FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" – UNIVEM
TRABALHO DE CONCLUSÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

DANILO QUIQUINATO PINHEIRO DA SILVEIRA

SISTEMA PARA AQUISIÇÃO E TRANSFERÊNCIA SEGURA DE DADOS DESTINADOS À ANÁLISE CALIGRÁFICA

MARÍLIA
2006

DANILO QUIQUINATO PINHEIRO DA SILVEIRA

SISTEMA PARA AQUISIÇÃO E TRANSFERÊNCIA SEGURA DE DADOS DESTINADOS À ANÁLISE CALIGRÁFICA

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha.

Orientador:
Prof. Dr. Márcio Eduardo Delamaro

MARÍLIA
2006

DANILO QUIQUINATO PINHEIRO DA SILVEIRA

SISTEMA PARA AQUISIÇÃO E TRANSFERÊNCIA SEGURA DE DADOS DESTINADOS À ANÁLISE CALIGRÁFICA

Banca examinadora da dissertação apresentada ao Programa de Graduação da UNIVEM,/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação

Resultado: _____

ORIENTADOR: Prof. Dr. Márcio Eduardo Delamaro

1º EXAMINADOR:_____

2º EXAMINADOR:_____

Marília, 16 de Junho de 2006.

Dedico este trabalho aos meus 4 anos de estudo intensivo no Curso de Ciência da Computação da UNIVEM, onde pude desenvolver meus conhecimentos, minha inteligência, minha força de vontade, minha paciência e outras virtudes que me impulsionam.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais pela oportunidade oferecida para me aprimorar e desenvolver meus conhecimentos e a confiança plena na minha capacidade e na minha disposição em aprender.

Em seguida e não distante, à minha companheira Aline, pela paciência e compreensão aos tempos que tive que me ausentar de sua presença para desenvolver esse trabalho.

Agradeço aos meus amigos de trabalho que me apoiaram e participaram do desenvolvimento desse projeto.

Agradeço também aos amigos de classe Rodrigo Zapaterra, Thiago, Hugo, Marco, Thalita, Rita e Mariana Feres e a todos que estiveram participando desses meus 4 anos de faculdade.

Aos professores que me estimularam.

Fixo meus agradecimentos neste trabalho para que eu possa demonstrar de uma maneira simples, mas sincera, a minha felicidade de ter convivido com eles durante esses tempos e que eu possa continuar o quanto for possível convivendo e aprendendo com todos.

Que assim Seja!

SILVEIRA, Danilo Quiquinato Pinheiro da. **Sistema para Aquisição e Transferência Segura de Dados Destinados à Análise Caligráfica**. 2006. 48 f. Monografia (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

RESUMO

Este projeto tem como objetivo desenvolver um sistema para facilitar e tornar segura a aquisição de assinaturas destinadas à análise caligráfica. A aquisição é feita através de dispositivos móveis Palm®, que permitem a digitalização de imagens, no caso assinaturas, através de telas sensíveis ao toque. Essas assinaturas são enviadas a um servidor, protegido pelo protocolo SSL, o qual ficará responsável por armazená-las para que possam ser utilizadas para testar a fidelidade do algoritmo encarregado da análise caligráfica.

Palavras-chave: Assinatura, Análise Caligráfica, SSL, PALM, Biometria.

SILVEIRA, Danilo Quiquinato Pinheiro da. **Sistema para Aquisição e Transferência Segura de Dados Destinados à Análise Caligráfica**. 2006. 48 f. Monografia (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2006.

ABSTRACT

The goal of this project is to develop a system to facilitate and make safe the acquisition of signatures to analysis. The acquisition is done through mobile devices, that allow the server capturing the images in sensitive screens. These signatures are sent to a server protected by the SSL protocol and are stored to be used in experiments that will be necessary to test the efficiency of the algorithm of signature analysis.

Keywords: Signature, Signature Analysis, SSL, PALM, Biometry.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

Análise Caligráfica é uma das técnicas de biometria utilizadas como forma de identificar as pessoas. Essa técnica é baseada na premissa de que cada indivíduo é único e possui características próprias que são impressas na simples ação de escrever manualmente símbolos que possam identificá-lo, como o seu próprio nome.

Na biometria é possível encontrar outros métodos muito utilizados, como a análise de retina, íris, impressão digital e face, que apesar de não sofrerem muitas alterações durante o tempo, exigem tecnologias de difícil aquisição, ao contrário da análise caligráfica, que não necessita de avançadas formas de aquisição. Uma outra vantagem é o fato de ser uma técnica utilizada há décadas e bem aceita na sociedade, enquanto as análises de retina e de impressão digital estão associadas às investigações criminais. (ANIL, 1999)

Com uma grande quantidade de repetições de sua assinatura, o indivíduo acaba criando os seus próprios padrões, que são melhorados e fixados com o passar do tempo, definindo formas e desenhos particulares. Entretanto, algumas características são muito passivas de alterações, como a mudança do objeto utilizado na escrita, a superfície em que ela é desenhada, a posição da pessoa no momento da escrita, a velocidade, o estado emocional, etc.

Com todos esses fatores interferindo na assinatura, a gama de variações torna-se uma barreira para os especialistas conseguirem identificar o autor, fazendo com que imitações sejam aceitas erroneamente. Estudando e treinando uma assinatura é possível reproduzi-la, e isso torna a análise caligráfica ineficaz.

O objetivo principal do algoritmo desenvolvido pelo Prof. Dr. Márcio Eduardo Delamaro é extrair o maior número de padrões que possam tornar a análise mais precisa e viável para a utilização em sistemas de identificação.

Neste projeto, foi implementado um sistema que permite a aquisição e transferência segura dessas assinaturas por meio da Internet, garantindo que os dados cheguem para análise, intactos e protegidos de qualquer interferência.

Foram feitos testes para garantir que a calibragem do algoritmo seja sensível à imitação de padrões, evitando a falsificação de assinaturas.

O intuito do projeto é viabilizar uma aplicação que permita a análise caligráfica ser incorporada a ambientes onde essa tecnologia ainda não tem acesso, como a Internet, tornando-a mais confiável, sem precisar de muitas informações para que a identidade do usuário seja reconhecida.

Além desta introdução, esta monografia é composta da seguinte forma:

- O Capítulo 2 faz uma introdução sobre a análise caligráfica, iniciando com uma explicação sobre a sua necessidade, como os padrões são identificados e o algoritmo utilizado nas comparações;
- No Capítulo 3 é feita uma descrição conceitual sobre as tecnologias utilizadas no projeto;
- O Capítulo 4 apresenta detalhes do sistema desenvolvido, dando ênfase na estrutura de implementação;
- No Capítulo 5 é realizado um conjunto inicial de testes para mostrar o funcionamento do projeto e obter uma análise simples do resultado.
- O Capítulo 6 apresenta considerações finais sobre o projeto fazendo uma conclusão sobre o resultado dos testes e comentando sobre a viabilidade do sistema.

CAPÍTULO 2 – CONTEXTO

Neste capítulo são tratados alguns aspectos importantes sobre a análise caligráfica, iniciando com a necessidade de utilização de sistemas que confirmem a origem das informações, passando pelos métodos já existentes de autorização e certificação na Internet, fazendo, em seguida, um estudo sobre o funcionamento lógico do sistema.

2.1. Segurança na Internet

Em meio à grande evolução tecnológica que a Internet trouxe para a sociedade, alguns aspectos se tornam preocupantes ao se tratar de segurança.

A impossibilidade de identificar o usuário que está envolvido nas ações gerou a necessidade de criação de métodos que certifiquem a autenticidade das solicitações (FERRO, 2006). Para isso foram desenvolvidos sistemas complexos que solicitam uma gama enorme de informações e cadastros contendo informações únicas, como:

- CPF;
- RG;
- Códigos Combinatórios;
- Senhas;
- Frases e respostas que devem coincidir com o padrão cadastrado.

Analisando sistemas bancários, que disponibilizam serviços *on-line* como transferências, depósitos e consultas, verifica-se que para se conseguir efetuar uma transferência é solicitado, por exemplo, o código do banco, agência, senha para acesso exclusivo on-line, frases. Há, ainda instituições que geram um número aleatório cujo código deve ser procurado em um cartão fornecido no momento do cadastro, contendo

aproximadamente 100 números. Com todas essas informações, o processo torna-se lento e cansativo, exigindo muita paciência do usuário e ainda assim podem ocorrer violações como senhas e cartões roubados e frases de fácil suposição.

A Tabela 1 sintetiza algumas das ameaças que estão envolvidas no ambiente da Internet, mostrando as conseqüências e as medidas que devem ser usadas para evitá-las. A personificação de usuários legítimos e a falsificação de dados estão entre as ameaças de autenticação conforme a Tabela 1, citada por Figueiredo (2003).

Tabela 1- Principais Formas de Ameaças

	Integridade	**Confidencialidade**	**Negação de Serviço**	**Autenticação**
Ameaças	- modificação de dados do usuário - browser cavalo de Tróia - modificação de memória - modificação de mensagens em trânsito	- *eavesdropping* (interceptação de conversas) - roubo de info/dado do servidor/cliente - informação da configuração da rede/máquinas... - informação de qual cliente "conversa" com servidor em trânsito	- bloqueio da conexão - inundação da máquina com solicitações bogus - isolamento da máquina por ataques a DNS	**- personificação de usuários legítimos** **- falsificação de dados**
Conseqüências	- perda de informação - compromete a máquina - vulnerabilidade para outras ameaças	- perda de informação - perda de privacidade	- interrupção -aborrecimento - impedir usuário realizar seu trabalho	- má representação do usuário - crença que informação falsa é verdadeira
Medidas	- *checksums* criptográfico	- encriptação, *Web proxies*	- difícil prevenir	- técnicas criptográficas

Esse sistema complexo de validação por informações é uma tentativa de suprir a falta de métodos eficientes para certificação como análise caligráfica, impressões digitais, reconhecimentos de íris e de retina.

Quanto menor o número de informações durante a autenticação, mais fácil será para o usuário efetuar as operações e menor o tempo de conclusão. Se os métodos fossem poucos,

porém confiáveis, haveria uma maior utilização de ambientes *on-line* para transferências de risco.

