Edré Quintão Moreira

Um Modelo Cooperativo para Aplicações Distribuídas Baseado na Web: Aplicação à Análise e Armazenamento de Registros Eletroforéticos

Dissertação apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte

04 de abril de 2002

Agradecimentos

A influência do ambiente de convívio em nosso comportamento e desempenho é inquestionável. São muitos aqueles que me ajudaram para que este trabalho pudesse ser desenvolvido, onde destacarei alguns deles.

Primeiramente, agradeço a Deus por minha existência.

Devo um obrigado muito especial ao Professor Osvaldo, que além de sua dedicação e paciência enquanto orientador, contribuiu enormemente para minha formação profissional.

Agradeço à Denise pelas revisões, à mamãe Neuza e ao *big brother* André pelo incentivo constante, ao *big brother* Sudré, que deixou muitos exemplos a serem seguidos durante sua estadia neste mundo e ao meu pai Édson que procurou me ajudar enquanto presente entre nós.

Os amigos do LCC/CENAPAD tiveram seu lugar de destaque tanto nos dias em que procurávamos nos divertir para aliviar o *stress* semanal, quanto na conclusão deste trabalho, quando assumiram também meu papel profissional.

Agradeço ao Chico pelas inúmeras vezes que interrompi suas tarefas para pedir explicações sobre termos biológicos e também pela revisão feita em meu trabalho.

Ao grande amigo Lenin, pelo companheirismo em todos estes anos de convivência, e ao amigo Lúcio França, que apesar de não estar hoje presente, continua a nos inspirar com sua fantástica inteligência e determinação para alcançar seus objetivos.

Esta pesquisa foi parcialmente financiada com recursos dos projetos TEC 995/96, da Fapemig, e 521742/94 do CNPq.

Resumo

Cientistas geram dados experimentais, e o interesse no compartilhamento destes dados é enorme, em especial na área biológica. É o compartilhamento que permite o trabalho cooperativo entre laboratórios espalhados pelo mundo. Este interesse gerou a formação de diversas bases de dados, como o *GenBank* e o *Protein Data Bank*. Cientistas de todo o mundo ali depositam seus dados, formando grandes repositórios globais que são usados para análise e recuperação de registros. Tais tarefas podem ser feitas de duas formas: utilizando serviços implantados nas bases centrais, como o *BLAST* para a recuperação de sequências homólogas, ou fazendo cópias locais das bases e implantando localmente novos algoritmos. Neste trabalho nós propomos a formação de um repositório global não real, mas sim virtual, através do uso de agentes móveis, e da fácil implantação por um usuário de funcionalidades adicionais para análise e recuperação de dados, utilizando carga dinâmica de classes Java.

Nós aplicamos este modelo na construção de uma nova versão do AnaGel, um sistema para armazenamento e análise de registros eletroforéticos. A eletroforese é um processo de separação de macromoléculas biológicas de extrema utilidade para bioquímicos e geneticistas. Várias amostras contendo diferentes misturas de macromoléculas - geralmente fragmentos de DNA, RNA ou de proteínas, com a mesma carga elétrica - são colocadas em uma placa coberta por um gel, e submetidas a um campo elétrico, que força a migração das macromoléculas através do gel. Moléculas menores encontram menor dificuldade para atravessar o gel, e seu deslocamento é consequentemente maior. O inverso acontece com as moléculas maiores. Um experimento de eletroforese gera então uma imagem, com várias *canaletas*, cada uma correspondendo a uma amostra, onde em cada canaleta há uma formação de *bandas* nos pontos de acúmulo de moléculas com o mesmo peso molecular.

Em sua primeira versão o AnaGel já era uma ferramenta bastante completa para o tratamento de registros eletroforéticos, oferecendo a deteção automática de canaletas e bandas, a correção de distorções, a determinação de pesos moleculares, além da recuperação de registros por diversos métodos. Era entretanto um programa que deveria ser instalado em um único computador, e este mesmo computador servia para a entrada e o armazenamento dos dados. A versão atual, toda construída em Java, apresenta como

vantagens a instalação em um servidor, com o uso por navegadores web, a possibilidade de análises e recuperação de registros através de agentes que se movimentam por todos os repositórios de uma rede de laboratórios que desejam compartilhar dados, e a facilidade de adição de novas funcionalidades pelos usuários, que necessitam apenas de construir classes Java obedecendo a uma classe abstrata exportada pelo AnaGel.

Nesta dissertação nós apresentamos uma discussão sobre o uso da Web para o trabalho cooperativo entre cientistas experimentais. A solução técnica adotada - agentes móveis, com o sistema Grasshopper - é comparada com outras possibilidades para estruturação de sistemas distribuídos. Nós procuramos mostrar como o sistema AnaGel se compara favoravelmente com seus concorrentes comerciais. Diversos detalhes da implementação são explicados, e apontadas direções para melhorias futuras.

Abstract

Scientists produce experimental data, and there is a big interest in sharing it, specially in the biological area. Is the sharing of data that allows the collaborative work among laboratories all over the world. From this interest, several databases were built, like the *GenBank* and the *Protein Data Bank*. Scientists use these databases to deposit their data, creating big global repositories that are used for analysis and recovering of records. Such tasks can be made in two ways: by using services provided in the central database, like *BLAST* for recovering homologous sequences, or by doing a local copy of the database and implementing new algorithms to work on it. In this work we propose the construction of a virtual global repository by using mobile agents, and the easy implementation of additional functionalities for analysis and recovering of data using dynamic loading of Java classes.

We applied this model in the development of a new version of AnaGel, a system for storage and analysis of electrophoretical records. Gel electrophoresis is a method that separates macromolecules on the basis of size, electric charge, and other physical properties. Many important biological molecules such as amino acids, peptides, proteins, nucleotides, and nucleic acids, exist in solution as electrically charged species. A gel is a colloid in a solid form. Gel electrophoresis refers to the technique in which molecules are forced by an electrical field to move across a span of gel. A molecule's properties determine how rapidly the molecule moves through a gelatinous medium. Multiple samples of substances containing macromolecules are separated by a single electrophoresis experiment; each sample gives rise to a pattern called a *lane*, which is composed by a lot of *bands* - places in which molecules with similar physical properties are accumulated.

In the first version of AnaGel several functionalities for analysis of electrophoretical records were included, such as automatic detection of lanes and bands, correction of distortions in the image, molecular weight (or size) determination, storage of records in a database, and recovering os records by similarity with many different comparison methods. It is a stand alone software, which must be installed in each client machine with its own database. The new version has been built in Java, being installed in one or more servers and accessed via web browsers. It also adds new functionalities like the recovering of records by similarity in different servers through the use of mobile agents, and the

possibility of inclusion of new comparison methods by the user.

In this dissertation we present a discussion about the use of the Web for cooperative work between scientists. The technical solution adopted - mobile agents, with the Grasshopper system - is compared against other possibilities of development of distributed systems. We also compare the AnaGel system with some commercial softwares. Several implementation details are explained and possible future work is presented.

Sumário

Capítulo 1

Introdução

1.1 Motivação

Trabalhos científicos demandam diferentes tipos de instrumentos e dados, que são dependentes de suas finalidades. O acesso a estes recursos pode demandar um grande esforço por parte do pesquisador, uma vez que podem estar situados em locais distantes da organização onde ele trabalha. Junte-se a isto o fato de, muitas vezes, serem desenvolvidos trabalhos conjuntos entre pesquisadores de várias organizações dispersas pelo mundo, dado que a colaboração é imprescindível para a ciência moderna.

O advento da Internet criou novas possibilidades para as organizações desenvolverem trabalhos científicos de maneira conjunta, especificamente entre colaboradores separados geograficamente. *GenBank*[1], *PDB*[2], *CERN*[3], são exemplos de uso da *Web* para o compartilhamento de dados experimentais obtidos por uma miríade de laboratórios espalhados por todo o mundo.

Um uso típico de ambientes cooperativos é para consulta sobre quais os registros presentes nos bancos de dados que mais se assemelham a um registro chave submetido para comparação. A determinação do grau de semelhança é feita por alguma métrica implícita em um algoritmo, tal qual o *BLAST* [7], que implementa uma métrica para comparação de sequências de bases ou aminoácidos para consultas no *GenBank*.

Estes bancos de dados experimentais têm como características principais a centralização e a disponibilização de métodos de pesquisa restrita a um conjunto de operações. A utilização de bancos de dados centrais faz necessário o uso de equipamentos com grande capacidade de armazenamento e potência computacional para atender a um volume de consultas e transações crescentes. A limitação quanto aos métodos de comparação utilizados

[1] http://www.ncbi.nlm.nih.gov/Genbank/index.html
[2] http://www.rcsb.org/pdb
[3] http://www.cern.ch

impede que os pesquisadores testem suas conjecturas, a não ser que façam uma cópia local dos dados e que implementem suas próprias ferramentas de análise. O valor do *BLAST* como métrica de distância vem do seu sucesso na descoberta de homologias, e não de algum resultado teórico que garanta a sua superioridade. Um cientista pode querer testar outras métricas, baseadas em algoritmos radicalmente diferentes.

Os dados experimentais que tratamos neste trabalho são *registros eletroforéticos*. O processo denominado *eletroforese* (vide Apêndice A) envolve o posicionamento de uma pequena quantidade de mistura a ser analisada no topo de um gel, normalmente de poliacrilamida ou agarose, e a aplicação de um campo elétrico por um período de tempo. O campo elétrico aplicado às moléculas carregadas provoca o deslocamento destas, fazendo com que algumas migrem mais que outras, dependendo do peso molecular e de suas dimensões. A aplicação de corantes nesse gel possibilita a visualização de padrões que identificam as concentrações de moléculas com características particulares (Figura 1.1). Esta técnica é aplicada, principalmente, à separação de moléculas de proteínas, fragmentos de DNA e fragmentos de RNA.

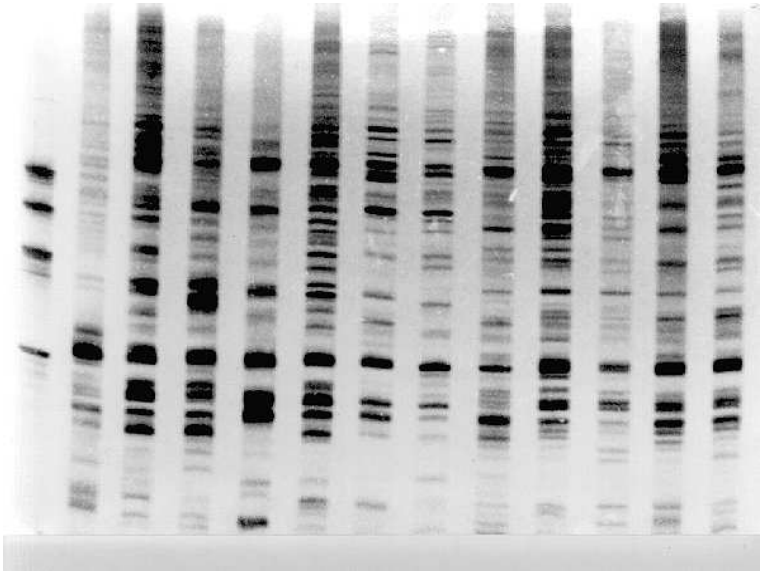

Figura 1.1: Padrão observado de um gel de eletroforese.

As faixas verticais que podem ser observadas na Figura 1.1 são denominadas *canaletas*.

O processamento de géis desta natureza pode ser automatizado, facilitando ao cientista o estudo de vários dados que dizem respeito às substâncias analisadas.

Em primeiro lugar, o *perfil* de cada canaleta deve ser extraído e normalizado de forma a possibilitar sua análise pelo *software* em questão.

Podem haver distorções mais acentuadas em determinadas regiões do gel, como é o caso do *efeito sorriso*. Com o intuito de tratar todas as canaletas de maneira uniforme, a normalização dos perfis tenta reparar as diferenças provocadas por tal deformação. Desta

forma os perfis podem ser alongados ou reduzidos, o que pode ser feito de maneira uniforme ou levando-se em consideração outras variáveis, como por exemplo o ponto onde a distorção é mais acentuada, etc. Isto é dependente do método de normalização utilizado.

A normalização pode também ser utilizada em outras situações onde deseja-se comparar canaletas de géis diferentes, os quais possuem características distintas, como a luminosidade por exemplo. Através do uso de algum algoritmo, os perfis podem ser compatibilizados para uma melhor análise.

Uma segunda fase importante diz respeito ao cálculo dos pesos moleculares de cada banda encontrada na imagem, uma vez definidas as curvas de calibração para o gel. Um método largamente utilizado, muito embora não seja o único, leva em consideração que o deslocamento é inversamente proporcional ao logaritmo do peso das moléculas. A Figura 1.2 mostra um exemplo de canaleta, destacando sua imagem, seu perfil extraído, os pesos moleculares estimados para cada banda e o perfil normalizado.

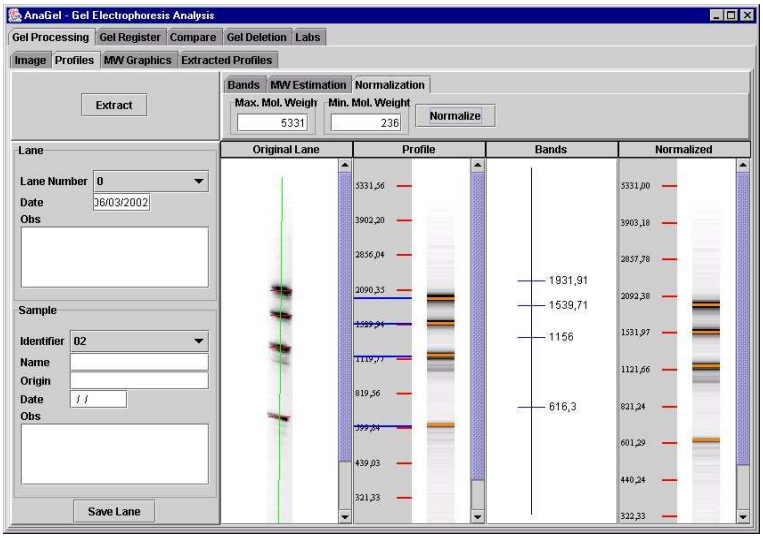

Figura 1.2: Processamento de uma canaleta.

Para se estimar o quanto uma amostra analisada se diferencia de outra, diversas métricas podem ser utilizadas. Normalmente essas métricas levam em consideração o número de características presente em ambas as amostras (a), o número de características presentes na amostra 1 e ausentes na amostra 2 (b), o número de características presentes na amostra 2 e ausentes na amostra 1 (c) e número de características ausentes em ambas as amostras (d). Alguns exemplos são o coeficiente de *Kulczynski*[22, 17], dado pela equação $s = \frac{a}{b+c}$, o de *Jaccard*[36], onde $s = \frac{a}{a+b+c}$, etc. sendo s o grau de similaridade. Para os perfis apresentados na Figura 1.3, a semelhança pelo coeficiente de *Kulczynski* é de 60%, enquanto que pelo método de *Jaccard* é de 27,27%.

É possível encontrar na literatura uma grande diversidade de outros coeficientes, os quais não são utilizados de uma maneira uniforme, bem como outras formas de análise

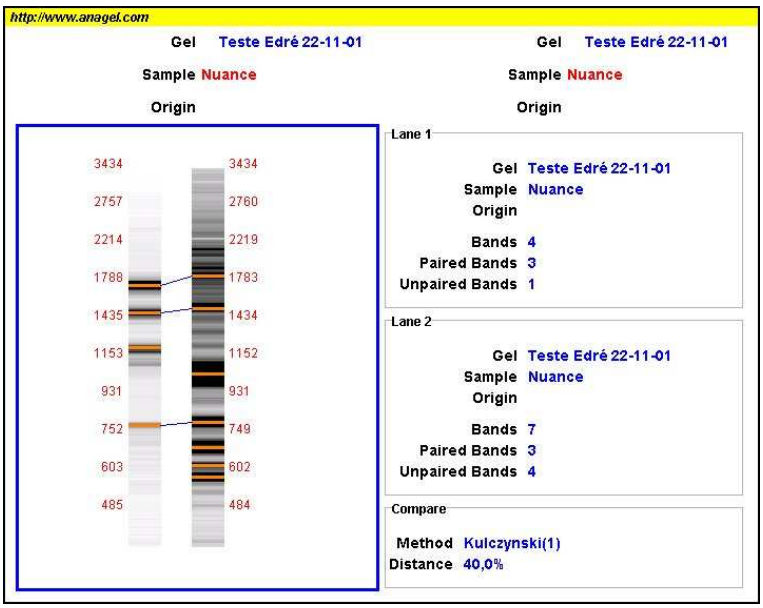

Figura 1.3: Cálculo da distância entre duas amostras.

que partem para a comparação direta dos tons de cinza dos perfis. Nada impede que um cientista desenvolva seu próprio método e o utilize em seus experimentos.

1.2 Contribuições do Trabalho

Neste trabalho explorou-se um modelo para desenvolvimento de *softwares* distribuídos, que possam funcionar independentemente uns dos outros, e que também sejam capazes de interagir quando necessário. Em síntese, as contribuições mais significativas desta dissertação são:

- Exploração de um modelo que possibilite a formação de repositórios globais virtuais de dados, constituídos por um conjunto de repositórios locais;
- Proposição de um ambiente para testes e validações de conjecturas que permita o uso de procedimentos criados pelo usuário, como uma métrica de distância entre registros experimentais, dentro de uma estrutura plenamente funcional e sem a necessidade de intervenção do desenvolvedor do sistema;
- Desenvolvimento de uma ferramenta útil e efetiva para armazenamento e análise de registros eletroforéticos, construída com base no modelo abordado;
- Exploração prática do uso de modernas técnicas de programação, como agentes móveis, introspecção e serialização de objetos, carga dinâmica de classes, etc. para a implementação dos conceitos.

1.3 Organização da Dissertação

O Capítulo 2 discorre sobre trabalhos desenvolvidos e tecnologias relacionadas com esta dissertação, como a utilização da *Web* para pesquisas científicas e tecnologias de comunicação para *softwares* distribuídos. Também são descritas algumas características de ferramentas comerciais para análise de registros eletroforéticos unidimensionais.

O desenvolvimento do sistema AnaGel tendo por base o modelo explorado é descrito no Capítulo 3. Neste capítulo são tratados aspectos detalhados do desenvolvimento de todos os módulos do *software*.

O Capítulo 4 apresenta as conclusões e resultados obtidos pelo desenvolvimento deste trabalho, bem como indica o que ainda pode ser feito para uma melhoria no sistema AnaGel.

O Apêndice A trata com um pouco mais de profundidade a eletroforese, o qual é indicado para leitores que desejem saber sobre o processo em mais detalhes.

O Apêndice B exibe o manual de utilização da versão do sistema AnaGel gerada como produto deste trabalho.

Por fim, no Apêndice C são dadas algumas características de *softwares* para análise de registros eletroforéticos comercializados atualmente.

Capítulo 2

Estado da Arte

2.1 A Internet como ambiente de Colaboração Científica

A Internet é ao mesmo tempo um mecanismo para disseminação de informação e um meio para colaboração e interação entre indivíduos e seus computadores sem a restrição de localização geográfica. Ela não foi projetada para apenas uma aplicação, mas como uma infraestrutura geral sobre a qual novas aplicações poderiam ser desenvolvidas, o que foi exemplificado anos após sua concepção pelo advento da *Web* [26].

2.1.1 Comunicação

O *e-mail* foi a primeira aplicação de grande uso desenvolvida para a Internet, tendo sido introduzido em 1972 pela necessidade dos desenvolvedores da ARPANET de um mecanismo de coordenação simples. É inquestionável a revolução causada no meio científico graças a esta forma de comunicação. Em adição ao e-mail, aplicações para transferência de arquivos, acesso remoto, transmissão de voz baseada em pacotes, vários modelos de compartilhamento de arquivos e discos e outras foram propostas no início da Internet [26].

Décadas mais tarde surgiu a *World Wide Web*, inicialmente planejada em 1990 para favorecer a comunicação entre a comunidade de físicos que pesquisava no CERN '(*Centre Européenne de Recherche Nucléaire* [1]) , os quais precisavam acessar bancos de dados comuns, trocar e editar documentos tais como artigos científicos, relatórios, etc. Todos possuíam acesso à Internet, mas necessitavam um *software* apropriado para prover tais funcionalidades. Hoje, a *Web* possui milhões de usuários acadêmicos e comerciais.

Várias aplicações para a *Web* foram sendo desenvolvidas ao longo do tempo de forma a atender às necessidades de seus usuários. Bibliotecas digitais, ferramentas de buscas, sites

[1] http://www.cern.ch

de comércio eletrônico, além de *softwares* plenamente funcionais podem ser hoje acessados remotamente.

Devido à característica de rápida propagação de informações e coordenação de numerosas e complexas interações em tempo real, a Internet tem o potencial de aprimorar a colaboração entre pesquisadores [38]. Um número crescente de projetos em medicina, biologia, física e outros têm surgido se aproveitando deste fato.

2.1.2 Compartilhamento de dados

Mais notoriamente na área biológica, vários bancos de dados centralizados são utilizados como repositórios globais. Esforços como o *PDB* e *GenBank* demonstram bem esta idéia.

O *Protein Data Bank (PDB)* [8] é um repositório mundial de dados estruturais de macromoléculas biológicas, sendo de responsabilidade do *Research Collaboratory for Structural Bioinformatics (RCSB)*. Biólogos, químicos, cientistas da computação, professores e alunos em todos os níveis, são os principais usuários.

Este repositório é utilizado para os mais diversos fins: determinação da estrutura tridimensional de uma proteína por homologia, procura por proteínas com estrutura tridimensional semelhantes à de estudo para descobrir as características (domínios funcionais conservados) da mesma, e outras. As funcionalidades são providas por *softwares* externos que acessam o banco, como o *Swiss-Model*, por exemplo.

Os dados para arquivamento são submetidos via e-mail ou pela *Web*, utilizando a ferramenta *ADIT*. Este último é um sistema integrado que ajuda a garantir que os dados submetidos estão consistentes com a representação entendida pelo *PDB*.

Alguns *softwares* de busca são disponibilizados no *site*, onde são acessados a partir de uma interface *CGI*, como o *Status Query*, *SearchLite* e *SearchFields*.

O *GenBank* [7] é um repositório de dados desenvolvido e distribuído pelo *National Center for Biotechnology Information (NCBI)*. Ele incorpora sequências genéticas de mais de 55.000 organismos diferentes. Os dados são recebidos de duas formas: (i) sequências submetidas por usuários individuais e (ii) grande volume de sequências submetidos por centros de sequenciamento. Virtualmente todos os registros são cadastrados por submissão eletrônica direta, utilizando os *softwares BankIt* (ferramenta para submissão para a *Web*) e *Sequin* (aplicação local de submissão para múltiplas plataformas).

Para recuperação dos registros armazenados, os usuários utilizam dos *softwares Entrez* e *BLAST*. *Entrez* é um sistema integrado ao banco de dados que acessa as sequências de DNAs e proteínas, permite a visualização de sequências em um mapa do genoma humano, e outros recursos. Já a família de programas *BLAST* é utilizada para cálculo do alinhamento local de sequências parametrizadas em relação àquelas presentes no banco de dados. Ambos serviços são providos através do *site* do *NCBI*, via servidor *Query Email*, e via clientes que

podem ser copiados do *site*.

Os usuários têm, dessa forma, um conjunto limitado de recursos que podem utilizar em suas pesquisas. A adição de novas funcionalidades (como diferentes algoritmos de alinhamento no *BLAST*, por exemplo) só é feita caso seja demandado pela comunidade como um todo, e não apenas pelo interesse de um usuário específico para seus experimentos.

2.1.3 Colaboratórios

Um *colaboratório* é um neologismo para designar sistema de pesquisa distribuído, criado através de uma rede de computadores, no qual cientistas de vários locais têm a habilidade de trabalharem juntos, com a assistência de tecnologias de comunicação e colaboração.

Pesquisadores em oceanografia, genoma de vermes e física espacial foram os primeiros a reconhecer o potencial de um colaboratório e a implementar os primeiros protótipos. As melhorias em *hardwares*, *softwares* e infraestrutura da tecnologia de redes proveram uma grande melhoria das condições para utilização de ferramentas baseadas na Internet para suportar pesquisas colaborativas.

A demanda por colaboratórios aumentou à medida que houve a necessidade de compartilhamento de conhecimentos e equipamentos pelos cientistas dispersos pelo planeta. A solução para o problema da epidemia global de AIDS, por exemplo, é sempre citado como uma pesquisa ampla, que requer um nível de cooperação muito grande de comunidades que, no passado, trabalhavam independentemente umas das outras (clínicos, ativistas, biólogos, etc.).

O restante desta seção descreve brevemente alguns casos de sucesso de colaboratórios, conforme descrito em [14]. Uma coleção de referências para os trabalhos aqui descritos podem ser encontradas no trabalho supra citado.

SCIENCEnet foi um serviço de rede proprietário iniciado em 1980 para suprir as necessidades de comunicação de pesquisadores em oceanografia. Assinantes da SCIENCEnet obtinham acesso aos seus colaboradores e aos dados. SCIENCEnet suportava lista de e-mails orientada a projetos que era usada para coordenar atividades entre os cientistas em diversas localidades. Adicionalmente, SCIENCEnet provia uma infraestrutura para o armazenamento e transporte de grandes conjuntos de dados.

Oceanógrafos colaboravam frequentemente para coordenar aquisição de dados de locais remotos. Também monitoravam arranjos de sensores automáticos, como bóias em oceanos ou sensores instalados nos círculos polares.

O *Worm Community System (WCS)* foi desenvolvido para o estudo do verme nematódio Caenorhabditis *elegans*, o qual possui algumas propriedades biológicas peculiares. Quando desenvolvido, a comunidade de cientistas que utilizava o colaboratório incluía 1400 pesquisadores em mais de 100 laboratórios. O *WCS* consistia em um conjunto de recursos ligados

por hipertexto, os quais usavam uma arquitetura *novel* antes da larga difusão da *WWW*. Estes recursos incluíam gráficos da estrutura física do *C. elegans*, um mapa genético, notas de pesquisa formais e informais, serviço de diretório, uma enciclopédia e um banco de dados chamado *acedb*. Um problema quanto ao uso do *WCS* foi o fato dos biólogos terem que fazer uma instalação relativamente complexa do sistema em um ambiente de computação pouco utilizado por eles.

O *Upper Atmospheric Research Collaboratory (UARC)* [31] foi iniciado em 1993 para servir à comunidade de físicos espaciais usuária de instrumentos instalados em um observatório localizado na costa da Groelândia. Os físicos interessavam-se no estudo da ionosfera terrestre. O objetivo do sistema era prover controle em tempo real dos instrumentos utilizados para estudar os eventos na atmosfera superior. Além disso, o *UARC* foi desenvolvido para suportar comunicação entre os pesquisadores distribuídos geograficamente, provendo dados em tempo real e acesso a dados arquivados.

O *Environmental Molecular Science Laboratory Collaboratory (EMSL)* teve seu início em 1993. *Environmental molecular science*, neste caso, se refere ao entendimento a nível molecular dos processos físicos, químicos e biológicos de descontaminação de solos e lençóis de água, processamento e distribuição de solo erodido, saúde humana e efeitos ecológicos de exposição a poluentes. O *EMSL* é dotado de recursos de dados (fita de armazenamento de arquivos de 20 terabytes), aparelhagem de ressonância magnética e espectômetro de massa. O colaboratório inclui aplicações para suportar operação remota dos aparelhos. Ainda, os usuários têm acesso a um conjunto de ferramentas de colaboração genéricas, incluindo quadro para escrita (*whiteboard*), salas de conversação, áudio e vídeo conferências e compartilhamento de aplicações. A versão corrente do sistema está implementada como uma aplicação *Java*.

Iniciado em 1998, *Great Lakes Regional Center for AIDS Research (GLRCFAR)* é um centro virtual para pesquisas sobre a AIDS que agrupa quatro universidades do meio oeste dos Estados Unidos: Northwestern, Minnesota, Wisconsin e Michigan. O colaboratório combina especialistas complementares dos vários centros, de tal forma que nenhum destes locais conseguiria sozinho reunir grupo com qualificação semelhante. Uma característica interessante do *GLRCFAR* é que ele representa a primeira tentativa de desenvolvimento de um colaboratório empregando componentes já existentes. Os participantes registram-se duas vezes por mês para uma série de seminários baseados no ambiente cooperativo, usando uma ferramenta denominada *PlaceWare*. Esta é uma ferramenta desenvolvida para a *Web* que simula uma sala de apresentações. Em adição a este serviço, os cientistas podem interagir via *NetMeeting*, tanto para escrever protocolos clínicos e propostas, quanto para ter visualização ao vivo de saídas de instrumentos remotos (imagens de tecidos de pacientes feitas por microscópio eletrônico, por exemplo). Por fim, o *GLRCFAR* é um repositório virtual de documentos e dados relevantes para o trabalho conjunto das quatro instituições

participantes.

O *Space Physics and Aeronomy Research Collaboratory (SPARC)* é um sucessor do projeto *UARC*. Visto por uma perspectiva de implementação, *SPARC* foi desenvolvido como uma aplicação cliente leve *(thin client)*. Isto significa que os usuários acessam todas as funcionalidades do colaboratório através de um navegador *Web* convencional, ao invés de um *software* especializado.

2.2 Características de Ferramentas de Armazenamento e Análise de Registros Eletroforéticos

A Figura 2.1 exibe um quadro comparativo dos *softwares* para análise e armazenamento de registros eletroforéticos mais utilizados atualmente (vide Apêndice C), onde a disponibilização de uma dada funcionalidade é marcada com "x":

2.3 Estruturas Sintáticas e Ferramentas para Programação Distribuída

2.3.1 Tecnologias de comunicação cliente/servidor

Esta seção descreve brevemente algumas tecnologias comumente utilizadas para interações em aplicações cliente/servidor, discutidas na referência [19].

Os métodos dominantes para comunicação em aplicações cliente/servidor são *mensagens* [12] e *remote procedure call (RPC)* [9]. A distinção primária entre estes métodos está no fato de serem protocolos *assíncronos*, como no caso de mensagens, ou *síncronos*, como ocorre no RPC.

Em ambos os tipos de comunicação, o cliente e o servidor trocam dados os quais serão processados por procedimentos específicos na CPU remota. Nenhuma das partes especifica *como* os dados serão processados - cada uma tem um conhecimento implícito das capacidades dos procedimentos remotos.

