UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GROMAXY: UMA FERRAMENTA PARA INTEGRAÇÃO DO GALAXY COM O GROMACS

ALFREDO GUILHERME DA SILVA SOUZA

ORIENTADOR: PROF. DR. LUIS CARLOS TREVELIN

São Carlos – SP

Agosto/2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GROMAXY: UMA FERRAMENTA PARA INTEGRAÇÃO DO GALAXY COM O GROMACS

ALFREDO GUILHERME DA SILVA SOUZA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Redes de Computadores e Sistemas Distribuídos
Orientador: Prof. Dr. Luis Carlos Trevelin

São Carlos – SP

Agosto/2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Alfredo Guilherme da Silva Souza, realizada em 08/08/2017:

Prof. Dr. Luis Carlos Trevelin
UFSCar

Prof. Dr. Julio Zukerman Schpector
UFSCar

Prof. Dr. Marcelo de Paiva Guimarães
UNIFESP

A Deus e minha mãe, Eliana Fátima.

AGRADECIMENTOS

A minha família, em especial minha mãe, que mesmo com todas as dificuldades que enfrentou ao longo de sua vida, me deu todo o suporte necessário para chegar até aqui, sem ela seria praticamente impossível traçar todos os caminhos necessários.

Ao meu orientador, Prof. Dr. Luís Carlos Trevelin, pela oportunidade, confiança, orientação, amizade, paciência e pelo incentivo. Agradeço também pelos conselhos que me deu ao longo do mestrado, não somente olhando no meu sucesso acadêmico, mas conselhos para carregar por toda minha vida. Serei eternamente grato por tudo.

Aos meus amigos do laboratório LaVIIC (Laboratório de Visualização Imersiva, Interativa e Colaborativa): Diego, Alexandre, Glésio, Jéssica, Jordam e João Paulo, por compartilharem comigo todo o conhecimento, a amizade e os momentos descontraídos durante o café.

Ao Prof. Dr. Júlio Zukerman Schpector, por abrir as portas de seu laboratório, LaCrEMM (Laboratório de Cristalografia, Estereodinâmica e Modelagem Molecular) para ajudar nesse trabalho. Agradeço também a sua aluna Dr. Stella Hernandez Maganhi, membro de seu laboratório, a quem serei eternamente grato, por ter participado de uma importante etapa desse trabalho. Sempre disposta a me ajudar, com toda a calma, educação e sorriso no rosto, me mostrou o caminho quando eu mais precisei e mais estava perdido.

Aos meus irmãos da República Etanóis, que me receberam de braços abertos quando cheguei em São Carlos, e que mesmo nos momentos de solidão estavam ali pra mostrar que eu não estava sozinho. Com eles eu vivi dias tristes, mas também vivi os melhores dias da minha vida.

Aos meus amigos Elias ”Baiano”, Amir, Rafael ”Sanches”, Cédrick ”Bamba”, Kleberson ”Man”, Jeffrey e Renato, pelos inesquecíveis momentos, e permitirem criar entre a gente um forte laço de amizade.

A todos funcionários, professores e amigos do PPGCC, que de alguma forma contribuíram para esta Dissertação. Agradeço também pela amizade.

Ao CNPq, pelo apoio financeiro.

Se andarmos pelos caminhos que outros já percorreram, chegaremos no máximo aos lugares que eles já atingiram.

Augusto Cury

RESUMO

Com o avanço dos estudos biomoleculares é previsto o surgimento de várias limitações e, com isso torna necessário o fortalecimento da aliança entre a ciência e a tecnologia, trazendo como base o desenvolvimento de novas ferramentas e métodos eficazes para auxiliar os diversos tipos de estudos, como o uso do método de Dinâmica Molecular para auxiliar no processo de pesquisas com foco na descoberta de fármacos. Esse método possibilita realizar simulações nas quais podem ser observadas ao longo do tempo o comportamento dos átomos que constituem a biomolécula em estudo. Na literatura podem ser encontrados diversos pacotes de ferramentas que possibilitam a realização de simulações de dinâmica molecular. O GROMACS é um desses e se destaca por ser bastante utilizado e *open-source*. Baseado na complexidade em manipular o GROMACS, este trabalho apresenta uma ferramenta chamada GromaXy, que tem como objetivo prover ao usuário, uma nova forma de interação com o GROMACS, oferecendo a ele uma ferramenta com interface gráfica, diferente da interface de linha de comandos do GROMACS. Além de uma nova forma de interação com o GROMACS, o GromaXy também oferece uma adaptação do Atom Validation, que é uma ferramenta encontrada na literatura e que permite a validação de arquivos PDB. O Atom Validation durante sua execução identifica os átomos ausentes da estrutura da proteína e apresenta uma lista desses para o usuário. O GromaXy foi desenvolvido por meio de um *framework web* bem difundido na Bioinformática chamado de Galaxy, que foi adotado neste trabalho por possuir um ambiente simples para a utilização, possibilitar o compartilhamento de histórico de trabalho, dados de experimentos realizados, e possibilitar o armazenamento do alto volume de dados de trajetória resultantes do processo de simulação de dinâmica molecular. O nome GromaXy foi adotado por fazer uma junção ao nome do GROMACS e o nome do Galaxy.

Palavras-chave: Dinâmica Molecular, GROMACS, Galaxy, Bioinformatica, Proteína.

ABSTRACT

Considering the significant advances in biomolecular researches, the raise of various limitations is predicted. Thereby, reinforcing the link between science and technology, by introducing the new and effective development tools which can be used in a variety of scientific studies (i.e. Molecular Dynamics methods for drug discovery), becomes necessary. This method enables the realization of simulations in biomolecular studies in which atoms' behavior can be observed during a specific period of time. In the state of art various researches related to development of tools, whose main goal is facilitating Molecular Dynamics studies, are encountered. The GROMACS is one of these tools that is known as an open-source tool that has been used in lots of research projects. Due to the complexity of manipulating the GROMACS, the current study presents a new tool called GromaXy, which aims at providing a new interaction way with GROMACS, by introducing a new graphical interface which is much more convenient than the GROMACS command line. In addition to providing a new interaction way with GROMACS, the GromaXy also offers an adoptation of Atom Validation, which is another tool that has been developed in one of the previous studies. Atom Validation identifies the absent atoms in protein structure and show the to the user in form of a list. GromaXy was developed using a well-known framework in bioinformatic area called Galaxy. This framework was preferred due to its simple environment, the ability of sharing the work history and the history of realized experiments, enabling the high-level storage of data related to the trajectory of simulation process. The GromaXy was chosen as an adequate name which presents the join of Galaxy and GROMACS.

Keywords: Molecular Dynamics, GROMACS, Galaxy, Bioinformatics, Protein.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

Com o crescimento na determinação de estruturas moleculares de biomoléculas, as ciências biomédicas estão passando por um desenvolvimento revolucionário (ZELLER et al., 1997). Desenvolvimento este que só está sendo possível com o uso de computadores de alto desempenho, e poderosas ferramentas computacionais, que podem ser utilizadas desde simulações até visualizações de complexas estruturas moleculares.

Entre os diferentes tipos de biomoléculas, as proteínas são encontradas com grande abundância e diversidade nos seres vivos em geral, estando presentes no organismo humano e atuando em diversos tipos de funções. Funções essas que estão fortemente ligadas a sua estrutura, ao processo de sua formação e ao seu comportamento com o meio que está inserida (NELSON; COX, 2014) (VOET; VOET, 2011).

A composição da proteína é dada pela ligação linear de compostos orgânicos chamados aminoácidos. Estes compostos possuem estruturas moleculares semelhantes entre si, sendo diferenciados apenas por sua cadeia lateral, também conhecida como radical R. As ligações de aminoácidos que ocorrem para a formação inicial da proteína são chamadas de ligações peptídicas (NELSON; COX, 2014) (VOET; VOET, 2011).

Como os aminoácidos são formados por ligações entre átomos e a proteína também é constituída por um conjunto de átomos, é a interação entre esses átomos e o meio inserido que a estrutura da proteína é definida.

Entre todos possíveis métodos que podem ser utilizados para estudar interações atômicas de um sistema, a Dinâmica Molecular (DM) é uma técnica que faz aplicação de cálculos físicos com a finalidade de determinar a evolução do sistema ao longo do tempo (RAPAPORT, 2004).

Durante o processo de simulação de DM, são capturados os dados sobre as posições e velocidades de cada partícula (átomos nessa Dissertação) da biomolécula (proteína nessa Dissertação) simulada. Estes dados são chamados de *trajetória* da simulação (MORGON; COUTINHO, 2007) (NAMBA; SILVA; SILVA, 2008). Por meio da trajetória que é possível visualizar o comportamento da biomolécula simulada. Para visualizar graficamente a *trajetória* da simulação é necessário o uso de um software com tal finalidade.

As estruturas moleculares de proteínas são resolvidas através de técnicas como: cristalografia, ressonância magnética nuclear, e são depositadas em um banco de dados de proteínas chamado de *Protein Data Bank* (banco de dados PDB), deixando disponível aos pesquisadores o modelo da proteína a ser utilizada por meio do formato de arquivo .pdb (arquivo PDB) (BERMAN et al., 2000). Nesse trabalho são usadas duas notações parecidas, e para não gerar conflitos de interpretações, são definidas a seguir:

- Arquivo PDB: arquivo na extensão .pdb; e
- Banco de dados PDB: *Protein Data Bank*.

Um arquivo PDB contém as informações de cada átomo que compõe a proteína, bem como os seus respectivos nomes, suas coordenadas, ligações que faz com outros átomos e outras informações (BERMAN et al., 2000). Os dados da proteína a ser simulada computacionalmente devem então ser extraídos do PDB, pois por meio das coordenadas de cada átomo é que se determina o posicionamento inicial de cada átomo no sistema a ser simulado.

1.2 Motivação

Para a realização de simulações de DM podem ser utilizados diversos programas, e esses são agrupados em um pacote. Atualmente há uma grande variedade de pacotes para a realização de simulações, que possibilitam tratar diversos tipos de problemas utilizando diferentes técnicas e métodos (MORGON; COUTINHO, 2007). Como citado em Makarewicz e Kazmierkiewicz (2013), em sua maioria as ferramentas de simulação de biomoléculas carecem de uma interface gráfica de usuário (GUI). Esta carência reflete na complexidade de manuseio destes pacotes por parte do usuário que pode não ter um bom conhecimento computacional, uma vez que sua utilização depende do manuseio por meio de linhas de comandos contendo vários parâmetros e a execução sequencial de vários programas, de forma que o *output* de um programa é *input* para o próximo programa. Entre os diversos pacotes encontrados, o GROMACS (APOL et al., 2010) (SPOEL

et al., 2005) se destaca por ser um dos mais utilizados no meio acadêmico e por ser *open-source*. Sendo esses motivos ele foi escolhido para ser utilizado nessa Dissertação, podendo em trabalhos futuros possibilitar o uso de diferentes pacotes.

Atualmente existem projetos que oferecem um ambiente gráfico para trabalhar com o GROMACS, porém todas as ferramentas encontradas não fazem parte do seu projeto padrão, sendo ferramentas desenvolvidas por terceiros. Todas estas ferramentas gráficas encontradas oferecem uma nova forma de interação com o GROMACS, o que pode simplificar a manipulação de todos os comandos necessários para a sua execução.

As ferramentas gráficas disponíveis são encontradas somente em versões para desktop, desta forma elas são executadas somente localmente no computador do usuário, exigindo sua instalação e atualização. Durante a pesquisa também foi encontrado um serviço robusto de simulação chamado Rescale (Rescale, 2014). Este é um serviço que dispensa a realização do processamento local na máquina do usuário, pois ele realiza todo o processo de simulação em nuvem. Embora o Rescale seja um serviço de alto desempenho em nuvem, ele não disponibiliza para o usuário uma interface na qual ele possa configurar e executar uma simulação manualmente. Diferente dos aplicativos para o GROMACS, este serviço é pago, em virtude do mesmo oferecer um ambiente para executar uma simulação.

Durante o desenvolvimento dessa Dissertação, foi identificado junto ao usuário, uma grande dificuldade em iniciar uma simulação de DM no GROMACS, pois raramente a estrutura da proteína que está contida no arquivo PDB, está completa, ou seja, muitas vezes há a ausência de átomos em sua estrutura.

O programa pdb2gmx do GROMACS, é responsável por fazer leitura do arquivo PDB e gerar um arquivo de saída na extensão .gro. Quando o pdb2gmx identifica que a estrutura da proteína está incompleta, ele impede que o usuário dê sequencia ao seu trabalho, gerando um erro e apresentando o átomo ausente da estrutura. Como esse programa interrompe sua execução para cada átomo identificado, esse processo pode se tornar-se muito repetitivo, tomando bastante tempo o do usuário, e esse tempo pode ser relativo ao tamanho da proteína e quantidade de átomos ausentes em sua estrutura.

1.3 Objetivos

Baseado na complexidade de utilização de pacotes de simulação de DM, a falta de ferramentas gráficas para trabalhar com estes pacotes, e a dificuldade de reprodução de experimentos realizados por outros pesquisadores, essa Dissertação apresenta uma ferramenta que provê uma

nova forma de interação do usuário com o GROMACS, junto ao ambiente do Galaxy, neste caso unindo o GROMACS ao Galaxy, e desta junção se dá também o nome da ferramenta desenvolvida: GromaXy.

Essa Dissertação também apresenta uma ferramenta que permite a validação de arquivos PDB. Essa ferramenta é uma adaptação de uma ferramenta já existente na literatura, o Atom Validation (LEITE, 2012). O Atom Validation permite que usuário valide uma proteína usando o campo de força CHARMM27, porém para essa Dissertação, o *software* foi modificado para que suporte a escolha de quaisquer campos de força do GROMACS.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos necessários para a compreensão dessa Dissertação, bem como os conceitos básicos de DM, a ferramenta Atom Validation, apresentação do framework adotado para o desenvolvimento do GromaXy e conceitos básicos de Usabilidade.

2.1 Conceitos de Dinâmica Molecular

A DM é uma técnica computacional que torna possível a realização de simulações, possibilitando analisar o comportamento dos átomos, computando suas interações ao longo do tempo por meio de equações de movimento de Newton, onde a cada passo de interação são capturados dados sobre as posições e velocidades de cada partícula (átomos nessa Dissertação) da biomolécula simulada, gerando ao longo do tempo a evolução temporal do sistema, ao qual é chamada de *trajetória* (RAPAPORT, 2004) (MORGON; COUTINHO, 2007) (NAMBA; SILVA; SILVA, 2008) .

As interações moleculares são representadas por meio de equações, que são parametrizadas pelo chamado Campo de Força, sendo assim a compressão destas interações é de grande importância, até para a escolha do campo de força mais adequado para cada tipo de situação.

2.1.1 Campos de força

O conjunto completo de parâmetros necessários para descrever os potenciais de interações entre os átomos pode ser definido como Campo de Força. Estes potenciais podem ser representados, por exemplo, entre a soma de interação entre duas moléculas, que pode ser obtida por meio da soma de movimentos internos da molécula e o potencial de interações intramolecula-

res que é chamada de Energia Potencial. Energia esta que está representada na Equação 2.1. Este fator citado está relacionado com a deformação do arranjo tridimensional dos átomos que compõem a biomolécula em estudo (MORGON; COUTINHO, 2007).

$$V_{total} = \sum V_{inter} + \sum V_{intra} \tag{2.1}$$

A parametrização do campo de força para a realização da simulação deve ser realizada conforme a necessidade do sistema em estudo, pois cada propriedade apresenta sua função no sistema (MORGON; COUTINHO, 2007).

Alguns dos campos de força que podem ser encontrados no GROMACS são: **amber03, amber94, amber96, amber99, amber99sb, amber99sb-ildn, amberGS, charmm27, encads, encadv, gmx2, gmx, gromos43a1,gromos43a2,gromos53a5,gromos53a6, gromos54a7, opls-saa**. Todos esses foram disponibilizados ao usuário para que ele possa escolher para validar suas proteínas. Todos esses campos de força podem ser encontrados dentro de um diretório de topologia no GROMACS, ao exemplo o diretório **/gromacs/top/** no sistema operacional Linux. Cada campo de força é descrito em um arquivo e este tem a extensão **.ff**, que é a abreviação de *Force Field*, tradução do nome Campo de Força.

Com toda a descrição realizada é compreensível que o resultado das trajetórias do sistema em estudo está fortemente ligado à parametrização do campo de força, que define os potenciais de interação entre os átomos, já que a evolução do sistema depende das forças atuantes sobre cada átomo.

2.1.2 Configuração inicial do sistema

O processo de simulação é realizado dentro de uma caixa, que geralmente é cúbica mas também podem ser utilizadas diferentes geometrias como: paralelepipédica ou octaédrica truncada. Esses são modelos usuais apenas para a compreensão, ou seja, apenas representações. Após definida a geometria é especificado o posicionamento inicial de cada átomo do sistema, definindo o posicionamento de forma que cada átomo tenha seu espaço, buscando não afetar o espaço de outros (NAMBA; SILVA; SILVA, 2008).

2.1.3 Minimização de Energia

A Minimização de Energia é um processo também conhecido como otimização de energia, que busca ao processo de partida encontrar uma posição geométrica adequada com o mínimo de energia sobre cada átomo, com a finalidade também de diminuir a energia potencial do sistema. Também necessário para eliminar os maus contatos entre os átomos, este processo é realizado na forma de um procedimento de caminhada sobre a superfície de potencial buscando o caminho na qual a energia vai decrescendo, organizando os átomos até alcançar o mínimo de energia (NAMBA; SILVA; SILVA, 2008).

2.1.4 *Constraints*

Visando a busca por eficiência na simulação, é possível aplicar regras de limitações (*constraints*) e restrições (*restraints*) que tem por objetivo identificar os movimentos que ocorrem com muita frequência, como: vibrações das ligações de átomos ou mesmo flexão dos ângulos de ligação dos átomos, e estabilizar estes movimentos de maneira que não ocorra alteração na trajetória dos átomos do sistema. Com a aplicação destas técnicas é possível aumentar o espaço do tempo usado nas equações, já que os movimentos que ocorrem com muita frequência estão presentes na maioria dos espaços de tempo analisados (FRENKEL; SMIT, 2001).

2.1.5 Pacotes de Dinâmica Molecular

A popularização dos métodos de DM está fortemente relacionada ao crescente desenvolvimento de pacotes de simulação de DM que traz avanços para pesquisas na área de Química Teórica, laboratórios experimentais, centros de pesquisas farmacêutica e também na área de materiais (MORGON; COUTINHO, 2007). Este crescente desenvolvimento de pacotes de DM acarreta na diversidade de pacotes de software para DM, onde cada um é desenvolvido com o objetivo de possuir suas próprias características de forma a atender um cenário específico (ADCOCK; MCCAMMON, 2006). Como já citado, a técnica de simulação de DM pode ser utilizada não somente para simulação biomoléculas, mas como também diversos outros tipos de moléculas. Nessa Dissertação o foco do uso de DM é apenas em simulação com proteínas.

2.1.5.1 Problemas com usabilidade

Entre os pacotes de DM mais difundidos na literatura é possível identificar problemas com dificuldade de uso, uma vez que boa parte dos programas que compõem um pacote de DM,

exigem do usuário um conhecimento preliminar em linha de comandos, por algumas situações até conhecimento básico em programação de computadores, que torna necessário para a escrita de *scripts* de execução, que são responsáveis por guiar todo o processo de configuração e simulação de DM. Por exemplo, é possível que o usuário escreva um *script* usando a linguagem Shell para determinar todo o processo de simulação, como os programas que deverão ser utilizados em ordens específicas e com arquivos de entradas (*input*) e saídas (*output*) específicos

Todas as ferramentas de pacotes de DM geralmente são invocadas pelo terminal, necessitando especificar para cada ferramenta invocada, os arquivos de entrada (*input*), arquivos de parametrização (.mdp) e arquivos de saída (*output*) que são comumente utilizados como arquivo de entrada também para o próxima ferramenta a ser invocada. Este processo sequencial que justifica o desenvolvimento de um *script* de execução, e esta não é uma tarefa simples para um usuário comum. Por estes e outros motivos conforme citado em Makarewicz e Kazmierkiewicz (2013), as ferramentas de simulações de biomoléculas carecem de uma interface gráfica de usuário (GUI). Enquanto em Knapp e Schreiner (2009) é destacado que a adoção de interfaces gráficas para software de DM pode trazer um acesso mais amplo e auxiliar o aumento do nível de usabilidade.

A **Usabilidade** está ligada a **facilidade de uso** de um sistema, o quanto é fácil para o usuário interagir com este sistema (NIELSEN, 1994). Além desta, é possível encontrar na literatura outras definições de usabilidade. Este assunto será melhor abordado na Seção 2.5.

2.2 Galaxy

O Galaxy é uma plataforma web que tem como principal objetivo tornar simples o manuseio de diversas ferramentas para a utilização, ferramentas que são complexas para o manuseio, configuração e instalação por um usuário comum (AFGAN et al., 2016) (GOECKS et al., 2010). A Figura 2.1 apresenta o seu ambiente de trabalho, possibilitando verificar que a direita fica o histórico dos trabalhos realizados e a esquerda o seu conjunto de ferramentas. Também possui um painel superior que contém os controles; neste local são encontradas opções para o usuário se conectar ao Galaxy, uma documentação para ajudar o usuário em suas dúvidas, entre outros recursos. O painel exibe o conjunto de objetivos que compõe a ferramenta selecionada para utilização.