2.2. Análise Caligráfica

A análise caligráfica ou de assinatura pode ser dividida em dois tipos, que variam conforme a maneira de aquisição utilizada. São eles: *off-line* e *on-line* (ANIL, 1999).

O método *off-line* consiste em utilizar um *scanner* 2D (Figura 1) para adquirir uma assinatura que tenha sido escrita em um documento. Já o método *on-line* necessita de uma mesa digitalizadora (Figura 2) ou um *hardware* que contenha uma tela sensível ao toque para efetuar a captação da assinatura (Figura 3).

Figura 1 – Aquisição *off-line* (Scanner Epson)

Figura 2 – Aquisição *on-line* (Mesa Digitalizadora Wacom)

Figura 3 – Aquisição *on-line* (Palm E2)

O método *on-line* se sobressai ao método *off-line*, pois permitem capturar a seqüência dos movimentos, a direção da escrita, o início e o fim dos traços, dados importantes para análise dos padrões e que dificultam a fraude.

Na Figura 4 é mostrada uma outra característica analisada pelo método *on-line*, que é a verificação no eixo Z.

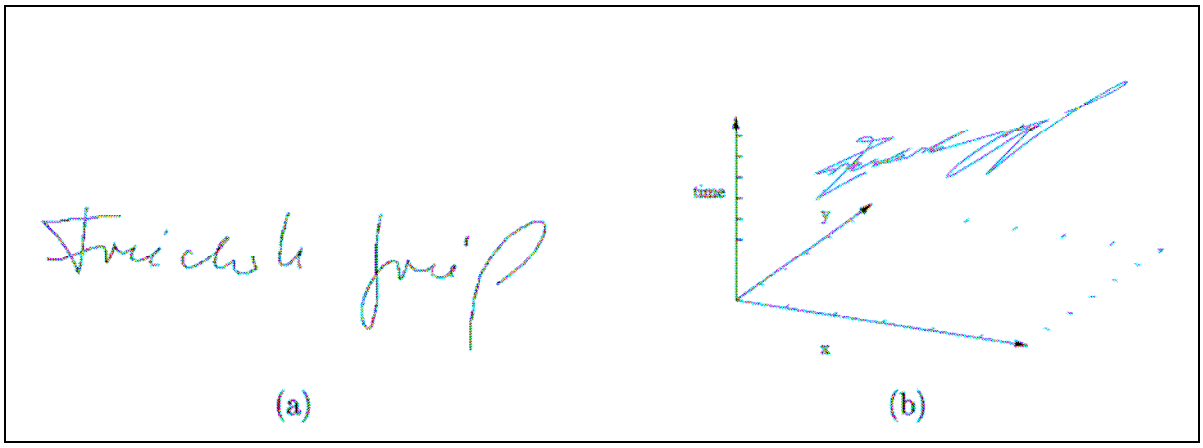

Figura 4 – Aquisição *off-line* (a) e *on-line* (b)

As assinaturas são extraídas ponto-a-ponto, gerando cadeias de caracteres que são submetidas ao algoritmo desenvolvido pelo Prof. Dr. Márcio Eduardo Delamaro.

Para poder avaliar o algoritmo, é preciso identificar dois tipos de erros: o número de assinaturas originais que são recusadas pelo sistema incorretamente, chamado taxa de rejeição falsa e o número de assinaturas falsificadas que são aceitas, chamado taxa de aceitação falsa. Essas taxas devem ser balanceadas para que o sistema seja eficiente.

Esse balanceamento é feito conforme o nível de segurança exigido no ambiente onde o sistema é aplicado. Em locais onde a segurança deve ser mais alta, o algoritmo deve aceitar uma taxa mínima de erro. (ANIL, 1999)

Para que a análise possa ser feita é necessário extrair as principais características da assinatura. Essas características são, por exemplo, as distâncias entre dois pontos consecutivos e o ângulo de curvatura. Na Figura 5 e Figura 6 é possível verificar alguns desses pontos.

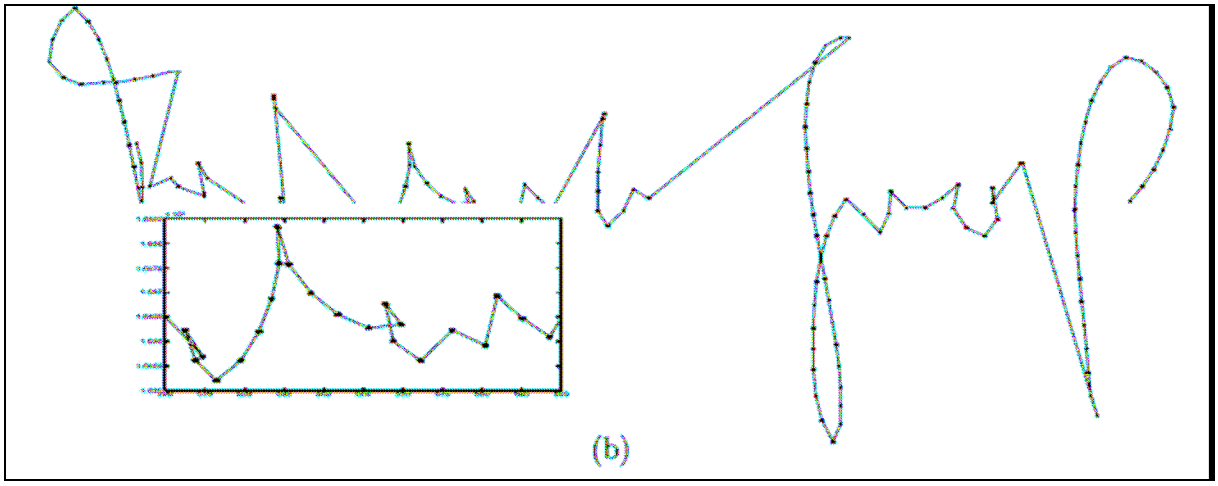

Figura 5 – Assinatura (Pontos Principais)

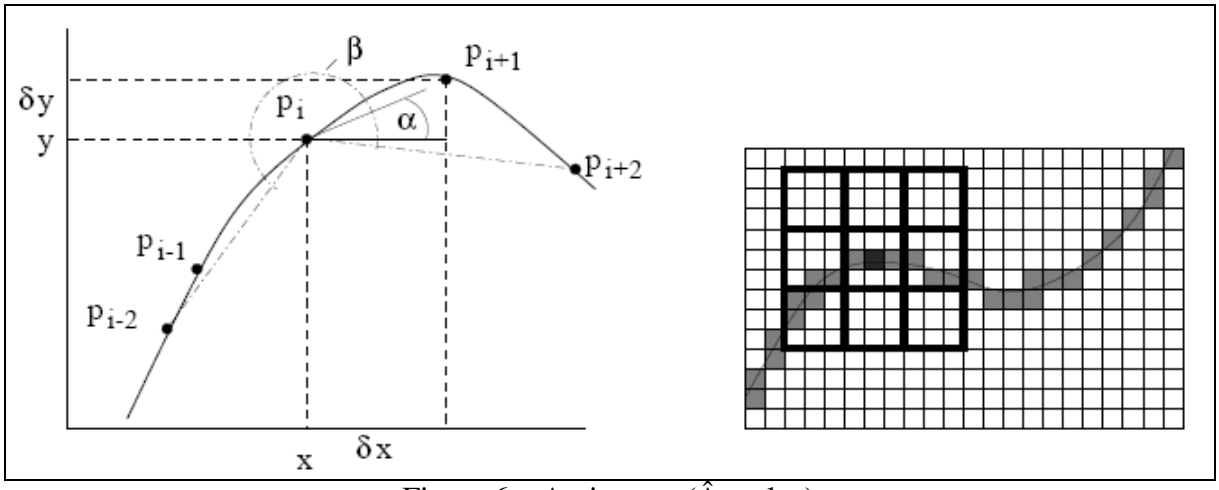

Figura 6 – Assinatura (Ângulos)

Após a extração dos pontos principais, eles podem ser representados por uma seqüência de caracteres.

Os métodos para comparação da assinatura utilizam o sistema de casamento de cadeias (*string matching*). Existem algumas abordagens que são utilizadas para verificação e podem ser divididas em Casamento Exato e Casamento Aproximado.

2.2.1. Casamento Exato

O casamento de cadeia exato consiste em obter todas as ocorrências exatas de uma cadeia inseridas em um texto. Alguns dos principais métodos de *string matching* exato são:

- Algoritmo de Boyer-Moore-Horspool (BMH);
- Algoritmo de Boyer-Moore-Horspool-Sunday (BMHS);
- Algoritmo *Shift-And* Exato (SAE);
- Algoritmo *Shift-Or;*
- Algoritmo Karp-Rabin.

2.2.2. Casamento Aproximado

O casamento aproximado tenta aumentar a área de verificação das cadeias de caracteres, assumindo que algumas variações nessa cadeia devem ser aceitas.

Podemos citar:

- Algoritmo *Shift-And* Aproximado (SAA);
- Algoritmo de Sellers;
- Algoritmo Ukkonen;
- Algoritmo Wu-Manber.

2.3. Considerações Finais

Neste capítulo foi relatado o contexto principal do trabalho, mostrando o ambiente em que é introduzida a análise caligráfica e os métodos de verificação incluindo os principais algoritmos para essa verificação.

Todos os algoritmos citados neste tópico são utilizados em múltiplos ambientes desde a Análise Caligráfica até Biologia Molecular. Uma melhor explicação em cima dos algoritmos pode ser feita através dos estudos de David Menoti Gomes (2004) em seu trabalho "Projeto e Análise de Algoritmos – Casamento Exato e Aproximado de Caracteres" pela UFMG (Universidade Federal de Minas Gerais) e no trabalho de Paulo Gustavo Soares da Fonseca (2003) em "Índices Completos para Casamento de Padrões e Ineferência Motifs" como dissertação de mestrado pela Universidade Federal de Pernambuco.

