**UNIVERSIDADE FEDERAL DE PERNAMBUCO**
**Programa de Pós-Graduação em Inovação Terapêutica**

**KLAUS RIBEIRO CAVALCANTE**

# DESENVOLVIMENTO DE UMA PLATAFORMA COMPUTACIONAL INTEGRADA PARA A REALIZAÇÃO DE PROCEDIMENTOS *IN SILICO* NA INOVAÇÃO TERAPÊUTICA: MODiMOL WORKBENCH

Recife
2016

**KLAUS RIBEIRO CAVALCANTE**

# DESENVOLVIMENTO DE UMA PLATAFORMA COMPUTACIONAL INTEGRADA PARA A REALIZAÇÃO DE PROCEDIMENTOS *IN SILICO* NA INOVAÇÃO TERAPÊUTICA: MODiMOL WORKBENCH

Tese apresentada ao Programa de Pós-Graduação em Inovação Terapêutica da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do Título de Doutor em Inovação Terapêutica

Orientador: Prof. Dr. Marcelo Zaldini Hernandes

Recife
2016

Catalogação na fonte
Elaine Barroso
CRB 1728

**Cavalcante, Klaus Ribeiro**
**Desenvolvimento de uma plataforma computacional integrada para a realização de procedimentos *in silico* na inovação terapêutica: MODiMOL *Workbench* / Klaus Ribeiro Cavalcante- Recife: O Autor, 2016.**

**155 folhas: il., fig., tab.**
**Orientador: Marcelo Zaldini Hernandes**
**Tese (doutorado) – Universidade Federal de Pernambuco. Centro de Biociências. Inovação Terapêutica, 2016.**
**Inclui referências e apêndices**

**1. Bioinformática 2. Banco de dados 3.Moléculas-modelos I. Hernandes, Marcelo Zaldini (orientador) II. Título**

**660.6 CDD (22.ed.) UFPE/CCB-2017-164**

**KLAUS RIBEIRO CAVALCANTE**

# DESENVOLVIMENTO DE UMA PLATAFORMA COMPUTACIONAL INTEGRADA PARA A REALIZAÇÃO DE PROCEDIMENTOS *IN SILICO* NA INOVAÇÃO TERAPÊUTICA: MODiMOL WORKBENCH

Tese apresentada ao Programa de Pós-Graduação em Inovação Terapêutica, da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutor em Inovação Terapêutica.

Aprovada em: 04 / 03 / 2016

**COMISSÃO EXAMINADORA**

_____

Prof. Dr. Marcelo Zaldini Hernandes / Universidade Federal de Pernambuco

_____

Prof. Dr. Antônio Mauro Rezende / Centro de Pesquisas Aggeu Magalhães

_____

Prof. Dr. Gustavo de Miranda Seabra / Universidade Federal de Pernambuco

_____

Prof. Dr. Ramiro Brito Willmersdorf / Universidade Federal de Pernambuco

_____

Prof. Dr. Carlos Henrique Madeiros Castelletti / Instituto Agronômico de Pernambuco

Dedico esta tese a minha família

## AGRADECIMENTOS

Agradeço a minha família pelo apoio e paciência concedida em todos os momentos que estive ausente para me dedicar à conclusão deste trabalho.

Ao professor e amigo Marcelo Zaldini Hernandes pelo aprendizado, orientação e maturidade adquirida ao longo destes anos de parceria. No LQTM, aprendi o significado da expressão "trabalho sério, responsável e de qualidade".

À professora Suely Lins Galdino (*in memoriam*) que sempre demonstrou grande interesse pelo meu trabalho, através de sugestões e apoio em todos os momentos presentes.

Aos colegas de LQTM, Marcelo Montenegro e Felipe Rebelo, pela amizade e experiências compartilhadas nestes anos de caminhada.

Aos colegas de PPGIT, que me apoiaram e dividiram conquistas ao longo deste período na UFPE.

À equipe do CENAPAD-PE pelo apoio ao projeto, através da disponibilização de uma máquina virtual para hospedar o servidor do sistema e da criação de um usuário especial criado para testes ("*workbench*").

Em especial, ao nosso Senhor Jesus Cristo, por sua infinita Graça e Misericórdia.

1 Ainda que eu falasse as línguas dos homens e dos anjos, e não tivesse amor, seria como o metal que soa ou como o sino que tine.
2 E ainda que tivesse o dom de profecia, e conhecesse todos os mistérios e toda a ciência, e ainda que tivesse toda a fé, de maneira tal que transportasse os montes, e não tivesse amor, nada seria.
3 E ainda que distribuísse toda a minha fortuna para sustento dos pobres, e ainda que entregasse o meu corpo para ser queimado, e não tivesse amor, nada disso me aproveitaria.
4 O amor é sofredor, é benigno; o amor não é invejoso; o amor não trata com leviandade, não se ensoberbece.
5 Não se porta com indecência, não busca os seus interesses, não se irrita, não suspeita mal;
6 Não folga com a injustiça, mas folga com a verdade;
7 Tudo sofre, tudo crê, tudo espera, tudo suporta.
8 O amor nunca falha; mas havendo profecias, serão aniquiladas; havendo línguas, cessarão; havendo ciência, desaparecerá;
9 Porque, em parte, conhecemos, e em parte profetizamos;
10 Mas, quando vier o que é perfeito, então o que o é em parte será aniquilado.
11 Quando eu era menino, falava como menino, sentia como menino, discorria como menino, mas, logo que cheguei a ser homem, acabei com as coisas de menino.
12 Porque agora vemos por espelho em enigma, mas então veremos face a face; agora conheço em parte, mas então conhecerei como também sou conhecido.
13 Agora, pois, permanecem a fé, a esperança e o amor, estes três, mas o maior destes é o amor.

1 CORÍNTIOS 13:1-13

## RESUMO

Metodologias computacionais vêm sendo cada vez mais empregadas com o objetivo de estudar sistemas moleculares, no que diz respeito à sua representação estrutural e explicação do seu comportamento, além da predição de suas propriedades físico-químicas. Desta forma, métodos "*in silico*" vêm se tornando mais utilizados, sobretudo, no âmbito da inovação terapêutica, devido principalmente a grande disponibilidade de programas gratuitos e equipamentos dedicados, com capacidades computacionais cada vez maiores, voltados para aplicações em modelagem molecular. Vale ressaltar que o uso de metodologias computacionais pode reduzir os custos envolvidos na pesquisa e desenvolvimento de novos fármacos numa indústria farmacêutica em até 50%. Neste contexto, portais científicos fornecem meios para gerenciar dados e executar aplicações científicas em ambientes de computação de alto desempenho (*clusters*, *grids* ou nuvens públicas/privadas), através de interfaces amigáveis baseadas na Web. Este presente trabalho apresenta o desenvolvimento de um portal científico, denominado MODiMOL *Workbench*, cujo objetivo é fornecer acesso a diferentes metodologias e abordagens de modelagem molecular, em ambientes de computação de alto desempenho, assim como suporte para dispositivos móveis (*smartphones* e *tablets*), através de interfaces de usuário responsivas, projetadas especialmente para estes tipos de dispositivos, por demonstrarem possuir um enorme potencial como ferramentas de pesquisa a favor da inovação terapêutica.



**Palavras-chave**: Portal científico. Computação de alto desempenho. MODiMOL *Workbench.* Modelagem molecular. Dispositivos móveis.

## ABSTRACT

Computational methodologies have been increasingly employed in order to study molecular systems, their structural representation and elucidation of the behavior, besides prediction of their physicochemical properties. Thus, *in silico* methods are becoming widely used, especially in the context of therapeutic innovation, mainly due to the huge availability of free softwares and hardware with increasing computational capabilities, focused on applications in molecular modeling. It is worth mentioning that "in silico" methodologies may reduce up to 50% the time spent and costs involved with research and development (R&D) of new drugs in a pharmaceutical industry. In this context, science gateways provide ways to manage data and execute scientific applications in high-performance computing environments (clusters, grids, or private/public clouds) through web-based friendly interfaces. The current work presents the development of a science gateway, named Modimol Workbench, whose goal is to provide access to different methodologies and molecular modeling approaches, in high-performance computing environments, as well as support for mobile devices (smartphones and tablets) through Web responsive user interfaces designed especially for these devices, by demonstrating enormous potential as new research tools for molecular modeling, particularly in therapeutic innovation.



**Keywords**: Science gateway. High-performance computing. Software. MODiMOL Workbench. Molecular modeling. Mobile devices.

# LISTA DE ILUSTRAÇÕES

# LISTA DE TABELAS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | *Application Programming Interface* |
| CADD | *Computer-Aided Drug Design* |
| CAMD | *Computer-Aided Molecular Design* |
| CDK | *Chemistry Development Kit* |
| CPU | *Central Processing Unit* |
| CSV | *Comma Separated Values* |
| GPU | *Graphics Processing Unit* |
| GUI | *Graphical User Interface* |
| HPC | *High Performance Computing* |
| INCHI | *IUPAC Chemical Identifier* |
| JAX-RS | API Java para Web Services do tipo RESTful |
| JAX-WS | API Java para Web Services baseado em XML |
| LDAP | *Lightweight Directory Access Protocol* |
| LQTM | Laboratório de Química Teórica Medicinal |
| MEP | *Molecular Electrostatic Potential* |
| MOL | *Symyx Molecule File Format* |
| MOL2 | *Tripos Sybyl MOL2 Format* |
| PC | *Personal Computer* |
| PDA | *Personal Digital Assistant* |
| PDB | *Protein Data Bank* |
| QSAR | *Quantitative Structure–Activity Relationship* |
| SASA | *Solvent Accessible Surface Area* |
| SCP | *Secure Copy* |
| SDF | *Symyx Structure Data Format* |
| SGBD | Sistema Gerenciador de Bancos de Dados |
| SMILES | *Simplified Molecular-Input Line-Entry System* |
| SSH | *Secure Shell* |
| UFF | *Universal Force Field* |
| WFMS | *Workflow Management Systems* |
| XML | *Extensible Markup Language* |

## SUMÁRIO

# 1 INTRODUÇÃO

A pesquisa científica vem se tornando cada vez mais multidisciplinar. Em especial, observa-se que a inserção de novas metodologias computacionais dentro dos diversos ramos da ciência e da tecnologia têm contribuído positivamente neste contexto. A aplicação destas metodologias vem sendo impulsionada principalmente pela crescente quantidade de dados experimentais que demandam análises em larga escala, e pela necessidade de simulações de sistemas e modelos teóricos cada vez mais realísticos e complexos.

A modelagem molecular foi uma das áreas que recebeu inestimável atenção e contribuição nos últimos anos, no sentido de serem desenvolvidas e disponibilizadas diversas ferramentas computacionais capazes de representar e simular o comportamento e propriedades químicas de estruturas, que vão desde partículas subatômicas até sistemas macromoleculares, como é o caso de proteínas e ácidos nucléicos. Esta representação "virtual", que utiliza o computador como plataforma de estudos, é conhecida comumente pelo termo "*in silico*", proveniente da alusão feita aos *chips* de silício encontrados na maioria dos computadores (RODRIGUES, 2001).

Em particular, no âmbito da inovação terapêutica, o planejamento de fármacos também vem recebendo ampla atenção no que diz respeito à disponibilidade de programas (*softwares*) gratuitos e com código-aberto (*open-source*), e equipamentos (*hardware*) com capacidades computacionais cada vez maiores, dedicados para aplicações em modelagem molecular de novos compostos com potencial atividade biológica ou farmacológica (CADD – *Computer-Aided Drug Design*) (HUANG *et al*., 2010; DALKAS *et al*., 2012; MACALINO *et al*., 2015).

Neste contexto, vale ressaltar que o uso de metodologias computacionais pode reduzir os custos envolvidos na pesquisa e desenvolvimento de novos fármacos numa indústria farmacêutica em até 50% (GELDENHUYS *et al*., 2006; MCGEE, 2005). Esta economia é normalmente explicada pelo alto nível de precisão e preditividade dos métodos computacionais (*in silico*) utilizados nestes processos. A aplicação destes métodos, muitas vezes, reduz drasticamente a quantidade de moléculas candidatas a novos fármacos que precisam ser efetivamente sintetizadas

e testadas experimentalmente, encurtando o tempo total necessário para a produção e disponibilização de um novo fármaco no mercado.

A maior dificuldade encontrada na aplicação de metodologias "*in silico*" está nos requisitos de hardware para a realização destes processos (por exemplo, em procedimentos de *virtual screening* de bibliotecas de moléculas ou em cálculos de química quântica em nível *ab initio*). Estes métodos são, normalmente, muito custosos computacionalmente (*CPU-intensive*), e portanto, seu uso, em certos casos, é limitado à existência de uma infraestrutura minimamente compatível com a demanda computacional requerida. Desta forma, novos esforços têm sido direcionados para o uso de tecnologias de computação de alto desempenho (*High Performance Computing* – HPC) para fornecer ambientes viáveis para a execução de aplicações científicas que requerem alta demanda computacional (OSG, 2014).

Estas iniciativas, normalmente associadas ao termo "*e-Science*", visam o uso de máquinas de grande poder de processamento em larga escala (através da implantação de poderosas infraestruturas de altíssimo desempenho), dentro do processo de descoberta e planejamento de novos fármacos. Estes ambientes dedicados são criados, geralmente, na forma de *clusters*, *grids* computacionais ou nuvens (*clouds*) públicas/privadas, e vêm se consolidando como elementos indispensáveis e fundamentais tanto na indústria como nas instituições acadêmicas (D-GRID, 2014; EGI, 2014).

Para estes propósitos, portais científicos (*science gateways* ou *science portals*) estão sendo cada vez mais empregados como uma ponte entre pesquisadores e aplicações científicas executadas (como "*jobs*") em ambientes de computação de alto desempenho (HPC). Um portal científico, neste caso, é um elemento mediador que disponibiliza um conjunto de aplicações científicas que são accessíveis remotamente, através de uma interface gráfica amigável e intuitiva, e cujo objetivo principal consiste em evitar, para os usuários finais interessados, a alta complexidade em:

1. Instalação das aplicações científicas, cujas especificidades podem variar bastante de aplicação para aplicação. Alguns programas, inclusive, possuem requisitos mínimos de recursos computacionais (*hardware*) e sistemas operacionais específicos para serem instalados e funcionarem apropriadamente;

2. Submissão e acompanhamento da execução dos "*jobs*" em ambientes dedicados de HPC. A maioria destes sistemas computacionais possuem escalonadores de recursos diretamente, cujas operações são realizadas em modo de linha de comando (por exemplo, no *prompt* de comando do Windows® ou no terminal do Linux/Unix), trazendo, desta forma, certa dificuldade ou até mesmo impedindo seu uso por parte daqueles usuários que possuem menos experiência em informática;
3. Manutenção da infraestrutura do sistema (*hardware*, sistema operacional, banco de dados) funcionando corretamente. Estes programas deixam de serem executados no computador pessoal (PC *desktop* ou *laptop*) do usuário, para serem executados remotamente em um ambiente computacional externo mantido por terceiros. Esta é uma vantagem principalmente para usuários que desejam utilizar estas ferramentas, mas que possuem pouca disponibilidade de recursos financeiros para a aquisição de máquinas com maior poder de processamento.

O principal aspecto positivo desta estratégia está no fato de que o usuário (pesquisador) pode interagir com uma determinada aplicação (ou algoritmo, no jargão conhecido nesta área), sem saber exatamente em que recurso(s) do ambiente computacional a aplicação será executada. Outro fator importante que deve ser considerado neste contexto consiste no fato de que o usuário não precisa interagir diretamente com a execução da aplicação através da digitação da linha de comando da aplicação (contendo parâmetros de entrada, variáveis de ambiente, etc.). Isto consequentemente acarreta na diminuição da curva de aprendizado em relação ao algoritmo utilizado, uma vez que ele apenas precisa preencher campos de um formulário (*desktop* ou Web), ao invés de escrever explicitamente um comando ou um *shell script* completo referente ao programa. Exemplos de portais científicos serão discutidos de maneira mais detalhada no capítulo 2.

Este presente trabalho de doutorado tem como principal objetivo o desenvolvimento de um sistema de portal científico (*science gateway*), denominado Modimol Workbench, cujo principal objetivo é prover um ambiente em que aplicações científicas (algoritmos) são disponibilizadas para que os usuários possam submeter e monitorar seus *jobs*, e onde todos os resultados obtidos no sistema são organizados para permitir, facilmente, a visualização de todos os dados gerados

nestes procedimentos, assim como, o rastreamento (histórico) das execuções (entradas, saídas, parâmetros utilizados, e outras informações básicas) realizadas por cada usuário.

Um dos principais diferenciais deste sistema, em comparação com outros exemplos de portais científicos, consiste na presença de interfaces de usuário responsivas, ou seja, adequadas para as telas dos dispositivos móveis (*smartphones* e *tablets,* por exemplo). É importante reforçar que, atualmente, programas (*apps*) para estes dispositivos estão cada vez mais disponíveis, incluindo aqueles com foco em aplicações científicas.

# 2 OBJETIVOS

## *2.1 Objetivo Geral*

- Desenvolvimento de uma plataforma computacional integrada para a realização de procedimentos *in silico* na inovação terapêutica: MODiMOL *Workbench*.

## *2.2 Objetivos Específicos*

- Desenvolver uma versão do sistema MODiMOL *Workbench* que seja capaz de criar, submeter, monitorar e exibir os resultados obtidos a partir da execução de cálculos de modelagem molecular ("*jobs*"), particularmente os relacionados com a inovação terapêutica;
- Desenvolver uma interface gráfica de usuário (GUI) amigável e intuitiva, com o objetivo de permitir o acesso ao sistema por meio da Web, ou seja, através de um navegador (*browser*), independente do sistema operacional;
- Permitir a integração do sistema com servidores externos que possuam infraestrutura de computação de alto desempenho (HPC), através de conexão segura, para fornecer meios de submeter cálculos ("*jobs*") remotos;
- Gerenciar todos os repositórios de dados do sistema (usuários, arquivos, algoritmos/serviços), através de um gerenciador de banco de dados e/ou sistema de arquivos;
- Possibilitar a manipulação dos dados (e arquivos) de usuários, contemplando, por exemplo, operações para visualizar e editar conteúdo, remover, renomear e duplicar arquivos.
- Possibilitar a conversão dos arquivos dos usuários entre vários formatos existentes, através de uma ferramenta de conversão, como por exemplo, o programa OpenBabel (O'BOYLE *et al.*, 2011);

- Permitir a visualização molecular tridimensional em arquivos que contenham, como conteúdo, a estrutura de um sistema molecular em formatos específicos, através de um programa apropriado para a visualização de estruturas químicas em 3D, tal como o *software* JSmol (JSMOL, 2014);
- Permitir o compartilhamento de informações de "*jobs*" entre os usuários do sistema, através da exploração do conceito de times ou grupos integrados de pesquisa;
- Realizar a integração com servidores externos de banco de dados públicos para estruturas moleculares, permitindo a obtenção direta destas estruturas químicas, a partir destes serviços;
- Gerenciar o repositório de algoritmos (*softwares*) disponíveis no sistema de forma dinâmica, ou seja, sem necessidade de recompilação do projeto ("*rebuild*"), quando uma mudança é realizada (inclusão, atualização ou remoção de um algoritmo);
- Criar perfis de usuários com permissões específicas para acesso aos serviços e restrições para submissões de "*jobs*"; e,
- Desenvolver uma versão da interface gráfica do sistema para seu acesso a partir de dispositivos móveis (*smartphones* e *tablets*, por exemplo).

# 3 REVISÃO DA LITERATURA

## *3.1 Portais Científicos*

Um portal científico (*science portal* ou *science gateway*) é um conjunto de ferramentas, aplicações e tecnologias associadas, cujo objetivo é permitir o acesso a infraestruturas de computação distribuída (*Distributed Computing Infrastructures* - DCI) sem a necessidade de que os usuários conheçam os mecanismos de funcionamento destes ambientes (KACSUK, 2014). Um portal científico é normalmente relacionado à ideia de uma camada de acesso (*front-end*) entre o usuário e a infraestrutura computacional, capaz de abstrair a complexidade inerente em se trabalhar diretamente com as tecnologias pertencentes aos ambientes de computação no qual se referem.

Neste sentido, portais científicos normalmente são desenvolvidos para acessar estruturas de computação de alto desempenho (*High-Performance Computing* - HPC) típicas, tais como, supercomputadores, *clusters*, *grids* computacionais e *clouds* (nuvens), servidores contendo coleções maciças de dados (*massive datacenters*), sensores, instrumentos, sistemas de *workflows*, *softwares*, ou quaisquer outros elementos nos quais o acesso de forma direta é inviável ou complexo (do ponto de vista do usuário), ou precisa ser controlado (autenticação e autorização exigidos) (SG, 2015).

Quando portais científicos são criados para acessar ambientes de HPC, sua funcionalidade primária consiste no gerenciamento relativo à submissão, monitoramento e visualização dos *jobs* dos usuários. Em complemento a esta função básica, a maioria dos portais também inclui formas de gerenciamento para algoritmos (programas científicos que são executados na infraestrutura computacional), armazenamento e acesso dos arquivos de entrada/saída dos *jobs*, gerenciamento de repositórios de arquivos e usuários, controle de acesso, compartilhamento de informações, dentre outras. Todos estes recursos tornam estas ferramentas extremamente versáteis e capazes de suprir as necessidades básicas de usuários interessados (pesquisadores) que necessitam utilizar determinadas infraestruturas de HPC, mas que possuem pouca ou nenhuma habilidade e experiência nestes ambientes.

### *3.1.1 Tecnologias para portais científicos*

Portais científicos podem ser desenvolvidos de forma livre (*ad hoc*) ou através de padrões e arcabouços de desenvolvimento apropriados. A seguir, serão apresentadas algumas tecnologias usadas no desenvolvimento de portais científicos.

#### *3.1.1.1 Globus Toolkit*

O projeto Globus (FOSTER & KESSELMAN, 1997) foi originalmente criado com o objetivo de desenvolver um *middleware* aberto (*open source*) para a criação de *grids* computacionais baseados em arcabouços e tecnologias padronizadas, em especial, através do modelo OGSA (*Open Grid Services Architecture*) (FOSTER *et al.*, 2002). Este *middleware* (conhecido como Globus *toolkit* - GT) (http://toolkit.globus.org/) consiste num conjunto de ferramentas e componentes de software que facilitam o desenvolvimento de um ambiente de computação distribuída, no que diz respeito ao gerenciamento e alocação apropriada dos recursos computacionais existentes, armazenamento e acesso eficiente a grandes quantidades de dados, e comunicação entre os elementos integrantes dentro da rede de computadores.

Um sistema desenvolvido com esta plataforma permite o compartilhamento de diversos tipos de recursos: ciclos de processamento, sistemas de armazenamento de dados, instrumentos científicos entre diversas instituições geograficamente separadas, levando em consideração as políticas específicas de cada instituição para o controle dos recursos disponibilizados (BUYYA & VENUGOPAL, 2005).

Além de ser usado para a criação de ambientes distribuídos para compartilhamento de recursos, o Globus *toolkit* também vem sendo usado para a criação de portais científicos, como é o caso do projeto XSEDE (TOWNS *et al.*, 2014). Nesta iniciativa, os recursos computacionais de diversas instituições de pesquisa norte-americanas são compartilhados para se obter um ambiente unificado de altíssimo desempenho. Através do projeto XSEDE (https://www.xsede.org/), é possível utilizar estes recursos através de um cadastramento no ambiente ou mesmo desenvolver portais científicos externos que se integrem diretamente à

infraestrutura provida pelo XSEDE de forma automatizada, através de interfaces de programação (API - *Application Programming Interface*). Neste sentido, o projeto XSEDE tende a se comportar, ao mesmo tempo, como um portal científico (desenvolvido com o Globus *toolkit*), e como provedor de recursos (ciclos de processamento pertencentes às máquinas compartilhadas no projeto) através da disponibilização de serviços de integração para portais externos que desejam acessar a infraestrutura do XSEDE.

#### *3.1.1.2 gUSE*

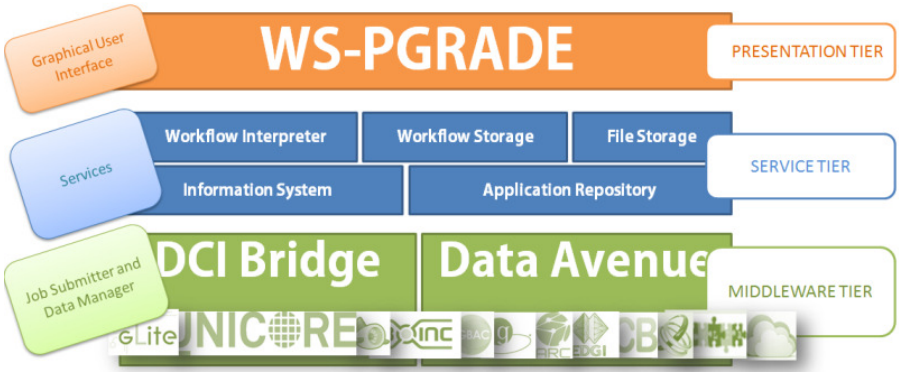
O projeto gUSE (*Grid and Cloud User Support Environment*) (BALASKO *et al*., 2013) consiste em um arcabouço aberto (*open source*), para desenvolvimento de portais científicos de propósito genérico, desenvolvido pelo Instituto de Pesquisas em Computação e Automação da Hungria (http://guse.hu/). Este projeto tem como foco principal, o suporte para execução e gerenciamento de *workflows* sobre diversos tipos de ambientes de computação distribuída: *clusters* (PBS e LSF), grids computacionais (ARC, gLite, GT2, GT4, XSEDE, entre outros), supercomputadores (UNICORE), *grids* de computadores *desktop* (BOINC), e nuvens (CloudBroker, GAE).

O sistema gUSE pode ser integrado com diferentes sistemas gerenciamento de *workflows* científicos, tais como Taverna ou Kepler, permitindo que usuários organizem suas aplicações específicas em estruturas de execução mais compreensivas. Estes exemplos de sistemas de gerenciamento de *workflows* serão discutidos no texto mais adiante.

A Figura 3.1 apresenta a arquitetura em camadas do arcabouço gUSE. A camada inferior deste modelo (*Middleware Tier*) é responsável por operações de manipulação de arquivos (*download*, *upload* e transferência de dados de/para serviços de armazenamento) (*Data Avenue*). Esta camada também trata da comunicação do sistema com os diferentes tipos de *middleware* das infraestruturas de computação suportadas nesta solução (*DCI Bridge*). A camada interna (*Service Tier*) é composta por diversos serviços de alto nível classificados como: i) serviços de dados (repositórios de usuários, arquivos, credenciais, aplicações, e *workflows*);

e, ii) serviços de controle (submissão de *jobs*, alocação de recursos, execução de *workflows*, etc.).

Figura 3.1. Arquitetura do projeto gUSE.




Fonte: http://guse.hu/about/architecture, 2016.

A camada superior (*Presentation Tier*) consiste na interface gráfica do usuário no arcabouço gUSE. Esta camada é formada por um portal Web, denominado WS-PGRADE (*Web Service – Parallel Grid Run-time and Application Development Environment*), que permite o usuário editar/salvar seus *workflows* através de um editor gráfico intuitivo. Esta camada também é responsável pela autenticação (certificado digital) e autorização dos usuários ao acessar os serviços e recursos computacionais do sistema.

Existem atualmente diversos portais científicos desenvolvidos com o arcabouço gUSE, dentre eles: AutoDock Portal (https://autodock-portal.sztaki.hu/), um portal científico para realizar docking molecular com o software AutoDock (MORRIS *et al.*, 2009); MoSGRID Portal (https://mosgrid.de/) cujo objetivo é fornecer acesso a serviços de química quântica, docking e dinâmica molecular; e, VisIVO Gateway (http://visivo.oact.inaf.it:8080/) que possibilita a utilização de ferramentas para a criação de imagens customizadas na área de astrofísica.

#### *3.1.1.3 PortEngin/PortEditor/CSGrid*

As ferramentas PortEngin e PortEditor (GOMES *et al.*, 2015) fazem parte da solução desenvolvida para a criação dos portais científicos da Rede Nacional de Processamento de Alto Desempenho (SINAPAD). Cada portal científico consiste em

um conjunto de arquivos de configuração criados para um determinado algoritmo (aplicação), por meio de editor de texto apropriado para arquivos no formato XML (PortEditor). Estes arquivos XML correspondem à interface gráfica exibida para o portal, contendo os campos de entrada necessários para a correta execução do algoritmo. Quando um portal é desenvolvido, estes arquivos de configuração são incluídos na ferramenta PortEngin, que disponibiliza os portais científicos através da Web, uma vez que este componente é executado como uma aplicação Web em um servidor de aplicações Web. Alguns exemplos de portais científicos disponibilizados pelo SINAPAD serão apresentados com maiores detalhes na seção 3.1.3.4.

A infraestrutura de computação distribuída do SINAPAD é acessada pelos portais científicos através do *middleware* CSGrid, uma instância do arcabouço CSBase (LIMA *et al*., 2006), criado pela Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO) com o objetivo de desenvolver *grids* computacionais customizados (isto é, com funcionalidades modificadas e/ou adicionadas). Este *middleware* é responsável por diversas funcionalidades básicas necessárias para o funcionamento do sistema, como por exemplo, gerenciamento do repositório de algoritmos, gerenciamento do diretório de usuários (através de um servidor LDAP), acesso e transferência de arquivos (através dos utilitários SSH e SCP) e submissão de jobs através de escalonadores de recursos.

Além das funcionalidades já providas pelo arcabouço CSBase, foram adicionados ao *middleware* CSGrid novos recursos com o objetivo de torná-lo mais adequado às necessidades do ambiente do SINAPAD. Alguns exemplos destes recursos: i) compartilhamento de arquivos entre usuários; ii) Acesso restrito anônimo; iii) submissão de *workflows*; e, iv) rastreamento de dados provenientes de submissões de jobs;

### *3.1.2 Sistemas de workflow científico*

Atualmente diversos portais científicos disponibilizam infraestrutura e meios para se trabalhar com *workflows* científicos. Esta necessidade surge, em especial, em diversas abordagens *in silico* que possuem protocolos bem definidos e conhecidos em áreas da biotecnologia (bioinformática, modelagem molecular, desenvolvimento de fármacos, etc.). Esta seção tem como principal objetivo discutir

acerca de sistemas de *workflows*, conceitos e tecnologias usadas para seu desenvolvimento.

Um *workflow* (fluxo de trabalho) consiste numa sequência ordenada e finita de etapas, em que cada etapa é executada uma após a outra até o final da sequência. Geralmente, cada etapa pode ser vista como uma ação transformadora individual sobre um elemento (ou conjunto de elementos) que normalmente representam objetos do mundo real, tais como documentos ou itens em produção (SULLIVAN & SHEFFRIN, 2003).

Historicamente, *workflows* começaram a ser estudados no âmbito da manufatura de produtos, a partir da segunda metade do século XIX, porém suas aplicações logo se espalharam por outras áreas do conhecimento, como por exemplo, na automação e modelagem de processos, gerenciamento de projetos, como também em diversos domínios da ciência. Atualmente, *workflows* científicos estão sendo largamente empregados em biologia computacional, bioinformática, genômica e descoberta de fármacos (*drug discovery*) (SHON *et al*., 2008), uma vez que vários estudos nestas áreas necessitam da execução sequencial de várias ferramentas sobre os dados de pesquisa.

Estes tipos de *workflow* estão sendo cada vez mais organizados dentro de sistemas de gerenciamento (WfMS – *Workflow Management Systems*), que permitem executar programas ou scripts usando dados de entrada selecionados, e onde os resultados gerados podem ser analisados e visualizados dentro do próprio sistema. Uma das grandes vantagens existentes neste tipo de abordagem consiste no fato dos fluxos poderem ser re-executados quando necessário, e os resultados obtidos a partir de execuções anteriores permanecerem disponíveis para consulta e análise a qualquer momento. A capacidade de compartilhamento destes dados e de *workflows* inteiros entre os usuários do sistema também é uma característica bastante comum destes ambientes, podendo motivar eventos de colaboração sinergística nos projetos de pesquisa.

Normalmente estes sistemas são desenvolvidos usando-se uma infraestrutura composta por diversos recursos computacionais (servidores *Web*, servidores de e-mail, bancos de dados, dispositivos de armazenamento de dados, etc.). Estes recursos podem ser reunidos em larga escala para serem usados colaborativamente

em *workflows* científicos que demandam grande poder de processamento (BOHLE, 2014; HEY & TREFETHEN, 2005; FOSTER, 2005).

Atualmente existem diversas tecnologias e padrões disponíveis para o desenvolvimento de sistemas de *workflow* científico. Comparar suas características, mensurar prós e contras entre as opções existentes e escolher a melhor delas para servir como base para o desenvolvimento de um sistema real, não é uma tarefa fácil. Neste ponto, vale ressaltar que não existem soluções prontas (*off-the-shelf*) que não necessitem de adequações mínimas para aplicação em sistemas desta natureza. O trabalho exigido nesta tarefa pode resultar, em algumas situações, na decisão de desenvolver um sistema de *workflow* científico sem o uso de nenhuma tecnologia ou padrão disponível.

Outro fator que pode pesar neste tipo de decisão é a possibilidade de descontinuidade (abandono) dos projetos de desenvolvimento/atualização por seus mantenedores (CURCIN & GHANEM, 2008; BARKER & VAN HEMERT, 2008). Desta forma, para evitar problemas desta natureza, muitas vezes vale a pena investir na implementação de uma plataforma própria, ao invés de utilizar como ponto de partida, iniciativas de terceiros. Além disso, no âmbito de um projeto de doutorado, acredita-se que a curva de aprendizado experimentada durante o desenvolvimento e a implementação de uma plataforma própria contribua muito mais para a formação do doutorando.

#### *3.1.2.1 Tecnologias para sistemas de workflow científico*

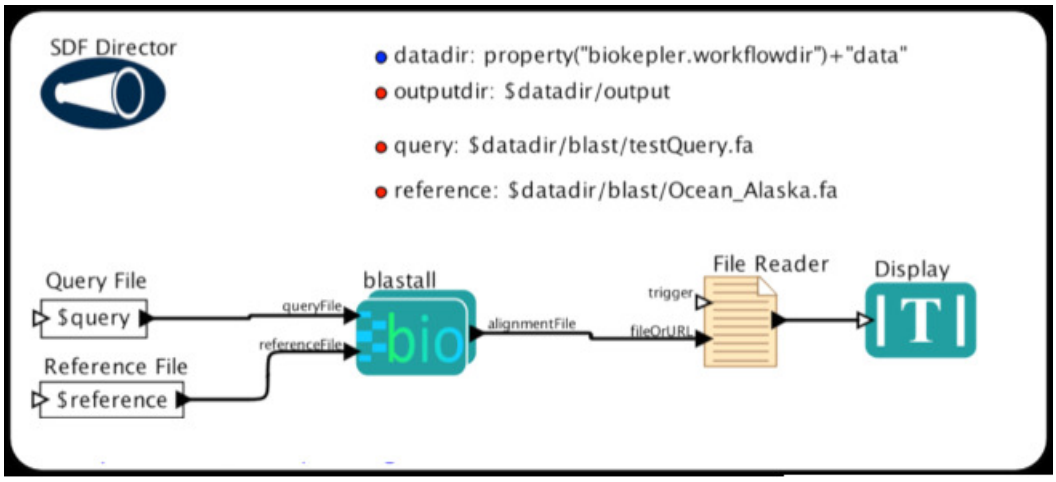
Atualmente, sistemas de gerenciamento de *workflows* podem ser desenvolvidos através de diversos arcabouços (*frameworks*) disponíveis para construção de *workflows* em qualquer domínio do conhecimento. A seguir, serão apresentados alguns exemplos de arcabouços bastante utilizados para estes propósitos.

#### *3.1.2.1.1 Kepler*

Kepler (ALTINTAS *et al.*, 2004; LUDÄSCHER *et al.*, 2006) é um arcabouço para a construção de sistemas de gerenciamento de *workflows* científicos, desenvolvido pela Universidade da Califórnia (https://kepler-project.org/). Ele permite criar, executar, arquivar e compartilhar dados provenientes de *workflows* de várias áreas do conhecimento, como por exemplo, biologia molecular, ecologia, genética, física, entre outros.

Através da interface gráfica do Kepler, é possível selecionar e conectar os componentes (analíticos e de dados) que irão compor um *workflow*, ou seja, formar a sequência completa de passos requerida para gerar os resultados desejados. Os *workflows* podem ser aninhados de forma que alguns possam ser conectados a outros, e onde tarefas complexas possam ser elaboradas a partir de componentes mais simples, promovendo com isso, a reusabilidade de *workflows* e componentes.

Figura 3.2. Exemplo de workflow criado no sistema Kepler que executa uma ferramenta de alinhamento (BLAST) para duas cadeias de caracteres no formato FASTA.




Fonte: http://www.biokepler.org/userguide, 2016.

A Figura 3.2 apresenta um exemplo de *workflow* cujo objetivo é realizar o alinhamento de duas cadeias de caracteres contendo informações biológicas (sequência de aminoácidos de diferentes proteínas ou de nucleotídeos de DNA) usando a ferramenta de alinhamento BLAST (ALTSCHUL *et al.*, 1990). Os parâmetros de entrada "$*query*" e "$*reference*" são as sequências de dados em formato FASTA (LIPMAN & PEARSON, 1985) que serão alinhadas. Após a

execução da ferramenta BLAST, o arquivo gerado "*alignmentFile*" é lido e finalmente exibido na interface gráfica do sistema.

Com o objetivo de obter melhor desempenho (menor tempo de resposta) na execução dos *workflows*, o módulo de execução (*run-time engine*) do sistema Kepler possibilita que os *workflows* sejam executados através de uma infraestrutura de computação distribuída disponível, ou seja, por meio de um *cluster* de computadores (SADASHIV & KUMAR, 2001) ou de um *Grid* computacional (FOSTER, 2002; FOSTER *et al*., 2001).

Uma das grandes vantagens do sistema Kepler é permitir a busca e utilização de componentes presentes em seu repositório, que possui cerca de 350 componentes prontos para uso. Cada um deles pode ser configurado, conectado a um *workflow* existente, e executado a partir da interface gráfica do sistema, ou através de linha de comando. O repositório de componentes do Kepler permite ainda que componentes e *workflows* possam ser compartilhados entre outros usuários da comunidade que utiliza esta plataforma.

#### *3.1.2.1.2 Taverna*

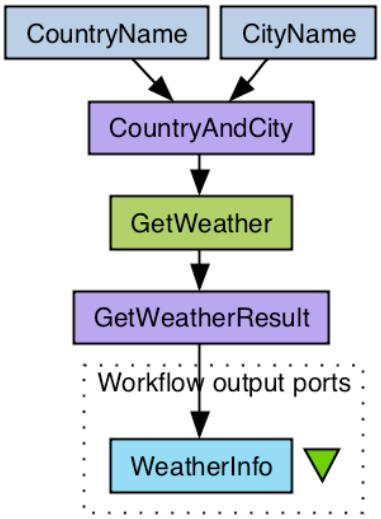
Taverna (WOLSTENCROFT *et al*., 2013; BELHAJJAME *et al*., 2008) é um arcabouço para construção de sistemas de gerenciamento de *workflow* científico semelhante ao Kepler, porém, desenvolvido e atualizado pela Universidade de Manchester (http://www.taverna.org.uk/). Taverna é usado para compor *workflows* em vários domínios do conhecimento, tais como, bioinformática, química, astronomia, mineração de dados (*data mining*) e análise de imagens.

Como qualquer sistema de gerenciamento de *workflows*, Taverna permite que o usuário defina como os dados vão percorrer as diversas etapas (serviços/componentes) do fluxo, convertendo-os de um formato para outro nos casos onde os serviços não são compatíveis entre si. Além disso, ele permite a execução dos workflows sem a preocupação de como seus serviços serão executados em sequência.

A Figura 3.3 apresenta um exemplo de *workflow* criado no sistema Taverna, cujo propósito é recuperar as informações sobre o clima de uma determinada cidade

a partir de dois parâmetros de entrada: nome da cidade e nome do país do qual a cidade pertence.

Figura 3.3. Exemplo de workflow criado no sistema Taverna que recupera informação do clima para uma determinada cidade.




Fonte: http://www.taverna.org.uk/introduction/why-use-workflows, 2016.

O sistema é composto por cinco módulos principais. Taverna *Engine*, que é o módulo responsável pela execução dos *workflows*. Este módulo pode ser acionado por dois outros módulos: Taverna *Workbench* e Taverna *Server*. O primeiro deles consiste em um programa com interface gráfica (GUI) onde o usuário tem acesso ao sistema, e, através dele, aciona a execução dos *workflows*. O segundo é o módulo "servidor", ou seja, aquele que recebe acionamentos de execuções remotas através de acesso *Web*. O quarto módulo é o *Comand Line Tool* que possibilita a execução de *workflows* através de um terminal (linha de comando). Por último, o módulo Taverna Online, que disponibiliza uma interface pela Web para a criação, edição e execução de workflows (OINN *et al.*, 2004; HULL *et al.*, 2006).

As principais funcionalidades e características do sistema de *workflows* Taverna são:

- Interface gráfica para criar/editar *workflows* arrastando componentes na tela (*drag-and-drop*) e conectando-os entre si;

- Recurso para execução de *workflows* em ambientes de computação distribuída (*Grids* computacionais), aumentando o desempenho através da aceleração da execução dos *workflows*;
- Arquitetura extensível, que permite adicionar novos serviços de um repositório que conta com mais de 3500 serviços disponíveis, e também, a partir de catálogos de serviços externos, como por exemplo, o BioCatalogue (BHAGAT *et al*., 2010);
- Compartilhamento de *workflows* que possibilita a busca e utilização de *workflows* criados por outros usuários do sistema, assim como a publicação dos próprios *workflows*; e,
- Monitoramento da execução de *workflows* e resultados, que permite a visualização de detalhes referentes à execução de cada serviço, e também dos resultados gerados em cada uma das etapas do *workflow*.

O sistema também permite adicionar novos módulos auxiliares (*plugins*) para expandir suas funcionalidades. Os *plugins* são desenvolvidos tanto internamente pela equipe do projeto Taverna, quanto por colaboradores externos. Um dos principais *plugins* é o CDK-Taverna (KUHN *et al*., 2010) que integra o sistema com a biblioteca CDK (*Chemistry Development Kit*) (STEINBECK *et al*., 2003), contendo esta biblioteca inúmeras rotinas (funcionalidades) usadas nas áreas de bioinformática e química computacional. Através do uso deste *plugin*, é possível incluir aos *workflows* criados no sistema os componentes que encapsulam as funcionalidades providas por esta biblioteca.

#### *3.1.2.1.3 KNIME*

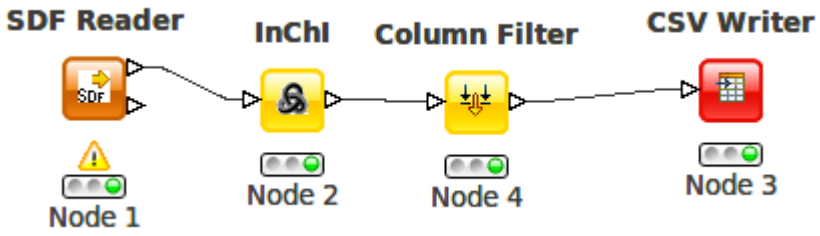
KNIME (BERTHOLD *et al*., 2008) é um sistema de gerenciamento de *workflows* inicialmente desenvolvido pela Universidade de Konstanz (http://www.knime.org/). Apesar de ser concebido para ser usado no desenvolvimento de *workflows* em qualquer área do conhecimento, KNIME vem sendo largamente empregado em pesquisas voltadas para descoberta de fármacos em grandes companhias farmacêuticas (TIWARIA & SEKHARB, 2007).

