SIGLA: UM LIMS BASEADO EM WORKFLOWS ADAPTÁVEIS COM SUPORTE A MÚLTIPLOS LABORATÓRIOS

ALEXANDRE SIMÕES DE MELO

SIGLA: UM LIMS BASEADO EM WORKFLOWS ADAPTÁVEIS COM SUPORTE A MÚLTIPLOS LABORATÓRIOS

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

Orientador: Sérgio Vale Aguiar Campos

Belo Horizonte

Fevereiro de 2010

Simões de Melo, Alexandre
D1234p SIGLa: Um LIMS baseado em workflows adaptáveis com suporte a múltiplos laboratórios / Alexandre Simões de Melo. — Belo Horizonte, 2010
xx, 97 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Sérgio Vale Aguiar Campos

1. LIMS. 2. Bioinformática. 3. fluxo de trabalho Trabalho. I. Tï£¡tulo.

CDU 519.6*82.10

[Folha de Aprovação]

Quando a secretaria do Curso fornecer esta folha,
ela deve ser digitalizada e armazenada no disco em formato gráfico.

Se você estiver usando o `pdflatex`,
armazene o arquivo preferencialmente em formato PNG
(o formato JPEG é pior neste caso).

Se você estiver usando o `latex` (não o `pdflatex`),
terá que converter o arquivo gráfico para o formato EPS.

Em seguida, acrescente a opção `approval={`*nome do arquivo*`}`
ao comando `\ppgccufmg`.

"Não há rosas sem espinhos, é um provérbio melancólico. Digamos em vez disso: Não há espinhos sem rosas."

(Chiara Lubich)

Resumo

A necessidade de gerenciar grandes quantidades de dados é uma exigência para os laboratórios. O uso de Sistemas de Gerenciamento de informação de Laboratórios (LIMS) para alcançar este objetivo cresce a cada dia. Um LIMS é um complexo sistema computacional de gerência de dados de laboratório, que enfatiza a garantia da qualidade. Vários LIMS estão disponíveis atualmente. No entanto, a maioria deles possui código fechado e são comercializados por um custo elevado. Além disso, devido à sua complexidade, os LIMS são geralmente concebidos para atender às necessidades de um tipo específico de laboratório, tornando muito difícil a reutilização do LIMS. Neste trabalho foi implementado o Sistema Integrado de Gerência de Laboratórios (SIGLa), um LIMS com código aberto e com uma nova abordagem destinada a permitir-lhe adaptar as suas atividades e processos para os diversos tipos de laboratórios. Para isso o SIGLa possui um sistema de gerenciamento de fluxos de trabalho integrado ao seu código, tornando possível criar e gerenciar fluxos de trabalho personalizados. Para cada novo laboratório um fluxo de trabalho é definido com suas atividades, regras e procedimentos. Durante a execução, para cada fluxo de trabalho criado, os valores dos atributos definidos em um arquivo XPDL (que descrevem o fluxo de trabalho) são armazenados no sistema de banco de dados do SIGLa, permitindo que sejam gerenciados e recuperados quando necessário. Para o desenvolvimento do SIGLA foi utilizado como primeiro modelo o laboratório de análise proteômica. Para verificar a capacidade do SIGLa de se adaptar a múltiplos laboratórios, foi definido um segundo fluxo de trabalho modelando as atividades de um laboratório de análise de microarranjos. Este fluxo de trabalho foi construído em poucos dias, ilustrando a flexibilidade e eficiência do método proposto. Com o desenvolvimento do SIGLa esperamos contribuir com o gerenciamento de dados complexos em laboratórios, garantindo a consistência dos dados e aumentando a produtividade do laboratório. Esperamos, também, possibilitar que laboratórios com poucos recursos financeiros possam ter acesso a um sistema de alto nível para o gerenciamento de dados.

Palavras-chave: LIMS, Fluxo de trabalho, Bioinformática.

Abstract

The need to manage large amounts of data is a demand for laboratories. The use of Laboratory Information Management Systems (LIMS) to achieve this is increasing continuously. A LIMS is a complex computational system used to manage laboratory data with emphasis in quality assurance. Several LIMS are available currently. However, most of them have proprietary code and are commercialized at a high cost. Moreover, due to its complexity, LIMS are usually designed to comply with the needs of one kind of laboratory, making it very difficult to reuse a LIMS. In this work we describe the *Sistema Integrado de Gerência de Laboratórios* (SIGLa), an open source LIMS with a new approach designed to allow it to adapt its activities and processes to various types of laboratories. To accomplish this SIGLa incorporates a workflow management system, making it possible to create and manage customized workflows. For each new laboratory a workflow is defined with its activities, rules and procedures. During the execution, for each workflow created, the values of attributes defined in a XPDL file (which describes the workflow) are stored in SIGLa's database, allowing them to be managed and retrieved upon request. The proteomic laboratory was used as first model for SIGLa development. To check the SIGLa capability of adapting to multiple laboratories, we defined a second workflow modeling a microarray laboratory. This workflow was constructed in a few days illustrating the flexibility and power of the method proposed. With SIGLa's development we hope to contribute positively to the area of management of complex data in laboratory by managing its large amounts of data, guaranteeing the consistence of the data and increasing the laboratory productivity. We also hope to make possible to laboratories with little resources to afford a high level system for complex data management.

Keywords: LIMS, Workflow, Bioinformatics.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Definição do Problema

Os avanços tecnológicos presentes na pesquisa biomédica resultaram em uma grande quantidade de dados sendo gerados por laboratórios de pesquisa, de testes e comerciais. Segundo Hinton [1995] isto se dá pelos seguintes motivos:

- Crescimento de regulamentações governamentais adicionando requisitos nos processos dos laboratórios.
- Avanço na instrumentação automática, o que permite uma coleta de dados muito mais rápida.
- Maior exigência nos processos de controle de qualidade, aumentando a quantidade de testes.
- Crescimento da demanda por análise estatística dos dados, que por sua vez aumenta a demanda por coleta de dados, armazenamento, recuperação, relatórios e arquivamento.

Com esta grande quantidade de dados, torna-se muito difícil controlar a qualidade dos processos e resultados gerados. Ainda é muito comum, principalmente em pequenos laboratórios de pesquisa, o uso de cadernos para o gerenciamento das atividades. Claramente esta solução apresentas inúmeras desvantagens como baixa produtividade, falta de segurança dos dados e alta ineficiência na busca de informação. Outros laboratórios buscam algum tipo de solução computacional para o gerenciamento dos seus dados, contudo, normalmente estes utilizam simples sistemas de cadastro genérico que não possuem mecanismo que visam melhorar a qualidade dos dados e que não oferecem

uma consulta de dados adequada. Existem também laboratórios que utilizam softwares que gerenciam a execução de apenas uma atividade específica. Este softwares podem auxiliar no gerenciamento dos dados do laboratório, contudo é um gerenciamento descentralizado, onde cada software gerencia uma atividade distinta. A desvantagem da descentralização é que os usuários devem aprender a manipular diversos sistemas complexos, além disso, neste caso não é possível gerar relatórios que retratem o processo do laboratório como um todo, para isto é necessário gerar um relatório manual, que por ser manual é altamente suceptível a erros.

1.2 Motivação

A fim de abordar estas questões, o conceito de Sistemas de Gerenciamento de Informação de Laboratórios (LIMS) foi desenvolvido. Um LIMS é um complexo sistema computacional utilizado por um laboratório para gerenciar seus dados com ênfase na melhoria da qualidade dos dados, auxiliando no processo do laboratório e buscando gerar resultados muito mais consistentes e confiáveis. Diversos LIMS estão disponíveis atualmente. Contudo a grande maioria possui código fechado e é comercializado por um custo elevado, dificultando seu uso por estudantes e pequenos laboratórios. Além disso, as atividades e procedimentos executados nos laboratórios são, em geral, bastante diferentes entre laboratórios distintos, sendo assim, para cada tipo de laboratório deve existir um LIMS específico, o que torna seu desenvolvimento ainda mais caro. Atualmente a maioria dos laboratórios ainda não possuem um LIMS que se adeque às suas atividades.

1.3 Contribuição

Diante deste cenário, neste trabalho foi implementado o Sistema Integrado de Gerência de Laboratórios (SIGLa). Trata-se de um LIMS com código aberto e com uma nova abordagem que permite adaptar o seu comportamento para diferentes tipos de laboratórios. Para isto foi usado um sistema de gerenciamento de fluxos de trabalho incorporado ao seu código. Fluxo de trabalho consiste na automação de um processo de negócios, durante o qual documentos, informações ou tarefas são passadas de um participante para outro por ações, de acordo com um conjunto de regras procedurais (WfMC [2009]). Um sistema de gerenciamento de fluxos de trabalho permite a definição e controle das atividades presentes no fluxo de trabalho. No SIGLa, para cada novo laboratório um novo fluxo de trabalho é definido com suas atividades, regras e

procedimentos. Depois disso, um arquivo com as definições do fluxo de trabalho será carregado no sistema, a fim de permitir que o SIGLa gerencie as atividades do laboratório. A definição de um novo fluxo de trabalho é bastante simples e pode ser executado por um usuário que conhece todos os procedimentos do laboratório, mas sem nenhum conhecimento de programação prévia. SIGLa é uma aplicação Web escrita em Java com o gerenciador de fluxo de trabalho Enhydra Shark incorporado ao seu código. Um banco de dados MySQL é usado para armazenar os dados dos laboratórios. Para definir um fluxo de trabalho, usamos o *Together Workflow Editor*. O fluxo de trabalho definido na primeira versão do SIGLa descreve as atividades de um laboratório de proteômica. Este laboratório foi escolhido por manipular grandes quantidades de dados que possuem uma complexa relação entre eles. Usando este tipo de laboratório como exemplo para o desenvolvimento, buscamos mostrar como o SIGLa pode ser usado para gerenciar experimentos reais e complexos. Para verificar a capacidade do SIGLa de adaptar-se a múltiplos laboratórios, foi desenvolvido um segundo fluxo de trabalho para gerenciar as atividades de um laboratório de microarranjos, este fluxo de trabalho foi construído em poucos dias, ilustrando a flexibilidade e eficiência do método proposto. O SIGLa está disponível em `http://luar.dcc.ufmg.br/sigla` no link SIGLa, e pode ser acessado com o login *guest* e senha *guest*. Nesta versão os fluxos de trabalho de proteômica e microarray estão disponíveis. Com o desenvolvimento do SIGLa esperamos contribuir positivamente para a área de gestão de dados complexos em laboratório.

1.4 Organização do texto

Para melhor compreensão do trabalho detalhamos aqui a sua estrutura. Este Capítulo 1 faz uma introdução onde foi apresentado o problema abordado, a motivação do trabalho e suas contribuições. O Capítulo 2 apresenta os trabalhos relacionados. Neste capítulo serão detalhados dois conceito fundamentais para a compreensão do trabalho. Primeiramente são detalhados os Sistemas de Gerenciamento de Informação de Laboratórios (LIMS). Além de gerenciar os dados do laboratório o SIGLa possui a capacidade de se adaptar a múltiplos laboratórios, para isso foi utilizado o conceito de fluxos de trabalho. Assim, o Capítulo 2 também detalha o conceito de fluxos de trabalho e de Sistemas de Gerenciamento de Fluxos de Trabalho. Por fim, no Capítulo 2, o presente trabalho é contextualizado com a apresentação dos LIMS existentes e de sistemas similares ao SIGLa. O Capítulo 3 aborda o projeto elaborado para o desenvolvimento do SIGLa, detalhando os requisitos do sistema e a sua análise. O Capítulo

4 apresenta o Sistema Integrado de Gerência de Laboratórios (SIGLa), mostrando o seu funcionamento, o processo de definição de fluxos de trabalho e a execução do fluxo de trabalho, ou seja, como é feito o gerenciamento dos dados. O Capítulo 5 aprofunda nos detalhes da implementação do SIGLa, apresentando as ferramentas utilizadas no seu desenvolvimento, sua estrutura de classes, suas funcionalidades e seu banco de dados. O Capítulo 6 apresenta os laboratórios de Proteômica e Microarranjo e descreve a definição dos seus fluxos de trabalho para serem utilizados no SIGLa. Por fim, o Capítulo 7 apresenta as conclusões do trabalho, relata o uso do SIGLa pelo laboratório de microarranjos da Universidade Federal de Minas Gerais, que já está utilizando o SIGLa com dados reais e apresenta algumas possibilidades que permitirão diversas extensões e melhorias no presente trabalho.

Capítulo 2

Trabalhos Relacionados

2.1 Sistemas de Gerenciamento de Informação de Laboratórios (LIMS)

Cada vez mais os laboratórios dependem de sistemas computacionais para o gerenciamento dos seus dados. Contudo, na maioria dos casos, são utilizados simples sistemas de cadastro de dados, e isto pode vir a ser um problema. A entrada de dados nestes sistemas é insegura, pois é bastante susceptível a erros. A consulta de dados não é uma tarefa fácil, além disso, os dados podem ser salvos em estados de transição, gerando inconsistências. É comum em laboratórios que possuem equipamentos de alta tecnologia o uso de softwares próprios de cada equipamento, estes softwares podem, inclusive, auxiliar no gerenciamento dos dados, contudo é um gerenciamento descentralizado, onde cada software gerencia uma atividade distinta. Contudo, é muito importante para os laboratórios uma análise global dos seus dados. Tudo isto pode comprometer seriamente a qualidade dos dados dos laboratórios, e como conseqüência pode trazer prejuízos consideráveis para os laboratórios.

Uma importante ferramenta para resolver este problema são os Sistemas de Gerenciamento de Informação de Laboratórios (LIMS). Um LIMS é um sistema utilizado por um laboratório para integrar e gerenciar seus dados, um sistema que dá ênfase na melhoria de qualidade dos dados e busca gerar seus resultados de maneira consistente e confiável. O LIMS acompanha o ciclo de vida dos dados, que inclui coleta de dados, armazenamento, análise, emissão de relatórios e arquivamento.

De acordo com as especificações dos principais LIMS do mercado os requisitos de um LIMS são:

- Armazenamento e rastreabilidade de amostras.

- Consulta de informação e geração de relatórios.
- Gerenciamento de atividades.
- Gestão de qualidade (auditoria, documentação, não-conformidades, etc.).
- Calibração e manutenção dos instrumentos dos laboratórios.
- Sistema de login a fim de restringir o acesso aos dados do LIMS.
- Integração com outros sistemas e equipamentos.
- Gerenciamento de materiais e insumos.

Um LIMS devidamente instalado e utilizado em um laboratório pode trazer inúmeros benefícios, os principais são: gerenciamento da explosão de informação, melhoria da qualidade dos dados e maior eficiência e eficácia na execução das atividades dos laboratórios. Hinton [1995]

2.1.1 Gerenciamento da Explosão de Informação

O gerenciamento da explosão de informação vai muito além da simples organização e armazenamento de dados. O LIMS provê um mecanismo avançado para consulta de dados, realiza a interface com a instrumentação do laboratório, inserindo automaticamente grandes quantidades de dados diretamente no seu banco de dados, realiza a interface com outros laboratórios com o compartilhamento de dados, provê a capacidade de importação e exportação de dados com outros *softwares* e ajuda a produzir relatórios e gráficos. Além disso o LIMS deve integra todas as atividades do laboratório, possibilitando o gerenciamento de todo o laboratório apenas com o LIMS.

2.1.2 Melhoria da Qualidade dos Dados

Cada vez mais os laboratórios estão implementado programas de melhoria de qualidade dos dados para atender às regulamentações governamentais e para atender à ênfase em qualidade dos dados requerida pelos pesquisadores, gerentes de laboratórios e pelo mercado de modo geral. Como exemplos de õrgãos governamentais que regulamentam as atividades laboratóriais temos: Agência Nacional de Vigilância Sanitária (ANVISA), Instituto Brasileiro do Meio Ambiente e dos Recursos Naturais Renováveis (IBAMA), Instituto Nacional de Metrologia (INMETRO), Fundação Oswaldo Cruz (Fiocruz) e a U.S. Food & Drug Administration (FDA), que regulamenta a produção de alimentos e a indústria farmacêutica. Um LIMS pode contribuir substancialmente para a melhoria

de qualidade dos dados, uma vez que é muito mais fácil gerenciar dados com um sistema computacional. Exemplos de contribuições que um LIMS pode dar para a melhoria de qualidade dos dados são:

- O LIMS diminui consideravelmente a possibilidade de erros na entrada de dados.
- O LIMS automatiza a verificação de especificações, tornando-a menos susceptível a erros. Por exemplo, dada uma amostra que necessita de 40 testes diferentes, e na execução destes a amostra passou em todos os testes menos um. Uma pessoa pode facilmente não perceber que a amostra está fora das especificações, mas o LIMS identifica este fato imediatamente.
- O LIMS gera dados para auditoria.
- O LIMS automatiza o processo do laboratório.
- O LIMS possibilita o acesso a padrões, calibração de instrumentos, procedimentos e registros.
- O LIMS disponibiliza a documentação das diversas etapas das atividades.

Dentre estes itens destacamos a redução de erros na entrada de dados. A entrada de dados é um ponto bastante crítico, uma vez que esta pode comprometer todas as demais etapas. Assim, um LIMS possui diversos elementos que visam à redução de erros na entrada de dados.

2.1.2.1 Redução de Erros na Entrada de Dados

Diversos mecanismos são adotados para a redução de erros na entrada de dados. Como exemplo podemos citar a restrição na entrada de dados. Existem diversas formas de restringir a entrada de dados, por exemplo, apenas números devem ser aceitos em um campo numérico. Com o LIMS, o usuário tem a possibilidade de definir limiares para os campos, impedindo que sejam inseridos valores impossíveis. Por exemplo, o campo pH pode ser restringido aos valores de 0 a 14. Outra possibilidade de restringir a entrada de dados é em campos que possuem um tamanho fixo de números significativos. Por exemplo, se um campo deve sempre possuir duas casas decimais, então o campo pode ser configurado com a vírgula decimal no local correto e os técnicos do laboratório não poderão inserir dados que não estejam neste formato. Isto irá eliminar erros na entrada de dados, como inserir 100 ao invés de 10,0.

Outro elemento do LIMS que visa à redução de erros na entrada de dados é o uso de códigos de barra. Usando um leitor de código de barras as informações das amostras

podem ser inseridas automaticamente no sistema, ajudando a eliminar a possibilidade de erros.

Outro elemento é o uso de caixas de seleção (*dropdown list*), que é uma lista de opções apresentadas para o usuário preencher um determinado campo, restringindo assim, que o usuário insira no sistema apenas uma destas opções. Para redução de erros na entrada de dados o LIMS também utiliza mensagens de alerta, que são mensagens que buscam evitar possíveis erros nas operações dos usuários. Por exemplo, se o usuário tenta salvar os dados que inseriu, porém algum campo não foi preenchido, o LIMS exibe uma mensagem informando que o campo não foi preenchido. O LIMS também possibilita a definição de campos que são calculados automaticamente. Ou seja, o valor de um campo é preenchido automaticamente a partir de uma fórmula definida pelo usuário, esta fórmula pode utilizar o valor de outros campos e/ou de valores fixos. Este cálculo automático no LIMS ajuda a reduzir erros matemáticos e erros de digitação.

Um último mecanismos adotado para a redução na entrada de dados que citaremos como exemplo, é a geração automática de relatórios e gráficos. O LIMS possui um componente integrado para geração de relatórios. Normalmente, nos laboratórios, os dados são armazenados em um simples sistema de cadastro, para apresentar estes dados de forma aceitável é necessário passá-los para um editor de texto, uma ferramenta gráfica ou em outra ferramenta do gênero. Claramente, este processo manual dá chances para a inserção de erros, que pode comprometer a qualidade dos relatórios e gráficos gerados. Além de gerar relatórios de forma segura, garantindo uma maior qualidade dos dados, o LIMS possibilita a personalização destes relatórios, a fim de adequar-se às necessidades mais específicas de cada laboratório. O LIMS também possibilita a importação e exportação de seus dados para outros sistemas mais comuns.

2.1.3 Maior Eficiência e Eficácia na Execução das Atividades do Laboratório.

O LIMS contribui bastante para o aumento da eficiência e eficácia na execução das atividades do laboratório. Como já foi dito, o LIMS oferece diversos mecanismos para aumentar a produtividade dos laboratórios, como o cálculo automático de campos, a geração automática de relatórios e gráficos e a verificação automática de especificações. Além destes, em muitos casos o LIMS pode tornar a entrada de dados totalmente automatizada, recuperando os dados dos instrumentos usados nos laboratórios diretamente para o LIMS. Em outros casos, os instrumentos geram arquivos que são importados para o banco de dados do LIMS. Além da entrada de dados automática, o LIMS au-

menta a produtividade dos laboratórios com mecanismos avançados de busca para a recuperação dos dados. Normalmente, nos laboratórios, para recuperar um registro, o técnico precisa encontrar arquivos de amostras antigas, analisar documentos, cadernos com o histórico das atividades, ou localizar a amostra física. O LIMS elimina todo esse tempo e trabalho, recuperando o registro desejado em apenas alguns segundos.

2.2 Sistemas de Gerenciamento de Fluxo de Trabalho

Segundo a *Workflow Management Coalition* (WfMC [2009]) fluxo de trabalho corresponde a um processo de negócio onde documentos, informações ou tarefas são passadas de um participante para o outro para execução de uma ação global. As atividades de um fluxo de trabalho podem ocorrer concorrentemente e eventualmente impactar umas nas outras, de acordo com um conjunto de regras. Como exemplo, a Figura 2.1 contém um fluxo de trabalho para realizar uma compra dentro de uma empresa. Primeiramente o pessoal do escritório faz uma solicitação para o setor de compras, em seguida este setor realiza várias pesquisas de preços até que um seja validado e aprovado pelo gerente de compras. Após a validação, a compra é realizada pelo setor de compras que passa o produto para o pessoal do escritório que deve receber o produto.

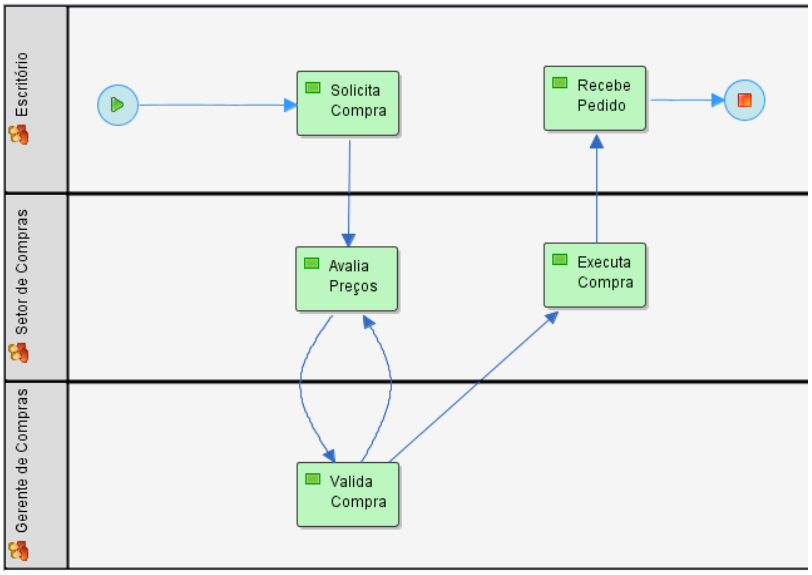
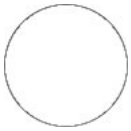


Figura 2.1. Exemplo de fluxo de trabalho

Para a definição de um fluxo de trabalho são utilizadas notações gráficas definidas pela *Business Process Modeling Notation* (BPMN) (BPMN [2009]). A BPMN foi

Tabela 2.1. Principais objetos da BPMN

Nome	Descrição	Representação
Evento	Um evento é uma ação que ocorre durante a execução de um fluxo de trabalho. Estes eventos afetam o fluxo do trabalho e normalmente têm uma causa (disparador) ou um impacto (resultado). Eventos são círculos com centros abertos para permitir indicadores internos para diferenciar diferentes disparadores ou resultados	
Atividade	Uma atividade é um termo genérico para o trabalho que a empresa executa. Uma atividade pode ser atômicas ou não-atômica (composta). Os tipos de atividades que fazem parte de um modelo de processo são: Processo, sub-processo e tarefa. Tarefas e sub-processos são retângulos arredondados. Os processos correspondem a um conjunto de tarefas e sub-processos.	
Fluxo Seqüencial	O Fluxo Seqüencial é representado por uma linha sólida com uma seta sólida e é usado para mostrar a ordem que as atividades serão realizadas em um processo	
Pool	O Pool representa um participante (ator) em um processo. Também pode atuar como um container para o particionamento de um conjunto de atividades	

desenvolvida pela *Business Process Management Initiative* (BPMI) (BPMI [2009]). O objetivo da BPMI foi disponibilizar uma notação que fosse realmente compreensível por todos os usuários de negócio, do analista de negócio que cria as atividades iniciais dos processos, aos desenvolvedores técnicos, responsáveis pela implementação da tecnologia que vai executar esses processos, e finalmente às pessoas das áreas de negócio que vão gerenciar e monitorar aqueles processos. A Tabela 2.1 possui uma descrição dos principais elementos da BPMN para a definição de fluxos de trabalho.

Após a definição de um fluxo de trabalho, a sua execução será controlada por um sistema de gerenciamento de fluxos de trabalho. Um sistema de gerenciamento de

fluxos de trabalho é um sistema que define, gerencia e executa fluxos de trabalho. Estes sistemas permitem controlar as várias atividades associadas ao processo do negócio. O controle das atividades possibilita um acompanhamento e uma análise de todo o processo, a fim de aumentar a eficiência e a eficácia da sua execução e de prover melhorias no processo. Por exemplo, na Figura 2.1, o sistema de gerenciamento de fluxo de trabalho irá automatizar cada passo do processo de compra. Cada atividade será disponibilizada para o ator correspondente, na ordem pré-determinada na definição do fluxo de trabalho. Este gerenciamento garante que todos os passos do processo sejam executados corretamente, evitando, por exemplo, que uma compra seja efetuada sem a validação do gerente de compras.

Com o sistema de gerenciamento de fluxo de trabalho é possível identificar qual o ponto de execução do fluxo de trabalho, ou seja, qual atividade está sendo executada e quem a está executando em um determinado momento. É possível também obter um histórico da execução do fluxo de trabalho, onde serão listadas todas as atividades executadas, seus executores, a ordem e o momento em que foram executadas. Este tipo de informação é fundamental para a análise e inserção de melhorias do processo do negócio.

Um sistema de gerenciamento de fluxo de trabalho possui uma ferramenta para definição de fluxos de trabalho, esta pode ser gráfica ou textual. Com esta ferramenta as atividades, as transições entre as atividades e as regras para execução das atividades são definidas. Este editor de fluxo de trabalho gera como saída um arquivo que contém todas as definições do fluxo de trabalho. Este arquivo será lido pelo sistema de gerenciamento de fluxos de trabalho para execução do mesmo. O arquivo de saída pode seguir vários padrões, como o XPDL (XML *Process Definition Language*) (XPDL [2010]), BPEL (*Business Process Execution Language*) (BPEL [2010]) e o BPML (*Business Process Modeling Language*) (BPML [2010]). Destas linguagens as principais são o XPDL e o BPEL. O padrão XPDL é o padrão mais antigo e foi criado pela *Workflow Management Coalition* (WfMC) (WfMC [2009]). A WfMC surgiu especificamente para promover e manter padrões de fluxos de trabalho. A BPEL foi desenvolvida pelas empresas Microsoft, IBM, Siebel, BEA e SAP, que resolveram se unir e propor um novo modelo, mais completo e universal. Um arquivo XPDL ou BPEL é basicamente um arquivo XML (*Extensible Markup Language*) que segue um conjunto de especificações para definir fluxos de trabalho. O BPEL é compatível e criado a partir da especificação de *Web Services*, consistindo em um sistema de estruturação de uma rede de serviços na WEB, isto possibilita o uso de *Service Oriented Architecture* (SOA [2010]) e *Enterprise Application Integration* (EAI [2010]). Apesar da última versão do XPDL, o XPDL 2.1, suportar *Web Services*, este não é seu foco. O padrão XPDL baseia-se na descrição de

um conjunto de atividades relacionadas entre si através de transições. Para a WfMC, uma atividade significa uma unidade de trabalho que será processada por um recurso (um participante/usuário/ator) e/ou uma aplicação computacional. O XPDL possui alguns conceitos que a BPEL não explora, entre eles, destacam-se as idéias relacionadas a tarefas realizadas por humanos. O XPDL provê formas concretas de especificar regras relacionadas ao envio de tarefas para participantes definidos de maneira dinâmica ou estática. Neste trabalho foram modeladas as atividades de laboratórios biológicos. Em geral, o processo de um laboratório consiste em um conjunto de experimentos que são realizados pelos técnicos nas bancadas dos laboratórios. Apesar de alguns laboratórios utilizarem *Web Services* este não é o foco das suas atividades. Assim, para implementação deste trabalho foi utilizado o padrão XPDL. O XPDL também foi escolhido por ser o padrão mais utilizado.

2.2.1 XML Process Definition Language

O XML Process Definition Language (XPDL) foi um formato padronizado em 1998 pela WfMC para o intercâmbio de definições de processos de negócio entre diferentes produtos de fluxos de trabalho. De acordo com a regra de negócio a ser modelada em um fluxo de trabalho, este pode tornar-se bastante complexo. Sendo assim, o XPDL possui uma série de conceitos importantes que possibilita a modelagem da regra de negócio nos mínimos detalhes. Estes conceitos também estão presentes na especificação do BPMN, contudo a espcificação do XPDL os aborda de forma mais técnica. O XPDL é compatível com o padrão BPMN. Para os criadores do XPDL, o BPMN é o padrão ideal para modelar o processo em nível visual e o XPDL para definir suas regras em nível técnico.

Segundo a WfMC [2009] um dos principais conceitos do XPDL é o processo, este contém todos os objetos que compõem um fluxo de trabalho, que serão descritos a seguir.

2.2.1.1 Processo

Basicamente, um processo é composto por um conjunto de atividades e transições. Para criar um processo é preciso definir seu identificador, nome, descrição, data de criação, autor e versão. A entidade processo fornece informações contextuais que se aplica a outras entidades dentro do processo. É um recipiente para o próprio processo e fornece informações relacionadas com a administração (data de criação, autor, etc) ou para ser utilizado durante a execução do processo (variáveis, participantes, aplicações)

2.2.1.2 Variáveis

Em um processo é possível definir variáveis que serão utilizadas durante sua execução. Para criar uma variável deve-se definir um identificador, um nome e deve-se definir se é uma variável do tipo vetor ou não. Existem outros parâmetros que são opcionais. Por exemplo, para uma variável é possível atribuir-lhe uma descrição, definir o seu valor inicial padrão ou determinar o tamanho da variável. Além destes parâmetros é preciso também definir o tipo da variável. Os principais tipos que uma variável pode assumir são: booleano, data, alfanumérico, inteiro e real. Existem outras possibilidades de tipos como o tipo enumerado ou o tipo declarado.

