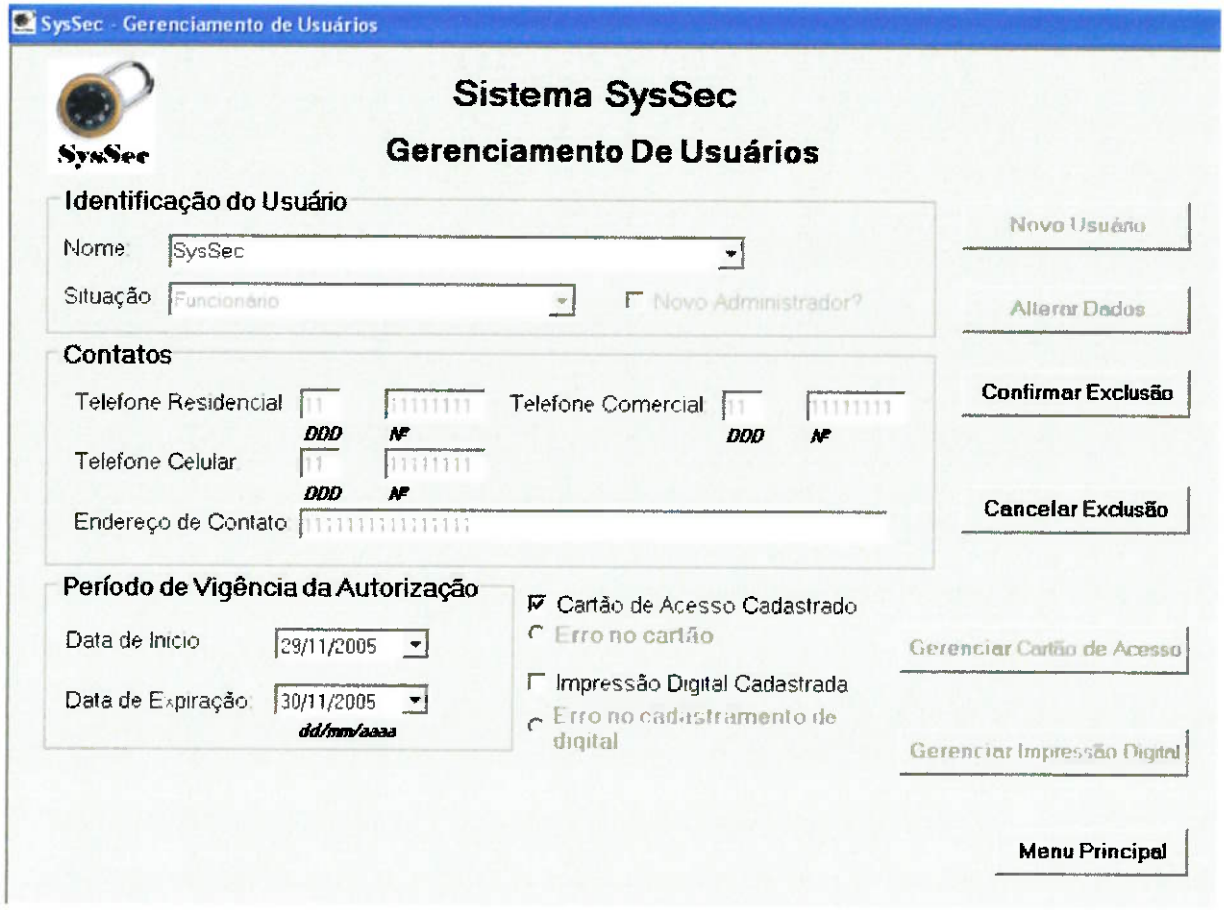

Ilustração 14 - Gerenciamento de Usuários

- Em seguida, deve-se apertar o botão Excluir Usuário. Pode-se confirmar a exclusão clicando em **Confirmar Exclusão,** ou cancelar a exclusão clicando em **Cancelar Exclusão.**

- Confirmada a exclusão, aparecerá uma janela perguntando se realmente deseja remover este usuário. Apertando o botão **OK,** o usuário será removido do sistema. Apertando o botão **Cancelar,** volta-se à tela de Cadastro de Usuários sem que a exclusão do usuário selecionado seja efetivada.