Assim como outros sistemas de gerenciamento de *workflows*, KNIME também permite a inclusão de novas funcionalidades por meio de "*plugins*". Entre os

"*plugins*" disponíveis, vale a pena destacar o KNIME-CDK (BEISKEN *et al*., 2013) que disponibiliza as funcionalidades da biblioteca CDK, já mencionada na seção anterior, e o WEKA *Integration*, que, por sua vez, disponibiliza rotinas da biblioteca WEKA (HALL *et al*., 2009). Esta biblioteca consiste numa coleção de mais de 100 rotinas com aplicação na área de inteligência artificial (algoritmos de regressão, classificação e mineração de dados).

A Figura 3.4 apresenta um exemplo de *workflow* criado no sistema KNIME, cuja finalidade é ler um arquivo no formato SDF (DALBY *et al*., 1992), extrair para cada molécula presente no arquivo o seu respectivo identificador InChi (HELLER *et al*., 2013) e exportar estes identificadores em um arquivo de saída no formato CSV (*Comma Separated Values*).

Figura 3.4. Exemplo de workflow criado no sistema KNIME que recupera identificadores InChi a partir de um arquivo SDF




Fonte: http://depth-first.com/articles/2010/10/22/cheminformatics-data-pipelining-with-knime-creating-a-custon-node-for-generating-inchis-and-inchi-keys, 2016.

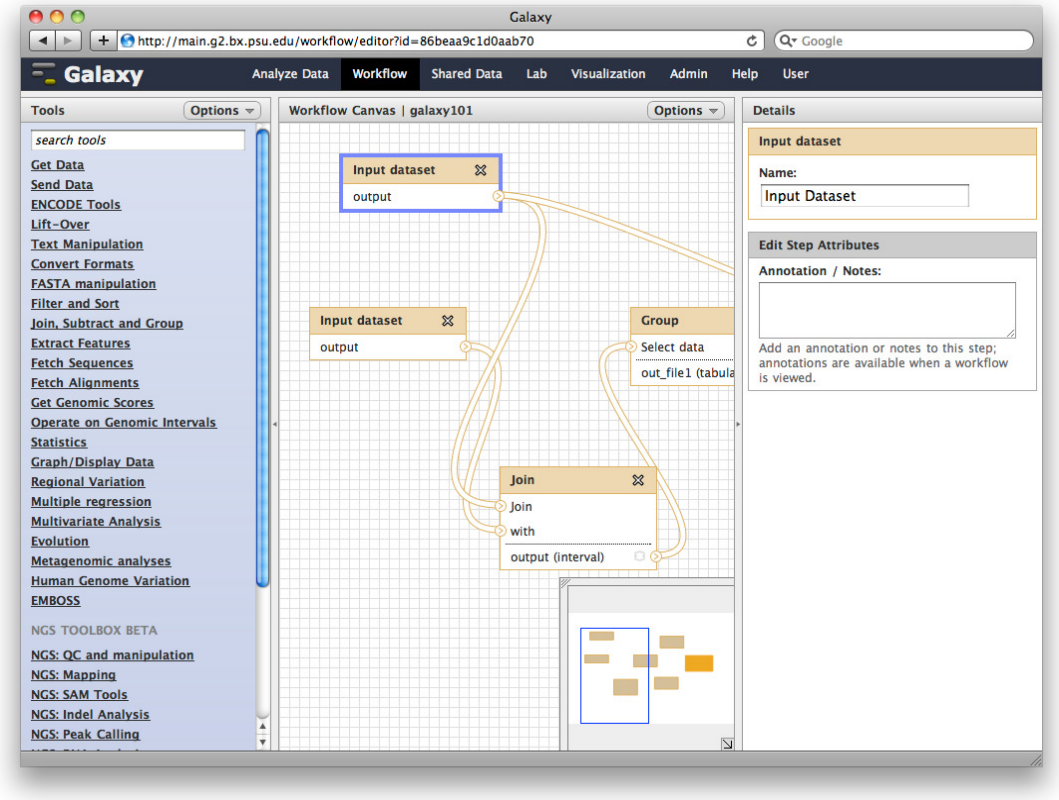
### *3.1.3 Exemplos de portais científicos*

Diversas iniciativas de aplicação de portais científicos podem ser encontradas na literatura. Serão apresentadas, a seguir, alguns exemplos destas aplicações. Vale ressaltar que, apesar da existência de várias tecnologias e padrões bem estabelecidos, cada uma das iniciativas segue direcionamentos próprios para a modelagem, construção e usabilidade dentro destes ambientes.

### *3.1.3.1 Galaxy*

Galaxy (GOECKS *et al.*, 2010; BLANKENBERG *et al.*, 2010; GIARDINE *et al.*, 2005) é um portal científico com aplicações na área de biologia computacional (bioinformática). Este sistema fornece uma coleção de ferramentas direcionadas para estudos de genômica que podem ser usadas como componentes na criação de *workflows* científicos. Para usar os serviços do Galaxy, o usuário pode tanto acessar diretamente o sistema que está hospedado no sítio do projeto (https://usegalaxy .org/), quanto instalar uma "instância" do sistema localmente em sua máquina pessoal.

Figura 3.5. Área de trabalho do sistema Galaxy onde é apresentado um workflow em edição.




Fonte: https://usegalaxy.org/u/aun1/p/galaxy101, 2016.

Por meio da interface Web, o sistema inclui, entre outras facilidades, a possibilidade de trabalhar com dados importados da máquina do usuário (*upload*), como também provenientes de fontes de dados públicas de sequências de nucleotídeos de DNA ou de aminoácidos de proteínas, como é o caso do BioMart

(KASPRZYK, 2011), do ENA (*European Nucleotide Archive*) (http://www.ebi.ac.uk/ena/) e do banco de dados do instituto NHGRI (*National Human Genome Research Institute*) (http://www.genome.gov/). É possível ainda realizar análises de alinhamento de sequências, filtragem (seleção), classificação e combinação de conjuntos de dados, geração de estatísticas e compartilhamento de informações entre os usuários.

A Figura 3.5 apresenta a área de trabalho do sistema Galaxy (no portal *Web* do projeto) onde é possível visualizar um exemplo de *workflow* criado nesta plataforma. Na parte à esquerda da tela, é apresentada a lista de serviços disponíveis no sistema. Na parte central da tela, o workflow que está sendo criado/editado, e na parte à direita, os detalhes e propriedades dos serviços (componentes) selecionados pelo usuário.

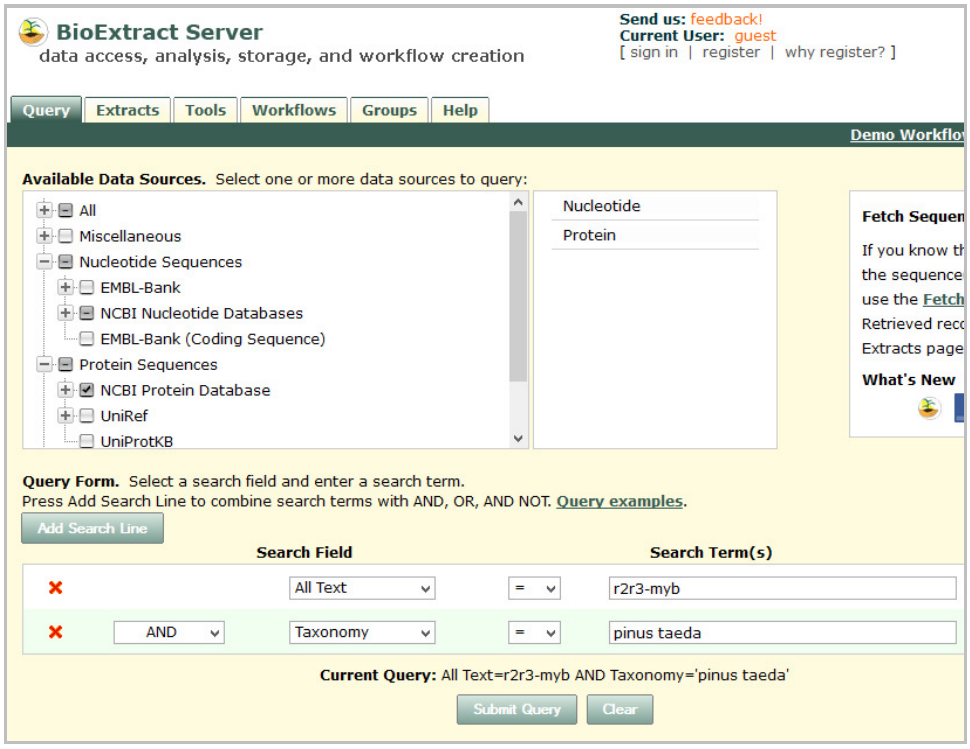
#### *3.1.3.2 BioExtract*

BioExtract (LUSHBOUGH *et al*., 2008; LUSHBOUGH & BRENDEL, 2010) é um portal científico disponibilizado na *Web* (http://bioextract.org/) que fornece acesso a um conjunto de ferramentas com aplicações em biologia computacional. O sistema permite que o usuário pesquise e recupere cadeias de nucleotídeos de DNA e aminoácidos de proteínas em vários bancos de dados públicos (NCBI *Protein Database*, NCBI *Nucleotide Database*, EMBL-Bank, e UniProt, por exemplo), para a realização de estudos sobre estas sequências.

A Figura 3.6 apresenta a área de trabalho do portal *Web* do BioExtract. O exemplo da figura consiste na pesquisa de uma determinada sequência em bancos de dados de nucleotídeos de DNA (NCBI *Nucleotide Databases*) e de aminoácidos de proteínas (NCBI *Protein Database*).

Embora com uma interface gráfica limitada, o BioExtract ainda permite a criação e execução de *workflows* formados pelos serviços presentes no sistema. O portal também possibilita que os dados biomoleculares extraídos (e analisados) e os *workflows* criados possam ser compartilhados entre os usuários do sistema.

Figura 3.6. Área de trabalho do sistema BioExtract




Fonte: http://bioextract.org, 2016.

### *3.1.3.3 Mobyle@RBPS*

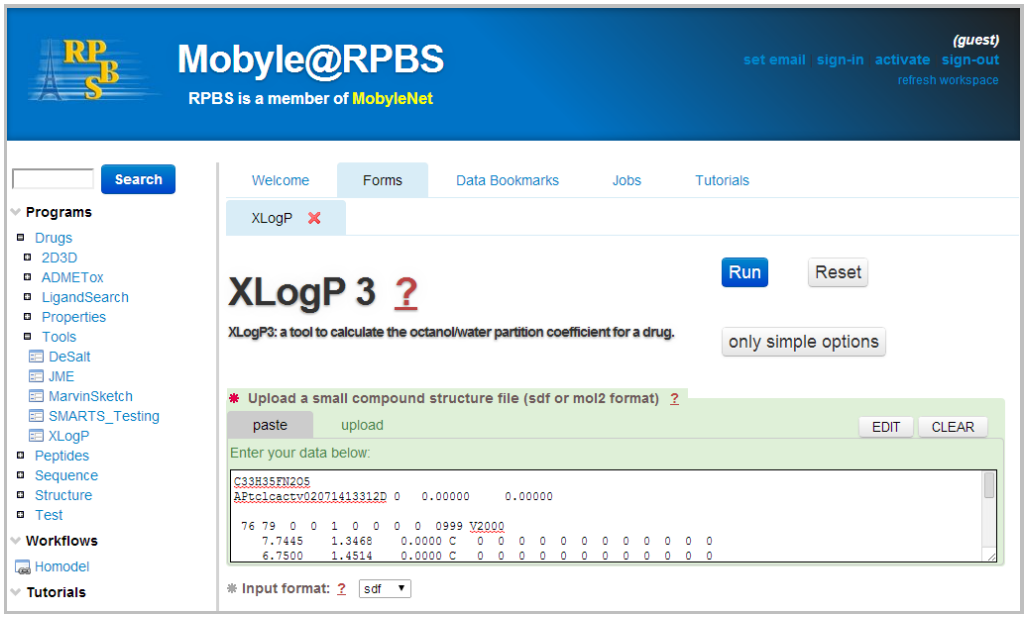
Mobyle@RBPS (NÉRON *et al.*, 2009; ALLAND *et al.*, 2005) é um portal científico na *Web* (http://mobyle.rpbs.univ-paris-diderot.fr/) que fornece serviços de bioinformática e química computacional. Diferente dos sistemas mencionados nas seções anteriores, Mobyle@RBPS não possui como foco principal, o desenvolvimento de *workflows*. Seu maior objetivo é disponibilizar um ambiente integrado de serviços centrado no usuário do sistema, ou seja, onde a usabilidade (conceito relacionado ao nível de intuição e facilidade de uso que uma interface gráfica possui) é considerado o aspecto mais importante do projeto.

Esta preocupação é motivada por um estudo que mostra como uma interface gráfica mal formulada pode impedir o uso de sistemas computacionais por pesquisadores (GORDON & SENSEN, 2007). Desta forma, o portal *Web* do Mobyle@RBPS foi criado para ser o mais simples possível, em detrimento de outros sistemas que exigem um maior nível de detalhamento na modelagem e execução

das tarefas, sobretudo, sistemas tradicionais de gerenciamento de *workflows*, tal como o Galaxy, discutido anteriormente neste trabalho.

A Figura 3.7 apresenta a área de trabalho do portal *Web* do Mobyle@RBPS. No painel da esquerda, encontram-se todas as ferramentas disponíveis no portal. Ao selecionar alguma delas, por exemplo, o serviço "XLogP 3", imediatamente um formulário é apresentado, solicitando do usuário, os parâmetros de entrada e saída necessários para o serviço ser executado apropriadamente.

Figura 3.7. Área de trabalho do portal Web Mobyle@RBPS onde é exibido o formulário para execução de uma ferramenta "XLogP 3".




Fonte: http://mobyle.rpbs.univ-paris-diderot.fr/ cgi-bin/portal.py#welcome, 2016.

Uma vez preenchidos estes parâmetros, o usuário solicita a execução do programa e aguarda, monitorando seu andamento na aba "*Jobs*", até ele ser concluído. Vale salientar que a todo instante, é possível enviar arquivos formatados com dados para o sistema (*upload*), assim como recuperar dados gerados a partir da execução das ferramentas no portal (*dowload*) na aba "*Data Bookmarks*".

### *3.1.3.4 Portais SINAPAD*

O Sistema Nacional de Processamento de Alto Desempenho (SINAPAD) mantém diversos portais científicos na Web, que permitem a submissão de *jobs* a

partir de diferentes aplicações científicas. Estes portais podem ser acessados diretamente por seus endereços (URLs) ou a partir da página de portais do SINAPAD (https://www.lncc.br/sinapad/portais.php), que reúne a lista de todos os portais disponíveis para a comunidade acadêmica.

Figura 3.8. Área de trabalho do portal científico ProFrager do SINAPAD.


Fonte: https://www.lncc.br/sinapad/Profrager, 2016.

Estes portais acessam as infraestruturas computacionais dos diversos Centros de Processamento de Alto Desempenho (CENAPAD) que compõem o sistema nacional, para realizar a execução dos *jobs* dos usuários. Os portais foram desenvolvidos usando o *middleware* CSGrid e as ferramentas PortEngin e PortEditor, já comentadas na seção 3.1.1.3. Dentre os diversos portais científicos disponibilizados pelo SINAPAD, pode-se destacar os portais: i) DockThor, usado para a submissão de cálculos de *docking* molecular; ii) ProFraGer, serviço destinado

à geração de bibliotecas de fragmentos de proteínas; e, iii) Gaussian, que permite a execução de cálculos de química quântica com o software de mesmo nome.

A Figura 3.8 apresenta a área de trabalho parcial da página Web do portal científico ProFrager. Este portal, em particular, requer um grande número de parâmetros de entrada, que, assim como os outros portais presentes no SINAPAD, são definidos através de arquivos de configuração em formato XML (através da ferramenta PortEditor) e refletidos na interface gráfica (em HTML) presente no portal.

### *3.2 Aplicações científicas em dispositivos móveis*

Atualmente, vivemos em um período imerso em inovações tecnológicas, onde temos à disposição um número cada vez maior de dispositivos eletrônicos (celulares, PDAs, *smartphones* e *tablets*) que facilitam o dia-a-dia ao agilizar tarefas rotineiras, assim como, permitem o acesso à uma quantidade crescente de informação, principalmente através do acesso à Internet. A mobilidade foi uma das principais vantagens presentes no uso destes dispositivos, em razão da possibilidade de acesso a informações a todo o momento e em qualquer lugar com acesso à Internet.

Outro importante aspecto que impulsionou a disseminação destes equipamentos foi o aumento de sua capacidade computacional (processamento de informações e armazenamento de dados) que abriu possibilidades para o desenvolvimento de aplicativos outrora tradicionalmente focados nos computadores pessoais (PC). Esta evolução tecnológica pode ser exemplificada pelos *tablets* mais recentes (WUEBBLING, 2014), que possuem processador NVIDIA® Tegra™ 4 com GPU GeForce™ de 72 núcleos (*cores*) e um CPU ARM® com quatro núcleos (*quad-core*).

Um exemplo de iniciativa que denota bem como as tecnologias móveis tendem a convergir em uma plataforma unificada, é o caso do Ubuntu Touch. Baseado no sistema operacional de código-aberto (*open-source*) Ubuntu (http://www.ubuntu.com/), esta versão é disponibilizada para ser usada em *smartphones* e *tablets* (ver Figura 9), com o intuito de aproveitar ao máximo o desempenho destes dispositivos móveis, que vem crescendo a cada dia. Um dos recursos do Ubuntu Touch, que vale a pena citar, é a sua capacidade de interconexão entre o *smartphone* e o computador (ambos usando o sistema

operacional Ubuntu), que possibilita transformar o *smartphone* numa extensão do computador, permitindo seu acesso direto pelo dispositivo de forma similar a um terminal remoto (*thin client*).

Figura 3.9. Exemplos de dispositivos móveis utilizando o sistema Ubuntu Touch.


Fonte: http://www.ubuntu.com/phone, 2016.

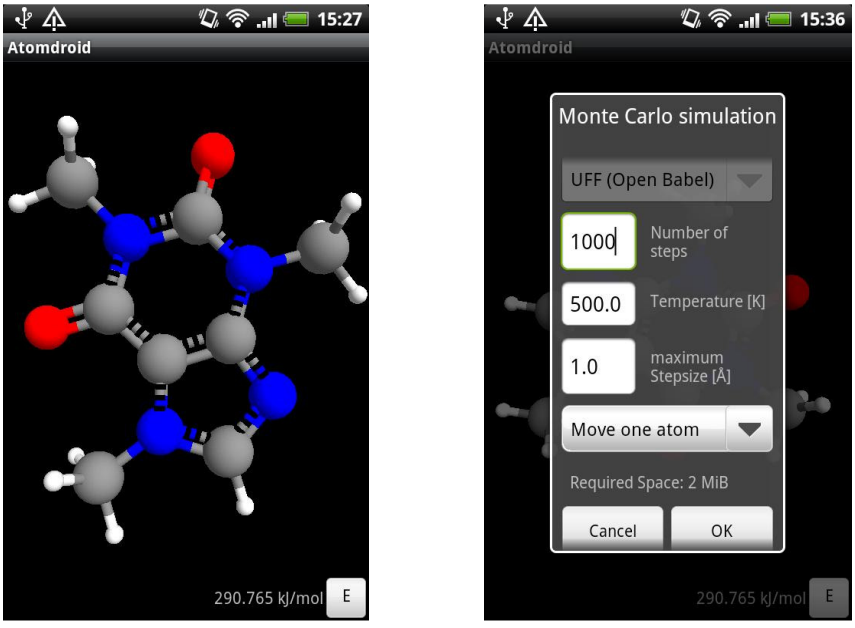
Exemplos de aplicativos para estes tipos de dispositivos podem ser encontrados atualmente para os segmentos comercial, educacional, de entretenimento, de governo eletrônico (e-GOV), e recentemente também na área científica. Em geral, aplicativos móveis podem ser desenvolvidos de duas formas diferentes: como uma aplicação nativa do dispositivo ou como uma aplicação Web.

No primeiro caso, o aplicativo é desenvolvido especificamente para uma plataforma (ou sistema operacional do dispositivo), onde é instalado como um aplicativo nativo daquela plataforma (app). Neste cenário, para atender várias plataformas diferentes, é necessário desenvolver o mesmo aplicativo para cada uma das plataformas requeridas. No segundo caso, o aplicativo é desenvolvido usando-se tecnologias Web (HTML5, Javascript e CSS), e acessado através de um navegador (*browser*). Desta forma, a aplicação pode rodar em diversas plataformas, sem a necessidade de desenvolvimento de múltiplas versões. Neste ponto, aplicações Web não dependem diretamente do usuário para serem atualizadas (a atualização ocorre no servidor), como é o caso de aplicações nativas, além de não sofrerem impacto direto com as constantes mudanças de versões das plataformas móveis. Em contrapartida, aplicações Web não possuem a interface nativa característica de cada dispositivo, uma vez que ela é desenvolvida para atender

múltiplas plataformas. Outra desvantagem está no fato de que estas aplicações geralmente necessitam de conexão com a Internet para funcionar (*on-line*).

Entre os aplicativos móveis com foco na área científica, apesar de poucas iniciativas de desenvolvimento até o presente momento, pelo menos em comparação com os aplicativos disponíveis para outras finalidades (entretenimento, e-mail, ferramentas, etc.), é possível visualizar alguns exemplos, inclusive, voltados para a área de modelagem molecular, como é o caso do aplicativo Atomdroid (FELDT *et al.*, 2012) (Figura 3.10) desenvolvido para a plataforma Android (http://www.android.com/). Este aplicativo permite, dentre outras coisas, o desenho (edição) e a visualização molecular, além da otimização de geometria com funções de potencial (campo de força) UFF (Universal Force Field) (RAPPE *et al.*, 1992), e o acesso das informações (*download*) de estruturas de proteínas e de outras bio-macromoléculas provenientes do banco de dados público PDB (Protein Data Bank) (PDB, 2014).

Figura 3.10. Interface gráfica do aplicativo móvel Atomdroid. A figura (a) apresenta a tela de visualização molecular para uma pequena molécula, e a figura (b) apresenta a tela com os parâmetros usados no campo de força para otimização de geometria.




(a) (b)

Fonte: https://play. google.com/store/apps/details?id=org.atomdroid&hl=pt_BR, 2016.

Outro aplicativo desenvolvido para plataformas móveis, com foco em modelagem molecular, é o *software* RCSB PDB *Mobile* (ROSE *et al*., 2012) (Figura 3.11), criado e distribuído gratuitamente para as plataformas Android e iOS (iPhone/iPad) (http://www.apple.com/). Este aplicativo permite a visualização das estruturas moleculares provenientes do banco de dados PDB, além de buscar através do nome da macromolécula, nome do autor ou código (ID) PDB, e de fazer a leitura de informações e notícias publicadas pelo portal PDB, dentre outros.

Figura 3.11. Interface gráfica do aplicativo móvel RCSB PDB Mobile. A figura (a) apresenta a tela principal do aplicativo com as diversas opções disponíveis, e a figura (b) apresenta a tela de visualização molecular para uma proteína (código PDB: 3SY7).

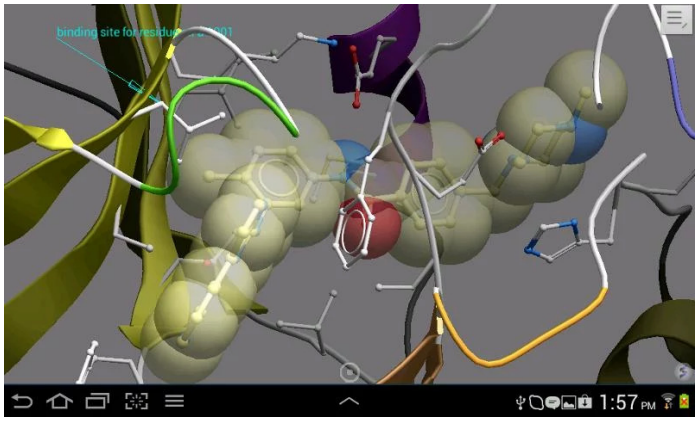

(a) (b)

Fonte: https://play. google.com/store/apps/details?id=org.rcsb.rcsbpdbapp, 2016.

Outros exemplos interessantes de aplicativos móveis (gratuitos) com aplicação em modelagem molecular incluem: Jmol Molecular Visualization (JMOL APP, 2014), iMolview Lite (IMOLVIEW, 2014) e NDKmol (NDKMOL, 2014). Estes aplicativos de visualização molecular permitem que o usuário explore estruturas moleculares usando diversos recursos de interação e visualização (renderização). A Figura 3.12, a seguir, apresenta a tela da interface gráfica do aplicativo iMolview Lite, cujo conteúdo exibe algumas estruturas moleculares em esquemas de

visualização diferentes (*ball-and-stick*, *ribbon diagram, cartoon*, e superfície molecular).

Figura 3.12. Interface gráfica do aplicativo móvel iMolview Lite exibindo estruturas moleculares em diferentes esquemas de visualização.




Fonte: https://play.google.com/store/apps/details?id=com.molsoft.imolview, 2016.

Estas iniciativas de aplicativos móveis corroboram a necessidade que vem surgindo de novos desenvolvimentos de aplicações na área de inovação terapêutica (*in silico*), que inclui, sobretudo, novas plataformas computacionais.

O desenvolvimento de aplicativos científicos voltados para dispositivos móveis possui bastante potencial ao longo dos próximos anos, visto que existe uma forte tendência que estes dispositivos cada vez mais façam parte do nosso dia-a-dia (FRAGA, 2012; DAQUINO, 2013; PLAZA, 2014). Este conceito aplicado à área de planejamento de fármacos também se mostra como uma proposta bastante promissora pela inovação intrínseca em disponibilizar, para estes novos tipos de dispositivos, aplicativos interativos e funcionais que permitam trabalhar com metodologias teóricas amplamente consolidadas dentro da química medicinal, como por exemplo, os estudos de relação quantitativa estrutura-atividade (QSAR) e de *docking* molecular, além de alinhamento, visualização molecular, e outras metodologias *in silico* (HUANG *et al.*, 2010; DALKAS *et al.*, 2012; MACALINO *et al.*, 2015).

# 4 METODOLOGIA

Portais científicos podem ser vistos como facilitadores (mediadores) entre recursos computacionais de uma instituição (ou de um conjunto delas definido através de uma organização virtual) e seus usuários de interesse. O sistema MODiMOL *Workbench* foi primariamente projetado e desenvolvido com este propósito, porém, também almejando dois importantes aspectos: i) tornar a experiência do usuário com este tipo de sistema computacional mais fácil e intuitiva, e ii) facilitar as tarefas administrativas do sistema.

Estas premissas são tratadas, neste trabalho, através de alguns conceitos-chave. Em primeiro lugar, pesquisadores frequentemente necessitam submeter seus cálculos ou simulações computacionais para execução ("*jobs*") diversas vezes antes de alcançar os resultados esperados. Este é um aspecto importante, visto que, por exemplo, em estudos de modelagem molecular, é comum a necessidade de refinar os resultados e, portanto, a disponibilidade das informações específicas de resultados anteriores auxilia no processo de análise e refinamento destes procedimentos em direção à obtenção de resultados aprimorados.

Desta forma, estes *jobs* estão logicamente associados entre si, e portanto, devem ser mantidos (organizados) juntos, uma vez que foram criados para o mesmo propósito final, ou seja, para uma mesma atividade de pesquisa. Este conceito é implementado neste trabalho, por meio da organização dos *jobs* em entidades denominadas "atividades". Uma atividade pode ser criada para mapear todas as ações realizadas (submissões de cálculos para uma ou mais aplicações científicas) ao longo do período de trabalho em uma determinada atividade (pesquisa) científica. Os dados gerados, resultantes destas execuções, são disponibilizados para análise e visualização, e permanecem armazenados para eventuais consultas futuras, e também como cópias de segurança (*backups*).

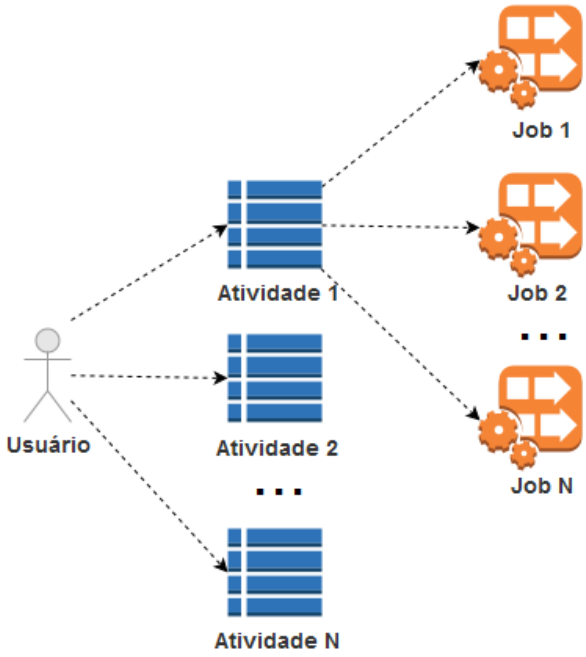
Adicionalmente, o número de *jobs* criados dentro do sistema, para um determinado usuário, pode crescer demasiadamente, portanto, a consulta e visualização de um determinado *job* pode se tornar uma tarefa bastante difícil, uma vez que *jobs* oriundos de estudos de modelagem molecular diferentes estariam disponíveis numa listagem única para o mesmo usuário. Desta forma, agrupá-los sob uma mesma "atividade", torna mais fácil a tarefa de acessar e visualizar

resultados históricos de uma mesma atividade de pesquisa e melhora o nível de compreensão e organização sobre os dados gerados pelos usuários.

A Figura 4.1 apresenta o esquema utilizado para a organização de *jobs* no sistema MODiMOL *Workbench*. Cada usuário pode criar diversas atividades, e cada uma destas atividades podem reunir um ou mais *jobs*.

De forma semelhante à iniciativa Mobyle@RBPS (NÉRON *et al.*, 2009) (ALLAND *et al.*, 2005), o sistema MODiMOL *Workbench* tem como principal preocupação, simplificar o acesso e o uso de ferramentas de modelagem molecular. Para alcançar este objetivo, faz-se necessário desenvolver uma interface com o usuário que seja o mais simples e intuitiva possível, no que diz respeito ao uso do sistema, isto é, no gerenciamento de usuários, atividades e *jobs*, e na visualização dos resultados gerados a partir da execução das ferramentas de modelagem.

Figura 4.1. Esquema de organização de atividades e jobs do sistema MODiMOL Workbench.




Fonte: O Autor, 2016.

Neste sentido, foi desenvolvida uma interface Web para disponibilizar o acesso do usuário ao sistema MODiMOL *Workbench*. Esta decisão foi motivada pelo fato do uso deste tipo de abordagem permitir que o sistema possa ser acessado através de um navegador *Web* (Ex.: Firefox, Internet Explorer e Chrome), dispensando a necessidade de instalação e utilização de um programa específico

(programa cliente) para esta finalidade. Além disto, um grande conjunto de tecnologias e padrões estão disponíveis atualmente para o desenvolvimento de sistemas de informação para a *Web*, fazendo com que este tipo de interface possa ser criada e atualizada com maior agilidade, tendência esta seguida pela maioria dos portais científicos existentes atualmente (ver capítulo 3).

Em relação aos aspectos administrativos, uma das principais funcionalidades providas por esta plataforma consiste na habilidade de incluir dinamicamente novos algoritmos (aplicações científicas) sem a necessidade de reinicialização do sistema, ou recompilação do código-fonte e geração de uma nova versão compilada do sistema, o que é um cenário muito típico em portais científicos existentes, nos quais novos algoritmos são inclusos como elementos estáticos ("*hard-coded*") no código-fonte com suas respectivas interfaces gráficas.

Portanto, nesta direção, todos os dados de configuração necessários para o algoritmo funcionar corretamente são providos durante a execução do sistema, incluindo elementos de interface gráfica e suas conexões com os parâmetros de entrada do serviço disponibilizado para o usuário. Nenhum arquivo de configuração externo é necessário, cabendo ao administrador do sistema a tarefa de gerenciamento dos algoritmos disponibilizados pelo portal MODiMOL *Workbench*.

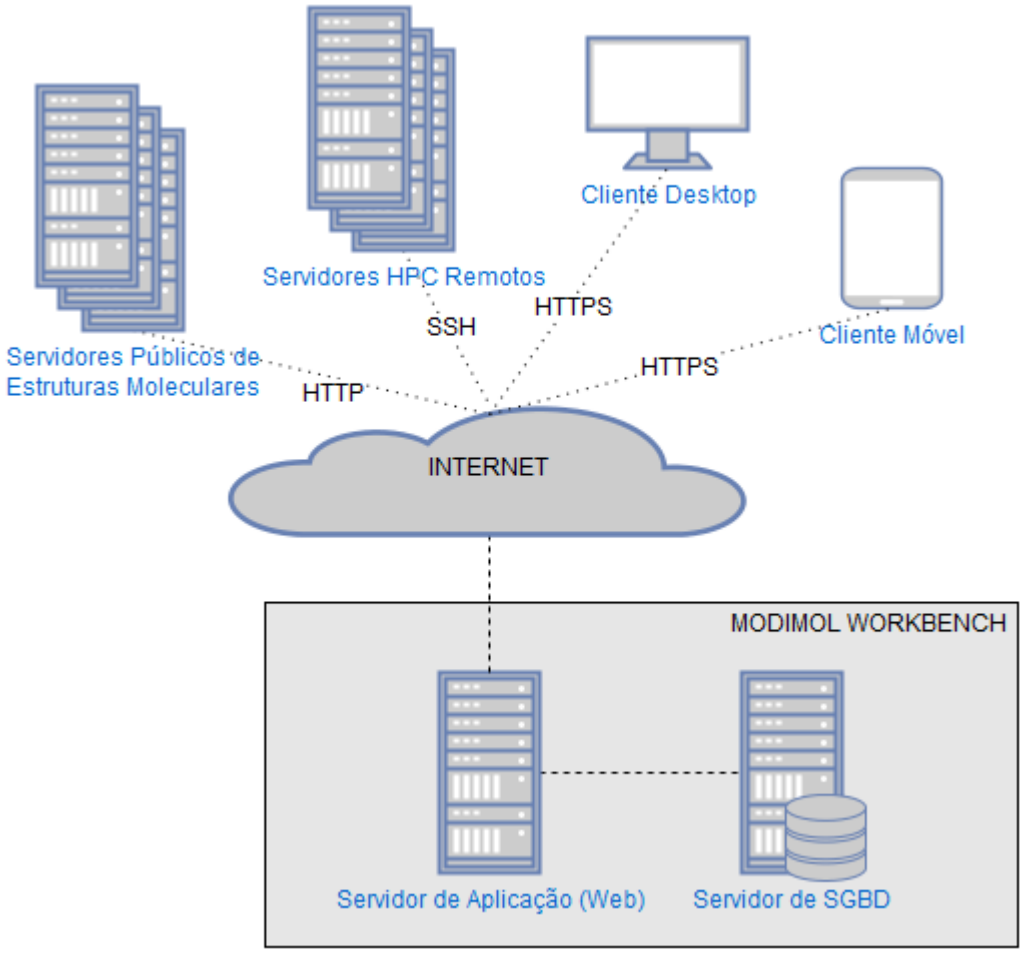
Dois importantes conceitos devem ser distinguidos neste ponto para uma melhor compreensão deste trabalho: algoritmo e serviço. Tipicamente, um algoritmo (ou simplesmente software, como é tratado no sistema) é um arquivo executável (binário ou script). Por outro lado, um serviço pode ser visto como uma instanciação de um algoritmo, onde é usado um conjunto específico de parâmetros de entrada.

Desta forma, para um determinado algoritmo, é possível existirem diversos serviços disponíveis no sistema, com diferentes conjuntos de parâmetros de entrada e, portanto, com propósitos diferentes para cada um deles. Neste sentido, o sistema MODiMOL Workbench permite o gerenciamento de algoritmos e serviços de forma dinâmica, conforme explicado anteriormente, o que torna mais simples e versátil a administração do sistema.

### *4.1 Arquitetura do sistema*

A Figura 4.2 ilustra a arquitetura geral do sistema. A plataforma MODiMOL *Workbench* é composta por dois principais elementos: i) um servidor de aplicação, onde o sistema está sendo executado (“rodando”), e ii) um servidor de banco de dados que contém um Sistema Gerenciador de Banco de Dados (SGBD), que é responsável por armazenar as informações do sistema.

Figura 4.2. Arquitetura do sistema MODiMOL Workbench.




Fonte: O Autor, 2016.

O portal do sistema é acessado através de um cliente Web a partir de um computador (*desktop* ou *laptop*) ou de um dispositivo portátil tipo *smartphone* ou *tablet* (Android, iPad, etc.). Todas as mensagens de rede trafegadas na comunicação realizada entre os equipamentos que acessam o sistema (clientes) e o

servidor são autenticadas e criptografadas usando o protocolo de comunicação HTTPS (SSL/TLS sobre o HTTP), usado para comunicações seguras na Web.

O sistema pode ser integrado com ambientes externos de computação de alto desempenho (HPC - *High-Performance Computing*) através de conexões em *shell* seguro (SSH). O sistema submete *jobs* para estas infraestruturas remotas, a fim de utilizar seus recursos computacionais disponíveis para realizar tarefas (cálculos) complexas e de alta demanda computacional. Além disso, o sistema provê integração, através de tecnologia de *Web Service* (WS), com alguns servidores de bancos de dados com o propósito de que os usuários do sistema possam recuperar estruturas moleculares diretamente destes serviços remotos. Este recurso será melhor detalhado na seção 4.9.

Como já discutido anteriormente, o principal objetivo do sistema é fornecer um ambiente de portal científico para permitir que usuários submetam *jobs* em infraestruturas de HPC, tipicamente externas ao domínio administrativo onde o servidor de aplicação do sistema está instalado. Porém, para pequenas aplicações utilitárias e ferramentas de baixa demanda computacional, foi incluída no sistema MODiMOL *Workbench*, a habilidade de executar estas aplicações na mesma máquina onde está sendo executado o servidor do sistema. Desta forma, o sistema inclui o conceitos de *job* “local” e “remoto” (e consequentemente, serviço local e remoto). Cabe ao administrador do sistema instalar e gerenciar estas aplicações no servidor, levando em consideração a capacidade computacional da máquina em questão, a fim de evitar sobrecarregá-la e, desta forma, comprometer o funcionamento do sistema.

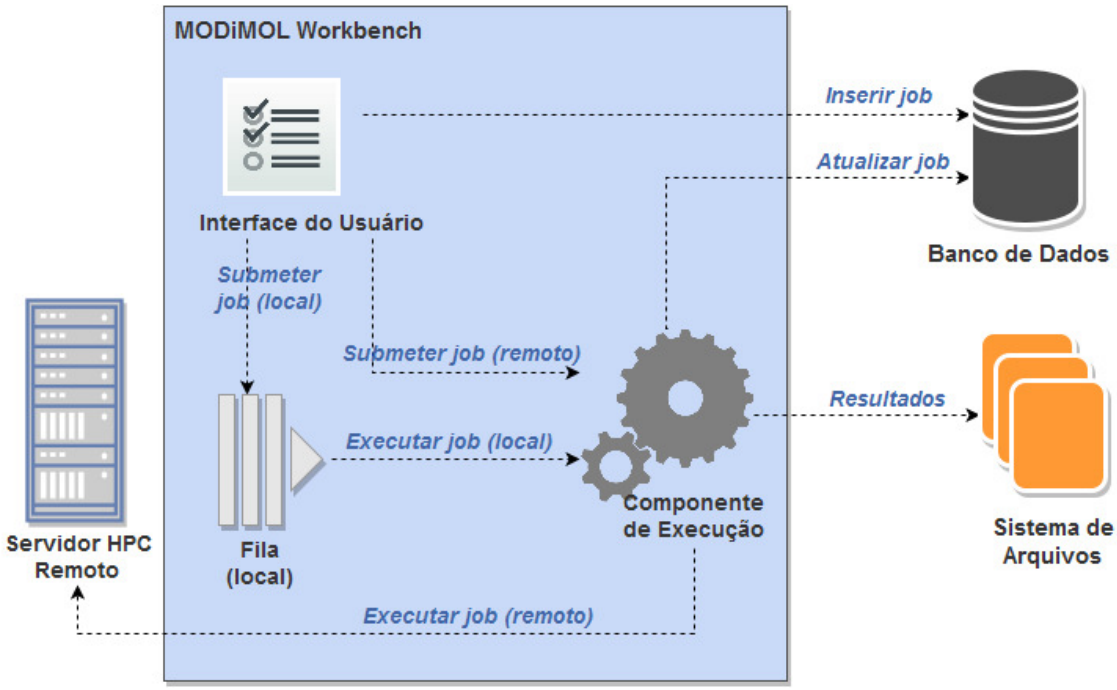
A manutenção do sistema é realizada exclusivamente por um usuário com permissões de administrador, que tem autorização para acessar e modificar todas as configurações do sistema (algoritmos, serviços, etc.). Vale a pena ressaltar que o usuário administrador é configurado enquanto uma nova instância do sistema é instalada. No Apêndice B, é possível visualizar com maiores detalhes os procedimentos técnicos para a instalação do sistema MODiMOL *Workbench*.

A Figura 4.3 apresenta o mecanismo interno para tratamento de solicitações de submissão de *jobs* para execução, dentro do sistema. Após selecionar um serviço de modelagem molecular disponível e preencher os parâmetros de entrada necessários através da interface gráfica, o usuário submete a solicitação de

execução do *job* no sistema. Em seguida, as informações relacionadas à submissão do *job* (identificador do serviço, linha de comando, data/hora da criação, nome dos arquivos de entrada) são armazenadas no banco de dados, e então, o *job* é adicionado a um contêiner ("*pool*") de *jobs* para ser processado. Existem duas destas estruturas no sistema, uma para cada tipo de *job* (local e remoto).

O contêiner de *jobs* locais é configurado de forma semelhante a uma estrutura de fila, porque limita o processamento destes *jobs* de forma seriada, ou seja, processando-os de um em um. Esta configuração é justificada pela necessidade de manter a máquina do servidor do sistema operando sem sobrecarga, uma vez que além de atender as requisições provenientes do uso normal do sistema (Web), ela também será responsável pela execução das aplicações locais.

Figura 4.3. Mecanismo de processamento dos jobs do sistema MODiMOL Workbench.




Fonte: O Autor, 2016.

Apesar disto, as configurações de ambos os contêineres de *jobs* podem ser configurados pelo administrador do sistema de acordo com os recursos disponíveis na máquina (CPU e RAM). Por exemplo, é possível aumentar arbitrariamente o número de *jobs* locais que serão processados paralelamente para dois ou mais, a

depender dos recursos computacionais presentes na máquina servidora. No caso do contêiner de *jobs* remotos, o número de *jobs* que podem ser executados paralelamente neste contêiner pode ser configurado com um valor muito maior se comparado ao correspondente no contêiner de *jobs* locais, uma vez que todos os *jobs* remotos são executados externamente ao ambiente do servidor do sistema, não onerando, neste caso, para uma sobrecarga de processamento na máquina utilizada.

Em todo o processo de submissão e execução, o *job* é armazenado e atualizado na base de dados do sistema. Para cada *job* submetido, é criada uma pasta no sistema de arquivos para manter todos os arquivos (entrada, saída, *scripts*, etc.) associados com o *job* em questão. Após finalizar o processamento do *job*, os arquivos de saída são armazenados nesta pasta e mantidos imutáveis, usados apenas para operações de leitura e consulta, devido à necessidade de registro histórico das ações ocorridas no sistema, em particular, sobre as atividades dos usuários. É possível, ao longo de todas as etapas percorridas pelo *job* (submissão, execução e pós-execução), acompanhar seu andamento e visualizar informações detalhadas, que estão disponíveis em cada etapa, através das interfaces gráficas do sistema.

