UNIVERSIDADE FEDERAL DO PARANÁ

MARCELO MOREIRA MENDES

MATHEUS ROSENDO

OESLEI TABORDA RIBAS

RODRIGO OTHÁVIO FARIAS

# J-ORFINDER

CURITIBA

2007

MARCELO MOREIRA MENDES

MATHEUS ROSENDO

OESLEI TABORDA RIBAS

RODRIGO OTHÁVIO FARIAS

# J-ORFINDER

Trabalho de graduação apresentado à disciplina Projetos do Curso de Tecnologia em Informática do Setor da Escola Técnica da Universidade Federal do Paraná.

Orientador: Prof. Dr. Roberto Tadeu Raittz

Co-Orientador: Dieval Guizelini

CURITIBA

2007

MARCELO MOREIRA MENDES

MATHEUS ROSENDO

OESLEI TABORDA RIBAS

RODRIGO OTHÁVIO FARIAS

# J-ORFINDER

Trabalho de conclusão de curso avaliado e aprovado como requisito parcial para obtenção de grau de Graduado no Curso Superior de Tecnologia em Informática, Setor Escola Técnica da Universidade Federal do Paraná, pela seguinte banca examinadora:

Orientador: Prof. Dr. Roberto Tadeu Raittz
Escola Técnica, UFPR

Prof. Mario de Paula Soares
Escola Técnica, UFPR

Prof. Jaime Wojciechowski
Escola Técnica, UFPR

## RESUMO

Um dos grandes problemas dos pesquisadores que trabalham com análise laboratorial de dados biológicos é o reconhecimento e a localização de informações. Muitas vezes os pesquisadores se deparam com um grande volume de dados e a extração de informações destes dados acaba se tornando uma tarefa árdua e complexa. Com o intuito de auxiliar na solução desse problema, foi desenvolvido o J-ORFinder, uma ferramenta avançada para reconhecimento de genes codificadores de proteínas em procariontes, que utiliza técnicas de Inteligência Artificial, como Algoritmos Genéticos para extração de características e Redes Neurais no reconhecimento de padrões. O projeto é dividido em duas partes interdependentes: Um módulo WEB e um Módulo Desktop. O primeiro é a ferramenta disponível para o usuário final, um software que utiliza redes neurais previamente treinadas para auxilar na identificação de genes codificadores de proteínas que pode ser utilizado pelos pesquisadores através de um browser de qualquer computador conectado a internet. O segundo é utilizado localmente apenas pelo(s) administrador(es) do sistema e tem como objetivo exportar redes neurais devidamente treinadas e configuradas para serem usadas no módulo WEB.



Palavras-chave: ORF, códon, base nitrogenada, bioinformática, rede neural, localização de proteínas no DNA.

## ABSTRACT

One of the major problems of the researchers who work with laboratory analysis of biological data is the recognition and location of information. Often the researchers are faced with a large volume of data and extracting information from these data has become an arduous task and complex. In order to assist in the solution of this problem, has been developed the J-ORFinder, a powerful tool for recognition of protein encoders genes in procariontes, which uses techniques of Artificial Intelligence, as Genetic Algorithms for extraction of features and Neural Networks in recognition of standards. The project is divided into two interdependent parts: A WEB module and a Desktop module. The first is the tool available to the end user, a software that uses neural networks previously trained to auxilar in identifying proteins encoders genes that can be used by researchers through a browser from any computer connected to Internet. The second is used locally only by the administrator (s) of the system and aims to export neural networks properly trained and configured to be used in the WEB module.



Key words: ORF, codon, nucleotide bases, bioinformatics, neural network, protein pesearch

## LISTA DE ILUSTRAÇÕES

# LISTA DE QUADROS

## LISTA DE ABREVIATURAS E SIGLAS

AG - Algoritmo Genético

API - Application Programming Interface

CE - Computação Evolutiva

CSS - Cascading Style Sheet

DAO - Data Access Object

DNA - Deoxyribonucleic Acid

FAN - Free Associative Neurons

HTML - HyperText Markup Language

HTTP - HyperText Transfer Protocol

IA - Inteligência Artificial

IDE - Ambiente de desenvolvimento integrado

J2EE - Java 2 Enterprise Edition

JDBC - Java Database Connectivity

JPEG - Joint Photographic Experts Group,

JVM - Maquina virtual java.

