CENTRO UNIVERSITÁRIO DE BRASÍLIA - UniCEUB
FACULDADE DE CIÊNCIAS EXATAS E TECNOLOGIA - FAET
CURSO DE ENGENHARIA DE COMPUTAÇÃO

FÁBIO RIBEIRO DE CASTRO

ESTUDO DO USO DE BIOMETRIA PARA AUTENTICAÇÃO REMOTA EM BLITZ DE TRÂNSITO

BRASÍLIA
2008

FÁBIO RIBEIRO DE CASTRO

ESTUDO DO USO DE BIOMETRIA PARA AUTENTICAÇÃO REMOTA EM BLITZ DE TRÂNSITO

Trabalho de conclusão de curso apresentado à banca examinadora da Faculdade de Ciências Exatas e Tecnológicas, do Centro Universitário de Brasília (UniCEUB), como exigência parcial à obtenção do grau de bacharel em Engenharia de computação, realizado sob a orientação do Professor: Mestre Roberto Schaefer.

BRASÍLIA
2008

FÁBIO RIBEIRO DE CASTRO

ESTUDO DO USO DE BIOMETRIA PARA AUTENTICAÇÃO REMOTA EM BLITZ DE TRÂNSITO

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB), como exigência parcial para obtenção do grau de bacharel em Engenharia de Computação.
Orientador: Professor Mestre Roberto Schaefer.

Brasília, ___ de _____ de 2008.

Banca Examinadora:

_____
Prof. Dr.:

_____
Prof. Dr.:

Agradecimentos

Primeiramente agradeço aos meus pais que sempre acreditaram em mim, me dando força e motivação para concluir o mais rápido possível o curso.

Aos meus irmãos que mesmo sem muito conhecimento me davam todo o suporte e conselhos que precisava.

Aos meus avós que nunca duvidaram da minha capacidade.

Aos meus amigos que estiveram comigo nessa longa caminhada, me apoiando e motivando sempre que necessário. Especialmente o Henrique Silva Moura que me ajudou muito após a minha ida pra São Paulo.

À minha namorada Daniela Pedrinha, por me incentivar, ajudar nos momentos mais difíceis, nas verificações de português e ortografia e também por toda sua compreensão ao longo desse último semestre.

Ao meu professor e orientador Roberto Schaefer pelo apoio e confiança em mim de que eu daria conta de terminar o trabalho a tempo.

Resumo

A impressão digital é uma das tecnologias mais seguras para autenticação, pois cada indivíduo possui características humanas e biológicas únicas. Por estes motivos tal modalidade é uma das formas mais rápidas, barata e eficaz de identificação das pessoas. Com base neste dado, esse trabalho busca demonstrar o método da biometria e os seus tipos de autenticação para substituir as atuais Carteiras Nacional de Habilitação, e propor uma solução capaz de aumentar a segurança pública. Pode contribuir para reduzir as multas e os incômodos por esquecimento dos documentos necessários para dirigir devidamente identificado. Sem contar com a facilidade de integração com outras bases de dados do governo. Para alcançar o objetivo deste trabalho foi desenvolvido em Java um sistema de cadastramento, consulta e alteração dos dados, da foto e da digital do motorista, onde a consulta ao banco de dados PostgresSQL é feita de forma remota através do uso da internet via modem 3G.

PALAVRAS-CHAVE: Impressão Digital. Biometria. Carteira Nacional de Habilitação. Autenticação remota. Blitz de trânsito.

SUMÁRIO

Introdução

Com o aumento da frota de carros e da população que dirige, é necessária uma maior verificação dos documentos dos motoristas. Para o Departamento Nacional de Trânsito é cada vez mais difícil fazer o controle dos mesmos, devido ao crescente número de fraudes aplicadas no Brasil. Por conta do alto índice de fraudes registradas, o DENATRAN em conjunto com outros órgãos públicos estão pesquisando um método mais seguro de autenticação da Carteira Nacional de Habilitação e dos outros documentos existentes.

Uma das técnicas pesquisadas a fim de implementar essa mudança é a do uso da biometria, que tem por finalidade utilizar as características biológicas individuais de cada motorista para identificar uma pessoa baseado em suas características próprias como as veias das mãos, a íris, a voz, o formato da face, a impressão digital.

A técnica da biometria é vantajosa para o indivíduo, pois desta forma ele não terá mais que memorizar números e muito menos se preocupar se está ou não portando os documentos necessários para dirigir dentro da lei, pois a fiscalização será possível apenas autenticando o próprio motorista.

Este trabalho monográfico tem por objetivo estudar o uso da biometria para autenticação remota em blitz de trânsito como uma solução para dar maior segurança na verificação da autenticidade de um cidadão, diminuindo o tempo de verificação, que hoje é feito via rádio. Para que possíveis crimes e criminosos sejam identificados e presos.

Para alcançar este objetivo foi escolhido começar o trabalho apresentando como é realizada a autenticação da atual Carteira Nacional de Habilitação, com os dados contidos nela, e também, demonstrando os riscos e vulnerabilidades. Em um segundo momento, buscou-se conceituar o que é a tecnologia biométrica e explicar alguns dos tipos de

autenticação possíveis. E por fim, demonstrar uma solução que cadastre o indivíduo com a sua impressão digital para que depois se faça a consulta de forma remota.

O primeiro capítulo apresenta a Carteira Nacional de Habilitação que é utilizada hoje, a sua evolução, os campos da Carteira de Motorista e suas vulnerabilidades.

Já o segundo capitulo aborda a biometria, alguns casos interessantes do uso da biometria, o futuro da biometria, os tipos de autenticação biométrica e o método escolhido.

O terceiro capítulo é uma explicação de como foi feito o sistema projetado, utilizando engenharia de software com base na modelagem UML.

O quarto capítulo fala sobre o banco de dados distribuído e porque foi adotado.

E por último, o quinto capítulo mostra a solução realizada para o problema, os dispositivos, sistema operacional, computadores e leitora utilizada para o desenvolvimento da solução. O capítulo também irá abordar as vantagens e desvantagens da solução proposta.

A principal motivação dessa monografia é a apreensão do carro do aluno por não portar o documento exigido, pois o havia esquecido em casa, por ter saído com pressa. Neste dia, além da apreensão do veículo, da multa por não portar o documento obrigatório, ainda teve que pagar outras taxas referentes ao depósito do carro e a vistoria para retirar o veículo do DETRAN.

Outro motivo para a construção da monografia é a falta de rigor e critérios nas fiscalizações rodoviárias e internas feitas no Brasil, pois alguns criminosos não vão presos porque a polícia não faz uma verificação adequada do documento ou não tem conhecimento sobre o criminoso. Em outras palavras, o policial somente faz uma simples conferência nos dados, olha a marca d’água e o talho doce na carteira de motorista.

Com esse novo sistema desenvolvido o cidadão parado numa BLITZ será obrigado a colocar o dedo no sensor para ver o seu cadastro e nele aparecerá se o cidadão em

questão está com a sua carteira de forma regular ou não ou se está sendo procurado pela polícia ou não. Além disso, o novo sistema poderá possibilitar ao governo integrar outras bases de dados à esse sistema. Como a base de eleitores que hoje em alguns lugares já utiliza a conferência pela digital.

1. Carteira Nacional de Habilitação – CNH

1.1. Conceituação

A Carteira Nacional de Habilitação é o documento oficial, que no Brasil, atesta a capacidade de um cidadão conduzir um veículo, sendo ela de porte obrigatório do condutor do veículo. A atual carteira possui foto e informações de diversos documentos do condutor e, ainda, serve como documento de identidade em todo Território Nacional.[1]

As primeiras carteiras de motoristas não possuíam foto e não serviam como documento válido para identificação pessoal, a validade dela dependia da apresentação em conjunto do documento de identidade. A fiscalização era similar ao que é realizado, ainda hoje, nos veículos náuticos onde os indivíduos habilitados devem apresentar a carteira de habilitação náutica (ARRAIS) junto com o documento pessoal identificador para que ela tenha validade. Na figura 1.1 podemos ver a diferença da CNH atual para a antiga que ainda está em circulação por alguns motoristas.

[1] Disponível em: https://denatran.serpro.gov.br/index2.htm. Acesso em: 1 nov.2008.

Figura 1.1 – Carteiras de Motorista de 1984 e Atual – Fonte: Detran

De acordo com o Código Nacional de Transito Brasileiro, Lei Nº 9.503, de 23 de setembro de 1997, somente o DENATRAN pode expedir a Carteira Nacional de Habilitação, conforme dispõe o seu artigo 19. Mas, é importante observar que, essa função na realidade é delegada aos DETRANs estaduais por conta da facilidade na fiscalização e na emissão da CNH.[2]

O candidato à Carteira de Motorista deve observar os seguintes requisitos para requerer a sua: saber ler e escrever, ter documento de identidade ou algum outro documento equivalente e ter mais de 18 anos. O candidato, que cumprir todos esses requisitos, deverá procurar uma auto-escola e agendar as aulas teóricas de direção defensiva, primeiros socorros e leis de trânsito. Passando por esta primeira exigência, o candidato deverá fazer no mínimo

[2] Disponível em: http://pt.wikipedia.org/wiki/Carteira_de_motorista/. Acesso em: 15 abr. 2008.

12 (doze) aulas práticas de direção e somente após o término dessas será possível agendar uma prova prática no DETRAN para obter a permissão que terá a validade de 1 (um) ano. Passado esse período, o condutor que não cometer nenhuma infração grave ou gravíssima, receberá automaticamente a CNH definitiva da categoria escolhida, já que existem 5 (cinco) diferentes tipos:

> • Categoria A - condutor de veículo motorizado de duas ou três rodas, com ou sem carro lateral;
>
> • Categoria B - condutor de veículo motorizado, não abrangido pela categoria A, cujo peso bruto total não exceda a três mil e quinhentos quilogramas e cuja lotação não exceda a oito lugares, excluído o do motorista;
>
> • Categoria C - condutor de veículo motorizado utilizado em transporte de carga, cujo peso bruto total exceda a três mil e quinhentos quilogramas;
>
> Para habilitar-se na categoria C, o condutor deverá estar habilitado no mínimo há um ano na categoria B e não ter cometido nenhuma infração grave ou gravíssima, ou ser reincidente em infrações médias, durante os últimos doze meses.
>
> • Categoria D - condutor de veículo motorizado utilizado no transporte de passageiros, cuja lotação exceda a oito lugares, excluído o do motorista;
>
> • Categoria E - condutor de combinação de veículos em que a unidade tratora se enquadre nas Categorias B, C ou D e cuja unidade acoplada, reboque, semi-reboque ou articulada, tenha seis mil quilogramas ou mais de peso bruto total, ou cuja lotação exceda a oito lugares, ou, ainda, seja enquadrado na categoria trailer.
>
> Segundo o artigo 145, para habilitar-se nas categorias D e E ou para conduzir veículo de transporte coletivo de passageiros, de escolares, de emergência ou de produto perigoso, o candidato deverá preencher os seguintes requisitos:
>
> 1 - Ser maior de vinte e um anos;
>
> 2 - Estar habilitado:
>
> 2.1 - No mínimo há dois anos na categoria B, ou no mínimo há um ano na categoria C, quando pretender habilitar-se na categoria D;
>
> 2.2 - E no mínimo há um ano na categoria C, quando pretender habilitar-se na categoria E;
>
> 3 - Não ter cometido nenhuma infração grave ou gravíssima ou ser reincidente em infrações médias durante os últimos doze meses;
>
> 4 - Ser aprovado em curso especializado e em curso de treinamento de prática veicular em situação de risco, nos termos da normatização do CONTRAN.[3]

[3] Disponível em: http://www.detran.df.gov.br/. Acesso em: 1 nov.2008.

1.2. Os campos da atual CNH

A Carteira de Motorista possui vários campos e marcas para torná-la mais confiável e, também, a fim de ser utilizada como um documento de identidade. Na figura 1.2 abaixo podemos identificar alguns dessas marcas e campos:

Figura 1.2: Carteira de Motorista Atual e seus campos – Fonte: DENATRAN

Na figura é possível identificar os campos Nome, CPF, Data de Nascimento, Filiação, Documento de Identidade com o órgão expedidor e a UF. Tais dados que fazem

da carteira de habilitação um documento de identidade, pois junto com a foto eles asseguram a veracidade da pessoa. Também existem campos específicos para uso do DETRAN, como o Permissão, Categoria, Validade, 1ª Habilitação, Número de Registro, Observações, Local e Data de Emissão.

Caso seja a primeira habilitação do sujeito aparecerá descrito PERMISSÃO, identificando que a carteira ainda não é definitiva é, apenas, uma permissão para dirigir durante 1 (um) ano. Passado esse tempo o condutor, caso não tenha cometido nenhuma infração grave ou gravíssima, receberá a carteira de habilitação definitiva.

Já o campo Categoria vai demonstrar o tipo da habilitação, ou seja, em qual categoria de veículo o sujeito estará habilitado a dirigir. No campo da validade fica explicito para o portador do documento a sua data de vencimento da carteira. No campo 1ª Habilitação mostra a data quando o condutor obteve a sua primeira habilitação, e nos campos local e data de emissão mostram o local e a data onde foi emitida, o que dá uma maior confiabilidade à carteira de motorista.

Um dos campos mais importantes para o DETRAN é o número de registro, pois ele contém o número registrado no banco de dados do DETRAN. No campo das observações é o local adequado para informar algum dado importante, como deficiência física, problemas auditivos ou visuais, etc. Existem também algumas marcas em Talho Doce, partes com fundo anti-scanner, marcas d’água e papel de segurança que servem para tentar evitar a clonagem e a falsificação.

1.2.1. Dados Sobre a CNH

Existem alguns dados disponibilizados pelo DETRAN do Distrito Federal que mostram a distribuição de Habilitações por idade, sexo e tipo de categoria, sendo que alguns desses dados podem ser observados nas tabelas abaixo.

Tabela 1.1: Quantitativo de Condutores por Sexo em dez/2005 – Fonte: DETRAN-DF

SISTEMÁTICA						
SEXO	SISTEMÁTICA ATUAL		SISTEMÁTICAANTERIOR		QTD TOTAL	
	Quantidade	Porcentagem	Quantidade	Porcentagem	Quantidade	Porcentagem
*	70	36,65%	121	63,35%	191	0,02%
FEMININO	322333	89,21%	38975	10,79%	361308	33,77%
MASCULINO	592507	83,65%	115793	16,35%	708300	66,21%
TOTAIS	914910	85,52%	154889	14,48%	1069799	100%

(*) Registro com informação incorreta e em depuração

Como é possível ver na tabela 1.1 até dezembro de 2005 no Distrito Federal o número de condutores homens era quase o dobro do número de condutores do sexo feminino.

Já na tabela 1.2, abaixo, podemos ver a distribuição dos condutores nos diversos tipos de categorias de Habilitação no Distrito Federal em dezembro de 2005, o que nos comprova que a categoria B, de veículos leves, é a que contém o maior número de registro de habilitações.

Tabela 1.2: Quantitativo de Condutores por Categoria dez/2005– Fonte DETRAN-DF

SISTEMÁTICA						
FAIXA ETÁRIA	QTD ATUAL		QTD ANTERIOR		QTD TOTAL	
	Quantidade	Porcentagem	Quantidade	Porcentagem	Quantidade	Porcentagem
*	23	69,70%	10	30,30%	33	0,00%
A	4943	88,24%	659	11,76%	5602	0,52%
B	656375	85,85%	108223	14,15%	764598	71,47%
C	12202	61,95%	7495	38,05%	19697	1,84%
D	89845	75,72%	28816	24,28%	118661	11,09%
E	7320	99,95%	4	0,05%	7324	0,68%
AB	94974	93,68%	6411	6,32%	101385	9,48%
AC	5815	90,85%	586	9,15%	6401	0,60%
AD	38778	93,53%	2683	6,47%	41461	3,88%
AE	4635	99,96%	2	0,04%	4637	0,43%
TOTAIS	914910	85,52%	154889	14,48%	1069799	100%

(*) Registro com informação incorreta e em depuração

Na tabela 1.3 podemos observar a distribuição quanto à faixa etária comprovando que o número de permissões emitidas na faixa de idade de 18 a 19 anos na sua maioria compõe apenas 1,88% do total. Também é possível comprovar que os condutores acima de 60

anos são 12,08% do total de motoristas e a faixa que compreende o maior número de condutores é a de 25 a 29 anos.

Tabela 1.3: Quantitativo de Condutores por Faixa Etária dez/2005 - Fonte: DETRAN

SISTEMÁTICA						
FAIXA ETÁRIA	QTD SISTEMÁTICA ATUAL		QTD SIST. ANTERIOR		QTD TOTAL	
	Quantidade	Porcentagem	Quantidade	Porcentagem	Quantidade	Porcentagem
*	3071	50,14%	3054	49,86%	6125	0,57%
18 A 19	20153	100,00%	0	0,00%	20153	1,88%
20 A 24	120603	99,99%	7	0,01%	120610	11,27%
25 A 29	150830	99,86%	207	0,14%	15037	14,12%
30 A 34	137747	97,75%	3175	2,25%	140922	13,17%
35 A 39	110558	86,94%	16603	13,06%	127161	11,89%
40 A 44	105056	90,32%	11254	9,68%	116310	10,87%
45 A 49	84047	83,32%	16824	16,68%	100871	9,43%
50 A 54	67220	75,86%	21389	24,14%	88609	8,28%
55 A 59	47550	69,21%	21156	30,79%	68706	6,42%
60 A 64	29784	63,43%	17174	36,57%	46958	4,39%
65 A 69	20579	57,70%	15089	42,30%	35668	3,33%
70 ou mais	17712	37,95%	28957	62,05%	46669	4,36%
TOTAIS	914910	85,52%	154889	14,48%	1069799	100%

(*) Registro com informação incorreta e em depuração

1.3. A vulnerabilidade da CNH

Segundo Jairo Mota Castro, gerente do Renach - DENATRAN, hoje em dia a autenticação da Carteira Nacional de Habilitação é feita por uma simples conferência visual, ou seja, o policial de trânsito apenas olha a carteira e confere a foto do motorista, verificando a veracidade da carteira, não conferindo alguns campos de verificação de autenticidade, esses campos são: as marcas d'aguas, os símbolos em alto relevo, os códigos e os números de registros.[4] A não realização da verificação correta facilita alguns falsificadores, que acabam pegando uma carteira de motorista roubada ou achada, no meio da rua, "scanneiam" a carteira com outra foto em cima e depois imprimem com um papel especial, parecido com o original.