Durante o processamento dos *jobs* remotos, o sistema trata automaticamente novas requisições para execução, processando-as concorrentemente. Para cada requisição, o sistema abre uma conexão com o sistema HPC externo e envia o *job* para ser executado remotamente. Após a conclusão da execução do *job*, o sistema transfere os resultados (arquivos de saída gerados) do servidor remoto para a pasta local do *job*, localizada na máquina do servidor do sistema MODiMOL *Workbench*. Mais detalhes sobre o processamento dos *jobs* (locais e remotos) podem ser encontrados na seção 4.5.

Neste ponto, é importante salientar que o repositório de arquivos pessoais do usuário é mantido no sistema de arquivos da máquina servidora, na qual está sendo executado o sistema MODiMOL Workbench. Para cada usuário do sistema, é criada e associada uma respectiva pasta pessoal. Durante a execução de um *job* em particular, cada vez que um arquivo é selecionado pelo usuário para ser usado, este arquivo é copiado de uma pasta (pasta pessoal do usuário) para outra (pasta

associada ao *job*). Desta forma, os arquivos do repositório do usuário são mantidos intactos e em segurança.

### *4.2 Tecnologias utilizadas no desenvolvimento do sistema*

Em relação às tecnologias utilizadas no projeto, o sistema MODiMOL *Workbench* foi desenvolvido utilizando-se a linguagem de programação Java (https://www.oracle.com/java/), por ser esta uma tecnologia largamente empregada na construção de sistemas de informação baseados nos padrões da *Web*, além de possuir diversas bibliotecas e interfaces de programação (API – *Application Programming Interface*) disponíveis.

O ambiente de programação utilizado nesta atividade foi o Eclipse (http://www.eclipse.org/), que consiste em uma ferramenta gratuita, amplamente utilizada e que conta com inúmeros recursos disponíveis que tornaram o desenvolvimento do sistema extremamente produtivo. Outra grande vantagem no uso destas tecnologias de desenvolvimento está no fato de serem multiplataforma, ou seja, possuem distribuições para as principais plataformas de software atuais: Linux, Windows e iOS (Apple).

Os dois principais elementos que formam a base da plataforma desenvolvida neste trabalho são o Apache Tomcat (http://tomcat.apache.org/) e o MySQL *Server* (https://www.mysql.com/), usados como contêiner de aplicações Web (*Servlet Container*) e Sistema Gerenciador de Banco de Dados (SGBD), respectivamente. A ferramenta MySQL *Server* foi utilizada neste trabalho por ser um SGBD robusto, gratuito e amplamente utilizado tanto no meio comercial, quanto na comunidade acadêmica.

A Tabela 4.1 apresenta um resumo das tecnologias (bibliotecas, arcabouços de desenvolvimento e suas respectivas versões) utilizadas na construção do sistema MODiMOL *Workbench*. Vale salientar que várias destas tecnologias são consideradas "estado-da-arte" na área de desenvolvimento de sistemas, uma vez que, sendo tecnologias predominantemente utilizadas em seus domínios de aplicação, consequentemente tornaram-se padrões bem estabelecidos dentro da comunidade de desenvolvimento de software.

Tabela 4.1. Tecnologias usadas no desenvolvimento do sistema MODiMOL Workbench.

| Tecnologia | Versão | Descrição / Aplicação |
|---|---|---|
| **Apache Tomcat Server** | **7.0.47** | Servidor de aplicação (e servidor Web) usado para execução de sistemas desenvolvidos na linguagem Java. |
| **Apache Maven** | **3.3.1** | Ferramenta utilizada dentro do processo de desenvolvimento de sistemas para o gerenciamento de dependências (bibliotecas de programação) e geração de *builds* (arquivos executáveis). |
| **Cascading Style Sheets (CSS)** | **2.0** | Tecnologia usada na adequação do estilo (bordas, margens, cores, posicionamento, etc.) dos componentes de uma página JSF. |
| **Eclipse** | **4.3 Kepler** | Ambiente de programação usado no desenvolvimento de sistemas baseados na linguagem Java. |
| **Facelets (XHTML)** | **1.1** | Tecnologia baseada em XML e HTML, usada em conjunto com o JSF. |
| **Hibernate** | **3.6.9** | Arcabouço utilizado para simplificar o acesso de aplicações Java à bancos de dados relacionais. |
| **Java** | **7.0** | Linguagem de programação usada no desenvolvimento dos componentes do sistema. |
| **Java Server Faces (JSF)** | **2.2.6** | Arcabouço baseado na linguagem Java, utilizado para a criação das interfaces (páginas Web) do sistema. |
| **JAX-WS (Web Services)** | **2.0** | Tecnologia usada na comunicação entre sistemas, onde as informações são trafegadas no formato XML dentro do corpo das mensagens. |
| **JAX-RS (RESTful Web Services)** | **2.0** | Tecnologia semelhante ao JAX-WS, entretanto, as informações são trafegadas como parâmetros das |

| | | |
|---|---|---|
| | | mensagens. |
| **JSmol** | **14.2.2** | Aplicativo desenvolvido em JavaScript usado para visualização molecular de estruturas químicas. |
| **Log4j** | **1.2.16** | Tecnologia usada para gerar históricos (mensagens detalhadas) a partir da execução de aplicações Java. |
| **MySQL Server** | **5.6.24 Community Edition** | Sistema gerenciador de banco de dados. |
| **OpenBabel** | **2.3.1** | Aplicativo usado para conversão de formatos de arquivos contendo estruturas químicas. |
| **Primefaces** | **5.0** | Biblioteca de componentes JSF com suporte a Ajax, usada na criação de páginas Web dinâmicas. |

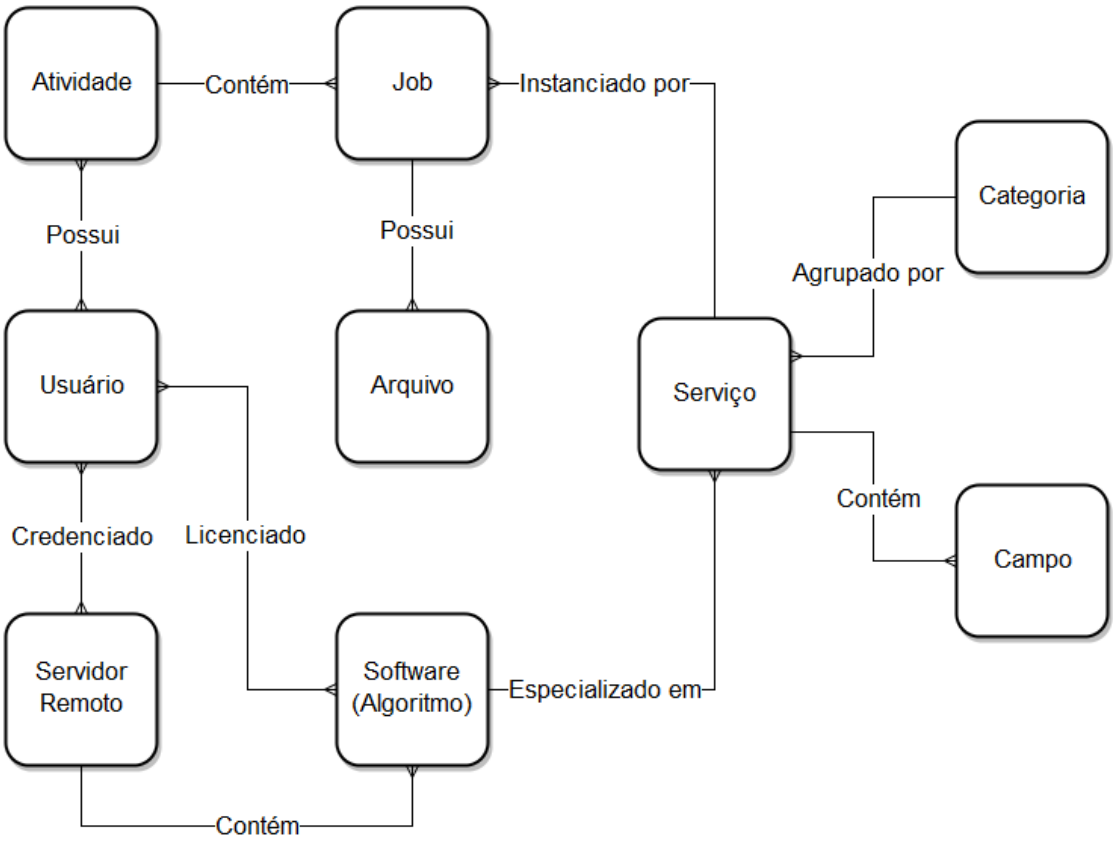
Fonte: O Autor, 2016.

A identidade visual das telas (tema) do sistema foi desenvolvida através da ferramenta *online* jQuery ThemeRoller (http://jqueryui.com/themeroller/) que permite a definição dos atributos visuais das interfaces construídas com os componentes da biblioteca PrimeFaces (baseados em *Java Server Faces* - JSF), como por exemplo, as cores, fontes e efeitos dos componentes gráficos.

### *4.3 Modelagem das informações armazenadas no banco de dados*

A Figura 4.4 apresenta o modelo conceitual do banco de dados utilizado no desenvolvimento do sistema MODiMOL *Workbench*. Como pode ser visto, um usuário é uma entidade que contém um conjunto de atividades e, onde cada atividade está associada a um ou mais *jobs*. Neste contexto, um *job* é uma instância de um serviço submetido para execução.

Figura 4.4. Modelo conceitual do banco de dados do sistema MODiMOL Workbench.




Fonte: O Autor, 2016.

Um serviço pode conter diversos campos e pode estar agrupado em categorias (com o intuito de organizar a apresentação na interface com o usuário). Como mencionado anteriormente, existem serviços locais e remotos. Um serviço é definido como remoto se um algoritmo (*software*) está localizado em um servidor remoto, caso contrário, como local, se este algoritmo encontra-se presente na máquina do servidor do sistema.

Conforme a Figura 4.4 apresentada anteriormente, o sistema deve armazenar e gerenciar as informações relativas a usuários, atividades, *jobs*, dentre outros. Os dados considerados relevantes para a modelagem de cada uma destas entidades, no desenvolvimento do sistema MODiMOL *Workbench*, são apresentados na Tabela 4.2.

Tabela 4.2. Detalhamento das informações para as entidades envolvidas no desenvolvimento do sistema MODiMOL Workbench.

| Entidade | Informações |
|---|---|
| **Arquivo** | • Identificador;<br>• Nome;<br>• Tipo (entrada ou saída); e,<br>• Tamanho (em *bytes*). |
| **Atividade** | • Identificador;<br>• Nome;<br>• Descrição;<br>• Data/hora de criação da atividade; e,<br>• Data/hora de encerramento da atividade. |
| **Campo** | • Identificador;<br>• Tipo (*text*, *integer*, *decimal*, *file*, *select* ou *check*);<br>• Argumento;<br>• Título;<br>• Valor; e,<br>• Dica. |
| **Categoria** | • Identificador; e,<br>• Nome. |
| **Job** | • Identificador;<br>• Linha de comando;<br>• Data/hora de criação do *job*;<br>• Data/hora de início da execução do *job*;<br>• Data/hora de fim da execução do *job*;<br>• Estado do *job* (pendente, em execução, finalizado); e,<br>• Resultado da execução do *job* (indefinido, sucesso, falha). |

| | |
|---|---|
| **Serviço** | • Identificador;<br>• Título;<br>• Descrição;<br>• Parâmetros;<br>• *Script*;<br>• Data/hora de criação do serviço;<br>• Data/hora de encerramento do serviço; |
| **Servidor Remoto** | • Identificador;<br>• Nome;<br>• Escalonador de recursos (SLURM);<br>• *Host*;<br>• Porta;<br>• Indicador de usuário “compartilhado”;<br>• Nome de usuário (para usuário “compartilhado”);<br>• Diretório de trabalho (para usuário “compartilhado”);<br>• Indicador de configuração de chave SSH;<br>• Data/hora de criação do servidor; e,<br>• Data/hora de encerramento do servidor. |
| **Software (Algoritmo)** | • Identificador;<br>• Nome;<br>• Descrição;<br>• Tipo (local ou remoto);<br>• Caminho do executável;<br>• Referências;<br>• Data/hora de criação do software; e,<br>• Data/hora de encerramento do software. |
| **Usuário** | • Identificador;<br>• Nome; |

| | |
|---|---|
| | • *Email*;<br>• Quota;<br>• Perfil (básico, intermediário e avançado);<br>• Informações de autenticação (nome de usuário e senha);<br>• Instituição;<br>• Coordenadas de localização (latitude e longitude);<br>• Data/hora de criação do usuário;<br>• Data/hora de inativação do usuário; e,<br>• *Link* para o currículo Lattes. |

Fonte: O Autor, 2016.

Para adequar o uso do banco de dados com o sistema MODiMOL *Workbench*, foi elaborado um modelo de dados relacional, mapeando-se as entidades supracitadas do sistema nos elementos necessários (tabelas, colunas, chaves primárias, relacionamentos e restrições), para a geração apropriada do banco de dados. Este modelo foi criado usando a ferramenta MySQL *Workbench* (http://dev.mysql.com/downloads/workbench). Esta ferramenta auxilia a tarefa de elaboração de modelos relacionais de sistemas, assim como na aplicação destes modelos em instâncias de bancos de dados.

Em seguida, o modelo relacional do sistema foi utilizado na criação de uma instância do banco de dados na ferramenta MySQL *Server*. No Apêndice C, é possível visualizar com maiores detalhes o modelo relacional do banco de dados do sistema MODiMOL Workbench, e o *script* SQL usado para a geração (fisicalização) das entidades e regras do banco de dados no MySQL *Server*.

### *4.4 Algoritmos e serviços*

Uma das maiores vantagens presentes na proposta deste trabalho consiste na sua habilidade de adicionar/editar serviços dinamicamente. Conforme discutido na seção anterior, algoritmos e serviços são armazenados como registros em uma base de dados. Portanto com o objetivo de atualizá-los, o administrador do sistema

precisa apenas acessar as telas de manutenção apropriadas, através da GUI, realizar as mudanças necessárias, e salvar as atualizações sem recompilar ou reinicializar o sistema, uma vez que, estas entidades não estão definidas diretamente no código-fonte do sistema ou estruturadas em arquivos de configuração (como no formato XML, por exemplo).

Algoritmos são cadastrados no sistema com configurações específicas. Um algoritmo local é criado com o caminho do arquivo executável (binário ou *script*), enquanto que o cadastramento de um algoritmo remoto demanda as informações de um servidor externo. Consequentemente, serviços também seguem a mesma ideia: para um dado algoritmo (local ou remoto) podem ser criados serviços (locais ou remotos, respectivamente).

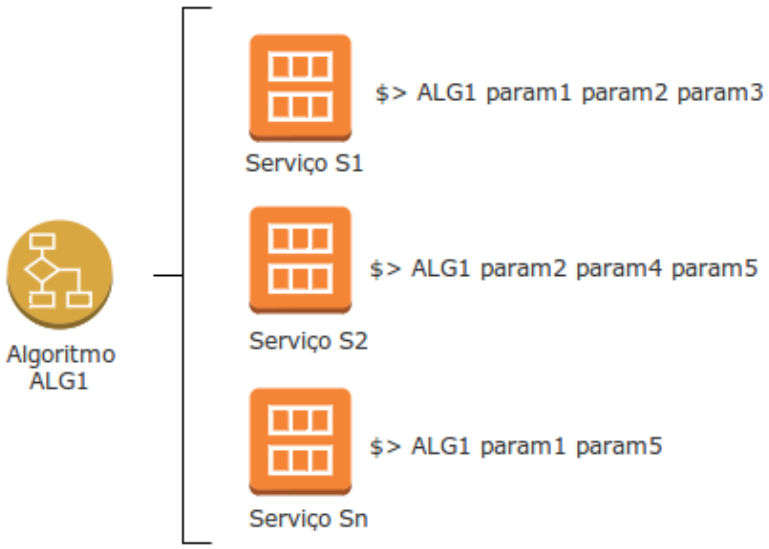
Serviços locais são definidos com os parâmetros de linha de comando utilizados (para o arquivo executável), enquanto que um serviço remoto é configurado com um *shell script* (que será executado remotamente). O conteúdo do *shell script* é editado com parâmetros que serão substituídos pelos valores de entrada preenchidos pelo usuário na interface, no momento da submissão do *job*. Este mesmo comportamento ocorre com os parâmetros de linha de comando, no caso dos serviços locais.

É possível que para um determinado algoritmo possam existir diversas formas de uso (aplicações práticas) associadas, de acordo com os parâmetros de entrada selecionados. Neste caso, existe a necessidade de que, para este dado algoritmo, sejam cadastrados diversos serviços no sistema, um para cada uso específico do algoritmo. Esta estratégia, no cenário em questão, evita o cadastramento de um mesmo algoritmo múltiplas vezes, sendo necessário apenas a criação de serviços específicos.

A Figura 17 exemplifica a situação discutida acima. Para um determinado algoritmo genérico ALG1, são cadastrados três serviços (S1, S2 e S3) que possuem parâmetros diferentes para suas respectivas linhas de comando.

Esta solução também pode ser aplicada para a situação de existirem versões diferentes para um mesmo algoritmo. Neste caso, todas as versões existentes podem coexistir normalmente no sistema, juntamente com seus respectivos serviços.

Figura 4.5. Exemplo de cadastro de vários serviços para um mesmo algoritmo.




Fonte: O Autor, 2016.

Dentro do sistema, os parâmetros de serviços são definidos como “campos” (*fields*). Durante a criação/atualização de um serviço, seus campos são definidos em relação a um tipo mapeado (ver Tabela 4.3), sendo: título (*label*), valor padrão (*default*), dica (*tooltip*), lista de títulos/valores (para o tipo SELECT) e lista de extensões de arquivos (para o tipo FILE).

Um campo é renderizado na interface do usuário com o respectivo componente gráfico (*editfield*, *combobox*, *checkbox*, etc.), de acordo com o tipo definido para o campo. A lista de extensões de arquivos é usada para filtrar os arquivos disponíveis no repositório de arquivos do usuário antes da montagem da lista final a ser exibida no formulário Web.

Tabela 4.3. Tipos de campos usados na criação dos serviços no sistema MODiMOL Workbench.

| Campo | Descrição |
|---|---|
| **TEXT** | Campo para valor textual simples |
| **INTEGER** | Campo para valor numérico inteiro |

| | |
|---|---|
| **DECIMAL** | Campo para valor numérico decimal (com casas decimais) |
| **FILE** | Campo para seleção de arquivo a partir do repositório pessoal de arquivos do usuário (*storage*) |
| **SELECT** | Campo para seleção de valor em uma lista de opções |
| **CHECK** | Campo para presença/ausência de parâmetro |

Fonte: O Autor, 2016.

Durante a criação dos serviços, é possível utilizar “parâmetros do sistema”, que representam variáveis de ambiente do sistema MODiMOL *Workbench* que serão substituídas nas linhas de comando, no caso dos serviços locais, ou nos *shell scripts,* no caso dos serviços remotos, antes da submissão dos *jobs* para execução. A Tabela 4.4 sumariza as variáveis de sistema existentes e suas respectivas descrições.

Tabela 4.4. Variáveis de sistema disponíveis no sistema MODiMOL Workbench.

| Nome | Descrição |
|---|---|
| **$MW_JOB_ID** | Corresponde ao identificador único de um determinado *job* dentro do sistema |
| **$MW_JOB_DIR** | Corresponde ao caminho da pasta local do *job* |
| **$MW_USER_NAME** | Corresponde ao nome do usuário |
| **$MW_USER_MAIL** | Corresponde ao e-mail do usuário |
| **$MW_USER_STORAGE** | Corresponde ao caminho da pasta do repositório pessoal do usuário |

Fonte: O Autor, 2016.

Para esclarecer melhor o conceito de parâmetros e variáveis de sistema durante o cadastramento de serviços no sistema MODiMOL *Workbench*, são apresentados nos Quadros 4.1 e 4.2 exemplos de substituição de parâmetros e variáveis de sistema para um serviço local e remoto, respectivamente. A substituição, conforme mencionado anteriormente, é realizada no momento que antecede a submissão do job para execução. Para ambos os exemplos, os parâmetros são denotados pelo padrão "$n", onde "n" é um número inteiro ($1, $2, etc.), enquanto que as variáveis de sistema, visualizadas apenas no exemplo do Quadro 4.2, são aquelas que fazem parte do conjunto apresentado na Tabela 4.4.

O Quadro 4.1 exemplifica especificamente a configuração de um serviço local no sistema MODiMOL *Workbench*, por meio da utilização do algoritmo AutoDock Vina (TROTT e OLSON, 2010), ferramenta usada em cálculos de *docking* molecular, bastante difundida na comunidade acadêmica. No exemplo, os parâmetros do serviço são substituídos por valores definidos pelo usuário na interface gráfica do sistema, para compor a linha de comando final que será usada para a execução do algoritmo.

O Quadro 4.2 apresenta um exemplo de configuração para um serviço remoto no sistema MODiMOL *Workbench*. Neste caso, um *shell script* é definido para a execução do algoritmo Gaussian 09 (FRISCH *et al.*, 2009) em um servidor remoto, cujo escalonador de recursos é o SLURM (JETTE *et al.*, 2003). O Gaussian é um programa de química quântica utilizado para a realização de cálculos de estrutura eletrônica, amplamente empregado na indústria e no meio acadêmico.

Quadro 4.1. Exemplo de substituição de parâmetros para uma linha de comando (serviço local).

| **Software (algoritmo)** | **Parâmetros** |
|---|---|
| `/opt/vina` | `--receptor $1 --ligand $2 --config $3` |
| **Linha de comando após substituições dos parâmetros** | |
| /opt/vina --receptor 1hwk-mod.pdbqt --ligand atorvastatin.pdbqt --config config.txt | |

Fonte: O Autor, 2016.

De forma análoga ao exemplo do Quadro 4.1, no *shell script* apresentado, os parâmetros também foram substituídos por valores definidos pelo usuário, e neste caso específico, as variáveis de sistema $MW_JOB_ID e $MW_USER_MAIL foram substituídas pelo identificador único do *job* no sistema MODiMOL *Workbench* e correio eletrônico (*e-mail*) do usuário, respectivamente.

Quadro 4.2. Exemplo de substituição de parâmetros e variáveis de sistema para um shell script (serviço remoto).

**_Shell script_ cadastrado no sistema (sem substituições)**

```
#!bin/bash
#SBATCH --job-name=job-$MW_JOB_ID
#SBATCH --exclusive
#SBATCH --nodes=$1
#SBATCH --partition=$2
#SBATCH --output=stdout.txt
#SBATCH --error=stdout.txt
#SBATCH --mail-type=ALL
#SBATCH --mail-user=$MW_USER_MAIL

INPUT=$3
OUTPUT=$4

module load softwares/gaussian09/d01
srun -v hostname -s | sort -u > tsnet.nodes.$SLURM_JOBID

g09 < $INPUT > $OUTPUT

rm tsnet.nodes.$SLURM_JOBID
```

**_Shell script_ após substituições dos parâmetros e variáveis de sistema**

```
#!bin/bash
#SBATCH --job-name=job-37
#SBATCH --exclusive
#SBATCH --nodes=2
#SBATCH --partition=long
#SBATCH --output=stdout.txt
```

```
#SBATCH --error=stdout.txt
#SBATCH --mail-type=ALL
#SBATCH --mail-user=john.doe@my-server.com

INPUT=h2.gjf
OUTPUT=h2.out

module load softwares/gaussian09/d01
srun -v hostname -s | sort -u > tsnet.nodes.$SLURM_JOBID

g09 < $INPUT > $OUTPUT

rm tsnet.nodes.$SLURM_JOBID
```

Fonte: O Autor, 2016.

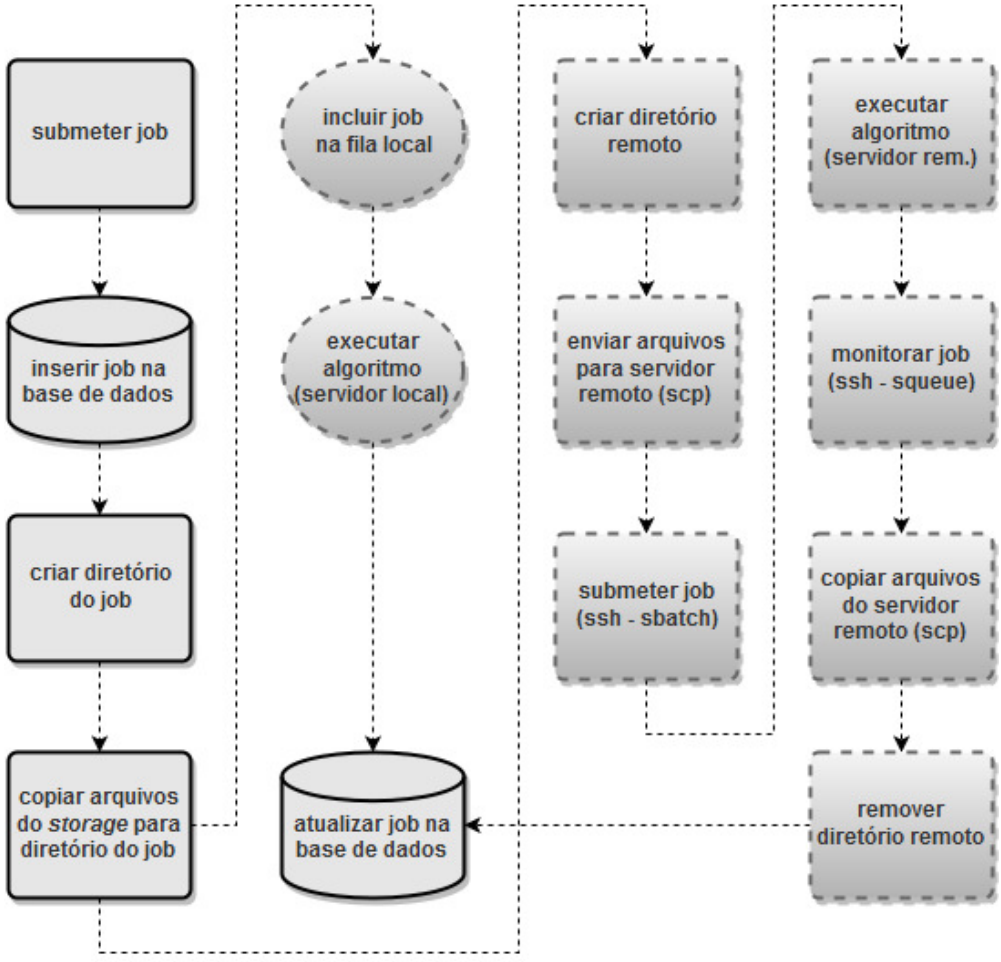
Adicionalmente, o sistema MODiMOL *Workbench* possibilita a edição do *shell script* de serviços remotos, no momento que imediatamente antecede a sua submissão para execução no servidor externo, e após as substituições dos parâmetros e variáveis de sistema. Este recurso pode ser bastante útil, pois permite que o usuário modifique o conteúdo do *script* para atender uma necessidade específica, que pode não ter sido contemplada originalmente pelo *script*, durante a criação do serviço pelo administrador do sistema.

### *4.5 Processamento dos jobs*

A Figura 4.6 apresenta um diagrama contendo o fluxo necessário para realizar a execução dos *jobs* submetidos (para ambos os serviços local e remoto) no sistema MODiMOL *Workbench*. Essencialmente, cada *job*, independente de sua origem, é iniciado com os mesmos passos básicos (representados por polígonos sólidos). O fluxo de execução toma diferentes caminhos para realizar passos específicos para *jobs* locais e remotos (representados por elipses e retângulos tracejados, respectivamente), finalizando em um mesmo passo comum (“atualizar *job* na base de dados”).

Após esperar por sua vez no "*pool*" correspondente, o *job* é enviado para execução pelo componente responsável por esta tarefa. Ele inicia o processamento, atualizando o banco de dados com informações deste *job* (estado: "em execução", data/hora do início da execução). A execução prossegue com a cópia dos arquivos de entrada necessários da pasta pessoal de arquivos do usuário (*storage*), para uma pasta de "trabalho" local criada para a execução do *job*, e que manterá permanentemente os arquivos (entrada e saída) para fins de consulta e visualização. A cópia dos arquivos é motivada, conforme já explicado anteriormente na seção 4.1, pela necessidade de se manter os arquivos da pasta pessoal do usuário inalterados e em segurança.

Figura 4.6. Fluxo de processamento dos jobs. Polígonos sólidos representam passos comuns a ambas execuções de jobs (local e remoto). Elipses tracejadas são passos presentes apenas em fluxos de execução local, enquanto que retângulos tracejados apenas em fluxos de execução remota.




Fonte: O Autor, 2016.

Neste ponto, um *job* pode ser executado local ou remotamente. O fluxo local considera a linha de comando (algoritmo + parâmetros) cadastrada no respectivo serviço, para ser executada na máquina onde está implantado o servidor do sistema MODiMOL *Workbench*. Em contrapartida, no fluxo remoto, torna-se necessária a realização de alguns passos precedentes, para tornar o sistema externo apto para a execução do *shell script* do serviço. Estes passos incluem a cópia dos arquivos de entrada do *job* para uma pasta no servidor remoto, criada especificamente para ser usada como pasta de "trabalho" no ambiente externo ao servidor do sistema MODiMOL *Workbench*.

Em seguida o *shell script* é submetido para execução no sistema remoto. Normalmente, sistemas de computação de alto desempenho são montados usando-se escalonadores de recursos, com a finalidade de gerenciar os seus recursos computacionais de forma mais eficiente. Em particular, o sistema MODiMOL *Workbench* suporta, por enquanto, o uso do SLURM (*Simple Linux Utility for Resource Management*) (JETTE *et al.*, 2003), um escalonador de recursos bem difundido e muito utilizando no gerenciamento de recursos de ambientes de computação de alto desempenho. Neste sentido, *jobs* remotos são submetidos e monitorados, neste ambientes, através de comandos do SLURM enviados remotamente.

Os principais comandos usados na versão atual do sistema MODiMOL Workbench são i) *sinfo*, que permite visualizar a lista das filas de execução (*partitions*). Isto é muito importante do ponto de vista do usuário, uma vez que é possível verificar quais filas (*partitions*) estão operacionais e disponíveis; ii) *sbatch*, que submete um *script batch* para execução. Um arquivo de *script* contém diretivas específicas que são interpretadas pelo SLURM; e, iii) *squeue*, que permite a visualização das informações sobre o estado ("*status*") de todos os jobs submetidos ao SLURM. É importante salientar que todas as operações de transferência de arquivos e execução remota de comandos do SLURM são realizadas através de conexões seguras (SCP e SSH, respectivamente), estabelecidas entre o sistema MODiMOL Workbench e o servidor externo em questão.

Quando o processamento do *job* é encerrado, suas informações (estado: "finalizado", resultado: "sucesso" ou "erro", data/hora da finalização da execução, nome dos arquivos de saída) são atualizadas no banco de dados. O encerramento

da execução do *job* produz, como resultados, arquivos contendo dados de saída do próprio *job*, assim como dados gerados a partir da saída padrão "*stdout*" e/ou da saída de erro padrão "*stderr*", uma vez que o *job* foi executado em modo de linha de comando, de maneira automática (não-iterativa), não necessitando, portanto, da intervenção do usuário em nenhum momento.

### *4.6 Perfis de usuários*

O sistema MODiMOL *Workbench* fornece meios para administrar o acesso aos serviços disponíveis, por exemplo, por meio de restrições específicas. O primeiro método é através da utilização de perfis de usuários. Quando um usuário é criado no sistema, associa-se um certo perfil que irá determinar quais serviços estarão acessíveis para este usuário. O sistema MODiMOL *Workbench* disponibiliza três tipos de perfis de usuário: Básico, Intermediário e Avançado.

#### *4.6.1 Perfil Básico*

O perfil 'Básico' permite que o usuário do sistema tenha acesso apenas a serviços locais, ou seja, durante a seleção de um serviço na interface do sistema, este usuário não terá visibilidade dos serviços remotos cadastrados no sistema. Um cenário que justifica a criação de um usuário com este tipo de perfil, consiste na situação onde o administrador do sistema necessita criar um usuário para uma pessoa acessar o sistema com a finalidade de navegar pelas interfaces e testar funcionalidades simples, incluindo submissão de *jobs* a partir de serviços locais que correspondam, por exemplo, à aplicações utilitárias ou algoritmos de baixa demanda computacional.

#### *4.6.2 Perfil Intermediário*

O perfil 'Intermediário' permite o acesso a serviços locais e remotos. Usuários com este tipo de perfil podem submeter *jobs* para serem executados em um ambiente remoto de HPC através de um usuário "compartilhado", que é

disponibilizado pelo servidor externo para ser usado como usuário de acesso a este servidor através de uma conexão segura, via SSH. Neste sentido, o usuário "compartilhado", é configurado dentro do sistema MODiMOL *Workbench* pelo administrador de forma que qualquer usuário com perfil 'Intermediário' possa utilizar os serviços remotos de uma infraestrutura de HPC, sem necessariamente possuir uma credencial (nome de usuário e senha) para acesso ao sistema externo em questão. Neste caso, um cenário plausível para o uso deste tipo de perfil, consiste na situação onde se deseja conceder o acesso aos serviços de um determinado servidor remoto, através de um usuário "temporário", de forma que este usuário possa submeter *jobs*, sem a necessidade da solicitação formal para criação de uma credencial (nome de usuário remoto) usada com o objetivo de se conectar ao sistema externo e acessar os seus recursos computacionais.

A princípio, um usuário "compartilhado" proveniente de um sistema externo não é um usuário tipo "*root*" ou com permissões especiais. Qualquer conta de usuário criada em um destes ambientes pode ser usada como usuário "compartilhado" dentro do sistema MODiMOL *Workbench,* para um determinado sistema externo. Vale ressaltar também que, não é primordialmente necessário que cada servidor externo cadastrado no sistema esteja configurado com um usuário "compartilhado", uma vez que o sistema fornece meios para uso de serviços remotos a partir de usuários pré-cadastrados (nome de usuário e senha) e cuja autenticação no sistema HPC externo, ocorre no momento da submissão do *job*, com a solicitação da senha na interface do sistema MODiMOL *Workbench*. Este cenário contendo submissão de *jobs* através de usuários que possuem credenciais remotas em servidores externos será discutido na seção 4.6.3

#### *4.6.2.1 Autenticação de chave pública (criptografia assimétrica)*

Para um usuário com Perfil Intermediário utilizar os serviços de um servidor externo, é necessário configurar a conta do usuário "compartilhado" dentro do sistema MODiMOL *Workbench*, de forma que as submissões de *jobs* provenientes destes serviços não exijam a entrada de credenciais (nome de usuário e senha) para a realização da operação, pelos usuários do sistema MODiMOL *Workbench*. Uma forma de atender esta necessidade consiste na autenticação do usuário

“compartilhado” através do mecanismo de criptografia assimétrica de chave pública. Neste método, um par de chaves criptográficas (pública e privada) é criado para realizar a autenticação de um determinado usuário (pertencente ao S.O. local) a um sistema remoto. Ao invés de utilizar sua senha, a chave pública do usuário é copiada para o sistema remoto, que a utiliza para autenticar e validar os comandos enviados, por este usuário, através de uma conexão SSH estabelecida entre as partes, neste caso, entre o servidor do sistema MODiMOL *Workbench* e o serviço remoto demandado.

O primeiro passo para a realização do mecanismo de autenticação por criptografia de chave pública consiste na geração do par de chaves SSH (pública e privada). Esta etapa é atendida pela execução do comando *ssh-keygen* (Quadro 4.3) no terminal do sistema operacional (S.O.), na máquina servidora do sistema MODiMOL *Workbench*.

Quadro 4.3. Uso do comando ssh-keygen para a criação do par de chaves SSH (pública e privada)

```
$ ssh-keygen -t rsa -b 4096
```

Fonte: O Autor, 2016.

O comando *ssh-keygen* gera o par de chaves através do algoritmo criptográfico RSA, e define o tamanho das chaves para 4096 bits. Vale mencionar que os parâmetros utilizados (algoritmo criptográfico e tamanho das chaves) podem ser definidos com outros valores, porém, para o sistema MODiMOL *Workbench*, foram usados os valores recomendados pela documentação oficial da ferramenta (http://www.openssh.com/manual.html).

O segundo passo consiste na cópia da chave pública para o servidor remoto que se deseja estabelecer a conexão. Este passo é realizado através do comando *ssh-copy-id* (Quadro 4.4), que possui como parâmetros, a porta de conexão SSH utilizada (“*port*”), o usuário remoto “compartilhado” (“*shared*”) e a máquina “*front-end*” (IP ou endereço qualificado) do servidor remoto (“*host*“):

Quadro 4.4. Uso do comando ssh-copy-id para a cópia da chave pública SSH para o servidor remoto.

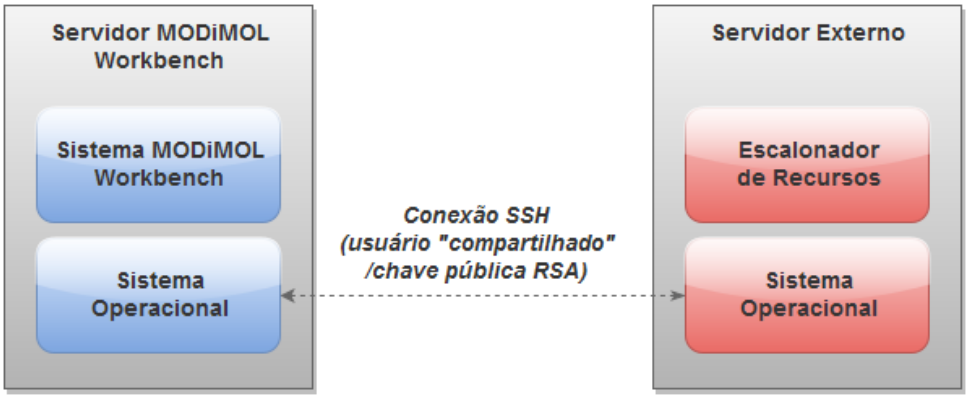
```
$ ssh-copy-id -p port shared@host
```

Fonte: O Autor, 2016.

Após a execução deste comando, é possível estabelecer conexões com o servidor remoto, através do usuário remoto (“*shared*”), sem a necessidade de informar sua senha, uma vez que a autenticação deste usuário no servidor remoto passa a ser realizada através da chave pública do usuário local (da máquina servidora do sistema MODiMOL *Workbench*). Desta forma, usuários do sistema MODiMOL *Workbench* com perfil Intermediário tornam-se aptos a acessar (submeter *jobs*) os ambientes remotos, sem a necessidade de possuir previamente usuários cadastrados nestes sistemas.

A Figura 4.7 apresenta a visão correspondente à comunicação realizada entre os sistemas (MODiMOL *Workbench* e servidor externo de HPC), com um usuário “compartilhado”, e cujo modo de autenticação utiliza criptografia de chave pública.

Figura 4.7. Comunicação entre o servidor MODiMOL Workbench e servidor (front-end) do ambiente de computação remoto, através de usuário “compartilhado” e autenticação por criptografia de chave pública.




Fonte: O Autor, 2016.

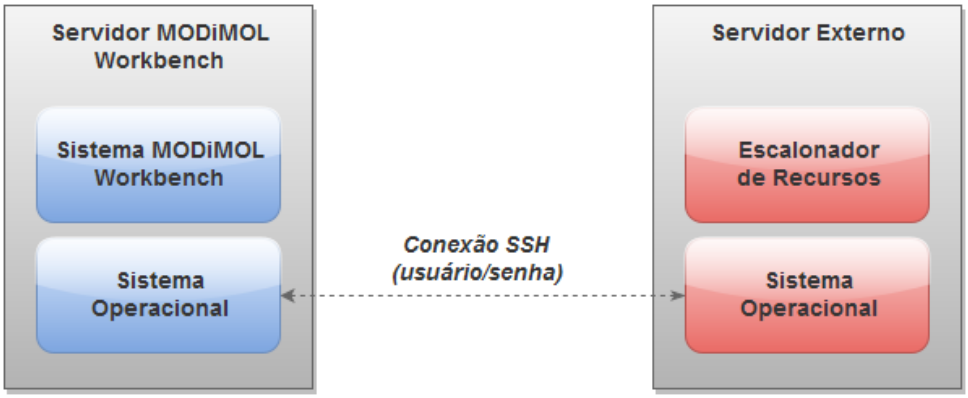
### *4.6.3 Perfil Avançado*

O perfil “Avançado”, assim como o perfil “Intermediário”, também permite o acesso a serviços locais e remotos. No entanto, diferentemente do perfil “Intermediário”, que realiza a autenticação a um servidor remoto através de um usuário “compartilhado” e chave pública SSH, o perfil “Avançado” requer uma

credencial remota previamente cadastrada no servidor a ser acessado. Desta forma, um usuário criado com este tipo de perfil terá acesso aos serviços disponibilizados em um determinado servidor externo, e usará sua própria credencial (nome de usuário e senha) para autenticação com aquele servidor, no momento da submissão de um *job*.

Vale ressaltar que esta mesma credencial, de forma independente ao sistema MODiMOL Workbench, pode ser usada a qualquer momento pelo usuário, para acessar diretamente os recursos do sistema externo, através de uma conexão segura (SSH) realizada através da linha de comando (terminal do Linux/Unix, ou *prompt* de comando no Windows). A Figura 4.8 apresenta o esquema de comunicação realizada entre os sistemas (MODiMOL *Workbench* e servidor externo de HPC), através de credencial própria (nome de usuário e senha), previamente cadastrada.

Figura 4.8. Comunicação entre o servidor MODiMOL Workbench e servidor (front-end) do ambiente de computação remoto, através de credencial (nome de usuário e senha) cadastrada.




Fonte: O Autor, 2016.

Mesmo para um usuário com perfil “Avançado”, serviços podem ser restritos por meio do conceito de licença. Uma licença, no contexto do sistema proposto, é uma permissão específica entre um algoritmo (*software* científico) e um usuário. Licenças são configuradas pelo administrador do sistema, que define quais softwares necessitam de licenças (reais) e quais usuários possuem autorização para acessá-los.

Esta estratégia permite que, para um determinado servidor remoto, possam coexistir serviços que exijam licença para seu uso, e outros que não possuam esta restrição. Neste sentido, serviços remotos, dentro do sistema MODiMOL *Workbench*,

podem ser ou não licenciados. A Tabela 4.5 sumariza as regras de permissão existentes entre os tipos de serviços e perfis de usuários.

Tabela 4.5. Regras de permissão para serviços e perfis de usuário.

| Serviço | Perfil Básico | Perfil Intermediário | Perfil Avançado |
|---|---|---|---|
| **Local** | X | X | X |
| **Remoto** | | X (*com usuário 'compartilhado' definido no servidor remoto*) | X (*com nome de usuário/senha para o servidor remoto*) |
| **Remoto (Licenciado)** | | | X (*com nome de usuário/senha para o servidor remoto e com licença para o algoritmo*) |

Fonte: O Autor, 2016.

## *4.7 Compartilhamento de informações*

O sistema MODiMOL *Workbench* permite que as atividades criadas por um usuário possam ser compartilhadas com outros usuários. Neste sentido, o compartilhamento de uma determinada atividade, resulta na condição em que todas as informações daquela respectiva atividade possam ser visualizadas pelo(s) usuário(s) selecionado(s) para o seu compartilhamento, e, além disso, esta atividade torna-se passível de ser atualizada, tanto em suas informações básicas (nome e descrição), quanto à submissão de novos *jobs*.

Para haver flexibilidade na forma de compartilhamento, a fim de se estabelecer restrições quanto à atualizações de uma atividade, foi incluído um atributo que pode ser usado para limitar o compartilhamento da atividade a apenas visualização (*read only*). Desta forma, um usuário proprietário de uma certa atividade

pode compartilhá-la com um conjunto de usuários, nos quais alguns possuem permissão de escrita (edição e submissão de *jobs*), enquanto outros não possuem esta prerrogativa, sendo possível apenas a visualização.