2.2.1.3 Parâmetros Formais

Além de variáveis um processo pode conter parâmetros formais, que são atributos passados durante a invocação do processo. Um parâmetro formal irá possuir um identificador, uma descrição e um modo que irá definir se trata-se de um parâmetro de entrada ou de saída. Como na definição de uma variável, para um parâmetro formal também deve ser definido um tipo.

2.2.1.4 Grupo de atividades

Para um processo também pode-se definir grupos de atividades. Um grupo de atividades possuirá um identificador e um conjunto de atividades e transições.

2.2.1.5 Participantes

Um processo também possui um conjunto de participantes, que serão os executores das atividades. Um participante deve possuir um identificador, um nome, uma descrição e um tipo definido. Um participante pode ser do tipo humano, sistema, cargo, unidade organizacional, recurso ou grupo de recursos.

2.2.1.6 Aplicações

Um processo pode possuir também um conjunto de aplicações. Estas fornecem uma descrição de aplicações automáticas que podem ser invocadas durante a execução do fluxo de trabalho. Uma aplicação será associada a uma atividade, então esta será executada automaticamente invocando a aplicação associada. A definição de aplicativos reflete a interface entre o motor(sistema de gerenciamento de fluxos de trabalho) e a aplicação, incluindo qualquer parâmetro a ser passado. Uma aplicação deve possuir um identificador, um nome, uma descrição, parâmetros formais e uma referência externa

que irá identificar a sintaxe da chamada externa da aplicação. No XPDL uma aplicação pode ser do tipo EJB (EJB [2010]), POJO (POJO [2010]), XSLT (XSLT [2010]), *Script* ou *Web Service*.

2.2.1.7 Atividades

As atividades irão representar um passo específico na execução do processo. Portanto, estas devem ser definidas dentro de um processo. Todas as atividades possuem um identificador, um nome, uma descrição e a definição do participante do processo que a executará.

2.2.1.8 Transições

Um processo possui um conjunto de transições que irão definir a ordem de execução das atividades. Uma transição possui um identificador, um nome, uma descrição, uma atividade de origem, uma atividade de destino e um tipo. Uma transição pode ser do tipo Condição ou Caso Contrário. O tipo Condição indica que a transição será executada quando sua condição for satisfeita. Esta condição também é definida durante a criação da transição. A condição pode ser, por exemplo, a expressão "pH = 7", ou seja a transição será executada quando a variável pH possuir um valor igual a 7. O tipo Caso Contrário identifica que será a transição padrão para execução caso nenhuma condição seja satisfeita.

2.2.1.9 Pacotes

Em certos contextos a execução de um processo pode ser apenas uma parte do processo global, normalmente isso ocorre nas grandes empresas. Para isto o XPDL define o conceito de pacotes. Um pacote é formado por um conjunto de processos. Ao criar um pacote deve-se definir o seu identificador, nome, descrição, versão e autor. Do mesmo modo que um processo, um pacote pode possuir variáveis, participantes e aplicações. Contudo, uma variável, participante ou aplicação declarada no pacote será visível para todos os seus processos. Além destes, em um pacote pode-se definir os tipos declarados, estes tipos serão usados na definição do tipo de uma variável. Por exemplo, é possível criar o tipo declarado Imagem, com isso é possível definir uma variável como sendo do tipo Imagem. Também os tipos declarados serão visíveis em todos os processos do pacote.

2.2.1.10 Atributos estendidos

Apesar da especificação XPDL conter a maioria das construções que possam ser necessárias na definição de processos, pode haver circunstâncias em que informações adicionais do usuário deverão ser incluídas. Deste modo, quando as extensões são necessárias o XPDL fornece uma forma padrão de estendê-lo. Trata-se dos atributos estendidos. Os atributos estendidos são definidos pelo próprio usuário para expressar qualquer entidade de características adicionais. Os atributos estendidos podem ser adicionados a qualquer um dos componentes do XPDL (pacotes, processos, atividades, variáveis, etc.). Um atributo estendido é composto por um nome e um valor. No decorrer deste trabalho são apresentados alguns exemplos do seu uso.

2.2.2 Enhydra Shark

Atualmente existem muitos sistemas de gerenciamento de fluxos de trabalho, tanto comerciais como gratuitos. Como exemplos temos o Enhydra Shark (Teamlösungen [2009]), ObjectWeb BONITA (BonitaSoft [2009]) e o WfMOpen (GmbH [2009]). Foram analisados diversos sistemas de gerenciamento de fluxos de trabalho e foi decidido utilizar neste trabalho o Enhydra Shark, uma vez que, em nossa análise, este se mostrou o mais completo e eficiente. Os critérios para a escolha do sistema de gerenciamento de fluxos de trabalho foram:

- Ser compatível com o JAVA.
- Possuir uma documentação razoável dos seus métodos.
- Possuir código aberto.
- Ser baseado no padrão XPDL.
- Possuir editor de fluxos de trabalho no padrão XPDL

O Enhydra Shark é uma biblioteca POJO baseada nas especificações da WfMC. Contudo o Shark também possui uma série de bibliotecas adicionais, específicas do Enhydra Shark, para o tratamento de fluxos de trabalho. Além da interface POJO o Shark pode ser usado como um serviço CORBA, EJB, RMI ou *Web Service*. A biblioteca Shark é incorporada ao código de uma aplicação possibilitando o gerenciamento do fluxo de trabalho. Dentre as principais funções da biblioteca Enhydra Shark, temos:

- O Enhydra Shark possibilita carregar e excluir arquivos XPDLs contendo definições de fluxos de trabalho.

- Uma vez carregado o Shark possibilita a inicialização e finalização de processos e atividades, além da sua execução de acordo com as regras definidas no XPDL.

- Após finalizar a execução de uma atividade o Shark irá verificar as transições que saem dessa atividade e irá validar as condições das transições, identificando assim a próxima atividade a ser executada.

- É papel do Shark também controlar os valores das variáveis definidas no fluxo de trabalho. Sendo assim, a biblioteca Shark possui métodos para recuperar e atribuir os valores dos atributos.

- Durante a instalação do Shark é criado um banco de dados específico que armazenará os dados de controle da execução dos fluxos de trabalho. Na configuração da instalação do Shark é possível definir qual o sistema de gerencimento de banco de dados que será utilizado. À medida que as atividades são executadas e finalizadas o Shark persiste no seu banco tais dados, com isso o shark torna possível recuperar o histórico das atividades executadas.

Para criar o fluxo de trabalho foi utilizado o editor de fluxos de trabalho do Enhydra Shark, o *Together Workfow Editor* TWE [2009]. O *Together Workflow Editor* simplifica a definição de um fluxo de trabalho através de uma interface gráfica. Basta selecionar a ferramenta *Activity* e clicar na área principal para criar uma atividade. Ao clicar duas vezes na atividade criada pode-se definir os parâmetros da atividade, como nome, identificador, descrição, etc. Selecionando a ferramenta *Transition* é possível criar uma transição entre duas atividades bastando apenas clicar nas duas atividades. Ao clicar duas vezes na transição criada também é possível definir os seus parâmetros. Com esta mesma facilidade também é possível definir as variáveis de um processo, parâmetros formais, os participantes do processo, aplicações, etc. À medida que o gráfico do fluxo de trabalho está sendo construído o Together gera automaticamente um arquivo XML, que segue o padrão XPDL, contendo as definições do fluxo de trabalho. A tela principal do Together pode ser visualizada na Figura 2.2.

2.3 Estado Atual dos LIMS

Devido à grande complexidade dos LIMS, normalmente estes são desenvolvidos por grandes empresas. Atualmente existem muitos LIMS, contudo a grande maioria possui código fechado e é comercializado com um custo relativamente alto. Devido a este custo, pequenos laboratórios, como a maioria dos laboratórios de pesquisa, utilizam softwares

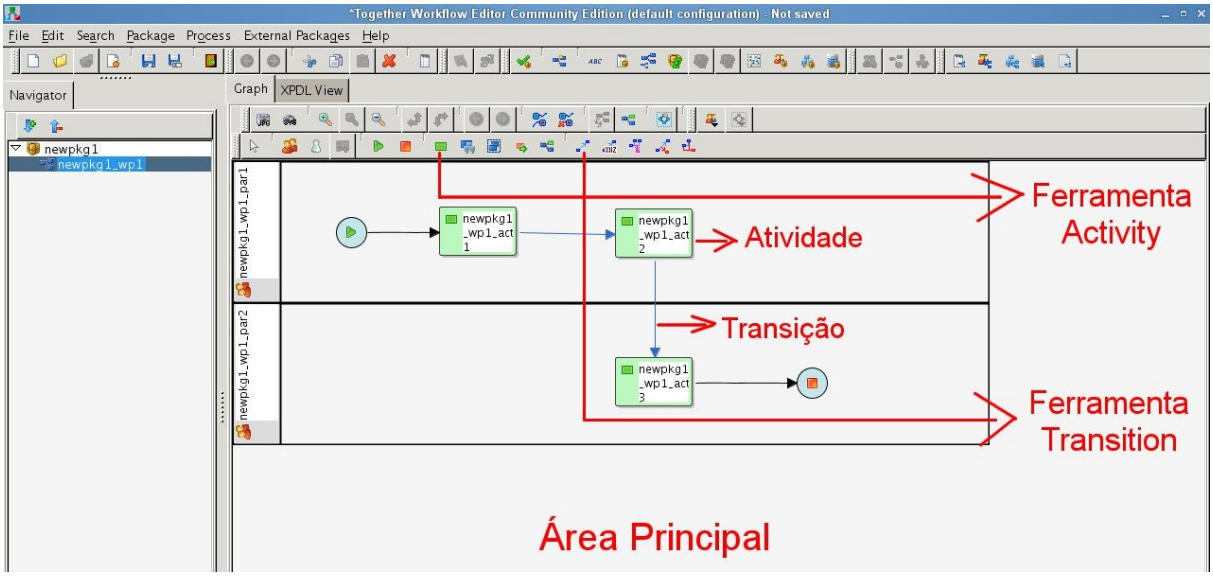


Figura 2.2. Tela Principal do *Together Workflow Editor*

mais simples. Já os laboratórios comerciais, que normalmente são maiores, são "obrigados" a investir em um LIMS para melhorar a qualidade dos seus dados. Como exemplo de LIMS comerciais temos o SQL LIMS (Solutions [2009b]), LabSoft LIMS (Solutions [2009a]) e o LabWare LIMS (LabWare [2009]), desenvolvidos por empresas particulares. Normalmente estes LIMS são específicos para um tipo de laboratório. O SQL LIMS, por exemplo, possui soluções distintas para laboratórios farmacêuticos, químicos, alimentícios, forenses e laboratórios de análise de água. As atividades e procedimentos são bastante diferentes entre os diversos tipos de laboratórios, por isso existe a necessidade, por parte das empresas, de especializar seus sistemas. A grande diversidade de laboratórios é uma razão para a existência de muitos LIMS disponíveis no mercado. Contudo, apesar das empresas procurarem tornar seus sistemas adaptáveis para cada cliente, a tarefa de um usuário encontrar o LIMS ideal para seu laboratório não é fácil. Mesmo laboratórios do mesmo tipo, possuem particularidades nos procedimentos que os diferenciam. A menos que seja um LIMS construído sob medida para o laboratório, sempre existirão adaptações a serem feitas no processo do laboratório para adequar-se ao LIMS. Em relação aos LIMS gratuitos normalmente estes são limitados, como o FreeLIMS, desenvolvido pela empresa alemã Labmatica. A Labmatica oferece uma versão gratuita e uma versão comercial, naturalmente, a versão gratuita possui limitações em relação à comercial. Alguns LIMS gratuitos foram construídos em trabalhos acadêmicos, contudo, a maior parte destes sistemas limitam-se a gerenciar apenas algumas atividades de um laboratório. Além disso a preocupação com a qualidade dos dados é muito pouca como em Hendricks & Learn [2003] ou Quo & B.M. Wu [2005]. Como

exemplos de trabalhos acadêmicos que desenvolveram LIMS temos Hendricks & Learn [2003], que desenvolveu um LIMS para laboratórios acadêmicos de fabricação de microchips. O trabalho Quo & B.M. Wu [2005] desenvolveu um LIMS para laboratórios de pesquisa em câncer, e Tharayil & T. Kalbfleisch [2007] e Morisawa et al. [2006] que desenvolveram LIMS para laboratórios biológicos. É importante notar que também estes LIMS acadêmicos são soluções para laboratórios específicos.

Na literatura também é possível identificar LIMS que, como o SIGLa, incorporam diversos conceitos de fluxo de trabalho, como em Wilkins et al. [2008] e Esterling & Overgaard [2009] que gerenciam dados de laboratórios que estudam proteínas. Contudo estes não possuem um sistema de gerenciamento de fluxo de trabalho diretamente incorporado ao código, como o SIGLa possui. O LIMS apresentado em Wilkins et al. [2008] e Esterling & Overgaard [2009] são específicos para laboratórios de proteômica. Em Esterling & Overgaard [2009] é mencionado que este possui a capacidade de se adaptar a outros tipos de laboratórios, contudo, para isso, é necessário modificar o código do sistema. Com o SIGLa essa adaptação pode ser feita sem modificar nenhuma linha de código.

Capítulo 3

Projeto do Sistema

3.1 Levantamento de Requisitos

Para o levantamento de requisitos do SIGLa foram realizadas entrevistas com a Dra. Alessandra Faria-Campos e com a Dra. Daiane de Laat. A Dra. Alessandra Faria-Campos possui formação em bioquímica e possui experiência.em laboratórios de proteômica. A Dra. Daiane de Laat é bióloga e possui experiência em laboratórios de microarranjos.

O laboratório de proteômica visa identificar e analisar proteínas expressas em diferentes condições e fases da vida de um organismo. Para isso são executadas diversas atividades que visam separar e identificar as proteínas presentes em uma amostra analisada. Cada uma das atividades possui um conjunto de informações que precisam ser armazenadas para análises futuras. Esta informações podem ser números inteiros ou reais, podem ser alfanuméricos, datas, arquivo de dados ou de imagens. Um projeto no laboratório de proteômica pode possuir várias amostras a serem analisadas. Em alguns casos é preciso separar as amostras analisadas em grupos de amostras. Várias análises são executadas em paralelo. Ao fim de cada análise é preciso criar relatórios que detalhem a execução da análise. A análise no laboratório é executada por uma equipe de técnicos, assim todos devem ter acesso às atividades já executadas. Por outro lado o acesso aos dados das análises deve ser restrito, os técnicos que não estiverem vinculados a uma determinada análise não poderão acessar as suas informações.

No laboratório de microarranjos é feita a análise de genes. Esta análise é feita com a execução de um conjunto de atividades. Da mesma forma, como no laboratório de proteômica, também é preciso armazenar as informações sobre cada atividade executada. Esta informações também podem ser números inteiros ou reias, podem ser alfanuméricos, datas, arquivo de dados ou de imagens. Cada uma das atividades irá

gerar um resultado que será utilizado na próxima atividade a ser executada. Como as atividades, os resultados também possuem informações que devem ser armazenadas para consultas posteriores. Um grupo específico de técnicos trabalham na análise de uma amostra, contudo também existe restrição de acesso aos dados. Também no laboratório de microarranjos várias análises são executadas em paralelo.

Atualmente, em ambos laboratórios, todo o registro dos dados das atividades e resultados são feitos em cadernos. Cada técnico possui o seu próprio caderno onde anota cada detalhe das atividades executadas. Os tecnicos registram a data de cada atividade e assinam todas as páginas do cardeno para comprovar que executaram as atividades. Para consultar um dado um técnico deve ter permissão para consultar o caderno de um outro técnico, caso a informação desejada não esteja no seu próprio caderno. A consulta no carderno é baseada na ordem cronológica das atividades executadas. Após cada análise os relatórios finais são feitos manualmente em editores de texto.

A proposta do SIGLa é automatizar a execução de todas as atividades dos laboratórios de proteômica e de microarranjos. Assim, no banco de dados do SIGLa serão armazenadas todas as informações das atividades executadas e dos seus resultados e entradas. Os dados das atividades serão consultados apenas por usuários que tenha permissão para tal consulta. Deste modo o banco de dados do SIGLa irá substituir e integrar as informações dos cadernos dos técnicos. O relatório final poderá ser emitido pelo próprio SIGLa.

Com base nas entrevistas realizadas os seguintes requisitos funcionais foram definidos:

- Possuir tela para o cadastro de atividades.
- Armazenar dados do tipo inteiro, real, alfanumérico, data ou arquivo.
- Possuir tela para o cadastro de resultados de uma atividades. Os resultados poderão ser excluídos ou modificados durante a definição de resultados.
- Possuir tela para a seleção dos resultados que servirão como entradas de uma determinada atividade.
- Possuir interface para visualização geral das atividades executadas.
- Possuir interface para a consulta dos dados das atividades executadas.
- Possibilitar a visualização simultânea de múltiplas amostras de um mesmo projeto.
- Possibilitar o agrupamento de amostras em um projeto.

- Cadastro, edição e exclusão de usuários do sistema. O gerenciamento de usuários deverá ser feito apenas por um usuario Administrador do Sistema.
- Possilitar a definição de quais dados cada usuário poderá ter acesso.
- Emissão de relatórios contendo os dados das atividades executadas em uma determinada análise.

A fim de que o SIGLa possua as principais características de um LIMS os seguintes requisitos funcionais foram definidos:

- O SIGLa deverá validar o tipo de cada atributo preenchido em uma atividade.
- O SIGLa deverá permitir a definição de exemplos de preenchimento a serem exibidos ao lados dos atributos das atividades.
- O SIGLa deverá validar o formato de cada atributo das atividades, caso este possua um formato específico.
- O SIGLa deverá validar se o preenchimento de um campo é obrigratório ou não.
- O SIGLa deverá disponibilizar para o usuário um lista de seleção para o preenchimento dos atributos de uma atividade, caso este atributo possua um conjunto restrito de valores com os quais pode ser preenchido.

Com base no objetivo deste trabalho e nas entrevistas realizadas os seguintes requisitos não-funcionais foram definidos:

- O SIGLa deverá ser um sistema flexível onde todo o processo de cada laboratório será definido em um fluxo de trabalho.
- O SIGLa deverá permitir que os técnicos manipulem o sistema em paralelo, a partir de terminais distintos.
- Todos os componentes do SIGLa deverão possuir código aberto.
- A base da dados do SIGLa deverá suportar grandes quantidades de dados.

3.2 Análise de Requisitos

Para que o SIGLa seja flexível todo o processo de cada laboratório é definido em um fluxo de trabalho. Assim, um fluxo de trabalho possui os seguintes conceitos:

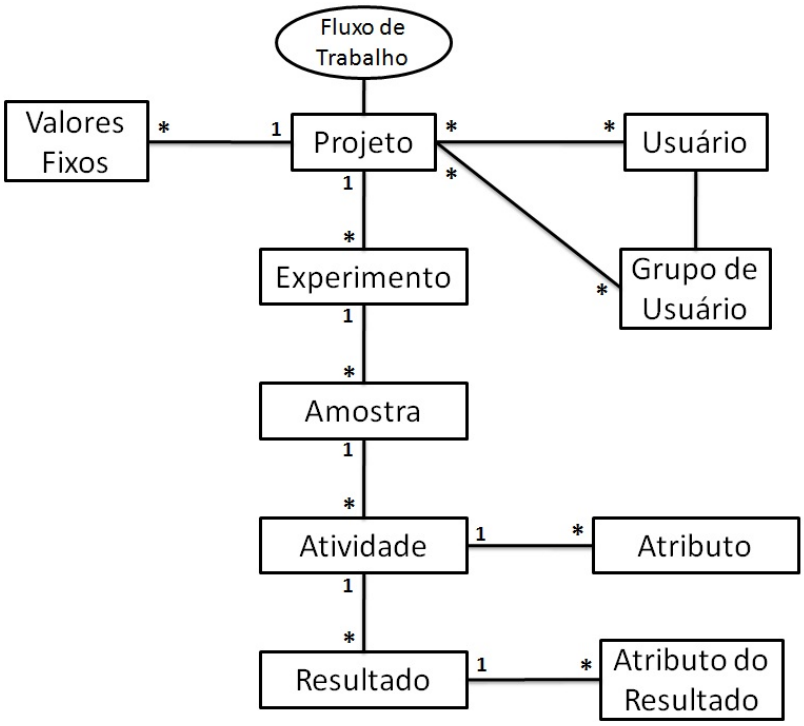


Figura 3.1. Modelo Conceitual do SIGLa

- O nome das atividades executadas no laboratório.
- A ordem de execução das atividades.
- O nome dos atributos de cada atividade.
- O tipo de cada atributo, que poderá ser inteiro, alfanumérico, data ou arquivo.
- O formato da cada atributo, caso este possua um formato pré-definido.
- O exemplo de preenchimento de cada atributo, caso seja necessário.
- Uma informação que defina se um atributo deverá ser preenchido obrigatoriamente ou não.

Baseado nos requisitos levantados foi definido o modelo conceitual do SIGLa representado na Figura 3.1.

As atividades a serem executadas no SIGLa são definidas no fluxo de trabalho, assim o SIGLa é ser capaz de executar diferentes atividades com atributos diversos. O SIGLa possui o conceito de atividade que representa as atividades executadas, cada atividade executada possui associada a ela um conjunto de atributos. Uma atividade possui resultados que podem ser as saídas da atividade ou as suas entradas. Como

as atividades um resultado também possui atributos. Um conjunto de atividades está associado à análise de uma amostra. Para possibilitar o agrupamento de amostras um conjunto de amostras está associado a um experimento. Os experimentos que estiverem vinculados a um mesmo estudo são associados a um mesmo projeto. Um projeto possui um fluxo de trabalho que define as atividades a serem executadas. Para possibilitar que o SIGLa disponibilize para o usuário um lista de seleção para o preenchimento dos atributos de uma atividade, um projeto possui um conjunto de valores fixos que representam os valores com os quais os atributos podem ser preenchidos. Para restringir o acesso aos dados é associado a cada projeto um conjunto de usuários que podem ter acesso aos dados do projeto. Um usuário pode ser associado individualmente a um projeto ou através de um grupo de usuários.

Para possibilitar o acesso ao SIGLa por vários técnicos em paralelo o SIGLa é um aplicação Web com uma arquitetura cliente-servidor. Esta arquitetura possibilita também uma maior segurança no acesso aos dados do laboratório. Por ser uma aplicação Web complexa e com código aberto, para o desenvolvimento do SIGLa foi utilizado como linguagem de programação o JAVA com JSP. Para possibilitar o armazenamento de grande quantidade de dados e manter o código aberto foi utilizado o Sistema de Gerenciamento de Banco de Dados MySQL. O SIGLa gerencia a execução de fluxo de trabalho, para facilitar o desenvolvimento do SIGLa o Enhydra Shark foi utilizado para executar as rotinas de gerência de fluxo de trabalho.

A Figura 3.2 contém o diagrama de caso de uso do SIGLa. Neste diagrama é possível identificar os usuários do sistema. Por razões de segurança dos dados apenas o usuário Administrador poderá manter os usuários, manter os grupos de usuários e editar ou excluir os experimentos e projetos. O coordenador será responsável por criar os projetos, seus experimentos e definir os valores fixos dos atributos das atividades. O coordenador também poderá consultar as atividades executadas e emitir relatórios. Por fim, o técnico será responsável por executar as atividades, definir suas entradas e resultados. O técnico também poderá consultar as atividades executadas.

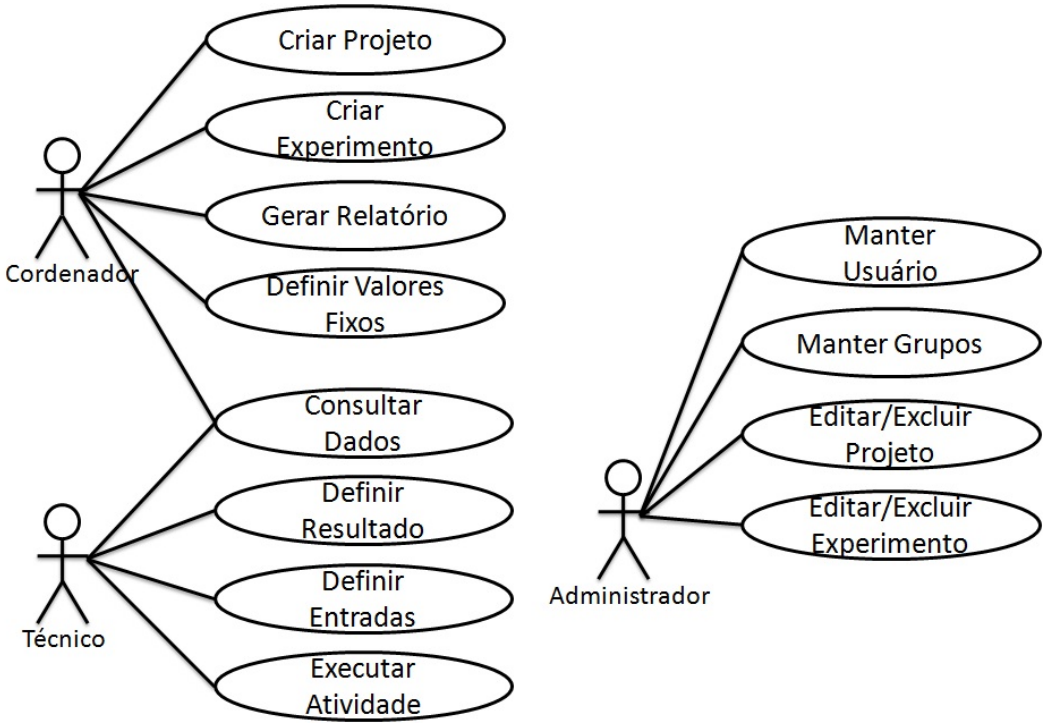


Figura 3.2. Diagrama de Caso de Uso

Capítulo 4

Sistema Integrado de Gerenciamento de Laboratórios (SIGLa)

Neste trabalho foi desenvolvido o Sistema Integrado de Gerência de Laboratórios (SIGLa), uma aplicação Web com alto grau de usabilidade, com código aberto e que se adapta a vários laboratório, ao contrário da maioria dos LIMS atuais que possuem código fechado e são específicos para um tipo de laboratório.

O SIGLa possui uma interface totalmente gráfica que propõe-se a integrar e gerenciar a execução de todas as atividades de um laboratório, dando ênfase na melhoria da qualidade dos dados. O SIGLa possui recursos que visam a melhorar de qualidade dos dados, desde a garantia da correta execução de pequenos detalhes, como um campo que deve ser preenchido corretamente, como garante também que aspectos importantes sejam executados corretamente, como as atividades do laboratório, que devem ser executas na ordem correta. Nas próximas seções deste capítulo serão explicadas diversas características do SIGLa que visam a melhoria da qualidade dos dados.

Para tornar o SIGLa adaptável a múltiplos laboratórios foi utilizado o conceito de fluxos de trabalho. A idéia do SIGLa é gerenciar a execução de atividades definidas em um fluxo de trabalho, deste modo basta definir um fluxo de trabalho modelando as atividades de cada laboratório e o SIGLa estará ápto para gerenciá-los. Uma vez que as atividades, regras e procedimentos de um laboratório foram definidos em um fluxo de trabalho, o editor de fluxo de trabalho gera o arquivo XPDL com todas as definições. Esse arquivo é carregado no SIGLa, então o LIMS estará pronto para gerenciar as atividades do laboratório. A flexibilidade do SIGLa proporciona não apenas a sua adaptação a múltiplos laboratórios, mas possibilita também que os laboratórios possam

especificar no seu fluxo de trabalho as suas peculiaridades. Esse mecanismo possibilita também modificar seu fluxo de trabalho com o passar do tempo, adaptando-o às novas necessidades do laboratório. Por exemplo, caso o processo do laboratório seja modificado e uma nova atividade seja requerida, ou simplesmente um novo atributo de uma atividade já existente seja necessário. Para atender a essas necessidades basta criar um novo fluxo de trabalho a partir do fluxo de trabalho existente acrescentando uma nova atividade ou o novo atributo. A partir daí todos os novos projetos irão carregar o fluxo de trabalho atualizado que contém o novo processo. Desta forma o SIGLa se mantém atualizado. No desenvolvimento do SIGLa foi utilizada a biblioteca Enhydra Shark para executar as rotinas específicas de gerenciamento de fluxos de trabalho.

4.1 Definição de Fluxos de Trabalho

Para definir um fluxo de trabalho foi utilizada a versão gratuita do *Together Workflow Editor* (TWE [2009]). Este é um editor gráfico, baseado no padrão XPDL que possibilita que qualquer usuário que conheça as atividades do laboratório e as especificações do XPDL, que foram descritas na sessão 2.2.1, seja capaz de definir o seu fluxo de trabalho.

Alguns dos LIMS, principalmente os comerciais, incorporam o conceito de fluxo de trabalho para a execução das atividades dos laboratórios. Contudo, estes fluxos de trabalho são definidos apenas com os componentes tradicionais, como atividades, transições, atores e regras de transições. A proposta do SIGLa é aproveitar todos os componentes tradicionais de um fluxo de trabalho, uma vez que estes componentes já definem, de maneira geral, todas as atividades de um laboratório. Contudo, o SIGLa possibilita, principalmente através do uso de atributos estendidos, definir também os detalhes existentes em cada processo. Por exemplo, durante a definição das atividades do laboratório, o próprio usuário pode definir atributos das atividades, os seus tipos, suas unidades de medida ou os seus formatos. Também é possível definir as entradas e saídas de cada atividade bem como a relação dessas entradas e saidas com as atividades. Além destes detalhamentos de processos, na definição do fluxo de trabalho é possível associar à cada atividade arquivos que poderão conter padrões, calibração de instrumentos, procedimentos e registros associados à atividade. É possível também definir sub-processos, o que possibilita criar vários caminhos paralelos na execução do fluxo de trabalho.