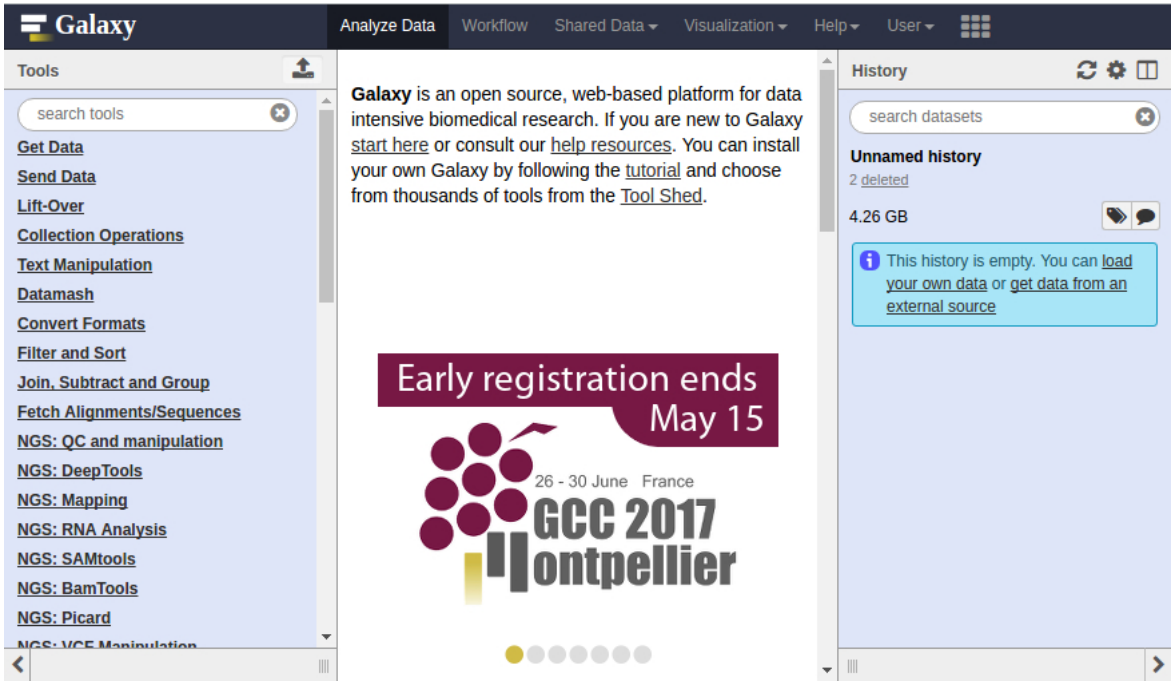

Figura 2.1: Ambiente de trabalho do Galaxy

Considerando que hoje um dos grandes problemas para a comunidade científica é a reprodução de experimentos já realizados por falta de documentação, padronização, ferramentas complexas para utilização e utilização de grandes conjuntos de ferramentas e dados necessários para o processamento, o Galaxy tem como característica positiva possibilitar a reprodução de experimentos, tendo como diferencial sua produção de dados, pois ele permite o armazenamento de dados individuais separados por usuário, de forma que o usuário possa configurá-lo para executar todo processamento necessário e depois utilizar estes dados em análises para preparar documentos para publicação (AFGAN et al., 2016) (GOECKS et al., 2010).

Outra característica importante do Galaxy é o fato dele ser um serviço web, podendo ser acessado de qualquer lugar, até mesmo por um dispositivo móvel, desta forma as ferramentas são disponíveis a qualquer pesquisador, desconsiderando a necessidade de instalação de qualquer programa localmente em sua máquina, ou até mesmo ter um conhecimento avançando sobre as ferramentas dispostas na plataforma do Galaxy (AFGAN et al., 2016) (GOECKS et al., 2010).

A plataforma do Galaxy é *open-source*, e esta disponibiliza seu ambiente para usuários e desenvolvedores, possibilitando efetuar o download de seu aplicativo, realizar implantação em qualquer ambiente computacional, desde um super computador a um computador pessoal. No Galaxy desenvolvedores podem desenvolver novas ferramentas ou até realizar modificações em suas ferramentas já existentes de maneira flexível, de forma a atender suas necessidades (AFGAN

et al., 2016) (GOECKS et al., 2010).

Além de todos seus pontos positivos há uma grande comunidade ativa de desenvolvedores espalhados por vários países que utilizam do Galaxy para seus projetos de Bioinformática e até projetos de outras áreas, por ele já possuir uma estrutura bem definida e já estar consolidado no meio acadêmico. Este é um fator importante para a vida e evolução do projeto, já que qualquer problema encontrado durante o desenvolvimento pode ser solucionado com a ajuda de membros da comunidade, além da contribuição de todos para fazer do Galaxy sempre um *framework* melhor, e cada vez mais difundido no meio.

Alguns grupos de pesquisas espalhados por Universidades de vários países utilizam do Galaxy para atender suas mais variadas necessidades. Na Tabela 2.1 são apresentados alguns dos projetos que foram desenvolvidos por estes grupos e disponibilizados para uso de forma pública.

Tabela 2.1: Apresentação de diferentes projetos públicos utilizando o Galaxy

Início da Tabela	
Projeto	Link
Biomina	http://biominavm-galaxy.biomina.be/galaxy/
CBiB Galaxy	http://services.cbib.u-bordeaux2.fr/galaxy/
DBCLS Galaxy	http://galaxy.dbcls.jp/
GalaxEast	http://www.galaxeast.fr/
Genboree	http://www.genboree.org/galaxy/
GigaGalaxy	http://galaxy.cbiit.cuhk.edu.hk/
GVL QLD	http://galaxy-qld.genome.edu.au/
GVL Tutorial	http://galaxy-tut.genome.edu.au/
INRA-URGI	http://urgi.versailles.inra.fr/galaxy
NELLY	http://www.bioinformatica.ucr.ac.cr:8080/
Pitagora-Galaxy	http://try.pitagora-galaxy.org/galaxy/
ballaxy	https://ballaxy.bioinf.uni-sb.de/
CAPER	http://www.bprc.ac.cn/CAPE
CardioVascular Research Grid (CVRG)	http://cvrg.galaxycloud.org/
Center for Phage Technology (CPT)	https://cpt.tamu.edu/galaxy-pub/
Cistrome Analysis Pipeline	http://cistrome.org/ap/root
CNIC.DarwinTree	http://galaxy.csdb.cn/
CoSSci	http://socscicompute.ss.uci.edu/
Galaxy-P	https://usegalaxyp.org/

Continuação da Tabela	
Projeto	Link
Galaxy PGTB (Virtual Biodiversity Lab)	https://galaxy-pgtp.pierroton.inra.fr/
Genomic Hyperbrowser	http://hyperbrowser.uio.no/hb/
Gene Ontololgy (GO)	http://galaxy.berkeleybop.org/
Image Analysis and Processing Toolkit	http://cloudimaging.net.au/
International Rice Research Institute (IRRI) Galaxy	http://175.41.147.71:8080/
MetaNET	http://metanet.osdd.net/
MISSISSIPPI	http://mississippi.fr/
Nebula	http://nebula.curie.fr/
Oqtans	https://galaxy.cbio.mskcc.org/
Orione	http://orione.crs4.it/
OSDDlinux LiveGalaxy	http://osddlinux.osdd.net:8001/
OSDD Molecular Property Diagnostic Suite (MPDS)	http://mpds.stage2.osdd.net/
PopGenIE / PlantGenIE	http://galaxy.plantgenie.org:8080/
RepeatExplorer	http://repeatexplorer.umbr.cas.cz/
ReproGenomics Viewer	http://rgv.genouest.org/galaxy/
RiboGalaxy	http://ribogalaxy.ucc.ie/
RNA-Rocket @ Pathogen Portal	http://rnaseq.pathogenportal.org/
Stem Cell Discovery Engine	http://discovery.hsci.harvard.edu/galaxy
South Green	http://gohelle.cirad.fr/galaxy/
VectorBase Galaxy	https://www.vectorbase.org/galaxy
VirAmp	http://viramp.com/
Whale Shark	http://whaleshark.georgiaaquarium.org/
Workflow4Metabolomics	http://galaxy.workflow4metabolomics.org/
ABiMS Tools	http://webtools.sb-roscoff.fr/
AB-OpenLab	http://ab-openlab.csir.res.in/frog/
AGEseq @ AspenDB	http://aspendb.uga.edu/ageseq
BioCiphers Lab Galaxy	http://avispa.biociphers.org/
BioMaS	http://galaxy.cloud.ba.infn.it:8080/
CTMM-TraIT Demo Galaxy	http://galaxy.ctmm-trait.nl/
deepTools	http://deeptools.ie-freiburg.mpg.de/
Dostie Lab Galaxy	http://galaxy.bci.mcgill.ca/
EpiToolKit	http://www.epitoolkit.de/
Fast UniFrac	http://unifrac.colorado.edu/

Continuação da Tabela	
Projeto	Link
Huttenhower Lab	http://huttenhower.sph.harvard.edu/galaxy/
IM-PET	http://impet.int-med.uiowa.edu/
In Silico Galaxy	http://insilico.utulsa.edu/galaxy/
kmer-SVM	http://kmersvm.beerlab.org/
LiSIs	http://lisis.cs.ucy.ac.cy/
Majewski Lab	http://genomequebec.mcgill.ca/exomeai
Martin Luther U. Halle-Wittenberg	http://galaxy.informatik.uni-halle.de/
MBAC Metabiome Portal	http://mbac.gmu.edu:8080/
MIRPIPE	https://bioinformatics.mpi-bn.mpg.de/
NGS-QC	http://galaxy.ngs-qc.org/
ODoSE	http://www.odose.nl/
OPPL Galaxy	http://biordf.org:8983/
Osiris	http://galaxy-dev.cnsi.ucsb.edu/osiris/
P-Galaxy	http://p-galaxy.ddbj.nig.ac.jp/
PIA	http://galaxy-dev.cnsi.ucsb.edu/pia/
PredPharmTox	http://galaxy.predpharmtox.org/
PreSTIGE	http://prestige.case.edu/
QBRC Galaxy and PIPE-CLIP	http://www.galaxy.qbrc.org/
SIFTED	http://thebrain.bwh.harvard.edu/sifted.html
SymD	http://symd.nci.nih.gov/
Vinther Lab	http://galaxy.bio.ku.dk/
Wageningen UR	http://galaxy.wur.nl/
ZBIT Bioinformatics Toolbox	http://webservices.cs.uni-tuebingen.de/
Final da Tabela	

A Tabela 2.1 apresentou alguns dos projetos disponíveis publicamente para uso, porém nem todos projetos desenvolvidos com o Galaxy precisam necessariamente de serem disponibilizados para uso geral, podendo desta forma haver outros projetos que foram ou estão sendo desenvolvidos utilizando o Galaxy. Este lista de projetos foi apresentada apenas para ilustrar o quanto o Galaxy já está difundido em projetos de Bioinformática.

2.3 GROMACS

O GROMACS é um pacote de simulação que permite realizar simulação de DM e minimização de energia (SPOEL et al., 2005). Ele é um software livre, disponível sob a licença GNU (*General Public License*), dessa forma o usuário tem total liberdade para modificar e distribuir o GROMACS, podendo alterar conforme sua necessidade e até realizar melhorias em seus programas, estando disponível gratuitamente a qualquer pessoa, podendo ser utilizado em qualquer computador pessoal, como um laptop ou desktop.

Esse pacote utiliza os princípios de equação de movimento de Newton para estudar os movimentos e interações entre os átomos do sistema em estudo. Essas equações são realizadas em pequenos espaços de tempo, de modo que torna possível analisar o comportamento de cada átomo de tempo gerando a sua trajetória no sistema (SPOEL et al., 2005).

2.4 Atom Validation

Durante o processo de determinação da estrutura de uma proteína, pode não ser possível identificar parte de sua estrutura, o que ocasiona em um modelo de proteína incompleta, ou seja, faltando átomos. O *Atom Validation* é um software que tem como proposta possibilitar que o usuário valide um arquivo PDB, afim de identificar quais átomos podem estar faltando na estrutura da proteína. Este software utiliza o campo de força Charmm27 que está presente no GROMACS, no seu processo de identificação de átomos ausentes na estrutura (LEITE, 2012).

Durante o processo simulação de DM usando o GROMACS, o programa pdb2gmx que faz parte do pacote do GROMACS, faz leitura e validação do arquivo PDB que foi dado como *input*, e ao identificar o primeiro átomo ausente o programa pausa o processo e informa o usuário a falta do átomo em questão, para que esse possa inserí-lo no arquivo PDB e tentar novamente. Este processo pode ter um custo cognitivo elevado, já que em uma proteína boa parte de sua estrutura pode estar incompleta. Para esse caso o usuário teria que efetuar o mesmo processo muitas vezes, o que te tomaria muito tempo.

Diferente do programa pdb2gmx, o *Atom Validation* realiza todo o processo de validação de uma só vez e apresenta uma lista com os resíduos que estão incompletos. O pdb2gmx valida um resíduo por vez, necessitando o usuário executar o programa toda vez que a última execução detectou que a estrutura da proteína está incompleta (LEITE, 2012).

A interface do programa *Atom Validation* é simples, contendo um menu com as instruções necessárias para sua manipulação. Além da opção que permite informar o arquivo PDB que

será utilizado, o *Atom Validation* permite também ao usuário definir se ele deseja mostrar ou omitir os átomos de hidrogênio durante a investigação (LEITE, 2012). A Figura abaixo apresenta o menu principal do *software*.

```
This program requires an input file with
an extension .pdb to start the protein validation,
with options to visualize or hide the hydrogen atoms.
example: AtomValidation.py <path and filename.pdb> [option]
options:
-v: visualize hydrogen
-o: hide hydrogen
```

Figura 2.2: Menu do *Atom Validation*

(LEITE, 2012)

2.5 Usabilidade

A usabilidade está relacionada com a facilidade que o usuário tem em utilizar um determinado sistema, bem como a eficiência na utilização e no quanto é agradável o seu uso (ROCHA; BARANAUSKAS, 2003).

A usabilidade de um sistema está relacionada com a aceitação do mesmo, que pode ser compreendida como uma questão na qual pode observar-se que o sistema possui os requisitos mínimos para atender as necessidades e exigências dos usuários e outras partes interessadas, como utilizadores, clientes e gerentes (NIELSEN, 1994).

A interface do utilizador pode possuir diversas propriedades, e a usabilidade não é a única. Essa última, por sua vez, possui vários componentes e está associada a vários atributos (NIELSEN, 1994), como:

- **Fácil aprendizagem**: O sistema deve ser de fácil aprendizagem, e em um curto tempo ele deve obter resultados com o sistema usado (NIELSEN, 1994);
- **Eficiência**: O sistema deve ser eficiente, de modo que após o usuário aprender a manuseá-lo, ele deve obter um alto nível de produtividade (NIELSEN, 1994);
- **Memorabilidade**: O sistema deve ser fácil para memorizar, e após um tempo o usuário deve ser capaz de voltar a usá-lo sem ter a necessidade de aprender tudo novamente (NIELSEN, 1994);

- **Erros**: O sistema deve ser livre ou ter uma baixa taxa de erros, evitando gerar erros graves durante a utilização do usuário (NIELSEN, 1994); e
- **Satisfação**: O sistema deve ser agradável ao usuário, de forma que deixe ele satisfeito após sua utilização (NIELSEN, 1994).

As interfaces passaram por evoluções ao longo dos anos, e cada estágio desta evolução a impressão é que as interfaces sempre ”saltam cada vez mais para fora do computador”, se considerar as primeiras interfaces apresentadas em Grudin (1990), até as interfaces usadas na atualidade. Na Figura 2.3 é apresentada a evolução das interfaces, onde o autor apresenta os 5 focos no desenvolvimento de interface. Mesmo em 1990, que é o ano da publicação, já era possível observar esta evolução. Os outros tipos de interfaces que são utilizadas nos dias de hoje não serão abordadas nessa Dissertação, já que não faz parte dos objetivos desta Dissertação.

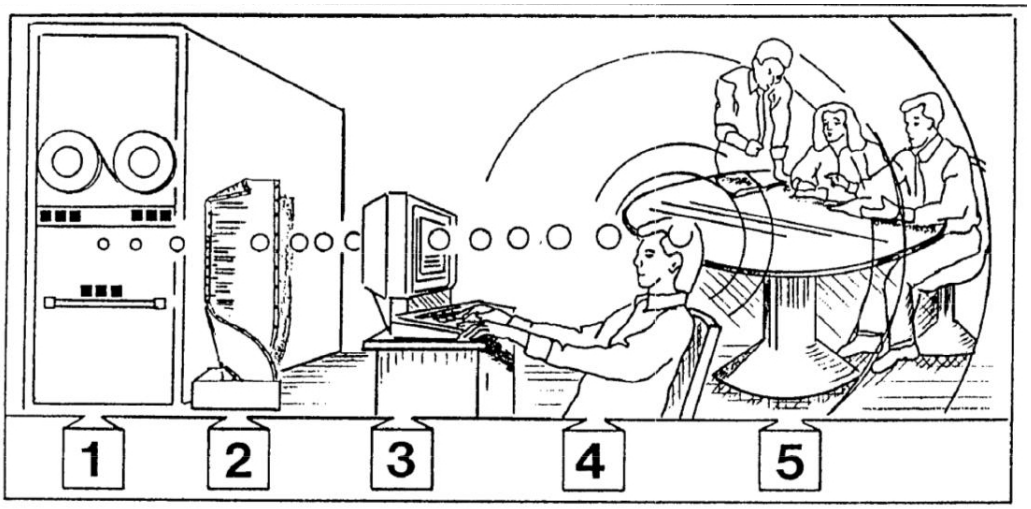

Figura 2.3: Representação da evolução das interfaces, e o foco em cada estágio de seu desenvolvimento

Adaptado de Grudin (1990)

Na Figura 2.3 foi apresentado o foco de desenvolvimento de cada tipo de interface, e estes estão melhor descritos a seguir, onde em cada item é apresentado um tipo de foco de desenvolvimento e suas principais características:

1. **A interface no hardware**: Os primeiros usuários eram Engenheiros que necessitavam ter uma visão completa do hardware; Os usuários trabalhavam em binário, sistemas de numeração hexadecimal, lidavam diretamente com registros específicos, posições de memória; Aos poucos a distância entre programadores (e outros usuários) e o hardware vem crescendo, trazendo a interface cada vez mais longe “para fora” do computador. Este crescimento está longe de chegar ao fim (GRUDIN, 1990);

2. **A interface nas tarefas de programação**: Entre os anos 60 e 70 a maioria dos usuários eram programadores; Naquele tempo os equipamentos de informática tinha um custo muito elevado; Melhorar a interface do usuário (programador) significava melhorar a sua eficiência; Os avanços neste tipo de interface estavam focados em linguagens de programação de alto nível, montadores, compiladores, depuradores e sistemas operacionais (GRUDIN, 1990);

3. **A interface no terminal**: Cuida da apresentação visual e capacidade interativa dos terminais, computadores e estações de trabalho; Responsável por possibilitar a interação do usuário com o computador por meio de comandos e estações de trabalho; A interface geralmente é desenvolvida em uma linguagem de *script* chamada **Shell** que permite a entrada de comandos e realiza a conversão destes comandos para funções do sistema operacional (GRUDIN, 1990);

4. **A interface no diálogo de interação**: Tem como objetivo desenvolver no computador interfaces que dê a sensação de diálogo com o usuário; Um dos principais objetivos é desenvolver uma interface adaptada ao usuário, baseado na retenção de ações dos últimos usuários, na adaptação ao longo do tempo; Pode-se dizer que o alcance ao computador neste tipo de interface deve ir além do teclado e a superfície da tela, devendo ampliar seu conhecimento para a mente do usuário; Para o desenvolvimento deste tipo de interface vários usuários devem ser assistidos ao longo do tempo, para ser analisada suas interações com o sistema, o registro das teclas utilizadas, além de filmar o usuário pensando em voz alta (GRUDIN, 1990); e

5. **A Interface com o ambiente de trabalho**: Tem foco para o desenvolvimento fora da interface, no ambiente social ou ambiente de trabalho; Baseado que boa parte dos trabalhos realizados em um computador são realizados em ambiente social, deve haver uma nova forma de pensar em desenvolvimento e design de interface, pensando nos diferentes usuários do grupo de trabalho, cada um com seu papel, suas habilidades, experiências e preferências (GRUDIN, 1990).

Como já mencionando, a principal proposta dessa Dissertação é desenvolver uma ferramenta de integração entre o GROMACS e a plataforma Galaxy, que visa oferecer uma nova forma de interação com o GROMACS, uma forma que oferece ao usuário um tipo mais atualizado de interface do que a **Interface de Linha de Comandos (CLI)**. Pode-se pensar também como uma atualização do tipo de interface.

2.6 Considerações Finais

Conforme mencionado neste capítulo, o Galaxy é um *framework web*, e este foi adotado para o desenvolvimento do protótipo proposto nessa Dissertação. Os motivos por sua adoção estão descritos no Capítulo 4.

Capítulo 3

LEVANTAMENTO DE DADOS E ESTADO DA ARTE

Este capítulo tem como objetivo apresentar o atual estado da arte no contexto em que esta pesquisa está inserida, dando destaque ao estado da arte em software que oferecem uma nova forma de interação com o GROMACS, pacotes de DM, tendo como foco de pesquisa o pacote de simulação GROMACS.

3.1 Software para Dinâmica Molecular

Como já citado, na literatura é possível encontrar diversos pacotes de simulação de DM, porém alguns são destaques por serem mais utilizados. Entre esses estão apresentados a seguir, cada pacote com seu nome e após na Tabela 3.1 são comparadas algumas das características de cada pacote.

- **AMBER**: o pacote AMBER (SALOMON-FERRER; CASE; WALKER, 2013) possui uma farta documentação, refere-se tanto ao campo de forças quanto ao pacote de DM e é disponibilizado para fins acadêmicos, porém não e gratuito (MORGON; COUTINHO, 2007);

- **CHARMM**: o CHARMM (BROOKS et al., 1983) possui boa documentação, refere-se tanto ao campo de forças quanto ao pacote de DM, é disponibilizado para fins acadêmicos e educacionais. Diferente da maioria dos pacotes de DM, o CHARMM tem uma abordagem mais integrada, diferenciando de outros pacotes de DM onde cada programa do pacote realiza uma tarefa específica desde a preparação, simulação e análise dos resultados, trajetória nessa Dissertação (MORGON; COUTINHO, 2007);

- **NAMD**: o NAND (PHILLIPS et al., 2005) é um pacote sofisticado, de alto desempenho, disponível para sistemas multi-processados, possui excelente escalabilidade, compatível

com diversos campos de força, possui excelente documentação e distribuição gratuíta (MORGON; COUTINHO, 2007);

- **TINKER**: o TINKER (PONDER et al., 2004) não tem versão paralelizada, tem boa documentação e é distribuído gratuítamente (MORGON; COUTINHO, 2007);
- **GROMOS**: o GROMOS (SCOTT et al., 1999) possui boa documentação, refere-se tanto ao pacote quanto ao campo de forças, e também deu origem ao pacote GROMACS. O GROMACS é um pacote mais recente, rápido, realiza o processamento em paralelo e é distribuído gratuitamente, porém não possui boa escalabilidade se comparado com o NAND. Diferente do GROMACS, o GROMOS é distribuído por um valor simbólico para fins acadêmicos (MORGON; COUTINHO, 2007); e
- **DL_POLY**: este pacote (SMITH; YONG; RODGER, 2002) possui documentação, é gratuíto, possui uma grande variedade de campos de força e possibilita realizar simulações de DM em paralelo (MORGON; COUTINHO, 2007).

Na lista apresentada, o GROMACS foi apresentado juntamente com o GROMOS, apenas por este ter dado origem ao GROMACS. Nessa Dissertação quando citado o pacote GROMACS, não se refere ao GROMOS e sim ao GROMACS.