[4] Jairo Mota Castro – Gerente do Renach – DENATRAN

Isso é uma forma de se clonar a carteira de motorista, mas podem também falsificar alterando nome e outros dados. Por isso e outras farsas que todos os métodos teriam que ser verificados pelos polícias, mas isso não acontece por que alguns policiais nem sabem todos os esses métodos de seguranças existentes na CNH.

Hoje em dia a autenticação pode ser feita através de uma conferência via rádio com a central a fim de conferir alguns dados, mas é importante observar que nesta conferência não é possível se conferir a foto. Por isso, alguns novos sistemas que estão em teste já estão dotados de um sistema portátil para que se possa fazer a verificação das informações. No entanto, a autenticação ainda continua sendo feita em cima da Carteira Nacional de Habilitação que pode ser passível de certas vulnerabilidades e pode ser molhada, rasgada, perdida, queimada e outros.

2. A Biometria e os tipos de autenticação viável para o projeto

2.1. A Biometria

A biometria é a ciência que estuda as formas de identificação dos seres humanos pelas partes do corpo. Essa tecnologia, depois de anos de testes laboratoriais, conquistou o mercado, e já é uma realidade, inclusive no Brasil.[5]

O processo de autenticação biométrica de uma forma geral é feita da mesma forma não importando o método. O processo envolvido nessa autenticação é simples, primeiro deve ser feito um cadastramento do usuário, onde um arquivo é gravado de forma criptografada com as principais informações biométricas. Num segundo momento quando o usuário se conectar ao sistema, o modelo é comparado com a nova informação obtida. Se as informações forem iguais, o acesso é liberado. Como pode ser visto na figura 2.1 abaixo.

Figura 2.1 – Método de autenticação – Fonte PC Magazine Brasil

[5] GUNNERSON, Gary. Pronto para a Biometria?: PC Magazine Brasil, São Paulo, v. 9, n. 3, p. 82, mar. 1999.

> *Como não poderia deixar de ser, toda essa tecnologia também é passível a erros. Nenhuma das técnicas biométricas disponíveis hoje é infalível. O reconhecimento de voz é o mais arriscado, podendo falhar uma vez a cada 3000 identificações, segundo dados da Sociedade Internacional de Biometria (www.tibs.org). Os motivos são vários, desde ruídos no ambiente até rouquidão. No caso da impressão digital, a proporção cai um pouco, passando para 1 em 10000, mas ainda existe. Mesmo assim as chances de erro são desprezíveis em comparação à possibilidade de esquecermos uma senha armazenada na memória.*
>
> *O grande risco dessas técnicas é de não reconhecer um usuário legítimo ou aceitar um farsante. É raríssimo isso acontecer, mas é possível. São dois os fatores principais que possibilitam os erros. O primeiro é o fato de que todas as características do corpo humano são transformadas em códigos matemáticos, que em tese podem ser decifrados se não houver criptografia auxiliar, principalmente nas aplicações de comércio eletrônico. O outro fator é que esses algoritmos requerem um mínimo de espaço para serem armazenados. Em bom português: quanto mais leves mais fáceis de decodificar, mas rápido será o processamento da identificação.*[6]

As metodologias biométricas podem ser avaliadas através de diversos parâmetros, nomeadamente, o grau de fiabilidade, aceitação, nível de aceitação e custo de implementação. O grau de fiabilidade pode ser aferido através dos valores FAR (False Acceptance Rate – Taxa de Falsas Aceitações) e o FRR (False Rejection Rate – Taxa de Falsas Rejeições). Estas variáveis são mutuamente dependentes, não sendo possível minimizar ambas num mesmo algoritmo. Assim, normalmente, procura-se o ponto de equilíbrio (figura 2.2) a que chamamos CER (Crossover Error Rate – Taxa de Intersecção de Erros). Quanto mais baixo for o CER mais preciso é um sistema biométrico.[7]

[6] VIEIRA, Eduardo. Você e a senha. Info Exame , São Paulo, v.16, n. 178, p 79-83, jan. 2001.
[7] Disponível em: http://www.via6.com/topico.php?tid=109871. Acesso em: 17 ago.2008.

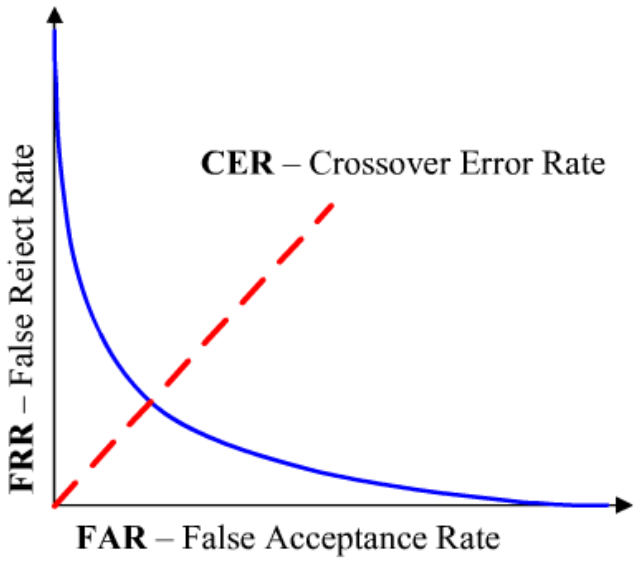

Figura 2.2: CER Crossover Error Rate – Fonte: (Liu, 2001)

A figura 2.3 abaixo apresenta-nos uma visão geral sobre a estrutura básica de reconhecimento biométrico.

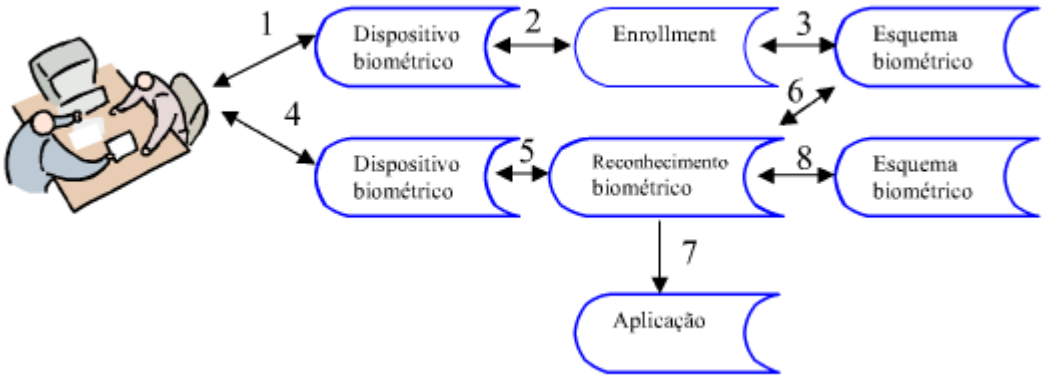

Figura 2.3: Processo de reconhecimento biométrico – Fonte (Liu, 2001)

O funcionamento de reconhecimento biométrico inicia-se com (1) captura da biometria em causa; (2) processa a biometria, extraindo a informação considerada relevante e inicia o processo de construção do esquema biométrico; (3) guarda o esquema biométrico num repositório central ou num sistema portátil como por exemplo um cartão; (4) captura da biometria em causa, mas no sentido de utilização; (5) processa a biometria extraindo o

esquema; (6) compara o esquema da biometria adquirida com o esquema biométrico existente; (7) fornece o resultado da comparação para uma aplicação cliente; (8) grava um registro da utilização.[8]

2.1.2. Histórico da Biometria

Segundo pesquisado no artigo da Folha UOL, a biometria existe a pelo menos um milênio e segundo relato do Ricardo Yagi, especialista de empresa focada em aparelhos biométricos ID-Tech:

> Na dinastia Tang (800 D.C.), na China, impressões digitais eram grafadas no barro para confirmar a identidade do indivíduo em transações comerciais.
>
> Em 1686, na Espanha, o professor de anatomia Marcelo Malpighi pesquisou com detalhes as linhas, curvas e espirais da impressão digital para que, em 1892, Francis Galton, um antropólogo inglês, publicasse a primeira classificação dos tipos de impressão digital utilizados até hoje.
>
> Yagi conta ainda que a impressão digital em tinta é usada para reconhecimento civil e criminal há ao menos cem anos. O FBI (Polícia Federal Americana) controla mais de 200 milhões de impressões digitais em seus bancos de dados há cerca de 30 anos.[9]

2.1.3. O futuro da Biometria

O futuro da biometria está nos estudos de vários pesquisadores e desenvolvedores, que buscando uma maior segurança e privacidade estão desenvolvendo novas técnicas de autenticação dependendo da sua viabilidade de uso em situações práticas e da análise de suas aplicações. No futuro, os novos meios de autenticação serão odores, DNA, salinidade do corpo, formato da orelha, e padrões das ondas cerebrais. Uma pesquisa feita

[8]Disponível em: http://www.via6.com/topico.php?tid=109871. Acesso em: 17 ago.2008.
[9] Disponível em: http://www1.folha.uol.com.br/folha/informatica/ult124u21496.shtml. Acesso em 18 set.2008.

pelo IDG Now! ao qual ouviu especialistas e listou tecnologias que podem ser empregadas no futuro, deixando os atuais sistemas obsoletos.

> Odor – Por mais que suscite centenas de piadinhas, a identificação pelo odor não tem relação nenhuma com um perfume usado ou a quantidade de suor impregnada no corpo humano.
> O sistema desenvolvido pela empresa Mastiff Electronic Systems, e ainda em testes, usa um sensor eletrônico extremamente sensível que imita o sentido do olfato humano para detectar partículas de odor liberadas por cada indivíduo. De tão sensível que é, o sistema, que ganhou o nome temporário de "Scentinel", cheira a palma da mão do usuário e "ignora" outras fragrâncias que não a detectada como a expelida pelo corpo humano.
>
> Arquitetura da orelha – Muitos sistemas usados atualmente em academias e clubes registram o formato da mão dos usuários.
>
> O sistema de arquitetura de orelha é similar, mas conta com uma grande vantagem: em vez de tocar no sensor, o sistema registra à distância particularidades do membro, como lóbulo e as formações dentro da concha auricular.
>
> Além de ser pouco intrusivo ao corpo humano, como é a tecnologia de reconhecimento de íris, por exemplo, a biometria por reconhecimento de orelha apresenta como vantagem a baixa mudança do membro no decorrer da vida do usuário.
>
> Comparação de DNA – Bem explorada no filme "Gattaca", a comparação entre amostras de DNA está no meio de uma acalorada discussão sobre sua validade como tecnologia biométrica.
>
> O uso de DNA como amostra biométrica é apontado como um sistema completamente seguro, já que seria praticamente impossível fraudar material genético humano.
>
> Críticos da tecnologia, no entanto, alegam que a obtenção de DNA seria um processo bastante intrusivo, com o usuário obrigado a ser tocado, e que não se basearia numa reprodução humana, mas sim em uma amostra
>
> Métodos de identificação pelo DNA, porém, contam com a desvantagem da agilidade – material genético humano leva atualmente meio hora, no mínimo, para que seja "decifrado" a ponto de poder se comparado com outras amostras.
>
> Ondas cerebrais – Mais do que qualquer outra parte do corpo, o cérebro humano carrega características individuais que dificilmente batem com as de outros indivíduos. Por que não, então, explorar essas particularidades?
>
> Os especialistas apostam em sistemas biométricos que medem ondas eletromagnéticas emitidas pelo cérebro humano.
>
> Em vez de se aproximar ou encostar-se no sensor, a tecnologia conseguiria medir os pulsos provenientes do cérebro sem que o usuário movesse um músculo.
>
> Mesmo que ainda não existam sistemas do tipo em desenvolvimento, pesquisas sobre o funcionamento elétrico do cérebro apontam para o reconhecimento biométrico pelo padrão das ondas geradas. [10]

[10] Disponível em: http://idgnow.uol.com.br/seguranca/2006/08/30/idgnoticia.2006-08-29.9472410830/. Acesso em 23/10/2008

2.1.4. Casos interessantes de uso da Biometria

A revista PC World, especializada em informática, traz alguns casos interessantes de uso da Biometria, como estes:

- O congresso Nacional brasileiro implantou um sistema de identificação pela impressão digital, para registrar a freqüência e a autenticidade dos deputados na votação.
- Nos jogos olímpicos de 1996, a geometria das mãos foi um dos critérios de segurança, usados para identificar os atletas que participaram das provas. Neste ano, em Sydney, a estratégia se repetirá e todos os 42 mil atletas, técnicos, funcionários, patrocinadores e pessoal de imprensa terão diferentes níveis de acesso.
- Nas últimas eleições presidenciais no México, a tecnologia de reconhecimento facial foi utilizada para impedir a duplicidade de votos de um mesmo eleitor.
- O departamento de imigração e naturalização dos Estados Unidos usa biometria facial e reconhecimento de voz para tornar mais rápido e eficiente o trânsito regular de cidadãos que trabalham na fronteira com o México.
- O Banco United of Texas foi a primeira unidade financeira dos Estados Unidos que implementou o reconhecimento de íris em seus caixas eletrônicos.
- A CIA, FBI e NASA exigem altíssimo nível de segurança para acesso a certas salas, usam a identificação pela retina.
- Proprietários de passaporte anuais e sazonais para acesso a Disney World, em Orlando, são checados pela impressão digital, para terem acesso ao parque. Os dizeres "pessoal e intransferível" finalmente podem ser efetivos.
- O presídio de Pentoville, na Inglaterra, adotou a identificação da assinatura para evitar que um preso se fizesse passar por outro na entrada de comida.
- Funcionários de um hospital em Chicago ganham acesso ao berçário por meio de reconhecimento de voz.
- Os aeroportos Charlote/Douglas, nos EUA, e Flughafen, na Alemanha, usam o reconhecimento da íris no embarque de passageiros. [11]

2.2. Os Tipos de Autenticação Biométrica

A identificação biométrica pode ser feita por características físicas ou comportamentais, variando o grau de complexidade da análise. Características físicas incluem impressão digital, reconhecimento da face, identificação da íris, identificação da retina, geometria da mão. Já as características comportamentais, fazem o reconhecimento da voz, reconhecimento da escrita ou assinatura.

[11] GOYA, Denise Hideko. Biometria: Esqueça todas as senhas. São Paulo, PC World, n. 98, p 58, ago.2000.

2.2.1. Autenticação pela Face

Estes sistemas não são intrusivos e oferece recursos extras de segurança como o registro de uma imagem de cada usuário, que se aproxima da máquina. Alguns exigem câmeras relativamente avançadas, o que aumenta o preço de comercialização ou utilizam várias fotos para comparação. Mesmo assim pode-se verificar que nem sempre a segurança é total.[12]

Figura 2.4: Programa de reconhecimento de face – Fonte PC Magazine Brasil

Identificar um indivíduo através da análise da face é um processo complexo que normalmente requer artifícios inteligentes sofisticados e técnicas de aprendizagem computacional (machine learning techniques). Uma quantidade de fornecedores biométricos está envolvida na venda desses sistemas, usando tanto vídeos padrões como imagens termais para capturar imagens faciais. A face é um componente chave da maneira como os seres humanos se lembram e reconhecem uns aos outros. A inteligência artificial é necessária para simular a interpretação humana das faces. As pessoas mudam todo o tempo. Pêlos faciais, óculos e a posição da cabeça podem afetar a forma como um sistema biométrico pode comparar uma face com a outra. A aprendizagem computacional é importante para a

[12] GUNNERSON, Gary. Pronto para a Biometria?: PC Magazine Brasil, São Paulo, v. 9, n. 3, p. 83, mar. 1999.

adaptação a essas mudanças e para comparar precisamente os novos exemplos com as *templates* previamente armazenados.[13]

2.2.2. Autenticação pela Retina

A retina é a parte posterior do olho formada em sua maior parte por células nervosas. A retina é o local onde se formam a imagem. Assim como a íris, a retina forma um padrão único e começa a se desintegrar logo após a morte. As biometrias de retina são geralmente tidas como o método biométrico mais seguro. O acesso não-autorizado em um sistema de retina é virtualmente impossível. Um procedimento preciso de cadastramento é necessário, o que envolve o alinhamento da vista para alcançar uma leitura otimizada. [14]

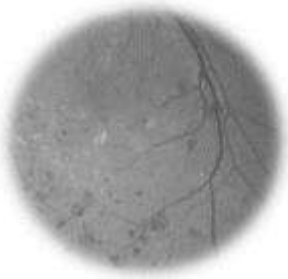

Figura 2.5: Análise da Retina – Fonte Infowester[15]

O processo de leitura da retina é muito parecido com o da íris e engloba os mesmos 3 (três) passos: Captura onde é feita a uma distancia de até 3 polegadas de um leitor ocular, onde o usuário deverá olhar fixamente para uma luz que vai aparecer atrás do leitor por alguns segundo. Na extração a única diferença é que o equipamento biométrico mapeia a posição das veias sangüíneas e as transforma em uma representação matemática única e

[13] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em 03 mar.2008.
[14] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em: 03 mar.2008.
[15] Disponível em: http://www.infowester.com/biometria.php. Acesso em: 18 set.2008.

depois é armazenada como uma *template*. Já a fase de Comparação é feita da mesma forma que na íris.[16]

2.2.3. Autenticação pelo Posicionamento das Veias

No método de comparação das veias é feita através de uma comparação do padrão das veias da pessoa que deseja o acesso com o cadastrado no banco de dados. Este método tem uma confiabilidade enorme, porém não é muito utilizado devido custo elevado de implementação. Esse sistema é um sistema relativamente novo e como qualquer sistema tem suas vantagens e desvantagens. Para a comparação não é preciso encostar-se ao leitor o que evita de manchas e riscos e arranhados na leitora, cadáveres e corpos mutilados não dão acesso porque para ter acesso o sangue precisa estar fluindo. Mas em compensação podemos ter alguns erros devido ao calor do corpo já alguns capilares são muito sensíveis ao calor.