Um ponto importante a ser mencionado consiste no fato de que o compartilhamento (lista de usuários e suas respectivas permissões de escrita) de determinada atividade pode ser atualizado a qualquer momento pelo seu proprietário, trazendo com isto maior flexibilidade na manutenção e definição de regras de acesso a atividades e suas informações.

### *4.8 Ativação e desativação de entidades*

O sistema MODiMOL *Workbench* disponibiliza uma funcionalidade para ativar e desativar determinadas entidades do sistema. Este recurso é útil, por exemplo, num cenário onde se deseja que um serviço ou servidor remoto permaneça temporariamente incessível, para o administrador realizar alguma tarefa de manutenção em alguma destas entidades.

Esta funcionalidade também pode ser usada diretamente por um usuário do sistema, no caso onde se deseja encerrar as ações sobre uma determinada atividade, por “inatividade” ou por considerá-la finalizada. A Tabela 4.6 apresenta as entidades que podem ser ativadas/desativadas no sistema, atores e reflexos de cada ação.

Tabela 4.1. Regras de ativação/desativação de entidades do sistema.

| Entidade | Ator | Regras (Ativação/Desativação) |
|---|---|---|
| **Atividade** | Usuário | Habilita/Desabilita a atualização de informações, compartilhamento e submissão de *jobs* |
| **Categoria** | Administrador | Habilita/Desabilita a categoria e serviços associados |
| **Serviço** | Administrador | Habilita/Desabilita o serviço |
| **Servidor Remoto** | Administrador | Habilita/Desabilita o servidor remoto, softwares e serviços associados |

| | | |
|---|---|---|
| **Software (Algoritmo)** | Administrador | Habilita/Desabilita o software e serviços associados |
| **Usuário** | Administrador | Habilita/Desabilita o acesso do usuário ao sistema |

Fonte: O Autor, 2016.

### *4.9 Integração com bancos de dados de estruturas moleculares*

Os algoritmos de modelagem molecular normalmente processam dados de entrada, que na maioria das vezes, são arquivos formatados contendo uma ou mais estruturas moleculares. Basicamente, um usuário pode obter estes tipos de dados através da transferência dos arquivos das moléculas da sua máquina pessoal para a máquina servidora do sistema (*upload*), ou digitando seu conteúdo diretamente na interface gráfica (em um editor de texto ASCII).

Além da opção do "*upload*" de arquivos, o sistema MODiMOL *Workbench* disponibiliza a inclusão de novas moléculas através do "*download*" de suas estruturas, diretamente de alguns bancos de dados públicos, onde cada estrutura é identificada através de um código único que será informado pelo usuário. Os bancos de dados utilizados na presente versão do sistema são: CACTUS (CACTUS, 2016), ChemSpider (CHEMSPIDER, 2016), PDB (PDB, 2016), PubChem (PUBCHEM, 2016) e ZINC (ZINC, 2016).

A recuperação da estrutura molecular ("*download*") é realizada por meio da integração entre o sistema MODiMOL *Workbench* e o sistema externo correspondente ao banco de dados utilizado. Esta integração é alcançada através de duas tecnologias de comunicação empregadas na disponibilização dos serviços destes bancos de dados: *Web Services* (JAX-WS, 2014) e RESTful *Web Services* (JAX-RS, 2014). Cada uma delas utiliza uma estratégia diferente para realizar a comunicação entre os sistemas.

O primeiro tipo requer um componente de software adicional para auxiliar na comunicação entre as partes (sistemas cliente e servidor do serviço se comunicam através de mensagens trocadas em formato XML), enquanto que o segundo tipo é completamente estruturado por meio de URLs, ou seja, através do envio de simples requisições ao servidor do serviço usando o protocolo de comunicação HTTP, e

cujas respostas correspondem às estruturas das moléculas requisitadas, em formato apropriado. Independentemente da tecnologia, a integração ocorre através do envio de um código que identifica unicamente uma estrutura molecular específica, e do retorno da estrutura selecionada de volta para o sistema, no formato de dados escolhido pelo usuário no momento da requisição.

Os formatos de dados SMILES (*Simplified Molecular-Input Line-Entry System*) (WEININGER, 1988) e InchI (IUPAC *Chemical Identifier*) (HELLER *et al*., 2013) são dois tipos de identificadores textuais empregados na codificação de estruturas químicas. Através destes, é possível representar qualquer entidade química através de uma sequência de caracteres, e cuja principal aplicação consiste na identificação e busca por informações em bancos de dados desta natureza. O formato InchI, em particular, possui também uma versão condensada de 27 caracteres, denominada InchIKey (gerada a partir de um código InchI), usada com o objetivo de facilitar a busca de moléculas em bancos de dados.

A Tabela 4.7 apresenta os bancos de dados externos de estruturas moleculares utilizados na integração com o sistema MODiMOL *Workbench*, assim como os tipos de formato de dados usados na entrada e na saída dos serviços fornecidos por estes sistemas. Os formatos de dados (entrada e saída) apresentados na Tabela 4.7 não refletem exatamente todos os formatos disponibilizados nestes sistemas, mas apenas aqueles selecionados para a integração específica com o sistema MODiMOL *Workbench*.

Tabela 4.2. Bancos de dados externos de estruturas moleculares usados na integração com o sistema MODiMOL Workbench.

| **Serviço** | **Tecnologia de Comunicação** | **Códigos de Identificação e Formatos de Dados** | |
|---|---|---|---|
| | | **Entrada** | **Saída** |
| **CACTUS** | JAX-RS | • InchI<br>• InchIKey<br>• SMILES | • Protein Data Bank (PDB)<br>• Symyx molecule file (MOL)<br>• Symyx Structure Data Format |

| | | | |
|---|---|---|---|
| | | | (SDF)<br>• Tripos Sybyl MOL2 format (MOL2) |
| **ChemSpider** | JAX-WS | • InchI<br>• InchIKey | • Symyx molecule file (MOL) |
| **Protein Data Bank (PDB)** | JAX-RS | • Código PDB | • Protein Data Bank (PDB) |
| **PubChem** | JAX-RS | • InchI<br>• InchIKey<br>• SMILES | • Symyx Structure Data Format (SDF) |
| **ZINC** | JAX-RS | • Código ZINC<br>• SMILES | • Symyx Structure Data Format (SDF)<br>• Tripos Sybyl MOL2 format (MOL2) |

Fonte: O Autor, 2016.

A Tabela 4.8 apresenta os bancos de dados de estruturas moleculares cujos serviços (*Web Services*) são baseados na tecnologia RESTful (JAX-RS) e suas respectivas URLs básicas, usadas na integração destes serviços externos com o sistema MODiMOL Workbench.

Tabela 4.3. Bancos de dados de estruturas moleculares e serviços (Web Services RESTful) correspondentes. Elementos em negrito correspondem aos parâmetros fornecidos pelo usuário na interface no momento da requisição do serviço. id = identificador da estrutura molecular; in = formato de entrada; out = formato de saída.

| Serviço | URL |
|---|---|
| **CACTUS** | http://cactus.nci.nih.gov/chemical/structure/**id**/file?format=**out** |
| **PDB** | http://www.rcsb.org/pdb/files/**id**.pdb |
| **PubChem** | http://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/**in**/**id**/**out** |

| | |
|---|---|
| **ZINC** | http://zinc.docking.org/results?in=**id**&page.format=**out** |

Fonte: O Autor, 2016.

### *4.10 Segurança*

Conforme já mencionado na seção 4.1, todas as mensagens trafegadas na comunicação entre os navegadores Web (clientes) e o servidor Web do sistema MODiMOL *Workbench* (Apache Tomcat), utilizam o protocolo de rede HTTPS (SSL/TLS sobre o HTTP) para realizar comunicação segura entre as partes. Para estabelecer um canal seguro em suas conexões usando SSL, um servidor Web deve possuir um "Certificado Digital" associado para cada uma das interfaces de rede que aceitam conexões seguras.

Um Certificado Digital é usado para encriptar/decriptar as mensagens trafegadas pela rede, além de poder ser usada no mecanismo de autenticação (identificação) de serviços na Web (por exemplo, a identificação de um sítio de comércio eletrônico). Quando a identificação de um serviço é importante, o Certificado Digital é "assinado" e emitido por uma Autoridade Certificadora (*Certificate Authority*), que garante a autenticidade do Certificado Digital, e consequentemente a identidade do serviço na Internet. No caso do sistema MODiMOL *Workbench*, o Certificado Digital é usado apenas para codificação das mensagens trafegadas na rede, não necessitando, portanto, no momento de sua criação, da "assinatura" de uma Autoridade Certificadora. Neste caso, o Certificado Digital criado é denominado "auto-assinado" (*self-signed*).

O procedimento de configuração para utilização do SSL no servidor Apache Tomcat é dividido em duas etapas básicas: i) Geração do Certificado Digital; e ii) Atualização dos arquivos de configuração do servidor Apache Tomcat. O Quadro 4.5 exibe a utilização da ferramenta "*keytool*" para a geração do Certificado Digital. A variável de ambiente "$JAVA_HOME" corresponde ao caminho da instalação do pacote Java na máquina em questão. Após a execução deste comando, um arquivo contendo o Certificado Digital é criado no sistema de arquivos da máquina do servidor Apache Tomcat.

Quadro 4.5. Uso do comando keytool para a geração da chave criptográfica a ser utilizada no servidor Apache Tomcat.

```
$ JAVA_HOME/bin/keytool -genkey -alias mworkbench -keyalg RSA
```

Fonte: O Autor, 2016.

A segunda etapa consiste na configuração do servidor Web Apache Tomcat para aceitar conexões com SSL, usando o Certificado Digital criado para criptografar as mensagens na Internet. Esta tarefa é realizada através da utilização de algumas diretivas em basicamente dois arquivos de configuração, sendo um deles pertencente à instalação do Apache Tomcat (*server.xml*), e outro, ao conjunto de arquivos de desenvolvimento do próprio sistema MODiMOL Workbench (*web.xml*).

O Quadro 4.6 apresenta o trecho acrescentado ao arquivo *server.xml* com a diretiva necessária para o servidor Apache Tomcat começar a aceitar conexões seguras usando SSL (*SSLEnabled="true"*), usando o Certificado Digital previamente criado (*keystoreFile="/home/user/.keystore"*). Neste ponto, o servidor passa a aceitar requisições de recursos através de conexões seguras usando o protocolo HTTPS (*scheme="https"*) na porta 8443 (*port="8443"*).

Quadro 4.6. Configuração adicionada ao arquivo server.xml do servidor Apache Tomcat para habilitar conexão segura via HTTPS.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
  maxThreads="150" scheme="https" secure="true"
  clientAuth="false" sslProtocol="TLS"
  keystoreFile="/home/user/.keystore"
  keystorePass="password"
/>
```

Fonte: O Autor, 2016.

O Quadro 4.7 exibe a diretiva adicionada ao arquivo de configuração *web.xml* do sistema, necessária para o uso do protocolo HTTPS pela aplicação. Especificamente, o sistema utiliza conexões seguras para a transferência das páginas Web (em formato HTML) (*transport-guarantee="CONFIDENTIAL"*), enquanto que, para outros recursos que não necessitam ser criptografados (por

exemplo, imagens, arquivos contendo código-fonte em JavaScript e CSS), pode utilizar o protocolo padrão HTTP sem segurança (*transport-guarantee="NONE"*).

Quadro 4.7. Configuração adicionada ao arquivo web.xml do sistema MODiMOL Workbench para habilitar o uso do protocolo HTTPS para a requisição de recursos (interfaces) em conexão segura.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>HTTPSOnly</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
    <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<security-constraint>
  <web-resource-collection>
    <web-resource-name>HTTPSOrHTTP</web-resource-name>
    <url-pattern>*.png</url-pattern>
    <url-pattern>*.gif</url-pattern>
    <url-pattern>*.css</url-pattern>
  </web-resource-collection>
    <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

Fonte: O Autor, 2016.

Outro ponto importante a ser considerado, no que diz respeito à segurança, consiste no armazenamento criptografado das senhas dos usuários do sistema. Com este objetivo, foi utilizado o SHA-256 (SHA-256, 2016), um algoritmo de criptografia largamente empregado para codificação de dados em geral. Basicamente, existem duas situações típicas que exigem a criptografia da senha do usuário, para seu devido armazenamento de forma segura no banco de dados. O primeiro cenário consiste na inclusão/alteração de um usuário pelo administrador do

sistema, e o segundo, na funcionalidade específica que permite a alteração da senha do usuário (quando este já se encontra autenticado no sistema).

Em ambos os cenários de uso, a senha que é manipulada na interface gráfica do sistema é previamente criptografada antes de ser armazenada. O Quadro 4.8 apresenta um trecho do código-fonte em linguagem de programação Java, que usa, durante a inclusão de um novo usuário no sistema, o algoritmo de criptografia SHA-256 para codificar sua senha antes do armazenamento no banco de dados do sistema.

Quadro 4.8. Trecho de código-fonte usado para criptografar a senha do usuário durante sua inclusão no banco de dados.

```
public class UserBC extends AbstractBC<User, Integer>{

  ...

  public Integer insert(User user, String typedPassword) {

    MessageDigest md = MessageDigest.getInstance("SHA-256");
    String hashedPassword = md.update(typedPassword.getBytes());

    user.setPassword(hashedPassword));

    this.update(user);

    ...
  }
}
```

Fonte: O Autor, 2016.

### *4.11 Versão para dispositivos móveis*

Como mencionado no capítulo de introdução deste trabalho, o sistema MODiMOL *Workbench* também fornece o acesso às suas funcionalidades, através de dispositivos móveis. Neste contexto, é importante discutir os possíveis caminhos para atingir este objetivo. As estratégias mais comuns incluem o desenvolvimento de

uma aplicação nativa (*app*), ou a adaptação da interface Web para aqueles tipos de dispositivos (telas responsivas), a fim de também serem acessíveis por um navegador Web (*browser*).

A primeira opção requer, normalmente, mais tempo para ser desenvolvido e mais conhecimento técnico acerca das especificidades dos dispositivos (múltiplos sistemas operacionais e suas versões, bibliotecas de desenvolvimento disponíveis, etc.). Por outro lado, a segunda opção provê uma alternativa mais rápida e menos custosa do que a primeira, por envolver apenas a adaptação das interfaces do usuário para dispositivos móveis. Atualmente, diversas bibliotecas de programação com esta finalidade dão suporte para criar interfaces responsivas sem um maior esforço por parte do desenvolvedor. Nesta opção, entretanto, ainda é requerida a presença de um navegador Web (aplicação nativa) no dispositivo móvel - o que é muito usual - para acessar as interfaces responsivas do sistema.

A estratégia deste trabalho em prover acesso ao sistema MODiMOL *Workbench,* através de dispositivos móveis ("*mobile*")*,* foi através da segunda opção mencionada anteriormente: adaptação das interfaces de usuário para suportar automaticamente as dimensões das telas dos dispositivos móveis (interfaces responsivas). Este resultado foi alcançado através do reconhecimento do tipo de cliente Web (*desktop* ou dispositivo móvel) pelo sistema MODiMOL *Workbench*, e apresentação apropriada das telas responsivas (para clientes móveis). A identificação é realizada por meio do cabeçalho "User-Agent" da mensagem de requisição do protocolo HTTP. O cliente Web é reconhecido pelo valor do cabeçalho, o qual é usado para decidir a interface de usuário adequada.

O Quadro 4.9 apresenta um trecho do código-fonte, em linguagem de programação Java, para o filtro (*Servlet Filter*) criado para detectar o tipo de dispositivo utilizado para requisitar um recurso (página em HTML) no servidor do sistema MODiMOL *Workbench*. Como pode ser observado no código-fonte, o sistema é capaz de redirecionar a requisição original para uma interface responsiva correspondente, quando o tipo do cliente Web, detectado a partir do cabeçalho "User-Agent" da requisição HTTP (representada pela classe HTTPServletRequest), for um dispositivo móvel.

Quadro 4.9. Trecho de código-fonte usado para identificação do cliente Web a partir do cabeçalho "User-Agent" do protocolo de comunicação HTTP

```
public class MobileDetectorFilter implements Filter {

  ...

  public void doFilter(ServletRequest request, ServletResponse response,
            FilterChain chain) throws IOException, ServletException {

    HttpServletRequest req = (HttpServletRequest) request;

    UserAgent ua = UserAgent.parseUserAgentString(
                              req.getHeader("User-Agent"));

    if(ua.getOperatingSystem().isMobileDevice()) {

        //redireciona para a interface responsiva correspondente
        ...
    } else {