Tabela 3.1: Comparação entre as características dos pacotes de DM mais difundidos na literatura

Início da Tabela		
Programa	**Distribuído gratuitamente**	**Possui Interface gráfica**
AMBER	Não	Não
CHARMM	Para fins acadêmicos	Não
NAMD	Sim	Não
TINKER	Sim	Não
GROMOS	Não	Não
GROMACS	**Sim**	**Não**
DL_POLY	Sim	Não
Final da Tabela		

Analisando a Tabela 3.1 é possível identificar que essa Dissertação também poderia utilizar alguns dos pacotes CHARMM, NAMD, TINKER e DL_POLY também como pacote de simulação de DM, além do GROMACS, sem ter um custo adicional em relação ao valor de sua distribuição. É possível estender essa Dissertação para no futuro possibilitar o uso desses

pacotes.

3.2 Softwares com interface gráfica para software de DM

Atualmente existem vários *software* que possuem interface gráfica e tem também como função oferecer uma nova forma de interação com o GROMACS, oferecendo ao usuário sua interface para interação com as ferramentas do GROMACS necessárias para simulação e até interpretação dos resultados. Para esta pesquisa foram encontrados alguns.

Foi notado durante as pesquisas que todos os *software* encontrados devem ser instalados no *desktop* do usuário, que são utilizados somente como interface com o objetivo de facilitar a manipulação do GROMACS. Todos eles funcionam somente localmente na máquina do usuário, dependendo dele a instalação, configuração e até atualização do *software* ao longo do tempo.

Em contra partida, a proposta do MyGromacs é que ele não seja instalado localmente no computador do usuário e sim instalado em um servidor, permitindo que o usuário acesse via *browser*, retirando dele também a necessidade de instalação do *software*, suas dependências e sua manutenção ao longo do tempo.

Como já descrito nessa Dissertação, foram encontrados *software* durante a pesquisa e esses estão citados a seguir:

- **Gromita**: a Gromita é uma ferramenta que possui uma interface de *workflow*, que possibilita a orientação do usuário através de cada passo lógico no processo de configuração da dinâmica, tornando acessível a todos os tipos de usuários, além de simples ela proporciona ao usuário uma maior funcionalidade devido a simplicidade na utilização de seus recursos, trazendo agilidade e simplificando o processo de configuração e execução das simulações de DM (SELLIS; VLACHAKIS; VLASSI, 2009);

- **Guimacs**: baseada em Java e de código aberto, o GUIMACS é uma ferramenta que torna possível a execução de diversas simulações e análises simultaneamente (KOTA, 2007);

- **Bioclipse**: com licença de código aberto, o Bioeclipse é uma plataforma de bioinformática e quimioinformática com rica funcionalidade, interface gráfica intuitíva e poderosa arquitetura de plugins (SPJUTH et al., 2007). Entre os plugins disponíveis para o Bioclipse existe um plugin que permite adicionar algumas das funcionalidades do GROMACS, disponibilizando ao Bioclipse diversos recursos de simulação de DM (Bioeclipse Wiki, 2014);

- **Pymol-gromacs**: usado por cientistas do mundo todo, o Pymol como é chamado é um ambiente de visualização e apresentação de modelos moleculares depositados no PDB (DELANO, 2002), que também possui plugin que permite estender suas funcionalidades por meio da integração com outras ferramentas, como algumas ferramentas do GROMACS, que possibilita a realização de DM e interpretação dos dados de trajetória de DM, permitindo visualizar o modelo molecular estaticamente e dinamicamente (MAKAREWICZ; KAZMIERKIEWICZ, 2013);

- **MYGromacs**: entre as ferramentas encontradas na literatura esta ferramenta é a que mais se aproxima do GromaXy (desenvolvida nessa Dissertação), porém o MYGromacs é uma ferramenta de interface gráfica de usuário que simplifica apenas a manipulação das ferramentas do GROMACS para análise das trajetórias. O projeto do MYGROMACS teve como principal objetivo possibilitar a análise de trajetórias de moléculas grandes, já que há disponível na literatura uma outra ferramenta chamada JGromacs que possibilita o mesmo serviço, porém devido ao alto consumo de memória não possibilita trabalhar com moléculas grandes (YU, 2012);

- **JsimMacs**: essa ferramenta desenvolvida em Java permite configurar e executar simulações de Dinâmica Molecular avançadas, oferecendo ao usuário uma interface intuitiva, além da capacidade de acesso remoto e interatividade 3D. Assim como outros projetos, este também visa que usuários podem desfrutar de um alivio em não precisar codificar *scripts* (ROOPRA et al., 2009); e

- **Gromacs GUI**: essa ferramenta desenvolvida em C++, só pode ser executada em Linux e tem dependência de bibliotecas externas, como Qt4 e Qwt. O Gromacs GUI possui uma interface bem completa, na qual oferece ao usuário diferentes possibilidades de configurações e tipos de simulações, conforme necessidades desejadas. O Gromacs GUI também permite que usuário plote e até expor gráficos, como também possui uma seção que facilita para o usuário configurar os arquivos mdps (.mdp) necessários para a simulação (Reza Salari, 2017).

Entre todos os *software* que oferecem interface para o GROMACS, encontrados na literatura, o MYGromacs é o que mais se aproxima do GromaXy, devido a sua interface intuitiva e uso do *e-mail* do usuário para informar o final de uma simulação, desta forma este foi escolhido entre todos outros para ser apresentado nessa Dissertação, possibilitando uma simples comparação de uma interface já existente na literatura com o MYGromacs.

O MYGromacs foi desenvolvido na linguagem de programação Java, que possibilita a

interação com as ferramentas do GROMACS escrita na linguagem C. A proposta do MYGromacs é semelhante as demais ferramentas, porém sua interface além de necessitar de instalação localmente na máquina do usuário, poderia ser mais intuitiva, não exigindo ao exemplo o usuário informar o caminho dos arquivos na extensão (.mdp) que são responsáveis pela parametrização e configuração da DM e outros processos. A Tabela 3.2 apresenta a comparação entre algumas das características do MYGromacs, e o GromaXy, tornando possível visualizar os pontos negativos e positivos de cada projeto.

Tabela 3.2: Comparação entre as características do MYGromacs e o GromaXy

Início da Tabela		
Características	**MYGromacs**	**GromaXy**
Pode ser acessado pelo browser ?	Não	Sim
Necessita instalação localmente ?	Sim	Não
Necessita alterar arquivos .mdp manualmente ?	Sim	Não
Necessita especificar o caminho dos arquivos ?	Sim	Não
Permite a criação de workflow de execução ?	Não	Sim
Armazena histórico de execução ?	Não	Sim
Permite compartilhar os resultados ?	Não	Sim
Final da Tabela		

Conforme já apresentado o MYGromacs, a Figura 3.1 apresenta a sua tela inicial que pode ser acessada por meio do botão lateral (*Home*).

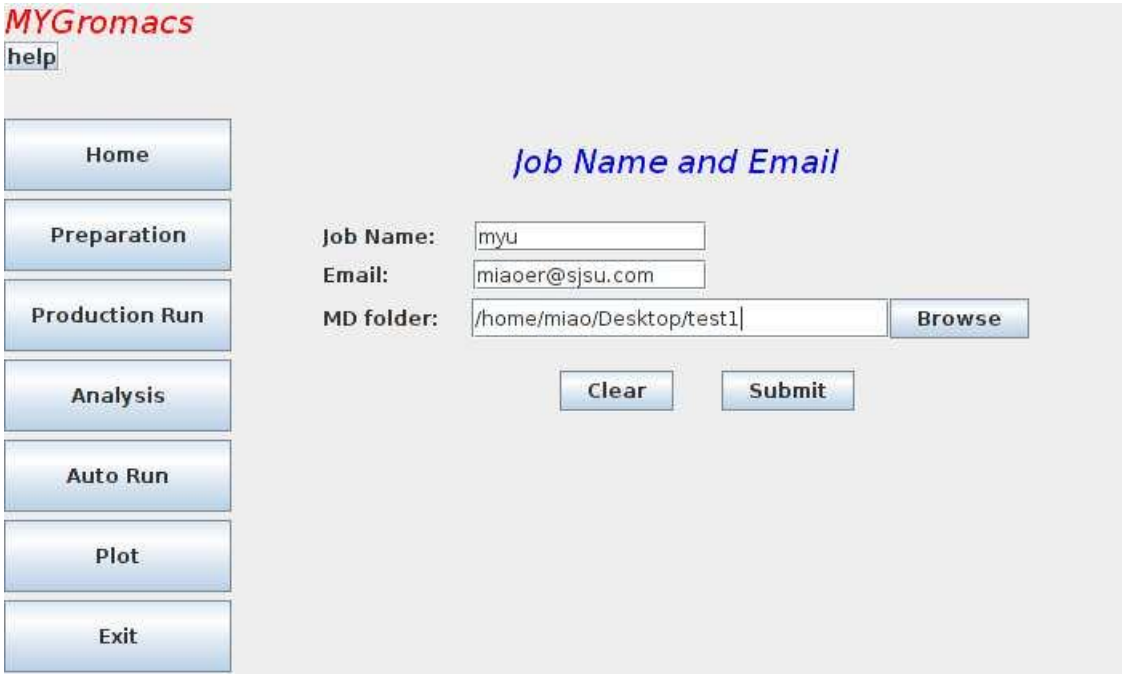

Figura 3.1: Tela inicial do MYGromacs

(YU, 2012)

A tela apresentada na Figura 3.2, permite que o usuário informe o caminho do arquivo PDB ou o id do modelo da proteína, também o caminho dos arquivos (.mdp) e parametrização da DM. Esta tela tem propostas semelhantes as da tela de simulação de DM do GromaXy, porém para o GromaXy o usuário não necessita manipular arquivos (.mdp) para configurar a DM ou a minimização de energia, pois a interface do GromaXy oferece ao usuário os parâmetros necessários, onde ele pode selecionar o parâmetro de sua necessidade, não necessitando manipular os arquivos que contém os parâmetros.

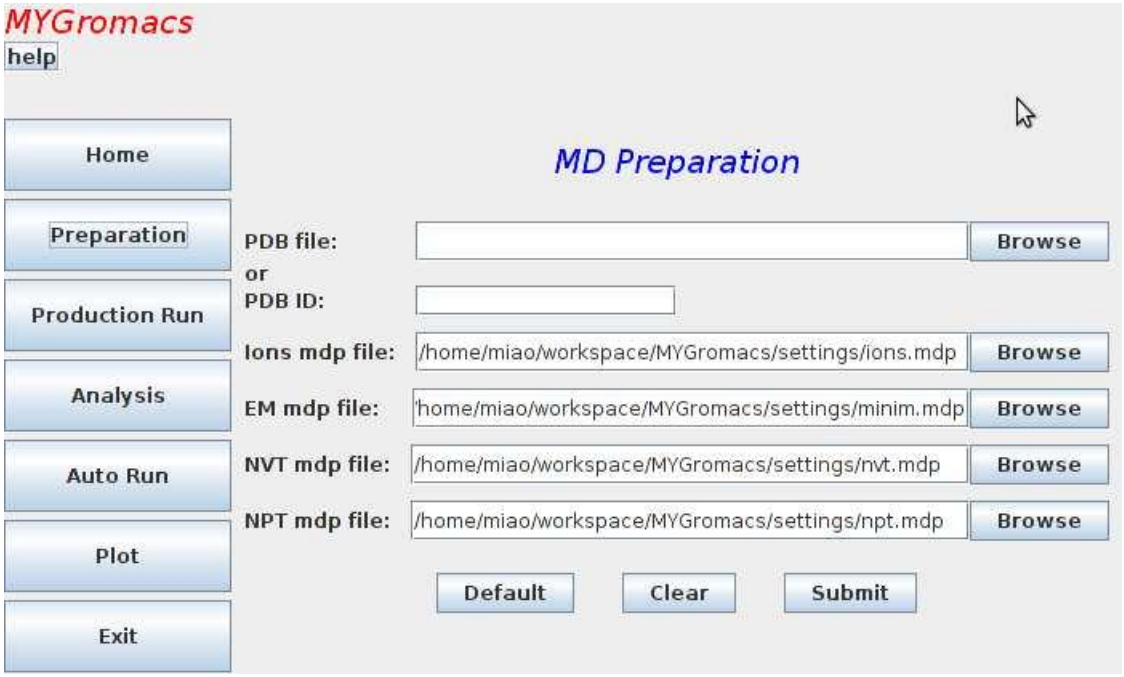

Figura 3.2: Processo típico de uma simulação de DM realizada no GROMACS

(YU, 2012)

Baseado nas comparações entre o MYGromacs e o GromaXy é possível identificar os recursos que não há no MYGromacs, e esses pontos positivos se tornam importantes para reforçar a aceitação do GromaXy por parte do usuário, considerando também que além do GromaXy os recursos do Galaxy também são disponibilizados ao usuário, assim como compartilhamento de dados, históricos de execuções de ferramentas e outros recursos do Galaxy, que enriquecem e auxiliam os trabalhos realizados pelo usuário.

Outro ponto importante para reforçar a aceitação do usuário é disponibilização de uma adaptação do Atom Validation (LEITE, 2012), que permite o usuário validar seu arquivo PDB antes de iniciar uma simulação de DM, com base em qualquer campo de força do GROMACS escolhido por ele.

3.3 Considerações Finais

Durante a pesquisa bibliográfica foi identificada a falta de material sobre os diversos pacotes de simulação de DM, acarretando em uma deficiência de se concluir uma pesquisa com um mais alto nível de qualidade sobre esses *software*. Mesmo os materiais encontrados estão desatualizados conforme suas devidas datas de publicação.

Entre as várias ferramentas encontradas, algumas das mais difundidas apresentadas nessa

Dissertação não apresentam uma interface gráfica intuitiva, dessa forma é desejável o desenvolvimento de um *software* que simplifique a manipulação desses pacotes de DM. Na literatura foi encontrado alguns projetos, porém todos exigem do usuário a instalação, configuração e atualização, quando necessário.

Em consideração a essas dificuldades encontradas e o atual estado da arte em que essa pesquisa se encontra, é possível julgar a relevância dessa Dissertação que apresenta o desenvolvimento do GromaXy, que tem como proposta principal oferecer uma nova forma de interação com o GROMACS, além de oferecer ao usuário uma adaptação do Atom Validation, para que ele possa validar suas proteínas antes de iniciar o processo de simulação no GROMACS.

Capítulo 4

GromaXy

Este capítulo faz apresentação do GromaXy, destacando seus requisitos, sua arquitetura, seu ambiente e funcionamento.

4.1 Descrição dos requisitos

Como já citado no Capítulo 2, hoje um dos grandes problemas na Bioinformática é a reprodução de experimentos já realizados por outros pesquisadores, devido ao grande número de ferramentas utilizadas durante todo o processo de pesquisa, o alto volume de dados gerados ao longo das execuções de diversos programas, e a complexidade e falta de padronização das ferramentas utilizadas. O Galaxy foi desenvolvido como uma forma de solucionar esses problemas, e esse foi disponibilizado para ser utilizado para o desenvolvimento de outras ferramentas na área de Bioinformática e até outras áreas.

Já foi mencionado também que entre os pacotes disponíveis na literatura, é possível identificar problemas de usabilidade, considerando que boa parte desses pacotes exigem do usuário um conhecimento preliminar em linha de comandos, ou melhor, Interface de Linha de Comandos (CLI). Além de possuir Interface de Linha de Comandos estas ferramentas para simulação de DM no geral geram alto volume de dados resultantes de simulações, que podem ser trajetórias de simulações ou outros tipos de dados. Entre os pacotes existentes na literatura, o GROMACS foi escolhido para ser utilizado nessa Dissertação, por ser bem difundido no meio acadêmico, e por ser também *open-source*.

Buscando oferecer uma nova forma de interação do usuário com o GROMACS e um ambiente que permita o usuário compartilhar seu histórico de trabalho, dados resultantes de simulações com outros usuários e outros recursos que o Galaxy pode oferecer, o GromaXy foi

desenvolvido nessa Dissertação como uma ferramenta de integração entre o GROMACS e o Galaxy.

O GROMACS é um pacote de simulações que contem inúmeras ferramentas, e possibilita a execução de simulações com diversos tipos de configurações, porém como motivo de validação da proposta do GromaXy, ele foi desenvolvido com o objetivo de possibilitar o usuário realizar uma típica simulação de DM, podendo nas próximas etapas de um trabalho futuro amadurecer este projeto, expandindo a disponibilização de recursos do GROMACS para o usuário, conforme suas principais e mais variadas necessidades.

Uma simulação de DM segue um fluxo de execução, seja ele sequencial ou lógico, conforme determinação do usuário em sua configuração. Como determinado que o GromaXy deverá permitir a realização de uma típica simulação, um fluxo de execução básico deve ser seguido, e este é apresentando na Figura 4.1, na qual é possível visualizar os principais programas que são utilizados, e sua ordem de execução.

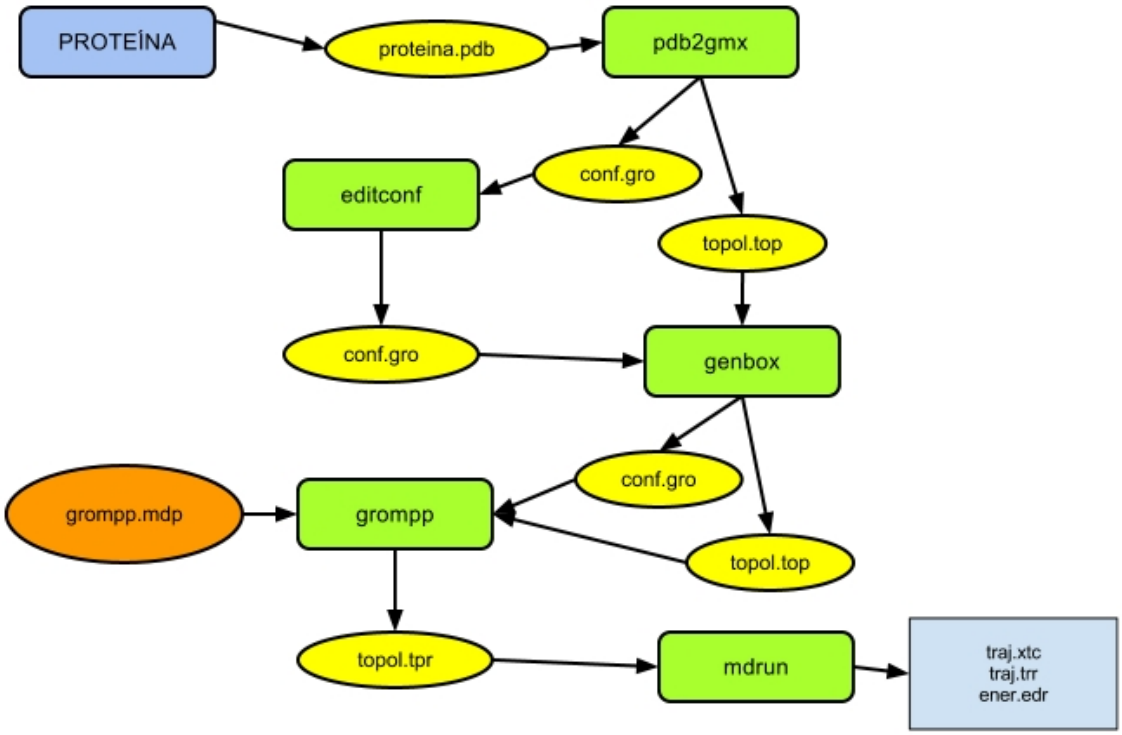

Figura 4.1: Processo típico de uma simulação de DM realizada no GROMACS

Com o objetivo de se minimizar a energia sobre cada átomo e a energia potencial do sistema, é desejável a realização do processo de minimização de energia, que deve ser executado antes da simulação de DM. Sendo assim, nessa Dissertação são realizadas duas minimizações com a finalidade de evitar alterações bruscas na energia potencial do sistema, pois na primeira minimização de energia, as cadeias laterais hidrofóbicas que estão na superfície da proteína a

ser simulada tendem a se relaxar e se posicionar no centro da proteína, diminuindo a possibilidade de ocorrer um reposicionamento forçado quando se realizada a segunda minimização com solvente. O diagrama do processo geral que deverá ser realizado nessa Dissertação é apresentado na Figura 4.2, destacando as duas minimizações e a DM com seus respectivos arquivos de saída.

