UNIVERSIDADE DE CAXIAS DO SUL

CENTRO DE CIÊNCIAS EXATAS E DA TECNOLOGIA

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

RODRIGO AUGUSTO BROMBATTI ZINI

MIGRAÇÃO DO BIOINFORMATIC TOOLS DESKTOP PARA A PLATAFORMA WEB

CAXIAS DO SUL, DEZEMBRO DE 2015.

RODRIGO AUGUSTO BROMBATTI ZINI

MIGRAÇÃO DO BIOINFORMATIC TOOLS DESKTOP PARA A PLATAFORMA WEB

Trabalho de Conclusão do Título de Bacharel em Sistemas de Informação pela Universidade de Caxias do Sul, como requisito parcial para obtenção do título de Bacharel, orientado pela Profa. Dra. Helena Graziottin Ribeiro.

CAXIAS DO SUL, DEZEMBRO DE 2015.

Dedico este trabalho a todas as pessoas que me auxiliaram para o desenvolvimento do mesmo, sem o auxílio destas pessoas não teria conseguido.

RESUMO

Com a evolução dos estudos do DNA humano a tecnologia fez-se cada vez mais necessária para auxiliar os profissionais da biologia. Com o surgimento da Bioinformática, a utilização de programas computacionais acabou auxiliando no entendimento de interações entre moléculas, células entre diversos outros organismos. Para facilitar mais ainda estas análises fez-se necessário o uso de *workflows.* Um *workflow* científico é uma experiência executada através de uma sequência de etapas e este conjunto de etapas caracterizam um fluxo de execução com propósitos de integrar e processar dados para um fim específico. A ferramenta *Bioinformatic tools,* foi desenvolvida através do conceito de *workflow* científico e é capaz de integrar diferentes bases de dados. A aplicação foi desenvolvida para a plataforma *desktop* utilizando Java. Para utilização da aplicação os usuários necessitam realizar o *download* da mesma. O objetivo deste trabalho é apresentar uma proposta de migração da ferramenta *desktop* para a plataforma *web.*

Palavras-chave: Migração de interface gráfica para *web*, Integração de banco de dados, *workflow* científico.

LISTA DE TABELAS

LISTA DE ILUSTRAÇÕES

.

LISTA DE SIGLAS

BLAST	*Basic Local Alignment Search Tool*
DNA	*Deoxyribonucleic Acid*
ERP	*Enterprise Resource Planning*
GNU	*General Public License*
GUI	*Graphic User Interface*
IDE	*Integrated Development Environment*
MVC	*Model Vision Controller*
OMIM	*Online Mendelian Inheritance in Man*
PDB	*Protein Data Bank*
PHP	*Personal Home Page*
RNA	*Ribonucleic Acid*
STRING	*Search Tool for the Retrieval of Interacting Ge-nes/Proteins*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo será realizada a contextualização do trabalho em questão, será apresentado a questão e o problema de pesquisa identificado. Além disso, serão destacados os objetivos, estrutura do trabalho.

1.1 CONTEXTUALIZAÇÃO GERAL DO TRABALHO

Durante o início do século passado geneticistas e químicos se questionavam a respeito do material genético. Após diversas pesquisas desenvolvidas, pode-se concluir que de que o DNA era responsável por armazenar a informação genética (LESK, 2008). Após a descoberta do material genético logo acabaram surgindo diversos métodos de sequenciamentos do DNA, que acabaram permitindo um estudo mais aprofundado de suas sequências monoméricas constituintes (LESK, 2008).

Na segunda metade da década de 90, após o surgimento dos sequenciadores automáticos de DNA, houve a necessidade de utilizar recursos computacionais cada vez mais sofisticados e eficientes tanto para realizar os sequenciamentos quanto para interpretar os resultados obtidos (JENSEN et. al., 2009). Desta forma nascia a bioinformática, uma ciência capaz de agrupar diversas áreas de conhecimento como engenharia de *software*, matemática, estatística, ciência da computação e a biologia molecular (JENSEN et. al., 2009).

A realização dos experimentos da biologia pode ser realizada através de *workflows* científicos, ou seja, sequência de passos que caracterizam um fluxo de execução (LESK, 2008). Os processos dos *workflows* são representados por passos, como por exemplo, um programa de *software* para executar buscas ou a execução e um *webservice*.

O *software Bioinformatic Tools* foi desenvolvido com a finalidade de auxiliar os biólogos nos *workflows* científicos, permitindo o acesso aos bancos de dados STRING e PDB como também às ferramentas BLAST através de uma aplicação GUI (*Graphic User Interface*) e outras ferramentas utilizando acessos URL (*Uniform Resource Locator)* (NOTARI et. al., 2012).

1.2 PROBLEMA E QUESTÃO DE PESQUISA

Tendo isto em mente se fez necessário criar ferramentas capazes de auxiliar o acesso ao banco de dados e ferramentas de bioinformática. Dentre as ferramentas

desenvolvidas podemos detalhar três delas, que serão o foco durante todo trabalho de conclusão, (NOTARI et al, 2012). As ferramentas são:

- Ferramenta para acesso ao banco de dados STRING (*Search Tool for the Retrieval of Interacting Ge-nes/Proteins*). Banco de dados aonde são armazenadas informações sobre as interações diretas (físicas) e indiretas (funcional) das proteínas.
- Ferramenta de pesquisa BLAST (*Basic Local Alignment Search Tool*). O programa compara as sequências de uma proteína com as diversas outras sequências que constam em seu banco de dados e retorna qual a importância estatística dos resultados encontrados.
- Ferramenta para acesso ao banco de imagens PDB (*Protein Data Bank*). Possível de se obter imagens das proteínas em formato 3D.

Estas ferramentas foram desenvolvidas na linguagem Java versão 1.6 para a plataforma *desktop.* Pode-se ter acesso às ferramentas e manuais através do *link* [http://www.danlian.com.br/tools/], estando licenciada pela GPL (GNU *General Public License*), sua distribuição, assim como seu uso e alteração são livres.

Visto que estas ferramentas foram desenvolvidas para apenas uma plataforma, as mesmas acabam ficando um pouco engessadas não sendo tão flexível ao modo como elas podem ser utilizadas e/ou acessadas. O que também acaba dificultando o acesso a estas ferramentas são as configurações que são necessárias como, por exemplo, diretórios locais para armazenamento dos arquivos, versão do Java pode ser incompatível.

Baseado no problema de pesquisa descrito anteriormente foi criado a seguinte questão de pesquisa:

“Como realizar a migração das ferramentas acima citadas da plataforma *desktop* para a plataforma *web* visando tornar o acesso a esta aplicação mais flexível diante dos usuários?”

1.3 OBJETIVOS

Baseando-se nos problemas descritos anteriormente, fez-se necessário buscar formas de resolver estas questões. Desta maneira o objetivo deste trabalho é migrar estas ferramentas que se encontram em uma plataforma *desktop* para uma plataforma *web*, e desta forma deixa-se a aplicação sempre disponível para acesso.

1.4 ESTRUTURA DO TEXTO

Para alcançar os objetivos citados acima, este trabalho foi estruturado da seguinte forma: uma conceituação referente aos elementos da bioinformática e recursos utilizados na versão *desktop* da ferramenta *Bioinformatic tools* com uma explanação de como a ferramenta foi desenvolvida e seu funcionamento na versão *desktop*. Além disso, foi realizado um levantamento referente às principais linguagens de programação que poderiam ser utilizadas, à questão de migração de aplicações, e como pode ser feita a representação das arquiteturas de *software*. Após essa conceituação, apresenta-se a proposta de migração para *web* a ser realizada na ferramenta *Bioinformatic tools*. Por fim descreve-se a implementação de mudança, como a migração foi realizada, juntamente com a descrição da estrutura implementada. Também são apresentados alguns estudos de casos das aplicações após a migração para a plataforma *web,* finalizando-se com as principais conclusões do trabalho realizado.

2 BIOINFORMÁTICA E FERRAMENTAS

Este capítulo apresenta uma revisão bibliográfica referente aos conceitos utilizados na concepção da ferramenta *Bioinformatic Tools* na área da bioinformática, apresentando como é o funcionamento das ferramentas, esclarecendo termos, para auxiliar no entendimento da proposta de migração das ferramentas para a plataforma *web*.

A ferramenta *Bioinformatic Tools*, que possui sua principal atuação na área da Bioinformática, trabalha com proteínas e suas iterações, utilizando a base de dados OMIM e programas de busca de informações como STRING, BLAST e PDB para a utilização de seus dados. Nesta sessão será realizada uma revisão dos conteúdos para um melhor entendimento da ferramenta e suas funções.

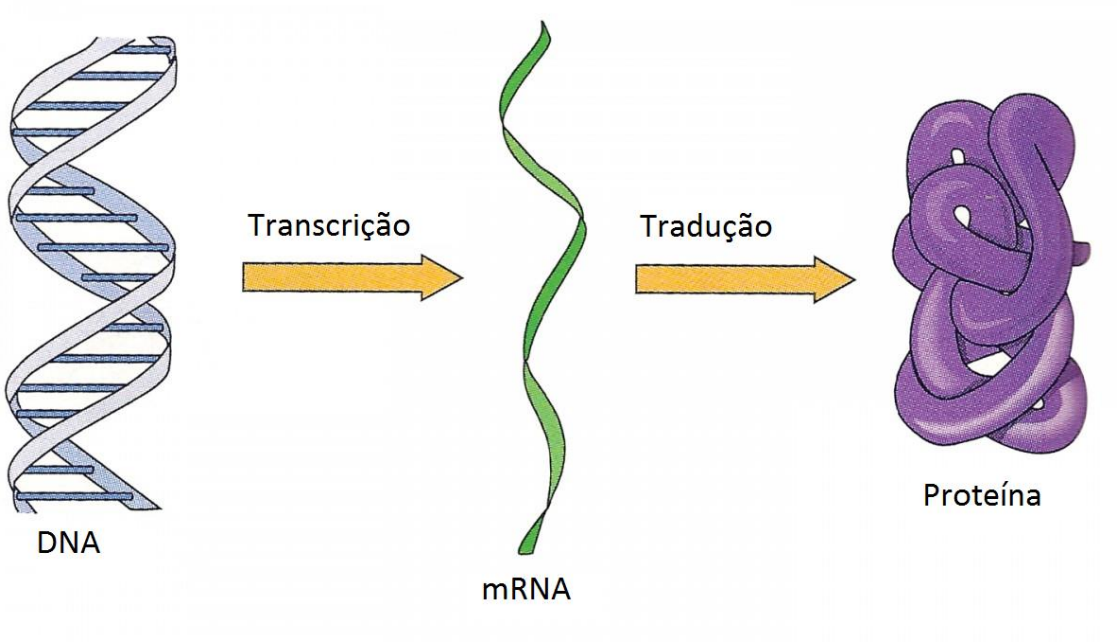
2.1 Proteínas, DNA e RNA

Proteínas são macromoléculas formadas por moléculas de aminoácido, capazes de desempenhar diversas funções no organismo, como: estrutural, hormonal, enzimática, imunológica, nutritiva, entre outras funções. Cada proteína é composta por milhares aminoácidos, o que determina a sequência destas unidades é a informação genética contida no gene. Portanto todo e qualquer funcionamento do organismo é comandado pelo controle das moléculas de DNA (LESK, 2008).