JSP - Java Server Pages

JSTL - JSP Standard Tag Library

JUDE - Java and UML Developer Environment

MVC - Model View Controller

NCBI - National Center for Biotechnology Information

ORF - Open Reading Frame

PNG - Portable Network Graphics

RNA - Ribonucleic Acid

SDK - Software Development Kit

SGBD - Sistema Gerenciador de Bancos de Dados

SMTP - Simple Mail Transfer Protocol

SQL - Structured Query Language

SVG - Scalable Vectorial Graphics

UFPR - Universidade Federal do Paraná

UML - Unified Modeling Language

VO - Value Object

XML - eXtensible Markup Language

W3C - World Wide Web Consortium

# SUMÁRIO

# 1 INTRODUÇÃO

Recentemente a área de Bioinformática tem assumido um papel de destaque no campo de pesquisa em Biologia Molecular. Principalmente em projetos que envolvam seqüenciamento genômico e proteômico. A Bioinformática está no centro de uma revolução científica que analisa, em larga escala, fenômenos ligados à vida através de abordagens multidisciplinares.

A quantidade de seqüências depositadas em bancos de dados públicos, tais como Genbank, DDBJ e EMBL nos dá uma idéia do volume de dados que os pesquisadores estão continuamente gerando com os projetos supracitados, só o GenBank, por exemplo excedeu 100 gigabases em agosto de 2005.

O principal resultado de projetos de seqüenciamento genômico é a seqüencia completa do *Deoxyribonucleic Acid* (DNA) de um cromossomo pertencente a determinado organismo, ou seja, o Genoma deste organismo. Esta seqüência é formada por um alfabeto composto por quatro letras: A, C, G e T. Estas letras são representações dos compostos químicos chamados de base nitrogenada, cujos nomes são, respectivamente: adenina, citosina, guanina e timina. Os genes, em sua grande maioria, são extensões específicas nestas seqüências que carregam a informação necessária para se traduzir uma proteína.

Sendo assim, o genoma de um organismo, ou seja, sua seqüência de bases nitrogenadas do DNA, traz sozinho pouco valor para o pesquisador se este não dispuser de poderosas ferramentas computacionais que tenham a capacidade de mostrar, em meio a uma gigantesca sequência de caracteres A C G T, quais regiões são genes ou seqüências regulatórias, ou quais regiões não tenham sentido biologicamente. Por isso o reconhecimento de padrões em cadeias de DNA torna-se praticamente impossível sem o auxilio da computação. Neste contexto alguns softwares já foram desenvolvidos para auxiliar os pesquisadores no seu trabalho. Entretanto, ao entrar em contato com o Departamento de Bioquímica e Biologia Molecular do setor de Ciências biológicas da Universidade Federal do Paraná (UFPR), constatamos que os softwares que estão atualmente disponíveis para reconhecimento de genes codificadores de proteínas não são eficazes como deveriam, ocorrendo, muitas vezes durante o processo de busca, a necessidade de

utilizar vários softwares para uma única tarefa, além do fato de que a maioria deles são baseados em métodos estatísticos o que torna os programas muito “generalistas”, gerando, neste caso, baixas taxas de acerto (em torno de 80%).

Com o desafio de aumentar este percentual, foi desenvolvido como projeto de graduação, o J-ORFinder. O nome foi resultado de uma fusão da tecnologia utilizada com o foco do sistema: “J” de Java, linguagem de programação utilizada e “ORFinder”, localizador de ORFs em inglês. Termo utilizado pelos pesquisadores para definir os possíveis genes codificadores de proteínas em uma cadeia de DNA.

O diferencial do sistema é a utilização de técnicas de Inteligência Artificial: Algoritmos Genéticos para extração de características e Redes Neurais no reconhecimento de padrões. O J-ORfinder foi implementado dividido em dois módulos interdependentes: Um módulo Desktop e outro WEB. O módulo Desktop foi desenvolvido para ser utilizado localmente apenas pelo(s) administrador(es) do sistema e tem como objetivo exportar redes neurais devidamente treinadas e configuradas que serão utilizadas no reconhecimento de ORFs por pesquisadores no módulo WEB.

# 2 FUNDAMENTOS E CONCEITOS

## 2.1 PREDIÇÃO DE GENES

A palavra predição vem do latim *praedicatione* significa vaticínio, profecia, prognóstico. Objetivo da predição genes é identificar regiões de DNA que codificam proteínas. Porém antes de tratarmos diretamente da predição de genes vamos conceituar alguns temas importantes relacionados para facilitar o entendimento.

### 2.1.1 Genoma

Um genoma pode ser definido como todo o conjunto de informações genéticas de um organismo, sendo constituído usualmente por uma ou mais moléculas de DNA, na vasta maioria dos seres, ou *Ribonucleic Acid* (RNA), no caso de algumas famílias de vírus.

