GERÊNCIA DE DADOS GENÔMICOS EM *WORKFLOWS* DE BIOINFORMÁTICA

Amanda de Mattos Sant'Ana Batista

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profª. Marta Lima de Queirós Mattoso, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Profª. Jonice de Oliveira Sampaio, D.Sc.

Prof. Alberto Martín Rivera Dávila, D.Sc.

RIO DE JANEIRO, RJ – BRASIL

AGOSTO DE 2008

BATISTA, AMANDA DE MATTOS SANT'ANA

Gerência de Dados Genômicos em Workflows de Bioinformática [Rio de Janeiro] 2008

XII, 187 p., 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 2008)

Dissertação – Universidade Federal do Rio de Janeiro, COPPE

1.*Workflows* Científicos

2.Bioinformática

3.Proveniência de Dados

I. COPPE/UFRJ II. Título (série)

À minha avó Joaquina (*in memoriam)* e aos meus pais,

exemplos sempre.

Agradecimentos

"Posso ter defeitos, viver ansioso e ficar irritado algumas vezes, mas não esqueço de que minha vida é a maior empresa do mundo. E que posso evitar que ela vá à falência. Ser feliz é reconhecer que vale a pena viver, apesar de todos os desafios, incompreensões e períodos de crise. Ser feliz é deixar de ser vítima dos problemas e se tornar um autor da própria história. É atravessar desertos fora de si, mas ser capaz de encontrar um oásis no recôndito da sua alma. É agradecer a Deus a cada manhã pelo milagre da vida. Ser feliz é não ter medo dos próprios sentimentos. É saber falar de si mesmo. É ter coragem para ouvir um não. É ter segurança para receber uma crítica, mesmo que injusta. Pedras no caminho? Guardo todas, um dia vou construir um castelo..."

(Fernando Pessoa)

Enfim, eis um desafio vencido. Mais uma pedra que é posta em seu lugar no castelo da minha vida. E o que seria desta vida sem todos os queridos que não somente a preenchem de alegria, mas que também ajudaram a construí-la...

Inicio este momento agradecendo a Deus, por ter me planejado, ter me ajudado em cada passo necessário em todas as minhas empreitadas. Por me dar oportunidades e me capacitar para abraçá-las. Por me amar tanto assim.

Aos meus avós, Joaquina e Ednor, que me ensinaram tanto, que me trazem sempre doces lembranças e que são referências inigualáveis de perseverança, de pessoas que vão muito além dos limites que lhes são impostos.

Não existem palavras para expressar a gratidão que tenho para com meu pai, que nunca deixou de ser meu herói, e minha mãe, meu principal apoio sempre. Eles abriram mão de muito por mim. Sofreram comigo e se alegraram comigo. Acompanharam-me sempre de muito perto e me atribuem uma importância que ninguém jamais me daria. Pai e mãe, este título não foi alcançado apenas por mim. As vidas de vocês também foram dedicadas a isto, mesmo antes de eu imaginar quais seriam os meus caminhos. Por isso, este título também é de vocês.

Agradeço às minhas irmãs, Juliana e Aline. Pessoas sem as quais a minha história não seria completa, não teria tanta graça. Elas me transmitiram força. Ouviram

meus desabafos. Me divertiram. Agradeço a elas por todas as lembranças que tenho dos momentos que passamos juntas. As travessuras da infância, as aventuras e indagações da adolescência, os dilemas da juventude, as conversas nas madrugadas. Por todas as coisas tão sem importância e ao mesmo tempo tão importantes. Tesouros refletidos em quem sou.

Anderson é outra pessoa sem a qual não teria sido possível lidar com tudo que assumi para minha vida. Meu amado, que me completa e me conquista todos os dias. Agradeço por ter me compreendido, me ajudado. Por cuidar de mim sempre com muito carinho. Agradeço porque escolheu me incluir em todos os seus planos e assumir os meus sonhos como seus também.

À minha orientadora Marta Mattoso agradeço, pois aceitou me conduzir no mundo da pesquisa, tão novo para mim. Porque acreditou em minha capacidade, me motivou e me orientou com paciência. Sempre demonstrando zelo, conhecimento e sabedoria indispensáveis para a conclusão deste trabalho.

Agradeço à COPPE, instituição de excelência, que sempre proveu os recursos para que esta pesquisa fosse bem sucedida. Em especial, aos professores da Linha de Banco de Dados, Geraldo Xexéo, Geraldo Zimbrão e Jano, porque a excelência de seu conhecimento e disponibilidade em sempre ensinar são os fatores que fazem refletir a qualidade do Programa de Engenharia de Sistemas e Computação da COPPE.

Em especial agradeço ao prof. Geraldo Xexéo, por aceitar participar da minha banca de avaliação. De igual modo, agradeço à professora Jonice, por se dispor a me ajudar sempre que precisei e também me dar a honra de obter suas críticas e observações sobre este trabalho.

Outro membro da minha banca avaliadora, que também merece meus sinceros agradecimentos é o prof. Alberto M. R. Dávila, da Fiocruz. Sua gentileza e paciência em me apresentar o "mundo genômico" são as características marcantes de alguém que nasceu para o ensino. Agradeço porque aceitou fazer parte desta banca e porque providenciou tudo quanto precisei para concluir meus experimentos.

À toda equipe do ProtozoaDB, que também sempre me atendeu com zelo e demonstrou paciência em todas as entrevistas que fiz, agradeço. Em especial ao Diogo e à Milene.

Por toda a ajuda que gentilmente me ofereceu, também agradeço ao Serra e ao Fabrício. Ao Nicolaas e ao Leo Arnt, agradeço por suas boas idéias que agregaram grande valor a este trabalho. Às professoras Maria Luiza e Yoko também agradeço,

assim como, à Linair. Ao prof. Marinho, do colégio Equipe Grau, que me estimulou aos estudos e que é uma inspiração como professor.

À minha família, Titia, Cleneide, tia Maria e tio Geraldo, agradeço porque oraram por mim e desejaram o melhor para a minha vida, me apoiando sempre. Em especial, agradeço à minha prima Cleide que me cativa, sempre demonstrando a importância que tenho em sua vida. Ela é de igual modo importante para mim.

Ao meu sogro Abel e à minha sogra D'Alva, por me amarem e me aceitarem como uma filha, pelos seus tão sábios ensinamentos, pelas orações e carinho que sempre dedicaram a mim. Também sou grata à minha cunhada Giselinha, outro grande exemplo de determinação e superação, por sempre orar e torcer por mim.

Não esqueceria jamais de agradecer a um grupo que é também minha família. Com nossas reuniões de todas as segundas-feiras, me tirando de um mundo onde tudo é muito sério, para um mundo bem mais tranqüilo, bem menos pesado, onde as músicas cantadas, as conversas compartilhadas, as gargalhadas, os segredos, os dramas, as meditações, são vividos com amor fraternal. Pessoas queridas, que fazem parte de mim, do que tenho para contar... Lu, Jonison, Fernanda Paula, Pablo, Clarisse, Natália, Rafael, Celso, Carol, Willy, Fernanda, Claudinho, Rose, Ronnie, Giselle, Gleice, Claudilene... meus "bests", que também me passaram muita força! Também agradeço aos pequenos, Gabriel, Alice e Eduarda, por toda a alegria que trazem para mim.

Agradeço aos meus amigos de sempre, EDK5, com nossos nove anos de história e histórias. Daniel Leitão, Strudel, Caio, Caputo, Higo (Raigo para mim...), Dudu e meu irmãozinho Daniel. Esse grupo tinha que se encontrar. Nós fomos significativamente influenciados uns pelos outros. A este grupo que torce por mim e se mantém presente, mesmo apesar da distância.

De modo único agradeço ao meu amigo Daniel Cardoso, por estar sempre ao meu lado, por me motivar mesmo quando eu nem mereço. Por ser alguém que me suporta mesmo quando estou insuportável.

Agradeço a todos que, de algum modo, trabalharam comigo ou tiveram presença marcante em minha vida, mesmo sem terem percebido. Gustavo Pinto, Márcio Duran, Paula Ballard, Rafael Brand, Rogea, Ramon, Joice, Márcia, Nice, Rodrigo Salvador, Leo Guerreiro, Ricardo, Ariadne, Milene, Moisés, Vinicios Pereira, Amanda Varella, Pedro Calisto, Vinicius Von Held, Vinicius Pereira, Felipe Leite, Robson, Vanessa Carla, Matheus, Talittão!, Camille, Vinagre, Carlinha, Martino, Nathália Marassi,

Claudinho, Rodrigo Porto, Ana Garcia, Vinícius Marques, Carol Barreiros e Patrícia Leal.

Ao prof. Blaschek, um motivador nato. Alguém com quem a cada minuto é possível aprender algo para a vida inteira. Agradeço por acreditar em mim e, também, me oferecer tantas oportunidades.

Agradeço ao CNPq pelo auxílio oferecido através de uma bolsa de estudos. Também agradeço à Fundação COPPETEC pelas oportunidades nos inúmeros projetos dos quais participei.

Meu muito obrigado, a todos que de maneira direta ou indireta colaboraram para este trabalho e que não foram citados aqui. Obrigada a todos que tiveram participação especial em minha vida e que me ajudaram, mesmo sem perceber, a enfrentar desafios, a vencer meus medos, a me superar, a chegar neste momento.

"Vem me pedir além do que eu posso dar
É aí que o aprendizado está
Vem de onde não sonhei
me presentear
Quando chega o fim da linha
e já não há aonde ir
Num passe de mágica
a vida nos traz sonhos pra seguir
Queima meus navios
pr'eu me superar
às vezes pedindo que ela vem nos dar
o melhor de si
E quando vejo,
a vida espera mais de mim
mais além, mais de mim
O eterno aprendizado é o próprio fim
Já nem sei se tem fim..."
(Jorge Vercilo)

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

GERÊNCIA DE DADOS GENÔMICOS EM *WORKFLOWS* DE BIOINFORMÁTICA

Amanda de Mattos Sant'Ana Batista

Agosto / 2008

Orientadora: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

A gerência dos dados envolvidos nos experimentos científicos é uma tarefa árdua para os Sistemas de Gerência de *Workflows* Científicos (SGWfCs), pois cada domínio de aplicação científica, como bioinformática, por exemplo, possui esquemas, metadados ou ontologias, mais adequados ao seu contexto. Os esquemas de dados propostos pelos SGWfCs são extremamente simples e não substituem os esquemas de dados dos domínios de aplicação, que por sua vez, não são facilmente incorporados aos SGWfCs. Além da heterogeidade de esquemas há também o problema da captura de dados semânticos ao longo do ciclo de vida do experimento. Esta dissertação propõe uma arquitetura para a gerência de dados genômicos em *workflows* de bioinformática, na qual são considerados o aproveitamento das soluções para definição, execução e proveniência de dados de *workflows* pelos principais SGWfCs e, como repositório para estes dados, o esquema padronizado para o domínio da bioinformática GUS (*Genomic Unified Schema*). São propostos serviços que, acoplados aos SGWfCs, possibilitam a definição do que deve ser armazenado e em que etapas do experimento esta tarefa deve ocorrer. É efetuado o mapeamento e armazenamento dos dados de proveniência gerados ao longo do ciclo de vida dos experimentos da bioinformática, permitindo análises mais sofisticadas sobre as informações geradas. O uso de um *workflow* real evidenciou as vantagens da solução apresentada.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

GENOMIC DATA MANAGEMENT IN BIOINFORMATICS WORKFLOWS

Amanda de Mattos Sant'Ana Batista

August / 2008

Advisor: Marta Lima de Queirós Mattoso

Department: Computer and Systems Engineering

The management of scientific data involved with *in-silico* experiments is a difficult task for Scientific Workflows Management Systems because each scientific domain, like bioinformatics or geology, has schemas, metadata and ontologies adjusted for its specific context. Scientific Workflows Management Systems (SWfMSs) commonly proposes simple data schemas that do not substitute domain specific data schemas and it is not easy to couple these schemas to SWfMSs.

In the molecular biology scientific scenario, an architecture is proposed to manage genomic data produced by bioinformatics workflows. This strategy considers definition, execution and data provenance solutions proposed by the main SWfMSs and the GUS schema (Genomic Unified Schema) as default bioinformatics domain schema, where these provenance data will be storage.

This architecture suggests services that can be connected to one of these SWfMSs. With these services it will be possible to specify which data to store in the provenance domain repository and, moreover, in which step of the workflow will this capture occurs.The solution provides the automatic workflow produced data mapping to the chosen schema, supporting data provenance and analysis.

Índice:

Capítulo 1 – Introdução

"Não sabendo que era impossível, foi lá e fez."
(Jean Cocteau)

1.1 – Motivação

Aproximadamente na década de 1930 a biologia molecular teve seu início com a convergência de algumas disciplinas da biologia: bioquímica, genética, microbiologia e virologia [1]. Seu foco de estudo se concentra nas macromoléculas e mecanismos macromoleculares presentes nos organismos como replicação, mutação e expressão de genes [1]. Ácidos nucléicos e proteínas são as duas categorias de macromoléculas mais pesquisadas na biologia molecular que busca caracterizar estruturas, funções e relacionamentos entre estes dois tipos de macromoléculas [1].

Geneticistas, físicos e químicos unem esforços em torno de um problema em comum: estrutura e função do gene. A maneira como as características biológicas são transmitidas de geração para geração é estudada pela Genética, termo que foi aplicado pela primeira vez em 1905 por William Bateson para descrever o estudo da variação e hereditariedade [2] em uma carta a Alan Sedgwick. Em 1906, Bateson popularizou o termo na *Third International Conference on Plant Hybridization* [3]. Vários estudos na área da genética foram conduzidos até que, em 1953, James Watson e Francis Crick descobriram a estrutura do DNA (*Deoxyribonucleic Acid*) [4], em português, ácido desoxirribonucleico, a molécula que contém todas as informações genéticas de cada ser vivo.

Desde então, o número de pesquisas genéticas aumentou consideravelmente e seus resultados se tornaram bastante expressivos tanto no contexto científico quanto na sociedade. Estes estudos colaboraram para a prevenção e tratamento de doenças genéticas, terapias genéticas que utilizam a substituição de genes doentes por genes sãos ou eliminação de genes, terapias definidas de acordo com o perfil genético, testes de

paternidade, alimentos transgênicos, assim como tantos outros assuntos relacionados à genética que têm se tornado populares e, em alguns casos, polêmicos.

Atualmente cientistas das mais diversas áreas encontram apoio para suas pesquisas em um vasto conjunto de ferramentas computacionais para acelerar seu processo de análise e descoberta. Essas tecnologias permitem que grupos de pesquisadores colaborem com dados, idéias e experimentos em tempo real, mesmo quando estão geograficamente dispersos. Para englobar as estratégias computacionais de apoio às ciências da vida, surgiu a Bioinformática [5] uma área de conhecimento multidisciplinar que se refere à criação e evolução de algoritmos, técnicas estatísticas e computacionais e teorias para resolver problemas práticos e formais envolvidos nos processos de análise e gerenciamento de informações biológicas.

Na bioinformática, áreas de pesquisa como matemática aplicada, informática, estatística, ciência da computação, inteligência artificial, química e bioquímica produzem estudos que convergem para solução de problemas biológicos. Seu principal objetivo é a automatização dos procedimentos de coleta, organização, armazenamento, recuperação e análise de dados biológicos gerados pelas ferramentas computacionais envolvidas no experimento científico, propiciando a inferência ou descoberta de informações.

São comuns na bioinformática, procedimentos chamados de experimentos *in silico*, ou seja, aqueles que são baseados em repositórios de informações científicas e que fazem uso intenso de ferramentas computacionais que podem ser encadeadas de modo a permitir a análise desses dados com o objetivo de testar uma hipótese, gerar conclusões, buscar padrões ou demonstrar um fato conhecido [6]. Cromatogramas, análises de alinhamentos, comparações de seqüências e análises filogenéticas podem ser citados como exemplos de experimentos *in silico*.

Apesar de todos os benefícios obtidos com o uso dessas tecnologias, como a otimização na execução de procedimentos e aumento da qualidade dos resultados, este cenário leva ao crescente aumento dos dados produzidos pelas pesquisas, assim como, grande heterogeneidade e falta de integração entre as diversas aplicações e bancos de dados existentes nos diferentes centros de pesquisa. Com isso surgem novas

dificuldades para as atividades de compreensão, integração e manipulação das informações científicas por parte dos pesquisadores.

Como exemplo deste aumento de dados, é possível citar os bancos de dados de biologia molecular, os quais multiplicaram-se ao longo dos anos e, atualmente, totalizam mais de 1000 bases de dados para os diversos organismos existentes [7]. As informações possuem origens e versões distintas assim como diferentes graus de qualidade e confiabilidade. Os dados são armazenados em locais e formatos variados, possuindo estruturas heterogêneas e muitos métodos de acesso. A mesma diversidade pode ser observada no conjunto de aplicações voltadas para experimentos científicos. Todos estes fatores caracterizam um quadro de grande elevação do nível de dificuldade das atividades de pesquisa dos biólogos. Na figura 1.1 é possível visualizar a taxa de crescimento dos dados armazenados no Genbank [23], um dos maiores repositórios de dados genômicos da comunidade científica.

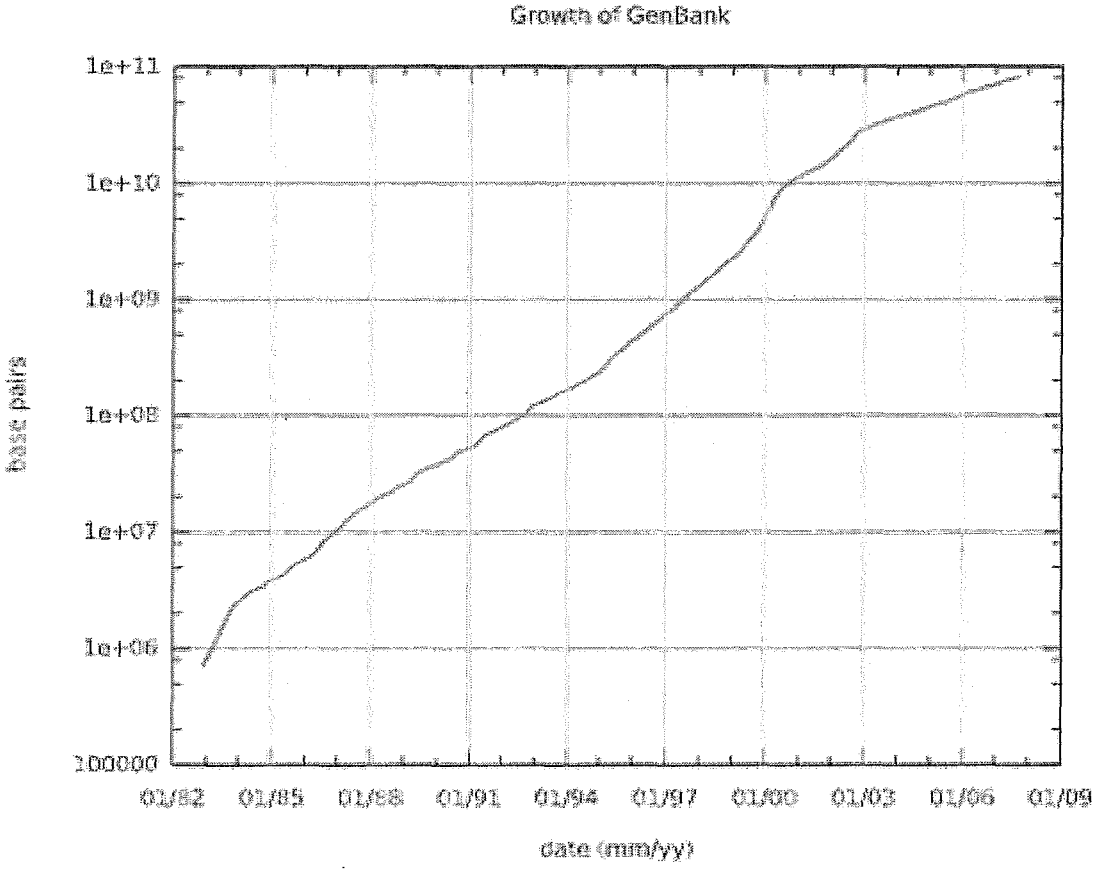


Figura 1.1. Crescimento exponencial de dados no Genbank [23].

Nos experimentos *in silico*, os dados de saída de cada etapa do processo podem ser utilizados como dados de entrada para as próximas etapas de maneira encadeada. As aplicações podem formar uma composição como se fosse apenas um sistema. Nesse contexto surge a necessidade de orquestrar as soluções computacionais utilizadas para que a integração dos resultados gerados ao longo do experimento não seja uma tarefa manual executada pelo cientista. Além disso, esta integração é necessária para que os dados produzidos por cada etapa do experimento possam ser eficientemente gerenciados de modo a prover análises dessas informações.

Um mecanismo computacional que permite a representação do encadeamento entre aplicações de maneira automatizada é o *Workflow*, que pode ser definido como uma série de tarefas que podem ser executadas em uma seqüência pré-definida através de regras de fluxo. O início da execução de um *workflow* se dá a partir de um conjunto de dados que serão necessariamente consumidos ao longo do processo. Dados intermediários podem ser gerados durante a execução de cada passo do *workflow* e repassados às tarefas que se seguem na seqüência. Ao final da execução, espera-se que seja gerado um resultado coerente com o propósito ao qual o *workflow* foi definido.

Utilizando *workflows* é possível descrever de maneira abstrata um conjunto de atividades e processos estruturados para fluxos que encadeiam as diversas análises de dados envolvidas em experimentos científicos de modo a prover ambientes eficientes para a resolução de problemas, utilizando recursos computacionais. Assim, *workflows* podem ser utilizados tanto para modelar experimentos *in silico* quanto para executá-los e monitorá-los. No meio científico, esta técnica já vem sendo adotada, de modo que os *workflows* são modelados através da composição de aplicações em uma seqüência de execução.

Para um *workflow* que representa um experimento científico foi aplicado o termo *workflow* científico, em oposição aos *workflows* comerciais utilizados nos processos de negócios das empresas. Os *workflows* científicos geralmente agregam maior dificuldade devido à necessidade de processamento de alta capacidade, enorme volume de dados transferidos, maior número de execuções a cada simulação do processo e à necessidade de freqüentes customizações.

Devido à falta de conhecimento computacional e à enorme necessidade de integração, atualmente a maioria das composições de ferramentas computacionais é realizada manualmente pelos cientistas. A análise dos dados gerados é pontual, ocorrendo no momento em que o biólogo gera o dado, quando muitas vezes, decide qual será o próximo passo do experimento ou necessita manipular o dado para deixá-lo compatível com o próximo programa a ser utilizado no fluxo. Uma maneira de tentar automatizar esse encadeamento de aplicações é a utilização de linguagens de programação de *script* como Perl [11]. Essa técnica é extremamente acoplada ao ambiente e muito complexa para os leigos em computação, o que dificulta as atividades do cientista. Além disso, informações de registro de execuções dos programas e a limitação no uso das aplicações, tendo em vista que as mesmas devem ser executadas localmente, são fatores que tornam esta estratégia menos favorável às pesquisas.

Vários sistemas de gerência de *workflows* científicos (SGWfCs) surgiram para oferecer esse tipo de suporte aos experimentos científicos, através da disponibilização de interfaces gráficas que permitem a modelagem e a execução de experimentos *in-silico*. Os SGWfCs que se tornaram mais populares para especificação e execução de análises que envolvem um intenso fluxo de dados são o Taverna Workbench [9], o Kepler [8], o Vistrails [10] e o Chimera [24]. Nesses sistemas, os *workflows* podem ser projetados graficamente através dos relacionamentos entre as tarefas, da configuração de parâmetros e identificação de fontes externas de dados. Cada tarefa pode consumir dados produzidos pelo processamento de tarefas anteriores. Os *workflows* podem ser visualizados como grafos direcionados, nos quais os nós representam módulos de uma análise e as arestas, o fluxo de dados entre estes módulos.

Paralelismo e distribuição são características que passam a ser requisitos para os *workflows* científicos. Os sistemas citados acima possuem soluções para a utilização de Serviços Web [18] para atender à necessidade de distribuição. Serviços web fornecem interfaces padrão de acesso que não dependem de linguagem de programação ou plataforma, permitindo a interoperabilidade entre recursos heterogêneos na Web. Os serviços web utilizam linguagem e protocolos baseados em XML (*eXtensible Markup Language*): WSDL (*Web Services Description Language*) [19], para descrever serviços e SOAP (*Simple Object Access Protocol*) [20] como protocolo de comunicação. Os recursos publicados como serviços web atuam como módulos de software reutilizáveis, pois ficam acessíveis por qualquer programa cliente na Web. Atualmente existem

diversos serviços web disponibilizados com o objetivo de apoiar as atividades científicas. Algumas estratégias para *workflows* são específicas para o uso de *workflows* científicos com serviços web [17, 21 e 22].

Outra característica que deve ser observada nos SGWfCs disponíveis é a proveniência de dados. Durante a execução de um experimento científico são gerados metadados que fornecem semântica aos *workflows* e possibilitam sua reutilização. Um dos tipos de metadados é a proveniência dos dados, também chamada de linhagem ou *pedigree* [25]. Estas informações adicionam um valor significativo para projetos científicos com intenso fluxo de dados. Dentre os sistemas mencionados acima, o Taverna, o Kepler e o Vistrails possuem diferentes alternativas para gerenciar dados de proveniência.

Dados de proveniência de um experimento científico são informações que permitem determinar o histórico do dado resultante ao final da análise a partir de sua origem, mesmo que esta informação tenha sofrido diversas transformações durante o processamento e se encontre em um formato qualquer como, arquivos texto, tabelas ou imagens [25]. No contexto de bancos de dados, proveniência de dados pode ser descrita como a origem do dado e do processo através do qual esta informação foi armazenada no banco de dados [26]. Em alguns casos, linhagem é interpretada como materiais e transformações que são utilizadas para produzir um dado [27]. Além de ser associada aos dados gerados, a proveniência pode ser atribuída ao processo que gera estes dados [28], ou seja, metadados que descrevem o processo e identificam sua origem.

Um volume significativo de dados pode ser gerado e processado em cada etapa da execução de um *workflow*. No contexto da bioinformática, para validar um experimento científico não se deve considerar apenas os dados retornados ao final da execução de um *workflow*. Os dados intermediários, gerados a partir da execução de cada etapa do fluxo de trabalho são essenciais para a validação do estudo. Alguns SGWfCs oferecem mecanismos que possibilitam a reutilização dos dados intermediários para a execução de novos experimentos, permitindo re-execuções parciais de *workflows*.

A gerência de dados intermediários de um *workflow* deve facilitar manutenções, acessos e análises dos dados gerados e permitir que dados irrelevantes gerados durante a

execução do experimento sejam eliminados de forma adequada. Os cientistas devem ter flexibilidade para optar pelo armazenamento ou eliminação do dado, pois diferentes pesquisas podem possuir necessidades distintas.

É árdua a tarefa de gerenciamento de dados de proveniência por parte dos SGWfCs, pois cada domínio de aplicação científica, como bioinformática ou geologia, por exemplo, possui esquemas, metadados ou ontologias, mais adequados ao seu contexto. Os esquemas de dados propostos pelos SGWfCs são, de um modo geral, extremamente simples e não substituem os esquemas de dados dos domínios de aplicação que, por sua vez, não são facilmente incorporados aos SGWfCs.

Alguns desses esquemas podem ser utilizados para *Datawarehousing*, contribuindo para análises estatísticas sobre os dados de um experimento. Um exemplo desse tipo de esquema é o GUS (*The Genomics Unified Schema*) [29], um esquema que possui um modelo conceitual padronizado, integrando e estabelecendo relações entre diversos tipos de dados e aplicações da bioinformática. O GUS possui um *framework* associado para armazenar, integrar, analisar e apresentar dados genômicos. Ontologias e tipos de dados como genomas, expressão de genes, proteomas e outros podem ser representados neste esquema. Um ambiente para desenvolvimento de sítios baseados em consultas que acessam a base de dados do GUS é disponibilizado através do GUS WDK (*Web Development Kit*). Projetos de pesquisas biológicas conceituados e de grande porte fazem uso desta tecnologia como o PlasmoDB [30], GeneDB [31], TcruziDB [32], BiowebDB [33] e vários outros.

1.2 – Objetivo

Analisando o cenário apresentado acima é possível perceber que o vasto conjunto de ferramentas computacionais disponíveis para auxiliar pesquisas científicas possibilita que estudiosos colaborem com dados e tarefas em experimentos. As aplicações de software dos experimentos *in silico* podem ser encadeadas formando um *pipeline*. Os SGWfCs como Taverna, Kepler e Vistrails surgiram para dar apoio à construção dessas cadeias de aplicações científicas. Do mesmo modo que os SGWfCs surgiram propondo soluções com ênfase em tarefas, esquemas de representação de dados de bioinformática como o GUS foram definidos almejando compartilhamento e consulta de informações relacionadas aos experimentos.

Dados de proveniência, tanto intermediários quanto metadados relacionados à história dos dados gerados, são extremamente relevantes para garantir a qualidade da pesquisa e são intensamente gerados e trafegados ao longo do processo. Cada SGWfC possui seu próprio modelo de representação de dados de proveniência. Estes modelos comumente são muito simples e não refletem semântica relacionada ao domínio científico. Nenhum dos SGWfCs analisados durante a pesquisa que produziu esta dissertação possui mecanismos de integração com esquemas padronizados para um domínio científico e, além disso, seus repositórios não permitem análises mais elaboradas sobre os dados armazenados durante as execuções. Estes sistemas não disponibilizam interfaces para a construção facilitada de consultas ou apresentam soluções instáveis.

Os esquemas, metadados ou ontologias padronizados, ou comumente adotados, que atendem às necessidades de cada domínio de aplicação científica (bioinformática, geologia, etc.) não são facilmente incorporados aos SGWfCs. Desse modo, cientistas que necessitam armazenar os dados gerados por seus experimentos em bases de dados que acomodem estes dados em um modelo de representação padronizado para a bioinformática não conseguem atingir este objetivo ao modelar seu experimento em SGWfCs.

Quando o cientista tem seu experimento projetado em um SGWfC ele tem a vantagem da execução automatizada dos recursos computacionais envolvidos no seu experimento, mas não terá os dados gerados nesta execução em um banco de dados centralizado e voltado para o compartilhamento de pesquisas científicas do mesmo domínio. Por contrapartida, se necessitar que os dados sejam armazenados na base de dados padrão, o cientista deverá executar as aplicações envolvidas manualmente ou construir *pipelines* dispensando grande parte do seu tempo de pesquisa para resolver problemas computacionais complexos.

Para produzir a solução apresentada neste trabalho mantivemos o foco no domínio científico da bioinformática e, dentre todos os tipos de dados de proveniência, apenas os dados intermediários serão considerados nesta arquitetura. Esta definição de escopo é necessária em virtude da impossibilidade de propor uma solução única para a gerência de dados intermediários para todas as áreas científicas. Além disso, nossa proposta engloba apenas os dados intermediários, pois estes são os mais negligenciados

pelos SGWfCs que não realizam armazenamento destes dados atribuindo-lhes a devida semântica, o que só é possível com a indicação do domínio científico. Os outros tipos de dados de proveniência encontram estratégias de gerência mais adequadas nos SGWfCs.

O objetivo deste trabalho é propor uma arquitetura de integração entre o SGWfC Kepler e o esquema de representação de dados da bioinfomática GUS de modo a aproveitar a tecnologia provida por ambas soluções tanto para definição do experimento e sua execução automatizada, quanto para o armazenamento em estruturas compatíveis com os padrões de representação dos dados gerados no domínio da bioinformática e persistência desses dados. É apresentada uma solução para gerência de dados intermediários em bioinformática que utiliza padrões como o GUS, serviços web e *workflows* de tal forma a obter a persistência automática durante a execução do *workflow* científico.

Optamos pelo Kepler como SGWfC a fazer parte da solução por apresentar uma interface de usuário amigável, contemplar as necessidades de definição e execução de *workflows* e ser construído sob uma plataforma facilmente expansível. O GUS foi o esquema de representação escolhido por ser muito abrangente, possuir representação para dados de proveniência compatível com qualquer *workflow* que represente experimentos biológicos, além de ser amplamente utilizado por diversos grupos de pesquisa renomados.

Através da análise das arquiteturas do Kepler e do GUS, assim como seus respectivos modelos de dados, foi elaborada uma arquitetura que engloba uma camada intermediária (*middleware*), nomeada ***Storage Layer***, entre o SGWfC e o repositório de dados do GUS e uma camada de integração, chamada de ***Integration Layer***, com o SGWfC para permitir que o usuário possa construir experimentos de modo simplificado, modelando-os no Kepler e indicando o armazenamento dos dados gerados quando for desejado.

As camadas incluídas na arquitetura são compostas por elementos desenvolvidos de modo a permitir que, ao projetar o *workflow*, o cientista possa definir qual informação será repassada para o GUS sem dificuldades. Além disso, estes componentes também são responsáveis pelo tratamento dos dados gerados durante a

execução, pelo mapeamento desses dados para as entidades do GUS, pela atualização e pela persistência dos dados no repositório.

Além disso, foi desenvolvido, em parceria com pesquisadores do BiowebDB, um estudo de caso que demonstra o uso da arquitetura proposta. Foi projetado um *workflow* que representa o processo do ProtozoaDB [34], um dos projetos do BiowebDB. Este *workflow* foi projetado utilizando os atores desenvolvidos para a camada de integração da arquitetura e os dados gerados durante a execução do experimento são armazenados no GUS através da *Storage layer*, que faz parte da arquitetura proposta.

Observando os pontos descritos acima, foi construída uma solução de apoio à proveniência de dados e sua análise capturando automaticamente os dados intermediários e utilizando o esquema GUS para o mapeamento e armazenamento dos dados de proveniência gerados ao longo do ciclo de vida dos experimentos da bioinformática. Desse modo, são permitidas análises mais sofisticadas sobre as informações geradas por experimentos científicos.

Com o processo de análise e levantamento do experimento envolvido com o projeto ProtozoaDB, necessário para implementação do estudo de caso, foi gerado um protótipo que otimizou as atividades de visualização e exploração dinâmicas dos genomas de protozoários e foi de grande importância para o grupo BiowebDB. Além disso, este trabalho também contribui em disponibilizar para a comunidade científica um protótipo funcional para armazenamento de dados gerados em *workflows* da bioinformática no GUS.

Finalmente, mesmo existindo diversas iniciativas para gerência de dados intermediários, nenhuma delas propõem a integração de SGWfCs, que mantém soluções estáveis para o problema de definir e de executar *workflows* científicos, com um esquema de representação padronizado e específico para o domínio da bioinformática, que soluciona muito bem o problema da acomodação dos dados dentro de um determinado contexto científico. Com este trabalho é apresentada uma estratégia para unir as duas vantagens de modo a otimizar e aprimorar as pesquisas científicas que envolvem experimentos *in silico*.

1.3 – Organização dos Capítulos

Para facilitar a compreensão deste trabalho, o segundo capítulo descreve as tecnologias e os conceitos relacionados tanto à biologia molecular quanto a *workflows* científicos, apresentação das características dos principais SGWfCs analisados durante esta pesquisa, modelos de representação de dados na bioinformática, proveniência e gerência de dados intermediários. Este capítulo também apresenta trabalhos correlatos nesta área.

O capítulo seguinte descreve as arquiteturas do Kepler e do GUS, estratégias computacionais relacionadas à solução proposta por este trabalho. Uma descrição detalhada da arquitetura proposta, apresentando todos os seus módulos e como os serviços web e atores foram desenvolvidos.

O quarto capítulo desta dissertação apresenta o ProtozoaDB, listando seus requisitos e descreve a implementação do protótipo produzido como estudo de caso para a arquitetura proposta. A metodologia utilizada para o estudo de observação ao qual a solução foi submetida e as análises geradas a partir deste estudo são apresentadas no capítulo cinco.

O sexto e último capítulo se propõe à conclusão desta dissertação, resumindo seu conteúdo, listando as contribuições obtidas pela pesquisa e apontando para futuros trabalhos.

Capítulo 2 – Biologia Molecular e Bioinformática

"A mente que se abre a uma nova idéia jamais volta ao seu tamanho original."
(Albert Einstein)

Uma estrutura que se assemelha a uma escada de pintor torcida, comumente descrita como "dupla hélice", após a bem conhecida H_2O da água, representa provavelmente a molécula mais famosa do mundo. O DNA, abreviação em língua inglesa da substância ácido desoxirribonucléico (ADN, em português) [35] teve sua descoberta em 1869 quando o bioquímico alemão Johann Friedrich Miescher buscava determinar os componentes químicos do núcleo celular, derrubando a teoria de que o material hereditário era composto por proteínas. A molécula se tornou um ícone em 2000, quando o Projeto Genoma Humano [36 e 37] e a empresa norte-americana Celera desvendaram os caracteres químicos da maioria dos três bilhões de "degraus", ou pares de bases, das cadeias de DNA nos 23 cromossomos da espécie humana [35].

A imagem da dupla hélice é o estereótipo da ciência natural da atualidade, assim como foi o desenho esquemático do átomo na década de 50, simbolizando a energia nuclear. Da identificação de genes envolvidos no câncer à polêmica dos alimentos transgênicos, com uma freqüência cada vez maior, temas de genética ou biotecnologia se apresentam para a sociedade, despertando esperança e temor ao redor da figura do DNA, que aparece associada a livros, ilustrações de jornal, páginas na internet ou em vinhetas da TV [35].

Este é o capítulo que introduz a base teórica para entender o DNA, o código genético, os genes e as proteínas, assim como, as técnicas mais conhecidas da biologia molecular que se relacionam com a obtenção, a identificação e a caracterização de genes. As principais aplicações da biologia molecular computacional também serão expostas neste capítulo. Conceitos da bioinformática, seus desafios e aplicações, descrição dos bancos de dados da biologia molecular e esquemas de dados padronizados para esta área científica serão expostos neste capítulo.

2.1 – Principais Conceitos da Biologia Molecular

Um processo de transformação que durou milhões de anos deu origem à **célula**, com os primeiros seres vivos que passaram a desenvolver mecanismos cada vez mais eficientes para captação, armazenamento e liberação de energia para sustentar suas atividades [41]. A célula é a unidade estrutural e funcional dos organismos vivos. Até hoje ainda existem seres vivos **unicelulares**, como bactérias. Outros organismos tais como os seres humanos são **pluricelulares**.

Os **vírus** dependem das células para que sua multiplicação possa ocorrer [44], pois não possuem metabolismo próprio. Estes seres são parasitas intracelulares formados apenas por um dos **ácidos nucléicos** (DNA ou RNA) envolvido por um revestimento protéico.

As células podem ser classificadas segundo sua organização estrutural como [44]:

Procariontes: não possuem envoltório para o **núcleo celular** ou **carioteca**;

Eucariontes: possuem duas partes bem distintas em sua composição, o **citoplasma**, envolvido pela **membrana plasmática** e o núcleo celular, envolvido pela carioteca.