O RPC estende o mecanismo de invocação de procedimentos tradicional de empilhar parâmetros, registradores e endereço de retorno e então desviar o fluxo de execução para o ponto de entrada do procedimento. No caso do RPC, um canal de comunicação é aberto entre a aplicação cliente e o processo servidor. Os parâmetros são passados para uma rotina de interface, a qual os codifica em uma forma apropriada para transmissão e os envia para o processo servidor. Os dados são recebidos por uma rotina de interface correspondente, decodificados e repassados ao procedimento servidor. O procedimento processa os parâmetros e, na maioria dos casos, produz um valor de retorno, o qual é transmitido de

Software / **Característica**	Phoretix	TotalLab	Gel-Pro	GelSite	GeneTools	SigmaGel
Detecção automática de canaletas	x	x	x	x	x	
Detecção automática de bandas	x	x	x	x	x	x
Correção de distorções em géis	x	x	x	x	x	x
Determinação de quantidade de massa das bandas	x	x	x		x	x
Determinação de peso molecular das bandas	x	x	x	x	x	x
Calibração Rf *(Retardation factor)*	x	x	x	x	x	x
Elaboração de histogramas	x	x	x	x	x	x
Construção de dendrogramas	x				x	
Cálculo de pontos isoelétricos	x	x				
Compartilhamento de bandas em um mesmo gel	x				x	
Armazenamento local dos dados	x	x	x	x	x	
Interação com o ambiente de trabalho	x	x	x	x	x	x
Tratamento da imagem	x	x	x		x	x
Múltiplas canaletas de calibração	x	x	x		x	x
Listas padrões de pesos moleculares	x	x	x		x	
Anotações sobre a imagem	x	x	x			
Personalização de algoritmos para cálculo de pesos moleculares			x			
Cálculo de coeficientes de similaridade					x	

Figura 2.1: Tabela comparativa dos *softwares* para análise de eletroforese.

volta ao processo cliente. Ambas as partes devem usar uma definição de interface comum, muito embora seja possível heterogeneidade de *hardware* e sistema operacional.

Enquanto uma chamada a procedimento local pode ser executada em poucos microsegundos (não incluindo o tempo de execução do procedimento em si), o RPC introduz um *overhead* para codificação, transmissão e decodificação do lado servidor, tendo tipicamente uma latência de alguns milisegundos.

No caso de comunicação via *mensagens*, a aplicação cliente escreve uma mensagem, tipicamente composta de texto estruturado ou contendo *tags*, a qual será enviada para um *software* de processamento indicado em seu cabeçalho. A mensagem é geralmente endereçada indiretamente, de forma que o cliente pode não saber o endereço de rede explícito ou mesmo a identidade do servidor de destino. A resolução do endereço é feita em passos intermediário do processamento.

Mensagens são inerentemente assíncronas - uma vez que o cliente tenha enviado a mensagem, ele continua sua execução. Se no futuro o cliente receber uma mensagem de resposta do servidor, ele deve restaurar o estado da aplicação para processar a resposta.

Uma vez que a comunicação é assíncrona, a latência em mensagens é ao mesmo tempo maior e menos previsível que no caso de RPC. Como resultado, mensagens podem ser menos efetivas para comunicação de um para um do que RPC, mas para comunicação de um para muitos, a qual é típica em serviços de rede, o aproveitamento pode ser maior, uma vez que o processo cliente não precisa ser suspenso enquanto espera pela resposta.

O ponto chave em relação a RPC está em sua alta eficiência e baixa latência, enquanto que no caso de mensagens reside no fato de ser robusta, particularmente em *wide-area networks*.

Outras técnicas mais avançadas foram desenvolvidas tendo-se como base *mensagens* e *RPC*, que é o caso das tecnologias *CORBA* [32] e *RMI* [10].

2.3.2 Agentes móveis

Definição

Agentes móveis são programas, na maioria das vezes escritos em linguagens interpretadas, que podem ser disparados em um computador cliente e transportados para um servidor remoto onde executam suas rotinas [19].

Cada agente tem sua própria linha de execução de forma que pode desacoplar-se do sistema onde está sendo executado e migrar para outro computador para continuar sua execução de maneira autônoma.

Em [5], agentes móveis são definidos como *entidades de software autônomas que podem interagir com o ambiente*. Isto significa que são autônomos e podem agir com outras entidades em vários ambientes e através de várias plataformas.

Basicamente, agentes são padrões de desenvolvimento de *software*. Ferramentas, linguagens e ambientes podem ser desenvolvidos especificamente para suportar este padrão baseado em agentes. Entretanto, este padrão de desenvolvimento pode também ser implementado usando ferramentas, ambientes e linguagens orientadas por objetos, ou qualquer outra ferramenta, linguagem e ambiente que seja capaz de suportar entidades de *software* autônomas, interativas e adaptativas.

Conceitos

Alguns conceitos básicos para o entendimento da tecnologia de agentes são dados em [4].

O *estado* de um agente pode ser ou seu estado de execução ou os valores dos atributos que determinam o que fazer quando a execução for retomada. Os valores dos atributos do agente incluem o estado do sistema de agentes associado a ele (tempo de vida, por exemplo).

O *estado de execução* de um agente é seu estado em tempo de execução, incluindo o contador de programa e sua pilha de execução.

A *autoridade* de um agente identifica a pessoa ou organização para a qual o agente está atuando. Uma autoridade deve ser autenticada.

Agentes requerem *nomes* para serem identificados em operações de gerenciamento e para poderem ser localizados via um *naming service*. Agentes são nomeados por sua autoridade, identidade e tipo de sistema de agentes. Uma identidade é um valor único dentro do escopo da autoridade que identifica uma instância particular do agente. A combinação da autoridade, identidade e tipo de sistema de agentes sempre forma uma identificação global única.

A *localização* de um agente é o endereço de um *local*. Um *local* está contido dentro de um sistema de agentes. Assim, a localização do agente deve conter o nome do sistema de agentes e do local onde o agente reside. Normalmente, quando a localização não contém o nome do local, o sistema de agentes aloca o agente em um local padrão.

Um *sistema de agentes* é uma plataforma que cria, interpreta, executa, transfere e finaliza um agente. Assim como um agente, um sistema de agentes está associado a uma autoridade que identifica a pessoa ou organização para a qual o sistema de agentes age.

O *tipo* do sistema de agentes descreve o perfil de um agente. Por exemplo, se o tipo do sistema é *Aglet* [23], o sistema de agentes é implementado pela IBM, suporta Java como linguagem para os agentes e usa serialização de objetos do Java. Um cliente requisitando uma função do sistema de agentes deve especificar o perfil do agente para identificar unicamente a função desejada.

Toda comunicação entre sistemas de agentes é através de uma *infraestrutura de comunicação*. O administrador da região define os serviços para comunicação interna e externa.

Uma infraestrutura de comunicação provê serviços de transporte (por exemplo RPC), *naming* e serviços de segurança para um sistema de agentes.

Quando um agente se auto transfere, ele viaja por ambientes de execução denominados *locais*. Um local é um contexto dentro de um sistema de agentes no qual um agente pode executar. Este contexto pode prover funcionalidades, como por exemplo controle de acesso. Os locais de origem e de destino podem residir no mesmo sistema de agentes ou em sistemas diferentes que suportem o mesmo perfil do agente. Um sistema de agentes pode conter um ou mais locais e um local pode conter um ou mais agentes.

Uma *região* é um conjunto de sistemas de agentes que têm a mesma autoridade, mas que não são necessariamente do mesmo tipo. O conceito de região permite mais que um sistema de agentes representar a mesma pessoa ou organização. Uma região provê um nível de abstração para clientes de outras regiões se comunicarem.

Interoperabilidade entre agentes

Como descrito na especificação da *OMG - Object Management Group* [2] - sobre agentes móveis [4], a diferença entre os diversos sistemas de agentes móveis dificulta a interoperabilidade e a rápida proliferação desta tecnologia, e tem provavelmente impedido o crescimento da indústria de agentes. Para promover a interoperabilidade entre os sistemas, alguns aspectos da tecnologia de agentes móveis devem ser padronizados.

Uma importante questão na tecnologia de agentes móveis é a interoperabilidade entre os sistemas de agentes dos vários fabricantes. Interoperabilidade torna-se mais alcançável se ações como transferência de agentes, transferência de classes e gerenciamento de agentes forem padronizadas.

Mobile Agent System Interoperability Facilities [4] (também denominado *MASIF*) trata sobre interoperabilidade entre sistemas de agentes implementados com a mesma linguagem, mas por diferentes fornecedores, e sistemas que necessitam de atualizações constantes. O suporte para diferentes linguagens pode ser dado em cada extremo.

A especificação aborda a padronização dos seguintes aspectos comuns aos diversos sistemas de agentes:

Gerenciamento de agentes - deve-se permitir a criação de agentes uma vez fornecido o nome de sua classe; suspensão, reativação e término da *thread* de execução de um agente de uma forma padrão. Definindo funções de gerenciamento comuns, um administrador pode gerenciar sistemas de agentes de diversos tipos.

Transferência de agentes - é vantajoso para dois agentes que se comuniquem na mesma localidade em invés de o fazerem através da rede por duas razões básicas: (i) número

[2] http://www.omg.org

de transações pela rede pode ser alto (e lento) e (ii) monitoramento dos dados pode levar dias. Permitindo a um agente fonte mover-se para um sistema de agentes mais próximo ao agente com o qual ele deseja se comunicar torna possível o aproveitamento da localidade.

Nomes de agentes e sistemas de agentes - quando invocada uma operação de gerenciamento, o agente sendo manipulado deve ser identificado. Assim, a sintaxe do nome dos agentes deve ser padronizada. Uma sintaxe padrão para a nomenclatura também provê dois outros benefícios. Primeiro, possibilita a um sistema de agentes determinar rapidamente se ele suporta um agente que deseja se instalar. Segundo, permite que dois agentes se identifiquem pelo nome.

Sintaxe do tipo do sistema de agentes e localização - a sintaxe da localização deve ser padronizada de modo que um agente possa acessar o tipo dos sistemas de agentes destino de uma mesma forma. A garantia de unicidade do nome dos tipos de sistemas de agentes prevê que duas companhias dupliquem os valores de seus tipos.

Interação entre agentes

Os tipos comuns de interação entre agentes relacionados com a interoperabilidade são:

Criação remota de agentes - um programa cliente interage com o sistema de agentes de destino para requisitar que um agente seja criado a partir de uma determinada classe.

Transferência de agentes - um agente solicita sua transferência para um determinado sistema de agentes. Para tanto, é necessário que o serviço de comunicação satisfaça às restrições exigidas pelo agente. Neste caso, a execução do agente é suspensa, o agente é transferido e sua execução é recomeçada no sistema de destino.

Invocação de método de agentes - um agente invoca um método de outro agente ou objeto, caso tenha a referência para aquele e autorização para fazê-lo. A infraestrutura de comunicação deve realizar a operação e retornar o resultado, caso as condições do serviço estejam de acordo com o exigido, ou indicar falha na execução.

Segurança

Agentes são entidades de *software* que frequentemente executam em um ambiente de computação distribuído e interagem com muitas outras entidades de *software*, incluindo outros agentes. Neste tipo de situação são muitos os aspectos de segurança a serem considerados. A possibilidade de encontrar problemas de segurança é claramente maior em ambientes abertos, como na Internet.

A referência [5] faz uma abordagem sobre os aspectos de segurança envolvidos ao ser utilizada a tecnologia de agentes móveis.

Geralmente, a palavra *segurança* refere a ações tomadas para garantir que alguma coisa esteja livre de perigo, não possa ser roubada, perdida ou danificada. Na prática, a segurança é usualmente aplicada apenas a itens que têm valor, uma vez que possui um certo custo associado.

Políticas de segurança referem-se a como é controlado o acesso a recursos importantes do sistema. Sistemas de agentes requerem políticas de segurança, as quais podem controlar onde os agentes estão aptos a executar, o que podem fazer e com quais outras entidades eles têm a habilidade de se comunicar.

As políticas de segurança são normalmente baseadas em *identidades*, as quais são simplesmente algo que serve para identificar uma entidade. Assim, um agente pode ser identificado pelo seu nome, por ser membro de alguma organização, etc. Identidades são baseadas em *credenciais*, que são um conjunto de dados que podem ser validados por uma terceira entidade para provar que a entidade a ser identificada é quem diz ser. Alguns mecanismos para identificação incluem certificados X.509 e chaves PGP.

Alguns ataques que podem ocorrer em sistemas de agentes são:

Revelação não autorizada - exposição dos dados privados de um agente, onde podem estar codificados dados sensíveis, como números de cartões de crédito, resultados de operações sigilosas, etc. Alguma entidade pode observar ocultamente a interação entre agentes e extrair informações sobre seus objetivos, planos, ou outras informações que pertencem aos agentes.

Alteração não autorizada - modificação da funcionalidade de um agente, seu estado ou seus dados. O conteúdo das mensagens pode ser interceptado, modificado, e repassado durante a transmissão.

Danificação - destruição de arquivos do servidor, de configurações, ou de um agente e sua missão.

Cópia ou retransmissão não autorizada - tentativa de copiar um agente e clonar ou retransmití-lo. Uma plataforma maliciosa pode criar uma cópia ilegal ou o clone de um agente, ou a mensagem originária de um agente pode ser ilegalmente copiada e retransmitida.

DoS *(Denial of service)* - ataque que tenta sobrecarregar o servidor, de forma a impedir que outros o utilizem. Um agente pode enviar diversas requisições, e o agente alvo fica impossibilitado de atender às requisições de outros.

Repudiação - um agente ou plataforma de agentes nega que tenha recebido ou enviado mensagens ou executado uma ação específica. Um encontro entre dois agentes como resultado da negociação de um contrato pode ser ignorado após o evento ter decorrido. Assim, o agente nega que a negociação tenha sido feita e se recusa cumprir sua parte do acordo.

Spoofing - um agente ou plataforma de agentes não autorizada assume a identidade de outra. Um agente pode se registrar como prestador de serviço de diretório e receber informações de outros agentes que tentem se registrar.

Estes ataques podem ocorrer não apenas a sistemas de agentes, mas a qualquer sistema distribuído que dependa de transmissão de mensagens que sejam confiáveis. A referência [37] aborda questões relacionadas a segurança de dados e de redes em geral, apresentando os ataques mais comuns e técnicas que podem ser utilizadas para evitá-los.

Com exceção do *DoS*, os demais ataques podem ser dificultados utilizando técnicas de criptografia e assinatura digital. Algoritmos de criptografia, sejam de chave pública/privada ou simétrica, podem ser utilizados para complicar o entendimento dos dados transmitidos, tornando possível o conhecimento do conteúdo em tempo hábil apenas pelas partes interessadas.

Assinaturas digitais podem ser utilizadas para verificação da integridade da mensagem transmitida, autenticação e garantia de participação da outra entidade nas eventuais negociações.

Por fim, políticas de controle de acessos baseadas nas credenciais do cliente podem ser utilizadas pelos sistemas que provêem os serviços, de modo a evitar que *softwares* maliciosos acessem e alterem recursos sensíveis.

Considerações quanto ao uso de agentes

Um agente móvel não está acoplado ao sistema no qual ele inicia sua execução. Contrariamente, ele é livre para trafegar entre os servidores na rede. Criado em um ambiente, ele pode transportar seu *código* e *estado* para outro, onde continua sua execução. *Estado* neste sentido significa os valores dos atributos do agente que determinam o que fazer quando a sua execução for retomada no ambiente de destino. *Código*, no contexto de orientação por objetos, significa a definição da classe que deu origem àquele agente, necessário para a sua execução.

Segundo [24], existem pelo menos sete bons motivos para usar agentes móveis, muito embora sempre possamos imaginar situações onde a aplicação desta tecnologia não traz os benefícios desejados:

1. **Redução de tráfego na rede -** Sistemas distribuídos frequentemente baseiam-se em protocolos de comunicação que envolvem interações múltiplas para cumprir uma

determinada tarefa, o que resulta no tráfego de muitos dados pela rede. Agentes móveis permitem aos usuários encapsular o algoritmo de conversação e enviá-lo a um servidor de destino, onde a interação será feita localmente. Outro cenário é aquele onde um grande volume de dados está armazenado em um servidor remoto e deve-se fazer um processamento sobre estes. O usuário pode, neste caso, enviar um agente contendo o algoritmo, fazer o processamento localmente, e retornar apenas o resultado, evitando que dados não processados trafeguem desnecessariamente pela rede. O princípio de processamento de dados baseado em agentes é simples: *mova a computação até os dados ao invés dos dados à computação.*

2. **Redução da latência de rede -** Sistemas de tempo real precisam responder imediatamente a mudanças em seus ambientes. Controlar tais sistemas através da rede de uma empresa de tamanho substancial envolve uma latência significativa. Para sistemas com estas características, tal latência não é aceitável. Agentes móveis oferecem uma solução, uma vez que podem ser disparados de um controlador central para agir localmente e responder às mudanças diretamente.

3. **Encapsulamento de protocolos -** Quando dados são trocados em um sistema distribuído, cada servidor possui o código que implementa os protocolos necessários para enviar e interpretar corretamente os dados que chegam. Entretanto, como protocolos expandem-se para agregar novos requisitos de eficiência ou segurança, é incômodo, senão impossível, atualizar o protocolo corretamente. Como resultado, protocolos sempre se tornam um problema legado. Agentes móveis, por outro lado, podem mover-se para servidores remotos para estabelecer canais de comunicação baseados em protocolos proprietários.

4. **Execução assíncrona e autônoma -** Dispositivos móveis normalmente baseiam-se em conexões de rede caras ou frágeis. Tarefas que requisitam uma conexão continuamente aberta entre um dispositivo móvel e uma rede fixa são provavelmente não econômicas ou tecnicamente impraticáveis. Para resolver este problema, tarefas podem ser encapsuladas em agentes móveis, os quais podem ser enviados pela rede. Após o envio, os agentes se tornam independentes do processo que os criou e podem operar assincronamente e autonomamente, sem a necessidade da conexão aberta. O dispositivo móvel pode se reconectar em outro instante para que o agente seja coletado.

5. **Adaptação dinâmica -** Agentes móveis podem perceber seus ambientes de execução e reagir autonomamente a mudanças. Múltiplos agentes móveis têm a habilidade exclusiva de se auto distribuírem entre os servidores na rede para manterem a configuração ótima para resolução de um problema em particular.

6. **Heterogeneidade -** Computação em rede é fundamentalmente heterogênea, tanto no que diz respeito a *hardware* quanto a *software*. Uma vez que agentes móveis são geralmente independentes da máquina e da camada de transporte (são dependentes apenas de seus ambientes de execução), eles provêem excelentes condições para integração de sistemas desta natureza.

7. **Robustez e tolerância a falhas -** A habilidade de agentes móveis de reagir dinamicamente a situações e eventos, incluindo aqueles desfavoráveis, torna fácil desenvolver sistemas distribuídos robustos e tolerantes a falhas. Se um servidor está sendo desligado, todos os agentes em execução naquela máquina são alertados e a eles é dado um tempo para migrarem e continuarem suas operações em outro servidor na rede.

A referência [19] aborda o uso de agentes móveis de uma forma bem ponderada. Os autores examinam vários argumentos que são dados a favor de agentes móveis, compara as vantagens individuais defendidas com métodos alternativos de atingir o mesmo resultado, e também considera os ganhos gerais alcançados através do uso da tecnologia de agentes para serviços de rede.

Os autores discutem algumas questões envolvidas ao serem utilizados agentes móveis, como avaliação da eficiência, flexibilidade e segurança. Segundo os autores, *"... as a general statement, we have not discovered any client server functions which are important for network services and which are uniquely enabled by the use of mobile agents"*.

Especificamente para o problema abordado, a utilização de agentes móveis apresentou-se como uma solução adequada, examinando-se os itens descritos no início desta seção e as observações feitas em [19].

Agentes móveis e Java

A referência [41] apresenta a linguagem Java como uma excelente escolha para implementação de agentes móveis. Nela, o autor defende vários argumentos em favor desta hipótese.

O suporte a múltiplas plataformas e a promessa de *escreva uma vez, execute em qualquer lugar*, faz de Java uma excelente opção para a tecnologia de agentes móveis. Além disso, a onipresença da máquina virtual Java pode, algum dia, facilitar a disseminação de agentes móveis através da Internet.

Java possui várias características que não são encontradas nas outras linguagens que suportam diretamente a implementação de agentes móveis. Por exemplo, a mobilidade dos agentes requer a habilidade de converter um agente e seu estado em uma forma apropriada para transmissão pela rede e de reconstruí-lo no servidor de destino. A serialização de objetos do Java [11] supre esta necessidade de uma forma quase que transparente.

Alguns sistemas de agentes móveis também provêem informação persistente do estado do agente. Persistência é suprida através da serialização do estado daquele agente, armazenando-o em um dispositivo persistente (disco rígido, banco de dados, etc.), e recuperando-o para reconstruir o agente em outra ocasião.

O suporte a rede do Java inclui *sockets*, comunicação por URLs e um protocolo para objetos distribuídos chamado *invocação de métodos remotos (RMI)* [10]. Aplicações *RMI* são normalmente compostas de dois programas distintos: um cliente e um servidor. Tipicamente o servidor cria alguns objetos, torna suas referências acessíveis, e aguarda a invocação de métodos nestes objetos pelos clientes. Uma aplicação cliente recupera referências para acesso aos objetos remotos e então invoca métodos nestes objetos. *RMI* provê um mecanismo pelo qual o cliente e o servidor se comunicam e trocam informações.

Adicione-se o fato de que Java facilita a migração do código via seu mecanismo de carregamento de classes. Carregadores de classes Java buscam dinamicamente as classes incluídas em uma aplicação, localmente através do *classpath* (lista de diretórios) ou através da rede. Para carga dinâmica das classes que compõem o agente, um carregador de classes especializado provê várias opções:

- A forma serializada do agente pode incluir sua classe como também aquelas referenciadas por ele.
- As classes que definem um agente podem ser carregada de um servidor *Web* ou de outro servidor.
- As classes de um agente podem ser carregadas através do *classpath*.

Todo código requerido através do carregador de classes está sujeito a restrições de segurança, as quais são de grande utilidade para sistemas de agentes móveis que têm que proteger seus agentes (e os servidores nos quais eles executam) de acessos não autorizados. O gerenciamento de segurança do Java suporta desenvolvimento de políticas de segurança altamente configuráveis e bem detalhadas para os diversos recursos [10]. Por exemplo, agentes disparados por um usuário em particular podem ter permissão para escrever arquivos, enquanto que aqueles disparados por outro podem ter acesso apenas para ler, e aqueles disparados por um terceiro usuário podem não ter acessos a arquivos.

Java também suporta o desenvolvimento de agentes que são altamente acoplados com a *Web*. *Applets* [11] podem disparar agentes móveis por navegadores da *Web* e podem receber os agentes disparados por elas após a finalização da execução remota. Java provê ainda uma tecnologia de servidor denominada *servlet* [10]. *Servlets* funcionam como um script CGI e podem disparar e receber agentes móveis.

O serviço de diretórios do Java - *JNDI* [25] - permite conectividade a informações de negócio através de um acesso unificado a múltiplos serviços de diretório e nomes. Agentes

móveis podem, por exemplo, usar provedores de serviços *JNDI* para localizar os serviços que eles precisam e então conectar-se aos sistemas legados.

Existem vários sistemas de agentes implementados para a linguagem Java. Dentre os utilizados atualmente podemos citar o *Voyager* da *Recursion Software, Inc.*[3], *Aglet* da IBM [23], *Grasshopper* da empresa alemã *IKV++ Technologies* [1, 21], dentre outros.

[3]http://www.recursionsw.com

Capítulo 3

O Sistema AnaGel

3.1 Descrição do Modelo

Na grande maioria dos sistemas, sejam eles isolados ou cliente/servidor, os usuários estão restritos aos algoritmos fornecidos pelo implementador. Em se tratando de trabalhos científicos, este fato pode forçar o pesquisador a procurar alternativas mais complexas para atingir seu objetivo ou até mesmo impedí-lo de continuar seu projeto. Para estes tipos de *software*, o usuário é obrigado a solicitar ao desenvolvedor a inclusão da funcionalidade desejada no sistema, o que pode não ser algo imediato.

O modelo estudado objetiva dar uma maior flexibilidade ao pesquisador para utilização de recursos remotos. Neste sentido, ele não fica limitado aos algoritmos do sistema servidor, sendo possível o envio de rotinas próprias para trabalhar sobre os dados. Dentre outros ganhos, um ambiente construído desta forma pode ser facilmente utilizado para testes e validação de conjecturas, sendo de grande utilidade para desenvolvimento de trabalhos científicos.

O cientista, em seu próprio laboratório, desenvolve seu algoritmo e solicita para que seja executado nos diversos laboratórios que compõem a rede. Após a execução nos servidores remotos, os resultados são remetidos de volta, onde o pesquisador pode avaliá-los e tirar suas conclusões.

A arquitetura explorada possibilita a criação de colaboratórios compostos por diversas entidades, as quais podem disponibilizar recursos dos mais diversos para o cientista utilizar da forma que lhe convier. Uma aplicação que se destaca é sua utilidade para criação de repositórios de dados globais virtuais, tal qual um *PDB* ou *GenBank* (Seção 2.1.2) distribuído e também mais flexível.

Para a implementação desta funcionalidade optou-se pela utilização de agentes móveis principalmente por ser uma tecnologia a qual desejava-se ter um maior conhecimento.

Ademais, a arquitetura apresentava-se como uma excelente aplicação para esta tecnologia (vide Seção 2.3.2). Aqui os agentes não servem simplesmente como "mensageiros", mas também utilizam o poder computacional do servidor de destino.

A Figura 3.1 mostra o funcionamento do modelo como um todo, exibindo um sistema hipotético com um servidor local (*Servidor Local* **A**) atendendo a diversos clientes e outros servidores (*Servidores Locais* **B** *e* **C**) que podem, eventualmente, atender a seus clientes. Cada servidor pode comunicar-se com um sistema de agentes (*Servidor* **A** $\rightarrow$ *Agência* **A**, *Servidor* **B** $\rightarrow$ *Agência* **B**, etc.), o qual possui a capacidade de enviar unidades de *software* para as demais agências que compõem a rede (*Agências* **B** *e* **C**) e interagir com seus respectivos servidores. Neste exemplo, um algoritmo é enviado pelo *Cliente 1*, o qual recebe o resultado final do processamento.

O pesquisador, a partir de sua estação, implementa o algoritmo segundo seus critérios, obedecendo apenas à interface de métodos permitidos. O módulo implementado é, então, submetido ao servidor **(1)** para que possa ser utilizado nos laboratórios remotos.

O algoritmo juntamente com outros dados necessários são repassados ao sistema de agentes **(2)**(aqui também denominado *agência*), o qual se encarrega de encapsular as informações em agentes móveis. Um agente é enviado para cada laboratório destino **(3a, 3b)**, sendo que todos são disparados simultaneamente, acrescentando um grau de paralelismo considerável em se tratando de cooperação com vários laboratórios remotos.

Uma vantagem observada com este modelo é que o próprio sistema de agentes se encarrega de localizar as agências dos demais laboratórios, dispensando a implementação de rotinas para tal fim.

Ao serem recebidos nos sistemas de agentes dos laboratórios, os agentes comunicam-se localmente com o servidor, repassando os dados e o algoritmo a ser utilizado naquele instante **(4a, 4b)**. Para tanto, o servidor deve possibilitar a carga do algoritmo enviado **(5a, 5b)**, o qual deve ser acessado através de uma interface previamente definida.

O próprio agente passa a interagir com o servidor, possibilitando o processamento dos dados localmente. Este fato traz ganhos à medida que diminui o tráfego de dados não processados pela rede.

Uma vez construído o resultado, este é armazenado pelo agente, o qual retorna para a agência de origem **(6a, 6b)**. Este processo ocorre paralelamente em cada servidor remoto, dado que cada agente é responsável por um deles. Uma vez que o processamento é local, não há a necessidade de uma conexão aberta com o servidor de origem durante a execução - o agente pode fazer seu processamento e retornar mais tarde, quando sua computação for terminada e a rede estiver disponível.

Na agência que originou a requisição os resultados são agrupados e repassados para o servidor local **(7)**, que por sua vez devolve à aplicação cliente do pesquisador **(8)**.

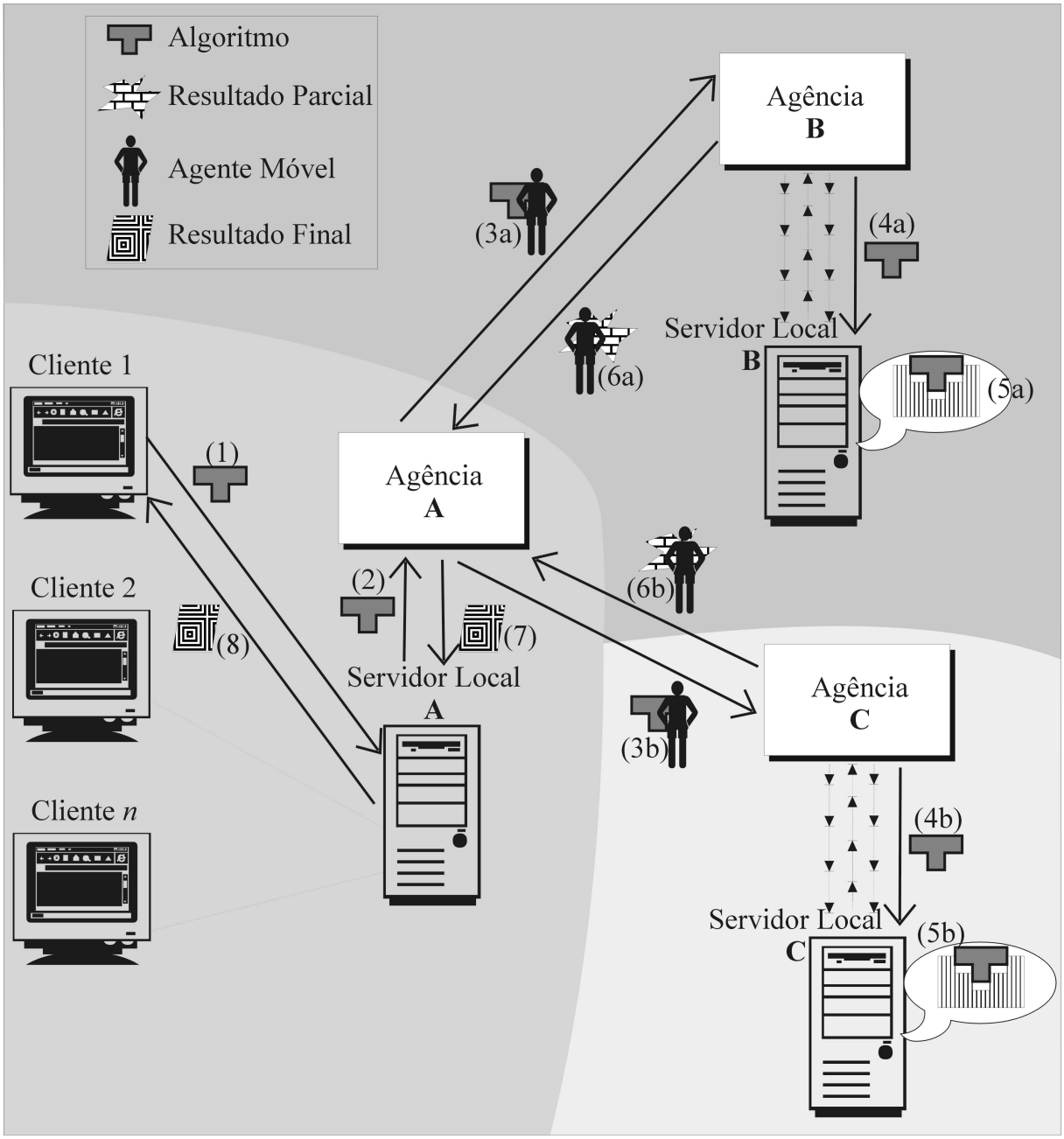

Figura 3.1: Arquitetura do modelo.