Figura 2.6: Veias da Palma da mão e dispositivo de leitura – Fonte: Fujitsu[17]

[16] Disponível em: http://www.consultoresbiometricos.com.br/05_Dbio_olho.php. Acesso em: 20 set.2008.
[17] Disponível em: http://www.fujitsu.com/pt/news/pr/20050512-01.html. Acesso em: 29/08/2008

2.2.4. Autenticação pela Íris

A íris é a parte colorida que fica ao redor da pupila do olho. Cada íris possui uma estrutura única, caracterizando um padrão complexo. Pode ser uma combinação de características como coroa, glândula, filamentos, sardas, sulcos radiais e estriamentos. É conhecido que uma duplicação artificial da íris é virtualmente impossível devido às suas propriedades únicas. A íris é estreitamente ligada ao cérebro humano e dizem ser uma das primeiras partes a se desintegrar após a morte. É, portanto muito improvável que uma íris artificial possa ser recriada ou que uma íris morta possa ser usada para fraudar a passagem no sistema biométrico.[18]

Figura 2.7: Identificação por leitura da Íris – Fonte Infowester[19]

O processo de reconhecimento da íris começa na captura da imagem da mesma por uma câmera de vídeo, o que deve ser feito em um ambiente com boa luminosidade. As lentes de contato não interferem na captura da imagem, entretanto óculos não podem ser

[18] Disponível em: http://www.consultoresbiometricos.com.br/05_Dbio_olho.php. Acesso em: 20 set.2008.
[19] Disponível em: http://www.infowester.com/biometria.php. Acesso em: 18 set.2008.

utilizados. Depois da captura da imagem começa o processo de extração, onde o equipamento biométrico extrai as características únicas da íris da imagem já capturada. Então essas características são convertidas em um código matemático único e armazenadas como um *template*. A ultima fase do reconhecimento é a comparação onde existe a comparação das características armazenadas para as da *template*.

2.2.5. Autenticação pela Geometria da Mão

Este é um método de identificação mais antigo, porém não é preciso. Este método é também um dos mais simples e rápidos, por isso ele é utilizado em locais de grande fluxo de pessoas. Na leitura a pessoa vai ter que colocar sempre a mesma mão para a identificação, porque o método consiste na medição da geometria da mão e a troca da mão pode gerar erro devido alguma diferença de medidas. Mas para isso existem pinos que indicam a posição de cada dedo no sensor de leitura.

A figura 2.8 abaixo mostra um dispositivo que faz identificação por meio de geometria de mão. Seu funcionamento é simples: o indivíduo digita um número único (número de funcionário, número de matrícula ou qualquer outro) e, em seguida, posiciona sua mão em um painel. Este possui pinos que indicam onde cada dedo deve ficar posicionado. Com isso, a posição da mão sempre vai ser a mesma e assim o aparelho consegue medir sua geometria e comparar com os dados gravados em seu banco de dados. Esse tipo de aparelho pode ser aplicado, por exemplo, em catracas e no controle de abertura de portas. Alguns dispositivos aceitam o uso de cartões (como crachás) ao invés da digitação de números, o que

tem como vantagem a possibilidade do usuário não ter que decorar uma combinação, e como desvantagem o risco de perda do cartão.[20]

Figura 2.8: Identificação por geometria da mão – Fonte: Infowester[21]

2.2.6. Autenticação pela voz

Autenticação por reconhecimento de voz refere-se ao processo de aceitação ou não da identidade de um indivíduo, sendo a informação para a autenticação, recolhida com recurso de um microfone e representada digitalmente a onda de som propagada. [Anil Jain 2000]. Esse processo tem recebido bastante atenção nas últimas décadas, pela sua conveniência, aceitação, simplicidade de substituição de autenticação normal de palavra-chave.

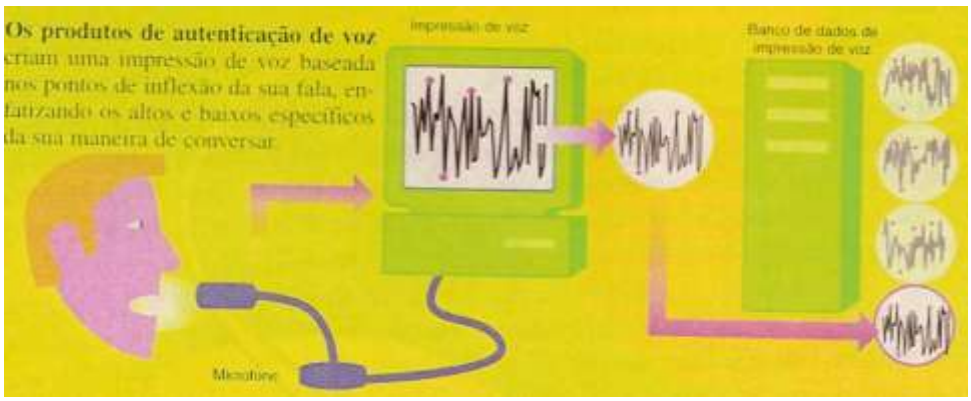

Foto 2.9: Autenticação por voz – Fonte PC Magazine Brasil

[20] Disponível em: http://www.infowester.com/biometria.php. Acesso em: 18 set.2008.
[21] Disponível em: http://www.infowester.com/biometria.php. Acesso em: 18 set.2008.

O potencial deste processo é bastante elevado devido ao baixo custo do hardware necessário, o qual está já presente em grande parte dos computadores existentes: um microfone. No entanto, a sua aplicação está limitada, atualmente, a aplicações com um baixo nível de segurança, em virtude das grandes variações na voz de um indivíduo e na baixa precisão dos atuais sistemas de autenticação por reconhecimento de voz.[22]

Talvez, seja o sistema de implementação mais barata, ele não exige hardware especial, e os centralizados são ideais para conceder acesso remoto a usuários móveis. Todavia os sistemas mais robusto podem apresentar um preço bastante alto.[23]

2.2.7. Autenticação pela Impressão Digital

Impressão Digital são pequenos sulcos na pele dos dedos, que são formados pelas elevações na pele. A impressão digital é usada a mais de mil anos para identificar pessoas, pois hoje, sabemos que as impressões são únicas, ou seja, ninguém tem uma digital igual a de outra pessoa. Até mesmo em caso de gêmeos univitelinos existe diferença na digital.

As elevações são formadas no feto e acompanham a pessoa pela vida toda, sem apresentar grandes mudanças. A impressão digital apresenta pontos característicos e formações que permitem a um perito (papiloscopista) identificar uma pessoa de forma confiável.[24]

---

[22] Disponível em: http://www.via6.com/topico.php?tid=109871. Acesso em: 17 ago.2008.
[23] GUNNERSON, Gary. Pronto para a Biometria?: PC Magazine Brasil, São Paulo, v. 9, n. 3, p. 86, mar. 1999.
[24] Disponível em: http://pt.wikipedia.org/wiki/Impressao_digital/. Acesso em: 15 abr.2008.

Foto 2.10: Impressão Digital – Fonte: Google Images[25]

O sistema biométrico de digitais são conhecidos por serem precisos no método de identificação e verificação. Os sistemas de digitais um-para muitos (1:n) e um-para-um (1:1), em sua maioria, analisam únicos e pequenos atributos na imagem do desenho da digital, conhecidos como minúcias. Elas podem ser definidas como os contornos das linhas papilares ou bifurcações (ramificações das linhas papilares).

Diversos outros métodos de impressões digitais verificam os pequenos poros dos dedos, que como as minúcias, são colocadas para diferenciar um indivíduo do outro de forma particular e única. Dessa forma, a distância entre as linhas papilares ou a densidade da imagem digital também podem ser analisadas.[26]

Algumas condições podem dificultar a coleta das impressões digitais dos indivíduos, pois elas podem acabar diminuindo a qualidade da captura das imagens, como por exemplo, no caso da sujeira, dos dedos secos ou rachados. Outros casos que também podem atrapalhar na qualidade das imagens digitais são a idade, o sexo e a etnia. É importante também observar que dependendo da forma que o indivíduo interage com o scanner que recolhe as digitais, pois caso a pressão feita na superfície do scanner seja muito forte, por exemplo, a imagem pode ficar distorcida. Já existe uma preocupação e um trabalho dos

[25] Disponível em: http://www.kimaldi.com/var/kimaldi/storage/images/media/images/biometria_1/50481-1-esl-ES/biometria_1.jpg Acesso em: 29/07/2008

[26] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em: 03 mar.2008.

fornecedores para solucionar essas dificuldades, assim alguns scanners estão sendo ergonomicamente desenhados para melhorar o sistema de captura das impressões digitais.[27]

A digital é uma das formas de identificação biométrica mais utilizada na autenticação, pois consiste em capturar a formação de sulcos na pele dos dedos e das palmas das mãos, afinal esses sulcos são formados por certas terminações e divisões que são completamente diferentes em cada pessoa.

Basicamente, existem três tipos de tecnologia para esse tipo de identificação, são elas: a óptica, que se utiliza de um feixe de luz para ler a impressão digital; a capacitiva, que é aquela que mede a temperatura exala da impressão; e a ultra-sônica, que mapeia a impressão digital por meio dos sinais sonoros. Exemplificando, é possível apontar para a identificação por impressão digital e seu uso em catracas, onde o usuário deve colocar seu dedo em cima de um leitor que confirmando a identificação, liberará o acesso.[28]

2.2.7.1. Verificação da imagem digital

Primeiramente, é necessário mostrar como é realizado o processo de verificação um-para-um (1:1):

> **- Captura:** A técnica de imagem óptica normalmente envolve a geração de uma fonte de luz, a qual é refracionada através de um prisma, em cuja superfície de vidro o dedo é colocado A luz brilha na ponta do dedo e a impressão é feita pela imagem do dedo que é capturada.[29]

Técnicas táteis ou termais usam tecnologia de chip de silicone, sofisticada, para conseguir os dados da imagem digital, assim o usuário posiciona um dos dedos no sensor que será ativado com o calor ou a pressão do dedo, capturando então os dados. Esses sensores de

[27] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em: 03 mar.2008.
[28] Disponível em: http://www.infowester.com/biometria.php. Acesso em: 18 set.2008.
[29] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em: 03 mar.2008.

captação de silicone medem as cargas elétricas e enviando um sinal elétrico quando o dedo é colocado em sua superfície.

Assim como métodos táteis e termais, o elemento chave da técnica de captação é o sensor, que é utilizado na captação das mínimas elevações e aprofundamentos das linhas papilares, enviando um sinal elétrico, e assim os vales das pontas dos dedos são analisados. Mas é importante salientar que nenhum sinal é gerado pelos vales e que essas variações na carga elétrica produzem a imagem digital.

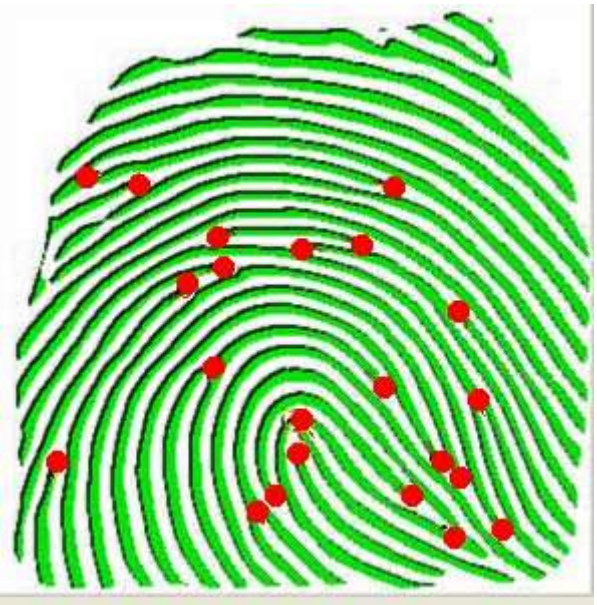

Foto 2.11: Retirando os pontos importantes da digital – Fonte:

Já na captura de imagem via ultra-som as ondas de som abaixo são utilizadas, obedecendo ao limite da audição humano. É realizado da seguinte forma: coloca-se um dedo no scanner e as ondas acústicas são utilizadas para medir a densidade do padrão da imagem digital.

- **Extração:** O equipamento biométrico extrai os dados contidos na imagem digital. Uma representação matemática única é então armazenada na forma de um template.
- **Comparação:** Durante o processo de comparação, um novo exemplo é comparado com o template.

Dependendo da base que está configurada para a aplicação, pode-se obter um par como resultado ou não. [30]

2.2.7.2. Identificação da Imagem Digital

Agora, passa-se para a análise do processo de identificação um-para-muitos que segue os seguintes passos para a capturação e extração da imagem digital:

Captura: para identificações padrões um-para-muitos (1:n), os indivíduos são cadastrados usando um processo de captura óptica em tempo real como o descrito acima na verificação da imagem digital. Sistemas AFIS de Forças Policiais, também conhecidos como estações de cadastramento, capturam as imagens de todos os dez dedos. Um AFIS civil, entretanto, não precisa capturar todas as imagens e pode operar efetivamente utilizando uma ou duas. Impressões latentes, tomadas de uma cena de um crime, ou imagens com tinta em um papel, também podem ser capturadas pelo AFIS utilizando-se um scanner rolado (flatbed scanner).

Extração: Para um AFIS (Automatic Fingerprint Identification Sistem), o processo de binning (processo de redução de erros e ruidos) das imagens digitais refina o processo de extração. Dados das minúcias são extraídos e armazenados na forma de um template no banco de dados.

- Comparação: Um novo exemplo, capturado tanto com técnicas em tempo real, quanto latente ou em papel, é comparado com um banco de dados de imagens digitais. Se um AFIS está sendo utilizado e o binning também, a comparação será com o quadrado que está com as características similares às da nova imagem apresentada.

É possível, até, obter um par como resultado ou não, para isso, apenas depende da base que foi configurada na aplicação.

[30] Disponível em: http://www.consultoresbiometricos.com.br. Acesso em: 03 mar.2008.

Tabela 2.1: Comparação de Biometria – Fonte Liu 2001

Características	Impressão Digital	Geometria da mão	Retina	Íris	Face	Assinatura	Voz
Facilidade de uso	Alta	Alta	Baixa	Média	Média	Alta	Alta
Incidência de erro	Sujidade, idade	Feridas na mão, idade	Óculos	Pouca luz	Luz, idade, óculos, cabelo	Alteração de assinaturas	Ruídos, meteorologia
Eficácia	Alta	Alta	Muito alta	Muito alta	Alta	Alta	Alta
Custo	*	*	*	*	*	*	*
Aceitação	Média	Média	Média	Média	Média	Muito alta	Alta
Nível de exigência pedido	Alto	Médio	Alto	Muito alto	Médio	Médio	Médio
Estabilidade a prazo	Alta	Média	Alta	Alta	Média	Média	Média
* - Não incluído dada a existência de demasiados factores que influenciam os custos							

3. Engenharia de Software com Modelagem UML

Engenharia de software é uma disciplina de engenharia voltada para todos os aspectos da produção de um software de qualidade. Envolve a escolha de métodos, ferramentas adequadas para o contexto do sistema, as especificações iniciais do sistema, sua manutenção e operação. É na engenharia de software que são definidos os processos, modelos, metodologias de desenvolvimento e gerência de projeto.

Mas por que é importante definir uma modelagem? É importante porque um modelo é uma exemplificação da realidade. Elaboramos modelos para entender melhor o sistema que estamos desenvolvendo. Com os modelos além de documentar as decisões tomadas, proporcionam um guia da construção do sistema. Ou seja, em sistemas muito complexos são construídos modelos para poder compreender o sistema na sua totalidade.

A escolha dos modelos que devem ser criados tem muita interferência sobre como um determinado problema é solucionado. Pois cada modelo pode ser desenvolvido em diversos níveis de exatidão. Os melhores modelos são aqueles que são associados à realidade. Mas um modelo único não é suficiente, para sistemas que não sejam triviais, ou seja, qualquer sistema será melhor analisado por um conjunto de modelos.

A UML é uma linguagem de modelagem que foi criada para acabar com o desentendimento e confusão dos analistas e desenvolvedores, por conta das suas diferenças de notação e conceituação.

Para resolver esse problema, Grady Booch, James Rumbaugh e Ivar Jacobson se reuniram e criaram uma metodologia unificada (UML) com o melhor de cada modelagem já existente: DER (Diagrama de Entidade e Relacionamento), WorkFlow (Modelagem de

Negócio), Modelagem de Objetos e Componentes. Assim foi criada a Unified Modeling Language a UML. A UML pode ser usada durante todas as fases do projeto.

Essas fases do projeto são dividas em Análise de requisitos, Análise geral, Design, Programação, Testes. A fase de Análise de requisitos é a que engloba a captura das idéias e necessidades dos usuários do sistema com as funções necessárias no sistema,. Na fase de análise geral é quando são criados os primeiros esboços de mecanismos presentes no "problema". Pois o resultado dessa fase é ampliado em soluções técnicas. Assim chamamos essa fase seguinte de Design. Onde também é feito um detalhamento para a Programação. Na Programação é onde os modelos criados são transformados em códigos de linguagem de programação. Após isso são feitos testes para ver se o sistema é seguro, aceitável e integrável concluindo os testes.

Mas por que utilizar UML? Usa-se UML porque a modelagem UML segue um padrão muito rigoroso e assim acabou se tornando uma notação padrão das indústrias de software.

A UML está divida em:

- Visões – Mostram os diferentes aspectos do sistema, dando ênfase aos diferentes ângulos e níveis de abstração, para construir uma visão completa do sistema a ser construído.
- Modelos de Elementos – São os conceitos que são utilizados para a elaboração dos diagramas que representam as definições comuns da Orientado à Objeto.
- Mecanismos Gerais – Provem comentários, informações extras sobre os elementos dos modelos.

- Diagramas – São os gráficos que descrevem o conteúdo em uma visão. A UML possui diferentes tipos de diagramas como, caso de uso, classe, atividade, estado etc que combinados, nos proporcionam todas as visões do sistema.

3.1. Diagramas da UML

Com a UML é possível modelar os projetos de sistemas, utilizando os diagramas. Os diagramas são divididos, basicamente, em:

- Diagramas Estruturais – diagramas de classes, diagrama de objetos, diagrama de componentes e diagrama de disponibilização.
- Diagramas de Comportamento – diagrama de caso de uso, diagrama de seqüência, diagrama de atividades, diagrama de colaboração e diagrama de estados.
- Diagramas de Gerenciamento do Modelo – diagrama de pacotes, diagrama de subsistemas e diagrama de modelos.

3.1.1. Diagrama de Caso de Uso

O diagrama de caso de uso é usado para visualizar os relacionamentos externos ao sistema, ou seja, as suas relações com o mundo exterior. Nesse diagrama não é necessário entender como o sistema implementa os casos de uso ou como ocorre o funcionamento, pois

os propósitos desse diagrama é oferecer possíveis situações reais onde o sistema possa ser utilizado, fornecer uma idéia clara sobre o que o sistema vai atender e o que deve ser implementado pelo sistema. Os elementos existentes em um caso de uso são: atores, interação e domínio.