2.1.1 – Cromossomos

Finíssimos filamentos enovelados, chamados de **cromossomos**, se encontram no interior dos núcleos celulares. O número de cromossomos presentes nas células varia de acordo com cada espécie, como por exemplo, o homem com 46 cromossomos nos núcleos de suas células e as moscas das frutas (*Drosophila Melanogaster*) com 8 cromossomos. Os cromossomos **homólogos**, ou seja, que se apresentam em pares que possuem as mesmas características e funcionalidades, estão presentes na maioria das células [43]. As células que apresentam os cromossomos homólogos são chamadas de células **diplóides** (2n). As células que contém apenas um dos representantes são chamadas de **haplóides** (n) e são, normalmente, formadas por **meiose** para a produção de **gametas**.

Cromossomos são estruturas formadas por uma dupla fita de DNA e proteína que contém vários genes, elementos reguladores e outras seqüências de **nucleotídeos**. Os nucleotídeos são compostos por uma **base nitrogenada**, uma **pentose** (açúcar) e um **grupo fosfato**. A orientação das ligações entre estas três moléculas é essencial para determinar o sentido da dupla fita de DNA. Abaixo se encontra a figura 2.1 que descreve a ligação entre estas moléculas, formando um nucleotídeo [42].

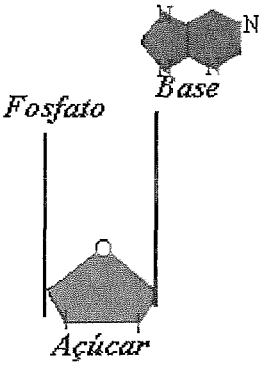
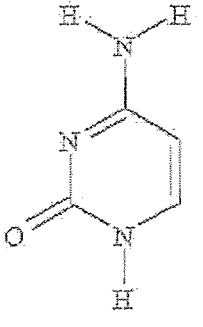
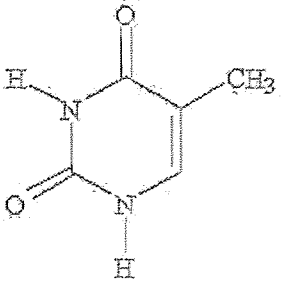


Figura 2.1. Moléculas que compõem os nucleotídeos.

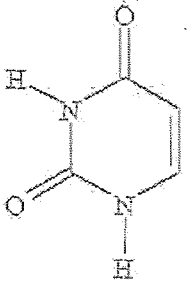
Com base na estrutura de suas bases nitrogenadas os nucleotídeos podem ser divididos em dois grupos: os que possuem **bases purinas** ou os que possuem **bases pirimidinas**. Tanto o DNA quanto o RNA possuem duas bases purinas: **Adenina (A)** e **Guanina (G)**. A base pirimidina **Citosina (C)** também faz parte dos nucleotídeos que compõem estes dois ácidos nucléicos. Entre as bases de DNA e RNA existe uma diferença na segunda base nitrogenada do tipo pirimidina. No DNA esta base será a **Timina (T)** e no RNA, a **Uracila (U)** [42]. Nas figuras 2.2 e 2.3 é possível visualizar a estrutura das bases purinas e pirimidinas encontradas nas moléculas de DNA e RNA.



CITOSINA



TIMINA



URACILA

Figura 2.2. Estrutura das bases pirimidinas: Citosina, Timina e Uracila.

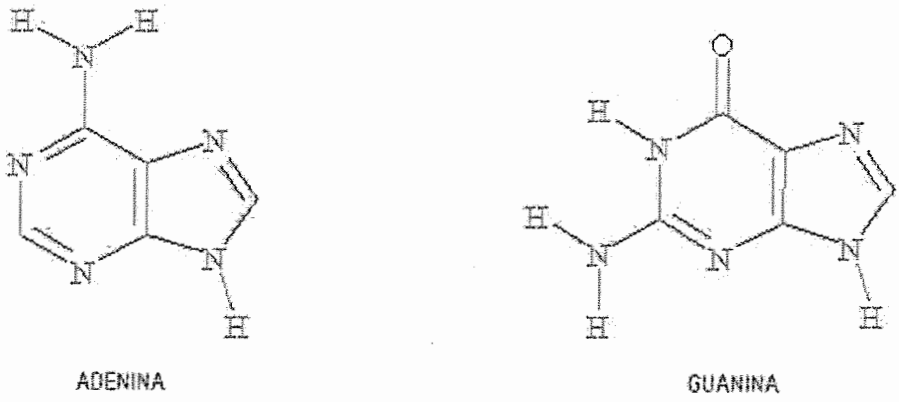


Figura 2.3. Estrutura das bases purinas: Adenina e Guanina.

Proteínas chamadas **histonas** fazem parte da composição do cromossomo. Elas são arranjadas em grupos de oito, envolvidos pela molécula de DNA. Estes grupos de proteínas aglomeradas envolvidos pela dupla hélice de DNA são chamados de **nucleossomos**. A partir dos vários nucleossomos presentes no cromossomo forma-se sua estrutura solenóide. Na figura 2.4 é exibida a composição descrita para os nucleotídeos [44].

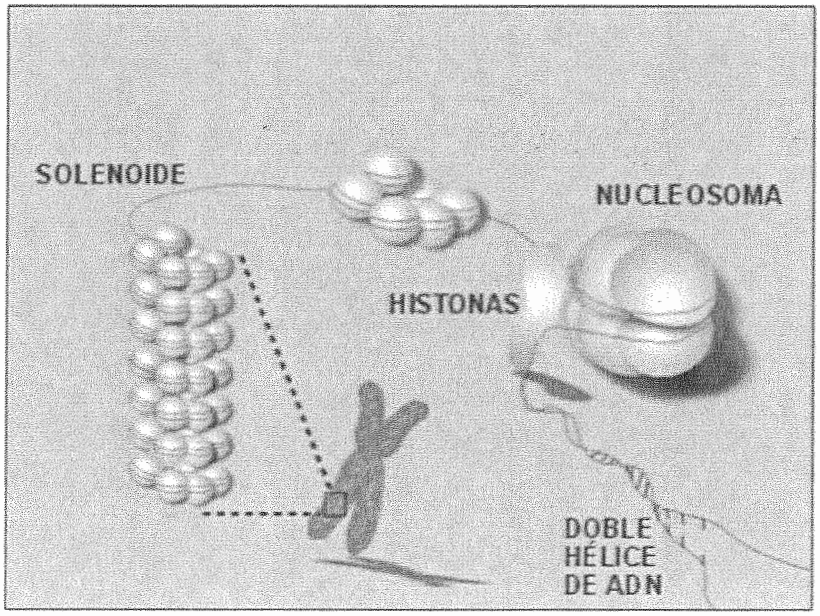


Figura 2.4. Composição dos nucleotídeos.

2.1.2 – DNA

O DNA é um ácido nucléico que contém todas as instruções genéticas para o desenvolvimento e funcionamento de todos os seres vivos e alguns vírus. Esta molécula é composta por uma **desoxirribose** e um **grupo fosfato**. As quatro bases nitrogenadas contidas nesta molécula são: adenina, citosina, guanina e timina [42]. Estas bases ficam localizadas no interior da hélice e se mantêm unidas pela fraca ligação entre duas bases complementares: adenina se junta com timina (A→T) e citosina com guanina (C→G).

De modo semelhante a um espiral, os dois filamentos que compõem o DNA se enrolam, formando uma dupla hélice, que pode ter milhares de nucleotídeos. As duas fitas formam sempre uma cadeia complementar. Se tivermos uma cadeia com a seqüência ATAGCTAAC, a cadeia complementar será TATCGATTG. Abaixo, na figura 2.5, visualizamos a estrutura do DNA e sua presença no cromossomo [44].

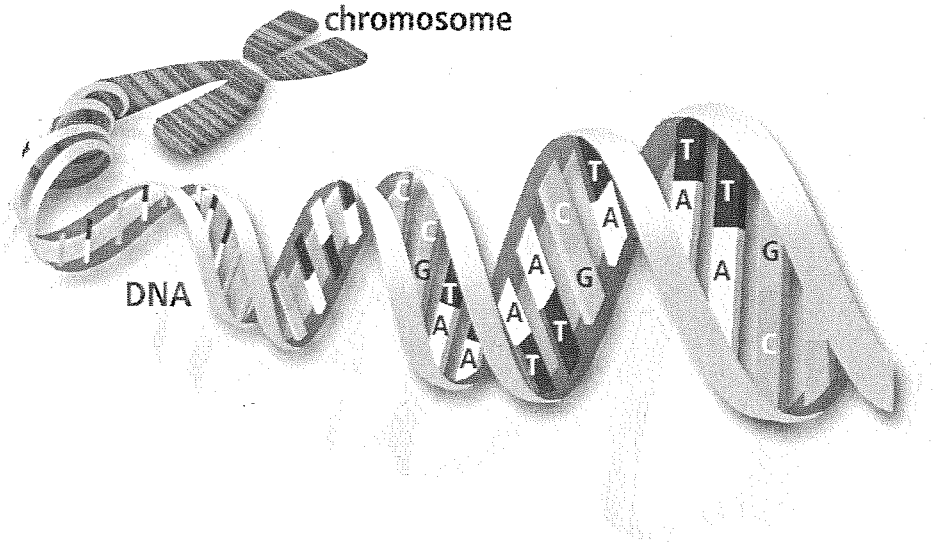


Figura 2.5. Estrutura do DNA [38].

Os trechos do DNA que são responsáveis por carregar a informação genética do organismo são chamados de **genes**. A parte do DNA que contém genes é chamada de **região codificante** [45]. Os segmentos da seqüência que não possuem carga genética têm importância estrutural ou estão envolvidos em atividades que regulam o uso da informação genética. Esta parte do DNA é chamada de **região não codificante**.

Os genes contêm todas as características biológicas de um organismo, que devem ser transmitidas aos seus descendentes e para as células filhas na proliferação celular, que faz parte do desenvolvimento do ser vivo. **Genoma** [45] é todo o conjunto de genes de um organismo.

2.1.3 – Síntese de Proteínas

As características de um indivíduo são resultado das **proteínas** que formam seu corpo. Estas são macromoléculas formadas por **aminoácidos**, que são moléculas orgânicas constituídas por dois agrupamentos especiais chamados amina e carboxila que se unem através de uma ligação peptídica [46].

São variados os tipos de proteínas existentes na natureza. No metabolismo dos seres vivos, podemos citar, como exemplo: as **enzimas** [46], responsáveis por acelerar e regular processos vitais; os **hormônios**, que transportam sinais químicos que regulam os ciclos vitais; os **anticorpos**, que protegem o corpo contra substâncias que o invadem; as **hemoglobinas**, que transportam oxigênio no sangue; a **actina** e a **miosina**, responsáveis pela contração muscular; e as **albuminas**, proteínas de reserva.

As moléculas de DNA possuem as instruções de como produzir cada parte que compõe um organismo, por isso, ela se torna um componente fundamental da **síntese de proteínas** [47]. O DNA coordena a produção de todas as proteínas necessárias ao ser vivo e, conseqüentemente, define todas as características físicas, fisiológicas e até mesmo comportamentais de um organismo. O processo da síntese protéica é dividido em duas etapas chamadas de **transcrição** e **tradução** de genes.

A transcrição é a primeira etapa deste processo. Seu objetivo é sintetizar a molécula chamada de **RNA mensageiro**, ou **mRNA**, a partir de uma das fitas do DNA, que serve como molde para o mRNA. Esse fluxo tem início em uma região do núcleo, chamada **nucléolo**, quando a enzima **RNA polimerase** encontra uma região específica do DNA, chamada de **promotor**. Nos organismos eucarióticos, esta cadeia de nucleotídeos é identificada pela presença da seqüência TATA, também chamada de **TATA box**. Um grupo de proteínas chamadas de **fatores de transcrição**, encontram e se conectam a este trecho da seqüência de DNA, com a meta de ajudar a RNA polimerase a localizar o início da seção que será transcrita [48]. A figura 2.6 ilustra esta etapa.

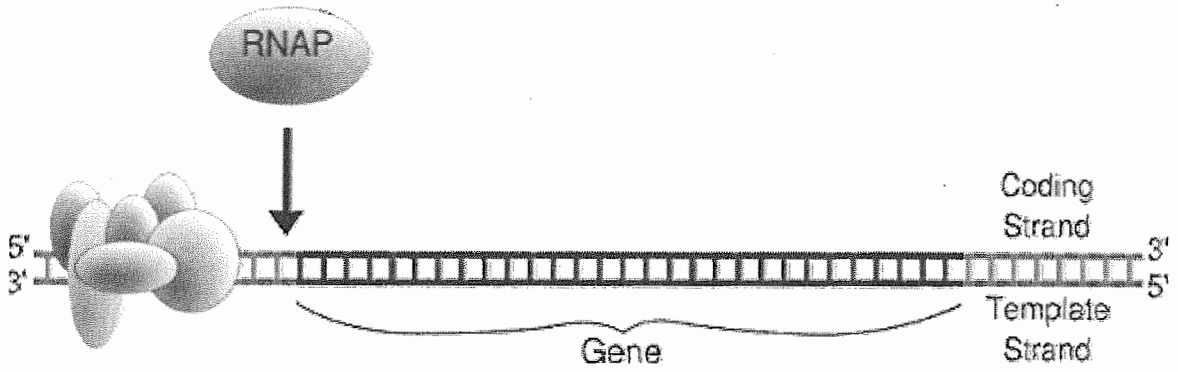


Figura 2.6. Início da fase de transcrição. RNAP = RNA Polimerase [52].

A enzima RNA polimerase II, responsável por transcrever a fita **pré-mRNA**, encontra o promotor e quebra a dupla hélice do DNA na região identificada. Esta enzima percorre a fita de DNA que serve de modelo para a transcrição e monta a molécula de pré-mRNA de acordo com as bases que são lidas na fita de DNA, substituindo apenas as bases nitrogenadas timina (T) por uracila (U), deixando claro que a molécula que está sendo sintetizada é uma cópia. A nova fita possui as bases A, U, G e C e é liberada quando encontra uma região do DNA, chamada de **terminador**. Este processo é representado na figura 2.7.

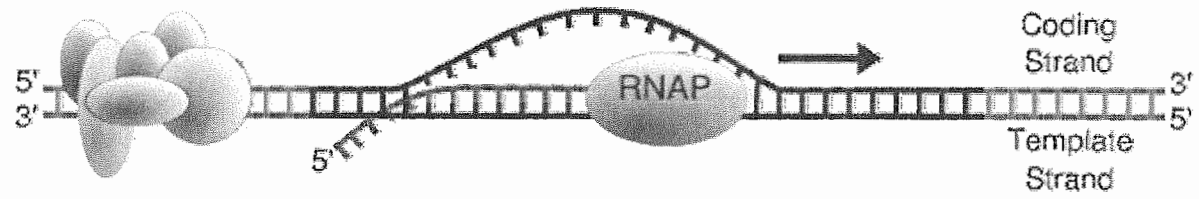


Figura 2.7. RNAP percorre a fita de DNA, gerando pré-mRNA [52].

Após este processo, molécula de DNA permanece no núcleo e, antes de deixar o núcleo celular, a fita pré-RNA é modificada em suas extremidades para evitar sua degradação no citoplasma. Em uma das pontas da fita é adicionada a região **5'**, identificada pela cadeia AUGAGCU, e na outra ponta é adicionada a região **3'**, que possui de 100 a 200 bases nitrogenadas de amina (A). A fita pré-mRNA se transforma então na fita de **mRNA** e deixa o núcleo celular [48]. Na figura 2.8 é possível visualizar a finalização da etapa de transcrição do DNA.

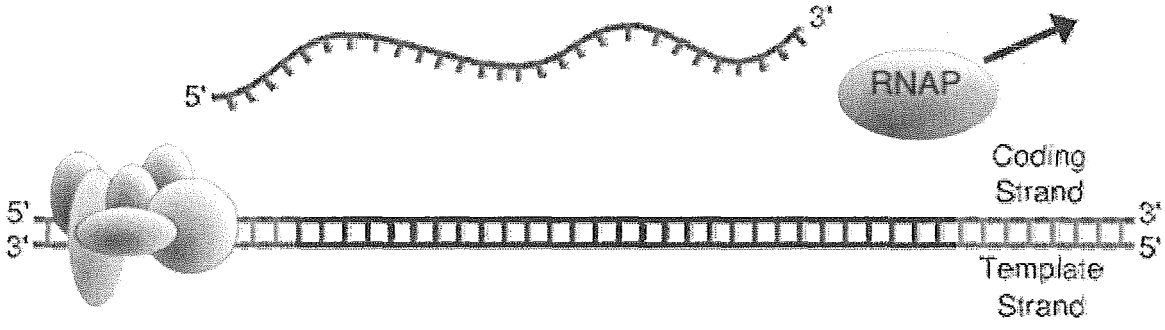


Figura 2.8. Finalização da etapa de transcrição [52].

A segunda etapa no processo de síntese de proteínas [49] é conhecida como tradução. Nesta fase ocorre a produção de uma **cadeia polipeptídica**, também chamada de **proteína**, utilizando a informação genética contida na molécula mRNA. Nesta fase, os nucleotídeos são traduzidos para aminoácidos, utilizando a tabela de código genético para determinar a seqüência de aminoácidos. Na tradução, cada grupo de três nucleotídeos, ou **códon**, especifica um aminoácido em particular. Por exemplo, a seqüência AAU indica que o aminoácido asparagina deve ser adicionado à cadeia polipeptídica.

O processo de tradução tem início quando uma molécula de **tRNA**, ou **RNA transportador**, é acoplado ao **start códon**, que identifica o início da região codificante da cadeia de mRNA. A moléculas de tRNA carrega um aminoácido que fará parte da seqüência que formará a proteína sintetizada ao final do processo. Na composição do tRNA existe uma seqüência de três nucleotídeos chamada de **anticódon**, que

complementa o códon na molécula de DNA. No interior dos **ribossomos** [49], códons e anticódons são ligados por pontes de hidrogênio. Os aminoácidos transportados pelos tRNAs acoplados à cadeia de mRNA são unidos por ligações peptídicas. O ribossomo se desloca ao longo da fita de mRNA liberando as moléculas de tRNA cujos aminoácidos foram acoplados à cadeia polipeptídica. A tradução é finalizada quando é alcançado o **stop códon**, e a proteína é liberada. Vários ribossomos podem percorrer a mesma cadeia de mRNA, sintetizando várias cópias da proteína simultaneamente. A imagem 2.9 oferece uma visão geral de todo o processo de síntese de proteínas.

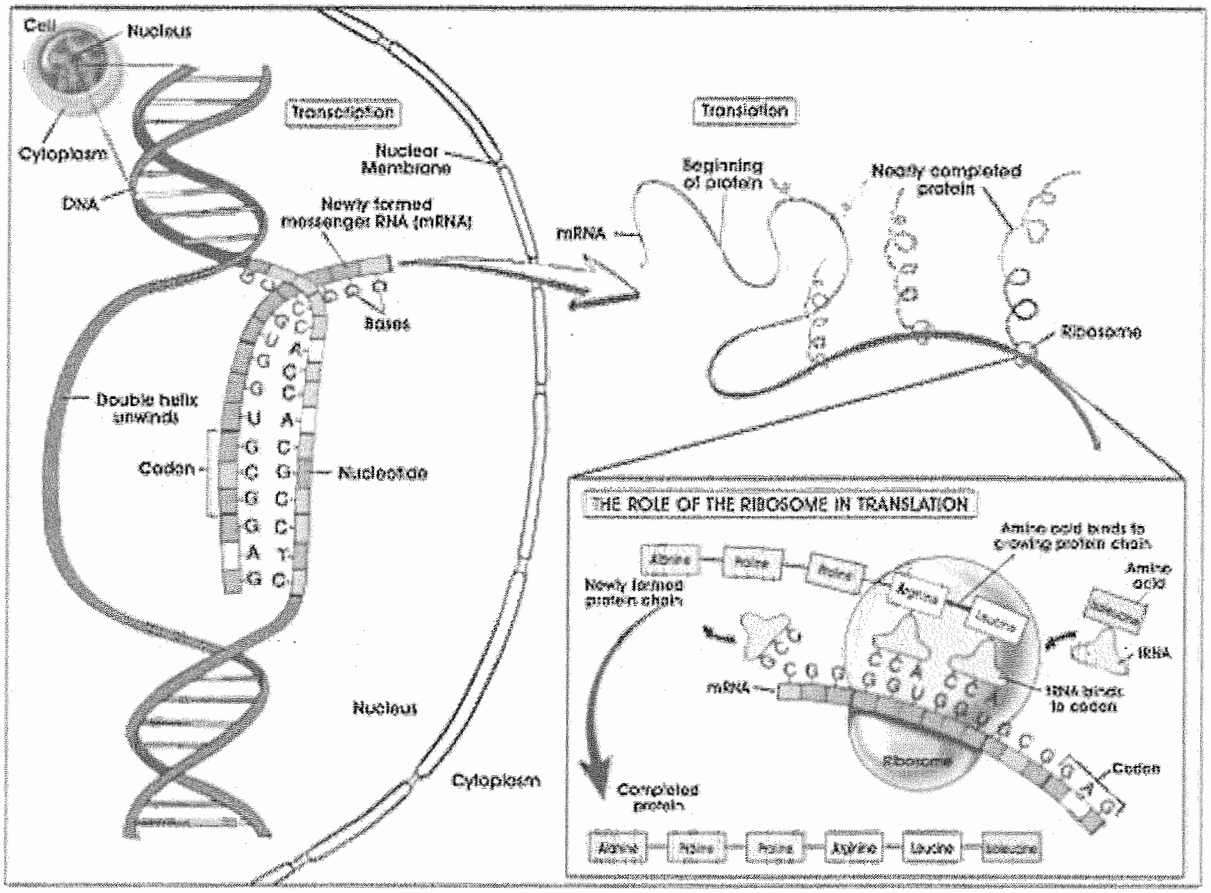


Figura 2.9. Transcrição e Tradução de Genes e síntese protéica [53].

Inferir a estrutura e a função das proteínas sintetizadas nas células dos organismos representa um grande problema [50]. A estrutura das proteínas é tridimensional e única para cada proteína e sua função depende de sua estrutura [51].

2.1.4 – Principais Técnicas da Biologia Molecular

A maior parte das pesquisas na biologia molecular se relaciona com a obtenção, identificação e caracterização de genes. As principais técnicas desenvolvidas no intuito de auxiliar estas atividades são [40]:

Eletroforese em Gel: técnica que utiliza papel, agarose e poliacrilamida para movimentar moléculas submetidas a um campo elétrico. Separa proteínas e ácidos nucléicos considerando cargas e pesos moleculares.

Reação em Cadeia da Polimerase (PCR): permite obter várias cópias de um trecho de molécula de DNA, introduzindo ou não alterações pretendidas nessa seqüência. É uma das técnicas mais comuns utilizadas em pesquisas médicas e biológicas para tarefas como o seqüenciamento de genes, diagnóstico de doenças hereditárias, testes de paternidade e medicina forense [39, 40].

***Southern Blotting*:** consiste na imobilização de DNA digerido com enzimas de restrição em membranas de nylon. Esta metodologia pode ser utilizada para análise do número de cópias de um gene, assim como para a identificação de polimorfismos.

***Northern Blotting*:** procedimento similar ao southern blotting que procura estudar o perfil quantitativo de expressão genética envolvido nas moléculas de mRNA.

***Western Blotting*:** utiliza os mesmos princípios das técnicas southern blotting e northern blotting para a pesquisa de proteínas.

2.2 – Bioinformática

A bioinformática é uma área de pesquisa interdisciplinar que surgiu a partir da enorme demanda de recursos computacionais para auxiliar na solução de problemas biológicos [50]. O termo bioinformática está inserido no contexto da biologia computacional como uma subárea que envolve a ciência da computação, a matemática,

a estatística e a física, para analisar e comparar seqüências genômicas ou protéicas. Para apoiar estas atividades de pesquisas biológicas, a bioinformática provê ferramentas da computação como bancos de dados, redes e técnicas de inteligência artificial [54].

Em algumas áreas de pesquisa científica como a bioquímica e a biologia molecular o uso das soluções de bioinformática é indispensável para a geração de resultados mais precisos. O uso da bioinformática cresceu de tal modo que a quantidade de informação gerada por suas ferramentas cresce exponencialmente apontando para pesquisas no sentido de gerenciá-la e manipulá-la. A biologia molecular é uma das áreas que mais utilizam recursos computacionais com o objetivo de determinar genomas de várias espécies através do seqüenciamento de cadeias de DNA, identificação da localização e função dos genes.

Um dos tópicos mais importantes para a biologia molecular envolve o trinômio *função do gene – função da proteína – estrutura da proteína*, pois a predição da função de um gene é possível a partir da determinação da função da proteína que depende de sua estrutura [51]. Para oferecer suporte a estas atividades de pesquisa, destacam-se três áreas de pesquisa em bioinformática: **bancos de dados da biologia molecular**; **comparação e análise de proteínas** e **predição de estruturas de proteínas**. As duas primeiras estão diretamente relacionadas ao contexto desta dissertação.

2.2.1 – Bancos de Dados da Biologia Molecular

Os dados biológicos geralmente são mais complexos do que dados da maioria dos outros domínios de aplicações. Eles possuem características próprias que fazem de seu gerenciamento uma atividade complexa. Algumas dessas principais características são [50]:

- Estes dados representam subestruturas complexas e os relacionamentos entre elas. O contexto desses dados deve ser armazenado para a correta interpretação.

- O volume de dados extremamente grande, assim como a variedade de tipos que os representam, implicando que sistemas biológicos devem ser flexíveis.

- Novas versões dos esquemas das fontes de dados biológicos são constantemente lançadas. Por isso, é necessário que as aplicações de bioinformática contemplem informações sobre diferentes versões dos dados, assim como seu histórico.

- O mesmo dado pode ser apresentado de maneiras distintas exigindo mecanismos para alinhar diferentes representações das informações.

- Os sistemas propostos para dar suporte aos dados biológicos devem suportar a definição e representação de consultas complexas sem exigir que o biólogo possua conhecimentos computacionais avançados.

Os bancos de dados da biologia molecular podem armazenar seqüências genéticas, estruturas de proteínas e funções de genes e proteínas, assim como anotações genômicas. Eles são divididos de acordo as mais variadas características. Existem bancos de dados específicos para genomas de organelas, como o GOBASE [55], bancos de dados de RNA, como o tRNA database [56], dados de classificação filogenética de proteínas, como o COG [57], ontologias para genes, como o AmiGO [58], dentre muitos outros específicos para cada tipo de informação.

Os principais bancos de dados são os que armazenam informações de seqüências de DNA, RNA e proteínas e também os que guardam dados de posicionamento da estrutura tridimensional das proteínas. No primeiro grupo, os que mais se destacam são o EMBL *Nucleodite Sequence Database* [59], mantido pela EMBL (*European Molecular Biology Laboratory*), o GenBank *Sequence Database*, mantido pelo NCBI (*National Center for Biotechnology Information*) [60] e o Swiss-Prot, ou *Annotated Protein Sequence Database* [62]. No segundo grupo, o principal é o PDB, ou *Protein Data Bank* [63], armazenando dados 3-D das macromoléculas de proteínas.

Muitos dos chamados bancos de dados pela comunidade da área biológica não são de fato bancos de dados, pois não utilizam sistemas gerenciadores de banco de dados e, sim, sistemas de arquivos do sistema operacional ou sistemas de arquivos próprios ou, até mesmo, uma combinação das três estratégias. Estes bancos são na verdade fontes de dados ou bases de dados, chamadas de bancos de dados neste trabalho somente para igualar a terminologia. Cada um desses bancos possui sua própria estrutura de armazenamento, manipulação e métodos de acesso, assim como esquemas

próprios e algoritmos de análise dos dados específicos, o que caracteriza uma grande heterogeneidade para o armazenamento dos dados nos diversos projetos de pesquisa.

O GenBank [61] é o banco de dados para seqüências genéticas do NIH (*National Institute of Health*) que foi estabelecido em 1978 como um repositório central para dados biológicos. Ele disponibiliza uma coleção de todas as seqüências de DNA publicadas, assim como suas anotações. Em fevereiro de 2008 foram estimadas 85.759.586.764 bases em 82.853.685 registros de seqüências nas seções tradicionais do GenBank e 108.645.736.141 bases em 27.439.206 registros de seqüências em suas seções de WGS (*Whole Genome Shotgun*), ou seja, projetos em andamento.

O sistema do GenBank é mantido como uma combinação de arquivos "*flat*", banco de dados relacionais e arquivos no formato *Abstract Syntax Notation One* (ASN.1). Inicialmente a submissão de dados de seqüência era realizado diretamente por seus autores, mas atualmente os dados de outros bancos públicos de seqüências, como o EMBL e o DDBJ [64] são incluídos automaticamente em seu repositório através da troca de dados diária entre esses três sistemas.

O *Protein Data Bank* (PDB) foi estabelecido em 1971 e consiste em um repositório universal de dados de estruturas de macromoléculas biológicas submetidas pelos pesquisadores de todo o mundo a partir dos resultados de suas pesquisas em laboratórios. Inicialmente o PDB continha apenas sete estruturas de proteínas em seu repositório. Desde então, as informações que armazena crescem de modo aproximadamente exponencial [65].

O Swiss-Prot é um banco de seqüências de proteínas com anotações que foi criado em 1986. O conhecimento das proteínas armazenadas no Swiss-Prot consiste de entradas de seqüências, que são compostas de diferentes tipos de linhas, cada qual com seu próprio formato. Com propósito de padronização, o formato do Swiss-Prot segue aproximadamente o formato do EMBL *Nucleodite Sequence Database*.

2.2.2 – Comparação e Análise de Seqüências

Quando um gene é seqüenciado por dois laboratórios distintos, geralmente, deseja-se comparar os dois resultados. Essa comparação de seqüências biológicas também é necessária quando duas seqüências consecutivas precisam ser agrupadas em

uma. A tarefa de encontrar trechos similares em várias seqüências é uma das mais importantes da biologia molecular [66]. A busca de seqüências similares que percorre todas as cadeias armazenadas em um banco de dados permite a comparação entre seqüências de várias espécies, permitindo a inferência de semelhanças entre elas ou para localizar diferenças que podem indicar doenças hereditárias.

A comparação de seqüências é um dos procedimentos da área biológica que mais demanda uso de recursos da computação. Vários algoritmos foram desenvolvidos utilizando a noção de **alinhamento** entre duas seqüências. De acordo com os métodos adotados por cada algoritmo, o alinhamento ótimo é aquele que possui o mais alto grau de **similaridade** entre duas seqüências.

Os algoritmos que processam todos os alinhamentos possíveis para descobrir o que possui similaridade ótima são chamados de **Algoritmos de Programação Dinâmica**. Os dois algoritmos mais conhecidos e que servem de base para a maioria dos métodos são Needdleman-Wunsch [67], utilizando alinhamento global, e Smith-Waterman [68], utilizando alinhamento local. O tempo de processamento desses algoritmos é proporcional ao tamanho das seqüências envolvidas. Por isso, esses métodos são utilizados apenas quando uma comparação precisa é requerida.

FASTA [70] e BLAST [69] são **Algoritmos Heurísticos**, que utilizam técnicas de heurística para inferir quais bases das seqüências não necessitam de comparação. Os dois algoritmos identificam segmentos pequenos e altamente similares que são então expandidos, assumindo que qualquer alinhamento significante cerca um ou mais desses segmentos que foram expandidos. O número de computações baseadas nas comparações é altamente reduzido. Apesar de não garantirem a descoberta de alinhamentos ótimos são eficazes e rápidos na comparação de seqüências.

2.2.3 – Predição de Estruturas de Proteínas

A função de um gene é determinada pela estrutura tridimensional da proteína sintetizada por ele. Por isso, as tarefas de **predição de estruturas de proteínas** são de extrema importância no âmbito da biologia molecular. Apesar desse imenso interesse, esta é uma das áreas que envolvem maior complexidade. As forças físicas que existem nas interações das diferentes combinações de aminoácidos também determinam a estrutura da proteína e ainda não são totalmente conhecidas.

Modelos teóricos produzidos por ferramentas computacionais geram estruturas que não são suficientemente precisas e se baseiam em dois métodos [71]. Um deles é a **homologia**, que executa comparações entre a estrutura 3-D da proteína pesquisada e as estruturas de outras proteínas com funções já determinadas. O outro método prediz a estrutura de uma seqüência isolada, sem se preocupar com similaridades. Esta técnica busca a estrutura que minimiza a energia livre de uma proteína, no método conhecido como ***ab initio*** ou ***de novo***. A precisão de modelos comparativos é maior que a dos métodos de energia livre, mas dependendo das circunstâncias, é possível uma boa predição em ambos os casos.

A popularização do uso da computação permitiu que diversas análises e descobertas fossem efetuadas por agilizar e facilitar as atividades envolvidas nas pesquisas, mas gerou um aumento exponencial da quantidade de informação produzida. Por isso, mesmo com todos os benefícios que se esperam, os projetos de biologia molecular tornaram-se um grande desafio.

2.2.4 – Esquemas de Dados Padronizados para a Bioinformática

Várias alternativas tentam produzir modelos dados para a bioinformática, mas consideram apenas o contexto científico dos seus centros de pesquisa. Os dois esquemas genéricos para dados biológicos que mais se destacam são o Chado [76] e o GUS [29]. O Chado é um dos subprojetos do GMOD (*Generic Model Organism Database*) cujo esquema relacional é simples e voltado principalmente para a representação de seqüências e ontologias. O GUS possui um esquema mais rico que permite representar, além de seqüências e ontologias, expressão e regulação gênica.

A plataforma GUS foi escolhida para compor a estratégia proposta neste trabalho devido à abrangência de seu esquema de dados que pode acomodar uma enorme variedade de dados biológicos, sendo mais compatível com qualquer *workflow* que represente experimentos biológicos. Além disso,diversos projetos vêm adotando o GUS, por este ser o mais próximo de um esquema genérico para a bioinformática.

O esquema do GUS é organizado em cinco módulos. Os sub-esquemas: DoTS, RAD, e TESS estão relacionados a dados genômicos básicos como seqüências de transcrição, microarranjos e regulação gênica, respectivamente. Os outros dois

esquemas, Core e SRes, representam projetos, usuários, algoritmos, bases de dados e ontologias. Estes dois últimos representam dados de proveniência dos experimentos, possibilitando, por exemplo, relacionar os dados gerados com os criadores desses dados, ou seja, quem gerou esses dados.

Além disso, a plataforma GUS disponibiliza um conjunto de ferramentas de desenvolvimento para consultas na *web*, chamado GUS WDK (*Web Development Kit*) [77]. Desse modo, desde que os dados estejam sendo armazenados a partir do experimento que está sendo executado, é possível efetuar consultas sobre esses dados.

2.2.5 – Desafios da Bioinformática

Com base no nível de dependência em relação a recursos computacionais pelos cientistas da biologia molecular nota-se uma demanda cada vez maior por ferramentas eficientes e sofisticadas que auxiliem nas tarefas de coleta, representação, organização, armazenamento e análise de todas as informações genéticas [72, 73 e 74]. Por isso, os principais desafios da área de bioinformática podem ser resumidos em:

Grande heterogeneidade de bancos de dados e aplicações – Devido à imensa quantidade e variedade dos dados biológicos, os diversos sistemas são projetados conforme as necessidades de cada centro de pesquisa gerando representações específicas em cada um deles. Também é comum encontrar formatos de dados em representações variadas. As bases são desenvolvidas utilizando diferentes mecanismos, como arquivos com textos semi-estruturados, coleções de páginas web e sistemas gerenciadores de banco de dados. A heterogeneidade também se faz presente nos programas e aplicações utilizados pelos cientistas, sendo comum encontrar várias soluções para um mesmo problema ou algoritmo. Por isso, questões como interoperabilidade e escalabilidade são grandes desafios nesta área.

Necessidade de integração entre diversos bancos de dados – compartilhar diferentes informações complementares e relacionadas levanta questões como autonomia das bases individuais, distribuição e consistência das informações científicas. Fatores que dificultam essa integração na área de bioinformática são: falta de um esquema global dos dados; necessidade do uso de texto puro nas consultas, além do amplo uso de um campo de texto livre, conhecido como

anotação [75], utilizado para guardar desde informações relevantes provenientes da análise individual do biólogo, até os dados sobre as condições do experimento (quantidade de material, luminosidade, pH, presença de alguma substância, etc.).

Definição e execução de experimentos com a composição de várias aplicações – Quando o processo de experimentação científica através dessas duas abordagens é executado pela utilização intensiva de recursos computacionais, convencionou-se chamar de experimento *in silico*, uma analogia ao termo *in vitro*. Os experimentos *in silico*, têm o objetivo de representar cada uma das atividades anteriormente feitas em laboratório, por programas de computadores que devem ser executados em um determinado fluxo, onde normalmente a saída de um dos programas servirá de entrada para o próximo na cadeia de execução. Dessa forma, efetua-se uma composição dos diversos programas disponíveis de forma que o conjunto deles satisfaça a um objetivo maior. Atualmente existem vários sistemas que procuram oferecer suporte a este tipo de atividade científica, mas ainda não se chegou a uma maneira adequada de armazenar e relacionar os dados de proveniência e os dados intermediários gerados no fluxo, ligando-os com o contexto da bioinformática.

Capítulo 3 – Gerência de Dados em *Workflows* Científicos

"Quem pensa pouco erra muito."
(Leonardo da Vinci)

Como já foi mencionado anteriormente, o uso de recursos computacionais em ambientes científicos vem apresentando um grande índice de aumento. Aplicações de bioinformática são desenvolvidas pela comunidade científica com o propósito de realizar simulações, comparações, cálculos e todo tipo de processamento necessário aos dados gerados pelos seus experimentos *in silico*.

Workflows representam um papel importante no sentido de coordenar a execução de um conjunto de ferramentas computacionais envolvidas com os experimentos científicos e, também, com o objetivo de permitir a gerência das informações geradas ao longo desse processo.

No meio científico, o uso de *workflows* tem sido adotado para descrever e executar experimentos *in silico*. Nos *workflows* científicos, os experimentos são modelados através da composição de recursos computacionais em uma seqüência de execução.

Este capítulo está dividido da seguinte forma: a primeira seção expõe os fundamentos de *workflows* e *workflows* científicos. As características dos principais Sistemas de Gerência de *Workflows* Científicos (SGWfCs) serão apresentadas na segunda seção deste capítulo. Em seguida, a necessidade de armazenar proveniência dos dados em experimentos e seu relacionamento com *workflows* científicos serão explicados. Por fim, será feita uma análise crítica em torno das ferramentas de gerência de *workflows* científicos com foco na gerência dos dados de proveniência e intermediários.