3.2 Plataforma de Desenvolvimento

3.2.1 Java

A plataforma de programação *Java* [27] foi criada para desenvolvimento e execução de *applets* e aplicações altamente interativos, dinâmicos e seguros, com um foco especial em sistemas de rede. O que diferencia a plataforma *Java* é o fato de esta prover sua funcionalidade em cima de outras plataformas de software e hardware. Isto é feito através

da geração de um código intermediário, o qual não é específico para quaisquer equipamentos ou sistemas operacionais, mas que contém instruções para um interpretador.

A linguagem Java foi escolhida para a implementação do sistema por diversos motivos, dentre os quais destacam-se:

- Possibilidade de execução pelo *Web* como *applet*;
- Execução em qualquer plataforma para a qual exista uma máquina virtual Java (*JVM*);
- Existência de biblioteca de componentes visuais bastante flexíveis que facilitam o desenvolvimento de interfaces gráficas;
- Disponibilização de primitivas para comunicação em rede e sincronização de processos;
- Funcionalidades para serialização de objetos;
- Possibilidade de carga dinâmica de classes para utilização de algoritmos do usuário;
- Desenvolvimento orientado por objetos, possibilitando redução no tempo de implementação e facilitando manutenção e atualização do *software*;
- Existência de diversos sistemas de agentes que a suportam.

3.2.2 Informix

Como servidor de banco de dados relacional foi escolhido o *Informix Dynamic Server* [3]. A escolha deu-se baseada na disponibilidade de acesso ao mesmo, uma vez que o CENAPAD-MG/CO possui uma licença de uso, sendo desnecessária a aquisição de outros *softwares* para este fim, como *Oracle*, *SQL Server*, etc.

É importante salientar que o servidor AnaGel não usa funcionalidades específicas deste banco de dados, possibilitando a troca por outro que seja capaz de armazenar os mesmos tipos de dados sem grandes modificações no código. Todos as consultas *SQL* foram construídas sem levar em consideração o servidor de banco de dados em questão para prover esta independência, mesmo porque servidores instalados nos diversos laboratórios provavelmente utilizarão bancos de dados de menor âmbito.

3.2.3 Grasshopper

O sistema de agentes *Grasshopper* [1] é um ambiente para criação, hospedagem, transmissão e execução de agentes móveis para a linguagem Java. A utilização de uma linguagem

não proprietária dá um grande incentivo à utilização deste sistema, uma vez que o conhecimento necessário para operá-lo pode ser adquirido de uma forma bem amigável, sem a necessidade da introdução de uma linguagem específica para ele - ao desenvolvedor cabe apenas aplicar os conceitos de agentes móveis e as definições do sistema utilizando uma linguagem que lhe servirá não apenas para este propósito, mas para implementar soluções para outros problemas onde a utilização de agentes não é necessária.

A independência de plataforma da linguagem Java é outro ponto de destaque. O sistema pode ser implantado em uma rede heterogênea, constituída por máquinas que executam diferentes sistemas operacionais, sem a necessidade de aquisição de uma compilação específica para cada plataforma. A única exigência é que a máquina virtual Java esteja instalada.

O sistema disponibiliza um mecanismo para comunicação e localização de agentes utilizando *proxy*, o que facilita o envio de mensagens para uma determinada entidade mesmo que ela migre para outra máquina da rede. *Proxies* também podem ser utilizados para comunicação com aplicativos legados, tornando possível a uma aplicação externa manipular o sistema de agentes de uma forma programável, utilizando em conjunto a *API* provida para tal fim.

O modelo de segurança adotado pelo *Grasshopper* aproveita-se do modelo de segurança da linguagem Java. Isto possibilita a definição de políticas de segurança muito bem detalhadas para cada usuário do sistema, provendo mecanismos de autenticação e segurança de dados de uma forma bastante flexível.

Por fim, um ponto interessante neste *software* é a possibilidade de utilização de módulos adicionais para compatibilização com a especificação *MASIF* da *OMG*, e *FIPA (Foundation of Intelligent Physical Agents)*, que visam tornar intercambiáveis agentes escritos em uma mesma linguagem criados em sistemas de agentes diferentes.

Os principais motivos que levaram à escolha do sistema de agentes *Grasshopper* foram compatibilidade com a versão atual do Java (1.3.1 quando do desenvolvimento do sistema), possibilidade de uso sem ônus para fins acadêmicos, além de implementar os conceitos de agentes móveis descritos na Seção 2.3.2 e de ser um *software* bem documentado e de fácil uso.

3.3 Estrutura do AnaGel

O sistema AnaGel (Apêndice B) foi originalmente implementado como um *software* para uso local [34, 33], disponibilizando uma gama de opções para o processamento de géis de eletroforese (Apêndice A), armazenamento dos géis analisados em um banco de dados local, e algoritmos pré-definidos para comparação de canaletas de géis.

Como estudo de caso, este *software* foi remodelado para aproveitar das vantagens oferecidas pela arquitetura abordada. Em vez de instalações locais optou-se pela implementação de um cliente que pudesse ser executado via *Web*, responsável por alguns processamentos simples, e de um servidor para o laboratório, o qual atenderia a diversos clientes, fazendo processamentos mais complicados e mantendo um banco de dados central compartilhado. Além disto, o servidor deveria possibilitar a utilização de métricas de comparação do próprio pesquisador. Cada laboratório seria, ainda, acrescido das funcionalidades de um sistema de agentes.

Em resumo, as funcionalidades adicionadas à nova versão do sistema AnaGel foram:

- Implementação na arquitetura cliente/servidor;
- Execução via *Web*;
- Utilização de algoritmos do usuário para comparação entre canaletas;
- Interoperabilidade entre servidores AnaGel para compartilhamento de dados e serviços.

A aplicação cliente foi implementada como uma *applet* Java. Cabe a ela o processamento local do gel para extração de todas as informações necessárias, o que demanda rotinas de processamento digital de imagens e visão computacional, além de interação com o usuário. A comparação entre canaletas é sempre feita no servidor, sendo as canaletas selecionadas no cliente e em seguida enviadas para processamento. A comunicação entre estas partes foi feita utilizando um tipo de *RMI* implementado também neste projeto (Seção 3.3.8).

É importante ressaltar que a máquina virtual Java (*JVM*) presente nos navegadores nem sempre é capaz de executar a aplicação por ser de uma versão diferente daquela que está no servidor. A incompatibilidade encontra-se mais especificamente no que diz respeito à serialização de componentes gráficos, cujas classes normalmente são alteradas a cada atualização. Para resolver este problema, a *applet* cliente foi convertida com o utilitário *htmlconverter* de forma a verificar se a versão da *JVM* do navegador é ou não compatível. Caso não seja, é sugerido de imediato ao usuário a atualização desta, redirecionando-o para o *site* onde a versão adequada pode ser encontrada.

Nas seções seguintes são dadas descrições detalhadas da implementação de cada módulo do sistema. Alguns dos módulos abordados foram remodelados a partir da versão anterior [34, 33] e reescritos na linguagem Java, e outros foram adicionados para prover as novas funcionalidades ao AnaGel.

3.3.1 Estrutura do banco de dados

A Figura 3.2 exibe o modelo entidade-relacionamento do banco utilizado para armazenamento das informações do sistema, o qual não apresenta complexidades. Este modelo foi implementado em um banco de dados *Informix* por se tratar de um servidor capaz de suportar um volume de dados muito grande e por existir uma licença de uso irrestrito. O modelo foi mapeado nas tabelas descritas a seguir:

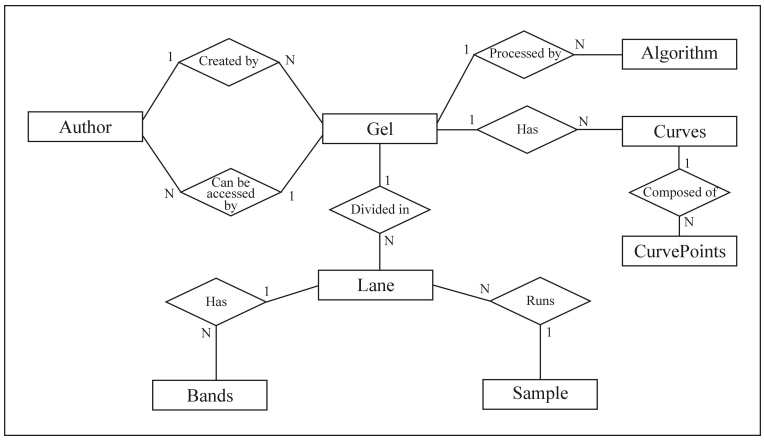

Figura 3.2: Modelo entidade-relacionamento do banco de dados.

- A tabela *Gel* armazena todas as informações peculiares de cada gel, como tipo de substância analisada, dados sobre a corrida e outras observações.
- A tabela *Lane* é responsável por guardar informações sobre a canaleta digitalizada. Os dados incluem o perfil extraído da imagem, os pesos moleculares a cada ponto deste perfil, além de coordenadas que permitem a localização da canaleta na imagem.
- Na tabela *Sample* ficam gravadas as informações sobre cada amostra utilizada nas canaletas.
- Cada registro na tabela *Bands* representa uma banda presente em uma canaleta, com seu peso molecular e seu deslocamento dentro da canaleta.
- As curvas de limites e de calibração são guardadas como registros em *Curves*.
- Cada ponto de uma curva de limite ou calibração é registrado na tabela *CurvePoints*, nas coordenadas da imagem, medidas em pixels.
- Os algoritmos utilizados para estimativa dos pesos moleculares das bandas e normalização dos perfis de um gel ficam gravados na tabela *Algorithms*. A finalidade desta tabela é possibilitar a verificação do processo de obtenção dos perfis por um outro usuário.

- A entidade *Author* é responsável por manter os dados, tais como login, senha, e-mail e outros dos usuários cadastrados no sistema.
- As entradas em *Permissions* indicam quais géis um usuário está autorizado a utilizar, embora não possa modificá-los.

3.3.2 Acesso ao banco de dados

Foi desenvolvida uma camada para acesso ao banco de dados para prover um maior nível de abstração com relação à estrutura mapeada nas tabelas. A possibilidade de execução de *SQL*s não foi impedida, dado que a aplicação cliente pode criar critérios e solicitar ao servidor AnaGel a execução das consultas.

As aplicações clientes não conectam-se diretamente com o banco de dados, de forma que qualquer transação desejada deve ser solicitada ao servidor AnaGel. Este desenvolvimento em três camadas, conforme exibido na Figura 3.3, possibilita uma maior flexibilidade quanto a mudanças na estrutura e até mesmo troca do servidor de banco de dados, uma vez que não demanda alterações na aplicação cliente.

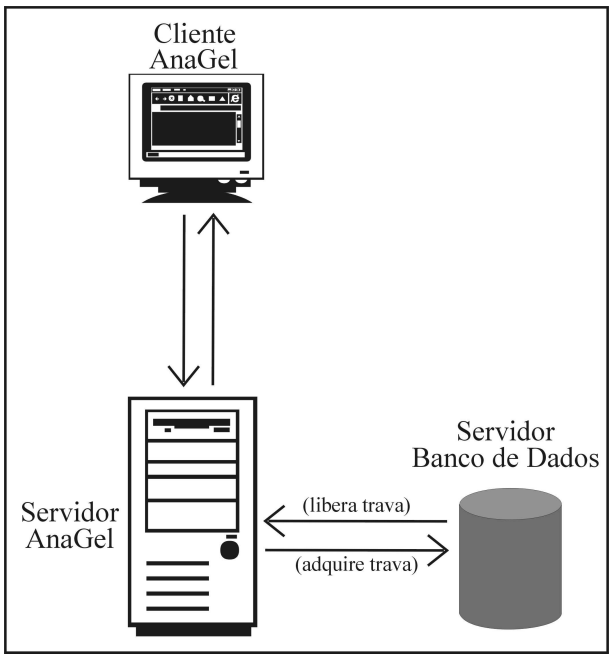

Figura 3.3: Modelo em três camadas.

Um aspecto que também foi considerado diz respeito ao acesso simultâneo ao banco por solicitação de vários clientes. Poderiam haver inconsistências caso mais de um usuário solicitasse operações de atualização no banco, dado que esta situação foi tratada no servidor AnaGel ao invés de delegada ao servidor de banco de dados para prover uma maior independência quanto a este último. Assim, sempre que uma aplicação solicita uma tran-

sação com o banco, uma trava deve ser adquirida para que ela se efetive, caso contrário a requisição espera em uma fila até que seja finalizada a transação em execução e a trava seja liberada.

3.3.3 Cadastro e autenticação de usuários

Ao ser cadastrado o usuário recebe um login e uma senha, que ficam armazenados no banco de dados. Esta foi a forma de implementação inicial, apesar de ser sabidamente vulnerável a ataques. Devido à modularidade proporcionada pelo desenvolvimento orientado a objetos, esta funcionalidade pode ser facilmente alterada, provendo formas mais seguras de autenticação.

Para cada usuário cadastrado é criado um diretório nomeado com o seu login. Este diretório possui quatro subdiretórios, cada um com uma funcionalidade específica:

1. `[instalação]/users/[login]/images` [1]: contém arquivos de imagens de géis submetidas pelo usuário;
2. `[instalação]/users/[login]/compare`: contém arquivos fontes e compilados dos algoritmos de comparação submetidos pelo usuário;
3. `[instalação]/users/[login]/normalize`: contém arquivos fontes e compilados dos algoritmos de normalização de perfis submetidos pelo usuário.
4. `[instalação]/users/[login]/molecularweight`: contém arquivos fontes e compilados dos algoritmos de estimativa de pesos moleculares para uma o gel.

Esta estrutura é útil para a listagem e carga dos respectivos ítens pelo servidor local e, em algumas situações, repasse para a aplicação cliente.

Para autenticação, o usuário fornece seu login e senha a partir de uma *applet* desenvolvida para este fim. Os dados são repassados ao servidor através da invocação do método `authenticateUser` do objeto remoto `DBAuthenticate`, o qual verifica os dados enviados com aqueles constantes na tabela *Author* do banco. Este método devolve um objeto `Author` em caso de sucesso na autenticação ou um objeto nulo (`null`), caso o login e/ou a senha estejam incorretos.

3.3.4 Upload de imagens e algoritmos

Pelo modelo de segurança do ambiente Java, *applets* não são capazes de utilizar recursos locais, tais como ler ou escrever arquivos, utilizar periféricos, fazer acesso a *dlls*, etc. a

[1] Neste contexto `[instalação]` significa o diretório no qual o *software* foi instalado e `[login]` significa o login do usuário

não ser que possuam uma assinatura digital e que ao dono desta assinatura sejam dadas tais regalias. A inclusão do certificado na máquina cliente certamente introduziria um complicador, o que dificultaria a difusão e utilização do sistema.

Como forma de prover meios alternativos de submissão de arquivos locais, tais como imagens ou algoritmos, optou-se pela utilização de formulários HTML. O formulário permite a inclusão de arquivos e sua submissão ao servidor *Web*, onde pode ser processado para recuperação dos dados. Com esta implementação, o servidor AnaGel pode ter acesso aos arquivos submetidos e repassá-los à *applet* cliente.

O formulário de submissão desenvolvido (Figura 3.4) possui um *tipo*, o qual é definido pelo usuário quando da submissão de algum arquivo, podendo ser de imagem de gel, algoritmo de comparação, normalização de perfis ou estimativa de pesos moleculares . Quando enviado, este é tratado por uma *servlet*, encarregada de extrair o arquivo anexado e processá-lo corretamente.

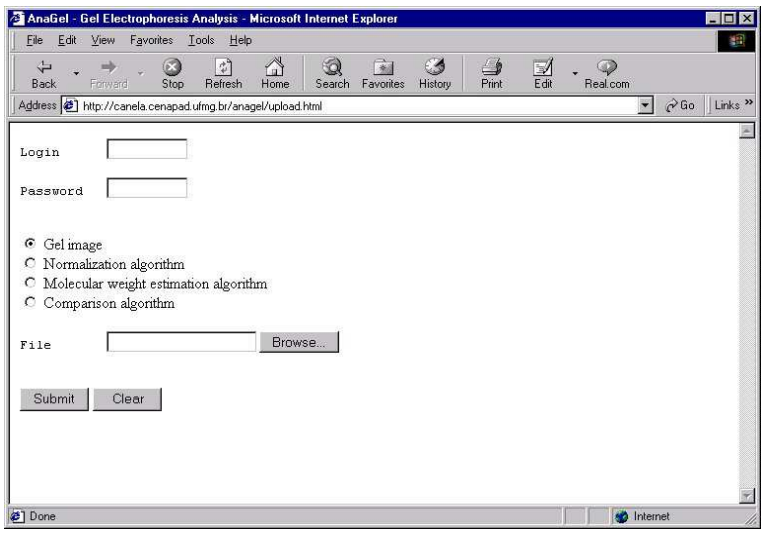

Figura 3.4: Formulário de submissão.

Uma *servlet* é responsável pelo processamento do formulário e extração das informações submetidas, redirecionando os dados para o diretório correto e fazendo as compilações quando necessário.

O formulário submetido também leva informações sobre o usuário em questão (login e senha) de forma a possibilitar a autenticação pela *servlet*. A *servlet* utiliza a mesma rotina `authenticateUser` descrita na Seção 3.3.3 para verificar os dados do submissor. Apenas os formulários submetidos por usuários devidamente cadastrados são aceitos.

Caso o formulário seja do tipo *imagem*, o arquivo anexado é redirecionado para o diretório que contém as imagens do gel para o usuário que fez a submissão, conforme explicado na Seção 3.3.3. O servidor AnaGel pode, assim, listar as imagens disponíveis e repassar esta lista para a *applet*, a qual pode solicitar quaisquer daquelas submetidas pelo

usuário corrente. O nome do arquivo destino é exibido ao usuário após o envio da imagem no navegador *Web*.

Algoritmos para comparação de canaletas podem ser submetidos em formulários do tipo *algoritmo de comparação*. O arquivo fonte submetido deve conter uma classe que herde da classe abstrata `CompareSamples`, a qual é compilada no momento da submissão, além de ser testada sua validade. Detalhes de implementação de algoritmos de comparação são dados na Seção 3.3.10.

O formulário pode ser do tipo *algoritmo de normalização*, indicando que o arquivo que segue em anexo é um fonte escrito na linguagem Java, o qual contém uma classe que descende da classe abstrata `Normalize`, podendo ser compilado e disponibilizado para utilização. O arquivo fonte é compilado através da *servlet* e o resultado da compilação é retornado no navegador do usuário, indicando sucesso ou exibindo as mensagens de erro, como na Figura 3.5. Após a compilação é testado dinamicamente se o algoritmo enviado é realmente um para normalização de perfis, caso contrário ele é rejeitado. Na Seção 3.3.7 é descrito em detalhes como algoritmos de normalização podem ser desenvolvidos.

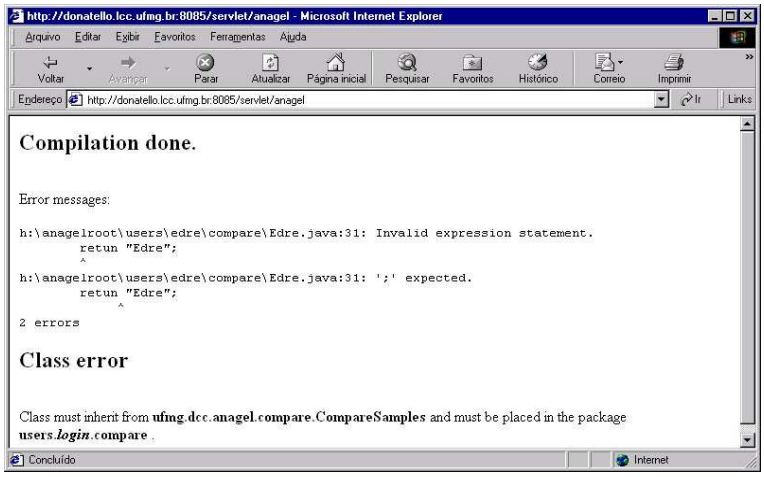

Figura 3.5: Resultado da compilação.

Assim como para algoritmos de normalização, aqueles para estimativa de pesos moleculares também podem ser submetidos em formulário do tipo *algoritmo de estimativa de pesos moleculares*. O arquivo contendo a classe que herda da classe abstrata `MolecularWeightEstimation` é compilado e testado para certificação de que realmente descende de tal classe. A Seção 3.3.6 dá detalhes de implementação e uso de tais algoritmos.

É importante ressaltar que, apesar do sistema AnaGel ter sido projetado considerando-se a flexibilização quanto ao uso de algoritmos de normalização e estimativa de pesos moleculares, estas funcionalidades ainda não estão disponíveis. Desta etapa foi desenvolvido apenas o módulo de submissão e compilação.

3.3.5 Extração das informações da imagem

A aplicação AnaGel não disponibiliza ferramentas para tratamento gráfico da imagem a ser processada, como filtros, redefinição de tamanho, rotação, etc. Este módulo foi desenvolvido para extrair as informações necessárias do gel. Assim, qualquer imagem a ser submetida deve ser devidamente tratada antes do envio. É importante salientar que, para o processamento correto, (i) a imagem deve estar em tons de cinza, nos formatos *gif* ou *jpg*; (ii) as canaletas devem estar dispostas verticalmente; (iii) os maiores pesos moleculares devem estar localizados na parte superior da imagem; (iv) as bandas devem ser escuras e o fundo do gel claro. A Figura 3.6 exibe um gel que pode ser tratado pelo sistema.

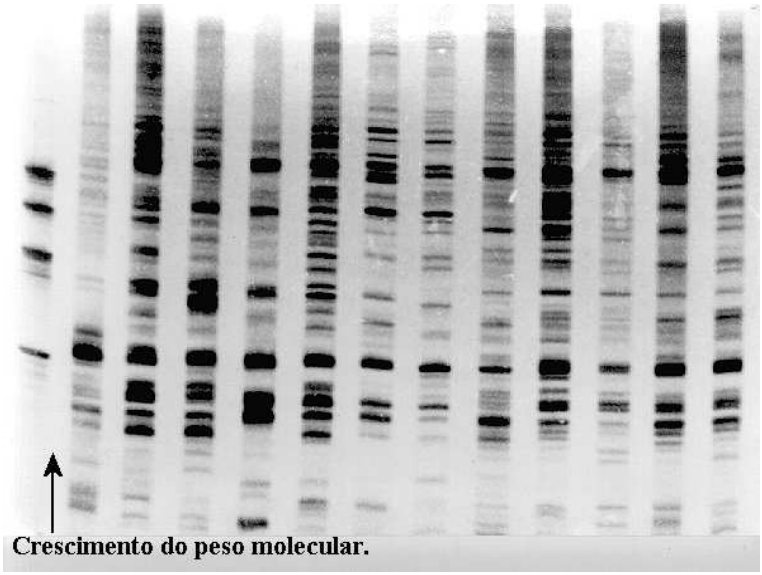

Figura 3.6: Imagem de gel válido.

Todo processo de extração de informações é feito localmente na *applet* cliente, uma vez que não demanda grande poder de computação e evita que muitos dados sejam enviados para o servidor AnaGel e então retornados ao cliente.

Um objeto da classe `LaneFinder` é responsável pela marcação das *canaletas* (Apêndice A), disponibilizando meios de detecção automática e inserção manual. O algoritmo para detecção automática foi desenvolvido baseado em uma heurística bastante simples, sendo analisado o gráfico das funções $f_1(x) = \sum_{k=1}^{\frac{a}{2}} I_{(x,k)}$ e $f_2(x) = a * 256 - \sum_{k=\frac{a}{2}+1}^{a} I_{(x,k)}, 1 \leq x \leq l$, onde a representa a altura da imagem, l representa a largura e I a intensidade do pixel ($0 \leq I \leq 255$). É feita uma correspondência entre os picos de $f_1(x)$ e depressões de $f_2(x)$, definindo assim uma "guia" para a marcação da canaleta. A Figura 3.7 ilustra esta idéia. Em algumas situações onde os picos não coincidem (por exemplo quando a metade inferior de uma das canaletas apresenta pouca tonalidade de cinza e a parte superior possui tons bastante fortes), a detecção pode falhar. Espera-se reimplementar esta rotina com o intuito de melhorar a detecção das canaletas seguindo o algoritmo proposto em [28]. A

Figura 3.8 exibe um gel com as canaletas demarcadas.

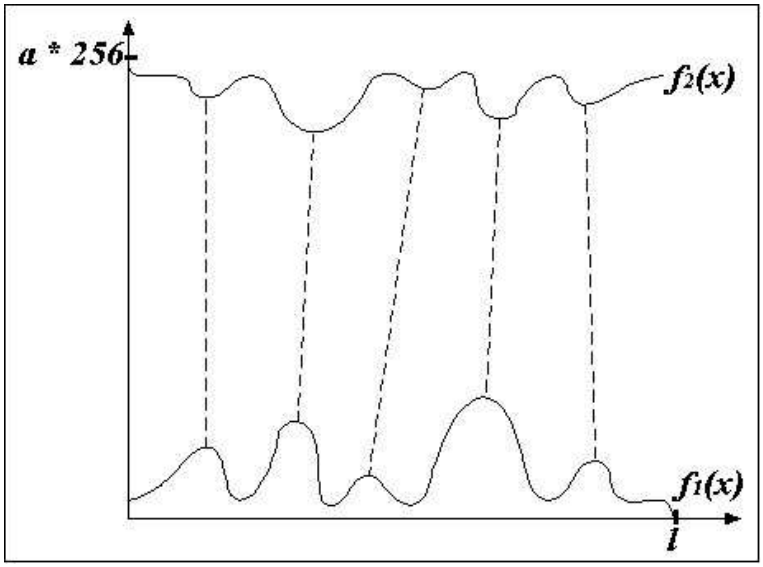

Figura 3.7: Esboço de detecção de canaletas.

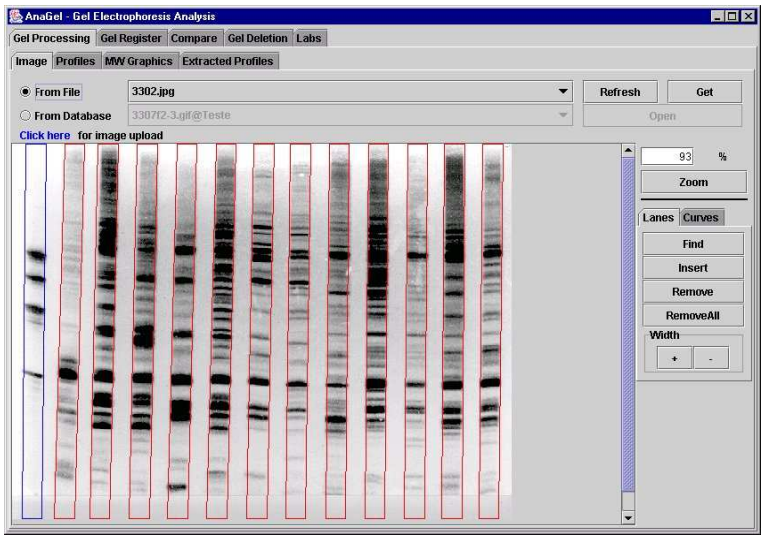

Figura 3.8: Gel com canaletas demarcadas.

Para correção de distorções na imagem, como por exemplo o efeito sorriso (Apêndice A), é possível a demarcação de curvas que delimitem a área útil da imagem e que acompanhem suas deformações. Um objeto da classe `MarkMWCurves` é responsável por tal tarefa, bem como por calcular as linhas de calibração de pesos moleculares. A Figura 3.9 exibe um gel com curvas limítrofes e de calibração de pesos moleculares marcadas.

A informação extraída da imagem que possibilita o tratamento pelo *software* é denominado *perfil*. Para cada canaleta demarcada é retirado seu perfil, o qual representa os tons de cinza que definem seu padrão. A extração é responsabilidade de uma instância da classe

Figura 3.9: Gel com curvas demarcadas.

`ProfileExtractor`, que considera as canaletas demarcadas pelo `LaneFinder` e as curvas que delimitam o gel. Os perfis extraídos para um gel podem ser observados na Figura 3.10.