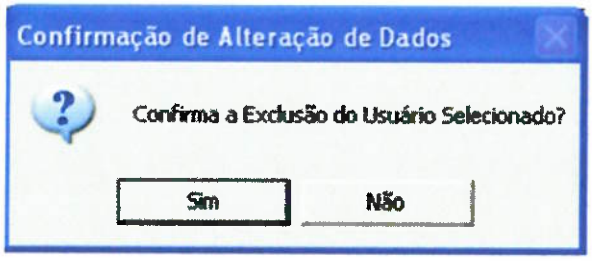

Ilustração 15 - Confirmação de exclusão

Exclusão de Administradores

Para excluir usuários administradores, deve-se proceder da mesma forma que com os usuários, exceto pelo fato de que apenas o administrador mestre possui poder para excluir um administrador do sistema.

Alterando dados de um usuário/administrador

Essa funcionalidade é usada para modificar um ou mais dados no registro de um usuário específico que necessite ser atualizado. O administrador pode alterar todos os dados, exceto a senha e o login de outros administradores. Esta funcionalidade (alterar login e senha de outros administradores) só pode ser feita pelo administrador mestre.

Para alterar os dados de um usuário ou administrador, deve-se proceder da seguinte maneira:

- Na tela de gerenciamento de usuários, deve-se escolher o usuário/administrador a ser alterado;
- Para poder alterar os dados deve-se clicar em **Alterar Dados**;
- A edição dos dados será permitida. Altere os campos desejados;
- Clique no botão **Confirmar Alteração** para que os dados sejam modificados ou no botão **Cancelar alteração**, caso deseje cancelar a alteração.

Consultando usuários

Esta funcionalidade tem como objetivo pesquisar usuários através de uma busca simples pelo nome ou por parte do nome, mostrando os dados cadastrados dos usuários encontrados, tais como Nome do Usuário completo, Data de início e de fim de autorização, *status* da biometria (indicativo do cadastro de impressão digital) e *status* do cartão (indicativo do cadastro do cartão).

Para realizar a consulta, deve-se:

- Na tela de Gerência Administrador, selecionar a opção **Consultar Usuários**. O sistema será redirecionado para a tela de consulta de usuário;

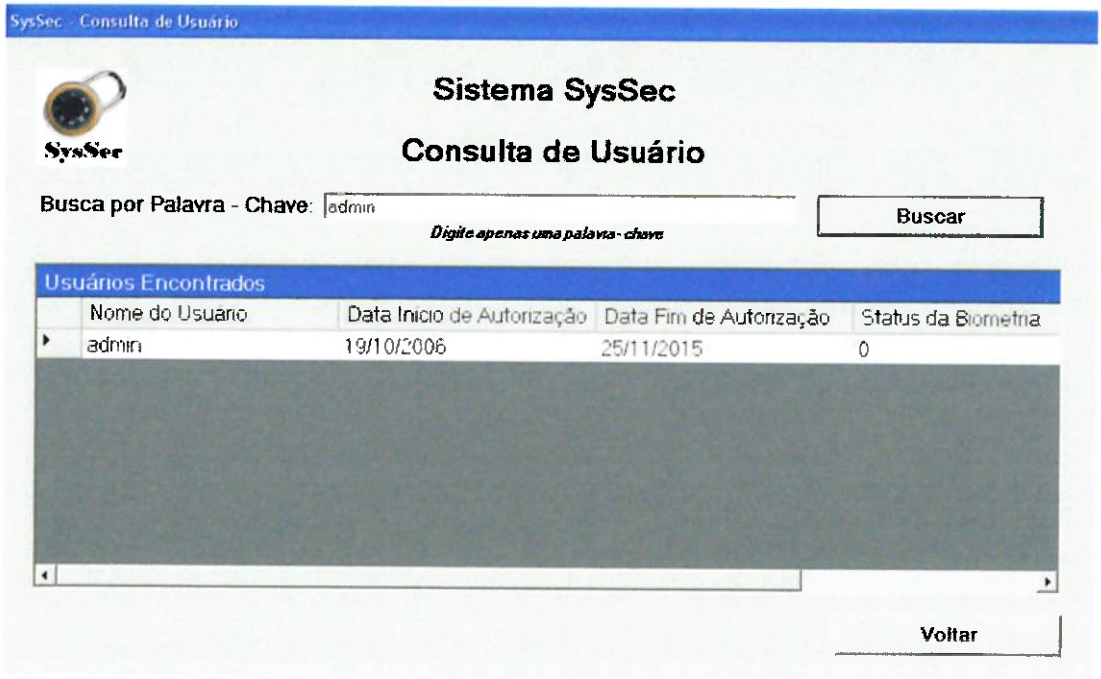

Ilustração 16 – Tela Consulta de Usuário

- No campo **Busca por Palavra - Chave**, colocar o nome do usuário que se deseja buscar, podendo realizar a busca por nomes completos ou incompletos;
- Clicar no botão **Buscar;**
- O sistema retornará uma lista com os usuários que satisfaçam a palavra-chave informada.

