Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Especialização em Informática: Ênfase: Análise de Sistemas

Eric Roberto Guimarães Rocha Aguiar

# GalaxyX - Software gerador de *interfaces* XML de programas de bioinformática para integração no *framework* Galaxy

Belo Horizonte
2011

ERIC ROBERTO GUIMARÃES ROCHA AGUIAR

# GalaxyX – Software Gerador de *Interfaces* XML de Programas de Bioinformática para Integração no *Framework* Galaxy

Monografia apresentada ao Curso de Especialização em Informática do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais, como requisito parcial para a obtenção do certificado de Especialista em Análise de Sistemas.

Área de concentração: Análise de Sistemas

Orientador: Prof. Roberto da Silva Bigonha

Belo Horizonte
2011

A282g Aguiar, Eric Roberto Guimarães Rocha.
GalaxyX: Software gerador de interfaces XML de programas de bioinformática para integração no *framework* Galaxy / Eric Roberto Guimarães Rocha Aguiar – 2011.

Orientador: Roberto da Silva Bigonha.
Monografia (Especialização)– Universidade Federal de Minas Gerais. Departamento de Ciências da Computação.

1. Computação – Monografia. 2. Frameworks – Monografias. 3. Bioinformática. I. Bigonha, Roberto da Silva. II. Universidade Federal de Minas Gerais. Departamento de Ciência da Computação. III. Título.

CDU 519.6*32(043)

## RESUMO

A bioinformática surgiu da necessidade de análise de grandes quantidades de dados biológicos. Considerando o contínuo avanço das tecnologias para geração de dados, como sequenciadores de DNA, a quantidade de dados produzidos tem aumentado consideravelmente, requerendo sistemas computacionais para analisá-los. Com a necessidade de utilização de softwares para executar essas análises surgiram outros pontos preocupantes, a falta de mão de obra especializada juntamente com a qualidade dos softwares desenvolvidos. Fatores como esses dificultam a utilização dos softwares de bioinformática visto que a maioria dos pesquisadores da área tem *background* em biologia e áreas afins. Com a identificação destes problemas iniciou-se a procura por novos recursos que facilitem a execução de softwares e a análise de dados; um exemplo dos recursos encontrados são os *frameworks.* Os *frameworks* voltados para a bioinformática dispõem de recursos que facilitam a criação e utilização de ferramentas, permitindo o desenvolvimento de soluções elaboradas através de uma série de funcionalidades intrínsecas. Dentre os *frameworks* de bioinformática, o Galaxy se destaca por apresentar características como interface unificada, validação de tipos de dados, possibilidade de integração de novas ferramentas, desenvolvimento de pipelines de forma gráfica e ampla documentação. Apesar de o Galaxy oferecer um conjunto considerável de ferramentas pré-instaladas, ele e todos os principais *frameworks* de bioinformática exigem o desenvolvimento de *scripts*/*interfaces* para que novos programas sejam integrados ao *framework.* Diante dos problemas identificados, propõe-se uma ferramenta, GalaxyX, que auxilie no processo de desenvolvimento das *interfaces* para o *framework,* visando diminuir sua complexidade. Com o GalaxyX, espera-se facilitar o processo de integração de *interfaces* para o *framework* Galaxy, refletindo em um aumento no número de *interfaces* disponíveis à comunidade.



**Palavras-chave: Bioinformática. *Framework. Interface.***

## ABASTRACT

The bioinformatics field arose from the necessity of analyzing large collections biologic data. Considering the recent advances in biological data generation technologies, such as DNA sequencers, the amount of data is considerably growing and requires computational systems to perform its analysis. Given the software requirement to execute these analyses other problems take place, for example, the lack of specialized staff and/or the low quality of the available software. As an outcome, the utilization of bioinformatics softwares becomes a complex task, since the most researches have biology-related background. The identification of these problems has led to the pursuit for new resources that facilitate the software execution and data analysis; for example, frameworks. The bioinformatics frameworks have resources that simplify the creation and utilization of tools, allowing the development of elaborated solutions through its intrinsic functionalities. Within the bioinformatics frameworks, Galaxy stands out by virtue of its features such as an unified interface, type validation, possibility of the new software integration, pipeline developing via graphical interface and wide documentation. Even though Galaxy contains a considerable set of pre-installed tools, it and all other frameworks require scripts/interfaces development in order to integrate new software. Based on the identified problems, we propose the GalaxyX tool to facilitate the Galaxy interface development process, aiming towards simplification. GalaxyX is expected to facilitate the Galaxy interface development process, consequently increasing the number of available interfaces to the community.



**Keywords: Bioinformatics. *Framework*. *Interface*.**

## LISTA DE FIGURAS

## LISTA DE QUADROS

# LISTA DE SIGLAS

API APPLICATION PROGRAMMING INTERFACE

DNA ACID DEOXYRIBONUCLEIC

DOM DOCUMENT OBJECT MODEL

JDK JAVA DEVELOPMENT KIT

JRE JAVA RUNTIME ENVIROMENT

PERL EXTRACT AND REPORT LANGUAGE

XML EXTENSIBLE MARKUP LANGUAGE

# SUMÁRIO

# 1. INTRODUÇÃO

Segundo BLANKENBERG et al., com o advento do sequenciamento de DNA acessíveis e de alto rendimento, o sequenciamento está se tornando um componente essencial em quase todos os laboratórios de genética. Dado essa grande quantidade de dados, novatos e especialistas estão enfrentando o grande desafio de tentar analisar os dados computacionalmente.

Conforme GOECKS, NEKRUTENKO e TAYLOR, assim como a quantidade de dados de sequenciamento de DNA tem crescido, ferramentas computacionais para análise desses dados vêm sido desenvolvidas continuamente. No entanto, a confiança repentina na computação criou uma "crise de informática" para investigadores das ciências da vida: os recursos computacionais podem ser difíceis de usar, e assegurar que os experimentos computacionais são confiáveis e, portanto, reproduzíveis é desafiador.

Os problemas de acessibilidade a ferramentas computacionais há muito tempo tem sido reconhecidos e estudados, sendo alvo de investimentos maciço de muitas empresas de desenvolvimento de software atingindo usuários nas mais diversas áreas.

Ainda segundo GOECKS, NEKRUTENKO e TAYLOR, sem conhecimentos de programação ou informática, os cientistas que necessitam utilizar abordagens computacionais são impedidos por problemas que vão desde instalação de ferramentas, determinar quais valores de parâmetro para o uso a combinar eficientemente múltiplas ferramentas juntas para executar uma análise. A gravidade desses problemas é fortemente evidenciada pela grande quantidade de soluções para tentar solucioná-lo, diversas soluções como BioPerl (STAJICH et al, 2002) e *frameworks* web que disponibilizam interface para ferramentas.

Após análise de diversas destas soluções, entendendo a necessidade dos pesquisadores nas mais diversas áreas, ficou claro a necessidade de um *framework* web para prover interface a diferentes ferramentas, o framework escolhido para este trabalho foi o Galaxy, desenvolvido na Universidade da Pensilvânia.

O Galaxy é um *framework* web desenvolvido com intuito de facilitar o acesso às ferramentas de análises de bioinformática (TAYLOR et al., 2007). Para alcançar o seu objetivo, o Galaxy adiciona uma interface gráfica extremamente necessária à pesquisa genômica, tornando a análise de dados acessíveis em um navegador web, e liberando os usuários de exames minuciosos dos parâmetros de linha de comando, formatos de dados e

linguagens de *script*. Entradas de dados e passos computacionais são selecionados a partir de menus gráficos e dinâmicos, e os resultados são exibidos em quadros intuitivos com resumos que incentivam os fluxos de trabalho interativos.

Como todos os *frameworks* de bioinformática para integrar novas ferramentas, o Galaxy necessita do desenvolvimento de *interfaces* para os softwares, essas *interfaces* são desenvolvidas em *Extensible Markup Language* (XML). Porém, segundo (SCHATZ, 2010), ainda são poucos os informatas que estão dispostos a dedicar tempo e recursos para o desenvolvimento de novas *interfaces* para o Galaxy, restringindo assim o número de *interfaces* disponíveis, sendo esse um dos grandes desafios do *framework* e da sua utilização.

Vendo a necessidade da comunidade acadêmica com relação a desenvolvimento de *interfaces* para programas, vislumbrei uma grande oportunidade de contribuir para o desenvolvimento da pesquisa através da possibilidade de prover o fácil desenvolvimento de novas *interfaces*, através do software GalaxyX, para serem utilizados pelos pesquisadores sem grande conhecimento de informática buscando facilitar o procedimento de desenvolvimento e integração de novas ferramentas ao Galaxy.

## 1.2 Organização do trabalho

O seguinte trabalho foi organizado da seguinte forma, incialmente será apresentado uma introdução à bioinformática, focando principalmente nos seus desafios e na importância dos *frameworks* para ultrapassá-los. Posteriormente será discutido um pouco mais sobre os frameworks de bioinformática, discutindo sobre o seu surgimento e aspectos da sua utilização. Assim que discutido sobre os principais frameworks de bioinformática, o foco será o Galaxy, onde aspectos da sua plataforma serão analisados, juntamente com características e especificidades. Entendido o funcionamento do Galaxy e aspectos de sua implementação, será discutido sobre o GalaxyX, programa gerador de *interfaces* para o Galaxy através de interface gráfica. Após discorrer sobre o GalaxyX, serão apresentados os resultados e discussão do projeto a fim de levantar dados para análise final do projeto. Finalmente, após a análise de resultados e discussão, a conclusão será apresentada e sugestões para trabalhos futuros aludidos.

# 2 OBJETIVOS

## 2.1 Objetivo Geral

Estudar o *framework* Galaxy (GIARDINE et al, 2005), artigos e documentação, de modo a obter embasamento teórico necessário ao desenvolvimento de um software gerador de *interfaces* XML, de modo gráfico, para que assim, pessoas com pouco conhecimento específico em informática ou programação possam desenvolver novas *interfaces* de softwares para integração no *framework*.

## 2.2 Objetivos específicos

- Realizar levantamento bibliográfico do *framework* Galaxy;
- Aprofundar conhecimento sobre meios de integração de programas no *framework*;
- Analisar técnicas de validação de *interfaces*;
- Estudar a linguagem XML utilizada para o desenvolvimento das *interfaces* dos programas.
- Definir tecnologias a se utilizar para o desenvolvimento do software.
- Desenvolver software gerador de *interfaces*;
- Validar software gerador de *interfaces* através de integração de programas no *framework*;

# 3. A BIOINFORMÁTICA

A bioinformática é uma área que vem crescendo constantemente nos principais centros de pesquisa devido ao seu papel de integrar os problemas biológicos às soluções computacionais. Mas o que é a bioinformática? Em que aspectos ela pode ajudar no desenvolvimento da pesquisa e quais os principais desafios desta área que promete tanto? Nesse tópico iremos discorrer sobre a bioinformática e sua importância.

