UNIVERSIDADE FEDERAL DO PARANÁ

ALLAN COSTA VIEIRA
FERNANDO DE OLIVEIRA MELO

SISTEMA PARA GERENCIAMENTO DE PROJETOS DE SEQUENCIAMENTO DE FUNGOS

CURITIBA
2016

ALLAN COSTA VIEIRA
FERNANDO DE OLIVEIRA MELO

SISTEMA PARA GERENCIAMENTO DE PROJETOS DE SEQUENCIAMENTO DE FUNGOS

Trabalho de conclusão de curso apresentado como requisito à conclusão do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnologia, Universidade Federal do Paraná.

Orientadora: Prof.ª Dr.ª Jeroniza Nunes Marchaukoski

Coorientador: Dr. Vinícius Weiss

CURITIBA
2016

RESUMO

Apesar dos fungos serem extremamente importantes na decomposição da matéria orgânica e no ciclo dos nutrientes, eles estão envolvidos em várias doenças que acometem seres humanos, por isso é de grande importância o estudo do organismo dos fungos realizado pelo PPG-BIOINFO, que tem como objetivo o sequenciamento genético de fungos patogênicos para melhor compreensão das causas da patogenicidade e possíveis curas.

Arquivos GBK, GFF e Fasta podem guardar as mais diversas informações genéticas de um organismo, sendo amplamente utilizados na Bioinformática, porém, cada formato de arquivo as armazena de forma diferente. Este trabalho apresenta um sistema web capaz de armazenar e converter os arquivos do formato GBK para os formatos Fasta, assim como disponibiliza-los para download. O formato GBK é de propriedade do NCBI GenBank (NCBI, 2016), um dos mais utilizados bancos de dados primários, e que tem centralizado boa parte das informações obtidas pelos processos de sequenciamento de DNA e/ou RNA de fungos patogênicos, separando em arquivos únicos e formato específico cada organismo, cromossomo ou plasmídeo completamente sequenciado com seus genomas e anotações. Por isso também a preferência por trabalhar nesse arquivo como fonte primária de disponibilização e armazenamento.

O sistema também funcionará como uma plataforma de trabalho, facilitando o armazenamento e visualização de arquivos genéticos e possibilitando que o andamento dos projetos seja facilmente consultado, arquivos disponibilizados e usuários gerenciados de acordo com uma hierarquia estabelecida.

Palavras chave: Bioinformática, Sequenciamento de DNA, Visualização de dados biológicos, NCBI, Fungos Patogênicos, Compartilhamento de Dados de Sequenciamento

ABSTRACT

Although fungi are extremely important in the decomposition of organic matter and in the nutrient cycle, they are involved in several diseases that affect human beings, so it is of great importance the study of the fungi organism carried out by PPG-BIOINFO, which aims to genetic sequence pathogenic fungi to better to understand the causes of pathogenicity and possible cures.

GBK, GFF and Fasta files can store the most diverse genetic information of an organism, being widely used in bioinformatics, and each file format has different ways of storing it. This paper presents a web system capable of store and convert files from GBK to Fasta, and offers them for download. The GBK format is owned by NCBI GenBank (NCBI, 2016), one of the most used primary databases, and has centralized much of the information obtained by processes of DNA sequencing and / or RNA of pathogenic fungi, separating it in unique file and specific format each organism, chromosome or plasmid completely sequenced with their genomes and annotations. Therefore, the preference to work with this kind of file as availability and storage primary source of.

The system will also behave as a working platform, facilitating the storage and visualization of genetic files and allowing querying projects progress, make files available and manage users according to an established hierarchy.

Keywords: Bioinformatics, DNA Sequencing, Visualization of biological data, NCBI, Pathogenic Fungi, Sequencing Data Sharing

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AJAX	-Asynchronous JavaScript and XML
DNA	-Ácido desoxirribonucleico
FASTA	-Formato utilizado para armazenar sequências de bases e de Aminoácidos em arquivo texto
GenBank	-Banco de dados público do National Center for Biology Information
GBK	-GenBank File
GFF	-General Feature Format
HTML	-HyperText Markup Language
IDE	-Integrated Development Environment
MVC	-Model-View-Controller
NCBI	-National Center for Biotechnology Information
NR	-Banco de dados não redundante disponibilizado pelo NCBI
PPG-BIOINFO	-Programa de Pós-graduação em Bioinformática
RNA	-Ácido ribonucleico
RUP	-Rational Unified Process
SGBD	-Sistema Gerenciador de Bancos de Dados
UML	-Unified Modeling Language
WBS	-Work Breakdown Structure
ZIP	-Formato de arquivo resultante de uma compressão de um ou mais arquivos e diretórios

SUMÁRIO

1. INTRODUÇÃO

O Programa de Pós-graduação em Bioinformática de Universidade Federal do Paraná foi criado em 2009 com intenção de atender a grande demanda por profissionais capazes de convergir entre os estudos de biologia molecular e computação, aprofundando-se cada vez mais no campo interdisciplinar chamado Bioinformática (PPG-BIOINFO, 2016).

A Bioinformática é uma ciência que atua na manipulação e extração de conhecimento de dados biológicos. O crescimento da área deve-se à necessidade de manipular quantidades enormes de dados genéticos e bioquímicos, utilizando recursos computacionais para catalogá-las, organizá-las e estruturá-las (RASHIDI, 2000).

1.1 CONTEXTUALIZAÇÃO

Segundo o levantamento realizado por Juliana Giacomazzi, da Universidade Federal de Ciências da Saúde de Porto Alegre, quase 4 milhões de pessoas no Brasil devem ter infecções fúngicas a cada ano (GIACOMAZZI, 2016), e os sistemas afetados podem ser tão superficiais quanto à camada externa da pele, ou tão profundas quanto o coração, o sistema nervoso central, ou as vísceras abdominais, e sendo que várias espécies de fungos vëm se mostrando resistente aos poucos medicamentos utilizados para combatê-los.

Por isso é de suma importância o trabalho realizado no PPG-BIOINFO com o sequenciamento genético de fungos patogênicos, afim de prover uma base para pesquisa de medicamentos mais eficazes.

Atualmente o NCBI Bank é o banco de dados mais utilizado para a publicação dos resultados de sequenciamento genéticos, porém, falta um local específico para armazenamento dessas informações enquanto a realização do trabalho. Agravando a situação, um dos principais bancos de dados de sequenciamento genético de fungos negros, o *Black Yeast Database* (BROAD INSTITUTE, 2016), esteve indisponível ao longo do desenvolvimento do nosso projeto, conforme FIGURA 37, no apêndice F deste trabalho. É neste ponto que pretendemos ofertar uma solução

para esse problema, propondo um banco de dados para armazenamento de arquivos genéticos durante o período da pesquisa.

1.2 JUSTIFICATIVA

O PPG-BIOINFO necessita cada vez mais de ferramentas que facilitem o trabalho de pesquisa e armazenamento das informações trabalhadas, e, visando criar um banco de dados para armazenamento das informações genéticas de fungos patogênicos, esperamos contribuir de maneira efetiva para que o trabalho de todo o grupo de pesquisa seja facilitado.

Nossa plataforma pretende trazer para o grupo, além de um o novo local para o armazenamento de informações, também uma nova forma de organização de trabalho, possibilitando assim um maior aproveitamento do tempo de seus colaboradores.

1.3 OBJETIVOS

Desenvolver uma plataforma web capaz de armazenar arquivos genéticos, organizando-os de acordo com o organismo estudado e o projeto em que se encontra, possibilitando compartilhar essas informações de maneira segura e dinâmica entre os membros do projeto e ao público quando necessário, e fazer com que essa plataforma tenha um sistema de gerenciamento hierárquico de usuários, possibilitando seguir os moldes organizacionais do PPG-BIOINFO, tendo o coordenador uma ferramenta capaz de não somente armazenar e compartilhar informação, como também gerenciar seus grupos de trabalho.

1.4 OBJETIVOS ESPECÍFICOS

Este trabalho tem os seguintes objetivos específicos:

- Possibilitar ao sistema fazer upload de arquivos GBK
- Possibilitar ao sistema fazer a conversão de arquivos GBK para Fasta

- Possibilitar ao sistema armazenar esses arquivos de forma compactada, afim de reduzir o espaço ocupado no banco de dados e possibilitar que os downloads sejam feitos de forma mais rápida pelos usuários
- Possibilitar ao visitante o acesso a dados públicos
- Possibilitar ao visitante o download de arquivos genéticos que tenham sido definidos como públicos
- Possibilitar ao visitante solicitar acesso de coordenador para aprovação de um administrador
- Possibilitar ao visitante solicitar acesso de pesquisador a um projeto para aprovação de um coordenador
- Possibilitar ao administrador do sistema aceitar ou rejeitar solicitações públicas de acesso como coordenador
- Possibilitar ao administrador atribuir um projeto a outro coordenador
- Possibilitar ao administrador editar e excluir coordenadores
- Possibilitar ao coordenador cadastrar, editar e excluir um projeto
- Possibilitar ao coordenador editar uma equipe
- Possibilitar ao coordenador aceitar ou rejeitar solicitação de pesquisadores ou convidados para um projeto
- Possibilitar ao coordenador editar um pesquisador
- Possibilitar ao coordenador incluir, editar e excluir um organismo
- Possibilitar ao coordenador fazer upload de arquivos genéticos de um organismo
- Possibilitar ao coordenador o download de arquivos genéticos de seus projetos
- Possibilitar ao pesquisador o acesso aos dados do projeto em que faz parte
- Possibilitar ao pesquisador o download de arquivos genéticos de seus projetos
- Possibilitar ao pesquisador solicitar acesso de convidado a um projeto de um outro coordenador

2. METODOLOGIA

A função deste capítulo é apresentar as metodologias e ferramentas que foram utilizadas para o desenvolvimento deste aplicativo. Também serão mostradas as responsabilidades dadas a cada membro.

2.1 MODELO DE PROCESSO DE ENGENHARIA DE SOFTWARE

Neste projeto foi utilizado o modelo RUP (*Rational Unified Process*), que fornece diretrizes para definir as tarefas e atribuir responsabilidades aos membros do projeto. O RUP possui quatro fases: iniciação, elaboração, construção e transição. Na iniciação define-se o escopo do projeto. Na elaboração é obtida uma visão abrangente do sistema, através da construção de protótipos e também é definida a arquitetura do sistema. Na construção, o foco está no desenvolvimento do sistema. Por fim, na transição, o produto é transferido ao usuário e o projeto é avaliado e pode ser concluído. Em todas as fases há o gerenciamento dos requisitos e dos recursos do projeto. Por isso o RUP é baseado no desenvolvimento iterativo, que por sua vez é mais flexível quanto às mudanças de escopo durante o desenvolvimento do projeto (IBM, 2014).

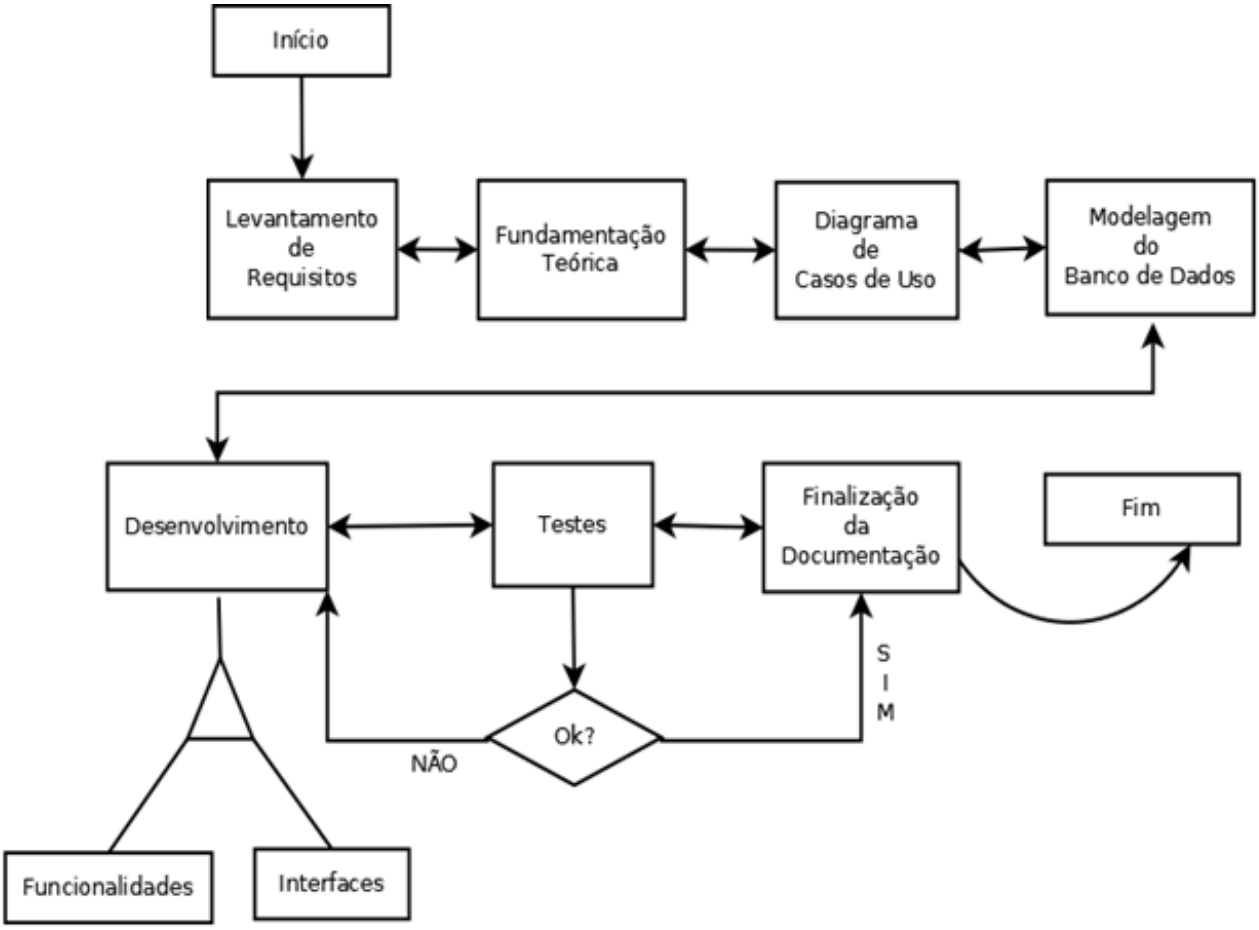

FIGURA 01 – METODOLOGIA DE DESENVOLVIMENTO.

Fonte: Os autores (2016).

2.2 MATERIAIS

2.2.1 Ambiente de Desenvolvimento

Para o desenvolvimento do aplicativo, foram utilizados dois computadores:

- Notebook CyberPower MS16-GC, processador Intel Core i7-4810MQ 2.80GHz, memória RAM 16GB, placa de vídeo Nvidia GTX 860M4GB, sistema operacional Windows 10 Home;
    - Oracle VM Virtual Box, Ubuntu 16.04
- Notebook Dell Latitude 3450, processador Intel Core i55200U 2.20GHz, memória RAM 8GB, placa de vídeo Intel HD Graphics 5500, sistema operacional Windows 8.1 Pro;
    - Oracle VM Virtual Box, Ubuntu 16.04

2.2.2 Linguagem de Programação

A linguagem de programação escolhida pelos usuários foi Ruby (versão 2.3.0p0). O Ruby é uma linguagem relativamente nova, criada em 1995 por Yuri Matsumoto, podendo ser base para diversos paradigmas de programação. Escolhemos trabalhar com orientação a objetos para poder aplicar boa parte do conhecimento adquirido em aula no curso de Análise e Desenvolvimento de Sistemas (RUBY, 2016).

A escolha dessa linguagem se deu pela necessidade do grupo de Pós-graduação em Bioinformática da UFPR em unificar o desenvolvimento de novas ferramentas sobre uma única linguagem, facilitando assim a manutenção e o aprendizado.

2.2.3 Framework de Desenvolvimento

Para um melhor uso da linguagem Ruby, utilizamos o framework Ruby on Rails (versão 4.2.6) durante o desenvolvimento do projeto.

Este framework facilita o desenvolvimento, implantação e manutenção, construindo uma aplicação web dentro da arquitetura MVC (RUBY ON RAILS, 2016).

2.2.4 Banco de Dados

Para implementação do banco de dados foi escolhido o PostgreSQL (versão 9.3.13), pois é uma ferramenta de fácil uso, com bastante documentação disponível e bem-conceituada no meio acadêmico e comercial, sendo já utilizado no PPG-BIOINFO em outros projetos.

O PostgreSQL é um SGBD que implementa um banco de dados SQL relacional. Além disso, é um software livre de domínio público e multiplataforma (POSTGRESQL, 2016), e através do framework Rails pode integrar-se de maneira conjunta ao desenvolvimento da aplicação (ANICAS, 2016).

2.2.5 Ferramenta para Modelagem UML

A UML (*Unified Modeling Language*) é uma linguagem visual utilizada para modelar software baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. Essa linguagem tornou-se, nos últimos anos, a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software (GUEDES, 2011).

Neste trabalho utilizamos o diagrama de caso de uso, o diagrama de classe, o diagrama de estados, o diagrama de atividades e o diagrama de telas. O diagrama de caso de uso permite organizar os requisitos do sistema dando uma visão geral. O diagrama de classes representa a estrutura do sistema orientado a objetos e o diagrama de telas apresenta o fluxo das atividades do aplicativo.

Para a criação dos diagramas UML utilizamos o Astah (versão 7.0), que é uma ferramenta específica para a criação desses diagramas. O Astah está há mais de 10 anos disponível no mercado (primeiramente com o nome JUDE) e com ele é possível criar todos os diagramas de uma forma simples, intuitiva e padronizada. A licença utilizada foi a de estudante, gratuita pelo período de um ano, solicitada no próprio site da Astah (ASTAH, 2016).

2.2.6 Ferramenta para Criação do Diagrama Entidade Relacionamento

A ferramenta escolhida para a modelagem do diagrama de entidade e relacionamento (Diagrama ER) foi o aplicativo Draw.io (website), que permite a criação on-line de diagramas (DRAW.IO, 2016).

2.2.7 Ferramenta para Controle de Versão

O GIT é um sistema de controle de versão de softwares focado no desenvolvimento envolvendo um ou mais desenvolvedores. Ele também é um sistema de gerenciamento de código fonte, ou seja, todos os membros da equipe podem enviar alterações. Todos os arquivos e todo histórico são armazenados num repositório único (GIT, 2016).

Neste projeto foi utilizado o GitHub (versão desktop 3.3.3.0), que além de gerenciar as versões e as alterações no código, possui um repositório online, facilitando as alterações para os membros da equipe e o acesso a informações necessárias a qualquer momento.

2.2.8 Ferramenta para Criação de Wireframe

Para a criação da estrutura do wireframe das telas, foi utilizado o software Balsamiq Mockups (versão 3.5.5). Com ele, conseguiu-se de uma forma ágil e realista o desenvolvimento do layout modelo, tudo isso devido às facilidades trazidas pela ferramenta. Como os mockups dão uma ideia inicial das telas do projeto, utilizamos somente a versão de teste de 30 dias do software, pois não entendemos ser necessário prolongar a extensão da licença (BALSAMIQ, 2016).