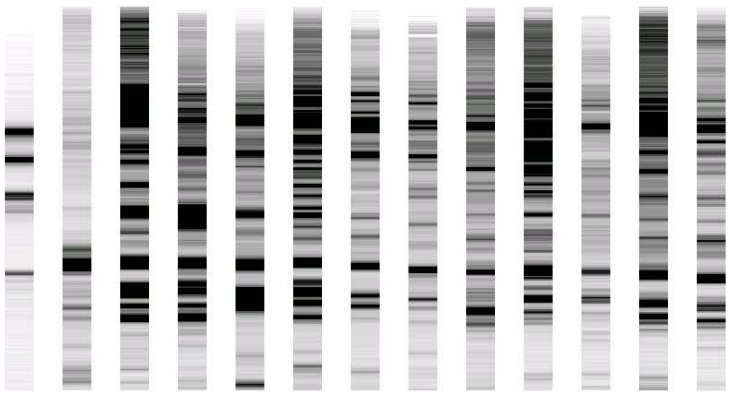

Figura 3.10: Perfis das canaletas de um gel.

Pelo processamento do perfil é possível a determinação das *bandas* (Apêndice A) presentes. Esta determinação pode ser feita automaticamente, uma vez definidos o tom de cinza mínimo e o número de elementos contíguos que possuam um tom de cinza igual ou superior ao especificado a ser considerado como banda, e também manualmente, pela interação com o usuário.

3.3.6 Estimativa dos pesos moleculares

Uma grande utilidade de sistema para processamento de géis de eletroforese é a habilidade de cálculo de pesos moleculares de bandas automaticamente. Em processos manuais

de cálculos de pesos moleculares de bandas, os cientistas baseiam-se em suas distribuições para uma substância de padrão conhecido que é corrida juntamente com as substâncias de interesse, para então calcularem as demais, tendo como base o deslocamento daquelas moléculas durante o processo. Uma forma muito utilizada baseia-se no fato de que as macromoléculas deslocam-se proporcionalmente ao inverso do logaritmo de seu peso, embora não seja o único método existente (veja Apêndice A para detalhes).

Uma vez demarcadas as curvas de calibração, ao longo das quais o peso molecular é o mesmo para as bandas interceptadas em cada canaleta, o sistema é capaz de estimar o peso de todas as bandas existentes. A Figura 3.11 mostra o peso molecular estimado para as bandas presentes na canaleta.

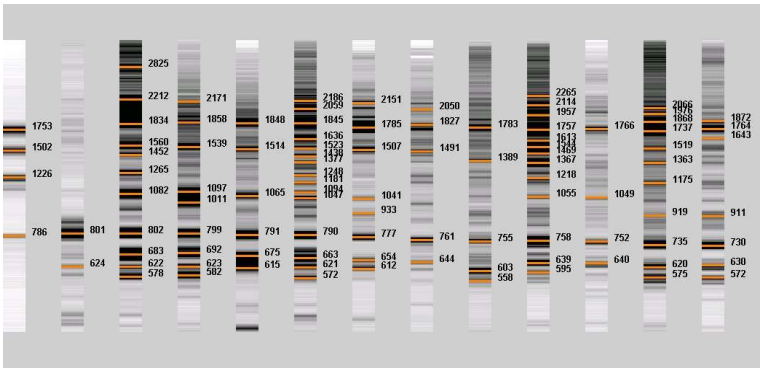

Figura 3.11: Peso molecular de bandas e canaleta normalizada.

Esta implementação do sistema AnaGel disponibiliza dois métodos para estimativa: um que considera o deslocamento das moléculas proporcional ao logaritmo do peso molecular e outro que avalia segundo o inverso da distância percorrida.

A implementação foi feita de forma bem modularizada, permitindo que novas métricas sejam adicionadas facilmente pelo implementador do sistema. Com este objetivo foi definida a classe abstrata `MolecularWeightEstimation`, exibida na Figura 3.12.

O método `estimateMolWeight` recebe como parâmetros um objeto do tipo `BandsCollection`, que contém as bandas do perfil, e um do tipo `Profile`, o qual representa o perfil em si. Este método retorna um objeto `BandsCollection` contendo as bandas com seus pesos moleculares associados, além de associar o peso molecular a cada ponto do perfil passado.

Devido ao *dynamic binding* de métodos provido pela linguagem Java, o módulo responsável pelo cálculo do peso molecular define uma referência do tipo `MolecularWeightEstimation` e atribui o objeto da subclasse equivalente à escolha do usuário (`LogEstimation` ou `InverseDistanceEstimation`, no caso deste implementação). Assim, ao ser invocado o método `estimateMolWeight` através da referência, é executada a implementação definida na classe do objeto referenciado. Espera-se possibilitar a definição de algoritmos para estimativa de pesos moleculares pelo próprio usuário em versões futuras.

```
package ufmg.dcc.anagel.molecularweight;
...
Public abstract class MolecularWeightEstimation {
    protected BandsCollection references;
    ...
    public MolecularWeightEstimation (BandsCollection r)
        throws MolecularWeightEstimationException {
        ...
        references = r;
        ...
    }
    public abstract float getDistance (float mw);
    public abstract float getMolWeight (float distance);
    public abstract BandsCollection estimateMolWeight (BandsCollection b,
                                                       Profile p);
}
```

Figura 3.12: Classe `MolecularWeightEstimation`.

3.3.7 Normalização de perfis

Em algumas situações as moléculas de mesmo peso molecular migram diferentemente em cada canaleta, causando uma deformação nos padrões denominado *efeito sorriso*. Desejando-se fazer uma compensação deste problema os perfis são normalizados, possibilitando tratar uniformemente cada perfil do gel em questão. Na Figura 3.11 também pode ser observado um perfil normalizado.

Assim como na implementação de rotinas para estimativa de pesos moleculares, o módulo para normalização de perfis foi desenvolvido de forma a possibilitar o desenvolvimento de novas técnicas e sua acoplação ao sistema sem grandes dificuldades. A classe abstrata `Normalize` fornece algumas funcionalidades básicas e delega ao desenvolvedor a implementação do método `normalizedLane`, o qual é utilizado para, dado um perfil original, fazer sua normalização. Na Figura 3.13 é dado um esboço de sua implementação.

A normalização pode ser feita apenas para um determinado intervalo compreendido entre dois pesos moleculares, possibilitando também o corte de regiões do perfil para comparações.

Foi dada uma implementação desta classe nesta versão do AnaGel, a classe `Linear-Normalize`. Nela, o método `normalizedLane` foi implementado de tal forma que o perfil é expandido linearmente, independentemente da deformação ser mais acentuada na parte superior ou inferior da imagem, fazendo com que todos os perfis fiquem com o mesmo tamanho, baseado no maior perfil extraído. Também aqui espera-se prover uma maior flexibilidade ao pesquisador, que poderá usar seus próprios critérios para normalização.

3.3.8 Protocolo de comunicação

O sistema AnaGel foi desenvolvido para funcionar em uma arquitetura cliente/servidor. Este último foi utilizado para prover uma camada intermediária entre o cliente AnaGel e

```
package ufmg.dcc.anagel.lane.normalize;
...
public abstract class Normalize {
    protected float maxDistance;
    protected float minDistance;
    protected int newLaneLength;

    public Normalize(float dMin, float dMax, int laneLen) {
        maxDistance = dMax;
        minDistance = dMin;
        newLaneLength = laneLen;
    }

    public abstract Lane normalizedLane (Lane l)
        throws NormalizationException;

    public float getMinDistance () {
        ...
    }

    public float getMaxDistance () {
        ...
    }
}
```

Figura 3.13: Classe `Normalize`.

o banco de dados e também para executar computações possivelmente mais complexas, como comparação de canaletas.

O servidor foi pensado como um hospedeiro para objetos remotos que poderiam ter seus métodos invocados diretamente pelo cliente com um certo grau de transparência.

Foi desenvolvido um protocolo para prover que o cliente pudesse acessar os serviços invocando métodos dos objetos residentes no servidor, um tipo de *RMI*. Como estratégia de implementação elaborou-se a classe `AnaGelClient` para encapsular a execução de métodos.

Ao ser invocada uma operação no cliente, este faz a chamada do método `execute` através de uma instância da classe `AnaGelClient` informando a classe à qual o método pertence, o nome do método e os seus parâmetros. Os dados fornecidos são encapsulados em uma instância da classe `Message`, que é serializada e repassados através de comunicação via *socket* ao servidor. Um objeto da classe `DecodeMessage` encarrega-se de recuperar os dados enviados, localizar em uma tabela *hash* de objetos aquele que representa a instância da classe informada, e invocar o método especificado no objeto encontrado com os devidos parâmetros. O resultado é, então, encapsulado em um objeto `Message` e devolvido ao cliente. A Figura 3.14 demonstra esta idéia.

O cliente foi desenvolvido para prover comunicação síncrona, de forma a bloquear o sistema até a finalização da tarefa especificada.

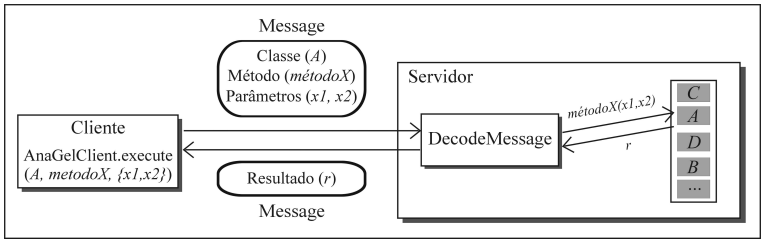

Figura 3.14: Execução de rotinas remotas.

3.3.9 Auditabilidade

Uma característica importante desta versão do AnaGel é a possibilidade de adição de módulos que permitam a um cientista auditar o processamento dos dados de um gel, através da verificação dos algoritmos de processamento utilizados, como aqueles para estimativa de pesos moleculares e normalização de perfis. Estes algoritmos são armazenados no banco juntamente com o gel processado.

3.3.10 Comparação de canaletas

Existem diversas alternativas para se quantificar a diferença entre duas canaletas. A maioria das métricas relaciona o total de bandas em cada canaleta com aquelas presentes em uma e ausentes em outra. Uma banda presente em duas canaletas[2] é dita ser *compartilhada*. Diversas métricas são descritas na literatura. Outros algoritmos computam a diferença levando-se em consideração os padrões de cinza de cada perfil diretamente (vide Apêndices A e B para maiores detalhes).

O sistema AnaGel foi implementado de forma a permitir comparação entre canaletas utilizando alguns algoritmos pré-definidos e também aqueles que o usuário queira porventura executar. Utilizando-se desta facilidade, o pesquisador pode desenvolver seu algoritmo e observar se os resultados são satisfatórios ou não, sem a necessidade da interferência do desenvolvedor do sistema para tal.

Para utilização de algoritmos próprios é necessário que o usuário escreva uma subclasse da classe `CompareSamples`, a qual é exibida na Figura 3.15, fornecendo a implementação para o método `compare` definido.

O algoritmo deve ser implementado na linguagem Java, uma vez que o sistema foi totalmente desenvolvido utilizando-se de recursos da mesma. O usuário deve colocá-la no pacote `users.[ login].compare`, sendo `[login]` o nome do usuário cadastrado que irá acessar o sistema.

Assim, para o usuário *antonio*, um exemplo de implementação é dado na Figura 3.16.

[2]Esta situação ocorre quando duas canaletas possuem bandas com o mesmo valor de peso molecular.

```
package ufmg.dcc.anagel.compare;
...
public abstract class CompareSamples {
    ...

    /**
     * Method to be implemented by the user to calculate the
     * distance between lane1 and lane2.
     */
    public abstract float compare(Lane lane1, Lane lane2);
    ...
}
```

Figura 3.15: Classe `CompareSamples`.

```
package users.antonio.compare;
...
public class MyCompare extends CompareSamples {
    public float compare(Lane lane1, Lane lane2) {
       // Corpo do metodo
       ...
    }
    public String toString() {
       return "MyCompare";
    }
}
```

Figura 3.16: Implementação da classe de comparação.

É interessante, embora não necessário, que seja implementado o método `toString`, para que este retorne o nome da classe ou algum *string* que possibilite sua identificação.

O arquivo fonte deve ser enviado ao servidor pela página de submissão. No momento do recebimento, a classe enviada é compilada e sua validação é feita. A validação consiste em testar se o pacote especificado está correto e se a classe herda as funcionalidades da superclasse `CompareSamples`. O usuário pode observar em seguida o reultado da submissão - se foi aceita corretamente ou se houve algum erro durante o processo.

Para a comparação entre as canaletas é necessário, primeiramente, que o usuário faça uma pré-seleção das canaletas de interesse, o que é efetivado através da execução de uma ou várias consultas no banco de dados, através de solicitações feitas ao servidor AnaGel. Uma vez pré-selecionadas, o intervalo de pesos moleculares das canaletas deve ser definido e todos estes dados são enviados ao servidor, acrescidos do *nome* do algoritmo de comparação a ser utilizado. A listagem de todos os algoritmos de comparação pode ser observada na interface cliente, porém as classes que definem os algoritmos não ficam carregadas, uma vez que a carga é feita dinamicamente no momento da execução.

No servidor AnaGel é feita a normalização das canaletas, dado que podem ser oriundas de diferentes imagens e seus tamanhos muito provavelmente não serão compatíveis. A normalização serve também para cortar a região da canaleta de interesse, definida pelo usuário quando da submissão.

Feito este processamento, a próxima tarefa é a comparação das canaletas para constru-

ção do relatório. Um aspecto de implementação interessante quanto à geração dos relatórios é que estes são criados como objetos, utilizando os componentes visuais da *API* do Java, e não diretamente como figuras. Isto torna possível a adição de funcionalidades ao relatório, de forma que este pode conter, por exemplo, botões que podem ser acionados para exibir informações mais detalhadas sobre cada canaleta.

O sistema é capaz de gerar três tipos de relatórios:

1. Um que exibe apenas duas canaletas, mostrando graficamente as bandas compartilhadas, no caso de métricas que considerem a presença e ausência de bandas em uma e outra (Figura 3.17), ou o resultado do alinhamento dos perfis (Figura 3.18) (Apêndice B). Este relatório é gerado apenas quando é feita a comparação entre duas canaletas.

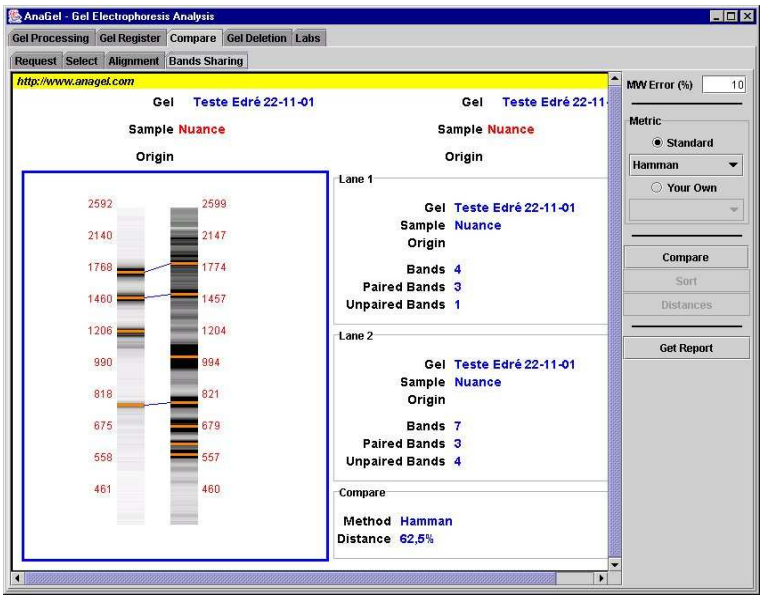

Figura 3.17: Relatório de compartilhamento de bandas.

2. Um relatório que mostra o perfil das canaletas. Este relatório pode ser obtido no caso de haver mais de duas canaletas selecionadas e alguma delas ter sido marcada como mestre (Figura 3.19). Assim, é computada a distância entre cada canaleta e a mestre e os perfis são desenhados daquele de menor distância para aquele de maior em relação à mestre.

3. Um terceiro tipo gera a entrada para o programa *Phylip* [13] (Figura 3.20), que é um programa para inferência de relações filogenéticas. Este só é o caso onde existe mais de duas canaletas selecionadas e não há canaleta mestre.

O algoritmo de comparação só é carregado no momento do cálculo da distância. Isto é possível devido à funcionalidade de carga dinâmica de classes provida pela máquina virtual

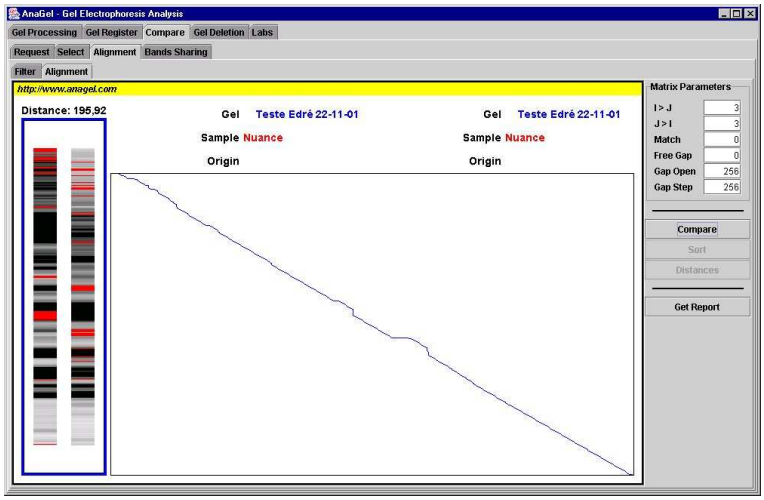

Figura 3.18: Relatório para alinhamento de perfis.

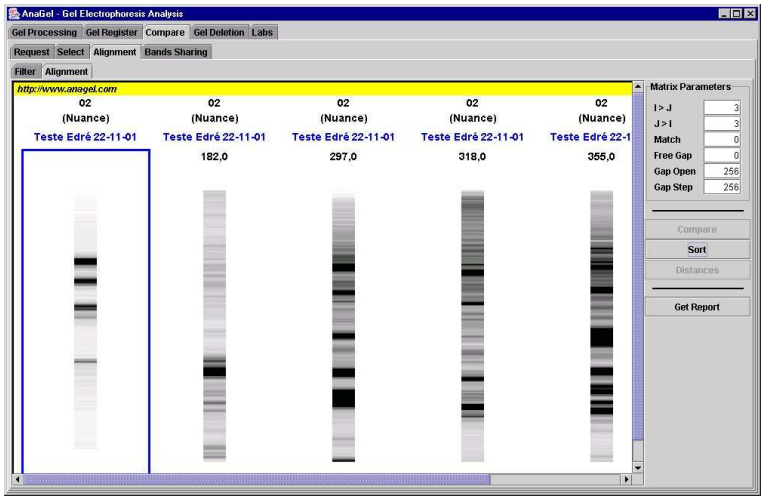

Figura 3.19: Relatório de comparação com canaleta mestre.

Java. Desta forma o *bytecode* da classe compilada representada pelo nome enviado pelo usuário é carregada e o objeto para comparação é instanciado dinamicamente.

Construído o relatório, este é retornado à aplicação cliente, sendo então exibido. Outro ponto interessante é que o usuário pode optar pela obtenção deste por e-mail, onde os componentes visuais são convertidos localmente em uma imagem *gif* a qual é então enviada através do servidor.

3.3.11 Colaboração entre laboratórios

Para a criação do ambiente de colaboração optou-se pela utilização da tecnologia de agentes móveis, onde o cientista pode fornecer seu algoritmo e um agente móvel se encarrega da execução nos laboratórios remotos.

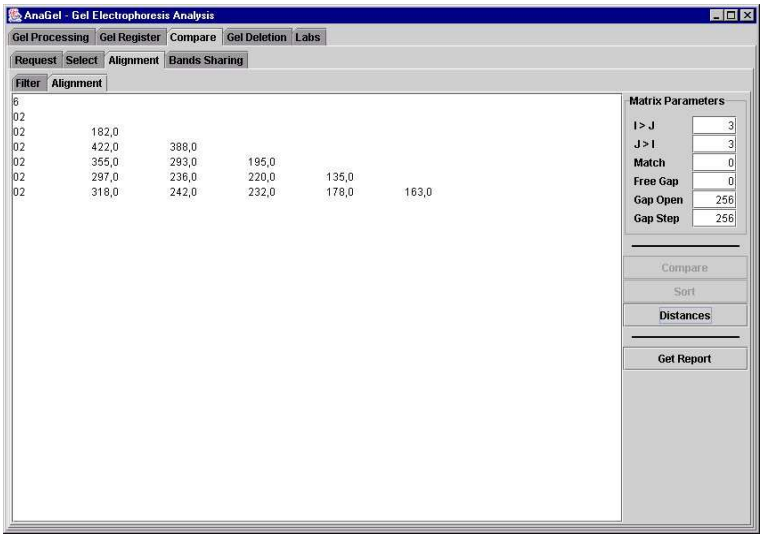

Figura 3.20: Relatório de comparação entre várias canaletas.

Os servidores localizados em cada laboratório possuem também acesso a um sistema de agentes (*agência*), que são usados para disparar e receber unidades de *software* a serem executadas.

A Figura 3.21 mostra a implementação do AnaGel utilizando o modelo proposto, destacando a interação entre os agentes e os servidores.

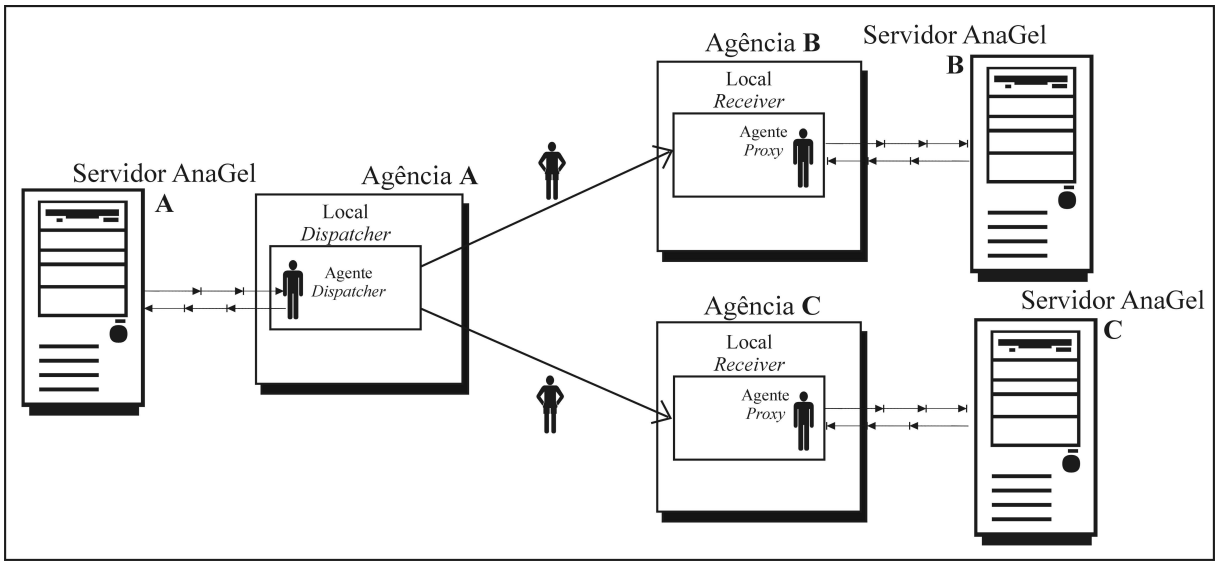

Figura 3.21: Interação entre servidor AnaGel e agências.

Todas as agências são registradas dinamicamente na *região AnaGel*, de forma a permitir a construção da lista de agências disponíveis ao pesquisador em tempo de execução, a qual contém apenas endereços de laboratórios que estão com seus servidores ativados. Ao ser ligado um servidor em qualquer laboratório, ele é automaticamente registrado nesta

região. O servidor que recebe os registros das agências é único, podendo estar localizado em qualquer parte da rede.

Um único servidor é responsável pelo registro de todas as regiões, o qual tem ativado um *region registry* fornecido pelo sistema de agentes *Grasshopper*. Quando as agências são criadas, um dos parâmetros é a região à qual ela vai se registrar, sendo indicada a mesma *região AnaGel* para todas.

Cada agência possui dois locais: *Dispatcher* e *Receiver*. O local *Dispatcher* provê funcionalidades para o envio e recolhimento dos agentes móveis. Já o *Receiver* disponibiliza meios de recebimento dos agentes visitantes que chegam para fazer suas computações.

No local *Dispatcher* reside um agente estacionário, o agente *Dispatcher*, responsável pela comunicação do sistema de agentes com o servidor local. Este é utilizado pelo servidor que origina a requisição para criar e enviar os agentes de buscas aos laboratórios remotos e agrupar os resultados das pesquisas após o retorno de cada um.

Um aspecto importante no que diz respeito ao sistema de agentes *Grasshopper* é que um agente não consegue ter acesso a outro na mesma agência através de sua referência (isto é, tratando-o simplesmente como um objeto naquele contexto). Entretanto, o agente tem acesso às funcionalidades do local onde está situado. O mecanismo encontrado para comunicação entre agentes de uma mesma agência foi através das funcionalidades do local - todo agente, no momento em que chega ao local desejado, providencia seu registro. Qualquer agente, quando necessitar comunicar-se com algum outro daquele local, acessa o serviço do local onde se encontra e adquire a referência desejada. Esta estratégia foi muito útil para o agrupamento do resultado dos processamentos remotos de cada agente, de forma a construir um único conjunto de dados a partir da fusão daqueles parciais. O local *Dispatcher* foi implementado baseado nesta idéia.

Seguindo os passos descritos em [21] para adição de funcionalidades a locais, criou-se a interface `DispatcherPlaceService`, conforme exibido na Figura 3.22.

```
package ufmg.dcc.anagel.agents;
...
public interface DispatcherPlaceService {

    public int startNewReq();

    public void addIdentifier(String id, int reqNum);

    public void addRegisteredAgent(Agent agt);

    public Agent next(int reqNum);

    public int percentDone(int reqNum);
}
```

Figura 3.22: Interface `DispatcherPlaceService`.

O método `startNewReq` cria uma nova entrada na lista de requisições daquele local.

A lista serve basicamente para armazenar informações que permitam a identificação do agente quando do seu retorno. Uma lista é criada para cada solicitação de utilização de recursos remotos. O retorno deste método é o número que identifica a requisição.

A funcionalidade do método `addIdentifier` é de adicionar o identificador de um agente a uma dada requisição.

Quando do retorno de um novo agente ao local, o método `addRegisteredAgent` é executado, permitindo a localização da requisição à qual pertence aquele agente fazendo-se a verificação pelo seu identificador.

Para um acompanhamento do progresso da execução de uma requisição, o método `percentDone` pode ser executado. O percentual concluído é calculado levando-se em consideração o número de agentes disparados e quantos deles retornaram até aquele momento.

Os agentes para cada requisição podem ser recuperados invocando-se o método `next`. Isto possibilita o tratamento dos agentes que retornaram como um objeto, o que facilita a recuperação das informações carregadas por eles para construção do resultado final. Este método possui o efeito colateral de retirar o agente da lista para aquela requisição e de finalizá-la quando não existirem mais agentes.

A funcionalidade descrita acima é realizada pela classe `AgentDispatcherPlaceService`. Com esta implementação o local pode atender simultaneamente até 100 requisições. É responsabilidade do usuário do serviço aguardar a disponibilidade do local para atender à requisição, caso o limite tenha sido atingido. Este fato é devidamente tratado pelo servidor AnaGel, ficando transparente para o usuário final.

Para a implementação do agente responsável por disparar as requisições foi necessária a definição da interface `AgentDispatcher`, a qual foi implementada pela classe `Comparator-AgentDispatcher`. Houve a necessidade da definição da interface devido às restrições impostas pelo sistema de agentes *Grasshopper* para criação de *proxies*, uma vez que este agente é utilizado para comunicação com o servidor AnaGel. Este mecanismo possibilita o intercâmbio de informações entre os dois *softwares* através de *sockets*. A interface é apresentada na Figura 3.23.

O método `addLocation` é invocado pelo servidor AnaGel para adicionar um novo endereço à lista dos laboratórios para onde os agentes serão enviados. A funcionalidade do método `clearLocations` é de excluir o endereço de todos os laboratórios anteriormente fornecidos.

Para associação dos parâmetros de comparação, da consulta *SQL* a ser executada nos servidores remotos e do usuário corrente pelo servidor AnaGel são utilizados os métodos `setCompareData`, `setQuery` e `setCurrentUser`, respectivamente.

Somente após a invocação dos métodos acima os agentes poderão ser disparados, através da execução da rotina `dispatchAgents` feita pelo servidor AnaGel. Este método permite a definição de um tempo máximo de espera pelos agentes. O valor retornado é o identificador

```
package ufmg.dcc.anagel.agents;
...
public interface AgentDispatcher {

    public void addLocation(String location);

    public void clearLocations();

    public void setCompareData(CompareData data);

    public void setQuery(QueryParameters query);

    public int dispatchAgents(long timeOut);

    public Object getResult(int i);

    public void setCurrentUser(Author user);
}
```

Figura 3.23: Interface `AgentDispatcher`.

para a requisição disparada. O fluxo sequencial do programa só é retomado após finalizada a execução desta rotina - sua execução é blocante, só sendo terminada quando da chegada de todos os agentes disparados ou uma porcentagem definida destes, ou expiração do tempo máximo de espera.

Pelo método `getResult` é possível a recuperação do conjunto de dados retornados, constituído pelos resultados individuais de cada agente disparado para atender àquela requisição. É importante ressaltar que todas as operações são executadas no objeto que está alocado dentro do sistema de agentes, e não no servidor AnaGel. este último consegue repassar as chamadas ao objeto real através de um *proxy*.

O local *Receiver* é utilizado pelos agentes visitantes que chegam a um servidor para se utilizarem dos recursos locais. Nele reside um agente estacionário, o *ProxyAgent*. Este agente é utilizado simplesmente como um objeto, o qual possui métodos que permitem a execução de rotinas no servidor.

Como no caso do local *Dispatcher*, foi necessária a definição da interface `ProxyHolderService` para provimento da funcionalidade de execução de rotinas no servidor AnaGel visitado, a qual é apresentada na Figura 3.24.

```
package ufmg.dcc.anagel.agents;
...
public interface ProxyHolderService {

    public void setProxy(Agent agt);

    public Agent getProxy();
}
```

Figura 3.24: Interface `ProxyHolderService`.

A implementação desta interface pela classe `ReceiverProxyHolderService` foi dada de tal forma que um agente estacionário fica registrado, servindo como *proxy* para execução

das rotinas no servidor AnaGel. O método `setProxy` associa o agente supra-citado e o método `getProxy` retorna a sua referência.