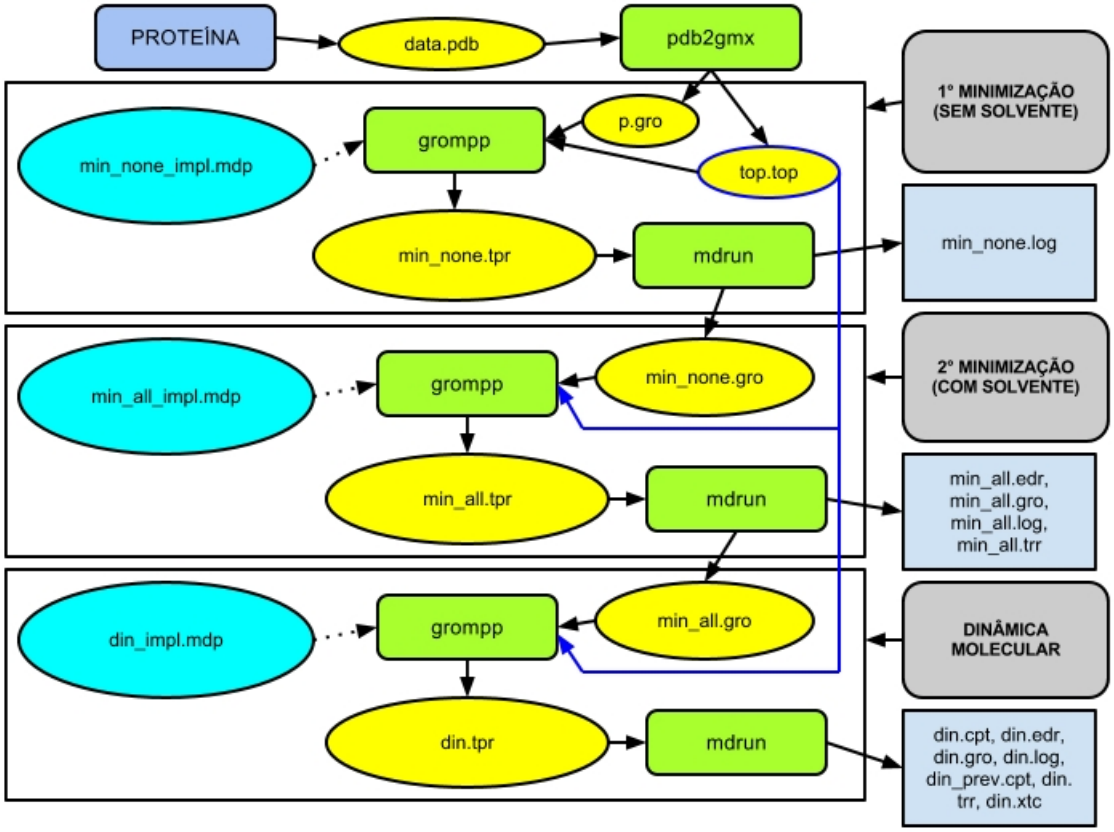

Figura 4.2: Fluxo de trabalho que é realizado pelo GROMACS

Por meio da Figura 4.2 é possível verificar na cor verde todos programas que são utilizados no processo de minimização de energia e simulação de DM, cada programa com seus respectivos arquivos de entrada e arquivos de saída, com a finalidade de destacar a importância de cada programa no processo geral e descrever o seu funcionamento, a seguir estão listados cada programa com sua descrição:

- **pdb2gmx**: é o primeiro programa que deve ser utilizado no processo de simulação de DM no GROMACS. Ele é responsável por fazer a leitura de um arquivo de entrada, o PDB, que contém os dados da proteína a ser analisada; gerar as coordenadas dos átomos da proteína em um arquivo de extensão (.gro); e gerar a topologia em um arquivo de extensão (.top). No arquivo de topologia são especificados os atributos constantes de cada átomo, entre estes, quais átomos ou combinações de átomos devem ser utilizados

para a aplicação das funções potenciais, e quais parâmetros devem ser aplicados a estas funções. Para preparar as coordenadas e a topologia, o pdb2gmx inicialmente verifica a existência de um arquivo que possui a parametrização de campo de força. O GROMACS possui uma flexibilidade de modo que torna possível a escolha de um campo de força pelo usuário. Nessa Dissertação, em todas as simulações serão utilizadas o campo de força amber99sb-ildn (APOL et al., 2010). Para gerar a topologia é possível especificar parâmetros conforme a necessidade. Nessa Dissertação será utilizado o parâmetro -ignh (para ignorar os átomos de hidrogênio), e - water none (para não utilizar nenhum tipo de água na primeira minimização de energia do sistema) (APOL et al., 2010);

- **editconf**: após gerada a topologia do sistema é necessário fazer a leitura do arquivo que contém a estrutura da proteína especificada inicialmente e converter em um novo arquivo do GROMACS, que é especificado também com a extensão (.gro). A partir desta opção é possível especificar o tipo de caixa de simulação que será utilizada na simulação da DM, como a especificação do diâmetro do sistema, os ângulos da caixa, coordenadas e velocidades, etc (APOL et al., 2010);

- **genbox**: é o programa que permite a geração de uma caixa de solvente, a configuração de um solvato solúvel e a inserção de número de moléculas extras em posições aleatórias. Se inseridas moléculas de solvente o genbox consegue alterar o arquivo de topologia de modo que ele identifique a linha que contém a informação com o número de moléculas de solvente e a remova, inserindo uma nova linha com a nova quantidade de moléculas (APOL et al., 2010);

- **grompp**: o grompp realiza a expansão da topologia de nível molecular para nível atômico, realiza a leitura de um arquivo de coordenadas para gerar velocidade a partir de uma distribuição de Maxwell, se necessário, e também a leitura de parâmetros dos arquivos (.mdp)'s gerados pelo usuário, que contém os parâmetros como número de passos, restrições, etç. Nessa Dissertação este arquivo (.mdp) é alterado pelo usuário por meio de uma interface gráfica, de modo que ele parametrize algumas informações para a dinâmica, sem ter o conhecimento total da manipulação destes comandos para gerar a dinâmica e parametrizar os arquivos. Após todo o processo descrito, o grompp gera um arquivo binário (APOL et al., 2010); e

- **mdrun**: é o programa responsável por realizar a simulação de DM do sistema, ele também possibilita a realização de minimização de energia, inserção de partículas de teste e cálculos de energias. Para realizar a DM ou minimização de energia do sistema o mdrun recebe um arquivo de entrada com a extensão (.tpr), e gera até quatro arquivos

de saída, entre eles, um arquivo para armazenar o log da DM ou log da minimização de energia, um arquivo de trajetória que contém as coordenadas, velocidades e forças; um arquivo de energia que contém energias, temperaturas e pressão; e um arquivo com as coordenadas e velocidades do último passo (APOL et al., 2010).

4.2 Arquitetura

Já difundido no meio acadêmico e amplamente utilizado por pesquisadores no cenário de Bioinformática, o Galaxy foi escolhido por suas qualidades e por ser um *framework* que atende os objetivos dessa Dissertação, possibilitando desenvolver por meio dele novas ferramentas (*tools*) em seu ambiente de forma padronizada, utilizando nas novas ferramentas todos os seus recursos, como: controle de usuário, armazenamento de dados, criação de *workflow*, compartilhamento de dados, histórico de execução e entre outros recursos interessantes.

Baseado na proposta dessa Dissertação, o GromaXy foi desenvolvido para atender os requisitos básicos já citados e ter aceitação por parte do usuário, que está habitualmente acostumado a manipular o GROMACS por interface baseada em linha de comandos e armazenar o grande volume de dados localmente em seu computador. Com o desenvolvimento do GromaXy, uma nova forma de interação com o GROMACS é disponibilizada ao usuário, trazendo possibilidades de aumentar nível de usabilidade comparado com o GROMACS.

Devido ao grande tempo de processamento de uma simulação de DM, para essa Dissertação foi necessário desenvolver duas ferramentas, como a finalidade de dividir a tarefa de simulação em dois cenários diferentes:

- Primeiro cenário: Realização da simulação de DM; e
- Segundo cenário: Download dos dados resultantes da simulação.

Com a divisão da tarefa de simulação em dois cenários, o usuário ao realizar uma simulação não necessita ficar aguardando a sua conclusão, pois todo o processo de simulação é executado em *background*, assim o usuário pode realizar outros trabalhos enquanto a simulação está sendo realizada. O usuário é informado sobre a conclusão da simulação por meio de um e-mail que é enviado a ele com uma chave única que identifica a simulação realizada, e esta chave que deve ser utilizada para por meio da ferramenta desenvolvida para no segundo cenário realizar o *download* dos dados resultantes da simulação.

A chave de identificação da simulação que é enviada via e-mail ao usuário, além de identificação também tem como objetivo organizar os resultados das simulações e controlar o acesso as informações de forma que não seja possível um usuário acessar os dados de outro.

A Figura 4.3 apresenta em dois cenários, todo o processo que é realizado pelo usuário, desde a escolha de um modelo de proteína no banco de dados PDB até a configuração, execução da simulação, notificação e *download* dos dados de trajetória da simulação de DM realizada.

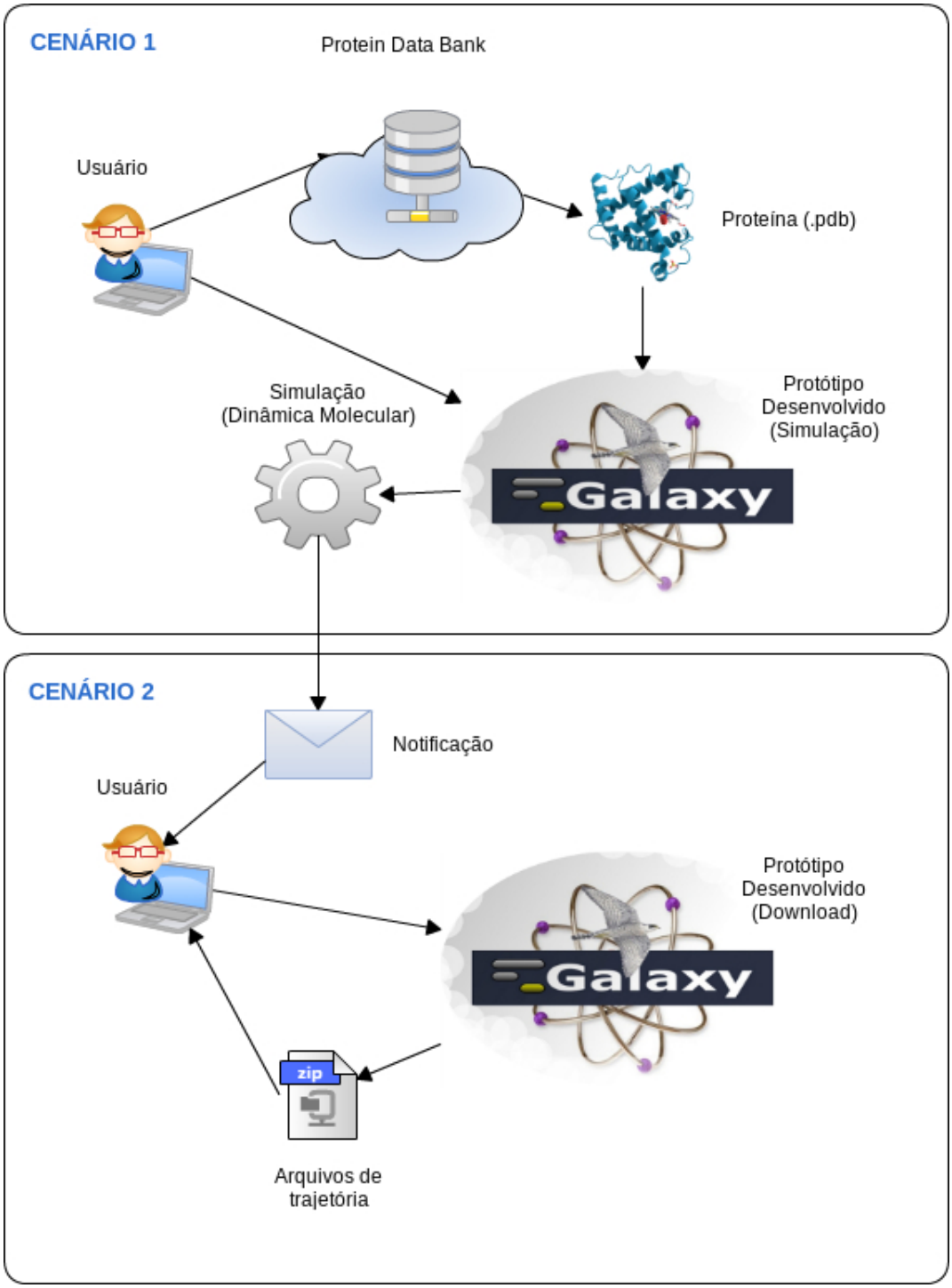

Figura 4.3: Representação do processo geral dividido por dois cenários distintos

Depois do usuário escolher um modelo de proteína por meio do PDB, ele pode fazer um *upload* desse modelo na ferramenta padrão do Galaxy usada para fazer *upload* de arquivos. Está ferramenta é apresentada na Figura 4.4. Somente depois de submeter este modelo ao Galaxy o usuário consegue selecionar um modelo de proteína no GromaXy para realizar sua simulação. Cada modelo submetido ao Galaxy fica registrado no perfil do usuário que está logado, permitindo que o usuário possa escolher um modelo que foi submetido em um antigo acesso que teve ao Galaxy.

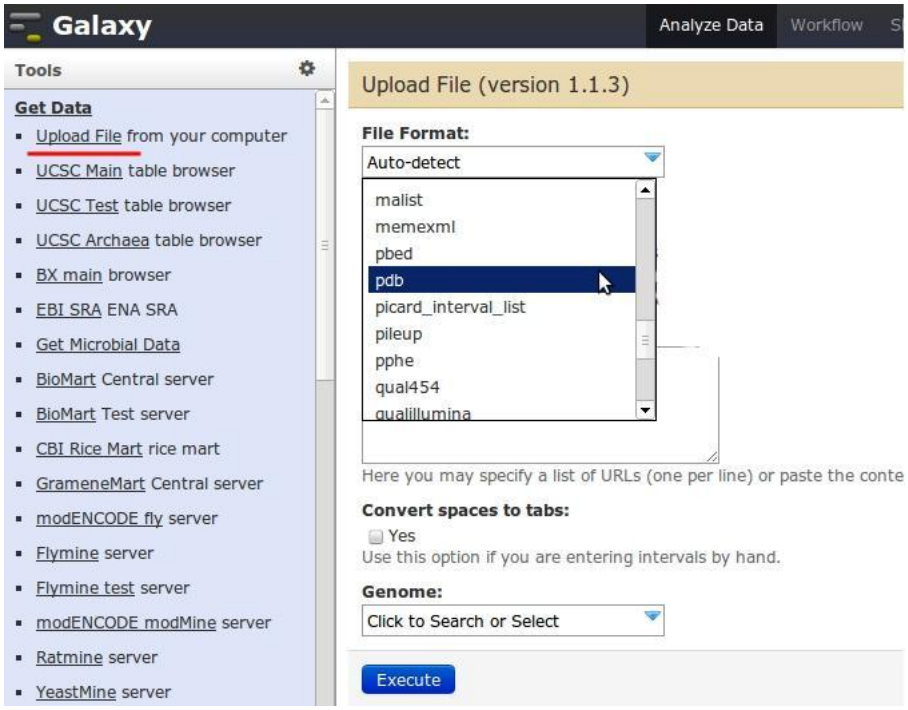

Figura 4.4: Tela para fazer um upload de um arquivo PDB no Galaxy

Ao acessar o GromaXy para realizar a simulação de DM, o usuário além de escolher a proteína submetida ao Galaxy, poderá realizar a parametrização da minimização de energia e DM, e poderá conforme a sua necessidade escolher a opção que garante a execução da simulação de DM com solvente implícito. É por meio da escolha desses parâmetros graficamente que o usuário estará interagindo indiretamente com o GROMACS, que será iniciado quando o usuário executar a simulação depois de parametrizada toda a tela do GromaXy que está apresentado na Figura 4.5.

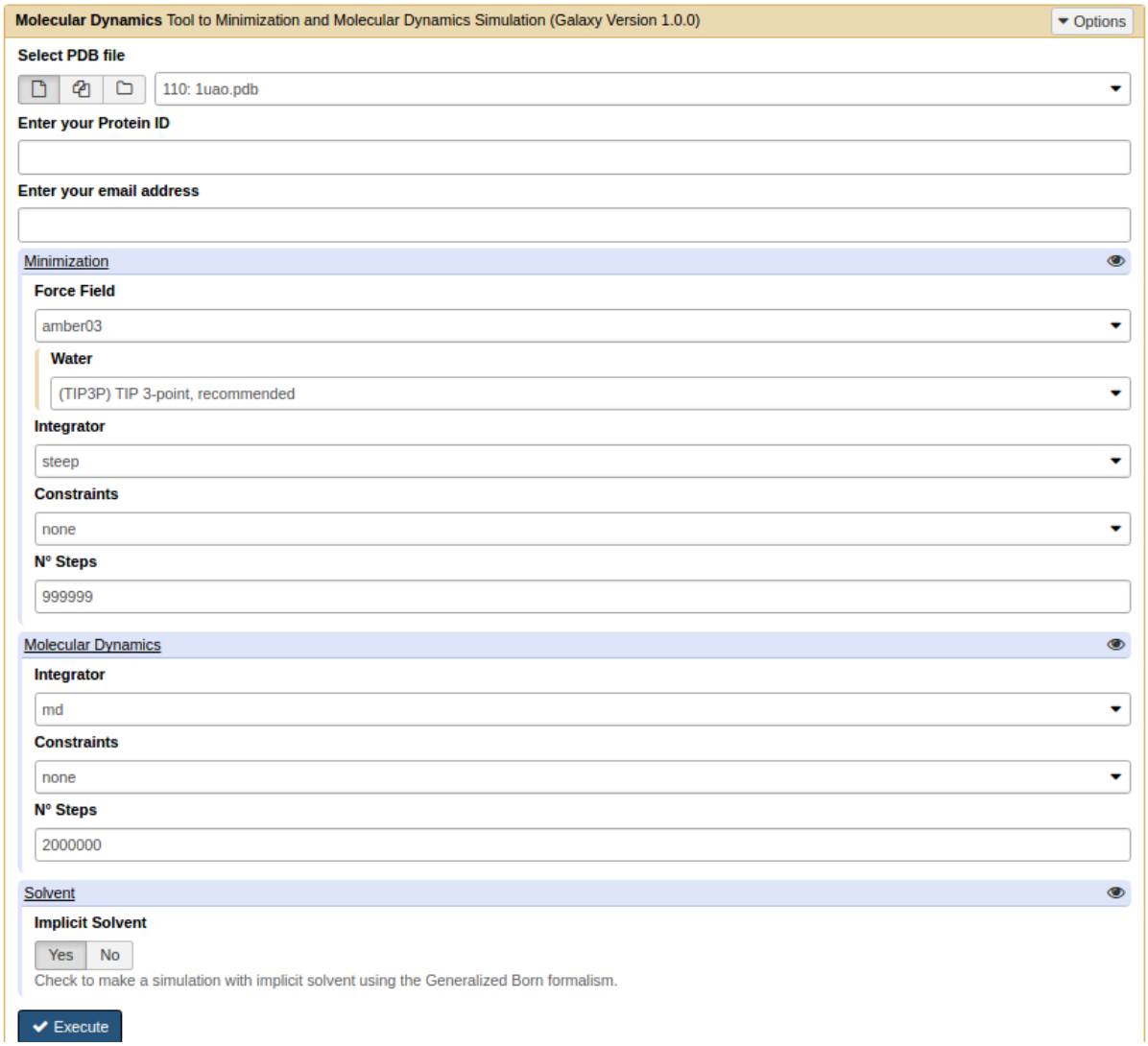

Figura 4.5: Tela principal do GromaXy, que possibilita a configuração e realização de DM.

Conforme apresentado na Figura 4.3, todo o processo desde a simulação até o *download* dos dados de trajetória da simulação é divido em dois cenários distintos, a partir da finalização da simulação de DM que se inicia os processos do segundo cenário, onde o usuário é notificado via e-mail que sua simulação foi concluída. Este processo de notificação é necessário devido ao tempo que pode levar uma simulação, que pode variar de minutos, horas e até meses, dessa forma este processo dispensa a necessidade do usuário consultar periodicamente o *status* da simulação. Após ser notificado o usuário pode entrar na tela que é apresentada na Figura 4.6 e iniciar o processo de *download* dos dados.

Figura 4.6: Tela da ferramenta desenvolvida para efetuar o download dos resultados

O GromaXy enviará a notificação para o e-mail que o usuário usou para criar sua conta no Galaxy, porém durante a configuração da simulação ele poderá informar um e-mail diferente, e este e-mail que será usado para ser enviado a notificação que está apresentada na Figura 4.7, que enviará consigo a *tag* que identificará a simulação realizada e será utilizada para o *download* dos dados na tela de *download* que está apresentada na Figura 4.6.

Figura 4.7: Exemplo de email que é enviado ao usuário notificando a finalização da simulação

É importante reforçar que mesmo com a configuração de simulação de DM no GromaXy, a simulação é realizada pelo GROMACS, pois a proposta inicial do GromaXy é oferecer ao usuário uma nova forma de interação com software de DM, no caso o GROMACS nessa Dissertação. Toda a parte de controle de dados por usuário não é tarefa do GROMACS, pois no Galaxy foram desenvolvidos métodos de armazenamento de dados por usuário, garantindo a organização e segurança dos dados que podem ser utilizados para diversos tipos de pesquisas.

Durante todo o processo de execução, tanto da simulação de DM ou *download*, é criado um *job* no histórico do usuário que pode ser compartilhado com outros usuários, e este recurso ajuda a preencher a lacuna na reprodução de experimentos, que hoje é um ponto importante a ser trabalhado, pois inúmeros trabalhos podem ser difíceis de serem reproduzidos por falta de informações sobre as ferramentas e métodos utilizados.

4.3 Considerações Finais

Atualmente uma das grandes dificuldades encontradas na Bioinformática é a reprodução de experimentos já realizados por outros pesquisadores, devido ao grande conjunto de dados gerados e uso das mais variadas ferramentas ao longo de um grande trabalho. Além destas dificuldades cada ferramenta tem suas características de interface com o usuário, que pode dificultar o trabalho por elas não serem padronizadas e não estarem centralizadas em um único ambiente. Baseado nessas dificuldades o *framework* Galaxy foi adotado nessa Dissertação, por preencher essas lacunas mencionadas e possuir um ambiente já aceito pela comunidade de Bioinformática pelo mundo. O preenchimento dessas lacunas pode trazer um grande ganho para o usuário e consequentemente para a Ciência.

O Galaxy diferente das ferramentas mais tradicionais de bioinformática pode ser instalado em um servidor, de forma que o usuário apenas acesse seus serviços (ferramentas nessa Dissertação) pelo *Browser*, descartando a necessidade de ter um programa instalado localmente em sua máquina e armazenar dados em disco.

E importante destacar que o GromaXy está em constante processo de desenvolvimento e amadurecimento. Nessa Dissertação ele foi desenvolvido como um protótipo baseado nas necessidades básicas do usuário.

Durante as consultas realizadas aos usuários foi detectado a necessidade de validação do modelo da proteína a ser utilizada na simulação, o que ocasionou o interrompimento do processo de desenvolvimento e amadurecimento da ferramenta que visa oferecer ao usuário uma nova forma de interação com GROMACS e iniciar o desenvolvimento de uma ferramenta que oferece ao usuário uma adaptação da ferramenta Atom Validation, uma vez que é necessário validar a estrutura da proteína antes de iniciar o processo de simulação na ferramenta desenvolvida anteriormente.

Alguns trabalhos de bioinformática são difíceis de serem reproduzidos por conta da forma que os dados são gerados, por exemplo, trajetórias de DM, que são conjuntos de dados que são gerados por meio de cálculos de interações, o que pode tornar muito difícil reproduzir os mesmos dados, já que a cada simulação o sistema de simulação pode apresentar um cenário diferente, como posição dos átomos e energias relacionadas durante a evolução temporal do sistema.

Capítulo 5

ASPECTOS DA IMPLEMENTAÇÃO

Nesse capítulo é apresentado o processo de desenvolvimento padrão de ferramentas no Galaxy, o processo que foi adotado para a implementação do GromaXy, e também os detalhes da implementação foi que necessária para adaptar o Atom Validation ao GromaXy e expandí-lo para possibilitar o usuário a utilizá-lo no Galaxy, permitindo usar todos os campos de força presentes no GROMACS. Nesse capítulo também são apresentadas as alterações que foram necessárias para que o Galaxy passasse a reconhecer arquivos PDB, que representa o modelo de proteína utilizada, que é extraída do banco de dados PDB.