## 3.1 O que é Bioinformática

O termo bioinformática foi utilizado pela primeira vez por Paulien Hogeweg *and* Ben Hesper em 1978 para o estudo de processos de informática aplicado a sistemas biológicos (HOGEWEG; HESPER, 1978). Em seu trabalho a bioinformática foi definida como a aplicação de estatística e ciências da computação no campo da biologia molecular. A multidisciplinaridade da bioinformática hoje pode ser vista pela abrangência da sua aplicação, sendo utilizada também em bioquímica e genética principalmente.

Existem inúmeras outras definições para a bioinformática, todas elas explicitando seu papel no armazenamento e relacionamento de dados biológicos, utilizando para isso métodos computacionais e algoritmos matemáticos.

## 3.2 Como surgiu a Bioinformática

Apesar do pouco tempo de plena utilização da bioinformática, já se faz presente normalmente em qualquer discussão relacionada à biologia, principalmente, pelo seu papel essencial na análise de dados biológicos, sejam eles de pequeno ou grande porte.

Para se explicar o surgimento da bioinformática basta analisar o crescente avanço das técnicas biotecnológicas, principalmente do sequenciamento de Ácido Desoxirribonucleico (DNA), figura 1. Este que, por sua vez, foi o estopim para a utilização da computação no auxilio a análise de dados biológicos.

Desde que os sequenciadores capilares surgiram na década de 90, a quantidade de dados biológicos alcançou níveis inimagináveis à sua análise manual; montagens de genomas ou o trans*cript*oma de um organismo se tornava cada dia mais viável, entretanto a sua análise era custosa e principalmente demorada, justificando assim o uso da computação principalmente na genética.

Com o avanço do sequenciamento capilar e de outras tecnologias que provem o sequenciamento de DNA gerando mais dados, de forma mais rápida e barata, a velocidade de dados gerados com necessidade de análise cresceu exponencialmente, assim, entendendo-se a utilização e crescimento constantes da bioinformática.

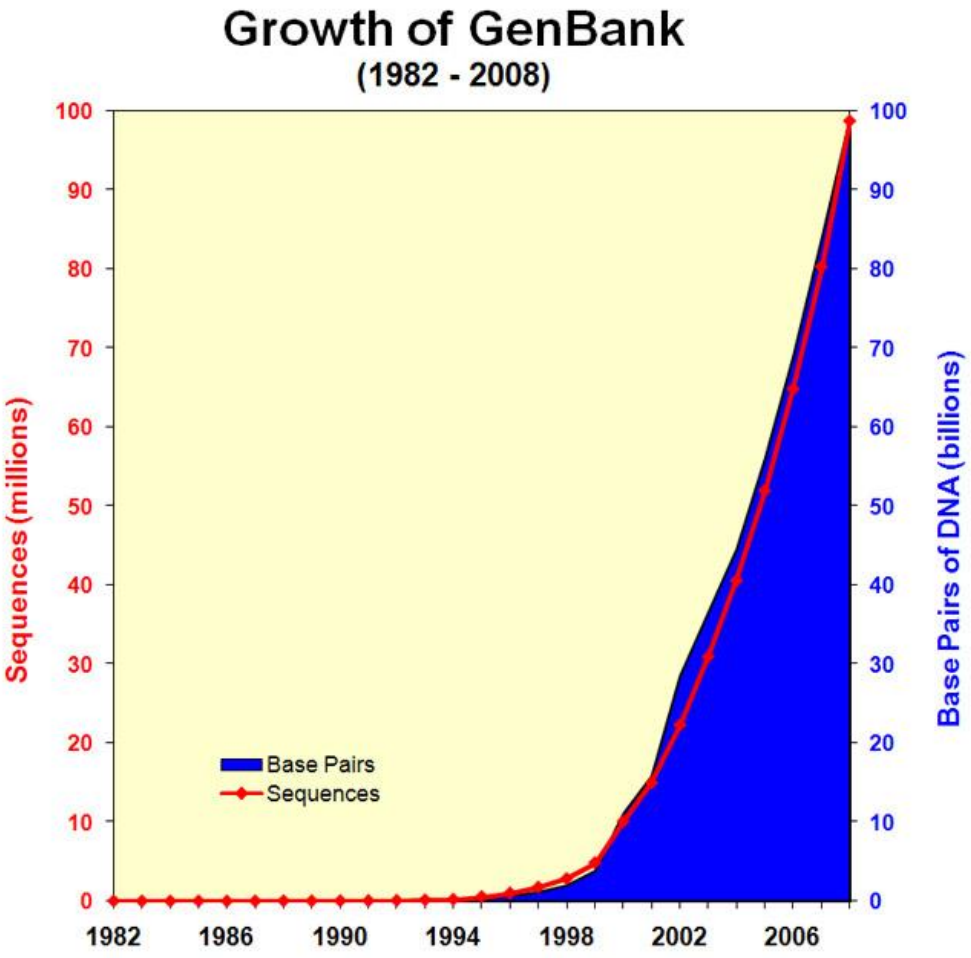


Figura 1. Crescimento do GenBank (Banco de dados de genomas de organismos).
Fonte: GenBank (http://www.ncbi.nlm.nih.gov/genbank/ genbankstats.html.), 2009.

## 3.3 Principais desafios

Os desafios da bioinformática vão desde a quantidade de dados gerada até a mão de obra qualificada para análise dos dados. Neste tópico vamos entender algumas das principais dificuldades e porque elas existem.

### 3.3.1. *Análise dos dados*

A tecnologia refletiu diretamente na bioinformática, todas as máquinas anteriormente utilizadas no inicio da bioinformática evoluíram, assim a quantidade de dados com necessidade de análise continua a crescer exigindo cada vez mais dos sistemas de análise. Com a facilidade de geração de dados, os dados tendem a aumentar muito rapidamente, e o crescimento do número de profissionais dispostos a trabalhar na área e as ferramentas de bioinformática não cresciam na mesma proporção.

Maior quantidade de dados implica em maior complexidade de análise. A complexidade das análises é resolvida através dos programas computacionais, entretanto, quanto mais complexos os dados, mais eficiente necessita ser o software para lidar com os dados, e mais robusto o hardware, para responder às necessidades do software.

Segundo Pennisi, os computadores são essenciais para análise e armazenamento de dados, mas seu poder não está aumentando rápido o suficiente e seus custos estão diminuindo muito lentamente para acompanhar a geração de dados.

### 3.3.2 *Mão de obra especializada*

A bioinformática ainda é considerada uma área nova no Brasil, visto que o seu crescimento está ligado diretamente à evolução das tecnologias, principalmente para geração de dados, como os sequenciadores. À medida que a quantidade de dados gerados crescia, observou-se a necessidade de mão de obra especializada para analisá-los, pois os biólogos e bioquímicos se empregavam em procedimentos de bancada, assim surgiam os primeiros

indícios de uma aproximação da biologia e da informática.

A análise de dados biológicos não se configura uma tarefa trivial para os informatas e a aplicação da informática nos dados biológicos nem sempre é uma tarefa muito fácil, assim, a necessidade de profissionais com embasamento biológicos e conhecimentos de informática se fez e faz cada vez mais presente no dia-a-dia dos grandes laboratórios.

Como explicitado por SCHNEIDER, à bioinformática vem se tornando cada vez mais central para a investigação nas ciências da vida molecular. Com a escassez de mão de obra qualificada, a necessidade de treinar não-bioinformatas para aproveitar ao máximo os recursos da bioinformática está crescendo e continuará a crescer.

### 3.3.3 *Ferramentas de bioinformática*

As ferramentas de informática tem um papel fundamental na análise dos dados biológicos; a quantidade e complexidade dos dados vêm tornando a análise dos dados cada vez mais custosa, assim, a necessidade de análises *in silico* cresce constantemente, exigindo cada vez mais dos programas acurácia e eficiência na sua utilização, além de *interfaces* amigáveis para prover amplo acesso da comunidade.

Além da complexidade dos dados, outro fator determinante para a pequena quantidade de softwares livres disponíveis é o número de programadores/pesquisadores disposto a ingressar na programação propriamente dita. Os principais provedores de softwares para análises de dados biológicos são instituições de pesquisa mantidas pelo governo e universidades, porém, como a bioinformática ainda está em fase de desenvolvimento e crescimento no mundo, o desenvolvimento e disponibilização desses softwares é lento quando comparado com a velocidade de geração de dados.

Ações como a do grupo de pesquisa (BRYANT, 2011) vêm buscando alcançar uma padronização que possa ser utilizada pelos bioinformatas no desenvolvimento de ferramentas. Com a padronização de código, o objetivo é acelerar o processo de desenvolvimento de ferramentas, assim, procurando sanar um dos principais problemas da bioinformática, a análise dos dados.

# 4. *FRAMEWORKS* DE BIOINFORMÁTICA

Os *Frameworks* surgiram inicialmente na informática com o intuito de reaproveitar código, utilizando para isso códigos genéricos que podem ser sobescritos ou especializados de maneira a adequá-los ao interesse da aplicação. Aqui vamos apresentar um pouco mais sobre qual o seu papel, aspectos do seu funcionamento e quais são os principais *frameworks* na bioinformática juntamente com suas especificidades.

## 4.1 O que é um *framework*

Um *framework* consiste em um conjunto de códigos genéricos que provem funcionalidades distintas incorporando características como alto grau de reusabilidade e abstração. Os *framework*s se diferem das bibliotecas por conter um conjunto de características distintas comparadas com estas ou até mesmo com aplicações normais.

As características principais que diferem *framework*s, bibliotecas e aplicações normais (criadas pelo usuário) são:

**Inversão de controle:** o fluxo de aplicação não é definida pelo programa que faz a requisição como em aplicações normais, mas sim pela estrutura interna do *framework.*

**Comportamento padrão:** os *framework*s seguem um comportamento padrão, determinados pelo seu fluxo interno, não se submetendo a fluxos incomuns.

**Extensibilidade:** os *frameworks* têm como principal característica a reusabilidade, podendo as suas classes ser sobrescritas ou especializadas, provendo funcionalidade específica segundo necessidade da aplicação.

**Código não modificável:** Os *framework*s, em geral, não permitem a modificação dos seus códigos, visto que desde a sua concepção, a reusabilidade seja sua característica predominante, para isso utiliza códigos genéricos.

Em geral os *framework*s são utilizados como suporte para o desenvolvimento e uso de aplicações, podendo incluir programas de suporte, bibliotecas de código, linguagens de *script* e outros.

## 4.2 Como funcionam os *frameworks* na bioinformática

Os *framework*s foram integrados à bioinformática segundo a necessidade de facilitar o desenvolvimento e utilização de softwares.

Devido ao pouco tempo da área "bioinformática", existem ainda poucos profissionais capacitados ao desenvolvimento de aplicações, assim, pesquisadores de áreas afins se aventuram no estudo de linguagens de programação e desenvolvimento de softwares; além da dificuldade no desenvolvimento de software, estes pesquisadores se encontram constantemente com problema na utilização de alguns softwares, sendo este um grande problema para a comunidade acadêmica.