O primeiro passo na definição de um fluxo de trabalho para o SIGLa é criar um pacote. Em seguida define-se os processos associados ao pacote, e então são criadas as

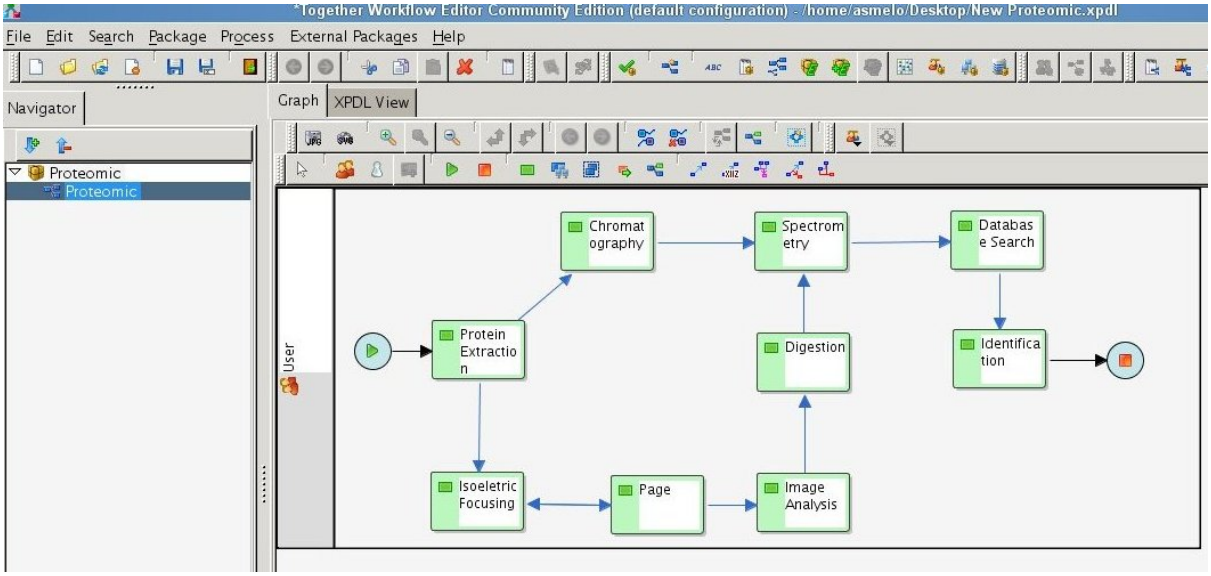


Figura 4.1. Exemplo de fluxo de trabalho definido no *Together Workflow Editor*

atividades dos processos. Com a criação de transições é definida a ordem em que as atividades serão executadas. A Figura 4.1 contém um exemplo de fluxo de trabalho definido no Together. As caixas representam as atividades do laboratório e as setas definem as transições. A definição das atividades e a ordem em que serão executadas é um passo muito importante, pois irá garantir que todas as atividades serão executadas, e mais, serão executadas no momento correto. Este controle é fundamental, principalmente em um laboratório onde o número de atividades é muito alto, onde uma atividade pode facilmente não ser executada, ou ser executada no momento errado.

Caso uma atividade possua mais de uma transição saindo para outras atividades distintas, é necessário definir condições nas transições para que a biblioteca Enhydra Shark identifique a próxima atividade a ser executada. Para isso deve-se definir uma variável com o nome Next Activity. Esta variável deve ser criada no pacote que define o fluxo de trabalho, para que possa ser acessível a todos os processos criados. O valor desta variável irá determinar a próxima atividade a ser executada. Assim, ao criar no fluxo de trabalho uma transição saindo de uma atividade que possui múltiplas saídas, deve-se definir a sua condição da seguinte forma: Next Activity==“<identificador da próxima atividade>”. Por exemplo, caso a transição aponte para a atividade Chromatogaphy, a sua condição será Next Activity=="Chromatography".

Além destas definições deve-se criar os atributos de cada atividade. Os atributos de uma atividade são as informações, referentes à execução da atividade, que devem ser armazenadas. Por exemplo, para a atividade *Protein Extraction* (Extração de Proteína), os atributos desta atividade poderiam ser: Organismo, que identificaria

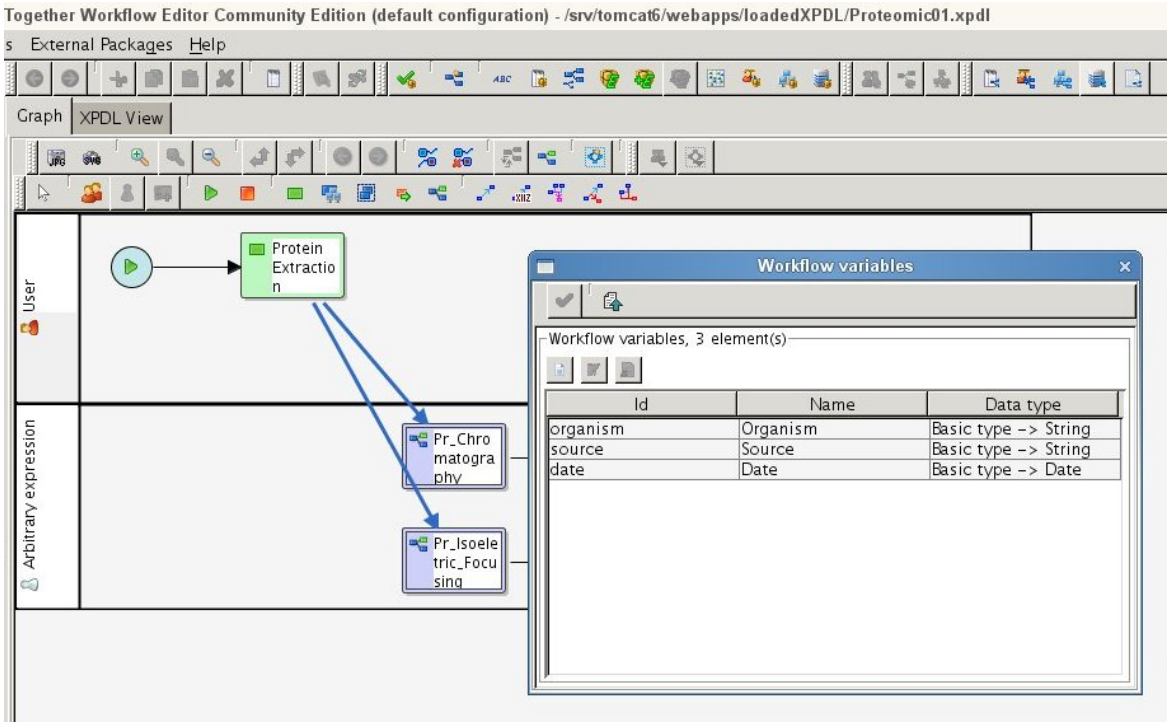


Figura 4.2. Definição das variáveis de um processo

o organismo do qual foi extraida a proteína, Fonte, que identificaria, mais especificamente, de onde a proteína foi extraída e Data, que informaria a data em que a atividade foi executada.

Em termos de definição, os atributos de uma atividade são criados como variáveis do processo que contém a atividade. Para cada variável deve-se definir um identificador, um nome e seu tipo, que pode ser *String*, Alfanumérico, Inteiro, Data ou Real. A definição dos tipos dos atributos é extremamente importante para o processo de melhoria de qualidade dos dados do laboratório, uma vez que o SIGLa irá validar se cada atributo foi preenchido corretamente, de acordo com seu tipo. A Figura 4.2 mostra o *Together Workflow Editor* no momento em que as variáveis do processo, que contém a atividade *Protein Extraction*, estão sendo definidas.

Para associar as variáveis do processo à atividade são criados atributos estendidos na atividade. Os atributos estendidos possuirão como valor os identificadores das variáveis do processo. O nome do atributo estendido deve ser Activity, para identificar que o atributo refere-se à atividade. A Figura 4.3 mostra a interface do *Together Workflow Editor* no momento em que os atributos estendidos da atividade *Protein Extraction* estão sendo definidos. Note que estes atributos estendidos estão referenciando os identificadores das variáveis criadas na Figura 4.2.

É possível também definir um atributo como sendo do tipo arquivo, fazendo isso

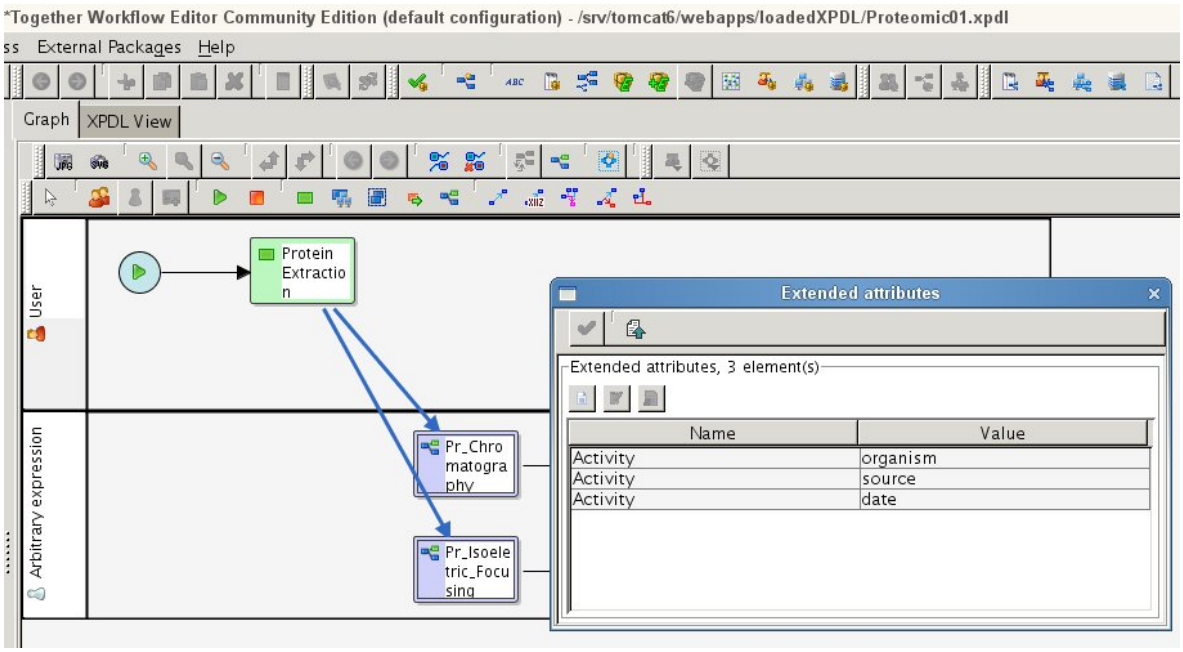


Figura 4.3. Definição dos atributos da atividade Extração de Proteínas

o SIGLa irá associar ao atributo um arquivo qualquer selecionado pelo usuário. Este arquivo pode conter, por exemplo, a descrição dos protocolos utilizados na atividade, ou pode ser uma imagem associada à atividade. Uma vez que a especificação XPDL não possui este tipo definido é preciso definir um tipo declarado no pacote do fluxo de trabalho. Este tipo deve possuir o nome *File* e o identificador *file*. Assim, durante a criação de uma variável do processo, na definição do seu tipo seleciona-se o tipo declarado *File*. Deste modo o SIGLa identificará que este é um atributo do tipo arquivo. A Figura 4.4 mostra o *Together* no momento em que o tipo declarado *File* está sendo criado. A Figura 4.5 mostra o momento em que o tipo da variável *Protocol* está sendo definido como *File*.

Além de definir o tipo de cada atributo também é possível definir algumas propriedades dos atributos. As propriedades são:

- *NOT NULL*: Define que o atributo deve ser preenchido obrigatoriamente. Uma vez que o atributo possua essa propriedade, o SIGLa irá verificar se o atributo foi preenchido, evitando assim que o usuário esqueça de preenche-lo.

- *FORMAT*: Define o formato de um atributo. A Figura 4.6 contém um exemplo onde o formato do atributo *Peak* está sendo definido como 0.##, o que significa que este atributo apenas poderá ser preenchido com um "0." seguindo de dois números. Utilizando números e o símbolo #, diversos formatos podem ser defi-

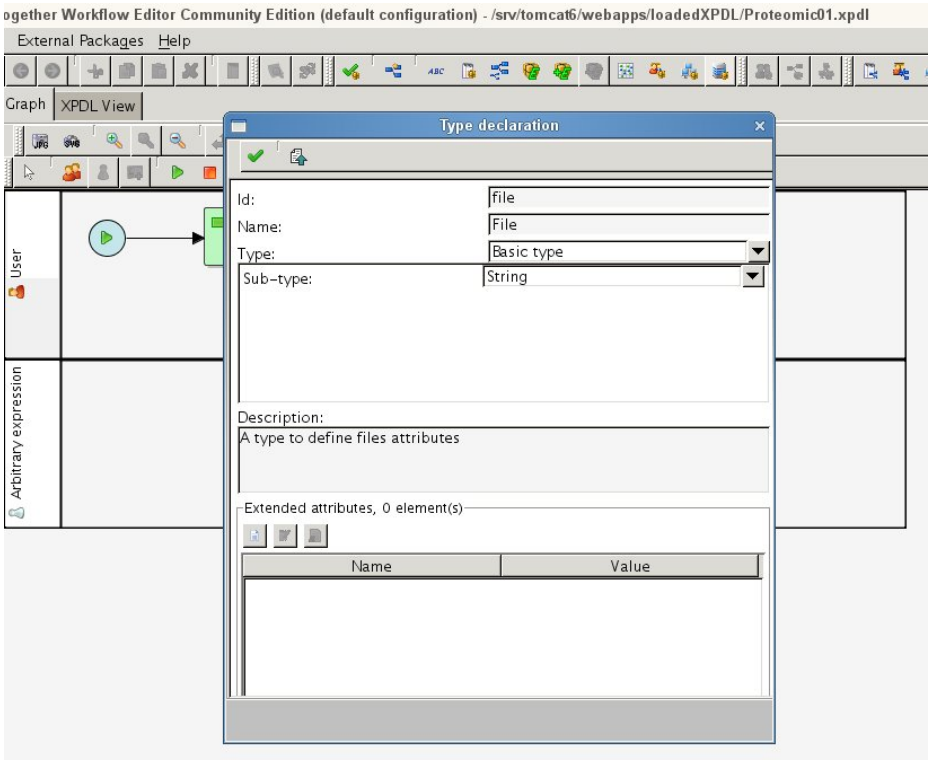


Figura 4.4. Definição do tipo declarado File

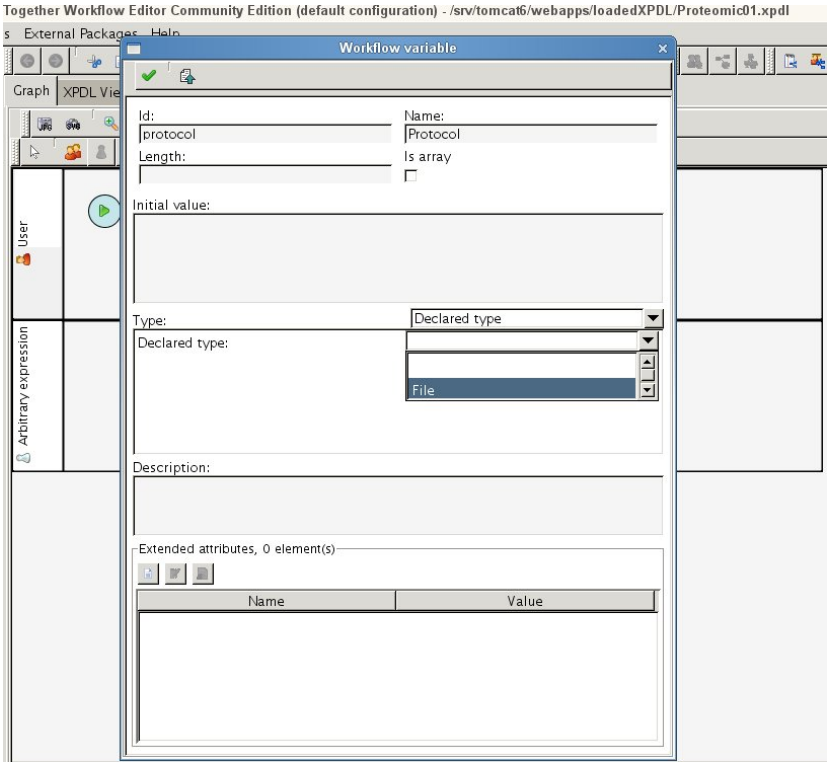


Figura 4.5. Atribuição do tipo File para a variável Protocol

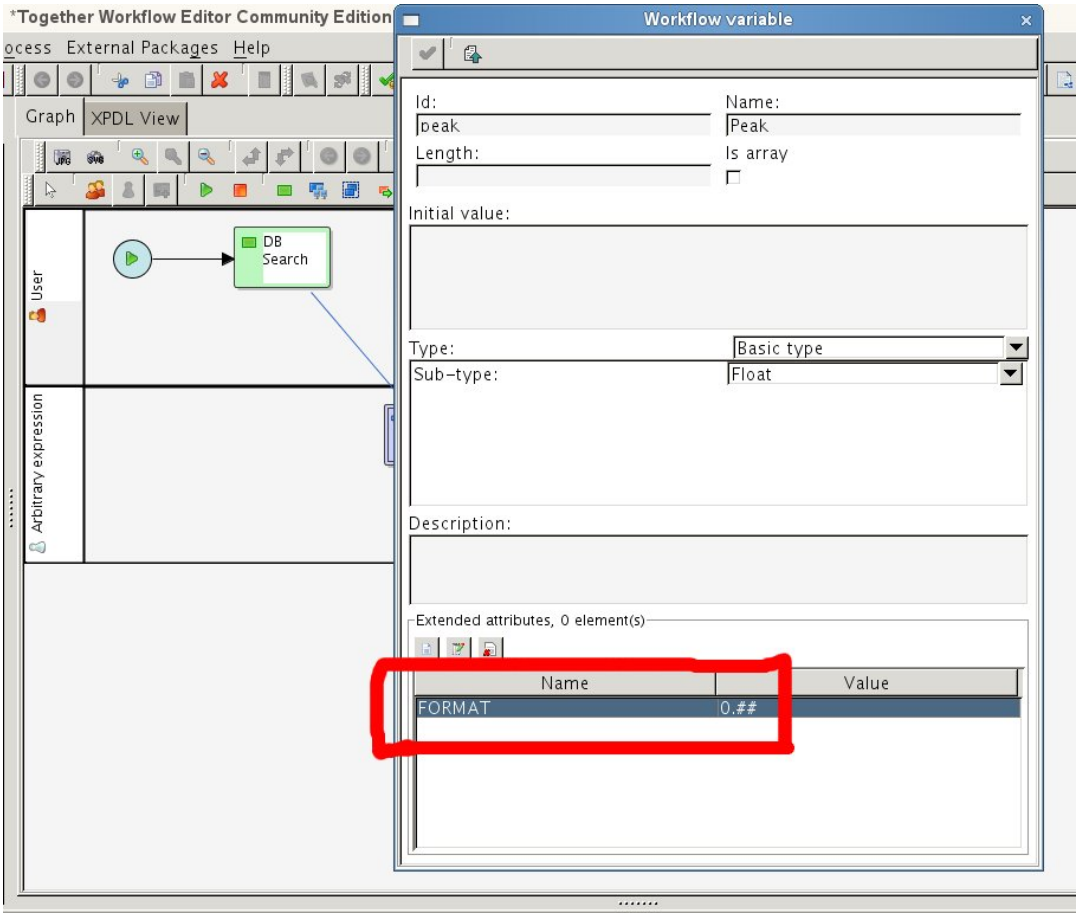


Figura 4.6. Definição do formato do atributo Peak

nidos. Esta propriedade diminue bastante a possibilidade de erros de digitação como colocar 09.3, ao invés de 0.93.

- *EXAMPLE*: Define um exemplo de dado ou uma medida, que aparecerá ao lado do campo, facilitanto assim o preenchimento do mesmo. Por exemplo, o atributo Condição pode ter como exemplo Congelado ou Líquido. Um outro exemplo seria o atributo Concentração que pode ter como medida $ng/\mu L$. Assim, esta medida ou exemplo aparecerá ao lado do campo, orientando assim, como o usuário deve preencher o campo.

Para criar um atributo de uma atividade deve-se criar uma variável no processo desta atividade. Assim, a definição de uma propriedade de um atributo é feita através de atributos estendidos definidos na variável do processo. Por exemplo, para atributir o exemplo $ng/\mu L$ para o atributo Concentração, deve-se criar na variável Concentração um atributo estendido que possua o nome Example e o valor $ng/\mu L$. Na Figura 4.6

pode-se perceber que para definir o formato de atributo Peak foi definido um atributo estendido.

Além de definir os atributos das atividades é possível definir os resultados/saídas das atividades. Cada atividade pode gerar resultados que serão entradas da próxima atividade a ser executada. Para definir que uma atividade possuirá saídas deve-se criar na atividade um atributo estendido com o valor output. O nome deste atributo estendido deverá ser Parameter, para identificar que este atributo estendido não está referenciando um atributo comum, da atividade. Como as atividades, os resultados também possuem atributos que são as informações associadas a cada saída. Da mesma forma, deve-se definir o tipo e as propriedades dos atributos dos resultados. Os atributos dos resultados também são definidos como variáveis do processo que contém a atividade. Da mesma forma, são criados na atividade os atributos estendidos para associar a atividade às variáveis. Para diferenciar os atributos das atividade dos atributos dos resultados da atividade, os atributos estendidos que associam os atributos dos resultados devem possuir o nome Result. A Figura 4.7 mostra um exemplo dos atributos estendidos definidos para a atividade *Protein Extraction* (Extração de Proteínas). De acordo com o nome dos atributos estendidos é possível identificar que esta atividade irá possuir os atributos *Organism*, *Source* e *Date*. Uma saída da atividade possuirá como atributo *Concentration*, *Volume*, e *Number*. O atributo estendido *Parameter* com valor *output* foi definido para identicar que esta atividade possui resultados.

É possível também definir se uma atividade possui entradas. Uma vez que esta definição seja feita, as entradas da atividade serão, automaticamente, as saídas geradas na atividade anterior. Para definir que uma atividade possui entradas deve-se criar, na atividade, um atributo estendido com o valor *input*. O nome deste atributo deve ser *Parameter*.

Por fim, ao definir as entradas e saídas no fluxo de trabalho também é possível definir o número máximo e mínimo de entradas e saídas das atividades. Para isso define-se um atributo estendido na variável do processo. A Figura 4.8 mostra um exemplo onde o número mínimo de saídas de uma atividade está sendo definido como 1. Para isso é preciso criar no processo da atividade a variável output. Ao criar a variável output define-se um atributo estendido que possuirá o nome *min number* e o valor será o número mínimo de resultados. Para definir o número máximo cria-se o atributo estendido *max number*. Para definir estas propriedade para as entradas de uma atividade deve-se criar a variável *input*.

Toda a definição do worflow, descrita aqui, torna-se bastante simplificada com o uso do *Together Workflow Editor*. Uma vez que este possui um interface gráfica que torna a criação muito simples. Por exemplo, para criar uma atividade basta selecionar

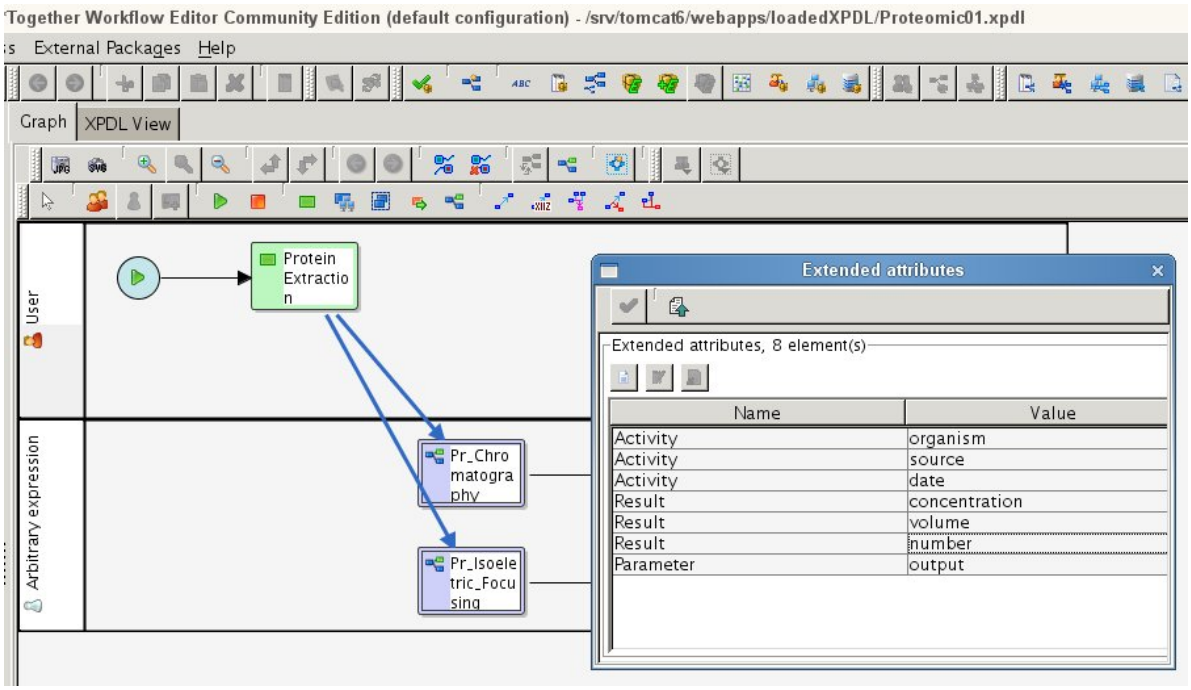


Figura 4.7. Exemplos de atributos estendidos

o ícone Activity e clicar na área de desenho do fluxo de trabalho. Para definir os parâmetros (identificador, nome, descrição, etc.) das atividades basta dar dois cliques nas atividades e adicionar os parâmetros através de uma interface totalmente gráfica.

Apesar de ser simples, a definição do fluxo de trabalho é uma etapa muito importante. Para que o SIGLa possa gerenciar com sucesso um laboratório é necessário que o seu fluxo de trabalho seja bem definido. Este deve conter todas as atividades com suas entradas, saídas e seus atributos devidamente declarados com seu tipo e propriedades. A definição do fluxo de trabalho é um passo bastante importante no processo de melhoria de qualidade dos dados do laboratório propciado pelo SIGLa.

4.1.1 Subprocessos

Ao definir todas as atividades em um único processo, o sistema de gerenciamente de fluxo de trabalho, o Enhydra Shark, irá executar cada atividade do fluxo de trabalho uma única vez até que chegue ao final do mesmo. Contudo no dia-dia dos laboratórios não é isto que acontece. Muitas vezes uma atividade não apresenta um bom resultado, sendo assim, a atividade deve ser executada novamente. Acontece também de um experimento executar uma mesma atividade para várias amostras, e cada uma dessas gerar uma execução distinta do fluxo de trabalho. A Figura 4.9 contém um exemplo da execução de um fluxo de trabalho onde a atividade *Image Analysis* precisou ser

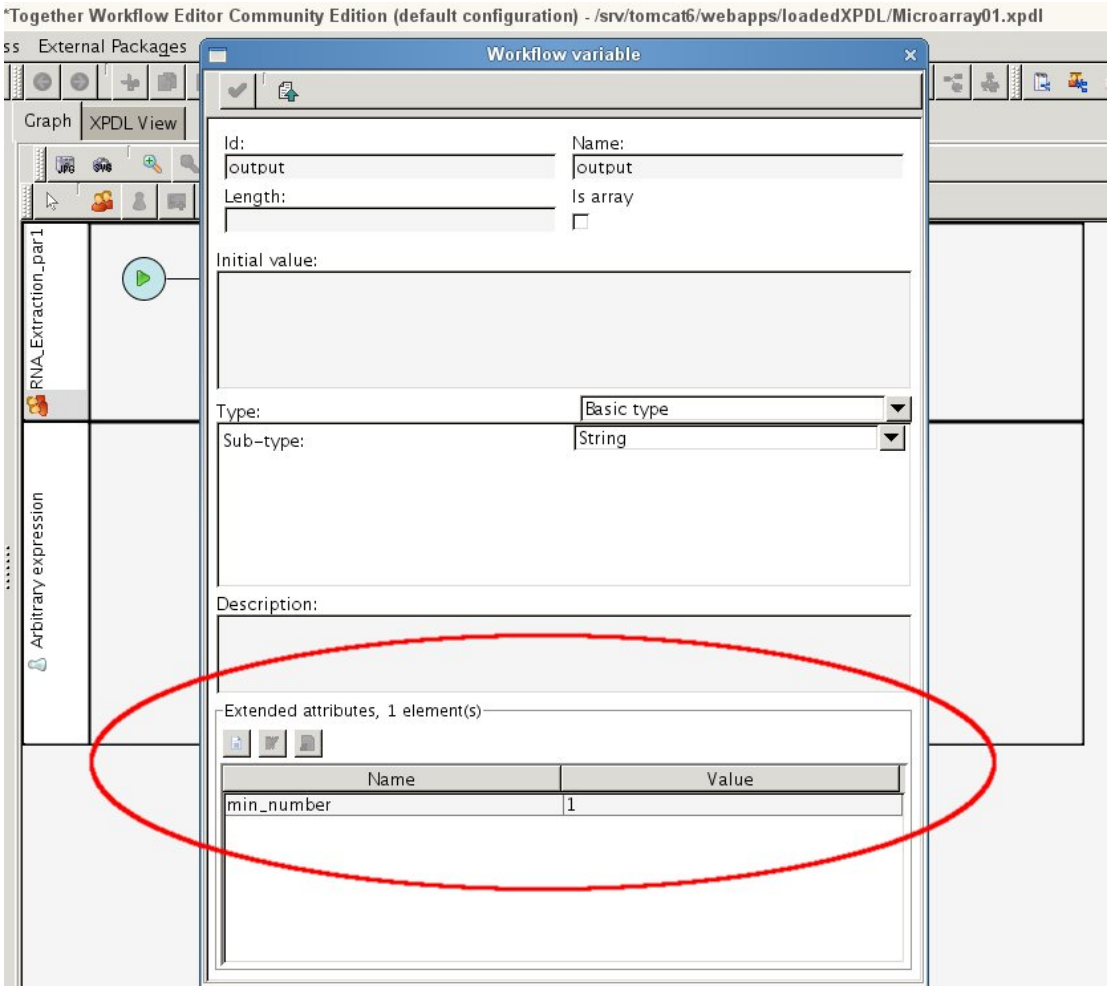


Figura 4.8. Definição do número mínimo de saídas que uma atividade pode ter

executada duas vezes. A Figura 4.10 mostra um exemplo onde três amostras foram analisadas depois da execução da atividade *Protein Extraction*.