O DNA (*Deoxyribonucleic Acid*), em português: ácido desoxirribonucleico é um composto orgânico que contêm todas as informações genéticas, informações estas que são responsáveis por coordenar o desenvolvimento e funcionamento de todos os seres vivos. Seu principal papel é coletar e armazenar informações capazes de construir proteínas e RNA. Os segmentos responsáveis por armazenar as informações genéticas são denominados genes. O restante da estrutura do DNA tem responsabilidade estrutural ou possui envolvimento na regulação do uso da informação genética (LESK, 2008).

Por sua vez, o RNA (*Ribonucleic Acid*), em português: ácido ribonucleico é o principal responsável pela síntese de proteínas da célula. As moléculas formadas pelo RNA possuem suas dimensões muito inferiores às formadas por DNA. O processo de produção das moléculas de RNA é chamado de transcrição. Existem três tipos de RNA são eles: RNA ribossômico, responsável por produzir ribossomos, que por sua vez produzem proteínas, RNA transportador, realiza o transporte dos aminoácidos até os ribossomos para que seja possível produzir as proteínas, e o RNA mensageiro, este possui as informações para que seja possível realizar a síntese do RNA (LESK, 2008).

Figura 1: Fluxo de informações a partir do DNA à Proteína.



Fonte: o Autor.

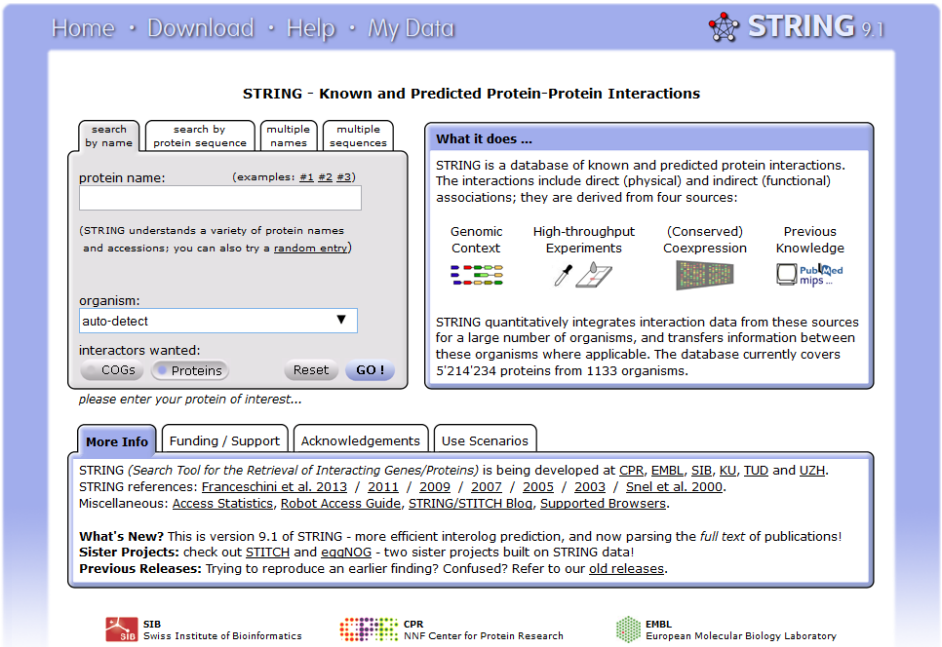
2.2 Workflow Científico

Um *workflow* científico é definido por uma especificação formal de um processo científico representando todos os passos que devem ser executados em um determinado experimento (DEELMAN et al. 2009). Na maioria dos casos estes passos são associados à seleção de dados, análise e visualização. Todos estes *workflows* podem ser gerenciados manualmente, porém a maioria dos *workflows* científicos são gerenciados por sistemas chamados Sistemas Gerenciadores de *Workflows* Científicos.

2.3 STRING

O STRING é uma ferramenta para realização de metabuscas com o intuito de se obter redes de interação de proteínas (JENSEN et. al., 2009). Possui um banco de dados sobre predições e interações de proteínas conhecidas, envolvendo associações físicas e funcionais (JENSEN et. al., 2009).

Figura 2: Site para acesso a base STRING.



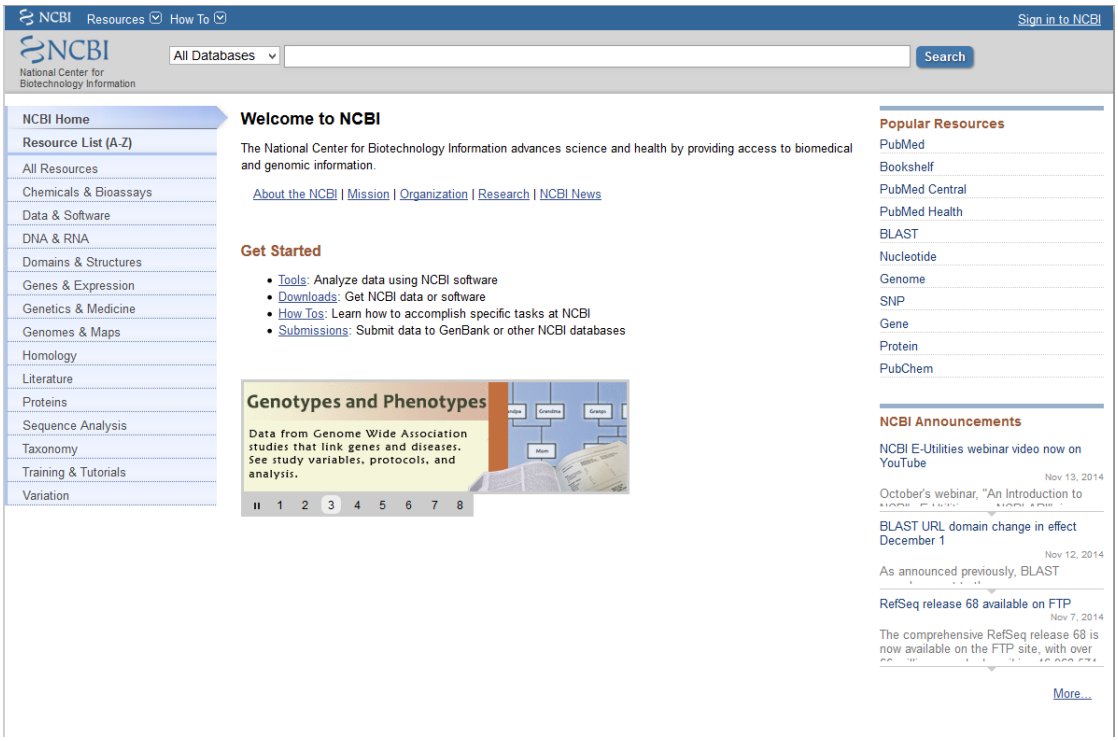
Fonte: Autor.

Atualmente, o STRING possui informações sobre mais de 1,1 mil organismos e mais de cinco milhões de proteínas. O STRING pode ser acessado pelo link [http://string-db.org/].

2.4 BLAST

O BLAST é uma ferramenta utilizada com o intuito de identificar e comparar sequências biológicas, como por exemplo, as sequências de aminoácidos de diferentes proteínas, ou até mesmo nucleotídeos de sequências de DNA (NOTARI et al, 2012). Com o BLAST é possível comparar uma sequência fornecida em uma consulta à base de dados e identificar as bibliotecas que se assemelham a sequência fornecida e que estejam acima de certo grau de semelhança (MARKEL et al, 2003). O BLAST pode ser acessado pelo link [http://www.ncbi.nlm.nih.gov/]

Figura 3: Site para acesso a ferramenta BLAST.



Fonte: Autor.

2.5 A FERRAMENTA BIOINFORMATIC TOOLS

A ferramenta *Bioinformatic tools* foi desenvolvida com o intuito de ser uma ferramenta de *workflow* científico específico para bioinformática, capaz de gerar uma rede de iterações com de dados de proteínas através de diversos e distintos conjuntos de dados.

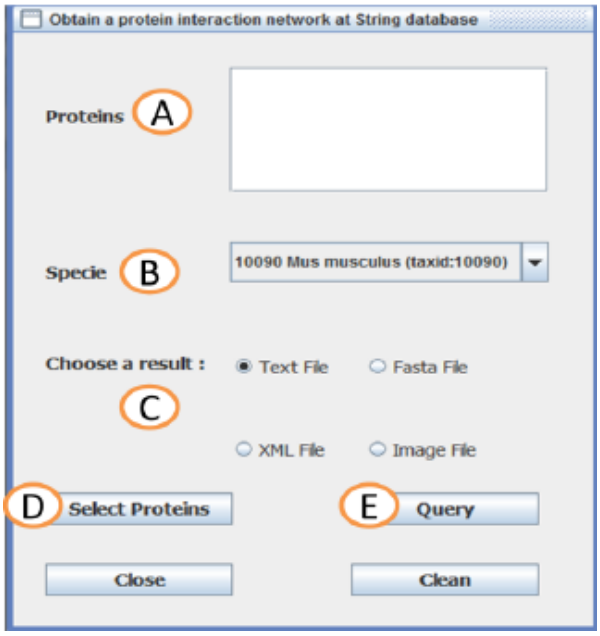
A ferramenta possui como principais funcionalidades o acesso às bases de dados STRING, para buscas de validações de sequências das proteínas e PDB, para buscas de imagens em 3D das proteínas, como também a ferramenta BLAST que realizar um alinhamento de sequências utilizando o algoritmo NCBI PSI-BLAST do NCBI, utilizando uma aplicação GUI (*Graphic User Interface*) e outras ferramentas que utilizam o acesso via URL (*Uniform Resource Locator*).