CAPÍTULO 3 – TECNOLOGIAS UTILIZADAS

Para conseguir aproveitar a análise caligráfica em todos os seus aspectos, os *Servlets* e *Applets* são utilizados para permitir interação com o ambiente *Web* e dispositivos Palm® para efetuar as aquisições.

Por serem tecnologias bem difundidas e de fácil acesso, promovem ao projeto versatilidade suficiente para aplicar o reconhecimento de assinaturas em múltiplos ambientes.

Os dispositivos Palm são computadores móveis que oferecem grande versatilidade às pessoas por serem ágeis devido à fácil utilização que as aplicações oferecem, características essas que os tornam atualmente muito bem aceitos no mercado de computadores pessoais.

Os *Servlets* e *Applets* são tecnologias que oferecem facilidades tanto no momento da implementação (por conterem bibliotecas bem estruturadas e com alto nível de detalhamento) quanto na utilização, devido à grande interação oferecida ao usuário.

3.1. Palm

Com a diminuição do tamanho do computador pessoal, a mobilidade se tornou um fator importante na hora de adquirir uma nova máquina. A primeira miniaturização gerou uma nova classe: os *notebooks*, que foram os propulsores para a tecnologia móvel, dando uma outra visão quanto à utilidade que um computador poderia ter para a sociedade.

Apesar do grande respeito que a sociedade tem pelos *notebooks*, para muitos eles acabam não tendo muito sentido durante o dia-a-dia. Um desses motivos é o seu custo-benefício acima do *Desktop* convencional e seu tamanho que ainda trás certo incômodo ao carregá-lo de um lado para outro.

Aproveitando essa fraqueza, a Palm Inc. lançou em 1996 os produtos Pilot 1000™ e Pilot 5000™ que nasciam para capturar os consumidores que necessitavam de um computador que coubesse na "*palma da mão*" (PALM TIMELINE, 2006).

Inicialmente eram apenas agendas eletrônicas, lentas e com pouca memória, mas hoje em dia podemos considerá-los como grandes organizadores pessoais que disponibilizam aplicações complexas e de grande utilidade.

A Palm Inc. completou 10 anos de existência em 17 de Março 2006 com um total de 34 milhões de computadores móveis vendidos (PALM, 2006).

A partir desse sucesso, é possível analisar uma tendência de que todos estarão carregando um computador pessoal, de fácil utilização e preço acessível. Por essa grande expansão, se torna uma ótima opção a utilização do computador de mão Palm® no projeto, já que ele é composto de uma tela sensível a toque que será essencial para a captura da assinatura.

Com relação à programação, os produtos Palm® utilizam o sistema operacional Palm OS®, desenvolvido pela empresa PalmSource (PALMSOURCE, 2006), que fornece um kit de desenvolvimento contendo todas as estruturas, funções e aplicações (inclusive simuladores e emuladores) que possam ser utilizadas para interagir com o sistema operacional, facilitando a criação de softwares (PALM OS SDK, 2006).

A linguagem nativa do Palm OS® é o C, que permite uma programação bem estruturada tanto em baixo quanto em alto nível.

A PalmSource disponibiliza uma ferramenta chamada PODS (Palm OS Developer Suite) que fornece um ambiente gráfico para facilitar o desenvolvimento. Essa ferramenta deve ser instalada em um computador *Desktop* para que a aplicação possa ser implementada.

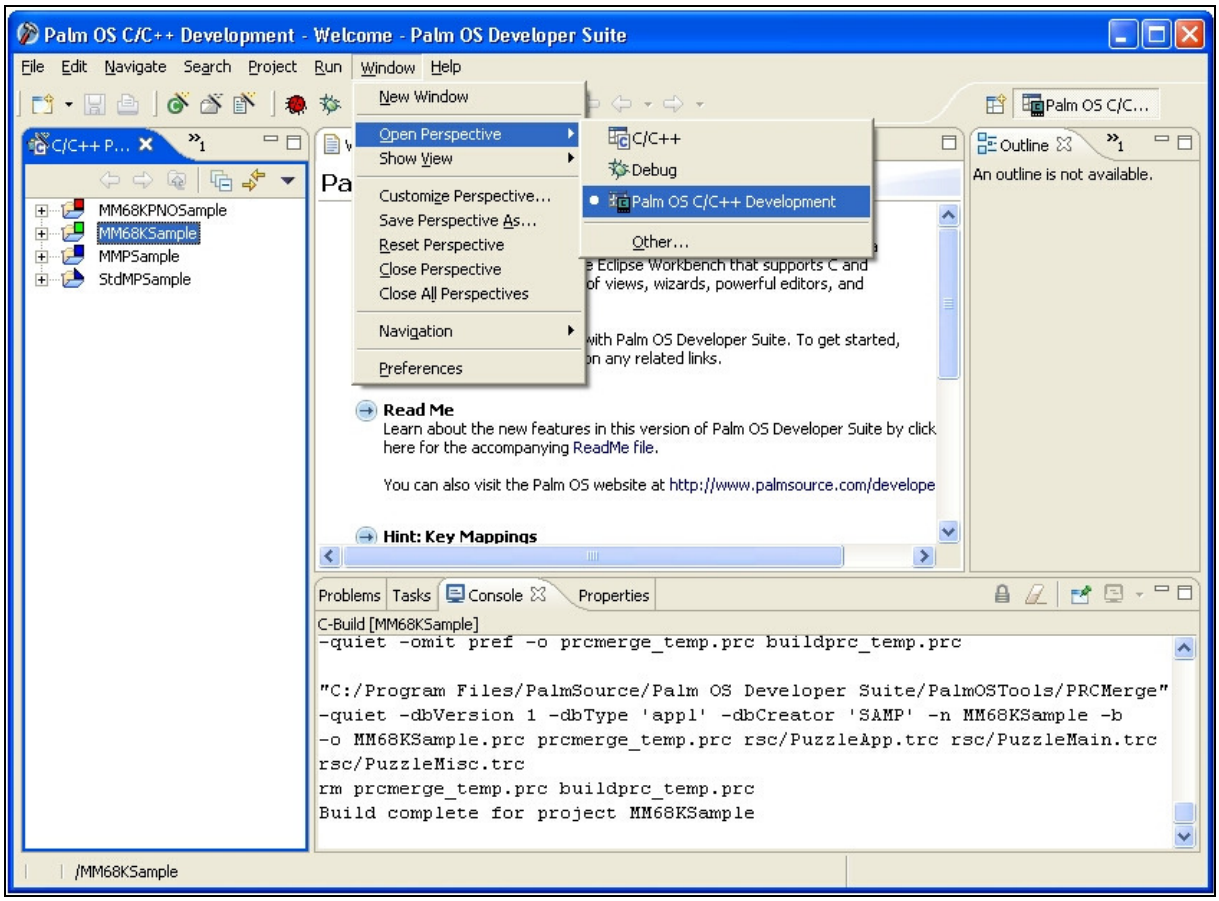

Figura 7 – PODS

Na Figura 7 o ambiente desenvolvido pela Palm é apresentado, indicando a criação de projetos como exemplo, desenvolvidos para facilitar o aprendizado na linguagem e no ambiente gráfico. A ferramenta conta também com utilitários para a depuração e simulação dos projetos nela desenvolvidos, que contêm as mesmas funcionalidades da aplicação real.

É possível depurar o código utilizando o emulador e/ou simulador (Figura 8). Para isso é necessário que o programa seja executado por meio dessas aplicações.

O emulador para Palm®, POSE (*Palm OS Emulator*), foi desenvolvido por programadores independentes, para evitar a necessidade de embarcar o *software* na fase de testes, podendo controlar as reações reais do sistema previamente. Já o simulador foi desenvolvido pela própria Palm®, não contando com todas as funções do POSE, mas havendo a necessidade de utilizá-lo por não ter sido disponibilizado suporte nas versões mais atuais do sistema operacional para o emulador.

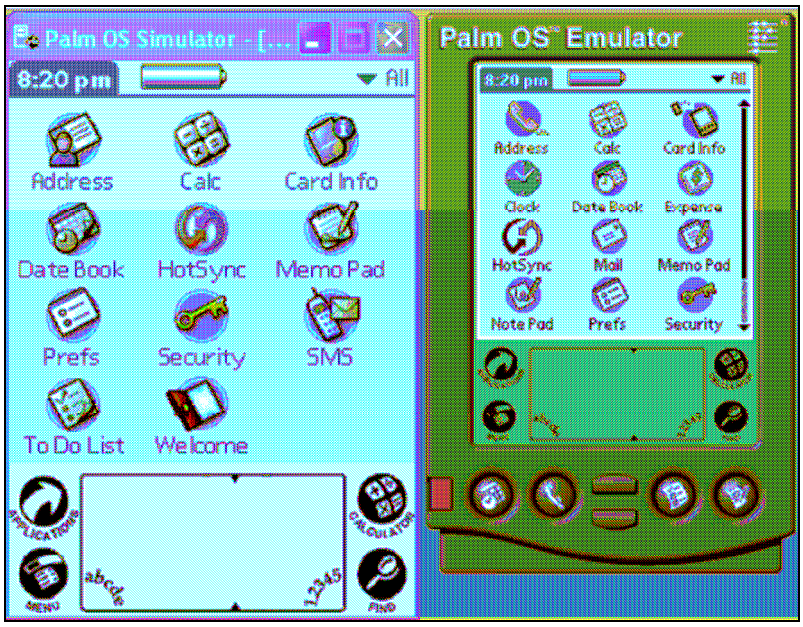

Figura 8 – Simulador e Emulador Palm OS

Após a implementação e os testes, a aplicação deve ser transferida para o dispositivo Palm conectando-o com o computador *Desktop* e utilizando as ferramentas de sincronização oferecidas pelo próprio dispositivo.

3.2. Applets

"Os Applets são programas Java que podem ser incorporados a documentos Hypertext Markup Language (HTML)." (DEITEL, 2005)

No contexto em que o projeto está baseado, *o Applet* se tornou uma grande solução. A sua característica de poder interagir com o ambiente *Web* e ao mesmo tempo dar toda a funcionalidade de um aplicativo comum, possibilitou a adaptação do reconhecimento de assinaturas na internet.