2.2.9 Ferramenta para Virtualização

A Virtual Machine da Oracle, VirtualBox (versão 5.0.20), é um software gratuito de virtualização. Com ela é possível rodar os mais diversos sistemas operacionais sem a necessidade de modificação ou emulação, possibilitando assim o desenvolvimento em um sistema Linux e permitindo assim termos um sistema mais parecido com o do PPG-BIOINFO do que o disponível em nossas máquinas (ORACLE, 2016).

2.2.10 RubyGems

Gems são bibliotecas utilizadas para resolver diversos tipos de problema no Ruby. Os desenvolvedores utilizam as Gems para não precisarem criar uma solução já existente.

Além dela outras gems foram utilizadas durante o projeto. As seguintes gems não se encontram no pacote do Ruby on Rails e tiveram que ser baixadas separadamente (RUBYGEMS, 2016):

- Bcrypt (3.1.11)
  - Biblioteca utilizada na criptografia de senhas armazenadas no banco, para que não fiquem como texto plano (menos seguro)
- Bio (1.5.0)
  - Biblioteca com funções específicas de biologia, mais especificamente genética (alinhamento, extração de características, etc.). Não é obrigatória, apenas se desejar implementar outras funções.
- Jquery-rails (4.1.1)
  - Necessária para a inclusão da biblioteca JQuery (JavaScript).
- JQuery-UI-Rails (5.0.5)
  - Utilizada para biblioteca de interface do sistema.
- JSON (1.8.3)
  - Serialização de dados e utilização de estruturas de dados para respostas através de AJAX (requisição assíncrona).
- PG (0.18.4)
  - Gem responsável pelo uso do banco de dados PostgreSQL.
- Rails (4.2.6)
  - Gem atualizada / Versão compatível com o projeto.
- Rake (11.2.2, 10.5.0)
  - Comandos para geração de conteúdo, inserção de dados no banco (como no comando *rake db:seed*) e alterações de tabelas / classes existentes através de *migrations* (*rake db:migrate*).
- Schema_to_scaffold (0.7.2)
  - Traduz objetos criados para o modelo MVC (*model, view* e *controller*), ou seja, cria os arquivos necessários.
- Sqlite3 (1.3.11)
  - Necessário para a compilação das gems: JQuery e JQuery-UI-Rails.

3. DESENVOLVIMENTO DO PROJETO

O projeto começou com a ideia de criação de um banco de dados capaz de armazenar o resultado de um parse de um arquivo genético com a intenção de ter as características de um organismo separadas e com isso não precisar recorrer ao arquivo completo todas as vezes que fosse necessário consultar uma informação. Porém, mudanças foram feitas no escopo ao decorrer do projeto e a ideia do parse foi deixada para uma implementação futura.

Com isso o projeto passou de um sistema de armazenamento de *parsers* para uma plataforma de trabalho, incluindo não somente as funcionalidades de download, conversão e upload de arquivos, como também o gerenciamento de usuários por hierarquia organizacional.

Um segundo problema relacionado aos arquivos genéticos era o seu tamanho. Podendo ocupar centenas de megabytes, armazená-los e transportá-los poderia ocupar bastante espaço e recurso computacional, por isso propusemos a compactação desses arquivos e implementamos essa funcionalidade. Devido ao fato dos arquivos conterem diversas sequências de caracteres repetidos a compactação atinge números excelentes, possibilitando desperdiçar menos memória no banco de dados e recurso computacional para transporte.

Outro problema recorrente durante o desenvolvimento, principalmente na fase de implantação do projeto no servidor da PPG-BIOINFO, foi a queda, por vezes constante, da conexão com o servidor pela VPN. Diversas vezes foi necessário efetuar um retrabalho devido a essas quedas.

A divisão de tarefas foi mudando de acordo com as mudanças requisito e aproximação do prazo final. Apesar das diversas mudanças, a equipe conseguiu se adaptar a elas.

A equipe decidiu retirar os diagramas de sequência e substitui-los por diagramas de tela, pois além de não termos sequencias tão grandes entre as classes os diagramas de telas mais fáceis de serem lidos e ilustram o que ocorre de forma mais simples e intuitiva. Além disso optamos por ter somente o diagrama de estado da inclusão de novo arquivo para o organismo, pois consideramos somente esse com um comportamento dinâmico relevante.

3.1 COMPACTAÇÃO DOS ARQUIVOS

Quando um novo arquivo GBK é carregado no sistema, ocorre uma compressão do seu conteúdo original e também do conteúdo que foi convertido (sofreu *parse* para o formato Fasta, adiante, no item 3.2). A classe responsável por tal operação é a GenFile.rb (Arquivo Ruby), conforme imagem:

```
135             # Caminhos de arquivos
136             f_nome = "#{Rails.root}/tmp/fasta.tmp"
137             f_zip  = "#{Rails.root}/tmp/fasta.zip"
138
139
140             # Grava o arquivo em um diretório temporário no servidor
141             f_tmp = File.new(f_nome,'w')
142             f_tmp.write(@str_fasta)
143             f_tmp.close
144
145             # Limpa o buffer para liberar memoria RAM do servidor
146             @str_fasta = ""
147
148
149             # Nome do arquivo que vai dentro do ZIP
150             mask    = ""
151             fz_nome = ""
152             if mask = f.original_filename.match(/(.+)([.].+)/)
153                 fz_nome = mask.captures[0]
154                 fz_nome = fz_nome.to_str
155                 fz_nome = fz_nome + ".fasta"
156             end
157
158             # Se um arquivo ZIP existir, exclui
159             if File.exist?(f_zip)
160                 File.delete(f_zip)
161             end
162
163             # Compacta o arquivo FASTA
164             #Zip.on_exists_proc = true
165             Zip::File.open(f_zip, Zip::File::CREATE) do |z|
166                 z.add(fz_nome, f_nome)
167                 # z.get_output_stream("myFile") { |os| os.write "myFile contains just this" }
168             end
169
170
171             # Apaga o arquivo .tmp
172             if File.exist?(f_nome)
173                 File.delete(f_nome)
174             end
175
```

FIGURA 02 – TRECHO DE CÓDIGO RUBY, COMPACTAÇÃO.

Fonte: Os autores (2016).

De uma forma sucinta, o algoritmo demonstrado na imagem acima se comporta da seguinte maneira:

O *stream* (sequência binária) do arquivo (acima representado como Fasta, porém ocorre o mesmo para o formato GBK) é salvo em um arquivo temporário no diretório "*tmp*" da aplicação, entre as linhas 136 e 146. A seguir, é extraído do arquivo original (o qual o usuário selecionou para upload) o nome deste para ser utilizado como nome do arquivo que estará dentro do ZIP, entre as linhas 150 e 156. É verificado então se já existe um arquivo com mesmo nome no diretório temporário, se houver, este será excluído para que o novo seja criado no lugar, linhas 159 a 161. Um arquivo ZIP vazio é então criado no diretório temporário, ele recebe o arquivo texto (Fasta ou GBK) que acabou de ser criado, linhas 165 a 168, e depois o arquivo texto é deletado no trecho que segue entre as linhas 171 a 173.

O resultado desta operação será um arquivo compactado contendo o arquivo texto que o usuário escolheu (e também o formato Fasta, convertido) para um determinado organismo (verificar o procedimento operacional no item 4 deste trabalho). O *stream* deste arquivo compactado é lido novamente e salvo na tabela do banco de dados, em um campo do tipo *bytea* (formato cujo PostgreSQL utiliza para armazenamento de longas sequências binárias), associado ao organismo ao qual o usuário optou por vincular o arquivo.

3.2 CONVERSÃO DO ARQUIVO GBK PARA FASTA

O procedimento que é amplamente conhecido como p*arse* na área da informática trata de uma conversão entre uma informação de entrada, ou *input* (geralmente uma estrutura, um conjunto de dados ou um arquivo, entre outros), e um formato esperado como saída, ou *output* (também podendo ser uma estrutura, um conjunto de dados ou um arquivo, entre outros), e consiste em criar regras de transformação para o conteúdo original que permitam que informações sejam extraídas e utilizadas conforme necessidade. É importante ressaltar que nem sempre haverá um *output* definitivo (conversão), pois em algumas situações, o *parse* será utilizado apenas para ler a informação e tomar uma decisão, como por exemplo uma validação.

Neste trabalho foi feito o *parse* do arquivo GBK para o formato Fasta, de forma que há um arquivo resultante da conversão e este arquivo fica disponível para utilização posterior. A conversão utilizada atualmente não compreende todas as informações contidas no arquivo GBK para geração de Fasta e Multifasta, a única exceção é para arquivos GBK com mais de um organismo contido, neste caso, o sistema irá processar a informação e gerar um fasta com tantos cabeçalhos conforme for a quantidade de organismos existentes no GBK, adotando para cada sequência de nucleotídeos, o nome do organismo como cabeçalho.

O trecho de código que representa a explicação acima consta na FIGURA 03.

```
83          # Percorre linha por linha do arquivo, tratando o que for
84          # necessario para cada formato
85          stream.each_line do |linha|
86
87              # contador
88              cont = cont + 1
89
90              # Busca pelo organismo no arquivo
91              if org_m = linha.match(/^[ ]*ORGANISM[ ]*(.*)$/)
92                  org = org_m.captures
93                  org = org.join.to_str
94
95                  # Monta a linha de cabeçalho do organismo do FASTA
96                  @str_fasta = @str_fasta + ">#{org}\n"
97
98
99                  # Senao, verifica onde começa a sequencia genética
100                 elsif org_m = linha.match(/^[ ]*ORIGIN[ ]*$/)
101
102                 # Marca um flag para início da sequencia
103                 ini_s = true
104
105
106                 # Depois do início da sequencia, verifica linha por linha
107                     # até que acabe, neste caso, marca o flag como falso e não
108                 # utiliza a linha
109                 elsif ini_s
110
111                 # Verifica se a linha toda é válida (contém apenas aminoácidos de DNA)
112                 # (Atualmente não processa RNA / Uracila, mas suporta mesmo assim)
113                 if org_m = linha.match(/^[ ]*[0-9]+([ actgu]*)$/i)
114
115                     dna = org_m.captures
116                     dna = dna.join.to_str
117                     dna = dna.delete(' ')
118                     dna = dna.upcase
119                     @str_fasta = @str_fasta + dna + "\n"
120
121                     # Se encontrar uma linha inválida, encerra a sequencia até que apareça
122                     # uma nova sequencia / organismo
123                     else
124                     ini_s = false
125
126                 end
127
128             end
129
130         end
```

FIGURA 03 – TRECHO DE CÓDIGO RUBY, CONVERSÃO PARA FASTA

Fonte: Os autores (2016).

De uma forma sucinta, o arquivo GBK (que neste momento encontra-se carregado e em memória através da variável “*stream*”) é lido linha a linha. Na linha 91 ocorre uma pesquisa, utilizando expressão regular, para identificar se aquela linha coincide com o padrão “ORGANISM”. Ao mesmo tempo, o resultado é armazenado no objeto “*org_m*”. Se a linha atual da iteração coincidir com este padrão, o valor subsequente à palavra chave “ORGANISM” será utilizado. Este será

o nome do organismo e irá compor o cabeçalho do arquivo fasta (onde o cabeçalho é identificado por ter o primeiro caractere da linha igual a “>”), isto ocorre entre as linhas 92 e 96. As iterações ao longo do arquivo continuam ocorrendo até que uma segunda expressão regular seja satisfeita, linha 100. Esta expressão valida o conteúdo da linha e preenche o objeto “*org_m*” quando aquele for igual a “ORIGIN”. Quando isto acontecer é possível determinar que as próximas linhas irão conter uma sequência genética. No arquivo GBK a sequência genética é constituída por um numerador, no início de cada linha após o atributo “ORIGIN”, e seis grupos de aminoácidos, representados por letras (como ACTG), em blocos de 10 caracteres. O que ocorre entre as linhas 113 e 126 é a identificação de uma linha de aminoácidos válidos e tratamento da informação para que resulte em uma *string* contendo 60 caracteres, apenas os aminoácidos. Esta *string* com apenas aminoácidos será inserida no arquivo Fasta, ainda em memória, para gravação e compactação posterior (conforme detalhado no item 3.1 deste trabalho).

4. APRESENTAÇÃO DO SOFTWARE

Neste capítulo será apresentada uma visão geral do software desenvolvido, mostrando suas telas e funcionalidades.

No momento as atividades possíveis a um pesquisador não variam muito comparado a de um usuário público, sendo a única diferença no momento a possibilidade de ver mais projetos devido a sua restrição. Por isso não há uma seção específica para esse tipo usuário.

4.1 PERFIL PÚBLICO DE ACESSO

O acesso de todos os usuários iniciasse pela tela da FIGURA 04, no qual é possível fazer *login* ou solicitar acesso de coordenador para o administrador.

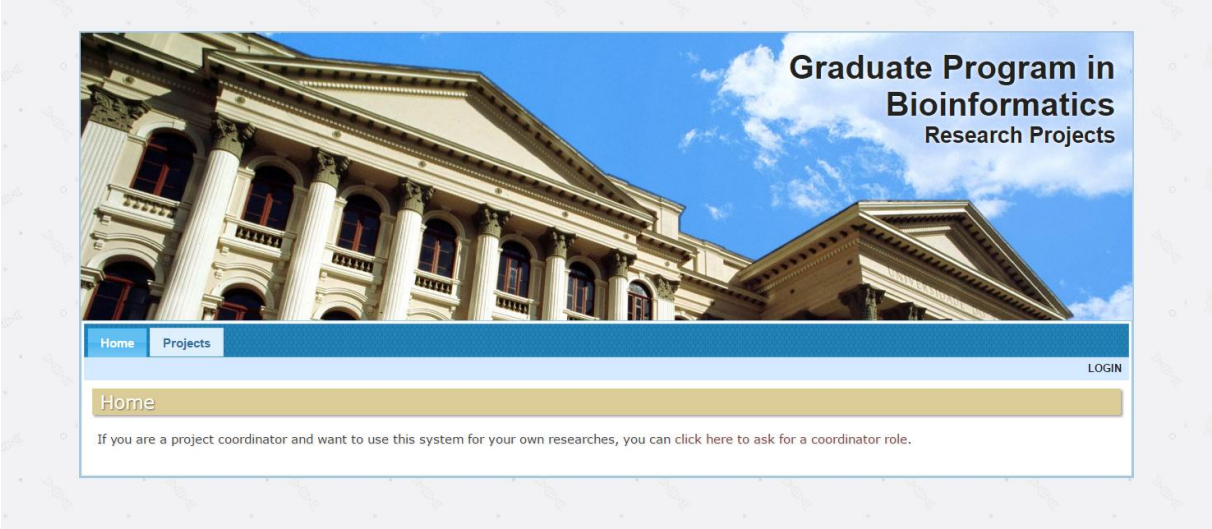

FIGURA 04 - TELA INICIAL PÚBLICO

Fonte: Os autores (2016).

Ao clicar em "*Projects*", temos a tela da FIGURA 05. Apesar dessa tela estar disponível para todos os usuários, os projetos podem possuir restrição de visualização de acordo com o tipo do usuário.

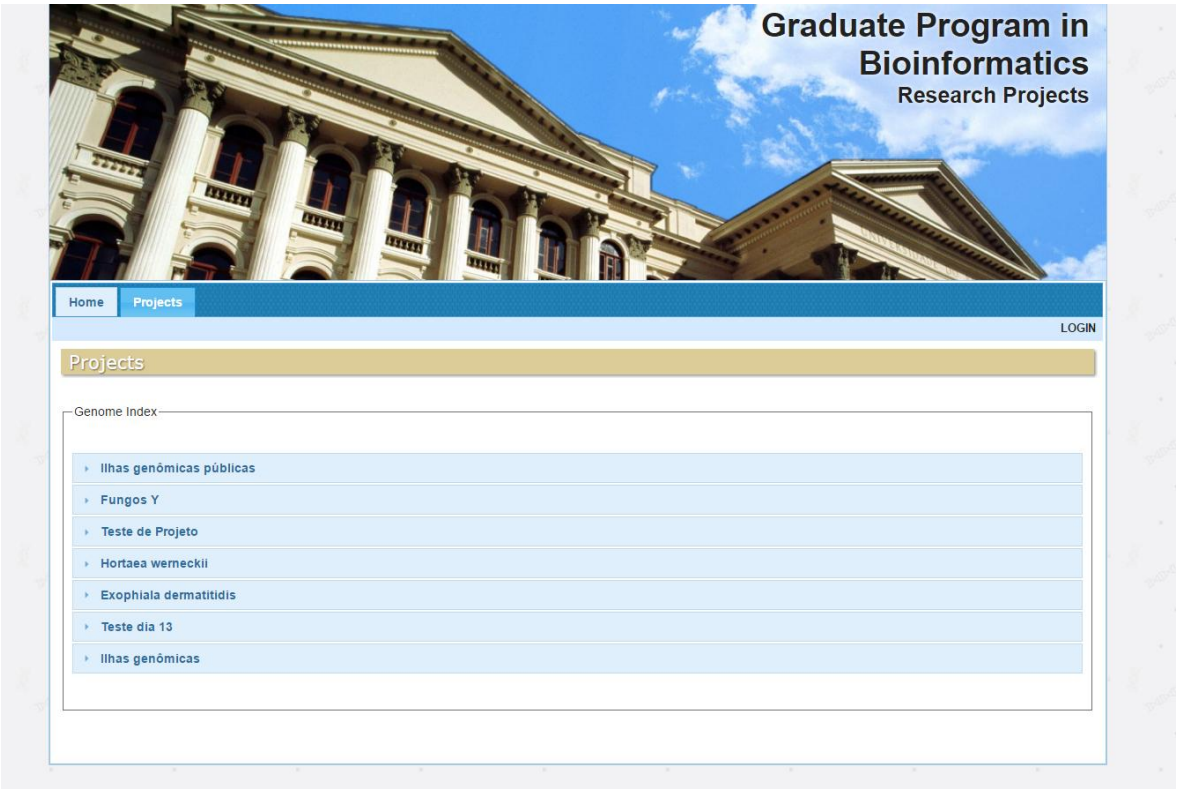

FIGURA 05 – TELA PROJETOS

Fonte: Os autores (2016).

É possível baixar os arquivos genéticos de acordo com as restrições de permissão.