2.5.1 Acesso à base de dados STRING

O acesso à base de dados STRING (Figura 4) é realizada através de informações de uma ou mais proteínas (uma por linha; 4A) ou através de uma seleção (4D) apresentada na Figura 5. Depois de selecionado a(s) proteína(s) é necessário

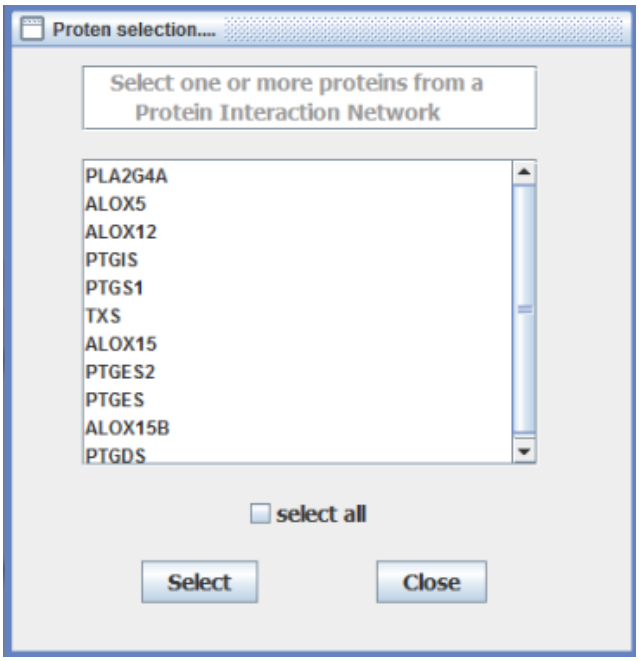
informar a espécie e/ou organismo (4B) e o formato em que o arquivo de retorno vai ser exportado (4C). Após o usuário informar os dados necessários, o mesmo executa a consulta dos dados (4D) e o resultado é mostrado no navegador *web* configurado na ferramenta (NOTARI et al, 2012).

Figura 4: Tela de consulta ao banco de dados STRING.



Fonte: (NOTARI et al, 2012).

Figura 5: Tela de seleção de proteínas.



Fonte: (NOTARI et al, 2012).

Para realizar o acesso à base de dados STRING é necessário montar uma URL para executar a consulta como, por exemplo:

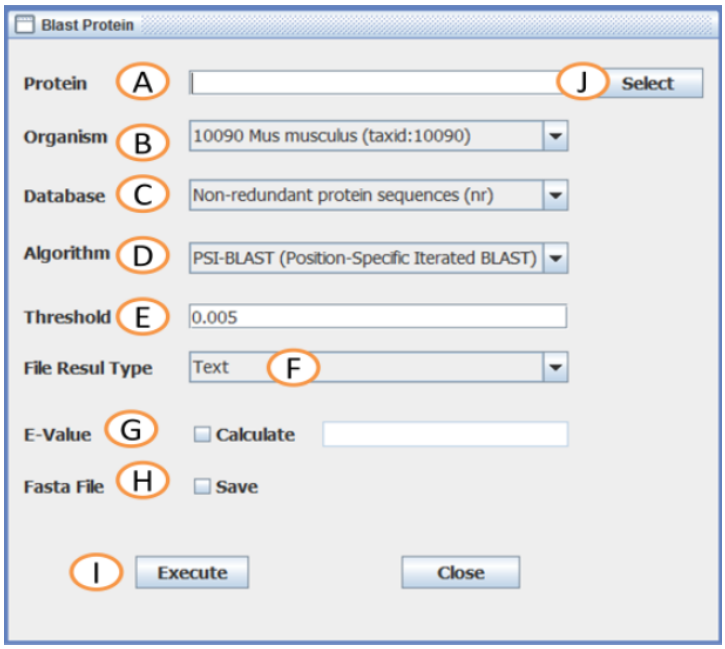
string-db.org/api/psi-mi-ab/interactionsList?identifiers=10090.ENSMUSP00000020268

A composição da URL é informada o *site* do banco de dados (string-db.org), o tipo de acesso (api), o tipo de consulta (neste caso foi utilizado para buscar uma rede no formato de arquivo texto: psi-mi-tab) e a lista de proteínas (código STRING para a proteína PS3 é ENSMUSP00000020268) e para a sua espécie (10090 é o código para a espécie *Musmusculus*) (NOTARI et al, 2012).

2.5.2 Acesso à ferramenta BLAST

Para realizar a consulta utilizando o algoritmo BLAST do NCBI, apresentado na Figura 6, inicialmente é necessário informar uma proteína (6A) ou escolher através da tela de seleção (6J) apresentada na Figura 5. Após selecionar a proteína o usuário deve informar o algoritmo (6B), o banco de dados do NCBI que será utilizado para realizar a consulta (6C), o algoritmo de alinhamento de sequências (6D), o valor do limite aceitável (6E) e, o formato do arquivo que será exportado o resultado (6F). Após preencher todos estes campos o usuário executa a consulta (6I), o resultado do valor do *e-value* é mostrado na tela (6G) e, se a opção de gravar as informações do arquivo FASTA (6H) estiver marcada, as informações são gravadas na pasta configurada na aplicação. O resultado do alinhamento das sequências é mostrado no navegador *web* configurado. O acesso ao algoritmo BLAST é realizado através das ferramentas do Entrez do NCBI (NOTARI et al, 2012).

Figura 6: Tela de execução de um alinhamento de sequências.

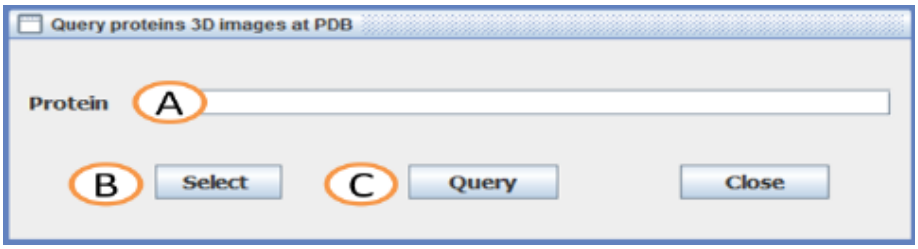


Fonte: (NOTARI et al, 2012).

2.5.3 Acesso à base de dados PDB

O acesso à base de dados PDB (Figura 7) é realizada através das informações de uma proteína (não é permitido informar mais de uma proteína; 7A) ou através da seleção de proteínas (7B) apresentada na Figura 5. Após o usuário informar todos os dados necessários o mesmo executa a consulta (7C) e o resultado (uma ou mais imagens da estrutura de uma proteína em formato tridimensional) será mostrado no navegador *web* configurado na aplicação. O acesso ao banco de dados se dá através de um serviço (*webservice*) fornecido pelo site (NOTARI et al, 2012).

Figura 7: Tela de consulta ao banco de dados PDB.



Fonte: (NOTARI et al, 2012).

3 EVOLUÇÃO E ARQUITETURA DE SOFTWARE

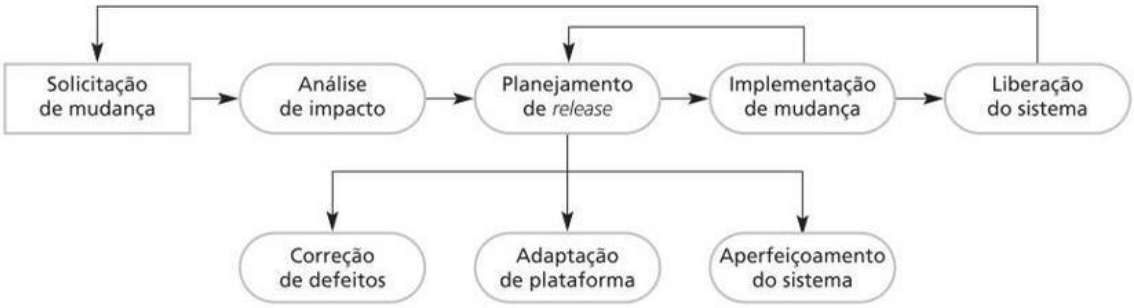
Neste capítulo serão explanados, de uma forma breve, os conceitos de evolução de *software* e arquitetura de *software,* conceitos estes que serão aplicados durante a proposta de migração da aplicação *Bioinformatic Tools.*

3.1 EVOLUÇÃO DE SOFTWARE

Conforme Sommerville (2011) o processo de evolução do *software* não é interrompido quando o mesmo é entregue, mas sim se estende por toda a sua vida útil. Desde uma simples mudança ou até mesmo uma correção pode tornar-se uma evolução no software, até mesmo uma melhoria de hardware pode acarretar em uma evolução no software para atender alguma demanda necessária (SOMMERVILLE, 2011).

Todo o processo de evolução de software se inicia com uma solicitação de mudança, conforme podemos ver na Figura 8. A partir desta solicitação é realizada toda uma análise de impacto, ou seja, aonde esta alteração terá efeito na aplicação. Depois de realizada a análise realiza-se o planejamento do *release*, considerando a mudança como uma melhoria no produto, adaptação de plataforma ou correção de defeitos. A partir do planejamento é realizada a implementação da mudança e, se necessário for, é refeito o planejamento do *release*. Ao finalizar é liberado o novo *release* para o sistema (SOMMERVILLE, 2011).

Figura 8: Processo de Evolução de Software.



Fonte: Sommerville, 2011.

Segundo Sommerville (2011), para executar o processo de mudança pode ser necessário executar a etapa de reengenharia de *software*, pois existem mudanças que podem alterar o escopo inicial do projeto, migrando-o para uma nova plataforma, tecnologia ou aprimorando as funcionalidades do *software*.