### 2.1.2 Seqüenciamento do genoma

O seqüenciamento do genoma (DNA) é um processo bioquímico que determina a ordem dos nucleotídeos (adenina A, citosina C, guanina G, timina T, que são os blocos que constituem a molécula de DNA) em uma amostra. Este processo pode ser realizado tanto manualmente quanto de maneira automatizada, através de máquinas sequenciadoras de DNA (*DNASequencers*). O esquema abaixo (FIGURA 1) demonstra de maneira simplificada o processo automatizado. Neste processo o DNA é fragmentado e recebe marcadores que permitem ao seqüenciador identificar os nucleotídeos por meio da diferença de cores entre eles,

que fica evidente quando submetidos a raios laser. Os dados obtidos pelos seqüenciadores são armazenados em meio digital para posterior analise.

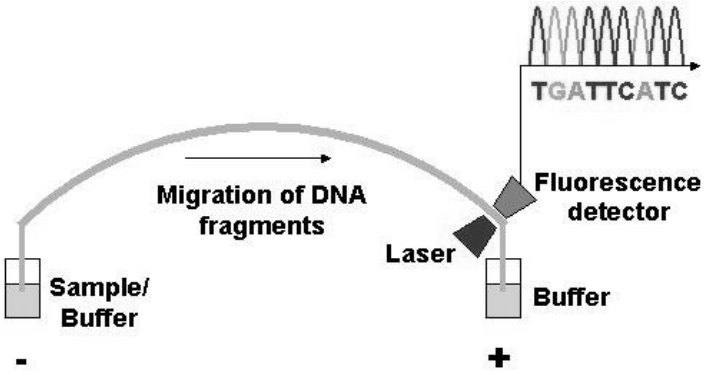



FIGURA 1 - ESQUEMA DO SEQUENCIAMENTO DO GENOMA

2.1.3 DNA

*Deoxyribonucleic acid* (DNA ou ADN em português Ácido desoxirribonucléico), é uma molécula orgânica que contém a "informação" que coordena o desenvolvimento e funcionamento de todos os organismos vivos. O seu principal papel é armazenar as informações necessárias para a construção das proteínas e RNA. Os segmentos de DNA que são responsáveis por carregar a informação genética são denominados genes. O restante da seqüência de DNA tem importância estrutural ou está envolvido na regulação do uso da informação genética. O DNA é um longo polímero formado por nucleotídeos, e nos organismos vivos, o DNA não existe como uma molécula única (fita simples), mas sim como um par de moléculas firmemente associadas. As duas longas fitas de DNA se enrolam em forma de uma dupla hélice (FIGURA 2).

FIGURA 2 - DNA OU ADN

### 2.1.4 Nucleotídeo

Nucleotídeo (FIGURA 3) é um composto químico e possui três partes: um grupo fosfato, uma pentose (molécula de açúcar com cinco carbonos) e uma base orgânica. Nas moléculas de DNA a pentose é uma desoxiribose. A base orgânica, também conhecida como base nitrogenada, é quem caracteriza cada um dos nucleotídeos, sendo comum o uso tanto do termo seqüência de nucleotídeos quanto o termo seqüência de bases. Os nucleotídeos estão presentes em ambas às fitas da dupla hélice, e são unidos os nucleotídeos da mesma fita por ligações fosfodiéster e unidos à fita complementar através das pontes de hidrogênio. Em geral, uma base ligada a um açúcar é chamada nucleosídeo e uma base ligada a um açúcar e um fosfato é chamada nucleotídeo. Portanto, o DNA pode ser referido como um polinucleotídeo.