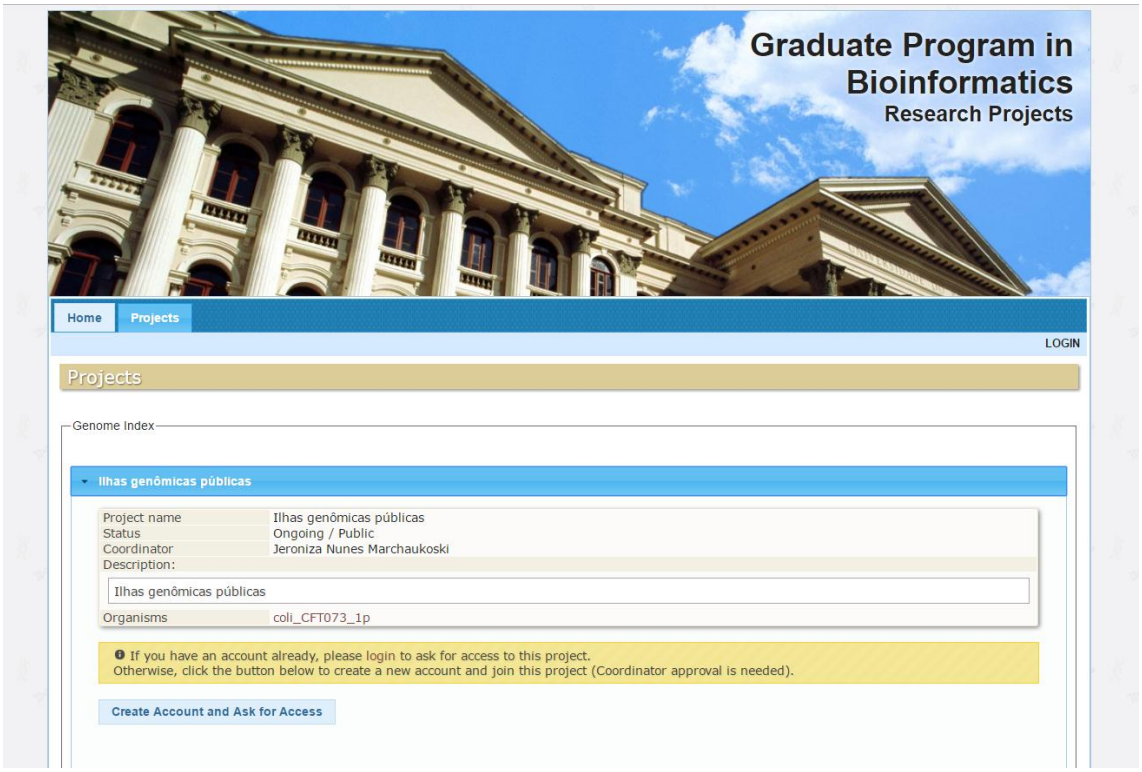

FIGURA 06 - TELA ARQUIVOS

Fonte: Os autores (2016).

E também é possível pedir acesso a um projeto. Caso o usuário esteja autenticado, é pedido um acesso de convidado (pois ele já é um pesquisador), caso o usuário não esteja autenticado é solicitado um acesso de pesquisador e prossegue para a tela da FIGURA 08.

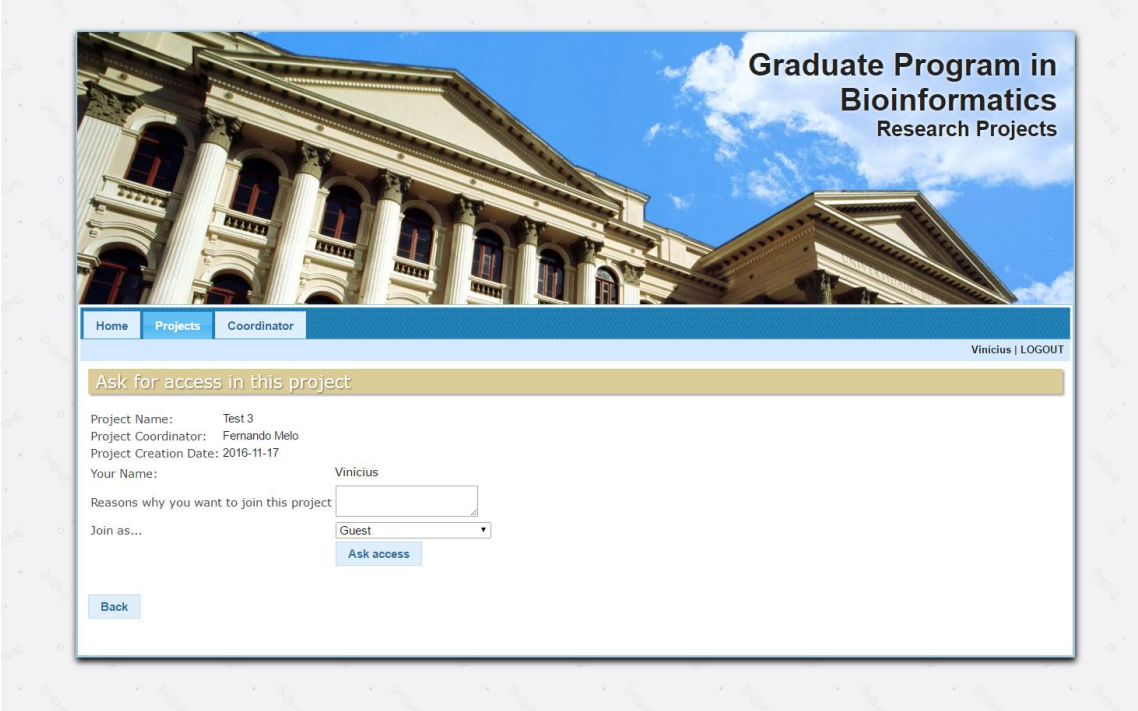

FIGURA 07 - TELA SOLICITAR ACESSO DE CONVIDADO

Fonte: Os autores (2016).

É preenchido os dados do solicitante e mostrado quais são os dados do projeto solicitado.

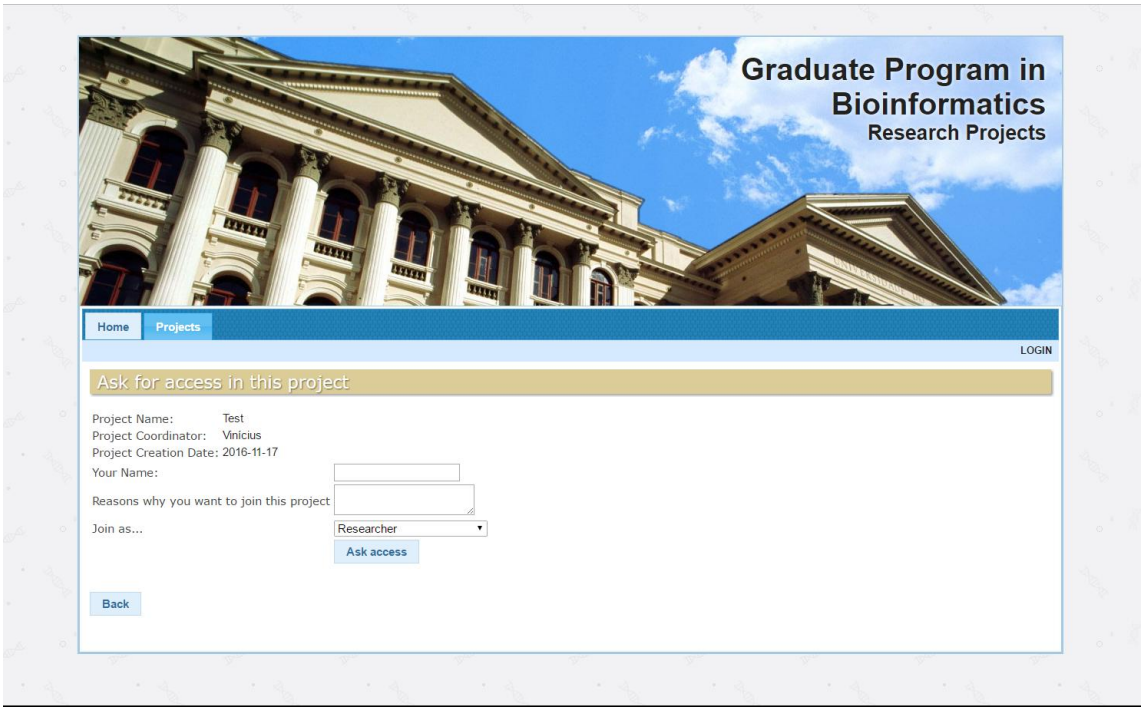

FIGURA 08 - TELA CADASTRO PESQUISADOR

Fonte: Os autores (2016).

Voltando a tela inicial, o visitante pode solicitar acesso como coordenador ao clicar em "*click here to ask for a coordinator role*". Temos na tela da FIGURA 09 o cadastro para solicitação do acesso de coordenador para o administrador.

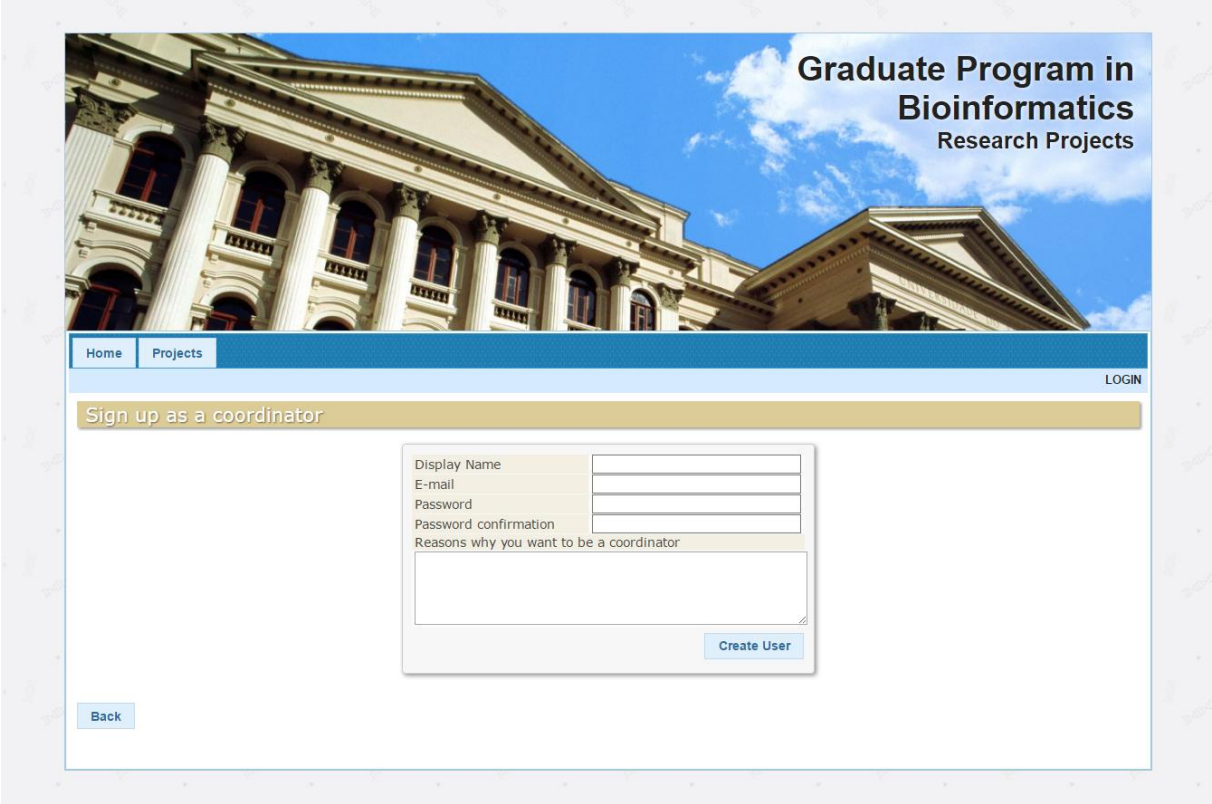

FIGURA 09 - TELA SOLICITAR ACESSO DE COORDENADOR

Fonte: Os autores (2016).

4.2 ACESSO DE ADMINISTRADOR

Ao entrar com o perfil de administrador, é apresentada a tela da FIGURA 10, mostrando as possibilidades de administração.

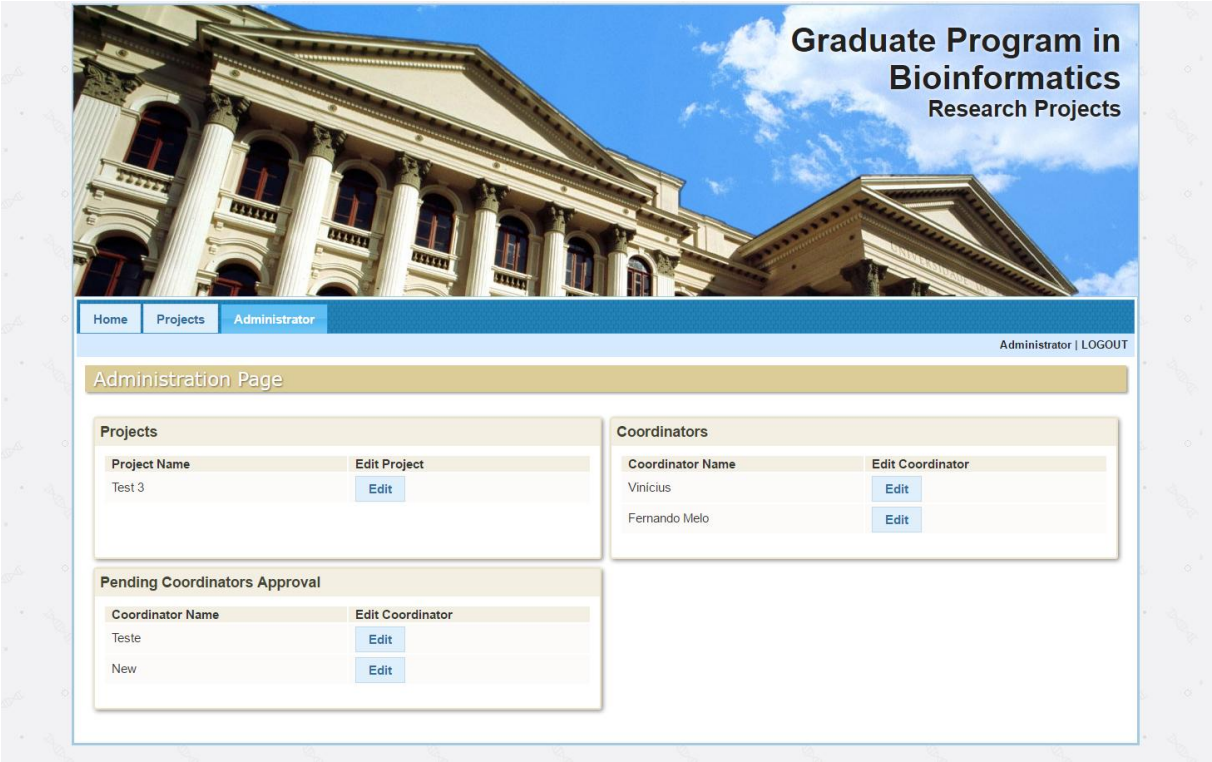

FIGURA 10 - TELA INICIAL ADMINISTRADOR

Fonte: Os autores (2016).

Caso escolha por editar um projeto, o administrador pode mudar o coordenador de um projeto utilizando o campo apropriado (Caixa de seleção: *“Project Leader”*).

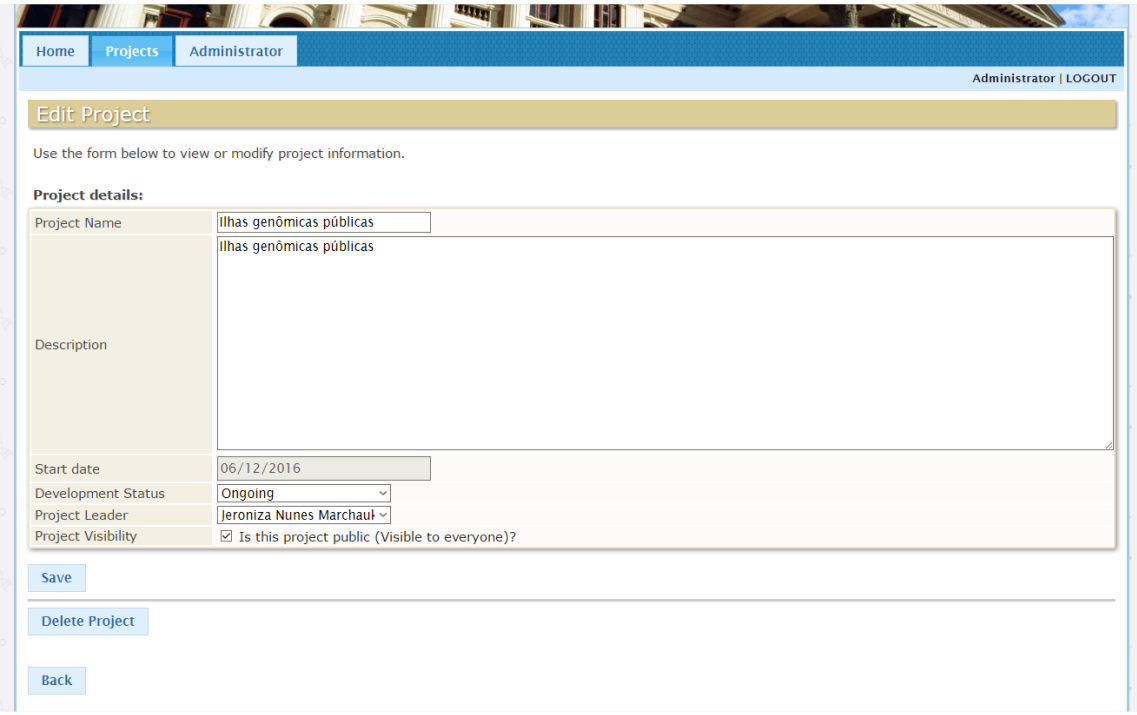

FIGURA 11 - TELA EDITAR PROJETOS

Fonte: Os autores (2016).

Na tela da FIGURA 12 o administrador pode aceitar novos coordenadores e novos projetos.

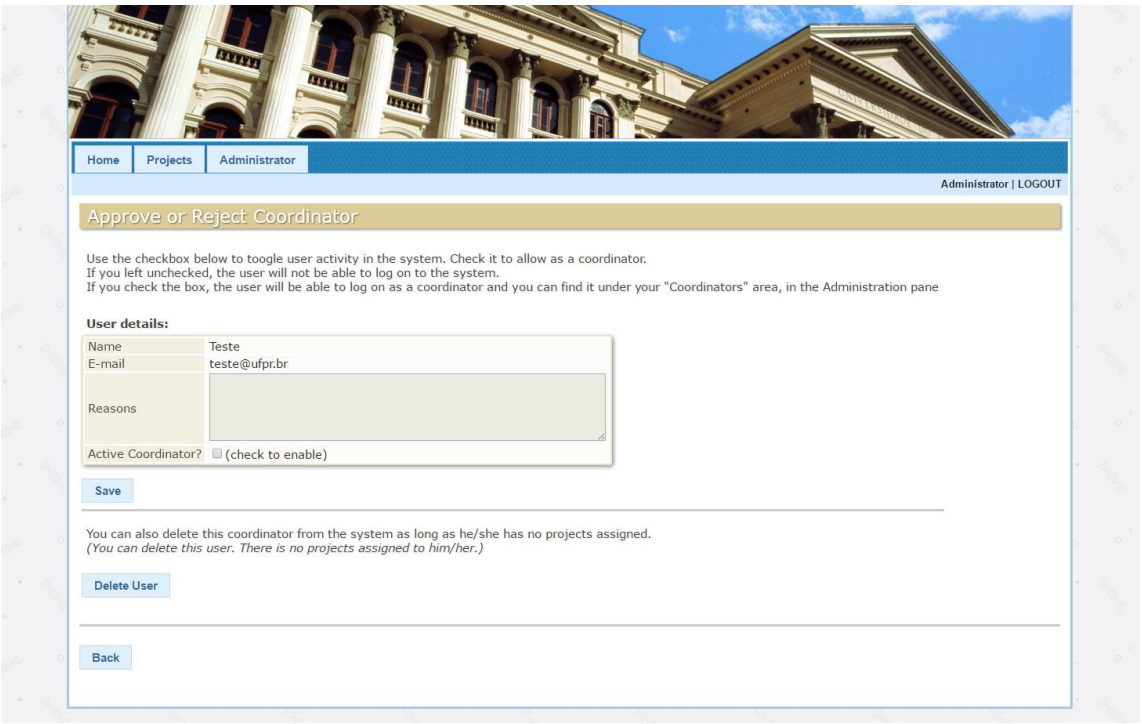

FIGURA 12 - TELA ACEITAR PROJETOS E NOVOS COORDENADORES

Fonte: Os autores (2016).