Outros autores como Pfleeger (2004) abordam o termo reengenharia de *software* como rejuvenescimento de *software*, pois se trata de um desafio realizar uma melhoria na estrutura e qualidade do *software* mantendo suas funcionalidades iniciais. Já Pressman (2011) aborda a reengenharia, citando que toda e qualquer mudança realizada pode acarretar danos em a sua estrutura, pois na maioria das vezes não é aplicada a melhor prática de engenharia de *software*, e que se deve utilizar a reengenharia para reconstituir a sua integridade visando sempre mantê-lo na forma mais legível e que sua usabilidade seja a mais simples possível.

A reengenharia de *software* pode ser aplicada também na alteração da documentação do *software*, mudança de linguagem, migração de plataforma, mudança de arquitetura. Os benefícios que podem ser visualizados quando aplica-se a reengenharia do *software* são: o baixo risco, pois quando é necessário reescrever um *software* crítico, com muita regra de negócio, como por exemplo um ERP (*Enterprise Resource Planning*), tem-se um risco elevadíssimo, e o baixo custo pois o tempo de aplicação e consequentemente o custo da reengenharia se tornam significativamente menores do que o custo do desenvolvimento de uma nova aplicação (SOMMERVILLE, 2011).

Figura 9: Processo de Reengenharia de Software.

Fonte: Sommerville, 2011.

O processo de reengenharia de software possui como principais atividades (SOMMERVILLE, 2011) (Figura 9):

- **Tradução de código-fonte**: processo onde o programa original é traduzido e convertido a partir de uma linguagem de programação antiga para uma linguagem mais moderna da mesma linguagem ou até mesmo de outra diferente;

- **Melhoria de estrutura do programa**: sua estrutura é analisada e são realizados ajustes para que sua leitura seja mais bem compreendida;
- **Engenharia reversa**: o *software* é analisado e são extraídas informações que auxiliam a realizar a sua documentação quanto à funcionalidade e estrutura.
- **Modularização do programa**: são agrupadas partes do sistema que possuem relação entre si, e caso seja possível eliminam-se redundâncias de programas e/ou códigos-fonte.
- **Reengenharia de dados**: os dados processados podem ser alterados para se adequar às mudanças aplicadas, isto pode significar alterações na estrutura relacional de tabelas do banco de dados.

Estas atividades não precisam ser todas aplicadas para realizar a reengenharia do *software*, deve-se adequar conforme a necessidade da mudança. Algumas destas atividades só serão necessárias conforme for realizada a adequação da arquitetura do *software* no momento de sua implementação (SOMMERVILLE, 2011).

3.2 ARQUITETURA DE SOFTWARE

Define-se arquitetura de *software* como a estrutura ou as estruturas de um sistema, que abrangem desde seus componentes de *software* até as propriedades extremamente visíveis juntamente com a conexão entre eles (PRESSMAN, 2011).

Pressman (2011) define que com a arquitetura de *software* permite-se:

- Analisar a efetividade de atendimento dos requisitos do projeto;
- Considerar alternativas de mudança de arquitetura de um projeto de uma forma relativamente fácil;
- Minimizar os riscos causados à construção de *softwares*.

Pode-se perguntar por que a arquitetura de *software* é importante para o desenvolvimento de um *software*. Bass e seus colegas (BASS, 2003) identificam três razões da importância da consideração da arquitetura de *software* em seu desenvolvimento:

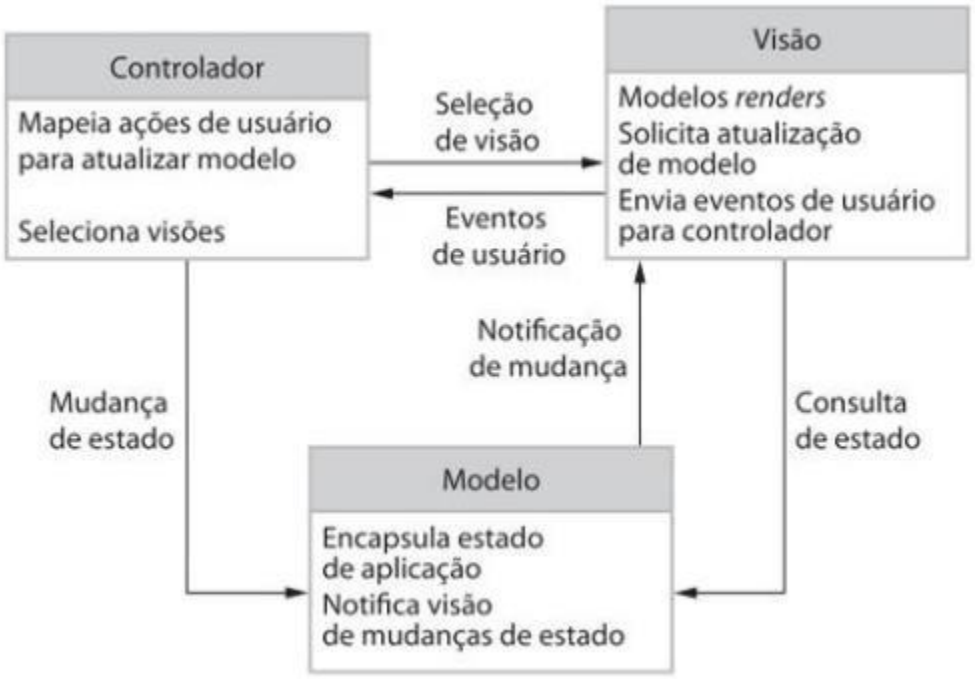
1. As apresentações da arquitetura de *software* atuam com facilitadores na comunicação as partes envolvidas com o projeto de desenvolvimento de um *software*.

2. A arquitetura evidencia decisões a serem tomadas nos primeiros passos de um projeto que terão suma importância no decorrer de todo o processo de engenharia de *software* até o sucesso final do sistema.
3. Com a arquitetura pode-se constituir um modelo relativamente pequeno se comparado com o tamanho total do projeto, porém que facilita a compreensão de como o sistema será estruturado e como os seus componentes atuam em conjunto.

Dentre os diversos padrões de arquitetura que Sommerville (2011) apresenta o MVC (*Model Vision Controller*) (Figura 10), a arquitetura em camadas, a arquitetura de repositórios e a arquitetura cliente-servidor.

O padrão MVC é muito utilizado em sistemas que possuem sua plataforma na *web.* O MVC é estruturado em três componentes lógicos que possuem interação entre si (SOMMERVILLE, 2011): a) componente modelo que gerencia os sistemas de dados e as operações associadas entre eles; b) o componente visão que gerencia como serão apresentados os dados ao usuário da aplicação; e por último c) o componente controlador que gerencia as interações do usuário e as transmite para o controlador visão.

Figura 10: Padrão MVC.



Fonte: Sommerville, 2011.

O padrão de arquitetura em camadas permite que seja possível separar e mesmo assim manter a independência das camadas, sendo as possíveis alterações localizadas na camada específica. O padrão MVC é um dos padrões que é organizado em camadas. Utilizando estes padrões as funcionalidades do sistema são mantidas e organizadas em camadas distintas, ou seja, cada camada é independente, porém fornece serviços e informações para camadas externas e atua como um cliente para a camada interna (PFLEEGER, 2004; SOMMERVILLE, 2011). Com a arquitetura de *software* apresentada no padrão MVC se for preciso realizar qualquer alteração em uma camada, apenas a sua camada adjacente sofrerá alterações. Isso ocorre devido à arquitetura ser mutável e portável (SOMMERVILLE, 2011).

4 PROPOSTA DE UMA VERSÃO WEB PARA A FERRAMENTA BIOINFORMATIC TOOLS

A proposta de migração da ferramenta *Bioinformatic tools* para a plataforma *web* seguirá a metodologia e princípios de evolução de *software* apresentados por Sommerville (Figura 9). Para isto, na definição da proposta serão detalhadas as três primeiras etapas da metodologia, solicitações de mudança, análise de impacto destas mudanças e o planejamento do *release*.

4.1 SOLICITAÇÕES DE MUDANÇAS

As solicitações de mudanças apresentadas a seguir correspondem aos objetivos específicos apresentados e definidos anteriormente:

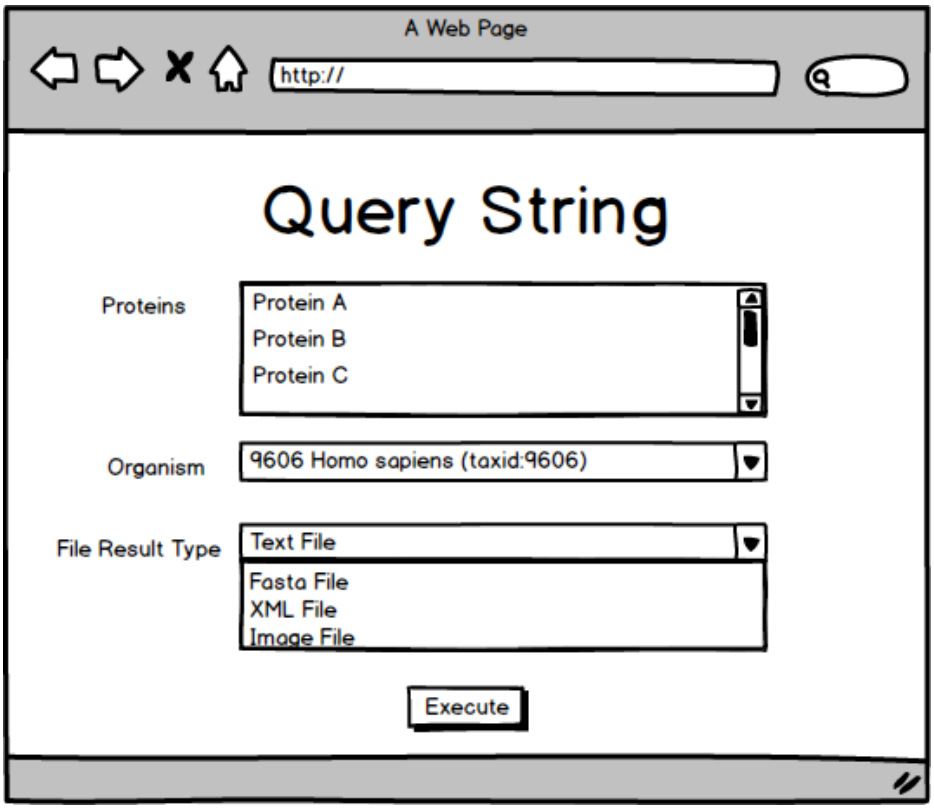
- Realizar a migração da interface para a plataforma *web*, de acordo com a técnica que será identificada para a implementação nas etapas seguintes;
- Remover a tela de configuração da ferramenta, pois a mesma não será mais utilizada;
- Realizar tratamento para os arquivos que são gerados através do processo analisado na plataforma *desktop*;
- Migrar a aplicação da versão 1.6 do Java para a versão mais recente;
- Providenciar hospedagem da aplicação *web*.