Quando observado o cenário atual da bioinformática e dos usuários de softwares de bioinformática, entende-se perfeitamente a mudança de foco dos *framework*s utilizados nestes; enquanto que na computação os *framework*s são empregados principalmente para o desenvolvimento de softwares, na bioinformática eles trazem consigo outro papel fundamental, a possibilidade de integração e execução de softwares através de uma interface unificada.

Os *framework*s que surgiram unicamente para integração de programas vêm ganhando espaço à medida que a quantidade de software de bioinformática cresce, assim, os *framework*s se tornam cada vez mais importantes como ferramenta facilitadora no uso da bioinformática.

## 4.3 Principais frameworks de bioinformática

Com o avanço da tecnologia, novos dados são gerados a todo o momento, dados esses que necessitam ser analisados. Para a análise destes dados são utilizados softwares específicos a cada área, porém como nem todos os pesquisadores que necessitam utilizar estes softwares tem um *background* computacional, estes constantemente se deparavam com problemas na execução de softwares, daí começaram a surgir alternativas para auxiliá-los na execução destes softwares.

Os *framework*s de bioinformática surgiram para auxiliar nas análises de dados biológicos provendo *interfaces* amigáveis, maleabilidade e, principalmente, integração de

vários programas em um único universo, conforme BLANKENBERG, características fundamentais no contexto da bioinformática.

Atualmente existem vários *frameworks* de bioinformática disponíveis para download e/ou utilização web, dentre eles estão o Galaxy, genePattern, REICH et al, Mobyle NÉRON et al, e Taverna, HULL et al.

Com o objetivo de facilitar a vida dos usuários, todos os *framework*s seguem um mesmo padrão, provendo integração de novas ferramentas e reutilização de resultados, porém todos eles exigem a programação de *interfaces* na integração de novos softwares.

O Taverna é considerado um dos precursores dentre os *framework*s de bioinformática, ele foi desenvolvido em 2006 na Universidade de Manchester. Também em 2006 surgiu o genePattern, desenvolvido pelo Broad Institute. O Instituo Pasteur apresentou o Mobyle em 2009. Quanto ao Galaxy, Universidade da Pensilvânia, apesar de ter sido apresentado inicialmente em 2005 apenas se tornou popular a partir de 2007, quando apresentou uma versão mais consistente e com suficientes diferenciais para atrair mais usuários ao *framework*.

# 5. O GALAXY

O Galaxy é um *framework* voltado para web, que se apresenta como uma plataforma de análise, realizando diversos tipos de análises nos mais diversos tipos de dados biológicos, podendo ser obtidos de diversas fontes diferentes, como bancos de dados biológicos e *web servers*. Além de oferecer interface para várias das ferramentas de bioinformática conhecidas, o Galaxy traz ainda a possibilidade de criação de pipelines, armazenamento de resultados e integração de novos programas. Neste tópico será apresentada a arquitetura do Galaxy, suas principais características/especificidades e como novos softwares podem ser integrados.

## 5.1 Arquitetura

O *framework* Galaxy pode ser dividido em duas partes fundamentais, o "*core"*, onde estão presentes as bibliotecas, e "*tools*", onde estão presentes arquivos relacionados às ferramentas integradas ao *framework,* figura 2. As bibliotecas do Galaxy foram desenvolvidas em Python, sendo responsáveis também pelo provimento do serviço de servidor web.

O estudo do *core* do Galaxy é essencial caso alguma modificação seja necessária no que se diz respeito ao funcionamento do Galaxy, exigindo conhecimento específico de Python e *web server*.

O módulo "*tools*" armazena todas as ferramentas nativas do Galaxy. Este módulo também é responsável por toda a gerência de ferramentas. Outra das suas atribuições é o controle dos tipos de dados e dados complementares das ferramentas, como arquivos externos necessários a execução de uma determinada ferramenta.

Para que seja integrado um novo software ao Galaxy normalmente são necessários dois arquivos além da instalação da ferramenta no servidor onde o Galaxy será instalado, o de *interface,* responsável pela definição dos parâmetros e um de *script,* para caso seja necessário adequar os dados vindos da *interface* à ferramenta.

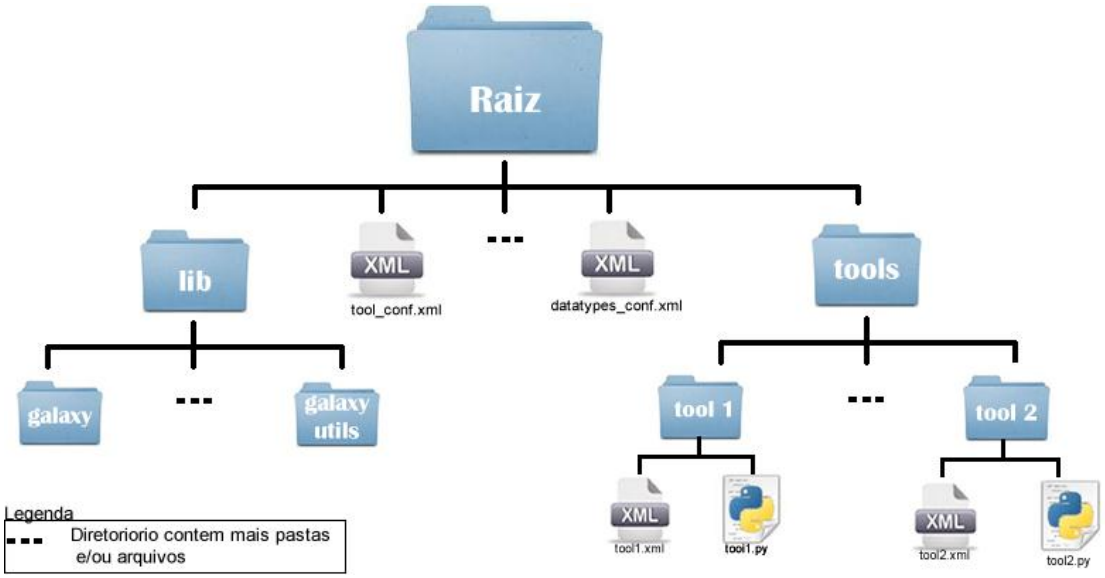


Figura 2. Estrutura básica de diretórios do Galaxy.
Fonte: próprio autor

O Galaxy mantém uma série de arquivos de configuração, normalmente escritos em XML, dentre eles estão o arquivo de configuração das ferramentas, "tool_conf.xml", e o arquivo de tipos de dados, "datatypes.xml". O arquivo de configuração das ferramentas contém entradas para todas as ferramentas disponíveis, especificando o caminho para a sua *interface*, arquivo XML, o descreve. O arquivo "datatypes_conf.xml" compreende todos os tipos de dados aceitos pelo Galaxy, tais como tabular, texto, dentre outros tipos conhecidos na bioinformática como *fasta* e *esd*, um exemplo do arquivo de tipos de dados pode ser visto no quadro 1.

**Quadro 1- Arquivo datatypes_conf.xml**

```
<datatypes>

      <datatype extension="ace" type="galaxy.datatypes.data:Text"/>
      <datatype extension="singlets"
      type="galaxy.datatypes.sequence:Fasta"/>
      <datatype extension="singlets.qual"
      type="galaxy.datatypes.qualityscore:QualityScore"/>
      <datatype extension="contigs"
      type="galaxy.datatypes.sequence:Fasta"/>
      <datatype extension="screen"
      type="galaxy.datatypes.sequence:Fasta"/>
      <datatype extension="gff3" type="galaxy.datatypes.interval:Gff3"
      display_in_upload="true"/>
      ...

<datatypes>
```

## 5.2 Características

O Galaxy, como *framework*, apresenta características fundamentais como reusabilidade e comportamento estável. Além destas, o Galaxy traz consigo funcionalidades fundamentais ao desenvolvimento do seu propósito que é prover de forma amigável e eficiente acesso a programas de bioinformática.

Maleabilidade, padronização e a unificada interface de usuário permitem ao usuário e desenvolvedores facilidade no manuseio do *framework*. O Galaxy ainda disponibiliza um conjunto amplo de características que ajudam no desenvolvimento de novas *interfaces*, como linguagem padronizada para integrar novos softwares e documentação ampla e detalhada.

Permitir o desenvolvimento de pipelines via interface gráfica se apresenta como característica fundamental ao sucesso do *framework*, principalmente considerando os principiantes na bioinformática, sendo uma alternativa eficaz para a utilização de pipelines criados anteriormente somente via linda de comando.

O armazenamento de resultados e a possiblidade de reutilização dos dados tornam o Galaxy um framework extremamente produtivo. Com a possibilidade da criação de pipelines e da reutilização de resultados, torna-se trivial a re-execução de pipelines, tarefa essa muito comum para análise de parâmetros e resultados.

Quando utilizando o Galaxy, o usuário não precisa aprender a programar ou se preocupar com detalhes de implementação, o *framework* provê uma interface unificada para análise genômica, o que acredita (GOECKS; NEKRUTENKO; TAYLOR, 2010) ser a sua principal característica.

## 5.3 Especificidades do Galaxy

O Galaxy como *framework* robusto que se apresenta trouxe consigo algumas especificidades interessantes, tanto do ponto de vista do desenvolvedor, quanto do ponto de vista do usuário, que sem duvida é o mais beneficiado com essas características.

Para viabilizar o desenvolvimento das *interfaces*, o Galaxy definiu uma sintaxe específica para o desenvolvimento dos seus componentes, componentes de interface, seguindo

padrões da XML. Com a sintaxe definida, é possível definir diferentes parâmetros que podem ser apresentados de modo diferentes, e diferentes tipos de dados, tipos de dados esses nativos do Galaxy ou integrados posteriormente.

O *framework* Galaxy permite que o desenvolvedor crie novos tipos de dados primitivos, tipos esses utilizados no momento de definição dos parâmetros. Além do desenvolvimento de tipo de dados diferentes, o Galaxy prove a criação e integração de mecanismo de validação para "reconhecimento automático" do tipo de dados que está sendo utilizado como entrada, no caso de arquivos.

Permitir criar pipelines via interface gráfica talvez se apresente como a especificidade mais bem recebida pelos usuários. Através da interface é possível integrar programas simplesmente "clicando e arrastando" como pode ser visto na figura 3. No ato do desenvolvimento, o usuário ainda conta com validação de tipos entre as ferramentas, onde somente tipos de dados compatíveis podem ser interligados. Isso acontece para que o usuário no momento do desenvolvimento do pipeline possa ter certeza que os dados são compatíveis e que os softwares funcionarão corretamente.