Caso escolha por editar ou excluir um coordenador, é possível ao administrador efetuar essa ação na tela da FIGURA 13.

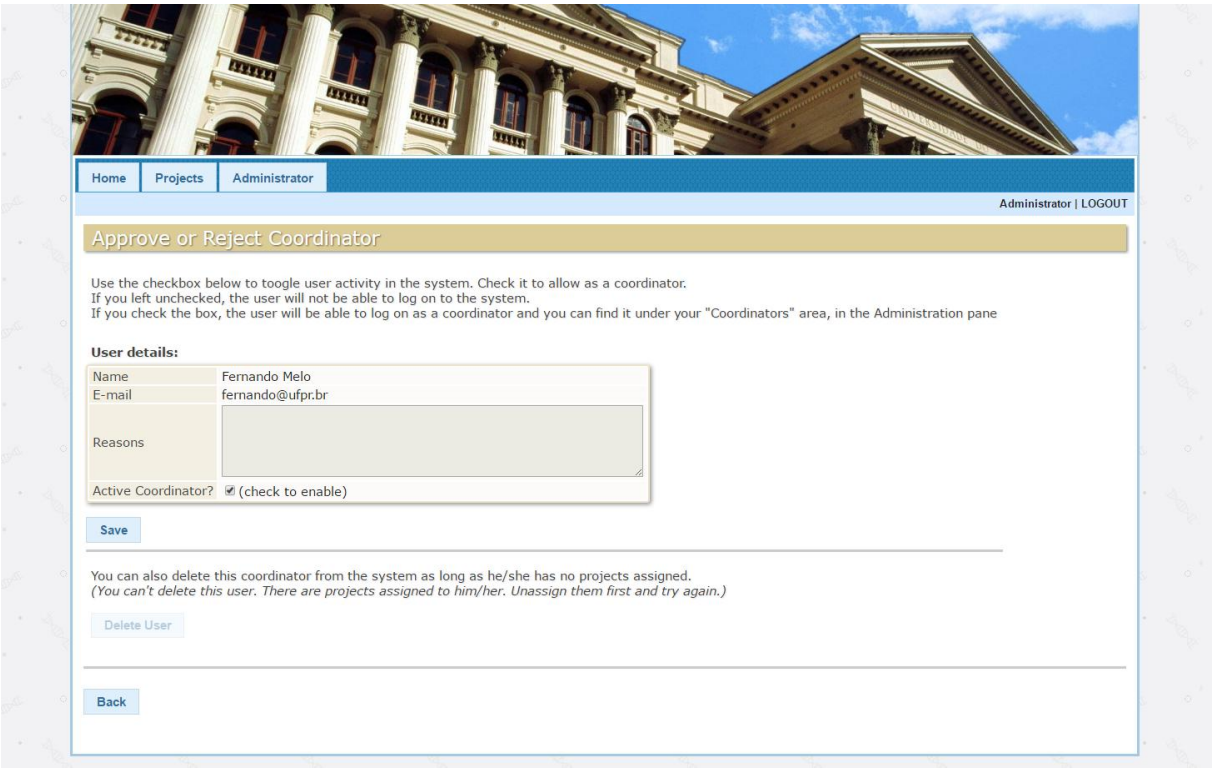

FIGURA 13 - TELA EDITAR/ EXCLUIR COORDENADORES

Fonte: Os autores (2016).

4.3 ACESSO DE COORDENADOR

Depois de autenticado, é apresentado ao coordenador a tela da FIGURA 14, mostrando as possibilidades de administração.

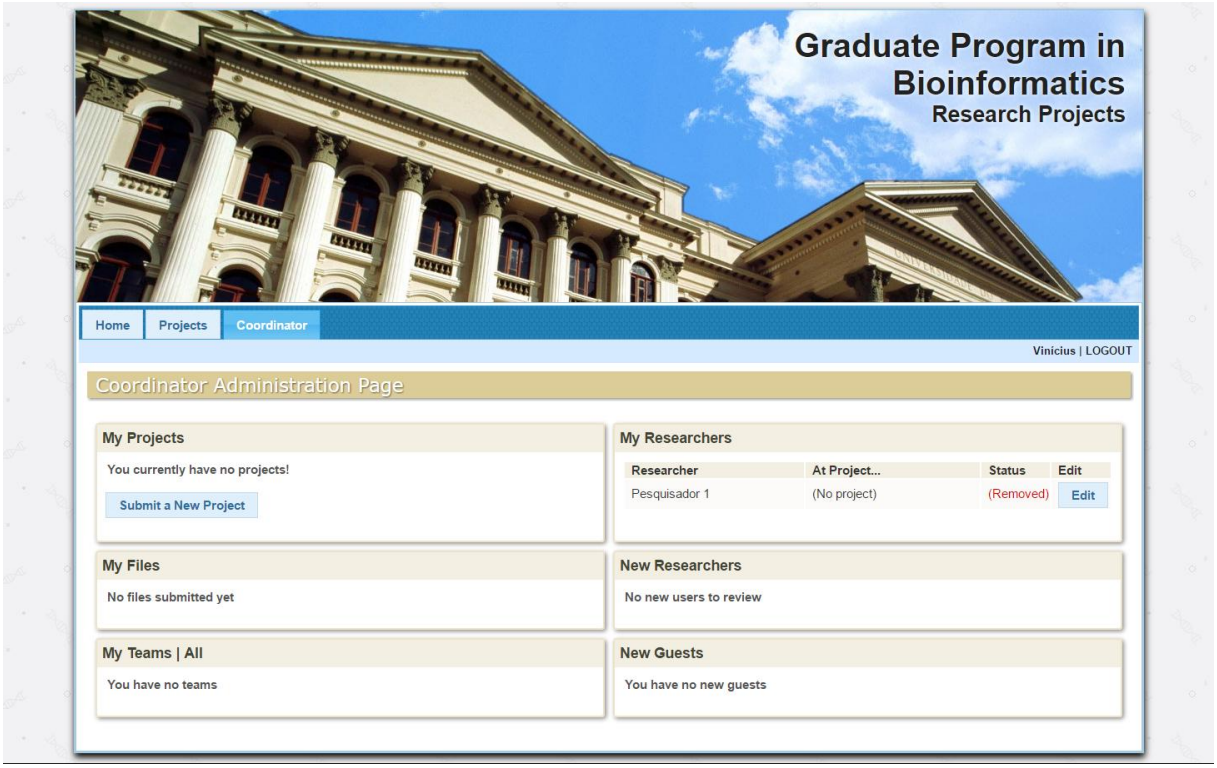

FIGURA 14 - TELA INICIAL COORDENADOR

Fonte: Os autores (2016).

Ao clicar em editar um projeto é possível ao coordenador editar as informações de um projeto ou exclui-lo.

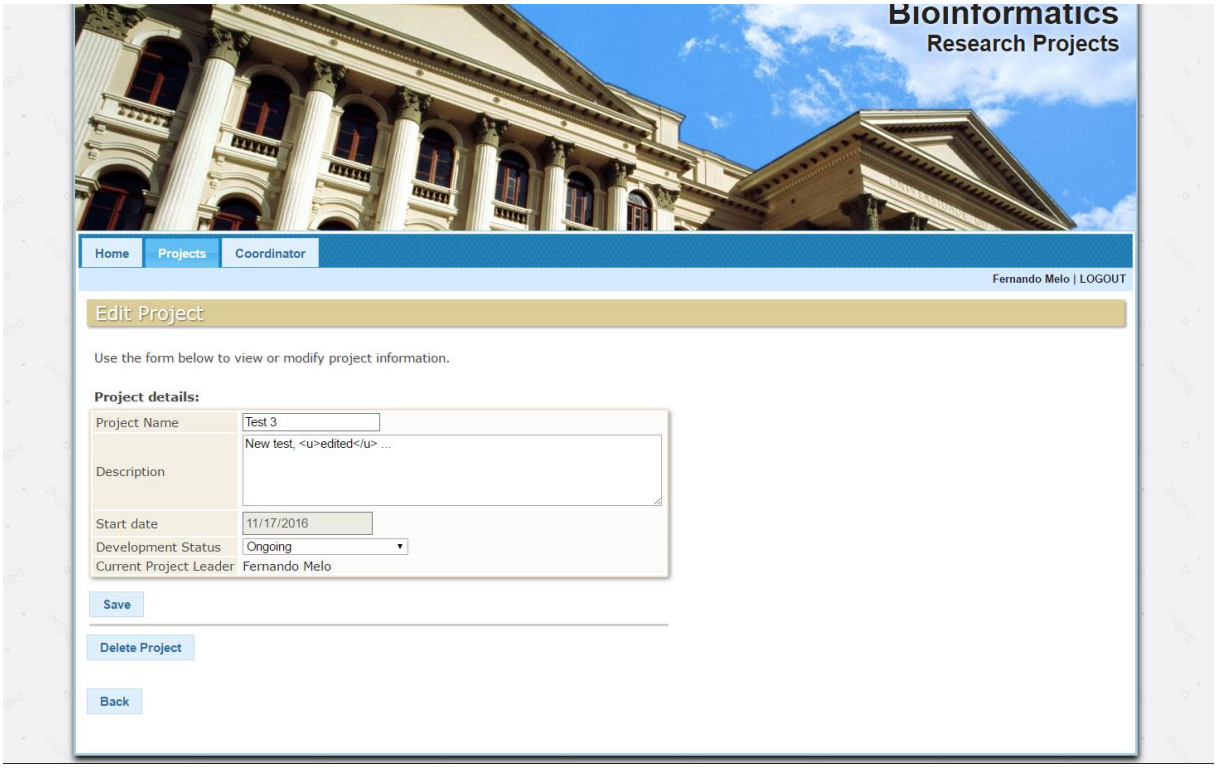

FIGURA 15 - TELA EDITAR/EXCLUIR PROJETOS

Fonte: Os autores (2016).

Ao clicar em editar um organismo é possível ao coordenador editar as informações de um organismo ou exclui-lo.

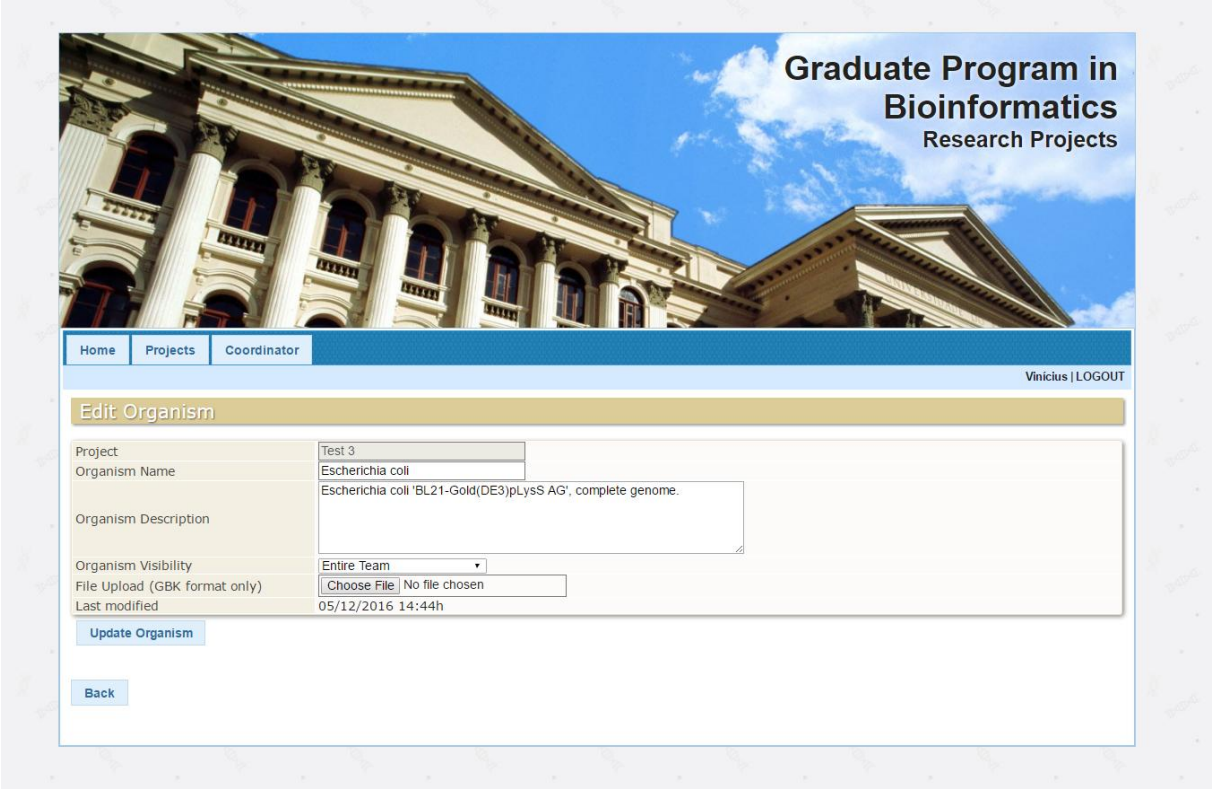

FIGURA 16 - TELA EDITAR/EXCLUIR ORGANISMOS

Fonte: Os autores (2016).

Enquanto edita um organismo o coordenador pode incluir um novo arquivo para aquele organismo, como mostrado na tela da FIGURA 17.

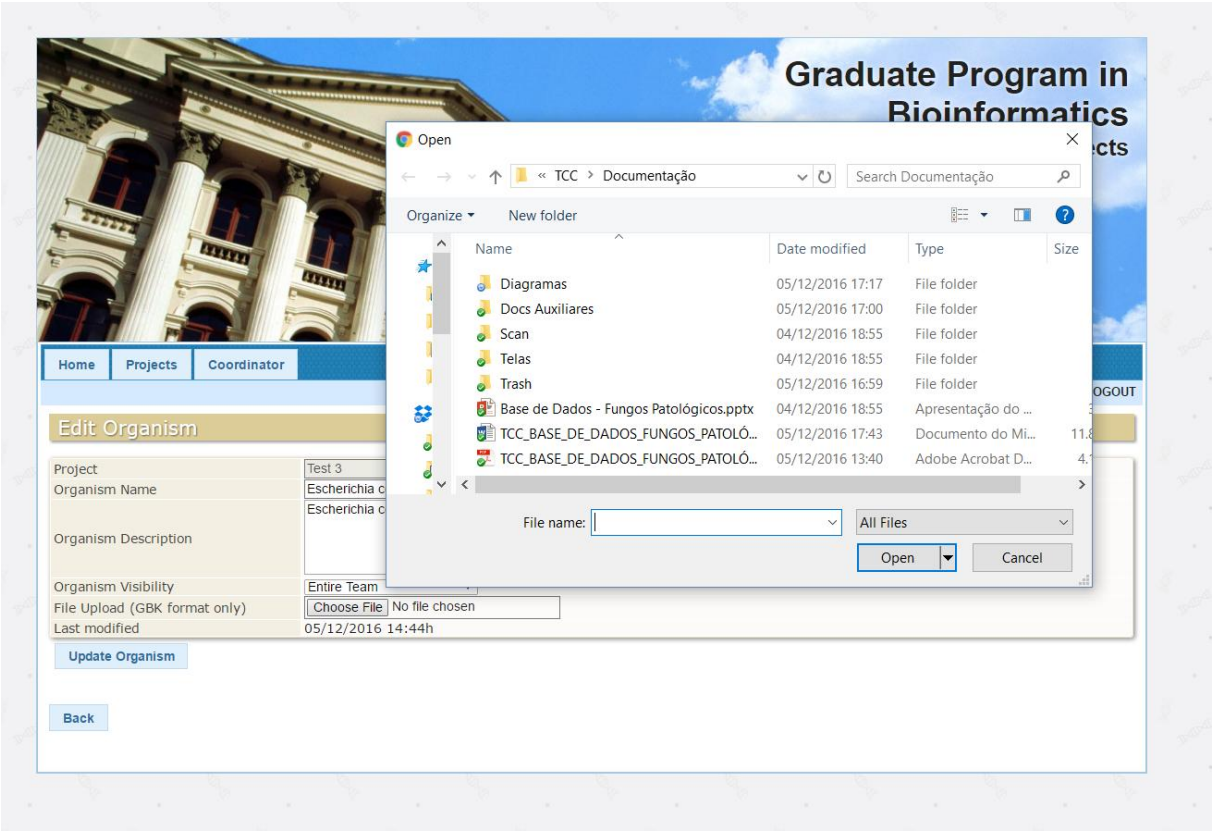

FIGURA 17 - TELA INCLUIR ARQUIVOS DE PROJETO

Fonte: Os autores (2016).

Após submeter um novo projeto (FIGURA 18) o coordenador monta a equipe para aquele projeto, como mostrado na tela da FIGURA 19.

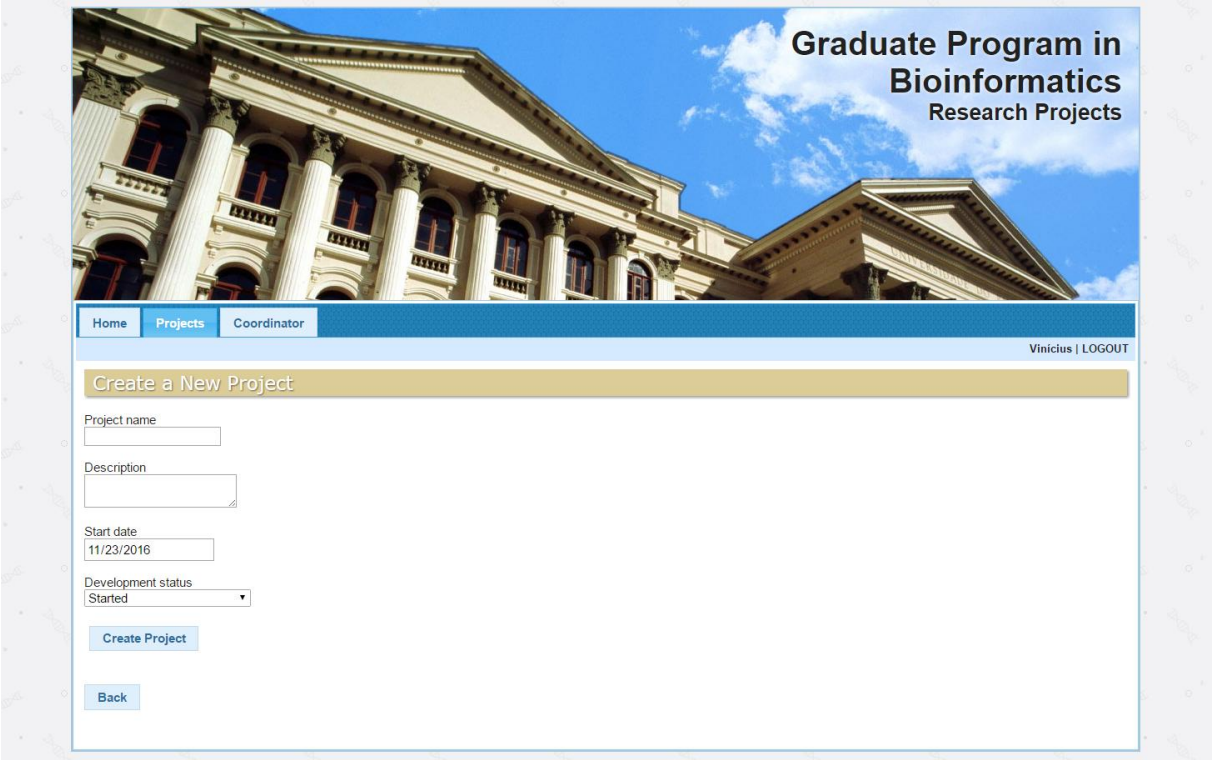

FIGURA 18 - TELA SUBMETER UM NOVO PROJETO

Fonte: Os autores (2016).