Consultando Vídeos gravados

A consulta de vídeos faz uma pesquisa em todos os vídeos de *log* gravados no servidor. A busca é feita utilizando-se uma Palavra – Chave que seja parte do nome do usuário, ou por uma data especifica, ou pelos dois parâmetros.

Para acessar a consulta:

- Clique no botão **Consultar Vídeos** que está na tela inicial do administrador, o sistema será redirecionado para a tela de Consulta de Vídeos.

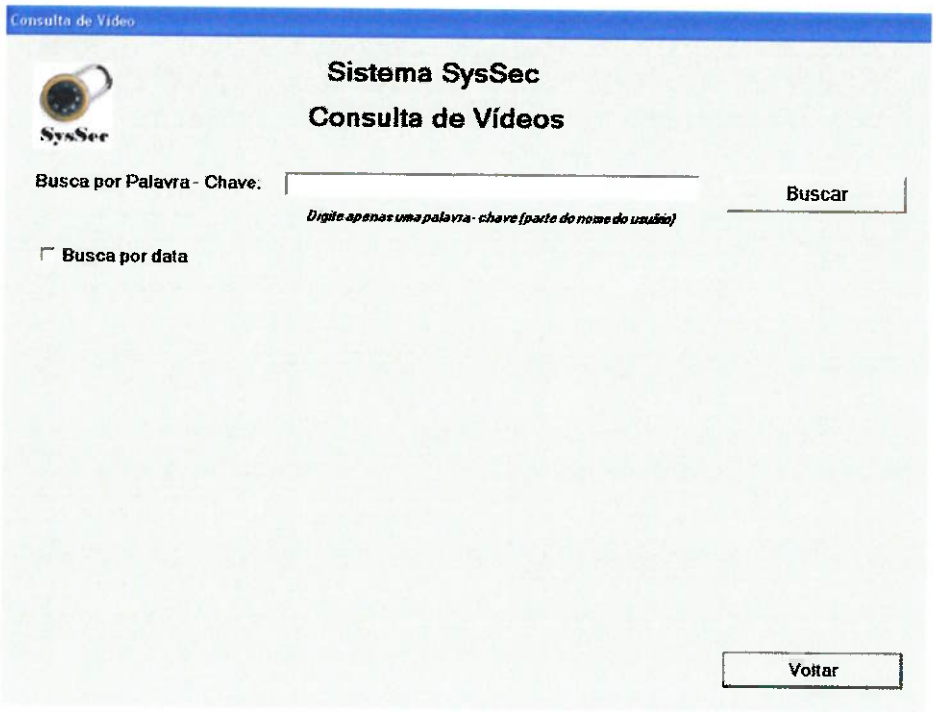

Ilustração 17 - Tela Consulta de Vídeo

- Preencha os parâmetros de busca. Existem dois parâmetros de busca que podem ser utilizados (pode-se utilizar um dos dois ou os dois simultaneamente).
    - Para usar a **Busca por Palavra – Chave** deve-se preencher o campo **Busca por Palavra – Chave** com o nome ou parte do nome do usuário.
    - Para usar a busca por Data, deve-se clicar no box **Busca por data** e escolher a data a ser buscada (busca por data).

- Clique no botão Buscar
- O sistema trará todos os vídeos associados aos parâmetros de busca associados.
- Para verificar o conteúdo dos vídeos, basta clicar no botão PLAY da linha referente ao vídeo que se deseja exibir. O sistema irá abrir o programa *windows media player* para executar/exibir o vídeo.