        //retorna a interface requisitada
        ...
    }
  }
}
```

Fonte: O Autor, 2016.

As interfaces de usuário responsivas do sistema MODiMOL *Workbench* foram desenvolvidas através da biblioteca de componentes PrimeFaces Mobile (http://www.primefaces.org/showcase/mobile/index.xhtml), um kit de componentes de interface com o usuário baseado no *Java Server Faces* (JSF), e implementado sobre o jQuery Mobile (http://jquerymobile.com/), um arcabouço de programação para a Web, compatível com um grande número de dispositivos móveis. A identidade visual das telas (tema), para versão móvel do sistema, foi desenvolvida através da ferramenta online jQuery Mobile ThemeRoller (http://themeroller. jquerymobile.com/), que permite a definição personalizada das cores, fontes e efeitos do tema criado para as interfaces do sistema.

É importante ressaltar que apenas parte das funcionalidades presentes no sistema foram contempladas com o desenvolvimento de interfaces responsivas para os dispositivos móveis. O escopo de implementação destas interfaces foi limitado a algumas funcionalidades direcionadas para o gerenciamento de arquivos pessoais do usuário, manipulação de atividades e submissão de *jobs*, não abrangendo o desenvolvimento das interfaces para funcionalidades administrativas do sistema, tais

como, gerenciamento de entidades básicas (usuários, algoritmos e serviços), configuração de servidores remotos, dentre outras.

# 5 RESULTADOS E DISCUSSÃO

## *5.1 Funcionalidades do sistema*

As funcionalidades presentes no sistema MODiMOL *Workbench* foram desenvolvidas levando em consideração a necessidade de gerenciamento de diversas informações (conforme descrito no capítulo de metodologia), que podem ser ou não de natureza administrativa, ou seja, manipuladas apenas pelo usuário administrador do sistema. Neste contexto, a Tabela 5.1 apresenta o conjunto de funcionalidades providas pelo sistema, indicando se cada uma delas possuem: i) natureza administrativa, e ii) interfaces responsivas também desenvolvidas (para dispositivos móveis).

Tabela 5.1. Funcionalidades do sistema MODiMOL Workbench.

| Funcionalidade | Administrativa? | Interface responsiva? |
|---|---|---|
| **Gerenciamento de Categorias** | X | |
| **Gerenciamento de Servidores Remotos** | X | |
| **Gerenciamento de Softwares (Algoritmos)** | X | |
| **Gerenciamento de Serviços** | X | |
| **Gerenciamento de Usuários** | X | |
| **Gerenciamento de Atividades** | | X |
| **Gerenciamento de Arquivos do Usuário** | | X |
| **Seleção de Serviços e Submissão de *Jobs*** | | X |

Fonte: O Autor, 2016.

Ainda em relação à Tabela 5.1, independente da natureza da funcionalidade do sistema, todas são acessíveis a partir de um navegador Web (*browser*). Se este acesso for realizado a partir de um dispositivo móvel, as funcionalidades marcadas na coluna “Interface responsiva?” respondem com interfaces de usuário responsivas, ou seja, que se adequam automaticamente às dimensões das telas dos dispositivos utilizados.

Entretanto, quando funcionalidades sem interfaces responsivas são acessadas por meio de um dispositivo móvel, elas são apresentadas de forma reduzida (menor *zoom*). Isto por que estas interfaces foram desenvolvidas apenas para navegadores disponíveis em computadores pessoais (*desktops* e *laptops*), onde a resolução da tela é relativamente maior que nos dispositivos móveis. Ao manuseá-las, o usuário necessita constantemente ampliar e direcionar o conteúdo visível, o que torna seu uso bastante trabalhoso e repetitivo.

Figura 5.1. Exemplo de interface não-responsiva acessada através de um navegador Web de dispositivo móvel.

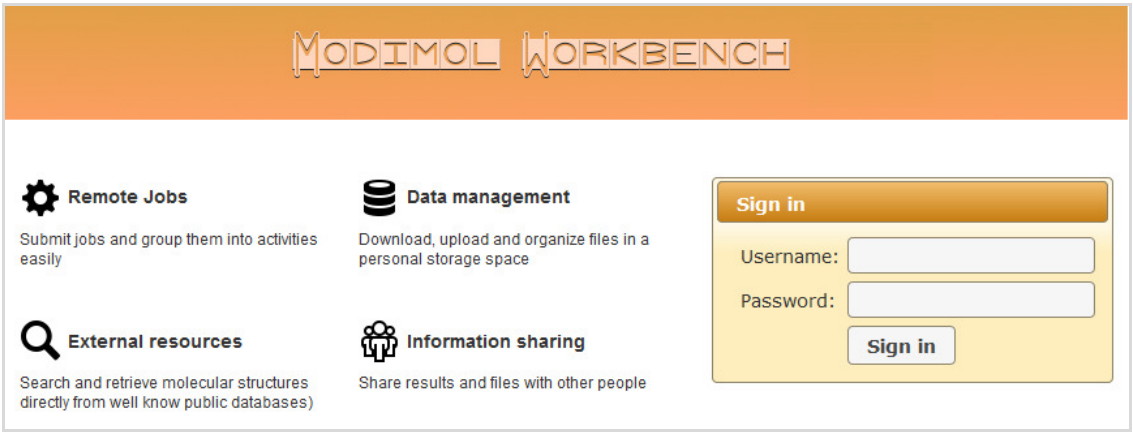

Fonte: O Autor, 2016.

Esta dificuldade, por exemplo, justifica bem a necessidade de desenvolvimento de interfaces responsivas para acessar todas as funcionalidades presentes no sistema. A Figura 5.1 exemplifica este cenário, apresentando uma interface não-responsiva, do sistema MODiMOL *Workbench*, acessada por meio de um navegador Web de um dispositivo móvel.

Após a autenticação do usuário no sistema MODiMOL *Workbench*, por meio da interface apresentada na Figura 5.2, será exibida a tela principal do sistema, que expõe para o usuário, todas as funcionalidades dispostas em painéis organizados por temas, por exemplo, funcionalidades referentes ao gerenciamento de arquivos no painel deste tema, ou aquelas referentes ao gerenciamento de atividades e submissão de *jobs*, neste tema específico.

As Figuras 5.3 e 5.4 apresentam as interfaces principais do sistema, sob a visão de um usuário administrador, e de um usuário comum do sistema (independente de seu perfil), respectivamente. A diferença básica entre as interfaces apresentadas consiste na visibilidade das funcionalidades administrativas (listadas na Tabela 9) que estão acessíveis apenas para o usuário administrador do sistema.

Figura 5.2. Interface de autenticação (login) do sistema MODiMOL Workbench.

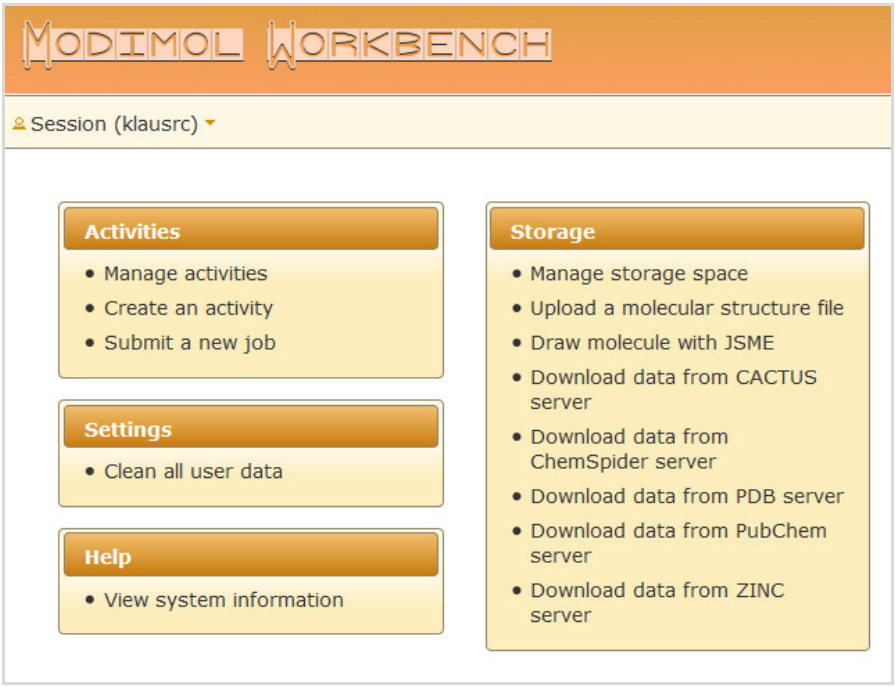



Fonte: O Autor, 2016.

Figura 5.3. Interface principal do sistema MODiMOL Workbench (Visão do administrador).


Fonte: O Autor, 2016.

Figura 5.4. Interface principal do sistema MODiMOL Workbench (Visão do usuário).




Fonte: O Autor, 2016.

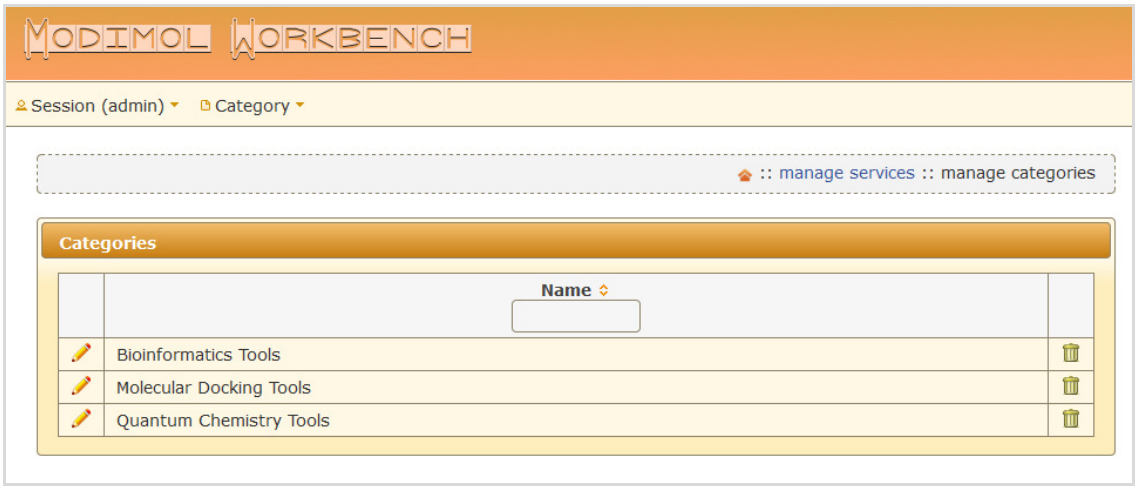
As próximas seções apresentam e discutem os resultados obtidos com o desenvolvimento das funcionalidades providas pelo sistema MODiMOL *Workbench* (Tabela 5.1) por meio da demonstração de suas interfaces de usuário (telas do sistema). Para as funcionalidades relacionadas ao gerenciamento de entidades, cada descrição contemplará, no mínimo, a presença de duas interfaces: a listagem dos registros cadastrados e o formulário para a edição da entidade.

### *5.1.1 Gerenciamento de categorias*

Uma categoria é uma entidade do sistema que é utilizada para o agrupamento de serviços comuns, ou seja, que pertençam a mesma natureza de aplicação. Neste caso, categorias são criadas no sistema para auxiliar na organização (classificação) dos serviços disponíveis pelo sistema MODiMOL *Workbench*. Esta organização pode ser melhor compreendida na seção 5.1.8, onde será apresentada a interface de usuário na qual um serviço é selecionado para posterior execução.

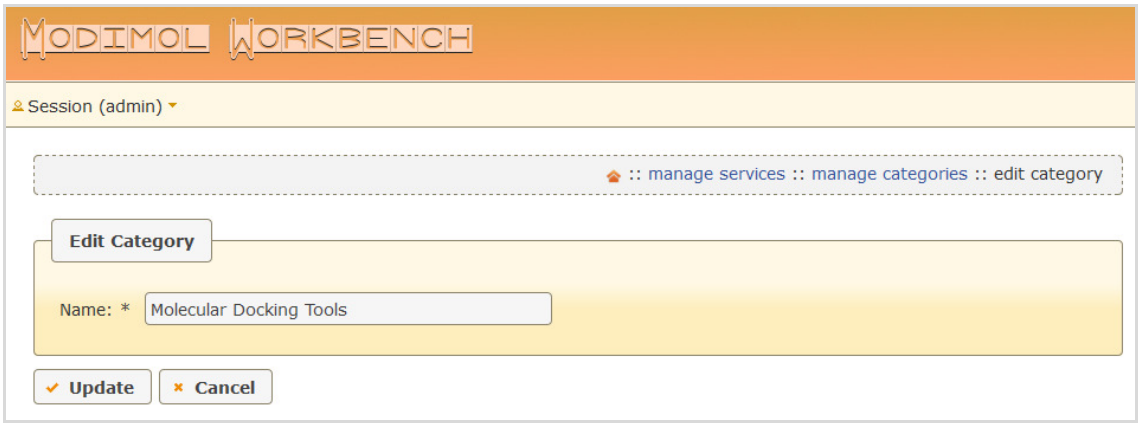
A Figura 5.5 apresenta a interface contendo a listagem de categorias cadastradas no sistema MODiMOL *Workbench*. Nesta interface, é possível remover uma categoria já existente, ou criar uma nova, a partir da opção presente no menu "*Category*". Por outro lado, a Figura 5.6 demonstra a interface utilizada para a edição de uma categoria (inclusão ou alteração). Neste ponto, a única informação a ser editada em uma categoria consiste no seu próprio nome.

Figura 5.5. Interface para listagem de categorias.




Fonte: O Autor, 2016.

Figura 5.6. Interface para edição de uma categoria.




Fonte: O Autor, 2016.

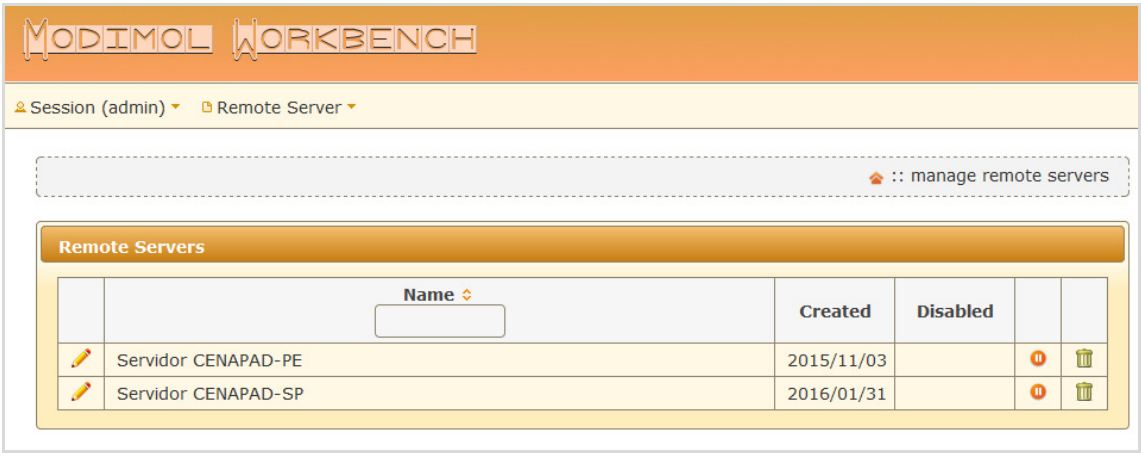
### *5.1.2 Gerenciamento de servidores remotos*

Para se utilizar serviços que são executados externamente em ambientes de computação de alto desempenho, é necessário que os servidores remotos usados estejam previamente cadastrados no sistema MODiMOL *Workbench*. Estas informações permitem que o sistema possa ter acesso a estes servidores externos para realizar a submissão dos *jobs*, de acordo com o mecanismo de execução de *jobs* descrito na seção 4.5.

A Figura 5.7 apresenta a interface correspondente à listagem dos servidores remotos cadastrados. Para esta entidade, além das operações básicas de inclusão, alteração e exclusão, também é possível realizar sua ativação/desativação. Neste caso, conforme discutido na seção 28, ao desativar um servidor remoto, também serão desativados todos os *softwares* (algoritmos) e serviços a ele associados. De forma análoga funciona o processo inverso: ao ativar um servidor remoto, também serão ativados todos seus *softwares* e serviços.

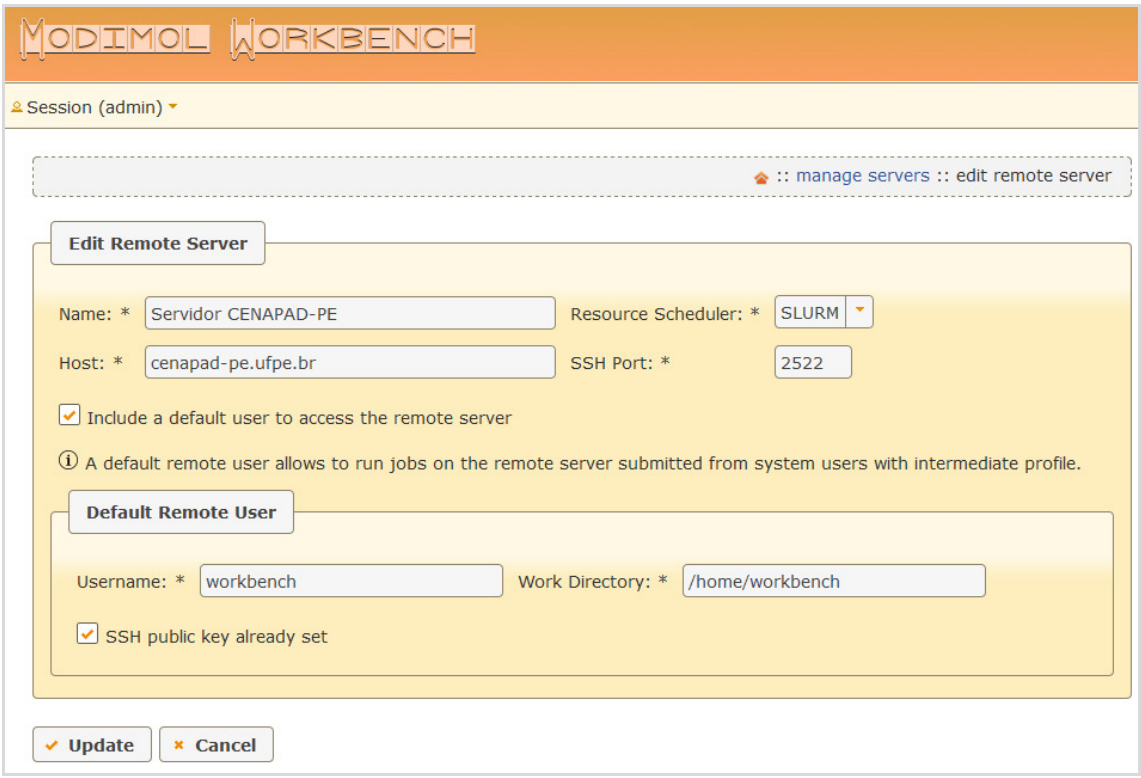
A Figura 5.8 exibe a interface de edição para um servidor remoto. Além do nome do servidor, também são incluídas algumas informações importantes, como por exemplo, o tipo de escalonador de recursos (*Resource Scheduler*), o endereço da máquina "*front-end*" (*Host*) do ambiente remoto e porta (*SSH Port*) para conexão SSH utilizada.

Figura 5.7. Interface para listagem de servidores remotos.




Fonte: O Autor, 2016.

Figura 5.8. Interface para edição de um servidor remoto.




Fonte: O Autor, 2016.

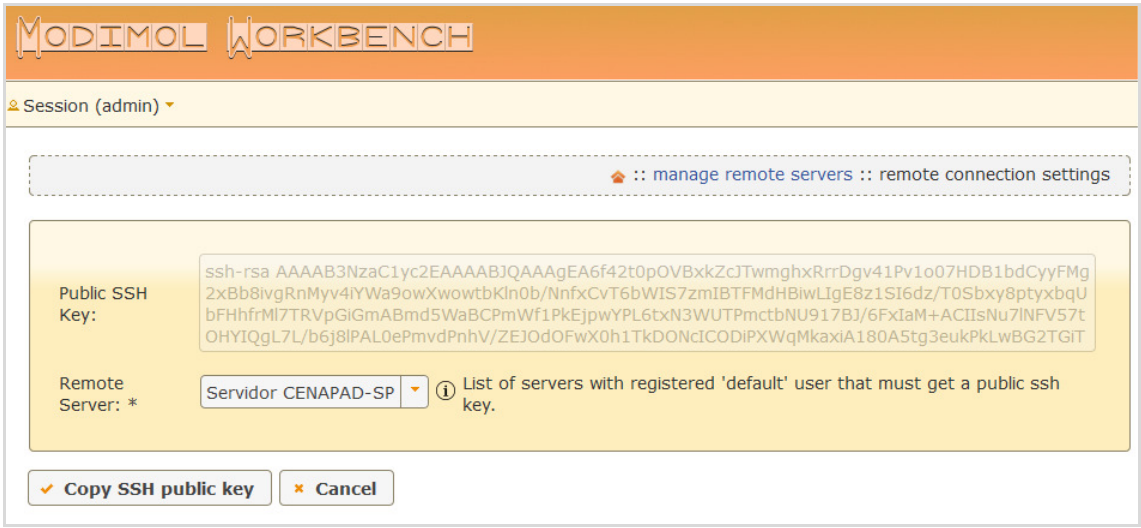
Como já discutido na seção 4.6.2 do capítulo de metodologia, é possível definir para um servidor remoto um usuário “compartilhado”. Para isto, é necessário definir o nome de usuário utilizado (*Username*) e a pasta de trabalho remota (*Work Directory*), na qual será criada uma pasta “temporária” para cada *job* processado no ambiente do servidor remoto.

Neste ponto, é importante salientar que, para um usuário “compartilhado”, não existe a necessidade do cadastro de sua senha, uma vez que o acesso a um ambiente remoto não se dará por meio de autenticação por senha, mas através de autenticação de chave pública (criptografia assimétrica).

A Figura 5.9 apresenta a interface presente no sistema MODiMOL *Workbench* para o procedimento de configuração da chave pública (SSH), utilizada para a conexão com um determinado servidor remoto, que se resume em copiá-la para o servidor remoto, usando internamente o comando *ssh-copy-id* (ver seção 4.6.2.1).

Primeiramente, o usuário precisa selecionar um servidor remoto da lista exibida pelo sistema, dos servidores remotos disponíveis para esta ação, ou seja, aqueles que possuem, necessariamente, um usuário “compartilhado” e que ainda não tiveram a chave pública (SSH) do servidor do sistema MODiMOL *Workbench* copiada. Ao selecionar o servidor remoto, e pressionar o botão “*Copy SSH public key*”, o sistema exibe uma janela modal contendo um campo para a entrada da senha do usuário “compartilhado” (Figura 5.10).

Figura 5.9. Interface de configuração para estabelecimento de conexão remota através de autenticação por chave pública.

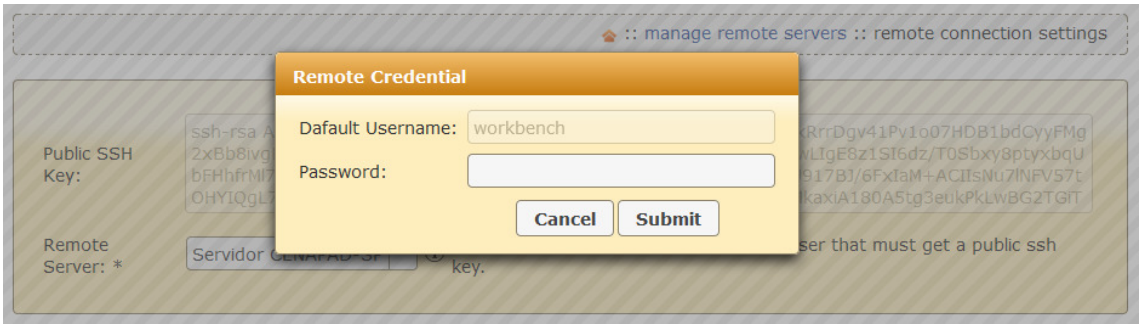



Fonte: O Autor, 2016.

Uma vez copiada a chave pública do sistema para o servidor remoto, o acesso ao sistema externo, através do usuário “compartilhado”, se dará sem a necessidade de sua senha. Desta forma, todos os usuários do sistema MODiMOL *Workbench* com perfil “Intermediário” passam a ser capazes de enviar *jobs* para

execução remota neste servidor, através de conexões estabelecidas por meio do usuário “compartilhado”.

Figura 5.10. Janela modal para digitação de senha do usuário “compartilhado” durante procedimento de configuração de conexão remota através de autenticação por chave pública.

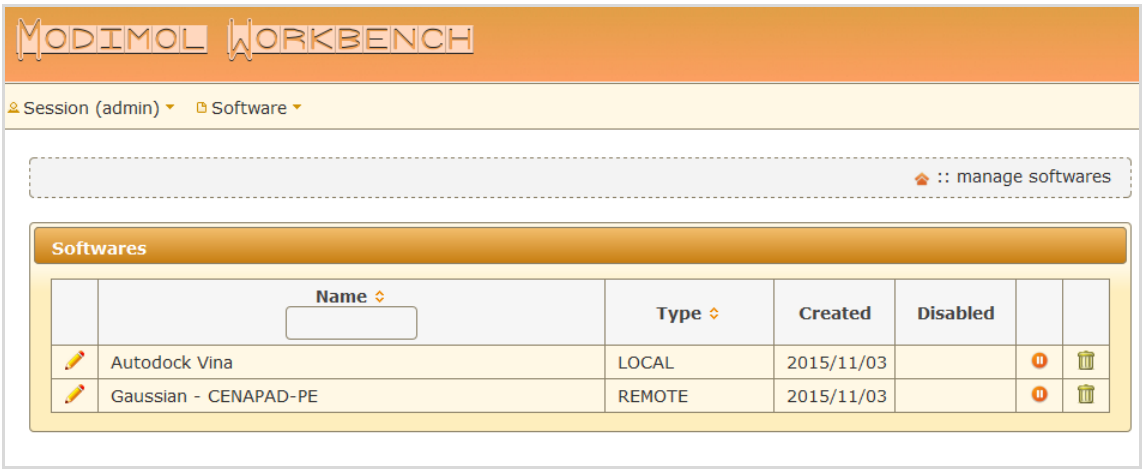



Fonte: O Autor, 2016.

### *5.1.3 Gerenciamento de algoritmos (softwares)*

A Figura 5.11 apresenta a interface contendo a listagem de softwares cadastrados no sistema MODiMOL *Workbench*. Da forma análoga ao exposto na seção anterior, um *software* também pode ser ativado/desativado pelo usuário. Estas ações, além de alterar o estado do software em questão, também propagam seus reflexos para os serviços associados.

Figura 5.11. Interface para listagem de softwares.



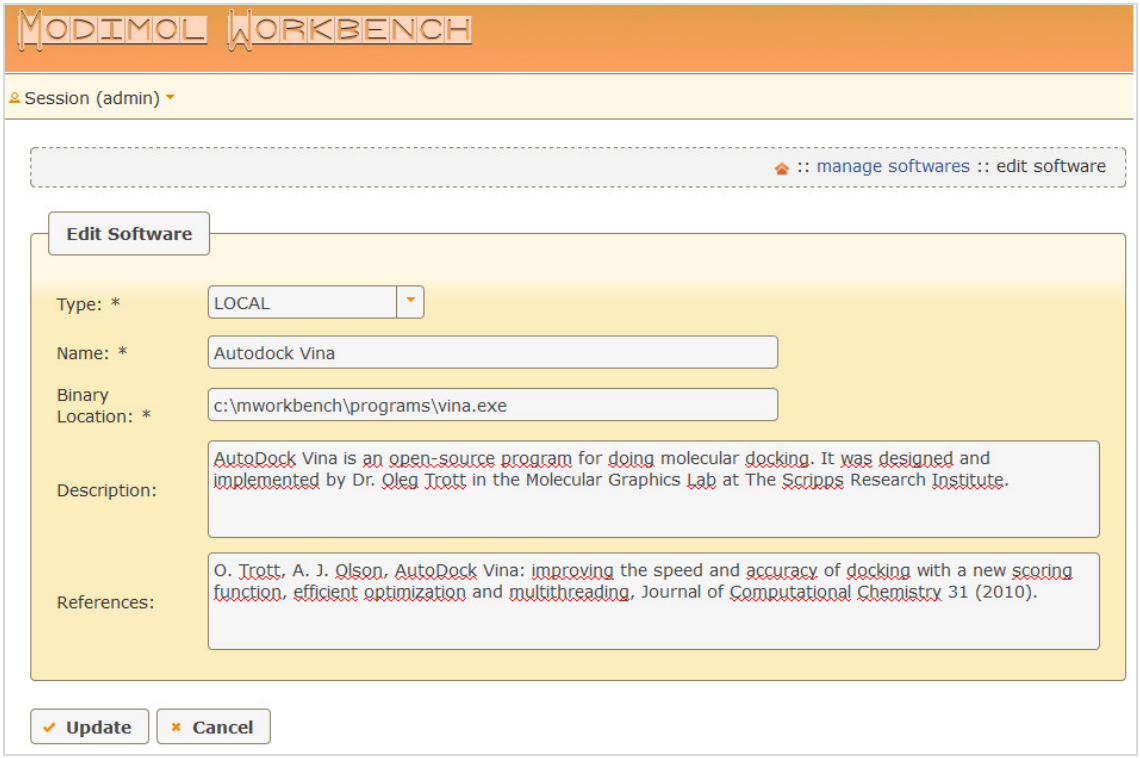

Fonte: O Autor, 2016.

Conforme já discutido no capítulo de metodologia, *softwares* podem ser executados local ou remotamente. Na listagem da Figura 5.11, é possível observar a coluna que exibe o tipo do software (*Type*) cadastrado, ou seja, softwares locais e remotos.

As Figuras 5.12 e 5.13 apresentam as interfaces do sistema MODiMOL *Workbench* para edição de um software local e remoto, respectivamente. Em ambos os casos, é necessário incluir as informações básicas: nome (*Name*), descrição (*Description*) e referências (*References*).

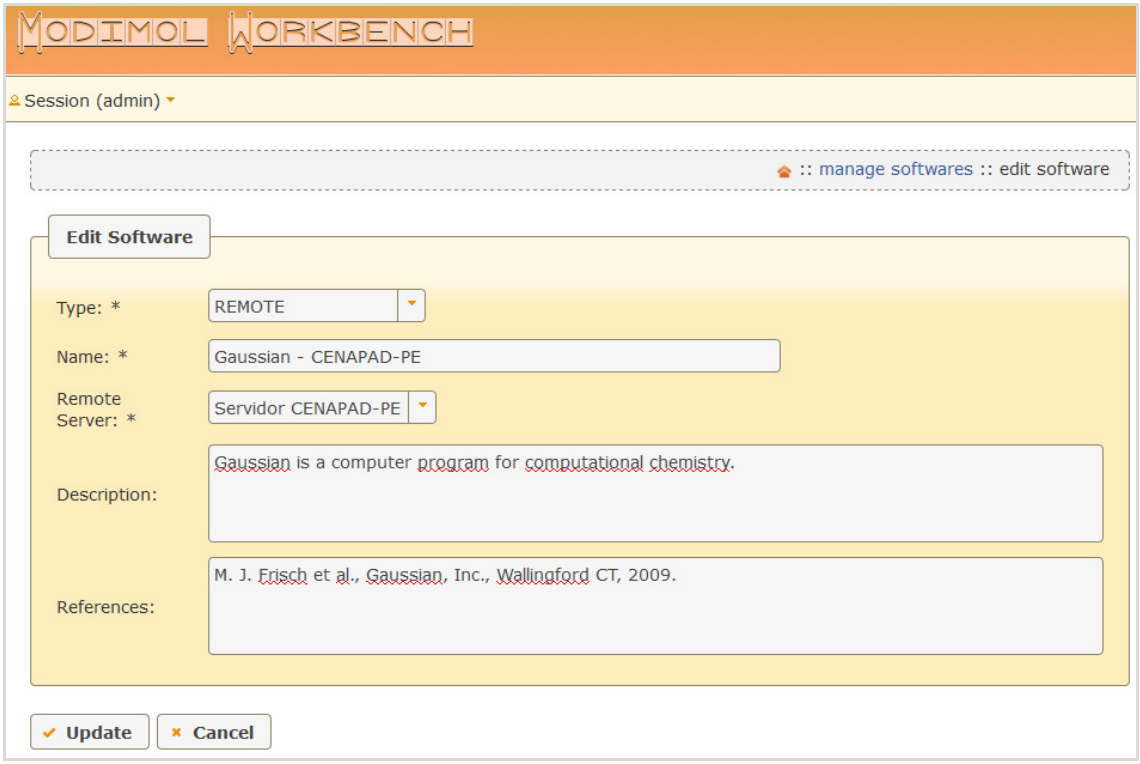
Para o cadastro de um *software* local, é necessário selecionar o tipo (*Type=LOCAL*) e definir o caminho, no sistema de arquivos, para o arquivo executável (binário ou *script*) correspondente ao *software* (algoritmo) a ser utilizado. Por outro lado, o cadastro de um software remoto (*Type=REMOTE*) requer a seleção do servidor (*Remote Server*) no qual se encontra o algoritmo que se deseja utilizar.

Figura 5.12. Interface para edição de um software local.




Fonte: O Autor, 2016.

Figura 5.13. Interface para edição de um software remoto.




Fonte: O Autor, 2016.

### *5.1.4 Gerenciamento de serviços*

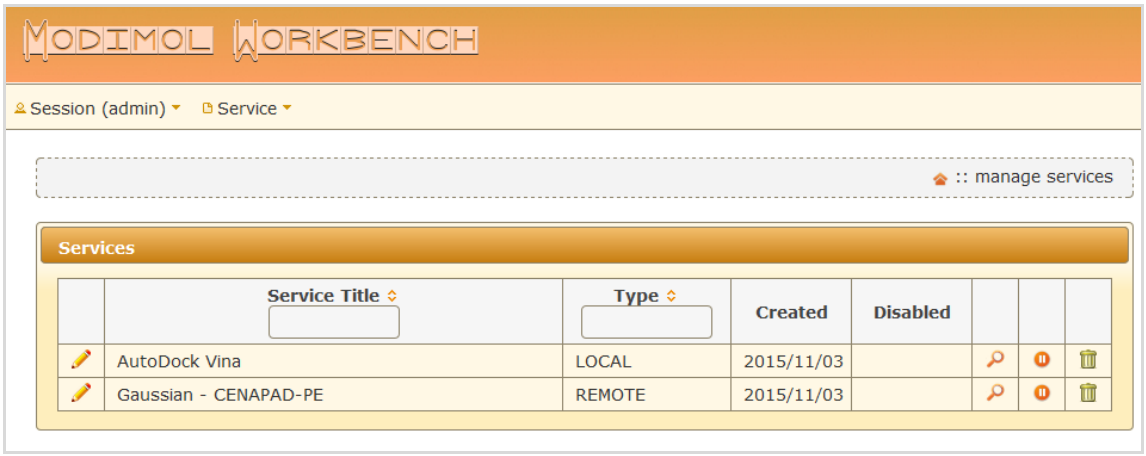
Um serviço, conforme já foi descrito, consiste em uma forma de flexibilização do *software* (algoritmo), no qual é possível se definir diversos cenários de uso para o mesmo *software*. No contexto do sistema MODiMOL *Workbench*, serviços podem ser cadastrados para *softwares* locais e remotos.

A Figura 5.14 apresenta a interface de listagem dos serviços cadastrados no sistema MODiMOL *Workbench*. Assim como na interface apresentada para a listagem de *softwares*, esta interface também exibe a lista de serviços que pode ser ordenada, de acordo com a necessidade do usuário, (através de "*links*" presentes nos títulos das colunas). Adicionalmente, é possível visualizar os detalhes de um serviço em uma janela modal separadamente, através do botão de "lupa" que se encontra na parte à direita de cada registro na tabela da listagem de serviços.

Durante o cadastro de um serviço, é necessário criar um mecanismo para facilitar a passagem dos parâmetros de entrada do usuário ao serviço. Este mecanismo consiste em duas estratégias: para um serviço local, é necessário se passar os parâmetros de entrada por meio de uma linha de comando a ser

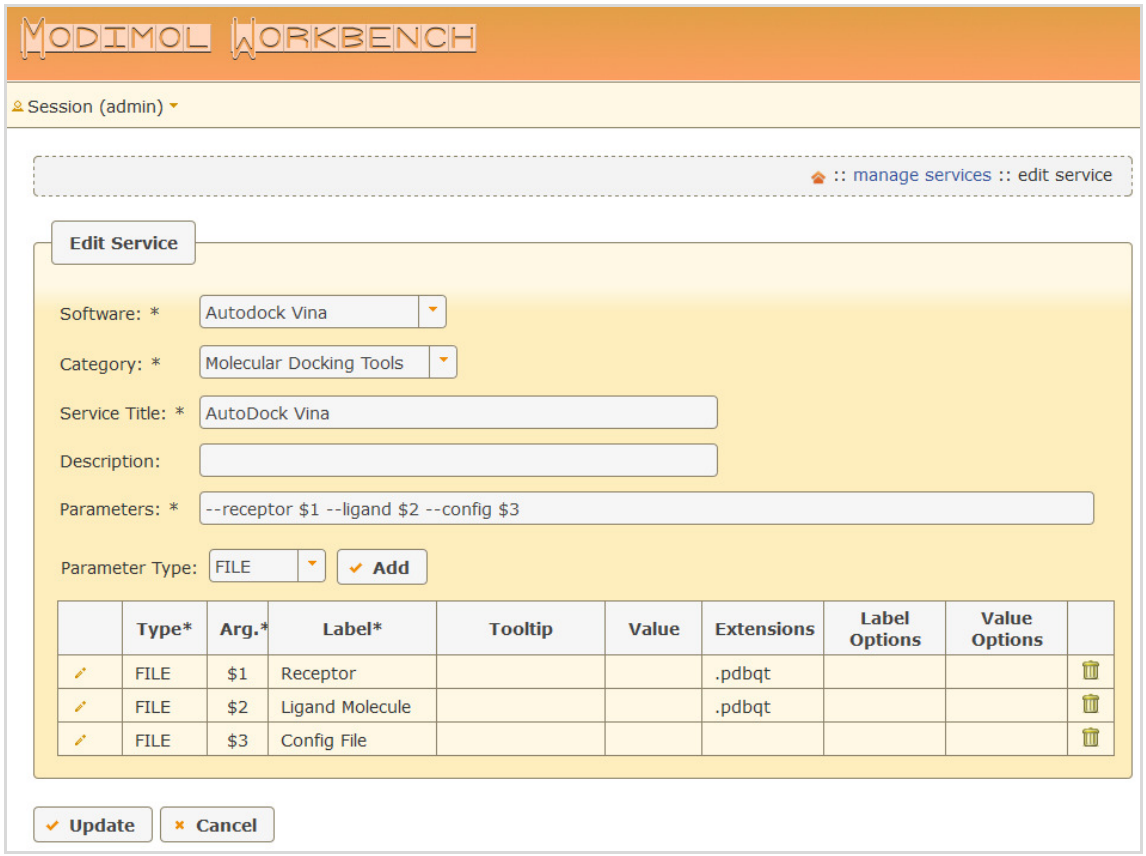
executada. Por outro lado, para um serviço remoto, é necessário a criação de um *shell script* a fim de ser executado no servidor externo, contendo as diretivas e comandos para a execução do algoritmo, no ambiente de computação remoto.

Figura 5.14. Interface para listagem de serviços.




Fonte: O Autor, 2016.

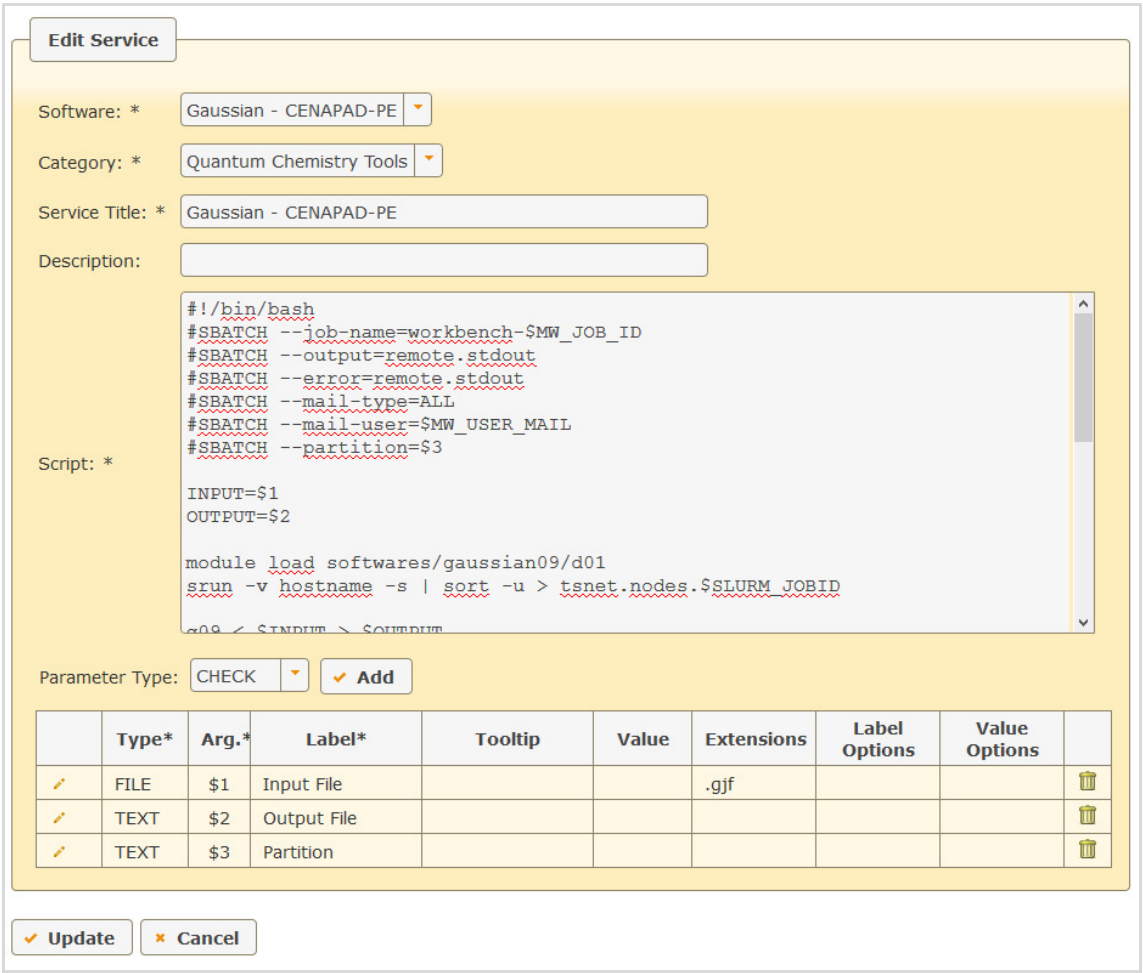
Figura 5.15. Interface para edição de um serviço local.




Fonte: O Autor, 2016.

As Figuras 5.15 e 5.16 apresentam as interfaces responsáveis pela edição de um serviço local e remoto, respectivamente. Em ambos os casos, o cadastro do serviço requer algumas informações básicas: algoritmo a ser utilizado (*Software*), categoria (*Category*), título (*Service Title*) e descrição (*Description*) do serviço. Para um serviço local (Figura 5.15), conforme explicado anteriormente, é necessário incluir os parâmetros (*Parameters*) que serão utilizados na linha de comando a ser executada pelo sistema localmente (usando os recursos computacionais da máquina onde se encontra o servidor do sistema MODiMOL *Workbench*).

Figura 5.16. Interface para edição de um serviço remoto.




Fonte: O Autor, 2016.

No caso do serviço remoto (Figura 5.16), a necessidade específica consiste na inclusão do conteúdo do *shell script* (*Script*) que será executado remotamente no ambiente do servidor remoto. É importante mencionar que o conteúdo do *script*

editado neste campo deve ser compatível com o interpretador de comandos existente no servidor remoto utilizado (*bash*, *tcsh*, etc.). Os parâmetros de entrada utilizados tanto na linha de comando do serviço local, quanto no *shell script* do serviço remoto são definidos em termos de "variáveis" presentes em ambos os casos. Estas "variáveis" são cadastradas levando-se em consideração seu tipo (*Type*), argumento (*Argument*), título (*Label*), dica para o usuário (*Tooltip*), valor inicial (*Value*), extensões para uso como filtro de arquivos (*Extensions*), e lista de opções (*Label Options/Value Options*).

Cada uma das "variáveis" cadastradas para um serviço deve ser mapeada para outra correspondente, na linha de comando (serviço local) ou no *script* (serviço remoto). Por exemplo, na Figura 5.16, a variável do tipo "FILE", cujo argumento é "$1", está mapeada no conteúdo do *shell script* do serviço correspondente à linha "INPUT=$1". Desta forma, quando o usuário selecionar este serviço para submeter um *job* para execução, ao preencher os parâmetros necessários para o serviço na interface gráfica do sistema, o valor correspondente à variável de argumento "$1", será usado para substituir o parâmetro presente no conteúdo do *script* (INPUT=$1) antes da submissão do *job*.

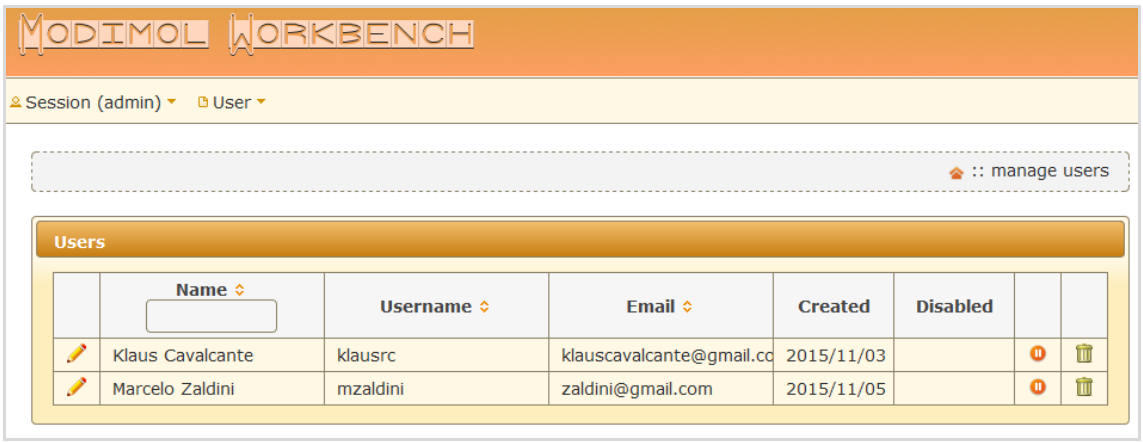
### *5.1.5 Gerenciamento de usuários*

A Figura 5.17 apresenta a interface de listagem de usuários cadastrados no sistema MODiMOL *Workbench*. Nesta listagem são exibidas algumas informações básicas de cada usuário: nome (*Name*), nome de usuário (*Username*) e correio eletrônico (*Email*). Nesta listagem, os usuários podem ser ordenados por qualquer um dos campos mencionados anteriormente.

A funcionalidade de ativação/desativação também está presente, permitindo que um usuário seja habilitado/desabilitado em realizar o acesso ao sistema, de acordo com alguma necessidade estipulada pelo administrador do sistema.

Conforme já discutido no capítulo de metodologia (seção 4.6), um usuário pode ser criado com o perfil "Básico", "Intermediário", ou "Avançado". A Figura 38 apresenta a interface de edição de um usuário no sistema MODiMOL *Workbench*. No exemplo da figura, o usuário que está sendo editado possui o perfil "Avançado".

Figura 5.17. Interface para listagem de usuários.



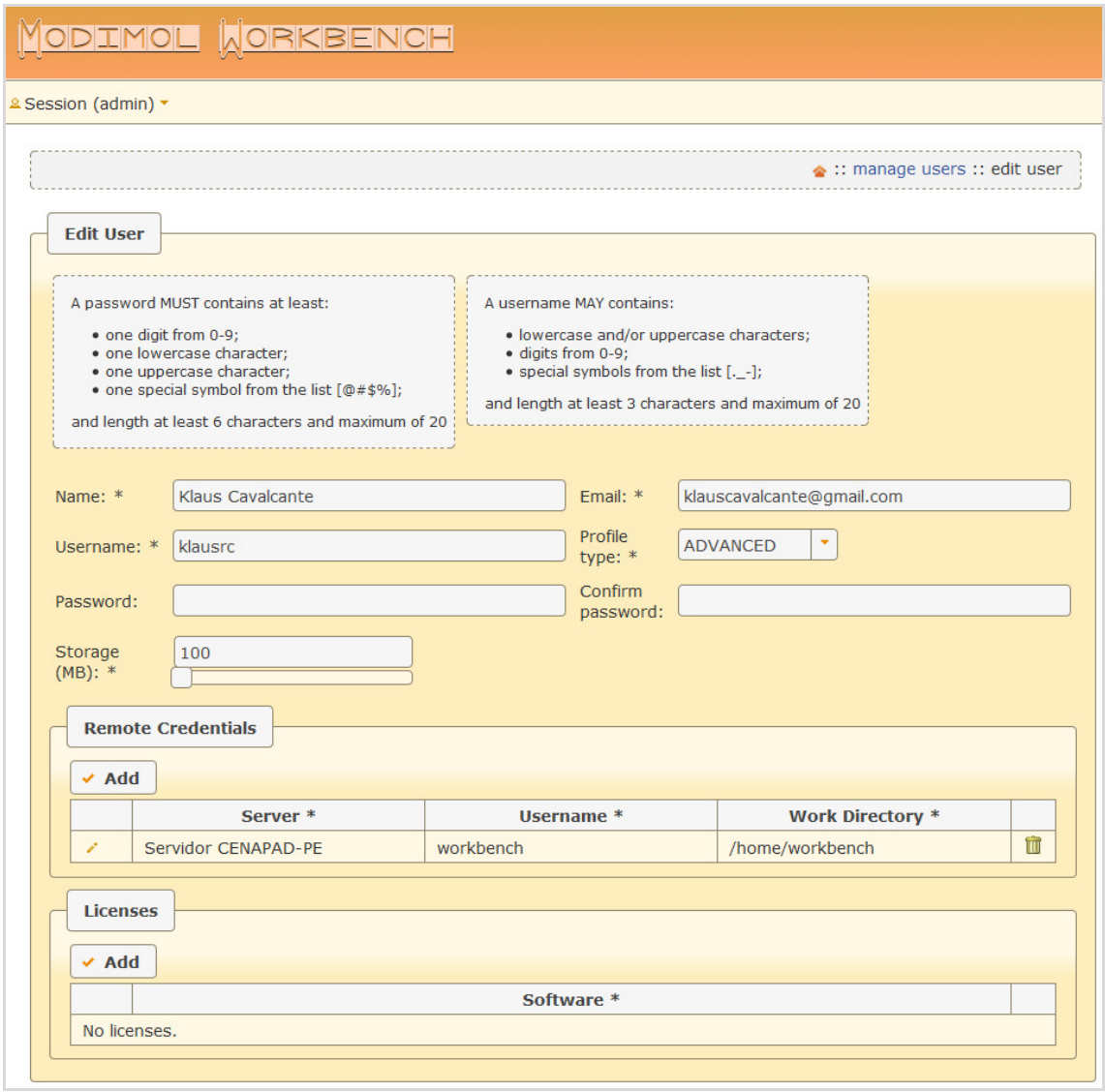

Fonte: O Autor, 2016.

Neste caso, além das suas informações básicas, como nome (*Name*), nome de usuário (*Username*), correio eletrônico (*Email*), senha (*Password*) e quota de armazenamento (*Storage*), é possível definir também as credenciais necessárias para autenticação em servidores remotos (*Remote Credentials*), além das permissões para utilização de algoritmos (*softwares*) licenciados (*Licenses*).

No sistema MODiMOL *Workbench*, uma credencial remota é cadastrada através da definição de três elementos: servidor remoto (*Server*), nome de usuário utilizado no servidor (*Username*) e pasta de trabalho remota, que será usada durante a execução dos *jobs* neste ambiente. Para este tipo de perfil, durante a submissão de um *job* remoto, será solicitado ao usuário do sistema, a digitação da senha do "usuário remoto", a fim de se obter permissão de acesso aos recursos computacionais para execução do *job* no sistema externo.

Para os outros tipos de perfis existentes, "Básico" e "Intermediário", a interface de edição de usuários do sistema MODiMOL *Workbench* exibe apenas os campos básicos de informações do usuário, ocultando os campos correspondentes ao preenchimento das informações referentes às credenciais remotas e licenças de *software*, visto que para estes tipos de perfis, estas informações não são necessárias.

Figura 5.18. Interface para edição (ou cadastramento) de um usuário.




Fonte: O Autor, 2016.

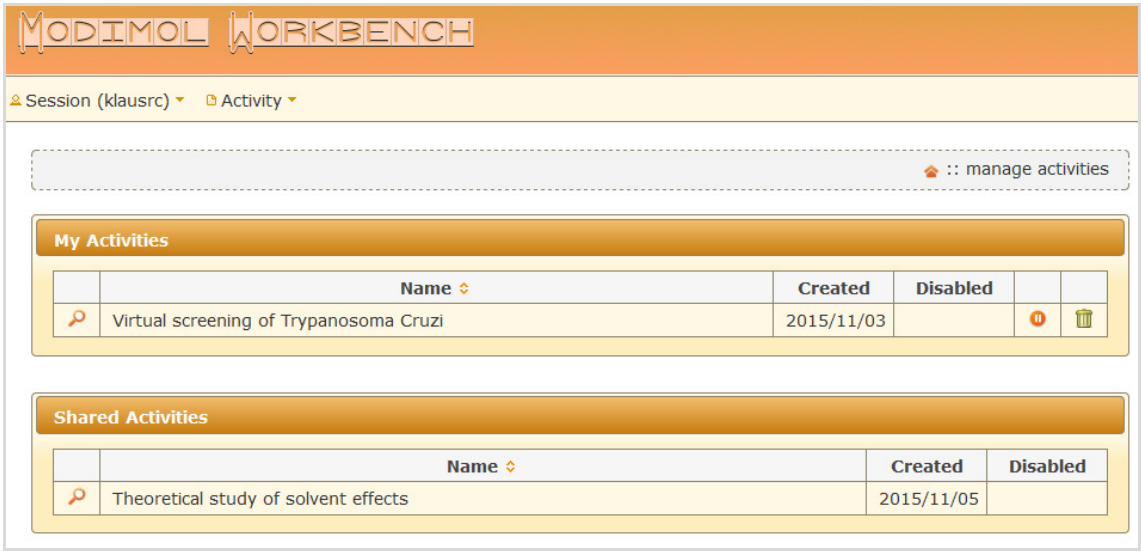
### *5.1.6 Gerenciamento de atividades*

A Figura 5.19 apresenta a interface do sistema MODiMOL *Workbench* correspondente às atividades relacionadas a um determinado usuário. Esta interface basicamente contém a listagem das atividades que foram criadas pelo próprio usuário que está autenticado no sistema, e a listagem das atividades compartilhadas por outros usuários. As atividades de propriedade do usuário autenticado no sistema podem ser editadas, excluídas, assim como, ativadas ou desativadas, tornando-as,

respectivamente, habilitadas ou desabilitadas para edição e submissão de novos *jobs* (Figura 5.20).

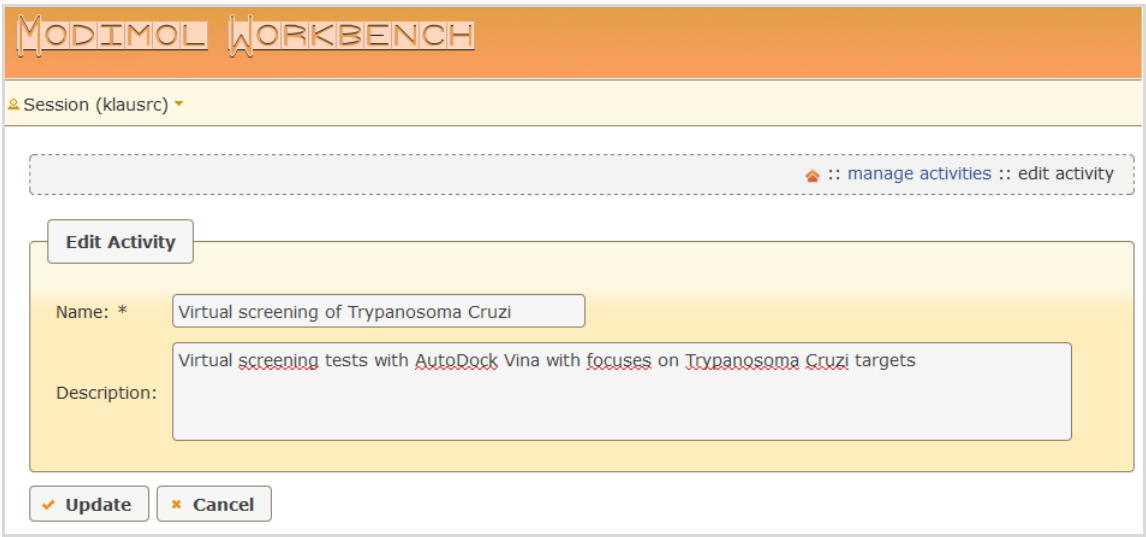
Por outro lado, as ações que dizem respeito às atividades compartilhadas dependem da permissão concedida ao usuário (somente leitura ou leitura/escrita), pelos proprietários destas atividades, durante o procedimento de compartilhamento. Nesta interface (Figura 5.19), é possível visualizar os detalhes das atividades no botão de “lupa”, presente em cada registro da listagem de atividades cadastrada no sistema.

Figura 5.19. Interface para listagem de atividades de um usuário.




Fonte: O Autor, 2016.

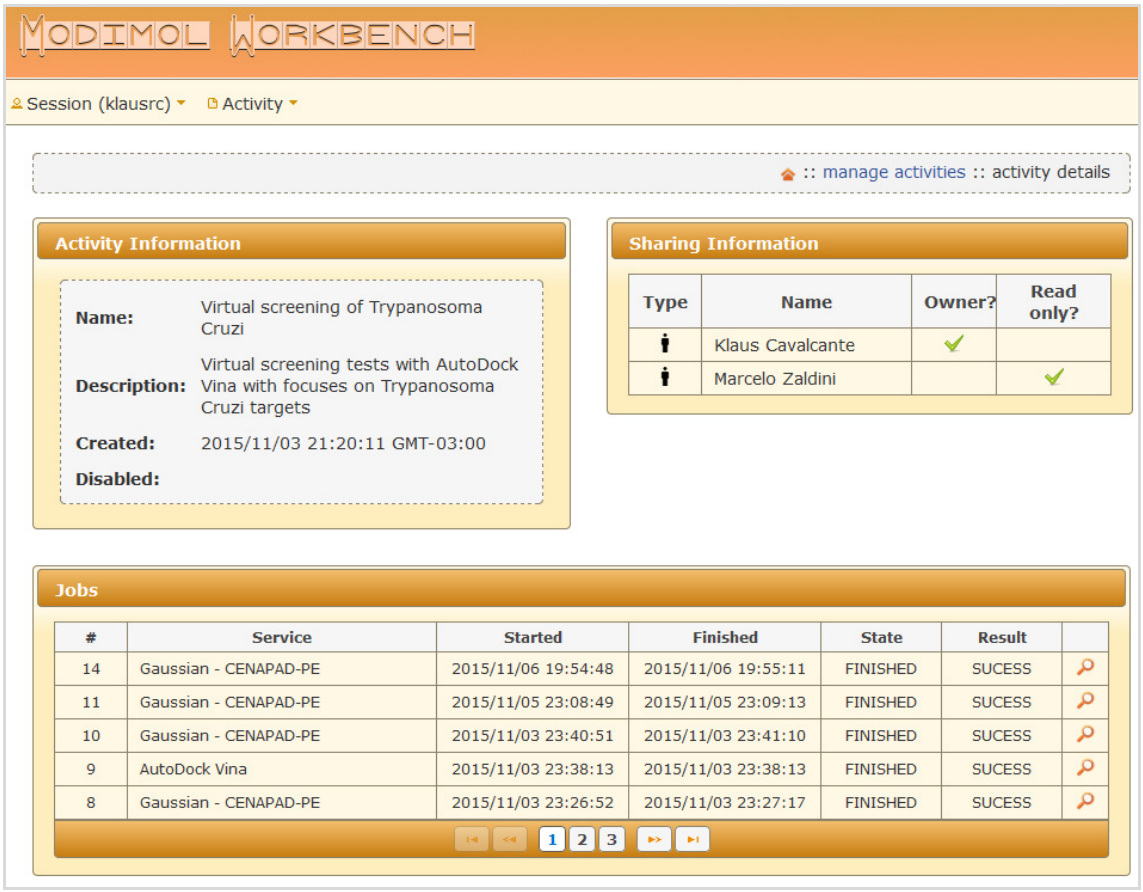
Figura 5.20. Interface para edição de uma atividade.




Fonte: O Autor, 2016.

O detalhamento das informações de uma atividade é ilustrado na Figura 5.21. Nesta interface são apresentadas os dados básicos da atividade (*Activity Information*), informações de compartilhamento (*Sharing Information*) e listagem dos *jobs* submetidos para execução (*Jobs*).

Figura 5.21. Interface para detalhamento de uma atividade.



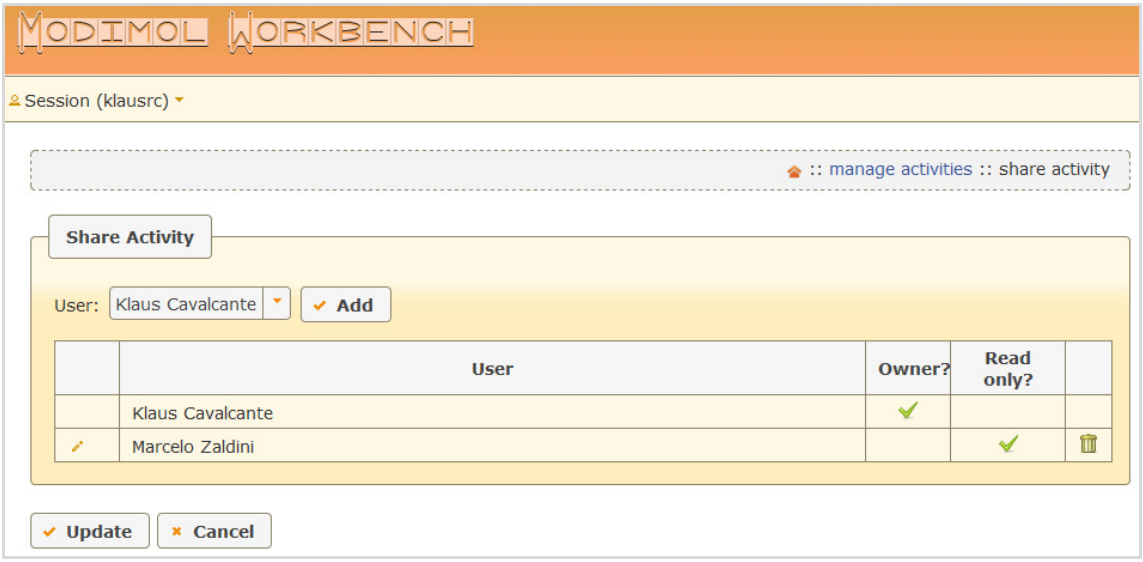

Fonte: O Autor, 2016.

O proprietário de uma atividade pode compartilhá-la através da interface apresentada na Figura 5.22. Nesta interface, o usuário proprietário da atividade pode incluir os usuários com os quais se deseja compartilhar o acesso à atividade, assim como, definir a regra de acesso para a mesma, no que diz respeito à permissão para edição da atividade e submissão de novos *jobs*.

Desta forma, é possível, por exemplo, que o proprietário de uma certa atividade a compartilhe com um grupo de usuários quaisquer, onde o acesso com permissão de atualização de dados é concedido para um determinado subconjunto

de usuários, enquanto que para os outros usuários restantes, o acesso é limitado apenas para a visualização da atividade.

Figura 5.22. Interface para edição das permissões de compartilhamento de uma atividade.



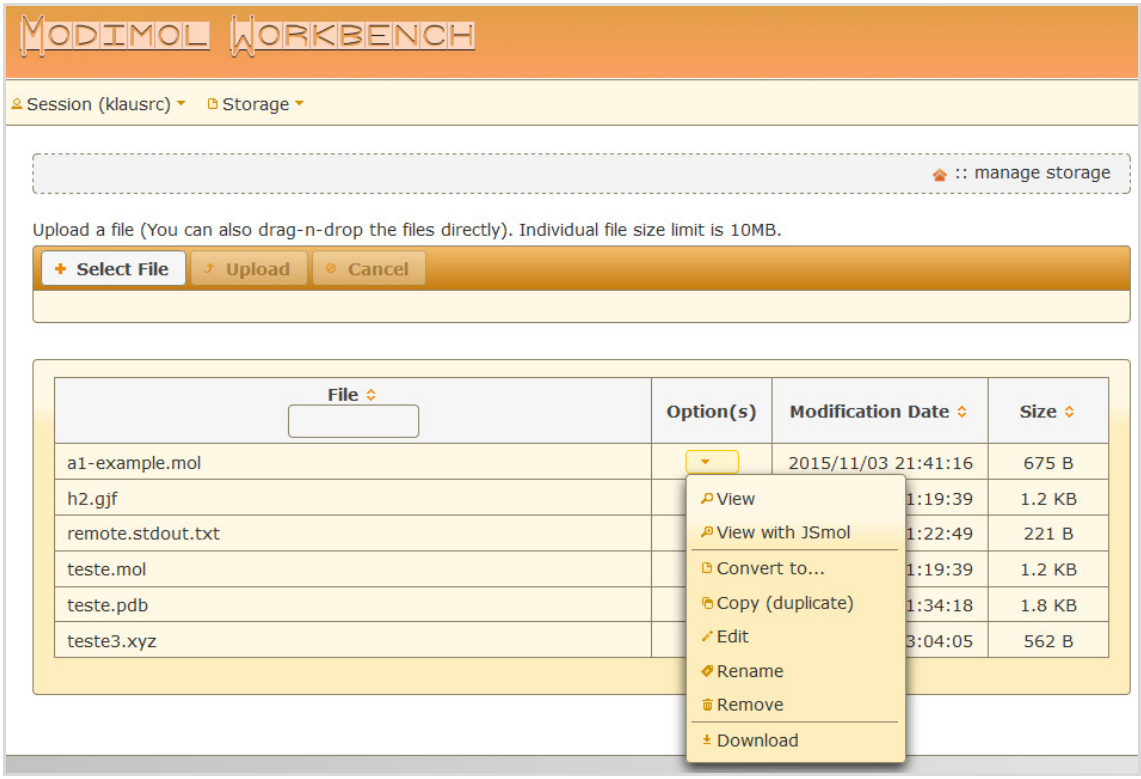

Fonte: O Autor, 2016.

### *5.1.7 Gerenciamento de arquivos do usuário*

Conforme já discutido no capítulo de metodologia, cada usuário do sistema MODiMOL *Workbench* possui uma área de armazenamento de dados individual, que se assemelha a um disco (*drive*) virtual, possibilitando que o usuário mantenha os arquivos usados em suas pesquisas, de forma organizada, diretamente no ambiente do sistema.

Além disso, quando serviços são definidos com o uso de arquivos como parâmetros de entrada para submissão de *jobs*, o sistema busca e utiliza unicamente os arquivos que estão disponíveis na área de armazenamento pessoal do usuário (conforme discutido na seção 4.5 do capítulo de metodologia). Desta forma, a área de armazenamento pessoal não é apenas um disco virtual destinado a manter os dados do usuário, mas é um dos componentes utilizados no mecanismo (“motor”) de submissão de *jobs* do sistema.

Figura 5.23. Interface de gerenciamento dos arquivos da área de armazenamento pessoal do usuário.

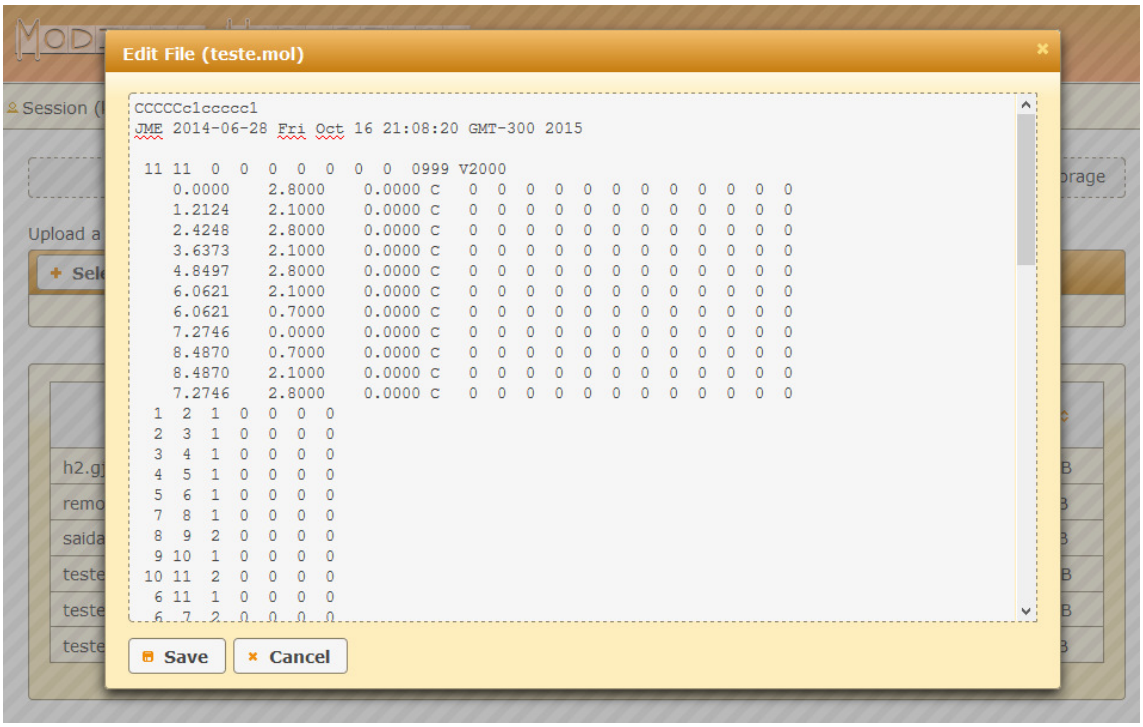



Fonte: O Autor, 2016.

A Figura 5.23 apresenta a interface do sistema MODiMOL *Workbench* utilizada no gerenciamento dos arquivos da área de armazenamento pessoal do usuário. Nesta interface, é possível realizar diversas ações sobre os arquivos (disponibilizadas em um menu de opções), como por exemplo, edição (Figura 5.24), visualização do conteúdo, renomeação, exclusão, conversão entre formatos, cópia (duplicação), e transferência do arquivo para a máquina do usuário ("*download*").

Esta interface (Figura 5.23) também permite que o usuário faça a inclusão de um arquivo, através da transferência do mesmo a partir de sua máquina para o servidor do sistema MODiMOL *Workbench* ("*upload*"). Em relação ao recurso de conversão do formato, o sistema permite que os arquivos possam ser convertidos entre diversos tipos de formatos existentes, por meio da utilização da ferramenta OpenBabel (O'BOYLE *et al.*, 2011). A Figura 5.25 apresenta a interface usada para a conversão de formato de arquivos, para um arquivo presente na área de armazenamento pessoal do usuário.

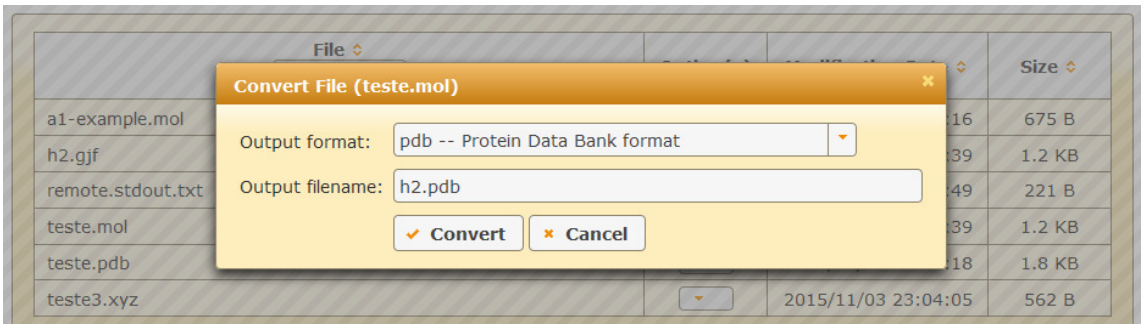
Figura 5.24. Interface para edição de arquivos.




Fonte: O Autor, 2016.

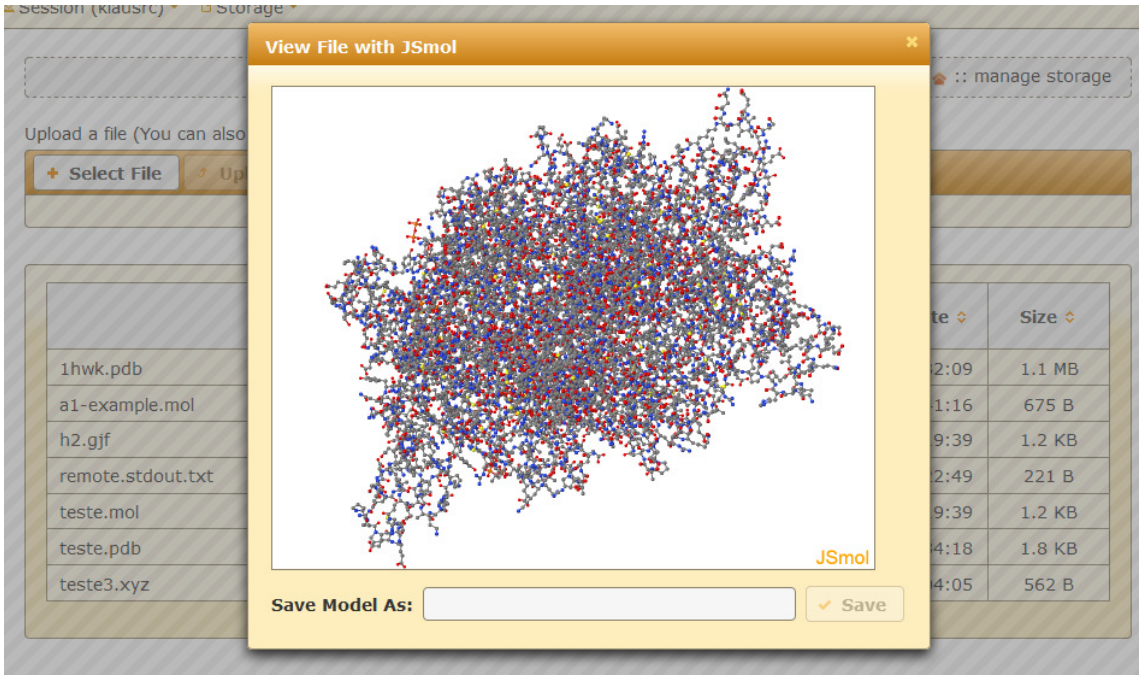
Outro recurso presente no sistema MODiMOL *Workbench* é a visualização molecular usando a ferramenta JSmol (Figura 5.26) (JSMOL, 2016). O JSmol é uma ferramenta desenvolvida em JavaScript que possui diversas funcionalidades, além da visualização molecular. Algumas destas funcionalidades permitem alterar a estrutura da molécula (apenas os dados que estão sendo visualizados), como por exemplo, o uso de procedimentos de otimização de geometria através de mecânica molecular com “campos de força”, como o UFF (*Universal Force Field*) (RAPPE *et al.*, 1992), presente no JSmol.

Figura 5.25. Interface para conversão de formatos de arquivos.




Fonte: O Autor, 2016.

Figura 5.26. Interface para visualização molecular tridimensional e interativa, usando a ferramenta JSmol.




Fonte: O Autor, 2016.

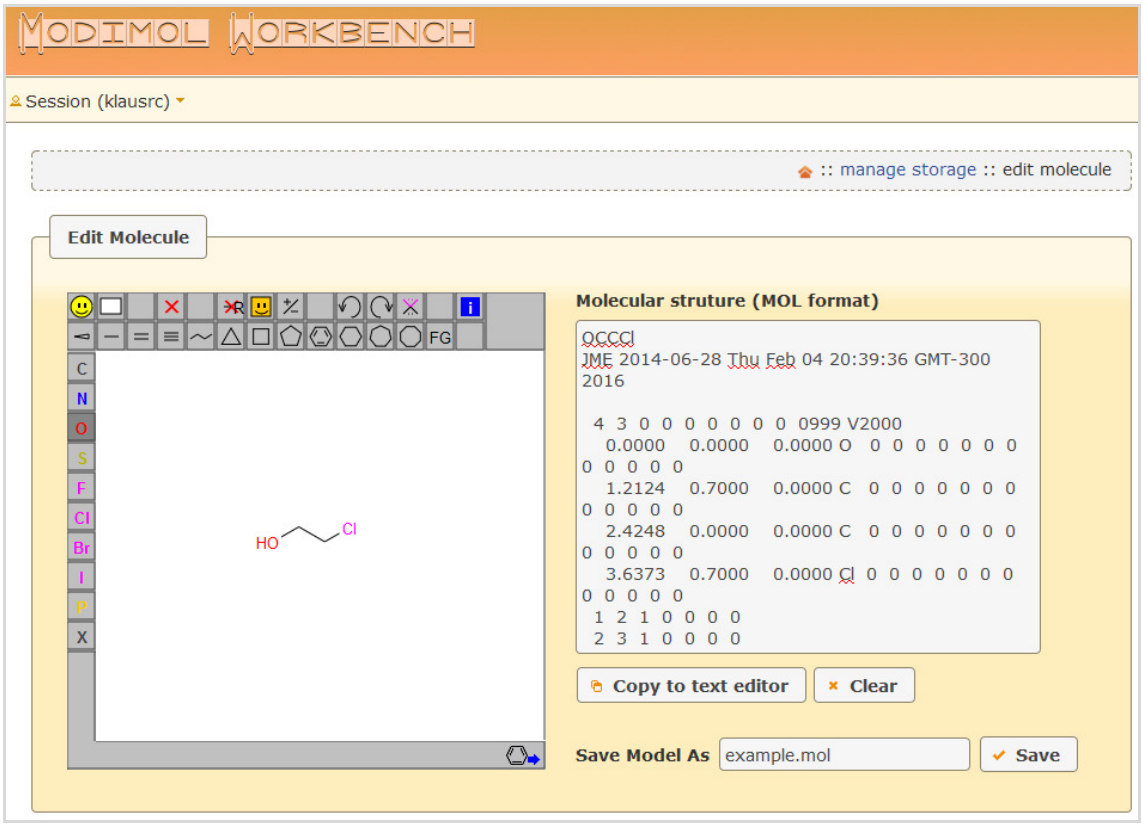
Portanto, em virtude da possibilidade de alteração da estrutura tridimensional da molécula no componente gráfico do JSmol, existe a necessidade de armazenamento das mudanças efetuadas. Desta forma, o sistema possibilita que o modelo molecular alterado possa ser salvo como um novo arquivo no formato MOL (DALBY *et al.*, 1992), na área de armazenamento pessoal do usuário.

A Figura 5.27 apresenta a interface usada para a inclusão de um novo arquivo através do editor molecular JSME (BIENFAIT & ERTL, 2013). Esta ferramenta gráfica permite o desenho de uma nova estrutura química diretamente no navegador Web, flexibilizando, desta forma, a obtenção de estruturas moleculares para utilização dentro do sistema. Esta interface também possibilita a edição da estrutura da molécula desenhada na ferramenta JSME em um editor de texto (botão "*Copy to text editor*"), permitindo que o usuário faça quaisquer alterações necessárias antes de salvar o conteúdo do arquivo na sua área de armazenamento pessoal.

O sistema MODiMOL *Workbench*, conforme apresentado no capítulo de metodologia, permite a possibilidade de acesso a bancos de dados públicos que contém estruturas moleculares disponíveis. O objetivo da integração do sistema com

estes bancos de dados consiste em flexibilizar ainda mais a obtenção de novas estruturas moleculares no sistema, além das formas já discutidas ("*upload*" e desenho molecular com JSME).

Figura 5.27. Interface para desenho e edição molecular, usando a ferramenta JSME.



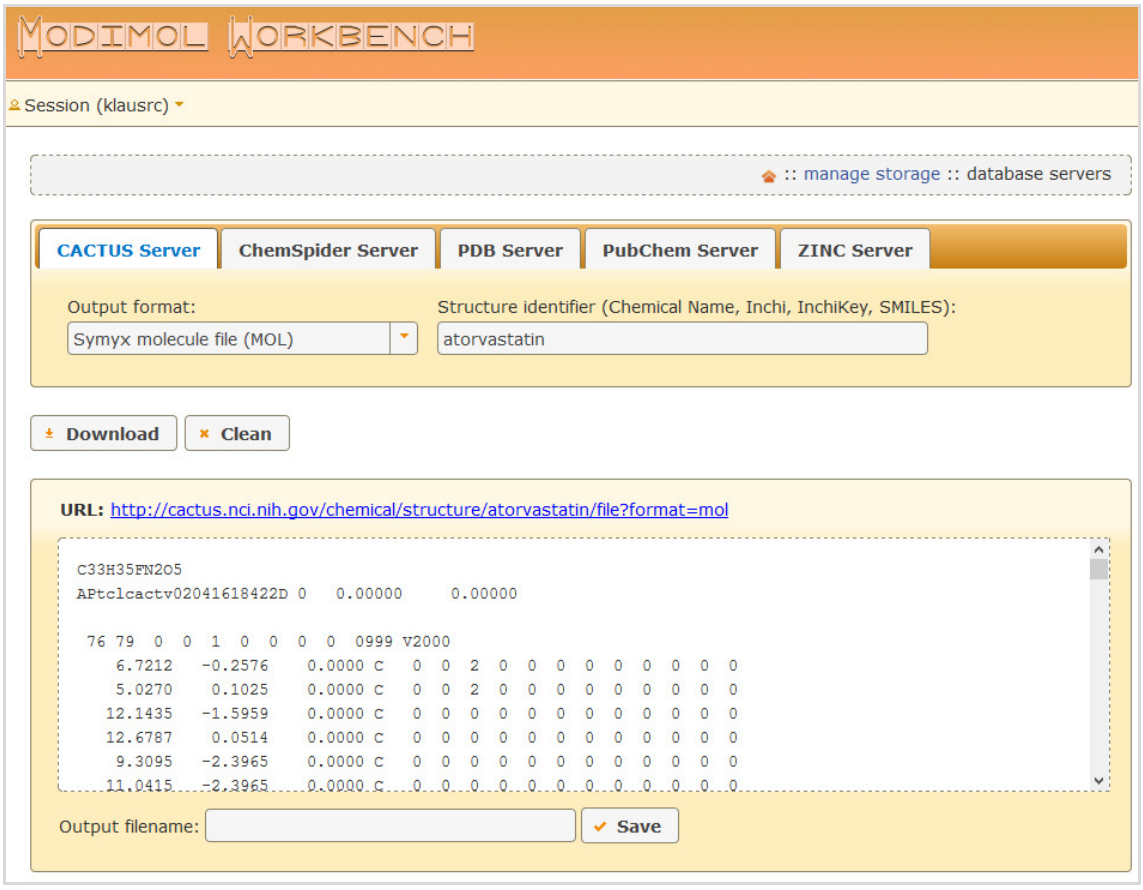

Fonte: O Autor, 2016.

As Figuras 5.28, 5.29, 5.30, 5.31 e 5.32 apresentam as interfaces correspondentes à integração do sistema MODiMOL *Workbench* com os serviços externos que disponibilizam estruturas moleculares para consulta e obtenção. Em todos os exemplos (distribuídos em "abas" na interface de usuário), as estruturas moleculares recuperadas a partir destes serviços podem ser visualizadas e salvas em arquivos na área de armazenamento pessoal do usuário, no formato escolhido.

No primeiro exemplo (Figura 5.28), é utilizado o banco de dados CACTUS (CACTUS, 2016) para a recuperação de uma estrutura molecular no formato MOL (DALBY *et al.*, 1992), passando como parâmetro o nome de uma estrutura química. Este banco de dados permite outros tipos de parâmetros de entrada e saída. Por exemplo, para o parâmetro de entrada, é possível passar, além do nome da estrutura química, também o identificador da estrutura desejada nos formatos Inchi,

InchiKey (HELLER et al., 2013) e SMILES (WEININGER, 1988). Este serviço permite recuperar os dados de estruturas moleculares em vários formatos que são amplamente utilizados: MOL, MOL2, PDB, SDF e XYZ.

Figura 5.28. Interface de integração com bancos de dados externos (CACTUS).

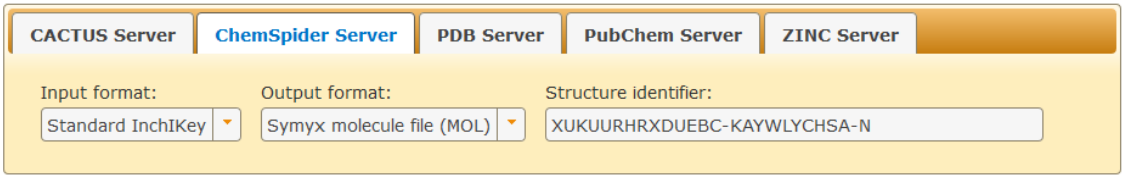



Fonte: O Autor, 2016.

No segundo exemplo (Figura 49), neste caso com foco no banco de dados ChemSpider, foi utilizado o mesmo parâmetro de entrada do exemplo anterior (CACTUS), porém inserido no formato InChiKey. Este identificador foi obtido a partir de uma consulta prévia ao sítio do ChemSpider na Internet[1], com a finalidade de exemplificar o uso do serviço a partir do sistema MODiMOL *Workbench*.