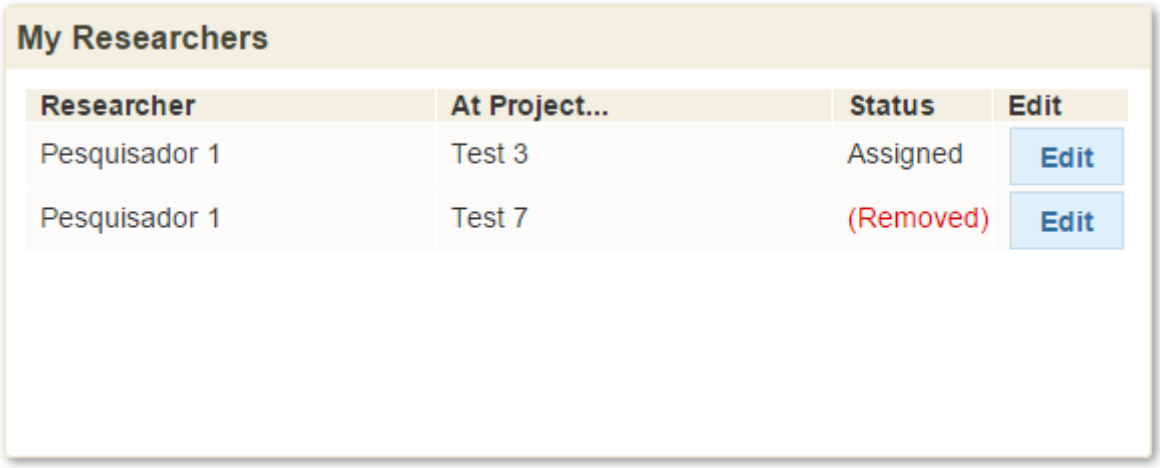

FIGURA 19 - TELA CADASTRA EQUIPE

Fonte: Os autores (2016).

Ao receber uma solicitação de acesso de pesquisador/convidado a um de seus projetos, o coordenador pode decidir se aceita ou não.

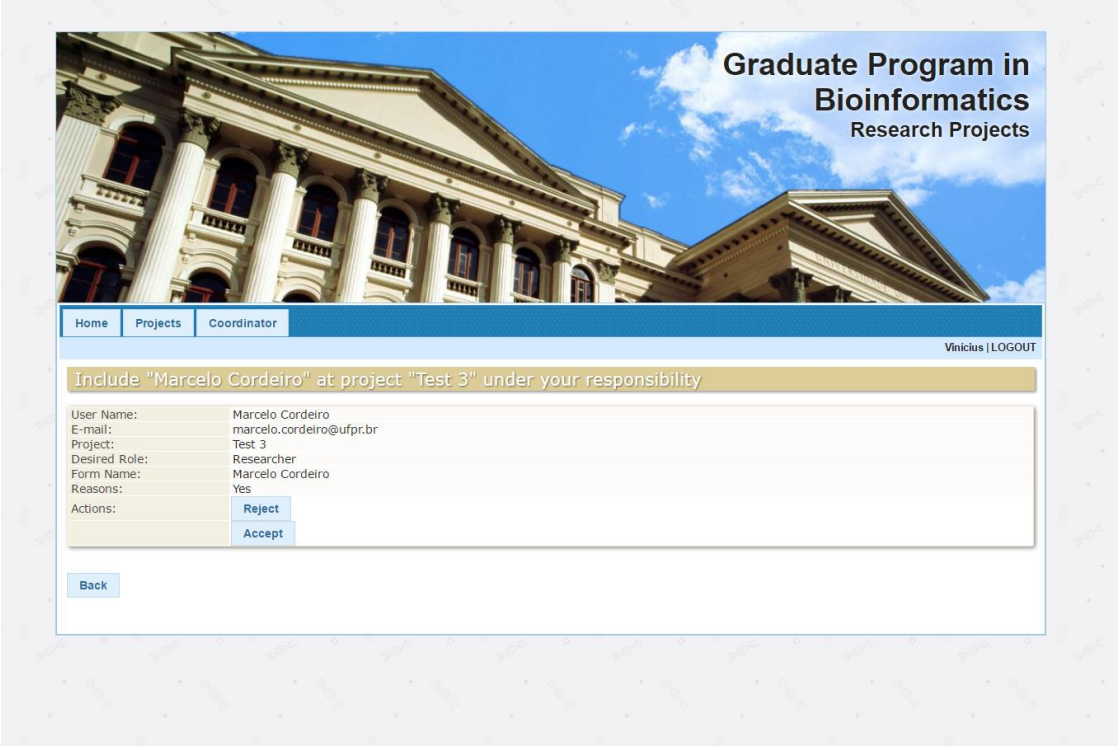

FIGURA 20 - TELA ACEITAR PESQUISADORES/CONVIDADOS

Fonte: Os autores (2016).

O coordenador também pode excluir seus pesquisadores, ao clicar no botão “Delete User” e confirmar a ação, como visto na tela da FIGURA 21.

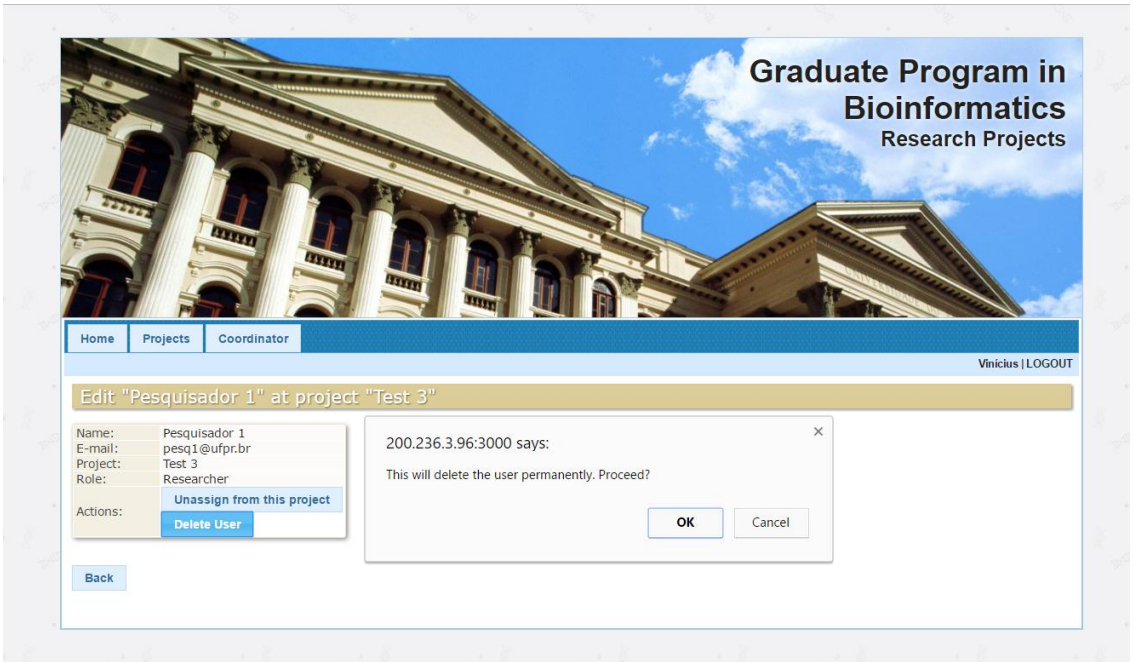

FIGURA 21 – TELA EXCLUIR PESQUISADORES

Fonte: Os autores (2016).

O coordenador também consegue editar as equipes de seus projetos

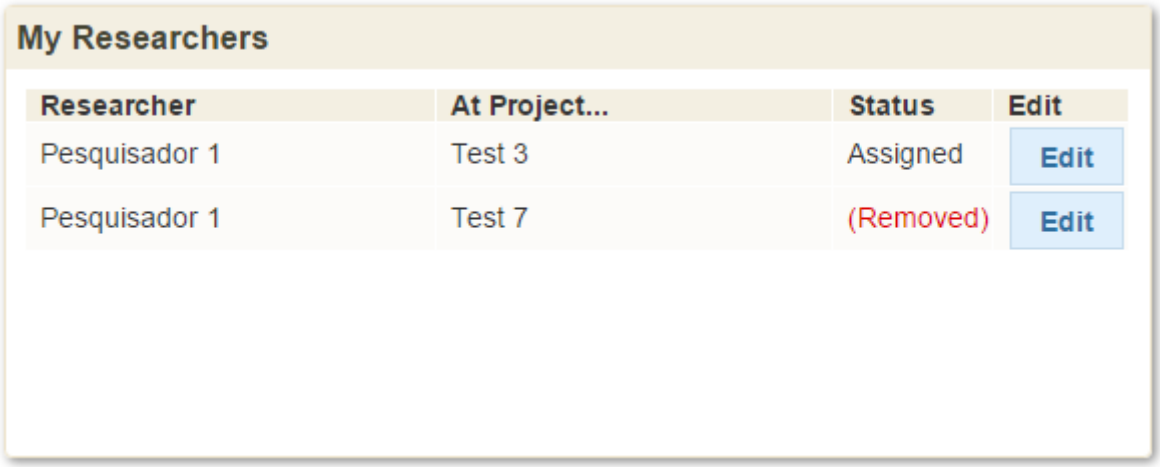

FIGURA 22 - TELA EDITAR EQUIPES

Fonte: Os autores (2016).

5. CONSIDERAÇÕES FINAIS

O desenvolvimento do sistema nos possibilitou ter um contato inicial com a área da bioinformática e perceber a importância desta, tanto para o desenvolvimento das mais diversas áreas da biologia como para a criação de novas tecnologias.

O aplicativo consegue cumprir as funções de armazenamento de informações genéticas dos organismos pesquisados assim como auxiliar no gerenciamento de times e projetos.

A conversão dos arquivos GBK para o formato Fasta é muito bem-vinda para os pesquisadores que precisam utilizar diferentes formatos de arquivos, principalmente para quem precisa somente utilizar a sequência genética de um organismo, e pode optar por fazer o download no formato Fasta. Porém, a impossibilidade de conversão entre os tipos de arquivo pode vir a parecer que a funcionalidade não está completa.

As funcionalidades de download e upload de arquivos genéticos consomem um alto valor computacional e de rede, por isso o sistema pode vir a enfrentar problemas de lentidão em caso de muitos acessos simultâneos a essas funções, mesmo com a implementação da compactação.

Durante o desenvolvimento, várias ideias de como o software poderia ser melhorado e ampliado surgiram, porém não foi possível implementá-las devido a complexidade e tempo. Algumas destas ideias para uma futura versão estão listadas abaixo:

- Criar uma ferramenta para busca por genomas ou descrição de genes
- Utilizar um banco de dados non-SQL afim de diminuir o impacto computacional para Download e Upload
- Implementar a conversão de GBK para GFF
- Implementar a conversão de GFF para GBK

REFERÊNCIAS

ANICAS, Mitchell. **How To Use PostgreSQL with Your Ruby on Rails Application on Ubuntu 14.04**. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-use-postgresql-with-your-ruby-on-rails-application-on-ubuntu-14-04>. Acesso em: 04 dez. 2016.

ASTAH. **Astah – Fundamentals**. Disponível em: <http://astah.net/fundamentals>. Último acesso em: 27/06/2016.

BALSAMIQ. **Balsamiq Mockups**. Disponível em: <https://balsamiq.com/products/mockups/>. Último acesso em: 26/06/2016.

BIORUBY. **BioRuby: bioinformatics software for the Ruby programming language**. Disponível em: <http://bioruby.org/>. Último acesso em: 20/06/2016.

BROAD INSTITUTE. **Black Yeast Database.** Disponível em: <http://www.broadinstitute.org/annotation/genome/Black_Yeasts/MultiHome.html>. Último acesso em: 26/06/2016.

GIACOMAZZI, J. et al. **The burden of serious human fungal infections in Brazil**. Mycoses. v. 59, n. 3, p. 145-50. 2016.

GIGLIO, Giuliano Prado M. **Página da Disciplina Análise de Sistemas I - UNIVERSO**. Último acesso em: <http://www.giulianoprado.xpg.com.br/analise1/material.htm>. Acesso em: 03 dez. 2016.

GIT. **About – Git.** Disponível em: <http://git-scm.com/about/>. Último acesso: 20/05/2016.

GUDWIN, Ricardo R. **Diagramas de Atividade e Diagramas de Estado**. Disponível em: <http://www.dca.fee.unicamp.br/~gudwin/ftp/ea976/AtEst.pdf>. Acesso em: 04 dez. 2016.

GUEDES, G. **UML 2: uma abordagem prática**. São Paulo: Novatec, 2011.

HOW include jquery-ui in your rails app. Disponível em: <https://agilewarrior.wordpress.com/2014/04/30/how-include-jquery-ui-in-your-rails-app/>. Acesso em: 04 dez. 2016.

IBM. **IBM – Definição de RUP.** Disponível em: <http://www-01.ibm.com/software/rational/rup/>. Último acesso: 21/05/2016.

MYSQL. **MySQL :: MySQL Workbench**. Disponível em: <https://www.mysql.com/products/workbench/>. Último acesso em: 25/06/2016.

NCBI. **National Center for Biotechnology Information**. Disponível em: <https://www.ncbi.nlm.nih.gov/>. Último acesso em: 04/12/2016.

ORACLE. **Oracle VM VirtualBox**. Disponível em: <https://www.virtualbox.org/>. Último acesso em: 24/06/2016.

PPG-BIOINFO. **Programa de Pós-Graduação em Bioinformática**. Disponível em: <http://www.bioinfo.ufpr.br/about.php>. Último acesso em: 26/06/2016.

POSTGRESQL. **PostgreSQL: About**. Disponível em: <https://www.postgresql.org/about/>. Último acesso em: 23/06/2016.

RASHIDI, H.; BUEHLER, L. **Bioinformatic Basics – Applications in Biological Science and Medicine.** Londres: CRC Press, 2000.

RUBY. **About Ruby**. Disponívelem: <https://www.ruby-lang.org/en/about/>. Último acesso em: 20/06/2016.

RUBYGEMS. **RubyGems.org | o host de gems da sua comunidade**. Disponível em: <https://rubygems.org/>. Acesso em: 02 nov. 2016.

RUBY ON RAILS. **Ruby on Rails**. Disponível em: <http://rubyonrails.org/>. Último acesso em: 20/06/2016.

DRAW.IO. **Flowchart Maker & Online Diagram Software.** Disponível em: http://draw.io/. Último acesso em: 17 dez. 2016.

APÊNDICE A - DIAGRAMA DE CASOS DE USO

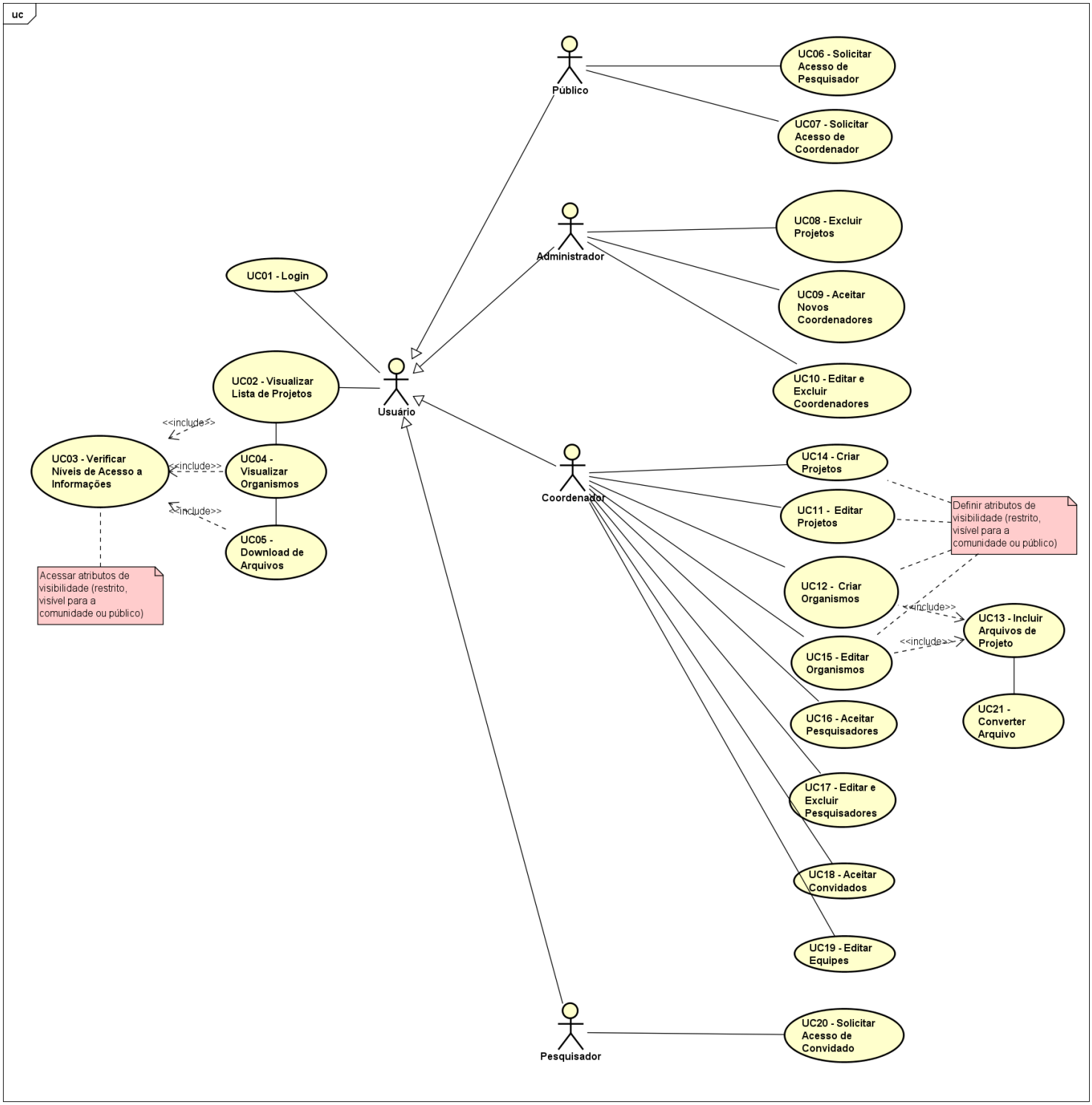

FIGURA 23 – DIAGRAMA DE CASOS DE USO

Fonte: Os autores (2016).