Ao fazer uma requisição de busca, o servidor repassa uma cláusula *SQL*, uma canaleta de referência, o algoritmo a ser utilizado e uma lista com os endereços dos laboratórios remotos, através do agente *Dispatcher*. A lista dos locais de destino é definida pelo usuário, através do *proxy* do aplicativo cliente. O agente *Dispatcher* cria vários agentes móveis, denominados *TransportAgent*, contendo o *SQL* e o algoritmo do usuário, e envia um para cada endereço da lista passada.

No sistema de agentes *Grasshopper* o agente móvel é recriado na agência de destino, de forma que o ponto onde a execução parou anteriormente à transferência não é conhecido. É responsabilidade do implementador monitorar o estado do agente para possibilitar a continuidade do processamento, caso contrário a execução é reiniciada. Assim, a classe `TransportAgent` foi implementado com mostrado na Figura 3.25.

```
package ufmg.dcc.anagel.agents;
...
public class TransportAgent extends MobileAgent {

    Private int state = 0;
    private Object result;
    ...

    public void live(){
        switch (state) {
            case 0: {
                // Moves to the destination agency
                ...
            }

            case 1: {
                // Do the processing locally
                ...
            }
            case 2: {
                // Indicates to the source agency that the
                // computation of this agent finished
                ...
            }
        }
    }

    public void setResult(Object result) {
        ...
    }

    public Object getResult() {
        ...
    }
}
```

Figura 3.25: Classe `TransportAgent`.

O ponto onde a computação deve ser retomada é monitorada pelo atributo `state`, dado que o método `live` é executado do princípio sempre que o agente visita uma agência.

Quando da chegada no servidor remoto, o *TransportAgent* espera em uma fila, podendo executar sua tarefa após os eventuais *TransportAgent*s de outros laboratórios que chegaram antes terem finalizado as suas.

Para invocar as rotinas do servidor AnaGel o *TransportAgent* utiliza o serviço do local *Receiver*, recuperando a referência para o objeto que provê acesso ao servidor. Este objeto é uma instância da classe `ProxyAgent`, a qual disponibiliza o método `execute` para executar a tarefa desejada. Em sua implementação este método acessa o servidor utilizando o esquema descrito na Seção 3.3.8

Em sua execução, o *TransportAgent* faz uma pré-seleção dos registros eletroforéticos que deseja trabalhar através da execução do *SQL* enviado. A pré-seleção provê meios de filtragem quanto ao tipo de gel e material em questão, dados gerais referentes ao ambiente onde a corrida se deu (temperatura, corrente elétrica, diferença de potencial, etc.), e outras características que podem ser expressadas com esta linguagem. O algoritmo de filtragem é então executado sobre estes dados, possibilitando a separação daqueles de interesse baseado na distância entre a canaleta de referência e cada uma das pré-selecionadas. Apenas as que possuem distâncias aceitáveis são aprovadas e adicionadas como resultado para aquele agente.

É importante observar que a linguagem *SQL* provê primitivas para a operação sobre os dados armazenados no banco levando em consideração a relação entre eles. Algoritmos mais complexos de filtragem, por exemplo para procurar por alguma característica específica em um dado, são difíceis ou até mesmo impossíveis de serem implementados (a não ser que seja adotada uma solução específica para o banco de dados utilizado). O modelo, como implementado no AnaGel, aumenta o poder de filtragem de canaletas, uma vez que algoritmos complexos podem ser utilizados para trabalhar sobre a *estrutura* dos dados.

Uma vez cumprida sua tarefa, o *TransportAgent* retorna ao sistema de agentes de origem levando o resultado do processamento remoto no atributo `result`. Ao chegar ele providencia seu registro no local *Dispatcher*, de forma a possibilitar a composição do resultado final pelo agente *Dispatcher*.

Por fim, o agente *Dispatcher* reúne os resultados parciais de cada agente e compõe um único conjunto de dados, repassando então ao servidor AnaGel que por sua vez transfere para a aplicação cliente.

O agente *Dispatcher* não espera indefinidamente pelo retorno de todos os agentes. Através de um arquivo de configuração o administrador do servidor AnaGel é capaz de modificar o tempo máximo que será dado aos *TransportAgent*s para executarem suas tarefas. Aqueles que chegam após expirado o prazo são descartados.

A partir da coleta feita pelos agentes o pesquisador pode avaliar sua métrica pela verificação das canaletas resultantes. Além disto, ele ainda pode efetuar o cálculo das distâncias entre as canaletas obtidas utilizando outros métodos, sendo que esta comparação

é agora feita no servidor AnaGel de seu laboratório e não mais remotamente.

Capítulo 4

Conclusões

Trabalhos cooperativos são de grande importância no âmbito da ciência. Muitas vezes os cientistas trocam informações e conhecimentos para darem suas contribuições à evolução da humanidade. Com o advento da Internet, a utilização de *colaboratórios* - ou *"laboratórios sem paredes"* - veio contribuir para que pesquisas conjuntas pudessem ser realizadas sem o desconforto do deslocamento do cientista de uma instituição para outra.

O trabalho desenvolvido veio estudar um modelo muito útil para construção de repositórios globais virtuais, os quais podem disponibilizar os mais diversos recursos para utilização remota. O modelo apresenta-se ainda mais interessante à medida que possibilita aos pesquisadores utilizarem de uma gama de recursos distintos para testarem e validarem seus algoritmos. Vale a pena salientar que o modelo proposto difere de um sistema gerenciador de banco de dados distribuído, onde os dados podem ficar dispersos em máquinas distintas, mas sob o controle de um único sistema.

A arquitetura explorada apresenta uma situação onde a tecnologia de agentes móveis encaixou-se perfeitamente, possibilitando um mapeamento direto dos conceitos numa estrutura sintática adequada de maneira muito simples e sólida. Primeiramente, uma vez que um agente móvel é uma unidade de *software* autônoma, capaz de transferir sua computação para outra máquina que constitui a rede, sua utilização torna-se plausível quando é necessário execução de rotinas no local de destino. Caso contrário, o agente funciona apenas como um mensageiro, introduzindo um grande *overhead* no que diz respeito ao tráfego de dados na rede, o que poderia ser feito utilizando tecnologias menos sofisticadas. Em segundo lugar, a comunicação local para tratamento dos dados evita que grandes quantidades de dados não tratados trafeguem pela rede. Isto está de acordo com o paradigma de agentes móveis, que pode ser resumido por *levar a computação até os dados ao invés dos dados à computação.*

A linguagem Java mostrou-se bastante adequada para implementar os conceitos da tecnologia de agentes móveis. A independência de plataforma tanto do sistema de agen-

tes quanto das classes implementadas possibilita a construção de um colaboratório bastante heterogêneo no que diz respeito aos sistemas operacionais e *hardwares* utilizados. Acrescente-se a isto o dinamismo da linguagem Java, principalmente no que diz respeito à introspecção e carga dinâmica de classes, o que possibilita utilização de algoritmos que não estejam embutidos na implementação original do sistema. Um ponto fraco na utilização destas tecnologias em conjunto é a exigência de que o implementador defina o *estado* da computação em cada agência, dado que a pilha de execução não é transferida juntamente com o agente móvel.

Uma característica importante do modelo reside no que tange à possibilidade de utilização de bancos de dados distribuídos, podendo ser vistos como um grande repositório virtual, e flexibilidade quanto ao uso de algoritmos. Projetos como *PDB* e *GenBakn* baseiam-se na utilização de um único banco de dados central, o que possibilita ganhos no que diz respeito à unificação dos dados, mas ao mesmo tempo limitam o pesquisador quanto aos algoritmos de comparação, como é o caso do *BLAST*. Nestes, a inclusão de novos algoritmos não é feita baseada na necessidade de pesquisadores isolados, mas da comunidade de usuários como um todo, pois demanda do desenvolvedor do sistema a implementação e disponibilização da técnica solicitada. Em particular no sistema AnaGel, apesar do objetivo ser bem diferente, a possibilidade de implementação e utilização de métricas do próprio usuário provê ao cientista muito mais autonomia para o desenvolvimento de suas pesquisas.

O modelo abordado pode ser aplicado em casos onde a flexibilidade quanto ao uso de métricas é desejada, como é o caso do *BLAST*. Neste *software*, o cientista fica limitado aos algoritmos de alinhamento local rigidamente implementados no sistema.

Utilizando-se das técnicas descritas neste trabalho, poderia-se criar um *GenBank* distribuído, o que facilitaria a expansão do sistema quando da saturação no armazenamento dos dados, e também um *software* com a funcionalidade de recuperação de registros por proximidade como o *BLAST*, porém mais flexível, onde qualquer cientista que desejasse implementar um algoritmo de comparação de sequências poderia fazê-lo segundo seus critérios, procurando por características específicas dentre as sequências presentes no banco, sem a necessidade da construção de um *software* completo especificamente para este fim.

Em aproximadamente 200 classes, num total de 26.000 linhas de código, o desenvolvimento do sistema AnaGel baseado no modelo abordado gerou como resultado uma aplicação de grande utilidade para biólogos e bioquímicos. As demais ferramentas comerciais utilizadas para análise de registros eletroforéticos disponibilizam, em sua grande maioria, utilitários mais sofisticados para análise de géis. No entanto, nenhum daqueles *softwares* avaliados possuem algumas características fundamentais encontradas no AnaGel, como capacidade de ser executado pela *Web* em diversas plataformas, flexibilidade quanto ao uso de algoritmos de comparação do usuário e formação de um repositório global virtual de dados. A tabela da Figura 4.1 compara os diversos *softwares* comercializados com esta

versão do AnaGel, onde a disponibilização de uma dada funcionalidade é marcada com "x":

Os colaboratórios descritos na Seção 2.1.3 apresentam funcionalidades bastante sofisticadas, como controle remoto de aparelhos e interações *online*. Neste sentido, o AnaGel pode ser considerado um colaboratório simplificado, onde os usuários são capazes de compartilhar dados utilizando seus próprios algoritmos de busca nos laboratórios remotos.

Com a implementação na arquitetura proposta o sistema AnaGel tornou-se de fácil manutenção, uma vez que não é necessário a redistribuição do código a cada usuário quando forem feitas atualizações, e também bastante expansível, permitindo que novos laboratórios utilizem dos recursos do *software* sem problema de saturação no que diz respeito ao número de requisições e quantidade de dados armazenados.

Um problema encontrado durante a implementação foi com relação ao sistema de agentes utilizado. Segundo a especificação do *Grasshopper*, este sistema não disponibiliza meios diretos de obtenção da referência para o objeto que representa o agente. A estratégia utilizada foi baseada na utilização dos serviços da agência, dado que o agente tem acesso às funcionalidades do local onde está situado. Assim, ao chegar em alguma agência de destino, cada agente se auto-registra, armazenando sua referência no objeto prestador de serviços para o local onde se encontra. Qualquer agente pode, a partir de então, recuperar a referência para outro seguindo o mesmo princípio.

Um aspecto de grande importância que não foi focado neste trabalho diz respeito à segurança envolvida quando da utilização desta tecnologia. Muito embora alguns aspectos tenham sido levantados na Seção 2.3.2, a implementação de políticas de segurança não foi sugerida no modelo e nem implementada nesta versão do sistema AnaGel. Trabalhos futuros podem abordar esta questão.

Outras funcionalidades poderão ser providas pelo sistema AnaGel, acrescentando-se ferramentas que podem ser encontradas nos *softwares* comercializados atualmente. Os módulos adicionais, tais como para análise de densitometria e geração de matriz de presença de bandas, são de fácil acoplação ao sistema, podendo ser considerados apenas detalhes dentro do projeto como um todo.

A flexibilização quanto ao uso de algoritmos para normalização e estimativa de pesos moleculares são outras funcionalidades que podem ser implementadas, dado que toda estrutura do AnaGel foi desenvolvida considerando-se estas possibilidades.

Característica \ Software	AnaGel	Phoretix	TotalLab	Gel-Pro	GelSite	GeneTools	SigmaGel
Detecção automática de canaletas	x	x	x	x	x	x	
Detecção automática de bandas	x	x	x	x	x	x	x
Correção de distorções em géis	x	x	x	x	x	x	x
Determinação de quantidade de massa das bandas		x	x	x		x	x
Determinação de peso molecular das bandas	x	x	x	x	x	x	x
Calibração Rf *(Retardation factor)*		x	x	x	x	x	x
Elaboração de histogramas		x	x	x	x	x	x
Construção de dendrogramas		x				x	
Cálculo de pontos isoelétricos		x	x				
Compartilhamento de bandas em um mesmo gel	x	x				x	
Interação com o ambiente de trabalho		x	x	x	x	x	x
Tratamento da imagem		x	x	x		x	x
Múltiplas canaletas de calibração	x	x	x	x		x	x
Listas padrões de pesos moleculares		x	x	x		x	
Anotações sobre a imagem		x	x	x			
Personalização de algoritmos para cálculo de pesos moleculares	x			x			
Cálculo de coeficientes de similaridade	x					x	
Execução via *Web*	x						
Banco de dados compartilhado	x						
Colaboração entre laboratórios	x						
Personalização de algoritmos de normalização	x						
Personalização de algoritmos de comparação	x						
Auditabilidade dos resultados	x						
Recuperação de registros por proximidade	x						

Figura 4.1: Tabela comparativa dos *softwares* para análise de eletroforese.

Apêndice A

Eletroforese

Este apêndice foi retirado em sua íntegra da referência [34], de forma a prover uma introdução à eletroforese para leitores não familiarizados com o processo.

A.1 Processo de Eletroforese

O termo eletroforese é geralmente aplicado ao movimento de pequenos íons e macromoléculas eletricamente carregadas em alguma solução geralmente aquosa sob influência de um campo elétrico [16]. A velocidade da migração de uma molécula depende de seu tamanho e forma, da carga que ela carrega, da corrente aplicada e da resistência do meio. Eletroforese em géis é a separação de moléculas carregadas eletricamente em um suporte inerte de gel, resultando na migração das mesmas para regiões distintas do gel chamadas *bandas*.

Um número diversificado de materiais pode ser utilizado na preparação do suporte de gel, incluindo o agar, a agarose, o amido, a celulose e a poliacrilamida. O poder de resolução do processo de eletroforese depende da concentração do material utilizado no suporte: géis de maior concentração dificultam a migração das moléculas, sendo utilizados para separação de moléculas de baixo peso molecular; géis de menor concentração possibilitam o maior deslocamento das moléculas, sendo utilizados na separação de moléculas de maior peso molecular [35].

Um gel é normalmente utilizado para separar diferentes amostras de moléculas. As amostras a serem separadas, normalmente uma mistura de algumas ou até milhares de moléculas de DNA ou de proteínas distintas, são colocadas em pontos distintos do gel (*slots*), próximos a um dos polos do campo elétrico. O volume aplicado a cada *slot* é da ordem de 10^{-6} litro (1 a 50 μl).

A Figura A.1 exemplifica um gel preparado com 10 *slots* prontos para receber as amos-

tras a serem separadas. As moléculas depositadas nos *slots* migram na direção do campo elétrico produzido pelos polos mostrados na figura, percorrendo um caminho em linha reta.

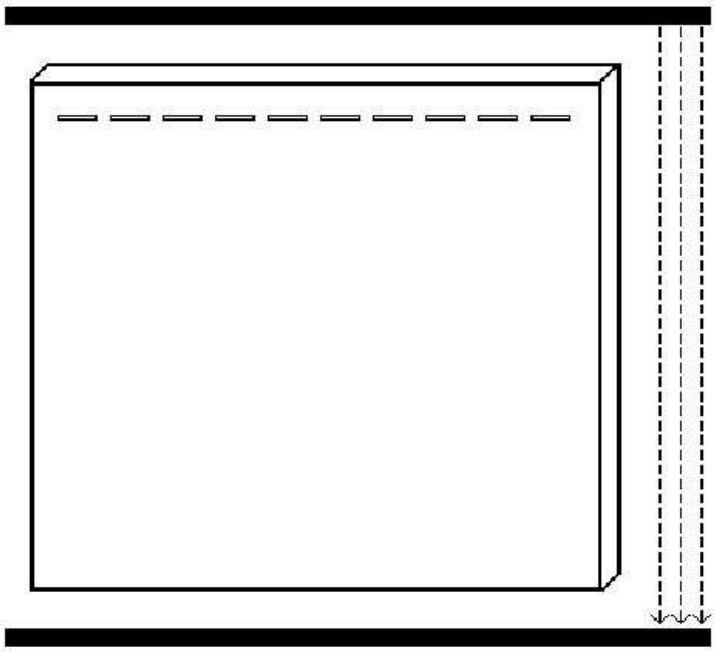

Figura A.1: Gel preparado com 10 *slots* e campo elétrico que será aplicado.

Os eletrólitos usados na eletroforese geralmente consistem em um tampão aquoso, no qual o gel é submerso. Este tampão, além de conter os eletrólitos que possibilitam a passagem da corrente elétrica, têm a finalidade de manter constante o pH do processo. Sob pH próximo da neutralidade, as moléculas a serem separadas têm normalmente carga negativa de maneira que, quando submetidas ao campo elétrico, tendem a migrar do cátodo em direção ao ânodo [16]. Os eletrólitos são utilizados também na composição do gel. Nos sistemas chamados contínuos, os eletrólitos utilizados na composição do gel são os mesmos eletrólitos que aparecem na substância tampão. Nos sistemas descontínuos os eletrólitos do gel são diferentes dos eletrólitos do tampão.

O processo de eletroforese pode ocorrer com o gel na posição vertical (*slab gel*) ou horizontal (*submarine gel*). Em ambos os casos, mantendo o pH da solução próximo a neutralidade, a mobilidade eletroforética das moléculas depende principalmente de seu tamanho e raramente de sua composição (moléculas maiores têm mais dificuldade de migrar pelo gel). O caminho percorrido pelas moléculas de um *slot* forma uma coluna reta chamada de *canaleta*.

A maioria das separações eletroforéticas acontece em temperatura ambiente e nenhum tipo de resfriamento é usado, mas, se ocorrer um sobreaquecimento, as bandas serão distorcidas [35]. A quantidade de calor gerado depende das dimensões do gel e da corrente aplicada. Uma corrente ótima, por sua vez, depende do grau de resolução desejado, do

tamanho das moléculas e do tempo disponível. Em geral, moléculas grandes são melhor separadas em corridas longas com baixa voltagem. Moléculas menores se difundem rapidamente e, portanto, a nitidez das bandas é melhorada com o uso de altas voltagens, havendo um compromisso entre nitidez e separação das bandas.

Existem situações análogas com relação à duração da corrida e concentração do gel. Quanto mais as moléculas migram, maior é a separação entre elas. Entretanto, aumentar a distância migrada pelas moléculas requer ou tempos maiores de corrida ou voltagens maiores, que podem levar à difusão maior de pequenas moléculas ou separação menor de moléculas maiores, respectivamente. Em geral, para distâncias equivalentes, a separação é melhor em géis de maior concentração. Entretanto se as moléculas a serem separadas forem muito grandes, géis de menor concentração são mais adequados [35].

Quando a corrida não acontece em condições ideais, vários problemas podem acontecer. Dos mais comuns pode-se citar a pouca separação entre as bandas e a pouca nitidez das mesmas. Um problema também comum é o chamado *efeito sorriso*: o campo elétrico fica mais fraco nas canaletas externas do gel, resultando em um menor deslocamento nas moléculas aplicadas nestas canaletas. Com o deslocamento maior nas canaletas centrais do que nas canaletas mais externas do gel, o efeito causado se assemelha a um sorriso (Figura A.2).

A distância migrada pelas moléculas no gel depende normalmente de seu tamanho ou peso molecular. Na maioria dos casos, esta distância é inversamente proporcional ao logaritmo do peso das moléculas ($d \propto -\log PM$). Para fragmentos de DNA que pouco variam na conformação esta proporção é largamente aceita [35, 29]. Para fragmentos de RNA e para moléculas de proteína, existem tentativas de se encontrar funções mais acuradas [30, 16]. Para todos os casos, porém, esta função é citada na literatura.

Para se estimar o peso molecular das diversas bandas resultantes da eletroforese, uma mistura de moléculas de pesos conhecidos é aplicada em uma ou mais canaletas do gel. Esta mistura, chamada de *padrão*, pode ter fragmentos de DNA de tamanhos conhecidos. Normalmente o padrão é aplicado na primeira ou última canaleta do gel (ou ambas), e serve de marcador de peso molecular para as demais canaletas. Este processo é chamado de *calibração do gel*.

Após a separação das moléculas o resultado deve ser visualizado. Dependendo da técnica utilizada, um corante específico para as moléculas separadas é aplicado ao gel, de maneira que apenas as bandas são visualizadas. Muitas vezes as moléculas separadas têm pesos moleculares tão variados que sua visualização resulta em um borrão contínuo nas canaletas. Nestes casos, a substância que se aplica ao gel deve permitir a visualização apenas das moléculas de interesse.

A Figura A.2 mostra a imagem de um gel de poliacrilamida (concentração 4%) que contém na primeira canaleta o padrão $\phi X174/HaeIII$ de pesos moleculares conhecidos e

nas demais canaletas fragmentos de DNA de amostras de *Fasciola hepatica*. Neste gel é possível observar um pequeno efeito sorriso.

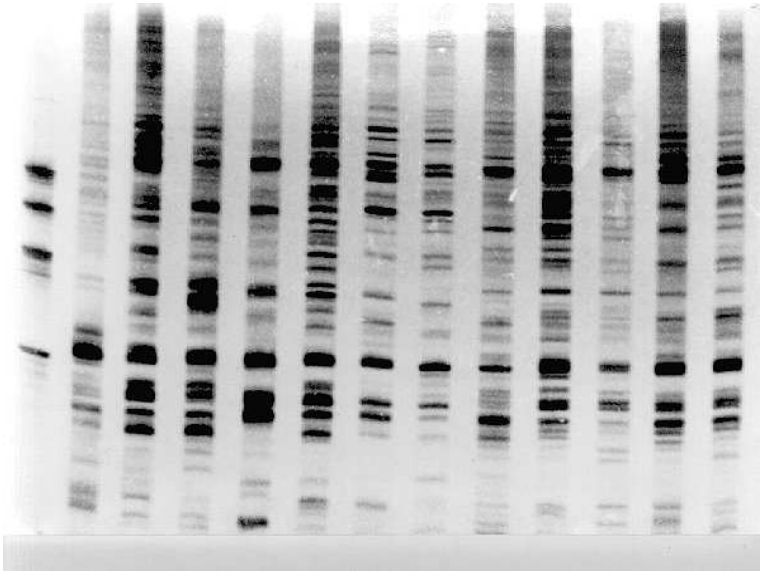

Figura A.2: Gel de poliacrilamida a 4% com DNA de amostras de *Fasciola hepatica* amplificado pela *Polymerase Chain Reaction*.

A.2 Aplicações e Técnicas de Eletroforese

As técnicas utilizadas em um processo de eletroforese são variadas e envolvem desde a preparação do material a ser separado até o processo final de visualização do gel. A escolha de uma técnica depende do material a ser separado (proteína, DNA ou RNA) e da finalidade da separação. A seguir serão descritas algumas aplicações da eletroforese e técnicas existentes para proteínas e fragmentos de DNA.

A.2.1 Eletroforese de proteínas

Vários métodos de separação de moléculas de proteínas existem na literatura. Alguns métodos separam proteínas nativas, enquanto outros são desnaturantes. A desnaturação de proteínas antes do processo de eletroforese tem por objetivo homogeneizar a forma e carga elétrica das moléculas, tornando a separação dependente principalmente do peso molecular.

Os principais usos da separação de proteínas desnaturadas são a determinação do tamanho de um componente protéico, a estimativa do grau de pureza de uma proteína em uma solução, a purificação de uma espécie de proteína para posterior estudo e o fracionamento

de uma mistura complexa de proteínas para comparação com separações não desnaturantes [2]. Em geral, os métodos deste tipo de separação não variam significativamente de aplicação para aplicação, sendo comum reagir as proteínas separadas com um reagente de fácil visualização. Um número grande de reagentes pode reagir com qualquer proteína sem afetar sua atividade biológica [18]. O método de visualização de moléculas de proteínas separadas pela eletroforese por um reagente corante é chamado de Coloração de Proteínas. Uma variação deste método é a Coloração pela Prata que aumenta a sensibilidade do processo de visualização [15]. Métodos de autoradiografia e fluorografia também aumentam a sensibilidade da visualização [39].

Quando se deseja estudar a atividade biológica de uma proteína, o processo de eletroforese não pode ser desnaturante. A coloração de proteínas não pode ser utilizada neste caso, pois os reagentes usados como corantes desnaturam as proteínas. Estas devem ser visualizadas após o isolamento daquelas de atividade biológica desejada.

Anticorpos são comumente usados para detectar antígenos em misturas complexas, em uma técnica de imunodetecção chamada *Immunobloting* [2]. Nesta técnica as proteínas nativas são transferidas do gel para uma membrana após a eletroforese, onde ficam imobilizadas. A esta membrana é aplicada uma solução com o anticorpo que reage apenas com a proteína que se deseja isolar. O passo seguinte é a aplicação de uma proteína que reconhece o anticorpo (anti-anticorpo), carregada com uma enzima que pode ser visualizada por coloração, quimioluminescência ou autoradiografia.

A eletroforese de proteínas pode ser utilizada para se explorar variabilidade genética, na medida em que sua estrutura primária é diretamente dependente da seqüência de bases do gene que dirige sua síntese [6]. Quando a proteína estudada é uma enzima, aplica-se ao gel o substrato a ser modificado apenas por esta enzima e revela-se o produto formado através de outra enzima que o transforma em produto corado e insolúvel. Desta forma apenas as bandas que possuem enzimas que modificam o primeiro substrato são coradas. Quando bandas em posições diferentes do gel são visualizadas as enzimas destas bandas são proteínas diferentes de mesma propriedade enzimática. Esta técnica de visualização de isoenzimas é chamada de *Histoquímica*.

Um sistema especial de gel, chamado de focalização isoelétrica, constitui-se de um gel no qual se estabelece um gradiente de pH. Nesse sistema as moléculas de proteína se deslocam em direção aos polos elétricos até atingirem uma região em que o pH do gel é igual ao ponto isoelétrico de cada proteína em particular. No ponto isoelétrico as cargas positivas e negativas das proteínas são igualadas e a atração elétrica desaparece. A solubilidade da proteína também atinge seu valor mínimo neste ponto, o que determina sua imobilização, formando uma banda.

A.2.2 Eletroforese de DNA

Uma grande aplicação da eletroforese é a separação de fragmentos de DNA, que pode ser usada para seqüenciamento, mapeamento de sítios de restrição, isolamento de genes e identificação, comparação e classificação de indivíduos.

Fragmentos de DNA são obtidos a partir de cortes na fita de DNA em sítios determinados. O corte da fita de DNA nestes sítios resulta em fragmentos de tamanhos diferentes, separados pelo processo de eletroforese. Quando o número de fragmentos cortados de uma fita é grande, o padrão de bandas é característico de cada indivíduo, e guarda semelhanças tanto maiores quanto forem as relações de parentesco entre indivíduos comparados. Este padrão pode ser utilizado para a identificação de indivíduos ou para taxonomia, após análise aproximada dos dados.

Para se saber se duas amostras de DNA pertencem a um mesmo indivíduo, deve-se submetê-las a cortes nos mesmos sítios de restrição e aplicar os fragmentos resultantes à eletroforese. Se o padrão de bandas das duas amostras for o mesmo, a probabilidade de que as amostras pertençam a um mesmo indivíduo é grande.

Esta técnica pode ser utilizada para se construir bancos de DNA para identificação humana mais seguros que os bancos de impressões digitais. Estes bancos têm a vantagem de se poder identificar um indivíduo a partir de qualquer célula do indivíduo, sendo desnecessária sua presença.

Exames de paternidade também se beneficiam da eletroforese de DNA. Como o genoma de um indivíduo é a combinação dos genomas de seus pais, o padrão de bandas resultante da eletroforese de seu DNA, cortado em determinados sítios de restrição, também é a combinação dos padrões de bandas resultantes da eletroforese do DNA de seus pais, cortados nos mesmos sítios de restrição. Para se fazer o exame de paternidade, subtrai-se do padrão de bandas do filho, todas as bandas pertencentes ao padrão de sua mãe. O exame indica alta probabilidade de paternidade apenas se todas as bandas restantes pertencerem ao padrão de bandas do suposto pai [20].

Testes de eletroforese podem ser utilizados também na identificação de doenças infecciosas. Como um indivíduo infectado com algum parasito carrega seu DNA, ao se separar seu DNA para um teste de eletroforese, separa-se também o DNA do parasito. O padrão de bandas decorrente da eletroforese destes DNAs conterá bandas oriundas de ambos os indivíduos. Se o padrão de bandas do DNA do parasito for conhecido, o diagnóstico da doença será positivo se todas as bandas estiverem presentes no resultado da eletroforese do material extraído do indivíduo infectado.

A eletroforese pode ser utilizada também em taxonomia biológica. O parentesco entre grupos taxonômicos, dado pela semelhança entre o DNA dos indivíduos pertencentes a estes grupos pode ser quantificado pela comparação entre os padrões de bandas resultantes da eletroforese destes DNAs. Amostras de DNA que compartilham um número grande de

bandas na análise eletroforética são consideradas semelhantes. Isto aponta a possibilidade de haver um ancestral comum aos grupos, cujo DNA, por mutações originou novas espécies, eliminando sítios de restrição e criando novos sítios. A comparação entre os padrões eletroforéticos gera medidas de distância (ou semelhança) entre grupos taxonômicos, utilizadas na construção de diagramas de relações taxonômicas.

Técnicas de análise de DNA

A técnica mais simples de eletroforese de DNA é a chamada *Restriction Fragment Length Polymorphism (RFLP)*. O DNA é cortado por uma enzima de restrição que reconhece sítios nos quais a seqüência de uma fita é igual ao reverso de seu complemento. Os fragmentos resultantes têm tamanhos diversos e são submetidos à eletroforese para separação. A visualização é feita sob irradiação ultravioleta após a aplicação de um corante que, ao intercalar com os ácidos nucléicos, fluoresce sob a luz ultravioleta [20].