Para dar esta flexibilidade de execução é preciso criar um processo para cada atividade do laboratório. Para o Shark um processo pode ser inicializado a qualquer momento. Assim, como cada atividade está associada a um processo, qualquer atividade pode ser inicializada a qualquer momento e quantas vezes for necessário. Para evitar que as atividades sejam executadas desordenadamente, na construção do fluxo de trabalho é definida a ordem de execução dos processos. Por exemplo, na Figura 4.11 temos a definição do processo que mostra a atividade de *Protein Extraction*, após a execução desta será possível inicializar apenas o processo da *Cromatography* ou da *Isoeletric Focusing*. Na Figura 4.12 temos a definição do processo que mostra a atividade de *Isoeletric Focusing*, após a execução desta será possível inicializar apenas o processo do PAGE. Construíndo o fluxo de trabalho desta forma possibilitamos a execução de

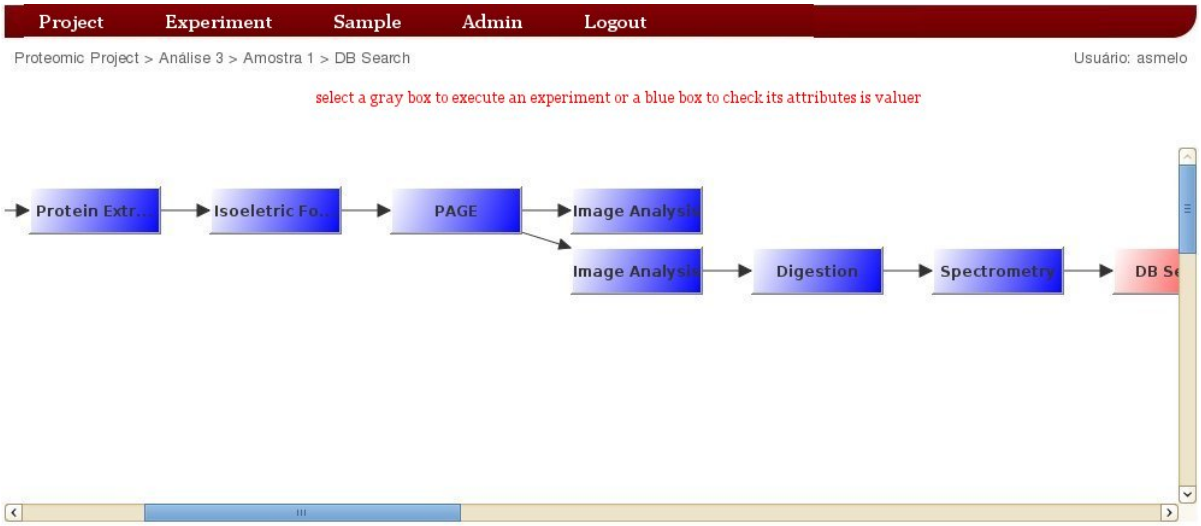


Figura 4.9. Análise onde foi preciso executar uma segunda Anáise de Imagem. Para isso foi iniciado um segundo processo

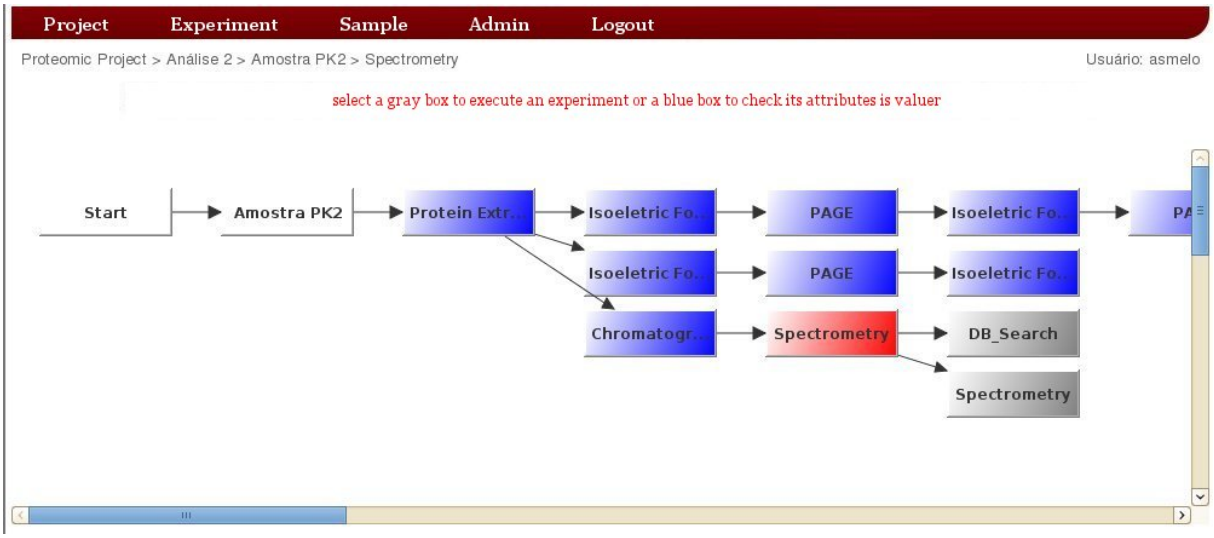


Figura 4.10. Análise onde três amostras foram analisadas a partir de uma mesma Extração de Proteína. Para isso foram iniciados três processos

casos como aqueles demonstrados na figuras 4.11 e 4.12.

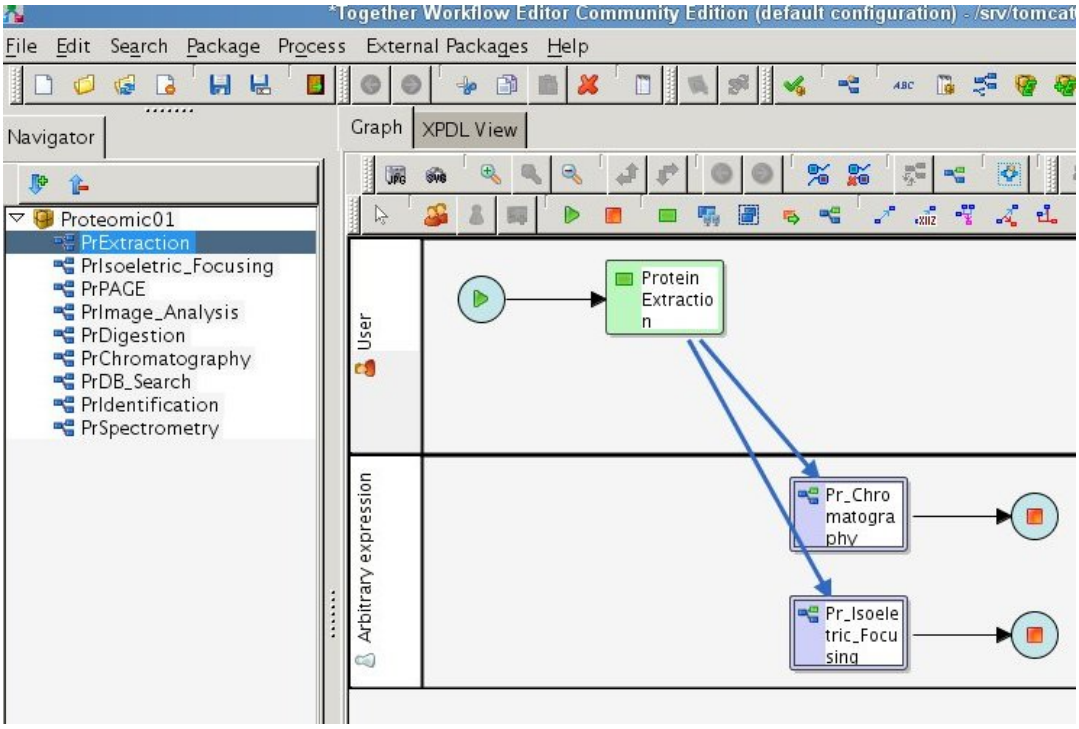


Figura 4.11. Definição do processo da Extração de Proteínas

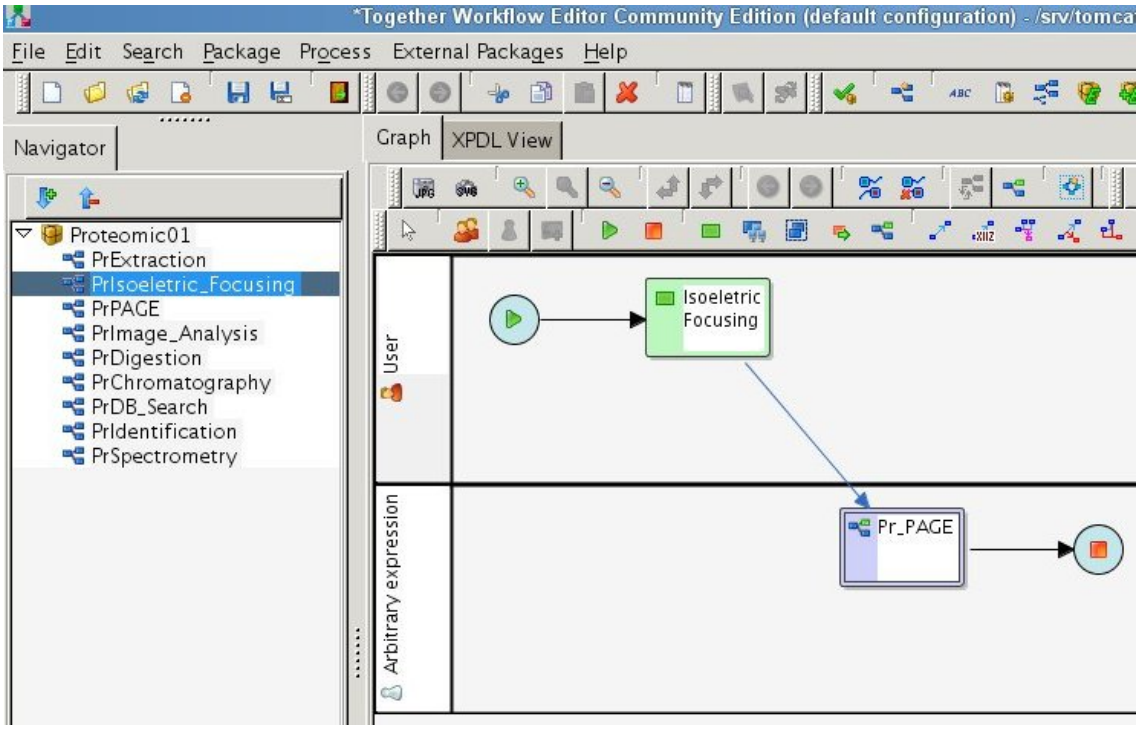


Figura 4.12. Definição do processo da Focalização Isoelétrica

Como dito, para o Shark um processo pode ser inicializado a qualquer momento. Contudo, ao se criar vários processos deve-se indentificar o processo que inicializará a execução do fluxo de trabalho. Para isso é definido um atributo estendido no pacote do fluxo de trabalho. Este atributo deverá ter o nome *First Activity* e o valor do atributo será o identificador da primeira atividade do fluxo de trabalho.

4.2 Execução do Fluxo de Trabalho

Uma vez que o fluxo de trabalho foi definido, durante a criação de um projeto, basta selecionar o arquivo XPDL que contém a definição do fluxo de trabalho desejado para executa-lo.

Para organizar os dados dos laboratórios o SIGLa possui a seguinte estrutura: Um projeto possui experimentos onde amostras são analisadas. Para iniciar a execução do fluxo de trabalho o primeiro passo é criar um projeto e seus experimentos. A Figura 4.13 mostra a interface para criar um projeto e a Figura 4.14 mostra a interface para criar um novo experimento. Por exemplo, um laboratório poderia criar um projeto para estudar certas doenças usando ratos como modelos de organismo. Esse projeto pode ter um experimento A para analisar ratos e um experimento B para analisar ratas. Em cada experimento seria executada a análise de duas amostras, uma amostra de ratos infectados e outra de ratos saudáveis. Cada grupo de amostra será analisado executando as atividades do worflow correspondente. Uma vez que o projeto e os experimentos foram criados, caso o usuário tenha permissão, o SIGLa também possibilita a edição e exclusão destes.

Após a seleção de um experimento, o SIGLa apresenta uma interface integrada (Figura 4.15), onde é possível gerenciar todo o fluxo de trabalho. Nesta interface pode-se visualizar as atividades de um determinado experimento. As atividades já executadas são representadas pela cor azul. No exemplo, a atividade *Chromatography* está selecionada e é representada pela cor vermelha, assim os valores dos seus atributos são exibidos na tabela abaixo do fluxo de trabalho. A atividade *Spectrometry* está em cinza pois indica que esta pode ser executada a partir do experimento *Chromatography* que está selecionado. Através desta interface é possível gerenciar múltiplas amostras, cada linha corresponde à análise de uma amostra. Esta interface também é responsável por uma característica muito importante do SIGLa: a consulta de dados. A representação gráficas das atividades facilita bastante a localização de uma atividade específica. Após a localização da atividade, com apenas um clique é possível recuperar todas as informações da atividade. Este processo de recuperação de dados é executado com bastante

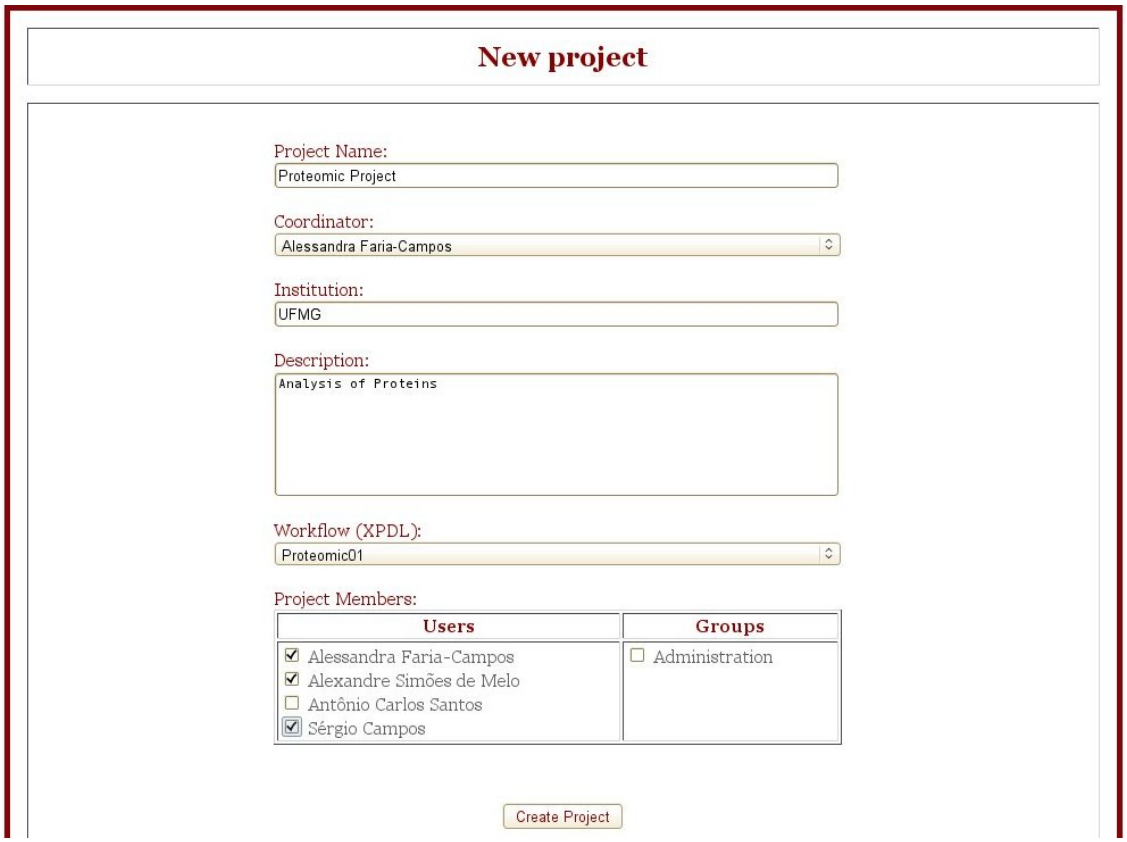


Figura 4.13. Interface para criar um novo projeto

Figura 4.14. Interface para criar um novo experimento

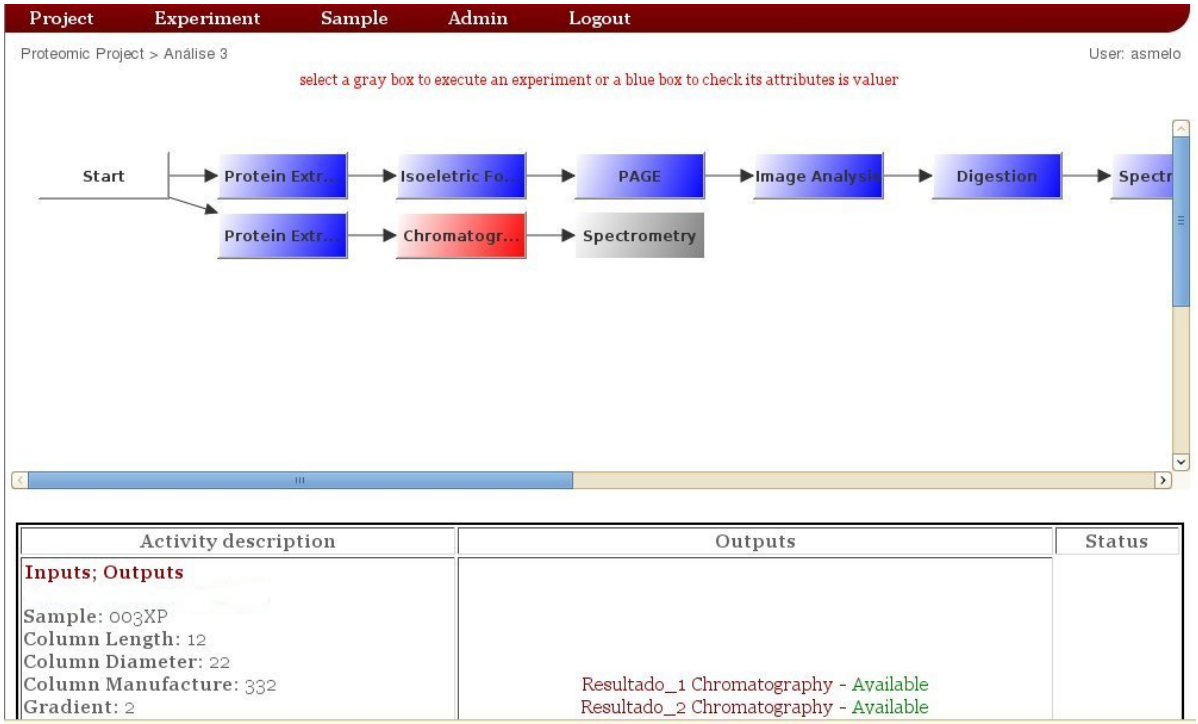


Figura 4.15. Execução do fluxo de trabalho. Através desta interface todo o fluxo de trabalho pode ser gerenciado. As caixas cinzas são as atividades disponíveis para execução. As caixas azuis são as atividades já executadas. A atividade selecionada é representada pela caixa vermelha, e seus atributos são exibidos na tabela abaixo do fluxo de trabalho.

frequencia nos laboratórios, assim é muito importante que seja um processo rápido e eficiente. Ao abrir um experimento o SIGLa instancia um novo fluxo de trabalho e a interface mostrada na Figura 4.16 é exibida. Nesta interface é disponibilizado para o usuário a execução da primeira atividade do fluxo de trabalho.

Para executar uma atividade o usuário precisa apenas clicar na atividade desejada (caixa cinza). Clicando nesta o SIGLa abre uma interface onde o usuário pode preencher todos os atributos da atividade. Estes atributos são os atributos definidos no fluxo de trabalho para esta determinada atividade, A Figura 4.17 mostra esta interface. O SIGLa possui outras caracteríticas importantes nesta interface, que é a validação dos atributos. Se o valor de algum atributo estiver inconsistente, o SIGLa apresenta uma mensagem de erro (Figura 4.18). Isto evita a inserção de dados inconsistentes no banco melhorando a sua qualidade. Para isso, ao concluir uma atividade o SIGLa confere o tipo de cada atributo e as suas propriedades, definidas no fluxo de trabalho, e verifica se os mesmos foram preenchidos corretamente. Também para evitar erros na entrada de dados, o SIGLa possibilita que o coordenador de um projeto possa definir valores para preencher os atributos das atividades. A Figura 4.17 mostra um exemplo desta

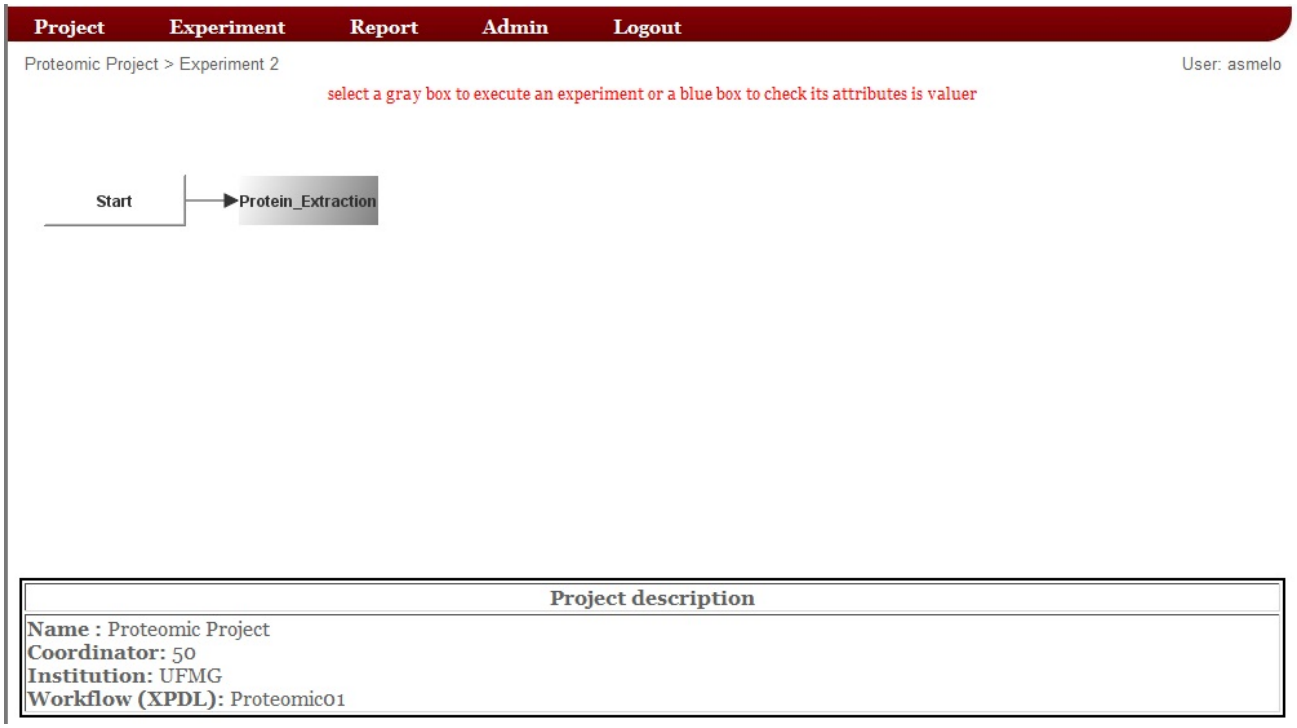


Figura 4.16. Inicialização de uma nova instância de um fluxo de trabalho. Para isso o SIGLa disponibiliza para execução a primeira atividade do fluxo de trabalho

propriedade do sistema: o atributo *Gas Pressure* é uma caixa de seleção com os valores definidos pelo coordenador. Usando esta propriedade o usuário não pode inserir valores aleatórios, mas apenas selecionar um valor pré-definido. Para definir os valores fixos de um atributo o SIGLa oferece a interface mostrada na Figura 4.19, onde basta selecionar a atividade em seguida o atributo desejado e então adicionar os valores fixos do atributo. Em qualquer momento o administrador pode excluir ou inserir novos valores fixos.

Quando uma atividade é definida para ter resultados, após a sua execução, o SIGLa abre a interface mostrada na Figura 4.20 para inserção dos resultado. Como na execução da atividade, na criação de um resultado o SIGLa realiza a validação dos atributos, melhorando a qualidade do resultado. A Figura mostra uma mensagem de erro informando que é necessário inserir pelo menos uma saída.

Quando uma atividade é definida para ter entradas, antes da execução da atividade o SIGLa exibe a interface da Figura 4.22 para seleção das entradas. As entradas de uma atividade correspondem às saídas da atividade anterior. Nesta interface o SIGLa valida o número de entradas que foram selecionadas.

Uma vez que as entradas e saídas foram definidas para uma atividade, estas podem ser acessadas através do link *Inputs* e *Outputs*, mostrado na Figura 4.15, acima

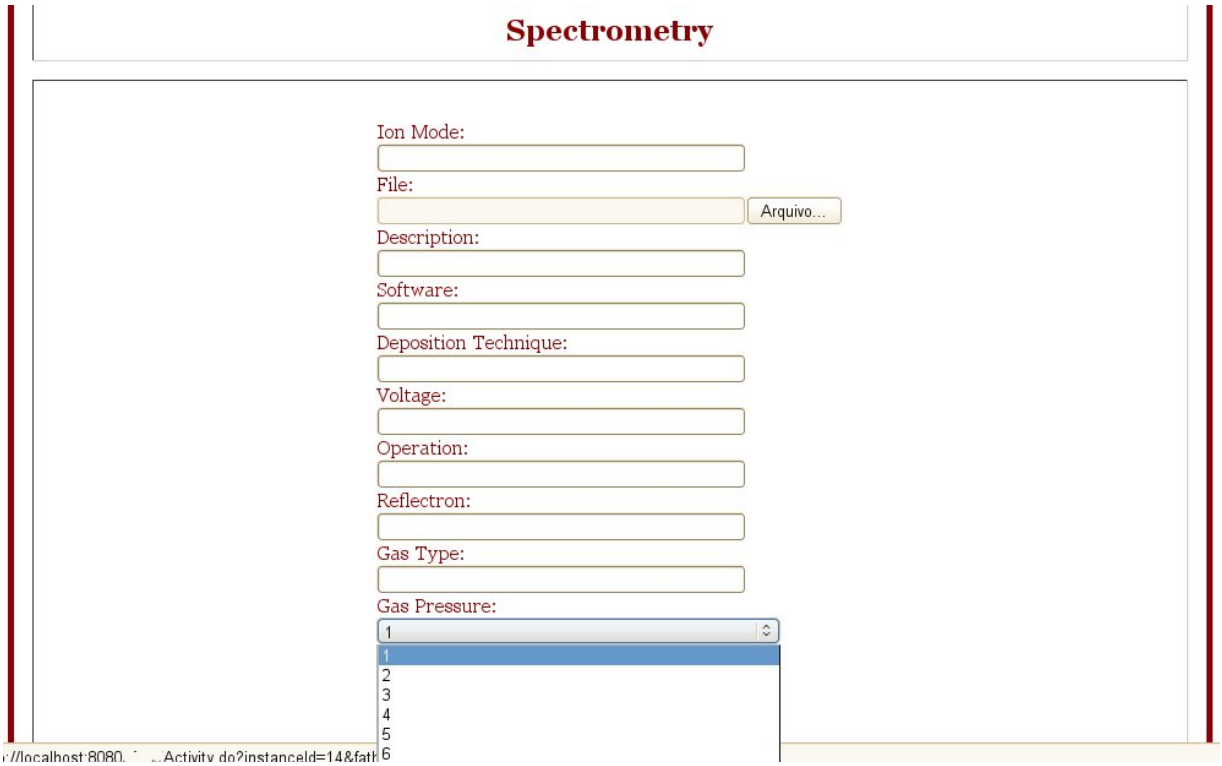


Figura 4.17. Execução da Atividade

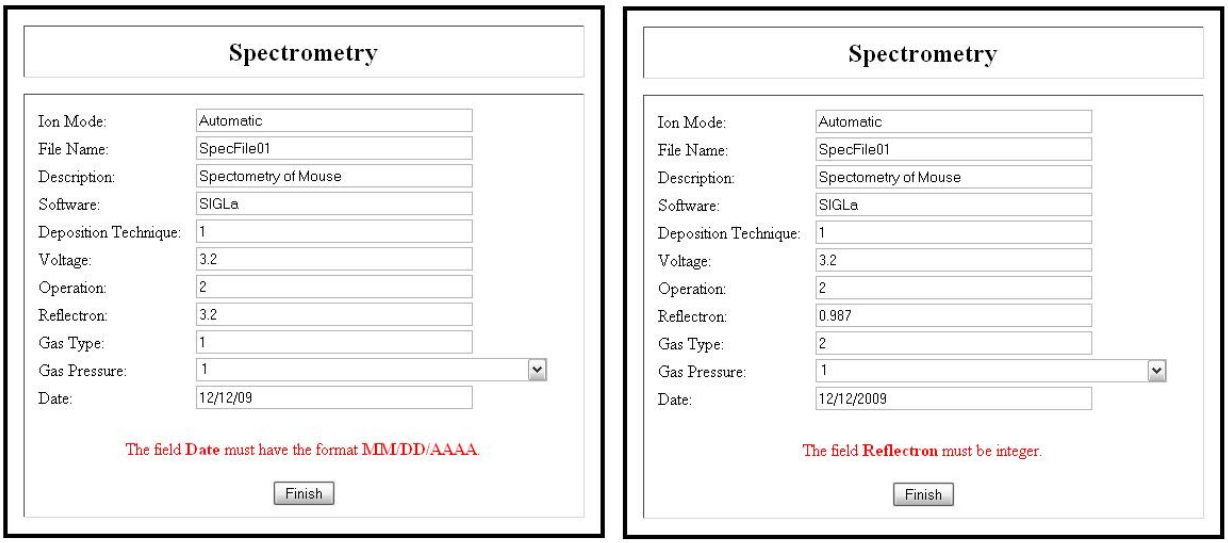


Figura 4.18. Validação dos atributos.

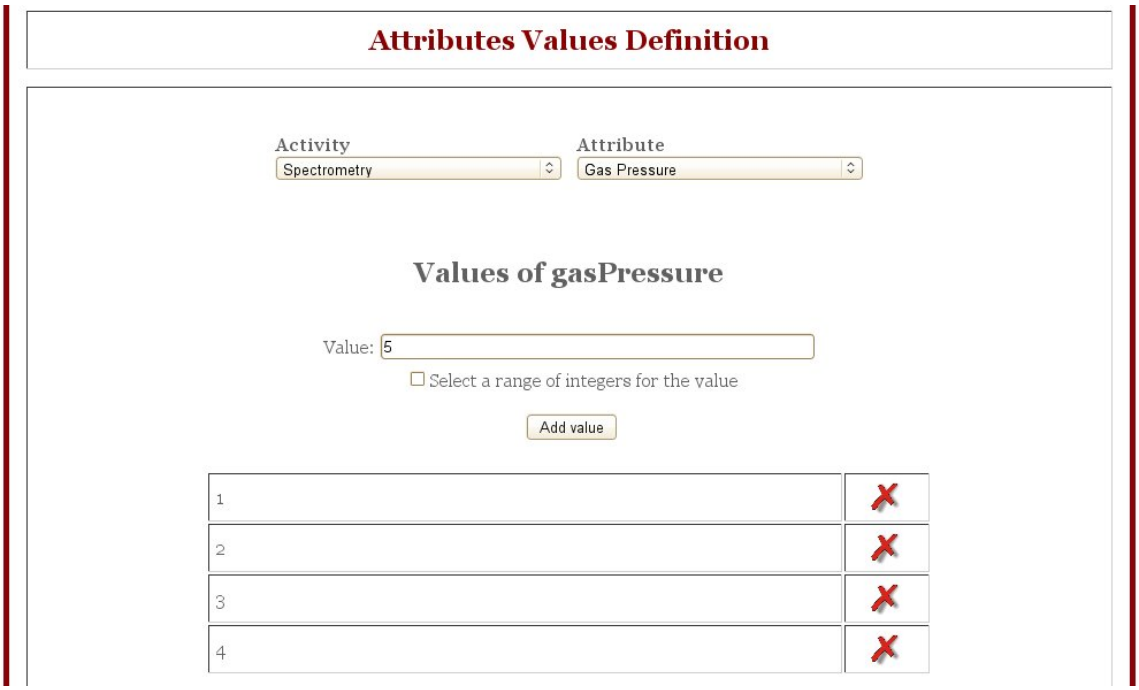


Figura 4.19. Definição de valores fixos dos atributos.