Para a realização da troca da plataforma *desktop* para a plataforma *web* será necessário realizar alterações na camada de visão do *software.* Como a aplicação foi desenvolvida baseando-se no padrão de arquitetura MVC, as alterações na aplicação serão executadas somente na camada de visão. Se necessário for serão realizadas adaptações de código nas camadas adjacentes.

Será realizada também a criação do novo leiaute para o pacote de visão, baseando-se na prototipação realizada entre a Figura 11 e Figura 14.

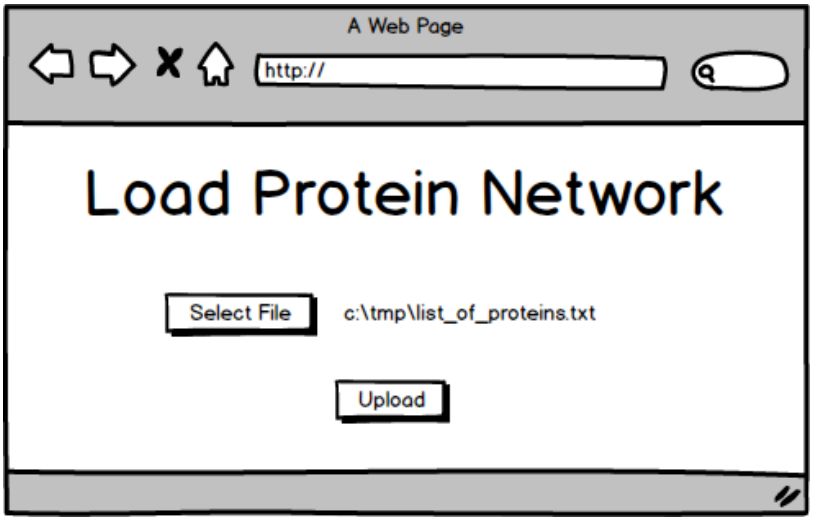
A Figura 11 mostra como será a nova tela de busca no STRING, aonde será possível selecionar uma ou mais proteínas, qual o organismo e em qual formato que será gerado o resultado, dentre as opções: texto, fasta, XML ou imagem.

Figura 11: Nova tela de consulta ao banco de dados STRING.



Fonte: autor.

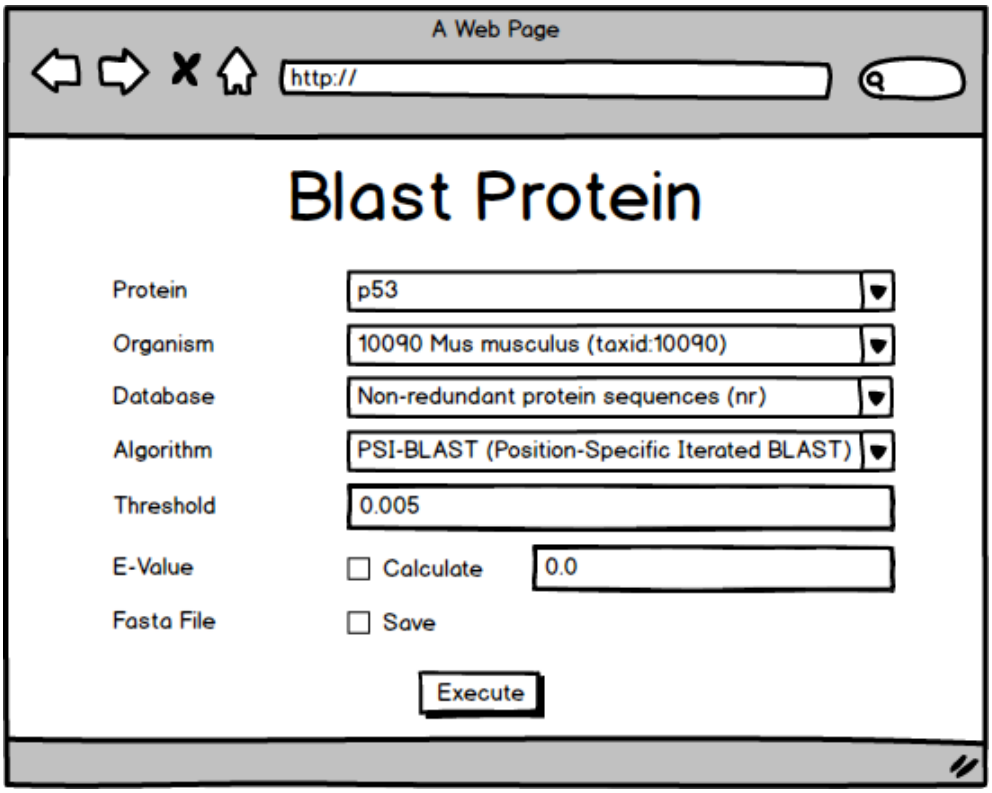
Figura 12: Nova tela de seleção de proteínas.



Fonte: autor.

Para a seleção de proteínas não existirá mais uma tela específica. Será desenvolvida uma nova tela (Figura 12) aonde será possível selecionar um arquivo texto contendo todas as proteínas que o usuário deseja importar. Estas proteínas serão carregadas automaticamente nos campos que serão referentes às proteínas.

Figura 13: Nova Tela de execução de um alinhamento de sequências.

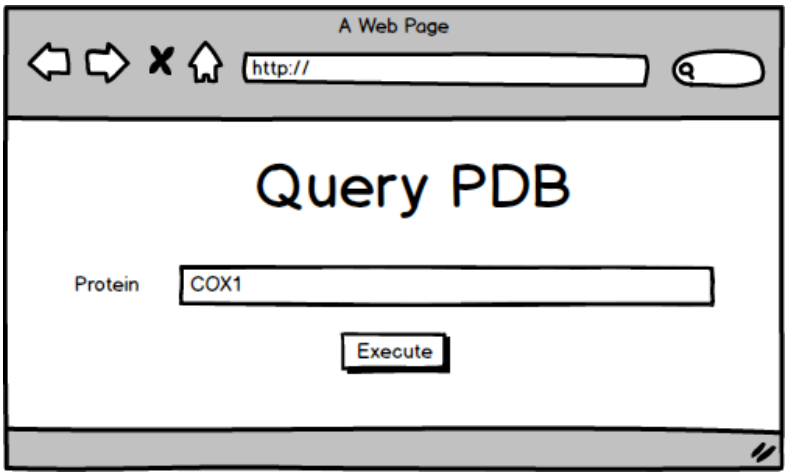


Fonte: autor.

Na tela para execução do BLAST (Figura 13) foi retirada a opção que o usuário possuía para escolher o formato que seria gerado o resultado da pesquisa. Agora o resultado será mostrado em uma nova aba no navegador.

A tela de busca no PDB será conforme protótipo apresentado na Figura 14, aonde o usuário irá escolher a proteína e o resultado será mostrado em novas abas no navegador.

Figura 14: Nova tela de consulta ao banco de dados PDB.



Fonte: autor.

Depois de realizada a implementação das alterações na ferramenta, será realizada a hospedagem da aplicação na *web*. O domínio aonde será hospedada a aplicação será o mesmo domínio onde a aplicação *desktop* está disponível [http://www.danlian.com.br/tools/].

A remoção da tela de configurações da aplicação é possível, pois a mesma solicita informar apenas informações sobre o caminho do navegador e informações sobre o *proxy*. Como a aplicação será migrada para a *web,* estas configurações já estão informadas no navegador que será utilizado.

Para realizar a migração da aplicação para a versão mais recente do Java, será utilizada a técnica de tradução de código-fonte utilizada na reengenharia de *software* citada no capítulo 2. A IDE (*Integrated Development Environment*) NetBeans, utilizada no desenvolvimento da aplicação *Bioinformatic tools,* permite que seja realizada essa migração.

4.2 ANÁLISE DE IMPACTO

A análise de impacto será realizada conforme critérios baseando-se em definições de Sommerville (2011) e Pfleeger (2004). Foram definidas cinco categorias que iniciam com um impacto baixo, representando uma simples alteração no código-fonte da aplicação, até um impacto alto que representa uma alteração complexa com diversos impactos na aplicação. As categorias são:

- **Baixo**: Alterações simples no código-fonte, baixo impacto na aplicação;
- **Médio-Baixo**: Alterações simples no código-fonte, com baixo impacto em alguns pontos da aplicação.
- **Médio**: Alterações com complexidade média no código-fonte, com um impacto médio em alguns pontos da aplicação.
- **Médio-Alto**: Alterações complexas no código-fonte, com alto impacto em um ponto da aplicação.
- **Alto**: Alterações complexas no código-fonte com alto impacto em diversos pontos da aplicação.

A análise de impacto será realizada de acordo com as alterações necessárias descritas anteriormente. As solicitações de mudança serão classificadas dentre as categorias descritas. Como a migração da versão do Java será realizada dentro da própria IDE do projeto inicial, a alteração terá um impacto baixo no projeto.

Para a realização da migração da interface da aplicação para a plataforma *web* serão necessárias diversas alterações, desta forma, esta alteração terá um

impacto médio-baixo. A remoção da tela de configuração terá um impacto baixo, pois suas funcionalidades já estão contempladas no navegador utilizado.

A alteração para que seja realizado o *download* dos arquivos gerados na aplicação terão um impacto médio-baixo. A hospedagem terá um impacto baixo, pois a aplicação já possui uma estrutura na *web* contendo manuais e *links* para realizar o *download* do Java. Desta forma será utilizada a mesma estrutura para hospedá-la.