APÊNDICE B - DIAGRAMA ENTIDADE RELACIONAMENTO

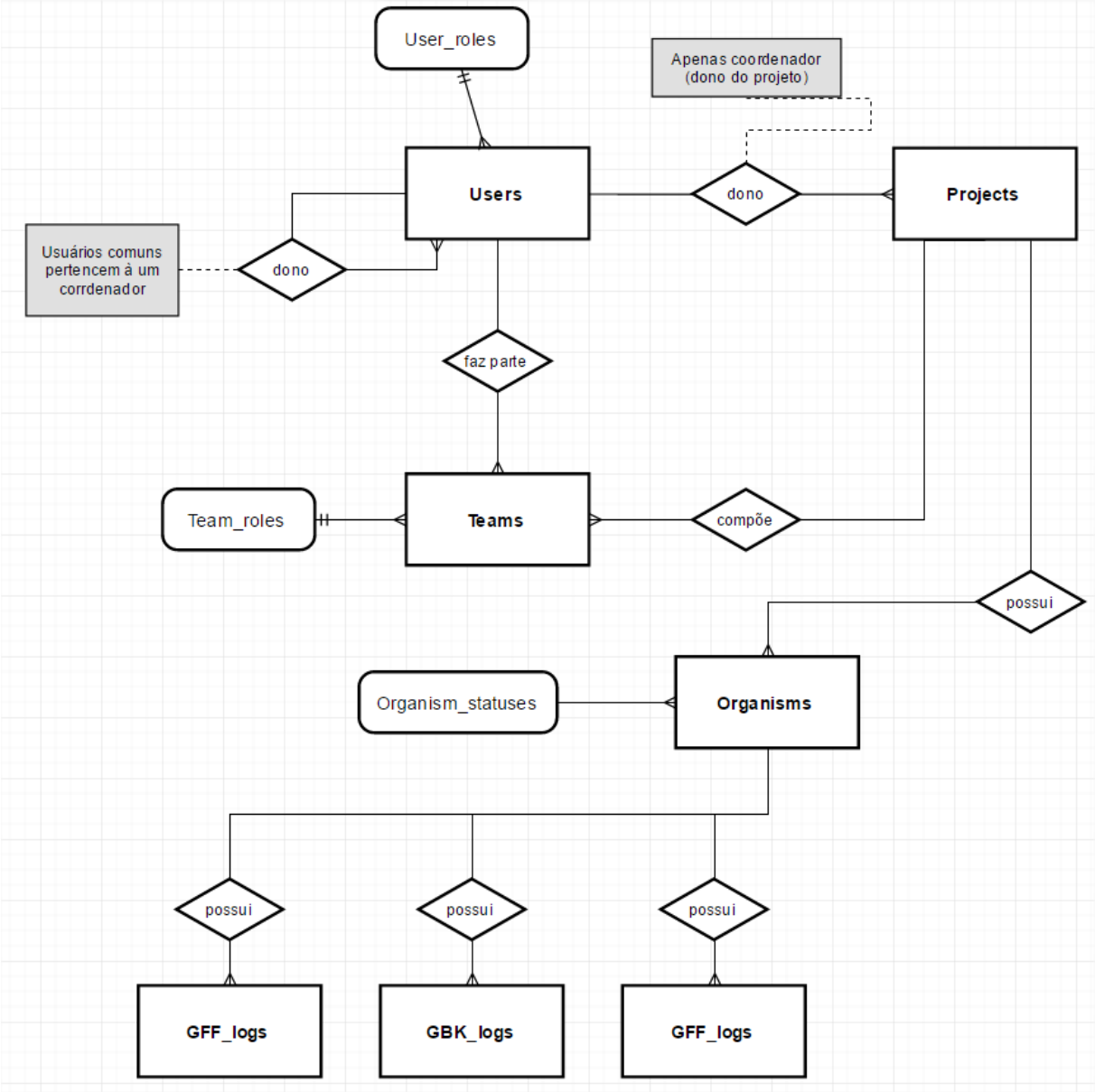

FIGURA 24 – DIAGRAMA ENTIDADE RELACIONAMENTO

Fonte: Os autores (2016).

APÊNDICE C - DICIONÁRIO DE DADOS

TABELA 1 – DICIONÁRIO DE DADOS

TABELA	teams					
Faz a ligação entre pesquisadores de uma equipe e um projeto						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Time	Id	INT	PK			
ID Usuário	user_id	INT		(users/ id)		Relacionamento com a tabela "users" através do campo "id"
ID Projeto	project_id	INT		(projects/id)		Relacionamento com a tabela "projects" através do campo "id"
ID Papel	team_role_id	INT		(team_roles/ id)		Relacionamento com a tabela "team_role" através do campo "id"
Nome	nome	VARCHAR(40)				
Descrição	descricao	VARCHAR (255)				
Status Acesso	status_ace	CHAR				Ativo ou Inativo no Projeto

TABELA	team_roles					
Define o papel de cada usuário na equipe						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Papel	id	INT	PK			
Nível	nivel	VARCHAR (20)				"Pesquisador" ou "Convidado"
Descrição	descricao	VARCHAR (255)				

TABELA	Projects					
Tabela com os dados de cada projeto						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Projeto	id	INT	PK			
ID Coordenador	user_id	INT		(users/ id)	(Usuário/ papel) = "Coordenador"	Relacionamento com a tabela "users", através do campo "user_role_id", porém somente se o papel do usuário for "Coordenador"
Nome	nome	VARCHAR (255)				
Descrição	descricao	VARCHAR (255)				
Data de Início	dt_ini	DATE				
Status de Aceitação	status_ace	CHAR				Ativo - Público Inativo - Restrito
Status de Andamento	andamento	CHAR				I – Started E – On Going C – Completed

TABELA	Users					
Tabela com a descrição dos usuários. Caso o Usuário seja "Pesquisador", somente o Coordenador responsável pode exclui-lo. Caso o Usuário seja "Coordenador" somente um Administrador pode exclui-lo.						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Usuário	id	INT	PK			
ID Coordenador	u_id	INT		(users / id)	(Usuário/ papel) <> "Coordenador" ou "Administrador"	Relacionamento com a tabela "user_roles", através do campo "id". Caso seja um "Coordenador" ou "Administrador", ID Coordenador será igual a *null*.
Papel	user_role_id	INT		(user_roles/ id)		Relacionamento com a tabela "user_roles", através do campo "id"
Nome de Usuário	nomeUsuario	VARCHAR (20)				
Nome de Exibição	nomeExibicao	VARCHAR (40)				
Email	email	VARCHAR (50)				*Login
Senha	senha	VARCHAR (40)				
Status de Aceitação	status_ace					Ativo ou Inativo

Motivo	motivo	VARCHAR				

TABELA	user_roles					
Tabela com a descrição de cada função do usuário						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Papel Usuário	id	INT	PK			
Nível	nivel	VARCHAR (45)				Administrador, Coordenador ou Pesquisador
Descrição	descricao	VARCHAR (45)				

TABELA	Organisms					
Tabela com a descrição de cada organismo						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Organismo	id	INT	PK			
ID Projeto	project_id	INT		(projects /id)		Relacionamento com a tabela “projects”, através do campo “id”
ID Status	organism_status_id	INT		(organism_statuses / id)		Relacionamento com a tabela “organism_statuses”, através do campo “id”
Nome	nome	VARCHAR (50)				
Descrição	descricao	VARCHAR (255)				
Status de Aceitação	status_ace					Ativou ou Inativo
Stream Fasta	stream_fasta	BLOB				
Stream GBK	stream_gbk	BLOB				
Stream gff	stream_gff	BLOB				

TABELA	organism_statuses					
Tabela com o status (visibilidade) de cada organismo						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Status	id	INT	PK			
Descrição	descricao	VARCHAR (255)				
Visibilidade	Visibilidade	CHAR(1)				P – Público C – Somente Coordenador E – Visível para Equipe

TABELA	GFF_logs					
Tabela de log para os arquivos GFF de cada organismo, mas não utilizada						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID GFF	id	INT	PK			
ID Organismo	organism_id			(organisms / id)		Relacionamento com a tabela “organisms”, através do campo “id”
Descrição	descricao	VARCHAR (255)				
Stream GFF	stream	BLOB				
Data	data	DATE				

TABELA	GBK_logs					
Tabela de log para os arquivos GBK de cada organismo						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID GBK	id	INT	PK			
ID Organismo	organism_id	INT		(organisms / id)		Relacionamento com a tabela “organisms”, através do campo “id”
Descrição	descricao	VARCHAR (255)				
Stream GBK	stream	BLOB				
Data	data	DATE				

TABELA	Fasta_logs					
Tabela de log para os arquivos Fasta de cada organismo						
C. Lógico	**C. Físico**	**Tipo**	**PK**	**FK (Tabela/Campo)**	**Restrições**	**Observações**
ID Fasta	id	INT	PK			
ID Organismo	organism_id	INT		(organisms / id)		Relacionamento com a tabela “organisms”, através do campo “id”
Descrição	descricao	VARCHAR (255)				
Stream Fasta	stream	BLOB				
Data	data	DATE				

APÊNDICE D - DIAGRAMA DE TELAS

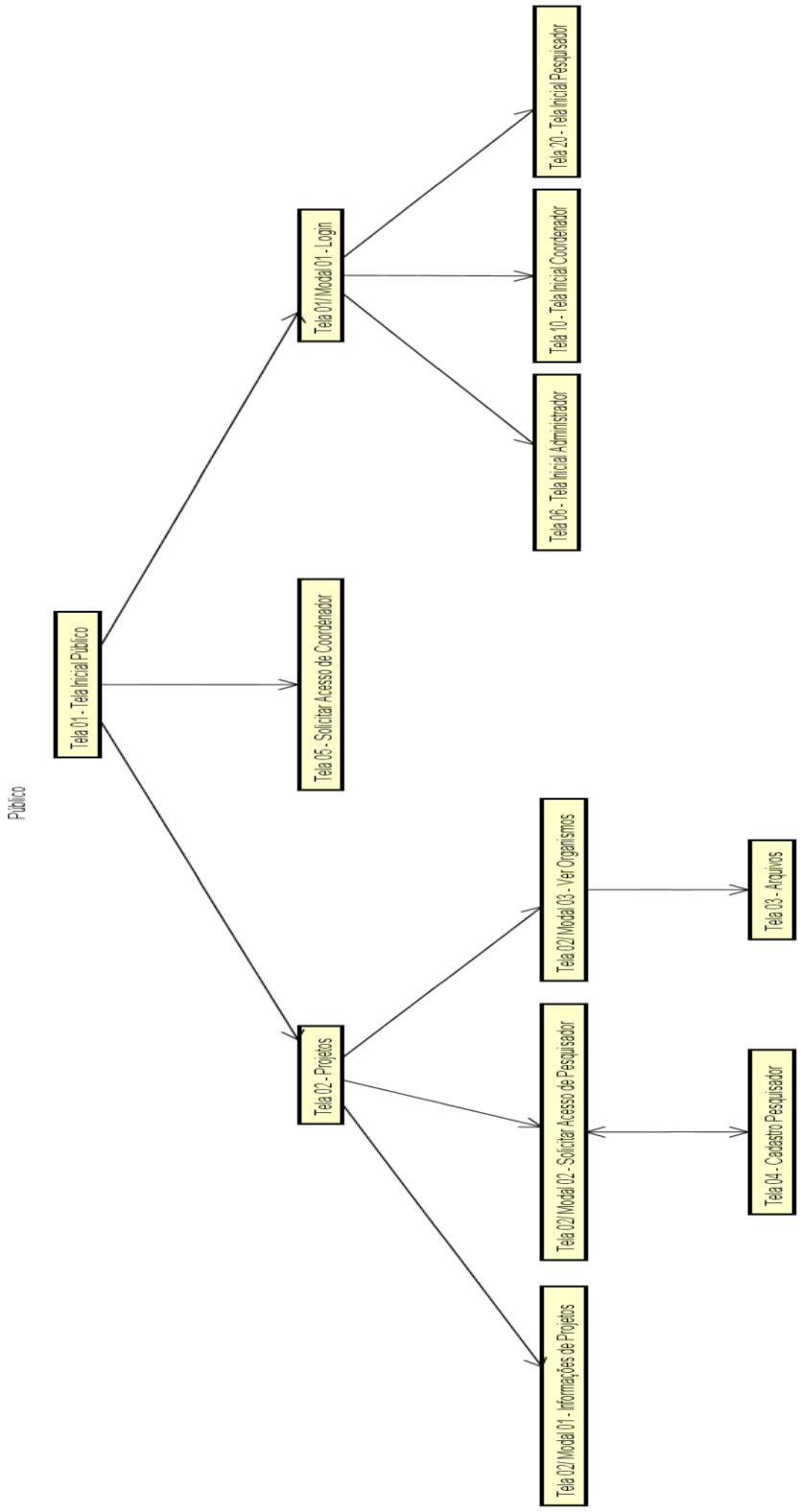

FIGURA 25 – DIAGRAMA DE TELAS – ACESSO PÚBLICO

Fonte: Os autores (2016).

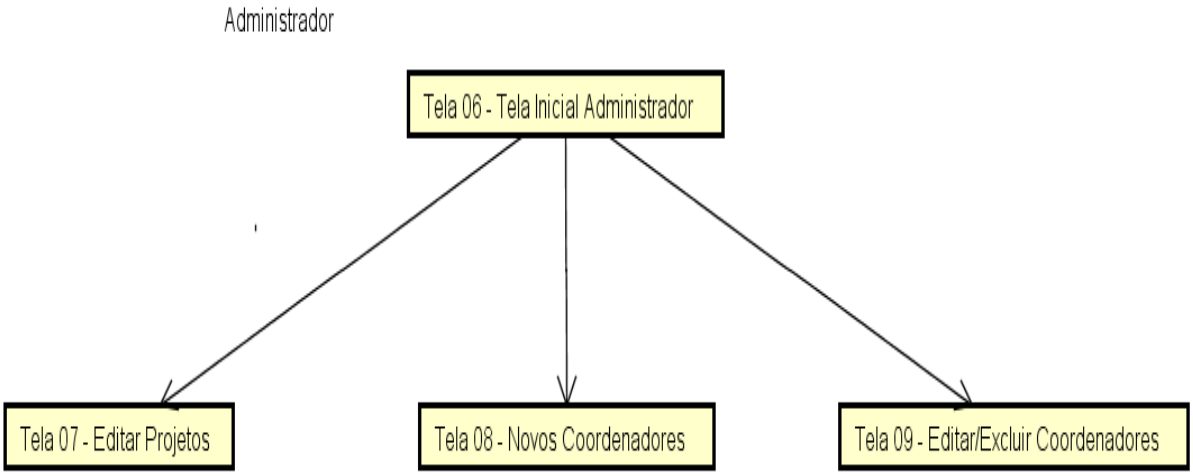

FIGURA 26 – DIAGRAMA DE TELAS – ACESSO DE ADMINISTRADOR

Fonte: Os autores (2016).

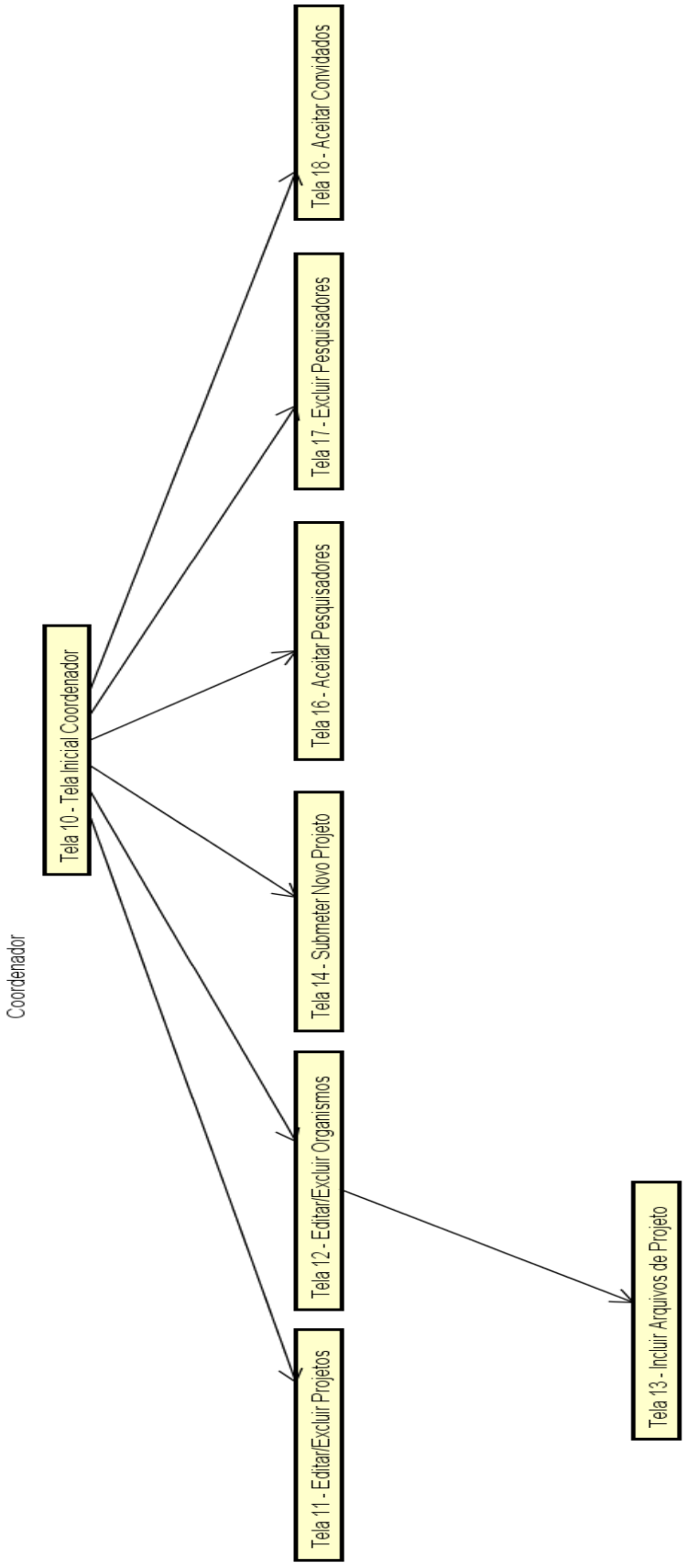

FIGURA 27 – DIAGRAMA DE TELAS – ACESSO DE COORDENADOR

Fonte: Os autores (2016).