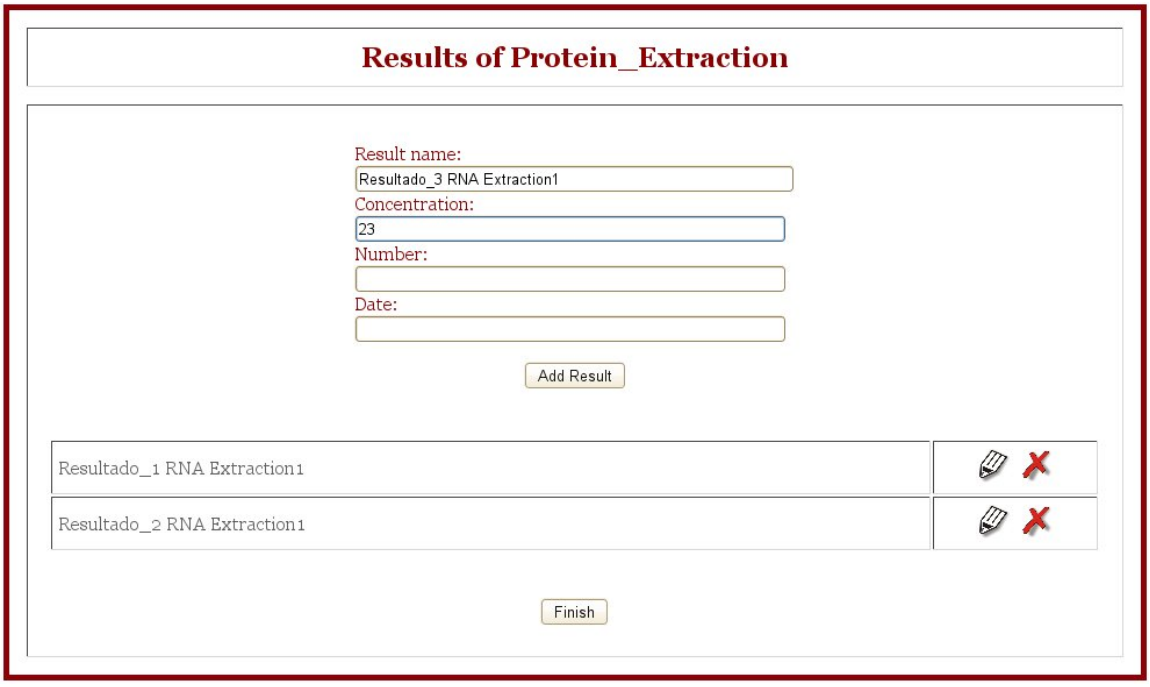


Figura 4.20. Definição de resultados. Nesta interface são listados os atributos do resultado definido na criação do fluxo de trabalho.

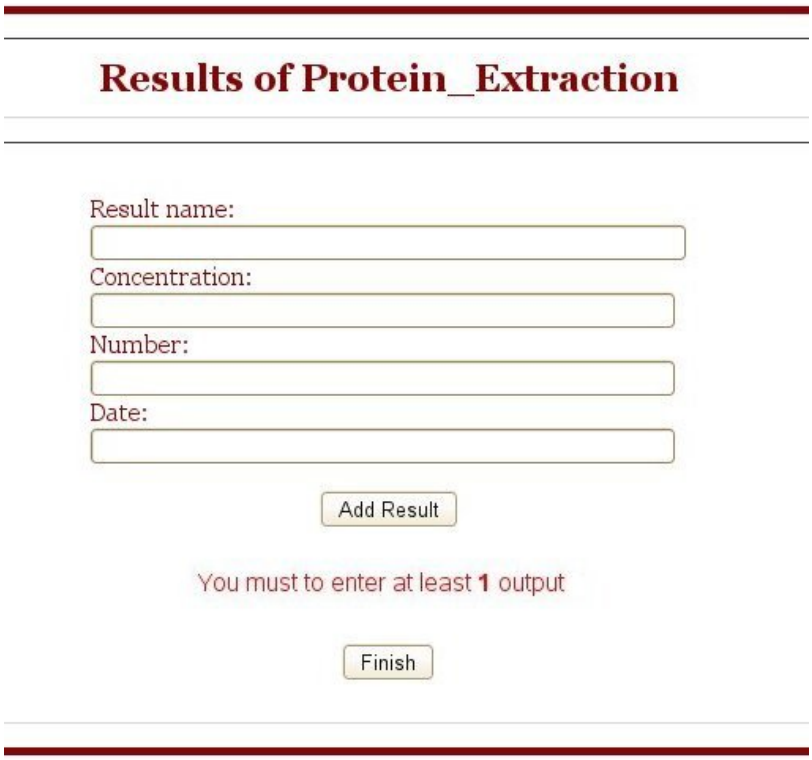


Figura 4.21. Validação do número de resultados definidos para uma atividade.

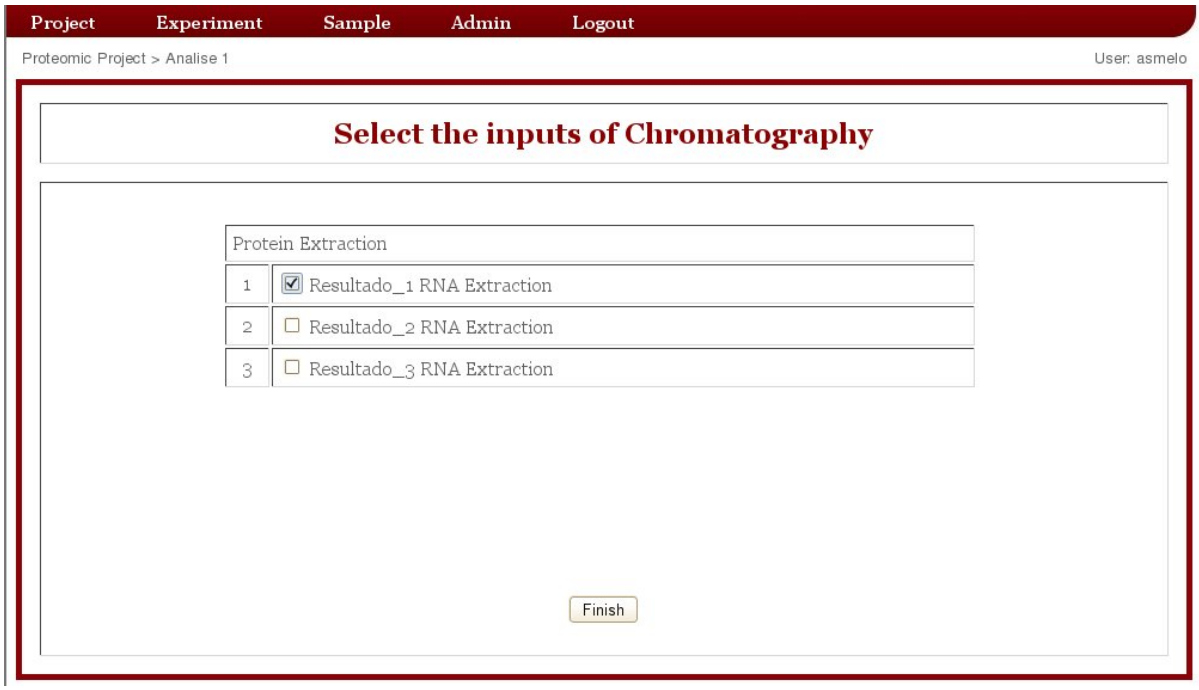


Figura 4.22. Definição de entradas. Nesta interface são listadas todas as saídas da atividade executada anteriormente.

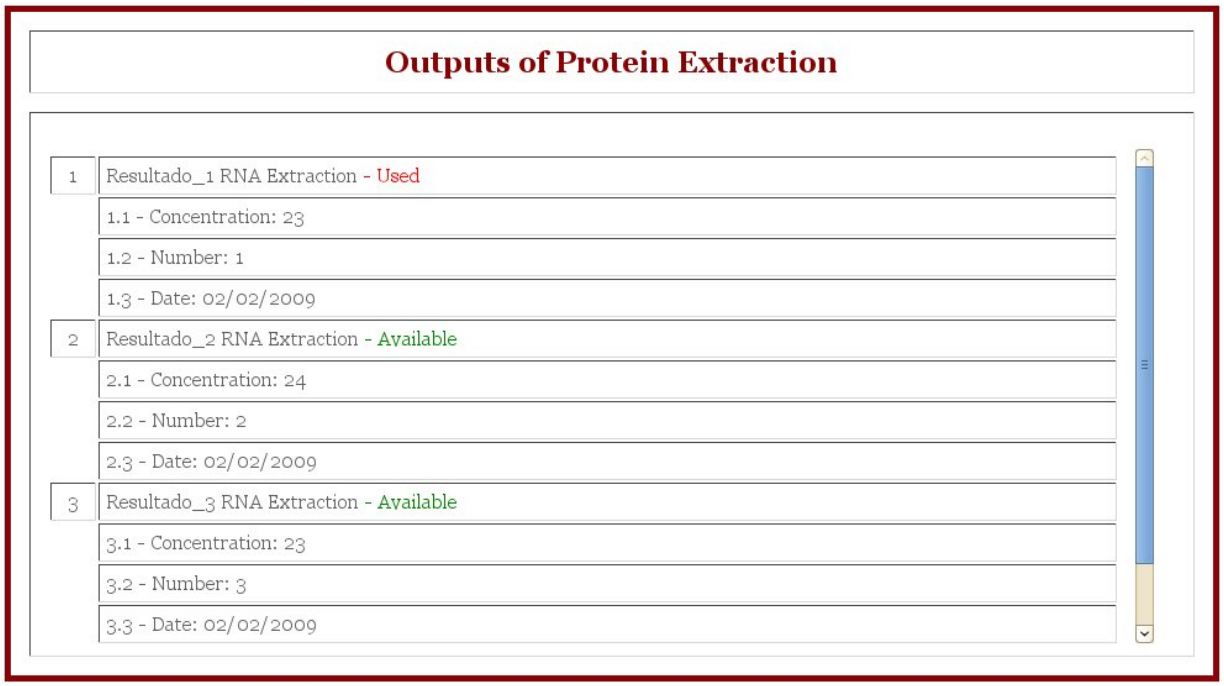
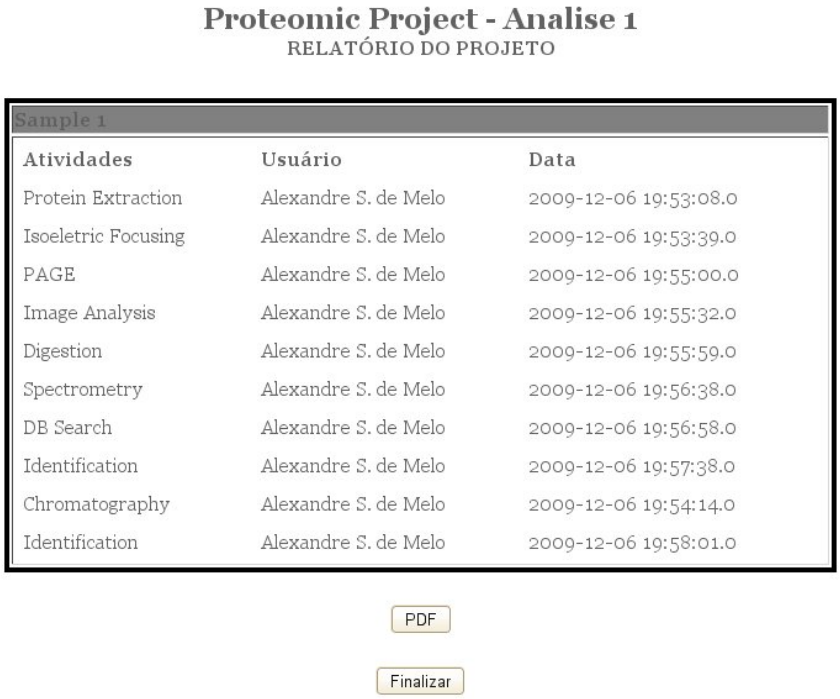


Figura 4.23. Visualização dos resultado de uma atividade.

dos atributos da atividade. Estes links irão listar as entradas ou saídas detalhando seus atributos. A Figura 4.23 mostra um exemplo desta interface. Nesta interface o SIGLa identifica se os resultado já foram utilizados como entrada de uma atividade ou não.

Para ajudar com o gerenciamento do laboratório o SIGLa oferece um relatório resumido, listando todas as atividades executadas em um determinado experimento. Neste relatório é possível verificar quando cada atividade foi executada e qual usuário a executou. O SIGLa também oferece um relatório completo que lista todas as atividade executadas em um experimento, detalhando os valores dos seus atributos. Para qualquer um dos relatórios o SIGLa possibilita exportá-los para o formato pdf. A Figura 4.24 mostra um exemplo de um relatório resumido, e a Figura 4.25 mostra um exemplo de um relatório completo.

Buscando melhorar a usabilidade do SIGLa foi implementado um menu para possibilitar o acesso a todas as funcionalidades do sistema a partir de todas as suas interfaces. A primeria opção do menu é a opção projeto que possibilita criar um novo projeto, abrir, editar ou excluir um projeto. Através desta opção também é possível definir os valores fixos de um projeto. A segunda opção é a opção Experimento, que permite criar um novo experimento, abrir, editar ou excluir um experimento. A terceira opção possibilita o acesso ao relatório simples e ao relatório completo. A quarta opção é a opção de Administração que permite gerenciar os usuários e os grupos de usuário.



Proteomic Project - Analise 1
RELATÓRIO DO PROJETO

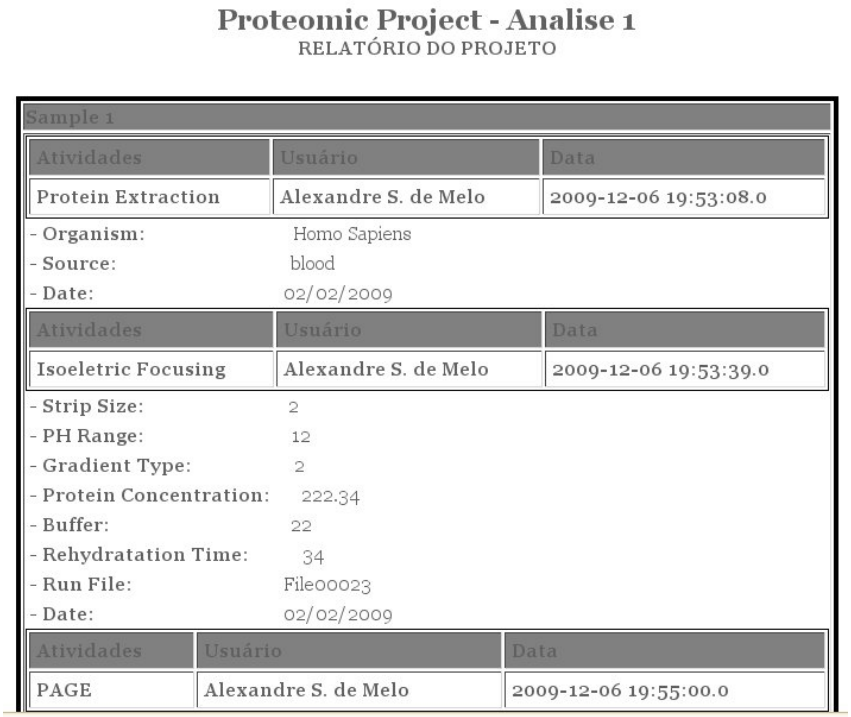
Sample 1

Atividades	Usuário	Data
Protein Extraction	Alexandre S. de Melo	2009-12-06 19:53:08.0
Isoeletric Focusing	Alexandre S. de Melo	2009-12-06 19:53:39.0
PAGE	Alexandre S. de Melo	2009-12-06 19:55:00.0
Image Analysis	Alexandre S. de Melo	2009-12-06 19:55:32.0
Digestion	Alexandre S. de Melo	2009-12-06 19:55:59.0
Spectrometry	Alexandre S. de Melo	2009-12-06 19:56:38.0
DB Search	Alexandre S. de Melo	2009-12-06 19:56:58.0
Identification	Alexandre S. de Melo	2009-12-06 19:57:38.0
Chromatography	Alexandre S. de Melo	2009-12-06 19:54:14.0
Identification	Alexandre S. de Melo	2009-12-06 19:58:01.0

PDF

Finalizar

Figura 4.24. Exemplo do relatório resumido gerado pelo SIGLa. São listadas todas as atividades de um experimento, identificando o momento da execução e o usuário responsável.



Proteomic Project - Analise 1
RELATÓRIO DO PROJETO

Sample 1

Atividades	Usuário	Data
Protein Extraction	Alexandre S. de Melo	2009-12-06 19:53:08.0

- **Organism:** Homo Sapiens
- **Source:** blood
- **Date:** 02/02/2009

Atividades	Usuário	Data
Isoeletric Focusing	Alexandre S. de Melo	2009-12-06 19:53:39.0

- **Strip Size:** 2
- **PH Range:** 12
- **Gradient Type:** 2
- **Protein Concentration:** 222.34
- **Buffer:** 22
- **Rehydratation Time:** 34
- **Run File:** File00023
- **Date:** 02/02/2009

Atividades	Usuário	Data
PAGE	Alexandre S. de Melo	2009-12-06 19:55:00.0

Figura 4.25. Exemplo do relatório completo gerado pelo SIGLa. São listadas todas as atividades de um experimento, detalhando os valores dos seus atributos.

A última opção do menu permite que usuário saia do sistema.

4.2.1 Mecanismo de Segurança

Além de melhorar a qualidade dos dados os LIMS precisam assegurar a privacidade dos dados. Entre os laboratórios é muito comum a necessidade de limitar o acesso aos dados, seja um laboratório acadêmico ou comercial. Normalmente na academia os LIMS armazenam dados que ainda estão sob pesquisa e não foram publicados, também os laboratório comerciais armazenam dados de grande valor e importância que não podem ser publicados.

Tendo em vista esta realidade, foi implementado no SIGLa um mecanismo de cadastro de usuários e controle de acesso aos dados. Antes de executar qualquer ação no SIGLa é preciso autenticar o usuário que está acessando com um login e senha. Mesmo que as páginas internas do SIGLa sejam acessadas diretamente, estas estão programadas para redirecionar para a tela de login, caso a autenticação de usuário ainda não tenha sido realizada. Assim, para utilizar o SIGLa, e então acessar seu banco de dados, é preciso se cadastrar no sistema. Apenas um usuário administrador do sistema pode cadastrar outros usuários. Uma vez feita a autenticação do usuário, todas as ações no sistema estarão associadas ao usuário logado. No relatório emitido pelo SIGLa pode-se verificar as atividades executadas, o momento em que foram executadas e qual usuário as executou.

Para isso o SIGLa oferece uma interface para definição de usuário e grupos de usuários. A Figura 4.26 mostra a interface de gerenciamento de grupos de usuários. Nesta interface é possível criar, editar e excluir os grupos de usuários. A Figura 4.27 mostra a interface de gerenciamento de usuários do sistema. Nesta interface também é possível criar, editar e excluir os usuários. O acesso a estas interfaces é dado apenas para o usuário que seja membro do grupo *Administration*.

O SIGLa também permite associar usuários aos projetos criados. Deste modo apenas os membros de cada projeto terão acesso ao mesmo. A Figura 4.28 mostra a interface de criação de projeto, abaixo pode-se ver a seleção dos membros dos projetos. É possível também selecionar um grupo de usuários ao invés de um usuário individualmente.

O SIGLa também proteje os dados do laboratório evitando que estes sejam excluídos ou modificados por qualquer usuário. Naturalmente, os experimentos e projetos criados podem ser modificados e excluídos, contudo apenas o administrador pode fazê-lo.

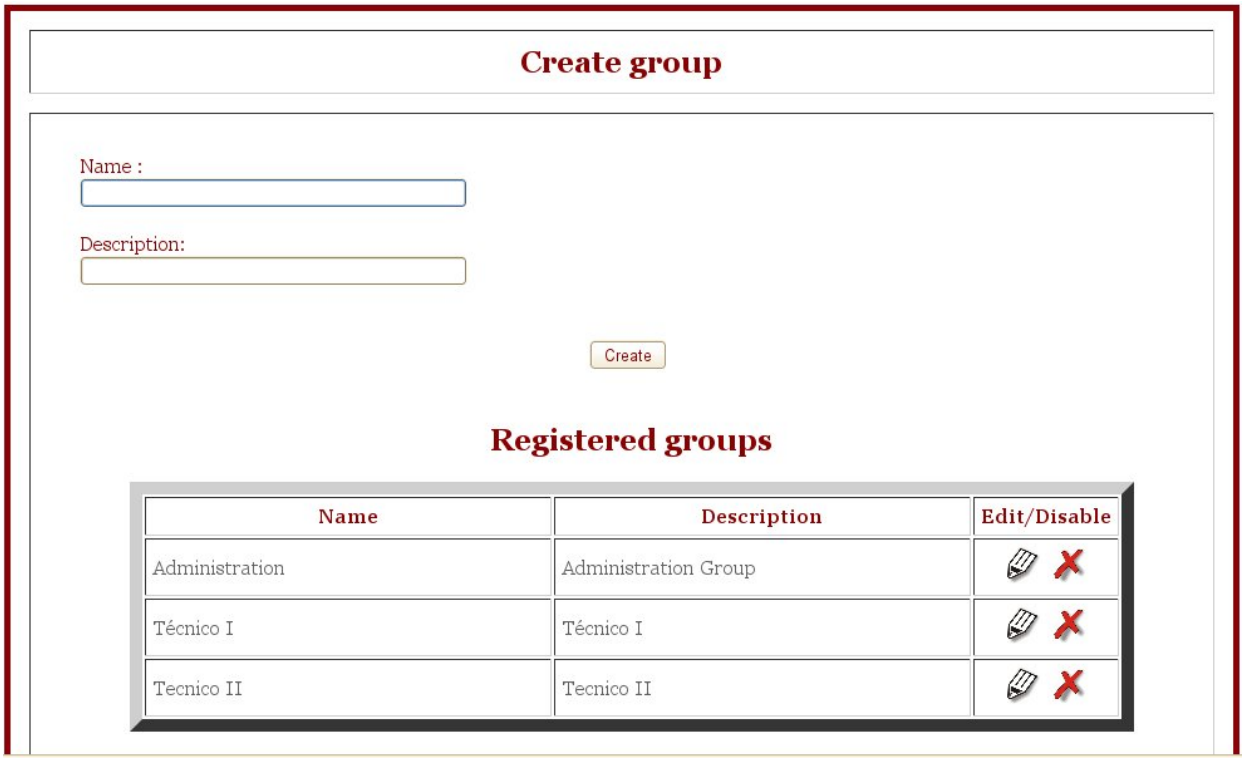


Figura 4.26. Interface de gerenciamento de grupos de usuários

Figura 4.27. Interface de gerenciamento de usuários

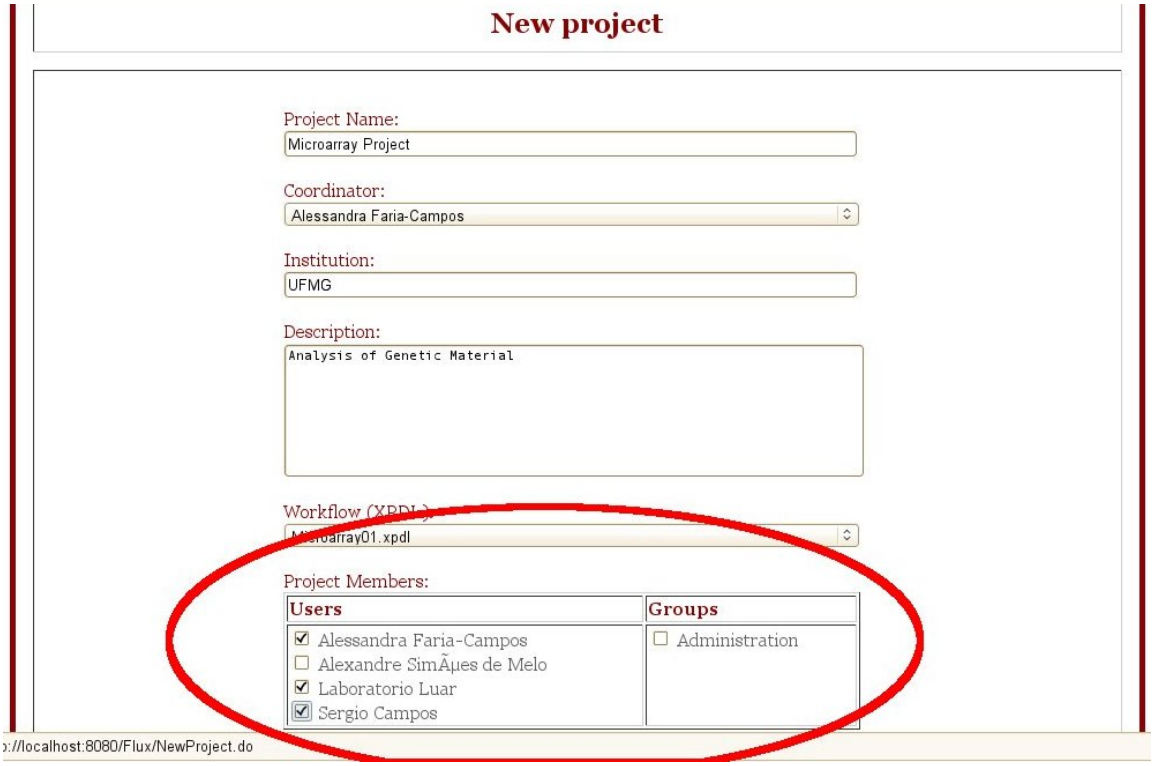


Figura 4.28. Seleção de membro de um projeto

Capítulo 5

Implementação

O SIGLa é uma aplicação Web com código aberto que possui uma interface gráfica robusta e complexa. Este foi desenvolvido com a linguagem de programação JAVA e o Tomcat 6.0 (Tomcat [2010]) como servidor Web. O seu desenvolvimento foi feito no Linux Open Suse 11.1 com o uso da IDE (`Integrated Development Environment`) Eclipse (Eclipse [2010]), contudo o SIGLa pode ser instalado em múltiplas plataformas e pode ser acessado por múltiplos browsers. Para a interface gráfica utilizamos Applets. O banco de dados foi desenvolvido com MySQL 5.1, por ser um banco estável e gratuito. Para definir um fluxo de trabalho, usamos o *Together Workflow Editor Community Edition* 2.4.1 que utiliza o padrão XPDL. O SIGLa possui o gerenciador de fluxo de trabalho *Enhydra Shark* 2.0.1 incorporado ao seu código. O Shark possui seu próprio banco de dados para armazenar as informações das execuções dos fluxos de trabalho.

5.1 Estrutura de Classes

O SIGLa foi desenvolvido baseado na arquitetura cliente-servidor, sendo assim, para garantir a eficiência no acesso aos dados, foi adotado o padrão de desenvolvimento *Data Transfer Objects* (DTO). Continuamente os clientes fazem inúmeras requisições ao servidor, sobrecarregando o seu processamento e gerando um grande tráfego na rede. Utilizando um objeto de transferência de dados (DTO) os dados do negócio são encapsulados. Para cada objeto é criado um DTO com os atributos do objeto. Quando o cliente realiza uma requisição ao servidor, este constroi um DTO e o preenche com os valores dos seus atributos e o passa para o cliente. Assim, o cliente faz apenas uma única chamada remota de método ao servidor que lhe retornar o DTO, ao invés de fazer inúmeras chamadas remotas de métodos para recuperar os valores dos atributos individualmente. Através do DTO o cliente pode acessar os valores de cada atributo,

uma vez que seus atributos são públicos. Dado que os DTOs são passados por valor para o cliente, todas as chamadas às instâncias de DTOs são locais, ao invés de invocações de métodos remotos.

Para obter um código organizado o SIGLa foi dividido em três camadas:

- Camada de Interface: Formada por classes que contém o código das páginas do SIGLa. Estas classes são Java Server Pages (jsp) que são compostas por código HTML e o mínimo de código JAVA e JavaScript.
- Camada de Negócio: Onde são declarados os objetos da aplicação. Os métodos dos objetos realizam o acesso aos dados no banco.
- Camada de Comunicação: Formada pelas classes *servlet* que processam requisiçoes e respostas do servidor web. As *servlets* são responsáveis por fazer a comunicação entre a camanda de Interface e a camada de Negócio. Cada jsp possui uma servlet associada.

A Figura 5.1 mostra o diagrama das classes da camada de negócio construída durante o desenho do projeto.

5.2 Sistema de Gerenciamento de fluxo de trabalho

Inicialmente, esperava-se que a biblioteca *Enhydra Shark* executasse facilmente todas as operações de gerenciameto de fluxos de trabalho. Contudo, à medido em que o SIGLa foi sendo desenvolvido o Shark apresentou várias limitações. Primeiramente, a documentação disponível é pouca e muito superficial. Já este fato dificultou bastante o processo de desenvolvimento do SIGLa. Além disso muitas das operações de gerenciamento de fluxos de trabalho não estão presente na versão *Open Source* do Shark, que foi utilizada neste trabalho. Uma vez que a nossa intenção é construir um sistema *Open Source*, não foi uma opção adquirir os módulos adicionais do Shark que são vendidos pela *Together Teamlösungen*. Sendo assim, foi desenvolvido no próprio código do SIGLa diversas funcionalidades para o gerenciamento de fluxos de trabalho, a fim de complementar a biblioteca *Open Source Enhydra Shark*. A maior parte destas funcionalidades consiste em um *parser* no arquivo XPDL para recuperar informações específicas sobre o fluxo de trabalho carregado. O arquivo XPDL é composto por um conjunto de marcadores (*tags*) predefinos pela WfMC. Uma vez que a estrutura do arquivo XPDL é conhecida, com a criação de *parsers*, é possível localizar automaticamente as informações contidas no XPDL. Para construção destes *parsers* foi utilizada a API DOM (DOM [2010]).