O DNA de genoma eucariótico cortado com enzima de restrição aparece como uma mancha contínua no gel após a eletroforese. Isto acontece pois o corte origina fragmentos de milhares de tamanhos. Para estes casos, a técnica de RFLP não surte efeito, uma vez que as diferenças entre perfis eletroforéticos não serão visualizáveis. O processo de visualização deve evidenciar um número limitado de fragmentos de DNA.

A técnica de *DNA Fingerprinting* procura por regiões polimórficas do genoma que possuem pequenas seqüências que se repetem em tandem por todos os cromossomos, chamadas minissatélites. Regiões diferentes do genoma e indivíduos distintos possuem diferentes números de repetições destas seqüências [20]. O corte do DNA deve ser feito por uma enzima de restrição que reconhece seqüências localizadas no início e fim das seqüências repetitivas, produzindo fragmentos que contêm as seqüências repetitivas e fragmentos que não contêm estas seqüências.

A visualização das bandas na técnica de *DNA Fingerprinting* envolve a utilização de uma sonda que reconhece a seqüência que se repete. A sonda constitui um fragmento de poucos a dezenas de nucleotídeos, marcada de forma a poder ser detectada por algum método indireto que pode ser autoradiografia, um método enzimático ou colorimétrico. O método indica as regiões do gel onde a sonda encontrou fragmentos com seqüência de nucleotídeos complementar à mesma, hibridizando-se a ela. O resultado será um número limitado de bandas passível de análise comparativa, diferentemente do padrão original que podia conter centenas de milhares de bandas justapostas, formando uma mancha contínua com aspecto idêntico para qualquer indivíduo.

A técnica de *Polymerase Chain Reaction (PCR)* permite a amplificação enzimática de seqüências específicas de DNA. Pequenos oligonucleotídeos chamados iniciadores são projetados de maneira a anelar nas duas extremidades da seqüência a ser amplificada. Esta é desnaturada, os iniciadores se anelam às suas seqüências complementares presentes no

DNA alvo e uma cópia de cada fita é produzida enzimaticamente por um DNA polimerase. Este processo é repetido dezenas de vezes de maneira que novas cópias são feitas das cópias originais e das cópias produzidas em passos anteriores, resultando em uma amplificação exponencial da seqüência desejada [20].

O número de fragmentos amplificados será exponencialmente maior do que o número de fragmentos não amplificados no material submetido a eletroforese. Desta forma, as amostras aplicadas no gel após amplificação conterão quantidades insignificantes das seqüências não amplificadas e apenas os fragmentos amplificados serão visualizados após a coloração do gel.

Uma variação da técnica de PCR é a chamada *Random Amplified Polymorphic DNA (RAPD)*. Esta técnica consiste na utilização de primers arbitrários que não necessitam de informação prévia da seqüência a ser amplificada. Os fragmentos amplificados são em número exponencialmente maior do que os não amplificados, de maneira que apenas estes são visualizados. Esta técnica pode ser utilizada para fragmentos de DNA de qualquer espécie e é útil na identificação e comparação de indivíduos [40].

Apêndice B

Sistema AnaGel - Manual do Usuário

B.1 Resumo

O sistema *AnaGel* [34, 33] foi desenvolvido para auxiliar pesquisadores interessados em analisar registros eletroforéticos em uma dimensão em geral. Esta nova versão foi elaborada para ser executável via Internet, possibilitando sua utilização sob diversas plataformas em qualquer ponto ligado à WWW.

Além de várias ferramentas para análise de registros eletroforéticos, o *AnaGel* permite a intercomunicação entre os laboratórios para desenvolvimento de trabalhos conjuntos, possibilitando, inclusive, utilização de algoritmos de comparação do próprio pesquisador.

B.2 Requisitos de Hardware/Software

Para a utilização do sistema é importante que a máquina possua pelo menos 32MB de memória RAM. É necessário a instalação do Java Plug-in 1.3.1, que é verificada e recomendada pelo próprio *AnaGel*. O software foi testado com os navegadores Netscape Communicator 4.7, Internet Explorer 5 e Hot Java 3.0.

B.3 Utilização do Sistema

B.3.1 Cadastro de usuários

Usuários interessados em utilizar o sistema devem enviar e-mail para anagel@cenapad.-ufmg.br, contendo nome, login preferencial e e-mail para contatos, solicitando a criação de uma conta.

B.3.2 Autenticação do usuário

Através de um *browser*, o *AnaGel* pode ser alcançado acessando-se a página *http://anagel.cenapad.ufmg.br/anagel/anagel.html* (1, Fig. B.1). Nesta página são solicitados o login (2, Fig. B.1) e senha (3, Fig. B.1) do usuário, que devem ser aqueles definidos no momento do cadastro. Clicando-se em **Ok** (4, Fig. B.1) é iniciada a autenticação do usuário. Estando corretos o login e a senha, a janela principal do sistema é exibida.

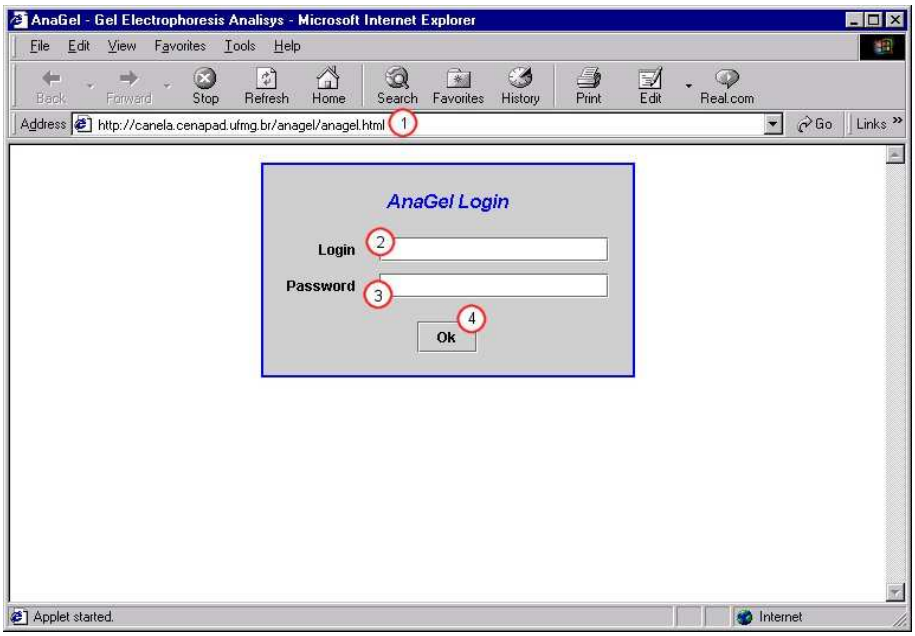

Figura B.1: Autenticação do usuário no sistema.

B.3.3 Processamento da imagem

Aquisição de imagens de géis

As imagens dos géis devem ser submetidas antes de serem processadas. Isto deve ser feito através de uma página específica que pode ser acessada clicando-se em **Click Here for Image Upload** , na paleta *Gel Image/Image* (4, Fig. B.3).

Nesta página (Fig. B.2), o usuário dever fornecer seu login e senha, selecionar a opção **Gel Image** , escolher o arquivo a ser submetido (os arquivos das imagens devem estar em formato *gif* ou *jpg*) e, em seguida, submeter as informações.

Após a submissão os nomes das imagens poderão ser visualizadas na paleta *Gel Image/Image*, quando a opção **From File** (1, Fig. B.3) estiver selecionada. Caso o gel submetido não esteja na lista, deve-se acionar a opção **Refresh** (5, Fig. B.3) nesta mesma paleta.

Uma vez localizado o arquivo do gel, é necessário acionar o botão **Get** (6, Fig. B.3). A imagem é então exibida em poucos instantes (9, Fig. B.3). É de suma importância que a

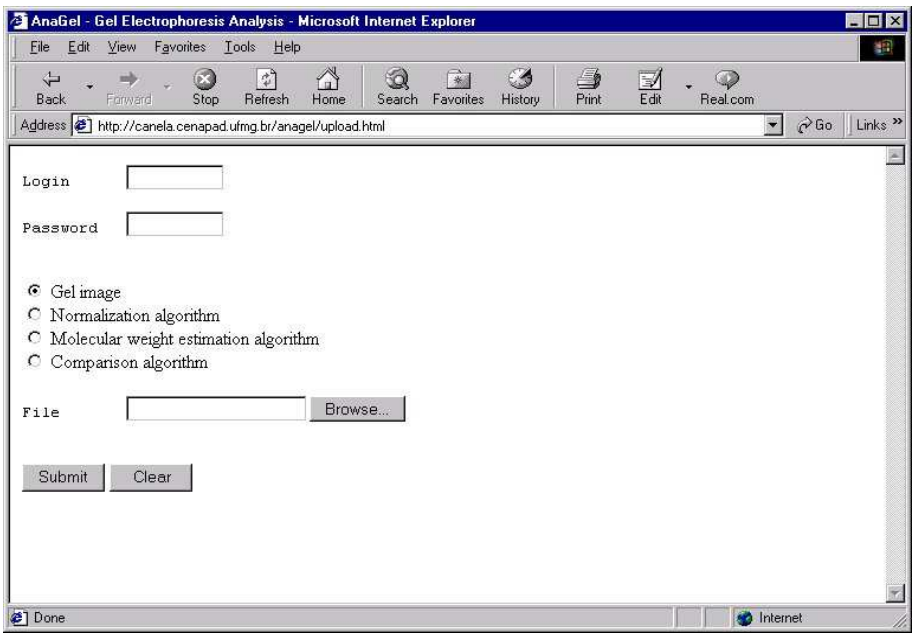

Figura B.2: Submissão de imagens de géis.

imagem esteja com as canaletas na posição vertical e com os pesos moleculares decrescendo de cima para baixo.

É permitido ao usuário a visualização da imagem ampliada ou reduzida, bastando alterar a porcentagem do tamanho original desejada (10, Fig. B.3) e acionar a opção **Zoom** (11, Fig. B.3).

Recuperação de géis do banco de dados

Escolhendo-se a opção **From Database** (2, Fig. B.3), é permitido ao usuário recuperar géis cadastrados no banco de dados, bastando para tanto que selecione da lista de géis (7, Fig. B.3) aquele de interesse e acione o botão **Open** (8, Fig. B.3). O gel é exibido em poucos instantes.

Marcação de canaletas

Em seguida à visualização do gel, as canaletas devem ser demarcadas. O sistema pode procurá-las e marcá-las automaticamente, caso seja acionado o botão **Find** (1, Fig. B.4).

O usuário pode optar por marcá-las sem a ajuda do software ou remover canaletas marcadas erroneamente. Para cada canaleta a ser demarcada, o usuário deve acionar o botão **Insert** (2, Fig. B.4) e pressionar o botão esquerdo do mouse aproximadamente no centro horizontal da canaleta. Para remover alguma delas, deve-se selecionar a de interesse (6, Fig. B.4), clicando-se com o botão esquerdo do mouse em quaisquer das extremidades

Figura B.3: Recuperação da imagem do gel para processamento.

da marcação (superior ou inferior esquerda, superior ou inferior direita), e, em seguida, acionar **Remove** (3, Fig. B.4).

As canaletas podem ser ajustadas para acompanhar a inclinação da imagem, clicando-se com o botão esquerdo em alguma das extremidades e arrastando-se o mouse até a posição desejada.

Todas as canaletas demarcadas possuem a mesma largura, podendo ser alteradas para mais largas ou mais estreitas, iteragindo-se com os botões do painel **Width** (5, Fig. B.4).

Limites do gel e curvas de calibração

Os limites do gel e as curvas de calibração de pesos moleculares só podem ser demarcadas após serem localizadas as canaletas (é necessário ser identificada pelo menos uma canaleta). As curvas que delimitam o gel são úteis para corrigir o efeito sorriso e delimitar a área útil da imagem.

Para a marcação do limite superior (4, Fig. B.5), deve-se acionar a opção **Upper Bound** (1, Fig. B.5). Uma semi-reta é traçada desde o último ponto ao ponto do clique do botão esquerdo. Estas semi-retas devem acompanhar os defeitos da imagem. Para finalizar-se a marcação, deve-se acionar o botão direito do mouse. O mesmo procedimento vale para a marcação do limite inferior (5, Fig. B.5), devendo-se apenas acionar a opção **Lower Bound** .

Em seguida à marcação dos limites, pelo menos três curvas de calibração de pesos moleculares (6, Fig. B.5) devem ser especificadas. Para tanto, deve-se acionar a opção

Figura B.4: Delimitação das canaletas.

Insert Curves (3, Fig. B.5) e, para cada curva a ser adicionada, clicar com o botão esquerdo sobre a banda de calibração, informando-se o peso molecular na caixa de diálogo exibida. São aceitos pesos moleculares apenas na ordem decrescente, de cima para baixo. Uma linha é exibida, indicando os pontos que possuem o mesmo peso molecular (7, Fig. B.5), acompanhando as linhas limítrofes.

Importante: *Após marcadas as curvas de calibração, uma vez acionadas as opções* **Upper Bound** *ou* **Lower Bound** *, todas as curvas anteriormente marcadas serão removidas. O acionamento destas opções após a marcação dos limites deve ser utilizado para que estes sejam apagados e remarcados em seguida.*

Extração de perfis

O perfil de cada canaleta é extraído a partir do acionamento da opção **Extract** (1, Fig. B.6) da paleta *Gel Image/Profile*. Para tanto, as canaletas devem estar identificadas e com as marcações de calibração e limite devidamente especificadas. O perfil é extraído desde o limite superior até o inferior, descartando as informações acima do limite superior e abaixo do inferior.

Os perfis extraídos do gel podem ser visualizados pela paleta *Gel Image/Gel Profile* (1, Fig. B.7). Pelo acionamento do comando **Get Profiles** um relatório com os perfis é gerado como figura *gif* e enviado para o usuário por e-mail.

Através da opção **Lane Number** do painel **Lane** (2, Fig. B.6) é escolhido o perfil

Figura B.5: Delimitação do gel e marcação de curvas de calibração.

a ser processado. A canaleta original é exibida no painel **Original Lane** (7, Fig. B.6), indicando a posição das bandas demarcadas. O painel **Profile** (8, Fig. B.6) exibe o perfil extraído que representa a canaleta, bem como a posição das bandas marcadas. No painel **Bands** (9, Fig. B.6) pode ser visualizado o peso molecular calculado para cada banda.

O campo **Date** do painel **Lane** deve indicar a data da corrida e o campo **Obs** pode ser utilizado para fornecer informações adicionais sobre a canaleta.

Uma amostra deve ser selecionada para a canaleta, pela opção **Identifier** do painel **Sample** (3, Fig. B.6).

Logo após o processamento de cada canaleta, deve-se acionar a opção **Save Lane** (10, Fig. B.6) para que os dados cadastrados sejam salvos.

Cadastro de dados sobre as amostras

Através do painel **Sample** deve-se fornecer informações sobre a amostra processada na canaleta. Pode-se escolher dentre quaisquer das cadastradas ou criar um novo registro para amostra.

Uma amostra já cadastrada pode ser associada selecionando-se o identificador da lista **Identifier** . Pode-se criar um novo identificador atravé da opção **New...** (1, Fig. B.8) ao final da lista. Deve-se então fornecer os diversos dados para a amostra, uma vez preenchido o campo **Identifier** (2, Fig. B.8). Os dados de amostras cadastradas não podem ser alterados uma vez registrados no banco de dados.

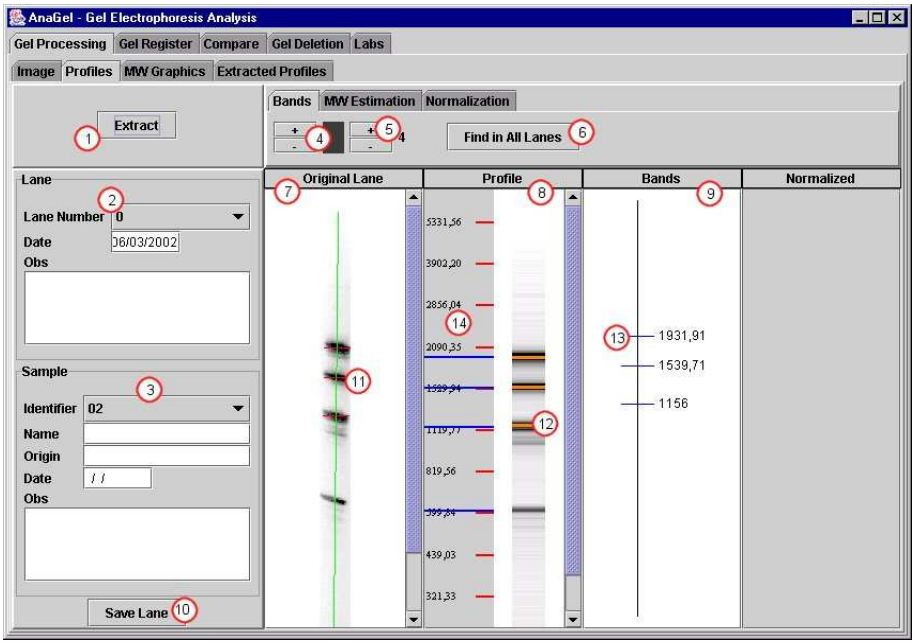

Figura B.6: Processamento das canaletas.

Marcação de bandas

As bandas podem ser localizadas automaticamente pelo sistema, ajustando-se a tonalidade (4, Fig. B.6) e espessura (5, Fig. B.6) mínimas a serem consideradas na paleta **Bands** (4, Fig B.6). Uma vez ajustados os parâmetros, a opção **Find in All Lanes** (6, Fig. B.6) pode ser acionada, de forma a marcar as bandas com as características indicadas em todas as canaletas.

Também é possível marcar bandas não encontradas, clicando-se com o botão direito na posição desejada, e remover bandas marcadas em posições incorretas, acionando-se o botão esquerdo sobre a banda a ser excluída. Estas operações só são possíveis quando feitas na representação do painel **Profile** (8, Fig. B.6).

Bandas marcadas e/ou inseridas manualmente são *descartadas* quando o sistema faz busca automática, o que ocorre quando alterados os parâmetros de procura de bandas (são perdidas apenas as marcações na canaleta corrente) ou acionada a opção **Find in All Lanes** (são perdidas as marcações manuais de todas as canaletas).

No painel **Profile** é exibida uma régua (14, Fig. B.6), que indica o valor do peso molecular para cada região do gel, bem como onde estão localizadas as marcações de calibração.

O painel **Original Lane** (7, Fig. B.6) exibe a canaleta original, sua inclinação e a localização das bandas encontradas (11, Fig. B.6).

Podem ser visualizados no painel **Bands** (9, Fig. B.6) os pesos moleculares de cada banda marcada.

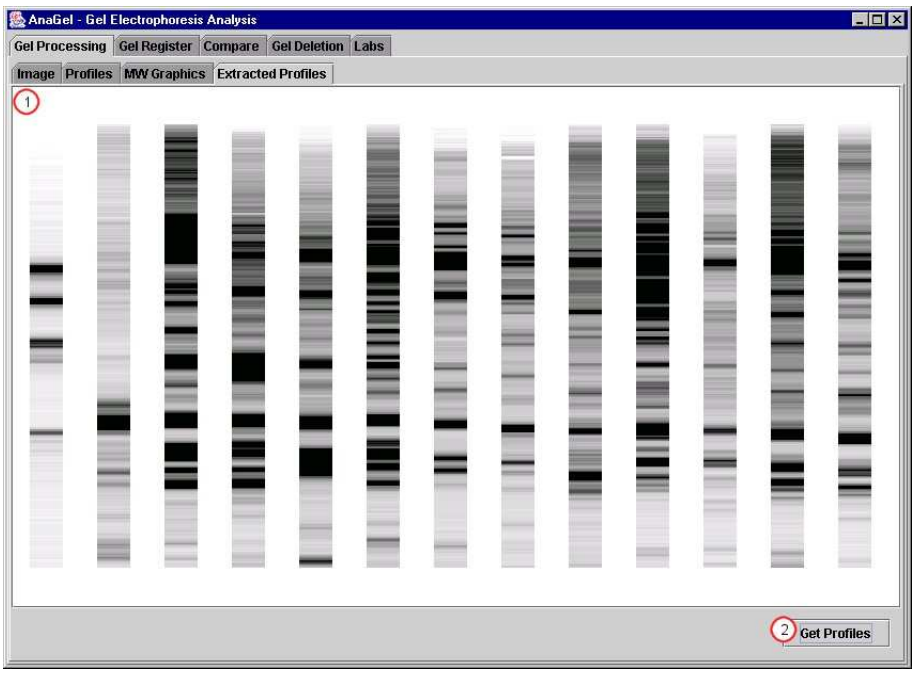

Figura B.7: Perfis normalizados extraídos do gel.

Estimativa de pesos moleculares

Pela paleta **MW Estimation** é possível mudar a forma de cálculo de peso molecular das bandas. Pode-se selecionar alguma da lista (1, Fig. B.9) e acionar o botão **Estimate** (2, Fig. B.9) para que os cálculos sejam feitos. A forma padrão de cálculo é utilizando-se a aproximação logarítmica.

Gráfico de estimativa de pesos moleculares

O gráfico que representa a estimativa dos pesos moleculares pode ser visualizado a partir da paleta *Gel Image/MW Graphics*. É exibido o gráfico de regressão linear (1, Fig. B.10) e a curva original (2, Fig. B.10). A inclinação da reta (**m**) e a interseção com o eixo y (**b**) também são fornecidas (4, Fig. B.10).

O usuário pode adquirir uma imagem no formato *gif* do gráfico por e-mail pelo acionamento do botão **Get Graphics** .

Normalização das canaletas

Como último passo no processamento das canaletas, estas devem ser normalizadas entre um peso molecular mínimo e máximo (1, 2, Fig. B.11) da paleta *Normalization*, que devem estar dentro da escala de pesos moleculares válidos para o gel. Este passo tem por objetivo corrigir o efeito sorriso, expandindo todas as canaletas de forma a ficarem do tamanho

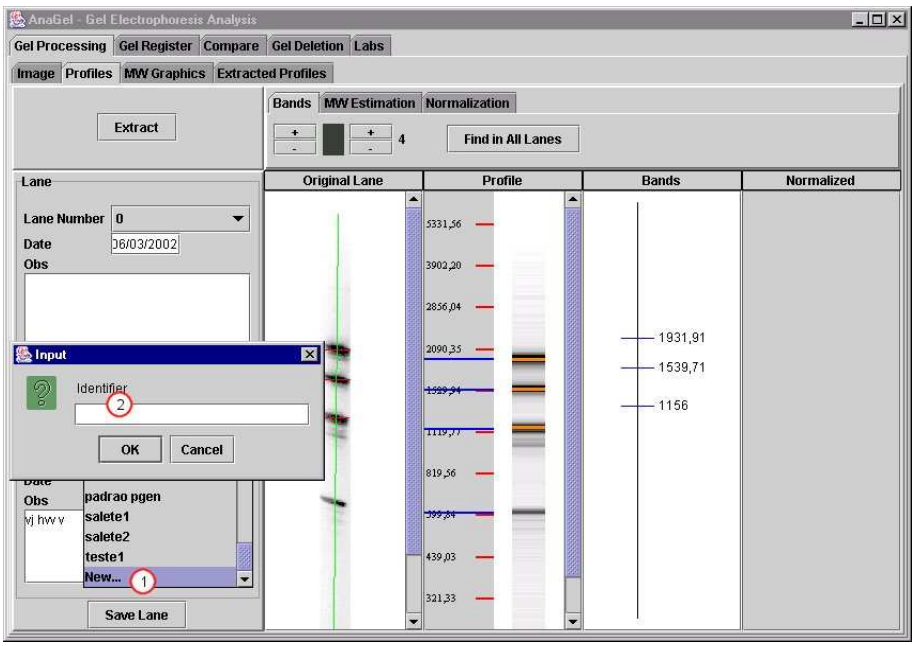

Figura B.8: Cadastro de amostras.

daquela que mais migrou durante o processo de eletroforese. Em seguida, deve-se acionar o botão **Normalize** (3, Fig. B.11) para efetivar a normalização.

A canaleta normalizada é exibida no painel **Normalized** , sendo mostrada também uma régua equivalente àquela presente no painel **Profile** .

Cadastro de dados do gel

Pela paleta *Gel Register/Gel Data* é possível o cadastro de dados relativos à corrida. É opcional o preenchimento do campo **Gel Name** (1, Fig. B.12), que pode facilitar futura identificação do gel. O campo **Date** (11, Fig. B.12) deve ser sempre preenchido com uma data válida, no formato *dd/mm/aaaa*.

Os demais campos são opcionais. Todos os campos numéricos podem ser preenchidos com valores decimais. No painel **Material** (2, Fig. B.12) deve ser selecionado o tipo do material utilizado na corrida.

Informaçõs adcionais do processo podem ser fornecidas no campo da parte inferior da tela (5, Fig. B.12).

A opção **Save** (10, Fig. B.12) registra todos os dados do gel bem como das canaletas processadas no banco de dados.

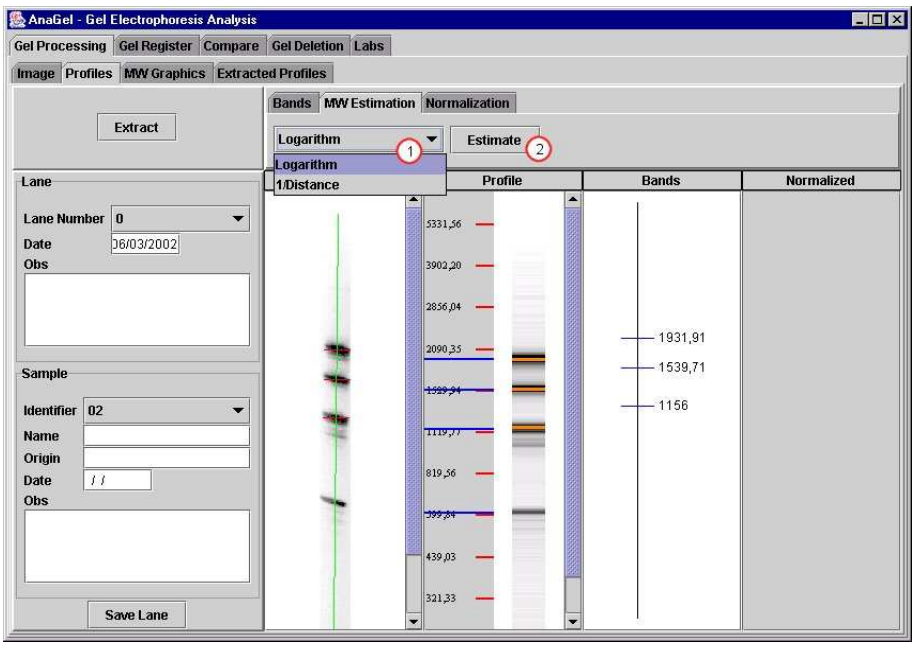

Figura B.9: Opções de estimativas de pesos moleculares.

Permissões

É possível, no momento do cadastro, especificar quais usuários poderão acessar os dados registrados para o gel, podendo utilizar as canaletas cadastradas para comparação com outras. Aos usuários habilitados é possível apenas a visualização e utilização das informações, não podendo ser estas modificadas.

Os usuário podem ser habilitados selecionando-os da lista **Users** (6, Fig. B.12) e adicionando-os à lista **Enabled** (7, Fig. B.12) através da opção **Add** (8, Fig. B.12). Estes podem ser removidos da lista **Enabled** pela opção **Remove** (9, Fig. B.12).

B.3.4 Remoção de géis

Através da paleta *Gel Deletion* o usuário pode excluir do banco de dados os géis cadastrados por ele que não têm mais utilidade ou que foram cadastrados erroneamente. Para tanto, deve selecioná-lo na lista (1, Fig. B.13) e acionar o botão **Remove** (2, Fig. B.13). Caso queira verificar, acionando-se a opção **View Gel** (3, Fig. B.13), a imagem do gel selecionado é exibida no painel lateral (4, Fig. B.13).

Ao ser removido um gel, todas os dados extraídos para ele, tais como perfis de canaletas, bandas, curvas de calibração, etc. são automaticamente excluídos do banco, sendo impossível recuperá-los novamente.

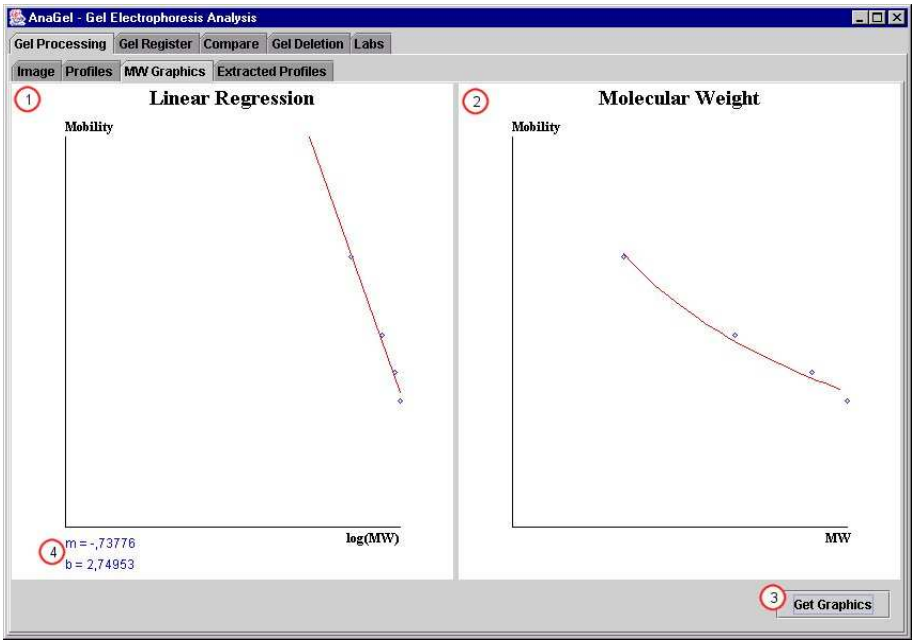

Figura B.10: Gráfico de estimativa dos pesos moleculares.