3.1 – Fundamentos de *Workflows* Científicos

Esta seção apresenta os principais conceitos relacionados aos *workflows* e *workflows* científicos.

3.1.1 – Workflows

A composição (seqüência) de programas de bioinformática faz parte das diversas tarefas realizadas pelos pesquisadores. Cada um dos programas utilizados no fluxo ou *pipeline* produz uma coleção de dados com determinada semântica e sintaxe, ligadas ao contexto científico. O recurso computacional que permite a composição de programas em uma seqüência de execução com o objetivo de gerar um resultado final é chamado de *workflow*.

O termo *workflow* tem sua origem associada ao processo de automação de escritórios, segundo [78] essa tecnologia teve seu início por volta dos anos 1970. O foco dessa tecnologia era oferecer soluções voltadas para a geração, armazenamento, compartilhamento e roteamento de documentos em uma organização, com o objetivo de reduzir a manipulação física de documentos.

Posteriormente o foco desta tecnologia passou a ser a gerência e automação de processos de negócio, e os *workflows* foram chamados de ***workflows* comerciais** (*Business Workflows*). Com isso foi possível integrar e organizar a realização de atividades interdependentes permitindo uma série de benefícios organizacionais. Melhorias nas funções de atendimento aos clientes e controle do andamento das atividades do fluxo de trabalho, aumento na produtividade por funcionário são exemplos destes benefícios.

Um *workflow* provê a abstração necessária para descrever uma série de processos estruturados e suas atividades com o objetivo de prover um ambiente robusto de resolução de problemas e assim, promover o uso efetivo e otimizado dos recursos computacionais [79]. Na literatura existem diversas definições para *workflows*. A definição, segundo [80], é:

> *"A automação de um processo de negócio, completo ou apenas parte dele, através do qual, documentos, informações ou tarefas são*

transmitidos de um participante a outro por ações, de acordo com regras procedimentais."

Para o desenvolvimento de *workflows*, destacam-se duas atividades: a **definição ou modelagem** e a **execução** de *workflows* [81]. Durante sua modelagem, o *workflow* é especificado de acordo com o fluxo de execução das tarefas do processo, através da indicação dos passos envolvidos, restrições, ordem de execução, desvios do fluxo normal da execução, tratamento de dados e tratamento de erros. A execução das atividades definidas na etapa de modelagem ocorre na fase de execução do *workflow*, que segue a seqüência estabelecida na sua definição. A execução considera a ordem definida para as tarefas do *workflow*, transformação de dados na transferência entre tarefas, invocação de tarefas de modo automático ou manual e, ainda, ferramentas ou recursos que permitam o monitoramento da execução e estatísticas.

3.1.2 – *Workflows* Científicos

Apesar de o contexto original dos *workflows* ter sido o ambiente de negócios [82 e 81], cada vez mais, áreas científicas como Biologia, Química, Física, Geografia, Engenharia, entre outras, passaram a utilizá-los para auxiliar suas pesquisas.

Na bioinformática, é comum o uso de *scripts* que definem a seqüência completa de execução e os parâmetros de execução do fluxo de trabalho, chamados de *pipelines*. No entanto, deve ser alertado que o uso de *scripts* pode tornar as atividades de pesquisa dos cientistas muito complexas. A grande especificidade de um *script* pode dificultar sua manutenção e reduzir a capacidade de reutilização. Além disso, registrar as execuções dos programas, a origem dos dados utilizados, as transformações aplicadas aos dados, os resultados obtidos, entre outras, é, muitas vezes, impossível.

Na tabela 3.1, encontram-se as principais características de *workflows* científicos apresentadas por [83, 12, 84 e 85].

Tabela 3.1 – Principais características dos *workflows* científicos

Característica	Descrição
Interface	A modelagem do *workflow* deve ser intuitiva e o ambiente de execução voltado para o usuário final. Detalhes de implementação não devem ser expostos, permitindo que o pesquisador concentre-se apenas do nível conceitual do *workflow.*
Reutilização	Os componentes deverão ser reutilizáveis e intercambiáveis. Deve ser possível adicionar ou remover novos processos dinamicamente.
Transformação de dados	Permitir transformações de dados entre as atividades de um processo.
Interações e processamento em lote	Deverá suportar "*process steering*" (*play, pause* e *stop*) durante a execução de um processo, permitindo monitoramento do experimento.
Execução parcial	Oferecer suporte à execução parcial. Apenas parte do experimento poderá ser executada.
Localidade	Deverá suportar processamento local e distribuído do *workflow.*
Complexidade	Deverá ser capaz de manipular complexos fluxos de dados, controles e eventos.
Dependência	Deverá estar associado a um sistema de gerência confiável, de alta disponibilidade e tolerante a falhas.
Verificação e Validação	Deverá verificar e validar a construção ou importação de *workflows.*
Alterações dinâmicas	Permitir alterações na definição do *workflow* durante a execução, pois a definição do *workflow* pode ser influenciada pelos resultados intermediários.

Os experimentos científicos podem ser executados inúmeras vezes, comumente sendo necessário efetuar modificações nos programas ou em parâmetros operacionais. Como *workflows* científicos modelam experimentos que podem ser modelados como um fluxo de ferramentas computacionais automatizado, alterações na definição conceitual de um *workflow* científico são constantes.

Os resultados produzidos em um experimento constituem uma fonte de dados imprescindível para os cientistas, por isso *workflows* científicos devem permitir o armazenamento de dados gerados ao final da sua execução, assim como dados intermediários gerados com a execução de cada etapa do processo. Os resultados finais são essenciais para comprovar ou negar as hipóteses postuladas no início do

experimento. Os dados intermediários também possuem grande importância, pois podem indicar erros experimentais ou mesmo erros na execução total ou parcial de um *workflow*.

O processo de definição de um *workflow* dependente do domínio de aplicação, requerendo amplas discussões entre os membros do grupo de pesquisa para especificá-lo. A modelagem de um *workflow* científico geralmente envolve tomada de decisão e análises refinadas sobre cada uma das etapas.

Na literatura é possível encontrar diversas implementações de *workflows* científicos. [86] descreve alguns *worklfows* de bioinformática como o MHOLline [87], para a modelagem estrutural de proteínas. Um outro exemplo é o GARSA [88] tem o objetivo de gerar um conjunto de seqüências anotadas, ou seja, identificação de uma lista de segmentos de seqüência com algum significado biológico como, por exemplo, a identificação de genes e de suas funções no genoma.

3.2 – Sistemas de Gerência de *Workflows* Científicos

As atividades de modelagem e execução de *workflows* são apoiadas por ferramentas que fornecem funcionalidades essenciais para a definição, execução e, em alguns casos, monitoramento. Essas ferramentas são conhecidas como Sistemas de Gerenciamento de *Workflows* (WfMS - *Workflow Management Systems*) e várias empresas de TI disponibilizam softwares de gerenciamento para *workflows* comerciais [89, 90 e 91].

Os Sistemas de Gerenciamento de *Workflows* Científicos (SGWfCs) são mais recentes e visam apoiar a descrição de *workflows* voltados para as aplicações científicas de quaisquer áreas de pesquisa, como biologia, física, química, geologia entre outras, que tenham necessidade de modelar computacionalmente seus experimentos. Os SGWfCs são responsáveis por invocar as aplicações, locais ou externas, que participam de um *workflow* científico, coordenando a transferência de dados entre elas.

Para atender às necessidades científicas, segundo [86], um SGWfC deve apresentar as seguintes características:

1. **Processos, Dados e Recursos** – deve contemplar processos, dados e recursos comumente usados e permitir a extensibilidade, visando acomodar novos processos, dados e recursos.

2. **Definição** - deve auxiliar os cientistas na definição e redefinição do *workflow*. A redefinição é um passo importante quando os resultados finais não forem considerados úteis ou interessantes pelos pesquisadores.

3. **Validação** - deve oferecer mecanismos para validação de *workflows*. Durante a validação, o sistema deve verificar se as entradas e saídas definidas pelos usuários para cada processo do *workflow* são consistentes, além de incluir, caso seja necessário, processos que possam converter os formatos dos dados e processos que verifiquem se os resultados gerados pelos programas são esperados ou não.

4. **Otimização, Monitoramento e Execução** - deve ser capaz de otimizar e executar o *workflow* definido pelo pesquisador. A execução do *workflow* pode ser monitorada e deve permitir a intervenção do pesquisador em qualquer ponto, o que é necessário se o pesquisador quiser avaliar os resultados intermediários para decidir se continua, ou não, a execução, ou para fazer alguma modificação na definição das próximas atividades do *workflow*.

5. **Agendamento** - oferecer agendamento da execução do *workflow*, permitindo que um *workflow* seja executado uma única vez em um determinado dia ou de tempos em tempos. Por exemplo, semanalmente uma equipe de pesquisa deseja verificar se existem novas seqüências nos bancos de dados públicos, caso afirmativo, novos alinhamentos serão produzidos.

6. **Dados e Metadados** - armazenar tanto os dados produzidos pelo *workflow* quanto os metadados, ou seja, uma informação sobre o dado que permite o acesso e gerenciamento deste dado de maneira eficiente e inteligente [92]. O sistema deve ser capaz de gerar estes metadados automaticamente e, sempre que possível, oferecer subsídios para que o cientista consulte-os e atualize-os.

3.2.1 – SGWfs Científicos x SGWfs Comerciais

Os SGWf comerciais e científicos compartilham algumas características comuns, mas também apresentam algumas diferenças. Na tabela 3.2, estão compiladas as principais diferenças a partir dos trabalhos de [83, 93, 81, 94, 95, 12, 96, 97, 85 e 98].

Tabela 3.2 – Diferenças entre SGWfs científicos e comerciais.

Caraterísticas	SGWf Científico	SGWf Comercial
Natureza dos fluxos	Foco no fluxo de dados	Foco no fluxo de Controle
Tipos dos fluxos	A maioria é lineares (*pipelines*)	A maioria é não linear (*workflows*)
Tipos de dados	Heterogêneos e muito complexos	Bem estruturados
Volume de dados manipulados	Alto	Baixo
Freqüência de mudanças nos Wfs	Elevada	Baixa
Volume de Processamento	Elevado	Médio
Permitem a intervenção humana	Sim, Wfs cuja execução é muito demorada podem ser avaliados.	Sim, raramente o processamento é menos intenso e demorado.
Validação de dados intermediários	Sim	Não possui equivalência.
Tolerâncias à falhas	Desejável	Desejável,
Wf executado de maneira rotineira	Sim	Sim
Execução parcial	Necessário, porém nem sempre suportado	Desnecessário
Modificação dinâmica	Desejável,	Não
Suporte ao reuso	Desejável, um Wf, ou parte dele, pode ser aproveitado por outros pesquisadores no mesmo experimento ou em experimentos futuros.	Não é necessário.
Suporte a proveniência e persistência de dados	Sim,	Não possui equivalência.
Suporte à anotação	Desejável, porém nem sempre suportado, cada vez mais essa característica é apreciada nos SGWf	Não possui equivalência.
Capaz de utilizar serviços Web	Necessário, porém nem sempre suportado.	Desejável, porém nem sempre suportado.
Suporte a transformação	Sim, os dados científicos são muito	Não possui equivalência.

de dados	heterogêneos.	
Suporte a ambientes distribuídos (Grids, P2P, Cluster)	Desejável, porém nem sempre oferecido pelos SGWfCs.	Não possui equivalência.

3.2.2 – Taverna

O **Taverna** é um projeto financiado pelo EPSRC (*Engineering and Physical Sciences Research Council*), envolvendo cinco universidades do Reino Unido, o grupo *European Bioinformatics* e alguns colaboradores da indústria privada. Este projeto mantém seu foco nos requisitos de gerência de *workflows* para experimentos de bioinformática, mas seu uso por outros domínios científicos não é impedido [99].

Este software pode ser utilizado em plataformas Windows ou Linux, foi desenvolvido em Java e *scripts shell* e seu código é aberto e orientado a serviços. A partir de uma interface gráfica de usuário, o *workflow* é modelado pelo cientista e traduzido para uma linguagem de definição de *workflows* do Taverna, chamada **Scufl** (*Simple Conceptual Unified Flow Language*) [100], baseada em WSFL (*Web Services Flow Language*) [102]. Em sua arquitetura possui uma máquina de *workflow*, chamada **FreeFluo** [101], desenvolvida pela equipe do Taverna e utilizada para gerenciar, de modo desacoplado da interface, a execução dos *workflows*.

Como é voltado para a bioinformática, provê uma série de recursos para facilitar a definição de *workflows* desta área, dentre eles: serviços pré-definidos, estruturas de dados específicas deste domínio, suporte semântico através de ontologias e serviços de proveniência dos dados.

3.2.3 – Kepler

O **Kepler** é um SGWfC que não foi projetado para contemplar apenas uma área de conhecimento [103]. Sua máquina de execução de *workflows* é baseada na máquina de execução do antigo projeto Ptolomy II, com código aberto e desenvolvido em Java com orientação a objetos. Seu modelo de classes é capaz de representar e operar sobre inúmeros tipos de dados heterogêneos e invocar serviços web.

Para permitir a modelagem visual de *workflows*, o Kepler possui uma interface gráfica, denominada **Vergil**, atendendo a grande parte dos requisitos propostos por [104]. Esta interface é dividida em três áreas principais: menu de comandos e a barra de ferramentas, que permitem que o usuário execute, pare ou interrompa a execução de um *workflow*, ou ainda altere a definição do mesmo; e a paleta de componentes, onde estão dispostos os objetos que podem ser acoplados no *workflow* que está sendo desenvolvido.

O Kepler possui arquitetura modular que permite a implementação de um número qualquer de modelos de computação e é composta por diversos pacotes que representam seus objetos disponibilizados para definição de *workflows* e modelos computacionais de execução. Sua arquitetura também foi projetada de modo a ser facilmente extensível.

3.2.4 – Vistrails

O **VisTrails** [105] é um SGWfC desenvolvido na Universidade de Utah cujo principal objetivo é prover suporte a exploração de dados e visualização. Seu principal avanço em relação aos outros SGWfCs é a parte gráfica que permite a visualização dos resultados de um *workflow* executado. É um software disponível para diversas plataformas como Windows XP, Mac OS X e Linux.

Através da interface gráfica, chamada de **Vistrails *Builder***, os cientistas podem projetar *workflows*. Esta camada gráfica possui três áreas principais: uma lista de módulos, ou objetos que podem compor os *workflows*; a área de design, onde o *workflow* é desenhado e uma área especifica para exibir as propriedades dos módulos que forem adicionados ao *workflow*.

O VisTrails possui uma arquitetura que provê uma infra-estrutura que pode ser utilizada com qualquer sistema de visualização existente, possuindo mecanismos que permitem a visualização múltipla de resultados de *workflows* executados.

3.3 – Proveniência de Dados em *Workflows* Científicos

Um enorme volume de dados e metadados pode ser gerado por *workflows* científicos. Estes metadados fornecem semântica aos *workflows* e facilitam sua reutilização. Um dos tipos de metadados é a **proveniência dos dados**, também chamada de **linhagem** ou **pedigree**.

O termo proveniência de dados, ou pedigree [107 e 108], tem sido aplicado para referenciar as origens de consultas e resultados de dados processados, enquanto linhagem denota o histórico de um dado ao longo de um processamento. Outros termos utilizados na literatura são: história derivada [109], dependência de conjunto de dados [110], filiação [111], genealogia de dados [112], rota de auditoria [113], entre outros.

No contexto científico, qualidade de dados está intimamente associada à proveniência de dados [106], pois a confiabilidade do resultado gerado pela pesquisa depende da qualidade das informações utilizadas e sua origem. Por isso, as informações de proveniência adicionam um valor significativo para projetos científicos com grande fluxo de dados.

Existem diferentes formas de proveniência de dados de *workflows* científicos. Estes metadados são utilizados para os mais diversos propósitos como, por exemplo, para representar a proveniência dos dados e resultados de um experimento, as publicações são comumente utilizadas. Abaixo são listadas algumas funcionalidades para os dados de proveniência, segundo [114]:

1. **Qualidade dos Dados**: informações de proveniência podem ser utilizadas para estimar a qualidade e a confiabilidade dos dados baseando-se na origem dos dados e suas transformações [115].

2. **Auditoria dos Caminhos**: os dados de proveniência podem traçar rotas dos dados [118], determinar a utilização de recursos [28] e detectar erros na geração de dados [117].

3. **Controle de replicação**: informações de proveniência detalhadas permitem a derivação de dados e ajudam a padronizar a replicação [116].

4. **Atribuição**: mantém controle sobre as informações do dono do experimento e seus dados. Também permite a citação [115] e atribui responsabilidades em caso de dados errados.

5. **Informacional**: permite realizar consultas baseadas nos metadados de origem para a descoberta de dados, além de prover o contexto necessário para interpretar os dados.

3.3.1 – Proveniência de Dados nos SGWfCs

Como foi apresentado nas seções anteriores, um *worflow* científico representa um experimento *in silico* e as informações de proveniência são essenciais para atestar resultados de experimentos no domínio científico. Por isso, é fundamental que os SGWfCs ofereçam recursos para armazenar, gerenciar e consultar esse tipo de informação relacionada aos *workflows* executados por ele. Neste momento serão apresentados os mecanismos de gerência de dados de proveniência dos principais SGWfCs mencionados nesta dissertação: Taverna, Kepler e Vistrails.

Para armazenar proveniência dos dados, um *plugin* pode ser instalado no Taverna. Este *plugin* se chama **Taverna LogBook** [119] e armazena os dados durante o progresso da execução de cada *workflow*. Ele permite a visualização de dados relacionados à execução do *workflow* e dados gerados pelo mesmo, tanto finais quanto intermediários.

O LogBook possui duas bases de dados relacionais, uma para dados e outra para metadados. Utiliza o MySQL [120], um banco de dados livre com suporte para diversas plataformas, para manter esses repositórios. O nível de detalhe para armazenamento dos dados é configurável de modo a permitir que o usuário opte por armazenar ou não os dados intermediários de um *workflow*. O armazenamento na base de dados é realizado por um serviço web que faz a conexão entre a máquina de execução e o banco de dados [121].

No Kepler, os componentes que oferecem suporte à proveniência são o ***Provenance Framework*** e o ***Smart Re-run*** que foram desenvolvidos de forma a oferecer suporte aos projetos multidisciplinares do Kepler [122]. O primeiro componente foi desenvolvido com o objetivo de registrar as informações relacionadas ao contexto de execução de um *workflow*, isto é, ele é capaz de registrar os dados de entrada e saída de um *workflow*, seus metadados, os dados intermediários, além das definições do *workflow* e algumas informações relacionadas à sua execução. O segundo componente

permite que uma determinada instância de *workflow* seja executada a partir de um determinado ponto.

O VisTrails foi o primeiro SGWfC a armazenar a evolução de *workflows*, mantendo o foco na proveniência da etapa de definição de *workflows*. Recentemente, foi desenvolvido um *plugin* [123] para armazenar os dados de execução de *workflows* em uma base de dados local, sendo arquivos XML a solução padrão.

A partir do armazenamento da proveniência no Vistrails, o sistema possibilita ao usuário definir *workflows* e controlar suas alterações, ou seja, as diferentes versões do *workflow* que foi gerado e posteriormente modificado podem ser exibidas em modo visual através de árvores. Essa característica permite que o cientista analise os parâmetros de entrada de um *workflow* e conclua quais geraram o melhor resultado, por exemplo.

3.4 – Análise da Gerência de Dados em Sistemas de Gerência de *Workflows* Científicos

Para avaliar as diferentes estratégias de gerência de dados propostas por cada um dos SGWfCs citados anteriormente, analisamos não somente como os dados de proveniência de definição e execução dos *workflows* modelados em cada SGWfC são tratados por estas ferramentas, mas também consideramos os mecanismos utilizados para armazenar os dados intermediários gerados ao longo de uma execução de experimento. Neste trabalho, estes aspectos são analisados dentro do contexto da bioinformática.

Além do tratamento de formatos heterogêneos de dados, da natureza distribuída de *workflows* e da necessidade de prover um caminho integrado para visualizar dados das diferentes etapas dos *workflows* de bioinformática, é importante observar quando se deve capturar o dado ou mesmo se este deve ser armazenado ou não. Além disso, definir, dentre todas as etapas de um *workflow*, quais dados devem ser transportados para um repositório, onde deve ficar guardado e como deve ocorrer essa captura são questões que devem ser observadas dentro de um contexto científico específico para que estas informações tenham sentido para o cientista.

Inicialmente o Taverna realizava a captura e o armazenamento da proveniência através de um serviço web chamado MIR que também definia um esquema de dados relacionados ao domínio da bioinformática extremamente simples, contendo os dados de execução do *workflow* e seus dados intermediários, de entrada e saída, sem armazenar esses dados intermediários em um esquema mais apropriado para o domínio da biologia.

A versão 1.7 do Taverna conta com uma outra alternativa para a gerência dos dados de proveniência. O controle da proveniência, assim como seu nível de detalhe passou a ser opcional para seus usuários. Os *logs* são armazenados durante a execução do *workflow*. Essa nova estratégia mapeia os metadados relacionados à execução do *workflow* a uma ontologia definida pela equipe do Taverna e, em seguida, armazenados em uma base de dados. Essa ontologia agrega semântica aos metadados relacionados ao *workflow*.

Os dados intermediários gerados para cada instância de *workflow* executada no Taverna não são classificados segundo uma ontologia e são armazenados apenas como objetos e relacionamentos entre eles. Estes relacionamentos indicam apenas qual dado foi gerado por cada dado, ou seja, a rota histórica do dado ao longo da execução do experimento. Além disso, existem apenas duas opções para definir granularidade de armazenamento. O cientista pode optar por armazenar proveniência sem captura de dados intermediários ou com dados intermediários. Se o pesquisador escolher armazenar dados intermediários, isto ocorrerá para todos os passos do *workflow*, e a performance da máquina de execução é razoavelmente comprometida.

O Kepler *Provenance Framework*, mecanismo para o gerenciamento da proveniência do Kepler, possui classes que observam a execução do *workflow*, armazenando os dados gerados durante a mesma e pode ser configurável para os parâmetros de granularidade e destino do dado, por exemplo. O cientista pode optar pelo uso do recurso para armazenamento da proveniência ou não, apenas incorporando um componente, chamado *Provenance Recorder*, à definição do seu *workflow*.

A estratégia do Kepler armazena os dados intermediários e possibilita a re-execução de um *workflow* cujos parâmetros eventualmente tenham sido alterados, ou que tenham obtido falhas em alguns de seus passos, aproveitando os dados de etapas

anteriores sem a necessidade de executá-las novamente. Este recurso é chamado de *Smart Re-run*. Neste caso, o armazenamento de dados intermediários também ocorre para todas as etapas do *workflow* e o pesquisador fica impedido de escolher quais dados ele deseja armazenar e quais ele deseja dispensar.

Para o Vistrails foi desenvolvido foi desenvolvido recentemente um *plugin* para armazenar os dados de execução de *workflows* em uma base de dados. Essa solução consiste em um mecanismo simples que armazena apenas alguns metadados de execução como máquina onde o *workflow* executou e data e hora de execução, sempre relacionados à instância armazenada. Esta solução não considera os dados intermediários gerados por cada etapa do *workflow*.

Os três SGWfCs descritos acima não possuem mecanismos de gerência de dados integrados com esquemas de dados para domínios de aplicação, como bioinformática. Além disso, cada um deles armazena a informação de modo particular, não respeitando nenhum esquema padronizado de proveniência de dados para *workflows*. Seus repositórios não permitem análises mais elaboradas sobre os dados armazenados, sendo possível apenas visualizá-los de modo atômico para cada execução de *workflow*.

Também não possuem soluções estáveis. Suas abordagens sofrem constantes alterações, ainda propõem muitas melhorias para suas alternativas ou não são disponibilizadas para uso efetivo, como é o caso do Kepler *Provenance Framework*, descrito em documentações, mas ainda não disponível para uso.

Outra deficiência ainda comum para estes SGWfCs é a falta de interfaces adequadas para a construção de consultas aos dados armazenados durante a execução. O Vistrails definiu uma linguagem de consulta chamada Vistrail *Query Language* [123] para possibilitar consultas a evoluções de *workflows* e informações armazenadas durante a execução, mas ainda não provê uma interface para construção facilitada de consultas.

Apesar dos problemas mencionados acima, cada um dos SGWfCs analisados possui vantagens que apóiam o ciclo de vida de um experimento *in silico*. O Taverna exibe uma listagem com vários serviços web de bioinformática em sua interface e uma ontologia para estes serviços, o que lhes provê um suporte semântico não encontrado nos outros SGWfCs. Além disso, possui mecanismo de tolerância a falhas, importante para experimentos que podem passar longos períodos de tempo em execução.

O Kepler possui componentes para as mais diversas funcionalidades que podem ser utilizados na definição de um *workflow* para representar uma grande gama de tarefas, como acesso rápido a bases de dados, invocação de serviços web e integração com ambientes de computação distribuída em *grid*. Já o VisTrails inova em possibilitar a visualização e o controle de versões dos *workflows*, considerando a definição do *workflow* como proveniência de dados.

Com base nesta análise, é possível concluir que uma boa solução para que cientistas da área da biologia molecular possam executar consultas voltadas para o seu contexto, a partir de dados gerados por experimentos *in silico*, seria a integração de um SGWfC com um esquema de dados que modele os conceitos do domínio da bioinformática apoiado por um SGBD (Sistema de Gerência de Banco de Dados). Assim, as funcionalidades e infra-estrutura disponibilizadas pelos SGWfCs para definição e execução de experimentos serão aproveitadas e aliadas às vantagens de um esquema de representação do domínio da bioinformática.

Capítulo 4 – Arquitetura para Gerência dos Dados em *Workflows* Científicos da Bioinformática

"Saber o que todos sabem é não saber nada."
(Remy Gourmont)

Todos os SGWfCs analisados possuem mecanismos já desenvolvidos e estabilizados para coordenar fluxos de atividades, integrando diversas ferramentas computacionais. Estes mecanismos atendem ao objetivo de automatizar experimentos que envolvem a utilização de softwares e tecnologias da computação e têm sido cada vez mais utilizados por diversos centros de pesquisa.

O GUS é um esquema de representação do domínio da bioinformática que abrange tanto dados comuns na biologia molecular, como informações relacionadas aos experimentos científicos. Este esquema é adotado em diversos projetos de pesquisa, além de ser o esquema padronizado que possui a representação mais completa desta área científica.

Aproveitar a tecnologia desenvolvida para os SGWfCs e unir esta solução ao tratamento e armazenamento de dados integrado com um esquema de dados apropriado para o domínio da biologia molecular é a estratégia proposta neste trabalho para tornar possível a gerência de dados intermediários gerados em um experimento.

Este capítulo está dividido da seguinte forma: a primeira seção explica a arquitetura da solução proposta. Os serviços que compõem esta arquitetura serão detalhados na segunda seção deste capítulo. Em seguida, os elementos que foram desenvolvidos para permitir a integração com a interface de usuário do SGWfC serão apresentados. Por fim, uma breve conclusão do capítulo, resumindo os elementos da arquitetura proposta e suas vantagens.

4.1 – Arquitetura Proposta

Esta seção apresenta a arquitetura proposta para automatizar a captura automática de dados intermediários gerados em um experimento através da integração de um SGWfC com o esquema padronizado para o domínio da bioinformática, GUS.

O objetivo desta arquitetura é permitir que um cientista possa modelar seu experimento em um *workflow*, utilizando um ambiente gráfico que possibilite a representação computacional do mesmo, assim como sua configuração. Além disso, este ambiente deve permitir que o cientista execute e ajuste seu experimento diversas vezes, fazendo com que a execução da cadeia de atividades seja automatizada.

O armazenamento dos dados produzidos durante a execução do *workflow* deve ser permitido e configurado pelo cientista, de modo que o mesmo possa especificar em que etapas de seu experimento deseja que a informação gerada seja armazenada na base de dados. Alguns dados podem ser irrelevantes para sua pesquisa e, desse modo, seriam eliminados. Também compete ao cientista especificar o que é o dado que está sendo gerado. Por exemplo, se este dado é uma seqüência de nucleotídeos no formato FASTA ou no formato Genbank, ou se são dados de Micro-arranjos, etc. Desse modo, os dados são redirecionados para as entidades que os representam no esquema do GUS, respeitando as questões de persistência envolvidas.

Abaixo, na figura 4.1, estão representados graficamente os elementos envolvidos na solução proposta:

- Ambiente para Modelagem e Execução do Experimento (SGWfC);
- Módulo de Integração com o SGWfC;
- Módulo de Transformação e Persistência de Dados;
- Serviço para Consultas;
- e Repositório para Dados do Experimento (GUS).

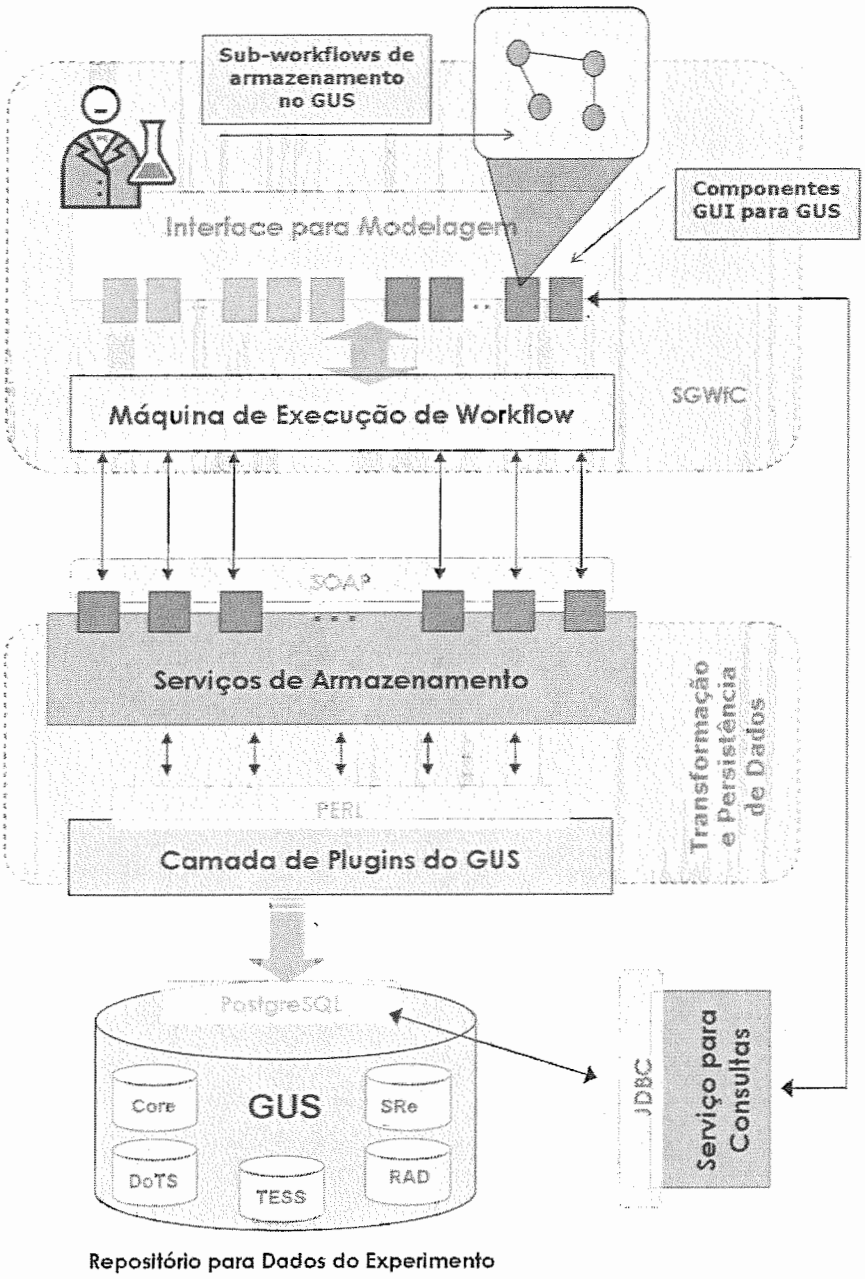


Figura 4.1. Arquitetura para Gerência de Dados Intermediários de *Workflows* da Bioinformática.

Para possibilitar a modelagem do experimento em um *workflow* através de uma interface gráfica, assim como, a execução coordenada das atividades modeladas neste *workflow*, um dos componentes essenciais da arquitetura é um SGWfC. Na estratégia apresentada nesta dissertação, foi escolhido o sistema Kepler por apresentar modelos computacionais capazes de representar qualquer experimento, apresentar uma interface gráfica intuitiva e uma documentação de uso satisfatória, além de ser facilitar a construção de componentes que possar ser acoplados à sua interface. Desse modo, permite que se adicionem componentes customizados à sua biblioteca de artefatos utilizados na representação dos experimentos.

Como repositório para os dados intermediários do experimento *in-silico* modelado no SGWfC, é utilizada uma base de dados PostgreSQL que contém os sete sub-esquemas do GUS para representação de dados biológicos e de experimentos: Core, DoTS, SRes, TEES, Prot, Study e RAD. Desse modo, qualquer experimento da biologia molecular poderá ter seus dados intermediários acomodados neste repositório em entidades que os representam, assim como seus relacionamentos.

O Módulo de Transformação e Persistência dos Dados faz uso de duas estratégias para tratar e persistir os dados no repositório: os *Plugins* do GUS (*GUS Plugin API*) e os Serviços de Armazenamento (*Storage Services*). Os Serviços de Armazenamento foram desenvolvidos para esta solução, integrando-se com a camada de *plugins* Perl do GUS, com o objetivo de armazenar dados nesta base de dados.

No momento da instalação do GUS *Schema*, a base de dados com seus sub-esquemas é montada e são instalados os *Plugins* do GUS. Esses *plugins* são implementados em Perl, utilizando os módulos do BioPerl, dentre outros, e têm o objetivo de persistir os dados no GUS. São disponibilizados diversos *plugins* para armazenar dados de acordo com seu tipo, por exemplo, existem *plugins* para inserir dados de Taxonomia, Ontologias, Seqüências, etc. Esses *plugins* realizam algum tratamento nos dados que recebem como entrada, dependendo do tipo do dado, realizando a conversão do tipo dessas informações para armazená-las como estão representadas no esquema do GUS.

Os *plugins* do GUS são comumente utilizados manualmente pelos cientistas que adotam o GUS *Schema* em seus projetos. Eles também possuem alguns problemas que

tornam sua utilização uma tarefa onerosa para os cientistas, como a necessidade de montar arquivos XML auxiliares ou alterar arquivos cujos formatos são comuns nesta área de domínio para ajustá-los aos *plugins*. Além de obrigar que os cientistas conheçam a linguagem de programação Perl.

Os Serviços de Armazenamento foram desenvolvidos como elementos do Módulo de Transformação e Persistência dos Dados do Experimento para funcionar como uma camada de nível mais alto de acesso ao repositório. Eles corrigem os problemas encontrados no uso dos *plugins*, realizam tratamento de alguns dados e ajustes em arquivos auxiliares, além de invocar os *plugins* que continuam sendo responsáveis por parte do tratamento dos dados e pela persistência dos mesmos na base de dados. A seção 4.2 explica detalhadamente o funcionamento dos Serviços de Armazenamento e suas vantagens.

Para permitir que os cientistas, ao modelar os *workflows* que representam seus experimentos, possam especificar em que etapas do *workflow* os dados devem ser armazenados no GUS, foi criado para esta dissertação o Módulo de Integração com o SGWfC que, no caso do Kepler, é composto por uma biblioteca de atores, ou *GUS Actors*. Estes atores serão detalhados na seção 4.3.

O cientista, ao modelar seu experimento *in-silico*, identificará quais tarefas de seu *workflow* devem ter seus dados armazenados, além disso, ele também deve escolher qual ator utilizar, dependendo do tipo do dado envolvido. Desse modo, é possível alcançar a modelagem do experimento e sua execução automatizada, integrando as ferramentas computacionais envolvidas e realizando o armazenamento dos dados intermediários em um repositório que os represente na biologia molecular.

Para permitir consultas às seqüências armazenadas no repositório, foi desenvolvido o Serviço de Consulta (*Query Service*), que se conecta ao GUS e dependendo de um de seus argumentos de entrada, retorna seqüências de nucleotídeos ou aminoácidos. Os resultados são armazenados em um arquivo, cujo caminho e nome também são passados em sua linha de comando.

4.2 – GUS Web Services

Os GUS *Web Services* foram desenvolvidos para resolver um conjunto de problemas envolvidos na utilização dos *Plugins* do GUS. Eles também funcionam como uma camada de nível mais alto que pode ser acessada por outras aplicações, permitindo que estes serviços representem objetos reutilizáveis.

Estes serviços foram implementados como serviços web para tornar seu uso independente de plataforma tecnológica, permitir acesso remoto e ser compatível com todos os SGWfCs analisados neste trabalho, tendo em vista que todos apresentam soluções para a utilização de serviços web em seus *workflows*. Desse modo, esses serviços podem ser utilizados por qualquer SGWfC, em qualquer experimento, sendo necessário apenas a passagem de parâmetros de entrada para os mesmos. O cientista não precisa executar linhas de comando manualmente para cada dado que desejar incluir na base de dados.

Devido à grande complexidade do GUS, pois o mesmo representa dados de diversas áreas da bioinformática, e considerando que para implementar os serviços de tratamento e armazenamento de dados no GUS é necessário compreender o significado de cada objeto de seu esquema, para este trabalho foram desenvolvidos serviços web apenas para os sub-esquemas **DoTS** (seqüências e anotação genômica), **SRes** (*Shared Resources*, ou seja, recursos compartilhados por todos os esquemas como Ontologias do domínio da bioinformática, Taxonomias, etc.) e **Core** (Administração, documentação e dados de proveniência).

Algumas informações de cada execução dos serviços web desenvolvidos para este trabalho é registrada no objeto **AlgorithmInvocation** do esquema Core. Esses dados contemplam data e hora iniciais e finais, resultado, tempo de CPU, etc.

Todos os serviços web foram implementados utilizando-se as tecnologias J2EE e axis2. Abaixo estão listados os atores desenvolvidos para o tratamento e armazenamento de dados no GUS e nas subseções que se seguem, cada um deles será apresentado detalhadamente. São eles:

- **InsertGusExternalDBWS:** insere no GUS um registro para um banco externo, Genbank, por exemplo;

- **InsertGusExternalDBRlsWS:** insere no GUS, a versão do banco externo. Por exemplo, uma determinada seqüência pode apresentar anotações diferentes em versões distintas do Genbank. Para possibilitar que o GUS mantenha essa informação de proveniência (qual a versão do banco externo) para as seqüências armazenadas;

- **InsertSequenceFeaturesWS:** insere seqüências e suas anotações a partir de arquivos extraídos do Genbank, no formato Genbank;

- **LoadFastaSequencesWS:** insere seqüências genômicas que se encontram em arquivos no formato FASTA, bem mais simples do que o formato Genbank, sem a maioria das anotações;

- **InsertBlastSimilarityWS:** insere os relacionamentos entre as seqüências que apresentam similaridade identificada pelo aplicativo BLAST, assim como as informações de similaridade;

- **LoadTaxonWS:** insere um conjunto de taxonomias para diversos organismos;

- e **InsertSequenceOntologyWS:** insere um conjunto de ontologias da *Sequence Ontology*.