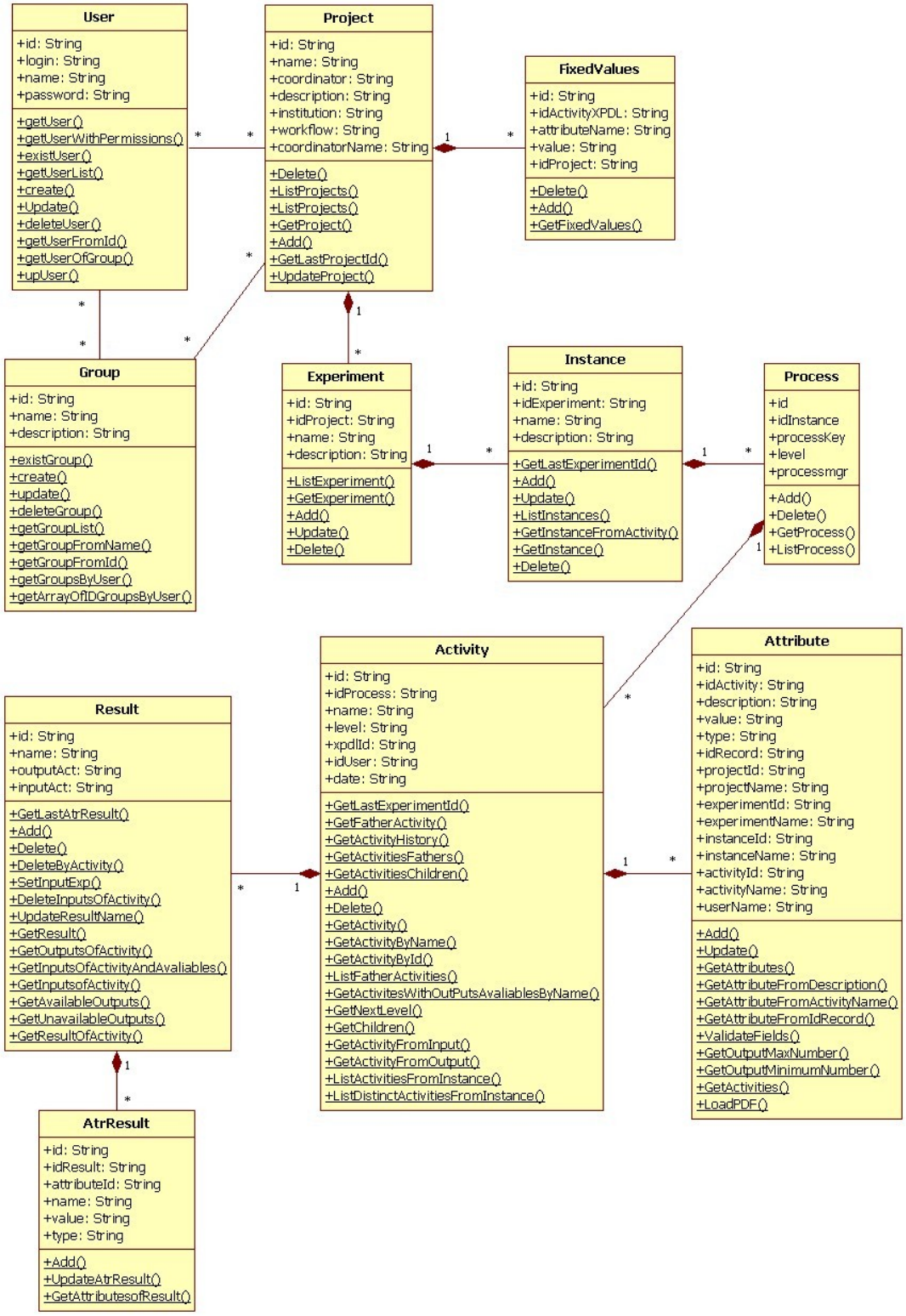


Figura 5.1. Diagrama de Classes do SIGLa

A API DOM fornece uma interface independente de plataforma e linguagem para estruturar o conteúdo de documentos XML. O DOM descreve uma linguagem neutra capaz de representar qualquer documento XML em forma de uma árvore e tratar a informação armazenada nesses documentos como um modelo de objetos hierárquicos. Ou seja, o DOM possibilita estruturar os marcadores do arquivo XPDL que o sistema irá analisar, facilitando assim este processo. Uma vez que os marcadores do XPDL estão mapeadas no objeto do tipo *Document*, da API DOM, através de uma simples chamada de método, é possível efetuar buscas de marcadores, filtrar um determinado maracador, ou um conjunto destes, excluir marcadores ou adiciona-los.

Para complementar a biblioteca *Open Source Enhydra Shark* foram criados os seguintes *parsers*:

- ***getFirstActivity*** - Definido para identificar, no arquivo XPDL, a primeira atividade do fluxo de trabalho. Como descrito na sessão 3.1.1, para possibilitar a existencia de subprocessos nos fluxos de trabalho do SIGLa, foi preciso criar um processo para cada atividade, durante a definição do fluxo de trabalho. De acordo com a especificação da WfMC os processos podem ser inicializados a qualquer momento. Contudo, para se adequar aos processos dos laboratórios, os fluxo de trabalho definidos para o SIGLa possuem vários processos, mas o fluxo de trabalho é inicializado através de uma única atividade pertencente a um determinado processo. Para definir a primeira atividade do fluxo de trabalho, durante a criação do fluxo de trabalho, é criado o atributo estendido *First Activity*, no pacote do fluxo de trabalho, identificando a primeira atividade que deverá ser executada. Assim, o *parser getFirstActivity* irá navegar nos marcadores do arquivo XPDL para recuperar o valor do atributo estendido *First Activity*, retornando assim a primeira atividade do fluxo de trabalho.

- ***getFirstProcess*** - Pela mesma justificativa do *getFirstActivity*, este *parser* foi definido para identificar, no arquivo XPDL, o primeiro processo do fluxo de trabalho. Para isto o *getFirstProcess* utiliza o *getFirstActivity* para identificar a primeira atividade do fluxo de trabalho, em seguida o *parser* verifica a qual processo a atividade pertence.

- ***getNextActivities*** - Definido para retornar as próximas atividades que podem ser executadas após uma determinada atividade. De acordo com a especificação da WfMC caso uma atividade possua duas transições saindo para duas, ou mais, atividades distintas, estas transições devem possuir condições que determinarão, exclusivamente, qual a próxima atividade a ser executada. Assim, a biblioteca

Shark somente retorna a próxima atividade a ser executada quando a condição de transição já foi resolvida. No caso do SIGLa esta condição não é necessária, pois o usuário pode escolher executar qualquer uma das próximas atividades. Por exemplo, no laboratório de proteômica, após a execução da atividade *Protein Extraction* o usuário pode escolher em executar a atividade *Chromatography* ou a *Isoeletric Focusing*, a depender da sua necessidade. Ou seja, ao finalizar uma atividade, é necessário retornar para o usuário as próximas atividades disponíveis, independente das condições das transições. Deste modo, o *parser getNextActivities* irá localizar no arquivo XPDL a atividade executada e retorna então as atividades de destino das transições da atividade executada.

- ***getNumberOfTransitions*** - Definido para retornar o número de transições que saem de uma determinada atividade. Este *parser* foi definido especificamente para verificar se uma determinada atividade é a última do fluxo de trabalho. Caso a atividade não possua transições, saindo dela, significa então que é a última atividade. Apesar do Shark disponibilizar formas de verificar se uma atividade é a última do fluxo de trabalho, o *parser getNumberOfTransitions* foi definido por possuir menos parâmetros na sua chamada, tornando assim o seu uso mais simples e mais claro que o comando da bibioteca Shark.

- ***getAttributes*** - Definido para retornar os atributos de uma atividade. De acordo com o padrão XPDL as variáveis são definidas para um processo. Contudo, para definir os fluxos de trabalho dos laboratórios foi necessário definir um conjunto de atributos (variáveis) por atividade, não por processo. Assim, durante a definição do fluxo de trabalho, são criados atributos estendidos, nas atividades, referenciando um grupo de variáveis do processo que correspondem aos atributos da atividade. O *Open Source Enhydra Shark* não manipula os atributos estendidos. Para isso é preciso comprar um módulo adicional para realizar tal manipulação. Assim, foi criado o *parser getAttributes* que analisa o arquivo XPDL para identificar os atributos estendidos de uma determinada atividade, em seguida, nos dados do processo correspondente, são recuperados os tipos e as propriedades dos atributos da atividade.

- ***getAttributesofResult*** - Definido para retornar os atributos dos resultados de uma atividade. Pela mesma justificativa do *parser getAttributes*, os atributos dos resultados de uma atividade são definidos como atributos estendidos da atividade correspondente e são marcados como atributos dos resultados. Assim, o *parser getAttributesofResult* analisa o arquivo XPDL para identificar os atribu-

tos estendidos de uma determinada atividade e recupera os atributos que foram marcados como sendo do tipo atributos dos resultado. Em seguida, nos dados do processo correspondente, são recuperados os tipos e as propriedades dos atributos dos resultados da atividade.

- ***getActivityName*** - Definido para retornar o nome de uma atividade a partir do seu identificador. Em algumas rotinas como o *parser getNextActivities* é retornado apenas o identificador da atividade, a fim de evitar conflitos caso duas atividades possuam o mesmo nome. No caso dos identificadores, estes não podem ser repetidos. Durante a definição de um fluxo de trabalho o *Together Workflow Editor* proíbe que sejam definidos identificadores idênticos, garantindo assim que eles sejam únicos. Assim, o *parser getActivityName* localiza no arquivo XPDL a atividade que possui um determinado identificador e retorna o seu nome.

- ***getAttributeName*** - Definido para retornar o nome de um atributo a partir do seu identificador. Pela mesma justificativa do *parser getActivityName* é necessário recuperar o nome de um atributo a partir do seu identificador. Assim o *parser getAttributeName* localiza no arquivo XPDL o atributo que possui um determinado identificador e retorna o seu nome.

- ***getPackageID*** - Definido para retornar o identificador de um fluxo de trabalho (um fluxo de trabalho pode ser visto como um pacote). A biblioteca Shark permite carregar fluxos de trabalho, contudo, caso este fluxo de trabalho já tenha sido carregado anteriormente o Shark lança uma exceção indefinida, que pode significar que o XPDL já foi carregado ou que o XPDL possui uma sintaxe errada. No SIGLa, caso um fluxo de trabalho seja carregado pela segunda vez, é informado para o usuário que o fluxo de trabalho já foi carregado, e o processo de carregamento do fluxo de trabalho não é executado, mas para isso é preciso saber se o erro lançado pelo Shark foi realmente devido ao caso em que o fluxo de trabalho já existe. O Shark possui um método que possibilita verificar se um fluxo de trabalho já foi carregado, para isso, deve-se passar como parâmetro o identificador do fluxo de trabalho que pretende-se carregar. Assim, para executar esta validação, o *parser getPackageID* recebe como parâmetro o arquivo XPDL que será carregado e retorna seu identificador.

- ***getActivities*** - Definido para retornar todas as atividades de um fluxo de trabalho. Na interface de definição de valores fixos é necessário listar todas as atividades do fluxo de trabalho para que o usuário possa definir os valores fi-

xos dos seus atributos. Esta rotina não é oferecida pelo Shark, assim o *parser getActivities* percorre o arquivo XPDL e retorna todas as atividades definidas.

De acordo com os *parsers* listados acima, percebe-se que a biblioteca *Open Source Enhydra Shark* mostrou-se bastante limitada para a criação de LIMS adaptáveis a múltiplos laboratórios, como o SIGLa. Contudo, ainda assim o Shark contribuiu para o desenvolvimento do SIGLa. As principais rotinas executas pelo Shark no SIGLa foram:

- Carregar um novo fluxo de trabalho.
- Listar os fluxos de trabalho já carregados.
- Gerenciar as variáveis definidas em um fluxo de trabalho.
- Inicializar e finalizar as atividades.
- Retornar o histórico e a ordem das atividades executadas.

Com exceção destas rotinas foi preciso implementar no SIGLa todas as demais funcionalidades do seu sistema de gerenciamento de fluxos de trabalho.

5.3 Funcionalidades do SIGLa

No SIGLa foram desenvolvidas as seguintes funcionalidades:

5.3.1 Criar/Editar Projeto

Nesta funcionalidade foi construída uma interface para obter os dados do projeto que são: nome, coordenador, instituição e descrição. O SIGLa valida se todos os campos foram preenchidos e armazena os dados do projeto no banco de dados do SIGLa. Na criação de um projeto também é definido o fluxo de trabalho a ser utilizado. Para isso, através da biblioteca Shark, são listados os fluxos de trabalho já carregados e é dada a opção de carregar um novo. Para carregar um novo fluxo de trabalho foi utilizado o método *OpenPackage* da biblioteca Shark. Este irá validar o arquivo XPDL e persistir os dados do fluxo de trabalho no banco de dados do Shark. Além do fluxo de trabalho, nesta interface são definidos os membros e grupos do projeto. Para isso são listados todos os usuário e grupos cadastrados do SIGLa. Quando o projeto é criado, o SIGLa faz a associação, no seu banco de dados, do projeto com os usuários e grupos

selecionados. Caso algum dos campos não tenha sido preenchido ou o arquivo XPDL seja inválido, o SIGLa gera uma mensagem de erro alertando o usuário.

A interface de edição de um projeto é similar à interface de criação, a única diferença é que na interface de edição o SIGLa não possibilita modificar o fluxo de trabalho utilizado. Modificar o fluxo de trabalho utilizado no meio de um projeto em execução iria tornar seus dados incosistentes. Assim, para executar um novo fluxo de trabalho é preciso criar um novo projeto. Para garantir a segurança dos dados o acesso ao link de edição de um projeto é exibido apenas se o usuário logado for membro do grupo *Administration*.

5.3.2 Abrir/Excluir Projeto

Após a tela de login o SIGLa abre a interface que lista os projetos cadastrados no sistema que o usuário logado seja membro. Nesta interface existem links para abrir os projeto, editá-los, definir seus atributos fixos e excluí-los. Antes de excluir o sistema exibi uma mensagem de confirmação para o usuário. Contudo, o link de exclusão de um projeto é exibido apenas se o usuário logado for membro do grupo *Administration*.

5.3.3 Gerenciamento de Grupos de Usuários

Ao criar um projeto deve-se selecionar todos os usuários que farão parte do projeto. Existem casos em que a lista de usuários é extensa, assim são criados grupos de usuários para facilitar este processo. Para isso o SIGLa possui um interface para definição de grupos. Nesta interface o SIGLa obtem o nome e a descrição do grupo. Uma vez que são preenchidos o SIGLa valida se os campos foram preenchidos e armazena os dados do grupo no banco de dados. À medida que os grupos são criados o SIGLa lista em uma tabela os grupos existentes. Nesta tabela foram definidos links para editar e excluir os grupos criados. A edição dos grupos funciona de forma similar: o sigla carrega os dados no grupo na interface, valida os campos e atualiza os dados no banco. Para possibilitar um controle dos grupos cadastrados no SIGLa, este permite que apenas os usuário membros do grupo *Administration* tenha acesso ao link de cadastro de grupos.

5.3.4 Gerenciamento de Usuários

Por questões de segurança, para todas as atividade executadas o SIGLa registra o usuário que as executou. Assim, para acessar o sistema é preciso possui um login e senha. Para isso foi definida no SIGLa uma interface para gerenciamento de usuários. Nesta interface o SIGLa obtém os dados do usuário que são: nome, login, senha e

confirmação de senha. Antes de persistir os dados do usuário no banco o SIGLa valida se os campos senha e confirmação de senha possuem o mesmo valor. A senha do usuário é criptografada no banco de dado. Para isso foi utilizado o MD5 no MySQL. Além dos dados do usuário o SIGLa lista nesta interface os grupos cadastros no sistema, possibilitando assim que o usuário defina a quais grupos participa. Para possibilitar um controle dos usuário cadastrados no SIGLa, este permite que apenas os usuário membros do grupo *Administration* tenha acesso ao link de cadastro de usuários.

5.3.5 Abrir/Excluir Experimento

Após abrir um projeto o SIGLa abre uma interface listando todos os experimentos do projeto selecionado. Nesta interface foram definidos links para criar um novo experimento, abrir um experimento existente, editá-lo e excluí-lo. Para garantir a segurança dos dados apenas os administradores do sistema podem excluir um experimento, assim, antes de excluir um experimento o SIGLa verifica se o usuário logado pertence ao grupo *Administration*.

5.3.6 Criar/Editar Experimento

Após a criação de um projeto o usuário deve criar um novo experimento. A interface para criação de um novo experimento obtem o nome e a descrição do experimento. Nesta interface o SIGLa valida se todos os campos foram preenchidos e persiste os dados do novo experimento no seu banco de dados, associando o experimento ao seu projeto correspondente. A interface de edição funciona de forma similar carregando os dados do experimento selecionado, validando as alterações e realizando a atualização no banco de dados.

5.3.7 Definição de Valores Fixos

A definição de valores fixos consiste em cadastrar valores pré-definidos para preencher os atributos das atividades. Na interface onde são listados os projetos cadastrados o SIGLa exibe um link para definir os valores fixos de cada projeto. Ao clicar neste link o SIGLa exibe uma interface com um *dropdown list* listando todas as atividades definidas no fluxo de trabalho do projeto selecionado. Para recuperar esta lista de atividades o SIGLa utiliza o *parser getActivities* que acessa o arquivo XPDL diretamente. Quando o usuário seleciona uma das atividades o SIGLa exibe um segundo *dropdown list* listando os atributos da atividade selecionada. Para recuperar os atributos o SIGLa utiliza o *parser getAttributes*. Uma vez que o atributo é selecionado o SIGLa exibe uma

caixa de texto para que o usuário possa cadastrar o valor fixo do atributo. À medida que os valores são cadastrados eles são associados, no banco de dados, ao projeto selecionado e são listados numa tabela abaixo da caixa de texto, para que o usuário possa verificar os valores já cadastrados. Nesta tabela foi implementado um link para excluir os valores cadastrados. Nesta interface o SIGLa oferece a opção de cadastrar uma faixa de números de uma só vez. Para este caso é exibida na interface um caixa de texto para receber o valor inicial e uma segunda caixa para receber o valor final. Esta funcionalidade pode ser utilizada para definir, por exemplo, os valores fixos do atributo pH, que correponde à faixa de 0 a 14. O usuário pode inserir cada um dos valores individualmente, mas com esta funcionalidade a definição torna-se muito mais rápida.

5.3.8 Gerenciamento da execução de um fluxo de trabalho

Após abrir um experimento o SIGLa exibe a interface de gerenciamento de fluxo de trabalho onde pode-se executar novas atividades e verificar as atividades já executas. O histórico das atividades executadas é representado em um *applet*, em forma gráfica, para facilitar a visualização do usuário. Na declaração do *applet* é passado como parâmetro as atividades já executadas e a ordem de execução. Para cada uma dessas o applet gera a representação da atividade e suas transições em um espaço ainda vazio no gráfico. A lista de atividades executadas é obtida através da biblioteca Shark, que possui o controle de todas as atividades que foram inicializadas e finalizadas.

Ao clicar em uma das atividades executadas o SIGLa exibe no *applet* as próximas atividades que podem ser executadas após a atividade selecionada. Para isso, no declaração do *applet*, além das atividades executadas, também é passado como parámetro as próximas atividades da atividade selecionada. Para cada uma dessas próximas atividades, o *applet* irá gerar a sua representação com uma cor diferenciada, para identificar que trata-se de uma atividade que ainda não foi executada. Para recuperar a lista de atividades que podem ser executadas após a atividade selecionada o SIGLa utiliza o *parser getNextActivities*, como descrito na sessão 4.2.

Ao clicar em uma das atividades executadas, representadas no *applet*, o SIGLa também recupera do seu banco de dados os valores dos atributos definidos para a atividade selecionada e os exibe em uma tabela abaixo do *applet*. Além dos atributos da atividade o SIGLa recupera os resultados gerados pela atividade, que também são listados na tabela. Junto aos atributos da atividade o SIGLa exibe também um link que dará acesso à lista de entradas da atividade, caso esta possua.

5.3.9 Executar Atividade

Ao clicar no *applet* para execução de uma nova atividade o SIGLa exibe um formulário com os atributos da atividade selecionada para serem preenchidos. Para recuperar a lista de atributos o SIGLa utiliza o *parser getAttributes* que irá recuperá-los diretamente do arquivo XPDL. Junto com a lista de atributos o *parser getAttributes* retorna o tipo e as propriedades de cada um dos atributos. O SIGLa utiliza estas informações para validar o preenchimento de cada um dos campos antes de persistir a atividade. Para validação dos atributos foram definidos 5 rotinas, são elas:

- ***Validação de Inteiros*** - Caso o atributo seja do tipo Inteiro, esta rotina irá verificar se todos os dígitos atribuídos ao campo são números.

- ***Validação de Números Reais*** - Caso o atributo seja do tipo Real, esta rotina irá verificar se todos os dígitos atribuídos ao campos são números, vírgula ou ponto.

- ***Validação da propriedade not null*** - Caso o atributo possua a proriedade *not null*, esta rotina irá verificar se foi atribuído algum dígito ao campo.

- ***Validação da propiedade format*** - Caso o atributo possua a propriedade *format*, esta rotina recebe como parâmetro o formato atribuído ao campo e realiza a validação do campo. O formato de um atributo pode ser #.##, por exemplo, neste caso a rotina irá verificar se o campo foi preenchido com 4 dígitos, se o segundo dígito é um ponto e se os demais dígitos são números.

- ***Validação de Datas*** - Caso o atributo seja do tipo Data, esta rotina utiliza a validação da propriedade *format* utilizando o formato ##/##/####. Uma vez que o formato esteja correto esta rotina valida se o campo foi preenchido com uma data válida.

Além de listar na interface os atributos da atividade, o SIGLa acrescenta ao formulário o campo Observação, que é um campo padrão para todas as atividades.

Ao recuperar a lista de atributos o SIGLa verifica se estes possuem a propriedade *Example*, caso possua, o exemplo é colocado na interface, ao lado do campo correspondente. Neste momento o SIGLa também verifica se os atributos possuem valores fixos. Caso um atributo possua valores fixos o SIGLa carrega os seus valores fixos definidos e os exibe em um *dropdown list*, para que o usuário selecione o valor fixo desejado. Por fim, o SIGLa verifica se algum dos atributos é do tipo *File*, caso seja é exibido

na interface um campo de *upload* de arquivos, possibilitando que o usuário selecione qualquer arquivo acessível a partir de sua máquina.

Caso algum dos campos não tenha sido preenchido corretamente o SIGLa exibe uma mensagem alertando o usuário. Uma vez que todos os campos foram validados, e estejam corretos, o SIGLa armazena no seu banco os dados definidos para a atividade e a associa com o seu experimento. Além dos valores dos atributos o SIGLa armazena no banco o usuário, a data e a hora em que a atividade foi executada.

Além de persistir os dados da atividade no banco do SIGLa é preciso persistí-los no banco de dados do Shark, para que este possa, posteriormente, retornar o histórico de atividades executadas. Para isso, quando usuário finaliza a execução de uma atividade, através da biblioteca Shark, esta atividade deve ser inicializada e finalizada. Além da atividade também é preciso inicializar, no Shark, uma instância do processo que contém a atividade.

Quando uma atividade, em termos de definição, possui mais de uma transição saindo para outras atividades distintas, é preciso resolver as condições dessas transições antes de finalizar uma instância desta atividade. Como descrito na seção 3.1, deve-se atribuir à variável global Next Activity o nome da próxima atividade que será executada. O valor desta variável é a condição que determina qual das transições será executada. Contudo, ao finalizar uma atividade que possui duas ou mais transições de saída, o usuário ainda não determinou qual a próxima atividade que será executada. Sendo assim, neste caso, ao final da execução da atividade o Shark apenas inicializa a atividade. Então, somente quando o usuário executar a próxima atividade, com o uso da biblioteca Shark, é definido o valor da variável Next Activity e então a atividade anterior é finalizada.

Dentro de um experimento, durante a execução de um fluxo de trabalho o usuário pode inicializar o fluxo de trabalho quantas vezes for necessário. Cada inicialização do fluxo de trabalho corresponde à análise de uma amostra diferente. Assim, para cada fluxo de trabalho inicializado o SIGLa associa ao experimento uma instância que irá representar uma análise.

5.3.10 Definição de Resultados

Após a execução de uma atividade o SIGLa verifica se esta possui o atributo *output*, caso possua isso significa que deverão ser definidas saídas para a atividade. Neste caso o SIGLa exibe um novo formulário com os atributos dos resultados da atividade. Para recuperar os atributos dos resultados de uma atividade o SIGLa utiliza o *parser getAttributesofResult* para obtê-los diretamento do arquivo XPDL. Além dos atributos

o *parser getAttributesofResult* retorna os tipos e propriedades de cada atributo. Da mesma forma como na execução de uma atividade o SIGLa valida se os campos foram preenchidos corretamente, de acordo com seus tipos e propriedades. Também como na definição de uma atividade o SIGLa acrescenta ao formulário da interface de definição de resultados o campo Observação. Ao recuperar a lista de atributos de resultados o SIGLa verifica se estes possuem a propriedade *Example*, caso possua, o exemplo é colocado na interface, ao lado do campo correspondente. Caso algum dos campos tenha sida preenchido incorretamete o SIGLa emite uma mensagem de erro. Uma vez que os campos estejam corretos o SIGLa armazena os dados do resultado no seu banco associando o resultado à sua atividade correspondente. Nesta interface o SIGLa disponibiliza dois botões: um para adicionar um novo resultado e um segundo botão para finalizar o processo de definição de resultados. À medida que os resultados são criados, estes são listados na tabela abaixo do formulário, para que o usuário saiba as saídas que já foram cadastradas. Nesta tabela foram definidos dois links: um para editar um resultado já criado e outro para excluí-lo. O processo de edição de um resultado é similar ao processo de criação: primeiramento o SIGLa carrega na interface os valores do resultado em edição, em seguida o SIGLa irá validar os campos preenchidos, caso estes não esteja preenchidos corretamente o SIGLa exibe uma mensagem de erro, caso contrário o SIGLa atualiza os dados no seu banco.

5.3.11 Definição de Entradas

Quando o usuário clica no *applet* para executar uma nova atividade, antes de executá-la, o SIGLa verifica se a atividade possui o atributo *input*. Caso possua isso significa que deverão ser definidas entradas para a atividade. Neste caso o SIGLa exibe uma interface onde serão listadas as saídas da atividade anteriormente executada. Estas saídas serão as possíveis entradas da atividade a ser executada. Após a seleção das entradas estas são armazenadas em memória. Quando a execução da atividade for finalizada as entradas selecionadas são associadas à atividade em banco de dados.

5.3.12 Exibir Entradas

Junto à lista de atributos de uma atividade o SIGLa exibe um link para listar as entradas de uma atividade. Ao clicar neste link o SIGLa exibe uma interface listando todas as entradas do experimento e detalha os valores preenchidos para seus atributos.

5.3.13 Exibir Saídas

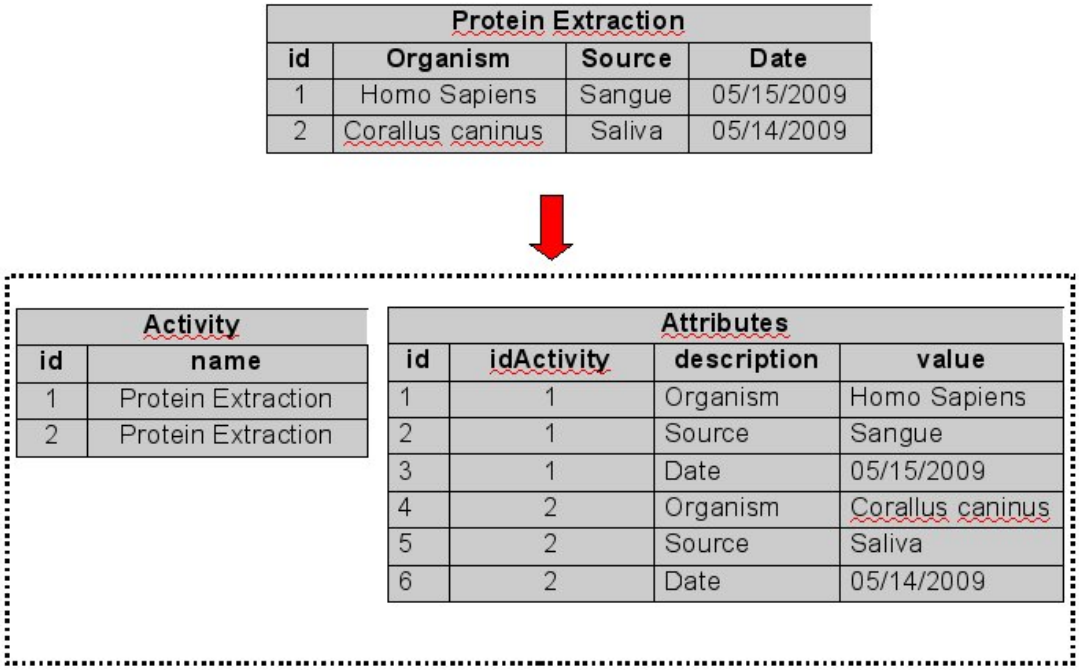
Junto à lista de atributos de uma atividade o SIGLa exibe um link para listar as saídas de uma atividade. Ao clicar neste link o SIGLa exibe uma interface listando todas as saídas do experimento e detalha os valores preenchidos para seus atributos. Nesta interface o SIGLa verifica no seu banco de dados se as saídas estão associadas como entradas de uma outra atividade, definindo assim o estado (disponível ou indisponível) de cada saída.

5.3.14 Relatórios

Ao abrir um experimento o SIGLa possibilita que o usuário gere um relatório do experimento. O relatório pode ser do tipo simples ou completo. No relatório simples o SIGLa recupera do seu banco todas as atividades executadas no experimento e exibe na interface o nome da atividade, o usuário que executou, a data e a hora em que foi executada. No relatório completo, além de recuperar as atividade executadas no experimento o SIGLa recupera os atributos das atividade e exibe na interface o nome da atividade, usuário, data, hora e os valores dos atributos de cada atividade. Para os dois tipos de relatórios é possível exportá-los para o formato pdf. Para gera o pdf foi utilizada a biblioteca XHTML Renderer. Esta biblioteca lê o texto de uma página HTML e a converte para o formato pdf.

5.4 Base de Dados

Ao contrário da marioria dos LIMS, que são específicos, o SIGLa deve possuir uma estrutura de armazenamento de dados suficientemente flexível para poder armazenar dados das atividades definidas no fluxo de trabalho, que podem possuir os mais diversos tipos de atributo. Assim, foram criadas as tabelas *Activity* e *Attribute*, que armazenarão atividades genéricas. A Figura 5.2 mostra um exemplo do armazenamento de duas atividades executadas. Ao invés de armazenar as atividades em um tabela própria, as informações sobre as atividades são armazenadas na tabela *Activity*, e na tabela *Attribute* são armazenados os atributos de forma vertical, onde a coluna *description* armazena o nome do atributo e a coluna *value* armazena o valor do atributo. A coluna *idActivity* é uma chave estrangeira para a tabela *Activity* que identifica a qual atividade o atributo está associado. A mesma idéia foi aplicada nos resultados das atividades, a tabela *Result* possui as informações sobre os resultados gerados e a tabela *AtrResult* armazena os atributos dos resultados. Cada resultado está associado a uma atividade.