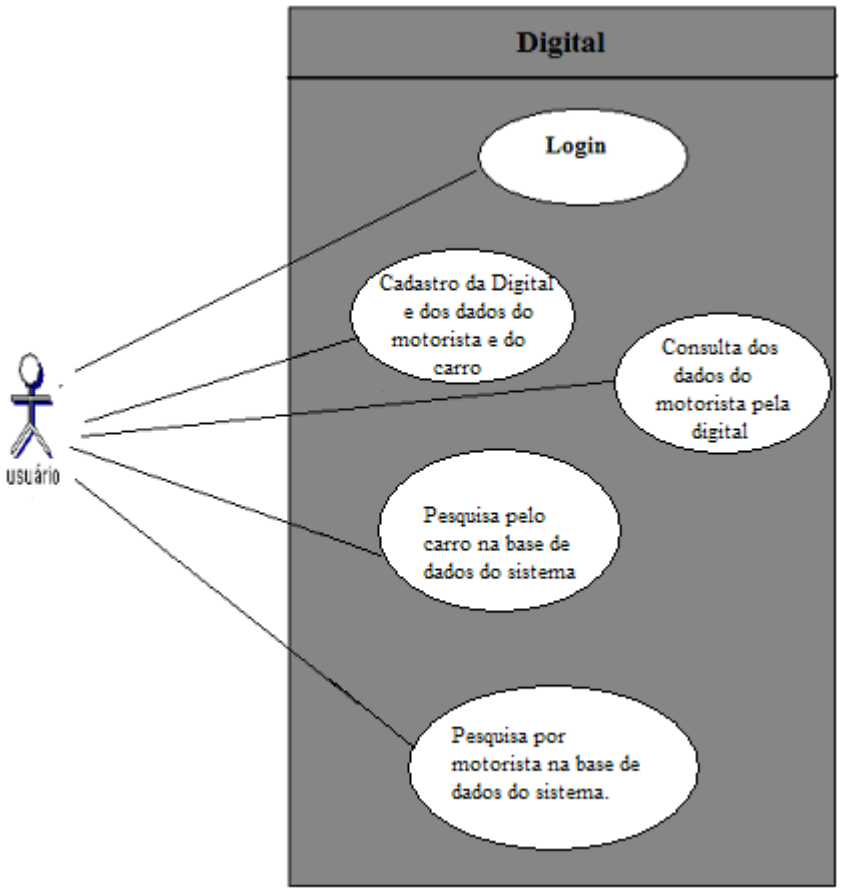

Figura 3.1 – Diagrama de Caso de Uso

O diagrama acima mostra o domínio do projeto desenvolvido, o relacionamento do sistema com o usuário externo. O usuário irá fazer um login para poder ter acesso ao sistema, após feito isso, poderá realizar o cadastramento da digital, consultar ou alterar ou incluir algum dado cadastral do motorista ou veículo.

3.1.2. Diagrama de atividade

O diagrama de atividade é um modelo que mostra o fluxo das atividades. As atividades são representadas por retângulos com cantos arredondados. Normalmente as atividades são estados de ação.

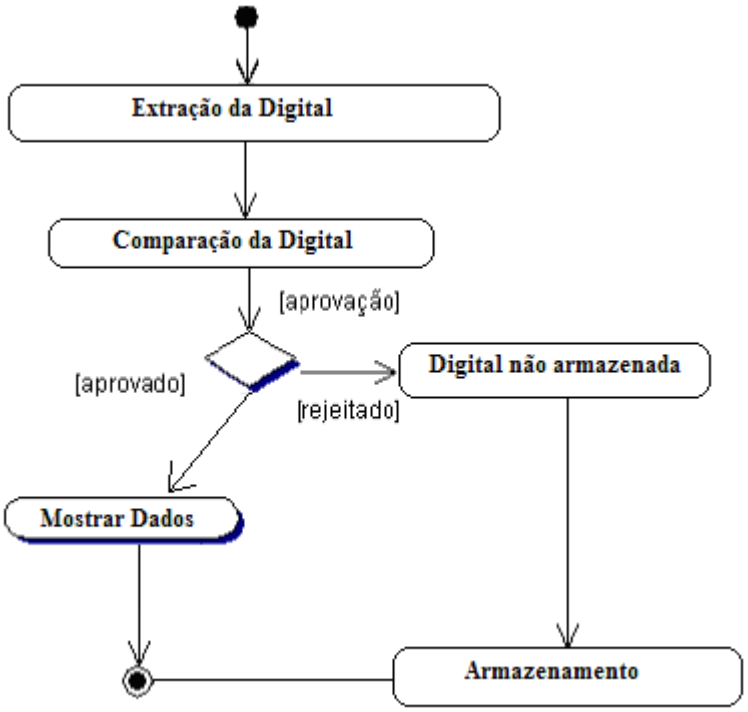

Figura 3.2 – Diagrama de Atividade

O diagrama de atividade é a representação das atividades executadas em um caso de uso. Sendo que cada etapa de um caso de uso tem uma atividade diferente. Neste caso está representado a atividade de consulta e cadastramento.

3.1.3 Diagrama de Classes

O diagrama de classes é a que mostra a estrutura estática do modelo da aplicação. Esse modelo exibe as classes do sistema e o grau do relacionamento entre elas.[31] Os

[31] DESTRO, DANIEL - Softech Network – UML Unified Modeling Language

diagramas de classe são importantes não só para visualização, especificação e documentação, mas também na construção de sistemas executáveis pela engenharia de software. Os elementos básicos em um diagrama de classes são: Classes, Interfaces e Relacionamentos.

3.1.4 Diagrama de Estados

O diagrama de estados mostra todos os possíveis estados dos objetos de uma classe. Mostra também quais eventos do sistema causam essas possíveis mudanças de estado. Porém não há necessidade de representar os estados dos objetos de todas as classes, já que alguns se repetem,[32] ou seja, o diagrama especifica a seqüência de estados pelos quais um objeto passa durante seu tempo de vida em resposta a eventos, juntamente com as suas respostas a esses eventos, porém uma desvantagem é a necessidade de definir todos os possíveis estados de um sistema, dessa forma, o Diagrama de Estados é utilizado para modelar apenas algumas situações.

[32] DESTRO, DANIEL - Softech Network – UML Unified Modeling Language

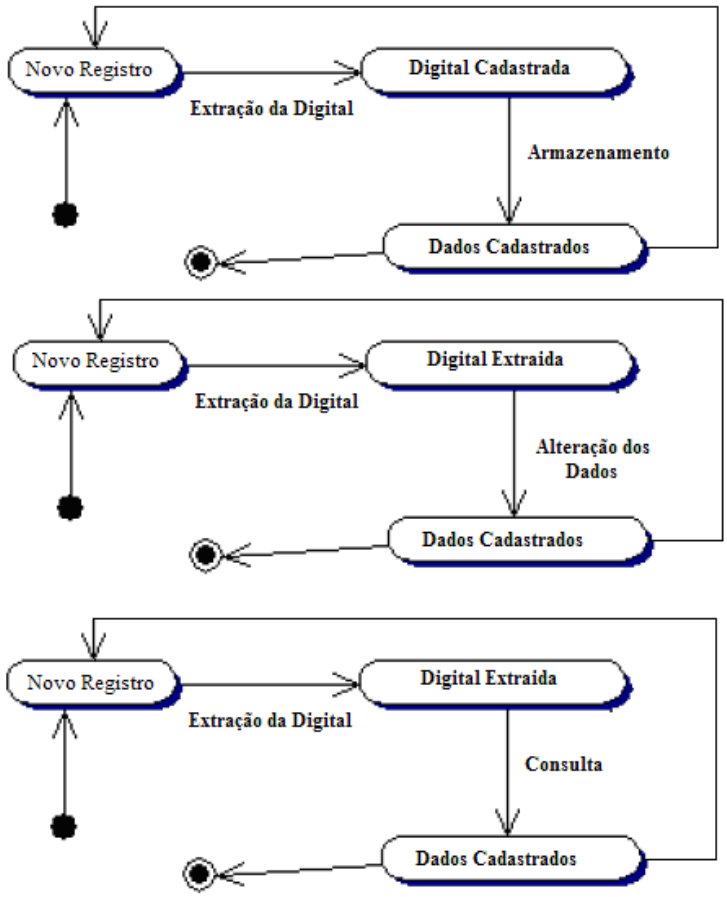

Foto 3.4 – Diagrama de Estados

Na figura 3.4 é possível ver algumas situação de estados do projeto em questão. É possível ver a passagem do estado novo para a digital extraída. E os eventos que causam essas mudanças.

4. Banco de Dados Distribuídos

A solução proposta utiliza Banco de Dados distribuídos, mas para entender melhor o projeto é preciso buscar o conceito de banco de dados distribuído.

O conceito básico de banco de dados distribuído é uma interligação de vários bancos de dados por uma rede de computadores, onde existem dois tipos de bancos de dados, os homogêneos e os heterogêneos. Os homogêneos são aqueles onde todos os bancos interligados na rede têm o mesmo modelo, já o heterogêneo existe mais de um modelo de banco dentro da mesma rede.

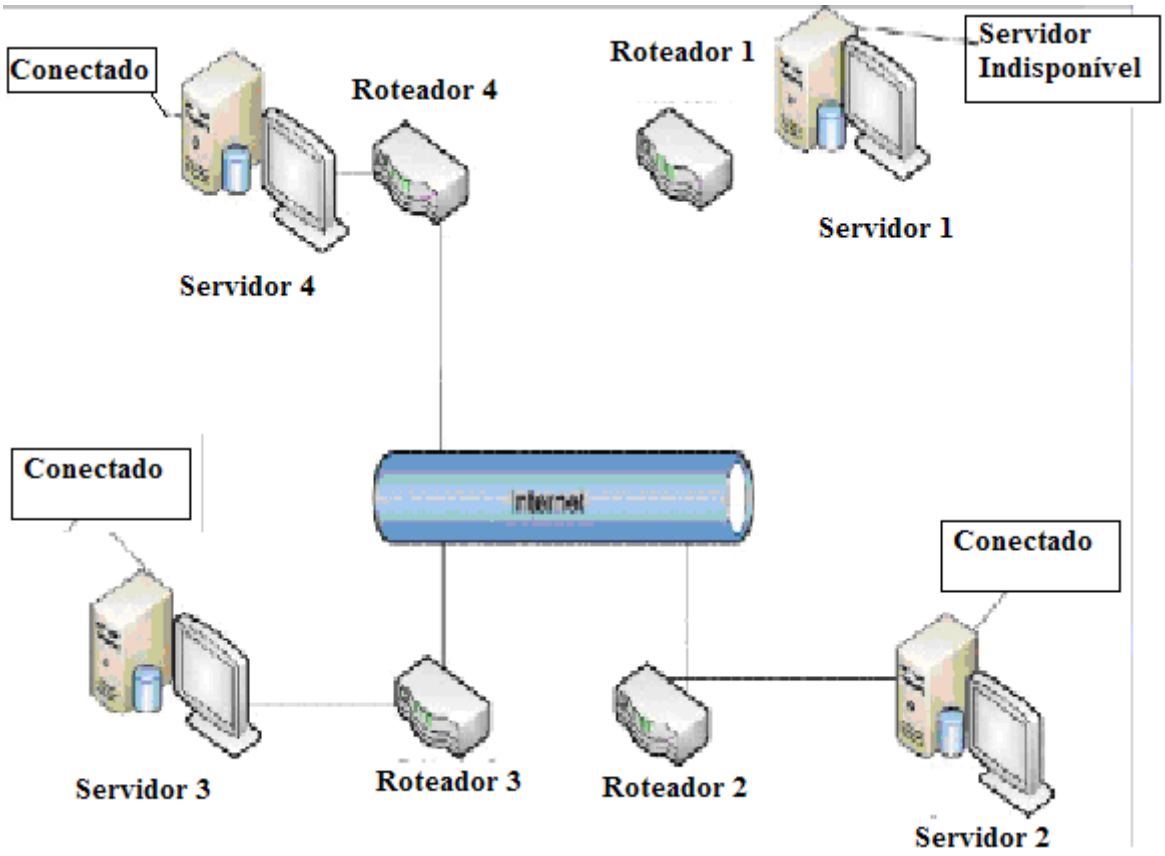

Figura 4.1 – Banco de Dados Distribuído – Fonte: www.inf.ufsc.br[33]

[33] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008

Em um banco de dados distribuído o dado é armazenado em vários computadores que se comunicam em alta velocidade e os arquivos podem estar fragmentados ou replicados na rede. Uma forma de armazenamento do dado em um banco de dados distribuído é a replicação que é quando um arquivo está replicado, ou seja, o arquivo está contido dentro de todos os modelos presentes na rede. A vantagem dessa forma de armazenamento é que fazendo isso aumenta-se a disponibilidade e o paralelismo, mas a desvantagem é que as atualizações dos dados tem que ser feitas em todos os servidores para manter a consistência dos dados. Essas atualizações de réplicas podem ser feitas de três formas diferentes: na primeira chamada de "*Snapshot"* é feita uma cópia completa das tabelas replicadas podendo ocorrer atualização ou não. Outra forma de atualização é a "Incremental", onde os dados alterados são transferidos pela rede, automaticamente, em um horário programado. E a terceira e última forma de atualização das réplicas é a do método "Transacional", onde os dados são atualizados nas réplicas no instante em que são modificados.[34]

Outra forma de armazenamento do dado é a fragmentação que é onde os arquivos se encontram divididos ao longo do sistema e esses dados que ficam dentro das tabelas podem ser divididos em dois ou mais fragmentos. E cada um desses fragmentos é armazenado em um servidor diferente. A fragmentação tem que ser transparente para que o usuário tenha a visão completa da tabela.

A fragmentação pode ser feita de três formas diferentes:

a) Fragmentação horizontal, cada "tupla", é armazenada em um servidor diferente. Para realizar a fragmentação horizontal existem três técnicas diferentes: o Round-Robin que divide igualmente os dados entre os servidores; Hash que usa uma função hash para

[34] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008

determinar o servidor que vai conter cada “tupla”; Por faixa que armazena no servidor as “tuplas” que estiverem dentro de uma faixa de valores. Além disso, essa fragmentação pode ser feita de duas formas diferentes: pode ser primária, onde os atributos usados para fragmentar a tabela fazem parte da mesma; ou pode ser derivada, onde os atributos usados para fragmentar a tabela fazem parte de outra tabela, para isso é necessário que as tabelas estejam interligadas através de uma atributo chave.[35] b) Outra forma de fragmentação é a vertical onde as relações são desfeitas em conjuntos de atributos mantidos em servidores diferentes. c) E na fragmentação mista é a combinação das outras duas fragmentações, a horizontal com a vertical.

Na figura 4.2, logo abaixo, é possível visualizar exemplos dos três tipos de fragmentações:

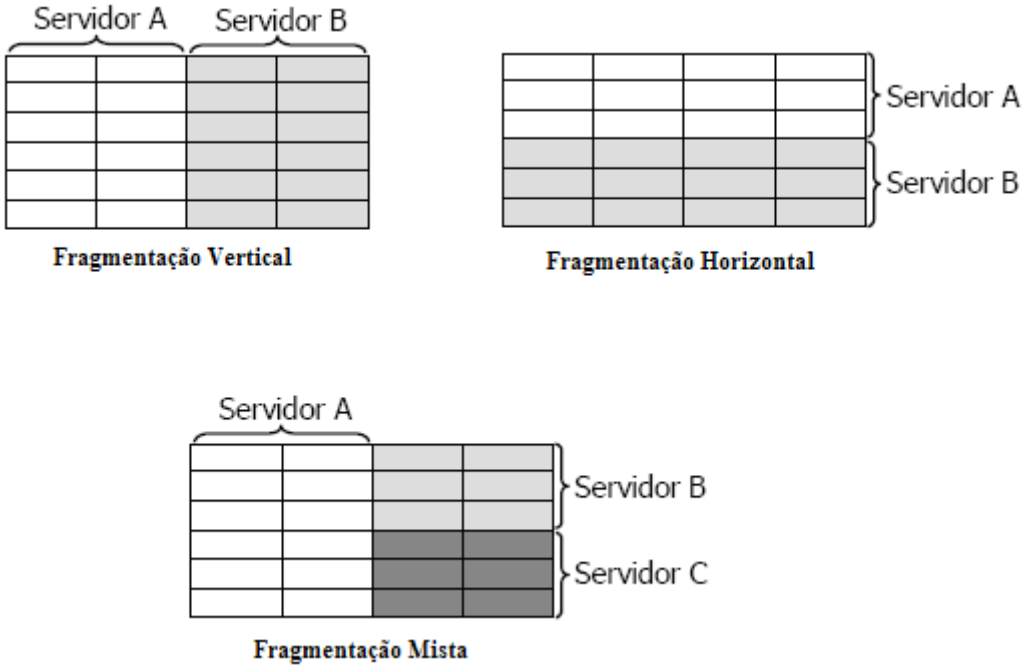

FIGURA 4.2 – Tipos de Fragmentação

[35] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008

Em um banco de dados distribuído existem dois tipos de usuários: os globais e os locais, sendo que o primeiro tem uma visão geral do sistema, ou seja, visualizam um esquema externo global. Já os locais acessam diretamente o servidor local, logo visualizam um esquema externo local. E, ainda, observa-se que dentro de cada um desses tipos existem os usuários administradores, programadores e os usuários finais.

Para se trabalhar com BDD é necessário cumprir alguns requisitos:

- Os mecanismos de controle e gerenciamento devem trabalhar de maneira integrada;
- É necessário a utilização de algoritmos e protocolos adequados para a otimização e processamento de consultas distribuídas;
- Modelos lógicos e linguagens de definição e manipulação de dados para BDD heterogêneos.[36]

4.1 Classificação de Banco de Dados

Existem vários tipos de bancos de dados, e assim é até possível fazer uma classificação dos Bancos e compará-los.

- Banco de Dados Distribuídos:
  - Banco de dados inter-relacionados localizados em diferentes servidores interconectados por uma rede.
- Banco de Dados Paralelos:

[36] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008

- O servidor utiliza uma máquina paralela, com vários processadores, para processar consultas/transações.

- Banco de Dados Cliente-Servidor:
  - Os dados podem ser acessados por clientes remotos ligados ao servidor por uma rede de comunicação.

- Banco de Dados Centralizados:
  - Acesso ao banco de dados apenas a partir da máquina na qual o mesmo encontra-se localizado.