4.2.1 – InsertGusExternalDBWS

Este serviço web é o mais simples dentre todos os GUS *Web Services*. Seu objetivo é apenas inserir nomes de bancos de dados externos ao GUS, ou seja, bancos de dados de origem das informações que estão sendo armazenadas no GUS. Esta informação no GUS se encontra na tabela **ExternalDatabase** do esquema **SRes**.

Este serviço possui apenas uma operação, chamada **InsertGusExternalDB**, que recebe como parâmetro de entrada, uma *string* contendo o **nome do novo banco** externo e retorna como parâmetro de saída, uma *string* contendo uma mensagem de erro ou a saída da execução do *plugin* do GUS relacionado (*InsertExternalDatabase.pm*). Este serviço também executa uma validação do nome do banco, retornando uma mensagem significativa no caso de não ser uma *string* válida. O código deste serviço se

encontra no anexo A e o arquivo WSDL de publicação deste serviço web se encontra no anexo B.

O InsertGusExternalDBWS pode ser utilizado tanto para inserir apenas um registro, quanto para realizar uma carga de bancos externos. Para inserir apenas um registro, basta utilizá-lo no *workflow* modelado, invocando o WS apenas uma vez. Para inserir vários bancos externos, modela-se um *loop* no *workflow*, passando em cada interação o nome de um dos bancos externos.

4.2.2 – InsertGusExternalDBRLSWS

Este serviço web insere versões de bancos externos já cadastrados no GUS, armazenando um registro na tabela **ExternalDatabaseRelease** do esquema **SRes** e seu apontamento para a tabela **ExternalDatabase** do mesmo esquema.

Sua única operação se chama **InsertGusExternalDBRls**, que recebe como parâmetros de entrada, uma *string* contendo o **nome do novo banco** externo e uma string representando a **versão do banco**. Este serviço retorna como parâmetro de saída, uma *string* contendo uma mensagem de erro ou a saída da execução do *plugin* do GUS relacionado (*InsertExternalDatabaseRls.pm*). Este serviço também executa uma validação do nome do banco e da versão do banco, retornando uma mensagem significativa no caso de não serem *strings* válidas. O código deste serviço se encontra no anexo A e o arquivo WSDL de publicação deste serviço web se encontra no anexo B.

Do mesmo modo que o InsertGusExternalDBWS, este serviço pode ser utilizado tanto para inserir apenas um registro, quanto para realizar uma carga de versões de bancos externos. Este serviço só executará as inserções caso os bancos já tenham sido inseridos, por isso, deve-se utilizar o InsertGusExternalDBWS para armazenar os bancos externos antes de suas versões.

4.2.3 – InsertSequenceFeaturesWS

Este serviço web insere seqüências genômicas contidas em arquivos texto no formato Genbank, assim como as anotações associadas às seqüências, chamadas de *Features* nos arquivos Genbank. As *Features* possuem atributos, chamados *Qualifiers*, também armazenados por este serviço.

Através do *plugin* relacionado à inserção de seqüências e suas *features* no GUS (*InsertSequenceFeatures.pm*), dependendo das anotações contidas no arquivo de entrada, este serviço pode incluir registros nas tabelas **TaxonName** e **SequenceOntology** do esquema **SRes**, nas tabelas **SequenceType**, **NASequence**, **ExternalNASequence**, **VirtualSequence**, **Assembly**, **SplicedNASequence**, **SecondaryAccs**, **NALocation**, **NASequenceRef**, **Keyword**, **NASequenceKeyword**, **NAComment** e **ExonFeature** do sub-esquema **DoTS**. Nestes registros também inclui os apontamentos para as tabelas **ExternalDatabase**, **ExternalDatabaseRelease** e **Reference** do esquema **SRes**. É possível perceber, mesmo através dos nomes dos objetos do GUS, que dados de proveniência são armazenados em relações que os representam, assim como os dados intermediários em si (a cadeia de nucleotídeos).

A operação **InsertSequenceFeatures** publicada com o serviço web possui os seguintes parâmetros de entrada, todos do tipo *string*:

- **fileGb:** string com o conteúdo do arquivo Genbank contendo apenas uma seqüência.
- **databaseName:** nome do banco externo de origem da seqüência.
- **databaseRls:** versão do banco externo de origem da seqüência.
- **soCvsVersion:** versão da *Sequence Ontology* previamente carregada no GUS, através do serviço *InsertSequenceOntologyWS*, que será explicado mais à frente.

Os dados acima são necessários para que o *plugin* do GUS possa montar o relacionamento entre os registros inseridos. O serviço executa validações para os parâmetros de entrada e retorna uma *string* com a mensagem de erro ou com a saída do *plugin*.

Este serviço web recebe o conteúdo do arquivo no formato Genbank e executa um tratamento no mesmo. Isto é necessário porque um *Qualifier* pode possuir mais de um valor no arquivo do Genbank e o *plugin* não consegue resolver esta situação, pois, no GUS, o *Qualifier* é um atributo de uma entidade, ou seja, não é possível inserir vários relacionamentos para *Qualifiers*. Para resolver este problema, o serviço percorre

o conteúdo do arquivo em busca de *Qualifiers* com vários valores, e concatena estes valores, separando-os pelo caractere ";". Após este tratamento do arquivo, o serviço salva este arquivo em um diretório temporário para passá-lo como argumento de entrada para o *plugin* do GUS.

Além do tratamento realizado no conteúdo do arquivo do Genbank, este serviço também fornece para o *plugin* do GUS um arquivo XML que possui todos os mapeamentos de todas as possíveis *Features* do Genbank, para as suas entidades correspondentes no GUS. Este arquivo também indica quais *Qualifiers* e quais *Features* serão armazenadas para cada arquivo do Genbank.

O grande problema envolvido com este arquivo de mapeamento é que diferentes seqüências em arquivos do Genbank possuem diferentes *Features* e, mesmo que possuam *Features* em comum, seus *Qualifiers* podem divergir. O *plugin* faz a leitura do arquivo de mapeamento e, se o mesmo indicar a inclusão de um *Qualifier* que não está no arquivo da seqüência, o armazenamento não ocorre.

Para resolver este problema, o serviço acessa um arquivo de mapeamento padronizado, produzido durante a pesquisa desta dissertação, chamado de **genbank2gus.XML** e instalado no diretório local **/usr/local/GUSfmw/** da máquina em que o serviço está instalado.

Para montar o mapeamento padronizado, foram incluídas *tags* XML para as *Features* e *Qualifiers* que não existiam no arquivo instalado com o GUS. Foi realizado um levantamento de todas as possíveis *Features* e seus *Qualifiers* de um arquivo do Genbank e um levantamento de *Features* e seus *Qualifiers* contemplados no arquivo padronizado disponibilizado na instalação do GUS. Assim, foram identificadas as *Features* e os *Qualifiers* que deveriam ser incluídos no arquivo de mapeamento padronizado. Junto à equipe de biólogos do BioWebDB, foram identificadas quais entidades do GUS deveriam acomodar as *Features* que foram incluídas no arquivo de mapeamento padronizado.

Um dos atributos das *tags* de elementos XML que representam *Qualifiers* desse arquivo de mapeamento é o ***ignore***, que pode ter o valor *true* ou *false*, indicando que o Qualifier não será armazenado, ou será armazenado, respectivamente. Para facilitar a posterior customização deste arquivo por parte do serviço web, todos os *Qualifiers*

possuem o valor *true* para o atributo *ignore*. Indicando que, por padrão, não será armazenado.

Este arquivo XML de mapeamento padronizado é apresentado no anexo C, assim como as listagens de *Features* e *Qualifiers* possíveis em um arquivo Genbank e as que são encontradas no arquivo de mapeamento disponibilizado pelo GUS.

O serviço percorre todas as *Features* e *Qualifiers* do arquivo do Genbank, montando um novo arquivo de mapeamento com base no arquivo padronizado. Ele retira as *Features* que não foram encontradas no arquivo e altera o valor do atributo *ignore* para *false* dos *Qualifiers* que foram encontrados para as *Features*. Em seguida, este arquivo de mapeamento, também é salvo em um diretório temporário e passado como argumento para o *plugin* do GUS.

O *plugin* do GUS, responsável pela inserção de seqüências e suas *Features* pode receber como argumento um arquivo do Genbank com várias seqüências. Este serviço web não recebe um arquivo do Genbank com várias seqüências para evitar a tramitação de *strings* muito grandes, ou mesmo o corte do conteúdo do arquivo. Isto ocorre porque, para manter compatibilidade entre as mais diversas tecnologias, o uso de arquivos como parâmetro de entrada para operações de serviços web não é aconselhável. Além disso, evita-se perda de informações ao minimizar o tempo de transferência de dados para o serviço, tendo em vista que a comunicação entre uma aplicação cliente e o serviço web pode ser interrompida.

Para inserir várias seqüências através do serviço é necessário modelar *loops* no *workflow*. Além disso, deve-se observar que é necessário invocar os serviços para inclusão de banco externo, versão do banco e ontologias, antes deste serviço ser invocado.

4.2.4 – LoadFastaSequencesWS

Este serviço web insere ou atualiza seqüências genômicas contidas em um arquivo texto no formato FASTA. Dependendo do tipo da seqüência contida no arquivo, o serviço armazenará a seqüência em relações distintas, por isso um dos parâmetros de entrada da operação **LoadFastaSequences** é a string **tableName**, contendo o nome da tabela na qual o dado será inserido. Essa tabela pode ser a **ExternalNASequence** (para

seqüências de nucleotídeos), a **ExternalAASequence** ou a **TranslatedAASequence** (para seqüências de aminoácidos), todas do sub-esquema **DoTS**.

Outro parâmetro de entrada para a operação do serviço web é **regexSouceId**, uma string cujo valor deve conter uma expressão regular que torne possível identificar o ID (identificador) da seqüência no banco de dados de origem. Isso ocorre porque vários bancos de dados genômicos (Genbank, RefSeq, EMBL, etc.), disponibilizam suas seqüências no formato FASTA para *download.*

Além dos parâmetros já citados, também são necessários como parâmetros de entrada o **ExtDBName**, *string* com o nome do banco externo de origem, o **ExtDBRls**, com a versão do banco externo de origem, e o **fastaFileStr**, string com o nome e localização do arquivo FASTA.

O serviço web salva um arquivo localmente com o conteúdo do arquivo passado como parâmetro de entrada e invoca o *plugin* relacionado (*LoadFastaSequences.pm*), passando o restante dos parâmetros como argumentos de entrada para o *plugin*. Além disso, são realizadas validações para os parâmetros de entrada e como resultado, retorna uma string contendo uma mensagem de erro ou a saída do *plugin*. O código do serviço web se encontra no anexo A e seu arquivo de publicação WSDL, no anexo B.

4.2.5 – InsertBlastSimilarityWS

Este serviço web insere resultados da execução do aplicativo BLAST nos objetos que os representam no GUS, assim como os relacionamentos com as seqüências envolvidas. Para possibilitar a inserção dos relacionamentos é necessário identificar qual atributo das relações do GUS é o identificador das seqüências (**na_sequence_id** e **aa_sequence_id**, como identificadores atribuídos pelo GUS e **souce_id**, identificador no banco de origem).

Para invocar a operação **InsertBlastSimilarity** deste serviço web, é necessário passar o conteúdo do arquivo resultante da execução do BLAST como uma string para o parâmetro de entrada **condensedFileStr**. O serviço armazenará o resultado no objeto **Similarity** e **SimilaritySpan** do sub-esquema **DoTS**.

Outros parâmetros de entrada são necessários para indicar informações do objeto do GUS onde se encontram armazenadas as seqüências contra as quais serão efetuadas

as buscas por similaridade. Esses parâmetros são: **subjectTable**, tabela que armazena as seqüências, **subjectTableSrcIdCol**, nome do atributo identificador da seqüência, **subjectExtDBName**, nome do banco externo de origem das seqüências e **subjectExtDBRls**, versão do banco externo de origem das seqüências. Todos recebem cadeias de caracteres (*strings*).

Também são necessários parâmetros para indicar informações do objeto do GUS que armazena as seqüências que serão comparadas (ou seja, são o objeto da busca por similaridades). São eles: **queryTable**, tabela que armazena as seqüências, **queryTableSrcIdCol**, nome do atributo identificador da seqüência, **queryExtDBName**, nome do banco externo de origem das seqüências e **queryExtDBRls**, versão do banco externo de origem das seqüências.

O serviço salva o conteúdo do arquivo de similaridades passado como parâmetro em um arquivo local e invoca o *plugin* relacionado (*InsertBlastSimilarities.pm*) passando o arquivo gerado como argumento para o *plugin*, assim como os demais parâmetros de entrada do serviço. Este serviço retorna como parâmetro de saída, uma *string* contendo uma mensagem de erro ou a saída da execução do *plugin*.

Este serviço também executa uma validação dos parâmetros de entrada, retornando uma mensagem significativa no caso de não serem *strings* válidas, caso contrário, retorna a saída gerada pela execução do *plugin* relacionado. O código deste serviço se encontra no anexo A e o arquivo WSDL de publicação deste serviço web se encontra no anexo B.

4.2.6 – LoadTaxonWS

Este serviço web insere dados relacionados à taxonomia dos organismos no GUS, armazenando registros nas tabelas **Taxon**, **GeneticCode** e **TaxonName** do esquema **SRes** e seus relacionamentos. Esse armazenamento é realizado a partir dos arquivos **nodes.dmp**, **names.dmp**, **gencode.dmp** e **merged.dmp**, que são disponibilizados para *download* no NCBI.

Sua única operação se chama **LoadTaxon**, que recebe como parâmetros de entrada, *strings* com os conteúdos dos arquivos mencionados acima. São eles

nodesFileStr, **namesFileStr**, **gencodeFileStr** e **mergedFileStr**, para os arquivos *nodes.dmp*, *names.dmp*, *gencode.dmp* e *merged.dmp*, respectivamente.

O serviço web executa uma validação para os parâmetros de entrada e, como parâmetro de saída, retorna uma *string* contendo uma mensagem de erro ou a saída da execução do *plugin* relacionado (*LoadTaxon.pm*). Os conteúdos passados como parâmetros de entrada são salvos em arquivos locais e passados como argumentos para o *plugin* do GUS. O código deste serviço se encontra no anexo A e o arquivo WSDL de publicação deste serviço web se encontra no anexo B.

4.2.7 – InsertSequenceOntologyWS

Este serviço web insere ontologias da OBO no objeto **SequenceOntology** do esquema **SRes**. Ele recebe como parâmetros de entrada: **soFileStr**, *string* que representa o conteúdo do arquivo que contém as ontologias e cujo *download* pode ser efetuado a partir da página da OBO, **soVersion**, versão das ontologias da OBO e **soCvsVersion**, string que representa a versão das ontologias no controlador de versões da OBO.

Sua única operação se chama **InsertSequenceOntology**, salva o conteúdo do arquivo localmente e executa o *plugin* relacionado (*InsertSequenceOntologyOBO.pm*), passando o arquivo e os demais parâmetros de entrada como argumentos para o *plugin*.

Este serviço retorna como parâmetro de saída, uma *string* contendo uma mensagem de erro ou a saída da execução do *plugin* do GUS relacionado (*InsertExternalDatabaseRls.pm*). Este serviço também executa validações para cada um de seus parâmetros de entrada, retornando uma mensagem significativa no caso de não serem *strings* válidas. O código deste serviço se encontra no anexo A e o arquivo WSDL de publicação deste serviço web se encontra no anexo B.

4.3 – ConectaGUS

Para permitir consultas ao repositório, foi desenvolvido um aplicativo em Java que utiliza a tecnologia JDBC para conectar-se ao GUS e executar uma consulta por seqüências de nucleotídeos ou aminoácidos que foram depositadas neste repositório. O código-fonte deste programa está disponível no anexo A.

É necessário informar um conjunto de parâmetros para utilizar este serviço. Estes argumentos devem ser passados ao aplicativo através de sua linha de execução na seguinte ordem: URL de conexão jdbc (jdbc:postgresql://ip:porta/nomebanco), nome de usuário do banco, senha do usuário do banco, tipo de seqüência (n para nucleotídeos ou p para proteínas), nome do banco para o qual a seqüência foi inserida no GUS, versão do banco para a qual a seqüência foi inserida no GUS e nome do arquivo de saída com os resultados da consulta.

4.4 – GUS *Actors*

Para facilitar o uso dos serviços web e permitir que o cientista possa configurar seus *workflows* de modo a permitir o armazenamento de dados intermediários no GUS é necessário prover uma camada de integração com o SGWfC. Dessa maneira, o cientista tem à mão os componentes necessários para indicar este armazenamento de dados no repositório.

Nesta dissertação, foi escolhido o SGWfC Kepler, pelos motivos explicados anteriormente. No Kepler, os Atores são as entidades que representam as tarefas do *workflow*. Na sua biblioteca de atores, o Kepler possui componentes para invocação de serviços web e atores para compor sub-*workflows*. Estes foram os dois principais atores utilizados neste trabalho para desenvolver os **GUS *Actors***.

Foi desenvolvido um ator para cada serviço web apresentado na seção anterior e seus objetivos também são os mesmos desses serviços web. Os GUS *Actors* são atores herdados a partir do ator **CompositeActor**, que é o ator que permite modelar e executar sub-*workflows* dentro de um fluxo de tarefas modelado no Kepler.

Para cada GUS *Actor* foi gerado um arquivo **".kar"** que pode ser importado no Kepler, o que disponibiliza o ator em sua biblioteca de componentes. Abaixo, na figura 4.2, é possível visualizar como os GUS *Actors* são apresentados na árvore de componentes que podem ser utilizados na modelagem de *workflows*. Estes atores são disponibilizados nas bibliotecas ***General Purpose*** e ***Workflow***.

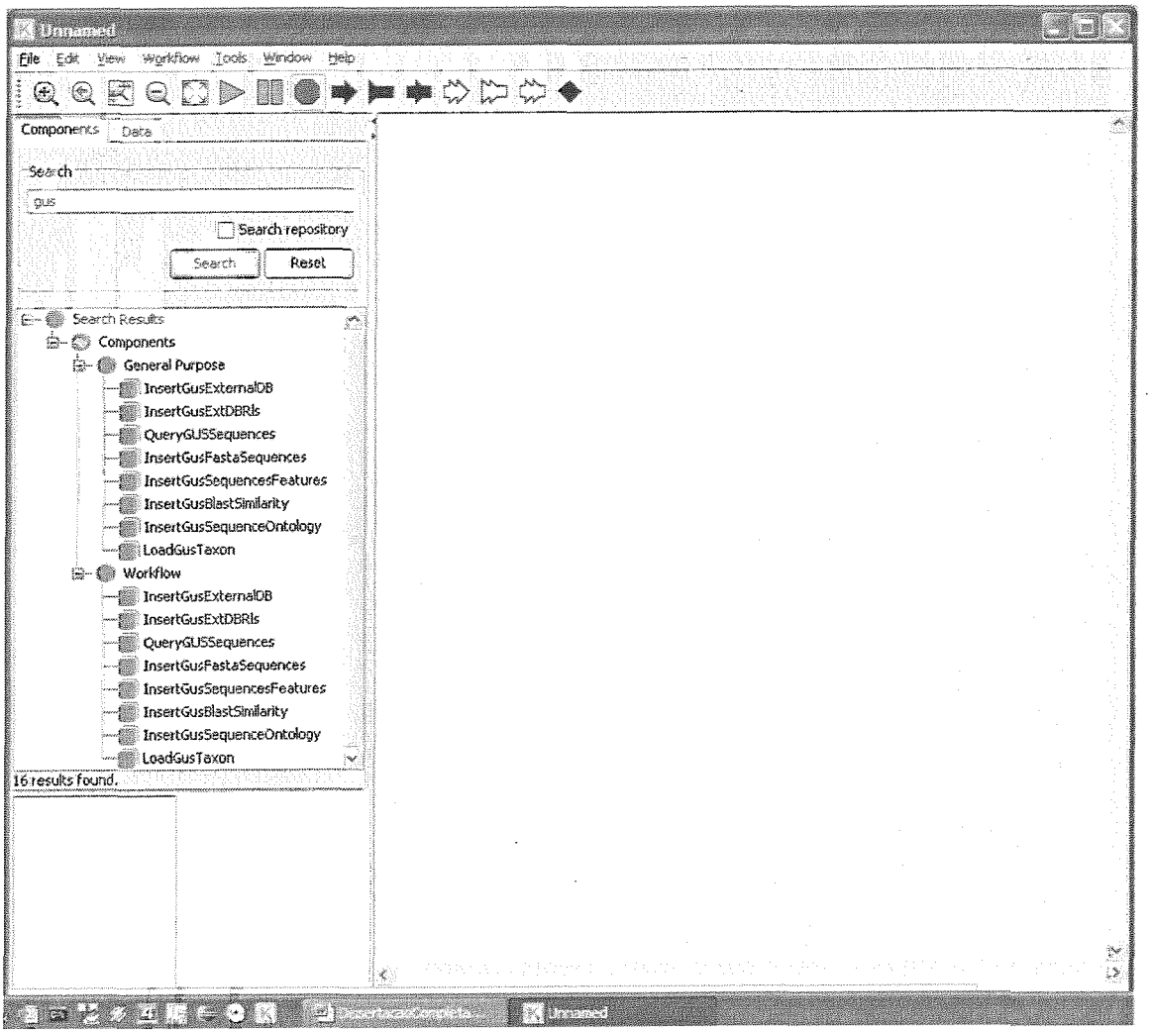


Figura 4.2. GUS *Actors* na biblioteca de componentes do Kepler.

Cada GUS *Actor* possui um sub-*workflow* associado. A seguir, será apresentado cada um destes atores e seus sub-*workflows*.

4.4.1 – InsertGusExternalDB

Este ator é responsável pelo armazenamento do nome do banco externo origem das seqüências envolvidas em um determinado experimento. Ele faz a invocação do serviço web **InsertGusExternalDBWS**, preenchendo seu parâmetro com um valor passado para o ator durante a execução do *workflow*, através de sua ***input port***, ou porta de entrada, GUSExternalDB. Na figura 4.3 é possível visualizar o sub-*workflow* deste ator.

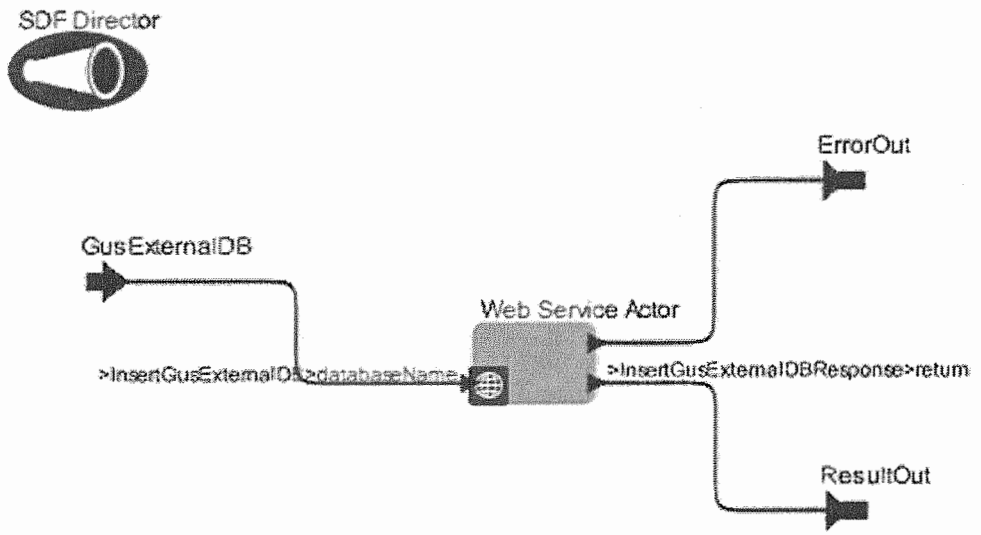


Figura 4.3. Sub-*workflow* do ator InsertGUSExternalDB.

Nas ***output ports***, ou portas de saída, **ErrorOut** e **ResultOut** do ator são repassadas as mensagens de erro e de saída da invocação do serviço web, respectivamente.

4.4.2 – InsertGusExternalDBRLS

Para armazenar a versão do novo banco inserido no GUS, foi desenvolvido este ator. Este ator invoca o serviço web **InsertGusExternalDBRlsWS**, preenchendo seus parâmetros de entrada com os valores de suas portas de entrada **ExternalDBName** e **ExternalDBRls**.

Como a inserção executada pelo serviço web só tem efeito se o nome do banco para o qual está sendo atribuída uma versão no repositório já estiver armazenado e, com o objetivo de facilitar o uso deste ator em experimentos, o sub-*workflow* modela uma etapa anterior à invocação do serviço web que insere a versão do banco externo. Nesta etapa anterior é invocado o serviço que insere o nome do banco no repositório, garantindo que, se nenhum erro ocorrer nesta inserção, o nome do banco para o qual está sendo inserida uma versão já está cadastrado. Na figura 4.4 é apresentado o sub-*workflow* correspondente a este ator.

Uma questão pode surgir: *Porque então, montar um ator apenas para inclusão de nomes de bancos externos se estes já são incluídos no ator que inclui as versões dos*

bancos? O ator que insere os nomes dos bancos pode ser utilizado em *workflows* de carga de dados para o GUS, por exemplo, para armazenar vários nomes de bancos externos, comuns aos mais diversos experimentos.

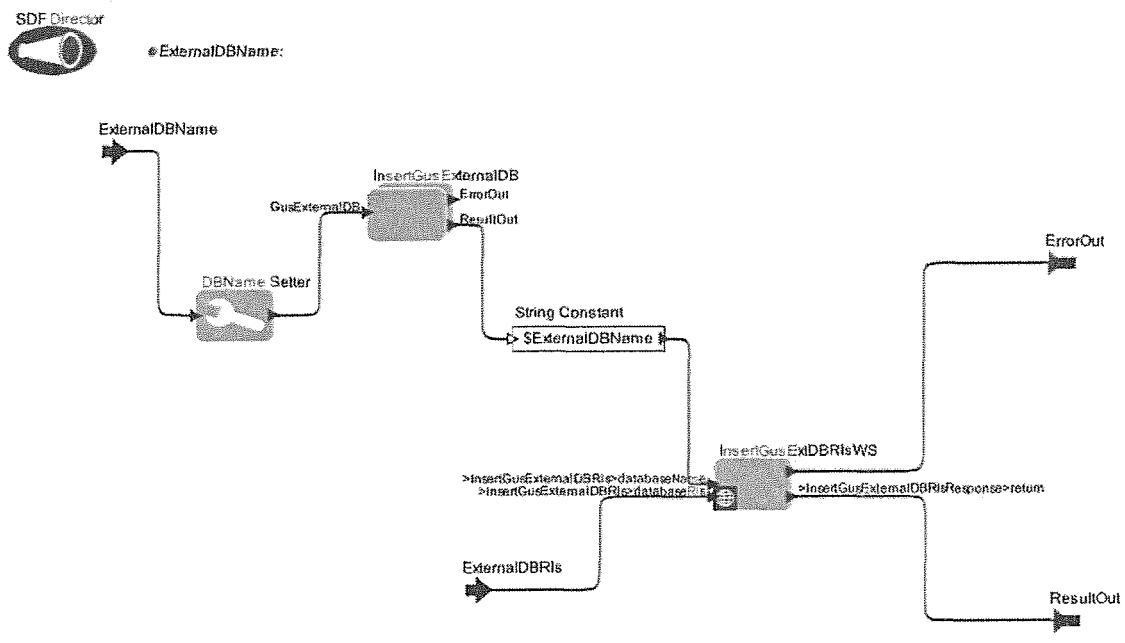


Figura 4.4. Sub-*workflow* correspondente ao ator InsertGUSExtDBRIs.

Este ator repassa as mensagens de erro e de saída da execução do serviço web para o *workflow* que o contém através de suas portas de saída, **ErrorOut** e **ResultOut**.

4.4.3 – InsertGusSequencesFeatures

O ator InserGUSExternalSequencesFeatures insere seqüências genômicas e suas *Features* e *Qualifiers* contidos em arquivos texto no formato Genbank. Ele invoca o serviço web **InsertSequenceFeaturesWS** e facilita seu uso, pois um arquivo do Genbank envolvido no experimento pode conter várias seqüências e para não sobrecarregar o serviço web com a transferência de arquivos muito grandes, o sub-*workflow* associado a este ator executa uma aplicação Java que funciona como cliente para o serviço web, percorrendo o arquivo do Genbank através de um *loop*, separando as seqüências em um conjunto de três seqüências e invocando o serviço web para cada um destes conjuntos. Este processo pode ser visualizado no sub-*workflow* associado ao ator, representado na figura 4.5.

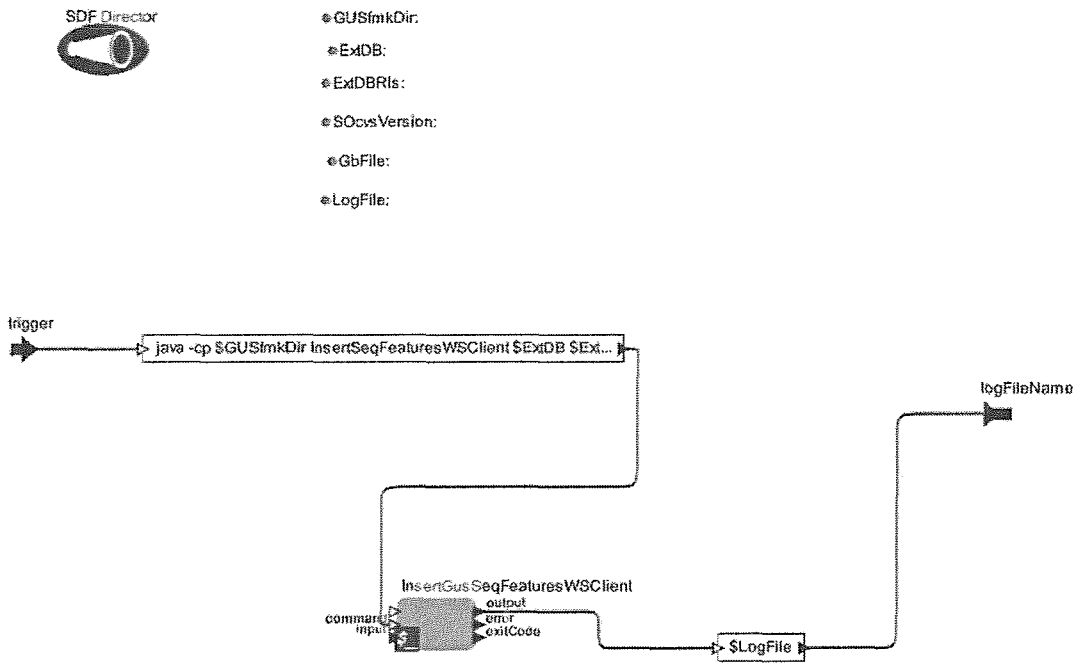


Figura 4.5. Sub-*workflow* associado ao ator InsertGUSSequenceFeatures.

Este ator repassa para o serviço web os valores recebidos através de seus parâmetros de entrada **GUSfmkDir**, **ExternalDB**, **ExternalDBRls**, **SOcvsVersion**, **GbFile** e **LogFile**. O diretório onde se encontra a aplicação que executa o *loop* nas seqüências e invoca o serviço web é indicado através do valor preenchido para o parâmetro **GUSfmkDir**. O arquivo que contém todas as seqüências é passado para o ator através de seu parâmetro de entrada **GbFile**. Os resultados do armazenamento de cada seqüência são registrados em um arquivo de *log*, cujo caminho e nome são passados pelo usuário para o ator através do parâmetro **logFile**. A porta de saída do sub-*workflow*, **logFileName** repassa o nome do arquivo de *log* para o *workflow*.

4.4.4 – InsertGusBlastSimilarity

Este ator é responsável pela inclusão de resultados de similaridade entre seqüências encontrados pelo BLAST. Para isso, ele invoca o serviço web **InsertBlastSimilarityWS**, preenchendo seus parâmetros de entrada com os valores passados para seu sub-*workflow* através de suas portas de entrada: **fileContent**,

subjectTable, **subjTableSrcIdCol**, **subjExtDBName**, **subjExtDBRls**, **qryTable**, **qryTableSourceIdCol**, **qryExtDBName** e **qryExtDBRls**. O sub-workflow associado a este ator pode ser observado na imagem 4.6.

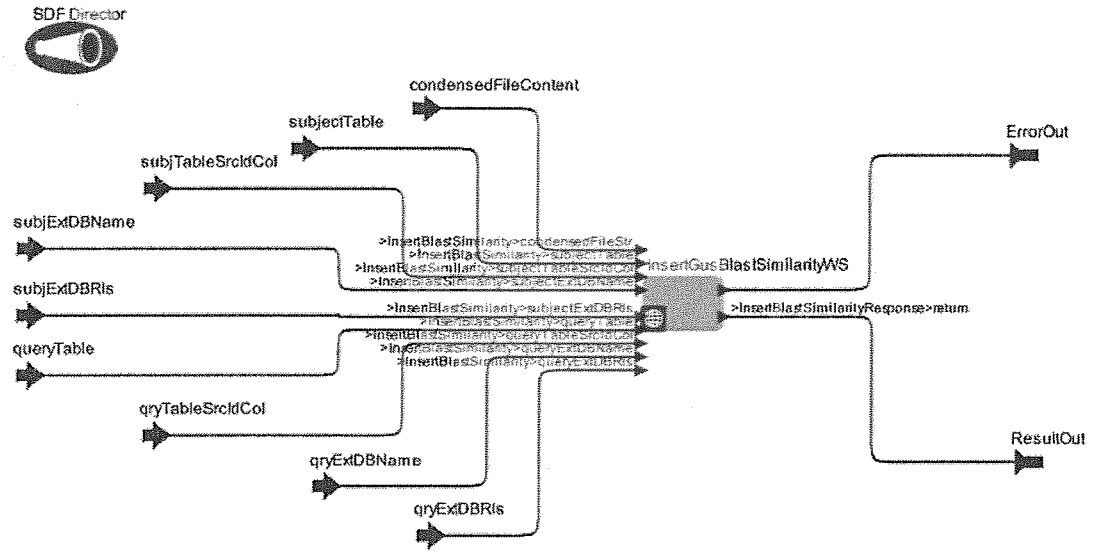


Figura 4.6. Sub-*workflow* associado ao ator InsertGusBlastSimilarity.

Este ator repassa as mensagens de erro e de saída da execução do serviço web para o *workflow* que o contém através de suas portas de saída, **ErrorOut** e **ResultOut**.

4.4.5 – InsertGusSequenceOntology

Para armazenar ontologias da OBO, o cientista pode utilizar este ator. Ele se torna útil em *workflows* que executam tarefas de carga de dados para o GUS. Estas cargas de dados são atividades periódicas em projeto de pesquisa, enquanto que *workflows* que modelam experimentos são executados com maior freqüência.

O sub-*workflow* deste ator invoca o serviço web **InsertSequenceOntologyWS**, preenchendo seus parâmetros com os valores recebidos através de suas portas de entrada **soFileContent**, **soVersion** e **soCVSVersion**. Abaixo, na figura 4.7 é possível visualizar este sub-*workflow*.

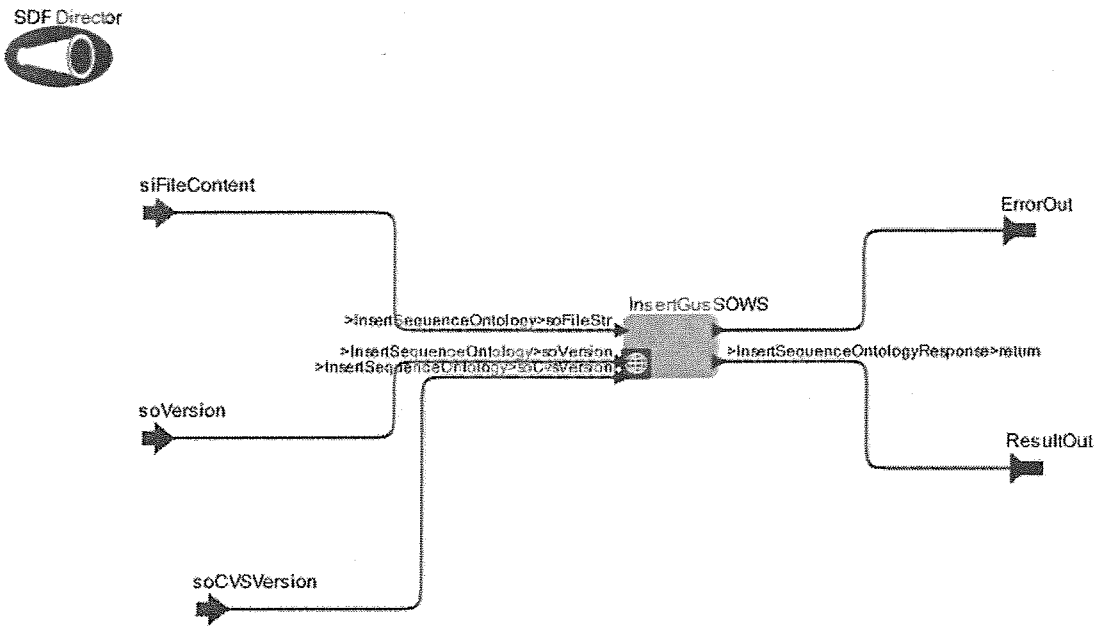


Figura 4.7. Sub-*workflow* associado ao ator InsertGusSequenceOntology.

Este ator repassa as mensagens de erro e de saída da execução do serviço web para o *workflow* que o contém através de suas portas de saída, **ErrorOut** e **ResultOut**.

4.4.6 – InsertGusFastaSequences

Este ator é responsável pela inclusão de seqüências genômicas contidas em arquivos texto no formato FASTA. Para isso, ele invoca o serviço web **LoadFastaSequencesWS**, preenchendo seus parâmetros de entrada com os valores passados para seu sub-*workflow* através de seus parâmetros de entrada: **ExternalDBName**, **ExternalDBRls**, **regexSourceId** e **tableName**. O nome do arquivo que contém as seqüências no formato FASTA, é repassado ao sub-*workflow* através de sua porta de entrada **fastaFileName**. O sub-*workflow* associado a este ator pode ser observado na imagem 4.8. Este sub-*workflow*, lê o arquivo FASTA e repassa seu conteúdo para o serviço web, que comandará o armazenamento no GUS.