Protein Extraction

id	Organism	Source	Date
1	Homo Sapiens	Sangue	05/15/2009
2	Corallus caninus	Saliva	05/14/2009

Activity

id	name
1	Protein Extraction
2	Protein Extraction

Attributes

id	idActivity	description	value
1	1	Organism	Homo Sapiens
2	1	Source	Sangue
3	1	Date	05/15/2009
4	2	Organism	Corallus caninus
5	2	Source	Saliva
6	2	Date	05/14/2009

Figura 5.2. Ao invés de armazenar os dados da Extração de Proteína em um tabela própria, as informações são armazenadas na tabela Activity, e na tabela Attribute são armazenados os atributos de forma vertical, possibilitando assim o armazenamento das atividades definidas no fluxo de trabalho.

Deste modo o SIGLa está preparado para executar e armazenar as atividades definidas no fluxo de trabalho.

A Figura 5.3 mostra o modelo de dados do SIGLa. A Tabela 5.1 detalha a função de cada tabela.

5.5 Execução de Testes

Após o desenvolvimento do SIGLa fora executados diversos testes para validação do sistema.

Inicialmente foi feito um teste de integridade do Banco de dados para verificar se o acesso ao banco de dados funciona adequadamente e sem inconsistência dos dados. Neste teste foram invocados todos os métodos de acesso ao banco, alimentando cada um com dados válidos e inválidos. Em seguida foi verificado se os dados nas tabelas do banco estavam de acordo com as ações realizadas.

Em seguinda foi realizado um teste funcional para verificar se as funcionalidades do sistema geram os resultados esperados. Neste teste foram executadas todas as funcionalidades do sistema usando dados válidos e inválidos, a fim de verificar se os resultados esperados ocorrem quando dados válidos são usados e se as mensagens de

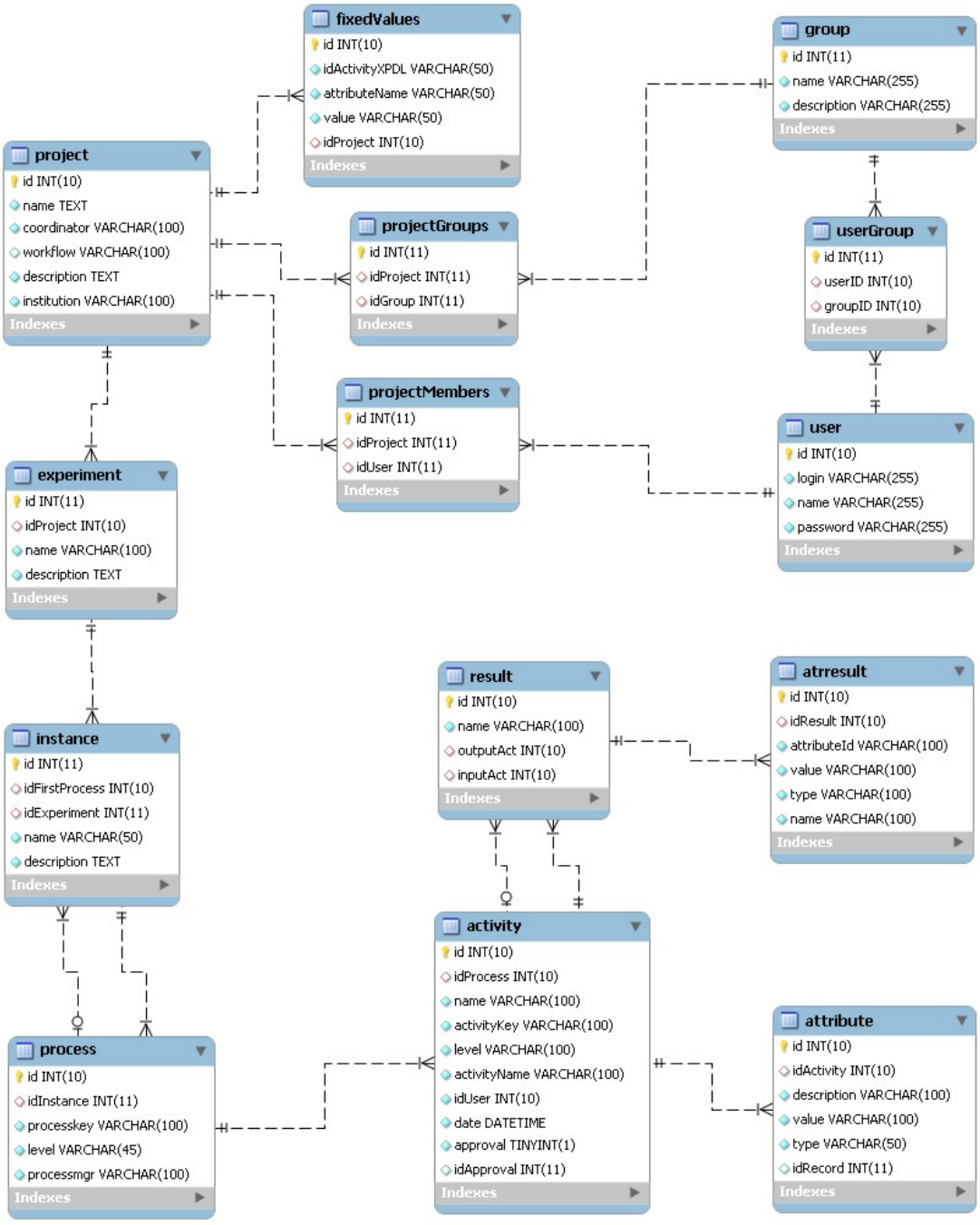


Figura 5.3. Modelo de Dados do SIGLa

Tabela 5.1. Tabelas do Banco de dados SIGLa

Tabela	Descrição
project	Armazena os projeto criados no SIGLa
fixedValues	Armazena os valores fixos dos atributos. Os valores fixos de um atributo são os possíveis valores que o atributo pode possuir, estes valores preencherão o *dropdownlist* do atributo na tela de execução da atividade.
user	Armazena os dados dos usuários do SIGLa
group	Armazena os grupos de usuários do SIGLa
userGroup	Define quais usuários pertencem a cada grupo
projectGroups	Define os grupos que terão acesso a um determinado projeto
projectMembers	Define os usuários que terão acesso a um determinado projeto
experiment	Armazena os dados dos experimentos associados aos projetos
instance	Armazena os dados das instâncias (amostras analisadas) associadas aos experimentos
process	Antes de inicializar uma atividade o Shark precisa inicializar um processo que contém a atividade. A tabela process armazena os processos criados na análise de uma amostra
activity	Armazena os dados das atividades executadas
attribute	Armazena os atributos das atividades executadas
result	Armazena os resultados das atividades executadas
atrResult	Armazena os atributos dos resultados das atividades executadas

erro são exibidas quando dados inválidos são usados. Para o teste de execução das atividades foram definidos no fluxo de trabalho exemplos de atributos de todos os tipos e com todas as propriedades: Not null, Format e Example. Em seguida foi verificado se o SIGLa estava validando corretamente os tipos e as propriedades dos atributos. O mesmo procedimento foi adotado para o teste de definição de resultados.

Em seguida foi realizado o teste de Segurança e controle de acesso para verificar que apenas aqueles usuários com acesso ao sistema possuem permissão de acessá-los. Este usuário devem acessar apenas aquelas funções ou dados para os quais o seu tipo de usuário tem permissão.

Por fim foi executado o teste de instalação onde o SIGLa foi instalado em outras máquinas, onde nunca fora instalado anteriormente. Neste teste a instalação do SIGLa foi validade em sistemas operacionais distintos.

Capítulo 6

Modelagem de Laboratórios

Para o desenvolvimento do SIGLa foi definido o fluxo de trabalho do laboratório de proteômica, que estuda as estruturas e funcionalidades das proteínas. Este laboratório foi escolhido por manipular uma grande quantidade de dados que possuem um complexo relacionamento entre eles. Usando este laboratório como exemplo para o desenvolvimento pretende-se aumentar a confiabilidade do SIGLa no gerenciamento de experimentos complexos e reais.

6.1 Proteômica

O sequenciamento genômico e o conseqüente conhecimento da sequência completa de todos os genes são contribuições importantíssimas para se estudar melhor os organismos vivos. Porém, estas informações não são suficientes para saber quais proteínas estão sendo realmente expressas na célula, num dado momento, numa determinada condição. Na era pós-genômica, tem ficado evidente que apenas com sequências completas de nucleotídeos no DNA não é possível elucidar muitos dos processos biológicos (Pandey & Mann [2000]). Adicionalmente às informações obtidas pela análise do genoma, é preciso ter conhecimento, por exemplo, sobre quais proteínas estão realmente sendo expressas, quando e em quais níveis esta expressão ocorre e sobre as eventuais modificações pós-traducionais. O fato das células de um organismo terem o mesmo genoma mas apresentarem as mais variadas funções e morfologias, fruto de diferentes composições de proteínas expressas, ilustra a importância de estudar não só a sequência dos genes, mas também a sua expressão para podermos compreender as suas funções biológicas. Desta forma, em paralelo a esta disponibilização de dados genômicos, foram desenvolvidas outras técnicas para o estudo do genoma funcional, como a análise dos transcritos de RNAm (análise de transcriptoma ou transcriptômica) e da análise do conjunto de

proteínas expressas a partir do genoma (análise de proteoma ou proteômica).

O termo proteômica foi inicialmente introduzido em 1995 e foi definido como sendo a caracterização em larga escala do conjunto de proteínas expressas em uma célula ou tecido (Wilkins et al. [1996]). A este conjunto de proteínas expressas em uma célula ou tecido, a partir do genoma, é que denominamos proteoma.

O proteoma de um organismo reflete o conjunto de proteínas expressas por este organismo em determinada situação e, ao contrário do genoma, não é estático e pode se modificar dependendo das condições e estímulos a que este organismo está exposto. O mesmo tipo de célula pode apresentar diferentes proteomas em resposta a ação de drogas, infecção por patógenos, poluição e diferentes tipos de estresse abiótico (Williams & D.F.Hochstrasser [1997]). A proteômica visa o estudo da estrutura, função e o controle dos sistemas biológicos pela análise das várias propriedades das proteínas, incluindo também os estudos da sequência(identidade), abundância, atividade e estrutura das proteínas expressas em uma célula, assim como as modificações, interações e translocações sofridas pelas proteínas (Williams & D.F.Hochstrasser [1997]).

Depois da genômica, a proteômica é geralmente considerada o próximo passo no estudo de sistemas biológicos. Este é muito mais complicado do que a genômica, principalmente porque enquanto o genoma de um organismo é mais constante, o proteoma muda de célula para célula ao longo tempo. Isto ocorre porque genes diferentes são expressos em diferentes tipos de células. Isto significa que mesmo os grupos básicos de proteínas que são produzidas em uma célula precisam ser determinadas.

6.1.1 Aplicações Práticas da Proteômica

Um dos desenvolvimentos mais promissores, a partir do estudo dos genes humanos e proteínas, tem sido a identificação de possíveis novas drogas para o tratamento de doenças. Isso depende das informações do genoma e proteoma para identificar proteínas associadas com as doenças, que os softwares podem usar como alvos para indicar novos medicamentos. Por exemplo, se uma determinada proteína está associada a uma doença, a sua estrutura 3D fornece a informação para projetar as drogas que interferem com a ação da proteína. Uma molécula que se encaixa no sítio ativo de uma enzima, mas não pode ser libertado pela enzima, vai inativar a enzima. Esta é a base das ferramentas de descoberta de novas drogas, que visam encontrar novas drogas para inativar proteínas associadas às doenças. Como diferenças genéticas entre os indivíduos são encontradas, os pesquisadores esperam usar essas técnicas para desenvolver drogas personalizadas, que são mais eficazes para o indivíduo.

Outra aplicação interessante da proteômica é usar proteínas como biomarcado-

res específicos para diagnosticar doenças. Biomarcadores ou marcadores biológicos são moléculas que podem ser medidas experimentalmente e indicam a ocorrência de um determinado processo em um organismo. Algumas técnicas permitem testar as proteínas produzidas durante uma determinada doença, o que ajuda a diagnosticar a doença rapidamente. Algumas doenças como a Alzheimer, doenças cardíacas e pulmonáres possuem biomarcadores característicos que podem ser utilizados para o seu diagnóstico.

6.1.2 Processo do Laboratórios de Proteômica

O primeiro passo no laboratório de proteômica é a extração da proteína a partir do organismo em análise. O método mais geral para obtenção de proteínas consiste na extração de proteínas pela ação combinada de solventes.

Após a extração as proteínas devem ser separadas, isto pode ser feito através de uma cromatografia ou de uma eletroforese. A cromatografia se baseia na diferença de características químicas das proteínas, a fim de separá-las. Isto faz com que cada proteína interaja diferencialmente com uma fase estacionária (que pode ser sólida ou líquida ou pode consistir de uma mistura de um sólido com um líquido) e com uma fase móvel (gás ou líquido). A atração diferencial das proteínas pelas fases móvel e estacionária afetará a sua mobilidade, fazendo com que estas se separem. A eletroforese consiste em separar, sob a influência de um campo elétrico, moléculas que possuem cargas. A velocidade de migração depende de fatores como: tamanho, forma, e carga elétrica da molécula em estudo. Na eletroforese bidimensional, as proteínas são submetidas a duas etapas de separação eletroforética consecutivas, baseado nas diferentes propriedades das proteínas. A primeira etapa de separação, denominada primeira dimensão, consiste em uma focalização isoelétrica (IEF), onde as proteínas são separadas pela sua carga elétrica em um gel de poliacrilamida com um gradiente de pH. Nesta etapa as proteínas migram no gel até atingirem uma posição estacionária onde possuem carga líquida zero, sendo este, o ponto isoelétrico da proteína. Na segunda dimensão, as proteínas previamente separadas pela IEF são submetidas a uma eletroforese desnaturante em gel poliacrilamida (SDS-PAGE), onde são separadas de acordo com suas massas moleculares. Como os parâmetros usuados na primeira dimensão (ponto isoelétrico) e na segunda dimensão (massa molecular) são independentes, a separação atinge um alto nível de resolução, o que permite a visualização de centenas de proteínas diferentes ao mesmo tempo. Após a eletroforese bidimensional as proteínas são visualizadas diretamente no gel, através de métodos de coloração com Azul de Comassie ou por coloração com nitrato de prata, por exemplo, resultando num perfil

bidimensional de pontos (denominados *spots*), cada um contendo múltiplas cópias de uma proteína. A identificação das proteínas presentes nos perfis de eletroforese bidimensional é feita por espectrometria de massa. Cada um dos *spots* é removido do gel (recortado com bisturi) e submetido à espectrometria de massa por uma metodologia denominada *peptide mass fingerprinting*.

A espectrometria de massa é uma técnica analítica de estudo de moléculas que se baseia no movimento de íons em campos elétricos e magnéticos para classificá-los de acordo com sua relação massa-carga, produzindo um espectro de massas. A espectrometria de massas já é utilizada há muitos anos para o estudo de moléculas inorgânicas, porém, somente na última década esta técnica foi aperfeiçoada para o estudo de moléculas biológicas (proteínas) sem que ocorra a destruição destas durante o processo de ionização (Yates [1998]) O aperfeiçoamento destas técnicas de eletroforese bidimensional e espectrometria de massa foi o que possibilitou o avanço no estudo de proteínas em larga escala e o surgimento do conceito atual de análise de proteomas (Tyers & Mann [2003]).

Uma vez que as proteínas foram separadas e submetidas à espectrometria de massa são realizadas consultas em bancos de dados biológicos a fim de identificá-las.

6.1.3 Fluxo de Trabalho de Proteômica

Utilizando o *Together Workflow Editor*, foi desenvolvido o fluxo de trabalho de Proteômica. Este fluxo de trabalho foi definido juntamente com especialistas da área para termos a certeza de que todos os requisitos do laboratório fossem atendidos. Durante o desenvolvimento do SIGLa, este foi validado constantemente pelos especialistas. A Figura 6.1 mostra o fluxo de trabalho definido.

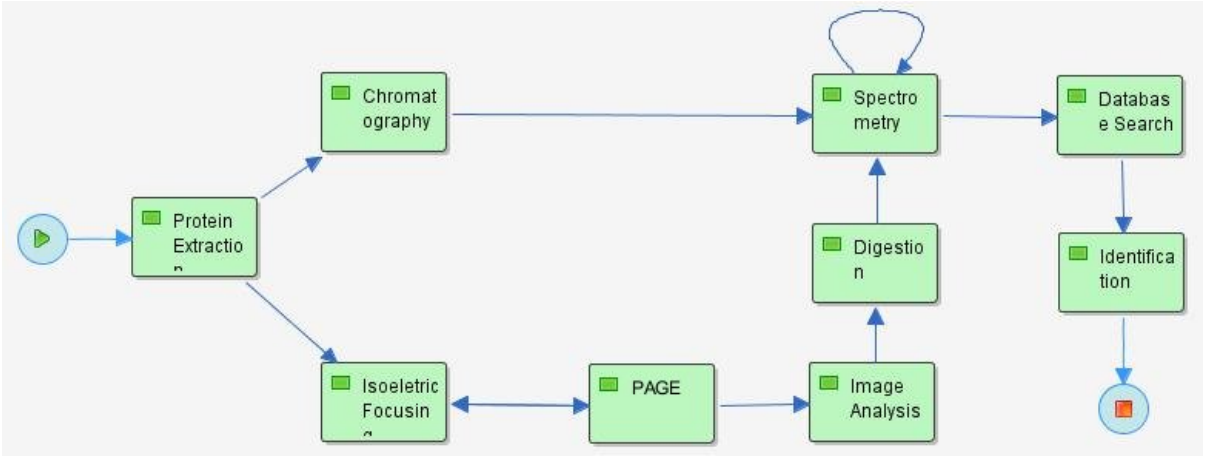


Figura 6.1. fluxo de trabalho do laboratório de Proteômica

Juntamente com os especialistas em proteômica, da Universidade Federal de Minas Gerais, foram listadas todas as atividades executadas no laboratório. As atividades são: *Protein Extraction* (Extração da Proteina), *Chromatography* (Cromatografia), *Isoeletric Focusing* (Focalização Isoelétrica), *PAGE* (Eletroforese), *Image Analysis* (Análise de Imagem), *Digestion* (Digestão), *Spectrometry* (Espectrometria), *Database search* (Busca em Banco) e *Identification* (Identificação). Além das atividades também foram definidos os atributos das atividades. Por exemplo, para a atividade *Protein Extraction* foram definidos os atributos *Organism*, *Source* e *Date*. O atributo *Organism* irá identificar o organismo do qual está sendo feita a extração de proteína. O organismo poderia ser preenchido como *Homo Sapiens*, caso a amostra em análise sejam gotas de saliva ou de sangue de um ser humano. O organismo poderia ser preenchido também como *Eunectes notaeus* (nome científico da cobra sucuri), caso a amostra em análise fosse o veneno de uma cobra sucuri. Uma vez que este campo será preenchido com um texto, o atributo *Organism* foi definido como sendo do tipo Alfanumérico. Durante a execução da atividade *Protein Extraction* é imprescindível que seja informado o organismo em questão, por isso foi atribuído a este atributo a propriedade *not null*, para tornar obrigatório o preenchimento deste campo. O atributo *Source* irá identificar de qual parte do organismo foi retirada a amostra em análise. Por exemplo, caso o organismo seja *Homo Sapiens*, o atributo *Source* poderia ser preenchido com os valores sangue, saliva, pele, ou seja, neste exemplo, este campo identificará de que parte do corpo humano foi retirada a amostra em análise. Como o atributo *Organism*, o atributo *Source* também foi definido como sendo do tipo alfanumérico, pois será preenchido com um texto. Segundo os especialistas em proteômica, também o campo *Source* é uma informação fundamental, por isso também foi atribuido ao *Source* a propriedade *not null*. Finalmente o atributo *Date* irá armazenar simplesmente a data de execução do experimento. Este atributo foi definido com o tipo Data e também possui a propriedade *not null*. O anexo A contêm todos os atributos definidos para as atividades do fluxo de trabalho de proteômica, bem como o tipo definido para cada atributo e as propriedades destes.

Para utilizar o fluxo de trabalho de proteômica, durante a criação do projeto seleciona-se o arquivo que contém a definição deste fluxo de trabalho. Em seguida criam-se os experimentos do projeto. Durante a execução do fluxo de trabalho o SIGLa irá executar as atividades definidas no fluxo de trabalho de proteômica na ordem pré-definida. A Figura 6.2 mostra um exemplo da execução deste fluxo de trabalho. Apenas carregando um arquivo, toda a estrutura da execução do fluxo de trabalho é carregada, então o SIGLa adapta sua interface para as novas definições. Na execução das atividades os atributos listados são os atributos das atividades da proteômica.

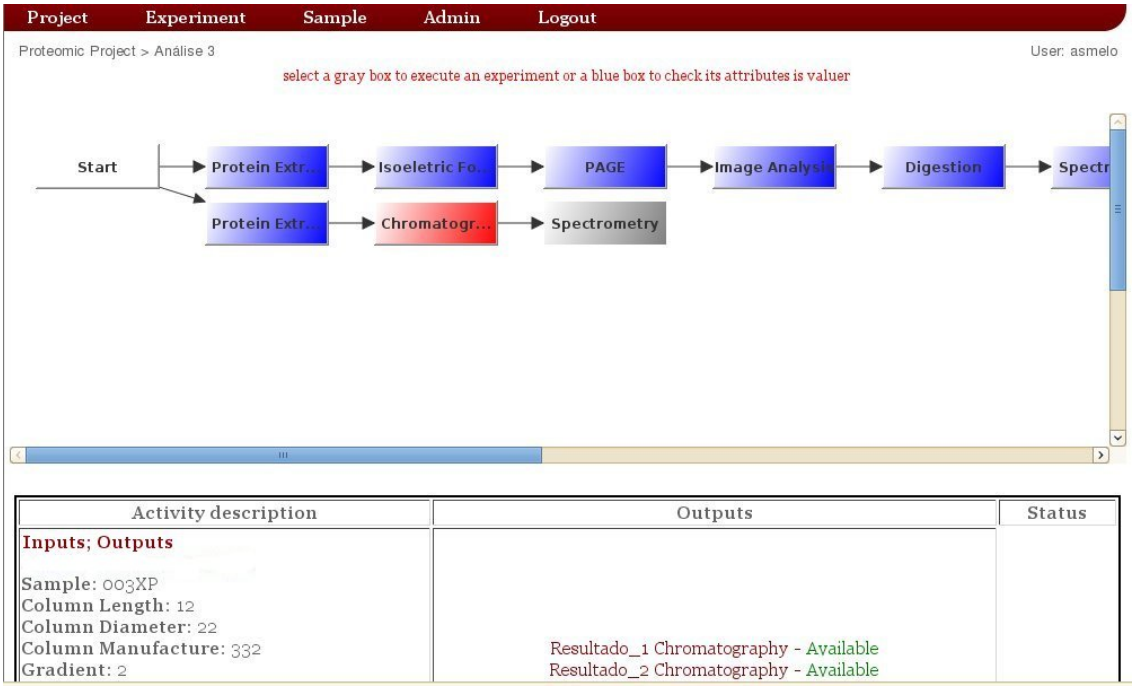


Figura 6.2. Exemplo de Execução do fluxo de trabalho de Proteômica

6.2 Microarranjos

A técnica de microarranjos de DNA é utilizada para estudar a expressão gênica em larga escala em várias espécies. Esta técnica permite o estudo de milhares de genes simultaneamente. Um microarranjo de DNA, ou DNA-chip, consiste num arranjo pré-definido de moléculas de DNA (fragmentos de DNA genômico, cDNAs ou oligonucleotídeos) quimicamente ligadas à uma superfície sólida, usualmente lâminas de microscópio revestidas com compostos que conferem carga positiva. Os microarranjos também podem ser preparados em membranas de nylon positivamente carregadas. As lâminas de microarranjo contendo sequências de nucleotídeos são utilizadas para duas finalidades principais: em estudo de genômica comparativa e na análise da expressão diferencial de genes. Neste caso, representa uma ferramenta muito poderosa para a descoberta de genes envolvidos em diversos processos biológicos, uma vez que é possível fazer uma varredura da expressão de milhares de genes ao mesmo tempo (Schena [1995]). A grande massa de dados que é obtida com a tecnologia de microarranjos de DNA proporciona pistas para entender processos moleculares ou vias complexas envolvidas no funcionamento da célula em determinada condição (Boothroyd [2003]).

Segundo Mayer [2010], em estudos de expressão gênica, a maior vantagem, é que é possível em um único experimento, mensurar a expressão de milhares de genes simultaneamente. Quando se deseja analisar a expressão de apenas alguns genes, a

técnica de real-time PCR é geralmente empregada, que consegue analisar a expressão de até 4 genes simultaneamente. Quando se deseja analisar o perfil da expressão gênica no nível de genoma, esta técnica é tecnicamente inviável, já que, por exemplo, para um genoma bacteriano de 4 mil genes, seriam necessários 1000 análises.

Microarranjos de DNA estão rapidamente se tornando padrão em ferramentas de biologia molecular, fornecendo uma poderosa abordagem para a análise da resposta transcricional global. Durante a última década, microarranjos têm sido amplamente utilizados em várias disciplinas biológicas e o número de estudos publicados utilizando a tecnologia ainda está aumentando (Yauk & Berndt [2007]).

6.2.1 Aplicações Práticas do Microarranjo

Esta tecnologia tem auxiliando pesquisas nas mais diversas áreas. Um exemplo importante é na pesquisa do câncer, onde normalmente se compara a expressão gênica de células saudáveis e de células do tumor. Com isto analisa-se as diferenças na sua expressão gênica e assim identifica-se genes que possuem uma função chave no desenvolvimento do câncer.

Atualmente, a tecnologia de microarranjos está sendo utilizada para análises clínicas, como por exemplo, na identificação rápida e com grande sensibilidade e especificidade de cepas virais (Mehlmann [2007]). Também se aprecia a possibilidade de utilizar esta tecnologia no prognóstico da resposta à terapia em pacientes com câncer de mama, individualizando desta forma o tratamento (Murphy et al. [2005]), assim como no prognóstico de doenças crônicas hepáticas e carcinoma hepatocelular (Lemmer et al. [2006]).

Além do uso nos estudos da expressão diferencial em genes, os experimentos de microarranjos podem ser usados para uma análise comparativa de genomas. Em parasitologia, os experimentos de microarranjos podem ser direcionados a entender a patogênese e estudar, por exemplo, o que torna a cepa de uma determinada espécie virulenta e a outra cepa avirulenta, como é o caso de *Entamoeba histolytica* e *Entamoeba dispar*, cujos genomas apresentam cerca de 98% de identidade. No entanto, a primeira é patogênica e a segunda coloniza o intestino do hospedeiro sem causar transtorno (Diamond & Clark [1993]).

Outra aplicação interessante, ainda na análise comparativa de genomas, é a utilização de microarranjos de DNA para estudar cepas atenuadas para vacina, como é o caso dos *bacillus Calmette-Géurin* (BCG), que são clones derivados da bactéria virulenta *Mycobacterium bovis* e que são comumente utilizados como vacina para previnir tuberculose (TB). Os experimentos de microarranjos realizados em lâminas *Affymetryx*

GeneChip contendo o genoma de *Mycobacterium tuberculosis* e utilizando DNA genômico marcado de BCG mostram que a falta de uma região chamada RD1 em BCG está associada com a atenuação da virulêcia (Mostowy [2003]).

6.2.2 Processo do Laboratório de Microarranjo

Um dos procedimentos do laboratório para a técnica de microarranjo segue os seguintes passos: O RNA total é extraído usando o reagente Trizol, por exemplo. Então o RNA é quantificado usando um espectrômetro e a qualidade do RNA total é avaliada através de uma eletroforese em gel desnaturante. Em seguida o RNA total é limpo usando o quite de limpeza *RNeasy MinElute*, o RNAm é então obtido usando o *Kit Oligotex*. Aproximadamente 50 ng de mRNA de cada amostra são amplificadas. Em seguida o RNA amplificado é rotulado usando o fluorocromos cianina 3 (Cy3), que emite a cor verde, e cianina 5 (Cy5), que emite a cor vermelha, através da marcação. Assim, a proporção de cores para cada elemento aponta as diferenças relativas na expressão gênica entre duas amostras.

A lâmina de microarranjo é usada para detectar o nível de expressão de RNAm relacionada ao DNA impresso em sua superfície. Essa resposta é obtida através da incubação do microarranjo com uma solução contendo DNA ou RNA que foram obtidas a partir de amostras biológicas. A idéia central por trás do método é que o DNA ou RNA na solução se ligará às seqüências depositadas sobre a superfície do microarranjo de acordo com a seqüência complementar (Draghici [2003]). A detecção é possível pois as amostras foram "marcadas"com Cy3 ou Cy5 quando utiliza-se microarrays em vidro ou com o isótopo 33-P quando os arranjos são preparados em membranas de nylon. Para ambos os casos se faz necessário a geração de uma imagem de hibridação, que é obtida por meio de leitores (*scanners*) a laser (para os fluorocromos) ou leitores de fósforo (para o isótopo 33-P). Durante a leitura o "scanner" digitaliza a imagem em alta resolução, para análise posterior. As imagens digitalizadas são então analisadas por um *software* específico que irá converter a cor e a intensidade da fluorescência em um conjunto de valores numéricos. Os arquivos resultantes (valores numéricos) são analisados posteriormente em outros *softwares* onde os dados são filtrados e tratados estatisticamente.

6.2.3 Fluxo de Trabalho de Microarranjos

A Técnica de microarranjos gera um grande número de informações, tanto os dados laboratoriais como arquivos de imagem e de dados. O processo de microarranjos tem

várias etapas e cada uma exige a anotação de informações diversas. Normalmente todas as informações são armazenadas em cadernos de laboratórios, o que pode tornar difícil o acesso à informação, já que eles estão em ordem cronológica, apenas. Depois da análise das lâminas de microarranjos, de imagens e dados, muitos arquivos de dados são produzidos e o número pode chegar a centenas de arquivos. A organização desses dados em uma única plataforma, como SIGLa, pode manter organizado todos os dados e arquivos, e é essencial para a manipulação dos dados. Além disso, como a técnica de microarranjos é normalmente feito por um grupo de pesquisadores, é importante que todos tenham acesso aos dados padronizados.