[1] http://www.chemspider.com/Chemical-Structure.54810.html?rid=0f06ff2a-9b4a-40e9-aa3d-26c66f26 4e81

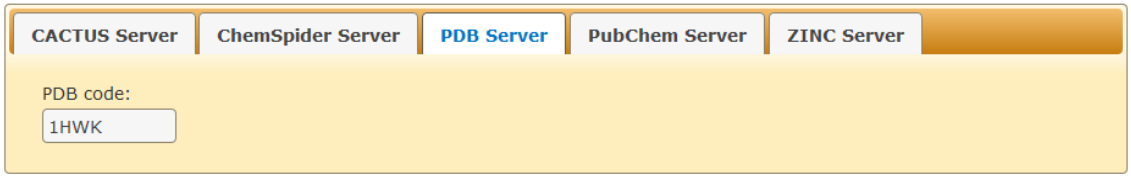
Figura 5.29. Interface de integração com bancos de dados externos (ChemSpider).




Fonte: O Autor, 2016.

A Figura 5.30 apresenta a interface responsável pela integração com o banco de dados PDB. Para este exemplo, o único campo de entrada disponível é o código identificador da estrutura (*PDB code*) a ser recuperada.

Figura 5.30. Interface de integração com bancos de dados externos (PDB).



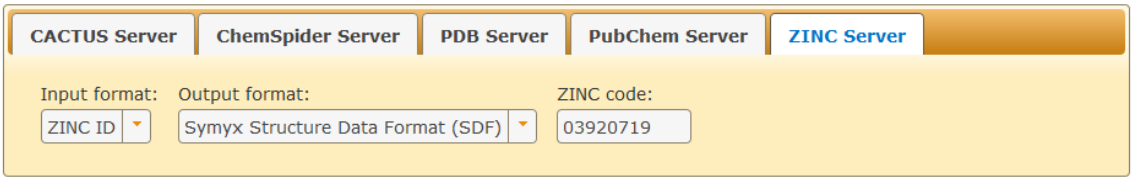

Fonte: O Autor, 2016.

As Figuras 5.31 e 5.32 apresentam as interfaces de integração com os bancos de dados PubChem e ZINC, respectivamente. Ambos os exemplos disponibilizam diversos opções de formatos a serem utilizados para a entrada e saída de dados.

Figura 5.31. Interface de integração com bancos de dados externos (PubChem).


Fonte: O Autor, 2016.

Figura 5.32. Interface de integração com bancos de dados externos (ZINC).

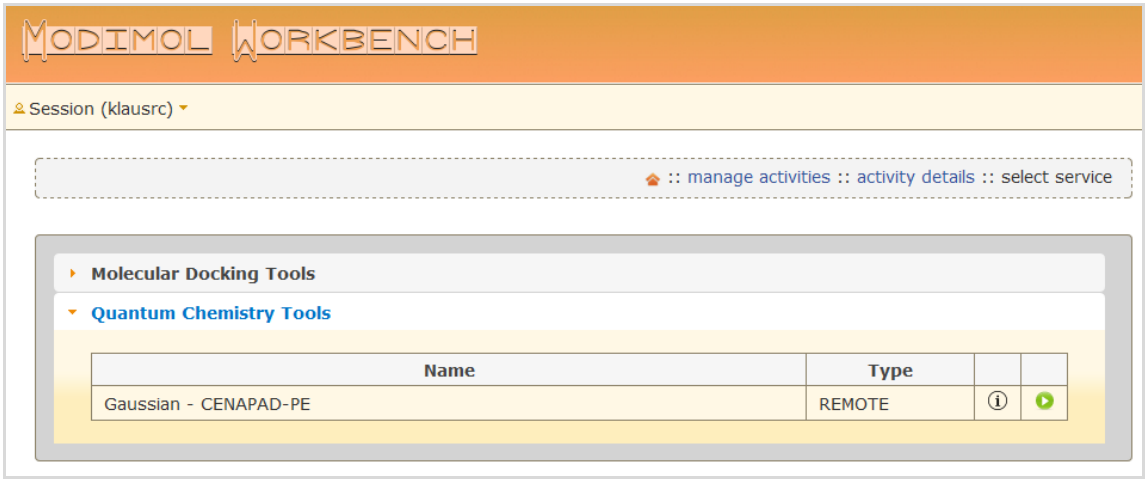



Fonte: O Autor, 2016.

### *5.1.8 Submissão de jobs*

Serviços são cadastrados e organizados em categorias. Esta organização pode ser melhor compreendida no momento em que um usuário seleciona um determinado serviço no sistema, com a finalidade de submetê-lo para execução. A Figura 5.33 apresenta a interface na qual o usuário seleciona um serviço para execução.

Figura 5.33. Interface para seleção de serviços.



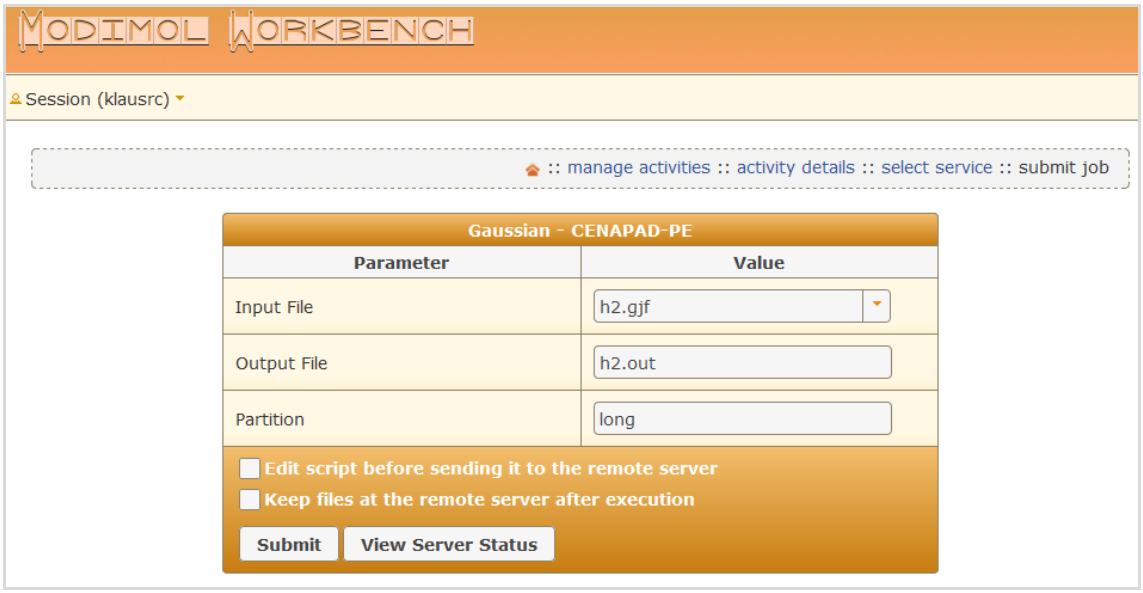

Fonte: O Autor, 2016.

Esta interface, conforme explicado anteriormente, expõe de forma classificada por categoria, todos os serviços disponíveis no sistema, dando maior percepção, e agilidade para a seleção do serviço, pelo usuário da plataforma MODiMOL *Workbench*. Este recurso é importante pois em um cenário no qual dezenas, ou até centenas, de serviços são cadastrados no sistema, a busca por um determinado serviço em uma longa lista pode se tornar mais lenta e cansativa para o usuário. Além disso, as categorias são exibidas de forma minimizada, com exceção daquela cujo foco foi requerido pelo usuário, para detalhamento de seus serviços.

Uma vez que o usuário seleciona um serviço para execução, o sistema apresenta a interface para preenchimento dos parâmetros de entrada necessários para a submissão do *job* para execução (Figura 5.34). Neste ponto vale ressaltar que esta interface é montada dinamicamente levando em consideração os parâmetros utilizados durante o cadastro do serviço selecionado. Este mecanismo

representa uma grande vantagem em termos de administração do sistema, uma vez que, surgindo a necessidade de atualização de um serviço (e seus parâmetros), esta interface não precisa ser também atualizada para se adequar à mudança no serviço.

Figura 5.34. Interface para submissão de jobs.




Fonte: O Autor, 2016.

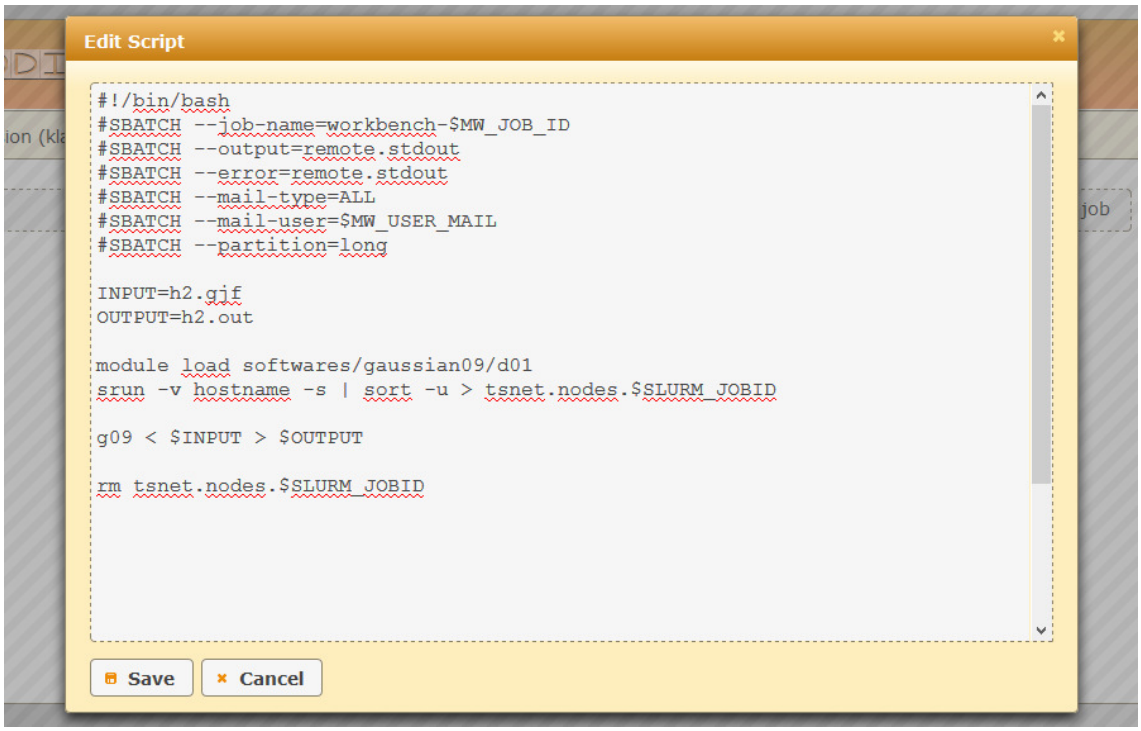
Além disso, como não há necessidade de reiniciar ou recompilar o sistema, após a atualização de um serviço, é possível utilizá-lo de forma imediata para submissão de novos *jobs* através da interface de submissão (Figura 5.34) já readequada para a nova configuração do serviço.

Uma das opções presentes na interface de submissão de *jobs* (Figura 5.34) é a possibilidade de edição do *shell script* antes de submetê-lo para execução no servidor remoto (campo de seleção "*Edit script before sending it to the remote server*"), sendo esta opção disponível apenas para serviços remotos. A Figura 5.35 apresenta a interface que permite a edição do *script* pelo usuário.

É possível observar que na interface do exemplo exibido na Figura 5.35, os valores dos parâmetros informados pelo usuário na interface de submissão (Figura 5.34) já foram usados para substituir as "variáveis" presentes no *script* (partition=long, INPUT=h2.gjf e OUTPUT=h2.out), restando apenas as "variáveis de sistema" ($MW_JOB_ID e $MW_USER_MAIL) que serão substituídas no momento da submissão do *job*. Ao realizar as modificações no conteúdo do *script*, o usuário

pode confirmar as alterações, fazendo com que o *script* seja finalmente submetido para execução no servidor remoto associado ao serviço selecionado.

Figura 5.35. Interface para edição do shell script de serviços remotos.




Fonte: O Autor, 2016.

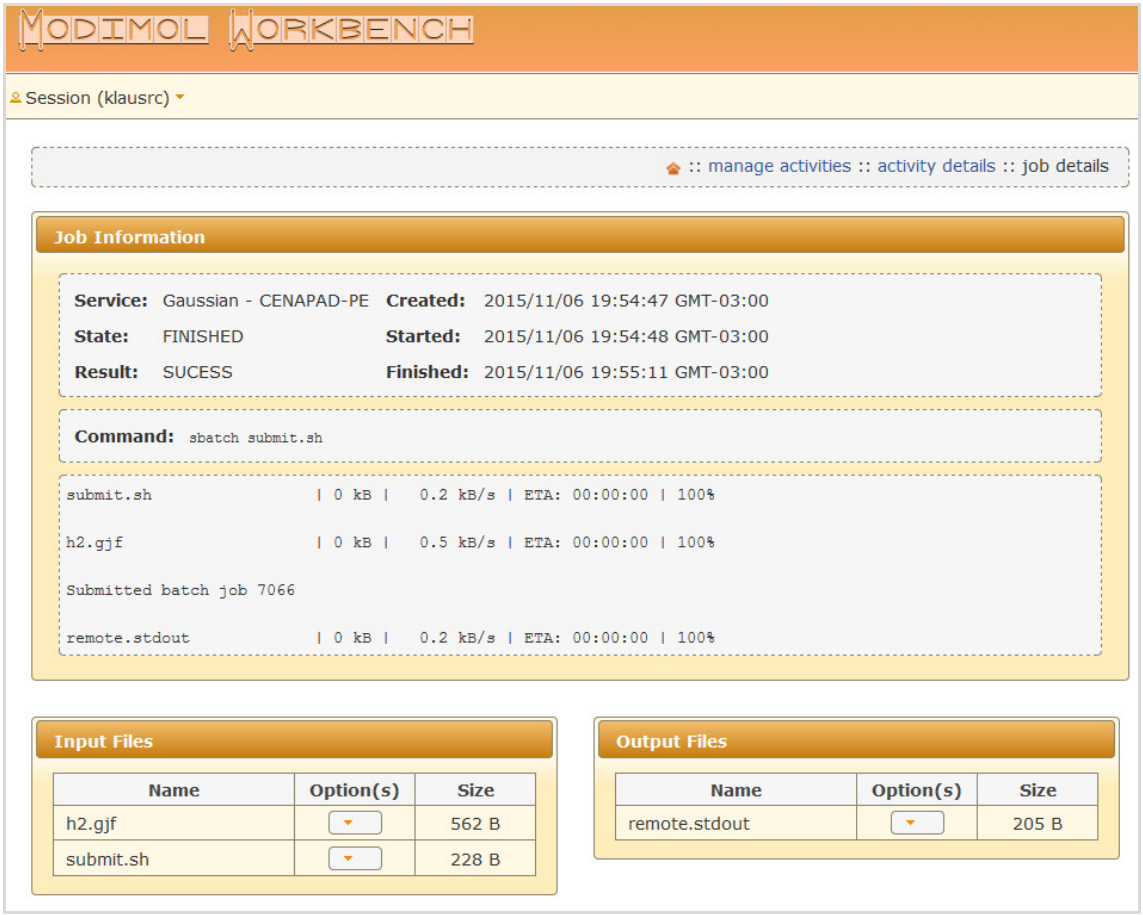
Outra opção existente na interface de submissão de *jobs* (Figura 5.34), também disponível apenas para serviços remotos, é a possibilidade de manter os arquivos de saída gerados na pasta de trabalho do servidor remoto, ao invés de excluí-los no final da execução do *job* (uma das etapas presentes no fluxo de execução padrão para *jobs* remotos - seção 4.5). Esta opção torna-se interessante quando existe a necessidade do usuário ter acesso aos arquivos gerados (de saída e temporários) pela execução do algoritmo no ambiente remoto.

Um exemplo típico de utilização de um recurso como este seriam os arquivos "*checkpoint*" intermediários de um programa de química quântica ("*.chk" do Gaussian, por exemplo), que sendo gerados durante um *job*, podem ser importantes para a continuação do cálculo em um próximo *job* submetido ao sistema. Poderíamos encontrar outros exemplos em vários programas, de diversas categorias (simulação computacional, química quântica, etc.).

A Figura 5.36 apresenta a interface do sistema MODiMOL *Workbench* usada para a visualização dos resultados obtidos após a execução dos *jobs*. Nesta interface é possível visualizar várias informações básicas sobre o *job*: nome do serviço (*Service*), estado (*State*), resultado da execução (*Result*), data/hora de criação do job (*Created*), data/hora de início da execução (*Started*), e data/hora de término da execução (*Finished*).

Em particular, para o campo "estado", os valores possíveis podem ser: pendente ("*PENDING*"), em execução ("*RUNNING*") e finalizado ("*FINISHED*"), enquanto que para o campo "resultado da execução" os valores são: indefinido ("*UNDEFINED*"), sucesso ("*SUCCESS*") e erro ("*ERROR*").

Figura 5.36. Interface para visualização dos resultados da execução de um job.




Fonte: O Autor, 2016.

Além das informações básicas, a interface de visualização dos *jobs* (Figura 56) também apresenta o comando usado na execução (*Command*), a saída padrão recuperada da execução do algoritmo, e os arquivos de entrada e saída

relacionados com o *job*. Neste ponto, é importante lembrar que as informações dos *jobs* (e arquivos) são imutáveis, ou seja, acessíveis apenas para consulta e visualização, respeitando desta forma, a premissa de se manter o histórico de ações sobre as atividades dos usuários. As ações presentes nesta interface consistem na obtenção dos arquivos para a máquina pessoal do usuário ("*download*"), e a visualização dos arquivos através da ferramenta JSmol.

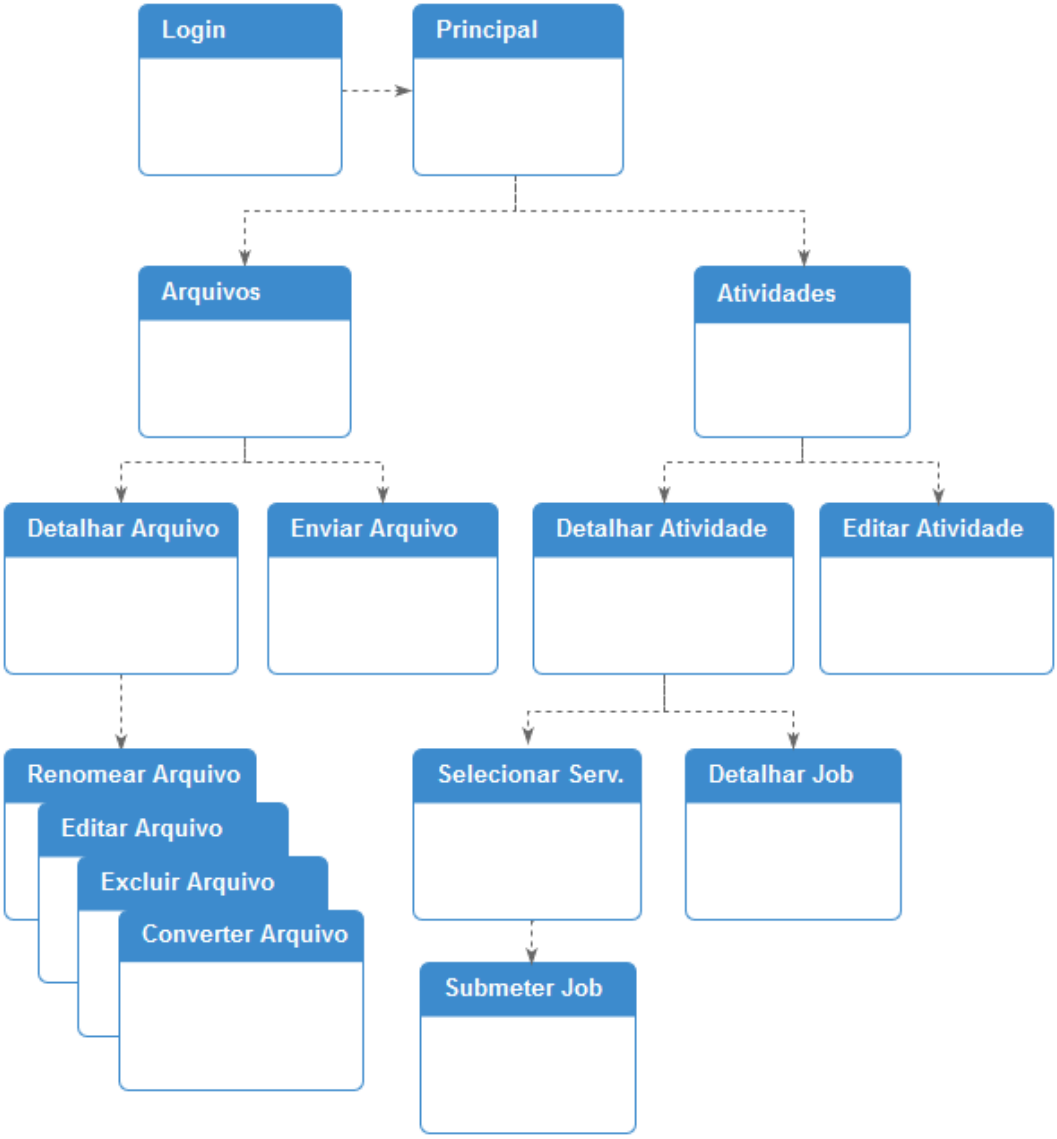
### *5.1.9 Interfaces do sistema para dispositivos móveis*

Conforme já exposto anteriormente, o desenvolvimento do sistema MODiMOL *Workbench* contemplou a elaboração de versões de interfaces que se adequam automaticamente às dimensões das telas dos dispositivos móveis (responsivas). O conjunto de interfaces desenvolvidas, utilizando esta abordagem, abrangeu apenas as funcionalidades diretamente acessadas pelos usuários comuns do sistema, ou seja, com exceção daquelas que possuem natureza administrativa (acessadas exclusivamente pelo usuário administrador do sistema).

Desta forma, as funcionalidades com interfaces responsivas desenvolvidas foram aquelas relacionadas ao gerenciamento de arquivos do usuário, assim como as funcionalidades de gerenciamento de atividades, e criação, submissão e visualização de *jobs*.

A Figura 5.37 apresenta o mapa de navegação da versão móvel do sistema MODiMOL *Workbench*. Ao realizar a autenticação no sistema (Figura 5.38-a), o usuário é direcionado para a interface principal, que permite o acesso às funcionalidades de gerenciamento de arquivos e de atividades (Figura 5.38-b). Ao selecionar uma destas opções, é exibida para o usuário a listagem dos itens da entidade escolhida, ou seja, a listagem dos arquivos existentes da área de armazenamento, no caso de ser selecionada a opção "*Storage*" (Figura 5.39-a), ou a listagem das atividades do usuário (próprias e compartilhadas), no caso da seleção da opção "*Activities*" (Figura 5.39-b).

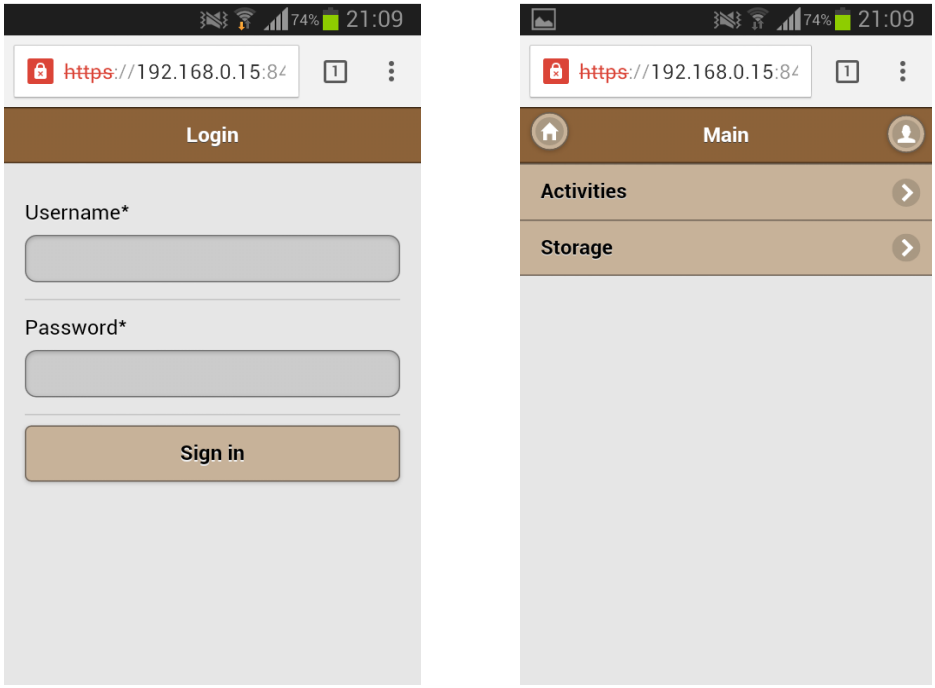
Figura 5.37. Mapa de navegação da versão móvel do sistema MODiMOL Workbench.




Fonte: O Autor, 2016.

Quanto às funcionalidades relacionadas ao gerenciamento de arquivos, é possível realizar as mesmas ações existentes na versão não-responsiva das interfaces (para *desktop*), ou seja, visualização do conteúdo do arquivo (Figura 60-a), edição, exclusão, renomeação, conversão com o OpenBabel, e visualização molecular com JSmol (Figura 5.40-b), além da transferência de arquivos entre o servidor do sistema e a máquina do usuário ("*upload*" e "*download*"). As únicas exceções, fora do escopo de desenvolvimento, são as interfaces para desenho molecular com o editor JSME e para a integração com bancos de dados públicos de estruturas químicas.

Figura 5.38. Interfaces de: a) autenticação do usuário; e, b) tela principal, na versão do sistema para dispositivos móveis.



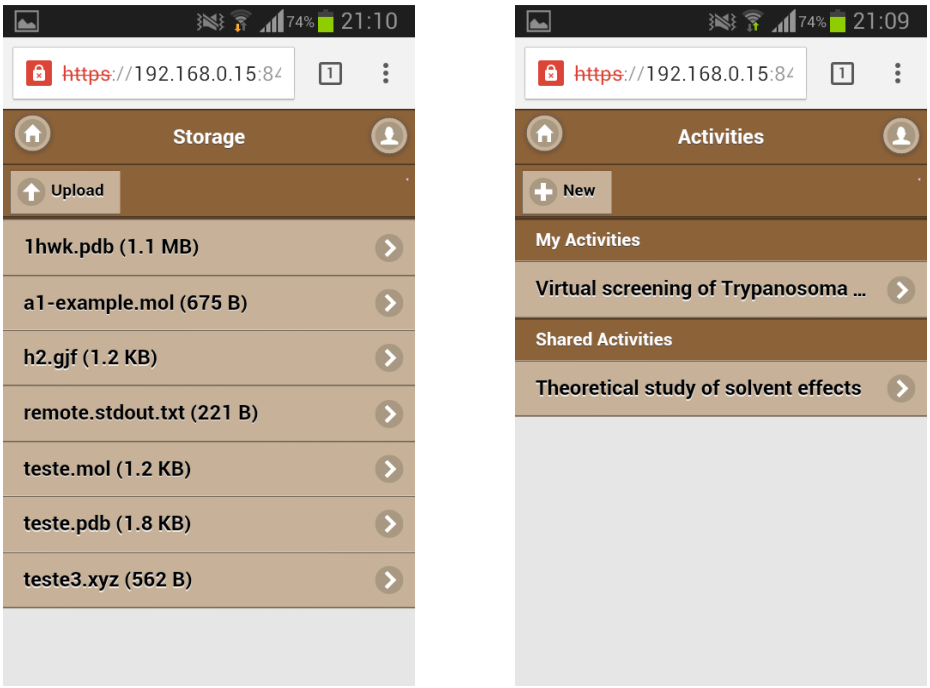

(a) (b)

Fonte: O Autor, 2016.

Quanto às funcionalidades voltadas para o gerenciamento de atividades, foram desenvolvidas interfaces para criação/edição e detalhamento de suas informações (Figura 5.40-c). Além disso, a partir de uma determinada atividade, o usuário pode selecionar um serviço dentre os disponíveis no sistema (Figura 5.40-d), e submetê-lo para execução local ou remota, de acordo com a natureza do serviço selecionado (Figura 5.41-a).

Após sua criação, o *job* pode ser visualizado com as mesmas informações que são apresentadas na interface da versão não-responsiva, ou seja, dados básicos, comando usado na execução, saída padrão e arquivos de entrada e saída relacionados (Figura 5.41-b).

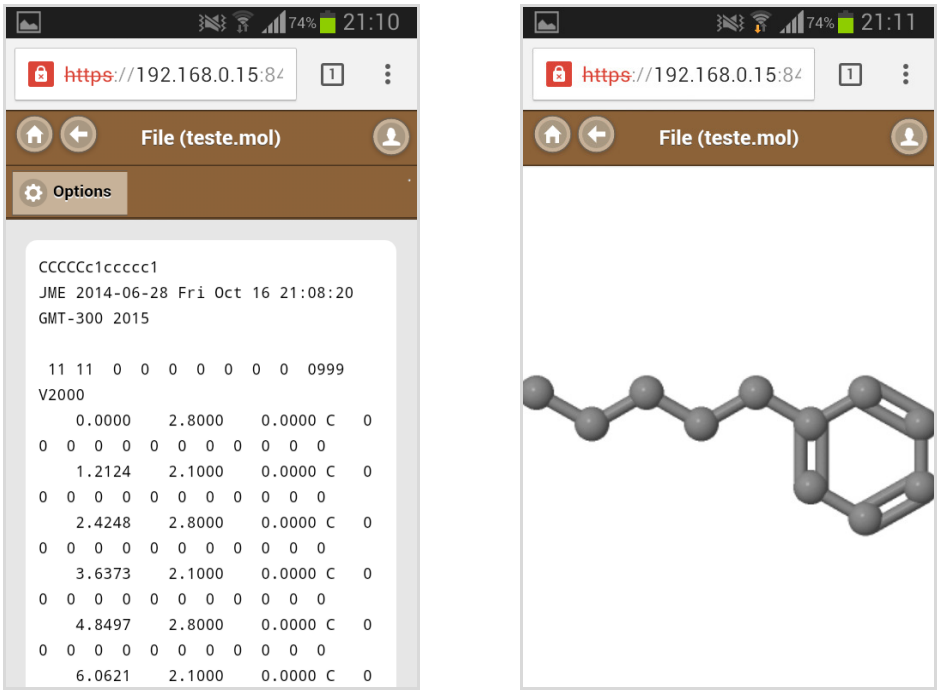
Figura 5.39. Interfaces de: a) visualização de arquivos da área de armazenamento do usuário; e, b) visualização de atividades, na versão do sistema para dispositivos móveis.



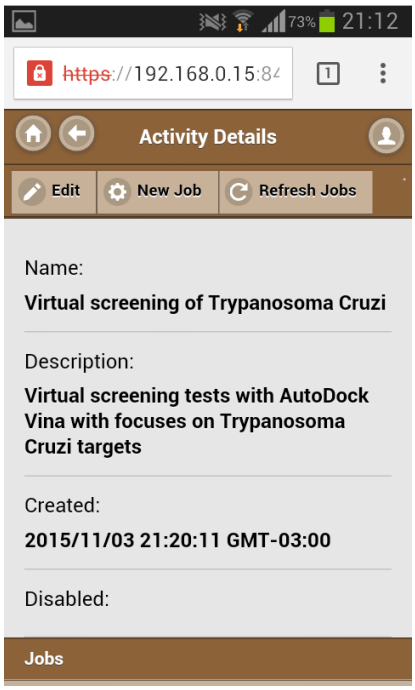

(a) (b)

Fonte: O Autor, 2016.

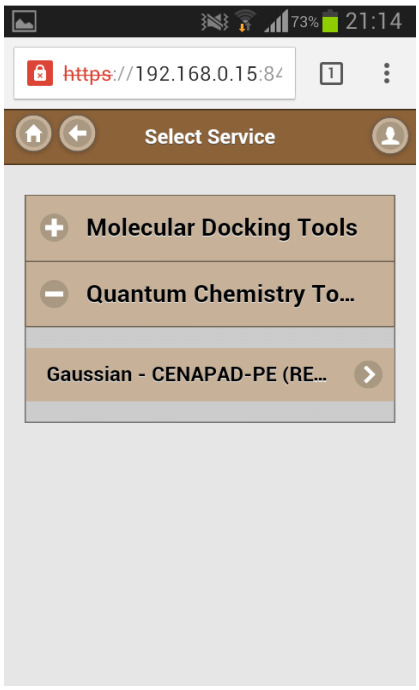
Figura 5.40. Interfaces de: a) visualização de conteúdo de arquivo; b) visualização molecular com JSmol; c) detalhes da atividade; e, d) seleção de serviços, na versão do sistema para dispositivos móveis.




(a) (b)




(c)

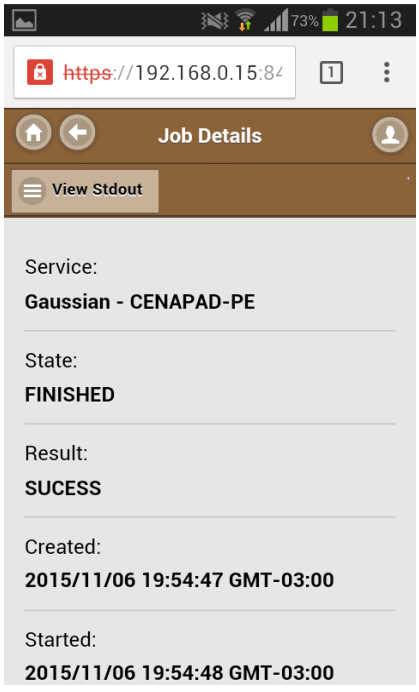



(d)

Fonte: O Autor, 2016.

Figura 5.41. Interfaces de: a) tela de submissão de jobs; e, b) detalhes do job, na versão do sistema para dispositivos móveis.


(a)




(b)

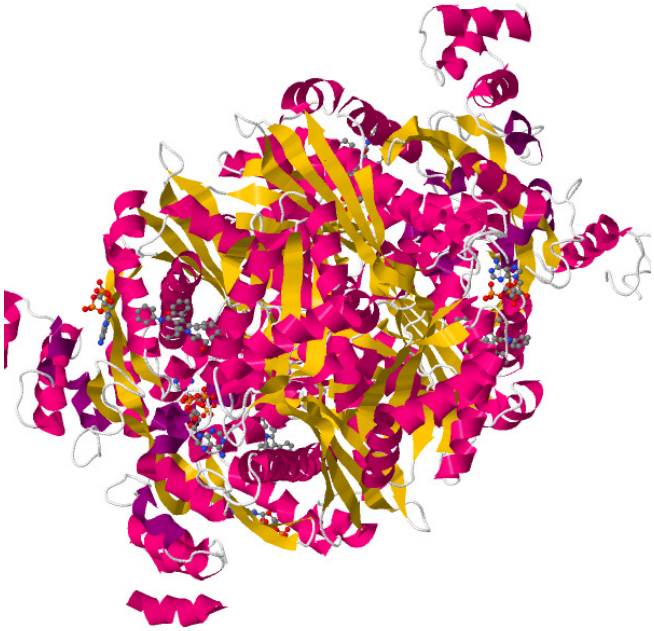
Fonte: O Autor, 2016.

# 6 ESTUDO DE CASO

Neste capítulo, será apresentado um estudo de caso, que tem como objetivo demonstrar a aplicabilidade do sistema MODiMOL *Workbench* para um cenário real de uso. Para esta finalidade, este estudo propõe um exemplo no qual um usuário deseja realizar um cálculo envolvendo a aplicação de métodos de *docking* molecular, através da utilização da ferramenta AutoDock Vina (TROTT e OLSON, 2010), para a molécula de Atorvastatina (ROTH, 2002), que é o princípio ativo de um importante medicamento redutor de colesterol.

Neste exemplo, o objetivo do usuário será utilizar os resultados obtidos com o *docking* molecular do complexo de Atorvastatina (ligante) com HMG-CoA redutase (enzima envolvida na biossíntese do colesterol no corpo humano) (receptor), através do uso do programa AutoDock Vina, para analisar e comparar com o posicionamento referente à molécula de Atorvastatina encontrada na estrutura co-cristalizada do complexo supracitado. Esta estrutura cristalizada, que será usada como referência, pode ser obtida (através de "*download*"), por exemplo, diretamente do banco de dados público PDB (cód:1hwk) (Figura 6.1).

Figura 6.1. Complexo Atorvastatina + HMG-CoA redutase, extraído do banco de dados PDB. Estruturas α-hélice em amarelo e folha-β em magenta.



Fonte: http://www.rcsb.org/pdb/explore/jmol.do?structureId=1HWK, 2016.

Com o propósito de demonstrar um cenário típico de uso do sistema MODiMOL *Workbench*, este exemplo será conduzido levando-se em consideração que o ponto de partida consistirá no usuário não possuir previamente nenhuma informação ou arquivo em sua área de armazenamento, contendo quaisquer dados relacionados ao estudo de caso apresentado neste capítulo. Desta forma, para o cenário deste exemplo, serão realizadas várias etapas que incluem, por exemplo, a obtenção das moléculas utilizadas, edição e conversão dos arquivos, e submissão de *jobs* para a obtenção dos resultados de *docking* molecular.

A Tabela 6.1 apresenta um resumo da sequencia de etapas necessárias para a realização deste estudo de caso. É possível observar nesta tabela que, além da ferramenta AutoDock Vina, também será utilizado o programa de química quântica Gaussian 09 (FRISCH *et al.*, 2009), neste caso, para realizar o procedimento de otimização de geometria da molécula de Atorvastatina (etapa 4), uma vez que sua estrutura química, quando extraída de um banco de dados, como o CACTUS (etapa 2), por exemplo, pode não estar adequada para ser usada diretamente em um cálculo de *docking* molecular, por não refletir os ângulos e distâncias de ligações entre os átomos, mais próximos da realidade. Desta forma, os dois algoritmos (AutoDock Vina e Gaussian) foram usados para exemplificar a submissão de *jobs* a partir do sistema MODiMOL *Workbench*.

Tabela 6.1. *Workflow* contendo as etapas necessárias para realização do estudo de caso apresentado.

| Etapa | Descrição |
|---|---|
| **1** | Criar atividade para organizar os *jobs* criados no estudo de caso. |
| **2** | Obter Atorvastatina do banco de dados CACTUS e salvar em arquivo no formato XYZ. |
| **3** | Converter Atorvastatina para o formato GJF e editar para inclusão dos parâmetros necessários para utilização com o programa Gaussian. |
| **4** | Otimizar geometria da Atorvastatina (GJF) com o programa Gaussian. |

| | |
|---|---|
| **5** | Converter o arquivo de saída do programa Gaussian para o formato PDBQT. |
| **6** | Obter complexo Atorvastatina + HMG-CoA redutase do banco de dados PDB (cód:1hwk) |
| **7** | Editar arquivo para manter apenas estrutura da enzima HMG-CoA redutase (receptor) e converter para o formato PDBQT, usando a ferramenta ADT (AutoDock Tools). |
| **8** | Realizar docking molecular para Atorvastatina (etapa 5) + enzima HMG-CoA redutase (etapa 7) com o programa AutoDock Vina. |
| **9** | Analisar resultado obtido através da comparação do AutoDock Vina com a estrutura encontrada no complexo cristalizado original, usando a ferramenta ADT. |

Fonte: O Autor, 2016.

Com o objetivo de ilustrar o uso de algoritmos locais e remotos neste estudo de caso, cada programa foi cadastrado como um tipo de algoritmo (*software*) no sistema MODiMOL *Workbench*. O AutoDock Vina foi criado como um algoritmo local, enquanto que o Gaussian como um algoritmo remoto. Para ambos os casos, também foram criados os serviços relacionados, respeitando-se a natureza (local ou remota) de cada algoritmo cadastrado. No caso do Gaussian, como algoritmo remoto, é necessário associá-lo a um servidor remoto que contenha este algoritmo instalado e com permissão para execução.

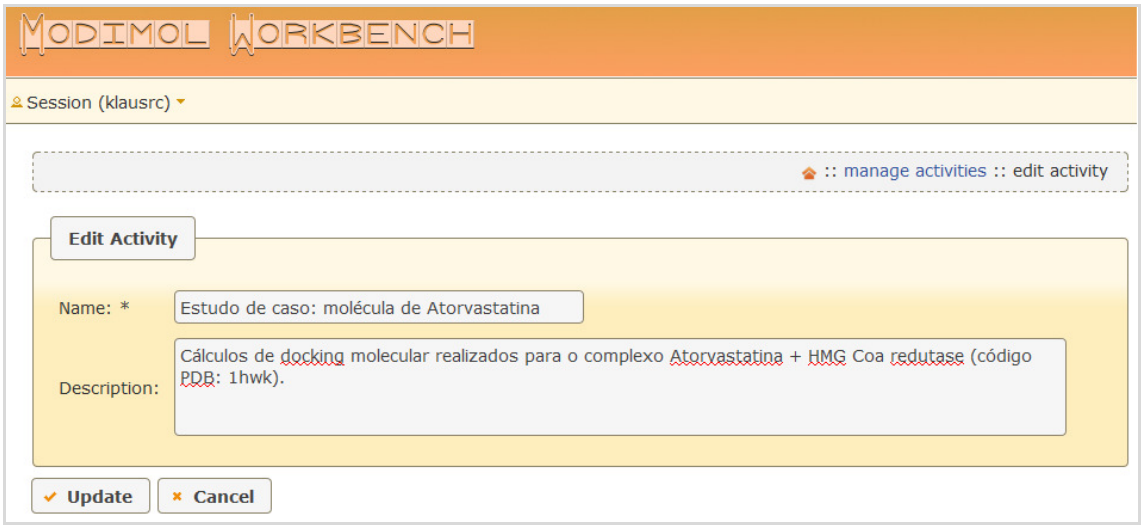
Neste exemplo, foi utilizado o servidor do CENAPAD-PE que possui o programa Gaussian instalado, e será acessado com um usuário específico (nome de usuário e senha previamente cadastrados). O CENAPAD-PE (www.cenapad-pe.ufpe.br), localizado na Universidade Federal de Pernambuco (UFPE), é um dos vários centros de computação de alto desempenho existentes no país, cuja infraestrutura é capaz de atender uma grande demanda computacional proveniente do uso de diversos *softwares* científicos.

Em relação ao ambiente computacional do estudo de caso, utilizou-se uma única máquina, cuja configuração consistiu de um processador Intel Core i3, 4 GB de memória RAM, 250 GB de disco rígido e sistema operacional Linux Ubuntu 14.10,

para servir tanto como servidor do sistema MODiMOL *Workbench* quanto para. o servidor de banco de dados. Nesta máquina foram instalados o servidor Apache Tomcat 7.0.47, o SGBD MySQL Server 5.6 e a máquina virtual Java 8, além da ferramenta OpenBabel 2.3.2.

Conforme apresentado na Tabela 6.1, a primeira etapa do presente estudo de caso consiste na criação da atividade que servirá para organizar os *jobs* submetidos para execução. A Figura 6.2 apresenta a interface de criação da referida atividade para a realização de *docking* molecular com a molécula de Atorvastatina.

Figura 6.2. Interface de criação da atividade no estudo de caso da molécula de Atorvastatina.



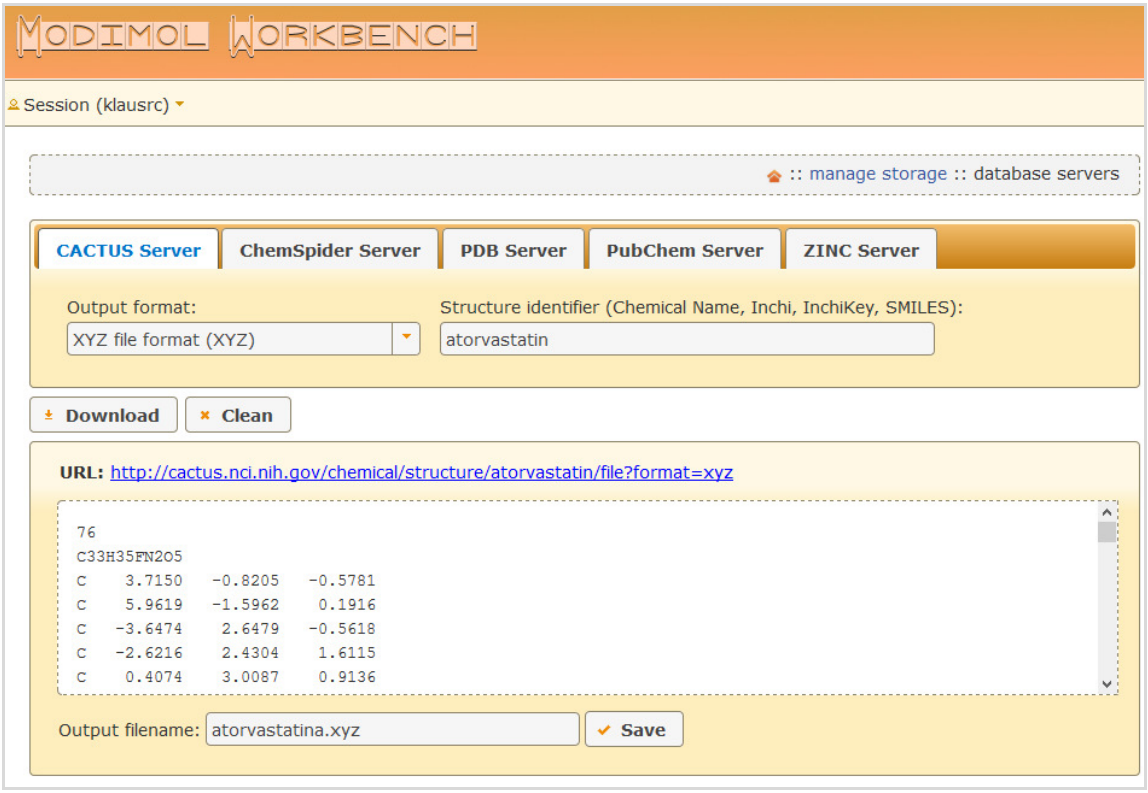

Fonte: O Autor, 2016.

O próximo passo (etapa 2) consiste em obter a molécula de Atorvastatina. Para esta tarefa, o usuário do sistema MODiMOL *Workbench* pode utilizar a interface de integração com bancos de dados externos, e desta forma buscar diretamente pela estrutura da referida molécula. Neste exemplo, foi utilizado o banco de dados CACTUS (Figura 6.3), para a obtenção da estrutura molecular da Atorvastatina e armazenamento do arquivo, contendo seus dados no formato XYZ (atorvastatina.xyz), na área de armazenamento pessoal do usuário.

Após a obtenção da Atorvastatina no banco de dados CACTUS, é necessário convertê-la para o formato GJF, que é aquele utilizado pelo programa Gaussian como formato de entrada de dados (etapa 3). Conforme explicado anteriormente, nesta etapa, aplica-se o programa Gaussian para a realização da otimização de geometria desta molécula, uma vez que sua estrutura, originalmente recuperada do

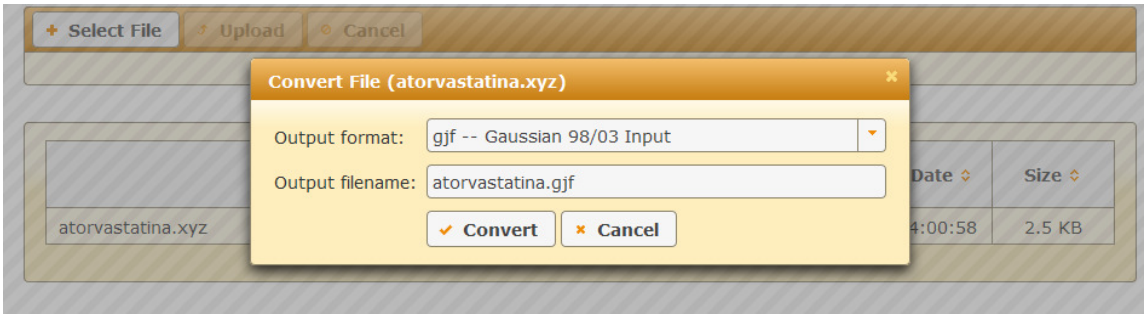
banco de dados CACTUS não se encontra necessariamente preparada para ser usada pelo AutoDock Vina. A Figura 6.4 apresenta a interface utilizada para a conversão da Atorvastatina para o formato GJF (atorvastatina.gjf), reconhecido pelo programa Gaussian.

Figura 6.3. Interface de obtenção da molécula de Atorvastatina a partir do banco de dados CACTUS.




Fonte: O Autor, 2016.

Figura 6.4. Interface para conversão da molécula de Atorvastatina no formato GJF.

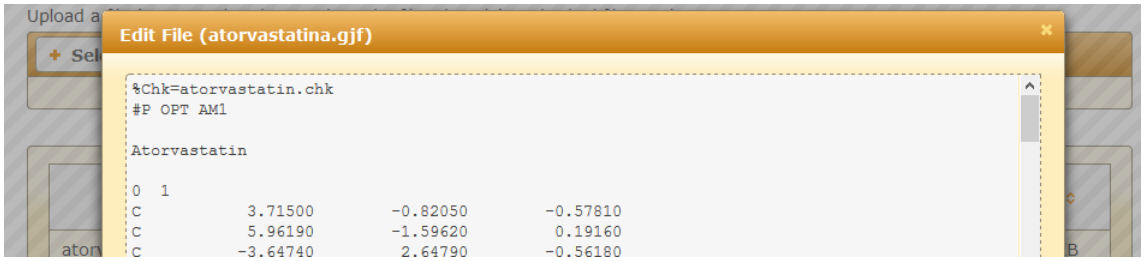



Fonte: O Autor, 2016.

A Figura 6.5 apresenta a edição do arquivo no formato GJF contendo a molécula de Atorvastatina, com o propósito de inclusão das palavras-chave

necessárias para a execução com o programa Gaussian. Neste caso, foram incluídas as palavras-chave “OPT” para realização da otimização de geometria da molécula do arquivo, e “AM1”, que corresponde ao método semi-empírico utilizado (DEWAR et al., 1985).

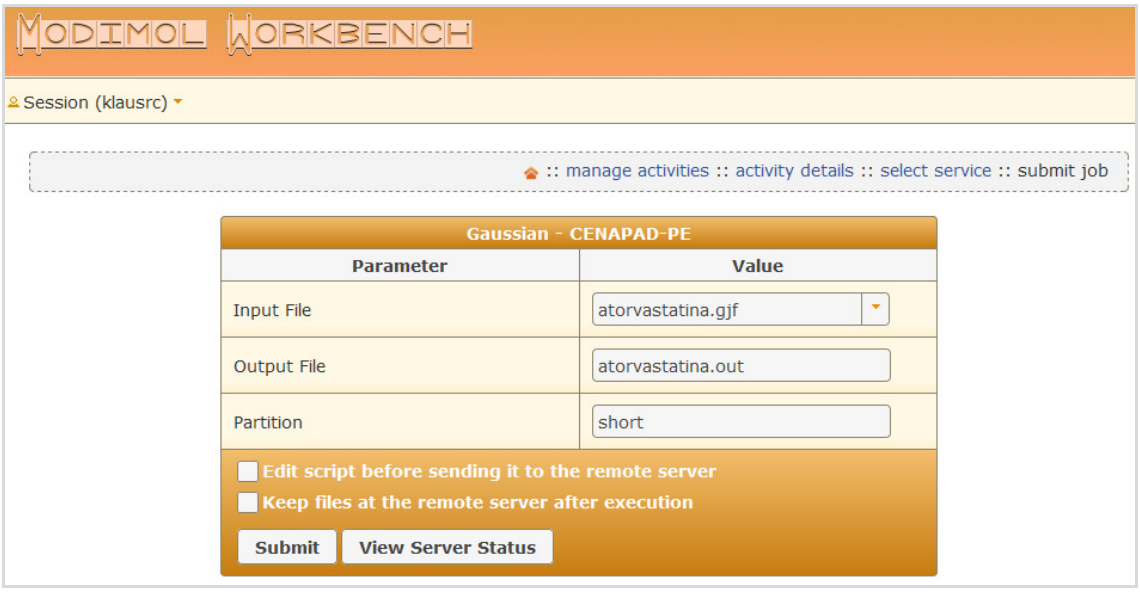
Figura 6.5. Interface de edição do arquivo no formato GJF da molécula de Atorvastatina.




Fonte: O Autor, 2016.

A Figura 6.6 apresenta a interface utilizada para a submissão do *job* remoto correspondente à otimização de geometria da molécula de Atorvastatina, usando o programa Gaussian presente no servidor remoto do CENAPAD-PE (etapa 4). Além do arquivo de entrada (editado na etapa anterior), a submissão do *job* utilizando o Gaussian demanda o nome do arquivo de saída (atorvastatina.out) e a fila (“*short*”) na qual o *job* será executado no ambiente do CENAPAD-PE.

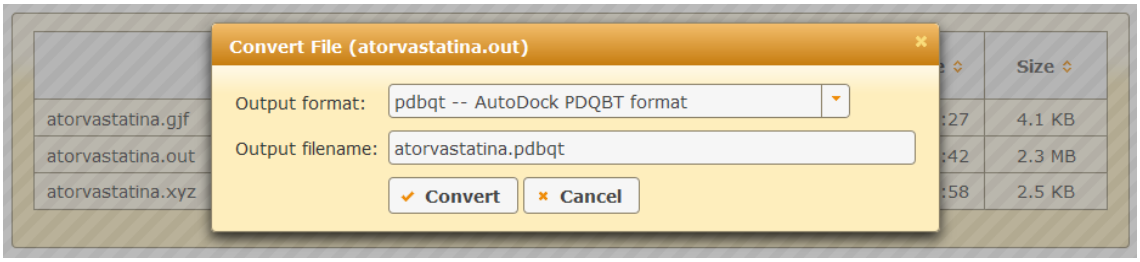
Figura 6.6. Interface para submissão de job remoto usando o programa Gaussian no servidor do CENAPAD-PE.




Fonte: O Autor, 2016.

Em seguida, o arquivo de saída do Gaussian (atorvastatina.out) contendo a molécula com sua geometria otimizada deve ser convertida para o formato PDBQT, reconhecido pelo programa AutoDock Vina (etapa 5). A Figura 6.7 apresenta a interface usada na conversão do arquivo citado para o formato PDBQT (atorvastatina.pdbqt).

Figura 6.7. Interface para conversão do arquivo de saída do Gaussian para o formato PDBQT.



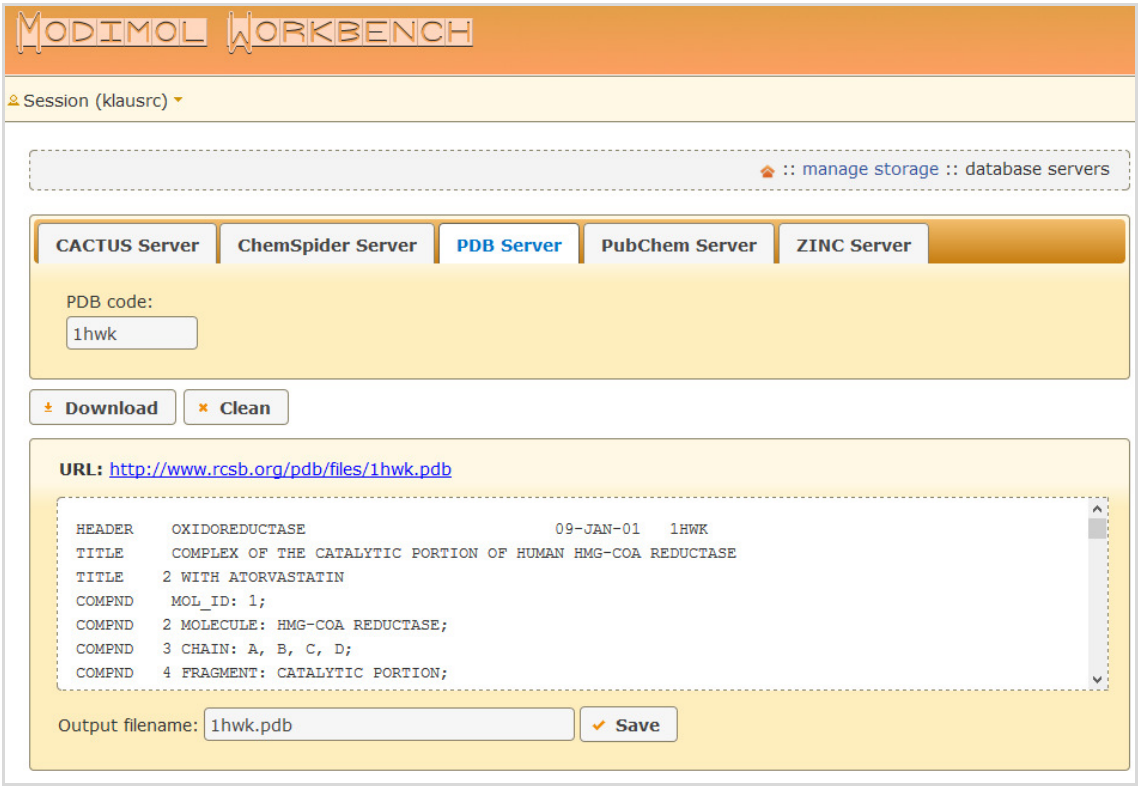

Fonte: O Autor, 2016.

A próximo passo (etapa 6) consiste em obter os dados referentes ao alvo farmacológico, neste caso, a enzima na qual se deseja realizar o cálculo de *docking* juntamente com a molécula de Atorvastatina. Esta tarefa, novamente, pode ser realizada a partir da interface de integração com os bancos de dados externos, neste caso, especificamente com o banco de dados PDB.

A Figura 6.8 apresenta a interface correspondente à integração com este banco de dados e na qual se busca pela estrutura molecular do complexo Atorvastatina + enzima HMG-CoA redutase, cujo código de identificação, é “1hwk”.

Após a obtenção dos dados do complexo e armazenamento na área de dados pessoal do usuário, é necessário editar o conteúdo deste arquivo com o propósito de remover os dados de todos os ligantes presentes (moléculas de Atorvastatina e de ADP - Adenosina Difosfato), mantendo apenas a estrutura da enzima (receptor) no arquivo (etapa 7). Em seguida, este arquivo também deve ser convertido para o formato PDBQT, a fim de ser utilizado como parâmetro de entrada para o programa AutoDock Vina (1hwk.pdbqt). Nesta etapa em particular, a conversão do formato PDB para PDBQT foi realizado no computador local do usuário, utilizando-se a ferramenta ADT (AutoDock Tools) (MORRIS, *et al.*, 2009), sendo esta a principal interface gráfica recomendada para a preparação de arquivos e análise dos resultados com o programa AutoDock Vina.

Figura 6.8. Interface de obtenção do complexo Atorvastatina + enzima HMG-CoA redutase a partir do banco de dados PDB (cód:1hwk).

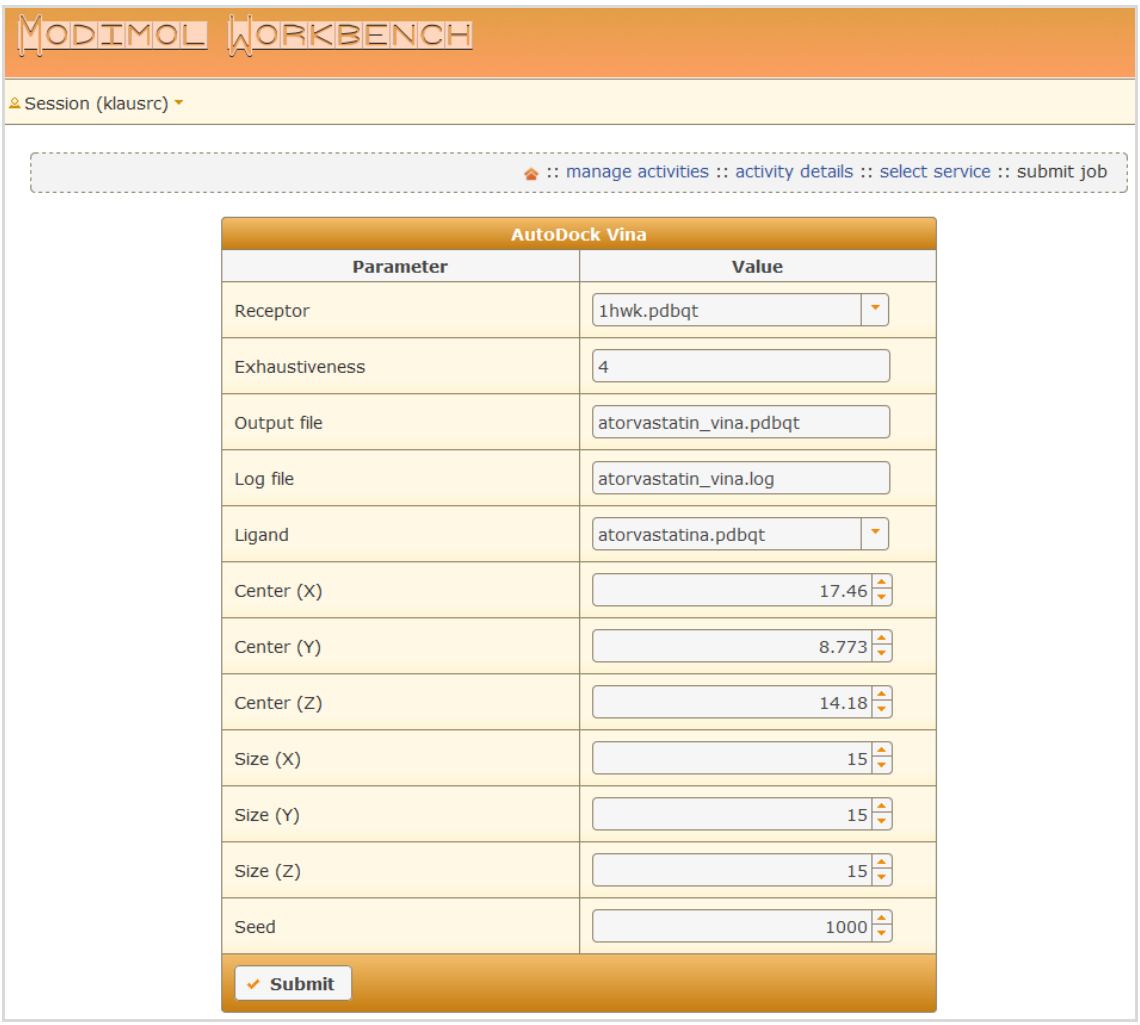



Fonte: O Autor, 2016.

A Figura 6.9 apresenta a interface de submissão do *job* local correspondente ao cálculo de *docking* molecular entre a molécula de Atorvastatina (*Ligand*) e a enzima HMG-CoA redutase (*Receptor*), usando o programa AutoDock Vina (etapa 8).

Este serviço requer vários parâmetros de entrada: arquivo de saída (*Output file*), arquivo para geração de *log* (*Log file*), semente para geração de números pseudo-aleatórios (*Seed*) necessários para a execução do algoritmo principal de busca do programa, nível de precisão da busca por soluções de docking (*Exhaustiveness*), e informações a respeito do sítio ativo da enzima, ou seja, o local esperado para acontecer o "*binding*" entre o ligante (Atorvastatina) e o receptor (enzima). Estas informações expressam os parâmetros que definem a caixa no espaço tridimensional no qual está situado o sítio ativo da enzima, isto é, o centro da caixa (*Center X, Y* e *Z*) e o tamanho de suas arestas (*Size X, Y* e *Z*) para cada um dos eixos cartesianos, em Ångstroms.

Figura 6.9. Interface de submissão de job local usando o programa AutoDock Vina.



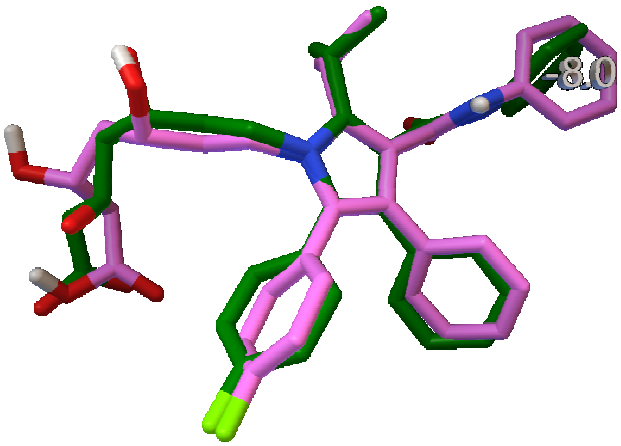

Fonte: O Autor, 2016.

O último passo (etapa 9) consiste na análise do resultado obtido após a execução do programa AutoDock Vina (atorvastatina_vina.pdbqt), em comparação com os dados extraídos da molécula co-cristalizada com a enzima HMG-CoA redutase, proveniente do arquivo obtido a partir do banco de dados PDB (1hwk.pdb). Com o objetivo de analisar os dados finais obtidos neste estudo de caso, o usuário deve realizar a cópia ("*download*") dos arquivos (atorvastatina_vina.pdbqt e 1hwk.pdb) para sua máquina pessoal e usar uma ferramenta de visualização molecular para comparar os resultados.

A Figura 6.10, gerada a partir da ferramenta ADT (AutoDock Tools), exibe a comparação entre o resultado obtido com o *docking* molecular para a Atorvastatina, e o posicionamento da mesma molécula retirado da estrutura co-cristalizada da

enzima extraída do arquivo PDB, confirmando que o resultado obtido pelo método *in silico* é muito parecido com o encontrado no cristal. Para este estudo de caso, em particular, a energia de ligação do complexo mais estável foi de -8.0 Kcal/mol.

Figura 6.10. Comparação do resultado obtido no docking molecular da Atorvastatina usando o programa AutoDock Vina (cor magenta) com a estrutura extraída do complexo cristalizado, sob código PDB:1HWK (cor verde).




Fonte: O Autor, 2016.

# 7 CONCLUSÕES

Neste trabalho foi apresentado o desenvolvimento do sistema MODiMOL *Workbench*, um portal científico projetado para servir como plataforma de execução de serviços de modelagem molecular com foco na inovação terapêutica. O objetivo central do projeto consiste em tornar a experiência do usuário com este tipo de sistema computacional mais fácil e intuitiva, além de facilitar as tarefas relacionadas à administração do sistema.

Estas premissas foram tratadas por meio de alguns conceitos-chave. *Jobs* gerados a partir de uma mesma atividade (pesquisa) científica estão logicamente associados entre si, e portanto, devem ser mantidos (organizados) juntos em atividades. Agrupar *jobs* sob uma mesma "atividade", torna mais fácil a tarefa de acessar e visualizar resultados históricos de uma mesma atividade de pesquisa e melhora o nível de compreensão e organização sobre os dados gerados pelos usuários. Em segundo lugar, foi desenvolvida uma interface Web para disponibilizar o acesso do usuário ao sistema MODiMOL *Workbench*. Esta decisão foi motivada pelo fato do uso deste tipo de abordagem permitir que o sistema possa ser acessado através de um navegador *Web* (Ex.: Firefox, Internet Explorer e Chrome) de qualquer lugar, através da Internet. Outro importante conceito consiste na inclusão dinâmica de novos algoritmos (aplicações científicas) sem a necessidade de reinicialização do sistema, ou recompilação do código-fonte e geração de uma nova versão compilada do sistema.

No sistema MODiMOL *Workbench*, *jobs* podem ser submetidos para ambientes externos de computação de alto desempenho, a fim de utilizar seus recursos computacionais disponíveis para realizar tarefas (cálculos) complexas e de alta demanda computacional, trazendo com isso, maior flexibilidade e aplicabilidade ao sistema.

Com o propósito de demonstrar um cenário típico de uso do sistema MODiMOL Workbench, foi realizado um estudo de caso no qual envolveu a aplicação de química quântica para otimização de geometria (Gaussian), e métodos de *docking* molecular (AutoDock Vina), para a molécula de Atorvastatina, princípio ativo de um medicamento redutor de colesterol. Este estudo de caso foi conduzido em várias etapas que incluíram, por exemplo, a obtenção das moléculas utilizadas,

edição e conversão dos arquivos, e submissão de *jobs* para a otimização de geometria e para a obtenção dos resultados de *docking* molecular.

Outro ponto relevante, obtido no desenvolvimento desta plataforma, consistiu no melhor gerenciamento dos dados resultantes da execução dos serviços, fornecendo ao usuário do sistema, a capacidade de recuperação das informações históricas de *jobs* processados, em qualquer momento e lugar, devido à facilidade de acesso ao sistema através de uma interface Web. Também foi contemplada a disponibilização de interfaces responsivas para acesso ao sistema a partir de dispositivos móveis (*smartphones* e *tablets*), tornando o sistema mais acessível e preparado para estas novas plataformas. Desta forma, o desenvolvimento de futuras soluções computacionais para estes tipos de aplicações devem almejar a inclusão de versões projetadas especificamente para estes tipos de dispositivos.

Assim, pode-se concluir que o esforço de desenvolvimento do sistema MODiMOL *Workbench*, permitiu disponibilizar um portal científico, onde ferramentas (serviços) de modelagem molecular podem ser utilizadas de modo mais ágil e intuitivo, principalmente por parte de usuários mais principiantes na área.