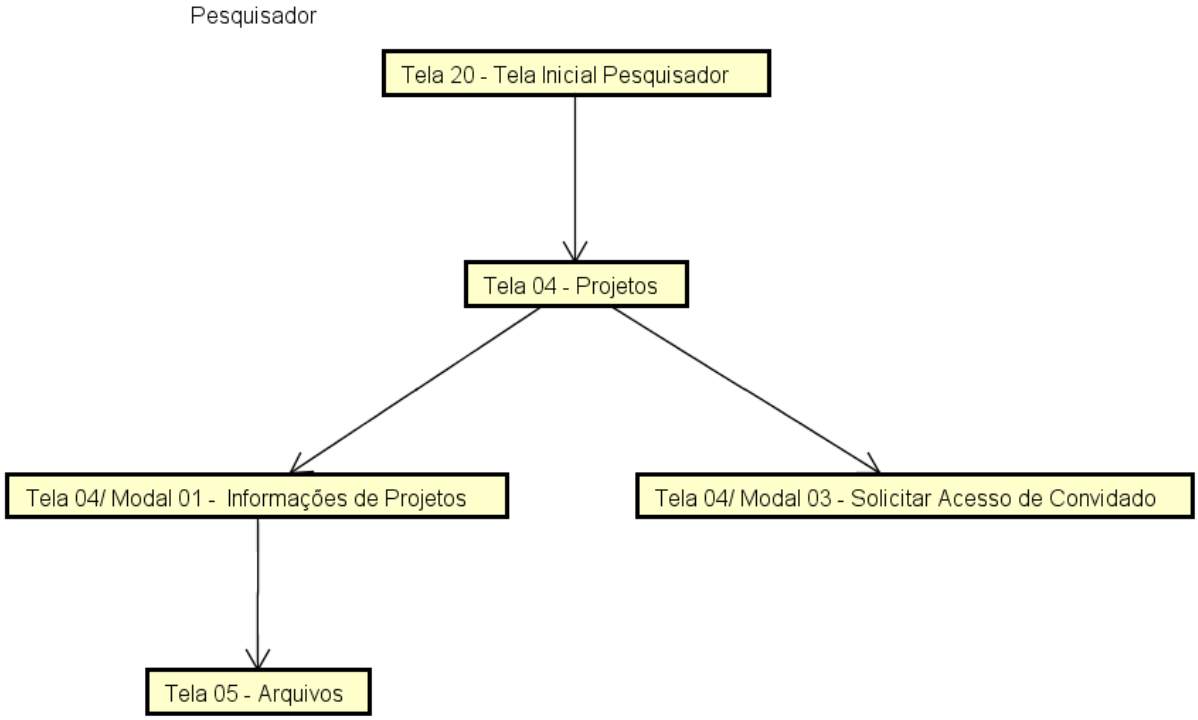

FIGURA 28 – DIAGRAMA DE TELAS – ACESSO DE PESQUISADOR

Fonte: Os autores (2016).

APÊNDICE E - DIAGRAMA DE CLASSES

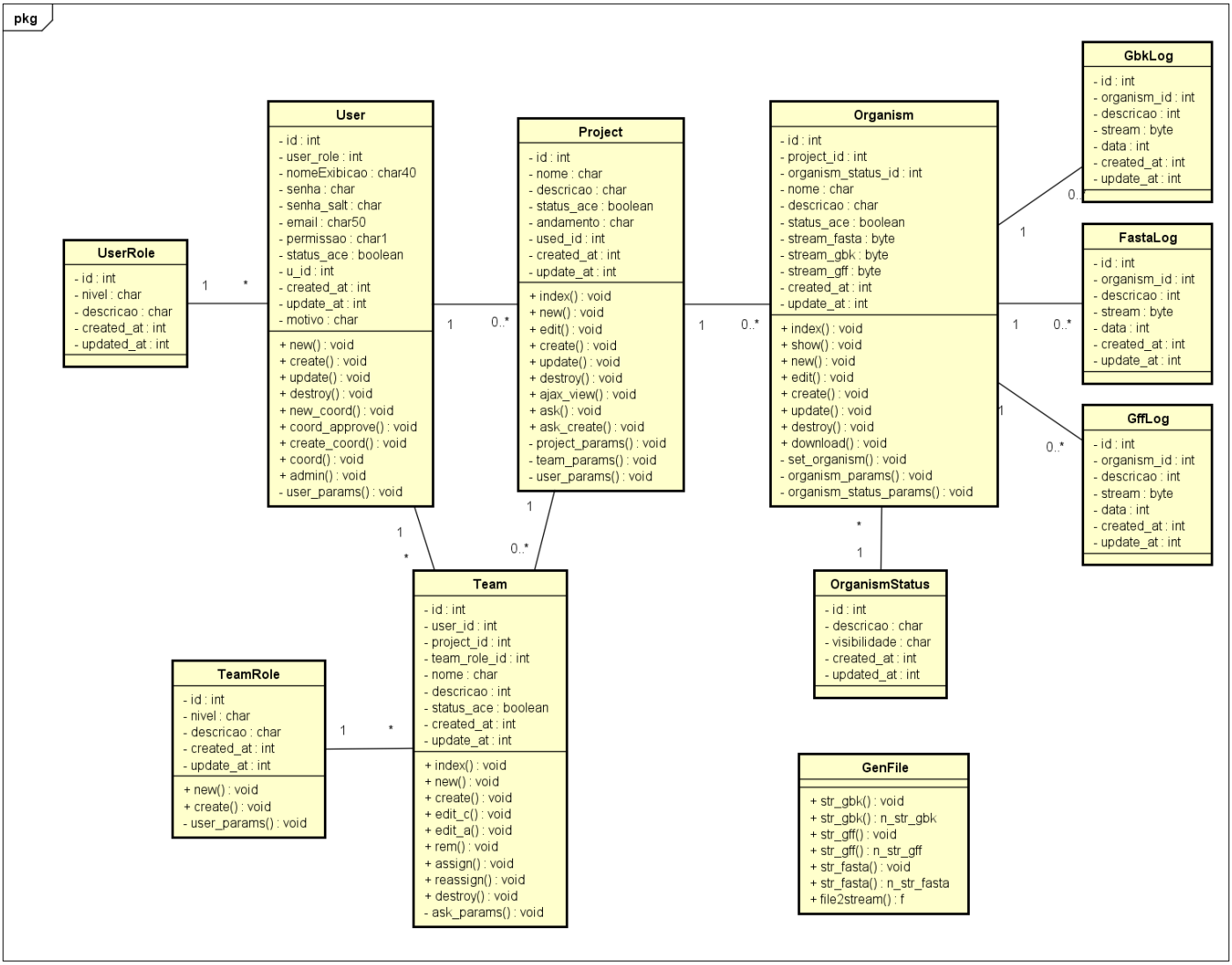

FIGURA 29 – DIAGRAMA DE CLASSES

Fonte: Os autores (2016).

APÊNDICE F – DIAGRAMAS DE ATIVIDADE

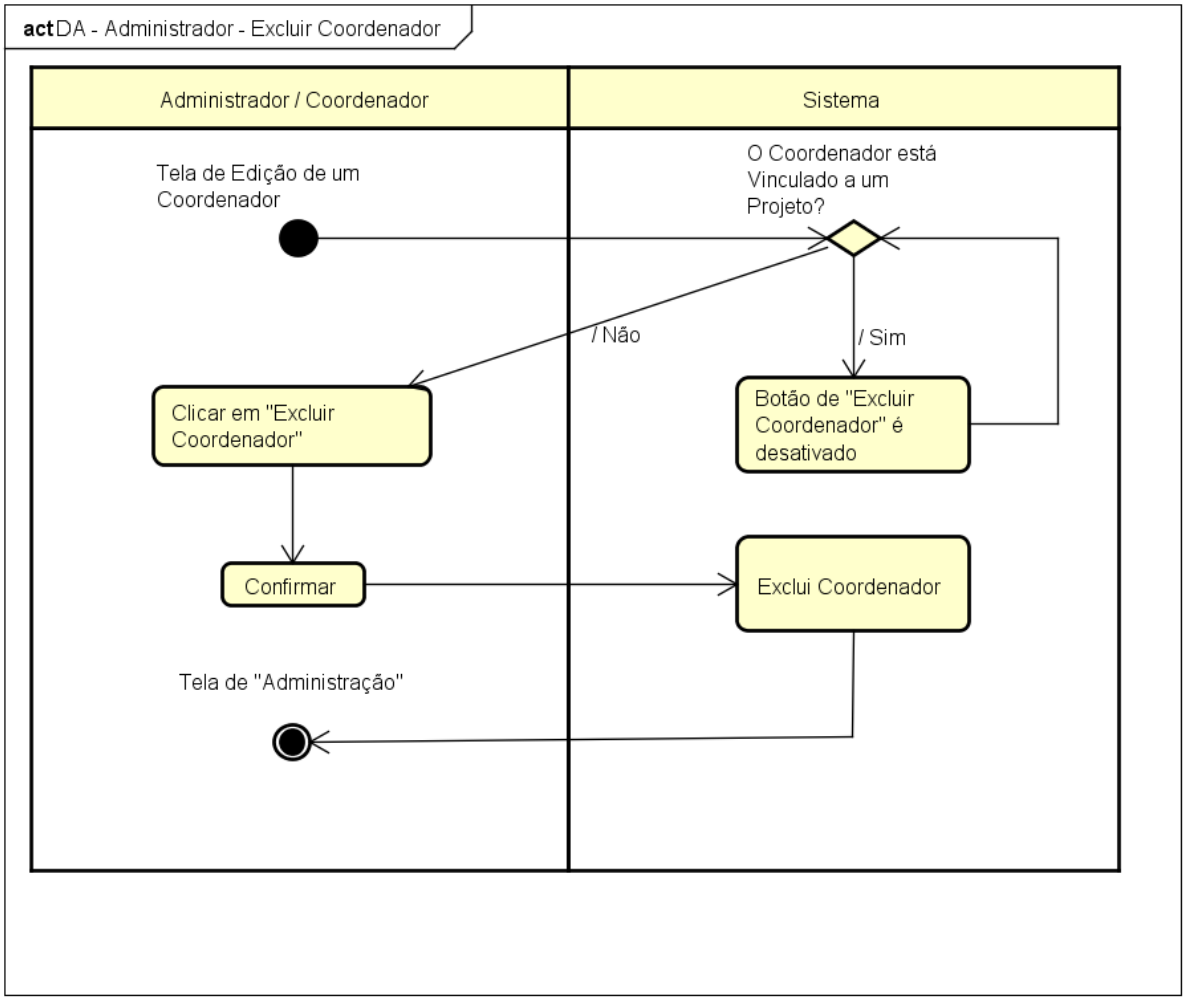

FIGURA 30 – DIAGRAMA DE ATIVIDADE – ADMINISTRADOR – EXCLUIR COORDENADOR

Fonte: Os autores (2016).

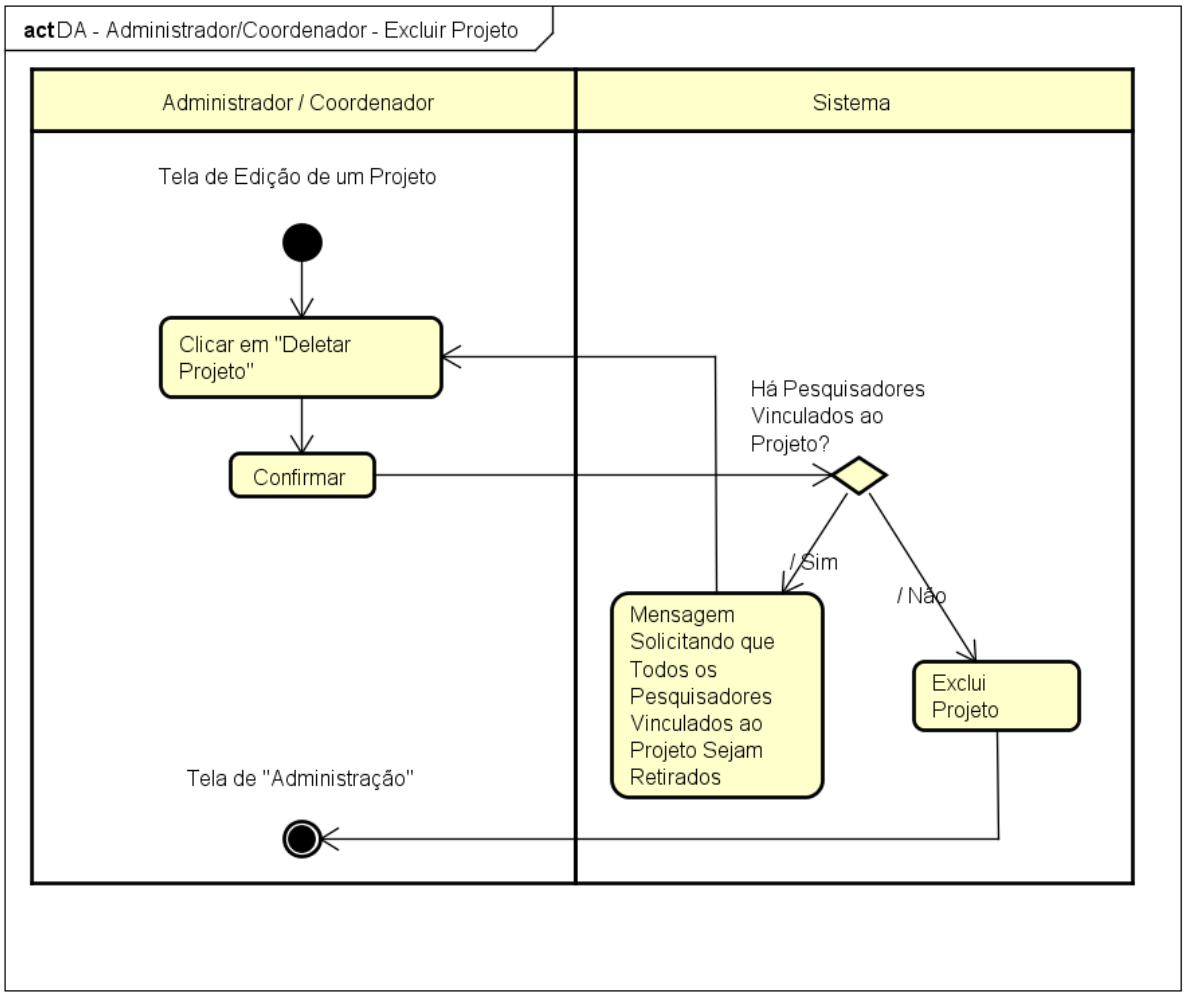

FIGURA 31 – DIAGRAMA DE ATIVIDADE – ADMINISTRADOR/COORDENADOR – EXCLUIR PROJETO

Fonte: Os autores (2016).

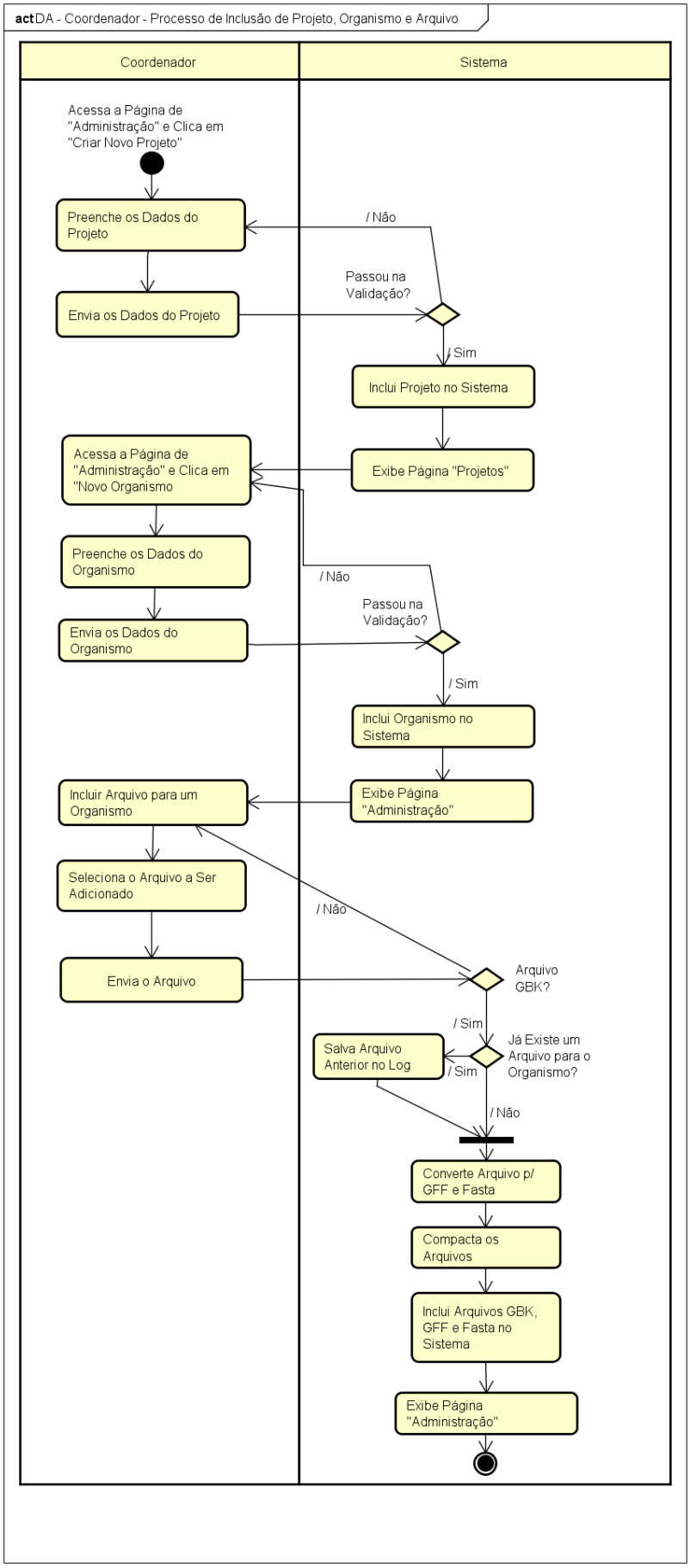

FIGURA 32 – DIAGRAMA DE ATIVIDADE – COORDENADOR – PROCESSO DE INCLUSÃO DE PROJETO, ORGANISMO E ARQUIVO

Fonte: Os autores (2016).