Baseando-se nestas informações será realizado o processo de planejamento do *release*, detalhando as alterações e prevendo alcançar os objetivos anteriormente relatados.

4.3 PLANEJAMENTO DO RELEASE

Para realizar os processos de planejamento do *release* será realizada uma análise nos objetivos detalhados anteriormente e identificar quais serão as mudanças a serem realizadas.

Baseando-se na análise realizada o planejamento do *release* foi definido com as seguintes atividades:

- Migração da aplicação para a versão 1.8 do Java utilizando a IDE NetBeans;
- Implementação da mudança da interface para a plataforma *web* seguindo a prototipagem mostrada anteriormente;
- Implementação da possibilidade de *download* dos arquivos gerados pela aplicação;
- Realização de testes de funcionalidade, comparando a facilidade de utilização entre a aplicação *desktop* e a aplicação *web*.

A partir do planejamento finalizado, iniciou-se o processo de planejamento de migração da aplicação *Bioinformatic tools*.

5 IMPLEMENTAÇÃO

Dando continuidade à metodologia de evolução de software de Sommerville (2011) detalhada na sessão 3.1 e a proposta desenvolvida para realização do trabalho, deu-se início o processo de implementação da mudança do sistema.

As principais funcionalidades da aplicação a serem consideradas nessa mudança são apresentadas na Tabela 1. Para cada uma delas descreve-se como estão, e se estão, na versão desktop, e como ficarão na versão web.

Tabela 1: Funcionalidades da Aplicação.

Funcionalidade	Descrição	Versão Desktop	Versão Web
1	**Upload de proteínas para aplicação**		**Realizada mudança na camada de visão**
2	**Informações carregadas dinamicamente através de arquivos texto**	**Informações estavam fixas no programa**	**Funcionalidade desenvolvida**
3	**Consulta no BLAST**	**Webservice**	**URL**
4	**Consulta ao PDB**	**Webservice**	**URL**
5	**Consulta ao STRING**	**URL**	**URL; Mudança na camada de Visão**

Fonte: o Autor.

5.1 IMPLEMENTAÇÃO DA MUDANÇA

A implementação de mudança foi iniciada primeiramente com a definição de quais linguagens de programação seriam utilizadas. Foi definido que as linguagens utilizadas seriam HTML, JavaScript, PHP e CSS. A escolha destas linguagens se deu devido ao método de busca nos bancos do BLAST e PDB não serem mais realizadas por *Webservice* e sim por URL, posteriormente será explicado como foi realizada esta alteração de método de pesquisa. Devido esta mudança, não existe mais a necessidade de realizar processamentos custosos na aplicação, todo o processamento fica de responsabilidade do servidor aonde serão realizadas as pesquisas, a nova aplicação *web,* apenas monta as URL e busca as informações necessárias conforme retorno do servidor.

Conforme Tabela 1, foi desenvolvida com a função de permitir realizar um *upload* das proteínas via arquivo texto para a aplicação web (Funcionalidade 1), aonde

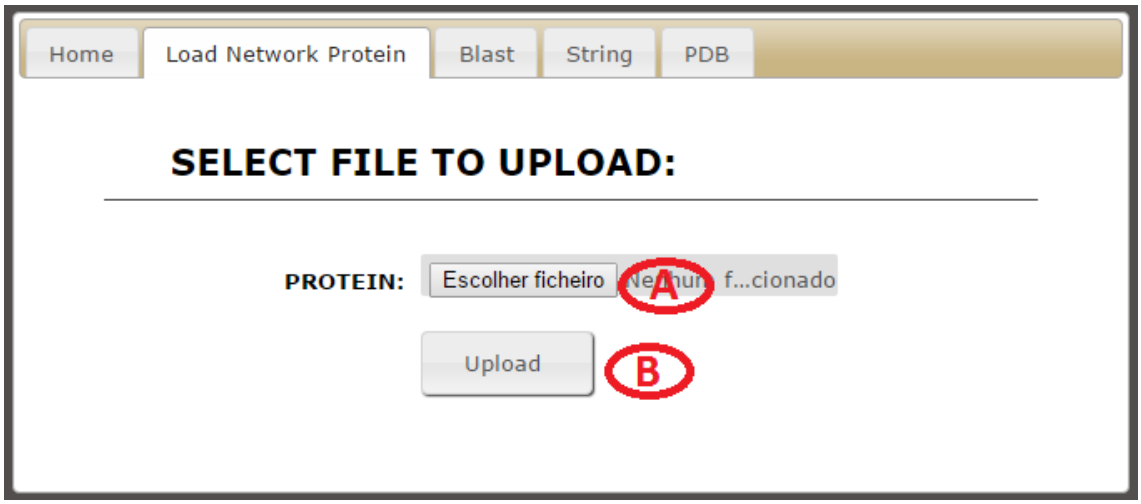
o usuário seleciona um arquivo texto contendo todas as proteínas que ele deseja importar para a aplicação (15A) e clica no botão “Upload” (15B). Após realizado este procedimento, em todos os lugares aonde existir a possibilidade de seleção de proteína, estará disponível para escolha estas proteínas que foram carregadas pelo arquivo texto.

Também foi adicionada uma função para permitir que as informações de organismos, algoritmos e bases sejam carregadas de forma dinâmica (Funcionalidade 2), aonde o usuário irá informar os dados via arquivo texto, e a aplicação irá buscar as informações de seus respectivos arquivos. Os layouts para importação foram definidos da seguinte forma:

- Lista de Organismos:
  - Arquivo texto separado por ponto e vírgula (;) com o nome “list_of_organism.txt”;
  - Código identificador do organismo;
  - Descrição do organismo;
  - Ex.: 10090;Mus musculus.
- Lista de Algoritmos:
  - Arquivo texto separado por ponto e vírgula (;) com o nome “list_of_algorithm.txt”;
  - Código identificador do algoritmo;
  - Descrição do algoritmo;
  - Ex.: psi;PSI-BLAST (Position-Specific Iterated BLAST).
- Lista de Base de Dados:
  - Arquivo texto separado por ponto e vírgula (;) com o nome “list_of_database.txt”;
  - Código identificador da base de dados;
  - Descrição da base de dados;
  - Ex.: nr;Non-redundant protein sequences.

Importante: os códigos de identificação serão utilizados para compor as URL de busca de informações, por este motivo eles devem estar de acordo com o que os sites de busca.

Figura 15: Nova Tela de Seleção de Proteínas.



Fonte: Autor.

Na Figura 16 é possível visualizar como são compostas as URL responsáveis pelas execuções de consulta do BLAST (Funcionalidade 3), aonde a URL (16A) é utilizada para buscar o identificador da proteína selecionada. A URL (16B) utiliza o identificador encontrado na URL anterior para conseguir identificar qual a sequência da proteína, que por sua vez é utilizada para compor a URL (16C) que possui o identificador de requerimento de busca que será utilizado na última URL (16D) que possuirá o resultado final da pesquisa.

Figura 16: URL de buscas no BLAST.

```
function executeBlast(){
  var term = $("#txtProtein").val() + "+AND+" + $("#cmbOrganism").children(":selected").text().replace(" ","+AND+");
  var url = "http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=protein&term=" + term;   A
  var proteinID = getProteinID(url);
  var url = "http://www.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=protein&id="+proteinID+"&complexity=0&rettype=text";   B
  var checked = document.getElementById("cbxSaveFastaFile").checked;
  if (checked) {
     window.open(requestStringFasta("http://www.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?db=protein&id="+
                 proteinID +"&complexity=0&rettype=fasta", $("#txtProtein").val()));
  }
  var sequence = getSequence(url);
  var url = "http://blast.ncbi.nlm.nih.gov/Blast.cgi?"+
  "QUERY=" + sequence +
  "&DATABASE=" + $("#cmbDatabase").val() +
  "&HITLIST_SIZE=10" +
  "&FILTER=L" +
  "&EXPECT=10" +
  "&FORMAT_TYPE=Text" +
  "&PROGRAM=blastp" +
  "&CLIENT=web" +
  "&SERVICE=" + $("#cmbAlgorithm").val() +
  "&NCBI_GI=on" +
  "&PAGE=Proteins" +
  "&I_THRESH="+ $("#txtThreshold").val() +
  "&ENTREZ_QUERY=txid"+ $("#cmbOrganism").val() +
  "&CMD=Put";                                                   C
  var rid = getRIDFile(url);
  window.open("http://blast.ncbi.nlm.nih.gov/Blast.cgi?RID=" + rid + "&CMD=Get&FORMAT_TYPE=Text");   D
}
```

Fonte: Autor.

Para buscar a informação do identificador de requerimento de busca foi necessário utilizar procedimentos encontrados na linguagem PHP devido a necessidade de busca da informação contida em uma página *web*, informação esta que não era obtida por nenhuma funcionalidade no JavaScript.

Figura 17: Comando PHP utilizado para buscar a informação do RID.

```php
<?php

$url = $_GET["url"];
$url2 = file_get_contents($url);
echo $url2;

?>
```

Fonte: Autor.

Outra mudança realizada foi na busca do PDB, aonde anteriormente era utilizado *Webservice* para se obter as informações das proteínas de diversos bancos de dados, dentre eles o PDB e o EMDataBank. Este Webservice foi desativado e as informações devem ser obtidas no PDB através de URL (Funcionalidade 4), por exemplo:

http://www.rcsb.org/pdb/images/4HHB_bio_r_500.jpg\

A composição da URL é informada o *site* do banco de dados (www.rcsb.org), o tipo de acesso (pdb/images), o identificador da proteína (4HHB) e o formato no qual a imagem será visualizada (_bio_r_500.jpg).

Para buscas no banco EMDataBank não foi localizado a nova forma de busca através de URL, esta funcionalidade pode ser pesquisada e/ou desenvolvida em outro projeto.

No processo de busca de informações no STRING (Funcionalidade 5) foram realizadas apenas mudanças de estruturação e linguagem de programação, a maneira de como as informações são obtidas foram mantidas.

Foram encontradas algumas falhas na nova versão encontradas durante os testes que já foram corrigidas antes da liberação do sistema. Mais detalhes sobre os testes estão disponíveis nas considerações finais deste trabalho.