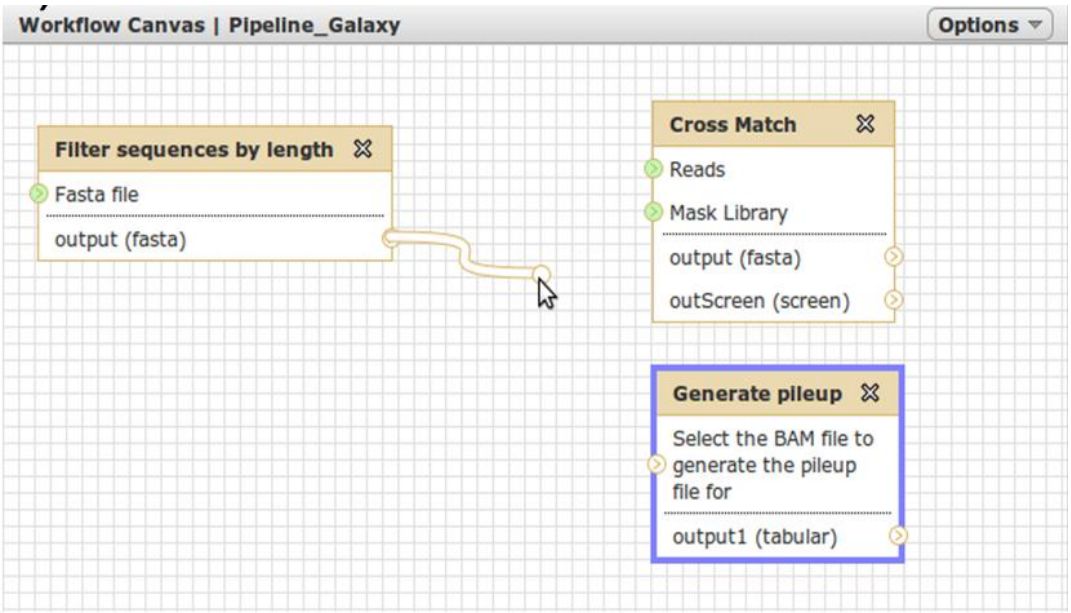


Figura 3. Exemplo de criação de workflow. Fonte: próprio autor.

## 5.4 Integração de novos programas

A integração de novos programas no Galaxy é feita de maneira padronizada.

Todos os passos necessários para integrar um novo software podem ser encontrados através do site oficial do projeto (www.usegalaxy.org).

Nas subseções seguintes serão apresentados os passos necessários para a integração de novas ferramentas ao *framework*, focando essencialmente no desenvolvimento das *interfaces.*

### 5.4.1 *Como integrar*

Para que um software seja passível de integração ele deve ser permitir sua de execução via linha de comando, visto que o papel do Galaxy é prover interface ao software. A partir do momento em que a ferramenta atende ao requisito de execução via linha de comando, pode-se definir como ela será apresentada no *framework*, discriminando tipos de dados, parâmetros e arquivos de saída.

Como medida para prover padronização os desenvolvedores do Galaxy adotaram XML como linguagem padrão para arquivos de configuração e *interfaces* de softwares. Basicamente para a inserção das ferramentas é necessário desenvolver a *interface* utilizando XML e alterar o arquivo de definição de ferramentas "tool_conf.xml" inserindo uma entrada para a interface desenvolvida, uma amostra deste arquivo pode ser visto no quadro 2 onde foi inserida a ferramenta "exemplo.xml".

No ato do desenvolvimento das *interfaces*, casualmente podem se fazer necessários *script*s intermediários que realizem o papel de parsear os dados entre a *interface* e o programa propriamente dito, esses *script*s são chamados de *parsers.* Falaremos mais sobre eles na sessão 4.3.

**Quadro 2 - Arquivo tool_conf.xml**

```
<?xml version="1.0"?>
<toolbox>
  <section name="mytools" id="mytool">
    <tool file="data_source/upload.xml"/>
    <tool file="data_source/upload.mod.xml"/>
    <tool file="data_source/ucsc_tablebrowser.xml" />
    <tool file="data_source/ucsc_tablebrowser_test.xml" />
    <tool file="data_source/ucsc_browser_archaea.xml" />
    <tool file="data_source/bx_browser.xml" />
    <tool file="data_source/microbial_import.xml" />
   <!--ferramenta inserida -->
    <tool file="minhasFerramentas/exemplo.xml" />
…
   </section>
</toolbox>
```

Nos tópicos seguintes serão apresentados um pouco mais sobre a linguagem XML e as especificidades do XML adotado pelo Galaxy.

### 5.4.2 *A linguagem XML e o Galaxy*

Para aplicações grandes que envolvem grande quantidade de pessoas, definir um padrão é necessário. O XML foi o padrão adotado para o desenvolvimento de novas *interfaces* para o Galaxy, provendo um padrão de desenvolvimento consolidado e funcional aos desenvolvedores. Ainda conforme (HOULDING, 2001), vale destacar a extensibilidade provida pela linguagem.

A XML é a recentemente introduzida metalinguagem padrão na Web. Ela provê um conjunto de regras para o desenvolvimento de metadados para transferência de informações em campos específicos. A XML ainda permite o desenvolvimento de linguagens de marcação que descrevem qual informação e como essas informações serão apresentadas, característica destacada já há algum tempo atrás por (HOULDING, 2001).

Utilizada como base para o desenvolvimento das *interfaces* e arquivos de configuração do Galaxy, conhecer a linguagem XML é de grande importância aos desenvolvedores. No quadro 3 é possível observar um exemplo de uma interface de software desenvolvida para o Galaxy e da linguagem utilizada que é derivada do XML.

Para o desenvolvimento das *interfaces* os desenvolvedores definiram tags especificas ao *framework*,

Nas seções seguintes vamos entender melhor as especificidades do Galaxy e de sua linguagem de desenvolvimento de *interfaces*.

**Quadro 3 - Exemplo de XML de interface de software**

```
<tool id="Exemplo" name="Ferramenta de exemplo">
  <description>Exemplificar uma interface de software</description>
  <command interpreter="perl">exemplificar.pl $input
$output</command>
  <inputs>
    <param format="fasta" name="input" type="data" label="Source
file"/>
  </inputs>
  <outputs>
    <data format="tabular" name="output" />
  </outputs>
  <help>
     Gerando um arquivo tabular com os identificadores das
sequências em uma coluna e o tamanho em outra coluna.
  </help>
</tool>
```

### 5.4.3 *Elementos básicos de uma interface*

Para que uma interface seja integrada ao Galaxy e funcione corretamente ela deve conter os elementos básicos de uma interface que são definidos na documentação do *framework*. Como dito anteriormente toda a interface é definida utilizando como base a linguagem XML.

Qualquer interface de programa, por mais básico que ele seja, deve conter as seguintes seções, subseções e *tags*:

- **Seção *tool***: é a ferramenta propriamente dita, toda a configuração da ferramenta fica inserida dentro da seção tool, incluindo subseções e *tags*. Ainda na *tag* tool definem-se o nome e um id único para a ferramenta.

- *Tag description*: define a descrição da ferramenta, descrição essa que será mostrada ao usuário na interface do *framework.*
- *Tag command*: nessa *tag* são definidos o interpretador do programa que será executado, por exemplo, Perl, e o comando propriamente dito seguido dos parâmetros que serão passados ao software.
- **Subseção *inputs***: seção sem necessidade de parâmetros de configuração onde são inseridos todos os *inputs*, elementos de entrada de dados, para o software em questão.
- *Tag param*: tag utilizada para definição de tipo de parâmetro e dado que será aceito para aquele determinado parâmetro de configuração do software.
- *Tag conditional*: elemento opcional utilizado para aplicar expressões condicionais do tipo "se selecionada a opção A então faça ISSO" baseadas em escolha em um parâmetro do tipo *select*. Dentro de cada opção devem existir parâmetros.
- **Subseção *outputs***: seção onde são inseridos todos os elementos que serão entendidos pelo Galaxy como elementos de saída. Devem ser definidos elementos do tipo *data* dentro da subseção.
- *Tag data*: elemento utilizado para definir qual o nome e o tipo de dados que se espera como saída da execução do programa.
- **Seção *help***: elemento responsável por armazenar informações referentes à ajuda do programa como principais parâmetros, explicação do funcionamento entre outros.

Todos os parâmetros utilizados para configurar características das *tags* e subseções são definidas através do sistema "chave=valor". Um exemplo da utilização desses elementos pode ser visto no quadro 2 na subseção 5.4.2.

### 5.4.4 *Tipos de dados dos parâmetros*

O *framework* Galaxy apresenta uma serie de tipos de dados primitivos possíveis, desde os mais comuns a qualquer linguagem de programação, como *text*, *integer* e *float*, até

tipos de dados mais específicos de linguagem web, como *select* e *hidden.*

Os tipos de dados são extremamente importantes no desenvolvimento e integração de ferramentas devido ao fato do Galaxy prover validação de tipos, e principalmente, pela compatibilidade do tipo de dado com o software em questão.

### 5.4.5 *Tipos de dados dos arquivos de entrada e saída*

Juntamente com a validação de tipos de parâmetros, o Galaxy fornece ao desenvolvedor das *interfaces* alguns tipos pré-definidos de dados, tipos esses mais comuns na bioinformática. Estes tipos de dados contam com validação pré-estabelecida como forma de prevenção de erros por parte do usuário da ferramenta.

Novos tipos de dados podem ser inseridos facilmente no *framework* através de derivação de tipos ou através de definição de tipos novos. Na derivação de tipos, tipos já pré-definidos podem ser utilizados como base para novos tipos, onde o que muda é somente o conteúdo, porém a estrutura continua a mesma, exemplo de um arquivo no formato *fasta*, quadro 4, e no formato *csfasta*, quadro 5. Quando da definição de novos tipos é necessário o desenvolvimento de um validador para o tipo, para que o tipo possa ser automaticamente reconhecido no momento da validação de tipos. O validador tem de ser escrito em Python e inserido no arquivo de tipos de dados, assim o tipo inserido poderá estar disponível para acesso.

Todo o descritivo do processo de desenvolvimento de novos tipos juntamente com o procedimento para escrita de novos validadores de tipos de dados podem ser encontrados na página wiki do projeto Galaxy (Galaxy wiki, 2011).

**Quadro 4 - Exemplo de arquivo no formato fasta**

```
>SMP_1234
gatcgaagaaaggtcgaagaaaggtagctagatcgaagaaaggaaccaa
>SMP_33
atcttattaaatctatggatgctagtgaagaaaggtcgaagaaactcgt
>SMP_323
aaaattcggatggaagctggtcgatggatgctagtgaagaaagcttaaa
>SMP_323
aaaattcggaacagtcgtagctagtccttgaatttttcggatcggggt
```

**Quadro 5 - Exemplo de arquivo no formato csfasta**

```
>SMP_767
t0102201002300220123221223012322122311112111
>SMP_690
t2003201022302103021221122220123221223012322
>SMP_329
A0320123220111020033321112230210302122112221
>SMP_323
A0301203020201111020200001010001222230202333
```

O uso correto dos tipos de dados é fundamental à utilização do usuário. Todos os programas que o usuário poderá utilizar nos seus dados estão relacionados diretamente aos tipos de dados de entrada e saída. O tipo de dado de saída ainda poderá ser reutilizado em outras ferramentas através de utilização simples, aquela em que o usuário executa um programa após o outro, ou utilização no pipeline, onde uma concatenação de programas é definida em uma ordem específica para que sejam executados um após o outro automaticamente.