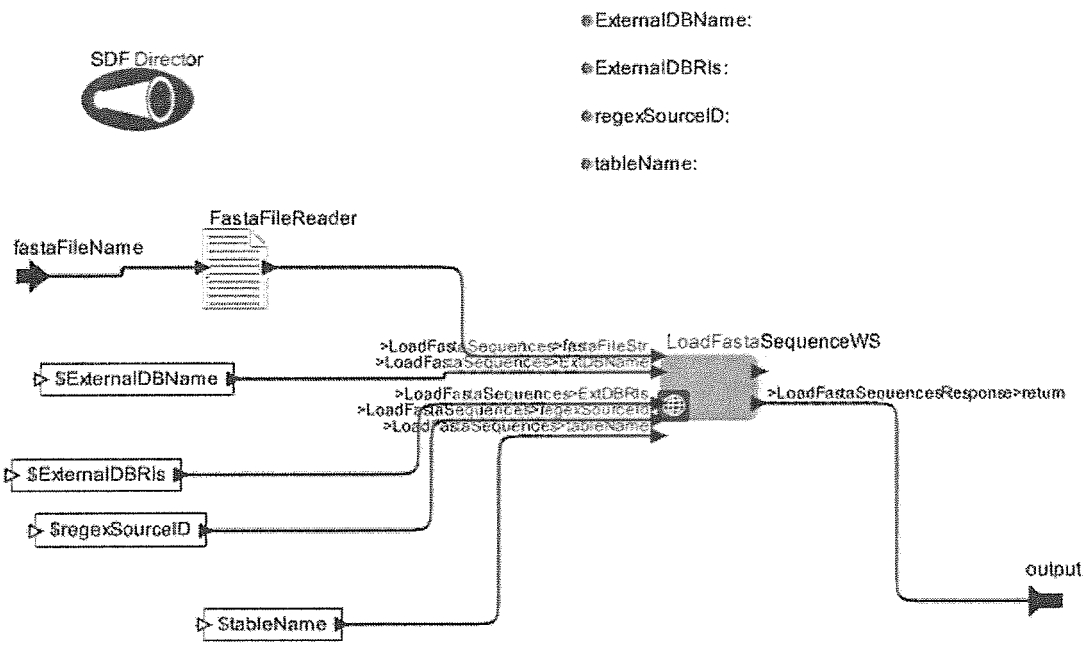


Figura 4.8. Sub-*workflow* associado ao ator LoadGusFastaSequence.

Através de suas portas de saída **ErrorOut** e **ResultOut**, as mensagens de erro e o resultado da execução do serviço web são repassados para o *workflow* que modela o experimento.

4.4.7 – LoadGusTaxon

Taxonomias são armazenadas no GUS através deste ator. Ele também é útil em *workflows* de carga de dados, executados periodicamente para apoiar a execução dos experimentos de um projeto de pesquisa.

Seu sub-*workflow* é apresentado na figura 4.9 e invoca o serviço web **LoadTaxonWS** para armazenar as taxonomias contidas em um conjunto de arquivos texto que são disponibilizados no NCBI. Os *workflows* passam o conteúdo destes arquivos para o sub-*workflow* associado a este ator através de suas portas de entrada **NodesFileContent**, **NamesFileContent**, **GencodeFileContent** e **MergedFileContent**. O sub-*workflow* preenche os parâmetros do serviço web antes de invocá-lo.

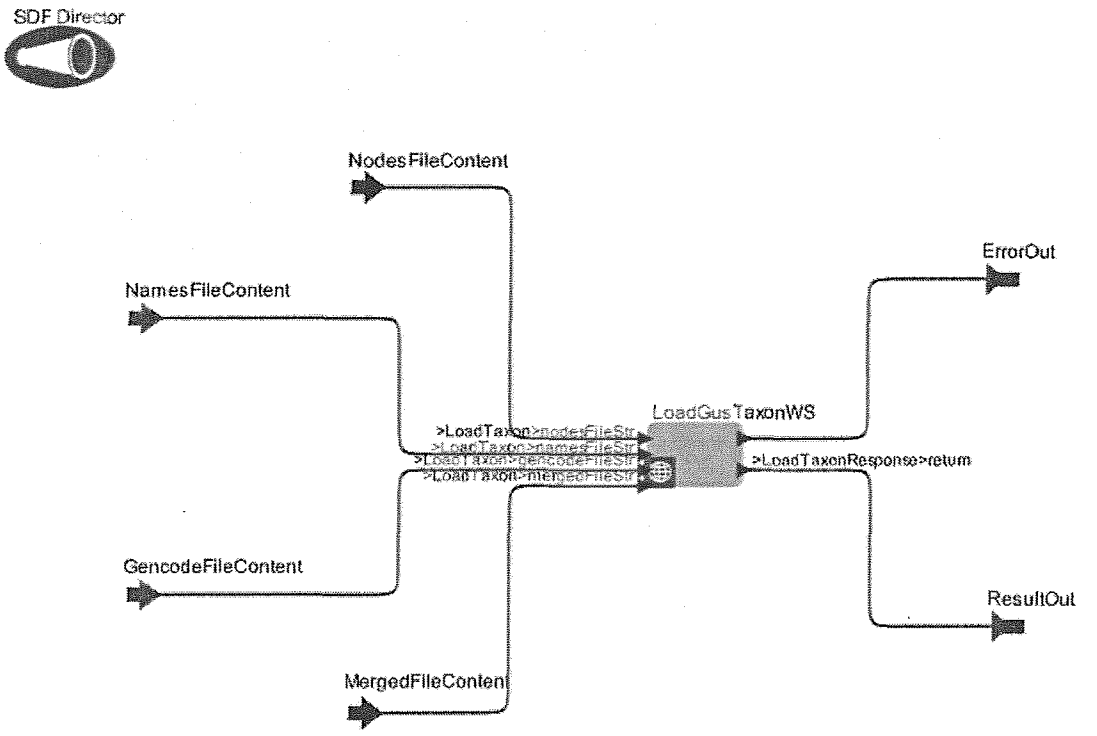


Figura 4.9. Sub-*workflow* associado ao ator LoadGusTaxon.

Este ator repassa as mensagens de erro e de saída da execução do serviço web para o *workflow* que o contém através de suas portas de saída, **ErrorOut** e **ResultOut**.

4.4.8 – QueryGusSequences

Este é o ator responsável por retornar seqüências que foram previamente armazenadas no GUS e salvá-las em um arquivo local. Seu sub-*workflow* contém uma tarefa para a execução do serviço ConectaGUS e pode ser observado na figura 4.10. Este ator possui um parâmetro , **GUSfmk**, que deve ser preenchido com o diretório onde se encontra o arquivo *ConectaGUS.class*, por padrão, localizado no diretório *$HOME/GUSfmk.*

Em suas portas de entrada, devem ser repassados todos os outros valores necessários para a execução do *Serviço de Consulta*. Sua porta de saída repassa para o *workflow*, o resultado da execução do serviço.

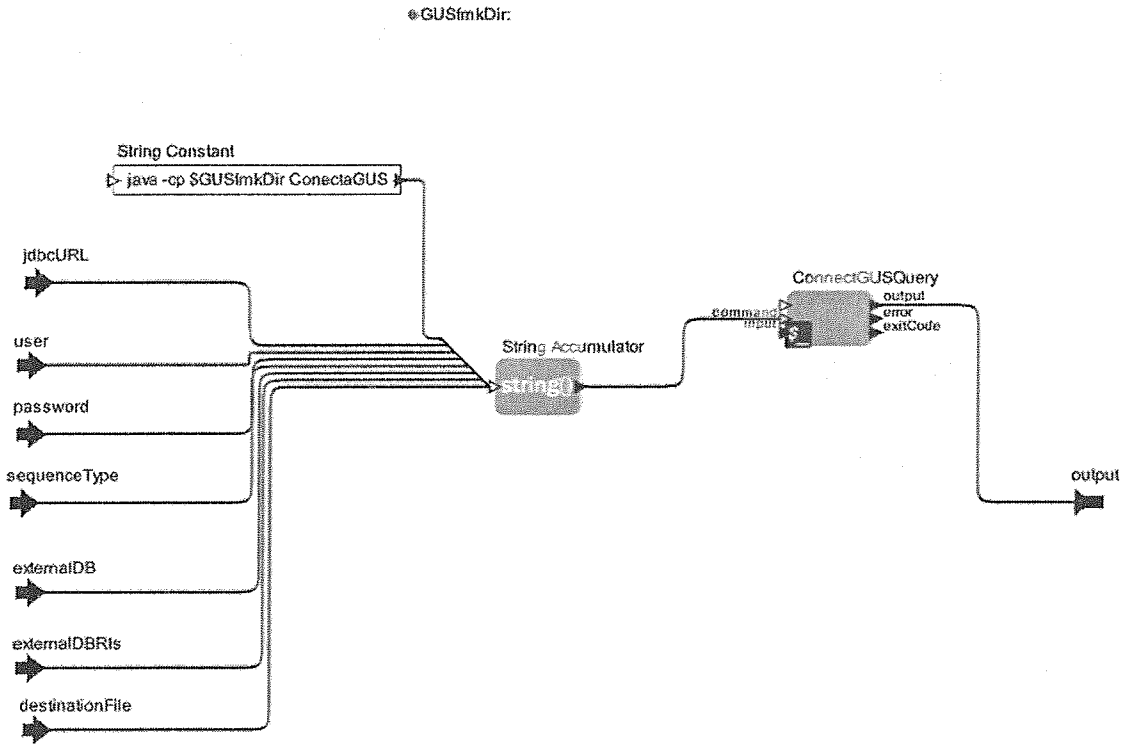


Figura 4.10. Sub-*workflow* do ator QueryGUSSequences.

4.5 – Conclusões

Para permitir a gerência de dados intermediários gerados em *workflows* da bioinformática, este trabalho apresenta uma arquitetura que aproveita os mecanismos de modelagem e execução já bem estabelecidos em SGWfCs, integrando esta tecnologia com um esquema de dados padronizado para o domínio da bioinformática. Como soluções tecnológicas para estes papéis dentro da arquitetura, foram escolhidos o SGWfC Kepler e o esquema GUS.

Para delimitar o escopo deste trabalho, em virtude da complexidade do esquema completo do GUS, foram adotados como domínio, os sub-esquemas **DoTS**, **SRes** e **Core**.

Com o objetivo de tornar esta integração possível, foram desenvolvidos serviços web que acessam um conjunto de *plugins* do GUS, responsáveis diretos pelo armazenamento, tratamento e persistência de dados no GUS. Estes serviços web também podem executar tratamentos essenciais nestes dados intermediários, além de prover recursos auxiliares para o correto funcionamento dos *plugins*. Facilitando a

inclusão de dados neste banco, que tem sido uma atividade executada manualmente pelos cientistas.

O Módulo de Integração com o SGWfC é composto por um conjunto de atores desenvolvidos para o Kepler, os GUS *Actors*, que representam sub-*workflows* que serão invocados durante a execução coordenada do *workflow* que os contém. Estes atores facilitam a modelagem de *workflows* que armazenam dados intermediários no GUS.

Esta solução apresenta algumas vantagens em relação às estratégias de captura e armazenamento de dados de proveniência dos SGWfCs analisados. A gerência de dados intermediários e de alguns dados de proveniência, tanto relacionados aos dados consumidos pelo experimento quanto informações relacionadas à execução do mesmo ocorre sob um esquema que representa os conceitos da área de domínio. Esta característica torna possível um leque de consultas e análises, impossíveis de serem realizadas com os repositórios de dados dos SGWfCs.

Uma outra vantagem é que o cientista pode determinar quais dados do experimento serão capturados e em que momento de sua execução estes dados serão armazenados. Isto evita que o repositório seja populado com dados desnecessários à pesquisa. A possibilidade de consultar as seqüências inseridas no repositório é alcançada através do serviço para consultas.

A adoção da tecnologia de serviços web para a implementação dos serviços de armazenamento torna possível a reutilização, o uso independente de plataforma tecnológica, assim como, a possibilidade de executar o *workflow* em máquinas diferentes do servidor do GUS.

Apesar das vantagens apresentadas, algumas atividades manuais continuam sendo necessárias aos cientistas, como realizar *downloads* periódicos de arquivos auxiliares ao experimento como os que contêm as ontologias e taxonomias.

Capítulo 5 – *Workflow* para Busca de Similaridades entre Seqüências Genômicas

"Noventa por cento do sucesso se baseia simplesmente em insistir."
(Woody Allen)

No capítulo anterior foi apresentada um arquitetura, cujo objetivo é permitir que dados gerados durante a execução de um experimento *in-silico* sejam capturados, tratados e armazenados em um repositório, cujo esquema represente o domínio da bioinformática. Um conjunto de serviços e atores foi desenvolvido no contexto desta dissertação para tornar a integração de um SGWfC com um esquema padronizado para esta área científica, o GUS.

Para validar a arquitetura proposta, assim como, os componentes desenvolvidos para a integração do SGWfC com o GUS, foi modelado um *workflow* para busca de similaridades entre seqüências genômicas. Este é um experimento real utilizado no projeto ProtozoaDB, do grupo BiowebDB e o levantamento de seus requisitos foi realizado através de entrevistas com esta equipe de cientistas.

Este capítulo está dividido da seguinte forma: a primeira seção apresenta o projeto ProtozoaDB, dentro do qual este experimento está inserido. A finalidade e os requisitos do experimento modelado serão explicados na segunda seção deste capítulo. Em seguida, o experimento modelado em um *workflow* no Kepler, utilizando a arquitetura proposta será descrito. Por fim, serão apresentadas as conclusões desta validação.

5.1 – ProtozoaDB

O ProtozoaDB, ou Protozoa *Database*, é um projeto mantido pelo consórcio BiowebDB, criado em 2003, envolvendo pesquisadores das instituições de pesquisa e universidades Fiocruz, UFRJ, Unirio e IME e financiado pelo CNPq.

É uma plataforma que contém inicialmente genomas de diversos protozoários: ***Trypanosoma cruzi***, ***Trypanosoma brucei***, ***Leishmania major***, ***Plasmodium falciparum*** e ***Entamoeba histolytica***. Com o seqüenciamento de genomas de outras espécies de protozoários, este conjunto de genomas acomodados no ProtozoaDB deve aumentar. Este projeto não visa a replicação de dados genômicos já armazenados em outras bases de dados como GeneDB e PlasmoDB, dentre outros. Seu objetivo é atuar de maneira complementar, provendo análises com foco em similaridades distantes e anotações baseadas em filogenias, incluindo análises de ortologia.

O projeto provê uma interface web intuitiva e flexível para consultas personalizadas dos dados genômicos de protozoários. Esta característica é alcançada através da integração de bancos de dados heterogêneos, ferramentas para análises e computação distribuída. Seu objetivo é a mineração de dados genômicos com o uso de estratégias baseadas em computação em grade e em um repositório PostgreSQL cujo esquema é o GUS.

As seqüências de nucleotídeos das cinco espécies de protozoários pesquisadas pelo ProtozoaDB estão disponíveis para *download* a partir dos bancos de dados Genbank, RefSeq e EST, no NCBI. Estes dados são carregados no GUS, versão 3.5, através dos *plugins* do GUS.

5.1.1 – Arquitetura do ProtozoaDB

A arquitetura atual do ProtozoaDB é composta por três módulos principais: **Serviços *Web***, **Sistema de Consultas** e **Plugins para Armazenamento de Dados** (*GUS Plugins*). A camada principal é constituída pelos serviços web baseados em REST (*Representational State Transfer*), através da qual todos os serviços do ProtozoaDB são disponibilizados e os dados acessados.

Cientistas pode interagir com uma página web para acessar os serviços do ProtozoaDB, assim como consultas pré-definidas que podem ser executadas sobre sua base de dados. Aplicações de terceiros podem reutilizar dados do ProtozoaDB através destes serviços. Notificações automáticas são enviadas aos usuários através de *feeds* RSS para informá-los de atualizações na base de dados.

O processamento e armazenamento de dados genômicos de diferentes origens (Genbank, OBO, etc.) é efetuado pelos *plugins* do GUS no repositório PostgreSQL. A figura 5.1, permite a visualização desta arquitetura.

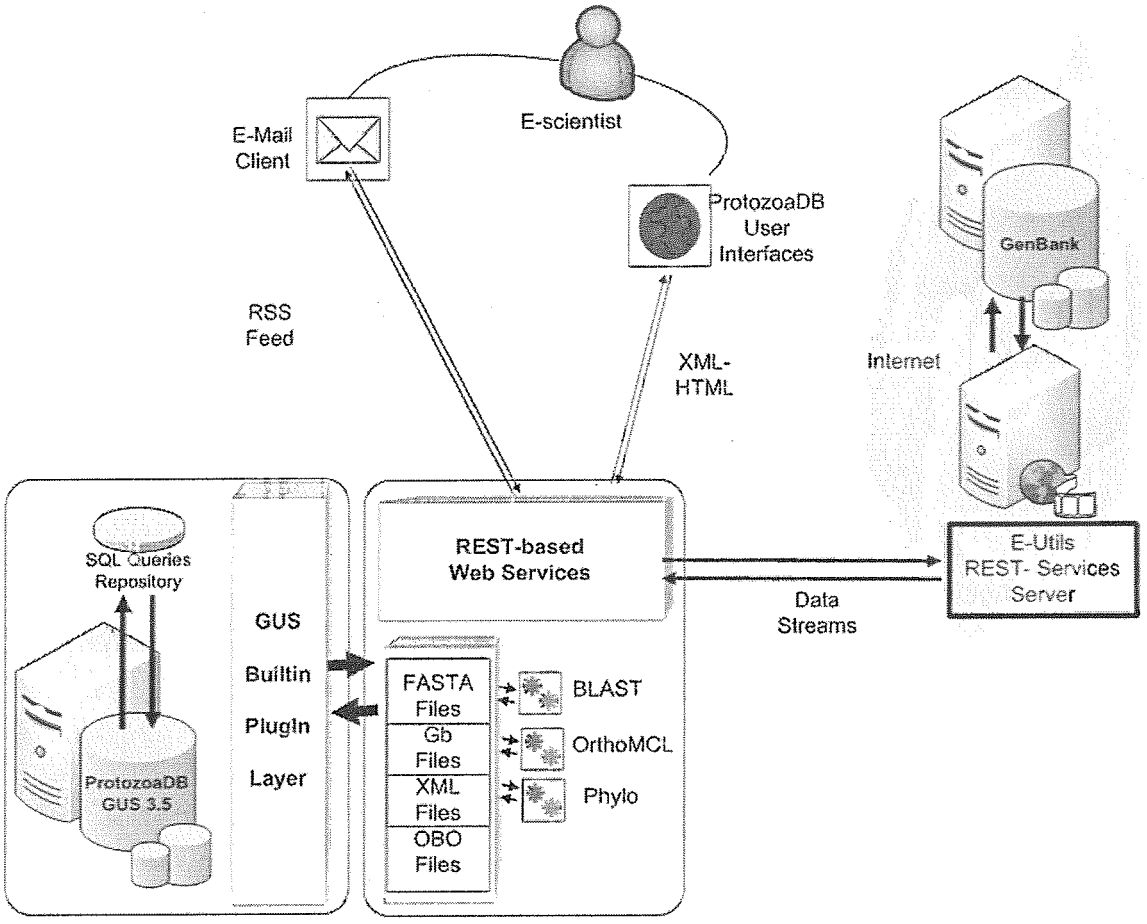


Figura 5.1. Arquitetura do ProtozoaDB.

Um dos problemas deste processo é que os cientistas precisam executar manualmente os *plugins* do GUS, assim como efetuar ajustes em arquivos de carga ou arquivos auxiliares para conseguir executar estes *plugins*, o que se torna uma tarefa onerosa.

5.2 – Requisitos do Experimento

Para construção deste estudo de caso, foram realizadas diversas entrevistas e validações com a equipe do ProtozoaDB, um projeto cuja equipe conta com seis pesquisadores, nove estudantes e três colaboradores. Esta equipe descreveu os processos

utilizados em seus experimentos, esclareceu diversas dúvidas que surgiram durante o levantamento de requisitos do experimento e participou de validações das soluções propostas. Além das entrevistas foram acessados sítios relacionados à pesquisa realizada no contexto do ProtozoaDB e estudo da literatura associada.

Um dos experimentos *in-silico* realizados com freqüência no contexto do projeto ProtozoaDB é o processamento de seqüências genômicas em busca de similaridades entre elas, fazendo uso da técnica de Alinhamento Local.

O Alinhamento Local é utilizado quando se deseja identificar quais seqüências em uma determinada coleção apresentam alinhamento com uma seqüência pesquisada, também chamada de ***Query***. O algoritmo computacional utiliza fragmentos da seqüência pesquisada e alinha com as seqüências de um banco de dados (***Subject***). Este algoritmo descarta todas as pesquisas com baixa pontuação (***Score***) e alinha a vizinhança dos fragmentos com boa pontuação até alcançar um valor máximo.

O objetivo deste algoritmo é identificar seqüências que possuam trechos similares de tamanho significativo, ou seja, com algo em comum. Tanto informações funcionais e evolucionarias das seqüências podem ser inferidas a partir desta pesquisa. O BLAST (*Basic Local Alignmetn Tool*) é uma ferramenta computacional que provê este tipo de algoritmo para seqüências de nucleotídeos e proteínas.

No ProtozoaDB, para executar o BLAST, os cientistas realizam um conjunto de atividades para manter estes dados no repositório GUS, de modo a permitir consultas avançadas sobre as informações armazenadas. É realizado um *download* das seqüências pesquisadas e das seqüências do banco a ser comparado que são armazenadas no GUS.

As seqüências dos genomas pesquisados pelo ProtozoaDB são trazidas em formato Genbank, pois este formato agrega um conjunto de informações de anotações genômicas padronizadas e, desse modo, os cientistas mantém em seu repositório GUS uma riqueza maior de informações. As outras seqüências são trazidas em formato FASTA, mais simples, contendo praticamente apenas as seqüências.

Em seguida, é executado o BLAST e seu resultado é armazenado no GUS. Abaixo, na figura 5.2, é possível visualizar em detalhe, o processo deste experimento.

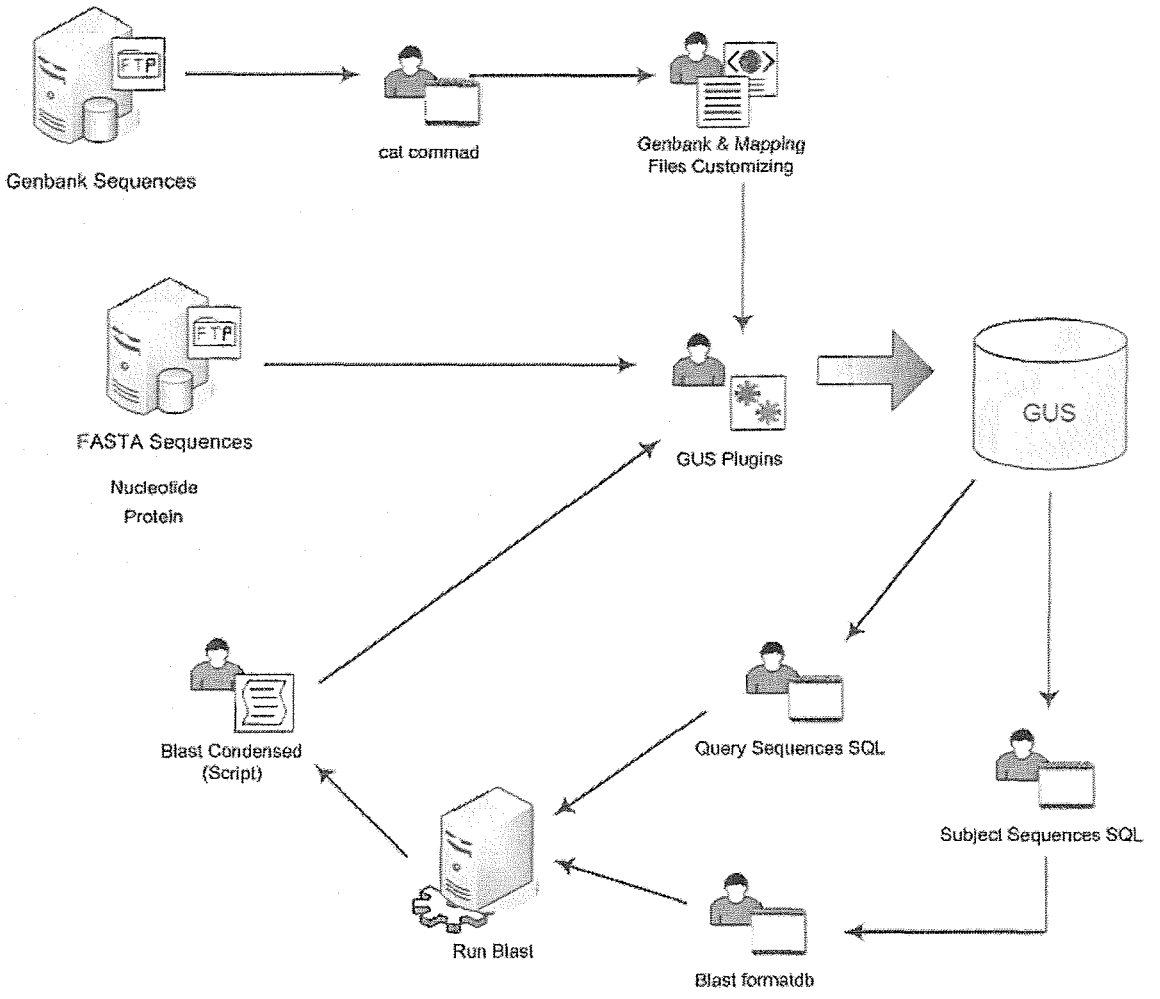


Figura 5.2. Processo para execução do BLAST e armazenamento no GUS.

Atualmente, os cientistas do grupo de pesquisas do ProtozoaDB, executam manualmente cada passo do processo deste experimento, que tem seu início com o *download* via protocolo FTP de vários arquivos texto no formato Genbank, de aproximadamente 250 MB, contendo seqüências de todos as espécies de invertebrados com genomas seqüenciados.

Uma tarefa manual se segue a esta com o objetivo de concatenar todos estes arquivos, gerando apenas um arquivo com todas as seqüências de invertebrados do Genbank. Esta tarefa consiste em executar uma linha de comando do sistema operacional Linux (comando ***cat***).

O arquivo gerado será passado como argumento de entrada para um *script* Perl, que percorre as seqüências deste arquivo gerando dois arquivos de saída no formato

Genbank: um arquivo com as seqüências de uma determinada espécie e outro arquivo com as seqüências de todas as outras espécies de invertebrados.

Este procedimento é realizado para cada uma das espécies pesquisadas pelo grupo e para gerar os arquivos com as seqüências de cada uma das espécies, o código-fonte deste *script* Perl é sempre ajustado, alterando-se os valores de algumas variáveis para especificar qual será a espécie que terá suas seqüências separadas em um arquivo.

Em seguida, os cientistas iniciam o processo de carga das seqüências genômicas no GUS. Esta é uma tarefa complexa e onerosa, pois envolve várias etapas de tentativa e erro e obriga que o pesquisador efetue diversos ajustes nos arquivos relacionados a esta carga.

Para montar o arquivo de mapeamento ***genbank2gus.xml***, necessário para que o *plugin* ***InsertSequenceFeatures.pm*** redirecione as *Features* das seqüências para os objetos corretos do GUS, o cientista deve descobrir quais *Features* se encontram no arquivo do Genbank. Para isso, ele executa um *script* Perl chamado ***gbfeatureslister.pl***. Em seguida, ajusta o arquivo de mapeamento padrão do GUS para contemplar as *Features* presentes no arquivo do Genbank. Esta é uma tarefa manual.

Após este ajuste no arquivo de mapeamento o cientista inicia a execução do *plugin* do GUS que, se encontrar algum *Qualifier* com mais de um valor, levanta uma mensagem de erro e não consegue proceder com a carga dos dados. A partir daí, o cientista deve alterar o arquivo do Genbank manualmente, resolvendo os valores de seus *Qualifiers*, até que o *plugin* consiga carregar todas as seqüências.

Para realizar o alinhamento das seqüências pesquisadas pelo grupo do ProtozoaDB com seqüências de outros bancos, os cientistas efetuam o *download* de arquivos texto no formato FASTA dos bancos RefSeq, Nucleotide, Protein, dentre outros. Estas seqüências são carregadas no GUS, via *plugin* ***LoadFastaSequences.pm***, manualmente, através da sua execução em linha de comando.

Após esta carga de dados, as seqüências do genoma pesquisado e as seqüências do banco de dados contra o qual o genoma será comparado em busca de similaridades devem ser extraídas do GUS através de consultas executadas diretamente na interface gráfica do PostgreSQL utilizada pelo grupo, ou seja, uma tarefa também manual. Isto se

faz necessário para que se mantenham os identificadores de seqüências do GUS e, ao inserir o resultado de similaridade no GUS, o *plugin* possa inserir os apontamentos para os registros das seqüências já armazenadas no GUS.

Uma consulta gera um arquivo com as seqüências do genoma pesquisado e outra consulta gera um arquivo com as seqüências do banco comparado. Estes arquivos são transformados para o formato FASTA, através da execução manual em linha de comando do *script* ***pqsql2fasta.pl***, para cada um dos arquivos.

Em seguida, o cientista deve executar um aplicativo auxiliar do BLAST, chamado ***formatdb***, passando como um dos argumentos, o arquivo que contém as seqüências do banco comparado. O *formatdb* transformará o arquivo FASTA em um banco de dados entendido pelo BLAST, visando a otimizar a execução de seus algoritmos.

O BLAST pode ser executado após a geração destes dois arquivos, através de uma tarefa manual de execução da sua linha de comando. Dependendo do número de seqüências comparadas, a execução do BLAST pode ser extremamente demorada, na maioria das vezes requerendo um processamento distribuído/paralelo e, ainda assim, levando um tempo considerável para produzir resultados. Isso requer que o cientista fique acompanhando sua execução e periodicamente tenha que pausar suas atividades para verificar o andamento da execução do BLAST.

O resultado da execução é um arquivo texto em um formato próprio do BLAST. Esse resultado deve ser armazenado no GUS, mas o *plugin* que insere as similaridades encontradas pelo BLAST, ***InsertBlastSimilarities.pm***, não recebe este arquivo como entrada. Para executar o *plugin*, o cientista deve executar um *script* Perl com o objetivo de condensar este arquivo, transformando-o para um formato que o *plugin* do GUS entenda. Após este conjunto de atividades, o cientista executa o *plugin* do GUS e realiza a carga do resultado da busca por similaridades entre as seqüências pesquisadas e um banco de seqüências.

5.3 – *Workflow* para Busca de Similaridades

É possível notar que várias das etapas descritas na seção anterior podem ser executadas em modo paralelo. Além disso, encadear estas tarefas modelando um *workflow* para representar este experimento é uma estratégia nitidamente vantajosa, pois retira do cientista a responsabilidade de disparar a execução de etapas.

Várias tarefas podem ser automatizadas e os diversos ajustes manuais, eliminados, ao aplicar a arquitetura proposta para gerência dos dados intermediários e os serviços implementados durante este trabalho de pesquisa. Nesta seção é apresentado o *workflow* que foi construído para modelar o experimento mencionado na seção anterior. Na figura 5.3, é possível visualizar o *workflow* completo, que será explicado em seguida.

O primeiro passo para construção deste *workflow*, foi visitar a Fiocruz e os laboratórios de pesquisa do ProtozoaDB com o objetivo de observar as atividades realizadas pelos pesquisadors. Diversas entrevistas com os cientistas, assim como pesquisas em diveros sítios do projeto ProtozoaDB e de bioinformática foram efetuadas para conhecer os diversos recursos computacionais utilizados neste experimento. Foi realizada uma modelagem abstrata do *workflow*, a qual foi validada pelo responsável do projeto.

Após esta etapa foi iniciada a modelagem concreta do *workflow*, fazendo uso dos componentes básicos do Kepler, assim como, dos GUS *Actors* para sua construção. Este *workflow* foi construído em etapas que foram testadas individualmente e, posteriormente, testadas em conjunto, no *workflow* completo. Também foram realizados testes inciais com arquivos e seqüências pequenos que se seguiram por testes reais com arquivos comumente utilizados nesta pesquisa.

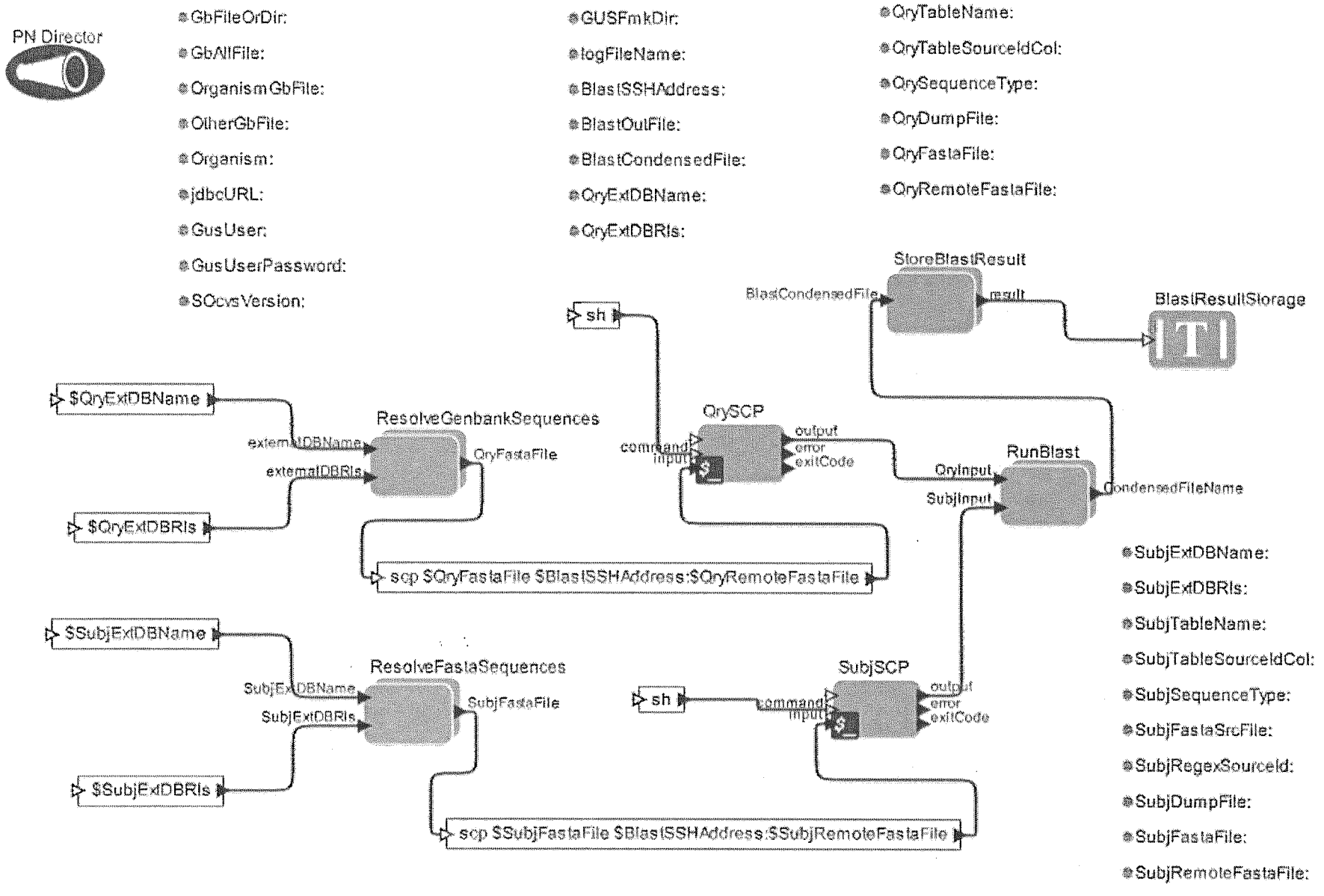


Figura 5.3. *Workflow* para busca de similaridade entre seqüências genômicas.

Este *workflow* utiliza o diretor PN do Kepler, pois dois fluxos são executados em paralelo e este é o diretor capaz de coordenar este tipo de modelo computacional. Um dos fluxos é o que trata as seqüências do Genbank, que são as seqüências pesquisadas pelo grupo do ProtozoaDB. O outro fluxo é o que trata as seqüências no formato FASTA, contra as quais as seqüências pesquisadas serão comparadas no algoritmo de alinhamento.

Para executar este *workflow*, o cientista deve ter efetuado o *download* dos arquivos nos formatos Genbank e FASTA que contém as seqüências envolvidas no experimento. Além disso, é necessário preencher um conjunto de parâmetros iniciais, com valores que serão utilizados ao longo de todo o experimento.

Este *workflow* pode ser executado para diferentes espécies de organismos, tipos distintos de seqüências, assim como, diversos bancos de dados de origem das seqüências comparadas. Essa reutilização é possibilitada através do preenchimento dos valores dos parâmetros de entrada, que podem ser customizados a partir da ***Runtime Window*** do Kepler, que se encontra no item de menu ***Workflow***. Esta janela pode ser visualizada na imagem 5.4.

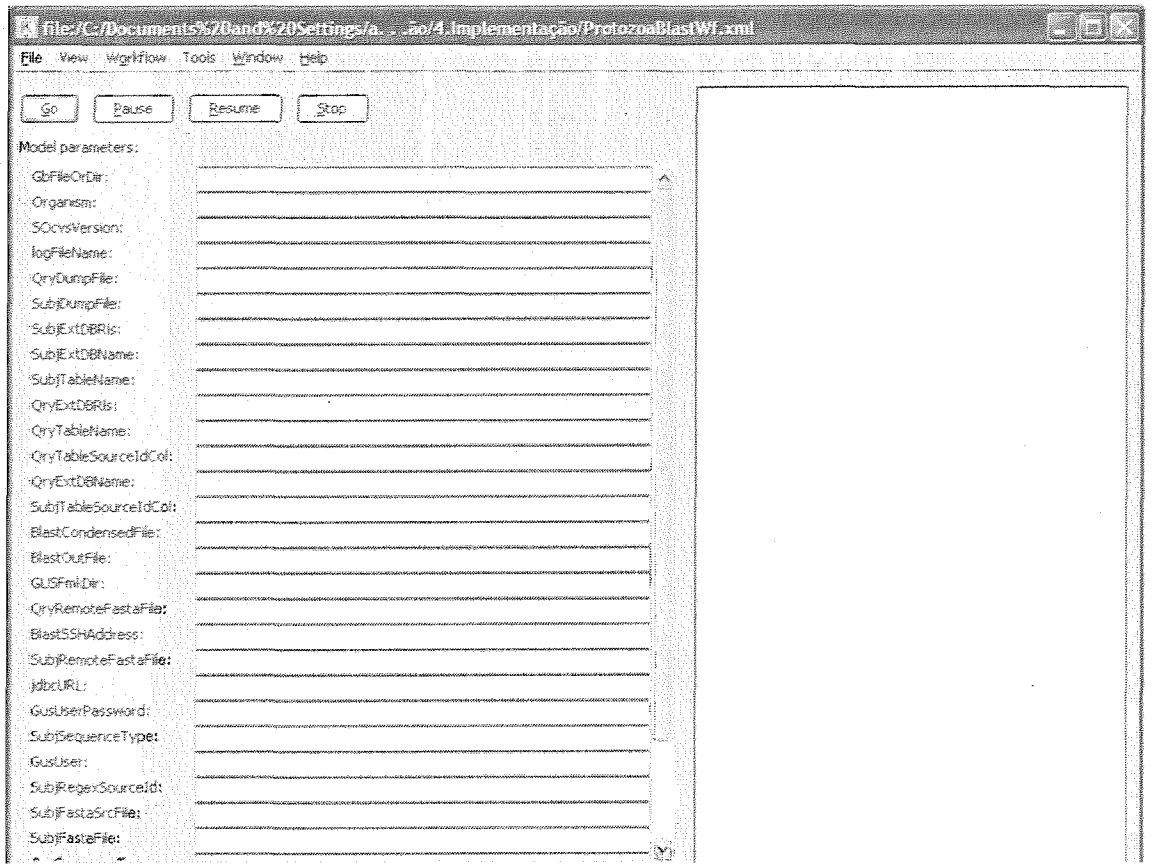


Figura 5.4. Customização de parâmetros do *workflow* na *Runtime Window*.