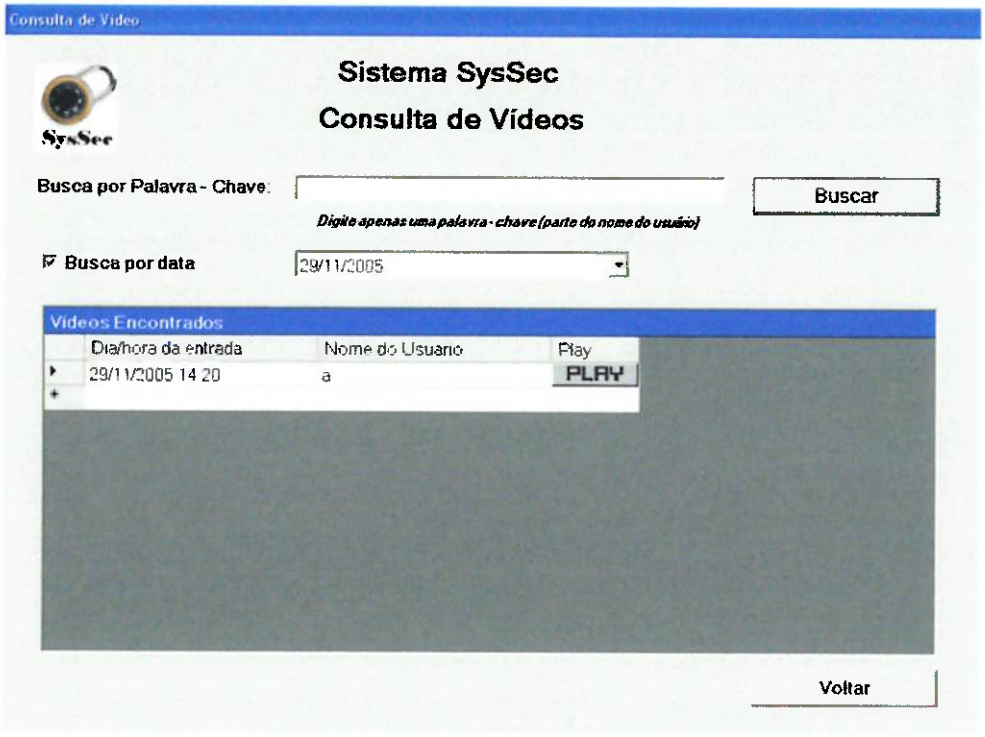

Ilustração 18 - Busca por Data

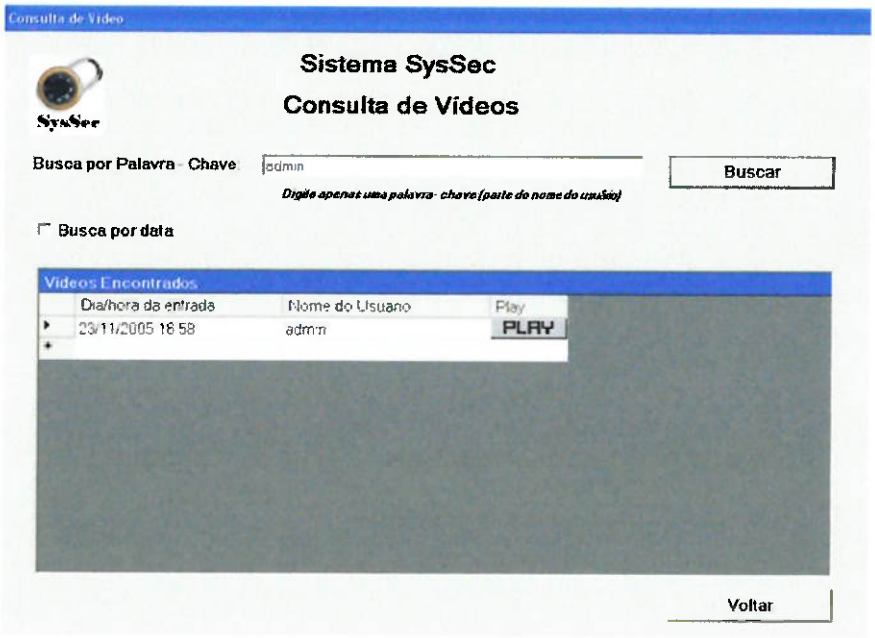

Ilustração 19 - Busca por Palavra-Chave

Iniciar/Parar Identificação

O botão **Iniciar Identificação,** na tela de Gerência de Identificação/Gerência Administrador, inicia o processo de identificação de usuários, habilitando a leitura de cartões. O botão **Parar Identificação** pára este processo.

Manual de Operação – Identificação

Este módulo é de uso dos usuários e administradores do sistema

O processo de identificação descreve como um usuário que deseja acessar o local restrito deve proceder.