6 ESTUDO DE CASO

Após finalizadas as mudanças de implementação, deu-se início ao estudo de caso das aplicações, utilizando as mesmas informações na versão *desktop* e na versão *web*, para poder realizar uma comparação de resultados.

Como pode ser visto na Figura 18, a nova tela para realizar o upload das proteínas para a aplicação sofreu mudanças visuais. Nela foi realizado o *upload* de um arquivo texto contendo diversas proteínas.

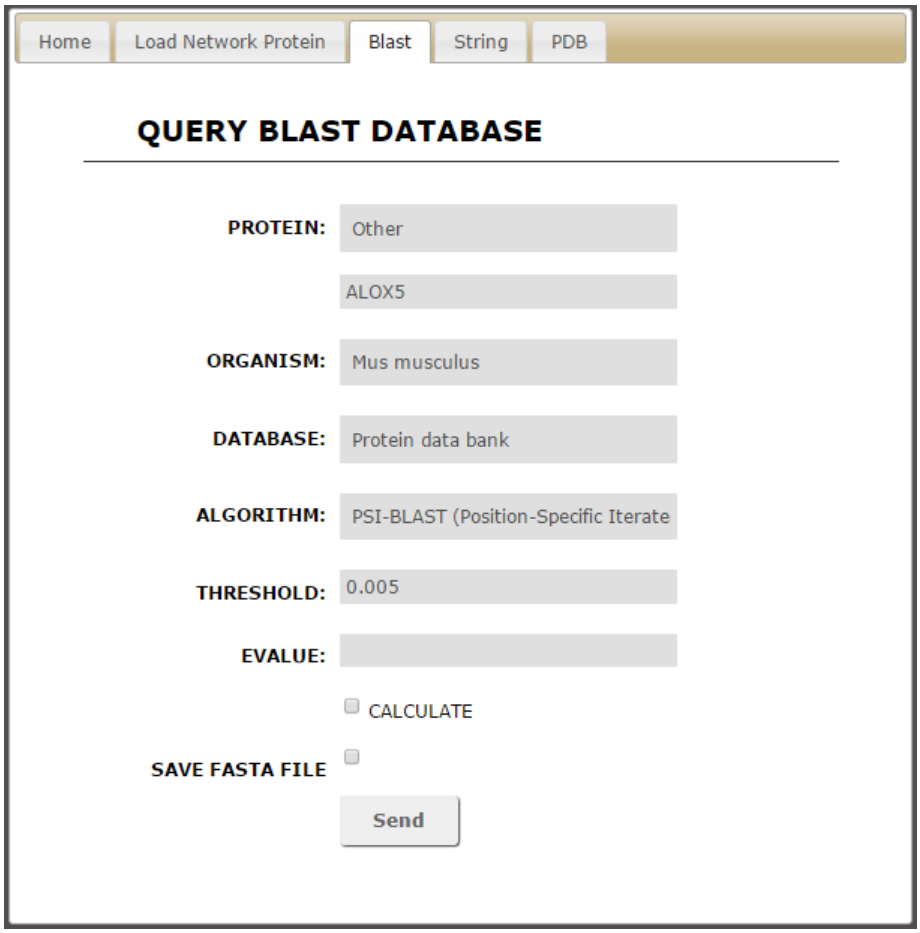
Figura 18: Nova tela de upload de proteínas.

Fonte: o Autor.

Ao clicar no botão “*Upload*” as proteínas que estão no arquivo texto serão importadas para a aplicação.

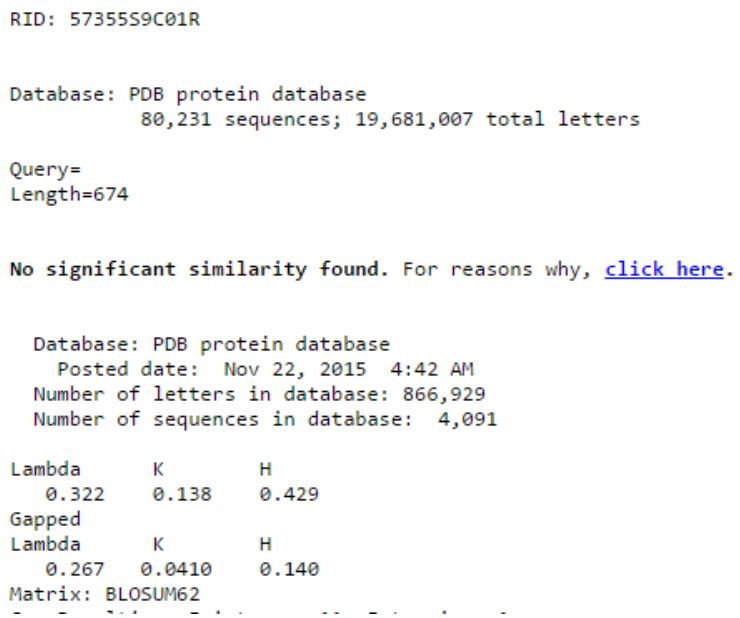
Na nova tela de busca no BLAST (Figura 19), foi realizada a pesquisa utilizando a proteína “ALOX5”, organismo “*Mus musculus*”, base de dados “*Protein data bank*” e o algoritmo “PSI-BLAST”. Ao pressionar o botão “*Send*” a pesquisa retornou as informações em uma nova aba do navegador, conforme pode ser visualizado na Figura 20.

Figura 19: Nova tela de busca ao BLAST.



Fonte: o Autor.

Figura 20: Resultado da pesquisa através do BLAST.



```
RID: 57355S9C01R


Database: PDB protein database
           80,231 sequences; 19,681,007 total letters

Query=
Length=674


No significant similarity found. For reasons why, click here.


  Database: PDB protein database
    Posted date:  Nov 22, 2015  4:42 AM
  Number of letters in database: 866,929
  Number of sequences in database:  4,091

Lambda      K        H
   0.322    0.138    0.429
Gapped
Lambda      K        H
   0.267   0.0410    0.140
Matrix: BLOSUM62
```

Fonte: o Autor.

Caso o *checkbox* “*Save Fasta File*” for marcado, a pesquisa retornará o arquivo FASTA, em uma nova aba no navegador, contendo as informações que podem ser visualizadas na Figura 21.

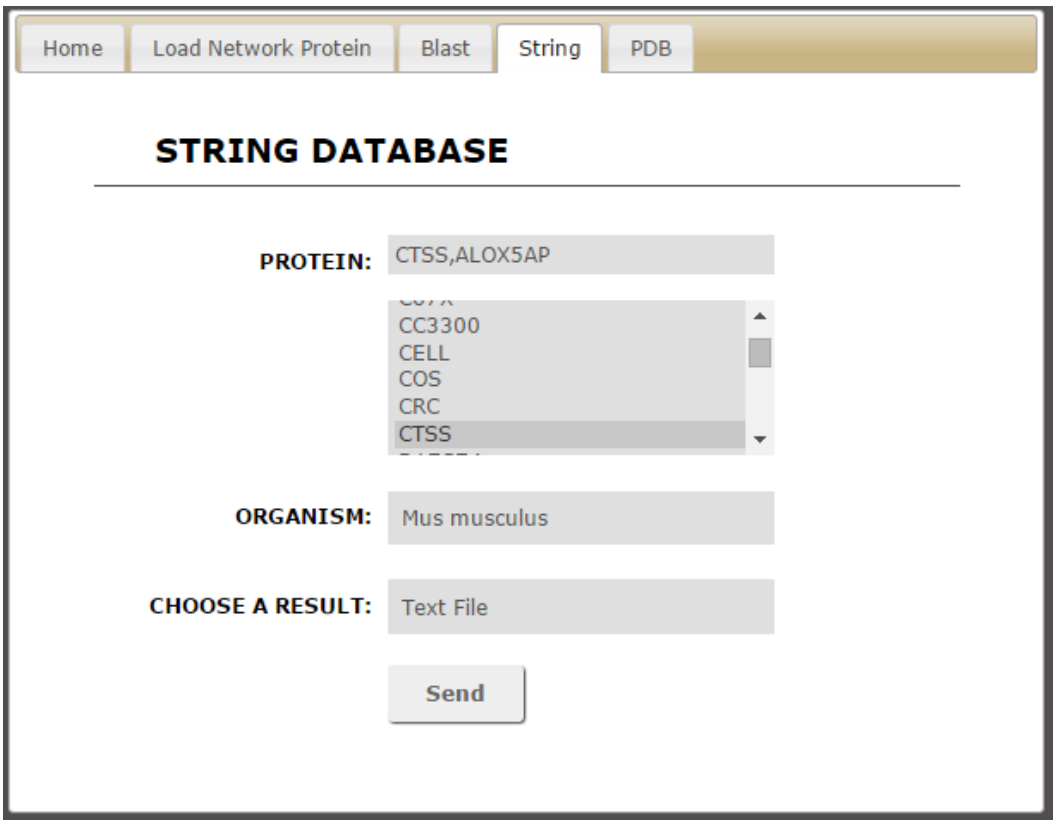
Figura 21. Arquivo FASTA resultante da pesquisa no BLAST.

```
>gi|341941017|sp|P48999.3|LOX5_MOUSE RecName: Full=Arachidonate 5-lipoxygenase; Short=5-LO; Short=5-lipoxygenase
MPSYTVTVATGSQWFAGTDDYIYLSLIGSAGCSEKHLLDKAFYNDFERGAVDSYDVTVDEELGEIYLVKI
EKRKYWLHDDWYLKYITLKTPHGDYIEFPCYRWITGEGEIVLRDGRAKLARDDQIHILKQHRRKELEARQ
KQYRWMEWNPGFPLSIDAKCHKDLPRDIQFDSEKGVDFVLNYSKAMENLFINRFMHMFQSSWHDFADFEK
IFVKISNTISERVKNHWQEDLMFGYQFLNGCNPVLIKRCTALPPKLPVTTEMVECSLERQLSLEQEVQEG
NIFIVDYELLDGIDANKTDPCTHQFLAAPICLLYKNLANKIVPIAIQLNQTPGESNPIFLPTDSKYDWLL
AKIWVRSSDFHVHQTITHLLRTHLVSEVFGIAMYRQLPAVHPLFKLLVAHVRFTIAINTKAREQLICEYG
LFDKANATGGGGHVQMVQRAVQDLTYSSLCFPEAIKARGMDSTEDIPFYFYRDDGLLVWEAIQSFTMEVV
SIYYENDQVVEEDQELQDFVKDVYVYGMRGKKASGFPKSIKSREKLSEYLTVVIFTASAQHAAVNFGQYD
WCSWIPNAPPTMRAPPPTAKGVVTIEQIVDTLPDRGRSCWHLGAVWALSQFQENELFLGMYPEEHFIEKP
VKEAMIRFRKNLEAIVSVIAERNKNKKLPYYYLSPDRIPNSVAI
```

Fonte: o Autor.