No banco de dados cliente-servidor os dados tornam-se amplamente acessíveis para os usuários, e permitem o acesso remoto a partir de máquinas conectadas ao servidor através da rede. Sendo que o acesso pode ser feito utilizando um aplicativo cliente do Sistema gerenciador de banco de dados ou uma aplicação *web* que acessa os dados no BD.[37]

No banco de dados paralelos, várias máquinas paralelas, vêm sendo usadas para suportar uma carga maior de trabalho dos SBDs, assim, vários processadores executam as operações em paralelo, devido o uso de controle de concorrência. O usuário não consegue visualizar a diferença entre um banco paralelo e um centralizado, devido à transparência. A memória e disco podem ser compartilhados ou não sem contar que o custo das máquinas multi-processadas está caindo cada vez mais, o que barateia esse tipo de banco de dados. O paralelismo pode ser utilizado no processamento das consultas, na entrada e saída de dados, no processamento de operações individuais. Como resultado é possível se processar mais transações tornando o processamento mais rápido, mas, em contra partida existem mais falhas devido à utilização de mais hardware. [38]

[37] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008
[38] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008

O banco de dados centralizados normalmente é usado por empresas de médio e pequeno porte que não tem bases de dados muito altas e nem muitos registros, mas o que se vê no mercado hoje em dia é o crescente uso de banco distribuído já que é mais eficaz quando se trabalha com um volume muito grande de dados. Esses bancos atualmente, exercem um papel vital na operação das empresas e assim, como qualquer outro sistema, tem as suas vantagens e desvantagens.

Para entender melhor a diferença entre os bancos foi feita uma comparação entre o banco de dados distribuído e os demais. Assim, comparando o BDs Distribuído com o Centralizado verificamos que:

- BDs centralizados possuem um ponto único de falha;
- BDs distribuídos podem aumentar a robustez do sistema e a disponibilidade dos dados
- BDs centralizados são limitados pela capacidade de processamento e armazenamento de uma máquina
- BDs distribuídos podem crescer em escala adicionando novos servidores ao sistema
- BDs distribuídos são mais sujeitos a apresentar falhas parciais de funcionamento e de segurança
- BDs distribuídos são mais difíceis de administrar pois os servidores estão em locais diferentes

Agora comparando o BDs distribuído com o cliente-servidor verificamos que:

- *BDs distribuídos também podem ser acessados remotamente por clientes*

E por último na comparação de um BDs distribuído com um paralelo concluímos que:

- *Ambos podem processar consultas em paralelo usando os vários processadores disponíveis;*
- *Em BDs paralelos os processadores podem trocar dados usando discos ou memória compartilhada;*
- *O uso da rede em BDs distribuídos pode prejudicar o desempenho do sistema ao processar consultas;*
- *BDs paralelos são mais vulneráveis a algumas falhas e não são tão escaláveis quanto BDs distribuídos;*[39]

[39] Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acesso em: 20 de Outubro de 2008.

4.2 Vantagens na Utilização de BDD

Além das diversas vantagens, explicitadas anteriormente, para o processamento e o desempenho, podemos citar outras vantagens no meio comercial para a utilização de banco de dados distribuídos, pois quando adotado o BDD fica mais fácil gerenciar um negócio de modo eficiente, além disso é possível verificar a eficiência de campanhas de marketing, definir preços padrões, promoções, condições de compra dos produtos, otimizar a quantidade de produtos no estoque, respostas rápidas a mudança do mercado, determinar o publico alvo de um produto, ou seja, é possível ganhar mais eficiência e lucratividade.

O acesso a banco de dados distribuídos pode ser feito por uma linguagem de consulta, utilizando aplicações gráficas, aplicações de rede, páginas e formulários da web demonstrando a facilidade de sua utilização até por leigos. Esses acessos podem ser: locais quando as aplicações não requerem dados de outros lugares; ou globais quando a aplicação requer dados de outros lugares.

A utilização de um BDD proporciona ao cliente uma resposta mais rápida, um armazenamento de dados maior, um alto fluxo de informações sem perda da confiabilidade. Proporciona, uma maior mobilidade, já que a maioria das empresas utiliza mais de um banco de dados e muitas vezes dispersos em vários locais. Mas para isso é preciso ter uma visão integrada dos dados, pois a operação de múltiplos bancos de dados pode levar a uma inconsistência de dados nos diversos bancos.

5. Soluções e Propostas

A solução proposta é a utilização de um scanner de impressão digital interligado a um computador portátil que utilizando um modem 3G possibilitaria acessar o servidor que estaria na WEB para fazer a conferência da digital durante a Blitz.

Para alcançar o objetivo deste trabalho foi desenvolvido um sistema de cadastramento, consulta e pesquisa dos dados, da foto e da digital do motorista, onde a consulta ao banco de dados é feita de forma remota através do uso da internet via modem 3G.

Foto 5.1 – Tela de consulta da digital do sistema desenvolvido

Em uma visão geral o projeto é simples, primeiramente, foi feito um cadastro do motorista, com a digital e seus dados assim como do carro que o motorista dirige. Portanto, quando o motorista for parado por uma blitz de trânsito, o agente responsável pela fiscalização conferirá os dados do motorista e adicionará ao seu cadastro no campo histórico de blitz, o local aonde foi feita a blitz e algumas observações referentes aquela fiscalização, para poder assim registrar e controlar os locais por onde aquele motorista costuma trafegar, pois caso ocorra algum acidente, infração ou crime onde não se consiga identificar a placa do

automóvel, a polícia poderá procurar o fugitivo fazendo uma pesquisa na base de dados para verificar os motoristas que dirigem os veículos com as mesmas características.

Para a elaboração desse sistema foi estudado e utilizado engenharia de software com modelagem UML, a fim de desenvolver um software de qualidade, para poder padronizar o sistema do Detran e para poder melhor entender o sistema que estamos elaborando. Com esse estudo de modelagem UML e a aplicação dessa modelagem o sistema será melhor entendido, desenvolvido e integrado aos outros órgãos do governo futuramente. A modelagem escolhida foi a UML, pois a modelagem UML já se tornou a modelagem padrão, logo é de fácil entendimento. E porque a modelagem UML proporciona aos desenvolvedores e aos clientes um fácil entendimento do sistema elaborado.

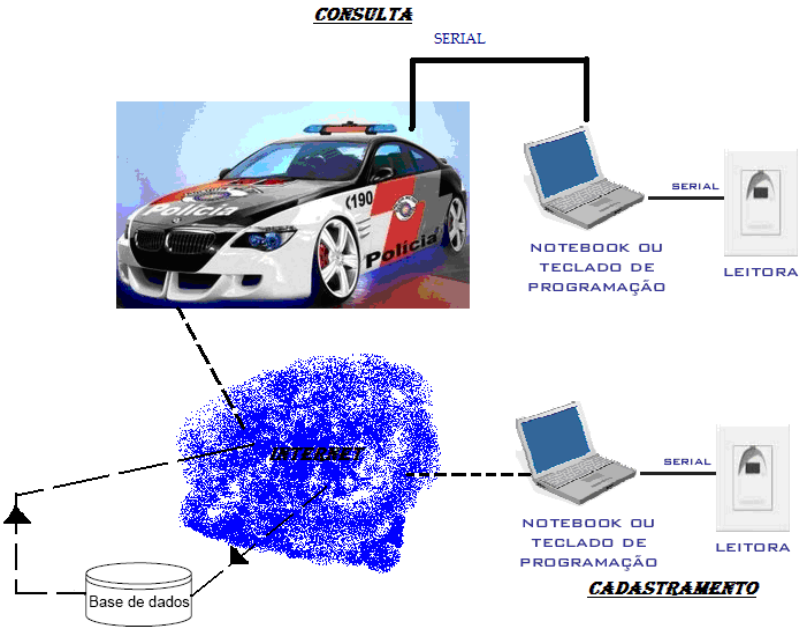

Figura 5.2: Esquema de funcionamento da Proposta

Um passo-a-passo do processo é demonstrado pela figura acima. Onde primeiramente é feito o cadastramento da digital e dos dados do motorista e do veículo que o motorista dirige. Esses dados são armazenados no banco de dados dentro de um servidor

WEB. Num segundo momento, durante a blitz, o Policial de trânsito faz a consulta da digital do motorista através de um outro computador que estaria dentro de uma viatura. Esse computador utilizando um modem 3G vai acessar o servidor e fazer a consulta da digital, alterar/ acrescentar alguns dados sobre a blitz ou cadastrar o usuário.

O sistema de autenticação é dividido em dois módulos: o módulo de comparação e o módulo de registro, sendo que no de comparação as características das digitais extraídas serão comparadas pelo drive DLL (programa da própria leitora) de acordo com os pontos característicos retirados na leitura, e no módulo de registro o sistema vai armazenar as informações do motorista no banco de dados e a digital extraída. No módulo de registro também será armazenado as informações do motorista com os campos da carteira de motorista e as informações do carro que o motorista dirige.

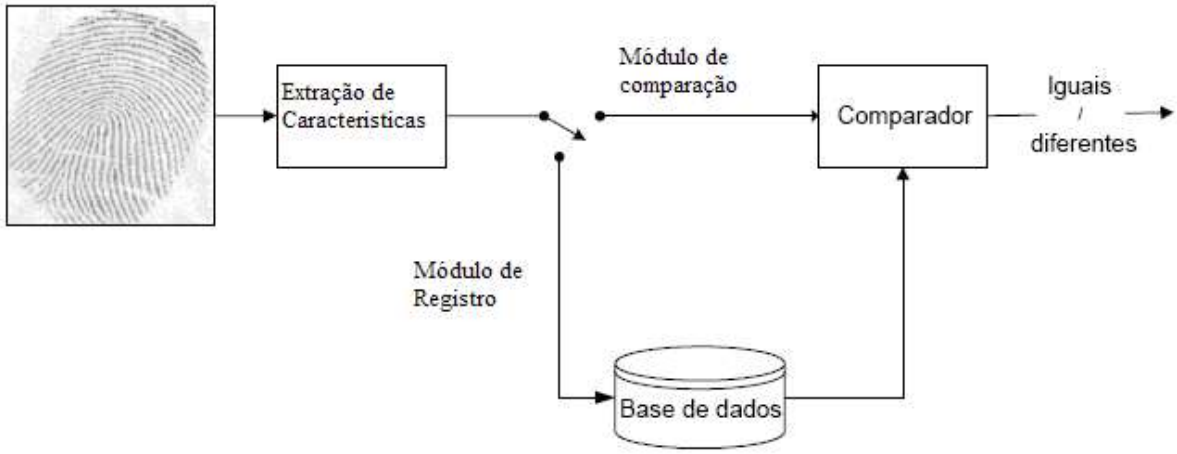

Figura 5.3: Esquema de funcionamento do Sistema

Para realizar o cadastramento do motorista, ele terá que colocar o dedo no scanner, onde será feita uma leitura da digital pela leitora, recolhendo assim os dados do dedo do indivíduo. O drive DLL da própria leitora irá retirar seis pontos únicos da digital e irá codificar esses pontos, isso para garantir a segurança da informação armazenada, e após a extração e codificação as informações serão armazenadas na base de dados. O cadastramento da digital é feito juntamente com o cadastro dos dados do motorista e do veículo que o mesmo

dirige. Esse processo de armazenamento poderá ser efetuado tanto nas sedes dos DETRAN como no meio da rua, mas caso o motorista seja parado no meio da rua e não tenha feito o cadastro o mesmo precisará mostrar a carteira de motorista, para o policial poder fazer uma pesquisa ao banco de dados com o CPF e verificar se o motorista não tem cadastro. O cadastramento da digital é feito juntamente com o cadastro dos dados do motorista e do veículo que o mesmo dirige.

Já, com relação à consulta da digital é um pouco diferente, para fazê-la é necessário que o motorista já tenha feito um cadastro prévio e, conseqüentemente, esteja com os dados na base de dados, pois a consulta é feita através da extração dos mesmos seis pontos únicos da digital que são codificados. Logo após essa codificação é feita uma pesquisa na base por dados com aquela mesma codificação no campo de digitais cadastradas, e quando encontrados, o sistema retorna com os dados do motorista que aquele código está apontando.

O banco de dados onde as informações serão armazenadas será um Banco de Dados distribuído homogêneo pois precisaremos de velocidade de transmissão e processamento, já que estaremos armazenando tanto a foto da digital como a foto do motorista de todos os motoristas do Brasil. O banco de dados distribuído estará interligado numa rede Partial Mesh do Detran. A rede Partial Mesh foi escolhida por garantir uma grande disponibilidade à rede sem afetar no seu desempenho e tem custo baixo. A utilização de banco de dados distribuídos ajuda na troca de sistema do DETRAN, ou seja, facilita a troca do sistema antigo e entrada no novo sem causar muito problemas para o governo.

O banco de dados irá receber os dados do motorista, carro, que o motorista dirige, e da blitz. Estes campos são: nome, RG, data de nascimento, CNH, categoria, data de validade da carteira de motorista, data de emissão da carteira de motorista, dedo1, dedo2, dedo3, primeira habilitação, filiação, observações, órgão emissor, UF, a foto do motorista, ACC, permissão, local de emissão, ano do veículo, modelo do veículo, cor do veículo, marca do veículo, placa

do veículo, nome da operação policial (Blitz), descrição da blitz, data da blitz, hora da blitz, local da blitz, um campo delito caso o motorista tenha cometido algum delito que tenha sido verificado durante a blitz e o campo observação para o agente policial acrescentar algo relevante à fiscalização. O modelo do banco pode ser visto na figura 5.5 abaixo.

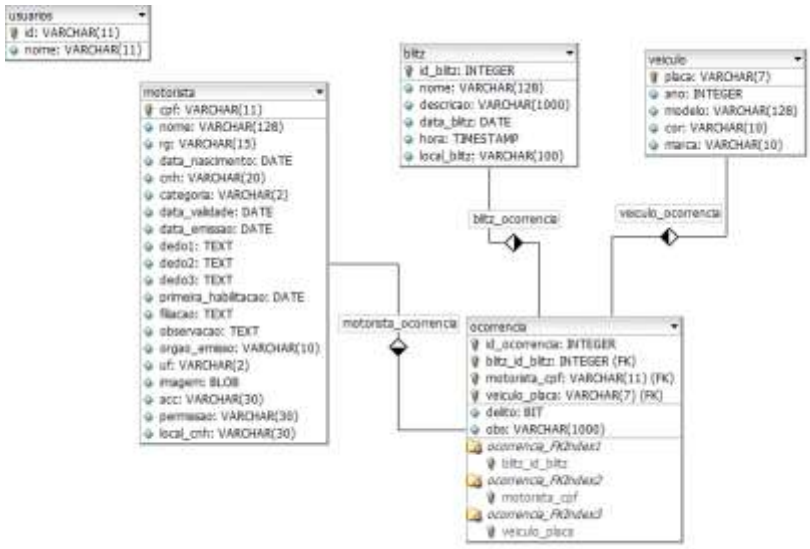

Figura 5.4 – Modelo do Banco de Dados

5.1 Requisitos para a aplicação

Alguns requisitos básicos serão necessários para o funcionamento do sistema. Entre eles será necessário que o condutor já tenha um cadastro no banco de dados do DETRAN, caso contrário ele necessitará fazer um cadastro que poderá ser feito na sede do DETRAN ou até mesmo durante a blitz. Para ser autenticado.

Caso o motorista não seja cadastrado e tentar se autenticar o sistema mostrará uma mensagem de usuário não identificado. Assim o policial fará uma consulta através do CPF do motorista que aparece na carteira de motorista. Caso a cadastro não seja encontrado quer dizer que o motorista realmente não tem cadastro e vai ser cadastrado ou ali na hora ou

depois numa das sedes do DETRAN. Caso o CPF seja encontrado será analisado os dados do motorista para ter certeza que a pessoa que esta ali dirigindo é a mesma pessoa que aparece no sistema.

A estrutura proposta está baseada no desenvolvimento de uma aplicação JAVA que estará interligada na WEB com o banco de dados, para a autenticação. E para isso foram utilizadas as seguintes ferramentas: o software Eclipse, que interligado com o Drive DLL da leitora e com o Banco de Dados irão fazer a interface gráfica com o usuário, o cadastramento dos dados do motorista, o cadastramento dos dados do veículo e o cadastramento da digital utilizando linguagem de programação JAVA. Para se fazer a consulta da digital remotamente será utilizado um modem 3G ZTE e será comprado um domínio na web e alugado um servidor para poder disponibilizar o banco de dados no SGBD Postgres SQL e o sistema operacional será o Microsoft Windows XP.

Na simulação da proposta, o resultado esperado é que o usuário faça o cadastramento com sucesso e depois fará uma consulta de outro computador no Banco através da internet com um modem 3G. Logo a obrigatoriedade no porte da carteira de motorista não mais existirá.

5.2 Leitor Biométrico utilizado no trabalho

No desenvolvimento deste projeto é utilizado o leitor biométrico da Microsoft® o Microsoft FingerPrint Reader. No Anexo I podem ser encontradas algumas características técnicas desse leitor. A utilização de um scanner de impressão digital foi utilizado por que é a que apresenta a melhor relação custo benefício, pelo fato de ser de simples implementação, de fácil compreensão e por ser portátil.

Figura 5.5: Leitor Biométrico – Fonte: Smartsec[40]

A escolha desse leitor se deu principalmente pelo seu baixo custo em relação a outros disponíveis no mercado, por não haver necessidade de tratamento de superfície, ser resistente a choques, superfícies sem riscos e sem corrosão, não apresenta problemas quando de encontro com eletricidade estática e por ser um leitor já testado com sucesso em monografias anteriores e ter resultados acima dos satisfatórios.

Este leitor, originalmente, é utilizado para autenticação em computadores pessoais em substituição de senhas de redes ou senhas de seguranças locais, o driver utilizado era gratuito até a apresentação desse projeto e a aplicação foi desenvolvida para esse fim.

5.3 O Computador, os sistemas operacionais e outros dispositivos utilizados

O modem 3G utilizado será um modem USB MF622 da BrasilTelecom. Tráfego de dados sobre HSDPA/UMTS/EDGE/GPRS, com velocidade de até 7.2Mbps sobre HSDPA, suporta envio de SMS e controle de grupo de envio em ambiente Windows. Trabalha na freqüência Tri-band e Quadri-Band e é uma tecnologia Plug & Play. Que utilizará o software ZTE CONNECTION MANAGER para fazer a conexão com a internet. Como o

[40] Disponível em: http://www.smartsec.com.br/Microsoft_FingerPrint_Reader_USB_Leitor_biometria.html Acesso em 29/07/2008

modem possui a tecnologia Plug & Play ele será instalado automaticamente, e quando inicializado abrirá a tela abaixo onde será feita a conexão com a internet.