## 5.5 *Parsers*

Para que as *interfaces* do Galaxy pudessem "conversar" com programas complexos, os desenvolvedores do Galaxy propuseram o uso de arquivos intermediários de *script*, arquivos esses denominados *parsers*. Os *parsers* tem o papel de diminuir a complexidade dos dados advindos das *interfaces* de modo que o software consiga entender esses dados e funcione corretamente.

Linguagens de *script* como Perl e Python são altamente recomendadas para o desenvolvimento dos *parsers*, visto que essas linguagens tem como ponto forte a manipulação de textos, principal propósito dos *parsers*. Um exemplo de *parser* em desenvolvido na linguagem Perl pode ser visto no quadro 6.

**Quadro 6 - Exemplo de *script parser***

```
#!/usr/bin/perl

use Switch;

#recebendo parametros da interface xml
my $sequence    = @ARGV[0];
my $output      = @ARGV[1];
my $log         = @ARGV[2];

my $lineExecution;

#monta linha de execução
$lineExecution = "patmatmotifs -sequence $sequence -outfile $output >
$log";

#executando a montagem do programa como linha de comando
my $exec = `$lineExecution`;
```

# 6. GALAXYX

## 6.1 Arquitetura

O Galaxy foi projetado para estar disponível nos principais sistemas operacionais como Windows, Unix/Linux e Mac OS, para isso foi utilizada a linguagem Java.

Como o framework Galaxy foi desenvolvido baseando seus arquivos de configuração e *interfaces* de softwares baseadas em XML, utilizou-se o *framework* JDOM para auxiliar no processo de manipulação de arquivos XML.

O software foi dividido em dois módulos principais, o módulo gerador de *interfaces*, responsável por tratar as informações inseridas pelo o usuário e gerar a interface XML propriamente dita e o módulo de validação de XML, responsável por validar as *interfaces* XML geradas pelo módulo gerador de *interfaces*. Além das *interfaces* geradas pelo GalaxyX, o módulo validador de *interfaces* poderá validar outras *interfaces* do *framework* Galaxy, mesmo que não tenham sido geradas através do software aqui proposto.

## 6.2 Java

A linguagem Java foi escolhida como linguagem de programação no desenvolvimento do GalaxyX por ser altamente portável. A mesma aplicação pode ser executada no Windows, Unix e Mac OS somente instalando o Java *Runtime Enviroment* (JRE).

Características como orientação a objetivo e vasta documentação são essenciais a projetos que envolvam bibliotecas de terceiros e interface gráfica. Assim acredito que a linguagem Java supre todas as necessidades do projeto garantindo ainda portabilidade.

A linguagem Java é mantida atualmente pela Oracle. O JRE está disponível para download através do site (www.java.com).

### 6.2.1 *O framework JDOM*

JDOM é um framework Java para manipulação de documentos XML. Suas funcionalidades se integram com DOM (*Document Object Model*). O JDOM foi desenvolvido especificamente para a utilização com a linguagem Java, assim tornando-se uma API (*Application Programming Interface*) muito completa para manipulação de arquivos XML.

A utilização do JDOM no projeto é justificada pelo fato de ter uma documentação ampla e utilização simples. Outras características como trabalhar com *namespaces* XML e apresentar validação para leitura de arquivos XML são extremamente importantes à sua utilização. Estas características foram amplamente utilizadas no desenvolvimento dos módulos gerador de interface e de validação de XML.

O JDOM é um framework *open source* e está disponível para *download* através do site do projeto (www.jdom.org).

### 6.2.2 *A API Log4J*

A Log4J é uma API desenhada para a linguagem JAVA voltada à geração de logs. Com essa API é possível criar logs de diferentes tipos definidos que pode ser definido pelo usuário. O usuário pode escolher várias características a serem exibidas no log além do tipo de mensagem apresentada segundo um *level* de log como *error* ou *debug*. No projeto GalaxyX a Log4J foi utilizada para auxilio a descoberta de erros e principalmente para manter um padrão nos logs, visto que está é uma boa prática de programação.

A Log4J é um projeto *open-source* mantido pela Fundação Apache e está disponível para download através do site do projeto (http://logging.apache.org).

## 6.3 Módulo de Geração de Interface

O módulo de geração de *interfaces* é responsável por transformar em interface XML os parâmetros definidos graficamente pelo usuário na utilização da ferramenta

GalaxyX. A interface foi desenvolvida voltada a prevenção de erros por parte do usuário. Toda a sequência de etapas é baseada nos elementos necessários para que uma interface funcione corretamente no *framework* Galaxy.

Para atingir um universo maior de usuários o idioma adotada na interface do GalaxyX foi o inglês, visto que o Inglês é o padrão da sociedade acadêmica e na maioria dos países é tido como segundo idioma, esperando assim alcançar um número maior de usuários.

Com o objetivo de prevenir qualquer erro do usuário, todos os campos obrigatórios foram sinalizados e validação dos mesmos é feita a cada tela do sistema como pode ser visto na figura 4.

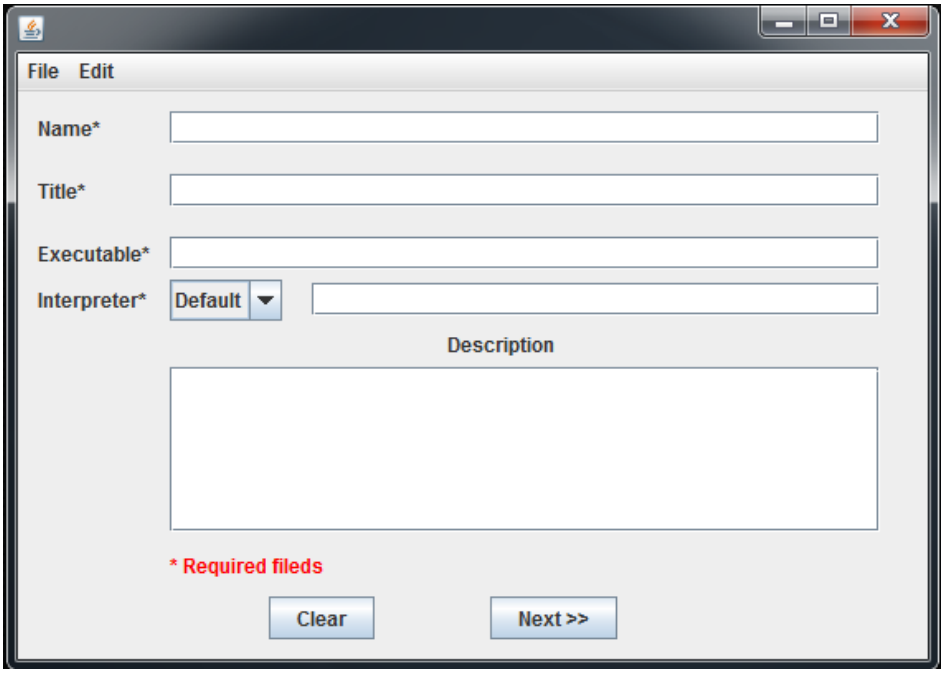


Figura 4. Tela de definição de ferramenta.
Fonte: próprio autor.

### 6.4 Módulo de Validação de XML

O modulo de validação de *interfaces* foi desenvolvido para validar as *interfaces* desenvolvidas através do GalaxyX sendo utilizado no momento da conclusão do processo de desenvolvimento. Além de utilizado neste processo, este módulo apresenta a possibilidade de ser utilizado para validar outras *interfaces* já desenvolvidas, mesmo que elas não tenham sido desenvolvidas através do GalaxyX, provendo assim mais uma característica útil ao usuário.

Para o desenvolvimento do módulo foi utilizado como base o validar interno integrado ao *framework* Galaxy que leva em consideração todas as opções disponíveis nas *interfaces* de softwares.

O processo de validação é composto de dois validadores diferentes, o primeiro é um validador de *tags* XML, responsável por garantir a utilização correta do padrão XML, e o segundo validador é utilizado para validar a sintaxe própria do Galaxy.

O validador de *tags* XML foi desenvolvido utilizando a biblioteca JDOM. O validador de sintaxe é o mesmo que é utilizado no framework Galaxy, garantindo assim que o software possa ser integrado ao *framework*. A interface do validador XML pode ser visto na figura 5.

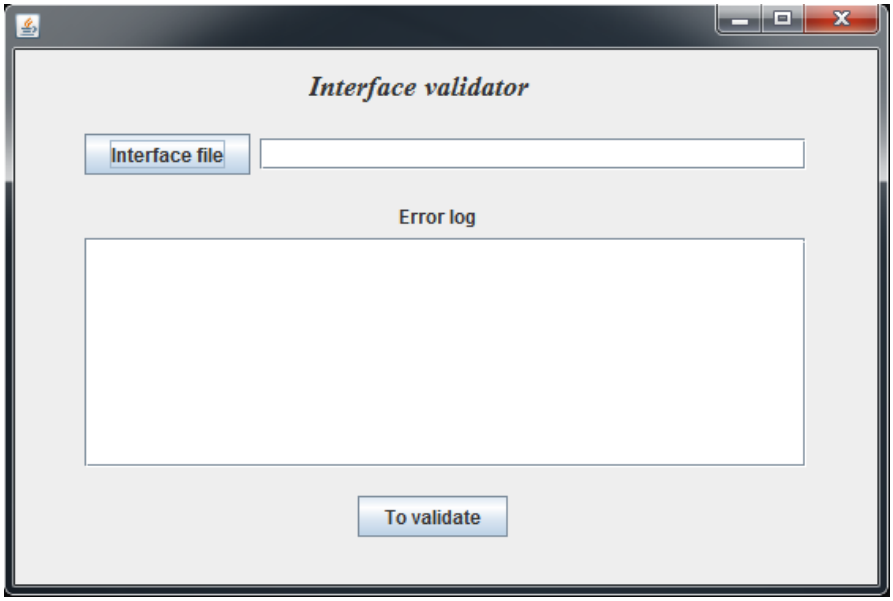


Figura 5. Tela de validação de *interfaces.*
Fonte:próprio autor.

## 6.5 Desenvolvendo uma interface de software

Para que o software possa ser validado, propõe-se a criação de uma interface de programa e posterior teste de integração no framework Galaxy. O programa em questão é o Blast (Basic Local Aligment Search Tool) provido pelo National Center for Biotechnology Information, Estados Unidos, popular programa para alinhamento local de sequências. Mais informações sobre o Blast podem ser encontradas na página do projeto

(http://blast.ncbi.nlm.nih.gov/Blast.cgi).