Para acessar o local restrito, o usuário cadastrado deve primeiramente utilizar o seu cartão de acesso. O usuário deve aproximar o cartão do leitor de cartão. Se o sistema reconhecer seu cartão como válido, aparecerá uma mensagem no terminal do usuário pedindo para que ele coloque um dos seus dedos, cadastrados previamente, no leitor de impressão digital.

Se a impressão digital for aceita, será gerado um vídeo de *log* do acesso, e, após o *log*, a fechadura eletrônica será destravada. Caso a impressão digital não seja aceita, o sistema não irá liberar o acesso do usuário ao sistema.

Para este protótipo, as principais mensagens relacionadas ao processo de identificação são exibidas em uma janela de *log* (ilustração 20) e armazenadas em um arquivo texto (denominado *log*.txt).


ADJUST
CANCEL

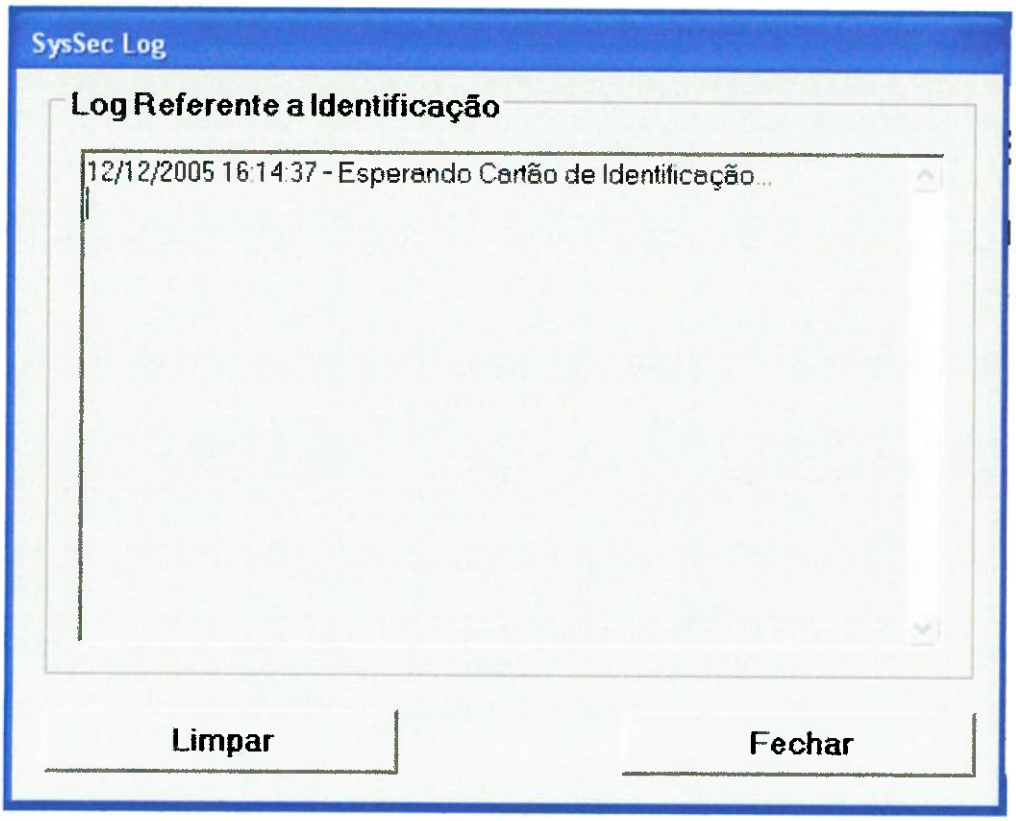

Ilustração 20 - Janela de *log* referente a identificação

Abaixo, segue um exemplo das mensagens de log extraído do arquivo *log*.txt:

- 9/12/2005 14:05:20 - Esperando Cartão de Identificação...
- 9/12/2005 14:05:27 - Identificado Cartão
- 9/12/2005 14:05:27 - Identificação Biométrica
- 9/12/2005 14:05:31 - Acesso Permitido
- 9/12/2005 14:05:31 - Vídeo de log gravado
- 9/12/2005 14:05:31 - Fechadura Eletronica Aberta
- 9/12/2005 14:05:32 - Esperando Cartão de Identificação...
- 9/12/2005 14:05:34 - Identificado Cartão
- 9/12/2005 14:05:34 - Identificação Biométrica
- 9/12/2005 14:05:41 - Acesso negado - tentativa nº: 1

Apêndice 2 – História da dactiloscopia

Um histórico da evolução da dactiloscopia e de alguns processos de identificação (no Brasil e no mundo):

Ano 650

A impressão digital é usada em processos de divórcio na China.