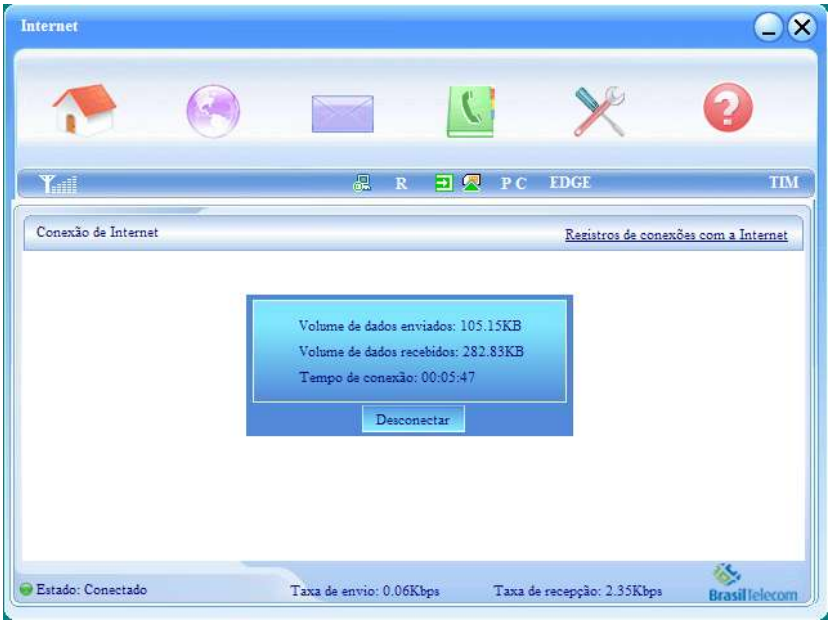

Figura 5.6: ZTE CONNECTION MANAGER

Os computadores e os sistemas operacionais que serão utilizados para processamento do sistema são:

- Dell Notebook Latitude D620
    - Processador Intel T2400 – 1,83GHz
    - 1 GB de RAM
    - HD 80 GB
    - Sistema Operacional Windows XP Professional

- Dell Vostro 1000
    - Processador AMD Turion 64 X 2 Mobile 1,80GHz
    - 2 GB de RAM

- HD 80 GB
- Sistema Operacional Windows XP Home Edition

5.4 As dificuldades encontradas

Durante o desenvolvimento do projeto foram encontradas algumas dificuldades entre elas:

1. Dificuldade na comparação da digital “scanneada” com a presente no Banco de dados.
2. Dificuldade de armazenamento da digital no Banco de Dados
3. Dificuldade de literatura sobre a CNH, já que o DETRAN não divulga os dados sobre a mesma.
4. Dificuldade de manipulação dos dados da digital pela leitora.

5.5 As vantagens e desvantagens da proposta

A principal vantagem da proposta não é apenas os motoristas não terem mais que se preocuparem com o documento obrigatório ao estar dirigindo e não passar por alguns inconvenientes ao esquecer a carteira de motorista em casa quando se é parado numa blitz. Mas também tem a vantagem do titular da carteira não ter mais que passar por alguns inconvenientes ao ter a carteira roubada, pois se implantando a proposta o ladrão não conseguirá mais falsificar o documento alheio para depois utilizá-lo para fins ilícitos. Assim tal vantagem seria a maior facilidade na fiscalização dos motoristas e dos carros, pois com o

procedimento terá um maior grau de confiabilidade, e assim se evitará o uso de carteiras falsificadas e de habilitação vencida.

Mais uma vantagem é o estudo dos motoristas e dos veículos que possibilita a polícia fazer pesquisa voltada para identificar crimes e acidentes de trânsito, onde o infrator tenha fugido do local. Dessa forma, a polícia em conjunto com a base de dados poderá filtrar os dados o que facilitará o trabalho na procura por suspeitos dos atos.

Uma outra vantagem que poderá ser utilizada no futuro é a facilidade de integração dos sistemas utilizados pelas polícias e pelo governo, ou seja, com a utilização desse sistema e alguns ajustes será possível criar um documento único para todos os cidadãos, o que dará um maior controle e segurança no controle da população brasileira.

No entanto, é preciso comentar também que o projeto possui algumas desvantagens, e dentre elas é possível citar: o alto custo que o governo terá para implantar o cadastramento das digitais de todos os motoristas que já possuem a atual carteira, o custo para comprar as leitoras biométricas e instalação em todas as viaturas. Sem contar no problema da falta de cobertura 3G em todas as regiões brasileiras, isso porque o Brasil tem uma área territorial enorme e em determinados lugares, infelizmente, não existe cobertura para se fazer a consulta remota à base de dados.

Conclusão

Com todo o estudo, pesquisa desenvolvida e, enfim, com a execução deste projeto é possível concluir que para a elaboração de um sistema é necessário que sejam estudados e analisados os problemas que se deseja resolver. Só após ser analisado o problema e a solução pensada e elaborada é possível começar a pensar em como desenvolver uma solução viável. Onde para esse projeto primeiramente foi feito um estudo do projeto como um todo, definindo a modelagem, o banco de dados e as ferramentas viáveis para a utilização.

Com a conclusão deste projeto foi possível entender melhor como funciona a fiscalização da polícia de trânsito sobre as carteiras de motorista, entendendo inclusive, para que serve cada um dos campos, as suas vulnerabilidades e a evolução da Carteira Nacional de Habilitação.

Também, foi possível, adquirir conhecimento mais bem apurado no que diz respeito a biometria, identificar alguns dos seus tipos, e ainda, demonstrar o porque a impressão digital foi utilizada como solução deste trabalho, verificando como é feita a identificação da digital pela leitora, e enfim, concluindo, que é um processo de extrema confiabilidade, onde, a cada dia que passa, a segurança das informações estão cada vez mais protegidas.

O banco de dados distribuído, pode ser analisado, entendido e visualizado na sua aplicação. Bem como entender a diferença de Banco de Dados Distribuídos para os outros tipos de Banco de Dados.

E, ainda, ver como o sistema criado e desenvolvido resolveu o problema da falta de coerência na fiscalização, a falta de confiabilidade na fiscalização da carteira de motorista e além de aumentar o número de verificações e diminuir o tempo dessas verificações. Ver como o sistema interliga todos os Departamentos de Trânsito do Brasil.

O projeto possibilita que trabalhos futuros possam vir a melhorar ou atualizar o sistema criado. Onde trabalhos futuros podem conectar a leitora a um GPS para armazenar a posição correta da blitz ou até mesmo fazer todos os sistemas para um celular com leitora digital embutida. Além disso, esse trabalho possibilita que no futuro outras bases de dados sejam incluídas nele. Assim pode-se chegar a um documento único. Onde todas as polícias estejam interligadas com a mesma base de dados. Para assim aumentar o combate ao crime. E interligar vários documentos e órgãos do governo.

Referências Bibliográficas

VIGLIAZZI, Douglas. **Biometria – Medidas de Segurança.** Editora Visual Books, 2006

PINHEIRO, Jose Mauricio. **Biometria nos Sistemas Computacionais – Você e a senha.** Editora Ciência Moderna, 2008.

DESTRO, Daniel - **Softech Network – UML Unified Modeling Language.**

DATE, C. J.. **Introdução a sistemas de bancos de dados**. 7ª ed. Rio de Janeiro: Campus, 2000.

GOYA, Denise Hideko. **Biometria: Esqueça todas as senhas.** São Paulo, PC World, n. 98, p 58, ago.2000.

GUNNERSON, Gary. **Pronto para a Biometria?** PC Magazine Brasil, São Paulo, v. 9, n. 3, p. 82-88, mar. 1999.

Consultores Biométricos <http://www.consultoresbiometricos.com.br>. Acesso em 15 de abril de 2008.

GrFinger 4.2 Developer's Manual 2006

PACHECO, César Alexandre Rodrigues dos Anjos. **Autenticação com Impressão Digital.** Lisboa 2003.

VIEIRA, Eduardo. **Você e a senha.** Info Exame, São Paulo, v.16, n. 178, p 79-83, jan. 2001.

Disponível em: http://www.via6.com/topico.php?tid=109871. Acesso em 9 de outubro de 2008.

Disponível em: http://www.inf.ufsc.br/~frank/BDD/BDDIntro.pdf Acessado em: 20 de Outubro de 2008.

Disponível em: http://www.slideshare.net/andrefachin/banco-de-dados-distribuidos-66037/. Acessado em: 20 de Outubro de 2008.

Disponível em: http://www.ece.uah.edu/biometric. Acesso em 9 de outubro de 2008.

Disponível em: http://www.infowester.com/biometria.php. Acesso em 18 de setembro de 2008.

Disponível em: http://www1.folha.uol.com.br/folha/informatica/ult124u21496.shtml. Acesso em 18 de setembro de 2008 – Matéria de 27/01/2007.

Disponível em: http://www.smartsec.com.br/Microsoft_FingerPrint_Reader_USB_Leitor_biometria.html/. Acesso em 29/07/2008

Disponível em: https://denatran.serpro.gov.br/index2.htm. Acesso em: 1 nov.2008.

Disponível em: http://www.detran.df.gov.br/. Acesso em: 1 nov. 2008

Disponível em: http://idgnow.uol.com.br/seguranca/2006/08/30/idnoticia.2006-08-29.9472410830/. Acesso em 23 out. 2008

Disponível em: http://www.fujitsu.com/pt/news/pr20050512-01.hmtl/. Acesso em 29 ago. 2008

Liu, S.e.S., M. "A. Practical Guide to Biometric Security Technology," (IEEE Computer Society) 2001.

Disponível em: http://www.ibc.gov.br/?itemid=121. Acesso em 01 nov. 2008

ANEXO I – Data Sheet da leitora

Microsoft®
Fingerprint
Reader

Version Information	
Product Name	Microsoft® Fingerprint Reader
Product Version	Microsoft Fingerprint Reader v1.0
Fingerprint Reader Version	Microsoft Fingerprint Reader v1.0
Product Dimensions	
Fingerprint Reader Length	3.23 inches (82.0 millimeters)
Fingerprint Reader Width	1.97 inches (50.0 millimeters)
Fingerprint Reader Depth/Height	0.60 inches (15.7 millimeters)
Fingerprint Reader Weight	3.78 ounces (107 grams)
Fingerprint Reader Cable Length	55.9 ± 1.18 inches (1420 ± 30.0 millimeters)
Compatibility and Localization	
Interface	USB Compatible
Operating Systems	Microsoft Windows® Vista™ and Windows XP Professional/Home/Media Center/Tablet PC Edition
Top-line System Requirements	• Requires a PC that meets the requirements for and has installed one of these operating systems: Microsoft Windows Vista or Windows XP Professional/Home Edition/Media Center Edition/Tablet PC Edition • 45 MB of available hard disk space (significant additional hard-disk space may be required if System Restore is enabled.) • Microsoft Internet Explorer® 6.0 and MSN® Explorer 8.0 and 9.0 • USB Port • CD drive • digitalPersona Fingerprint Password Manager software version 2.0
Compatibility Logos	• Designed for Microsoft Windows XP • Certified USB logo
Product Feature Performance	
Storage Temperature & Humidity	-40 °F (-40 °C) to 140 °F (60 °C) at <5% to 65% relative humidity (non-condensing)
Operating Temperature & Humidity	32 °F (0 °C) to 104 °F (40 °C) at <5% to 80% relative humidity (non-condensing)
Fingerprint Reader Technology	
Scanner Type	Optical
Fast User Switching	Yes
Browser Password Replacement	Yes
Certification Information	
Country of Manufacture	People's Republic of China (PRC)
ISO 9001 Qualified Manufacturer	Yes
Agency and Regulatory Approvals	• FCC Declaration of Conformity (USA) • UL and cUL Listed Accessory (USA and Canada) • ICES-003 report on file (Canada) • TUV-GS Certificate (Germany) • CE Declaration of Conformity, Safety and EMC (European Union) • GOST Certificate (Russia) • VCCI Certificate (Japan) • ACA Declaration of Conformity (Australia) • BSMI Certificate (Taiwan) • MIC Certificate (Korea) • NOM Certificates (Mexico) • CB Scheme Certificate (International)
Windows Hardware Quality Labs (WHQL)	ID: 1210759 Microsoft Windows XP (x86) and Windows Vista (x86)