B.3.5 Comparação de amostras

Requisição (filtragem) das amostras

Pela paleta *Compare/Request* são selecionadas as canaletas a serem comparadas. Vários parâmetros podem ser utilizados para seleção das canaletas, que são tipo de material, nome, data de cadastro, autor, tipo, matriz, concentração, tensão, corrente, potência, temperatura, duração e buffer do gel; identificador, nome, origem e data da amostra. O parâmetro a ser utilizado pode ser escolhido da lista disponível no painel **Parameters** (1, Fig. B.14). Uma vez selecionada a regra, esta pode ser adicionada à lista de regras a serem utilizadas do painel **Rules** (3, Fig. B.14) através da opção **Add Rule** (2, Fig. B.14). A opção **Remove Rule** (4, Fig. B.14) remove as regras selecionadas.

Acionando-se a opção **Filter** (5, Fig. B.14) são solicitadas do banco de dados as amostras que satisfazem a todas as regras indicadas no painel **Rules** , sendo exibidas no painel **Matches** (6, Fig. B.14).

Pode-se selecionar apenas aquelas de interesse e adicionar à lista **Selected** (9, Fig. B.14) pela opção **Add Selected** (7, Fig. B.14), ou adicionar todas, através da opção **Add All** (8, Fig. B.14).

Clicando-se em **Filter** com a lista de parâmetros vazia causa a requisição de todas as amostras que são permitidas ao usuário.

Novas regras podem ser criadas para seleção e outras amostras podem ser adicionadas da mesma forma que explicado anteriormente. A lista **Matches** é temporária, servindo

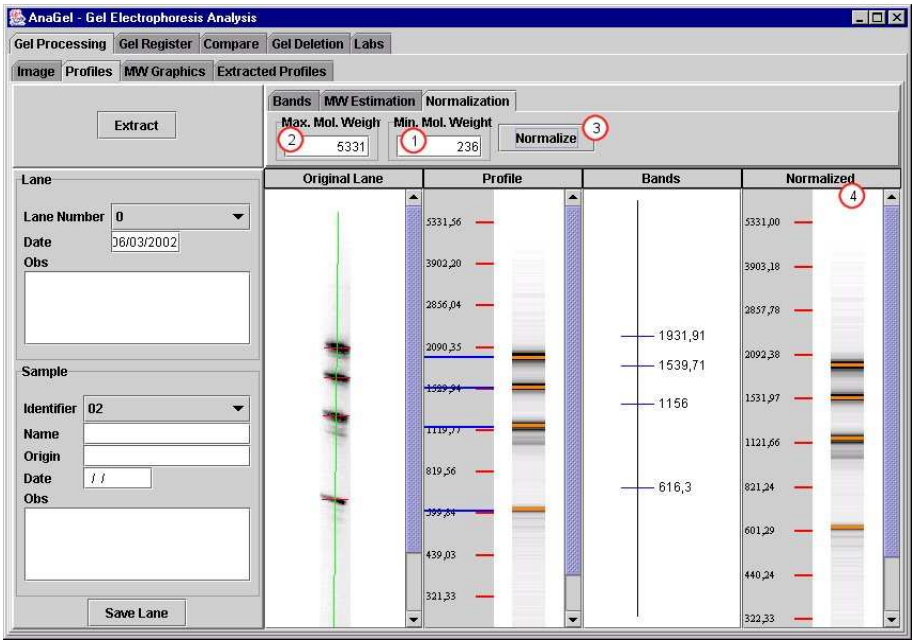

Figura B.11: Normalização das canaletas.

apenas para listar as amostras compatíveis com uma dada pesquisa. A lista **Selected** contém todas as canaletas escolhidas no decorrer das pesquisas.

Acionando-se **Remove Selection** (10, Fig. B.14) do painel **Selected** causa o remoção dos registros selecionados dessa lista.

Através das opções do painel **Lab's Cooperation** (11, Fig. B.14) o usuário pode solicitar que amostras sejam recuperadas de bancos de dados de outros laboratórios. Para tanto, é necessário que sejam fornecidos a amostra de calibração para comparações, o intervalo de pesos moleculares de interesse, a distância máxima permitida para que sejam incluídas no resultado, e a faixa de erro admitida para pesos moleculares. Acionando-se o botão **Search in Labs** , os laboratórios são consultados e o resultado da busca remota é exibido no painel **Matches**. Juntamente com a canaleta de calibração e os dados supra-citados, também as regras presentes no painel **Rules** são enviados, sendo utilizados para uma pré-seleção antes da avaliação das distâncias.

Seleção das amostras

Uma vez requisitadas ao banco de dados, pode-se comparar quaisquer subconjuntos de canaletas de interesse, através da paleta *Compare/Select*.

Canaletas podem ser selecionadas clicando-se sobre o perfil exibido (1, Fig. B.15) ou pela lista que contém os dados da canaleta (2, Fig. B.15).

Pode-se marcar uma canaleta como mestre (4, Fig. B.15), acionando-se a opção **Set Master** (5, Fig. B.15). Neste caso, havendo uma única canaleta selecionada, esta é

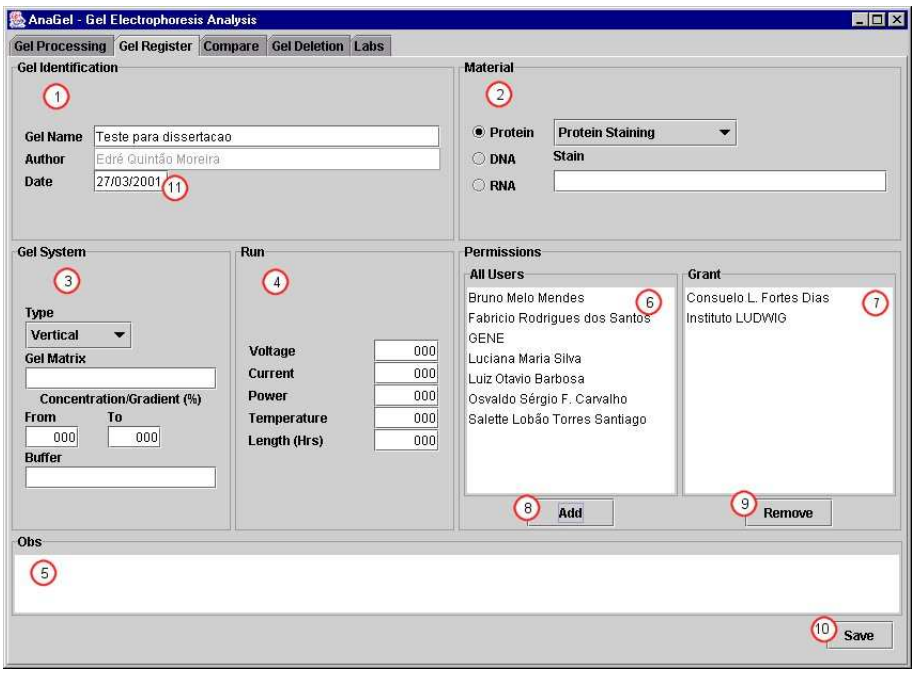

Figura B.12: Entrada de dados sobre o gel e a corrida.

assinalada como mestre, e havendo mais de uma, aquela que aparecer primeiro na lista é escolhida para tal.

Pela opção **Remove Master** (6, Fig. B.15) a canaleta é desmarcada como mestre. Havendo uma canaleta selecionada como mestre, a indicação de outra substitui a anterior.

Para comparação, é necessário que se tenha selecionado pelo menos duas canaletas, incluindo-se a mestre. O usuário deve especificar a região de comparação de interesse no painel **MW Range** (7, Fig. B.15). Esta deve ser uma região comum a todas as amostras selecionadas, delimitando o intervalo a ser utilizado no momento das comparações.

As opções **Spaced Marks** e **Bands MW** (8, Fig. B.15) podem ser usadas para visualização de detalhes relativos ao peso molecular das bandas. A primeira opção, caso selecionada, faz com que sejam exibidos espaçadamente os valores de pesos moleculares. A segunda opção, caso marcada, força o programa *AnaGel* a mostrar o valor dos pesos moleculares de cada banda.

Caso acionado o botão **Get Report** (9, Fig. B.15), um relatório é enviado por e-mail, sendo um arquivo *gif* com a mesma visualização exibida ao usuário.

Comparação por alinhamento de perfis

Através da paleta *Compare/Alignment/Filter Profile* é possível aplicar filtros aos perfis selecionados, possibilitando a obtenção de melhores resultados.

Os perfis a serem tratados podem ser selecionados clicando-se sobre sua imagem (1, Fig.

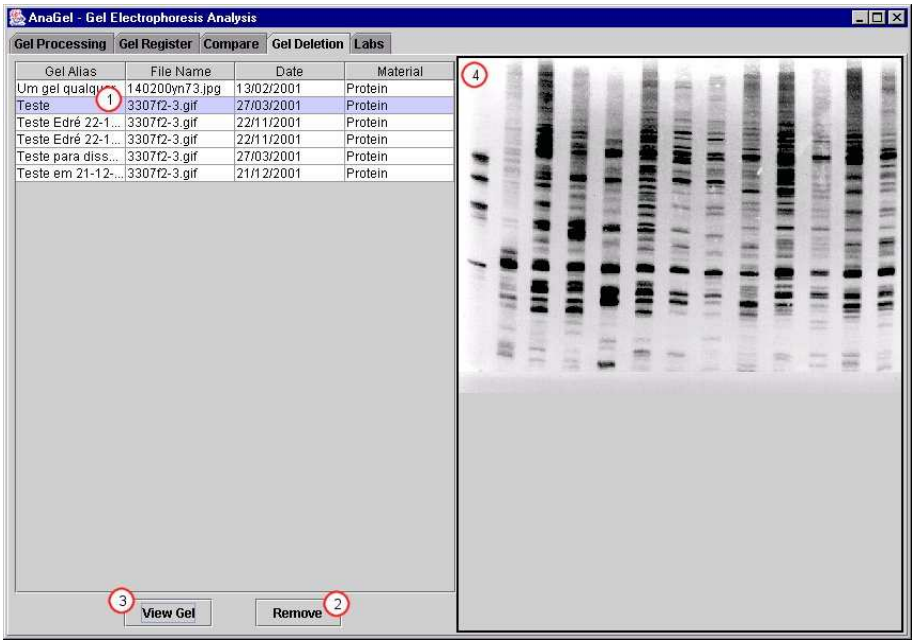

Figura B.13: Remoção de gel cadastrado.

B.16). Pode-se selecionar todos os perfis pela opção **Select All** (2, Fig. B.16) e desfazer a seleção através de **Deselect All** (3, Fig. B.16).

Sobre estes, pode-se fazer a equalização histogrâmica (**Histogram**), a expansão linear (**Linear**), a imagem negativa (**Negative**), suavisação (**Smooth**) e a normalização relativa à uma média especificada (**Normalize**) (5, 6, 7, 8, 9, Fig B.16).

Pelo botão **Reset** (4, Fig. B.16) as características originais das canaletas selecionadas são reestabelecidas.

As modificações feitas não são salvas no banco de dados, servindo apenas para a análise por alinhamento de perfis corrente.

O acionamento da opção **Get Report** (10, Fig. B.16) faz com que seja um enviado um relatório ao usuário contendo os perfis assim como exibidos na tela.

Feitos os devidos ajustes, a paleta *Compare/Alignment/Alignment* disponibiliza funções para comparação por alinhamento de perfis. Vários parâmetros podem ser ajustados para tal (8, Fig. B.17).

A opção **Compare** (1, Fig. B.17) só é habilitada quando existem exatamente duas canaletas selecionadas. Quando acionada é feito o alinhamento de perfis com os parâmetros indicados. O resultado do alinhamento é exibido ao lado (4, Fig. B.17), mostrando o caminhamento na matriz de edição (9, Fig. B.17) e o resultado do alinhamento em si (7, Fig. B.17), bem como o valor da distância entre os perfis (11, Fig. B.17).

A opção **Sort** (2, Fig. B.17) fica disponível no caso de haver uma canaleta mestre e mais de um perfil selecionado na lista. É gerado um relatório contendo a canaleta mestre

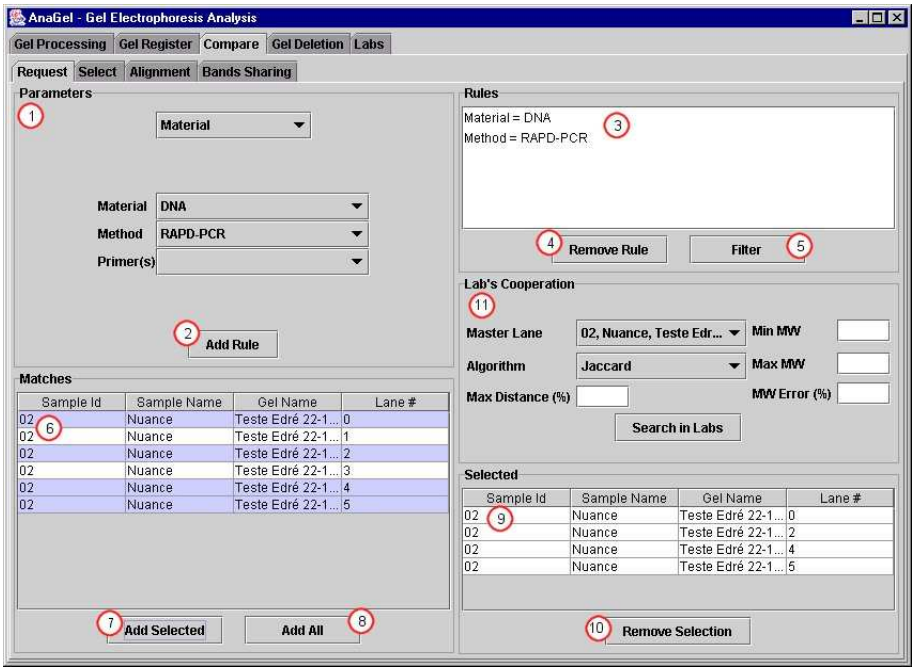

Figura B.14: Requisição de amostras ao banco de dados.

(1, Fig. B.19) e as demais canaletas (3, Fig. B.19), ordenadas pela distância de cada uma à mestre.

Clicando-se no campo que indica o nome do gel (2, Fig. B.19), é exibida uma janela contendo as informações sobre este (Fig. B.18).

A opção **Distances** (3, Fig. B.17) só é habilitada quando não existe canaleta selecionada como mestre e são marcadas mais de duas canaletas na lista. As canaletas são comparadas entre si, gerando uma matriz de distâncias (1, Fig. B.20). O formato exibido é o utilizado pelo software Philip para análise taxonômica [13], onde a primeira coluna indica o identificador da amostra de cada canaleta e as demais as distância encontradas.

Todos os relatórios gerados podem ser adquiridos por e-mail pelo acionamento do botão **Get Report** ((10, Fig. B.17).

Comparação por compartilhamento de bandas

Pela paleta *Compare/Bands Sharing* pode-se fazer a comparação entre as canaletas por compartilhamento de bandas.

O campo **MW Error(%)** (6, Fig. B.21) indica o erro percentual tolerado para que bandas sejam consideradas como compartilhadas.

A métrica a utilizada pode ser escolhida dentre as disponibilizadas pelo sistema (**Standard**) ou implementadas pelo usuário (**Other**) (7, 8, Fig. B.21).

A funcionalidade das opções **Compare** , **Sort** e **Distances** é a mesma descrita para

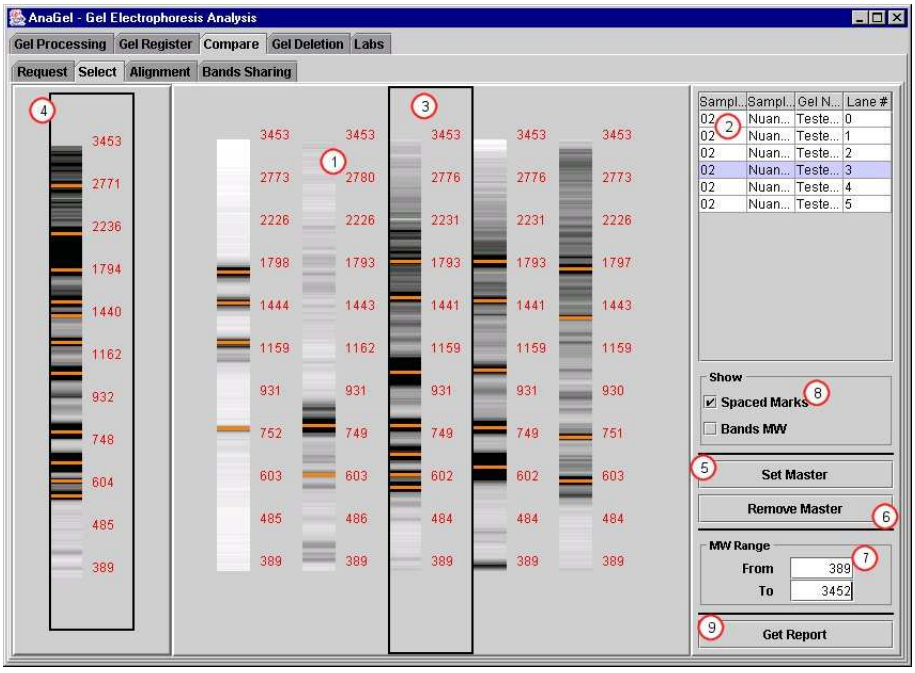

Figura B.15: Seleção das amostras a serem tratadas.

para alinhamento de perfis, porém utilizando a métrica escolhida.

Quando utilizada a opção **Compare** , o relatório gerado exibe as bandas compartilhadas (2, Fig. B.21), informações sobre as amostras (3, 4, Fig. B.21) e a distância entre elas, com respeito à métrica utilizada (5, Fig. B.21)

Assim como para os demais relatórios que apresentam recursos gráficos, clicando-se no campo relativo ao nome do gel (9, Fig. B.21) exibe uma janela contendo informações sobre o gel (Fig. B.18).

Aquisição dos relatórios

Os dados exibidos como resultados da comparação, tanto por alinhamento de perfis quanto por compartilhamento de bandas, podem ser adquiridos como uma imagem *gif* (Fig. B.22), para aquelas opções que geram informações gráficas, ou como arquivo texto, para as informações geradas como matrizes.

Acionando-se **Get Report** (10, Fig. B.17) o relatório é enviado para o e-mail cadastrado para o usuário.

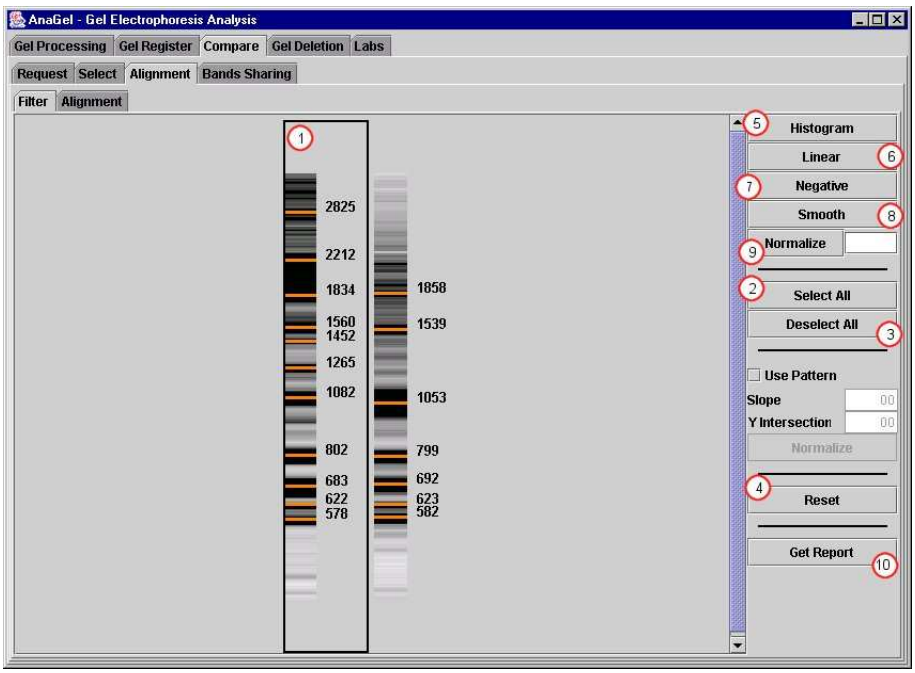

Figura B.16: Ajustes do perfil para alinhamento.

B.3.6 Fornecimento de algoritmos pelo usuário

É permitido ao usuário a utilização de seus próprios algoritmos de comparação de canaletas. É necessário, para tanto, possuir conhecimento da linguagem Java e adquirir o arquivo *anagel.jar*, que disponibiliza os recursos necessários para tal.

Para que seja possível a utilização de algoritmos não integrantes do sistema, é necessário que algumas regras sejam seguidas.

Os algoritmos devem ser implementados utilizando-se a linguagem Java, uma vez que o sistema foi totalmente desenvolvido utilizando-se de recursos da mesma e esta provê meios de carga dinâmica de classe de uma forma direta.

Uma classe para comparações deve ser descendente da classe abstrata `CompareSamples` do pacote `ufmg.dcc.anagel.compare`, onde a rotina de comparação deve ser o corpo do método `public float compare(Lane lane1, Lane lane2)`. O usuário deve colocá-la no pacote `users.`*login*`.compare`, sendo *login* o nome do usuário cadastrado para acesso ao sistema.

Assim, para o usuário *antonio*, um exemplo de implementação seria

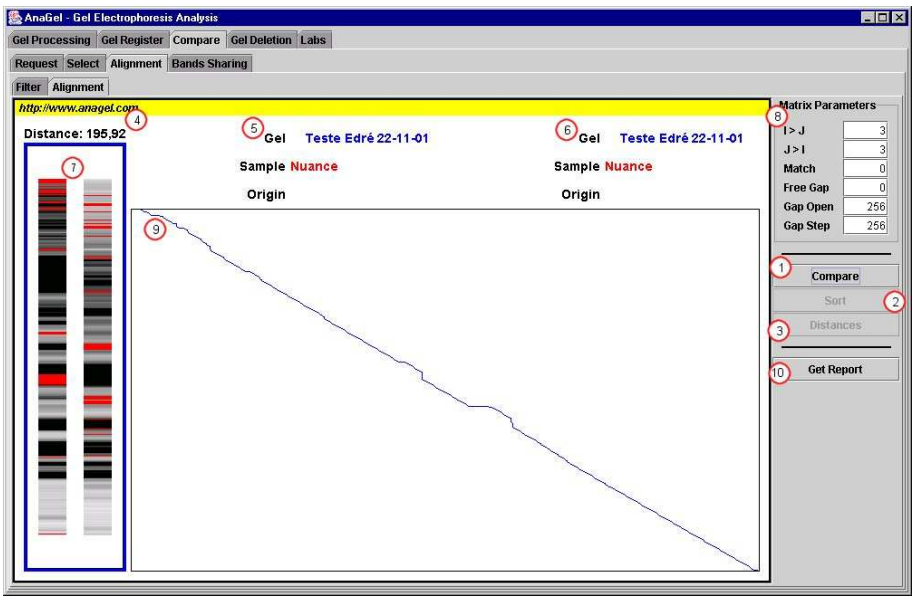

Figura B.17: Relatório para comparação por alinhamento de perfis, quando acionada a opção **Compare** .

```
package users.antonio.compare;
import ufmg.dcc.anagel.compare.*;

public class MyCompare extends CompareSamples {

   public float compare(Lane lane1, Lane lane2) {
      // Corpo do m\'{e}todo
      ...
   }

   public String toString() {
      return "MyCompare";
   }

}
```

É interessante, embora não necessário, que seja implementado o método `public String toString()`, para que este retorne o nome da classe ou algum *string* que possibilite sua identificação.

Esta classe deve ser fornecida sem ser compilada através da página para fornecimento

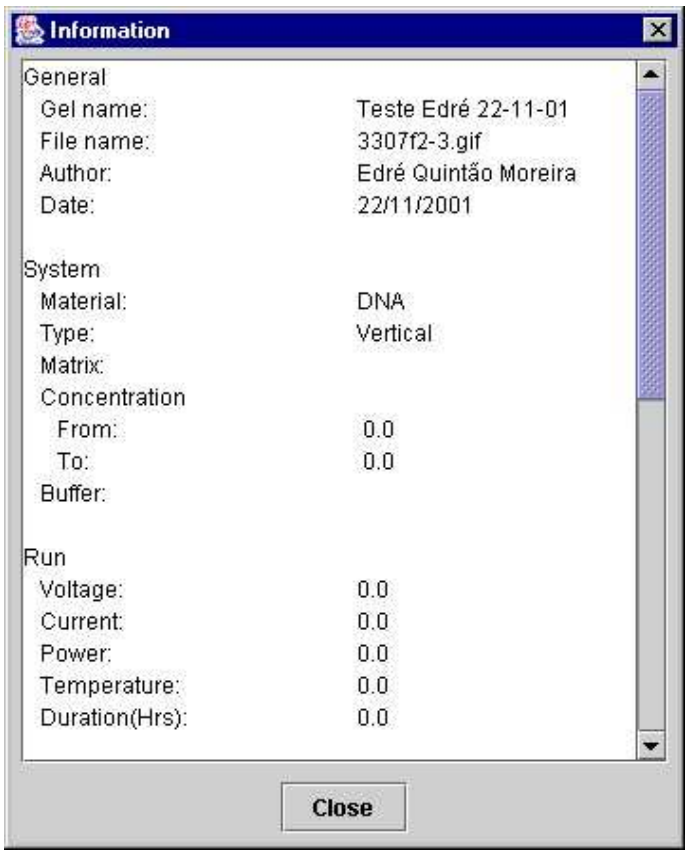

Figura B.18: Informações cadastradas sobre o gel.

de imagens (Fig. B.2), bastando selecionar a opção **Comparison algorithm** e selecionar o arquivo fonte criado.

A descrição das classes pode ser encontrada a partir do *site http://anagel.cenapad.ufmg.br/anagel/doc/index.html*.

B.3.7 Cooperação entre laboratórios

Na paleta **Labs** o usuário pode escolher os laboratórios nos quais deseja executar seu algoritmos (Fig. B.23). Todos os laboratórios nos quais os servidores AnaGel estão ativados são listados.

No momento da requisição de recuperação de canaletas em laboratórios remotos (Seção B.3.5) são consultados apenas naqueles selecionados na lista (1, Fig. B.23).

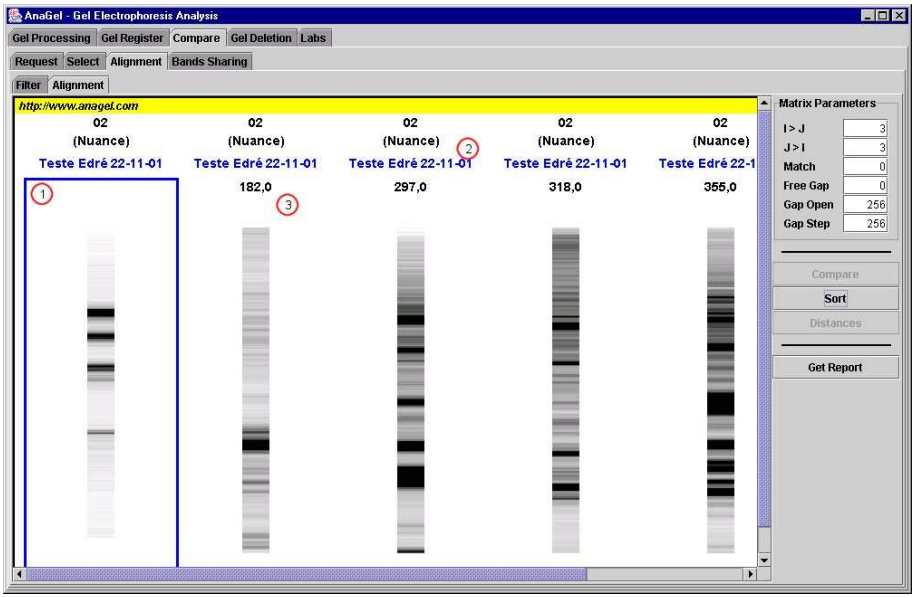

Figura B.19: Relatório criado a partir da utilização da opção **Sort** .

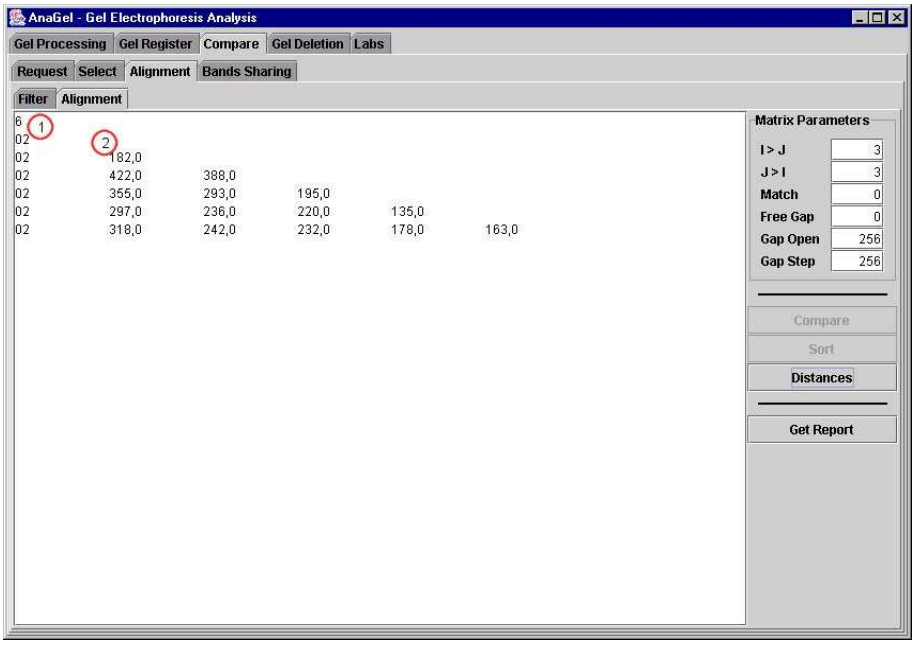

Figura B.20: Matriz de distâncias criada pelo acionamento da opção **Distances** .