Ano 782

No Turquestão, são descobertas placas de cerâmica lavradas que demonstram que impressões digitais eram usadas em assinatura de acordos (para firmar acordos).

Ano 800

Na Índia, as impressões digitais são utilizadas por analfabetos para legalizar os seus papéis.

Ano 1300

Os chineses empregam a impressão digital nos casos de crimes (além de processos de divórcio).

Ano 1664

Um médico italiano chamado Marcelo Malpighi publica um trabalho intitulado "Epístola sobre o órgão do tato" que contém um estudo sobre os desenhos digitais e palmares.

Ano 1823

João Evangelista Purkinje apresenta à Universidade de Breslau, na Alemanha, uma tese na qual analisa os caracteres externos da pele e estuda o sistema déltico.

Ano 1856

José Engel publica o "Tratado do Desenvolvimento da Mão Humana" em que faz estudos sobre os desenhos digitais.

Ano 1858

Henry Faulds, inglês, médico de hospital em Tóquio, prevê a possibilidade de se descobrir um criminoso pela identificação das linhas papilares e preconiza uma técnica para a tomada de impressões digitais, utilizando-se de uma placa de estanho e tinta de imprensa.

Ano 1882

É lançado em Paris por Alfonse Bertillon o "Sistema Antropométrico", considerado o primeiro sistema científico de identificação (está baseado nos elementos antropológicos do homem: diâmetro da cabeça, comprimento da orelha direita, entre outros).

Ano 1888

Francis Galton, nobre inglês, tentando estabelecer um sistema de identificação mais seguro que a antropometria, lança as bases científicas da impressão digital. Seus estudos serviram como base (ponto de partida) para os demais sistemas dactiloscópicos.

Ano 1891

Em 1° de Setembro, Juan Vucetich apresenta seu sistema de identificação com o nome de "Icnofalangometria".

Ano 1894

Dr. Francisco Latzina publica no jornal "La Nacion", de Buenos Aires, um artigo no qual sugere que o nome Icnofalangometria fosse substituído por Dactiloscopia (do grego, *DA'KTYLOS* = dedos; *SKOPÊIN* = examinar).

Ano 1900

Edward Richard Henry publica na Inglaterra seu livro "*Classification and Uses of Finger Prints*" onde expõe seu novo sistema de identificação dactiloscópico.

Ano 1901

A Scotland Yard, na Inglaterra, adota o sistema dactiloscópico de Henry oficialmente.

Ano 1902

No Rio de Janeiro, José Alves Felix Pacheco inicia a tomada da impressão digital nas fichas antropométricas.

Ano 1903

É regulamentada uma lei instituindo o sistema dactiloscópico Vucetich no Rio de Janeiro.

Ano 1904

Em 29 de Julho, é expedida a primeira carteira de identidade no Brasil, na época denominada "Ficha Passaporte" ou "Cartão de Identidade" (eram ainda usados assinalamentos antropométricos junto com a dactiloscopia).

Ano 1907

Por meio de um decreto, o sistema Dactiloscópico Vucetich foi adotado no Estado de São Paulo.

Ano 1935

São criados o "Arquivo Dactiloscópico Monodactilar" e o "Laboratório de Locais do Crime" em São Paulo.

Ano 1941

É promulgado o Código de Processo Penal no Brasil em que estabelece a identificação dactiloscópica nos indiciados em inquérito policial.

Ano 1960

A partir desse ano, o FBI começa a investir em um sistema biométrico automático (AFIS).

Ano 1963

É inaugurado o "Instituto Nacional de Identificação" em Brasília.

Ano 1998

O "Instituto de Identificação do Estado de São Paulo" possui um acervo com, aproximadamente, 40.000.000 prontuários e expede, diariamente, em média 15.000 Cédulas de Identidade e 300 Atestados de Antecedentes Criminais.

Apêndice 3 – Material eletrônico

O material eletrônico (documentação e o programa) se encontra em anexo a esta documentação (CD **PCS2502-B1-CD-2005**).

O material eletrônico está dividido nas seguintes seções:

- **Documentação** – Documentação desenvolvida para o projeto (monografia, folhas de rosto e apresentação) no formato PDF (*Portable Document Format*);
- **SysSec** – Programa SysSec executável;
- **Fontes** – Código-fonte desenvolvido neste projeto.