Apêndice I – Código da Aplicação

```
package br.com.detran.negocio.blitz;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class BliitzBO {
	public List<Blitz> getBlitzs() throws ConsultaDaoException {
		BliitzDao usu = new BliitzDaoImp();
		return usu.getBlitzs();
	}
	public Blitz getBlitz(int id) throws ConsultaDaoException {
		BliitzDao usu = new BliitzDaoImp();
		return usu.getBlitz(id);
	}
	public void setBlitz(Blitz blitz) throws ConsultaDaoException {
		BliitzDao usu = new BliitzDaoImp();
		usu.setBlitz(blitz);
	}
	public void delBlitz(int id) throws ConsultaDaoException {
		BliitzDao usu = new BliitzDaoImp();
		usu.delBlitz(id);
	}
}

package br.com.detran.negocio.veiculo;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public interface VeiculoDao {
	public List<Veiculo> getVeiculos() throws ConsultaDaoException;
	public Veiculo getVeiculo(String placa) throws ConsultaDaoException;
	public void setVeiculo(Veiculo veiculo) throws ConsultaDaoException;
	public void delVeiculo(String placa) throws ConsultaDaoException;

}

package br.com.detran.util;

import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;

public class Base64Utility {

	public static String encodeObject(Object object) throws IOException {
		String result = null;
		ByteArrayOutputStream ostream = new ByteArrayOutputStream();
		ObjectOutputStream os = null;
		os = new ObjectOutputStream(ostream);
		os.writeObject(object);
		byte[] datatoEncode = ostream.toByteArray();
		result = Base64.encode(datatoEncode);
```

```
                os.close();
                ostream.close();
                return result;
        }

        public static Object decodeObject(String data) throws IOException,
                        ClassNotFoundException {
                byte[] content = Base64.decode(data);
                ObjectInputStream oistream = new ObjectInputStream(
                                new ByteArrayInputStream(content));
                Object obj = oistream.readObject();
                oistream.close();
                return obj;
        }
}

package br.com.detran.negocio.veiculo;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class VeiculoFacade {

        public static List<Veiculo> getVeiculos() throws ConsultaDaoException {
                VeiculoBO usu = new VeiculoBO();
                return usu.getVeiculos();
        }

        public static Veiculo getVeiculo(String placa) throws ConsultaDaoException {
                VeiculoBO usu = new VeiculoBO();
                return usu.getVeiculo(placa);
        }

        public static void setVeiculo(Veiculo veiculo) throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                usu.setVeiculo(veiculo);
        }
        public static void delVeiculo(String placa) throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                usu.delVeiculo(placa);
        }
}

package br.com.detran.negocio.veiculo;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.commons.beanutils.BeanUtils;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.apache.struts.actions.DispatchAction;
import br.com.detran.negocio.motorista.Motorista;
import br.com.detran.negocio.motorista.MotoristaFacade;

public class VeiculoDispatchAction extends DispatchAction  {

        @Override
        public ActionForward execute(ActionMapping arg0, ActionForm arg1,
```

```
                HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
            // TODO Auto-generated method stub
            return super.execute(arg0, arg1, arg2, arg3);
        }

        @SuppressWarnings("unchecked")
        public ActionForward p(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

            DynaActionForm frm = (DynaActionForm) form;
            frm.getMap().put("lista", VeiculoFacade.getVeiculos());

            return mapping.findForward("lista");
        }
        public ActionForward e(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

            DynaActionForm frm = (DynaActionForm) form;

            if (frm.getString("placa")!=null && !frm.getString("placa").equals("")) {
                BeanUtils.copyProperties(frm,
VeiculoFacade.getVeiculo(frm.getString("placa")));
            }
            return mapping.findForward("edit");
        }

        public ActionForward i(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

            DynaActionForm frm = (DynaActionForm) form;
            Veiculo veiculo = new Veiculo();
            try {
                BeanUtils.copyProperties(veiculo, frm);
                VeiculoFacade.setVeiculo(veiculo);
            } catch (Exception e) {
                e.printStackTrace();
                return mapping.findForward("erro");
            }
            return mapping.findForward("sucesso");
        }

        public ActionForward d(ActionMapping mapping, ActionForm form,
                HttpServletRequest request, HttpServletResponse response)
                throws Exception {

            DynaActionForm frm = (DynaActionForm) form;
            try {
                VeiculoFacade.delVeiculo(frm.getString("placa"));
            } catch (Exception e) {
                e.printStackTrace();
                return mapping.findForward("erro");
            }
            return mapping.findForward("sucesso");
        }


}
```

```
package br.com.detran.negocio.veiculo;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.List;
import br.com.detran.comum.Dao;
import br.com.detran.conexao.Conn;
import br.com.detran.exception.ConsultaDaoException;

public class VeiculoDaoImp extends Dao implements VeiculoDao {

	public List<Veiculo> getVeiculos() throws ConsultaDaoException {
		CallableStatement query = null;
		Connection con = null;
		ResultSet res = null;
		List<Veiculo> result = new ArrayList<Veiculo>();

		try {
			con = Conn.getInstance().getConnection();
			con.setAutoCommit(false);
			query = con.prepareCall("{? = call f_getveiculos()}");
			query.registerOutParameter(1, Types.OTHER);

			query.execute();

			res = (ResultSet) query.getObject(1);
			while (res.next()) {
				Veiculo veiculo = new Veiculo();
				veiculo.setAno(res.getInt("ano"));
				veiculo.setPlaca(res.getString("placa"));
				veiculo.setModelo(res.getString("modelo"));
				veiculo.setMarca(res.getString("marca"));
				veiculo.setCor(res.getString("cor"));

				result.add(veiculo);
			}
		} catch (SQLException e) {
			e.printStackTrace();
		} finally {
			attemptClose(res);
			attemptClose(query);
			attemptClose(con);
		}
		return result;
	}
	public Veiculo getVeiculo(String placa) throws ConsultaDaoException {
		CallableStatement query = null;
		Connection con = null;
		ResultSet res = null;
		Veiculo result = new Veiculo();

		try {
			con = Conn.getInstance().getConnection();
			con.setAutoCommit(false);
			query = con.prepareCall("{? = call f_getveiculo(?)}");
			query.registerOutParameter(1, Types.OTHER);
```

```
                query.setString(2, placa);

                query.execute();

                res = (ResultSet) query.getObject(1);
                if (res.next()) {
                        result.setAno(res.getInt("ano"));
                        result.setPlaca(res.getString("placa"));
                        result.setModelo(res.getString("modelo"));
                        result.setMarca(res.getString("marca"));
                        result.setCor(res.getString("cor"));
                }
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
        }
        return result;
}
public void setVeiculo(Veiculo veiculo) throws ConsultaDaoException {
        CallableStatement query = null;
        Connection con = null;
        ResultSet res = null;

        try {
                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{call f_insertveiculo(?, ?, ?, ?, ?)}");
                query.setString(1, veiculo.getPlaca());
                query.setInt(2, veiculo.getAno());
                query.setString(3, veiculo.getModelo());
                query.setString(4, veiculo.getCor());
                query.setString(5, veiculo.getMarca());

                query.execute();
                con.commit();

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
        }
}
public void delVeiculo(String placa) throws ConsultaDaoException {
        CallableStatement query = null;
        Connection con = null;
        ResultSet res = null;

        try {
                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{call f_deleteveiculo(?)}");
                query.setString(1, placa);

                query.execute();
                con.commit();
```

```
                } catch (SQLException e) {
                        e.printStackTrace();
                } finally {
                        attemptClose(res);
                        attemptClose(query);
                        attemptClose(con);
                }
        }


}

package br.com.detran.negocio.veiculo;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class VeiculoBO {
        public List<Veiculo> getVeiculos() throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                return usu.getVeiculos();
        }
        public Veiculo getVeiculo(String placa) throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                return usu.getVeiculo(placa);
        }

        public void setVeiculo(Veiculo veiculo) throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                usu.setVeiculo(veiculo);
        }
        public void delVeiculo(String placa) throws ConsultaDaoException {
                VeiculoDao usu = new VeiculoDaoImp();
                usu.delVeiculo(placa);
        }
}

package br.com.detran.negocio.veiculo;

public class Veiculo {

        private int ano;
        private String modelo;
        private String marca;
        private String cor;
        private String uf;
        private String placa;


        public int getAno() {
                return ano;
        }
        public void setAno(int ano) {
                this.ano = ano;
        }
        public String getModelo() {
                return modelo;
        }
        public void setModelo(String modelo) {
```

```
		this.modelo = modelo;
	}
	public String getMarca() {
		return marca;
	}
	public void setMarca(String marca) {
		this.marca = marca;
	}
	public String getCor() {
		return cor;
	}
	public void setCor(String cor) {
		this.cor = cor;
	}
	public String getUf() {
		return uf;
	}
	public void setUf(String uf) {
		this.uf = uf;
	}
	public String getPlaca() {
		return placa;
	}
	public void setPlaca(String placa) {
		this.placa = placa;
	}

}

package br.com.detran.negocio.usuario;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class UsuarioFacade {

	public static List<Usuario> getUsuarios() throws ConsultaDaoException {
		UsuarioBO usu = new UsuarioBO();
		return usu.getUsuarios();
	}
}

package br.com.detran.negocio.usuario;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.apache.struts.actions.DispatchAction;

public class UsuarioDispatchAction extends DispatchAction {

	@Override
	public ActionForward execute(ActionMapping arg0, ActionForm arg1,
			HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
		// TODO Auto-generated method stub
		return super.execute(arg0, arg1, arg2, arg3);
	}
```

```
	@SuppressWarnings("unchecked")
	public ActionForward p(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		frm.getMap().put("lista", UsuarioFacade.getUsuarios());

		return mapping.findForward("lista");
	}
}

package br.com.detran.negocio.usuario;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.List;
import br.com.detran.comum.Dao;
import br.com.detran.conexao.Conn;
import br.com.detran.exception.ConsultaDaoException;

public class UsuarioDaoImp extends Dao implements UsuarioDao {

	public void deleteUsuarios() throws ConsultaDaoException {
	}

	public List<Usuario> getUsuarios() throws ConsultaDaoException {
		CallableStatement query = null;
		Connection con = null;
		ResultSet res = null;
		List<Usuario> result = new ArrayList<Usuario>();

		try {
			con = Conn.getInstance().getConnection();
			con.setAutoCommit(false);
			query = con.prepareCall("{? = call f_getusuarios()}");
			query.registerOutParameter(1, Types.OTHER);

			query.execute();

			res = (ResultSet) query.getObject(1);
			while (res.next()) {
				Usuario usu = new Usuario();
				usu.setCpf(res.getString("cpf"));
				usu.setCpf(res.getString("nome"));
				result.add(usu);
			}
		} catch (SQLException e) {
			e.printStackTrace();
		} finally {
			attemptClose(res);
			attemptClose(query);
			attemptClose(con);
		}
		return result;
```

```
	}

	public void insertUsuarios() throws ConsultaDaoException {
		// TODO Auto-generated method stub

	}

}

package br.com.detran.negocio.usuario;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public interface UsuarioDao {
	public List<Usuario> getUsuarios() throws ConsultaDaoException;
	public void insertUsuarios() throws ConsultaDaoException;
	public void deleteUsuarios() throws ConsultaDaoException;

}

package br.com.detran.negocio.usuario;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class UsuarioBO {
	public List<Usuario> getUsuarios() throws ConsultaDaoException {
		UsuarioDao usu = new UsuarioDaoImp();
		return usu.getUsuarios();
	}
}

package br.com.detran.negocio.usuario;

public class Usuario {

	private String cpf;
	private String nome;

	public String getCpf() {
		return cpf;
	}
	public void setCpf(String cpf) {
		this.cpf = cpf;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}

}

package br.com.detran.negocio.ocorrencia;


public class OcorrenciaMotorista {
```

```
	private int id_blitz;
	private String blitz_nome;
	private String cpf_motorista;
	private String placa_veiculo;
	private String modelo;
	private String obs;
	private boolean delito;
	private String data;

	public int getId_blitz() {
		return id_blitz;
	}
	public void setId_blitz(int id_blitz) {
		this.id_blitz = id_blitz;
	}
	public String getBlitz_nome() {
		return blitz_nome;
	}
	public void setBlitz_nome(String blitz_nome) {
		this.blitz_nome = blitz_nome;
	}
	public String getCpf_motorista() {
		return cpf_motorista;
	}
	public void setCpf_motorista(String cpf_motorista) {
		this.cpf_motorista = cpf_motorista;
	}
	public String getPlaca_veiculo() {
		return placa_veiculo;
	}
	public void setPlaca_veiculo(String placa_veiculo) {
		this.placa_veiculo = placa_veiculo;
	}
	public String getModelo() {
		return modelo;
	}
	public void setModelo(String modelo) {
		this.modelo = modelo;
	}
	public String getObs() {
		return obs;
	}
	public void setObs(String obs) {
		this.obs = obs;
	}
	public boolean getDelito() {
		return delito;
	}
	public void setDelito(boolean delito) {
		this.delito = delito;
	}
	public String getData() {
		return data;
	}
	public void setData(String data) {
		this.data = data;
	}


}
```

```
package br.com.detran.negocio.ocorrencia;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class OcorrenciaFacade {

	public static List<OcorrenciaMotorista> getOcorrenciasMotorista(String cpf) throws
ConsultaDaoException {
		OcorrenciaBO usu = new OcorrenciaBO();
		return usu.getOcorrenciasMotorista(cpf);
	}
	public static List<Ocorrencia> getOcorrencias() throws ConsultaDaoException {
		OcorrenciaBO usu = new OcorrenciaBO();
		return usu.getOcorrencias();
	}
	public static void setOcorrencia(Ocorrencia ocorrencia) throws ConsultaDaoException {
		OcorrenciaBO usu = new OcorrenciaBO();
		usu.setOcorrencia(ocorrencia);

	}
	public static void delOcorrencia(int id) throws ConsultaDaoException {
		OcorrenciaBO usu = new OcorrenciaBO();
		usu.delOcorrencia(id);
	}
}

package br.com.detran.negocio.ocorrencia;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.beanutils.BeanUtils;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.apache.struts.actions.DispatchAction;

import br.com.detran.negocio.blitz.BliitzFacade;
import br.com.detran.negocio.blitz.Blitz;
import br.com.detran.negocio.motorista.Motorista;
import br.com.detran.negocio.motorista.MotoristaFacade;
import br.com.detran.negocio.veiculo.Veiculo;
import br.com.detran.negocio.veiculo.VeiculoFacade;

public class OcorrenciaDispatchAction extends DispatchAction {

	@Override
	public ActionForward execute(ActionMapping arg0, ActionForm arg1,
			HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
		return super.execute(arg0, arg1, arg2, arg3);
	}

	@SuppressWarnings("unchecked")
	public ActionForward p(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {
```

```
		DynaActionForm frm = (DynaActionForm) form;
		frm.getMap().put("lista", OcorrenciaFacade.getOcorrencias());

		return mapping.findForward("lista");
	}
	@SuppressWarnings("unchecked")
	public ActionForward e(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		List<Blitz> lista = BliitzFacade.getBlitzs();
		List<Motorista> listaMotorista = MotoristaFacade.getMotoristas();
		List<Veiculo> listaveiculo = VeiculoFacade.getVeiculos();


		frm.getMap().put("listablitz", lista);
		frm.getMap().put("listamotorista", listaMotorista);
		frm.getMap().put("listaveiculo", listaveiculo);

		return mapping.findForward("insert");
	}

	public ActionForward i(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		Ocorrencia ocorrencia = new Ocorrencia();
		try {
			BeanUtils.copyProperties(ocorrencia, frm);
			OcorrenciaFacade.setOcorrencia(ocorrencia);
		} catch (Exception e) {
			e.printStackTrace();
			return mapping.findForward("erro");
		}
		return mapping.findForward("sucesso");
	}
	@SuppressWarnings("unchecked")
	public ActionForward po(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		try {
			List<OcorrenciaMotorista> lista =
OcorrenciaFacade.getOcorrenciasMotorista(frm.getString("cpf"));
			System.out.println(lista.size());
			frm.getMap().put("listaocorrencia", lista);

		} catch (Exception e) {
			e.printStackTrace();
			return mapping.findForward("erro");
		}
		return mapping.findForward("tableocorrencia");
	}

	public ActionForward d(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
```

```
                    throws Exception {

                DynaActionForm frm = (DynaActionForm) form;
                try {
                    OcorrenciaFacade.delOcorrencia((Integer)frm.get("id"));
                } catch (Exception e) {
                    e.printStackTrace();
                    return mapping.findForward("erro");
                }
                return mapping.findForward("sucesso");
        }
}

package br.com.detran.negocio.ocorrencia;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public interface OcorrenciaDao {
        public List<OcorrenciaMotorista> getOcorrenciasMotorista(String cpf) throws
ConsultaDaoException;
        public List<Ocorrencia> getOcorrencias() throws ConsultaDaoException;
        public void setOcorrencia(Ocorrencia ocorrencia) throws ConsultaDaoException;
        public void delOcorrencia(int id) throws ConsultaDaoException;

}

package br.com.detran.negocio.ocorrencia;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class OcorrenciaBO {
        public List<Ocorrencia> getOcorrencias() throws ConsultaDaoException {
                OcorrenciaDao usu = new OcorrenciaDaoImp();
                return usu.getOcorrencias();
        }
        public List<OcorrenciaMotorista> getOcorrenciasMotorista(String cpf) throws
ConsultaDaoException {
                OcorrenciaDao usu = new OcorrenciaDaoImp();
                return usu.getOcorrenciasMotorista(cpf);
        }
        public void setOcorrencia(Ocorrencia ocorrencia) throws ConsultaDaoException {
                OcorrenciaDao usu = new OcorrenciaDaoImp();
                usu.setOcorrencia(ocorrencia);

        }
        public void delOcorrencia(int id) throws ConsultaDaoException {
                OcorrenciaDao usu = new OcorrenciaDaoImp();
                usu.delOcorrencia(id);
        }

}

package br.com.detran.negocio.ocorrencia;

import java.util.Date;

public class Ocorrencia {
```

```
	private int id ;
	private int id_blitz;
	private String nome_blitz;
	private String cpf_motorista;
	private String placa_veiculo;
	private boolean delito;
	private String obs;
	private String data;

	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public int getId_blitz() {
		return id_blitz;
	}
	public void setId_blitz(int id_blitz) {
		this.id_blitz = id_blitz;
	}
	public String getCpf_motorista() {
		return cpf_motorista;
	}
	public void setCpf_motorista(String cpf_motorista) {
		this.cpf_motorista = cpf_motorista;
	}
	public String getPlaca_veiculo() {
		return placa_veiculo;
	}
	public void setPlaca_veiculo(String placa_veiculo) {
		this.placa_veiculo = placa_veiculo;
	}
	public boolean isDelito() {
		return delito;
	}
	public void setDelito(boolean delito) {
		this.delito = delito;
	}
	public String getObs() {
		return obs;
	}
	public void setObs(String obs) {
		this.obs = obs;
	}
	public String getNome_blitz() {
		return nome_blitz;
	}
	public void setNome_blitz(String nome_blitz) {
		this.nome_blitz = nome_blitz;
	}
	public String getData() {
		return data;
	}
	public void setData(String data) {
		this.data = data;
	}

}
```

```
package br.com.detran.negocio.motorista;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public interface MotoristaDao {
	public List<Motorista> getMotoristas() throws ConsultaDaoException;
	public Motorista getMotorista(String cpf, String caminho) throws ConsultaDaoException;
	public void setMotorista(Motorista motorista) throws ConsultaDaoException;
	public void delMotorista(String cpf) throws ConsultaDaoException;

}

package br.com.detran.negocio.motorista;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class MotoristaFacade {

	public static List<Motorista> getMotoristas() throws ConsultaDaoException {
		MotoristaBO usu = new MotoristaBO();
		return usu.getMotoristas();
	}

	public static Motorista getMotorista(String cpf, String caminho) throws ConsultaDaoException {
		MotoristaBO usu = new MotoristaBO();
		return usu.getMotorista(cpf, caminho);
	}

	public static void setMotorista(Motorista motorista) throws ConsultaDaoException {
		MotoristaBO usu = new MotoristaBO();
		usu.setMotorista(motorista);
	}
	public static void delMotorista(String cpf) throws ConsultaDaoException {
		MotoristaBO usu = new MotoristaBO();
		usu.delMotorista(cpf);
	}
	public static String findMotorista(String hashConsulta)  {
		MotoristaBO usu = new MotoristaBO();
		return usu.findMotorista(hashConsulta);
	}

}

package br.com.detran.negocio.motorista;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.beanutils.BeanUtils;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.apache.struts.actions.DispatchAction;

import br.com.detran.negocio.blitz.BliitzFacade;

import com.mchange.v2.beans.BeansUtils;
```

```
public class MotoristaDispatchAction extends DispatchAction {

	@Override
	public ActionForward execute(ActionMapping arg0, ActionForm arg1,
			HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
		// TODO Auto-generated method stub
		return super.execute(arg0, arg1, arg2, arg3);
	}

	@SuppressWarnings("unchecked")
	public ActionForward p(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		frm.getMap().put("lista", MotoristaFacade.getMotoristas());

		return mapping.findForward("lista");
	}
	public ActionForward e(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;

		if (frm.get("cpf")!=null) {
			if (!frm.getString("cpf").equals(""))
				BeanUtils.copyProperties(frm,
MotoristaFacade.getMotorista(frm.getString("cpf"), request.getRealPath("")));
		}
		return mapping.findForward("edit");
	}
	public ActionForward m(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;

		if (frm.get("cpf")!=null) {
			if (!frm.getString("cpf").equals(""))
				BeanUtils.copyProperties(frm,
MotoristaFacade.getMotorista(frm.getString("cpf"), request.getRealPath("")));
		}
		return mapping.findForward("motorista");
	}

	public ActionForward i(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		Motorista motorista = new Motorista();
		try {
			BeanUtils.copyProperties(motorista, frm);
			MotoristaFacade.setMotorista(motorista);
		} catch (Exception e) {
			e.printStackTrace();
			return mapping.findForward("erro");
		}
```

```
			return mapping.findForward("sucesso");
		}