Como todo conteúdo da Internet, para que um *Applet* possa ser executado, ele deve ser baixado pelo navegador (*client-side*). Isso é feito automaticamente e por isso é importante dar atenção ao problema de segurança. O Applet é um programa e, como todo programa, se não for utilizado de maneira responsável, pode acarretar sérios problemas. Devido a esse

motivo, o *Applet* foi envolto em um modelo chamado *SandBox*. Esse modelo garante que o *Applet* só consiga acessar recursos locais caso esteja "assinado", ou seja, caso seja adicionado junto ao *Applet* um certificado comprovando a sua segurança. Ainda é possível detalhar quais recursos serão utilizados pelas classes por meio de um arquivo chamado *Policy*.

O arquivo Policy indica quais as permissões um código terá no momento de sua execução. As ações desse código serão barradas caso ele tente executar algo que não lhe seja permitido (JAVA PERMISSIONS, 2006).

A Figura 9 retrata o processo de validação e execução de um aplicativo Java envolto pelas permissões.

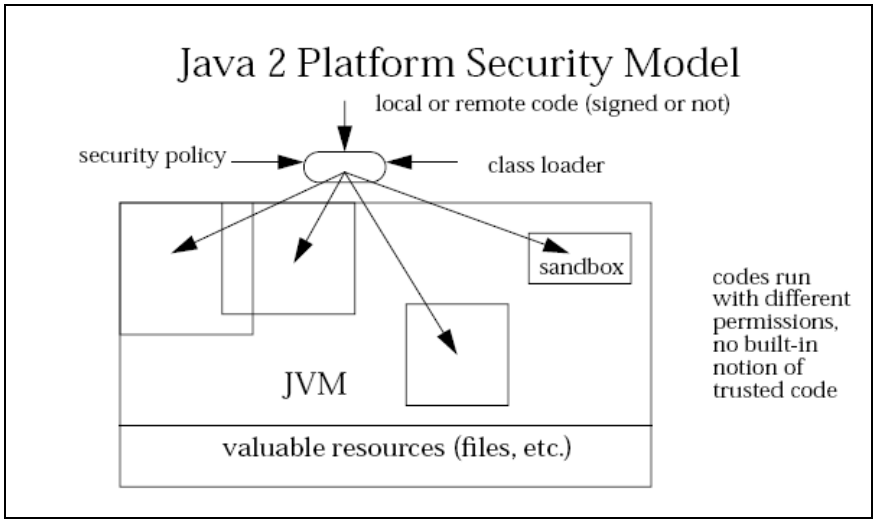

Figura 9 – Modelo *SandBox*

Durante o processo de inicialização de um *Applet*, o *browser* responsável se comunica por meio de avisos, informando ao usuário sobre a situação do programa que está sendo carregado, para evitar códigos iniciados ilegalmente. Essa comunicação é apresentada pela Figura 10, na qual um *Applet* assinado, porém, com certificado não reconhecido, tenta ser executado.

Figura 10 – Certificado de um Applet Assinado

Um *Applet* assinado pode executar ações no computador como manipulação de arquivo, acesso a dispositivos e inclusive gerenciar conexões do tipo *Socket*, que foram utilizadas no projeto para receber as assinaturas do aplicativo Palm.

O *Applet* ficará encarregado também de chamar um outro aplicativo do tipo *Servlet* (*server-side*) que executará o algoritmo de análise, fará o armazenamento e a geração de relatórios com o resultado da análise que será tratado na próxima sessão.

3.3. Servlets

Servlets são programas Java que estão envolvidos no relacionamento cliente-servidor. O papel do *Servlet* é dar assistência às solicitações feitas pelo cliente, retornando informações que tenham sido requisitadas. Essas informações podem ser geradas dinamicamente e serem enviadas através de dados do tipo XML, XHTML, o que garante mais flexibilidade para aplicações *Web.*

Uma grande vantagem dos *Servlets* é o fato de estarem no lado do servidor (*server-side*) podendo se conectar a banco de dados, ler informações locais, entre outros, ampliando, assim, as funcionalidades do servidor (DEITEL, 2005).

A troca de informações é baseada no mecanismo de requisição-resposta que é visualizado na Figura 11.

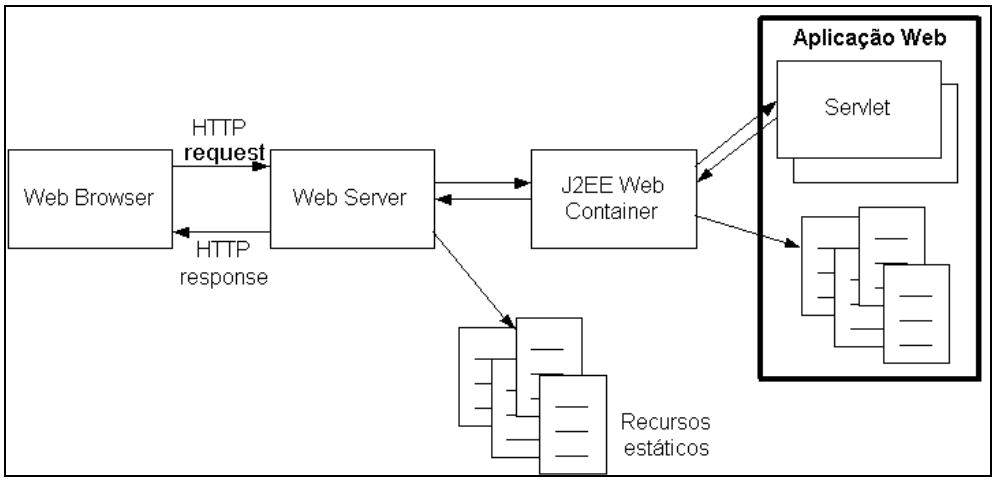

Figura 11 – Requisição e Resposta de um Servlet

Também na Figura 11 é indicada a presença de um elemento chamado de *Web* Container, que é o aplicativo encarregado de hospedar os *Servlet*, controlando seu ciclo de vida e dando suporte a eventos externos por ele invocados, como conexão ao banco de dados.

A Figura 12 mostra o ciclo de vida do *Servlet* sinalizando todos os estados atravessados por ele durante a execução (SERVLET, 2006).

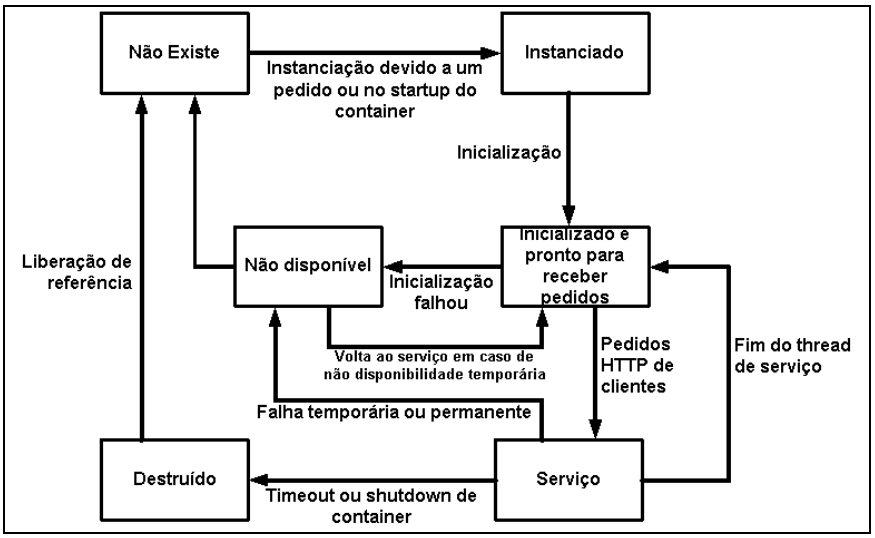

Figura 12 – Ciclo de Vida do Servlet

Os *Servlets* também fornecem suporte ao acesso a banco de dados e devido sua execução ser feita no servidor, esse acesso ganha desempenho e segurança por não precisar de acesso externo.

3.4. Considerações Finais

Todas essas tecnologias estão sempre sofrendo evoluções, trazendo inovações para o setor de desenvolvimento de aplicações *Web* e Móveis, garantindo segurança na escolha de sua utilização.

Será visto a seguir, como essas tecnologias são utilizadas no projeto.

CAPÍTULO 4 – IMPLEMENTAÇÃO

Neste capítulo é apresentado o desenvolvimento do projeto é analisado, mostrando a utilização das tecnologias citadas.

A implementação é baseada no modelo mostrado na Figura 13.

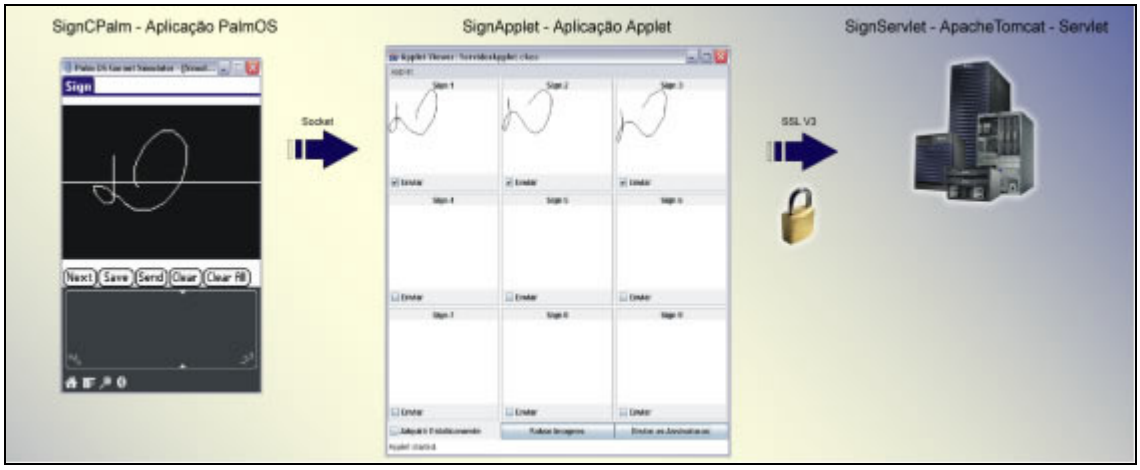

Figura 13 – Lógica de Implementação

O sistema é composto de três aplicações, são elas:

- SignCPalm:
  - Aquisição das assinaturas;
  - Envio para o SignApplet ou armazenamento local.
- SignApplet:
  - Recepção das assinaturas emitidas através da conexão com o SignCPalm ou através de arquivo contendo os dados brutos da assinatura;
  - Transmissão das assinaturas para o SignServlet
- SignServlet:
  - Recepção das assinaturas emitidas pelo SignApplet;
  - Armazenamento;

- Comparação;
- Retorno para o SignApplet sobre a conclusão do processo em formato de XML.