Os parâmetros de entrada do *workflow* se encontram na área ***Model Parameters***. Alguns desses parâmetros possuem valores padronizados que são utilizados durante a execução sem a necessidade de que o cientista altere seus valores, porém estes podem ser customizados, caso o cientista deseje.

Dentre os parâmetros requeridos para a execução do *workflow* estão:

- **GbFileOrDir:** diretório em que se encontram os arquivos do Genbank ou nome do arquivo;
- **Organism:** nome do organismo cujo genoma está sendo pesquisado;
- **SOcvsVersion:** número da versão das ontologias carregadas previamente no GUS;
- **logFileName:** caminho e nome do arquivo onde serão registrados os logs do *workflow*;
- **QueryDumpFile:** caminho local e nome do arquivo que conterá as seqüências *Query* retornadas do GUS;
- **SubjDumpFile:** caminho local e nome do arquivo que conterá as seqüências *Subject* retornadas do GUS;
- **SubjExtDBRls:** versão do banco no GUS onde serão armazenadas as seqüências do banco contra o qual será executado o alinhamento de seqüências;
- **SubjExtDBName:** nome do banco no GUS onde serão armazenadas as seqüências do banco contra o qual será executado o alinhamento de seqüências;
- **SubjTableName:** nome da tabela do GUS na qual as seqüências *Subject* foram inseridas (DoTS::TranslatedAASequence para seqüências de proteínas e DoTS::ExternalNASequences para seqüências de nucleotídeos);
- **QryExtDBRls:** versão do banco no GUS onde serão armazenadas as seqüências do organismo pesquisado;
- **QryTableName:** nome da tabela do GUS na qual as seqüências *Query* foram inseridas (DoTS::TranslatedAASequence para seqüências de proteínas e DoTS::ExternalNASequences para seqüências de nucleotídeos);

- **QryTableSourceIdCol:** nome da coluna que identifica as seqüências *Query* no GUS;
- **QryExtDBName:** nome do banco no GUS onde serão armazenadas as seqüências do organismo pesquisado;
- **SubjTableSourceIdCol:** nome da coluna que identifica as seqüências *Subject* no GUS;
- **BlastCondensedFile:** caminho e nome do arquivo que conterá o resultado condensado do Blast;
- **BlastOutFile:** caminho e nome do arquivo onde será gravado o resultado do Blast;
- **GUSFmkDir:** diretório onde se encontram todos os scripts e programas auxiliares a este *workflow*, desenvolvidos ou adaptados neste trabalho;
- **QryRemoteFastaFile:** arquivo que conterá as seqüências *Query* na máquina onde será executado o Blast;
- **BlastSSHAddress:** URL do SSH para conectar à maquina onde será executado o Blast e para onde serão transferidos os arquivos necessários à sua execução;
- **SubjRemoteFastaFile:** arquivo que conterá as seqüências *Subject* na máquina onde será executado o Blast;
- **jdbcURL:** URL jdbc para postgreSQL que será utilizada para apontar para o repositório do GUS;
- **GusUserPassword:** senha do usuário que conecta ao repositório do GUS;
- **SubjSequenceType:** tipo das seqüências do banco contra o qual será executado o alinhamento de seqüências, podendo ser nucleotídeos ou proteína;
- **GusUser:** usuário que conecta ao repositório do GUS;
- **SubjRegexSourceId:** expressão regular que identifica o source_id de cada seqüência *Subject* contida no arquivo fasta de origem;
- **SubjFastaSrcFile:** caminho e nome do arquivo de origem que contém as seqüências *Subject* no formato FASTA;
- **SubjFastaFile:** arquivo no formato FASTA que contém as seqüências *Subject* extraídas do GUS;

- **QuerySequenceType:** tipo das seqüências pesquisadas, podendo ser **n**, para nucleotídeos, ou **p**, para proteína;
- **GbAllFile:** caminho e nome do arquivo que conterá as seqüências concatenadas do Genbank;
- **OtherGbFile:** caminho e nome do arquivo que conterá as seqüências dos outros organismos no formato Genbank;
- **OrganismGbFile:** caminho e nome do arquivo que conterá as seqüências no formato Genbank do organismo pesquisado;
- **QryFastaFile:** caminho e nome do arquivo no formato FASTA que conterá as seqüências Query que foram retornadas do GUS.

Todos os outros parâmetros têm seus valores deduzidos ao longo da execução do *workflow*, a partir dos valores preenchidos para os parâmetros acima.

Para modelar o *worflow* foram utilizados vários componentes do Kepler, dentre eles: ***String Parameter***, ***String Constant***, ***External Execution***, ***Simple File Reader*** e ***Relation***. Além dos componentes nativos do Kepler, foram utilizados os GUS *Actors*: ***InsertGusExtDBRls***, ***InsertGusSequenceFeatures***, ***LoadGusFastaSequence*** e ***InsertGusBlastSimilarity***.

Na imagem 5.5 é possível visualizar que, iniciando um dos fluxos paralelos do *workflow*, foi colocado o ator *composite* do Kepler, *ResolveGenbankSequences*, que contém um sub-*workflow* que executa um conjunto de tarefas com o objetivo de tratar as seqüências do .Genbank, armazenar esses dados e metadados no GUS e preparar o arquivo de seqüências para a execução do Blast.

A primeira tarefa desse sub-*workflow* insere um registro para o banco das seqüências pesquisadas no GUS, assim como o registro indicando a versão deste banco. Em seguida, são utilizados os atores do Kepler para execução de linhas de comando no sistema operacional Linux. A etapa ***Cat Command***, é responsável pela concatenação dos arquivos no diretório em que se encontram os arquivos no formato Genbank das seqüências pesquisadas. A etapa ***SplitGb Perl*** faz uma chamada a um *script* Perl que gera um arquivo com as seqüências do organismo pesquisado e outro com as seqüências de todos os outros organismos.

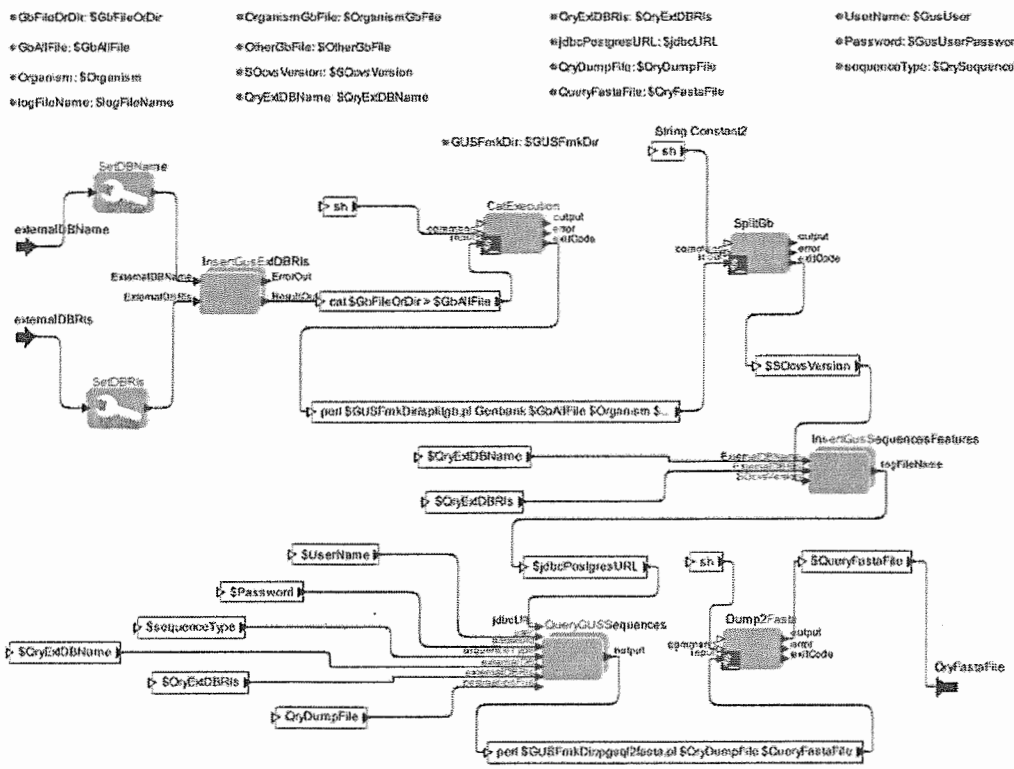


Figura 5.5. Trecho do *workflow* que inclui seqüências do Genbank no GUS.

Após estas tarefas executadas em seqüência, o *workflow* segue com o Gus *Actor* *InsertGusSequenceFeatures*, que executará os devidos tratamentos no arquivo Genbank e armazenará as seqüências e suas *Features* no GUS. Em seguida, será invocado o Serviço para Consultas, através do GUS *Actor* *QueryGUSSequences*, que retornará um conjunto de registros com as seqüências pesquisadas e seu campo de identificação no GUS (source_id). Após este passo, este arquivo é convertido para o formato FASTA através da tarefa *Dump2Fasta*, que executa um *script* Perl. Finalizado este processo, o arquivo com as seqüências *Query* está pronto para ser utilizado na execução do Blast.

O segundo trecho executado em paralelo no início do *workflow* principal é exibido na imagem 5.6. A princípio é utilizado o GUS *Actor* *InsertGUSExtDBRls* que incluirá um registro para o banco externo das seqüências, assim como sua versão no GUS. Em seguida, encontra-se uma tarefa que armazenará os dados das seqüências em formato FASTA no GUS, representada pelo GUS *Actor* *InsertGusFastaSequences*. É armazenado no arquivo de log o resultado da execução desta tarefa. Com as seqüências devidamente armazenadas no GUS, executa-se o GUS *Actor* *QueryGUSSequences*, que retorna os registros das seqüências com seus identificadores do GUS. Este resultado é

convertido para o formato FASTA através da atividade Dump2Fasta que, para isso, executa um *script* Perl.

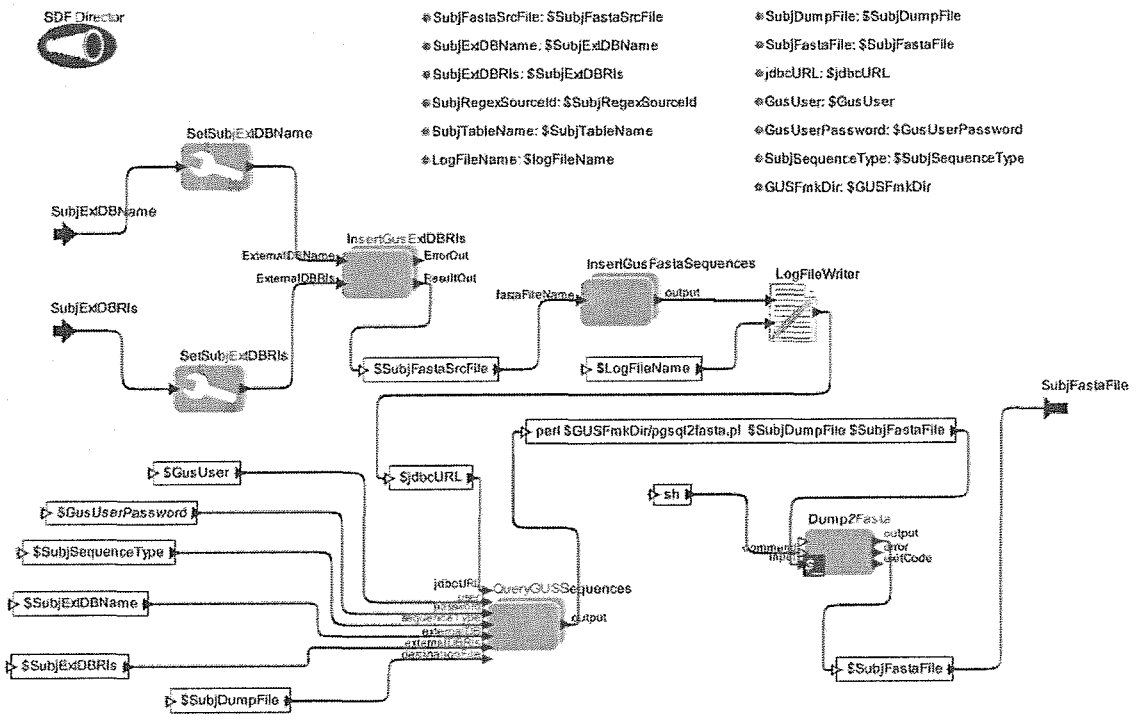


Figura 5.6. Trecho do *workflow* que carrega as seqüências FASTA no GUS.

Após a execução dos dois sub-*workflows* descritos acima, os dois arquivos FASTA com as seqüências que serão comparadas pelo Blast estão prontos e são executadas tarefas que transferem estes arquivos para a máquina onde será executado o Blast via SCP.

Quando a transferência dos arquivos é concluída, os fluxos paralelos se encontram seguindo um caminho comum. Este caminho é iniciado com a tarefa é representada pelo ator *composite* e é chamada *RunBlast* neste *workflow*. Este sub-*workflow* representa um conjunto de passos para executar remotamente o Blast e preparar seu resultado para ser armazenado no GUS. É possível visualizar este sub-*workflow* na imagem 5.7.

Este sub-*workflow* tem seu início com a execução remota do comando *formatdb* do Blast, que prepara as seqüências *Subject* para serem utilizadas pelo Blast. Esta linha de comando é montada a partir dos parâmetros do *workflow*.

O Blast é executado em seguida, via SSH, e sua linha de comando também é montada a partir dos parâmetros do *workflow*. Dependendo do tamanho dos arquivos de entrada, a execução do Blast pode ser muito demorada, levando horas para sua conclusão. O *workflow* aguarda a finalização desta tarefa e, em seguida, transfere o resultado do Blast para a máquina local, via SCP. Finalizando este sub-workflow, é executada uma tarefa que transforma o resultado do Blast em um arquivo que condensa estas informações e é um formato necessário para que seja possível armazenar estes resultados no GUS.

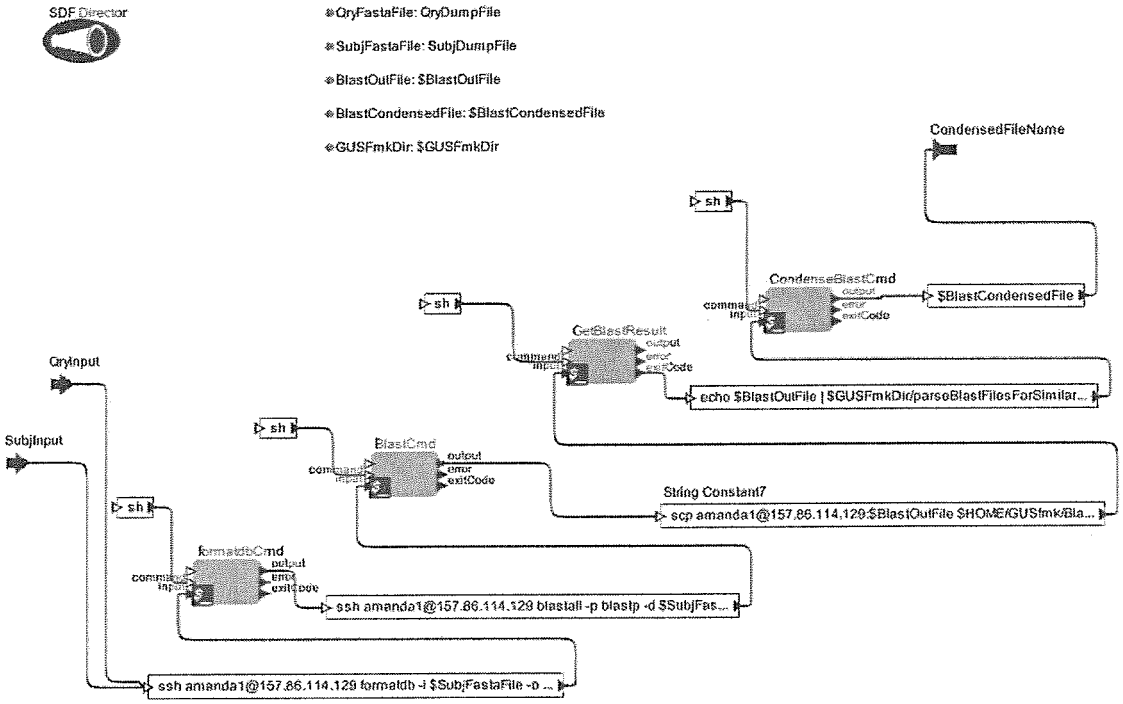


Figura 5.7. Trecho do *workflow* que executa o Blast remotamente.

Na figura 5.8, localizada abaixo, visualiza-se o trecho que finaliza a execução do experimento. Ele é representado pelo sub-*workflow* do ator *composite* do Kepler *StoreBlastResult*, no *workflow* principal. Este sub-*workflow* lê o arquivo condensado com os resultados do Blast e o GUS *Actor InsertGusBlastSimilarity* armazena estes resultados no GUS, utilizando um conjunto de parâmetros definidos pelo usuário antes da execução do *workflow* completo.

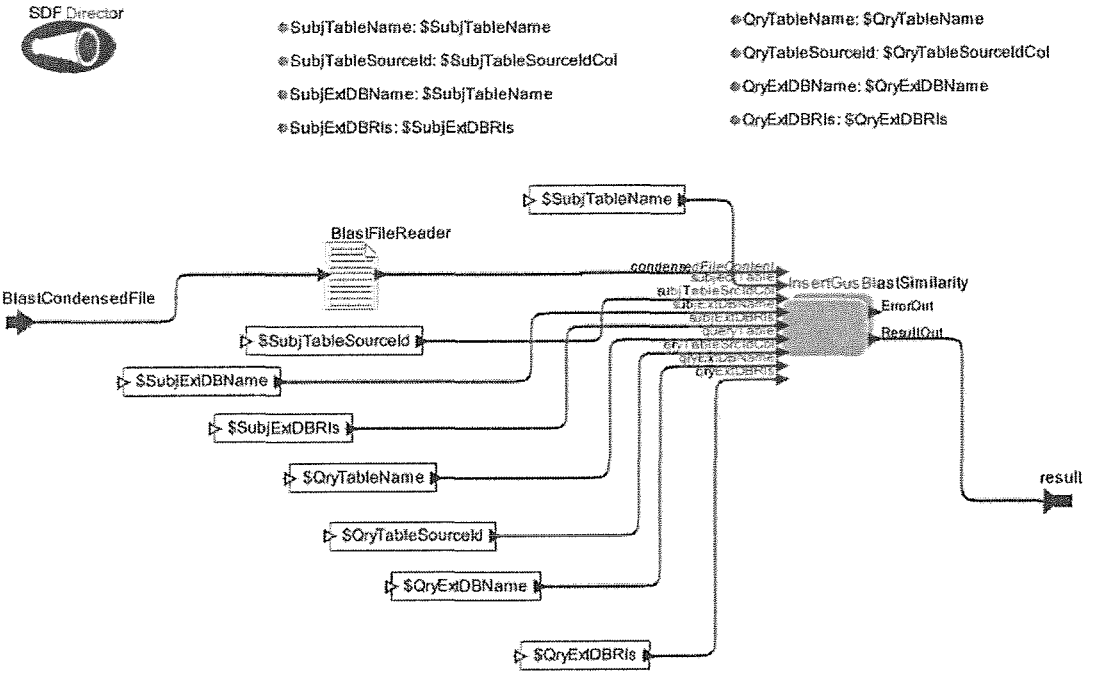


Figura 5.8. Trecho final do *workflow* que armazena o resultado do Blast no GUS.

A finalização da execução ocorre com a exibição do resultado do armazenamento desses resultados no GUS. Isto permite que o cientista verifique se o armazenamento foi efetuado com sucesso ou se ocorreu algum problema no caminho.

5.4 – Conclusões

Nesta seção foi apresentado um estudo de caso que faz uso da arquitetura proposta como solução para gerência de dados genômicos em *workflows* de bioinformática. Este experimento está inserido no contexto da pesquisa realizada pelo projeto ProtozoaDB, cujo objetivo é prover não somente dados genômicos de protozoários como também interfaces de consulta a estes dados e serviços para retorná-los.

O experimento escolhido envolve a busca por similaridades entre seqüências genômicas através de algoritmos de alinhamento executados pelo BLAST do NCBI. O grupo de pesquisas do ProtozoaDB já fazia uso do GUS como seu repositório de dados,

mas não utilizava SGWfCs para automatizar seus experimentos. Neste cenário, foi adequado aplicar a arquitetura proposta.

Foi realizado um levantamento dos requisitos do experimento, assim como os problemas envolvidos. Os cientistas executavam cada etapa do processo manualmente e sequencialmente, ajustando arquivos auxiliares quando necessário.

A construção de um *workflow* que modela este experimento otimizou sua execução, pois os cientistas passaram a atuar apenas na fase inicial do experimento, configurando seus parâmetros de entrada e disparando sua execução.

A execução do experimento foi automatizada e coordenada pelo SGWfC e os dados intermediários gerados foram armazenados em uma base de dados que possui objetos com semântica no contexto da bioinformática. Além dos dados intermediários, alguns dados de proveniência das seqüências pesquisadas e alguns dados de proveniência da execução do experimento também foram armazenados.

Os serviços para carga de dados de taxonomia e ontologias não foram utilizados na modelagem deste *workflow*. O uso destes componentes foi efetuado em uma fase anterior com o objetivo de preparar o repositório para receber os dados gerados durante a execução, relacionando-os com suas taxonomias e ontologias.

Foram realizados diversos testes com seqüências reais extraídas do Genbank e no banco de dados de proteínas não-redundantes do NCBI com o objetivo de validar o *workflow* e o armazenamento dos dados gerados no GUS.

Capítulo 6 – Conclusões

"No final tudo dá certo. Se ainda não deu é porque não chegou ao final."
(Jean Rostand)

A área da bioinformática apresenta vários avanços devido ao crescimento exponencial do uso de recursos computacionais em ambientes científicos. Aplicações de bioinformática são desenvolvidas pela comunidade científica com o propósito de realizar simulações, comparações, cálculos e todo tipo de processamento necessário aos dados gerados pelos seus experimentos *in silico*.

As aplicações construídas para apoiar as atividades da biologia molecular são aplicadas para os mais diversos propósitos e geram dados heterogêneos e em grande volume. Por esse motivo, a necessidade de métodos e processos para automatizar, gerenciar e apoiar o trabalho diário de cientistas e biólogos é proporcional ao índice de crescimento das soluções computacionais para a área. Além disso, capturar dados relevantes à pesquisa gerados durante os experimentos científicos que utilizam este ferramental tecnológico e manter estes dados em repositórios capazes de acomodá-los em objetos que os representem de modo significativo para sua área de domínio se tornou um grande desafio.

Os SGWfCs surgiram para oferecer apoio à modelagem e execução de experimentos cujas tarefas envolvem o uso de ferramentas computacionais. Eles permitem que a execução de um experimento seja automatizada ao modelar tarefas encadeadas que invocam estas aplicações da bioinformática.

Os SGWfCs mais conhecidos na área são o Kepler, o Taverna e o Vistrails. Para este trabalho, estes sistemas foram analisados de modo a verificar sua usabilidade, flexibilidade, assim como, sua capacidade para atender às necessidades dos mais diversos tipos de *workflows*, considerando principalmente, as características dos *workflows* científicos. Todos os sistemas atenderam aos requisitos para modelagem e execução de experimentos *in-silico*, mas apenas o Kepler se mostrou o mais facilmente extensível.

Também foram analisados aspectos relacionados à proveniência dos dados e à gerência dos dados gerados em experimentos modelados nestes ambientes. Todos apresentaram alguma solução para proveniência de dados, mas nenhum destes mecanismos atende às exigências dos *workflows* da bioinformática. O Kepler não disponibiliza junto com seu ambiente uma estratégia para gerência de dados de proveniência.

O Taverna captura dados de execução do experimento, assim como algumas anotações que os cientistas podem atribuir aos seus experimentos, sua solução também provê o armazenamento de dados intermediários gerados durante as execuções, mas estes dados são armazenados assim como são gerados em estruturas que não lhes remete significado algum. Além disso, seu mecanismo de captura de dados de proveniência não permite que o cientista escolha quais dados gerados devem ser armazenados. Todos os dados gerados são armazenados e isso faz com que seu desempenho seja significativamente abalado.

O Vistrails inova em seu mecanismo de captura de proveniência, pois armazena informações do experimento que nenhum outro armazena, como o versionamento do mesmo. Além disso, provê uma interface na qual estas informações podem ser visualizadas e facilmente compreendidas. Porém, as informações geradas durante a execução do experimento não possuem uma solução do mesmo nível. Elas são armazenadas em esquemas muito simples que não atendem ao domínio da bioinformática e o Vistrails não disponibiliza maneiras para consultá-las.

Com esta análise é possível notar que a gerência dos dados envolvidos nos experimentos científicos é uma tarefa árdua para os SGWfCs, pois cada domínio de aplicação científica, como bioinformática ou geologia, por exemplo, possui esquemas, metadados ou ontologias, mais adequados ao seu contexto. Os esquemas de dados propostos pelos SGWfCs são, de um modo geral, extremamente simples e não substituem os esquemas de dados dos domínios de aplicação, que por sua vez, não são facilmente incorporados aos SGWfCs.

Considerando os aspectos mencionados, esta dissertação propõe uma arquitetura para permitir a gerência de dados genômicos em *workflows* de bioinformática. Esta arquitetura considera o aproveitamento das soluções desenvolvidas para definição,

execução e proveniência de dados de *workflows* pelos principais SGWfCs e, como repositório para estes dados, o esquema padronizado para o domínio da bioinformática GUS.

O GUS é um esquema que possui um modelo conceitual padronizado, integrando e estabelecendo relações entre diversos tipos de dados e aplicações da bioinformática. Ele possui um *framework* associado para armazenar, integrar, analisar e apresentar dados genômicos. Ontologias e tipos de dados como genomas, expressão de genes, proteomas e outros podem ser representados neste esquema.

Para prover a integração entre o SGWfC e o repositório GUS a arquitetura prevê um conjunto de serviços web capazes de tratar os dados gerados, armazenar esses dados e efetuar consultas sobre os registros inseridos no GUS. Estes serviços acessam uma camada de *plugins* Perl disponibilizados com o *framework* do GUS e que executam o acesso direto à base de dados, inserindo registros em seus objetos e mantendo a persistência desses dados. Estes *plugins* também realizam algum tratamento nos dados antes de enviá-los para o repositório.

Estes serviços foram desenvolvidos com a tecnologia de serviços web para permitir uma maior compatibilidade com os mais diferentes tipos de aplicações que desejem utilizá-los. Esta tecnologia é abordada por componentes em todos os SGWfCs analisados, por isso, esta solução pode ser utilizada com qualquer SGWfC. Para esta dissertação, foi utilizado o Kepler, mas se outros centros de pesquisa desejarem utilizar a solução para outros SGWfCs, basta utilizar seus componentes para invocação de serviços web, posicionando-os de modo correto no fluxo do experimento. Serviços web também permitem que estes serviços sejam utilizados em qualquer plataforma.

O serviço implementado para consultas no GUS foi implementado em Java, utilizando JDBC para a conexão com a base de dados. Isto permite que este serviço seja independente de plataforma e que possa acessar qualquer base GUS.

Junto à interface do Kepler foram desenvolvidos atores para facilitar o uso dos serviços web e permitir sua reutilização. Estes atores são derivados do ator *Composite* do Kepler, capaz de representar sub-*workflows*. Cada ator possui um sub-*workflow* associado que inclui tarefas preliminares à invocação dos serviços web para armazenamento no GUS.

O uso dos atores permite que o cientista, ao definir o *workflow*, escolha em que etapas o armazenamento dos dados deve ocorrer. Esta característica se torna vantajosa em relação às abordagens adotadas pelos SGWfCs, pois não armazena dados desnecessários no repositório de dados do GUS.

Utilizar a camada de *plugins* do GUS também constitui uma vantagem, pois estes *plugins*, além de prover a persistência dos dados por conhecerem o esquema do GUS, também executam *parsers* nos dados gerados, capturando vários dados de proveniência relacionados aos dados intermediários e armazenando-os no GUS com seus relacionamentos. Estes *plugins* também armazenam alguns dados de proveniência relacionados à execução do experimento.

Desse modo, foi construída uma solução de apoio à proveniência de dados e sua análise, capturando automaticamente esses dados no esquema do GUS. É efetuado o mapeamento e armazenamento dos dados de proveniência gerados ao longo do ciclo de vida dos experimentos da bioinformática, permitindo análises mais sofisticadas sobre as informações geradas.

A tabela 6.1 abaixo, analisa as características desejáveis para *workflows* científicos apresentadas na tabela 3.1 e permite identificar quais destas características foram atingidas com a proposta desta dissertação.

Tabela 6.1. Características de workflows científicos alcançadas com a solução proposta.

Característica	Descrição
Interface	Componentes de integração com a interface do SGWfC, tornando a modelagem com armazenamento de dados no GUS mais intuitiva.
Reutilização	Serviços e componentes de interface reutilizáveis e independentes de plataforma e de SGWfC.
Transformação de dados	Serviços de armazenamento que tratam e transformam os dados antes do seu armazenamento no repositório.
Interações e processamento em lote	Contemplada de acordo com o SGWfC escolhido.
Execução parcial	Contemplada de acordo com o SGWfC escolhido.
Localidade	Componentes e serviços que permitem a execução distribuída.
Complexidade	Contemplada de acordo com o SGWfC escolhido.
Dependência	Contemplada de acordo com o SGWfC escolhido.
Verificação e Validação	Contemplada de acordo com o SGWfC escolhido.

Alterações dinâmicas	Contemplada de acordo com o SGWfC escolhido.

Para validar a solução foi modelado um *workflow* que representa um experimento para buscar similaridades entre seqüências genômicas. Este *workflow* foi testado e utilizado no contexto do projeto ProtozoaDB. Através deste estudo de caso, foi possível verificar a eficiência da arquitetura proposta. Várias tarefas anteriormente executadas manualmente pelos cientistas foram automatizadas com o *workflow* e seus dados foram armazenados do repositório do GUS, onde ficam disponíveis para consultas.

O *workflow* modelado não eliminou a tarefa de efetuar *downloads* dos arquivos que contém as seqüências do Genbank e as seqüências do banco contra o qual as seqüências pesquisadas serão comparadas. Porém, é possível incluir tarefas automatizadas que realizem o *download* das seqüências via FTP através de linhas de comando.

Apesar de capturar alguns dados de proveniência de execução do experimento, a solução apresentada neste trabalho mantém seu foco nos dados intermediários. Por isso, não provê solução para a captura de dados de proveniência relacionados à definição do *workflow* e metadados do experimento. Como um dos trabalhos futuros, é possível construir um serviço que leia as saídas dos serviços web de armazenamento e descubra o valor do campo identificador do registro de execução do experimento no GUS. Tendo em mãos este valor, é possível preencher outras informações já previstas no esquema do GUS, assim como em possíveis extensões deste esquema, através de outro serviço web que pode ser desenvolvido. Dessa forma, é possível integrar uma solução completa para gerência de dados de proveniência para experimentos da bioinformática.

Além das melhorias mencionadas acima, o desenvolvimento de outros serviços que contemplem o armazenamento de dados nos esquemas do GUS que não foram abordados nesta solução também é uma boa proposta. A estratégia apresentada nesta dissertação propõe armazenamento apenas para os sub-esquemas Core, DoTS e SRes. Outros serviços e atores podem para armazenar e persistir dados nos sub-esquemas TEES, Prot, Study e RAD podem ser futuramente desenvolvidos, visando contemplar outras áreas da biologia molecular.

Em relação ao estudo de caso desenvolvido durante esta pesquisa, melhorias futuras também são possíveis. Por exemplo, a implementação de uma interface gráfica amigável, através da qual o cientista possa preencher os parâmetros de entrada do experimento, comandar e acompanhar a execução do *workflow*, invocando a máquina de execução do Kepler via linha de comando, um recurso disponibilizado pelo Kepler.

Também para o experimento de busca de similaridades entre seqüências modelado para o ProtozoaDB, é possível desenvolver consultas sofisticadas e publicar seus resultados em interfaces web construídas com o GUS WDK, o que permitiria uma análise avançada dos dados armazenados através da arquitetura proposta.

Capítulo 7 – Referências Bibliográficas

[1] MORANGE, M., **A History of Molecular Biology**, Hardcover Edition. Cambridge, Harvard University Press, 1998.

[2] **The John Innes Centre, Letter from William Bateson to Alan Sedgwick in 1905**. In: http://www.jic.ac.uk/corporate/about/bateson.htm, 2008.

[3] DZAU, V. J., LIEW, C. C., "The gene in the twenty-first century". In: Dzau, V. J., Liew, C. C. (eds), **Cardiovascular Genetics and Genomics for the Cardiologist**, chapter 1, Durham, USA, Blackwell Publishing, 2007.

[4] WATSON, J. D., CRICK, F. H. C. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid", **Nature** v. 171, n. 4356, pp. 737-738, Apr. 1953.

[5] KANEHISA, M.. "Databases of Biological Information". In: **Trends Guide to Bioinformatics**, Elsevier Science, p. 24-26, 1998.

[6] GOBLE, C.A.,PETTIFFER, S., STEVENS, R., et al., "Knowledge Integration: In silico Experiments in Bioinformatics". In: Foster, I. and Kesselman, C. (eds), **The Grid: Blueprint for a New Computing Infrastructure,** 2 ed., Morgan Kaufman, New York, NY, USA, 2003.

[7] GALPERIN, M. Y., "The Molecular Biology Database Collection: 2008 update", **Nucleic Acids Research** v. 36, Nov. 2007.

[8] LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., et al., "Scientific Workflow Management and the Kepler System", **Concurrency and Computation: Practice & Experience** v. 20, n. 18, 2005.

[9] OINN, T., ADDIS, M., FERRIS, J., et al., "Taverna: a tool for the composition and enactment of bioinformatics workflows", **Bioinformatics** v.20, n.17, Jun. 2003.

[10] CALLAHAN, S. P., FREIRE, J., SANTOS, E., et al., Vistrails: Visualization meets Data Management, Proceedings". **The 2006 ACM SIGMOD international conference on Management of Data**, Chicago, USA, 2006.

[11] **PERL**. Disponível em URL: http://www.perl.org/, visitado em Março de 2008.

[12] ALTINTAS, I., BHAGWANANI, S., BUTTLER, D., et. al., "A Modeling and Execution Environment for Distributed Scientific Workflows". **15th Intl. Conference on Scientific and Statistical Database Management (SSDBM)**, Boston, Massachussets, 2003.

[13] GOBLE, C., WROE, C., STEVENS, R., et al., "The myGrid Project: Services, Architecture and Demonstrator". In: **UK e-Science All Hands Meeting 2003**, pp. 595-603, United Kingdom, England, 2003.

[14] KÜNZL, J., **Development of a Workflow-based Infrastructure for Managing and Executing Web Services**, Universität Stuttgart, Fakultät Informatik, Diplomarbeit n.1997, 2002.

[15] STEIN, L., "Creating a Bioinformatics Nation", **Nature** v. 417, n.9, pp. 119-120, May 2002.

[16] WILKINSON, M. D., LINKS, M., "BioMOBY: an Opensource Biological Web Services Proposal", **Bioinformatics** v. 3, n. 4, pp. 331-341, 2002.

[17] SILVA, F. N., CAVALCANTI, M. C., DÁVILA, A., "In Services: Data Management for In Silico Workflows". **17th International Workshop on Database and Expert Systems Applications (DEXA 2006)**, v. 04-08, pp.206-210, Cracóvia – Polônia, 2006.

[18] **W3C, Web Services Architecture, 2004**. Disponível em URL: http://www.w3.org/TR/ws-arch/, visitado em Janeiro de 2008.

[19] **W3C, Web Services Description Language (WSDL) 2.0**. Disponível em URL: http: //ww w.w3.org/TR/wsdl20/, visitado em Janeiro de 2008.

[20] **W3C. Note on Simple Object Access Protocol (SOAP) 1.2**. Disponível em URL: http: //www.w3.org/TR/SOAP/, visitado em Janeiro de 2008.

[21] CAVALCANTI, M. C., TARGINO, R., BAIÃO, et al., "Managing Structural Genomic Workflows Using Web Services". In: **Data & Knowledge Engineering**, v. 53, n. 1, pp. 45-74, Amsterdam, 2005.

[22] TARGINO, R. S., **O Ambiente 10+C para definição e execução de workflows in silico através de serviços Web**. Tese de M.Sc., COPPE/UFRJ, Rio de Janeiro, RJ, Brasil, 2004.

[23] **Genetic Sequence Data Bank**. Disponível em URL: ftp://ftp.ncbi.nih. gov/ genbank/gbrel.txt, visitado em Fevereiro de 2008.

[24] FOSTER, I., VOCKLER, J., WOILDE, M., et al., "Chimera: A virtual data system for representing, querying, and automating data derivation". In: **SSDBM - 14th International Conference on Scientific and Statistical Database Management**, pp. 37–46, Edinburgh, Scotland, 2002.

[25] SHIMMHAN, Y., PLALE, B., GANNON, D., "A survey of data provenance in e-science", **SIGMOD Record** v. 34, pp. 31-36, 2005.

[26] BUNEMAN, P., KHANNA, S., TAN, W. C., "Why and Where: A Characterization of Data Provenance". **ICDT -8th International Conference on Database Theory**, 2001.