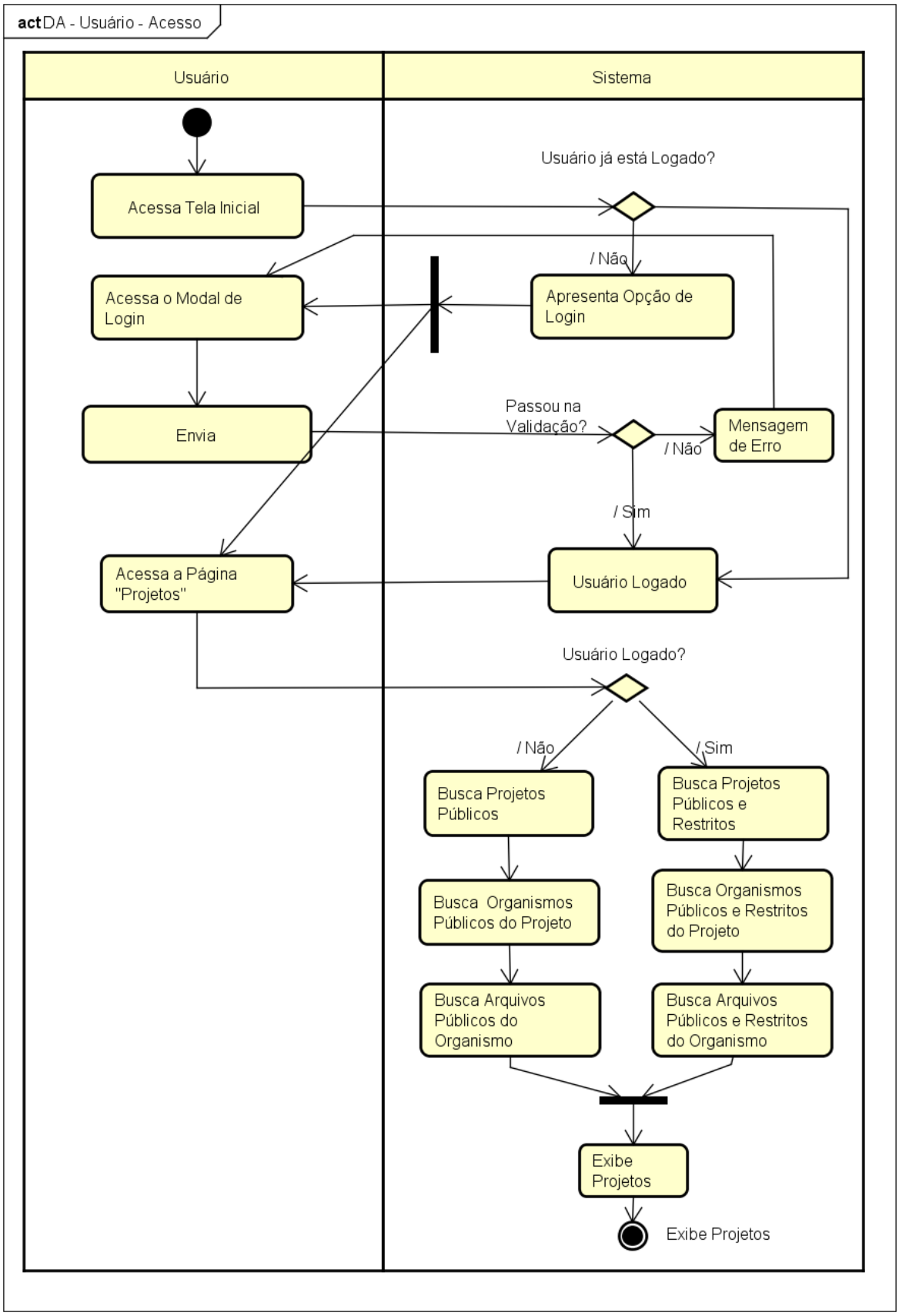

FIGURA 33 – DIAGRAMA DE ATIVIDADE – USUÁRIO - ACESSO

Fonte: Os autores (2016).

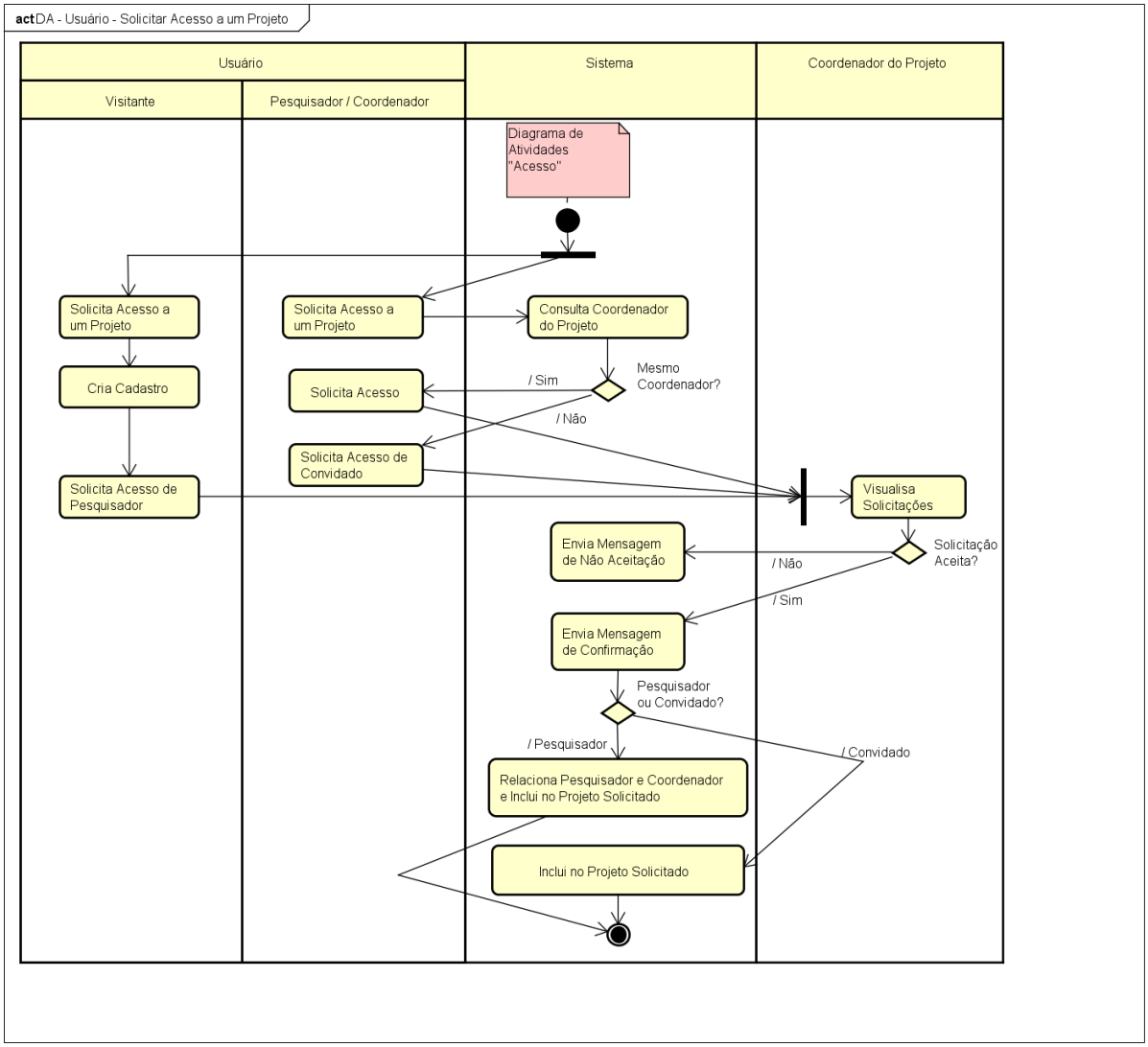

FIGURA 34 – DIAGRAMA DE ATIVIDADE – USUÁRIO – SOLICITAR ACESSO A UM PROJETO

Fonte: Os autores (2016).

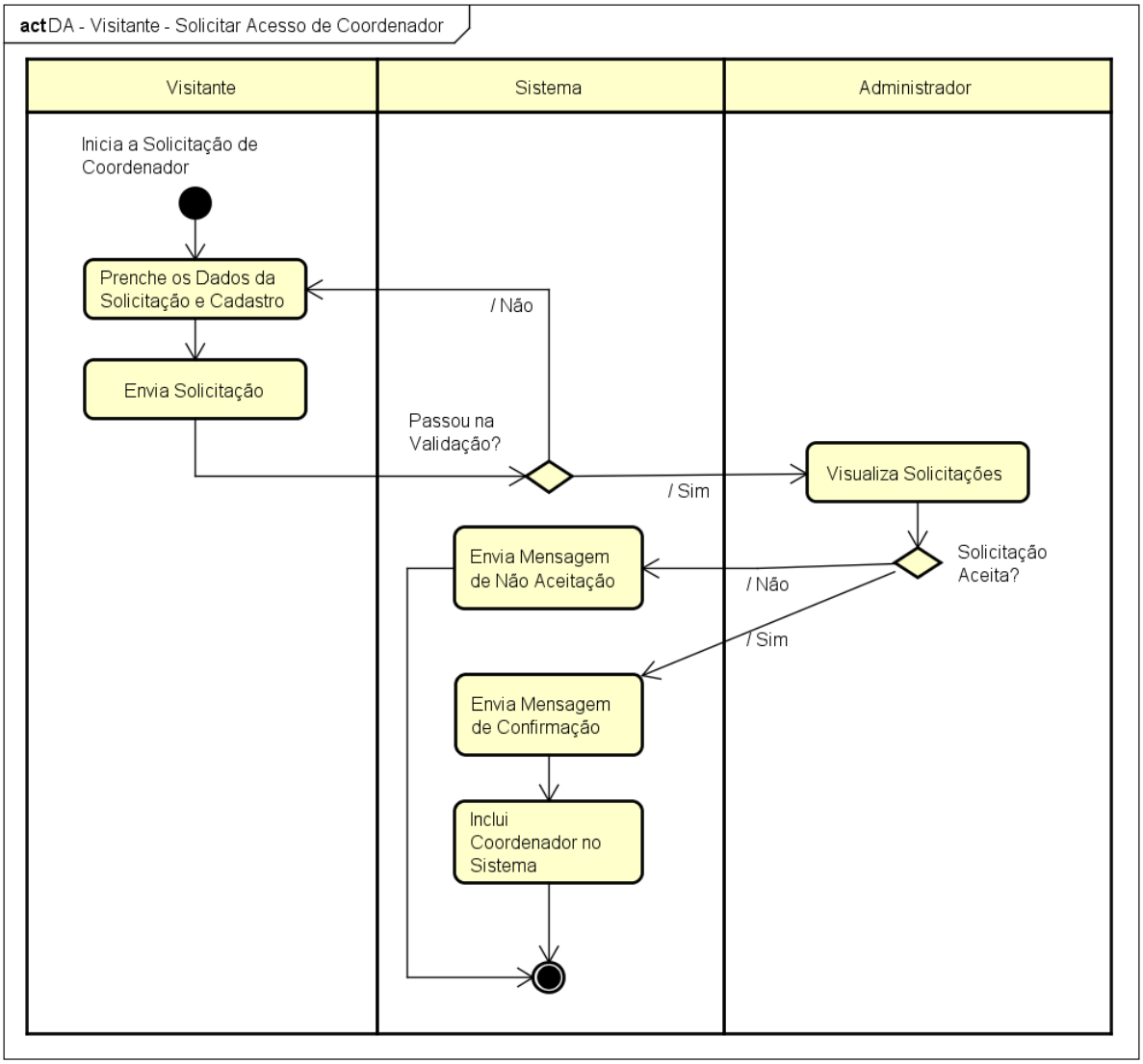

FIGURA 35 – DIAGRAMA DE ATIVIDADE – VISITANTE – SOLICITAR ACESSO DE COORDENADOR

Fonte: Os autores (2016).

APÊNDICE G – DIAGRAMA DE ESTADO

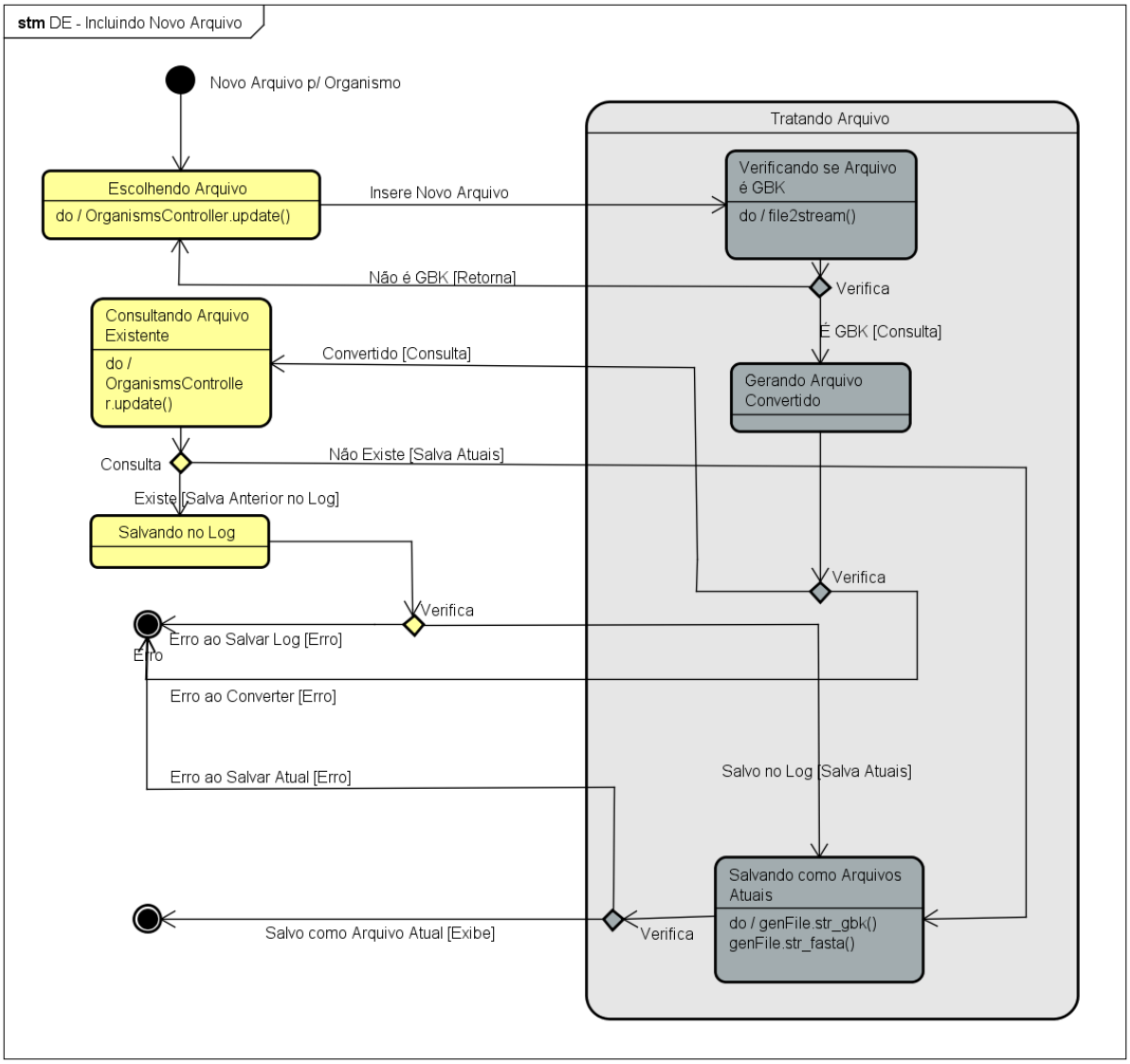

FIGURA 36 – DIAGRAMA DE ESTADO – INCLUINDO NOVO ARQUIVO

Fonte: Os autores (2016).

APÊNDICE H – PLANO DE RISCOS

TABELA 2 – PLANO DE RISCOS

Nº	Risco	Consequência	Ação	Probabilidade	Impacto	Classificação
1	Falta de tempo para o desenvolvimento do aplicativo.	Impossibilidade de entregar ou atraso na entrega.	Divisão de tarefas e monitoramento do andamento.	Alto	Alto	7
2	Mudança de requisitos do aplicativo.	Necessidade de mudar o direcionamento do projeto.	Mudanças na estrutura do projeto.	Alto	Alto	7
3	Conhecimento limitado em bioinformática.	Dificuldade no desenvolvimento do projeto.	Estudo e pesquisa na internet, livros e em outros matérias da PPG-BIOINFO.	Alto	Moderado	6
4	Perda de dados.	Impossibilidade de continuar o projeto.	Manter o repositório do GIT atualizado	Moderado	Alto	6
5	Desistência de um membro da equipe.	Não haveria a possibilidade de entrega do trabalho, devido a resolução do curso.	Conseguir outro membro para o grupo.	Moderado	Alto	6
6	Infraestrutura insuficiente.	Falta de equipamentos para desenvolvimento e testes.	Utilizar emuladores para as necessidades específicas.	Moderado	Moderado	4

APÊNDICE I – INSTALAÇÃO NO SERVIDOR

A instalação foi realizada no ambiente da PPG-BIOINFO rodando em servidor Linux, para instalação em outros tipos de plataformas configurações diferenciadas serão necessárias. O que estiver entre <> é variável e deve ser substituído por valores desejados:

Instalação do Ruby e da Gem do Rails:

- `sudo apt-get install ruby2.3 ruby2.3-dev`
- `gem install rails`

Instalação do JavaScript Runtime NodeJS:

- `sudo apt-get install nodejs`

Instalação do banco de dados PostgreSQL:

- `sudo apt-get install aptitude`
- `sudo apt-get install postgresql postgresql-contrib`
- `sudo apt-get install libpq-dev` (Para conseguir compilar a gem "PG" do PostgreSQL)
- `sudo gem install pg`

Configuração do banco de dados PostgreSQL:

- `sudo -u postgres createuser -s <aluno>`
    - (cria o usuário "aluno", usar o mesmo da conta do sistema ou criar outro usuário no Linux)
- `sudo -u postgres psql`
    - (entra no console do postgres, digitar os comandos abaixo para criar uma senha e sair do console do Postgres)
        - `\password <aluno>`
        - `<inserir uma senha>`
        - `<confirmar>`
        - `\q`

Configuração da aplicação:

Deve-se navegar até a pasta raiz da aplicação, ou seja base_fungos, e alterar o arquivo do banco de dados com usuário e senha criados:

- `sudo vim config/database.yml`

Alterar as linhas abaixo com os valores de usuário e senha, prestando atenção para não usar tabulação por '*tab*' no arquivo, e sim utilizar o espaçamento correto, pois se trata de um formato de serialização YAML:

- `username: <aluno>`
- `password: <senha do <aluno> no PostgreSQL>`

No diretório da aplicação (base_fungos):

- `bundle install` (esse passo irá ler a configuração da aplicação e instalar as gems necessárias)

Instalar o banco de dados da aplicação no PostgreSQL:

- `rake db:create`
- `rake db:migrate`
- `rake db:seed`

Depois de instalar e executar o rake db:seed, alguns dados serão inseridos nas tabelas:

- Valores iniciais para as tabelas de:
    - user_roles
    - team_roles
    - organism_statuses

- Usuário de administrador:
    - username: admin
    - password: senha

Estará tudo configurado e pronto. Para iniciar a aplicação, basta dar o comando abaixo, ainda na raiz da aplicação:

- `rails server` (ou rails server -b xxx.xxx.xxx.xxx para executar o servidor no IP do host atual permitindo acesso externo)
- Para parar o servidor, basta apertar CTRL+C no console

APÊNDICE J – DISPONIBILIDADE DO BLACK YEAST DATABASE

FIGURA 37 – INDISPONIBILIDADE DO SISTEMA “BLACK YEAST DATABASE”, PROJETO DO BROAD INSTITUTE, https://www.broadinstitute.org, Copyright © 2016 Broad Institute. All rights reserved.

Fonte: Os autores (2016).