Para iniciar, é necessário acessar um endereço de *Web* (Figura 14), hospedado em um servidor HTTP protegido pelo protocolo SSL, no caso a hospedagem é feita no *Laboratory Of Software Testing* (LOST) da Univem.

O protocolo SSL é mais uma garantia de que todas as informações trocadas durante o processo serão criptografadas, dificultando a interceptação dos dados (JSSE, 2006). O servidor HTTP utilizado é a implementação Apache-Tomcat, que é o *Web Container* responsável por hospedar a aplicação SignServlet.

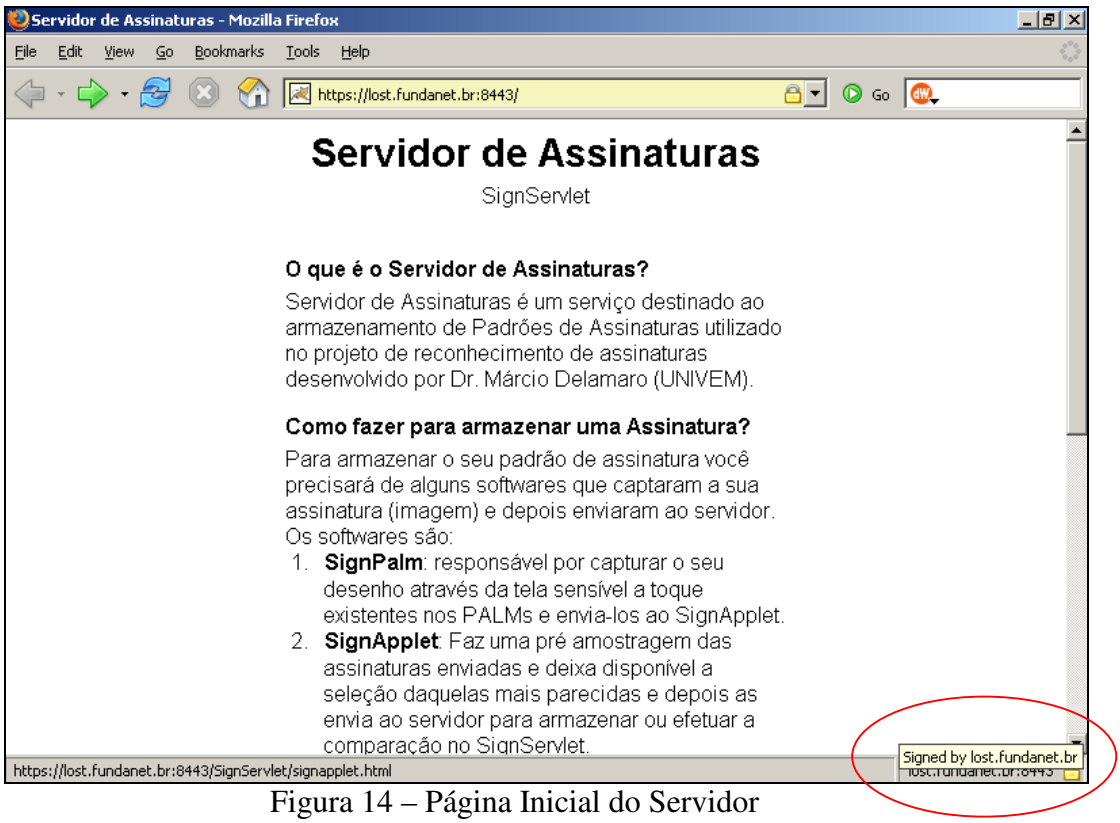

Figura 14 – Página Inicial do Servidor

O Servidor hospeda um web site que levará ao SignApplet.

Esse web site é apenas uma página HTML que fornece um *Applet*, chamado de SignApplet. Esse *Applet* foi "assinado", ou seja, durante sua inicialização, ele emite um certificado que virá no formato de uma tela, mostrada anteriormente (Figura 10), questionando se o certificado é reconhecido pelo usuário.

A assinatura de *Applets* é feita através de um aplicativo chamado *jarsigner* disponibilizado pelo Java SDK.

O método completo de assinatura de *Applets* pode ser verificado em JAVA SECURITY, a partir do capítulo 6.3.1.

4.1. SignPalm

SignPalm é a aplicação responsável por fazer a aquisição das assinaturas no Palm. Ela é baseada praticamente em eventos que captam as ações da caneta utilizada para fazer a escrita. Em cada ação ocorrida, um método é chamado tendo como parâmetro o tipo do evento.

Quando a caneta entra em contato com a tela, o evento se trata de um *penDownEvent* e tem como uma das propriedades a posição em que a tela foi pressionada (eixo x e y). A mesma coisa acontece quando a caneta é movida (*penMoveEvent)* ou quando é retirada (*penUpEvent*). (PALMOS REFERENCE, 2004)

Depois da captura, o ponto é armazenado em um objeto chamado SignTable, cuja estrutura é mostrada na Figura 15.

```
typedef struct {
      int   cont;
      int   max;
      Int16 *size;
      UInt16      **table;
} SignTable;
```

Figura 15 – *Struct* SignTable

Para esse armazenamento é utilizada uma função que adiciona os pontos em uma matriz do tipo SignTable, fazendo os devidos tratamentos.

Em seguida os pontos são desenhados (Figura 16). Os desenhos são baseados em retas e para isso é preciso saber a posição do ponto anterior (lastX e lastY) e do ponto corrente (newX e newY).

```
static void TracaLinha(Int16 lastX, Int16 lastY, Int16 newX,
Int16 newY)
{
      WinPushDrawState();
      WinSetForeColor(white);
      WinDrawLine(lastX, lastY, newX, newY);
      WinPopDrawState();
}
```

Figura 16 – TracaLinha

No Palm é possível saber se o ponto é início ou fim da reta, utilizando os eventos, porém, se for armazenado apenas os eixos x e y, não será possível obter essa informação, para isso é feita uma adição de +10000 para os pontos lastX e lastY que sejam adquiridos nos eventos *penDownEvent* (Figura 17) e +20000 para os pontos newX e newY no *penUpEvent* (Figura 18).

```
case penDownEvent:
      if(RctPtInRectangle(pEvent->screenX, pEvent->screenY, &bounds))
      {
            lastX = pEvent->screenX;
            lastY = pEvent->screenY;
            penDown = true;
            addPoint(MainTable, lastX+10000, lastY+10000);
            Envia(-3,lastX, lastY, 0, 0);
            handled = true;
      }
      break;
```

Figura 17 – Evento *penDown*

```
case penUpEvent:
      if(penDown&&RctPtInRectangle(pEvent->screenX, pEvent->screenY,&bounds))
      {
            Int16 newX = pEvent->screenX;
            Int16 newY = pEvent->screenY;
            TracaLinha(lastX, lastY, newX, newY);
            Envia(-1,lastX, lastY, newX, newY);
            addPoint(MainTable, newX+20000, newY+20000);
            lastX = -1;
            lastY = -1;
            penDown = false;
            handled = true;
      }
      break;
```

Figura 18 – Evento *penUp*

Na Figura 19 a assinatura para teste é demonstrada, onde o ponto azul simboliza o início e o ponto vermelho o fim.

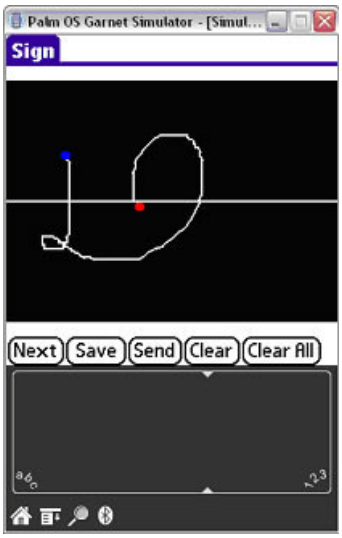

Figura 19 – Pontos no Palm

Os pontos podem ser armazenados no próprio Palm, gerando um arquivo com os dados brutos ou podem ser enviados para a aplicação SignApplet através de uma conexão por *socket*.

Em seguida será apresentada a transmissão dos dados. A abordagem se baseará em dois algoritmos de envio: dinâmico e estático.

4.1.1. Transmissão Dinâmica

A transmissão dinâmica trata do envio simultâneo dos pontos no momento da captura. Esse envio foi desenvolvido para evitar que os pontos sejam armazenados, evitando assim que as informações básicas da assinatura possam ser utilizadas em momento posterior, dando margens para a falsificação.

Nesse tipo de transmissão o processamento dos pontos para seu envio torna-se um fator de atraso que dificulta a captura, diminuindo o número de pontos da assinatura e alterando a qualidade da comparação.

Esses dois fatores, segurança e qualidade, devem ser bem analisados para que possa ser verificado a relação entre custo e benefício de sua utilização.

A conexão é efetuada através do pacote NetLib inserido na API do sistema operacional PalmOS, que fornece suporte para *sockets*.

O *software* envia através da conexão efetuada o ponto capturado informando o evento em que ocorreu, o ponto inicial e o final para que na aplicação SignApplet o tratamento seja correto.

4.1.2. Transmissão Estática

Na transmissão estática, ao invés de enviados, os pontos são armazenados localmente em arquivo do tipo texto. Esses pontos são guardados em ordem seqüencial dos fatos.