5.1 Desenvolvimento de ferramentas no Galaxy

A comunidade de desenvolvimento e usuários do Galaxy é constituída de muitas pessoas, e essas cooperam para o projeto principal, também ajudam uns aos outros em suas mais diversas dificuldades e dúvidas a cerca de seus projetos usando o Galaxy. Essa comunidade é bem ativa e nesse projeto também foi consultada para tirar algumas dúvidas encontradas no decorrer do desenvolvimento do GromaXy.

Para desenvolver uma ferramenta no Galaxy é necessário seguir um processo de desenvolvimento. Nas Subseções 5.1.1 e 5.1.2, é apresentado o exemplo de desenvolvimento de uma ferramenta do Galaxy, para que seja possível compreender as técnicas adotadas para o desenvolvimento do GromaXy.

5.1.1 Galaxy Tool XML File

No Galaxy as ferramentas são geradas por meio de arquivos XML, e esses são cadastrados em arquivos de configuração do Galaxy. Durante a execução do Galaxy, são essas configurações

que permitirão que o Galaxy saiba quais ferramentas ele deve montar em seu ambiente, para que fique disponível ao usuário.

Para cada ferramenta deve haver um arquivo XML correspondente, e este deve ser usado para definir a interface com o usuário. No arquivo XML que podem ser definidos elementos da interface, como: campos de entrada, textos e *help* da ferramenta, que serve como suporte para possíveis dúvidas referente sua utilização e até outras informações importantes para apresentar ao usuário. Quaisquer detalhes que devem ser apresentados ao usuário deverão ser definidos no arquivo XML.

O arquivo XML de cada ferramenta também tem a finalidade de passar ao *script* os parâmetros que foram definidos pelo usuário na interface gráfica da ferramenta. A linguagem Python atualmente é a mais adotada pela comunidade para escrever seus *scripts*, porém é possível também escrever em outras linguagens, como exemplo, a linguagem Perl, que é atualmente uma linguagem muito usada na Bioinformática, por conta do seu poder em processamento de *strings*.

5.1.2 Galaxy Tool *Script* File

Como apresentado na Seção 5.1.1, uma ferramenta deve ser constituída de um arquivo XML e pelo menos um arquivo de *script*, onde esse receberá os parâmetros definidos na interface, na qual o usuário definirá por meio de suas escolhas. Os parâmetros são passados do arquivo XML para o *script* e esse deve usá-los para suas finalidades.

Uma ferramenta pode ter mais de um arquivo de *script*, já que no código de um arquivo pode haver uma chamada de outro arquivo e assim consecutivamente, conforme necessidade do desenvolvedor. No desenvolvimento do GromaXy foi necessário usar mais de um arquivo de *script* por ferramenta, por conta de organização de código e até mesmo para definir melhor suas funcionalidades.

Durante o desenvolvimento do GromaXy foi observado que algumas funcionalidades padrões da linguagem Python pode não funcionar dentro do ambiente do Galaxy, assim como algumas bibliotecas. Isso pode fazer com o que o processo de desenvolvimento seja um pouco mais complicado se comparado ao desenvolvendo uma ferramenta fora do ambiente do Galaxy, uma vez que pode ser necessário testar várias possibilidades ou recursos diferentes para chegar em um mesmo resultado, por falta de compatibilidade. Esse foi um problema enfrentado durante o desenvolvimento do GromaXy.

5.1.3 Galaxy Tool Shed

Na comunidade científica pode ser comum encontrar mesmas necessidades entre vários usuários, e com isso pode tornar-se interessante que um usuário compartilhe sua ferramenta para outros usuários, onde esses tenham acesso aos mesmos serviços que tem acesso o desenvolvedor da ferramenta. O Galaxy permite que desenvolvedores compartilhem suas ferramentas com a comunidade por meio de um serviço, dessa forma muitas ferramentas implementadas podem ser disponibilizadas para a comunidade por meio de um serviço do Galaxy, chamado de *Galaxy Tool Shed* (BLANKENBERG et al., 2014).

Após compartilhada uma ferramenta qualquer usuário pode instalar essa em sua instância, seja ela local (no seu computador pessoal) ou até mesmo em seu servidor. Esse é um recurso que pode ser muito interessante se pensar no sentido de facilitar a reprodução de experimentos, que já é uma das propostas do Galaxy. Uma vez que um pesquisador usou um conjunto de ferramentas para chegar até um resultado que foi publicado, ele pode compartilhar com outro pesquisador essa ferramenta, e não somente o histórico de trabalho, dados gerados ao longo do trabalho e outras informações.

Para instalar uma ferramenta por meio do *Galaxy Tool Shed* é necessário que o usuário acesse o serviço e que tenha privilégios de administrador, o que pode ser concedido por meio de configuração em arquivos de configurações do Galaxy, onde o usuário pode determinar qual usuário do sistema terá tais privilégios.

Para exemplificar melhor o que já foi descrito, na Seção 5.2 é apresentado um exemplo de desenvolvimento de uma ferramenta no Galaxy, desde o desenvolvimento de um *script*, definição do arquivo XML até o cadastramento de uma nova ferramenta nos arquivos de configuração do Galaxy. Esses exemplos são apresentados para facilitar a compreensão do leitor em relação ao desenvolvimento do GromaXy, e também o desenvolvimento de uma nova ferramenta no Galaxy.

5.2 Desenvolvendo uma ferramenta no Galaxy

Esta seção demonstra um exemplo de implementação de uma ferramenta no Galaxy. No exemplo é apresentada uma ferramenta que tem por finalidade computar sequencia de letras 'CG' de uma sequencia em um arquivo FASTA, que é um arquivo que possibilita o armazenamento de sequencias, nesse exemplo nucleotídios. Esse exemplo e todos trechos de códigos dessa seção foram extraídos do tutorial do Galaxy, que pode ser consultado em GalaxyToolTu-

torial (2017).

Toda e qualquer ferramenta desenvolvida no Galaxy, que será cadastrada nos arquivos de configuração, deve ser criada em diretórios específicos para que o Galaxy consiga encontrar a ferramenta para montar em seu ambiente durante a execução da plataforma. Seguindo as regras de integração de novas ferramentas no Galaxy, é necessário sempre adicionar o *script* e o arquivo XML ao diretório '/tools', onde todos os arquivos relacionados as ferramentas devem ser armazenados. No exemplo, é necessário criar uma subdiretório chamado '/myTools' dentro do diretório atual, no caso o diretório '/tools'. No exemplo o nome do diretório está como '/myTools', porém cada desenvolvedor pode atribuir qualquer nome de seu interesse.

No Linux é possível criar um diretório graficamente ou por meio de um terminal, onde o usuário interage diretamente com uma Interface de Linha de Comandos. No terminal os comandos necessário para a criação do diretório, são : 'cd tools', 'mkdir myTools' e 'cd myTools'. O comando 'cd' é usado para entrar em um diretório específico, no exemplo o diretório 'tools'. Já o comando 'mkdir' é usado para a criação de diretório, no exemplo, o diretório 'myTools', onde ficará os arquivos da ferramenta desenvolvida. No sistema operacional Linux é usado o conceito de diretório, porém para uma compreensão mais didática, é possível considerar que 'diretório' no Linux é equivalente a 'pastas' no sistema operacional Windows.

O Trecho de código 5.1 apresenta o arquivo de *script* que será necessário ser escrito, no exemplo o arquivo tem o nome de 'toolExample.pl', e este é um *script* escrito na linguagem Perl.

Código-fonte 5.1: *Script* toolExample.pl (GalaxyToolTutorial, 2017)

```
1  #!/usr/bin/perl -w
2  # usage : perl toolExample.pl <FASTA file> <output file>
3
4  open (IN, "<$ARGV[0]");
5  open (OUT, ">$ARGV[1]");
6  while (<IN>) {
7      chop;
8      if (m/^>/) {
9          s/^>//;
10         if ($. > 1) {
11             print OUT sprintf("%.3f", $gc/$length) . "\n";
12         }
13         $gc = 0;
14         $length = 0;
15     } else {
```

```
16          ++$gc while m/[gc]/ig;
17          $length += length $_;
18      }
19  }
20  print OUT sprintf("%.3f", $gc/$length) . "\n";
21  close( IN );
22  close( OUT );
```

No Trecho de código 5.2 é apresentado o arquivo XML que deverá ser criado para que o Galaxy use seus dados para montar a interface gráfica da ferramenta. Essa interface será usada pelo usuário e será a única forma dele interagir diretamente com a ferramenta.

Código-fonte 5.2: Conteúdo do arquivo toolExample.xml (GalaxyToolTutorial, 2017)

```
1  <tool id="fa_gc_content_1" name="Compute GC content" version="0.1.0">
2    <description>for each sequence in a file</description>
3    <command interpreter="perl">toolExample.pl $input $output</command>
4    <inputs>
5      <param format="fasta" name="input" type="data" label="Source file"/>
6    </inputs>
7    <outputs>
8      <data format="tabular" name="output" />
9    </outputs>
10
11   <tests>
12     <test>
13       <param name="input" value="fa_gc_content_input.fa"/>
14       <output name="out_file1" file="fa_gc_content_output.txt"/>
15     </test>
16   </tests>
17
18   <help>
19 This tool computes GC content from a FASTA file.
20   </help>
21
22 </tool>
```

No Trecho de código 5.2 é possível identificar uma *tag* responsável pela invocação do arquivo 'toolExample.pl', e por determinar qual interpretador deve ser usado para executar o *script* informado. Durante a execução do XML os parâmetros '$input' e '$output' são passados para o *script*, e esses indicam quais parâmetros serão passados para o *script*, e qual será

o retorno do *script* para o XML, para ser entregue ou apresentado ao usuário. Essa tag está apresentada no Trecho de código 5.3.

Código-fonte 5.3: Associação do *script* ao XML, com seu interpretador e parâmetros (GalaxyToolTutorial, 2017)

```
1 <command interpreter="perl">toolExample.pl $input $output</command>
```

Após criados os arquivos 'toolExample.pl' e 'toolExample.xml' dentro do diretório '/tools/myTools', é necessário cadastrar a ferramenta desenvolvida no arquivo 'tool_conf.xml' que pode ser encontrada no diretório '/config', que fica dentro do diretório raiz do Galaxy. A localização de alguns diretórios no Galaxy pode variar conforme sua versão.

Para adicionar a ferramenta no arquivo de configuração é necessário adicionar um trecho de código responsável por indicar ao Galaxy o nome da ferramenta, como também o caminho para ele encontrar o arquivo XML da ferramenta. Esse código é apresentado no Trecho de código 5.4.

Código-fonte 5.4: Código necessário para cadastrar a ferramenta MyTools (GalaxyToolTutorial, 2017)

```
1 <section name="MyTools" id="mTools">
2     <tool file="myTools/toolExample.xml" />
3  </section>
```

Após implementação da ferramenta e sua configuração no ambiente do Galaxy, é possível instanciar o Galaxy e acessar a nova ferramenta desenvolvida. Para instanciar o Galaxy é necessário executar o arquivo **run.sh** que pode ser encontrado no diretório raiz. Para execução é necessário executar o comando **sh run.sh** no Terminal do Linux. Se durante o processo de execução não ocorrer algum erro grave que impossibilite a conclusão, o Galaxy pode ser acessado pelo endereço **http://localhost:8080**. Esse é o endereço que vem pré-configurado no Galaxy, porém pode ser alterado conforme necessidade.

Após acessar o Galaxy, a ferramenta já pode ser visualizada, e a interface dessa deve ser semelhante a apresentada na Figura 2.2.

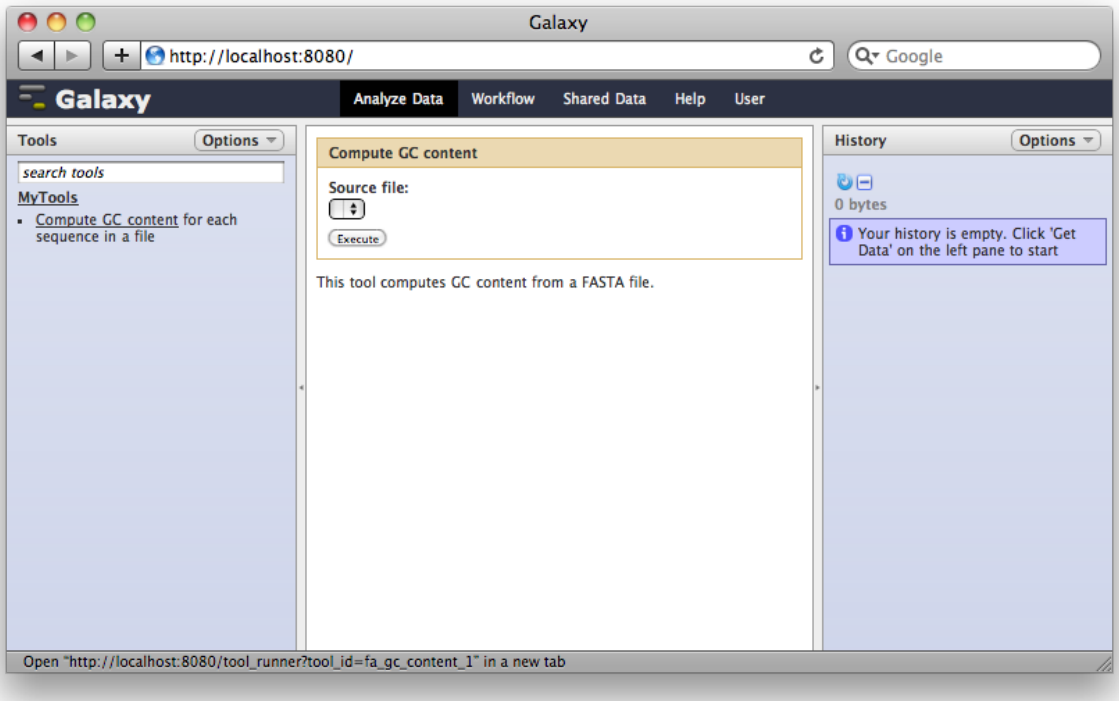

Figura 5.1: Ferramenta MyTools

(GalaxyToolTutorial, 2017)

Nessa seção foi apresentado um exemplo do desenvolvimento de uma ferramenta simples no Galaxy. Cada desenvolvedor pode desenvolver sua ferramenta conforme suas necessidades, o que pode fazer do processo de desenvolvimento da ferramenta um tanto mais complexo do que o apresentado aqui. Portanto, é possível compreender que o exemplo foi usado apenas para fins didáticos com foco em demonstrar para o leitor como se desenvolve uma nova ferramenta no Galaxy, para facilitar a compreensão do trabalho realizado durante o desenvolvimento do GromaXy.

5.3 Desenvolvimento do GromaXy

Para o desenvolvimento do GromaXy foi adotado o mesmo processo que foi apresentado na Seção 5.2, e os detalhes de implementação são apresentados nessa seção. É importante destacar que no exemplo apresentado, o *script* foi desenvolvido na linguagem de programação Perl, porém para o desenvolvimento do GromaXy foi utilizado a linguagem Python. Também foi utilizada linguagem Shell para invocar as ferramentas necessárias do pacote GROMACS.

5.3.1 Estrutura do GromaXy

Durante o desenvolvimento do GromaXy foi necessário desenvolver um padrão a ser utilizado para armazenar os dados das simulações, organizando cada simulação por usuário, de forma que cada usuário deverá acessar somente os dados que referem as simulações realizadas por ele. Baseado nessa necessidade foi criado um processo que gerará uma *tag* formada por meio da concatenação do e-mail do usuário logado com um *underline* (_) e um número que é gerado aleatoriamente. Pensando na pequena probabilidade de gerar um identificador repetido foi desenvolvido um método para verificar a existência de identificador, se confirmado a sua existência o sistema gerará novamente repetindo todos os passos necessários.

Como a *tag* é utilizada para organizar os dados dos usuários, e o processo de simulação terá como resultado vários arquivos como *output*, antes de se iniciar o processo de minimização de energia e simulação de DM é criado um diretório com o nome da *tag*, e todo o processo é realizado dentro desse diretório, assim não é necessário ao longo da simulação ficar movendo os arquivos de *output* para a diretório.

O controle dos dados por usuário é realizado por meio do inicio da *tag* por iniciar com o e-mail do usuário logado, portando ao realizar o *download* dos dados o sistema verificará se a *tag* informada tem o seu inicio correspondente ao e-mail do usuário logado, evitando outro usuário acessar seus dados.

Como já apresentado, o Galaxy possui seu padrão de desenvolvimento, onde cada nova ferramenta é desenvolvida por meio de marcações em XML e em códigos desenvolvidos em linguagens de interesse do usuário, por exemplo: Python, Perl ou outras. O XML é responsável por definir os componentes da interface gráfica da nova ferramenta e a passar dados inseridos nos componentes para o código escrito em Python (no GromaXy) que define o funcionamento da ferramenta desenvolvida.

Atualmente os usuários utilizam o GROMACS por meio da criação de *Script* usando a linguagem Shell, e nessa Dissertação, o GromaXy realiza esta etapa que pode ser um tanto complexa para o usuário, deixando-o interagir somente com a interface gráfica do GromaXy. Para a integração do GROMACS com o GromaXy foi necessário guiar o processo de desenvolvimento baseado na arquitetura original do Galaxy, porém acrescentando o *Script* em Shell que é manipulado pela *engine* desenvolvida em *Python*. A Figura 5.2 representa esta atual arquitetura, apresentando a interação do usuário com o GromaXy, destacando a composição do GromaXy e execução da simulação dentro do diretório que é gerado baseado na *tag* criada para identificar a simulação.

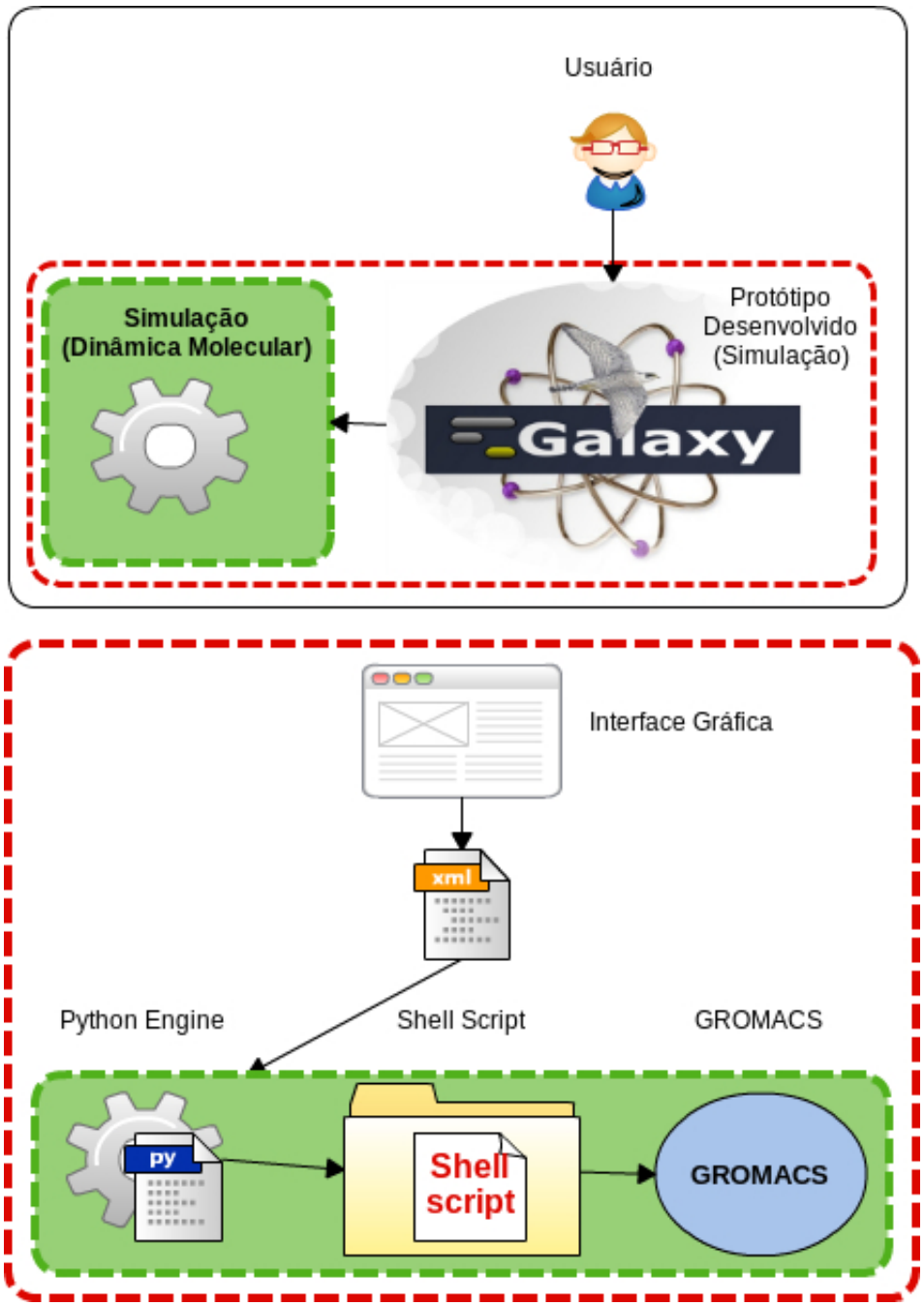

Figura 5.2: Representação da arquitetura de software do GromaXy

Um problema encontrado no armazenamento de dados foi a manipulação dos dados que é nomeado e armazenado automaticamente pelo Galaxy. Para suprir esta dificuldade foi criado um método já mencionado, podendo nos próximos passos deste trabalho alterar o método original de armazenamento do Galaxy apenas para identificar cada simulação por usuário e permitir o download do resultado, visando aproveitar todos os recursos do modelo de armazenamento padrão do Galaxy.

O processo de *upload* do Galaxy não reconhece o tipo de arquivo PDB, portanto foi ne-

cessário alterar o arquivo XML que define o tipo de dados a ser reconhecido na ferramenta de *upload* do Galaxy para que ele possibilite fazer a submissão desse tipo de arquivos. O Trecho de código 5.5 apresenta a alteração que foi realizada para o reconhecimento do tipo de arquivo PDB.

Código-fonte 5.5: Definição do tipo de dados PDB

```
1 <?xml version="1.0"?>
2 <datatypes>
3   <registration converters_path="lib/galaxy/datatypes/converters"
        display_path="display_applications">
4     <datatype extension="pdb" type="galaxy.datatypes.sequence:Pdb"
          display_in_upload="true"/>
5   </registration>
6   <sniffers>
7     <sniffer type="galaxy.datatypes.sequence:Pdb"/>
8   </sniffers>
9 </datatypes>
```

A criação do tipo de dados PDB foi necessário também para que o Galaxy consiga ao realizar o processo de submissão permitir o usuário visualizar o conteúdo do arquivo antes de iniciar uma simulação. Essa alteração realizada no Galaxy pode ser vista como uma solução temporária, portanto em trabalhos futuros podem ser estudadas outras formas de fazer o Galaxy reconhecer o tipo de arquivo PDB.