### 6.5.1 *Analisando programa a ter interface desenvolvida*

Executável: blastall

Parâmetros principais:

*-p*: programa a ser utilizado. Opções: blastn, blastp, blastx, tblastn, tblastx

*-d*: banco de dados onde será realizada pesquisa (arquivo)

*-i*: *query* a qual será pesquisado no banco de dados (arquivo)

*-e*: probabilidade do alinhamento ter sido ao acaso (logaritmo)

*-o*: Nome do arquivo de saída

### 6.5.2 *Desenvolvendo interface*

#### 6.5.2.1 *Escolhendo a opção definir nova ferramenta*

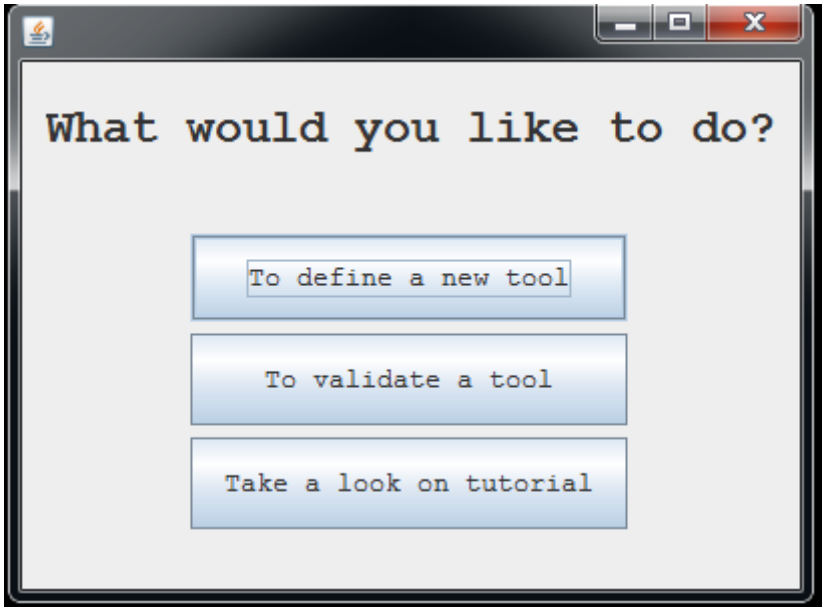


Figura 6 – Menu.
Fonte: próprio autor.

### 6.5.2.2 *Definindo informações sobre a ferramenta*

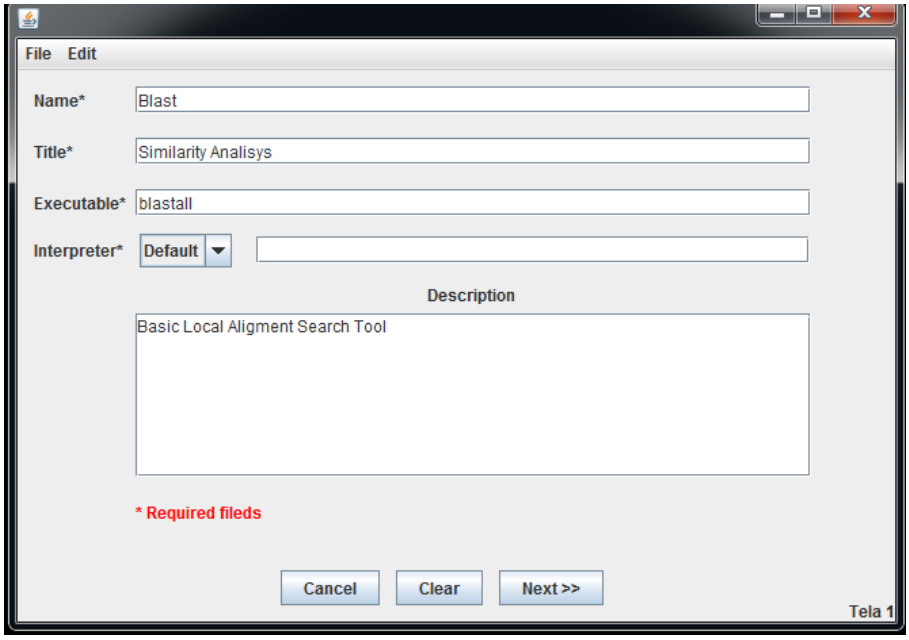


Figura 7- Inserindo informações sobre ferramenta.
Fonte: próprio autor.

### 6.5.2.3 *Definindo parâmetros de entrada "input"*

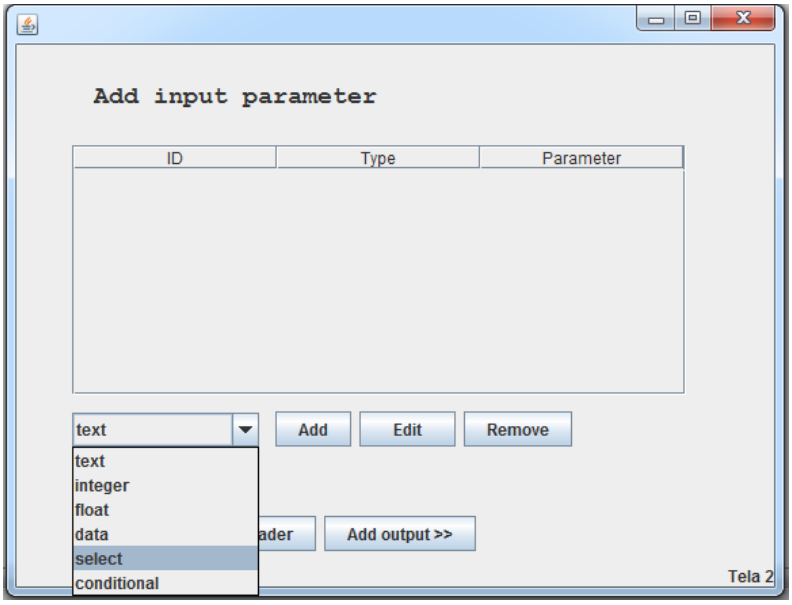


Figura 8 - adicionando parâmetros de entrada.
Fonte: próprio autor.

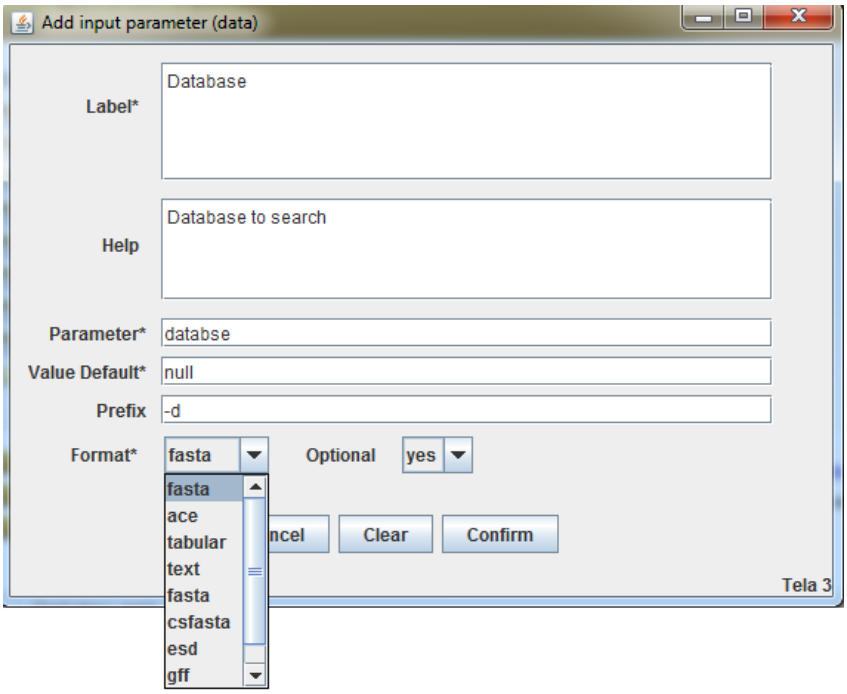


Figura 9 - Adicionando parâmetro de entrada do tipo data.
Fonte: próprio autor.

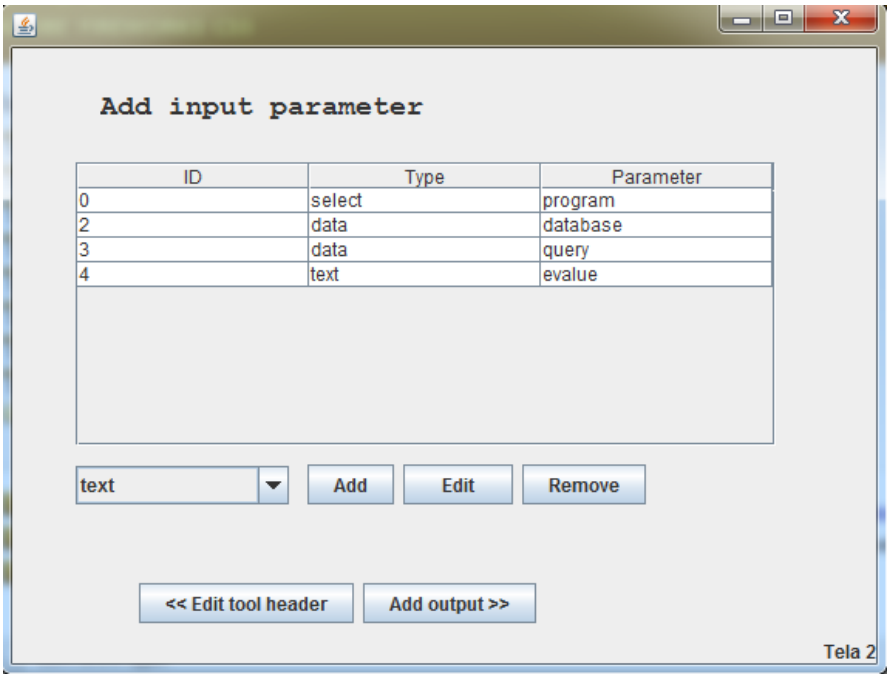


Figura 10 - Tela de parâmetros de entrada já definidos.
Fonte: próprio autor.

*6.5.2.4 Definindo parâmetros de saída "output"*

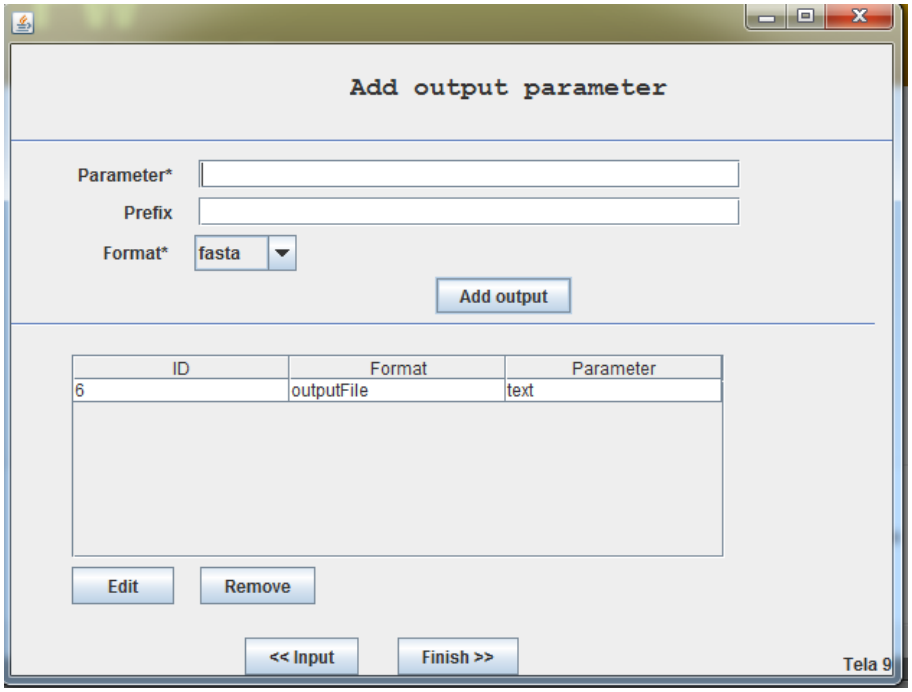


Figura 11 - Tela de definição de parâmetros de saída.
Fonte: próprio autor.