Para que possa ser identificado o evento *penUp* e *penDown,* o valor 10000 ou 20000 são adicionados ao ponto, respectivamente.

Ao contrário do método dinâmico, a transmissão estática apenas armazena os valores na memória e só é descarregado em arquivo caso o usuário solicite para posterior sincronização com o computador e carregado para o *Applet* em formato de arquivo.

A vantagem desse método é o fato da aquisição dos pontos ser mais rápida, resultando em maior número de pontos e melhorando a qualidade da assinatura. Devido à qualidade, esse método será utilizado nos testes.

4.2. SignApplet

A aplicação SignApplet é responsável por receber as assinaturas emitidas pelo SignCPalm, tanto através da conexão *socket* quanto pelo armazenamento de arquivo.

SignApplet é uma classe do tipo *JApplet* que fornece ao usuário um ambiente contendo painéis que mostram as assinaturas que foram carregadas por ele. (Figura 20)

Internamente, após os dados serem recebidos (seqüência de pontos) eles são refeitos ponto a ponto em cada painel referente.

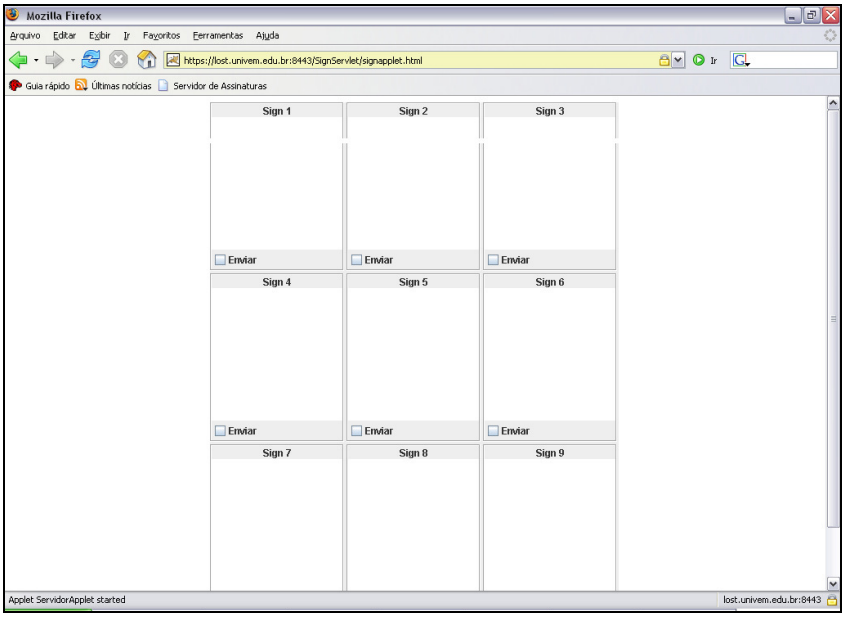

Figura 20 – ServidorApplet(Painéis)

Para que haja a conexão com o Palm, no *layout* é disponibilizado um botão cujo título é “Adquirir Assinaturas” (Figura 21).

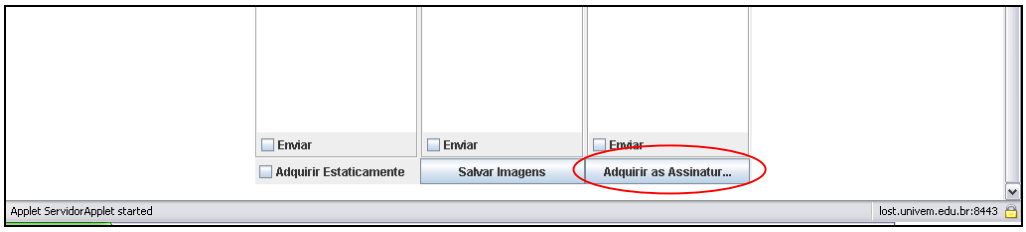

Figura 21 – SignApplet(Adquirir Assinatura)

Neste botão está contida uma rotina que inicia uma *Thread.*

As *Threads* correspondem a processos que terão a execução independente da classe principal, no caso SignApplet.

Há a necessidade de que sejam criados *Threads* para que o recebimento dos dados seja independente das outras operações, pois o desenho da tela feito pelo Applet, não ocorre se estiver sendo executado, por exemplo, uma leitura de dados ou um loop estiver sendo executado. As *Threads* evitam um *freeze* (travamento da tela) durante o processo (APPLET, 2006).

Para que seja possível fazer a leitura dos dados enviados pela aplicação SignCPalm, é necessário criar um servidor socket (Figura 22).

```
public ServerSocket CriaServidorSocket() throws Exception
{
      ServerSocket servidor = new ServerSocket(porta);
      servidor.setSoTimeout(TIMEOUT);

      return (servidor);
}
```

Figura 22 – CriaServidorSocket

Depois de criado o servidor, o Palm deve se conectar ao computador através de um cliente *socket*, na mesma porta onde foi instalado o servidor.

Depois de feito o recebimento, as assinaturas serão mostradas nos painéis da interface do *Applet* (Figura 23).

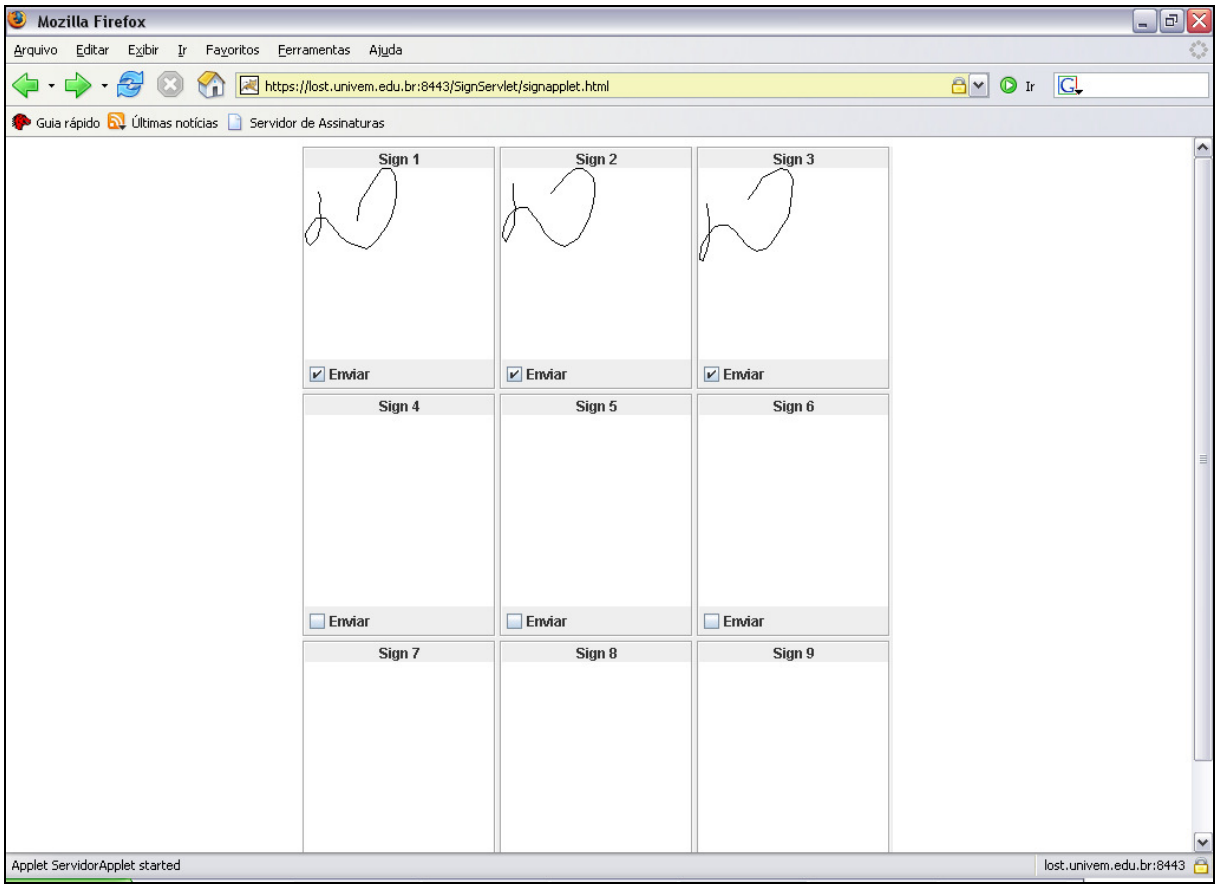

Figura 23 – SignApplet(Assinaturas Recebidas)

O próximo passo é enviar as assinaturas para o SignServlet. Isto é feito através de uma conexão HTTP onde os seguintes dados são passados:

- Número de identificação do usuário para o devido armazenamento;
- Número de identificação caso a assinatura seja uma imitação;
- Porcentagem mínima de acertos que deve existir entre duas assinaturas (*MinCorrectRate*);
- Indicação de utilização de método inserção ou comparação;
- Assinaturas.