Durante o desenvolvimento do GromaXy ele foi adaptado para permitir que o usuário utilizasse o identificador da proteína (ID) em estudo, para que o GromaXy efetuasse o *download* da proteína diretamente do banco de dados PDB, desconsiderando a necessidade do usuário fazer esse processo manualmente e até armazenar em sua máquina para após fazer *upload* no Galaxy.

Junto ao usuário foi identificado que poucas proteínas do banco de dados PDB estão completas, muitas vezes é necessário que o usuário complete sua estrutura antes de usar o arquivo para iniciar uma simulação de DM, excluindo a necessidade de colocar o GromaXy para efetuar *download* sempre diretamente do banco de dados, deixando então a opção para o usuário baixar a proteína em seu computador, fazer as alterações necessárias, para somente após fazer *upload* do arquivo no Galaxy para usar no GromaXy. Esses são os fatos que levaram a adoção do software Atom Validation para essa Dissertação.

Como já mencionado, o Atom Validation foi adaptado para ser usado no GromaXy, permitindo o usuário selecionar todos os campos de força disponíveis no GROMACS, e não somente

o campo de força CHARMM27, que é o atual campo de força que o Atom Validation possibilita que o usuário utilize para validar suas proteínas.

A subseção 5.3.2 apresenta as modificações realizadas no Atom Validation que foram necessárias para expandir seu uso para vários campos de força, como também para disponibilizar a ferramenta no GromaXy. É importante destacar que o Atom Validation em sua versão original não possui Interface Gráfica de Usuário (GUI), o que torna necessário o usuário interagir com uma Interface de Linha de Comandos (CLI), como no GROMACS. Esses pontos reforçam os argumentos que indicam que essa adaptação além de aumentar a possibilidade de uso do Atom Validation por parte do usuário, pode aumentar também seu nível de usabilidade.

5.3.2 Adaptação e modificação do Atom Validation no Galaxy

Antes de utilizar o *script* do Atom Validation dentro do ambiente do Galaxy, foi necessário fazer as devidas alterações no *script* original, permitindo que ele ofereça ao usuário a opção de escolher um campo de força específico de uma lista de todos os disponíveis no GROMACS. Atualmente o Atom Validation só possibilita o uso do campo CHARMM27.

Foi necessário modificar alguns trechos de códigos para que o Atom Validation consiga identificar os arquivos necessários dentro do diretório da ferramenta desenvolvida no Galaxy, uma vez que a nova ferramenta irá utilizar desses mesmos arquivos.

O Trecho de código 5.6 apresenta uma alteração que foi necessária para que o Atom Validation possa encontrar o arquivo com a lista de possíveis aminoácidos a serem utilizados durante a validação. Essa lista de aminoácidos é usada no *script* para determinar o que é ou não um aminoácido no arquivo de campo de força, já que o arquivo pode conter não somente dados de aminoácidos. A variável 'path_atom_validation' contém o caminho da ferramenta dentro do Galaxy.

Código-fonte 5.6: Caminho para a lista de aminoácidos

```
1 path_amino = str(path_atom_validation+'aminoacids.txt')
```

Assim como foi necessário modificar seu código para indicar onde estará o arquivo de aminoácidos dentro do Galaxy, também foi necessário realizar uma modificação para permitir que a ferramenta encontre o arquivo que contém os dados do campo de força escolhido pelo usuário. O Trecho de código 5.7 apresenta essa alteração realizada. A variável 'force_field' armazena o valor escolhido pelo usuário, no caso o nome do campo de força escolhido, e esse é

concatenado com o caminho em que os campos de força se encontram dentro Galaxy.

Código-fonte 5.7: Caminho para o arquivo de campo de força

```
path_force_field = str(path_atom_validation+force_field+'.ff/'+'aminoacids.
    rtp')
```

Durante a alteração do código do Atom Validation, foi identificado que o seu código original não permite que ocorra a leitura de todos os campo de força do GROMACS, uma vez que alguns campos de força possui uma estrutura diferente do campo CHARMM27, o único campo que atualmente está disponível no Atom Validation. Para isso foi necessário estudar o código do *software* afim de identificar em qual ponto poderia ser realizada uma mudança para permitir que ele consiga realizar a leitura de qualquer campo de força, de forma genérica, ou seja, independente do campo de força escolhido pelo usuário. O Trecho de código 5.8 apresenta o código que é responsável pela leitura das linhas necessárias desse arquivo.

Código-fonte 5.8: Código responsável pela Expressão Regular

```
while  readline !=' [ bonds ]\n':
    if readline !=' [ atoms ]\n':
        readline = ER.sub(' ',readline)
        readline = re.search(r'^\s+\w+\s',readline)
        readline = readline.group()
        readline = ER.sub('',readline)
        atoms.append(readline)
    readline = ForceFieldBase.readline()
```

No código do Atom Validation é utilizado recursos de Expressão Regular da linguagem Python, e por conta de uma regra definida em sua expressão regular que estava gerando erros ao tentar ler alguns campos de força diferente do CHARMM27. O erro identificado foi que a expressão regular definia ao processo, para ignorar apenas o primeiro espaço em branco encontrado na linha em questão, porém alguns arquivos de campos força possuía mais que um espaço em branco ao início da linha, o que consecutivamente ocasionava a geração de um erro. O Trecho de código 5.9 apresenta o código já atualizado.

Código-fonte 5.9: Linha responsável pela Expressão Regular

```
readline = re.search(r'^\s+\w+\s',readline)
```

Observando o Trecho de código 5.9 é possível identificar a expressão regular usada, e para essa foi necessário apenas inserir um sinal de positivo (**+**) após a letra **s**. A letra **s** na expressão regular indica que deve ser ignorado o primeiro espaço em branco encontrado, já o sinal de positivo (**+**) após a letra, indica que deve ser ignorado um ou mais espaços em branco quando encontrado. Essa alteração foi necessária para permitir o Atom Validation conseguir ler o arquivo que contém os dados do campo de força escolhido.

Após adaptar o Atom Validation para permitir que o usuário escolha qualquer um dos campo de força disponíveis no GROMACS, foi necessário modificar o *script* resultante da adaptação para o Galaxy, criando assim uma nova ferramenta, com uma nova interface, diferente da interface original do Atom Validation.

Durante o desenvolvimento da nova ferramenta no Galaxy com o *script* adaptado do Atom Validation, foi detectado alguns problemas, como dificuldade em encontrar a biblioteca do Biopython na máquina que está instanciado o Galaxy. A biblioteca é utilizada no código do Atom Validation, e indispensável para seu funcionamento. Para isso foi criada uma instalação da biblioteca de forma dedicada somente ao ambiente do Galaxy, para que esse possa reconhecê-la.

Assim como o criada as ferramentas que já compõem o GromaXy, foi criada uma nova ferramenta dentro do diretório do GromaXy, para oferecer ao usuário os recursos do Atom Validation adaptado. Os mesmos processos foram adotados para todas as ferramentas, assim como foram usadas as mesmas linguagens: XML e Python. .

Na versão original do Atom Validation ele busca o arquivo do campo de força dentro do diretório de seu *script*, mas como para a nova ferramenta desenvolvida nessa Dissertação deve permitir a escolha de vários campos de força, foi necessário fazer uma cópia de cada arquivo de campo de força do GROMACS e armazenar em um diretórios com o seu nome, portanto foi necessário dentro do diretório do GromaXy criar outros novos diretórios. Essas cópias de arquivos foram realizadas apenas para fazer os testes necessários para o bom funcionamento da ferramenta, porém em versões futuras é interessante buscar o arquivo de campo de força sempre dentro do GROMACS instalado na máquina, considerando que o GROMACS pode ter sua versão atualizada e os dados de cada campo de força também, assim como podem surgir novos campos de força futuramente, e com isso o GromaXy estará atualizado automaticamente.

A Figura 5.3 apresenta a tela principal do Atom Validation adaptado, já dentro do Galaxy, funcionando junto ao GromaXy. Para a apresentação da ferramenta foram submetidos dois arquivos PDB, com proteínas distintas, que foram extraídas do banco de dados PDB. Na Figura 5.3 é possível visualizar em uma lista as duas proteínas disponíveis.

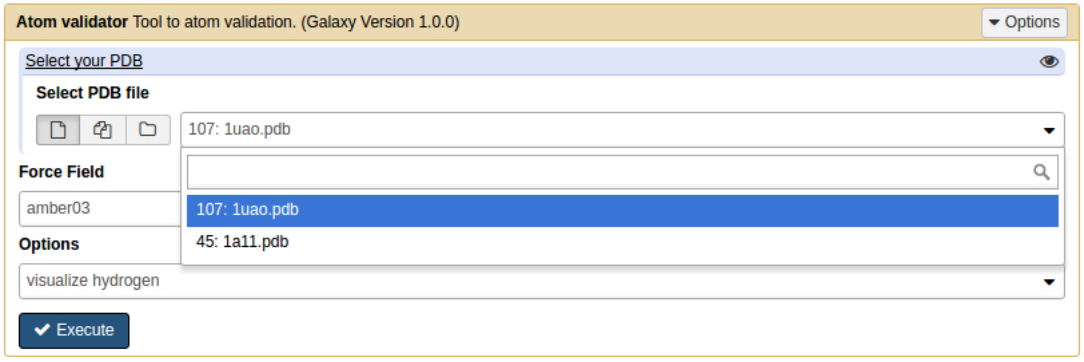

Figura 5.3: Tela principal do Atom Validation adaptado, destacando a escolha do arquivo PDB

Já a Figura 5.4 apresenta a tela principal do Atom Validation adaptado, porém apresentando a ferramenta com uma lista dos campos de força disponíveis para o usuário utilizar. Na figura não foi possível apresentar todos, pelo motivo da lista ser maior que o elemento pode apresentar. É necessário que o usuário utilize a barra de rolagem para visualizar todos os campos de força disponíveis.

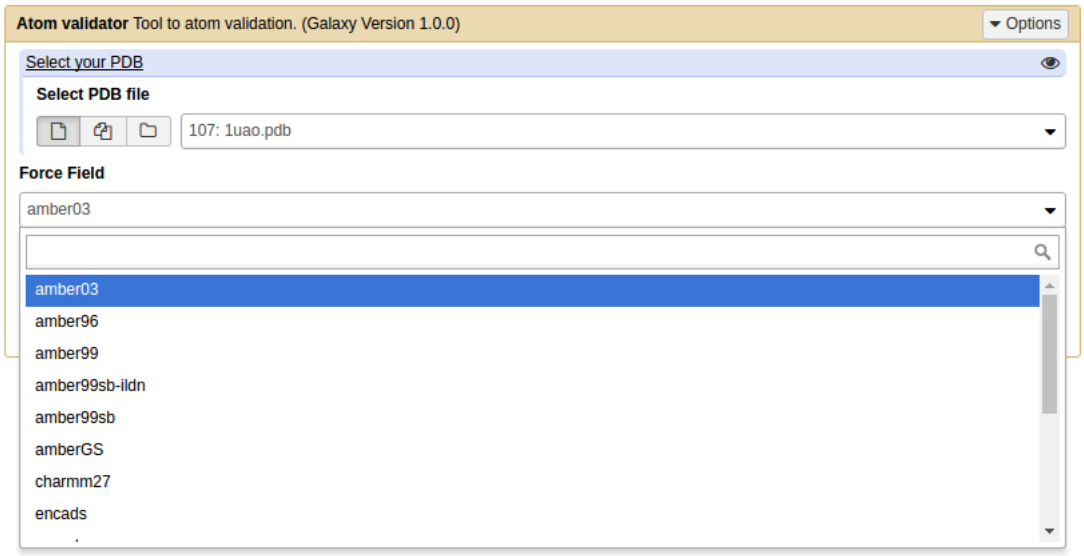

Figura 5.4: Tela principal do Atom Validation adaptado, destacando a escolha do arquivo campo de força

Além das modificações já citadas, o *script* do Atom Validation foi alterado para que ao invés de imprimir as mensagens para o usuário ao longo de sua execução, essas mesmas mensagens sejam armazenadas em um arquivo, e ele seja disponibilizado ao usuário ao final da execução, para que ele possa analisar o resultado da validação. Esse recurso foi pensado no sentido do usuário poder querer alterar os dados do arquivo, conforme vai atualizando seu arquivo PDB, que foi usado na validação. Apenas apresentar o resultado da validação em uma

tela no GromaXy poderia não ser uma boa opção, já que o usuário não teria opção de alterar esses dados.

O Atom Validation além de imprimir as informações da validação para o usuário, também organiza essas informações em arquivos XML, porém para essa Dissertação estão sendo disponibilizadas ao usuário apenas as mensagens, por considerar que o usuário do GromaXy pode não se familiarizar com o formado de arquivo XML, o que pode gerar gerar uma dificuldade ao interpretar os dados. Em trabalhos futuros esses arquivos podem ser disponibilizados, como também podem ser usados por outras novas ferramentas.

Assim como para o GromaXy, para utilizar a adaptação do Atom Validation é necessário fazer *upload* de um arquivo PDB para o ambiente do Galaxy, e para isso o Galaxy oferece uma ferramenta para essa finalidade. A Figura 5.5 apresenta o processo de envio concluido de duas proteínas que foram submetidas para o Galaxy.

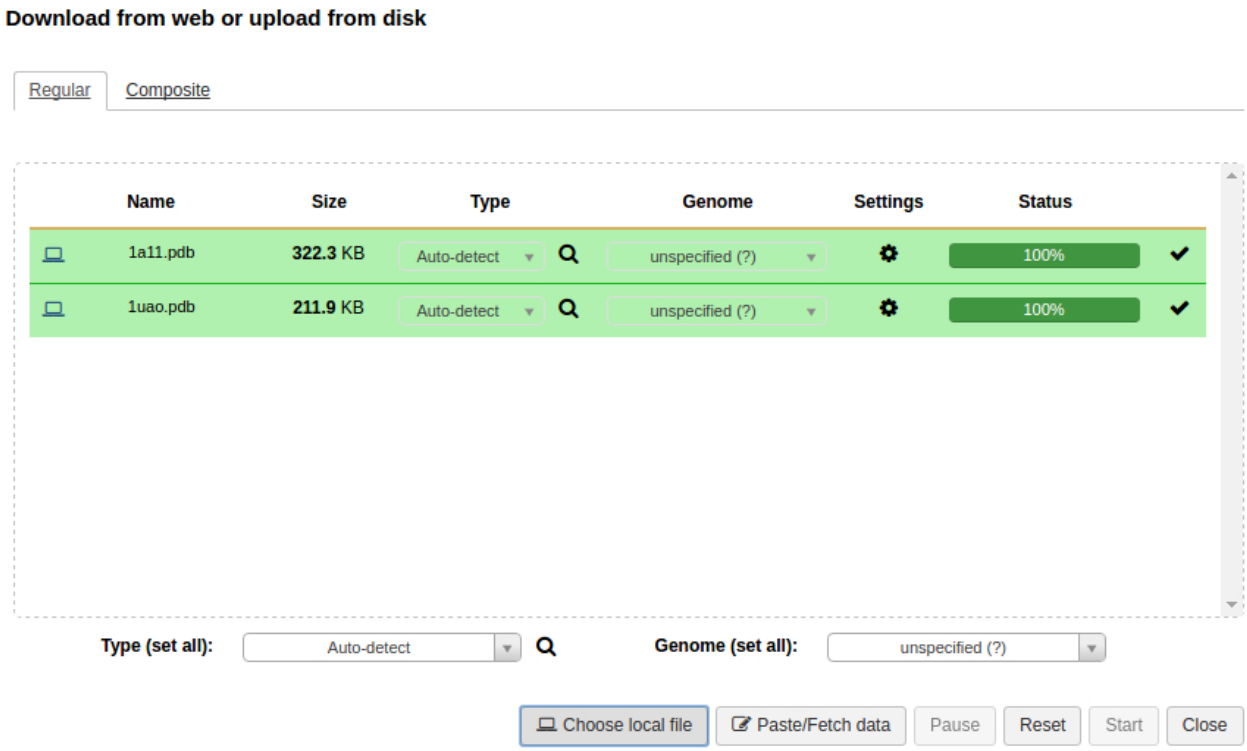

Figura 5.5: Tela de *upload* do Galaxy, destacando dois arquivos PDB que foram submetidos

Após a submissão de um arquivo PDB para o Galaxy, é criado um *job* para cada arquivo que foi enviado, e nesse *job* há a opção de visualizar o conteúdo do arquivo. Essa opção pode ser acessada pelo usuário, para que ele possa verificar o conteúdo do arquivo PDB. Se o processo de submissão for concluído com sucesso, o mesmo é renderizado no ambiente na cor verde, caso contrário ele terá a cor vermelha, que indicará erros que poderão ser visualizados pelo

usuário. A Figura 5.6 apresenta dois *jobs* que foram criados no Galaxy ao final do processo de submissão dos arquivos PDB.

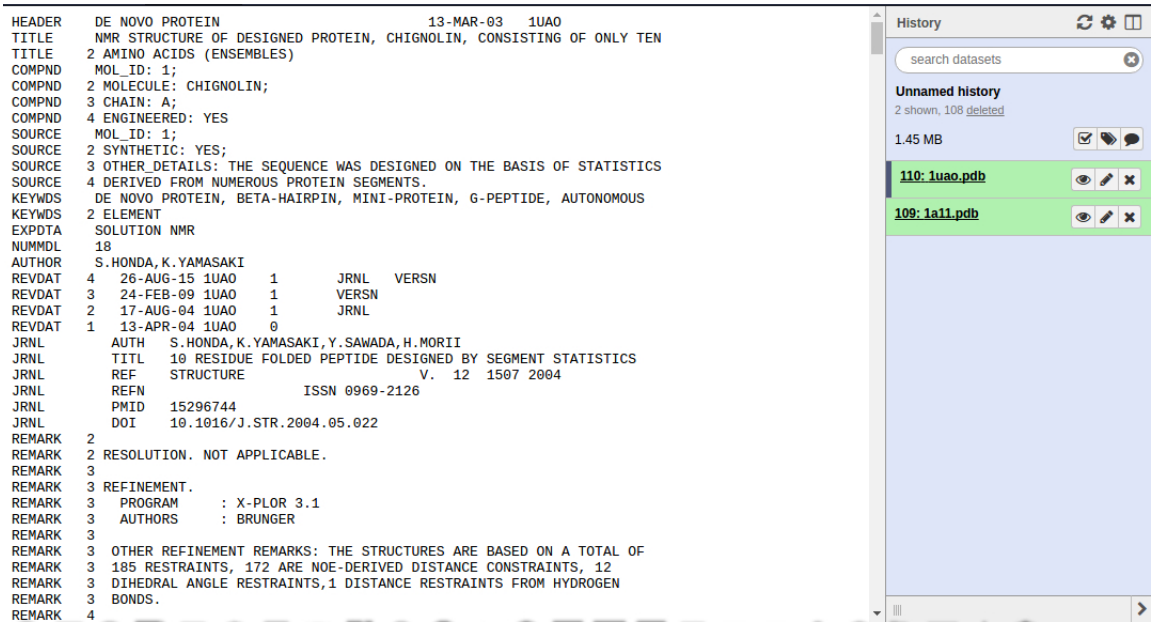

Figura 5.6: Apresentação do conteúdo de um dos dois arquivos PDB submetidos

Com arquivos PDB já disponíveis no Galaxy é possível acessar a ferramenta desenvolvida e configurar uma validação, selecionando o arquivo PDB desejado, o campo de força e até outras opções. Todas os arquivos PDB que foram submetidos ao Galaxy ficam disponíveis para uso, quando os *jobs* associados a suas submissões não foram removidos do histórico. A Figura 5.7 apresenta a ferramenta desenvolvida junto ao histórico de *jobs* que é possível visualizar os dois *jobs* associados aos dois arquivos PDB submetidos ao Galaxy.

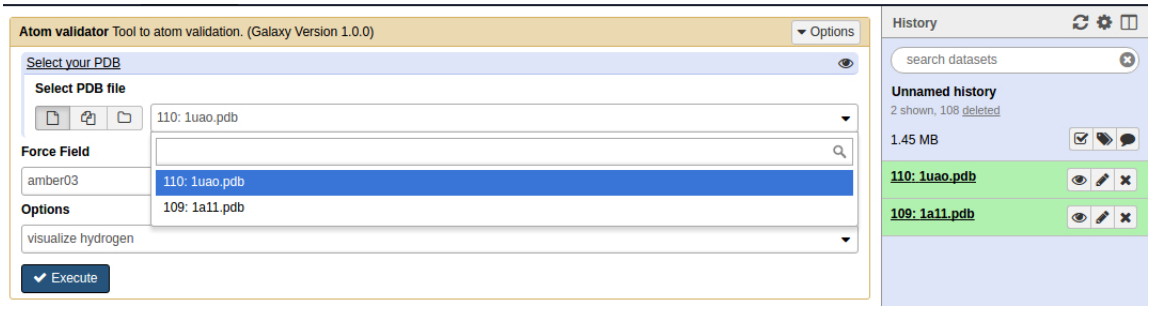

Figura 5.7: Tela principal do Atom Validation adaptado com histórico de *jobs*

Ao escolher a proteína desejada para validação e selecionar as opções necessárias, é possível executar a validação, e após o início de sua execução é criado um novo *job* associado a validação, e esse se tiver sucesso em sua conclusão ficará na cor verde e oferecerá ao usuário a opção de *download* dos resultados, que no caso serão as informações relacionadas aos átomos que estão

faltando na estrutura da proteína, e até mesmo os átomos classificados como desconhecidos. A Figura 5.8 apresenta o ambiente após a conclusão da validação, destacando ao centro do ambiente uma mensagem padrão gerada pelo Galaxy, a direita o *job* da validação e logo acima do histórico o arquivo que contém os resultados da validação. Foi realizado o *download* desse arquivo para visualizar externamente.