# 8 PERSPECTIVAS

As perspectivas para futuro trabalho e desenvolvimento de novas versões do sistema MODiMOL *Workbench* incluem, mas não se limitam a, novas funcionalidades e melhorias para aquelas existentes, como, por exemplo:

*7.1 Atividades*. O comportamento natural para uma atividade é manter um grande número de *jobs* no início de seu tempo de vida, e este valor gradualmente decrescer até alcançar uma situação na qual a atividade se torna menos ativa e já não exista mais necessidade de submeter mais *jobs* para execução (quando o objetivo da atividade é atingida, ou a atividade é abandonada, por exemplo). Neste cenário, a liberação de espaço da área de armazenamento pessoal do usuário poderia ser realizada através de uma funcionalidade para arquivar e exportar atividades "inativas" para a máquina pessoal do usuário. Além disso, outra proposta interessante neste contexto, consiste na geração de um tipo de relatório contendo a sequência histórica dos *jobs* processados de uma determinada atividade. Neste caso, o relatório seria similar ao conceito de "caderno de laboratório" ou "registro científico", ou seja, um documento formal que poderia ser utilizado para apresentar e confirmar os resultados alcançados pela realização de um experimento (incluindo os "*in silico*"), em particular.

*7.2 Grupos de trabalho*. A inclusão de grupos de trabalho (*workgroups*), como conjunto de usuários com objetivos comuns e/ou pertencentes a mesma organização, simplificaria e facilitaria a colaboração entre as pessoas envolvidas em uma determinada atividade científica. Por exemplo, uma atividade (com informações relevantes de interesse comum a diversas pessoas) poderia ser compartilhada para uma equipe inteira rapidamente através de uma simples operação, ao invés de ser necessário compartilhá-la individualmente para cada um dos membros do time, como já está implementado na versão atual do sistema.

*7.3 Mensagens*. Um sistema de mensagens facilitaria a comunicação entre os usuários do sistema, assim como, também poderia ser usado como um mecanismo de envio de alertas para informar os usuários sobre certos eventos do sistema, tais como, mudança de estado dos *jobs* (em execução, sucesso, erro), situação de cota

de armazenamento de usuário prestes a se esgotar ou, tempo remanescente para a senha do usuário expirar, por exemplo.

*7.4 Estatísticas*. Coletar e apresentar informações relativas ao uso do sistema e dos recursos computacionais existentes. Poderiam ser coletados dados como tempo de CPU utilizado (total, por atividade, por usuário, etc.), espaço de disco utilizado (total, por usuário, etc.), *jobs* submetidos, número de usuários e suas respectivas localizações geográficas (através das coordenadas armazenadas), dentre outras.

*7.5 Execução remota*. Fornecer um controle mais detalhado para monitorar, realizar ações e consultar informações detalhadas sobre as execuções remotas dos *jobs*. Adicionalmente, também sugere-se incluir suporte para outros escalonadores de recursos bem difundidos e utilizados para esta finalidade, além do SLURM, como por exemplo o LoadLeveler (SKOVIRA *et al.*, 1996), PBSPro (NITZBERG *et al*, 2004) e TORQUE (STAPLES, 2006), a fim de tornar o sistema mais flexível para integração com outros ambientes remotos de HPC.

*7.6 Bancos de dados externos*. Ampliar a integração do sistema com outros bancos de dados públicos de estruturas moleculares, tais como os serviços ChEBI (CHEBI, 2016), e SwissDock (GROSDIDIER *et al.*, 2011), sendo este último, mantenedor de uma extensa base de dados de proteínas e estruturas de moléculas com potencial de serem usadas em aplicações de *virtual screening*, ou em outros estudos de modelagem molecular.

*7.7 Usuários*. Tornar o gerenciamento de usuários mais completo e otimizado, através da inclusão de alguns recursos essenciais ao sistema, como por exemplo, mecanismo de recuperação de senha (através de e-mail de notificação com *link* de confirmação), e formulário de solicitação de novos usuários do sistema.

7.8 *Segurança*. Aplicar novas medidas visando uma maior proteção do sistema, principalmente no que se refere ao controle de acesso, e integridade e privacidade dos dados dos usuários. Para tal, pretende-se implementar, autenticação de usuário através de certificado digital, mecanismo anti-robô (*captcha*) nas interfaces públicas

(páginas de autenticação, recuperação de senha, solicitação de novos usuários, etc.), além de outras regras de segurança, dispostas pela OWASP, para aplicação no desenvolvimento de sistemas computacionais. A OWASP é uma organização sem fins lucrativos formada por pessoas e instituições, com foco no desenvolvimento e evolução de novas metodologias, ferramentas e tecnologias para a segurança de *softwares* em geral (OWASP, 2016).

*7.9 Testes*. Com a perspectiva do desenvolvimento de novas versões da plataforma MODiMOL *Workbench*, existe a perspectiva de inclusão de algumas metodologias de testes dentro do processo de construção do sistema, para garantir que as próximas versões sejam cada vez mais robustas e menos suscetíveis a falhas. Tais metodologias incluem a utilização de testes para trechos do código-fonte do sistema que possuem alta complexidade (testes unitários), bem como ferramentas capazes de testar, de forma automatizada (sem intervenção do usuário), as interfaces do sistema, como é o caso, por exemplo, das ferramentas Selenium (SELENIUM, 2016) e Demoiselle Behave (DBEHAVE, 2016). Além disso, deseja-se utilizar a ferramenta JMeter (JMETER, 2016) para realizar testes de carga, que são fundamentais para detectar limitações e potenciais “gargalos” de desempenho do sistema, e desta forma, ser capaz de propor manutenções evolutivas adequadas.

*7.10 Workflows*. Implementar infraestrutura necessária para editar/salvar *workflows*, através de uma interface gráfica intuitiva, e mecanismo de execução própria (interno ao sistema), ou por meio da adoção de ferramentas bem conhecidas e difundidas com esta finalidade, como é o caso do Kepler (LUDÄSCHER *et al*., 2006) e Taverna (WOLSTENCROFT *et al*., 2013).

*7.11 Tecnologias*. Pesquisar outras tecnologias e arcabouços (*frameworks*) referentes à programação de sistemas com o objetivo de desenvolver novas versões do sistema MODiMOL *Workbench* com tecnologias emergentes. Alguns exemplos bastante difundidos e utilizados de arcabouços incluem o Demoiselle (DEMOISELLE, 2016), Grails (GRAILS, 2016) e Play (PLAY, 2016), que possuem diversos recursos avançados que facilitam o desenvolvimento de sistemas baseados na tecnologia Java. Outra perspectiva pretendida consiste no estudo de diferentes

padrões arquiteturais de sistemas, em especial, com foco na arquitetura REST, cuja filosofia consiste em sistemas nos quais o servidor (*back-end*) fornece um conjunto de serviços diretamente sobre o protocolo de comunicação HTTP, e a interface do sistema (*front-end*) é desenvolvida de forma desacoplada e menos dependente do servidor, através de tecnologias baseadas tipicamente em HTML, CSS e JavaScript. Atualmente, existem diversos arcabouços para desenvolvimento de interfaces de sistemas utilizando estas tecnologias, como por exemplo, o Bootstrap (BOOTSTRAP, 2016), AngularJS (ANGULARJS, 2016) e IONIC (IONIC, 2016).

# REFERÊNCIAS

ALLAND, C.; MOREEWS, F.; BOENS, D.; CARPENTIER, M.; CHIUSA, S.; LONQUETY, M.; RENAULT, N.; WONG, Y.; CANTALLOUBE, H.; CHOMILIER, J.; HOCHEZ, J.; POTHIER, J.; VILLOUTREIX, B. O.; ZAGURY, J. F.; TUFFÉRY, P. RPBS: a web resource for structural bioinformatics. **Nucleic Acids Research**, 33, 44-49, 2005.

ALTINTAS, I.; BERKLEY, C.; JAEGER, E.; JONES, M.; LUDÄSCHER, B.; MOCK, S. Kepler: An Extensible System for Design and Execution of Scientific Workflows. Proceedings of the The Future of Grid Data Environments, **Global Grid Forum** 10, 2004.

ALTSCHUL, S.; GISH, W.; MILLER, W.; MYERS, E.; LIPMAN, D. Basic local alignment search tool. **Journal of Molecular Biology**, 215(3), 403–410, 1990. doi:10.1016/ S0022-2836(05)80360-2

ANGULARJS, **AngularJS**, Google Inc., 2016. Disponível em: <https://angularjs. org/>. Acesso em: 10 fev. 2016.

BALASKO, A.; FARKAS, Z.; KACSUK, P. Building Science Gateways by Utilizing the Generic WS-PGRADE/gUSE Workflow System. **Computer Science**, 14, 2, 307, 2013. doi:10.7494/csci.2013.14.2.307.

BARKER, A.; VAN HEMERT, J. (2008), Scientific Workflow: A Survey and Research Directions, **Parallel Processing and Applied Mathematics**, 7th International Conference, PPAM 2007, doi:10.1007/978-3-540-68111-3_78.

BLANKENBERG, D.; VON KUSTER, G.; CORAOR, N.; ANANDA, G.; LAZARUS, R.; MANGAN, M.; NEKRUTENKO, A.; TAYLOR, J. Galaxy: a web-based genome analysis tool for experimentalists. **Current Protocols in Molecular Biology**, 19, 19.10, 1-21, 2010. doi:10.1002/0471142727.mb1910s89

BEISKEN, S.; MEINL, T.; WISWEDEL, B.; FIGUEIREDO, L. F.; BERTHOLD, M.; STEINBECK, C. KNIME-CDK: Workflow-driven cheminformatics. **BMC Bioinformatics**, 14, 257, 2013. doi:10.1186/1471-2105-14-257

BELHAJJAME, K.; WOLSTENCROFT, K.; CORCHO, O.; OINN, T.; TANOH, F.; WILLIAM, A.; GOBLE, C. Metadata Management in the Taverna Workflow System. **Cluster Computing and the Grid**, CCGRID08, Lyon, France. 2008. doi:10.1109/CCGRID.2008.17

BERTHOLD, M. R.; CEBRON, N.; DILL, F.; GABRIEL, T. R.; KÖTTER, T.; MEINL, T.; OHL, P.; SIEB, C.; THIEL, K.; WISWEDEL, B. KNIME: The Konstanz Information Miner. Studies in Classification, **Data Analysis, and Knowledge Organization**, 319-326, 2008. doi:10.1007/978-3-540-78246-9_38

BHAGAT, J., TANOH, F., NZUOBONTANE, E., LAURENT, T., ORLOWSKI, J., ROOS, M., WOLSTENCROFT, K., ALEKSEJEVS, S., STEVENS, R., PETTIFER, S., LOPEZ, R., GOBLE, C.A. BioCatalogue: a universal catalogue of web services for the life sciences, **Nucleic Acids Research**, 38, 689-694, 2010. doi:10.1093/nar/gkq394

BIENFAIT, B. e ERTL, P. JSME: a free molecule editor in JavaScript. **Journal of Cheminformatics**, 5:24, 2013. doi:10.1186/1758-2946-5-24

BOHLE, S. What is E-science and How Should it Be Managed?, 2014. **Nature.com**. Disponível em: <http://www.scilogs.com/scientific_and_medical_ libraries/what-is-e-science-and-how-should-it-be-managed/>. Acesso em: 11 jan. 2014.

BOOTSTRAP, **Bootstrap Project**, 2016. Disponível em: <http://getbootstrap. com/>. Acesso em: 10 fev. 2016.

BUYYA, R. & VENUGOPAL, S. A Gentle Introduction to Grid Computing and Technologies. **CSI Communications**, 29(1):9-19, 2005.

CACTUS. **CADD Group Chemoinformatics Tools and User Services**, National Cancer Institute, 2014. Disponível em: <http://cactus.nci.nih.gov/chemical/ structure>. Acesso em: 08 fev. 2014.

CHEBI. **Chemical Entities of Biological Interest. European Molecular Biology Laboratory**, 2016. Disponível em: <http://www.ebi.ac.uk/chebi/>. Acesso em: 10 fev. 2016.

CHEMSPIDER. **Royal Society of Chemistry**, 2014. Disponível em: <http://www.chemspider.com/>. Acesso em: 08 fev. 2014.

CURCIN, V.; GHANEM, M. Scientific workflow systems – can one size fit all?, **Biomedical Engineering Conference**, 2008. doi:10.1109/CIBEC.2008.4786077.

D-GRID. **D-Grid Development and Operating Company (D-Grid GmbH)**, 2014. Disponível em <http://www.d-grid-gmbh.de/>. Acesso em: 11 jan. 2014.

DALBY, A.; NOURSE, J. G.; HOUNSHELL, W. D.; GUSHURST, A. K. I.; GRIER, D. L.; LELAND, B. A.; LAUFER, J. Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited. **Journal of Chemical Information and Computer Science**., 32 (3), 244–255, 1992. doi:10.1021/ci00007a012

DALKAS, G. A.; VLACHAKIS, D.; TSAGKRASOULIS, D.; KASTANIA, A.; KOSSIDA, S. State-of-the-art technology in modern computer-aided drug design. **Briefings in Bioinformatics** 14(6):745-752, 2012. doi:10.1093/bib/bbs063

DAQUINO, F. **O que está acontecendo com o mercado de PCs?**, TecMundo, 2013. Disponível em: <http://www.tecmundo.com.br/computador-desktop/39009 -o-que-esta-acontecendo-com-o-mercado-de-pcs-.htm>. Acesso em: 28 jan 2016.

DBEHAVE. **Demoiselle Behave**. Serviço Federal de Processamento de Dados - SERPRO, 2016. Disponível em <http://dbehave.com/>. Acesso em: 10 fev. 2016.

DEMOISELLE. **Framework Demoiselle**. Serviço Federal de Processamento de Dados - SERPRO, 2016. Disponível em <https://www.frameworkdemoiselle.gov.br/>. Acesso em: 10 fev. 2016.

DEWAR, M. J. S.; ZOEBISCH, E. G.; HEALY, E. F.; STEWART, J. J. P. (1985). Development and use of quantum mechanical molecular models. 76. AM1: A new general purpose quantum mechanical molecular model. **Journal of the American Chemical Society** 107 (13): 3902. doi:10.1021/ja00299a024

EGI. **European Grid Infrastructure**, 2014. Disponível em <http://www.egi.eu/>. Acesso em: 11 jan. 2014.

FELDT, J.; MATA, R. A.; DIETERICH, J. M. Atomdroid: A Computational Chemistry Tool for Mobile Platforms. **Journal of Chemical Information and Modeling**, 52(4), 1072-1078, 2012.

FOSTER, I. Service-Oriented Science. **Science**, 308(5723), 814-817, 2005. doi:10.1126/science.1110411.

_____. What is the Grid? A Three Point Checklist. **GRIDToday**, July 20, 2002.

FOSTER, I. & KESSELMAN, C. Globus: A Metacomputing Infrastructure Toolkit. International **Journal of Supercomputer Applications**, 11(2):115-128, 1997. doi:10.1177/109434209701100205

FOSTER, I.; KESSELMAN, C.; TUECKE, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International **J. Supercomputer Applications**, 15(3), 2001.

FOSTER, I.; KESSELMAN, C.; NICK, J. & TUECKE, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Open Grid Service Infrastructure WG, **Global Grid Forum**, Edinburgh, Scotland, 2002.

FRAGA, R. **Venda de smartphones ultrapassa a de PCs pela primeira vez**, TechTudo, 2012. Disponível em <http://www.techtudo.com.br/noticias/noticia/2012/02/venda-de-smartphones-ultrapassa-de-pcs-pela-primeira-vez.html>. Acesso em: 28 jan 2016.

FRISCH, M. J.; TRUCKS, G. W.; SCHLEGEL, H. B.; SCUSERIA, G. E.; ROBB, M. A.; CHEESEMAN, J. R.; SCALMANI, G.; BARONE, V.; MENNUCCI, B.; PETERSSON, G. A.; NAKATSUJI, H.; CARICATO, M.; LI, X.; HRATCHIAN, H. P.; IZMAYLOV, A. F.; BLOINO, J.; ZHENG, G.; SONNENBERG, J. L.; HADA, M.; EHARA, M.; TOYOTA, K.; FUKUDA, R.; HASEGAWA, J.; ISHIDA, M.; NAKAJIMA, T.; HONDA, Y.; KITAO, O.; NAKAI, H.; VREVEN, T.; MONTGOMERY, J. A., JR.; PERALTA, J. E.; OGLIARO, F.; BEARPARK, M.; HEYD, J. J.; BROTHERS, E.; KUDIN, K. N.; STAROVEROV, V. N.; KOBAYASHI, R.; NORMAND, J.; RAGHAVACHARI, K.; RENDELL, A.; BURANT, J. C.; IYENGAR, S. S.; TOMASI, J.; COSSI, M.; REGA, N.; MILLAM, J. M.; KLENE, M.; KNOX, J. E.; CROSS, J. B.;

BAKKEN, V.; ADAMO, C.; JARAMILLO, J.; GOMPERTS, R.; STRATMANN, R. E.; YAZYEV, O.; AUSTIN, A. J.; CAMMI, R.; POMELLI, C.; OCHTERSKI, J. W.; MARTIN, R. L.; MOROKUMA, K.; ZAKRZEWSKI, V. G.; VOTH, G. A.; SALVADOR, P.; DANNENBERG, J. J.; DAPPRICH, S.; DANIELS, A. D.; FARKAS, Ö.; FORESMAN, J. B.; ORTIZ, J. V.; CIOSLOWSKI, J.; FOX, D. J. Gaussian, Inc., Wallingford CT, 2009.

GELDENHUYS, W. J., GAASCH, K. E., WATSON, M., ALLEN, D. D., Van DER SCHYF, C. J. Optimizing the use of open-source software applications in drug discovery. **Drug Discovery Today**, 11, 3-4, 127-132, 2006. doi:10.1016/S1359-6446(05)03692-5.

GIARDINE, B.; RIEMER, C.; HARDISON, R. C.; BURHANS, R.; ELNITSKI, L.; SHAH, P.; ZHANG, Y.; BLANKENBERG, D.; ALBERT, I.; TAYLOR, J.; MILLER, W.; KENT, W. J.; NEKRUTENKO, A. Galaxy: a platform for interactive large-scale genome analysis. **Genome Research**. 15(10), 1451-5, 2005. doi:10.1101/gr.4086505

GOECKS, J.; NEKRUTENKO, A.; TAYLOR, J.; and the GALAXY TEAM. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. **Genome Biology**, 11(8), 86, 2010. doi:10.1186/gb-2010-11-8-r86

GOMES A. T. A.; BASTOS B. F.; MEDEIROS V. e MOREIRA V. M. Experiences of the Brazilian national high-performance computing network on the rapid prototyping of science gateways, **Concurrency Computat.: Pract. Exper.**, 27:271-289, 2015. doi:10.1002/cpe.3258

GORDON, P. M. K.; SENSEN, C. W. A Pilot Study into the Usability of a Scientific Workflow Construction Tool, Technical Report #2007-874-26, Department of Computer Science, University of Calgary, Canada, 2007.

GRAILS. **Grails Project**, 2016. Disponível em <https://grails.org/>. Acesso em: 10 fev. 2016.

GROSDIDIER, A.; ZOETE, V.; MICHIELIN, O. SwissDock, a protein-small molecule docking web service based on EADock DSS. **Nucleic Acids Res.**, :W270-7, 2011. doi:10.1093/nar/gkr366

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The WEKA Data Mining Software: An Update; **SIGKDD Explorations**, 11, 1, 2009. doi:10.1145/1656274.1656278

HELLER, S.; MCNAUGHT, A.; STEIN, S.; TCHEKHOVSKOI, D.; PLETNEV, I. InChI - the worldwide chemical structure identifier standard. **Journal of Cheminformatics**, 5, 7, 2013. doi:10.1186/1758-2946-5-7

HEY, T.; TREFETHEN, A. E. Cyberinfrastructure for e-Science. **Science**, 308(5723), 817-21, 2005. doi:10.1126/science.1110410.

HUANG, H. J.; YU, H, W.; CHEN, C. Y.; HSU, C. H.; CHEN, H. Y.; LEE, K. J.; TSAI, F. J.; Chen, C. Y. C. Current developments of computer-aided drug design, **Journal of the Taiwan Institute of Chemical Engineers**, 41(6):623-635, 2010, doi:10.1016/j.jtice.2010.03.017.

HULL, D.; WOLSTENCROFT, K.; STEVENS, R.; GOBLE, C. A.; POCOCK, M. R.; Li, P.; OINN, T. Taverna: A tool for building and running workflows of services. **Nucleic Acids Research**, 34, 729–732, 2006. doi:10.1093/nar/gkl320

IONIC. **IONIC Project**, 2016. Disponível em: <http://ionicframework.com/>. Acesso em: 10 fev. 2016.

IMOLVIEW. **iMolview a Mobile App for iPhone/iPad and Android**, Molsoft, L.L.C., 2014. Disponível em: <http://www.molsoft.com/iMolview.html>. Acesso em: 07 fev. 2014.

JAX-RS. **JSR 339: JAX-RS 2.0: The Java API for RESTful Web Services**. Disponível em: <https://jcp.org/en/jsr/detail?id=339>. Acesso em: 27 jan. 2014.

JAX-WS, **JSR 224: Java API for XML-Based Web Services (JAX-WS) 2.0**. Disponível em: <https://jcp.org/en/jsr/detail?id=224>. Acesso em: 27 jan. 2014.

JETTE, M.A.; YOO, A.B.; GRONDONA, M. SLURM: Simple Linux Utility for Resource Management In In Lecture Notes in Computer Science: **Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP)** 2003, 44-60.

JMETER. **Apache JMeter - Apache Software Foundation**, 2016. Disponível em: <http://jmeter.apache.org/>. Acesso em: 10 fev. 2016.

JMOL APP. **Jmol Molecular Visualization**, 2014. Disponível em: <https://play. google.com/store/apps/details?id=org.openscience.jmolandroid&hl=pt_BR>. Acesso em: 07 fev. 2014.

JSMOL. **A JavaScript-Based Molecular Viewer From Jmol**, 2014. Disponível em: <http://sourceforge.net/projects/jsmol/>. Acesso em: 11 jan. 2014.

KACSUK, P. (2014). **Science Gateways for Distributed Computing Infrastructures: Development Framework and Exploitation by Scientific User Communities**. Springer International Publishing. ISBN: 978-3-319-11267-1 doi:10.1007/978-3-319-11268-8

KASPRZYK, A. BioMart: driving a paradigm change in biological data management. **Database**, 2011. doi:10.1093/database/bar049

KUHN, T.; WILLIGHAGEN, E. L.; ZIELESNY, A.; STEINBECK, C.; CDK-Taverna: an open workflow environment for cheminformatics. **BMC Bioinformatics**, 11, 159, 2010. doi:10.1186/1471-2105-11-159

LIMA, M.J.; URURAHY, C.; de MOURA, A.L.; MELCOP, T.; CASSINO, C.; SANTOS, M.N.; SILVESTRE, B.; REIS, V.; CERQUEIRA, R. CSBase: A Framework for Building Customized Grid Environments. **Enabling Technologies: Infrastructure for Collaborative Enterprises**, 2006. WETICE '06. 15th IEEE International Workshops, 187-194, June 2006, Manchester, UK. ISSN 1524-4547 doi:10.1109/WETICE.2006.26

LIPMAN, D. J.; PEARSON, W. R. Rapid and sensitive protein similarity searches. **Science**, 227 (4693): 1435–41, 2985. doi:10.1126/science.2983426

LUDÄSCHER, B.; ALTINTAS, I.; BERKLEY, C.; HIGGINS, D.; JAEGER-FRANK, E.; JONES, M.; LEE, E.; TAO, J.; ZHAO, Y. Scientific Workflow Management and the Kepler System. Special Issue: Workflow in Grid Systems. **Concurrency and Computation: Practice & Experience**, 18(10): 1039-1065, 2006.

LUSHBOUGH, C.; LAWRENCE, C. J.; BERGMAN, M.; JENNEWEIN, D.; BRENDEL, V. BioExtract Server – An Integrated System to Access and Analyze Heterogeneous, Distributed Biomolecular Data. **Computational Biology and Bioinformatics**, 7(1), 12-24, 2008. doi:10.1109/TCBB.2008.98

LUSHBOUGH, C. M.; BRENDEL, V. P. An overview of the BioExtract Server: a distributed, Web-based system for genomic analysis. **Advances in Experimental Medicine and Biology**. 680, 361-369, 2010. doi:10.1007/978-1-4419-5913-3_41

MACALINO, S. J. Y.; GOSU, V.; HONG, S.; CHOI. S.; Role of computer-aided drug design in modern drug discovery. **Archives of Pharmacal Research**. 38(9):1686-1701, 2015. doi:10.1007/s12272-015-0640-5

MCGEE, P. Modeling Success with In Silico Tools. **Drug Discovery and Development**, 8(4), 24-28, 2005.

MORRIS, G. M.; HUEY, R.; LINDSTROM, W.; SANNER, M. F.; BELEW, R. K.; GOODSELL, D. S. e OLSON, A. J. Autodock4 and AutoDockTools4: automated docking with selective receptor flexiblity. **Journal of Computational Chemistry**, 16:2785-91, 2009. doi:10.1002/jcc.21256

NDKMOL. NDKmol - Molecular Viewer for Android, 2014. Disponível em: <http://webglmol.sourceforge.jp/android-en.html>. Acesso em: 07 fev. 2014.

NÉRON, B.; MÉNAGER, H.; MAUFRAIS, C.; JOLY, N.; MAUPETIT, J.; LETORT, S.; CARRERE, S.; TUFFERY, P.; LETONDAL, C. Mobyle: a new full web bioinformatics framework. **Bioinformatics**, 25(22), 3005-3011, 2009. doi:10.1093/bioinformatics/btp493

NITZBERG, B.; SCHOPF, J. M.; and JONES. J. P.; 2004. PBS Pro: Grid computing and scheduling attributes. In **Grid resource management**, Kluwer Academic Publishers, Norwell, MA, USA 183-190.

O'BOYLE, N. M.; BANCK, M.; JAMES, C. A.; MORLEY, C.; VANDERMEERSCH, T.; and HUTCHISON, G. R. Open Babel: An open chemical toolbox. **Journal of Cheminformatics** 2011, 3:33. doi:10.1186/1758-2946-3-33

OINN, T.; ADDIS, M.; FERRIS, J.; MARVIN, D.; SENGER, M.; GREENWOOD, M.; CARVER, T.; GLOVER, K.; POCOCK, M. R.; WIPAT, A.; LI, P. Taverna: A tool for the composition and enactment of bioinformatics workflows. **Bioinformatics**, 20 (17), 3045–3054, 2004. doi:10.1093/bioinformatics/bth361

OSG. **Open Science Grid**, 2014. Disponível em <http://www.opensciencegrid. org/>. Acesso em: 11 jan. 2014.

OWASP. **Open Web Application Security Project**, 2016. Disponível em <https://www.owasp.org/>. Acesso em: 10 fev. 2016.

PDB. **Protein Data Bank, Research Collaboratory for Structural Bioinformatics (RCSB)**, 2014. Disponível em <http://www.rcsb.org/pdb/>. Acesso em: 17 jan. 2014.

PLAY, **Play Framework Project**, 2016. Disponível em <https://www.playframe work.com/>. Acesso em: 10 fev. 2016.

PLAZA, W. R. **Vendas de ultrafinos e smartphones aumentam, desktops e notebooks caem**, Guia do Hardware, 2014. Disponível em: <http://www. hardware.com.br/noticias/2014-08/vendas-de-ultrafinos-smartphones-aumentam-desktop-notebooks-caem.html>. Acesso em: 28 jan 2016

PUBCHEM. **National Center for Biotechnology Information**, U.S. National Library of Medicine, 2014. Disponível em: <http://pubchem.ncbi.nlm.nih.gov/>. Acesso em: 07 fev. 2014.

RAPPE, A. K.; CASEWIT, C. J.; COLWELL, K. S.; GODDARD III, W. A.; SKIFF, W. M. UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations. **Journal of American Chemical Society**, 114, 10024–10035, 1992.

RODRIGUES, C. R. Modelagem Molecular. Cadernos Temáticos de Química. **Quimica Nova na Escola**, São Paulo, n. 3, p. 43-49, 2001.

ROSE, P. W.; BI, C.; BLUHM, W. F.; CHRISTIE, C. H.; DIMITROPOULOS, D.; DUTTA, S.; GREEN, R. K.; GOODSELL, D. S.; PRLIĆ, A.; QUESADA, M.; QUINN, G. B.; RAMOS, A. G.; WESTBROOK, J. D.; YOUNG, J.; ZARDECKI, C.; BERMAN, H. M.; BOURNE, P. E. The RCSB Protein Data Bank: new resources for research and education. **Nucleic Acids Research**, 41(D1): D475-D482, 2012.

ROTH, B. D. The Discovery and Development of Atorvastatin, A Potent Novel Hypolipidemic Agent. **Progress in Medicinal Chemistry**, 40:1–22, 2002. doi:10.1016/S0079-6468(08)70080-8

SADASHIV, N.; KUMAR, S. M. D. Cluster, Grid and Cloud Computing: A Detailed Comparison, **The 6th International Conference on Computer Science & Education** (ICCSE 2011) August 3-5, 2011

SELENIUM. **SeleniumHQ Browser Automation**, 2016. Disponível em <http://www.seleniumhq.org/>. Acesso em: 10 fev. 2016.

SG. **ScienceGateways.org**, 2015. Disponível em <http://sciencegateways.org/>. Acesso em: 27 dez. 2015.

SHA-256. **National Institute of Standards and Technology**. http://csrc.nist.gov/groups/ST/toolkit/secure_hashing.html

SHON, J.; OHKAWA, H.; HAMMER, J. Scientific workflows as productivity tools for drug discovery. **Current Opinion in Drug Discovery & Development**, 11(3), 381-8, 2008

SKOVIRA, J.; CHAN, W.; ZHOU, H.; and LIFKA, D. A. 1996. The EASY - LoadLeveler API Project. In **Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (IPPS '96)**, Springer-Verlag, London, UK, UK, 41-47.

STAPLES G. TORQUE resource manager. **Proceedings of the ACM/IEEE conference on Supercomputing (SC '06)** 2006. ACM, New York, NY, USA. doi:10.1145/1188455.1188464

STEINBECK, C.; HAN, Y.; KUHN, S.; HORLACHER, O.; LUTTMANN, E.; WILLIGHAGEN, E. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. **Journal of Chemical Information and Computer Sciences**, 42(2), 493-500, 2003. doi:10.2174/138161206777 585274

SULLIVAN, A.; SHEFFRIN, S. (2003). **Economics: Principles in action**. Upper Saddle River, New Jersey 07458: Pearson Prentice Hall. p. 472. ISBN 0-13-063085-3

TIWARIA, A.; SEKHARB, A. K. T. Workflow based framework for life science informatics. **Computational Biology and Chemistry**. 31(5–6), 305–319, 2007. doi:10.1016/j.compbiolchem.2007.08.009

TOWNS, J.; COCKERILL, T.; DAHAN, M.; FOSTER, I.; GAITHER, K.; GRIMSHAW, A.; HAZLEWOOD, V.; LATHROP, S.; LIFKA, D.; PETERSON, G. D.; ROSKIES R.; SCOTT J. R.; WILKINS-DIEHR N. XSEDE: Accelerating Scientific Discovery, **Computing in Science & Engineering**, 16(5):62-74, 2014, doi:10.1109/MCSE.2014.80

TROTT, O.; OLSON, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading, **Journal of Computational Chemistry** 31 (2010) 455-461

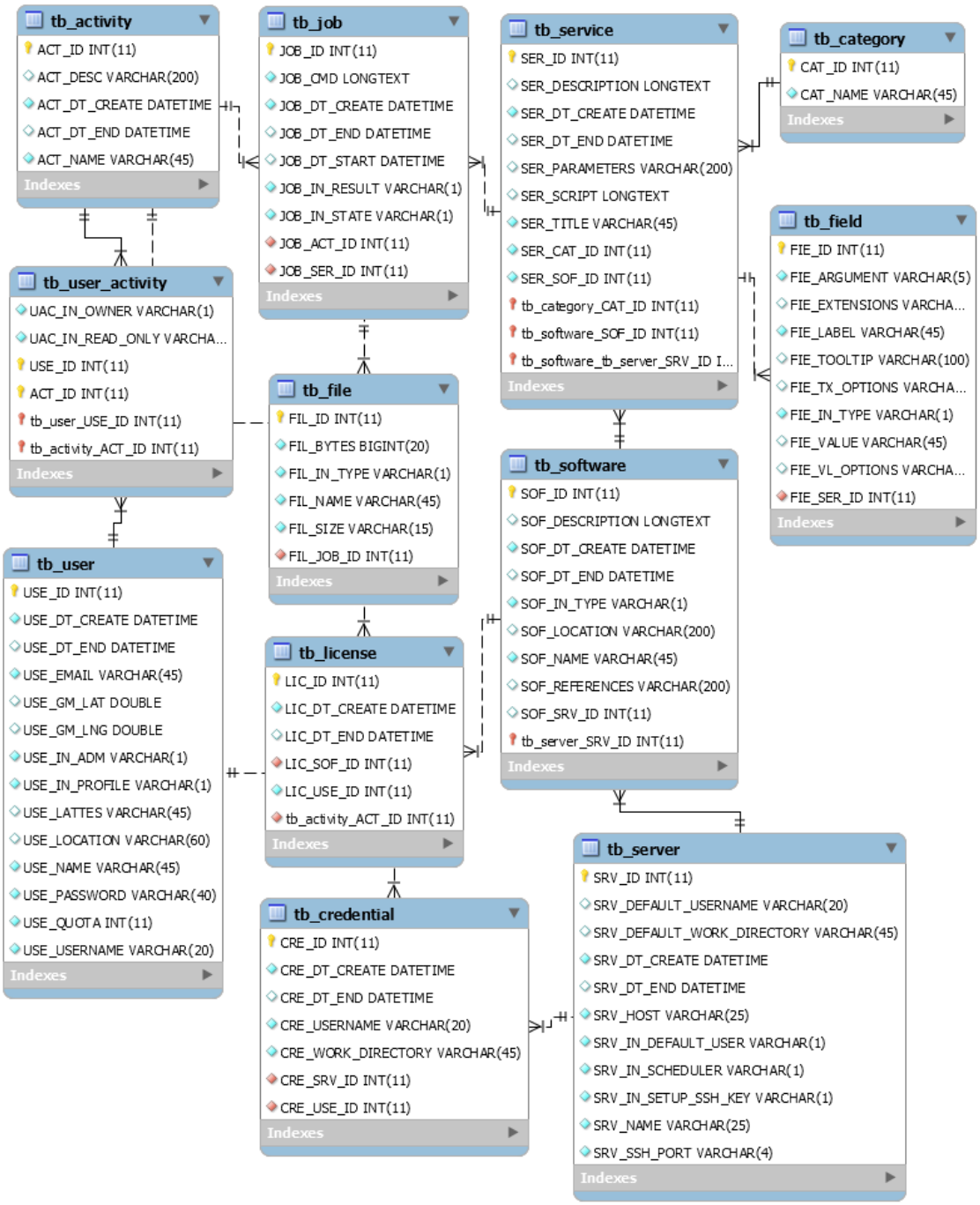
WEININGER, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. **Journal of Chemical Information and Modeling**, 28(1), 31–36, 1988. doi:10.1021/ci00057a005

WOLSTENCROFT, K.; HAINES, R.; FELLOWS, D.; WILLIAMS, A.; WITHERS, D.; OWEN, S.; SOILAND-REYES, S.; DUNLOP, I.; NENADIC, A.; FISHER, P.; BHAGAT, J.; BELHAJJAME, K.; BACALL, F.; HARDISTY, A.; HIDALGA, A. N.; VARGAS, M. P. B.; SUFI, S.; GOBLE, C. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. **Nucleic Acids Research**, 41(W1), W557-W561, 2013. doi:10.1093/nar/gkt328

WUEBBLING, M., 2014. **NVIDIA Reveals Tegra Note – Complete Tablet Platform**, Powered by Tegra 4. Disponível em <http://blogs.nvidia.com/blog/ 2013/09/18/tegra-note/>. Acesso em: 11 jan. 2014.

ZINC. **ZINC Database**, University of California, UCSF, 2014. Disponível em: <http://zinc.docking.org/>. Acesso em: 08 fev. 2014.

# APÊNDICE A - MODELO DE DADOS DO SISTEMA MODIMOL WORKBENCH



# APÊNDICE B - MANUAL DE INSTALAÇÃO DO SISTEMA

## 1. INFORMAÇÕES BÁSICAS

### *1.1 Ambiente de instalação:*

| Tipo de Componente | Elemento |
|---|---|
| **Sistema Operacional** | Linux (Ubuntu 14.10) |
| **Servidor de Aplicação** | Apache Tomcat 7.0.xx |
| **Sistema Gerenciador de Banco de Dados** | MySQL Server 5.6 |
| **Sistema MODiMOL Workbench** | v1.0.0 (mworkbench.war) |

### *1.2 Premissas usadas no manual:*

| Parâmetro | Valor | Observação |
|---|---|---|
| **Usuário Linux** | USUARIO | A ser substituído pelo usuário correspondente no sistema Linux utilizado. |
| **Senhas** | SENHA | A ser substituída, em cada caso, por uma senha apropriada. |

## 2. SSHPASS

### *2.1 Instalar pacote do sshpass:*