Na Figura 24, a implementação é detalhada.

```
public String enviaParaServlet(int id, int id_imitacao, double porcentagem,
int opcao) throws Exception
{
       URL url = new URL(super.getCodeBase()+SERVLET);
       HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
       connection.setRequestMethod("POST");
       connection.setDoOutput(true);
       connection.setDoInput(true);
       DataOutputStream out = new
DataOutputStream(connection.getOutputStream());

       //ENVIA O TIPO DE OPERAÇÃO A SER EFETUADA NO SERVIDOR
       //PODE SER INSERÇÃO DE MODELO, COMPARACAO DE MODELO, INSERCAO DE
      //IMITACAO E COMPARAÇÃOD E IMITAÇÃO
       out.writeInt(opcao);

       //ENVIA A IDENTIFICAÇÃO DE QUEM INSERIU A ASSINATURA
       out.writeInt(id);

       //ENVIA ID DO USUÁRIO DONO DA IMITAÇÃO
       if(opcao==II)
       {
           out.writeInt(id_imitacao);
       }

       //ENVIA PORCENTAGEM DE MINCORRECTRATE QUE FOI ESCOLHIDA
       out.writeDouble(porcentagem);

       //ENVIA AS ASSINATURAS
       if(opcao!=CI)
       {
           enviaParaSocket(out);
       }
       // PEGA RETORNO DA ANÁLISE FEITA NO SERVLET
       DataInputStream in = new DataInputStream(connection.getInputStream());
       byte[] retorno = new byte[in.available()];
       in.readFully(retorno);
       String b = new String(retorno);
       connection.disconnect();
       return b;
}
```

Figura 24 - enviaParaServlet

O retorno dessa função é uma *String* que armazena o endereço de um web site definido pelo *Servlet*. O *Applet* pega o endereço, verifica sua validade e faz com que o *browser*, onde está o *Applet*, carregue o endereço retornado (Figura 25).

```
String desvio = enviaParaServlet(new Integer(a).intValue(),func);
if(desvio.length() > 0)
{
      super.getAppletContext().showDocument(new URL(desvio));
}
else
{
      JOptionPane.showMessageDialog(null,"Não Foi Possível Interpretar
Retorno.");
}
```

Figura 25 – Desvia Web Site

Na próxima sessão será mostrado o funcionamento interno do *Servlet*.

4.3. SignServlet

Quando o *Servlet* identifica uma comunicação, tanto pelo método *Post* quanto pelo método *Get,* o evento

protected void processRequest(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException

é ativado e com ele são carregadas as informações necessárias para a comunicação.

Os dados são recebidos na forma de uma matriz de inteiros, que se referem à assinatura, portanto, para armazená-las é utilizada a classe SignSet, contida no pacote que faz a comparação entre as assinaturas, guardando todas as propriedades necessárias na análise.

Caso a opção escolhida seja a de inserção de um modelo, é preciso acessar o banco de dados (MySql) para fazer o armazenamento dos dados.

Se a opção for comparação, é preciso extrair o modelo do banco de dados e fazer a comparação com assinatura recebida (Figura 26).

```
public boolean[] comparaAssinaturas(SignSet v1, SignSet modelss, double
porcentagem)
{
      v1.setMinCorrectRate(porcentagem);
      v1.computeParameters();
      modelss.setMinCorrectRate(porcentagem);
      modelss.computeParameters();
      int[][] vs = v1.getSignatures();
      int[][] model = modelss.getSignatures();
      boolean[] assinaturasOk = new boolean[vs.length];
      for (int i = 0; i < vs.length ; i++)
      {
            boolean bate = true;
            for (int j = 0; j < model.length; j++)
            {
                  SignSet newss = new SignSet("");
                  newss.setMinCorrectRate(porcentagem);
                  newss.addSignature(vs[i]);
                  newss.addSignature(model[j]);
                  newss.setMinCorrectRate(porcentagem);
                  newss.computeParameters();
                  if ( newss.getMaxMax()> modelss.getMaxMax() ||
                       newss.getMaxMin()> modelss.getMaxMin() ||
                       newss.getMaxPU()> modelss.getMaxPU()  ||
                       newss.getMinMin()< modelss.getMinMin() ||
                       newss.getMinMax()< modelss.getMinMax() ||
                       newss.getMinPU()< modelss.getMinPU()  ||
                       newss.getMaxThreshold()>modelss.getMaxThreshold())
                  {
                        bate = false;
                  }
                  else
                  {
                        bate = true;
                  }
            }
            assinaturasOk[i]=bate;
      }
      return assinaturasOk;
}
```

Figura 26 – comparaAssinaturas

O método comparaAssinaturas retornar um valor, que corresponde ao resultado, informando quais assinaturas foram aceitas (*true*) e quais recusadas *(false*). Em seguida é enviado um resultado sobre a comparação, mostrado a partir de um arquivo XML gerado pelo *Servlet* que contém todos os dados da aquisição, recepção e comparação, incluindo as imagens das assinaturas (Figura 27).

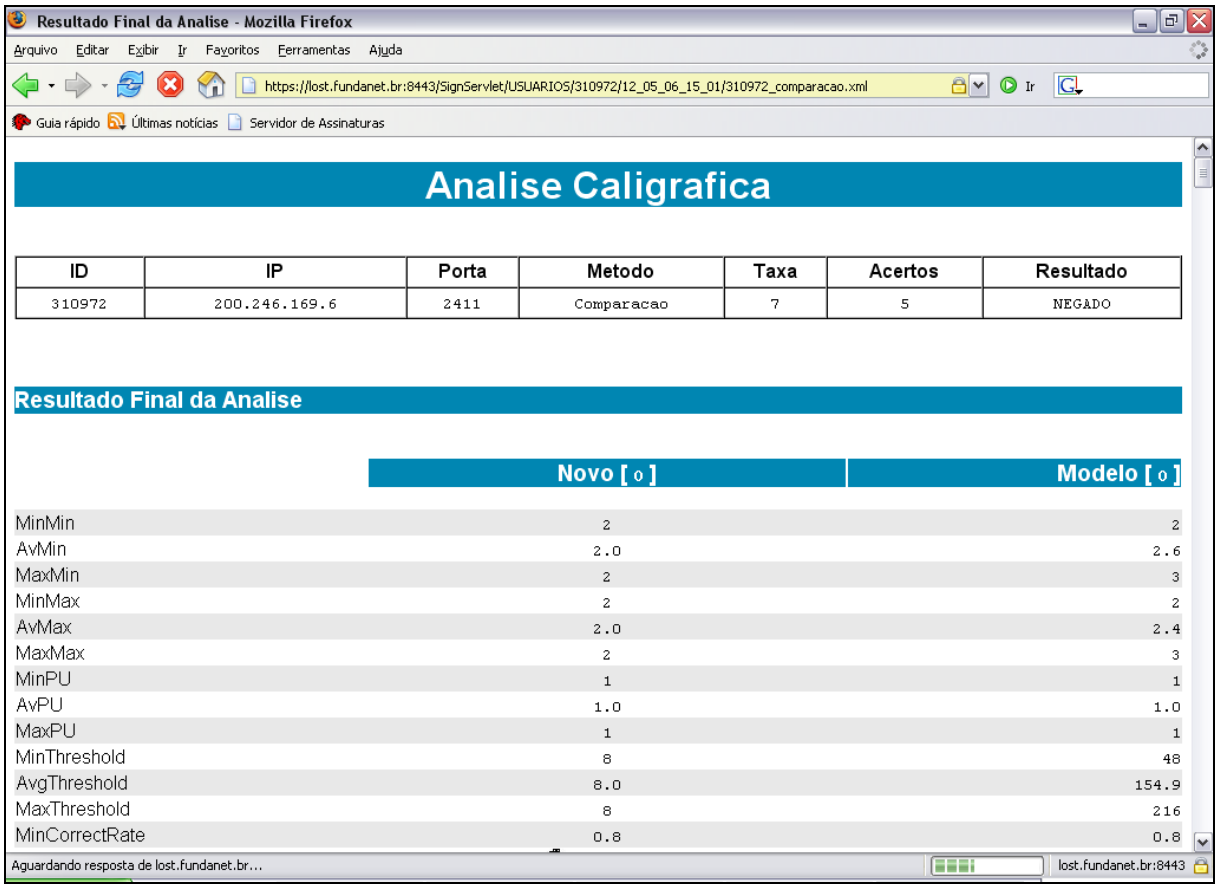

Figura 27 – Resultado da Análise

No relatório final da análise é listada toda a comparação feita pelo SignServlet. Se forem armazenadas, como modelo, 9 assinaturas e transmitidas para comparação 3 imitações então tem-se um total de 27 comparações efetuadas, podendo saber quais os valores resultantes nessas comparações e com isso verificar o porque de terem sido aceitas ou negadas.

4.4. Considerações Finais

O relacionamento dos 3 aplicativos resultou em uma interação fácil e clara permitindo que as fases de aquisição, transferência e comparação se tornassem eficientes para completar a análise caligráfica com eficiência. No próximo capítulo, novos testes mostraram conclusões mais eficientes sobre o projeto.

CAPÍTULO 5 – EXPERIMENTOS

A elaboração de experimentos se torna crucial para comprovar o funcionamento do projeto. A execução de todas as fases, desde a aquisição até a comparação, é necessária para verificar possíveis melhorias ao sistema. Um grupo de pessoas foi selecionado aleatoriamente para efetuar os testes e assim adquirir diferentes padrões de assinatura e diferentes críticas sobre o sistema.

5.1. Objetivo

O propósito do teste é a verificação do sistema, analisando se o processo de captura, transmissão e armazenamento das assinaturas são feitos corretamente e aproveitar os dados para analisar o índice de aceitação falsa, ou seja, a quantidade de assinaturas imitadas que foram identificadas como verdadeiras e assim poder selecionar a melhor taxa mínima de acerto.

É importante frisar que esse teste não garantirá uma perfeita análise do algoritmo, trata-se apenas de uma prévia para os testes futuros, que devem ser feitos de maneira mais completa.

5.2. Método

Para os testes foram selecionadas trinta pessoas dispostas a efetuar a digitalização de nove assinaturas próprias e três imitações. Após o recolhimento das nove assinaturas de cada pessoa, as imagens foram impressas e distribuídas entre elas para realizar um treinamento baseado na assinatura que deveram imitar. Essa distribuição foi feita de forma aleatória.

Após o treinamento, as três imitações, referentes às falsificações, foram recolhidas e submetidas a seis testes de comparação.

Para fazer essa comparação, foram utilizados 6 valores para a porcentagem mínima de acerto (*MinCorrectRate*). Esse parâmetro é a porcentagem mínima de acerto que deve existir entre duas assinaturas. Por exemplo, se ambas as assinaturas contiverem 100 pontos, e usar uma taxa de 0.80, então pelo menos 80 pontos devem coincidir. Os valores foram 0.70, 0.75, 0.80, 0.85, 0.90 e 0.95.

O resultado do teste mostra qual o melhor valor para o *MinCorrectRate*.