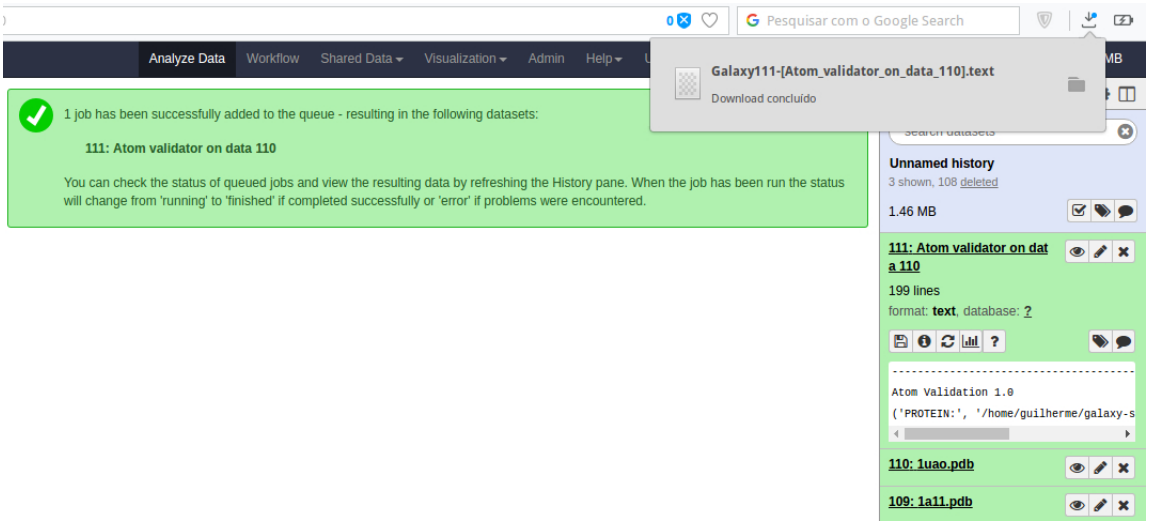

Figura 5.8: Mensagem de conclusão da validação, histórico de *jobs* e arquivo com os resultados da validação

5.4 Considerações Finais

Na Subseção 5.3.2 além de apresentar algumas das adaptações e modificações que foram necessárias realizar no Atom Validation, foi apresentado também o resultado final da ferramenta desenvolvida. Essa ferramenta foi apresentada apenas como objetivo de demonstrar a diferença entre o Atom Validation padrão e sua versão modificada e adaptada dentro do ambiente do Galaxy, na ferramenta GromaXy. O capítulo 6 apresenta algumas validações que foram realizadas usando o Atom Validation e sua versão adaptada no GromaXy.

Capítulo 6

VALIDAÇÃO

Este capítulo apresenta uma comparação entre o Atom Validation tradicional e sua adaptação. Para essa comparação são apresentadas algumas validações que foram realizadas em ambas as ferramentas, afim de validar a proposta de sua adaptação e inserção dentro do ambiente do Galaxy, juntamente com o GromaXy.

Nesse capítulo são apresentadas algumas validações usando duas proteínas distintas, que foram selecionadas apenas para demonstração de uso da ferramenta e comparação do resultado final entre o Atom Validation e sua adaptação.

Na Seção 6.1 estão apresentadas algumas validações que foram realizadas usando o campo de força CHARMM27, afim de demonstrar a funcionalidade da ferramenta e possibilitar a realização de comparações e viabilidade do uso de sua adaptação no GromaXy. Já na Seção 6.2 são apresentadas validações que foram realizadas apenas no GromaXy, usando todos os campos de força diferentes do CHARMM27.

Para realizar o processo de validação usando a nova ferramenta foi necessário obter os arquivos PDB desejados. Esses arquivos contém os dados das coordenadas atômicas de duas proteínas: as proteínas com identificador (*id*) 1UAO (HONDA et al., 2004) e 1A11 (OPELLA et al., 1999). Essas proteínas encontram-se depositadas no banco de dados PDB.

Afim de ilustrar as estruturas das proteínas 1UAO e 1A11, a Figura 6.1 apresenta lado a lado uma ilustração gráfica de suas estruturas. Para gerar as ilustrações foi usado o *software* PyMOL (DELANO, 2002), que permite fazer a leitura de um arquivo PDB e apresentar graficamente a estrutura da proteína. Essas ilustrações foram inseridas no texto apenas para conduzir melhor o leitor em seu universo de imaginações, portanto não tem objetivo de serem usadas para ilustrar comparações de resultados nessa Dissertação.

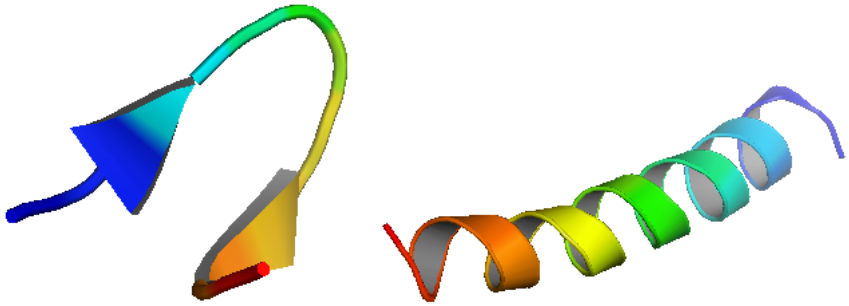

Figura 6.1: Estruturas das proteínas: 1UAO (a esquerda), e 1A11 (a direita).

Na Seção 6.1 são apresentados os dados de saída (*outputs*), após a execução do Atom Validation, em sua versão original e sua versão adaptada no GromaXy. Os dados foram inseridos em figuras e estão apresentados lado a lado, para facilitar a comparação entre os *outputs* de ambas ferramentas.

É importante reforçar que na versão adaptada do Atom Validation, não houve alteração na forma com que os dados são apresentados e no processo de validação. Foram realizadas apenas alterações necessárias para adaptar o *software* para funcionar com o GromaXy e possibilitar o uso de todos os campos de força do GROMACS. Portanto, o Atom Validation e sua versão adaptada apresentam os dados da validação seguindo a seguinte estrutura:

- cadeias da Proteína;
- resíduos da cadeia; e
- os átomos do resíduo.

Os arquivos estão divididos em duas partes, que estão separados apenas por uma linha tracejada. Essa linha tracejada divide a apresentação de duas informações:

- Apresentação dos átomos desconhecidos (*atoms_unknown*); e
- Apresentação dos átomos ausentes (*atoms_absence*).

Nesse capítulo, nos *outputs* apresentados, foram inseridas apenas as cinco (5) primeiras linhas referentes aos átomos de cada resíduo, no caso, 5 linhas de átomos ausentes e 5 linhas de átomos desconhecidos. Esse processo foi adotado apenas pelo motivo do arquivo ser muito extenso para apresentar nessa Dissertação por completo. Os três pontos (...) inseridos em cada arquivo foram inseridos manualmente, e esses representam apenas que há mais resíduos no arquivo original que foi extraído após o processo de validação.

No trabalho de (LEITE, 2012) é descrito que o Atom Validation não completa o arquivo PDB com os átomos ausentes, portanto a adaptação do Atom Validation segue essa mesma ideia, por esse motivo essa é uma tarefa que deve ser realizada pelo usuário, seja manualmente ou com o auxílio de um *software* terceiro.

6.1 Validações usando apenas o campo CHARMM27

Nessa seção são apresentadas apenas validações usando o campo CHARMM27, uma vez que apenas esse campo está disponível na versão original do Atom Validation, o que impede de comparar o *software* com sua versão adaptada, usando todos os campos de força do GROMACS.

Na subseção 6.1.1 são apresentadas validações usando a proteína 1UAO, com o Atom Validation e sua versão adaptada. Já na subseção 6.1.2 são apresentadas validações usando a proteína 1A11.

Para usar o Atom Validation é necessário interagir com o ”terminal”do sistema operacional Linux, necessitando escrever uma ”linha de comandos”, para que o sistema interprete e execute uma chamada no sistema, do interpretador padrão da linguagem Python, o Atom Validation e seus parâmetros.

Para executar o Atom Validation, foi necessário escrever no ”terminal”do sistema operacional, uma ”linha de comandos”, para que o sistema interprete e execute uma chamada no sistema, do interpretador padrão da linguagem Python, o Atom Validation e seus parâmetros. Tanto na subseção 6.1.1, quanto na 6.1.2, são apresentados exemplos dos comandos que foram executados para realizar a validação das proteínas.

6.1.1 Validações usando a proteína 1UAO

O Trecho de código 6.1 apresenta o comando que foi executado para validar a proteína 1UAO, gerando um arquivo de saída com o nome ’saida.txt’, para facilitar a apresentação dos resultados. O processo original apresenta os dados de saída no próprio terminal do Linux. Dessa forma, ele não gera um arquivo de saída, e sim gera a impressão dos dados de saída (*output*) na própria seção do ”terminal”que o usuário está interagindo.

Código-fonte 6.1: Código do comando usado para validar a proteína 1UAO no Atom Validation

```
1 python AtomValidation.py 1uao.pdb -v > saida.txt
```

Também foi executado o comando apresentado no Trecho de código 6.1, porém sem gerar seu *output* no arquivo 'saida.txt'. A Figura 6.2 apresenta uma imagem da tela após a execução do Atom Validation, usando a proteína 1UAO e o parâmetro '**-v**', que indica ao programa que o usuário deseja também visualizar os átomos de Hidrogênio (H). Por meio dessa figura é possível visualizar que o Atom Validation não possui Interface Gráfica de Usuário (GUI), e sim Interface de Linha de Comando.

```
guilherme@guilherme-Rev-1-0:~/Downloads/gilson/final$ python AtomValidation.py 1uao.pdb -v


--------------------------------------------------
Atom Validation 1.0
PROTEIN: 1uao
CHAIN:  ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A']
RESIDUE: 180
ATOM:   2484

Expresion:
Residue model chain residue_number atoms_unknown
('ASP', 17, 'A', 3) ['H', 'HB3']
('THR', 16, 'A', 8) ['H']
('GLY', 4, 'A', 1) ['H1', 'H2', 'H3', 'HA3']
('GLY', 5, 'A', 10) ['H', 'HA3']
('THR', 2, 'A', 8) ['H']
('TRP', 7, 'A', 9) ['H', 'HB3']
('GLY', 9, 'A', 1) ['H1', 'H2', 'H3', 'HA3']
('TYR', 2, 'A', 2) ['H', 'HB3']
('THR', 1, 'A', 6) ['H']
('ASP', 5, 'A', 3) ['H', 'HB3']
('TYR', 17, 'A', 2) ['H', 'HB3']
('TRP', 17, 'A', 9) ['H', 'HB3']
('THR', 6, 'A', 6) ['H']
('GLY', 9, 'A', 10) ['H', 'HA3']
('GLY', 2, 'A', 10) ['H', 'HA3']
('GLY', 8, 'A', 10) ['H', 'HA3']
('THR', 13, 'A', 8) ['H']
('TRP', 14, 'A', 9) ['H', 'HB3']
('ASP', 16, 'A', 3) ['H', 'HB3']
('TYR', 12, 'A', 2) ['H', 'HB3']
('TYR', 15, 'A', 2) ['H', 'HB3']
('TYR', 5, 'A', 2) ['H', 'HB3']
('TRP', 16, 'A', 9) ['H', 'HB3']
('ASP', 8, 'A', 3) ['H', 'HB3']
('GLU', 11, 'A', 5) ['H', 'HB3', 'HG3']
```

Figura 6.2: Validação da proteína 1UAO usando o Atom Validation

Como já descrito no texto dessa Dissertação, na Figura 6.3 estão apresentados lado a lado parte dos *outputs* das validações usando o Atom Validation e sua adaptação. Do lado esquerdo está o *output* do Atom Validation, já do lado direito o *output* de sua adaptação. Para cada validação, é possível notar que os trechos de texto 'Atom Validation 1.0' aparecem duas vezes, separado por uma linha tracejada. Essa linha tracejada indica a divisão entre os dois tipos de validações: apresentação dos átomos desconhecidos (*atoms_unknown*), e apresentação dos átomos ausentes (*atoms_absence*).

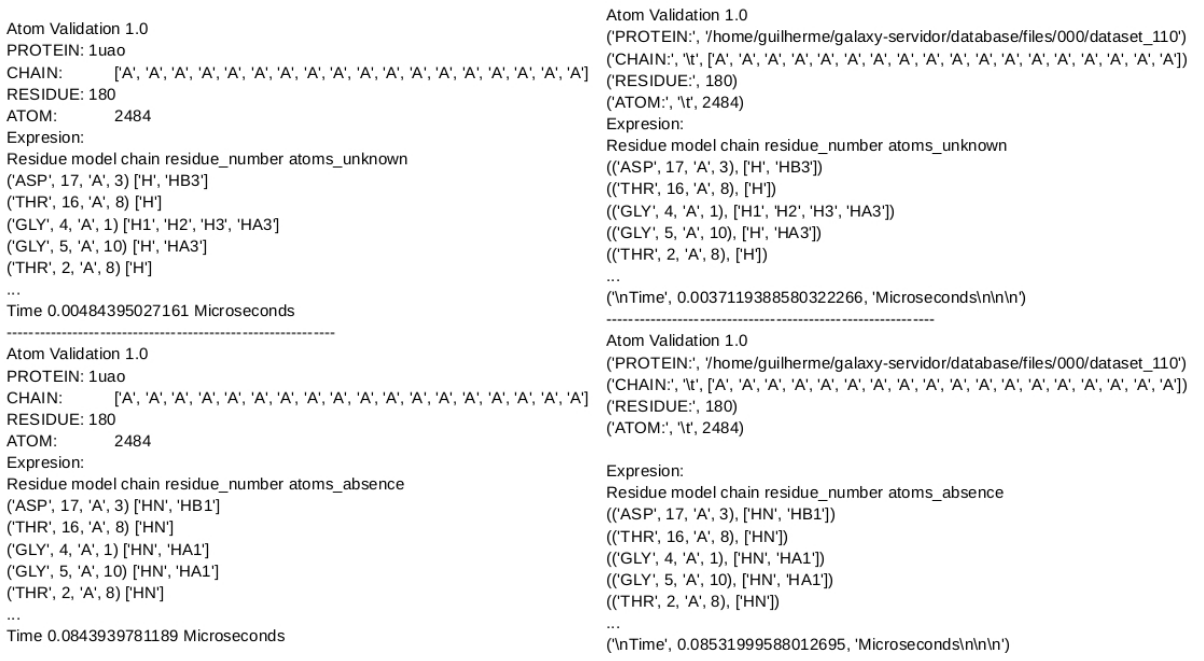

Figura 6.3: Comparação do *output* do Atom Validation e de sua versão adaptada, usando a proteína 1UAO e o campo de força CHARMM27. A esquerda o *output* do Atom Validation, a direita do GromaXy

6.1.2 Validações usando a proteína 1A11

Assim como na subseção 6.1.1, foi necessário executar o Atom Validation no ”terminal”do sistema operacional para iniciar sua execução, e esse está apresentado no Trecho de código 6.2.

Código-fonte 6.2: Código do comando usado para validar a proteína 1A11 no Atom Validation

```
1 python AtomValidation.py 1a11.pdb -v > saida.txt
```

Após a execução do comando apresentado no Trecho de código 6.2, os dados são gerados no arquivo de saída como o nome ’saida.txt’. Para exemplificar como seria a saída seguindo o processo de execução padrão do Atom Validation no linux, a Figura 6.4 ilustra a tela após a execução do Atom Validation, usando a proteína 1A11 e o parâmetro ’**-v**’, que indica ao programa que o usuário também deseja visualizar os átomos de Hidrogênio (H).

```
guilherme@guilherme-Rev-1-0:~/Downloads/gilson/final$ python AtomValidation.py 1a11.pdb -v


--------------------------------------------------
Atom Validation 1.0
PROTEIN: 1a11
CHAIN:  ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A']
RESIDUE: 250
ATOM:   3900

Expresion:
Residue model chain residue_number atoms_unknown
('ALA', 2, 'A', 8) ['H']
('ALA', 7, 'A', 16) ['H']
('LYS', 4, 'A', 4) ['H', 'HB3', 'HG3', 'HD3', 'HE3']
('ALA', 5, 'A', 16) ['H']
('LEU', 9, 'A', 19) ['H', 'HB3']
('GLN', 6, 'A', 24) ['H', 'HB3', 'HG3']
('GLY', 4, 'A', 1) ['H1', 'H2', 'H3', 'HA3']
('VAL', 2, 'A', 17) ['H']
('ILE', 7, 'A', 9) ['CD1', 'H', 'HG13', 'HD11', 'HD12', 'HD13']
('LEU', 6, 'A', 21) ['H', 'HB3']
('LYS', 7, 'A', 4) ['H', 'HB3', 'HG3', 'HD3', 'HE3']
('GLY', 9, 'A', 1) ['H1', 'H2', 'H3', 'HA3']
('SER', 7, 'A', 23) ['H', 'HB3', 'HG']
('LEU', 6, 'A', 19) ['H', 'HB3']
('SER', 1, 'A', 2) ['H', 'HB3', 'HG']
('SER', 2, 'A', 2) ['H', 'HB3', 'HG']
('ALA', 4, 'A', 14) ['H']
('LEU', 6, 'A', 12) ['H', 'HB3']
('GLU', 9, 'A', 3) ['H', 'HB3', 'HG3']
('LEU', 0, 'A', 13) ['H', 'HB3']
('VAL', 4, 'A', 11) ['H']
('LEU', 9, 'A', 12) ['H', 'HB3']
('ALA', 2, 'A', 16) ['H']
('GLY', 5, 'A', 1) ['H1', 'H2', 'H3', 'HA3']
('SER', 7, 'A', 10) ['H', 'HB3', 'HG']
```

Figura 6.4: Validação da proteína 1A11 usando o Atom Validation

Na Figura 6.5 estão apresentados lado a lado parte dos *outputs* das validações usando o Atom Validation e sua adaptação. Do lado esquerdo está o *output* do Atom Validation, já do lado direito o *output* de sua adaptação.

```
Atom Validation 1.0
PROTEIN: 1a11
CHAIN:          ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A']
RESIDUE: 250
ATOM:           3900
Expresion:
Residue model chain residue_number atoms_unknown
('ALA', 2, 'A', 8) ['H']
('ALA', 7, 'A', 16) ['H']
('LYS', 4, 'A', 4) ['H', 'HB3', 'HG3', 'HD3', 'HE3']
('ALA', 5, 'A', 16) ['H']
('LEU', 9, 'A', 19) ['H', 'HB3']
...
Time 0.0126760005951 Microseconds
------------------------------------------------------------
Atom Validation 1.0
PROTEIN: 1a11
CHAIN:          ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A']
RESIDUE: 250
ATOM:           3900
Expresion:
Residue model chain residue_number atoms_absence
('ALA', 2, 'A', 8) ['HN']
('ALA', 7, 'A', 16) ['HN']
('LYS', 4, 'A', 4) ['HN', 'HB1', 'HG1', 'HD1', 'HE1']
('ALA', 5, 'A', 16) ['HN']
('LEU', 9, 'A', 19) ['HN', 'HB1']
...
Time 0.265933036804 Microseconds
```