6.5.2.5 *Definindo pasta de destino onde a interface vai ser salva*

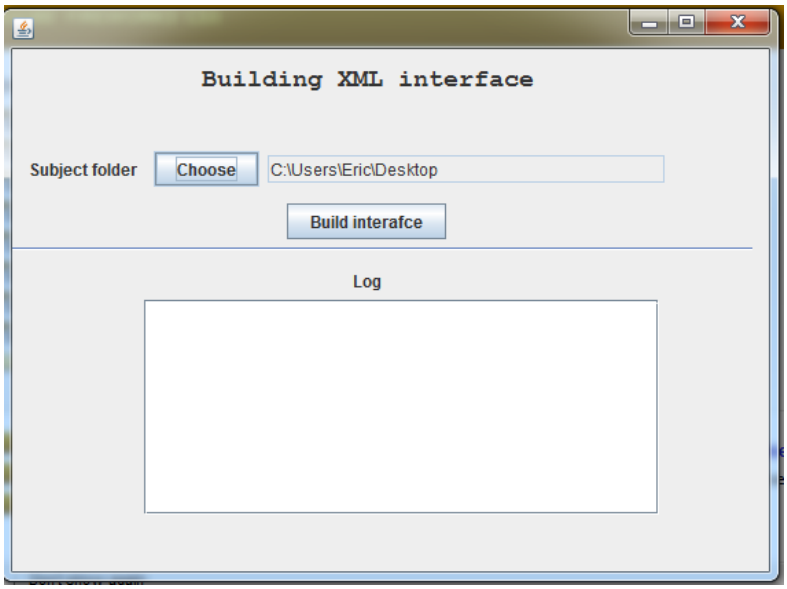


Figura 12 - Definindo pasta onde a *interface* será salva.
Fonte: próprio autor.

### 6.5.2.6 *Gerando arquivo XML*

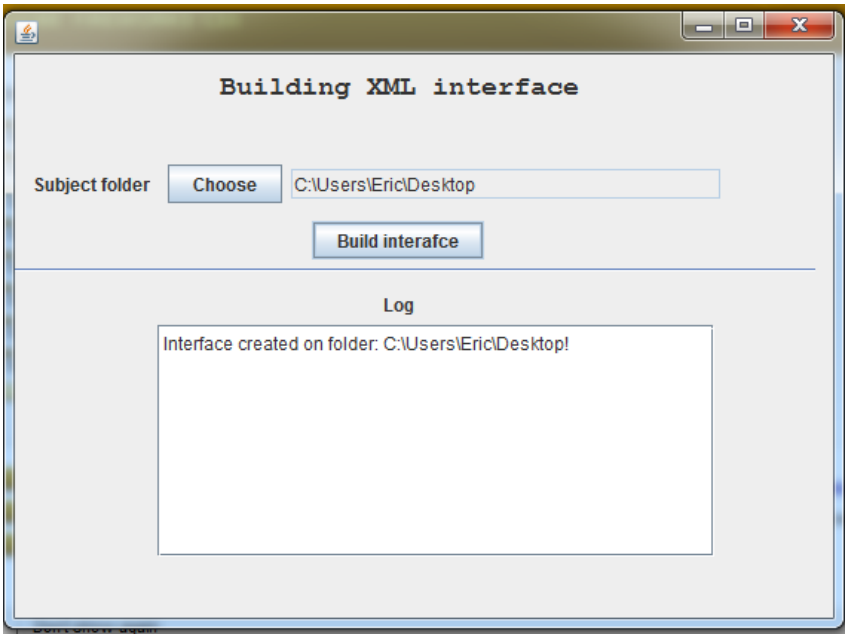


Figura 13 - Gerando *interface*.
Fonte: próprio autor.

### 6.5.2.7 *Abrindo arquivo de XML gerado*

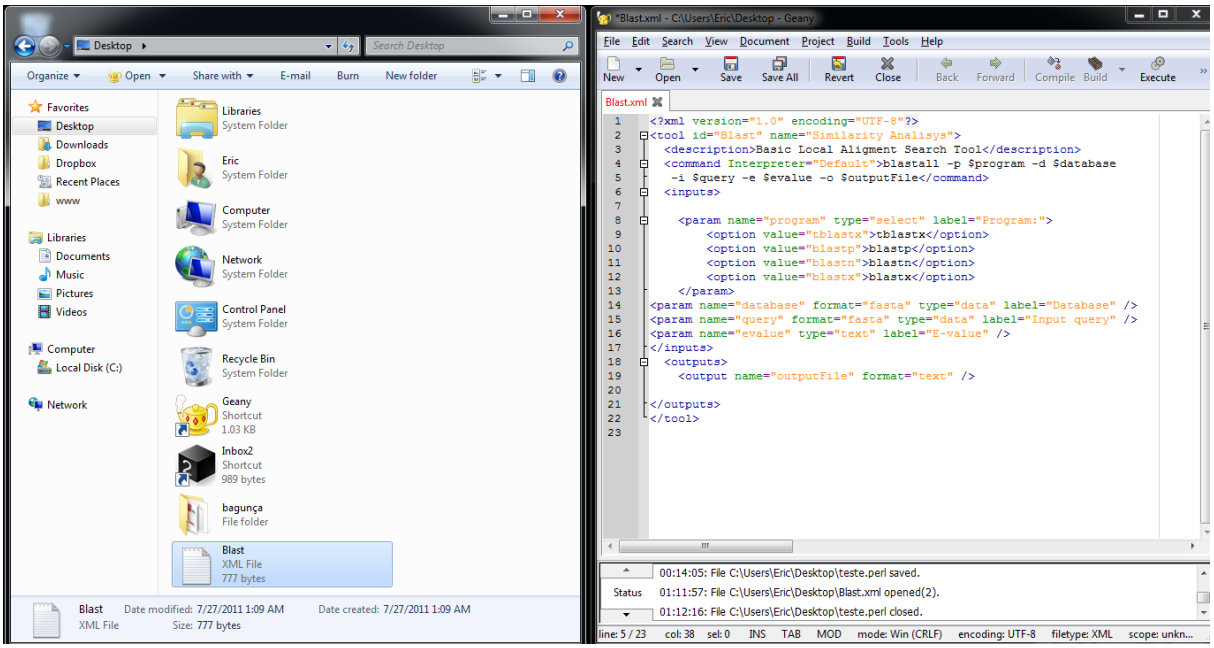


Figura 14 - Arquivo XML gerado.
Fonte: próprio autor.

## 6.6 Integrando *interface* gerada ao Galaxy

### 6.6.1 *Copiando para o servidor*

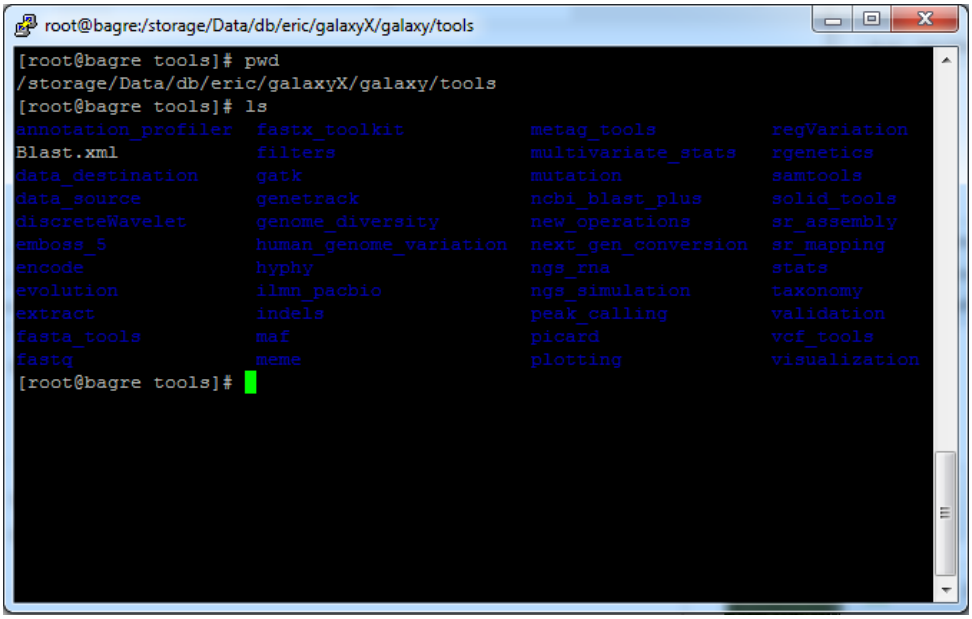


Figura 15 – Interface Blast.xml dentro da pasta tool.
Fonte: próprio autor.

### 6.6.*2 Inserindo entrada para a interface*

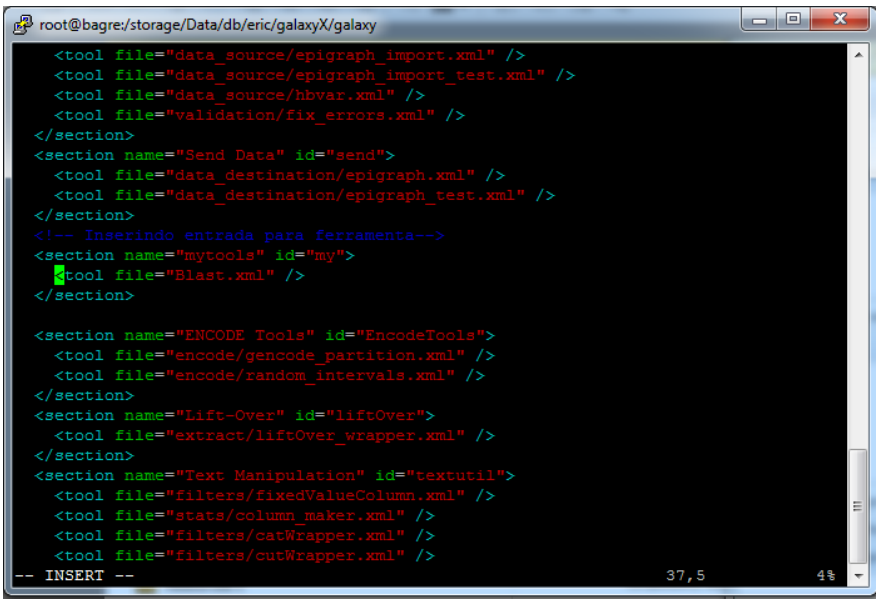


Figura 16 - Inserindo entrada para nova *interface.*
Fonte: próprio autor.