Na Figura 22 podemos visualizar como ficou a nova tela de busca ao STRING, pode-se notar também que as proteínas que foram importadas via arquivo texto, estão sendo mostradas na lista, aonde é possível selecionar uma ou mais proteínas desta lista ou também informar manualmente uma que não esteja nela, basta apenas separar as proteínas por vírgulas.

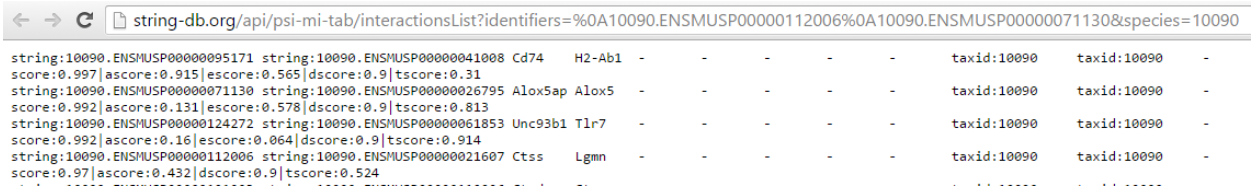
Figura 22: Nova tela de busca ao STRING.



Fonte: o Autor.

Para este caso de uso foi utilizado uma proteína que estava na lista “CTSS” e uma outra que foi inserida manualmente “ALOX5AP”, selecionado o organismo e qual o formato que a aplicação irá mostrar o resultado (Texto, FASTA, XML ou Imagem), neste caso foi escolhido em formato Texto. Após pressionado o botão “*Send*” a pesquisa retornou as informações apresentadas na Figura 23.

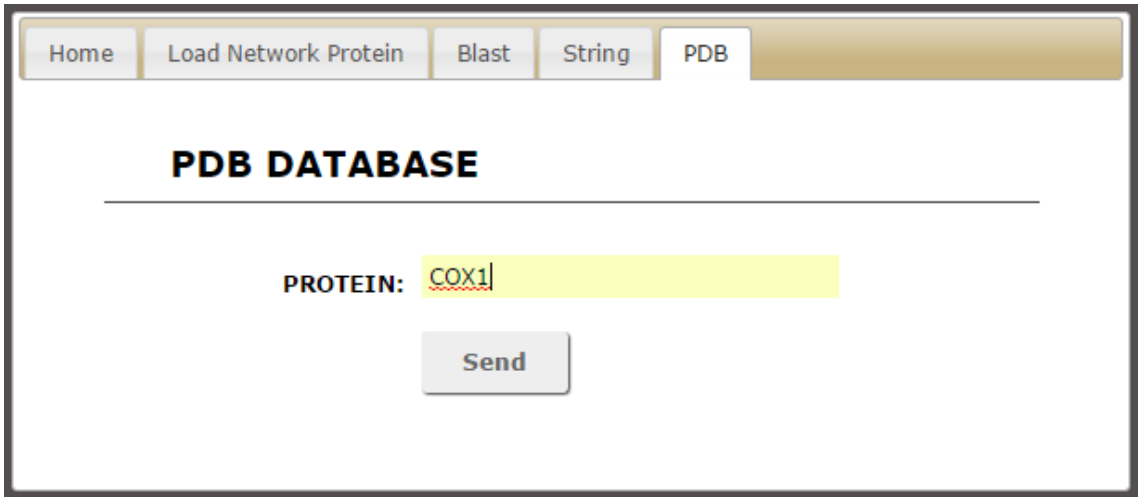
Figura 23: Resultado da pesquisa no STRING formato texto.



Fonte: o Autor.

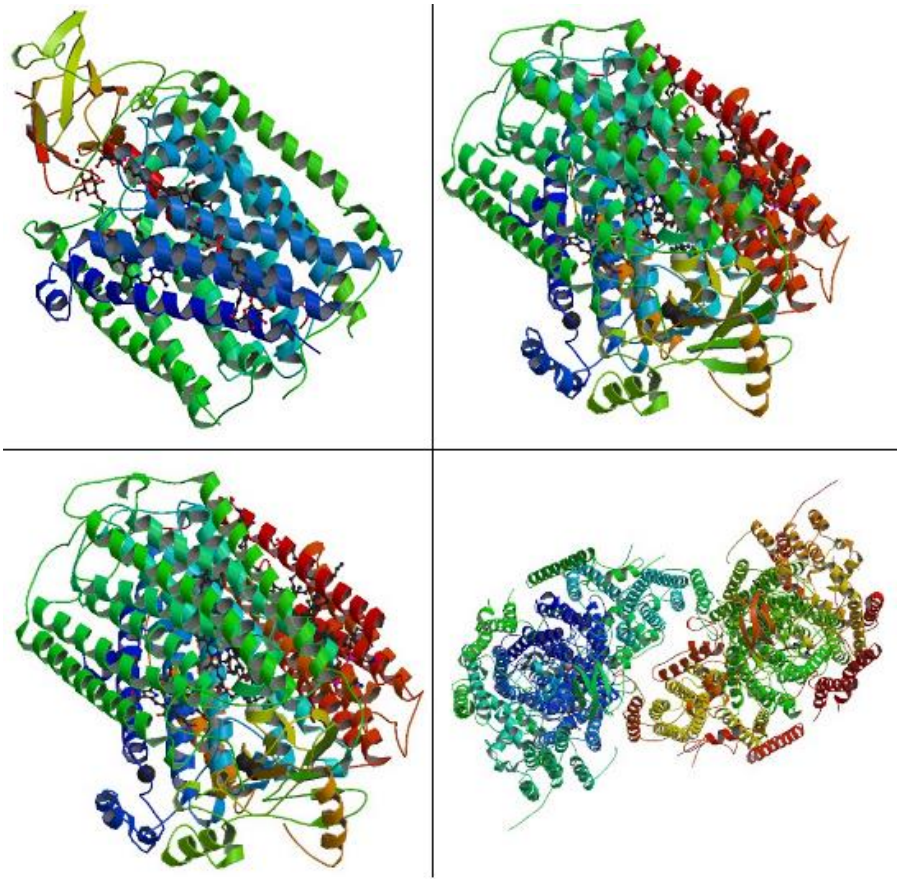
Por último, a Figura 24 mostra a nova tela de busca ao PDB, aonde foi escolhida a proteína “COX1”. Ao pressionar o botão “*Send*” a aplicação irá retornar a estrutura de uma ou mais proteínas em formato tridimensional (caso a proteína possua mais de quatro proteínas na sua estrutura, a aplicação irá mostrar apenas as primeiras quatro). O resultado desta pesquisa pode ser visto na Figura 25.

Figura 24: Nova tela de busca ao PDB.



Fonte: o Autor.

Figura 25. Resultado da busca do PDB para a proteína COX1.



Fonte: o Autor.

7 CONCLUSÕES

Após a aplicação dos estudos de caso concluiu-se que:

i) O objetivo principal deste projeto foi alcançado, a migração da aplicação *desktop* para a plataforma *web* está funcionando e mostrados os resultados esperados;
ii) As ferramentas que sofreram mudanças na sua forma de pesquisa estão funcionando de forma satisfatória e com uma performance melhor em geral, sujeita à disponibilidade e velocidade de internet;
iii) Não existe mais a necessidade de realizar qualquer configuração de *proxy,* deixando a aplicação acessível a qualquer profissional da biologia sem precisar ter conhecimentos de informática para instalação e utilização da ferramenta;
iv) A aplicação ficou menos engessada devido as funcionalidades adicionadas como por exemplo o carregamento de informações através de arquivos texto;
v) A aplicação foi homologada para os navegadores *Google Chrome, Mozilla Firefox e Internet Explorer* deixando a aplicação disponível para diversos navegadores;
vi) A aplicação ficou com um aspecto mais amigável ao usuário, com um visual mais moderno do que o desenvolvido na versão *desktop.*

Além disto foram identificadas melhorias que podem ser implementadas neste projeto, tais como:

i) Controle de usuários com registro de *login* e consultas realizadas;
ii) Busca de proteínas no EMDataBank, funcionalidade que não foi desenvolvida por falta de informações no site da ferramenta.

As dificuldades apresentadas durante o desenvolvimento do projeto foram apenas com as linguagens de programação, as quais não se tinha muito conhecimento, mas com pesquisas na internet em fóruns as dificuldades foram superadas e o trabalho pode ser implementado de maneira satisfatória.

8 REFERÊNCIAS

AMBERGER, Joanna et. al. McKusick's online Mendelian inheritance in man (OMIM®). Nucleic acids research, v. 37, n. suppl 1, p. D793-D796, 2009.

BARNES, Michael R.; GRAY, Ian C. (Ed.). Bioinformatics for geneticists. John Wiley & Sons, 2003.

BASS L., CLEMENTS P.; Kazman, R. Software Architecture in Practice. Addison-Wesley Professional, 2ª edição, Abril 2003.

KARP, Gerald. Biologia celular e molecular. Editora Manole Ltda, 2005.

NOTARI, Daniel Luis et. al. Dis2PPI: A Workflow Designed to Integrate Proteomic and Genetic Disease Data. International Journal of Knowledge Discovery in Bioinformatics (IJKDB), v. 3, n. 3, p. 67-85, 2012.

PRESSMAN, Roger S. Engenharia de software: uma abordagem profissional. 7ª Edição. Ed: McGraw Hill, 2011.

PFLEEGER, Shari Lawrence. Engenharia de software: teoria e prática. 2ª Edição, Prentice Hall, 2004.

SOMMERVILLE, Ian. Engenharia de Software. 9º ed.: Addison Wesley, 2011. 532 pp.