```
$ sudo apt-get install sshpass
```

## 3. OPENBABEL

### *3.1 Baixar arquivo contendo código-fonte do openbabel:*

http://sourceforge.net/projects/openbabel/files/openbabel/2.3.2/openbabel-2.3.2.tar.gz/download

### *3.2 Descompactar arquivo e criar pasta para a geração da build local:*

```
$ tar zxvf openbabel-2.3.2.tar.gz
$ mkdir build
```

### *3.3 Instalar g++ (opcional):*

```
$ sudo apt-get install g++
```

### *3.4 Instalar cmake (opcional):*

```
$ sudo apt-get install cmake
```

### *3.5 Instalar eigen (opcional):*

```
$ sudo apt-get install libeigen2-dev
$ sudo apt-get install libeigen3-dev
```

### *3.6 Configurar a build:*

```
$ cd build
$ cmake ../openbabel-2.3.2 -DCMAKE_INSTALL_PREFIX=~/openbabel-
install
```

### *3.7 Compilar arquivos:*

```
$ make
```

ou

`$ make -j<n-cores>` (para máquinas com mais de um núcleo (*multicore*)

### *3.8 Instalar o pacote (configurado para ser instalado na pasta ~/openbabel-install):*

```
$ sudo make install
```

Fonte:
http://open-babel.readthedocs.org/en/latest/Installation/install.html#compiling-open-babel

## 4. JAVA 8 (ORACLE JDK)

### *4.1 Instalar pacote do Java*

```
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update

$ sudo apt-get install oracle-java8-installer
```

### *4.2 Configurar versão default do Java na máquina:*

Quando existem múltiplas instalações de Java em seu ambiente, a versão Java para utilizar como padrão pode ser escolhida através dos seguintes comandos:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

ou

```
$ sudo update-java-alternatives -s java-8-oracle
```

### *4.3 Definir variável de ambiente JAVA_HOME:*

```
$ sudo apt-get install oracle-java8-set-default
```

ou

```
$ sudo gedit /etc/environment
```

Incluir a seguinte linha:

```
JAVA_HOME="/usr/lib/jvm/java-8-oracle"


$ source /etc/environment
$ echo $JAVA_HOME
```

Fontes:
http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-via-ppa.html
https://www.digitalocean.com/community/tutorials/como-instalar-o-java-no-ubuntu-com-apt-get-pt

## 5. APACHE TOMCAT 7.0.xx

*5.1 Baixar arquivo de instalação do Apache Tomcat:*

http://tomcat.apache.org/download-70.cgi

*5.2 Descompactar arquivo:*

```
$ tar -zxvf apache-tomcat-7.0.xx.tar.gz
```

*5.3 Definir variável de ambiente CATALINA_HOME:*

```
$ sudo gedit /etc/environment
```

Incluir no arquivo a seguinte linha:

```
CATALINA_HOME="/home/USUARIO/apache-tomcat-7.0.xx"
```

Atualizar com nova variável de ambiente:

```
$ source /etc/environment
$ echo $CATALINA_HOME
```

*5.4 Desabilitar a opção de "persistir sessões" ao reiniciar Apache Tomcat:*

Descomentar linhas do arquivo `conf/context.xml` contendo:

```
<Manager pathname="" />
```

*5.5 Incluir usuário para acessar página de gerenciamento do Apache Tomcat (opcional):*

Incluir no arquivo `conf/tomcat-users.xml`, a seguinte linha:

```
<role rolename="manager-gui">
<user username="mworkbench" password="SENHA" roles="manager-gui"/>
```

*5.6 Incluir parâmetros de inicialização do Apache Tomcat:*

Criar arquivo de `script bin/setenv.sh`:

```
$ cd apache-tomcat-7.0.xx/bin
$ gedit setenv.sh
```

Incluir a seguinte linha:

```
export CATALINA_OPTS="-Xms256m -Xmx1024m -Dcom.sun.management.jm
xremote -Dorg.apache.el.parser.COERCE_TO_ZERO=false"
```

Habilitar execução do arquivo de script:

```
$ chmod +x setenv.sh
```

### *5.7 Configurar HTTPS no Apache Tomcat:*

Gerar um arquivo para armazenar o certificado digital (*self-signed*) `keystore` a partir da ferramenta `keytool`:

```
$ keytool -genkey -alias mworkbench -keyalg RSA -keystore /home/
USUARIO/apache-tomcat-7.0.xx/conf/mworkbenchkeystore
```

Descomentar linhas do arquivo `conf/server.xml`,correspondentes ao conector HTTPS e acrescentar os seguintes parâmetros:

```
<Connector port="8443" ... keystoreFile="conf/mworkbenchkeystore"
keystorePass="SENHA" />
```

Fontes:
https://tomcat.apache.org/tomcat-7.0-doc/ssl-howto.html
http://www.mkyong.com/tomcat/how-to-configure-tomcat-to-support-ssl-or-https/
http://www.loiane.com/2011/06/configurando-ssl-no-tomcat-em-5-minutos-httpslocalhost8443/

## 6. MYSQL SERVER 5.6

### *6.1 Desinstalar versão anterior do MySQL (opcional):*

```
$ sudo apt-get remove --purge mysql-server mysql-client mysql-common
$ sudo apt-get autoremove
$ sudo apt-get autoclean
$ sudo rm -rf /var/lib/mysql
$ sudo rm -rf /etc/mysql
```

### *6.2 Instalar MySQL Server:*

```
$ sudo apt-get install mysql-server
```

*6.3 Criar banco de dados do sistema MODiMOL Workbench:*

```
$ mysql -u root -p

mysql> create database if not exists `mworkbenchdb`;
```

*6.4 Aplicar estrutura do banco de dados (script do Apêndice A)(`script.sql`):*

```
$ mysql -u root -p mworkbenchdb < script.sql
```

ou

```
$ mysql -u root -p

mysql> use mworkbenchdb;
mysql> source mworkbenchdb.sql
```

*6.5 Criar usuário no MySQL:*

```
$ mysql -u root -p

mysql> create user 'mworkbench'@'localhost' identified by 'SENHA';
mysql> create user 'mworkbench'@'127.0.0.1' identified by 'SENHA';
mysql> grant all on mworkbenchdb.* to 'mworkbench'@'localhost';
mysql> grant all on mworkbenchdb.* to 'mworkbench'@'127.0.0.1';
```

*6.6 Criar registro para administrador do sistema (user: admin, pass: admin):*

```
$ mysql -u mworkbench -p

mysql> insert into `tb_user` values (1, '2015-01-01 12:00:00', null,
'admin@ufpe.br', NULL, NULL, 'Y', 'A', null, null, 'Administrator',
'd033e22ae348aeb5660fc2140aec35850c4da997', 100, 'admin');
```

Fontes:
http://www.k19.com.br/artigos/como-instalar-o-mysql-server-no-ubuntu-13-10/
https://help.ubuntu.com/community/MysqlPasswordReset
http://www.howtogeek.com/howto/programming/mysql/dump-just-the-table-structure-to-a-file-in-mysql/
https://dev.mysql.com/doc/refman/5.5/en/default-privileges.html

# 7. MODIMOL WORKBENCH

## *7.1 Configurar pastas do sistema:*

Criar pasta para armazenar os jobs:

```
$ mkdir mworkbench
$ cd mworkbench
$ mkdir jobs
```

Criar pasta para armazenar os arquivos dos usuários:

```
$ mkdir storage
$ cd storage
$ mkdir 1
```
(pasta para o usuário administrador)

## *7.2 Iniciar Apache Tomcat junto com o sistema MODiMOL Workbench:*

Copiar arquivo contendo a aplicação do sistema para pasta de aplicações do Apache Tomcat:

```
$ cp mworkbench.war /home/USUARIO/apache-tomcat-7.0.xx/webapps
```

Iniciar servidor Apache Tomcat:

```
$ cd apache-tomcat-7.0.xx/bin
$ ./startup.sh
```

## *7.3 Configurar arquivo `apache-tomcat-7.0.xx/webapps/mworkbench/WEB-INF/web.xml`:*

Incluir os seguintes parâmetros:

```
<context-param>
 <param-name>javax.faces.PROJECT_STAGE</param-name>
 <param-value>Production</param-value>
</context-param>

<context-param>
 <param-name>javax.faces.FACELETS_REFRESH_PERIOD</param-name>
 <param-value>-1</param-value>
</context-param>
```

*7.4 Configurar arquivo `apache-tomcat-7.0.xx/webapps/mworkbench/WEB-INF/classes/application.properties`:*

Definir caminho para a pasta raiz que contém os arquivos dos jobs:

```
mworkbench.job.directory=/home/USUARIO/mworkbench/jobs
```

Definir caminho para a pasta raiz que contém os arquivos pessoais dos usuários:

```
mworkbench.storage.directory=/home/USUARIO/mworkbench/storage
```

*7.5 Configurar arquivo `apache-tomcat-7.0.xx/webapps/mworkbench/WEB-INF/classes/log4j.properties`:*

Definir nível de verbosidade e tipos de saída do componente de log do sistema:

```
log4j.rootLogger=WARN, stdout, file
```

*7.6 Configurar arquivo `apache-tomcat-7.0.xx/webapps/mworkbench/WEB-INF/classes/hibernate.cfg.xml`:*

Incluir os seguinte parâmetros:

```
<property name="hibernate.connection.username">mworkbench</property>
<property name="hibernate.connection.password">SENHA</property>
```

*7.7 Reiniciar Apache Tomcat com as novas configurações:*

```
$ cd apache-tomcat-7.0.xx/bin
$ ./shutdown.sh
$ ./startup.sh
```

## 8. REDIRECIONAMENTO DE PORTAS COM IPTABLES (OPCIONAL)

Para executar o Apache Tomcat em um servidor Linux com permissão de acesso para usuários externos nas portas 80 (HTTP) e 443 (HTTPS), é necessário realizar os seguintes passos:

*8.1 Configurar a máquina para aceitar conexões (de entrada) na porta 80, na porta 8080 e para realizar o redirecionamento da porta 80 para a 8080:*

```
$ iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
```

```
$ iptables -A INPUT -i eth0 -p tcp --dport 8080 -j ACCEPT
$ iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j
REDIRECT --to-port 8080
```

*8.2 Idem para as portas 443 e 8443:*

```
$ iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT
$ iptables -A INPUT -i eth0 -p tcp --dport 8080 -j ACCEPT
$ iptables -A PREROUTING -t nat -i eth0 -p tcp --dport 80 -j
REDIRECT --to-port 8080
```

Fonte:
http://proghowto.com/iptables-redirect-port-80-to-port-8080