Para implementar o processo do laboratório de microarranjos no SIGLa, foi criado o seu fluxo de trabalho (Figura 6.3). Para isso foram listadas as atividades executadas no laboratório de microarranjos. As atividades são: *RNA Extraction* (Extração de RNA), *Amplification* (Amplificação), *Labeling* (Marcação), *Hybridization* (Hibridização), *Cleaning and Concentration* (Limpeza e Concentração), *Isolation* (Isolamento), *Scanning*, *Image Analysis* (Análise de Imagem), *Data Analysis* (Análise de Dados) e *Data Validation* (Validação dos Dados). Foram definidos também os atributos de cada atividade. Por exemplo, para a atividade *Labeling* foram definidos os atributos *Protocol*, *Volume for Labeling*, *Fluorescent Dye* e *Date*. O atributo *Protocol* irá identificar o protocolo utilizado no experimento. Neste âmbito, um protocolo é uma seguencia de passos que define a execução do experimento. A depender do objetivo da análise em questão pode ser utilizada uma quantidade maior ou menor de *fluorocromos cianina*, por exemplo. Ou seja, apesar de ser o mesmo experimento, este pode ser executado de formas diferentes. O que irá identificar o modo como o experimento foi executado é justamente o protocolo utilizado. Este campo será preenchido com o nome do protocolo, *Trizol*, por exemplo, logo, o atributo foi definido como sendo do tipo Alfanumérico. Neste experimento específico sempre haverá um protocolo associado, logo foi atribuído a este campo a propriedade *not null*. O atributo atributo *Volume for Labeling* irá identificar a quantidade de DNA utilizada no experimento. Este atributo foi definido como sendo do tipo Real, uma vez que será preenchido com o volume de DNA utilizado. Segundo os especialistas esta é uma informação imprescendível na execução do experimento, deste modo, foi atribuído a este a propriedade *not null*. Além desta propriedade, também foi atribuído ao campo *Volume for Labeling* a propriedade *Example*, com o valor μL. Deste modo, durante a execução da atividade, aparecerá ao lado deste campo a abreviação μL (microlitro), indicando ao usuário do SIGLa que o campo *Volume for Labeling* deverá ser preenchido com o volume expresso em microlitros. Já o atributo *Fluorescent Dye* identificará o tipo do fluorocromos cianina utilizado, que pode ser Cy3 ou o Cy5. O processo de marcação (*Labeling*) é feito primeiramente

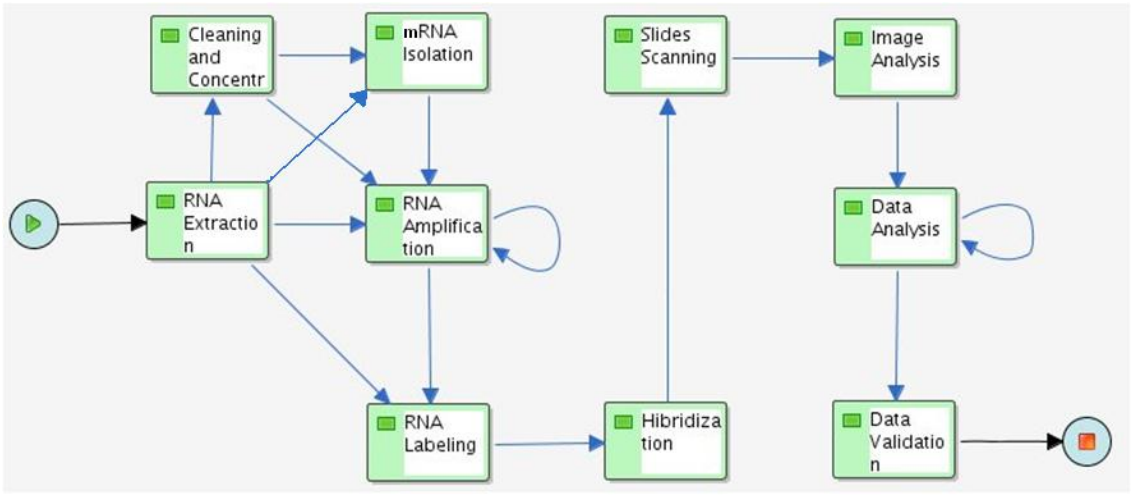


Figura 6.3. Definição do fluxo de trabalho de Microarranjos

em amostras separadas e cada uma é marcada com um tipo de fluorocromos cianina diferente. Posteriormente estas amostras são misturadas e é dado prosseguimento ao experimento. O atributo *Fluorescent Dye* foi definido como sendo do tipo Alfanumérico e foi atribuído a este a propriedade *not null*, pois não é uma informação opcional na execução da atividade. Por fim, o atributo *Date* irá armazenar a data de execução do experimento. Este atributo foi definido com o tipo Data e também possui a propriedade *not null*. O Anexo B contém os atributos definidos para cada atividade do fluxo de trabalho de Microaranjos.

Neste fluxo de trabalho houve a utilização das entradas e saídas da atividades. Cada atividade gera saídas que serão entradas da próxima atividade a ser executada. Por exemplo, no caso da Extração de RNA são gerados os RNAs extraídos como saída, destes é necessário armazenar a concentração total de RNA, o volume final, a imagem do Gel e o método de dosagem utilizado. Todos esses atributos são definidos na atividade *RNA Extraction*, no fluxo de trabalho de microarranjos. Após a *RNA Extraction* é possível executar a *Cleaning and Concentration*, uma *Amplification* ou uma *Labeling*. No fluxo de trabalho foi definida uma entrada para cada uma dessas atividades. Deste modo as saídas da *RNA Extraction* serão as entradas para as atividades de *Cleaning and Concentration*, *Amplification* ou *Labeling*. Ao definir as entradas e saídas no fluxo de trabalho é possível definir o número máximo e mínimo de entradas e saídas das atividades. No caso do fluxo de trabalho de microarranjos foi definido para todas as atividades o número mínimo de um resultado, obrigando ao usuário a inserir ao menos uma saída.

Para todas as atividades do fluxo de trabalho de microarranjo, com exceção da

Hybridization e *Data Validation*, foi definida uma saída. Por exemplo, a saída da atividade *Amplification* deverá armazenar os atributos *Amplification Final Contentration*, *Amplification Final Volume* e *Dosage Method*. Durante a execução da *Amplication* é acrescentado ao DNA em análise a enzima polimerase. Este processo é feito com o objetivo aumentar o número de DNA. Assim, o atributo *Amplification Final Concentration* irá determinar o volume de DNA após ser amplificado. Para este atributo foi definida a propriedade Exemplo com o valor $ng/\mu L$. O atributo *Amplification Final Concentration* foi definido como sendo do tipo Real. O atributo *Amplification Final Volume* irá determinar o volume final após a mistura do DNA com a enzima polimerase. Para este atributo foi definida a propriedade Exemplo com o valor μL. Este atributo também foi definido como sendo do tipo Real. Já o atributo *Dosage Method* ira informar o método utilizado para identificar a concentração final de DNA. Existem várias formas de identificar esta concentração, um exemplo seria com o uso do Nanodrop. Assim o *Dosage Method* foi definido como sendo do tipo Alfanumérico. Para todos os atributos do resultado da atividade *Amplification* foi definida a propriedade *not null*, para tornar obrigatório o preenchimento do atributo. O Anexo C contém os atributos dos resultados das atividades do fluxo de trabalho de microarranjo. Os resultados gerados pelas atividades serão as entradas das próximas atividades a serem executadas.

Para utilizar o fluxo de trabalho de microarranjos, durante a criação do projeto seleciona-se o arquivo que contém a definição do fluxo de trabalho de microarranjos. Em seguida cria-se os experimentos do projeto. Durante a execução do fluxo de trabalho percebe-se que o SIGLa apenas permite a execução das atividades definidas no fluxo de trabalho de microarranjos, e na ordem pré-definida. A Figura 6.4 mostra um exemplo da execução do fluxo de trabalho de microarranjos. Apenas carregando um arquivo, toda a estrutura da execução do fluxo de trabalho é carregada, então o SIGLa adapta sua interface para as novas definições. Na execução das atividades os atributos listados são os atributos das atividades do microarray.

O processo de adaptação do SIGLa para o laboratório de microarranjos foi bastante eficiente. Uma vez que contamos com o apoio de especialistas em microarranjos levamos apenas algumas semanas para compreender as necessidades do laboatório e definir suas atividades. Em cerca de quatro dias o fluxo de trabalho foi construído. Ou seja, com o SIGLa foi possível construir um LIMS de microarranjos em poucos dias, daí pôde-se perceber a flexibilidade e eficácia do método proposto no presente trabalho.

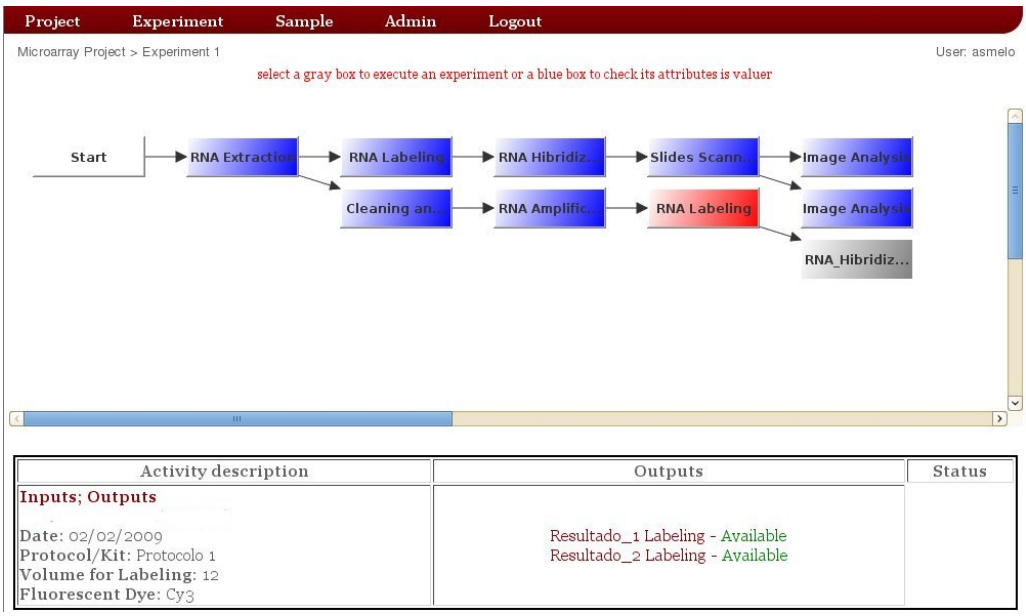


Figura 6.4. Execução do fluxo de trabalho de Microarranjos

Capítulo 7

Conclusão

Hoje em dia é muito difícil um laboratório de médio ou grande porte se desenvolver sem um sistema computacional para gerenciamento dos seus inúmeros dados. Atualmente estes laboratórios são obrigados a adquirir um LIMS por um alto custo, ou trabalhar com sistemas limitados e ineficientes. Neste trabalho desenvolvemos o SIGLa, um sistema baseado em fluxos de trabalho que atende os principais requisitos de um LIMS a fim de integrar, gerenciar e melhorar a qualidade dos dados do laboratório, gerando assim resultados mais confiáveis e consistentes. Além disso o SIGLa não uma é solução apenas para um tipo de laboratório, mas para diversos tipos de laboratórios biológicos, uma vez que o usuário pode facilmente personalizar o SIGLa para seu laboratório, simplesmente definindo seu fluxo de trabalho. O primeiro fluxo de trabalho a ser definido para o SIGLa define as atividades de um laboratório de proteômica. Para avaliar a capacidade do SIGLa de adaptar-se a múltiplos laboratórios foi desenvolvido um segundo fluxo de trabalho modelando as atividades de um laboratório de microarranjos. Com o SIGLa foi possível construir um LIMS de microarranjos em apenas poucos dias, daí pôde-se perceber a flexibilidade e eficácia do método proposto no presente trabalho. A flexibilidade do SIGLa proporciona não apenas a sua adaptação a múltiplos laboratórios, mas possibilita também que os laboratórios possam modificar seu fluxo de trabalho com o passar do tempo, adaptando-o às novas necessidades do laboratório, mantendo assim seu sistema de gerenciamento mais atualizado.

Para executar as atividades de gerenciamento de fluxos de trabalho, no desenvolvimento do SIGLa, foi utilizada a biblioteca Enhydra Shark, que utiliza o padrão XPDL. O conceito de fluxo de trabalho se mostrou bastante útil para a modelagem de laboratórios, contudo a especificação XPDL, e portanto também o Enhydra Shark, apresentaram limitações para modelar as atividades dos laboratórios. Assim, foi preciso criar no próprio código do SIGLa rotinas de gerenciamento de fluxos de trabalho. Ou

seja, para desenvolver o SIGLa foi preciso praticamente desenvolver todo um sistema de gerenciamento de fluxos de trabalho.

No momento o SIGLa está sendo utilizado por pesquisadores de microarranjo da Universidade Federal de Minas Gerais. Espera-se que com a difusão deste trabalho o SIGLa possa ser cada vez mais utilizado e melhorado, uma vez que possui o código fonte aberto. O SIGLa está disponível para teste em `http://luar.dcc.ufmg.br/sigla` no link SIGLa. Este pode ser acessado com o usuário *guest* e a senha *guest*. Com o desenvolvimento do SIGLa espera-se contribuir positivamente para a área de gerenciamento de dados complexos em laboratórios, além de possibilitar que laboratórios com pouco recurso possam utilizar uma ferramenta de gerenciamento de dados de alto nível, aumentando assim a sua produtividade e a qualidade de seus resultados.

7.1 Validação do SIGLa em Laboratórios Reais

Após a definição do fluxo de trabalho de microarranjos um especialista em microarranjos executou vários experimentos utilizando o SIGLa. A Dr. Daiane de Laat, do laboratório de Genética Bioquímica da UFMG, testou o sistema com dados reais de um experimento de microarranjos, com doze amostras iniciais de RNA e seis lâminas de microarranjos. Todas as etapas do fluxo de trabalho foram executadas para todas as doze amostras. A interface de execução do fluxo de trabalho guiou o usuário quanto à sequência de atividades que deveriam ser executadas, facilitando assim a execução das atividades para o usuário. Depois de completar todas as etapas, o usuário pôde recuperar, sem dificuldades, as informações das atividades executadas. Segundo a Dr. Daiane de Laat, este fácil acesso a todas as informações de uma atividade é importante quando se trabalha com uma técnica complexa e cheia de etapas, como o microarranjo. Além disso, o sistema permitiu armazenar os protocolos e os arquivos gerados durante a execução das atividades, tais como imagens de géis ou arquivos de dados. O SIGLa também contribuiu para o laboratório com outras características importantes, como permitir a escolha entre duas ou mais opções possíveis em um campo (valores fixos), como a escolha de corantes fluorescentes Cy3 e Cy5, o que facilita o trabalho dos pesquisadores do laboratório de microarranjos, quando estão preenchendo esses campos. Além disso, foi possível gerar relatórios no formato PDF com todas as informações do experimento, que pôde ser facilmente visualizado e impresso. A validação dos valores dos atributo feitas pelo SIGLa também foi muito útil. Em muitos momentos ele evitou o preenchimento dos campos com valores errados.

7.2 Trabalhos Futuros

A construção dos fluxos de trabalho exige a presença de um especialista nas atividades do laboratório, para definir as atividades a serem executadas e seus atributos. Devido a este fato neste trabalho foram definidos dois fluxos de trabalho, o de proteômica e de microarranjos. Sendo assim, a primeira contribuição a ser feita a este trabalho certamente é a construção de fluxos de trabalho que definam as atividades de outros laboratórios, fazendo assim com que o SIGLa possa contribuir para mais laboratórios, gerenciando os seus dados.

A modelagem do SIGLa através de fluxos de trabalho é bastante poderosa, possibilitando a definição dos mais diversos tipos de atividades. Isto é suficiente para integrar e gerenciar as atividades experimentais de um laboratório. Sabe-se, porém, que os laboratórios precisam também gerenciar outros dados além dos seus experimentos. Por exemplo, um laboratório necessita ter um controle de estoque dos materiais utilizados nos experimentos. Em laboratórios comerciais é necessário também um controle dos pedidos realizados pelos clientes, além de um gerenciamento financeiro. Seria interessante se o SIGLa executasse também estas atividades.

No projeto inicial do SIGLa previam-se algumas funcionalidades extras, contudo a codificação do SIGLa se mostrou bastante complexa, principalmente por ser um sistema adaptável, ou seja, todas as suas funcionalidades deveriam ser contruídas de tal modo que se adaptassem a laboratórios biológicos distintos. Sendo assim a sua codificação levou mais tempo do que o previsto, impossibilitando o desenvolvimento de funcionalidades como a alocação, aprovação e rejeição das atividades. É muito comum que equipes executem os experimentos dos laboratórios, sendo assim é importante que as atividades possam ser alocadas para os membros das equipes, possibilitando assim o gerenciamento do trabalho executado pelos técnicos dos laboratórios. Quando se trata de atividades críticas no processo do laboratório, muitas vezes é desejável a aprovação ou rejeição destas atividades por um supervisor. Esta funcionalidade também seria importante, pois ajudaria a melhorar a qualidade dos resultados gerados. Outra funcionalidade prevista era a *internationalization*, que consiste na criação de arquivos contendo os textos das interfaces do SIGLa, uma vez que estes arquivos fossem definidos o SIGLa iria consulta-los para imprimir os textos das suas interfaces. Este processo possibilita que o SIGLa possa ter seu idioma modificado facilmente, para isto bastaria criar estes arquivos em diversos idiomas. Deste modo o SIGLa poderia contribuir para laboratórios de outros países. Para amenizar a falta desta funcionalidade as interfaces do SIGLa foram desenvolvidas em inglês, a lingua mais utilizada no meio científico. Uma última funcionalidade prevista foi o gerenciamento de placas. Diversos

laboratórios utilizam placas para a execução dos seus experimentos. Normalmente são utilizadas placas de 96 poços onde são depositadas amostras a serem analisadas simultaneamente por um determinado equipamento. Esperava-se criar uma interface para estas placas onde fosse possível definir qual amostra estaria em cada posição da placa. Contudo, apesar desta interface não ter sido criada, esta pode ser implementada através do fluxo de trabalho, basta definir como atributos da atividades os poços da placa. A intenção do desenvolvimento desta funcionalidade é principalmente para conceber uma interface mais amigável para o usuário.

Uma vez que o SIGLa possui um sistema de gerenciamento de fluxos de trabalho incorporado ao seu código, este possui a capacidade de gerenciar vários tipos de fluxos de trabalho, não apenas aqueles que modelam as atividades de um laboratório. Dessa forma o SIGLa poderia perfeitamente ser utilizado para gerenciar outros processos, como as atividades administrativas de uma empresa, por exemplo. Uma outra contribuição para este trabalho seria a definição de fluxos de trabalho de outra natureza para serem executados no SIGLa.

Referências Bibliográficas

BonitaSoft (2009). Bonita: Open source workflow. http://wiki.bonita.objectweb.org/xwiki/bin/view/Main/WebHome.

Boothroyd, J. C. (2003). Dna microarray in parasitology: strengths and limitations. *Trends Parasitol*, 19(10):470–476.

BPEL (2010). Business process execution language. http://www.oracle.com/technology/products/ias/bpel/index.html.

BPMI (2009). Business process management initiative. http://www.bpmi.org/.

BPML (2010). Business process modeling language. http://www.ebpml.org/bpml.htm.

BPMN (2009). Business process modeling notation. http://www.bpmn.org/.

Diamond, L. & Clark, C. (1993). A redescription of entamoeba histolityca schaudinn.; 1903(emended walker.;1911) separating it from entamoeba dispar brumpt.; 1925. *J. Eukaryot. Microbiol.*, 40(3):340–344.

DOM (2010). Document object model. http://www.w3.org/DOM/.

Draghici, S. (2003). *Data Analysis Tools for DNA Microarrays*. Chapman and Hall/CRC.

EAI (2010). Enterprise application integration. http://www.omg.org/technology/documents/formal/eai.htm.

Eclipse (2010). Eclipse. http://www.eclipse.org/.

EJB (2010). Enterprise java beans. http://java.sun.com/products/ejb/.

Esterling, L. & Overgaard, B. (2009). Scierra proteomics lws—a flexible lims for managing complex experimental data. In *Cimarron Software, Inc.* Salt Lake City, UT, USA.

GmbH, D. (2009). Wfmopen project. http://wfmopen.sourceforge.net/.

Hendricks, R. & Learn, M. (2003). A laboratory information management system (lims) for an academic microchip fabrication facility. In *University/Government/Industry Microelectronics Symposium.*

Hinton, M. D. (1995). *Laboratory Management Systems.* Marcel Dekker, inc. New York.

LabWare (2009). Labware. http://www.labware.com/lwweb.nsf.

Lemmer, E.; Friedman, S. & J.M, L. (2006). Molecular diagnosis of chronic liver disease and hepatocellular carcinoma: the potential of gene expression profiling. *Seminars in liver disease*, 26(4):373–384.

Mayer, L. (2010). Tecnologia microarray (biochips de dna). http://artigos.netsaber.com.br/resumo_artigo_13846/artigo_sobre_tecnologia_microarray_(biochips_de_dna).

Mehlmann, M. (2007). Comparation of the mchip to viral culture.; reverse transcription-pcr.; and the quickvue influenza a+b test for rapid diagnosis of influenza. *J. Clin. Microbiol.*, 45(4):1234–1237.

Morisawa, H.; Hirota, M. & Toda, T. (2006). Development of an open source laboratory information management system for 2-d gel electrophoresis-based proteomics workflow. volume 7, pp. 430+.

Mostowy, S. (2003). The in vitro evolution of bcg vaccines. *Vaccine*, 21:4270–4274.

Murphy, N.; Millar, E. & Lee, C. (2005). Gene expression profiling in breast cancer: towards individualising pacient management. *Pathology*, 37:271–277.

Pandey, A. & Mann, M. (2000). Proteomics to study genes and genomes. *Nature*, 405(6788):837–846.

POJO (2010). Plain old java objects. http://sun.systemnews.com/articles/145/3/opt-dev/22871.

Quo, C. & B.M. Wu, D. W. (2005). Development of a laboratory information system for cancer collaboration projects. *27th Annual International Conference of the Engineering in Medicine and Biology Societ*, 22:108–112.

Schena, M. (1995). Quantitative monitoring of gene expression patterns with a complementary dna microarray. *Science*, 270(5235):467–470.

SOA (2010). Service oriented architecture. `http://www-01.ibm.com/software/solutions/soa/`.

Solutions, C. (2009a). Labsoft. `http://www.labsoftlims.com/`.

Solutions, L. (2009b). Sql lims. `http://www.sqllims.com/`.

Teamlösungen, T. (2009). The ehydra shark project. `http://www.enhydra.org/workflow/shark/index.html`.

Tharayil, S. & T. Kalbfleisch, A. K. (2007). Service-oriented laboratory information management system for life sciences research. In *International Conference on Services Computing*, pp. 621–627. Salt Lake City, UT.

Tomcat (2010). Apache tomcat. `http://tomcat.apache.org/`.

TWE (2009). Together workflow editor. `http://www.together.at/together/prod/twe/`.

Tyers, M. & Mann, M. (2003). From genomics to proteomics. *Nature*, 422(6928):193–197.

WfMC (2009). Workflow management coalition. `http://www.wfmc.org/`.

Wilkins, M.; Pasquali, C. & Appel, R. (1996). From proteins to proteomes: Large scale protein identification by two-dimensional electrophoresis and amino acid analysis. *Biotechnology*, 14(1):61–65.

Wilkins, M. R.; Arthur, J. W. & Junius, K. K. (2008). Information management for proteomics: a perspective. *Expert Review of Proteomics*, 5(5):663–678.

Williams, K. & D.F.Hochstrasser (1997). Proteome research: New frontiers in functional genomics. In *Springer*, pp. 1–12.

XPDL (2010). Xml process definition language. `http://www.wfmc.org/xpdl.html`.

XSLT (2010). Extensible stylesheet language for transformation. `http://www.w3.org/TR/xslt`.

Yates, J. (1998). Mass spectrometry and the age of the proteome. *J. Mass Spectrom*, 33(1):1–19.

Yauk, C. & Berndt, M. (2007). Review of the literature examining the correlation among dna microarray technologies. *Environ Mol Mutagen*, 48(5):380–394.

Anexo A

Atributos das Atividades do Fluxo de Trabalho de Proteômica

Tabela A.1. Atributos das Atividades do Fluxo de Trabalho de Proteômica - I

Protein Extraction		
Atributo	**Tipo**	**Propriedades**
Organism	Alfanumérico	not null
Source	Alfanumérico	not null
Date	Data	not null.
Isoeletric Focusing		
Atributo	**Tipo**	**Propriedades**
Strip Size	Real	not null
pH Range	Alfanumérico	not null
Gradient Type	Alfanumérico	not null
Protein Concentration	Real	not null example: $ng/\mu L$
Buffer	Alfanumérico	not null
Rehydratation Time	Alfanumérico	not null
Run File	Alfanumérico	not null
Date	Data	not null.
PAGE		
Atributo	**Tipo**	**Propriedades**
Raw Image	Alfanumérico	not null
Description	Alfanumérico	not null
Annotated Image	Arquivo	not null
Software Version	Alfanumérico	not null
Equipment	Alfanumérico	not null
Percent Acrylamide	Real	not null
Buffer	Alfanumérico	not null
Stain Details	Alfanumérico	not null
Gradient	Alfanumérico	not null
Mol Weigth Marker	Alfanumérico	not null
Run Time	Inteiro	not null
Voltage	Real	not null
Current	Real	not null
Power	Real	not null
Date	Data	not null.

Tabela A.2. Atributos das Atividades do Fluxo de Trabalho de Proteômica - II

Image Analysis		
Atributo	**Tipo**	**Propriedades**
Name	Alfanumérico	not null
File	Alfanumérico	not null
Analyse Mode	Inteiro	not null
Date	Data	not null.
Digestão		
Atributo	**Tipo**	**Propriedades**
File	Alfanumérico	not null
X	Inteiro	not null
Y	Inteiro	not null
PL	Real	not null
MW	Real	not null
Intensity	Inteiro	not null
Area	Real	not null
Vol	Real	not null
Porcent Intensity	Real	not null
Percent Vol	Real	not null
Vol Ratio	Real	not null
Slope	Inteiro	not null
Date	Data	not null.
Chromatography		
Atributo	**Tipo**	**Propriedades**
Sample	Alfanumérico	not null
Column Length	Real	not null
Column Diameter	Real	not null
Column Manufacture	Alfanumérico	not null
Gradient	Real	not null
Time	Alfanumérico	not null
FlowRate	Inteiro	not null
Temperature	Real	not null
Separation Purpose	Inteiro	not null
Date	Data	not null.

Tabela A.3. Atributos das Atividades do Fluxo de Trabalho de Proteômica - III

Database Search		
Atributo	**Tipo**	**Propriedades**
Intensity	Real	not null
SN	Real	not null
Quality Fac	Real	not null
Res	Real	not null
Area	Real	not null
Rel Intensity	Real	not null
FWHM	Real	not null
Chi	Real	not null
Peak	Real	not null
Sample Id	Inteiro	not null
Date	Data	not null.
Identification		
Atributo	**Tipo**	**Propriedades**
Sample Id	Inteiro	not null
Protein Name	Alfanumérico	not null
Protein MW	Real	not null
Protein PI	Real	not null
Pep Count	Inteiro	not null
Protein Score	Inteiro	not null
Protein Score CI	Real	not null
Total Ion Score	Inteiro	not null
Total Ion Species CI	Real	not null
Date	Data	not null.
Spectrometry		
Atributo	**Tipo**	**Propriedades**
Ion Mode	Alfanumérico	not null
File Name	Alfanumérico	not null
Description	Alfanumérico	not null
Software	Alfanumérico	not null
Deposition Technique	Alfanumérico	not null
Voltage	Inteiro	not null
Operation	Inteiro	not null
Reflectron	Inteiro	not null
Gas Type	Inteiro	not null
Gas Pressure	Inteiro	not null
Date	Data	not null.

Anexo B

Atributos das Atividades do Fluxo de Trabalho de Microarranjo

Tabela B.1. Atributos das Atividades do Fluxo de Trabalho de Microarranjo - I

RNA Extraction		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Amount of Material	Real	not null
Condition	Alfanumérico	not null, example:fresh, frozen
Sample Description Intensity	Alfanumérico	not null
Sample Name	Alfanumérico	not null
Source	Alfanumérico	not null, example: blood, root
Date	Data	not null.
Amplification		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Volume for Amplification	Real	not null, example: μL
Date	Data	not null.
Labeling		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Volume for Labeling	Real	not null example: μL
Fluorescent Dye	Alfanumérico	not null
Date	Data	not null.
Hybridization		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Slide id	Inteiro	not null
Date	Data	not null.
Cleaning and Concentration		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Volume for Cleaning	Real	not null example: μL
Date	Data	not null.

Tabela B.2. Atributos das Atividades do Fluxo de Trabalho de Microarranjo - II

Isolation		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Volume for Isolation	Real	not null example: μL
RNA Messenger Isolation Concentration	Real	not null example: $ng/\mu L$
Date	Data	not null.
Scanning		
Atributo	**Tipo**	**Propriedades**
Scanner	Alfanumérico	not null
Slide id	Real	not null
Slide Map	Real	not null
Scanning Software	Real	not null
Scanning Resolution	Real	not null
Date	Data	not null.
Image Analysis		
Atributo	**Tipo**	**Propriedades**
Image Analysis Software	Alfanumérico	not null
Date	Data	not null.
Data Analysis		
Atributo	**Tipo**	**Propriedades**
Data Analysis Software	Alfanumérico	not null
Date	Data	not null.
Data Validation		
Atributo	**Tipo**	**Propriedades**
Protocol	Alfanumérico	not null
Publication	Alfanumérico	not null
Date	Data	not null.

Anexo C

Atributos dos Resultados das Atividades do Fluxo de Trabalho de Microarranjo

Tabela C.1. Atributos dos Resultados das Atividades do Fluxo de Trabalho de Microarranjo - I

RNA Extraction		
Atributo	**Tipo**	**Propriedades**
RNA Total Concentration	Real	not null, example: $ng/\mu L$
Final Volume	Real	not null, example: μL
Gel Image	Arquivo	not null
Dosage Method	Alfanumérico	not null
Amplification		
Atributo	**Tipo**	**Propriedades**
Amplification Final Concentration	Real	not null example: ug
Amplification Final Volume	Real	not null example: μL
Dosage Method	Alfanumérico	not null.
Labeling		
Atributo	**Tipo**	**Propriedades**
Sample Description	Alfanumérico	not null
Cleaning and Concentration		
Atributo	**Tipo**	**Propriedades**
Cleaning Final Volume	Real	not null example: μL
Cleaning Final Concentration	Real	not null example: ng
Dosage Method	Alfanumérico	not null.

Tabela C.2. Atributos dos Resultados das Atividades do Fluxo de Trabalho de Microarranjo - II

Isolation		
Atributo	**Tipo**	**Propriedades**
Isolation Final Volume	Alfanumérico	not null example: μL
Isolation Final Concentration	Real	not null example: ng
Dosage Method	Alfanumérico	not null.
Scanning		
Atributo	**Tipo**	**Propriedades**
Scanning Resolution	Alfanumérico	not null
Cy3 File Localization	Alfanumérico	not null
Cy5 File Localization	Alfanumérico	not null
Image Analysis		
Atributo	**Tipo**	**Propriedades**
Data File	Arquivo	not null.
Data Analysis		
Atributo	**Tipo**	**Propriedades**
Data Analysis Field	Alfanumérico	not null
Data Analysis File	Alfanumérico	not null