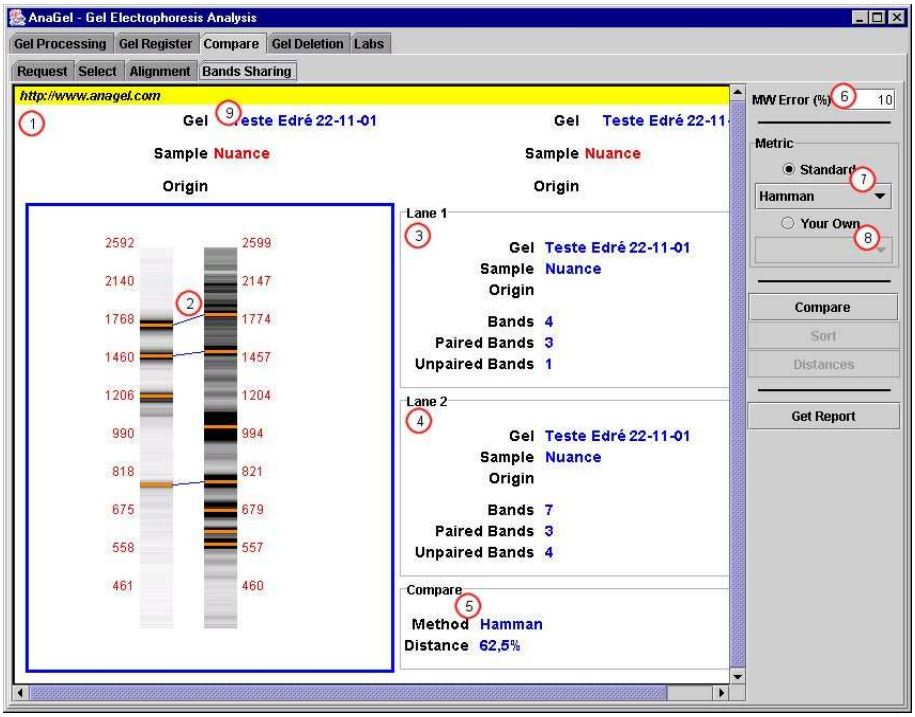

Figura B.21: Resultado gerado quando acionado o botão **Compare** para compartilhamento de bandas.

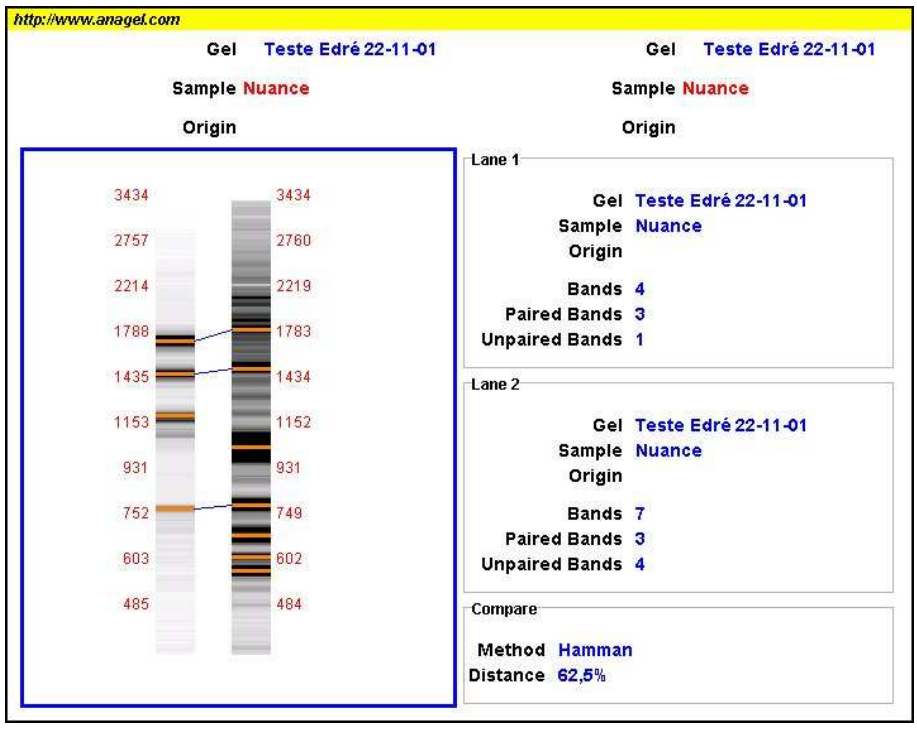

Figura B.22: Arquivo *gif* enviado por e-mail para o usuário.

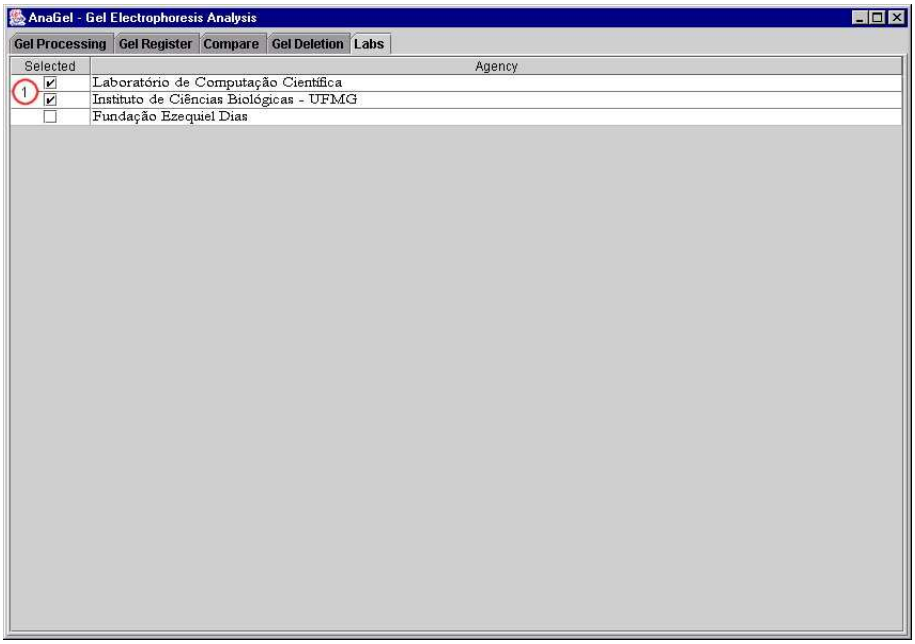

Figura B.23: Lista de laboratórios com servidores *AnaGel* ligados .

Apêndice C

Ferramentas para Análise de Géis de Eletroforese

Existem diversas ferramentas para análise de géis que podem ser encontradas no mercado. Dentre elas, destacam-se *Phoretix*, *TotalLab*, *Gel-Pro*, *GelSite*, *GeneTools* e *SigmaGel*.

C.1 Phoretix

O *Phoretix 1D Advanced*[1] (Figura C.1) da empresa *Nonlinear Dynamics* é um programa para análise de registros eletroforéticos desenvolvido para ser o mais flexível possível e também de fácil utilização. Possui uma interface gráfica intuitiva, com diversos ícones para facilitar o acesso às operações.

Dentre as principais características deste *software* pode-se destacar:

- Execução nos ambientes *Windows 95, 98, NT* e *2000*;
- Detecção automática de canaletas e bandas;
- Janelas para acompanhamento das análises;
- Caixas de texto para documentação em cima da imagem;
- Correção de distorções em géis usando canaletas e linhas de referência flexíveis;
- Determinação de quantidade de massa e peso molecular de bandas;
- Cálculo de histogramas;

[1] http://www.phoretix.com

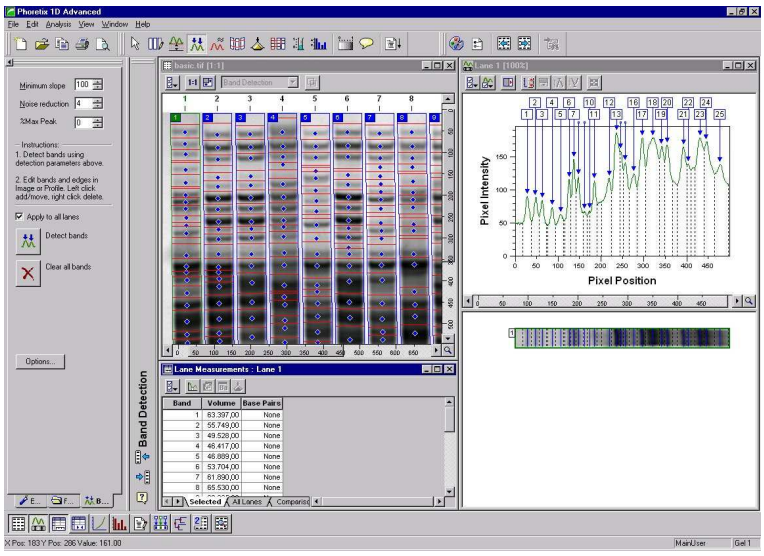

Figura C.1: Phoretix 1D Advanced.

- Construção de dendrogramas;
- Cálculo de pontos isoelétricos (pI's);
- Diversas técnicas de normalização e calibração;
- Compartilhamento de bandas;
- Armazenamento local dos dados;
- Exportação de dados para o *Microsoft Excel* e outros aplicativos para *Windows*.

C.2 TotalLab

A ferramenta *ImageMaster TotalLab 1D* [2] (Figura C.2) , também desenvolvido pela empresa *Nonlinear Dynamics*, possui uma interface gráfica semelhante ao *software Phoretix*. Com exceção de construção de dendrogramas e compartilhamento de bandas, suas características são as mesmas citadas na Seção C.1.

C.3 Gel-Pro

Gel-Pro Analyzer [3] (Figura C.3) é um *software* para análise de géis de eletroforese unidimensional para computadores pessoais. É capaz de adquirir e analisar dados de câ-

[2] http://www.amershambiosciences.com
[3] http://www.mediacy.com/gppage.htm

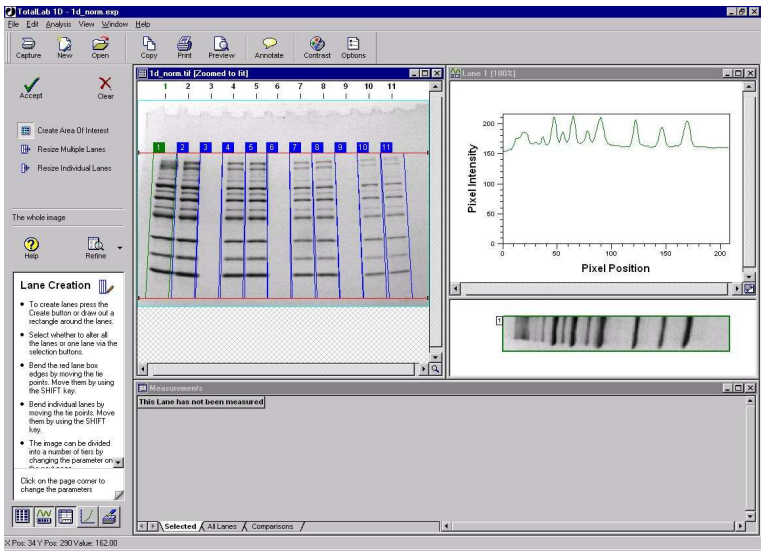

Figura C.2: ImageMaster TotalLab.

meras e *scanners* ou ler arquivos de imagens armazenadas e operar simplesmente como ferramenta de análise.

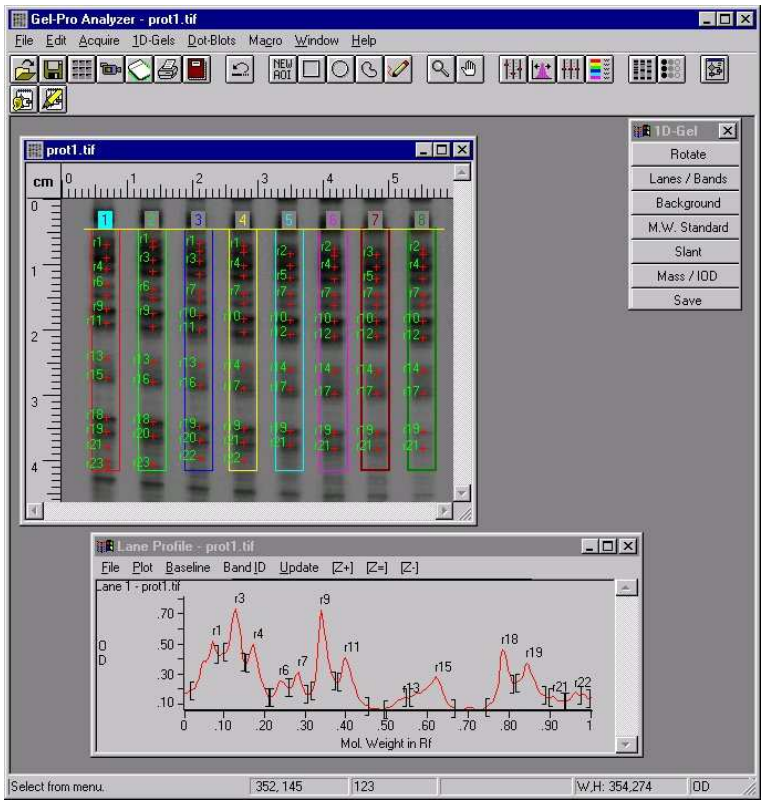

Figura C.3: Gel-Pro Analyser.

As principais características deste produto são:

- Execução nos ambientes *Windows 95, 98, NT* e *2000*;

- Possibilidade de personalização de macros escritas em *Visual Basic* para processamento da imagem (estimativa de pesos moleculares, detecção de canaletas e bandas, etc.);
- Armazenamento de imagens e anotações em um banco de dados local;
- Reconhecimento automático de canaletas, bandas, e fundo da imagem;
- Determinação da área das bandas;
- Cálculo automático da massa em cada banda da canaleta;
- Correção de efeito sorriso;
- Determinação de pesos moleculares com múltiplas canaletas padrões;
- Utilização de listas padrões de pesos moleculares para DNA, RNA ou proteína;
- Criação de listas padrões de pesos moleculares personalizadas;
- Disposição das informações sobre as bandas em linhas de igual mobilidade para melhor comparação entre canaletas;
- Geração de relatórios em formatos tabular e gráfico;
- Exportação de dados para o *Microsoft Excel* e outros aplicativos para *Windows*;
- Capacidade de importação e exportação de arquivos em vários formatos;
- Recuperação de dados pelo título, tipo, data, hora, nome do arquivo e comentários do experimento;
- Ferramentas para documentação em cima da imagem, como caixas de texto, linhas, círculos, etc.

C.4 GelSite

Assim como os *softwares* descritos anteriormente, *GelSite* [4] (Figura C.4) é uma ferramenta para análise de géis de eletroforese, possuindo como caracteristicas principais:

- Execução nos ambientes *Windows 95, 98, NT* e *2000*;
- Localização automática de bandas baseado na diferença de tons de cinza da imagem;

[4] http://www.nucleicassays.com/gelsite

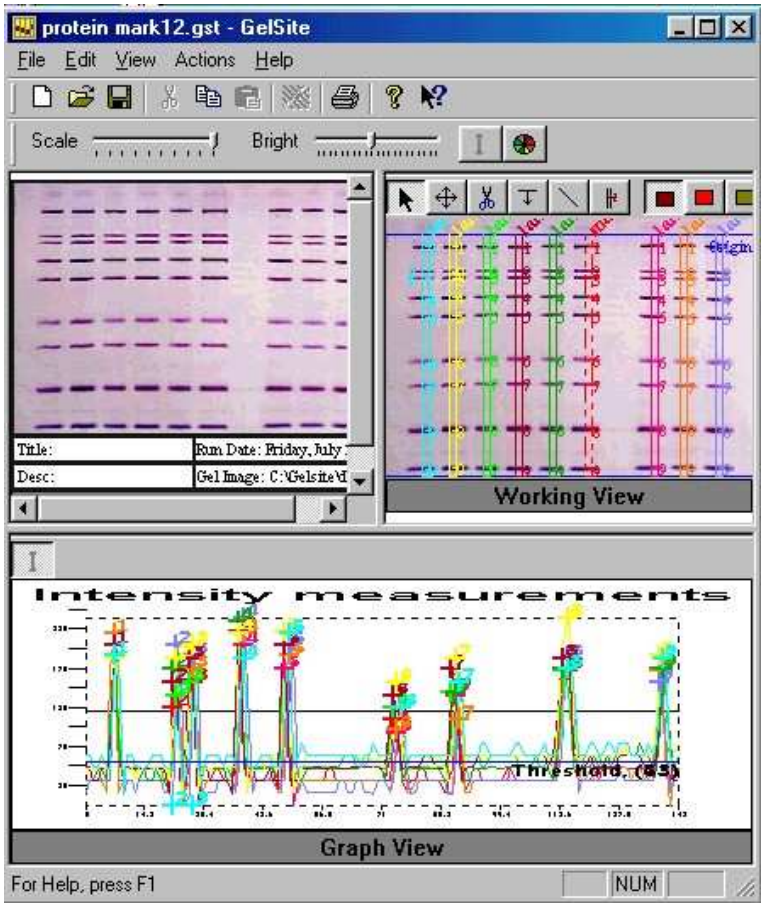

Figura C.4: GelSite.

- Interação com o *Windows*, permitindo operações de copiar e colar;
- Cálculo automático do peso molecular, tamanho do DNA em pares de bases, e concentração da solução;
- Seleção de múltiplas canaletas de calibração;
- Ajuste independente de cada canaleta para compensação de variações térmicas durante a corrida;
- Possibilidade de redefinição de bandas pelo usuário;
- Armazenamento dos dados em um banco local.

C.5 GeneTools

O *software* de análise *GeneTools* [5] (Figura C.5) desenvolvido pela *Syngene* é uma ferramenta automática, de fácil uso e muito rápida. Sua interface é muito intuitiva, facilitando

[5] http://www.syngene.com/genetools.asp

sua utilização por usuários que não possuem muita familiaridade com computadores.

Dentre as principais características no que se refere à análise de géis unidimensionais destacam-se:

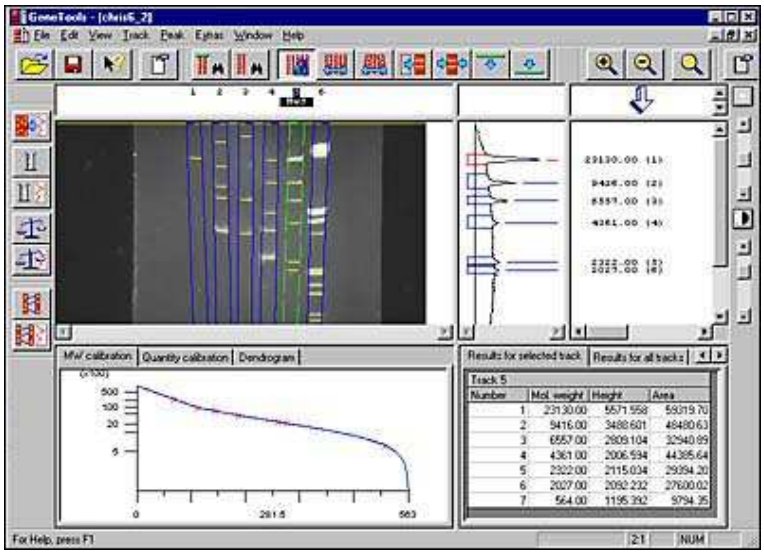

Figura C.5: GeneTools.

- Execução nos ambientes *Windows 95, 98, NT* e *2000*;
- Armazenamento dos dados em um banco local.
- Cálculo automático da densitometria, peso molecular e quantidade de massa das bandas;
- Todas as funcionalidades estão em um mesmo pacote, dispensando *softwares* adicionais;
- Análise de imagens em diversos formatos (*jpg*, *gif*, *bmp*, etc.);
- Demarcação automática de canaletas e bandas com detecção de efeito sorriso;
- Adição e remoção manual de bandas;
- Exportação de resultados diretamente para o *Microsoft Excel*;
- Personalização de configurações por usuário;
- Impressão em todas impressoras padrões;
- Exibição de imagens, resultados e histogramas em uma única janela;
- Visualização de curvas de calibração de pesos moleculares;

- Biblioteca de padrões de pesos moleculares para assinalamento automático ou manual, configurável pelo usuário;
- Uso ilimitado de marcações de calibração por gel;
- Interpolação automática de múltiplas marcações de calibração para correção de efeito sorriso ou distorções;
- A versão completa disponibiliza ferramentas para cálculo de compartilhamento de bandas;
- Dendrogramas e análise de *clusters*;
- Comparação de presença e ausência de bandas com geração da matriz;
- Cálculo de coeficientes de similaridade;
- Geração de relatórios dos experimentos.

C.6 SigmaGel

SigmaGel[6] (Figura C.6) é uma ferramenta simples desenvolvida pela *Jandel Scientific* para análise de géis, sendo uma alternativa de baixo custo em comparação aos demais *softwares*.

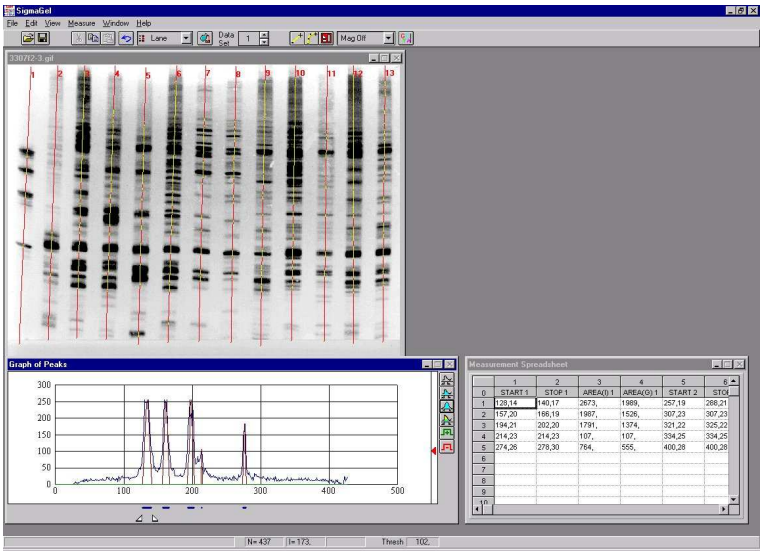

Figura C.6: SigmaGel.

As funcionalidades que mais se sobressaem nesta ferramenta são:

[6]http://www.spssscience.com/SigmaGel/index.cfm

- Execução nos ambientes *Windows 3.1x, 95, 98, NT* e *2000*;
- Interface de fácil uso pela presença de ícones, teclas de atalho, exibição de dicas sobre as funcionalidades e ajuda sensível ao contexto;
- Tratamento de imagens nos formatos *tiff*, *jpeg*, *pcx*, *eps* e *bmp*;
- Suporte a *scanners* compatíveis com TWAIN;
- Integração com o *Windows* para captura de imagens;
- Salvamento de dados em arquivos texto;
- Impressão de imagens, gráficos de medidas e planilhas;
- Demarcação manual de canaletas;
- Detecção automática de bandas;
- Inserção e remoção manual de bandas;
- Geração de densitogramas;
- Marcação de várias curvas de calibração;
- Cálculo da área dos picos;
- Cálculo dos pesos moleculares das bandas;
- Geração de planilha com diversas informações, como pesos moleculares, área, posição das bandas, etc.

Referências Bibliográficas

[1] Grasshopper – An Intelligent Mobile Agent Platform. White Paper.

[2] Promega protein guide: Tips and techniques, 1983. Promega Corporation.

[3] *Getting Started with Informix Dynamic Server*, February 1998. Version 7.3.

[4] MASIF - Mobile Agent System Interoperability Facilities Specification. ftp://ftp.omg.org/pub/docs/orbos/98-03-09.pdf, March 1998. Standard proposal.

[5] Agent technology — green paper. OMG Document agent/00-09-0. http://www.objs.com/agent/agents_Green_Paper_v100.doc, September 2000.

[6] S. B. ABDERRAZAK, F. GUERRINI, F. MATHIEU-DAUDÉ, P. TRUC, K. NEUBAUER, K. LEWICKA, C. BARNABÉ, and M. TIBAYRENC. *Isoenzyme Electrophoresis for Parasite Characterization. In Methods in Molecular Biology*, volume 21: Protocols in Molecular Parasitology. Humana Press, 1993.

[7] Dennis A. BENSON, Ilene KARSCH-MIZRACHI, David J. LIPMAN, James OSTELL, Barbara A. RAPP, and David L. WHEELER. Genbank. In *Nucleic Acids Research*, volume 28. Oxford University Press, 2000.

[8] Helen M. BERMAN, John WESTBROOK, Zukang FENG, Gary GILLIAND, T. N. BHAT, Helge WEISSIG, Ilya N. SHINDYALOV, and Philip E. BOURNE. The protein data bank. In *Nucleic Acids Research*, volume 28. Oxford University Press, 2000.

[9] Andrew D. BIRRELL and Bruce Jay NELSON. Implementing remote procedure calls. *ACM Transactions on Computer Systems*, 2(1):39–59, February 1984.

[10] Mary CAMPIONE, Kathy WALRATH, and Alison HULM. *The Java tutorial continued: the rest of the JDK*. Addison-Wesley, Reading, MA, USA, 1999.

[11] Mary CAMPIONE, Kathy WALRATH, and Alison HULM. *The Java Tutorial: A Short Course on the Basics*. Addison-Wesley, 3rd edition, 2000.

[12] R. J. CYPSER. *Communications for Cooperating Systems.* Addison Wesley, 1991.

[13] J. FELSENSTEIN. Phylip (phylogeny inference package) version 3.5., 1993. Distribuído pelo autor.

[14] Thomas A. FINHOLT. Collaboratories. Technical report, School of Information, University of Michigan, 2001. Collaboratory for Research on Electronic Work.

[15] K. GOODERHAM. *High-Sensitive Silver Stainig of Proteins Following Polyacrilamide Gel Electrophoresis. In Methods in Molecular Biology*, volume 1: Proteins. Humana Press, 1984.

[16] D. GRIERSON. *Gel Electrophoresis of RNA. In Gel Electrophoresis of Nucleic Acids: A Practical Approach.* IRL Press Limited, 1982.

[17] J. C. GROWER. Measures of similarity, dissimilarity and distance. In S. Klotz e N. L. Johnson, editor, *Encyclopedia of Statistical Sciences*, volume 5, pages 397–405. Wiley, 1985.

[18] B. D. HAMES. *An Introduction to Polyacrilamide Gel Electrophoresis. In Gel Electrophoresis of Proteins: A Practical Approach.* IRL Press Limited, 1981.

[19] Colin G. HARRISON, David M. CHEES, and Aaron KERSHENBAUM. Mobile Agents: Are they a good idea? Technical report, IBM, T. J. Watson Research Center, Yorktown Heights, New York, March 1995.

[20] A. R. HOELZEL and G. A. DOVER. *Molecular Genetic Ecology.* Focus Series. IRL Press, 1991.

[21] IKV++. *Grasshopper Programmer's Guide - Release 2.2*, 2001.

[22] D. A. JACKSON, K. M. SOMERS, and H. H. HARVEY. Similarity coefficients: Measures of co-occurrence and association or simply measures of occurrence? *The American Naturalist*, 133(3):436–453, 1989.

[23] Danny B. LANGE and Mitsuru OSHIMA. Mobile agents with java: The aglet API. *World Wide Web Journal*, 1998.

[24] Danny B. LANGE and Mitsuru OSHIMA. Seven good reasons for mobile agents. *Communications of the ACM*, 42(3):88–89, March 1999.

[25] Rosanna LEE and Scott SELIGMAN. *JNDI API Tutorial and Reference: Building Directory-Enabled Java Applications.* The Java Series. Addison-Wesley, 2000.

[26] Barry M. LEINER, Vinton G. CERF, David D. CLARK, Robert E. KAHN, Leonard KLEINROCK, Daniel C. LYNCH, Jon POSTEL, Lawrence G. ROBERTS, and Stephen S. WOLFF. The past and future history of the internet. *Communications of the ACM*, 40(2):102–108, February 1997.

[27] T. LINDHOLM and F. YELLIN. *The Java Virtual Machine Specification*. Addison-Wesley, Reading, USA, 1997.

[28] A. MACHADO, M. CAMPOS, A. SIQUEIRA, and O. CARVALHO. An iterative algorithm for segmenting lanes in gel electrophoresis images. In *Proceedings of the XI SIBGRAPI*. IEEE Press, 1997.

[29] T. MANIATIS, E. F. FRITSCH, and J. SAMBROOK. *Molecular Cloning: A Laboratory Manual*. 1982.

[30] E. L. V. MAYES. *Determination of Protein Molecular Weights by Gel Permeation High Pressure Liquid Chromatography. In Methods in Molecular Biology*, volume 1: Proteins. Humana Press, 1984.

[31] Gary M. OLSON, Daniel E. ATKINS, Robert CLAUER, Thomas A. FINHOLT, Farnam JAHANIAN, Timothy L. KILLEN, Atul PRAKASH, and Terry WEYMOUTH. The upper atmospheric research collaboratory (UARC). *interactions*, 5(3):48–55, 1998.

[32] Robert ORFALI and Dan HARKEY. *Client/Server Programming with Java and CORBA*. John Wiley and Sons, New York, NY, USA; London, UK; Sydney, Australia, second edition, January 1998.

[33] Maria Luiza Assunção PIMENTA. *Anagel - Armazenamento e Análise de Registros Eletroforéticos*, 1996.

[34] Maria Luiza Assunção PIMENTA. Anagel: Um sistema de análise de registros eletroforéticos. Master's thesis, Universidade Federal de Minas Gerais - Departamento de Ciência da Computação, 1996.

[35] P. G. SEALEY and E. M. SOUTHERN. *Electrophoresis of DNA. In Gel Electrophoresis of Nucleic Acids: A Practical Approach*. IRL Press Limited, 1982.

[36] P. H. A. SNEATH and R. R. SOKAL. *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. Freeman, 1973.

[37] William STALLINGS. *Cryptography and network security: principles and practice*. Prentice-Hall, Inc., Upper Saddle River, NJ 07458, USA, 2nd edition, 1999.

[38] Stephanie TEASLEY and Steven WOLINSKY. COMMUNICATION: Scientific collaborations at a distance. In *Science Magazine*, volume 292, pages 2254–2255, June 2001.

[39] J. M WALKER, editor. *Methods in Molecular Biology*, volume 1: Proteins. Humana Press, 1984.

[40] J. WELSH and M. MCCLELLAND. Fingerprinting genomes using PCR with arbitrary primers. *Nucleic Acids Research*, 18(24):7213–7218, 1990.

[41] David WONG, Noemi PACIOREK, and Dana MOORE. Java-based mobile agents. *Communications of the ACM*, 42(3):92–102, February 1999.