[27] LANTER, D. P., "Design of a Lineage-Based Meta-Data Base for GIS". **Cartography and Geographic Information Systems** v. 18, n. 4, pp. 255-261, Oct 1991.

[28] GREENWOOD, M., GOBLE, C., STEVENS, R., et al., "Provenance of e-Science Experiments - experience from Bioinformatics". **The UK OST e-Science 2nd AHM**, Nottingham, September 2003.

[29] **The Genomics Unified Schema - A Platform for Genomics Databases**. Disponível em URL: http://www.gusdb.org/, visitado em Fevereiro de 2008.

[30] **Plasmo DB**. Disponível em URL: http://www.plasmodb.org, visitado em Março de 2008.

[31] **Gene DB**. Disponível em URL: http://www.genedb.org, visitado em Março de 2008.

[32] **Tcruzi DB**. Disponível em URL: http://www.tcruzidb.org, visitado em Março de 2008.

[33] **Bioweb DB Consortium**. Disponível em URL: http://www.biowebdb.org, visitado em Março de 2008.

[34] DÁVILA, A. M. R., MENDES, P. N., WAGNER, G., et al., "ProtozoaDB: dynamic visualization and exploration of protozoan genomes", **Nucleic Acids Research** v.36, pp.547-552, 2008.

[35] LEITE, M., **O DNA**, 1 ed., São Paulo, Publifolha, 2003.

[36] **Human Genome Project Information**. Disponível em URL: http://www.ornl.gov /sci/techresources/Human_Genome/home.shtml, visitado em Abril de 2008.

[37] **Human Genome Project**. Disponível em URL: http://www.nih.gov/about/ re search resultsforthepublic/HumanGenomeProject.pdf, visitado em Abril de 2008.

[38] **U.S. DOE Genome Image Gallery**. Disponível em URL: http://www.doegenomes.org/, visitado em Maio de 2008.

[39] SAIKI, R. K., GELFAND, D. H., STOFFEL, S., et al., "Primer-Directed Enzymatic Amplification of DNA with a Thermostable DNA Polymerase.", **Science** v.239, n. 4839, pp.487-491, 1998.

[40] ROBERTS, K., RAFF, M., ALBERTS, B., et al., **Molecular Biology of the Cell**. 4 ed. Routledge, Hardcover, 2002.

[41] MALACINSKI, G., **Fundamentos de Biologia Molecular**. 4 ed. Rio de Janeiro, Guanabara Koogan, 2005.

[42] DARBRE, P. D., **Basic Molecular Biology: Essential Techniques, John Wiley & Sons**. Spiral ed. Chichester, NY, 1999.

[43] ZAHA, A. et al., **Biologia Molecular Básica**. 3 ed. Porto Alegre, Editora Mercado Aberto, 2003.

[44] LODISH, H., BERK, A., MATSUDAIRA, P., et al., **Molecular Cell Biology**. 5 ed. New York, Scientific American Books, 2003.

[45] GRIFFITHS, A., WESSLER, S., LEWOTIN, R., et al., **Introdução à Genética**. 8 ed. Rio de Janeiro, Guanabara Koogan, 2006.

[46] FUTUYMA, D. J., **Biologia Evolutiva**. 2 ed. Ribeirão Preto, Sociedade Brasileira de Genética, 1992.

[47] BURNS, G. W., BOTTINO, P. J., **Genética**. 6 ed. Rio de Janeiro, Guanabara Koogan, 1991.

[48] BROOKER, R. J.. "Gene transcription and RNA modification". In: Brooker, R. (ed), **Genetics: analysis and principles**, 2 ed., chapter 12, New York, McGraw-Hill, 2005.

[49] NIERHAUS, K. H., WILSON, D. N.. **Protein Synthesis and Ribosome Structure: Translating the Genome**. Hardcover ed., Weinheim, 2004.

[50] GOODMAN, N., "Biological Data Becomes Computer Literate: New Advances in Bioinformatics", **Current Opinion in Biotechnology** v. 13, n. 1, pp. 68-71, Feb. 2002.

[51] HEAD-GORDON, T., WOOLEY, J. C., "Computational challenges in structural and functional genomics", **IBM Systems Journal** v. 40, n. 2, pp. 265-296, 2001.

[52] **Genetic Transcription. Wikipedia**. Disponível em URL: http://en. wikipedia.org /wiki/Transcription_(genetics), visitado em Março de 2008.

[53] **Stem Cell Information, The National Institutes of Health**. Disponível em URL: http://stemcells.nih.gov/info/scireport/appendixA.asp, visitado em Abril de 2008.

[54] BAXEVANIS, A.D., OUELLETTE, B.F.F., **Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins**. 3 ed. New York, John Wiley & Sons, 2005.

[55] **GOBASE, The Organelle Genome Database**. Disponível em URL: http:// mega sun.bch.umontreal.ca/gobase/gobase.html, visitado em Abril de 2008.

[56] **GtRDB: The Genomic tRNA Database**. Disponível em URL: http://www.bioex plo rer.net/Databases/RNA_Databases/, visitado em Abril de 2008.

[57] **Clusters of Orthologous Groups of proteins (COGs).** Disponível em URL: http://www.ncbi.nlm.nih.gov/COG/, visitado em Abril de 2008.

[58] **The Gene Ontology**. Disponível em URL: http://www.godatabase.org/cgi-bin/ go. cgi, visitado em Maio de 2008.

[59] **European Molecular Biology Laboratory - EMBI**. Disponível em: http://www. em bl.org/, visitado em Maio de 2008.

[60] **National Center for Biotechnology Information**. Disponível em URL: http://www. ncbi.nlm.nih.gov/, visitado em Maio de 2008.

[61] **GenBank Overview**. Disponível em URL: http://www.ncbi.nlm. nih.gov/Gen ban k/, visitado em Maio de 2008.

[62] BAIROCH, A., APWEILER, R., "The SWISS-PROT protein sequence data bank and its supplement TrEMBL", **Nucleic Acids Research** v. 26, n. 1, pp. 38-42, 1998.

[63] BERMAN, H., WESTBROOK, J., FENG, Z., et al, "The Protein Data Bank". **Nucleic Acids Research** v. 28, n. 1, pp. 235-242, 2000.

[64] Okayama, T., Tamura, T., Gojobori, T., et al., "Formal Design and Implementation of an Improved DDBJ DNA Database with a New Schema and Object-oriented Library", **Bioinformatics** v. 14, n. 6, pp. 472-478, 1998.

[65] **RCSB Protein Data Bank**. Disponível em URL: http://www.rcsb. org/pdb/statistics / contentGrowthChart.do?content=total&seqid=100, visitado em Junho de 2008.

[66] ROBBRINS, R. J., "Challenges in the Human Genome Project", **IEEE Engineering in Biology and Medicine**, pp. 25-34, March 1992.

[67] NEEDLEMAN, S., WUNSCH, C., "A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins", **Journal of Molecular Biology** v. 48, pp. 444-453, 1970.

[68] SMITH, T., WATERMAN, M., "Identification of Common Molecular Subsequences", **Journal of Molecular Biology** v. 147, pp. 195-197, 1981.

[69] ALTSCHUL, S. F., GISH, W., MILLER, W., et al., "Basic local alignment search tool", **Journal of Molecular Biology** v. 215, pp.403-410, 1990.

[70] WILBUR, W., LIPMAN, D. J. "Rapid Similarity Searches of Nucleic Acid and Protein Data Banks". In: **The National Academy of Sciences**, v.80, pp. 726–730, Feb.1983.

[71] BAKER, D., SALI, A., "Protein Structure Prediction and Structural Genomics", **Science Magazine** v. 294, pp. 93-96, Oct. 2001.

[72] HEALTH, L., RAMAKRISNAN, N., "The Emerging Landscape of Bioinformatics Software Systems", **IEEE Computer** v. 35, n. 7, pp. 41-45. Jul. 2002.

[73] LANE, M., EDWARDS, J. L., NIELSEN, E., "Biodiversity Informatics: The Challenge of Rapid Development, Large Databases, and Complex Data". In: **26th International Conference on Very Large Data Bases**, v. 10-14, pp. 729-732, Cairo, Egypt, September 2000.

[74] SCHNASE, J. L. "Research Directions in Biodiversity Informatics". In: **26th International Conference on Very Large Data Bases**, V. 10-14, pp. 697-700, Cairo, Egypt , September 2000.

[75] MOUGENOT, I., LIBOUREL, T., DÉHAIS, P., "Genetic Sequence Annotation within Biological Databases". In: **The Fourth International Conference on Databases Systems for Advanced Applications (DASFAA)**, v. 10-13, pp. 333-341, Singapore, April 1995.

[76] **Generic Model Organism Database Construction Set (GMOD).** Disponível em URL: http://www.gmod.org, visitado em Maio de 2008.

[77] **The Genomics Unified Schema**. Disponível em: http://www.gusdb.org/, visitado em Junho de 2008.

[78] ARAÚJO, R. M., BORGES, M. R. da S., "Sistemas de Workflow". In: **XX Jornada de Atualização de Informática - Congresso da SBC**, Fortaleza, Brasil, 2001.

[79] **WFMC, "Workflow Management Coalition Terminalogy & Glossary" WFMC-TC 1011, 1995.** Disponível em URL: http://www.wfmc.org/standards/ docs/ Ref _Model _ 10_years_on_Hollingsworth.pdf, visitado em Maio de 2008.

[80] **WfMC, WfMC – Workflow Management Coalition, "The workflow referece model".** Disponível em URL: http://www.wfmc.org, visitado em Maio de 2008.

[81] AALST, W. e HEE, K., **Workflow Management: Models, Methods, and Systems**. 1 ed., MIT Press, January 2002.

[82] CRUZ, T., **Workflow: A tecnologia que vai revolucionar processos**. 2 ed. São Paulo, Atlas, 2000.

[83] WESKE, M., VOSSEN, G., MEDEIROS, C. M. B., "Scientific Workflows Management: WASA Architecture and Applications", **Fachberich Angewandte Mathematik und Informatik**, Münster, January 1996.

[84] LUDÄSCHER, B., ALTINTAS, I., **On Providing Declarative Design and Programming Constructs for Scientific Workflows based on Process Networks**. In: Technical report, San Diego Supercomputer Center, 2003.

[85] SANGEETA, B., **An Evaluation of End-User Interfaces of Scientific Workflows Management Systems**. Tese de M.Sc., North Carolina State University, EUA, 2005.

[86] LEMOS, M., **Workflow para Bioinformática**. Tese de D.Sc., Pontifícia Universidade Católica - PUC-Rio, Rio de Janeiro, RJ, Brasil, 2004.

[87] MEYER, L. A. V., ROSSLE, S. C., et. al., " Parallelism in Bioinformatics Workflows". In: **VECPAR'2004: 6th International Conference, Valencia, Spain, Revised Selected and Invited Papers**, v. 3402, pp. 583-597, 2005, Valencia, Jun 2005.

[88] DÁVILA, A. M. R., et. al., "Garsa: genomic analysis resources for sequence annotation", **Bioinformatics** v. 21, n. 23, pp. 4302-4303, 2005.

[89] **IBM, Lotus Workflow**. Disponível em URL: http://www.lotus. com/ produc ts/pro duct3.nsf/wdocs/wfhome, visitado em Janeiro de 2005.

[90] **Oracle, Oracle Workflow: Feature overview**. Disponível em: http:// www.oracle .com/technology/products/ias/workflow/, visitado em Janeiro de 2005.

[91] **BEA. BEA Aqualogic**. Disponível em URL: http://www.bea.com/framework.jsp? CN T=index.htm&FP=/content/products/aqualogic/, visitado em Maio de 2008.

[92] SUMPTER, R., "Whitepaper on Data Management", Lawrence Livermore National Laboratory. **The IEEE Metadata Workshop**, 1994.

[93] SINGH, M. P., VOUK, M. A., "Scientific Workflows: Scientific Computing Meets Transactional Workflows". In: **The NSF Workshop on Workflow and Process Automation: State-of-the-Art and Future Directions**, SUPL28—34, Athens, GA, May, 1996.

[94] SANDEEP, C., **Service Based Support for Scientific Workflows**. Dissertação de M.Sc., North Carolina State University, North Caroline, EUA, 2002.

[95] **Introduction to Workflows and Use of Workflows in Grids and Grid Portals, 2003**. Disponível em URL: http://www .extre me .indiana.edu/swf-survey/Intro duction ToWorkflowsInGridsAndPortals_GGF9_2003.ppt, visitado em Junho de 2008.

[96] AALST W. V., DUMAS, M., HOFSETE, A. H. M., "Web Service Composition Languages: Old Wine in New Bottles?". In: **29th Euromicro Conference**, pp. 298-305, 2003.

[97] RANA, O. F., "Creating and Managing Distributed Scientific Workflows: Techniques and Tools". **Euro-Par**, 2005.

[98] YU, J., BUYYA, R. "A Taxonomy of Scientific Workflow Systems for Grid Computing", **SIGMOD Record** v. 34, n. 3, Sept 2005.

[99] **Taverna Project Website, Taverna**. Disponível em URL: http://taverna.sour cefor ge.net/, visitado em Abril de 2008.

[100] ZHAO, J., GOBLE, C., STEVENS, R., "Semantic *web* applications to e-science *in silico* experiments". **The 13th international World Wide Web conference on Alternate track papers & posters**, pp. 284-285, New York, NY, USA, 2004.

[101] **Freefluo Workflow Enactor**. Disponível em URL: http://freefluo. sourceforge.net/, visitado em Maio de 2008.

[102] **IBM, Web Services Flow Language**. Disponível em URL: http://www.ibm. com/devel operworks/webservices/library/ws-ref7/index.html, visitado em Junho de 2008.

[103] ALTINTAS, I., et al., "Kepler: An Extensible System for Design and Execution of Scientific *Workflows*". In: **The 16th Intl. Conf. on Scientific and Statistical Database Management (SSDBM'04),** pp.21-23, Santorini Island, Greece, June 2004.

[104] AALST, W. M. P., HOFSTEDE, A. H. M, BARROS, A. P, et. al., "*Workflow* Patterns". In: **Distributed and Parallel Databases**, v.14 n.1, pp. 5-51, Hingham, USA, July 2003.

[105] **Vistrails**. Disponível em URL: http://www.vistrails.org/index.php/Main_Page, visitado em Abril de 2008.

[106] CHEN, Y., "Data Quality and Provenance in Information Integration". **Workshop on Information Integration**, Arizona State University, 2006.

[107] FRENCH, J. C., "What is metadata?". In: **The Secure Data Management (SDM'92) Workshop: The Role of Metadata in Managing Large Environmental Science Datasets**, pp. 3-8, 1992.

[108] BUNEMAN, P., MAIER, D., WIDOM, J. 2000b. "Where was your data yesterday, and where will it go tomorrow? Data Annotation and Provenance for Scientific Applications". **NSF Workshop on Information and Data Management (IDM '00)**, Chicago, Illinois, 2000.

[109] HACHEM, N. I., QUI, K., GENNERT, M.,et al., "Managing derived data in the Gaea scientific DBMS". In: **The 19th International Conference on Very Large Databases (VLDB '93)**, pp. 1-12, Dublin, Irland, Aug.1993.

[110] ALONSO, G., HAGEN, C., SCHEK, H.-J., et al., "Towards a platform for distributed application development". In: A. Dogac, L. Kalinichenko, M. T. Ozsu and A. Sheth (eds), **Workflow Management Systems and Interoperability**, NATO ASI Series, v. 164 Berlin, Springer, 1998.

[111] SPERY, L., CLARAMUNT, C., LIBOUREL, T., "A lineage metadata model for the temporal management of a cadastre application". In: **The 10th International Workshop on Database and Expert Systems Applications (DEXA '99),** pp. 466-474, Florence, Italy, Sept.1999.

[112] BARKSTROM, B. R., **Digital archive issues from the perspective of an Earth Science data producer**, NASA-98-dadw-brb, 1998.

[113] BROWN, P., AND STONEBRAKER, M. 1995. "Big Sur: A system for the management of Earth science data". In: **The 21st International Conference of Very Large Data Bases (VLDB '95)**, pp. 720–728, Zurich, Switzerland, September 1995.

[114] Goble, C., "Position Statement: Musings on Provenance, Workflow and (Semantic Web) Annotations for Bioinformatics", **Workshop on Data Derivation and Provenance**, Chicago, 2002.

[115] JAGADISH, H., OLKEN, F., "Database Management for Life Sciences Research", **SIGMOD Record** v. 33, n. 2, pp. 15-20, Jun. 2004.

[116] FOSTER, I., VÖCKLER, J., WILDE, M., et al., "The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration". In: **Conference on Innovative Data Systems Research**, Asilomar, CA, USA, January 2003.

[117] GALHARDAS, H., FLORESCU, D., SHASHA, D., et al., "Improving Data Cleaning Quality Using a Data Lineage Facility". In: **Design and Management of Data Warehouses**, Interlaken, Switzerland, June 2001.

[118] WROE, C., GOBLE, C., GODERIS, A., et al., "Recycling workflows and services through discovery and reuse". In: **Concurrency and Computation: Practice & Experience**, v. 19, n. 2, pp. 181-194, Feb. 2005.

[119] **Taverna LogBook**. Disponível em URL: http://www.umanitoba.ca/afs/plant_ scie nce/psgendb/doc/taverna/manual/logbook.html, visitado em Junho de 2008.

[120] **MySQL Database**. Disponível em URL: http://www.mysql.com/, visitado em Junho de 2008.

[121] **LogBook Web Service**. Disponível em URL: http://www.mygrid.org.uk/wiki/ My grid/LogBookWebService, visitado em Julho de 2008.

[122] **Kepler Provenance Framework**. Disponível em URL: http://kepler-project. org/ Wiki.jsp?page=KeplerProvenanceFramework, visitado em Julho de 2008.

[123] FREIRE, J., et al., "VisTrails: visualization meets data management". In: **SIGMOD Conference 2006**, pp 745-747, Chicago, Illinois, 2006.

InsertGusExternalDBWS.java

```
package gus;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class InsertGusExternalDBWS {

     public String InsertGusExternalDB(String databaseName){

          String msg = "Inclusão do Banco Externo: "+
               databaseName +" realizada com sucesso!";

          if ((databaseName != null) &&
               (databaseName.trim() != "")){

               try{
                    // Execute a command without arguments

                    String command =
                    "ga
               GUS::Supported::Plugin::InsertExternalDatabase
               "+"--name " + databaseName + " --commit";

                    Process child =
Runtime.getRuntime().exec(command);
                    BufferedReader brError = new
              BufferedReader(new
                         InputStreamReader(child.getErrorStrea
               m()));
                    String lineError;
                    String Erro = "";
                    while((lineError = brError.readLine()) !=
null) {
                         Erro = Erro + lineError + "\n";
                    }

                    if (Erro != ""){
                         msg = Erro;
                    }


               } catch (Exception e) {
                    System.err.println("Erro no tratamento do
arquivo - " +
                                   e.getMessage());
                    msg = e.getMessage();
               }

          }
          else{
               msg = "Erro. Preencha um nome válido para o nome
do Banco.";
```

```
            }
            return msg;
      }
}
```

InsertGusExternalDBRlsWS.java

```
package gus;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class InsertGusExternalDBRlsWS {

      public String InsertGusExternalDBRls(String databaseName,
String
      databaseRls){

            String msg = "sucesso!";

            if ((databaseName != null) && (databaseName.trim() !=
"")){

                  try{
                        // Execute a command without arguments
                        String command =
                        "ga
                  GUS::Supported::Plugin::InsertExternalDatabaseRl
                  s -
                        databaseName " + databaseName + " --
                  databaseVersion "
                        + databaseRls + " --commit";

                        Process child =
Runtime.getRuntime().exec(command);

                        BufferedReader brError = new
BufferedReader(new

      InputStreamReader(child.getErrorStream()));

                        String lineError;
                        String Erro = "";
                        while((lineError = brError.readLine()) !=
null) {
                              Erro = Erro + lineError + "\n";
                        }

                        if (Erro != ""){
                              msg = Erro;
                        }
                  } catch (Exception e) {
                        System.err.println("Erro no tratamento do
arquivo - " +
                                          msg);
                  throw new RuntimeException(e);
                }
            }
            else{
```

```
                msg = "Erro. Preencha um nome válido para o nome
do Banco.";
            }
            return msg;
        }
}
```

InsertSequenceFeaturesWS.java

```
package gus;

import java.util.*;
import java.io.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.w3c.dom.Node;

import com.sun.org.apache.xml.internal.serialize.OutputFormat;
import com.sun.org.apache.xml.internal.serialize.XMLSerializer;


public class InsertSequenceFeaturesWS {


      /**
       * @author $Amanda Mattos$
       * @version $1.0$
       * @objective $Gravar fisicamente os arquivos envolvidos no
processo$
       */

     private static Boolean GravaArquivo(String strFile, String
nomeArquivo) throws IOException{

            File file = new File(nomeArquivo);
            Boolean sucesso = file.createNewFile();

            BufferedWriter out = new BufferedWriter(new
FileWriter(file));
            out.write(strFile);
            out.close();

            return sucesso;
      }


      /**
       * @author $Amanda Mattos$
       * @version $1.0$
       * @objective $Varrer arquivo do Genbank e tratá-lo para
concatenar múltiplos valores de qualifiers$
       */

     private static String RetornaGbTratado(String strfileGb,
String nomeArquivo) throws Exception{
```

```
            StringBuffer fileGb = new StringBuffer(strfileGb);

            int posIni = fileGb.indexOf("FEATURES");
            posIni = fileGb.indexOf("\n", posIni) + 1;

            String linha = null;
            String qlfAnterior = null;

            int posFeaturesInicial = posIni;

            while ((fileGb.indexOf("\n", posIni) > -1) &&
(fileGb.charAt(posIni) == ' ')){

                  linha = fileGb.substring(posIni,
fileGb.indexOf("\n", posIni));

                  String qualifier = null;
                  if (linha.indexOf("                        /") > -
1){ //é cabeçalho de qualifier
                        qualifier =
linha.substring(linha.indexOf("/") + 1, linha.indexOf("="));

                        if ((qlfAnterior != null) &&
(qualifier.equals(qlfAnterior))){

                              if (fileGb.charAt(posIni - 2) ==
'"'){
                                    fileGb.setCharAt(posIni - 2,
';');
                              }else{
                                    fileGb.insert(posIni - 1, ";");
                              }

                              int posBarra = fileGb.indexOf("/",
posIni);

                              fileGb.delete(posBarra,
fileGb.indexOf("=", posIni)+1);

                              if (fileGb.charAt(posBarra) == '"'){
                                    fileGb.deleteCharAt(posBarra);
                              }

                        }

                        qlfAnterior = qualifier; //pois o conteúdo
do qualifier pode ter várias linhas
                  }
                  else{
                        if (linha.indexOf("                        ")
== -1){ //se não for conteúdo de qlf é feature
                              qlfAnterior = null;
                        }
                  }

                  //Seta próximos valores antes de sair
                  posIni = fileGb.indexOf("\n", posIni) + 1;


            }
```

```
            GravaArquivo(fileGb.toString(), nomeArquivo); //grava
arquivo do genbank tratado

            return fileGb.substring(posFeaturesInicial, posIni);

      }


      /**
       * @author $Amanda Mattos$
       * @version $1.0$
       * @objective $Operação do Web Service que trata o arquivo
do Genbank e monta o arquivo de mapeamento para o GUS, além de
       * armazenar os dados no GUS$
       */

     public String InsertSequenceFeatures(String fileGb, String
databaseName, String databaseRls, String soCvsVersion){

            String tmpNomeDir = this.toString();
            tmpNomeDir =
tmpNomeDir.substring(tmpNomeDir.indexOf("@")+1);

            String diretorio = System.getProperty("user.dir") +
tmpNomeDir;

            System.out.println(System.getProperty("user.dir"));

            //salvar arquivo temporário
            File diretorioTemp = new File(diretorio);
            diretorioTemp.mkdir();

            //String nomeArquivo = diretorioTemp.getPath() +
"\\gbTemp_trat" + tmpNomeDir + ".gb"; //nome do arquivo do Gb
tratado
            String nomeArquivo = diretorioTemp.getPath() +
"/gbTemp_trat" + tmpNomeDir + ".gb"; //nome do arquivo do Gb
tratado

            System.out.println(nomeArquivo);


            String msg = "";

            try{

                  String features = null;
                 try{

                  features = RetornaGbTratado(fileGb,
nomeArquivo);

                 }catch (Exception e){
                  e.printStackTrace();
                 }

                  //cria lista para armazenar as features
                  List<String> lstFeatures = new
ArrayList<String>();

                  String feature = null;
                  String linha = null;
                  String qualifier = null;
```

```
                //Lendo XML padrão
                DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
                dbf.setNamespaceAware(false);
                DocumentBuilder docBuilder =
dbf.newDocumentBuilder();
                //Document doc = docBuilder.parse(new
File("c:\\temp\\genbank2gus.xml"));
                Document doc = docBuilder.parse(new
File("/usr/local/GUSfmw/genbank2gus.xml"));

                //retornando elementos com a tag feature
                Element raiz = doc.getDocumentElement();
                NodeList featuresList =
raiz.getElementsByTagName("feature");

                //1.Percorrer as features e qualifiers do
arquivo Gb e configurar os nós destas features no XML padrão
                Element featureXML = null;

                //cria lista de feature novas para não deixar
que seus qualifiers novos não sejam ignorados
                List<String> lstQlfFeaturesNovas = new
ArrayList<String>();

                while (features.indexOf("\n") != -1) {
//Enquanto houver quebra de linha

                    linha = features.substring(0,
features.indexOf("\n") - 1);

                    features =
features.substring(features.indexOf("\n") + 1);

                    if (linha.indexOf("                    ")
== -1){ //não é linha de qualifier, é feature
                        feature =
linha.substring(linha.indexOf("     ")+5);
                        feature = feature.substring(0,
feature.indexOf(" "));

                        //Percorre as features do XML
buscando pela atual
                        Boolean achouFeature = false;
                        for (int i = 0; i <
featuresList.getLength(); i++) {
                            if
(((Element)featuresList.item(i)).getAttribute("name").equals(fea
ture)) {
                                featureXML =
(Element)featuresList.item(i);
                                achouFeature = true;
                                break;
                            }
                        }

                        //Se achouFeature = false, criar
elemento no XML
                        if (achouFeature == false){

                            Element newFeature =
doc.createElement("feature");
```

```
                                    newFeature.setAttribute("name",
feature);
                                    newFeature.setAttribute("table",
"DoTS::SeqVariation");

      newFeature.setAttribute("so","");
                                    raiz.appendChild(newFeature);
                                    featureXML = newFeature;

                               }

                               if ((feature.length() > 0) &&
(lstFeatures.indexOf(feature) == -1)){
                                    lstFeatures.add(feature);
                               }
                          }
                          else{ //é qualifier
                               if ((linha.indexOf("/") != -1) &&
(linha.indexOf("=") != -1)){
                                    qualifier =
linha.substring(linha.indexOf("/")+1, linha.indexOf("="));

                                    //Retorna lista de qualifiers da
feature
                                    NodeList qualifierList =
featureXML.getElementsByTagName("qualifier");

                                    Boolean achouQlf = false;
                                    for (int j = 0; j <
qualifierList.getLength(); j++) {

                                         if
(((Element)qualifierList.item(j)).getAttribute("name").equals(qu
alifier)){
                                              Element qualifierXML
= (Element)qualifierList.item(j);

                                              //não deixa que
features novas tenham qualifiers não ignorados
                                              if
(lstQlfFeaturesNovas.indexOf(feature + "-" + qualifier) == -1){

      qualifierXML.setAttribute("ignore", "false");
                                              }
                                              else{

      qualifierXML.setAttribute("ignore", "true");
                                              }

                                              achouQlf = true;
                                              break;
                                         }
                                    }

                                    //se achouQlf = false, criar
qualifier com ignore = true
                                    if (achouQlf == false){

                                         Element newQlf =
doc.createElement("qualifier");

      newQlf.setAttribute("name", qualifier);
```

```
                newQlf.setAttribute("ignore", "true");

                featureXML.appendChild((Node)newQlf);


                lstQlfFeaturesNovas.add(feature + "-" + qualifier);

                                        }

                                    }
                                }

                            }

                            //2.Apagar os nós das features que não existem
no arquivo Gb
                            String nomeFeature = null;
                            String strListaFeatures =
lstFeatures.toString();
                            int i = 0;

                            while (i < featuresList.getLength()){

                                nomeFeature =
((Element)featuresList.item(i)).getAttribute("name");

                                if (strListaFeatures.indexOf(nomeFeature)
== -1){

                                    Node nodeDeletar =
featuresList.item(i);
                                    raiz.removeChild(nodeDeletar);

                                }
                                else{
                                    i++;
                                }

                            }

                            //salva localmente o arquivo do genbank e o
arquivo de mapeamento
                        try{

                            //String nomeMapFile =
diretorioTemp.getPath()+"\\genbank2gus.xml"; //nome do arquivo
de mapeamento
                            String nomeMapFile =
diretorioTemp.getPath()+"/genbank2gus.xml"; //nome do arquivo de
mapeamento
                                StringWriter wrtXML = new StringWriter();

                                XMLSerializer serializer = new
XMLSerializer(wrtXML, new OutputFormat(doc, "iso-8859-1",
true));
                                serializer.serialize(doc);

                                wrtXML.close();


                GravaArquivo(wrtXML.getBuffer().toString(), nomeMapFile);
//grava arquivo de mapeamento
```

```
                    //System.out.println(nomeMapFile);

                    if ((databaseName != null) &&
(databaseName.trim() != "")
                              && (databaseRls != null) &&
(databaseRls.trim() != "")
                              && (soCvsVersion != null) &&
(soCvsVersion.trim() != "")){

                         try{
                              // Executa a linha de comando
que invoca o plugin do GUS
                              //String command = "java -cp
c:\\temp\\ TestarFile "+ nomeMapFile + " "+ nomeArquivo;


                              String command = "ga
GUS::Supported::Plugin::InsertSequenceFeatures --mapfile " +
nomeMapFile +" --inputFileOrDir " +

nomeArquivo + " --fileFormat genbank --extDbName " +
databaseName + " --extDbRlsVer "+ databaseRls +
                                                       " --
soCvsVersion " + soCvsVersion  + " --commit";

                              //System.out.println(command);

                              Process child =
Runtime.getRuntime().exec(command);

                              BufferedReader brError = new
BufferedReader(new InputStreamReader(child.getErrorStream()));

                              String lineError;
                              String Erro = "";
                              while((lineError =
brError.readLine()) != null) {
                                   Erro = Erro + lineError +
"\n";
                              }

                              if (Erro != ""){
                                   msg = Erro;
                              }

                         } catch (Exception e) {
                         throw new RuntimeException(e);

                    }

               }
               else{

                    throw new Exception("Erro. Preencha
um nome válido para o nome do Banco.");

               }

          }catch (IOException e){
               e.printStackTrace();
          }
          }catch (Exception e) {
```

```
            e.printStackTrace();
        }

        return msg;

    }
}
```

LoadFastaSequencesWS.java

```
package gus;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

public class LoadFastaSequencesWS {

    private static Boolean GravarArquivo(String strFile, String
nomeArquivo) throws IOException{

            File fastaFileTemp = new File(nomeArquivo);
            Boolean sucesso = fastaFileTemp.createNewFile();

            BufferedWriter out = new BufferedWriter(new
FileWriter(fastaFileTemp));
            out.write(strFile);
            out.close();

        return sucesso;
    }

    public String LoadFastaSequences(String fastaFileStr,
                                                    String
ExtDBName,
                                                    String
ExtDBRls,
                                                    String
regexSourceId,
                                                    String
tableName){
```

```
            String msg = "sucesso!";

            if ((fastaFileStr != null) && (fastaFileStr.trim() !=
"") && (ExtDBName != null)
                        && (ExtDBName.trim() != "") && (ExtDBRls
!= null) && (ExtDBRls.trim() != "")
                        && (regexSourceId != null) &&
(regexSourceId.trim() != "") && (tableName != null)
                        && (tableName.trim() != "") ){

                //cria diretorio temporario para arquivo
condensado

                String tmpNomeDir = this.toString();
                tmpNomeDir =
tmpNomeDir.substring(tmpNomeDir.indexOf("@")+1);

                String diretorio =
System.getProperty("user.dir") + tmpNomeDir;

                File diretorioTemp = new File(diretorio);
                diretorioTemp.mkdir();

                //Carregar string em arquivo temporario
                String nomeArquivo = diretorioTemp.getPath() +
"/fastaFile" + tmpNomeDir + ".fasta";
                try{

                      GravarArquivo(fastaFileStr.toString(),
nomeArquivo);

                }catch (IOException e){
                      return e.getMessage();
                }

                try{
                      // Execute a command without arguments
                      String command = "ga
GUS::Supported::Plugin::LoadFastaSequences --
externalDatabaseName " + ExtDBName +
```

```
                                                    " --
externalDatabaseVersion " + ExtDBRls + " --sequenceFile " +
nomeArquivo +
                                                    " --regexSourceId "
+ regexSourceId + " --tableName " + tableName + " --commit";


                    Process child =
Runtime.getRuntime().exec(command);

                    BufferedReader brError = new
BufferedReader(new InputStreamReader(child.getErrorStream()));

                    String lineError;
                    String Erro = "";
                    while((lineError = brError.readLine()) !=
null) {
                         Erro = Erro + lineError + "\n";
                    }

                    if (Erro != ""){
                         msg = Erro;
                    }

               } catch (Exception e) {

                    return e.getMessage();

               }

          }
          else{
               msg = "Erro. Preencha um nome válido para o nome
do Banco.";
          }

          return msg;

     }
}
```

InsertSequenceOntologyWS.java

```
package gus;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

public class InsertSequenceOntologyWS {

     private static Boolean GravarArquivo(String strFile, String
nomeArquivo) throws IOException{

          File soFileTemp = new File(nomeArquivo);
          Boolean sucesso = soFileTemp.createNewFile();

          BufferedWriter out = new BufferedWriter(new
FileWriter(soFileTemp));
          out.write(strFile.toString());
          out.close();

          return sucesso;
     }

     public String InsertSequenceOntology(String soFileStr,
                                                    String
soVersion,
                                                    String
soCvsVersion){

          String msg = "sucesso!";

          if ((soFileStr != null) && (soFileStr.trim() != "")
&& (soVersion != null)
                    && (soVersion.trim() != "") &&
(soCvsVersion != null) && (soCvsVersion.trim() != "")){

//             cria diretorio temporario para arquivo SO

               String tmpNomeDir = this.toString();
               tmpNomeDir =
tmpNomeDir.substring(tmpNomeDir.indexOf("@")+1);

               String diretorio =
System.getProperty("user.dir") + tmpNomeDir;

               File diretorioTemp = new File(diretorio);
               diretorioTemp.mkdir();

//             Carregar string em arquivo temporario
               String nomeArquivo = diretorioTemp.getPath() +
"/so_file" + tmpNomeDir + ".so";

               try{

                    GravarArquivo(soFileStr.toString(),
nomeArquivo);
```

```
                }catch (IOException e){
                     return e.getMessage();
                }

                try{
//                   Execute a command without arguments
                   String command = "ga
GUS::Supported::Plugin::InsertSequenceOntologyOBO --inputFile "
+ nomeArquivo +
                   " --soVersion " + soVersion + " --
soCvsVersion " + soCvsVersion + " --commit";


                   Process child =
Runtime.getRuntime().exec(command);

                   BufferedReader brError = new
BufferedReader(new InputStreamReader(child.getErrorStream()));

                   String lineError;
                   String Erro = "";
                   while((lineError = brError.readLine()) !=
null) {
                        Erro = Erro + lineError + "\n";
                   }

                   if (Erro != ""){
                        msg = Erro;
                   }

                } catch (Exception e) {

                   return e.getMessage();

                }

           }
           else{
                msg = "Erro. Preencha um nome válido para o nome
do Banco.";
           }

           return msg;
     }
}
```

LoadTaxonWS.java

```
     package gus;

     import java.io.BufferedReader;
     import java.io.BufferedWriter;
     import java.io.File;
     import java.io.FileWriter;
     import java.io.IOException;
     import java.io.InputStreamReader;

     public class LoadTaxonWS {
```

```
     private static Boolean GravarArquivo(String strFile,
String nomeArquivo) throws IOException{

          File fileTemp = new File(nomeArquivo);
          Boolean sucesso = fileTemp.createNewFile();

          BufferedWriter outFile = new BufferedWriter(new
FileWriter(fileTemp));
          outFile.write(strFile);
          outFile.close();

          return sucesso;
     }

     public String LoadTaxon(String nodesFileStr,
                                   String namesFileStr,
                                   String
gencodeFileStr,
                                   String
mergedFileStr){

          String msg = "sucesso!";

          if ((nodesFileStr != null) &&
(nodesFileStr.trim() != "") && (namesFileStr != null)
                    && (namesFileStr.trim() != "") &&
(gencodeFileStr != null) && (gencodeFileStr.trim() != "")
                    && (mergedFileStr != null) &&
(mergedFileStr.trim() != "")){

//             cria diretorio temporario para arquivo SO

               String tmpNomeDir = this.toString();

               tmpNomeDir =
tmpNomeDir.substring(tmpNomeDir.indexOf("@")+1);

               String diretorio =
System.getProperty("user.dir") + tmpNomeDir;

               File diretorioTemp = new File(diretorio);
               diretorioTemp.mkdir();

//             Carregar string em arquivo temporario
               String nomeArquivoNodes =
diretorioTemp.getPath() + "/nodes" + tmpNomeDir + ".dmp";
               String nomeArquivoNames =
diretorioTemp.getPath() + "/names" + tmpNomeDir + ".dmp";
               String nomeArquivoGencode =
diretorioTemp.getPath() + "/gencode" + tmpNomeDir + ".dmp";
               String nomeArquivoMerged =
diretorioTemp.getPath() + "/merged" + tmpNomeDir + ".dmp";

               try{

                    //grava arquivo nodes.dmp

     GravarArquivo(nodesFileStr.toString(),
nomeArquivoNodes);

                    //grava arquivo names.dmp
```

```
      GravarArquivo(namesFileStr.toString(),
nomeArquivoNames);

                        //grava arquivo gencode.dmp

      GravarArquivo(gencodeFileStr.toString(),
nomeArquivoGencode);

                        //grava arquivo merged.dmp

      GravarArquivo(mergedFileStr.toString(),
nomeArquivoMerged);

                  }catch (IOException e){
                        return e.getMessage();
                  }

                  try{
//                      Execute a command without arguments
                        String command = "ga
GUS::Supported::Plugin::LoadTaxon --nodes " +
nomeArquivoNodes +
                        " --names " + nomeArquivoNames + " --
gencode " + nomeArquivoGencode +
                        " --merged " + nomeArquivoMerged + "
--commit";

                        Process child =
Runtime.getRuntime().exec(command);

                        BufferedReader brError = new
BufferedReader(new
InputStreamReader(child.getErrorStream()));

                        String lineError;
                        String Erro = "";
                        while((lineError =
brError.readLine()) != null) {
                              Erro = Erro + lineError + "\n";

                        }

                        if (Erro != ""){
                              msg = Erro;
                        }

                  } catch (Exception e) {

                        return e.getMessage();

                  }

            }
            else{
                  msg = "Erro. Preencha um nome válido para
o nome do Banco.";
            }

            return msg;
      }
}
```