5.3. Resultados

Após a finalização das aquisições, foi elaborada uma tabela com o resultados. A Tabela 2 indica o número de assinaturas que foram aceitas de forma incorreta em cada porcentagem e para seu respectivo usuário, ou seja, são imitações que passaram pelo sistema como verdadeiras. Por exemplo, para o usuário de código 3, na porcentagem 0.85 foram aceitas erroneamente 25 comparações de um total de 27, já na porcentagem 0.90, esse número cai para 16, com isso é possível verificar que para essa comparação a melhor porcentagem de aceitação é de 0.90.

Tabela 2 – Aceitação Falsa

Usuario \ MinCorrectRate	**0.70**	**0.75**	**0.80**	**0.85**	**0.90**	**0.95**
2	18	16	6	10	8	13
3	19	23	19	25	16	26
7	16	13	8	17	15	18
285943	12	6	18	19	19	18
297208	6	5	5	5	5	7
302708	0	0	0	0	0	0
303461	1	4	8	6	8	9
312754	0	0	0	0	0	0
312789	0	0	0	0	0	0
314196	7	6	8	7	8	8
314323	0	0	0	0	0	0
314633	8	14	7	15	14	18
315591	0	0	1	0	18	18

316059	9	9	9	6	9	9
343900	9	9	9	9	9	9
5	0	0	0	0	0	0
4	9	7	9	9	9	9
8	0	0	0	0	0	0

A partir dos dados referentes à Tabela 2, foi construído um gráfico (Figura 28) que mostra a porcentagem de aceitação falsa em cada taxa de *MinCorrectRate*.

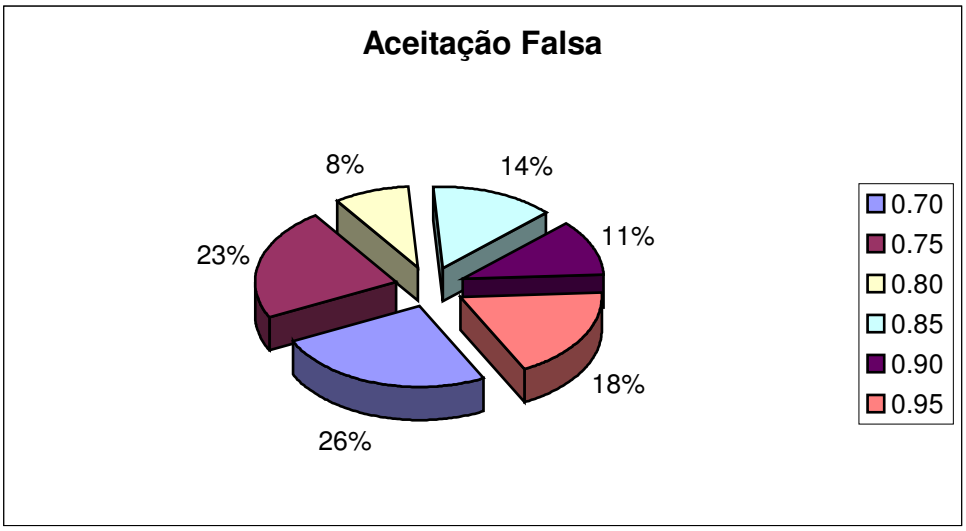

Figura 28 – Gráfico Aceitação Falsa

Na Figura 28 pode-se verificar que na taxa 0.8 foi onde ocorreu o menor índice de aceitação falsa, portanto aquela que mais se apresentou confiável.

5.4. Considerações Finais

Os testes devem ser melhorados, mas o motivo principal, garantir a confiabilidade dos dados, foi confirmado, onde o processo de aquisição, transferência e armazenamento portaram-se positivamente, garantido a informação por completa.

CAPÍTULO 6 – CONCLUSÕES

O processo de aquisição e transferência foi implementado. O sistema se comportou bem durante os testes iniciais, podendo garantir que as informações adquiridas são armazenadas de forma segura e disponibilizadas para comparação.

O dispositivo Palm® utilizado para a aquisição da assinatura conseguiu suprir as necessidades do projeto, porém, deixou a desejar na qualidade da assinatura, pois a tela de captação não é tão sensível a ponto de imitar uma escrita utilizando caneta esferográfica e papel comum. Uma solução para esse problema é a adaptação para a captura em mesas digitalizadoras.

O aplicativo SignPalm efetuou conexões facilmente com o Applet e tornou o processo de aquisição simples, tendo como único inconveniente o fato de o gerenciador de conexões do sistema operacional Palm OS® ter um *TimeOut* padrão relativamente baixo, fazendo com que a conexão seja cortada. Para evitar esse problema seria necessário configurar o *TimeOut* para manter um prazo mais longo de espera.

A quantidade de pontos de uma assinatura acabou sendo influenciada pelo processamento caso seja usado o método de transferência ponto a ponto (muito processamento, poucos pontos). Para evitar esse problema nos testes, foi utilizada a versão de transmissão estática, que não ocasiona atraso, porém, existem algumas modificações que podem ser feitas na transmissão dinâmica que diminuirá o atraso, pois essa versão também está apta a desviar os dados para um arquivo, sendo necessário gravá-los em memória. Removendo essas funções o envio por *socket* fica mais limpo, e gerando ganho no processamento.

O ServidorApplet conseguiu estabelecer o servidor *socket* e efetuar a aquisição das assinaturas corretamente, tanto na transmissão dinâmica quanto na estática.

O SignServlet foi implementado e sua utilização trouxe ao projeto uma flexibilidade muito grande, tornando o sistema facilmente acessível ao ambiente Web. Seu desempenho foi excelente considerando o grande processamento necessário, desde a comunicação com banco de dados até a análise das assinaturas. Baseados nisso podemos concluir que a tecnologia dos Servlets vai muito além de um simples gerador de conteúdo dinâmico para a internet.

De uma forma geral, o sistema conseguiu cumprir as necessidades do projeto trazendo boas expectativas para que este possa ser utilizado em futuros projetos para o aprimoramento do algoritmo de análise caligráfica desenvolvido pelo Prof. Dr. Márcio Eduardo Delamaro.

O projeto trouxe ao aluno grande aprendizado, pois necessitou de um grande nível de pesquisa na área de programação com linguagem Java e para dispositivos Palm®, passando por conexões *sockets*, conexões a banco de dados e interação no ambiente *Web* com *Servlets* e *Applets*, mais especificamente na criação de *Applet* seguros.

Espera-se que o sistema desenvolvido possa permitir que a análise caligráfica seja implantada em sistemas WEB, trazendo mais segurança para o ambiente e tornando a vida das pessoas mais simples, sem a necessidade de memorização de grande número de códigos, senhas para que possa efetuar operações importantes.

REFERÊNCIAS BIBLIOGRÁFICAS

ANIL, Jain K.; GRIESS, Friederike D.; CONNEL, Scott D. On-Line Signature Verification, Department of Computer Science and Engineering. Michigan State University, 1999.

GOMES, David Menoti. Projeto e Análise de Algoritmos, Casamento Exato e Aproximado de Cadeias de Caracteres, 2004.

FIGUEIREDO, A. Revista Unicamp. Administração de Sistemas e Segurança, nº. 6, Setembro de 1999. Disponível em http://www.revista.unicamp.br/infotec/admsis/admsis6-1.html. Acessado em 06/06/2006.

FERRO, Wanderson Roberto. Comércio Eletrônico e a Segurança da Rede: Uma visão Tecnológica, 10/05/2006.

FONSECA, Gustavo Soares da. Índices Completos para Casamento de Padrões e Ineferência Motifs, Dissertação de Mestrado, Universidade Federal de Recife, 2003

PALM TIMELINE. Historical Timeline. Disponível em http://www.palm.com/us/company/corporate/timeline.html. Acessado em 11/06/2006.

PALM. Palm Celebrates 10. Year Anniversary of the Pilot, the Mobile Computer That Changed the Way People Work and Live. Disponível em http://www.palm.com/us/company/pr/news_feed_story.epl?reqid=835698. Acessado em 11/06/2006.

PALMSOURCE. Disponível em http://www.palmsource.com. Acessado em 11/06/2006.

PALM OS SDK. Palm OS Software Development Kit. Disponível em http://www.palmos.com/dev/tools/core.html. Acessado 01/05/2006.

JAVA SECURITY. Java™2 Platform Security Architecture. Disponível em http://java.sun.com/j2se/1.3/pdf/security-spec.pdf. Acessado em 07/06/2006.

DEITEL. Java: Como Programar, 6ª Edição, Deitel, 2004.

SERVLET. Introdução a Containers Web: A Primeira Aplicação Web com Servlets. Disponível em http://jacques.dsc.ufcg.edu.br/cursos/j2ee/html/servlets/techsupport.htm. Acessado em 14/06/2006.

PALMOS REFERENCE. Palm OS® Programmer's API Reference, Palm OS® 68k SDK, 2004

J2SE API. Java™ 2 Platform Standard Edition 5.0 API Specification. Disponível em http://java.sun.com/j2se/1.5.0/docs/api/.

J2EE API. Java™ 2 Platform Enterprise Edition, v 1.4 API Specification. Disponível em http://java.sun.com/j2ee/1.4/docs/api/.

JSSE. Java ™ Secure Socket Extension (JSSE). Reference Guide, for the Java™ 2 SDK, Standard Edition, v 1.4.2. Disponível em http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html.

APPLETS. Lesson Applets Tutorial. Disponível em http://java.sun.com/docs/books/tutorial/deployment/applet/index.html. Acessado em 04/05/2006.

JAVA PERMISSIONS. Permissions in the Java™ 2 SDK http://java.sun.com/j2se/1.4.2/docs/guide/security/permissions.html. Acessado em 22/10/2006.

SILVEIRA, Danilo Quiquinato Pinheiro da

Sistema para Aquisição e Transferência Segura de Dados Destinados à Análise Caligráfica / Danilo Quiquinato Pinheiro da Silveira; orientador: Prof. Dr. Márcio Eduardo Delamaro. Marília, SP: [s.n], 2006

48 f.

Monografia (Graduação em Ciência da Computação) - Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha – UNIVEM

1. Engenharia de Software 2. Biometria 3. Java 4. Palm

CDD: 005.1