		public ActionForward d(ActionMapping mapping, ActionForm form,
				HttpServletRequest request, HttpServletResponse response)
				throws Exception {

			DynaActionForm frm = (DynaActionForm) form;
			try {
				MotoristaFacade.delMotorista(frm.getString("cpf"));
			} catch (Exception e) {
				e.printStackTrace();
				return mapping.findForward("erro");
			}
			return mapping.findForward("sucesso");
		}
		public ActionForward fu(ActionMapping mapping, ActionForm form,
				HttpServletRequest request, HttpServletResponse response)
				throws Exception {

			DynaActionForm frm = (DynaActionForm) form;
			try {
				frm.set("cpf", MotoristaFacade.findMotorista(frm.getString("hash")));
			} catch (Exception e) {
				e.printStackTrace();
				return mapping.findForward("erro");
			}
			return mapping.findForward("cpf");
		}
}

package br.com.detran.negocio.motorista;

import java.io.File;
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.List;

import br.com.detran.comum.Dao;
import br.com.detran.conexao.Conn;
import br.com.detran.exception.ConsultaDaoException;

public class MotoristaDaoImp extends Dao implements MotoristaDao {

		public List<Motorista> getMotoristas() throws ConsultaDaoException {
			CallableStatement query = null;
			Connection con = null;
			ResultSet res = null;
			List<Motorista> result = new ArrayList<Motorista>();



			try {
				con = Conn.getInstance().getConnection();
				con.setAutoCommit(false);
				query = con.prepareCall("{? = call f_getmotoristas()}");
```

```
                query.registerOutParameter(1, Types.OTHER);

                query.execute();

                res = (ResultSet) query.getObject(1);
                while (res.next()) {
                        Motorista moto = new Motorista();
                        moto.setCpf(res.getString(                      "cpf"));
                        moto.setNome(res.getString(                     "nome"));
                        moto.setRg(res.getString(                       "rg"));
                        moto.setData_nascimento(
res.getString("data_nascimento"));
                        moto.setCnh(res.getString(                      "cnh"));
                        moto.setCategoria(res.getString( "categoria"));
                        moto.setDedo1(res.getString(            "dedo1"));
                        moto.setDedo2(res.getString(            "dedo2"));
                        moto.setDedo3(res.getString(            "dedo3"));
                        moto.setData_validade(
res.getString("data_validade"));
                        moto.setData_emissao(
res.getString("data_emissao"));
                        moto.setOrgao_emissor(res.getString("orgao_emissor"));
                        moto.setUf(res.getString("uf"));
                        moto.setFiliacao(res.getString("filiacao"));
                        moto.setObservacao(res.getString("observacao"));
                        moto.setPermissao(res.getString("permissao"));
                        moto.setAcc(res.getString("acc"));
                        moto.setP_habilitacao(res.getString("p_habilitacao"));
                        moto.setLocal(res.getString("local"));

                        result.add(moto);
                }
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
        }
        return result;
}
public Motorista getMotorista(String cpf, String caminho) throws ConsultaDaoException {
        CallableStatement query = null;
        Connection con = null;
        ResultSet res = null;
        Motorista moto = new Motorista();
        try {
                File file = new File(caminho+"/jsp/foto/foto.jpg");
                if(file.exists())
                        file.delete();

                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{? = call f_getmotorista(?, ?)}");
                query.registerOutParameter(1, Types.OTHER);
                query.setString(2, cpf);
                query.setString(3, caminho.replaceAll("\\\\","/")+"/jsp/foto/");

                query.execute();
```

```
                    res = (ResultSet) query.getObject(1);
                    if (res.next()) {
                            moto.setCpf(res.getString(                  "cpf"));
                            moto.setNome(res.getString(                 "nome"));
                            moto.setRg(res.getString(                   "rg"));
                            moto.setData_nascimento(
        res.getString("data_nascimento"));
                            moto.setCnh(res.getString(                  "cnh"));
                            moto.setCategoria(res.getString( "categoria"));
                            moto.setDedo1(res.getString(          "dedo1"));
                            moto.setDedo2(res.getString(          "dedo2"));
                            moto.setDedo3(res.getString(          "dedo3"));
                            moto.setData_validade(
        res.getString("data_validade"));
                            moto.setData_emissao(
        res.getString("data_emissao"));
                            moto.setOrgao_emissor(res.getString("orgao_emissor"));
                            moto.setUf(res.getString("uf"));
                            moto.setFiliacao(res.getString("filiacao"));
                            moto.setObservacao(res.getString("observacao"));
                            moto.setAcc(res.getString("acc"));
                            moto.setP_habilitacao(res.getString("p_habilitacao"));
                            moto.setLocal(res.getString("local"));
                    }
            } catch (SQLException e) {
                    e.printStackTrace();
            } finally {
                    attemptClose(res);
                    attemptClose(query);
                    attemptClose(con);
            }
            return moto;
    }
    public void setMotorista(Motorista motorista) throws ConsultaDaoException {
            CallableStatement query = null;
            Connection con = null;
            ResultSet res = null;


            try {
                    con = Conn.getInstance().getConnection();
                    con.setAutoCommit(false);
                    query = con.prepareCall("{call f_insertmotoristas(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?)}");



                    query.setString(1,  motorista.getCpf());
                    query.setString(2,  motorista.getNome());
                    query.setString(3,  motorista.getRg());
                    // passando null para o rg
                    query.setString(4,  motorista.getData_nascimento());
                    query.setString(5,  motorista.getCnh());
                    query.setString(6,  motorista.getCategoria());
                    query.setString(7,  motorista.getDedo1());
                    query.setString(8,  motorista.getDedo2());
                    query.setString(9,  motorista.getDedo3());
                    query.setString(10, motorista.getData_validade());
                    query.setString(11, motorista.getData_emissao());
                    query.setString(12, motorista.getOrgao_emissor());
```

```
                query.setString(13, motorista.getUf());
                query.setString(14, motorista.getFiliacao());
                query.setString(15, motorista.getObservacao());
                query.setString(16, motorista.getPermissao());
                query.setString(17, motorista.getAcc());
                query.setString(18, motorista.getP_habilitacao());
                query.setString(19, motorista.getLocal());
                query.setString(20, motorista.getFoto().replaceAll("\\\\","/"));


                query.execute();
                con.commit();

            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
            }
        }
        public void delMotorista(String cpf) throws ConsultaDaoException {
            CallableStatement query = null;
            Connection con = null;
            ResultSet res = null;

            try {
                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{call f_deletemotoristas(?)}");
                query.setString(1, cpf);

                query.execute();
                con.commit();

            } catch (SQLException e) {
                e.printStackTrace();
            } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
            }
        }


}

package br.com.detran.negocio.motorista;

import java.util.List;

import br.com.detran.conexao.Finger;
import br.com.detran.exception.ConsultaDaoException;

import com.griaule.grFinger.Context;
import com.griaule.grFinger.FingerprintTemplate;
import com.griaule.grFinger.GrErrorException;
import com.griaule.grFinger.MatchingResult;
import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
```

```
public class MotoristaBO {
	public List<Motorista> getMotoristas() throws ConsultaDaoException {
		MotoristaDao usu = new MotoristaDaoImp();
		return usu.getMotoristas();
	}
	public Motorista getMotorista(String cpf, String caminho) throws ConsultaDaoException {
		MotoristaDao usu = new MotoristaDaoImp();
		return usu.getMotorista(cpf, caminho);
	}
	public void setMotorista(Motorista motorista) throws ConsultaDaoException {
		MotoristaDao usu = new MotoristaDaoImp();
		usu.setMotorista(motorista);
	}
	public void delMotorista(String cpf) throws ConsultaDaoException {
		MotoristaDao usu = new MotoristaDaoImp();
		usu.delMotorista(cpf);
	}
  public String findMotorista(String hashConsulta)  {
	List<Motorista> lista;
	String result = null;
		try {
			lista = MotoristaFacade.getMotoristas();
			byte[] tempbit = Base64.decode(hashConsulta);
			FingerprintTemplate template = new FingerprintTemplate(tempbit,
tempbit.length);
		for (Motorista motorista : lista) {
			byte[] bit = Base64.decode(motorista.getDedo1());
			byte[] bit2 = Base64.decode(motorista.getDedo2());
			byte[] bit3= Base64.decode(motorista.getDedo3());


			FingerprintTemplate teste = new FingerprintTemplate(bit, bit.length);
			MatchingResult ver = Finger.getInstance().getFigner().verify(template, teste,
Context.DEFAULT_CONTEXT);
			if (ver.doesMatched()) {
				return motorista.getCpf();
			}
			teste = new FingerprintTemplate(bit2, bit.length);
			ver = Finger.getInstance().getFigner().verify(template, teste,
Context.DEFAULT_CONTEXT);
			if (ver.doesMatched()) {
				return motorista.getCpf();
			}
			teste = new FingerprintTemplate(bit3, bit.length);
			ver = Finger.getInstance().getFigner().verify(template, teste,
Context.DEFAULT_CONTEXT);
			if (ver.doesMatched()) {
				return motorista.getCpf();
			}

			}
		} catch (ConsultaDaoException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (GrErrorException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}// errado... deixar na clausula trhows
	return result;
  }
```

```
}

package br.com.detran.negocio.motorista;

import java.util.Date;

public class Motorista {

	private String	cpf;
	private String	nome;
	private String	rg;
	private String	data_nascimento;
	private String	cnh;
	private String	categoria;
	private String	dedo1;
	private String	dedo2;
	private String	dedo3;
	private String	data_validade;
	private String	data_emissao;
	private String	orgao_emissor;
	private String	uf;
	private String	filiacao;
	private String	observacao;
	private String	permissao;
	private String	acc;
	private String foto;
	private String	p_habilitacao;
	private String	local;

	private String pastaImg;

	public String getCpf() {
		return cpf;
	}
	public void setCpf(String cpf) {
		this.cpf = cpf;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getRg() {
		return rg;
	}
	public void setRg(String rg) {
		this.rg = rg;
	}
	public String getData_nascimento() {
		return data_nascimento;
	}
	public void setData_nascimento(String data_nascimento) {
		this.data_nascimento = data_nascimento;
	}
	public String getCnh() {
		return cnh;
	}
	public void setCnh(String cnh) {
```

```
            this.cnh = cnh;
}
public String getCategoria() {
            return categoria;
}
public void setCategoria(String categoria) {
            this.categoria = categoria;
}
public String getDedo1() {
            return dedo1;
}
public void setDedo1(String dedo1) {
            this.dedo1 = dedo1;
}
public String getDedo2() {
            return dedo2;
}
public void setDedo2(String dedo2) {
            this.dedo2 = dedo2;
}
public String getDedo3() {
            return dedo3;
}
public void setDedo3(String dedo3) {
            this.dedo3 = dedo3;
}
public String getData_validade() {
            return data_validade;
}
public void setData_validade(String data_validade) {
            this.data_validade = data_validade;
}
public String getData_emissao() {
            return data_emissao;
}
public void setData_emissao(String data_emissao) {
            this.data_emissao = data_emissao;
}
public String getOrgao_emissor() {
            return orgao_emissor;
}
public void setOrgao_emissor(String orgao_emissor) {
            this.orgao_emissor = orgao_emissor;
}
public String getUf() {
            return uf;
}
public void setUf(String uf) {
            this.uf = uf;
}
public String getFiliacao() {
            return filiacao;
}
public void setFiliacao(String filiacao) {
            this.filiacao = filiacao;
}
public String getObservacao() {
            return observacao;
}
public void setObservacao(String observacao) {
```

```
		this.observacao = observacao;
	}
	public String getPermissao() {
		return permissao;
	}
	public void setPermissao(String permissao) {
		this.permissao = permissao;
	}
	public String getAcc() {
		return acc;
	}
	public void setAcc(String acc) {
		this.acc = acc;
	}
	public String getP_habilitacao() {
		return p_habilitacao;
	}
	public void setP_habilitacao(String p_habilitacao) {
		this.p_habilitacao = p_habilitacao;
	}
	public String getLocal() {
		return local;
	}
	public void setLocal(String local) {
		this.local = local;
	}
	public String getFoto() {
		return foto;
	}
	public void setFoto(String foto) {
		this.foto = foto;
	}
	public String getPastaImg() {
		return pastaImg;
	}
	public void setPastaImg(String pastaImg) {
		this.pastaImg = pastaImg;
	}

}

package br.com.detran.negocio.blitz;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.beanutils.BeanUtils;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
import org.apache.struts.action.DynaActionForm;
import org.apache.struts.actions.DispatchAction;

public class BlitzDispatchAction extends DispatchAction  {

	@Override
	public ActionForward execute(ActionMapping arg0, ActionForm arg1,
			HttpServletRequest arg2, HttpServletResponse arg3) throws Exception {
		// TODO Auto-generated method stub
		return super.execute(arg0, arg1, arg2, arg3);
```

```
	}

	@SuppressWarnings("unchecked")
	public ActionForward p(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		frm.getMap().put("lista", BliitzFacade.getBlitzs());

		return mapping.findForward("lista");
	}

	public ActionForward e(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;

		if (frm.get("id")!=null) {
			BeanUtils.copyProperties(frm, BliitzFacade.getBlitz((Integer)frm.get("id")));
		}

		return mapping.findForward("edit");
	}

	public ActionForward i(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		Blitz blitz = new Blitz();
		try {
			BeanUtils.copyProperties(blitz, frm);
			BliitzFacade.setBlitz(blitz);
		} catch (Exception e) {
			e.printStackTrace();
			return mapping.findForward("erro");
		}
		return mapping.findForward("sucesso");
	}

	public ActionForward d(ActionMapping mapping, ActionForm form,
			HttpServletRequest request, HttpServletResponse response)
			throws Exception {

		DynaActionForm frm = (DynaActionForm) form;
		try {
			BliitzFacade.delBlitz((Integer)frm.get("id"));
		} catch (Exception e) {
			e.printStackTrace();
			return mapping.findForward("erro");
		}
		return mapping.findForward("sucesso");
	}


}

package br.com.detran.negocio.blitz;
```

```
public class Blitz {

	private int id;
	private String nome;
	private String local;
	private String descricao;



	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
	public String getDescricao() {
		return descricao;
	}
	public void setDescricao(String descricao) {
		this.descricao = descricao;
	}
	public String getLocal() {
		return local;
	}
	public void setLocal(String local) {
		this.local = local;
	}

}

package br.com.detran.negocio.blitz;

import java.util.List;
import br.com.detran.exception.ConsultaDaoException;

public class BliitzFacade {

	public static List<Blitz> getBlitzs() throws ConsultaDaoException {
		BliitzBO usu = new BliitzBO();
		return usu.getBlitzs();
	}

	public static Blitz getBlitz(int id) throws ConsultaDaoException {
		BliitzBO usu = new BliitzBO();
		return usu.getBlitz(id);
	}

	public static void setBlitz(Blitz blitz) throws ConsultaDaoException {
		BliitzDao usu = new BliitzDaoImp();
		usu.setBlitz(blitz);
	}
	public static void delBlitz(int id) throws ConsultaDaoException {
```

```
                BliitzDao usu = new BliitzDaoImp();
                usu.delBlitz(id);
        }
}

package br.com.detran.negocio.blitz;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.List;

import br.com.detran.comum.Dao;
import br.com.detran.conexao.Conn;
import br.com.detran.exception.ConsultaDaoException;

public class BliitzDaoImp extends Dao implements BliitzDao {

        public List<Blitz> getBlitzs() throws ConsultaDaoException {
                CallableStatement query = null;
                Connection con = null;
                ResultSet res = null;
                List<Blitz> result = new ArrayList<Blitz>();

                try {
                        con = Conn.getInstance().getConnection();
                        con.setAutoCommit(false);
                        query = con.prepareCall("{? = call f_getblitzs()}");
                        query.registerOutParameter(1, Types.OTHER);

                        query.execute();

                        res = (ResultSet) query.getObject(1);
                        while (res.next()) {
                                Blitz bli = new Blitz();
                                bli.setId(res.getInt(                   "id"));
                                bli.setNome(res.getString(                      "nome"));

                                result.add(bli);
                        }
                } catch (SQLException e) {
                        e.printStackTrace();
                } finally {
                        attemptClose(res);
                        attemptClose(query);
                        attemptClose(con);
                }
                return result;
        }
        public Blitz getBlitz(int id) throws ConsultaDaoException {
                CallableStatement query = null;
                Connection con = null;
                ResultSet res = null;
                Blitz result = new Blitz();

                try {
                        con = Conn.getInstance().getConnection();
```

```
                con.setAutoCommit(false);
                query = con.prepareCall("{? = call f_getblitz(?)}");
                query.registerOutParameter(1, Types.OTHER);
                query.setInt(2, id);

                query.execute();

                res = (ResultSet) query.getObject(1);
                if (res.next()) {
                        result.setId(res.getInt("id"));
                        result.setNome(res.getString("nome"));
                        result.setLocal(res.getString("local"));
                        result.setDescricao(res.getString("descricao"));

                }
        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
        }
        return result;
}
public void setBlitz(Blitz blitz) throws ConsultaDaoException {
        CallableStatement query = null;
        Connection con = null;
        ResultSet res = null;

        try {
                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{call f_insertblitz(?, ?, ?, ?)}");

                query.setInt(1, blitz.getId());
                query.setString(2, blitz.getNome());
                query.setString(3, blitz.getLocal());
                query.setString(4, blitz.getDescricao());

                query.execute();
                con.commit();

        } catch (SQLException e) {
                e.printStackTrace();
        } finally {
                attemptClose(res);
                attemptClose(query);
                attemptClose(con);
        }
}
public void delBlitz(int id) throws ConsultaDaoException {
        CallableStatement query = null;
        Connection con = null;
        ResultSet res = null;

        try {
                con = Conn.getInstance().getConnection();
                con.setAutoCommit(false);
                query = con.prepareCall("{call f_deleteblitz(?)}");
                query.setInt(1, id);
```

```
                        query.execute();
                        con.commit();

                } catch (SQLException e) {
                        e.printStackTrace();
                } finally {
                        attemptClose(res);
                        attemptClose(query);
                        attemptClose(con);
                }
        }


}

package br.com.detran.negocio.blitz;

import java.util.List;

import br.com.detran.exception.ConsultaDaoException;

public interface BliitzDao {
        public List<Blitz> getBlitzs() throws ConsultaDaoException;
        public Blitz getBlitz(int id) throws ConsultaDaoException;
        public void setBlitz(Blitz blitz) throws ConsultaDaoException;
        public void delBlitz(int id) throws ConsultaDaoException;

}

package br.com.detran.conexao;
import com.griaule.grFinger.GrErrorException;
import com.griaule.grFinger.GrFinger;

public class Finger  {

        private static Finger _instance;
        private static GrFinger finger     =  null;

        private Finger() throws GrErrorException {
                finger = new GrFinger();
        }

        public GrFinger getFigner() throws GrErrorException {
                        return finger;
        }

        public static Finger getInstance() throws GrErrorException {
                if (_instance == null) {
                        _instance = new Finger();
                }
                return _instance;
        }
        public Object clone() throws CloneNotSupportedException {
                throw new CloneNotSupportedException();
        }
}

package br.com.detran.exception;
```

```
public class ConsultaDaoException extends Exception {
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;
	private static final String message = "Erro ao acessar base de dados";

	public ConsultaDaoException() {
		super(message);
	}

	public ConsultaDaoException(String arg0) {
		super(message + ": " + arg0);
	}

	public ConsultaDaoException(Throwable arg0) {
		super(arg0);
	}

	public ConsultaDaoException(String arg0, Throwable arg1) {
		super(message + ": " + arg0, arg1);
	}
}

package br.com.detran.comum;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;

public class Dao {

	protected static void attemptClose(ResultSet o) {
		try {
			if (o != null)
				o.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	protected static void attemptClose(Statement o) {
		try {
			if (o != null)
				o.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}

	protected void attemptClose(Connection o) {
		try {
			if (o != null)
				o.close();
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

```
package br.com.detran.conexao;
import java.sql.Connection;
import java.sql.SQLException;
import javax.sql.DataSource;
import com.mchange.v2.c3p0.DataSources;

public class Conn  {

	private static Conn conn;
	private static DataSource pooled		=  null;

	private Conn() throws SQLException {
		pooled = DataSources.pooledDataSource(
DataSources.unpooledDataSource("jdbc:postgresql://localhost:5432/detran","detran","detran"));
	}

	public Connection getConnection() throws SQLException {
			return pooled.getConnection();
	}

	public static Conn getInstance() throws SQLException {
		if (conn == null) {
			conn = new Conn();
		}
		return conn;
	}
	public Object clone() throws CloneNotSupportedException {
		throw new CloneNotSupportedException();
	}
}
```