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['H'])
(('ALA', 7, 'A', 16), ['H'])
(('LYS', 4, 'A', 4), ['H', 'HB3', 'HG3', 'HD3', 'HE3'])
(('ALA', 5, 'A', 16), ['H'])
(('LEU', 9, 'A', 19), ['H', 'HB3'])
...
('\nTime', 0.005525112152099609, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ALA', 2, 'A', 8), ['HN'])
(('ALA', 7, 'A', 16), ['HN'])
(('LYS', 4, 'A', 4), ['HN', 'HB1', 'HG1', 'HD1', 'HE1'])
(('ALA', 5, 'A', 16), ['HN'])
(('LEU', 9, 'A', 19), ['HN', 'HB1'])
...
('\nTime', 0.13966608047485352, 'Microseconds\n\n\n')
```

Figura 6.5: Comparação do *output* do Atom Validation e de sua versão adaptada, usando a proteína 1A11 e o campo de força CHARMM27. A esquerda o *output* do Atom Validation, a direita do GromaXs

Tanto a Figura 6.5, quanto 6.5, ambas apresentam apenas parte dos dados resultantes das simulações, considerando apresentar ao leitor apenas o *output* de validações usando o Atom Validation e sua versão adaptada no GromaXy. O objetivo dessas informações também é apresentar que o processo original do Atom Validation não foi modificado no GromaXy, de forma a afetar os resultados originais do Atom Validation. Na Seção 6.2, também são apresentados apenas parte dos dados resultantes das simulações.

6.2 Validações usando apenas o GromaXy

Além das validações das proteínas 1UAO e 1A11 usando o campo de força CHARMM27, também foram realizadas validações com outros campos de força presentes no GROMACS. Pelo motivo do Atom Validation não possuir outros campos de força disponíveis além do CHARMM27, não foi possível realizar comparação de *output* do Atom Validation com o GromaXy, quando selecionados todos os campos do GROMACS.

Nesta seção são apresentadas uma validação para cada campo de força. Algumas foram realizadas usando a proteína 1UAO, já outras usando a proteína 1A11. Seria ideal realizar uma validação para cada campo de força usando uma proteína, porém para evitar apresentar um conjunto muito grande de informações para o leitor, apenas uma parcela das validações são

apresentadas usando a proteína 1UAO e as outras restantes usando a proteína 1A11.

Nessa Dissertação, foram realizadas uma validação no GromaXy para cada campo de força do GROMACS. Algumas foram realizadas usando a proteína 1UAO, já outras usando a proteína 1A11. Seria ideal realizar uma validação para cada campo de força usando uma proteína, porém para evitar apresentar um conjunto muito grande de informações para o leitor, apenas uma parcela das validações são apresentadas usando a proteína 1UAO e as outras restantes usando a proteína 1A11. A Tabela 6.1 apresenta a ordem das simulações, ilustrando cada proteína e o campo de força que foi utilizado para a validação.

Tabela 6.1: Apresentação das validações usando o GromaXy

Início da Tabela			
Proteína	Campo de força	Proteína	Campo de força
1UAO	Amber03	1A11	Gmx
	Amber94		Gmx2
	Amber96		Gromos43a2
	Amber99		Gromos45a3
	Amber99sb-ildn		Gromos53a5
	Amber99sb		Gromos53a6
	AmberGS		Gromos54a7
	Encads		Oplsaa
	Encadv		
Final da Tabela			

O Apêndice A, apresenta alguns documentos com parte do *output* gerado na validação após execução do GromaXy. Esses documentos apresentam apenas parte do conteúdo apenas com o objetivo de demonstrar o funcionamento da ferramenta desenvolvida, sem dar enfoque a todos os dados gerados, o que ocasionaria na geração de um volume muito grande dados a serem apresentados no Apêndice.

Os dados apresentados no Apêndice A, seguem a mesma estrutura dos arquivos apresentados na Seção 6.1. Os documentos estão apresentados por Seções e Subseções, seguindo a estrutura Seções (Apresentações a nível de proteína), e Subseções (Apresentações a nível de Campo de força).

6.3 Considerações Finais

Durante o processo de validação da proteína 1A11 usando o campo de força Oplsaa, foi identificado que o a ferramenta ao montar a lista de átomos ausentes para alguns resíduos, não foram inseridos os átomos e sim uma *String* com o conteúdo 'atoms'. Com a finalidade de identificar o erro e verificar se o erro foi gerado por conta da adaptação realizada no Atom Validation, porém foi identificado que o erro já existe na versão original do Atom Validation, e esse pode ser corrigido em trabalhos futuros. Para evitar qualquer alteração nos resultados atuais do Atom Validation, nesse trabalho foi evitado alterar a parte funcional do *software*.

Nas Figuras 6.3 e 6.5, é possível notar o tempo de execução total da validação para cada caso, e observar que os tempos de execuções entre o Atom Validation e sua versão adaptada dentro do ambiente Galaxy não possui tanta diferença, o que indica que o fato do *script* estar dentro do Galaxy pode não afetar o tempo de execução de forma que o usuário note tamanha diferença.

Capítulo 7

DISCUSSÕES FINAIS

7.1 Considerações Finais

Neste capítulo são apresentadas as considerações finais dessa Dissertação, as contribuições para o estado da arte em que essa pesquisa se encontra, e ao final, quais os trabalhos futuros que podem ser realizados, dando continuidade ao que já foi alcançado com essa Dissertação.

Atualmente, a interação com o GROMACS se dá por meio de interface de linha de comandos (CLI). Na literatura podem ser encontradas ferramentas que oferecem uma interface gráfica de usuário (GUI) para o GROMACS, porém essas em sua maioria precisam ser instaladas localmente na máquina do usuário, serem mantidas ao longo do tempo por meio atualizações e configuradas para que consigam se comunicar com o GROMACS, que também precisa ser instalado na máquina do usuário. Nesse sentido, o GromaXy tem seu diferencial, por poder ser instalado no servidor, e o usuário poder acessa-lo pelo seu *browser*, tirando dele a responsabilidade de instalação de qualquer das ferramentas necessárias para configurar e executar sua simulação.

Durante o amadurecimento do GromaXy, usuários GROMACS foram consultados periodicamente para que fosse possível realizar o desenvolvimento centrado nas reais necessidades cotidianas do usuário. Foi identificado uma dificuldade no uso do programa pdb2gmx que faz parte do pacote do GROMACS, pelo fato de muitas vezes o arquivo PDB, que contém a estrutura da proteína, não estar completo, ou seja, estar faltando átomos em sua estrutura. Como a estrutura de uma proteína pode variar desde uma estrutura simples até uma estrutura complexa, muitos átomos podem não estar presentes em sua estrutura, o que pode ocasionar em uma necessidade repetitiva de tentativas de execução do pdb2gmx até uma execução com sucesso, ou seja, sem interrupção, por conta de estrutura incompleta.

Cada vez que o usuário tenta executar o pdb2gmx usando um arquivo PDB incompleto, este identifica que está faltando átomos na estrutura da proteína, ele apresenta ao usuário o átomo que está faltando, e o usuário precisa remodelar a proteína, ou seja, precisa inserir o átomo ausente no arquivo PDB. Para cada átomo ausente é necessário executar o pdb2gmx, inserí-lo no arquivo PDB e executar o programa novamente. Esse processo se repete até que o usuário consiga completar totalmente a proteína, o que faz com que ocasionalmente o programa pdb2gmx aceite a estrutura já completa. Esse processo pode levar muito tempo, com relação a quantidade de átomos ausentes na estrutura.

7.2 Contribuições da Dissertação

A principal contribuição dessa dissertação foi a implementação e amadurecimento de uma ferramenta que provê uma nova forma de interação com o GROMACS, disponibilizando ao usuário recursos do Galaxy como: possibilitar o compartilhamento de histórico de trabalho e dados de experimentos realizados, possibilitar o armazenamento do alto volume de dados de trajetória resultantes do processo de simulação de dinâmica molecular e permitir a configuração de *workflow* de execução de ferramentas, onde o usuário pode configurar um ciclo de execução de várias ferramentas de forma lógica, baseado em suas regras definidas conforme dados de *input* e *output* de cada ferramenta. Foi dado o nome de GromaXy para essa ferramenta, por fazer junção aos nomes GROMACS e Galaxy.

Baseado na dificuldade do usuário em identificar os átomos ausentes na estrutura da proteína que está no seu arquivo PDB, o *software* Atom Validation foi adaptado no GromaXy, para que o usuário possa executar essa ferramenta afim de identificar todos ou parte dos átomos que estão ausentes na estrutura da proteína. O Atom Validation foi desenvolvido para validar proteínas usando somente o campo de força CHARMM27, porém nessa Dissertação ele foi adaptado para permitir que o usuário use qualquer um dos campos de força disponíveis no GROMACS, já que o usuário pode necessitar usar um campo de força que não seja o CHARMM27.

As contribuições dessa Dissertação vão além da área de Ciência da Computação, alcançando a Ciência em Geral, uma vez que os resultados dessa Dissertação pode impactar positivamente na área de Bioinformática Estrutural, auxiliando usuários, muitas vezes pesquisadores, em suas tarefas cotidianas de simulações de DM com o GROMACS, que podem ser usadas desde etapas de um processo de desenvolvimento de um fármaco e até mesmo estudos para compreensão do comportamento de diversas doenças.

7.3 Trabalhos Futuros

Para trabalhos futuros é possível e interessante amadurecer ainda mais o GromaXy junto ao usuário, baseado nas suas necessidades. Como o GROMACS pode ser usado de diversas formas, para várias finalidades e com diversos variações de configurações, é interessante amadurecer o GromaXy baseado em um grupo de usuários, não somente um e também sempre pensando em desenvolver de uma forma genérica, que possa atender uma grande parcela de usuários.

Assim como o Atom Validation e sua adaptação realizada nessa Dissertação, não completam o arquivo PDB com os átomos ausentes, necessitando do usuário depender do uso de *software* terceiro para auxiliar nessa tarefa. É interessante em trabalhos futuros explorar esse barreira existente, no sentido de desenvolver uma ferramenta, associada ao GromaXy, para ao validar a proteína, apontar os átomos ausentes e desconhecidos, e fornecer mecanismos ao usuário, para permitir que ele guie um processo automático de complementação do arquivo PDB.

Como o processo de Avaliação de Usabilidade foi pausado na Dissertação para focar em outro problema identificado junto ao usuário, é importante e até desejável que em trabalhos futuros esse processo seja retomado. Os resultados finais, mesmo que não comprovar aceitação do GromaXy por parte do usuário, os dados podem ser usados para atualizar o estado da arte na área de IHC, com foco em ferramentas para Bioinformática.

Apendice A

APÊNDICE

Esse apêndice apresenta os documentos com a saída output, após a execução da versão adaptada do Atom Validation, no GromaXy, usando todos os campos de força do GROMACS, exceto o CHARMM27.

A.1 Validações usando a proteína 1UAO

Nessa seção apresentadas partes dos *outputs* que foram gerados após as execuções das validações da proteína 1UAO no GromaXy. As próximas subseções estão organizadas por campos de força, sendo assim em cada subseção é apresentado parte do *output* para o campo de força usado.

A.1.1 Validações usando o campo de força Amber03

Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 126)
('ATOM:', '\t', 2052)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('TYR', 2, 'A', 2), ['HB3'])
(('ASP', 5, 'A', 3), ['HB3'])
(('TYR', 17, 'A', 2), ['HB3'])
('\nTime', 0.002924203872680664, 'Microseconds\n\n\n')

Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 126)
('ATOM:', '\t', 2052)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('TYR', 2, 'A', 2), ['HB1'])
(('ASP', 5, 'A', 3), ['HB1'])
(('TYR', 17, 'A', 2), ['HB1'])
('\nTime', 0.08080101013183594, 'Microseconds\n\n\n')

A.1.2 Validações usando o campo de força Amber94

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.0037031173706054688, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.09548211097717285, 'Microseconds\n\n\n')
```

A.1.3 Validações usando o campo de força Amber96

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.00403904914855957, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08771395683288574, 'Microseconds\n\n\n')
```

A.1.4 Validações usando o campo de força Amber99

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.0035839080810546875, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08441400527954102, 'Microseconds\n\n\n')
```

A.1.5 Validações usando o campo de força Amber99sb-ildn

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.0038270950317382812, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08444714546203613, 'Microseconds\n\n\n')
```

A.1.6 Validações usando o campo de força Amber99sb

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)

Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.003576040267944336, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)

Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.0891561508178711, 'Microseconds\n\n\n')
```

A.1.7 Validações usando o campo de força AmberGS

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.0035982131958007812, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08653092384338379, 'Microseconds\n\n\n')
```

A.1.8 Validações usando o campo de força Encads

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)

Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.003607034683227539, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08585810661315918, 'Microseconds\n\n\n')
```

A.1.9 Validações usando o campo de força Encadv

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_unknown
(('ASP', 17, 'A', 3), ['HB3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLY', 5, 'A', 10), ['HA3'])
(('TRP', 7, 'A', 9), ['HB3'])
(('GLY', 9, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
...
('\nTime', 0.0034809112548828125, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_110')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 180)
('ATOM:', '\t', 2484)
Expresion:
Residue model chain residue_number atoms_absence
(('ASP', 17, 'A', 3), ['HB1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLY', 5, 'A', 10), ['HA1'])
(('TRP', 7, 'A', 9), ['HB1'])
(('GLY', 9, 'A', 1), ['H', 'HA1'])
...
('\nTime', 0.08584809303283691, 'Microseconds\n\n\n')
```

A.2 Validações usando a proteína 1A11

Assim como na Seção A.1, nessa seção são apresentadas partes dos *outputs* que foram gerados após a execução das validações, porém usando a proteína 1A11 no GromaXy. As próximas subseções estão organizadas por campos de força, sendo assim em cada subseção é apresentado parte do *output* para o campo de força usado.

A.2.1 Validações usando o campo de força Gmx

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.005981922149658203, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.12082099914550781, 'Microseconds\n\n\n')
```

A.2.2 Validações usando o campo de força Gmx2

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('LYS', 4, 'A', 4), ['HB3', 'HG3', 'HD3', 'HE3', 'HZ3'])
(('LEU', 9, 'A', 19), ['HB3'])
(('GLN', 6, 'A', 24), ['HB3', 'HG3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLN', 9, 'A', 15), ['HB3', 'HG3'])
...
('\nTime', 0.005719184875488281, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('LYS', 4, 'A', 4), ['HB1', 'HG1', 'HD1', 'HE1'])
(('LEU', 9, 'A', 19), ['HB1'])
(('GLN', 6, 'A', 24), ['HB1', 'HG1'])
(('GLY', 4, 'A', 1), ['H', 'HA1'])
(('GLN', 9, 'A', 15), ['HB1', 'HG1'])
...
('\nTime', 0.13664603233337402, 'Microseconds\n\n\n')
```

A.2.3 Validações usando o campo de força Gromos43a1

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.006386995315551758, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.1199331283569336, 'Microseconds\n\n\n')
```

A.2.4 Validações usando o campo de força Gromos43a2

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.006403923034667969, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.12426400184631348, 'Microseconds\n\n\n')
```

A.2.5 Validações usando o campo de força Gromos45a3

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)

Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.006173849105834961, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.11807394027709961, 'Microseconds\n\n\n')
```

A.2.6 Validações usando o campo de força Gromos53a5

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.006186008453369141, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.11929011344909668, 'Microseconds\n\n\n')
```

A.2.7 Validações usando o campo de força Gromos53a6

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.006638050079345703, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.13900113105773926, 'Microseconds\n\n\n')
```

A.2.8 Validações usando o campo de força Gromos54a7

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('ALA', 2, 'A', 8), ['HA', 'HB1', 'HB2', 'HB3'])
(('ALA', 7, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LYS', 4, 'A', 4), ['HA', 'HB2', 'HB3', 'HG2', 'HG3', 'HD2', 'HD3', 'HE2', 'HE3', 'HZ3'])
(('ALA', 5, 'A', 16), ['HA', 'HB1', 'HB2', 'HB3'])
(('LEU', 9, 'A', 19), ['HA', 'HB2', 'HB3', 'HG', 'HD11', 'HD12', 'HD13', 'HD21', 'HD22', 'HD23'])
...
('\nTime', 0.00618290901184082, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ILE', 3, 'A', 9), ['CD'])
(('GLY', 3, 'A', 1), ['H'])
(('ILE', 9, 'A', 9), ['CD'])
(('GLY', 7, 'A', 1), ['H'])
(('ILE', 5, 'A', 9), ['CD'])
...
('\nTime', 0.1254270076751709, 'Microseconds\n\n\n')
```

A.2.9 Validações usando o campo de força Oplsaa

```
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_unknown
(('LYS', 4, 'A', 4), ['HB3', 'HG3', 'HD3', 'HE3', 'HZ3'])
(('LEU', 9, 'A', 19), ['HB3'])
(('GLN', 6, 'A', 24), ['HB3', 'HG3'])
(('GLY', 4, 'A', 1), ['H1', 'H2', 'H3', 'HA3'])
(('GLN', 9, 'A', 15), ['HB3', 'HG3'])
...
('\nTime', 0.005319118499755859, 'Microseconds\n\n\n')
------------------------------------------------------------
Atom Validation 1.0
('PROTEIN:', '/home/guilherme/galaxy-servidor/database/files/000/dataset_109')
('CHAIN:', '\t', ['A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A', 'A'])
('RESIDUE:', 250)
('ATOM:', '\t', 3900)
Expresion:
Residue model chain residue_number atoms_absence
(('ALA', 2, 'A', 8), ['atoms'])
(('ALA', 7, 'A', 16), ['atoms'])
(('LYS', 4, 'A', 4), ['HB1', 'HG1', 'HD1', 'HE1'])
(('ALA', 5, 'A', 16), ['atoms'])
(('LEU', 9, 'A', 19), ['HB1'])
...
('\nTime', 0.13040709495544434, 'Microseconds\n\n\n')
```

GLOSSÁRIO

AMBER – *Assisted Model Building with Energy Refinement*

CHARMM – *Chemistry at Harvard Marcromolecular Mechanics*

CLI – *Command-line User Interface*

DL_POLY – *Daresbury Laboratory Polyatomic Simulator*

DM – *Dinâmica Molecular*

GROMACS – *Groningen Machine for Chemical Simulations*

GROMOS – *Groningen Molecular Simulations*

GUI – *Graphical User Interface*

NAMD – *Not Another Molecular Dynamics*

PDB – *Protein Data Bank*

XML – *Extensible Markup Language*

REFERÊNCIAS

ADCOCK, S. A.; MCCAMMON, J. A. Molecular dynamics: survey of methods for simulating the activity of proteins. *Chemical reviews*, ACS Publications, v. 106, n. 5, p. 1589–1615, 2006.

AFGAN, E.; BAKER, D.; BEEK, M. Van den; BLANKENBERG, D.; BOUVIER, D.; ČECH, M.; CHILTON, J.; CLEMENTS, D.; CORAOR, N.; EBERHARD, C. et al. The galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic acids research*, Oxford University Press, v. 44, n. W1, p. W3–W10, 2016.

APOL, E.; APOSTOLOV, R.; BERENDSEN, H.; BUUREN, A. V.; BJELKMAR, P.; DRUNEN, R. V.; FEENSTRA, A.; GROENHOF, G.; KASSON, P.; LARSSON, P. et al. Gromacs user manual version 4.5. 4. *Royal Institute of Technology and Uppsala University: Stockholm and Uppsala, Sweden*, 2010.

BERMAN, H. M.; WESTBROOK, J.; FENG, Z.; GILLILAND, G.; BHAT, T.; WEISSIG, H.; SHINDYALOV, I. N.; BOURNE, P. E. The protein data bank. *Nucleic acids research*, Oxford Univ Press, v. 28, n. 1, p. 235–242, 2000.

Bioeclipse Wiki. *Gromacs plugin*. 2014. Access date: 30 mar. 2014. Disponível em: <http://wiki.bioclipse.net/index.php?title=Gromacs_plugin>.

BLANKENBERG, D.; KUSTER, G. V.; BOUVIER, E.; BAKER, D.; AFGAN, E.; STOLER, N.; TAYLOR, J.; NEKRUTENKO, A. Dissemination of scientific software with galaxy toolshed. *Genome Biology*, v. 15, n. 2, p. 403, Feb 2014. ISSN 1474-760X. Disponível em: <http://dx.doi.org/10.1186/gb4161>.

BROOKS, B. R.; BRUCCOLERI, R. E.; OLAFSON, B. D.; STATES, D. J.; SWAMINATHAN, S.; KARPLUS, M. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of computational chemistry*, Wiley Online Library, v. 4, n. 2, p. 187–217, 1983.

DELANO, W. L. The pymol molecular graphics system. 2002.

FRENKEL, D.; SMIT, B. *Understanding molecular simulation: from algorithms to applications*. [S.l.]: Academic press, 2001.

GalaxyToolTutorial. Jun 2017. Disponível em: <https://galaxyproject.org/admin/tools/add-tool-tutorial/>.

GOECKS, J.; NEKRUTENKO, A.; TAYLOR, J. et al. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*, v. 11, n. 8, p. R86, 2010.

GRUDIN, J. The computer reaches out: the historical continuity of interface design. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 1990. p. 261–268.

HONDA, S.; YAMASAKI, K.; SAWADA, Y.; MORII, H. 10 residue folded peptide designed by segment statistics. *Structure*, Elsevier, v. 12, n. 8, p. 1507–1518, 2004.

KNAPP, B.; SCHREINER, W. Graphical user interfaces for molecular dynamics—quo vadis? *Bioinformatics and biology insights*, Libertas Academica, v. 3, p. 103, 2009.

KOTA, P. Guimacs–a java based front end for gromacs. *In silico biology*, IOS Press, v. 7, n. 1, p. 95–99, 2007.

LEITE, G. J. S. Validação de proteínas no campo de força charmm27. *Trabalho de conclusão de curso - Curso de Ciência da Computação. Centro Universitário Barão de Mauá. Ribeirão Preto, SP*, 2012.

MAKAREWICZ, T.; KAZMIERKIEWICZ, R. Molecular dynamics simulation by gromacs using gui plugin for pymol. *Journal of chemical information and modeling*, ACS Publications, v. 53, n. 5, p. 1229–1234, 2013.

MORGON, N. H.; COUTINHO, K. *Métodos de química teórica e modelagem molecular*. [S.l.]: Morgon & Coutinho, 2007.

NAMBA, A.; SILVA, V. D.; SILVA, C. D. Dinâmica molecular: teoria e aplicações em planejamento de fármacos. *Eclética Química*, SciELO Brasil, v. 33, n. 4, p. 13–23, 2008.

NELSON, D.; COX, M. *Lehninger : principios de bioquímica*. [S.l.]: Ediciones Omega, S.L., 2014. ISBN 9788428216036.

NIELSEN, J. *Usability engineering*. [S.l.]: Elsevier, 1994.

OPELLA, S.; MARASSI, F.; GESELL, J.; VALENTE, A.; KIM, Y.; OBLATT-MONTAL, M.; MONTAL, M. Structures of the m2 channel-lining segments from nicotinic acetylcholine and nmda receptors by nmr spectroscopy. *Nature Structural & Molecular Biology*, Nature Publishing Group, v. 6, n. 4, p. 374–379, 1999.

PHILLIPS, J. C.; BRAUN, R.; WANG, W.; GUMBART, J.; TAJKHORSHID, E.; VILLA, E.; CHIPOT, C.; SKEEL, R. D.; KALE, L.; SCHULTEN, K. Scalable molecular dynamics with namd. *Journal of computational chemistry*, Wiley Online Library, v. 26, n. 16, p. 1781–1802, 2005.

PONDER, J. W. et al. Tinker: Software tools for molecular design. *Washington University School of Medicine, Saint Louis, MO*, v. 3, 2004.

RAPAPORT, D. C. *The Art of Molecular Dynamics Simulation*. 2nd. ed. New York, NY, USA: Cambridge University Press, 2004. ISBN 0521825687, 9780521825689.

Rescale. *Rescale*. 2014. Access date: 31 mar. 2014. Disponível em: <http://www.rescale.com/>.

Reza Salari. *Gromacs GUI*. 2017. Access date: 4 jan. 2017. Disponível em: <https://resal.wordpress.com>.

ROCHA, H. V. D.; BARANAUSKAS, M. C. C. *Design e avaliação de interfaces humano-computador*. [S.l.]: Unicamp, 2003.

ROOPRA, S.; KNAPP, B.; OMASITS, U.; SCHREINER, W. jsimmacs for gromacs: a java application for advanced molecular dynamics simulations with remote access capability. *Journal of chemical information and modeling*, ACS Publications, v. 49, n. 10, p. 2412–2417, 2009.

SALOMON-FERRER, R.; CASE, D. A.; WALKER, R. C. An overview of the amber biomolecular simulation package. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, Wiley Online Library, v. 3, n. 2, p. 198–210, 2013.

SCOTT, W. R.; HÜNENBERGER, P. H.; TIRONI, I. G.; MARK, A. E.; BILLETER, S. R.; FENNEN, J.; TORDA, A. E.; HUBER, T.; KRÜGER, P.; GUNSTEREN, W. F. van. The gromos biomolecular simulation program package. *The Journal of Physical Chemistry A*, ACS Publications, v. 103, n. 19, p. 3596–3607, 1999.

SELLIS, D.; VLACHAKIS, D.; VLASSI, M. Gromita: a fully integrated graphical user interface to gromacs 4. *Bioinformatics & Biology Insights*, n. 3, 2009.

SMITH, W.; YONG, C.; RODGER, P. Dl_poly: Application to molecular simulation. *Molecular Simulation*, Taylor & Francis, v. 28, n. 5, p. 385–471, 2002.

SPJUTH, O.; HELMUS, T.; WILLIGHAGEN, E. L.; KUHN, S.; EKLUND, M.; WAGENER, J.; MURRAY-RUST, P.; STEINBECK, C.; WIKBERG, J. E. Bioclipse: an open source workbench for chemo-and bioinformatics. *BMC bioinformatics*, v. 8, n. 1, p. 59, 2007.

SPOEL, D. V. D.; LINDAHL, E.; HESS, B.; GROENHOF, G.; MARK, A. E.; BERENDSEN, H. J. Gromacs: fast, flexible, and free. *Journal of computational chemistry*, Wiley Online Library, v. 26, n. 16, p. 1701–1718, 2005.

VOET, D.; VOET, J. *Biochemistry*. John Wiley & Sons, 2011. ISBN 9781118139929. Disponível em: <http://books.google.com.br/books?id=-GwUPP2C8ysC>.

YU, M. Computational modeling of protein dynamics with gromacs and java. 2012.

ZELLER, M.; PHILLIPS, J. C.; DALKE, A.; HUMPHREY, W.; SCHULTEN, K.; SHARMA, R.; HUANG, T.; PAVLOVIC, V.; ZHAO, Y.; LO, Z. et al. A visual computing environment for very large scale biomolecular modeling. In: IEEE. *Application-Specific Systems, Architectures and Processors, 1997. Proceedings., IEEE International Conference on*. [S.l.], 1997. p. 3–12.