6.6.3 *Abrindo o framework Galaxy*

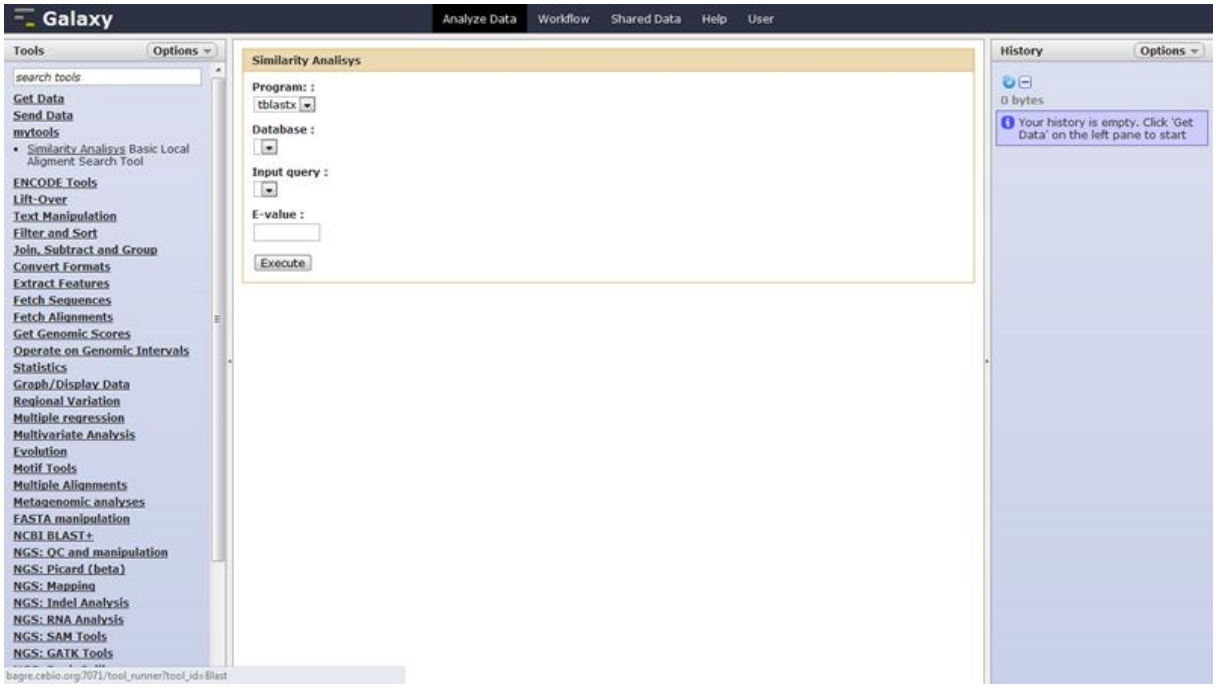

Figura 17 - Visualizando *interface* inserida.
Fonte: próprio autor.

## 6.7 Resultados e Discussão

Após o desenvolvimento do projeto, o principal e almejado produto resultante foi o software gerador de *interfaces* GalaxyX. O GalaxyX contém os elementos básicos principais para criar *interfaces* de programas para serem integrados ao *framework* Galaxy.

O GalaxyX pode ser considerada uma alternativa interessante para a definição de *interfaces* para o *framework* Galaxy devido a características como interface simplificada, prevenção de erros, validação de *interfaces* já definidas e possibilidade de criação de novas *interfaces*. Com estas características, o desenvolvimento de *interfaces* se torna menos complexa e com maior chance de sucesso, utilizando um conjunto de restrições no desenvolvimento que visão garantir que a *interface* seja compatível com os padrões do Galaxy.

Para que o software fique ainda mais amigável e útil aos usuários, normalmente sem um *background* computacional, outras características se fazem necessárias, como

tutoriais mais completos de utilização, possibilidade de definição de outros tipos de dados, mesmo que menos comuns, e principalmente disponibilização de um framework genérico para substituição dos parsers quando estes se fizerem necessários.

Enfim, o GalaxyX é uma mais uma ferramenta para auxiliar os pesquisadores na utilização de softwares de bioinformática. Entretanto são necessários muito mais esforços para que novas ferramentas continuem aparecendo com uma qualidade maior, tornando menos difícil a utilização de software de bioinformática.

# 7. CONCLUSÃO E TRABALHOS FUTUROS

## 7.1 Conclusão

Na escolha do Galaxy como framework alvo do desenvolvimento das *interfaces* é nitidamente perceptível à importância de uma bolsa documentação. Todos os aspectos relacionados ao desenvolvimento de *interfaces* puderam ser encontrados e os detalhes técnicos do software puderam ser desenhados sem nenhuma dúvida a respeito de compatibilidade.

Com o desenvolvimento do GalaxyX, percebeu-se uma necessidade ainda muito grande de programas de bioinformática e de pessoas interessadas a ingressar nessa área. Ainda existem poucas pessoas dispostas a gastar alguma energia para entender os problemas biológicos e desenvolver ferramentas computacionais que auxiliem nesse processo.

Durante a programação propriamente dita do GalaxyX, nota-se a importância de ferramentas para auxiliar no desenvolvimento do software, como o *framework* JDOM, para manipulação de arquivos XML, e a API Log4J, que é utilizada para gerar logs, muito útil ao desenvolvedor para "debugar" o programa a procura de erros.

Finalmente após a conclusão do GalaxyX foi possível avaliar, através de testes simples de utilização, que a ferramenta resultante poderá ser de grande ajuda à sociedade acadêmica, trazendo a simplicidade como uma de suas características e um desenho voltado para a prevenção de erros. Assim, o GalaxyX se mostra uma ferramenta útil à diminuição de complexidade no processo de desenvolvimento de novas *interfaces* de softwares para integração no *framework* Galaxy.

## 7.2 Trabalhos Futuros

Acredito que ainda exista muitos aspectos a se melhorar no GalaxyX, desde questões de interface, como estudo de usabilidade, até métricas de eficiência. Porém, acredito os pontos primordiais ao software seriam:

- Desenvolver uma bateria de testes para validação estatística das vantagens da utilização do GalaxyX no desenvolvimento de *interfaces* de *softwares* para o framework Galaxy.
- Portar para a web o programa para que atinja um número maior de usuários.
- Desenvolver um módulo interno para o Galaxy, para que isso possa ser feito diretamente no framework.
- Desenvolver módulo/página web para compartilhamento de *interfaces*.
- Desenvolver um framework para comunicação entre as *interfaces* e os programas, eliminando a necessidade de programação de *wrapers*.

**REFERÊNCIAS**


BARILLOT, E.; ACHARD, F. *XML: a lingua franca for science?* Trends in Biotechnology, v. 18, n. 8, p. 331-333, Aug. 2000. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0167779900014657>. Acesso em: 22/6/2011.

BLANKENBERG, D.; VON KUSTER, G.; CORAOR, N.; et al. *Galaxy: a web-based genome analysis tool for experimentalists*. Current protocols in molecular biology / edited by Frederick M. Ausubel ... [et al.], v. Chapter 19, p. Unit 19.10.1-21, Jan. 2010. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20069535>. Acesso em: 19/7/2010.

BLAST, E. Human genome 10th anniversary. *Will computers crash genomics?* Science, New York, v. 331, n. 6018, p. 666-8, Feb. 2011.

BRYANT, S. P.; SOLANO, E.; CANTOR, S.; COOLEY, P. C.; WAGENER, D. K. *Sharing Research Models: Using Software Engineering Practices for Facilitation*. Methods report (RTI Press), p. 1-16, Mar. 2011. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/21687780>. Acesso em: 22/6/2011.

Galaxy Project. *Galaxy Adding Tools to Galaxy*. Disponível em <http://bitbucket.org/galaxy/galaxy-central/wiki/AddToolTutorial> . Acesso em: 10 maio 2011.

*Galaxy wiki.* GalaxyProject Documentation. *Disponível em:*
< https://bitbucket.org/galaxy/galaxy-central/wiki/ToolConfigSyntax>. Acesso em: 20/11/2010.

GIARDINE, B.; RIEMER, C.; HARDISON, R. C.; et al. *Galaxy: a platform for interactive large-scale genome analysis*. Genome research, v. 15, n. 10, p. 1451-5, Oct. 2005. Disponível em:
<http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1240089&tool=pmcentrez&rendertype=abstract>. Acesso em: 24/2/2011.

GOECKS, J.; NEKRUTENKO, A.; TAYLOR, J. *Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences*. Genome biology, v. 11, n. 8, p. R86, Jan. 2010. Disponível em: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2945788&tool=pmcentrez&rendertype=abstract>. Acesso em: 22/5/2011.

HOGEWEG, P; HESPER, B. *Simulating the growth of cellular forms*. SIMULATION, v. 31, n. 3, p. 90-96, Sep. 1978. Disponível em: <http://sim.sagepub.com/cgi/doi/10.1177/003754977803100305>. Acesso em: 10/3/2011.

HOULDING, S. *XML — an opportunity for <meaningful> data standards in the geosciences*. Computers & Geosciences, v. 27, n. 7, p. 839-849, Aug. 2001. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S009830040000145X>. Acesso em: 15/4/2011.

HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; et al. *Taverna: a tool for building and running workflows of services*. Nucleic acids research, v. 34, n. Web Server issue, p. W729-32, Jul. 2006. Disponível em: <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1538887&tool=pmcentrez&rendertype=abstract>. Acesso em: 10/11/2010.

NÉRON, B.; MÉNAGER, H.; MAUFRAIS, C.; et al. *Mobyle: a new full web bioinformatics framework*. Bioinformatics (Oxford, England), v. 25, n. 22, p. 3005-11, Nov. 2009. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/19689959. Acesso em: 14/11/2010.

SCHATZ, M. C. *The missing graphical user interface for genomics*. Genome biology, v. 11, n. 8, p. 128, Aug. 2010. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20804568>. Acesso em: 9/9/2010.

SCHNEIDER, M. V.; WATSON, J.; ATTWOOD, T.; et al. *Bioinformatics training: a review of challenges, actions and support requirements*. Briefings in bioinformatics, v. 11, n. 6, p. 544-51, Nov. 2010. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/20562256>. Acesso em: 30/3/2011.

STAJICH, J. E.; BLOCK, D.; BOULEZ, K.; et al. *The Bioperl toolkit: Perl modules for the life sciences*. Genome research, v. 12, n. 10, p. 1611-8, 2002. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/12368254>. Acesso em: 15/01/2011.

TAYLOR, J.; SCHENCK, I.; BLANKENBERG, D.; NEKRUTENKO, A. *Using galaxy to perform large-scale interactive data analyses*. Current protocols in bioinformatics / editorial board, Andreas D. Baxevanis ... [et al.], v. Chapter 10, p. Unit 10.5, Sep. 2007. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/18428782>. Acesso em: 22/6/2011.

*REICH*, M.; LIEFELD, T.; GOULD, J.; et al. *GenePattern 2.0*. Nature genetics, v. 38, n. 5, p. 500-1, May 2006. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/16642009>. Acesso em: 22/6/2011.