InsertBlastSimilarityWS.java

```
package gus;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;

public class InsertBlastSimilarityWS {

     private static Boolean GravarArquivo(String strFile, String
nomeArquivo) throws IOException{

          File cndBlastTemp = new File(nomeArquivo);
          Boolean sucesso = cndBlastTemp.createNewFile();

          BufferedWriter out = new BufferedWriter(new
FileWriter(cndBlastTemp));
          out.write(strFile);
          out.close();

          return sucesso;
     }

     public String InsertBlastSimilarity(String
condensedFileStr,
                                                    String
subjectTable,
                                                    String
subjectTableSrcIdCol,
                                                    String
subjectExtDBName,
                                                    String
subjectExtDBRls,
                                                    String
queryTable,
                                                    String
queryTableSrcIdCol,
                                                    String
queryExtDBName,
                                                    String
queryExtDBRls){

          String msg = "sucesso!";

          if ((condensedFileStr != null) &&
(condensedFileStr.trim() != "") && (subjectTable != null)
                    && (subjectTable.trim() != "") &&
(subjectTableSrcIdCol != null) && (subjectTableSrcIdCol.trim()
!= "")
                    && (subjectExtDBName != null) &&
(subjectExtDBName.trim() != "") && (subjectExtDBRls != null)
                    && (subjectExtDBRls.trim() != "") &&
(queryTable != null) && (queryTable.trim() != "")
                    && (queryTableSrcIdCol != null) &&
(queryTableSrcIdCol.trim() != "") && (queryExtDBName != null)
                    && (queryExtDBName.trim() != "") &&
(queryExtDBRls != null) && (queryExtDBRls.trim() != "")){
```

```
                //cria diretorio temporario para arquivo
condensado

                String tmpNomeDir = this.toString();
                tmpNomeDir =
tmpNomeDir.substring(tmpNomeDir.indexOf("@")+1);

                String diretorio =
System.getProperty("user.dir") + tmpNomeDir;

                File diretorioTemp = new File(diretorio);
                diretorioTemp.mkdir();

                //Carregar string em arquivo temporario
                String nomeArquivo = diretorioTemp.getPath() +
"/blast_condensed" + tmpNomeDir + ".blast";

                try{

                     GravarArquivo(condensedFileStr.toString(),
nomeArquivo);

                }catch (IOException e){
                     return e.getMessage();
                }

                try{
                     // Execute a command without arguments
                     String command = "ga
GUS::Supported::Plugin::InsertBlastSimilarities --file " +
nomeArquivo +
                                                  " --subjectTable " +
subjectTable + " --subjectTableSrcIdCol " + subjectTableSrcIdCol
+
                                                  " --subjectExtDbName
" + subjectExtDBName + " --subjectExtDbRlsVer " +
subjectExtDBRls +
                                                  " --queryTable " +
queryTable +     " --queryTableSrcIdCol " + queryTableSrcIdCol +
                                                  " --queryExtDbName "
+ queryExtDBName + " --queryExtDbRlsVer " + queryExtDBRls + " --
commit";


                     Process child =
Runtime.getRuntime().exec(command);

                     BufferedReader brError = new
BufferedReader(new InputStreamReader(child.getErrorStream()));

                     String lineError;
                     String Erro = "";
                     while((lineError = brError.readLine()) !=
null) {
                          Erro = Erro + lineError + "\n";
                     }

                     if (Erro != ""){
                          msg = Erro;
                     }

                } catch (Exception e) {
```

```
                    return e.getMessage();

               }

          }
          else{
               msg = "Erro. Preencha um nome válido para o nome
do Banco.";
          }
          return msg;
     }
}
```

ConectaGUS.java

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.sql.*;

public class ConectaGUS {

     public static void main(String args[])
     {

     String url = args[0];
     System.out.println(" Esta é a URL:  " + url);

     try
     {
          Class.forName( "org.postgresql.Driver" );
     }catch ( java.lang.ClassNotFoundException e ){
          System.err.print( "ClassNotFoundException: " );
          System.err.println( e.getMessage () );
     }

     System.out.println("Driver do PostgreSQL selecionado. ");

     try
     {
          Connection db = DriverManager.getConnection(url,
args[1], args[2]);//"amanda1", "amanda197");
          //db = DriverManager.getConnection( url, "postgres",
"" );

          System.out.println("Conexão aberta. ");

          Statement sq_stmt = db.createStatement();

          String sql_str = "";

          System.out.println(args[0]);

          if (args[3].equals("n")){
```

```
                sql_str = "SELECT source_id, sequence FROM
dots.externalnasequence "+
                "WHERE external_database_release_id IN (SELECT
external_database_release_id "+
                "FROM sres.externaldatabaserelease WHERE version
= '"+ args[5] + "' "+
                "AND external_database_id IN (SELECT
external_database_id FROM sres.externaldatabase "+
                "WHERE name = '" + args[4]+"'))";


          }else if (args[3].equals("p")){

                sql_str = "SELECT source_id, sequence FROM
dots.translatedaasequence "+
                "WHERE external_database_release_id IN (SELECT
external_database_release_id "+
                "FROM sres.externaldatabaserelease WHERE version
= '"+ args[5] + "' "+
                "AND external_database_id IN (SELECT
external_database_id FROM sres.externaldatabase "+
                "WHERE name = '" + args[4]+"'))";


          }else System.err.print("É necessário escolher um tipo
de sequência...");



          ResultSet rs = sq_stmt.executeQuery(sql_str);
          String strArquivo = "";

          while (rs.next())
          {
                String id = rs.getString("source_id");
                String name = rs.getString("sequence");

                strArquivo = strArquivo +id + "\t" + name+"\n";

                //System.out.println(id + "\t" + name);
          }

          System.out.println("Consulta efetuada. ");

          File file = new File(args[6]);
          try{
                Boolean sucesso = file.createNewFile();
                BufferedWriter out = new BufferedWriter(new
FileWriter(file));
                out.write(strArquivo);
                out.close();
          }catch (IOException e){
                System.out.println(e.getMessage());
          }

          sq_stmt.close();
          rs.close();
          db.close();

     }catch ( SQLException ex ){
          System.err.println( "SQLException: " +
ex.getMessage() );
```

```
            System.out.println("Conexão fechada. ");
        }
}
```

Anexo B – Arquivos WSDL dos Serviços Web

InsertGusExternalDBWS

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="InsertGusExternalDB">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="databaseName" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="InsertGusExternalDBResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
```

```
  </wsdl:types>
- <wsdl:message name="InsertGusExternalDBRequest">
  <wsdl:part name="parameters" element="ns0:InsertGusExternalDB"
/>
  </wsdl:message>
- <wsdl:message name="InsertGusExternalDBResponse">
  <wsdl:part name="parameters"
element="ns0:InsertGusExternalDBResponse" />
  </wsdl:message>
- <wsdl:portType name="InsertGusExternalDBWSPortType">
- <wsdl:operation name="InsertGusExternalDB">
  <wsdl:input message="ns0:InsertGusExternalDBRequest"
wsaw:Action="urn:InsertGusExternalDB" />
  <wsdl:output message="ns0:InsertGusExternalDBResponse"
wsaw:Action="urn:InsertGusExternalDBResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="InsertGusExternalDBWSSOAP11Binding"
type="ns0:InsertGusExternalDBWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertGusExternalDB">
  <soap:operation soapAction="urn:InsertGusExternalDB"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertGusExternalDBWSSOAP12Binding"
type="ns0:InsertGusExternalDBWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertGusExternalDB">
  <soap12:operation soapAction="urn:InsertGusExternalDB"
style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
```

```
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertGusExternalDBWSHttpBinding"
type="ns0:InsertGusExternalDBWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="InsertGusExternalDB">
  <http:operation
location="InsertGusExternalDBWS/InsertGusExternalDB" />
- <wsdl:input>
  <mime:content type="text/xml" part="InsertGusExternalDB" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="InsertGusExternalDB" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InsertGusExternalDBWS">
- <wsdl:port name="InsertGusExternalDBWSSOAP11port_http"
binding="ns0:InsertGusExternalDBWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/InsertGusExternalDBWS/servi
ces/InsertGusExternalDBWS" />
  </wsdl:port>
- <wsdl:port name="InsertGusExternalDBWSSOAP12port_http"
binding="ns0:InsertGusExternalDBWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/InsertGusExternalDBWS/servi
ces/InsertGusExternalDBWS" />
  </wsdl:port>
- <wsdl:port name="InsertGusExternalDBWSHttpport"
binding="ns0:InsertGusExternalDBWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/InsertGusExternalDBWS/servi
ces/InsertGusExternalDBWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

InsertGusExternalDBRlsWS

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="InsertGusExternalDBRls">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="databaseName" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="databaseRls" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="InsertGusExternalDBRlsResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="InsertGusExternalDBRlsRequest">
```

```
  <wsdl:part name="parameters"
element="ns0:InsertGusExternalDBRls" />
  </wsdl:message>
- <wsdl:message name="InsertGusExternalDBRlsResponse">
  <wsdl:part name="parameters"
element="ns0:InsertGusExternalDBRlsResponse" />
  </wsdl:message>
- <wsdl:portType name="InsertGusExternalDBRlsWSPortType">
- <wsdl:operation name="InsertGusExternalDBRls">
  <wsdl:input message="ns0:InsertGusExternalDBRlsRequest"
wsaw:Action="urn:InsertGusExternalDBRls" />
  <wsdl:output message="ns0:InsertGusExternalDBRlsResponse"
wsaw:Action="urn:InsertGusExternalDBRlsResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="InsertGusExternalDBRlsWSSOAP11Binding"
type="ns0:InsertGusExternalDBRlsWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertGusExternalDBRls">
  <soap:operation soapAction="urn:InsertGusExternalDBRls"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertGusExternalDBRlsWSSOAP12Binding"
type="ns0:InsertGusExternalDBRlsWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertGusExternalDBRls">
  <soap12:operation soapAction="urn:InsertGusExternalDBRls"
style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
```

```
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertGusExternalDBRlsWSHttpBinding"
type="ns0:InsertGusExternalDBRlsWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="InsertGusExternalDBRls">
  <http:operation
location="InsertGusExternalDBRlsWS/InsertGusExternalDBRls" />
- <wsdl:input>
  <mime:content type="text/xml" part="InsertGusExternalDBRls" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="InsertGusExternalDBRls" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InsertGusExternalDBRlsWS">
- <wsdl:port name="InsertGusExternalDBRlsWSSOAP11port_http"
binding="ns0:InsertGusExternalDBRlsWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/InsertGusExternalDBRlsWS/se
rvices/InsertGusExternalDBRlsWS" />
  </wsdl:port>
- <wsdl:port name="InsertGusExternalDBRlsWSSOAP12port_http"
binding="ns0:InsertGusExternalDBRlsWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/InsertGusExternalDBRlsWS/se
rvices/InsertGusExternalDBRlsWS" />
  </wsdl:port>
- <wsdl:port name="InsertGusExternalDBRlsWSHttpport"
binding="ns0:InsertGusExternalDBRlsWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/InsertGusExternalDBRlsWS/se
rvices/InsertGusExternalDBRlsWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

InsertSequenceFeaturesWS

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="InsertSequenceFeatures">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="fileGb" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="databaseName" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="databaseRls" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="soCvsVersion" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="InsertSequenceFeaturesResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
```

```
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="InsertSequenceFeaturesRequest">
  <wsdl:part                                    name="parameters"
element="ns0:InsertSequenceFeatures" />
  </wsdl:message>
- <wsdl:message name="InsertSequenceFeaturesResponse">
  <wsdl:part                                    name="parameters"
element="ns0:InsertSequenceFeaturesResponse" />
  </wsdl:message>
- <wsdl:portType name="InsertSequenceFeaturesWSPortType">
- <wsdl:operation name="InsertSequenceFeatures">
  <wsdl:input        message="ns0:InsertSequenceFeaturesRequest"
wsaw:Action="urn:InsertSequenceFeatures" />
  <wsdl:output      message="ns0:InsertSequenceFeaturesResponse"
wsaw:Action="urn:InsertSequenceFeaturesResponse" />
  </wsdl:operation>
  </wsdl:portType>
-   <wsdl:binding   name="InsertSequenceFeaturesWSSOAP11Binding"
type="ns0:InsertSequenceFeaturesWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertSequenceFeatures">
  <soap:operation        soapAction="urn:InsertSequenceFeatures"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
-   <wsdl:binding   name="InsertSequenceFeaturesWSSOAP12Binding"
type="ns0:InsertSequenceFeaturesWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertSequenceFeatures">
  <soap12:operation      soapAction="urn:InsertSequenceFeatures"
style="document" />
```

```
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
-    <wsdl:binding    name="InsertSequenceFeaturesWSHttpBinding"
type="ns0:InsertSequenceFeaturesWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="InsertSequenceFeatures">
  <http:operation
location="InsertSequenceFeaturesWS/InsertSequenceFeatures" />
- <wsdl:input>
  <mime:content type="text/xml" part="InsertSequenceFeatures" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="InsertSequenceFeatures" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InsertSequenceFeaturesWS">
-    <wsdl:port    name="InsertSequenceFeaturesWSSOAP11port_http"
binding="ns0:InsertSequenceFeaturesWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/InsertSequenceFeaturesWS/se
rvices/InsertSequenceFeaturesWS" />
  </wsdl:port>
-    <wsdl:port    name="InsertSequenceFeaturesWSSOAP12port_http"
binding="ns0:InsertSequenceFeaturesWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/InsertSequenceFeaturesWS/se
rvices/InsertSequenceFeaturesWS" />
  </wsdl:port>
-       <wsdl:port       name="InsertSequenceFeaturesWSHttpport"
binding="ns0:InsertSequenceFeaturesWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/InsertSequenceFeaturesWS/se
rvices/InsertSequenceFeaturesWS" />
  </wsdl:port>
  </wsdl:service>
```

```
</wsdl:definitions>
```

InsertBlastSimilarityWS

```
<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="InsertBlastSimilarity">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="condensedFileStr"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="subjectTable" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="subjectTableSrcIdCol"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="subjectExtDBName"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="subjectExtDBRls"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="queryTable" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="queryTableSrcIdCol"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="queryExtDBName"
nillable="true" type="xs:string" />
```

```
  <xs:element minOccurs="0" name="queryExtDBRls" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="InsertBlastSimilarityResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="InsertBlastSimilarityRequest">
  <wsdl:part name="parameters"
element="ns0:InsertBlastSimilarity" />
  </wsdl:message>
- <wsdl:message name="InsertBlastSimilarityResponse">
  <wsdl:part name="parameters"
element="ns0:InsertBlastSimilarityResponse" />
  </wsdl:message>
- <wsdl:portType name="InsertBlastSimilarityWSPortType">
- <wsdl:operation name="InsertBlastSimilarity">
  <wsdl:input message="ns0:InsertBlastSimilarityRequest"
wsaw:Action="urn:InsertBlastSimilarity" />
  <wsdl:output message="ns0:InsertBlastSimilarityResponse"
wsaw:Action="urn:InsertBlastSimilarityResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="InsertBlastSimilarityWSSOAP11Binding"
type="ns0:InsertBlastSimilarityWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertBlastSimilarity">
  <soap:operation soapAction="urn:InsertBlastSimilarity"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
```

```
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertBlastSimilarityWSSOAP12Binding"
type="ns0:InsertBlastSimilarityWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertBlastSimilarity">
  <soap12:operation soapAction="urn:InsertBlastSimilarity"
style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertBlastSimilarityWSHttpBinding"
type="ns0:InsertBlastSimilarityWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="InsertBlastSimilarity">
  <http:operation
location="InsertBlastSimilarityWS/InsertBlastSimilarity" />
- <wsdl:input>
  <mime:content type="text/xml" part="InsertBlastSimilarity" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="InsertBlastSimilarity" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InsertBlastSimilarityWS">
- <wsdl:port name="InsertBlastSimilarityWSSOAP11port_http"
binding="ns0:InsertBlastSimilarityWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/InsertBlastSimilarityWS/ser
vices/InsertBlastSimilarityWS" />
  </wsdl:port>
```

```
- <wsdl:port name="InsertBlastSimilarityWSSOAP12port_http"
binding="ns0:InsertBlastSimilarityWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/InsertBlastSimilarityWS/ser
vices/InsertBlastSimilarityWS" />
  </wsdl:port>
- <wsdl:port name="InsertBlastSimilarityWSHttpport"
binding="ns0:InsertBlastSimilarityWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/InsertBlastSimilarityWS/ser
vices/InsertBlastSimilarityWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

LoadFastaSequencesWS

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="LoadFastaSequences">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="fastaFileStr" nillable="true"
type="xs:string" />
```

```
  <xs:element minOccurs="0" name="ExtDBName" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="ExtDBRls" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="regexSourceId" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="tableName" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="LoadFastaSequencesResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="LoadFastaSequencesRequest">
  <wsdl:part name="parameters" element="ns0:LoadFastaSequences"
/>
  </wsdl:message>
- <wsdl:message name="LoadFastaSequencesResponse">
  <wsdl:part name="parameters"
element="ns0:LoadFastaSequencesResponse" />
  </wsdl:message>
- <wsdl:portType name="LoadFastaSequencesWSPortType">
- <wsdl:operation name="LoadFastaSequences">
  <wsdl:input message="ns0:LoadFastaSequencesRequest"
wsaw:Action="urn:LoadFastaSequences" />
  <wsdl:output message="ns0:LoadFastaSequencesResponse"
wsaw:Action="urn:LoadFastaSequencesResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="LoadFastaSequencesWSSOAP11Binding"
type="ns0:LoadFastaSequencesWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="LoadFastaSequences">
```

```
  <soap:operation soapAction="urn:LoadFastaSequences"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="LoadFastaSequencesWSSOAP12Binding"
type="ns0:LoadFastaSequencesWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="LoadFastaSequences">
  <soap12:operation soapAction="urn:LoadFastaSequences"
style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="LoadFastaSequencesWSHttpBinding"
type="ns0:LoadFastaSequencesWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="LoadFastaSequences">
  <http:operation
location="LoadFastaSequencesWS/LoadFastaSequences" />
- <wsdl:input>
  <mime:content type="text/xml" part="LoadFastaSequences" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="LoadFastaSequences" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="LoadFastaSequencesWS">
```

```
- <wsdl:port name="LoadFastaSequencesWSSOAP11port_http"
binding="ns0:LoadFastaSequencesWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/LoadFastaSequencesWS/servic
es/LoadFastaSequencesWS" />
  </wsdl:port>
- <wsdl:port name="LoadFastaSequencesWSSOAP12port_http"
binding="ns0:LoadFastaSequencesWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/LoadFastaSequencesWS/servic
es/LoadFastaSequencesWS" />
  </wsdl:port>
- <wsdl:port name="LoadFastaSequencesWSHttpport"
binding="ns0:LoadFastaSequencesWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/LoadFastaSequencesWS/servic
es/LoadFastaSequencesWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

InsertSequenceOntologyWS

```
  <?xml version="1.0" encoding="UTF-8" ?>
-                                                    <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
  <wsdl:documentation>Please   Type   your   service   description
here</wsdl:documentation>
- <wsdl:types>
```

```
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="InsertSequenceOntology">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="soFileStr" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="soVersion" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="soCvsVersion" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="InsertSequenceOntologyResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="InsertSequenceOntologyRequest">
  <wsdl:part name="parameters"
element="ns0:InsertSequenceOntology" />
  </wsdl:message>
- <wsdl:message name="InsertSequenceOntologyResponse">
  <wsdl:part name="parameters"
element="ns0:InsertSequenceOntologyResponse" />
  </wsdl:message>
- <wsdl:portType name="InsertSequenceOntologyWSPortType">
- <wsdl:operation name="InsertSequenceOntology">
  <wsdl:input message="ns0:InsertSequenceOntologyRequest"
wsaw:Action="urn:InsertSequenceOntology" />
  <wsdl:output message="ns0:InsertSequenceOntologyResponse"
wsaw:Action="urn:InsertSequenceOntologyResponse" />
  </wsdl:operation>
  </wsdl:portType>
```

```
- <wsdl:binding name="InsertSequenceOntologyWSSOAP11Binding"
type="ns0:InsertSequenceOntologyWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertSequenceOntology">
  <soap:operation soapAction="urn:InsertSequenceOntology"
style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertSequenceOntologyWSSOAP12Binding"
type="ns0:InsertSequenceOntologyWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="InsertSequenceOntology">
  <soap12:operation soapAction="urn:InsertSequenceOntology"
style="document" />
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="InsertSequenceOntologyWSHttpBinding"
type="ns0:InsertSequenceOntologyWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="InsertSequenceOntology">
  <http:operation
location="InsertSequenceOntologyWS/InsertSequenceOntology" />
- <wsdl:input>
  <mime:content type="text/xml" part="InsertSequenceOntology" />
  </wsdl:input>
- <wsdl:output>
  <mime:content type="text/xml" part="InsertSequenceOntology" />
```

```
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="InsertSequenceOntologyWS">
-    <wsdl:port    name="InsertSequenceOntologyWSSOAP11port_http"
binding="ns0:InsertSequenceOntologyWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/InsertSequenceOntologyWS/se
rvices/InsertSequenceOntologyWS" />
  </wsdl:port>
-    <wsdl:port    name="InsertSequenceOntologyWSSOAP12port_http"
binding="ns0:InsertSequenceOntologyWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/InsertSequenceOntologyWS/se
rvices/InsertSequenceOntologyWS" />
  </wsdl:port>
-       <wsdl:port       name="InsertSequenceOntologyWSHttpport"
binding="ns0:InsertSequenceOntologyWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/InsertSequenceOntologyWS/se
rvices/InsertSequenceOntologyWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

LoadTaxonWS

```
 <?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:ns0="http://gus"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://gus">
```

```
  <wsdl:documentation>Please Type your service description
here</wsdl:documentation>
- <wsdl:types>
- <xs:schema xmlns:ns="http://gus"
attributeFormDefault="qualified" elementFormDefault="qualified"
targetNamespace="http://gus">
- <xs:element name="LoadTaxon">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="nodesFileStr" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="namesFileStr" nillable="true"
type="xs:string" />
  <xs:element minOccurs="0" name="gencodeFileStr"
nillable="true" type="xs:string" />
  <xs:element minOccurs="0" name="mergedFileStr" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
- <xs:element name="LoadTaxonResponse">
- <xs:complexType>
- <xs:sequence>
  <xs:element minOccurs="0" name="return" nillable="true"
type="xs:string" />
  </xs:sequence>
  </xs:complexType>
  </xs:element>
  </xs:schema>
  </wsdl:types>
- <wsdl:message name="LoadTaxonRequest">
  <wsdl:part name="parameters" element="ns0:LoadTaxon" />
  </wsdl:message>
- <wsdl:message name="LoadTaxonResponse">
  <wsdl:part name="parameters" element="ns0:LoadTaxonResponse"
/>
  </wsdl:message>
- <wsdl:portType name="LoadTaxonWSPortType">
- <wsdl:operation name="LoadTaxon">
  <wsdl:input message="ns0:LoadTaxonRequest"
wsaw:Action="urn:LoadTaxon" />
```

```
  <wsdl:output message="ns0:LoadTaxonResponse"
wsaw:Action="urn:LoadTaxonResponse" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="LoadTaxonWSSOAP11Binding"
type="ns0:LoadTaxonWSPortType">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="LoadTaxon">
  <soap:operation soapAction="urn:LoadTaxon" style="document" />
- <wsdl:input>
  <soap:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="LoadTaxonWSSOAP12Binding"
type="ns0:LoadTaxonWSPortType">
  <soap12:binding
transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
- <wsdl:operation name="LoadTaxon">
  <soap12:operation soapAction="urn:LoadTaxon" style="document"
/>
- <wsdl:input>
  <soap12:body use="literal" />
  </wsdl:input>
- <wsdl:output>
  <soap12:body use="literal" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:binding name="LoadTaxonWSHttpBinding"
type="ns0:LoadTaxonWSPortType">
  <http:binding verb="POST" />
- <wsdl:operation name="LoadTaxon">
  <http:operation location="LoadTaxonWS/LoadTaxon" />
- <wsdl:input>
  <mime:content type="text/xml" part="LoadTaxon" />
  </wsdl:input>
```

```
- <wsdl:output>
  <mime:content type="text/xml" part="LoadTaxon" />
  </wsdl:output>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="LoadTaxonWS">
- <wsdl:port name="LoadTaxonWSSOAP11port_http"
binding="ns0:LoadTaxonWSSOAP11Binding">
  <soap:address
location="http://157.86.114.129:8080/LoadTaxonWS/services/LoadTa
xonWS" />
  </wsdl:port>
- <wsdl:port name="LoadTaxonWSSOAP12port_http"
binding="ns0:LoadTaxonWSSOAP12Binding">
  <soap12:address
location="http://157.86.114.129:8080/LoadTaxonWS/services/LoadTa
xonWS" />
  </wsdl:port>
- <wsdl:port name="LoadTaxonWSHttpport"
binding="ns0:LoadTaxonWSHttpBinding">
  <http:address
location="http://157.86.114.129:8080/LoadTaxonWS/services/LoadTa
xonWS" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```

Anexo C – Mapeamento Genbank para GUS

Lista de *Features* no Genbank

-10_signal
-35_signal
3'UTR
5'UTR
attenuator
C_region
CAAT_signal
CDS
conflict
D_segment
D-loop
enhancer
exon
gap
GC_signal
gene
iDNA
intron
J_segment
LTR
mat_peptide
misc_binding
misc_difference
misc_feature
misc_recomb
misc_RNA
misc_signal
misc_structure

modified_base

mRNA

N_region

ncRNA

old_sequence

operon

oriT

polyA_signal

polyA_site

precursor_RNA

prim_transcript

primer_bind

promoter

protein_bind

RBS

rep_origin

repeat_region

repeat_unit

rRNA

S_region

satellite

sig_peptide

source

stem_loop

STS

TATA_signal

terminator

tmRNA

transit_peptide

tRNA

unsure

V_region

V_segment

variation

Lista de *Features* no Mapeamento do GUS

-10_signal

-35_signal

3'clip

3'UTR

5'clip

5'UTR

attenuator

C_region

CAAT_signal

CDS

conflict

D_segment

D-loop

enhancer

exon

gap

GC_signal

gene

iDNA

intron

J_segment

LTR

mat_peptide

misc_binding

misc_difference

misc_feature

misc_recomb

misc_RNA

misc_signal

misc_structure

modified_base

mRNA

N_region

old_sequence

operon

oriT

polyA_signal

polyA_site

precursor_RNA

prim_transcript

primer_bind

promoter

protein_bind

RBS

rep_origin

repeat_region

repeat_unit

rRNA

S_region

satellite

scRNA

sig_peptide

snoRNA

snRNA

source

stem_loop

STS

TATA_signal

terminator

transit_peptide

tRNA

unsure

V_region

V_segment

variation

Diferenças entre *Features*

Features apenas no arquivo de mapeamento original do GUS:

3'clip
5'clip
scRNA
snoRNA
snRNA

Features apenas nas Referencias do Genbank:

ncRNA
tmRNA

Arquivo de Mapeamento Padronizado

```
<mapping>
    <specialCaseQualifierHandler

class="GUS::Supported::SpecialCaseQualifierHandlers"
name="standard"/>
    <feature name="attenuator" so="attenuator"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier ignore="true" name="map"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="usedin"/>
```

```
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
    </feature>
    <feature name="C_region" so="C_gene"
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier ignore="true" name="map"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
    </feature>
    <feature name="CAAT_signal" so="CAAT_signal"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier ignore="true" name="map"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
```

```
    <feature name="CDS" so="CDS" table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="codon"/>
        <qualifier ignore="true" name="codon_start"/>
        <qualifier ignore="true" name="EC_number"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="exception"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="number"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="protein_id"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier handler="standard" ignore="true"
method="translation" name="translation"/>
        <qualifier ignore="true" name="transl_except"/>
        <qualifier ignore="true" name="transl_table"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="conflict" so="sequence_difference"
table="DoTS::SeqVariation">
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="compare"/>
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
```

```
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier column="Substitute" ignore="true"
name="replace"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="D-loop" so="D_loop"
table="DoTS::DNAStructure">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="D_segment" so="" soId="SO:0000492"
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
```

```
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="enhancer" so=""
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="bound_moiety"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="label"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="exon" so="exon"
table="DoTS::ExonFeature">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="EC_number"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
```

```
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier column="Num" ignore="true"
name="number"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier handler="standard" ignore="true"
method="product"
            name="product" note="everyone else puts it in
product field"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="gap" so=""
table="DoTS::ScaffoldGapFeature">
        <qualifier handler="standard" ignore="true"
method="gapLength" name="estimated_length"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
    </feature>
    <feature name="GC_signal" so="GC_rich_region"
table="DoTS::DnaRegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
```

```
    </feature>
    <feature name="gene" so="gene"
table="DoTS::GeneFeature">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="iDNA" so="iDNA"
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="number"/>
        <qualifier ignore="true" name="old_locus_tag"/>
```

```
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="intron" so="intron"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="cons_splice"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier column="num" ignore="true"
name="number"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="J_segment"
so="J_gene_recombination_feature"
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
```

```
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="LTR" so="long_terminal_repeat"
table="DoTS::Repeat">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="mat_peptide" so="mature_peptide"
table="DoTS::ProteinFeature">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="EC_number"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
```

```
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_binding" so="binding_site"
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="bound_moiety"/>
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_difference"
so="sequence_difference" table="DoTS::SeqVariation">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="clone"/>
        <qualifier ignore="true" name="compare"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
```

```
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier column="Substitute" ignore="true"
name="replace"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_feature"
so="located_sequence_feature" table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier column="Num" ignore="true"
name="number"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_recomb" so="" soId="SO:0000298"
table="DoTS::SeqVariation">
        <qualifier ignore="true" name="insertion_seq"/>
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
```

```
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="organism"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_RNA" so="transcript"
table="DoTS::RNAStructure">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_signal" so="" soId="SO:0005836"
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
```

```
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="misc_structure"
so="sequence_secondary_structure"
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="modified_base" so="modified_base_site"
table="DoTS::SeqVariation">
        <qualifier ignore="true" name="mod_base"/>
        <qualifier ignore="true" name="allele"/>
```

```
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="frequency"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="mRNA" so="mRNA" table="DoTS::RNAType">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="N_region" so="" soId="SO:0000563"
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
```

```
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="old_sequence"
table="DoTS::SeqVariation">
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="compare"/>
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier column="Substitute" ignore="true"
name="replace"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="operon" so="operon">
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="allele"/>
```

```
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier ignore="true" name="label"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="oriT" so="origin_of_transfer">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="bound_moiety"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="direction"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="rpt_family"/>
        <qualifier ignore="true" name="rpt_type"/>
        <qualifier ignore="true" name="rpt_unit"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier ignore="true" name="direction"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="polyA_signal" so="polyA_signal_sequence"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
```

```
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="polyA_site" so="polyA_site"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="precursor_RNA" so="primary_transcript"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
```

```
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="prim_transcript" so="primary_transcript"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="primer_bind" so="primer_binding_site"
table="DoTS::DNAStructure">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
```

```
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="PCR_conditions"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="promoter" so="promoter"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="bound_moiety"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="protein_bind" so="protein_binding_site"
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="bound_moiety"/>
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
```

```
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="RBS" so="ribosome_entry_site"
table="DoTS::RNAStructure">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="repeat_region" so="repeat_region"
table="DoTS::Repeats">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
```

```
        <qualifier ignore="true" name="insertion_seq"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="rpt_family"/>
        <qualifier ignore="true" name="rpt_type"/>
        <qualifier ignore="true" name="rpt_unit"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="transposon"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="repeat_unit" table="DoTS::Repeats">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="rpt_family"/>
        <qualifier ignore="true" name="rpt_type"/>
        <qualifier ignore="true" name="rpt_unit"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="rep_origin" so="origin_of_replication"
table="DoTS::DNAStructure">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
```

```
            <qualifier ignore="true" name="direction"/>
            <qualifier ignore="true" name="evidence"/>
            <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
            <qualifier ignore="true" name="label"/>
            <qualifier column="source_id" ignore="true"
name="locus_tag"/>
            <qualifier ignore="true" name="map"/>
            <qualifier handler="standard" ignore="true"
method="note" name="note"/>
            <qualifier ignore="true" name="old_locus_tag"/>
            <qualifier ignore="true" name="standard_name"/>
            <qualifier ignore="true" name="usedin"/>
            <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
        </feature>
        <feature name="rRNA" so="rRNA" table="DoTS::RNAType">
            <qualifier ignore="true" name="allele"/>
            <qualifier ignore="true" name="citation"/>
            <qualifier ignore="true" name="evidence"/>
            <qualifier ignore="true" name="function"/>
            <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
            <qualifier ignore="true" name="label"/>
            <qualifier column="source_id" ignore="true"
name="locus_tag"/>
            <qualifier ignore="true" name="map"/>
            <qualifier handler="standard" ignore="true"
method="note" name="note"/>
            <qualifier ignore="true" name="old_locus_tag"/>
            <qualifier ignore="true" name="product"/>
            <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
            <qualifier ignore="true" name="standard_name"/>
            <qualifier ignore="true" name="usedin"/>
            <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
        </feature>
        <feature name="S_region" table="DoTS::Immunoglobulin">
            <qualifier ignore="true" name="allele"/>
            <qualifier ignore="true" name="citation"/>
            <qualifier ignore="true" name="evidence"/>
```

```
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="satellite" so="satellite_DNA"
table="DoTS::Repeats">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="rpt_type"/>
        <qualifier ignore="true" name="rpt_family"/>
        <qualifier ignore="true" name="rpt_unit"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="scRNA" so="scRNA" table="DoTS::RNAType">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
```

```
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="sig_peptide" so="signal_peptide"
table="DoTS::ProteinFeature">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="snRNA" so="snRNA" table="DoTS::RNAType">
        <qualifier ignore="true" name="allele"/>
```

```
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="snoRNA" so="snoRNA"
table="DoTS::RNAType">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
```

```
    </feature>
    <feature name="source" so="" table="DoTS::Source">
        <qualifier ignore="true" name="record"/>
        <qualifier ignore="true" name="organism"/>
        <qualifier ignore="true" name="mol_type"/>
        <qualifier ignore="true" name="cell_line"/>
        <qualifier ignore="true" name="cell_type"/>
        <qualifier ignore="true" name="chromosome"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="clone"/>
        <qualifier ignore="true" name="clone_lib"/>
        <qualifier ignore="true" name="country"/>
        <qualifier ignore="true" name="cultivar"/>
        <qualifier ignore="true" name="dev_stage"/>
        <qualifier ignore="true" name="ecotype"/>
        <qualifier ignore="true"
name="environmental_sample"/>
        <qualifier ignore="true" name="focus"/>
        <qualifier ignore="true" name="frequency"/>
        <qualifier ignore="true" name="germline"/>
        <qualifier ignore="true" name="haplotype"/>
        <qualifier ignore="true" name="lab_host"/>
        <qualifier ignore="true" name="isolate"/>
        <qualifier ignore="true" name="isolation_source"/>
        <qualifier ignore="true" name="label"/>
        <qualifier ignore="true" name="macronuclear"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier handler="standard" ignore="true"
method="organelle" name="organelle"/>
        <qualifier ignore="true" name="plasmid"/>
        <qualifier ignore="true" name="pop_variant"/>
        <qualifier ignore="true" name="proviral"/>
        <qualifier ignore="true" name="rearranged"/>
        <qualifier ignore="true" name="segment"/>
        <qualifier ignore="true" name="serotype"/>
        <qualifier ignore="true" name="serovar"/>
        <qualifier ignore="true" name="sex"/>
        <qualifier ignore="true" name="specimen_voucher"/>
        <qualifier ignore="true" name="specific_host"/>
        <qualifier ignore="true" name="strain"/>
```

```
        <qualifier ignore="true" name="sub_clone"/>
        <qualifier ignore="true" name="sub_species"/>
        <qualifier ignore="true" name="sub_strain"/>
        <qualifier ignore="true" name="tissue_lib"/>
        <qualifier ignore="true" name="tissue_type"/>
        <qualifier ignore="true" name="transgenic"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier ignore="true" name="variety"/>
        <qualifier ignore="true" name="virion"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="stem_loop" so="stem_loop"
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="STS" so="STS" table="DoTS::STS">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
```

```
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="TATA_signal" so="TATA_box"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="terminator" so="terminator"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
```

```
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="transit_peptide" so="transit_peptide"
table="DoTS::ProteinFeature">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="tRNA" so="tRNA" table="DoTS::RNAType">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="anticodon"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
```

```
        <qualifier ignore="true" name="product"/>
        <qualifier column="is_pseudo" ignore="true"
name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="unsure" so=""
table="DoTS::Miscellaneous">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="compare"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="replace"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="V_region" so=""
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
```

```
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="V_segment" so=""
table="DoTS::Immunoglobulin">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="product"/>
        <qualifier ignore="true" name="pseudo"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="variation" so="sequence_variant"
table="DoTS::SeqVariation">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="compare"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="frequency"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
```

```
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="phenotype"/>
        <qualifier ignore="true" name="product"/>
        <qualifier column="Substitute" ignore="true"
name="replace"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="3'clip" so="three_prime_clip"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="3'UTR" so="three_prime_UTR"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
```

```
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="5'clip" so="five_prime_clip"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="5'UTR" so="five_prime_UTR"
table="DoTS::Transcript">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier ignore="true" name="function"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
```

```
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="-10_signal" so="minus_10_signal"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
        <qualifier ignore="true" name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
    <feature name="-35_signal" so="minus_35_signal"
table="DoTS::DNARegulatory">
        <qualifier ignore="true" name="allele"/>
        <qualifier ignore="true" name="citation"/>
        <qualifier ignore="true" name="evidence"/>
        <qualifier handler="standard" ignore="true"
method="gene" name="gene"/>
        <qualifier ignore="true" name="label"/>
        <qualifier column="source_id" ignore="true"
name="locus_tag"/>
        <qualifier ignore="true" name="map"/>
        <qualifier handler="standard" ignore="true"
method="note" name="note"/>
```

```
        <qualifier column="source_id" ignore="true"
name="old_locus_tag"/>
        <qualifier ignore="true" name="operon"/>
        <qualifier ignore="true" name="standard_name"/>
        <qualifier ignore="true" name="usedin"/>
        <qualifier handler="standard" ignore="true"
method="dbXRef" name="db_xref"/>
    </feature>
</mapping>
```