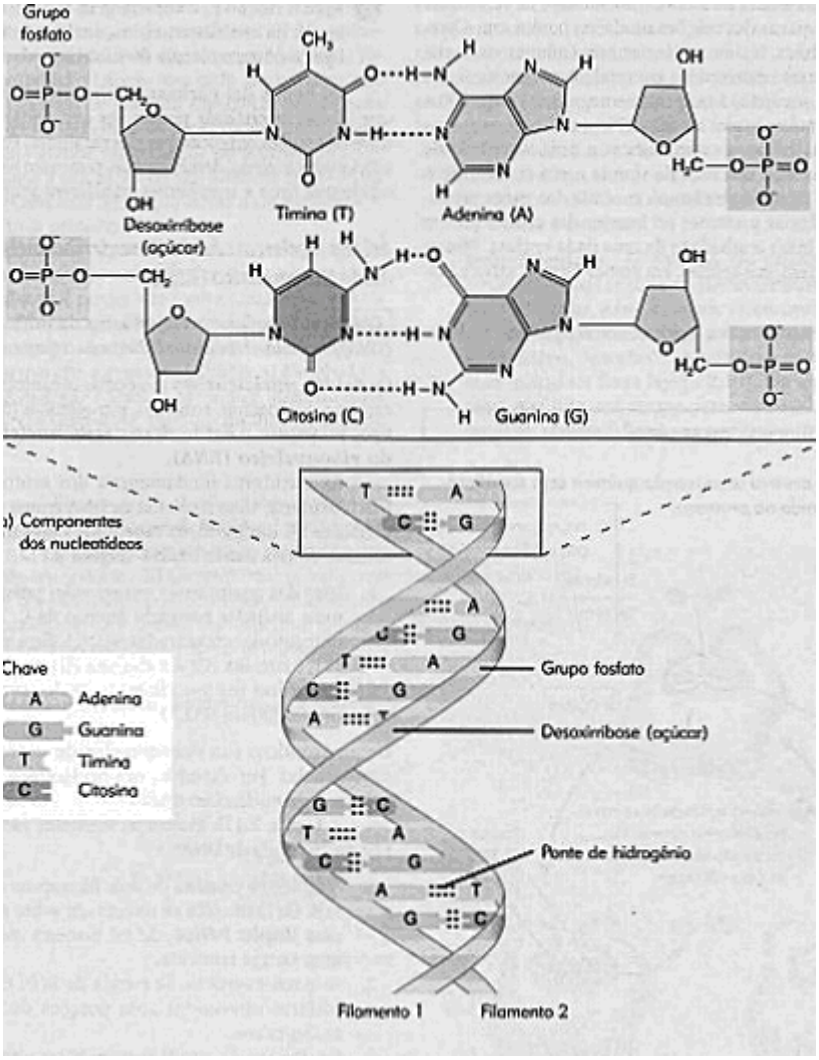



FIGURA 3 - ESTRUTURA QUÍMICA DO DNA

### 2.1.5 Códon

Códon é a combinação três a três dos nucleotídeos (bases), que são as unidades do código genético. Os quatro nucleotídeos combinados três a três formam 64 códons possíveis, que orientam a montagem dos aminoácidos na proteína.

### 2.1.6 Proteína

Proteínas são grandes compostos de unidades menores chamadas aminoácidos. Os aminoácidos contêm carbono, hidrogênio, oxigênio, nitrogênio e ocasionalmente sulfa (FIGURA 4). Todos os aminoácidos têm um grupo ácido e um grupo amino agregados a um átomo de carbono. Amino é o nome químico para a combinação de nitrogênio e hidrogênio nestes compostos. O grupo amino de um aminoácido pode se unir ao grupo ácido de outro aminoácido para formar um dipeptídeo. Esta união é chamada ligação peptídica. Quando mais de dois aminoácidos se juntam, forma-se um polipeptídeo. Uma proteína típica pode conter 500 ou mais aminoácidos agregados. O tamanho e a forma de cada polipeptídeo determina a proteína e a sua função. Algumas proteínas são feitas de vários polipeptídeos. Cada espécie de ser vivo tem suas proteínas características.

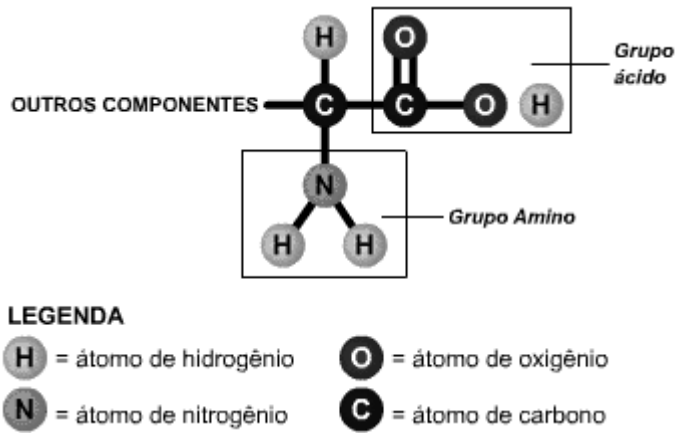



FIGURA 4 - ESTRUTURA QUÍMICA DE UM AMINOÁCIDO

Elas são os constituintes básicos da vida, nos animais as proteínas correspondem a 80% do peso dos músculos desidratados, cerca de 70% da pele e 90% do sangue seco.

### 2.1.7 Gene

Genes são segmentos de DNA (FIGURA 5) com um determinado número de nucleotídeos (bases) e com uma ordem própria, constituem uma unidade de linguagem química, e é neles que estão armazenadas as informações para produzir uma determinada proteína ou controlar uma característica, por exemplo, a cor dos olhos.

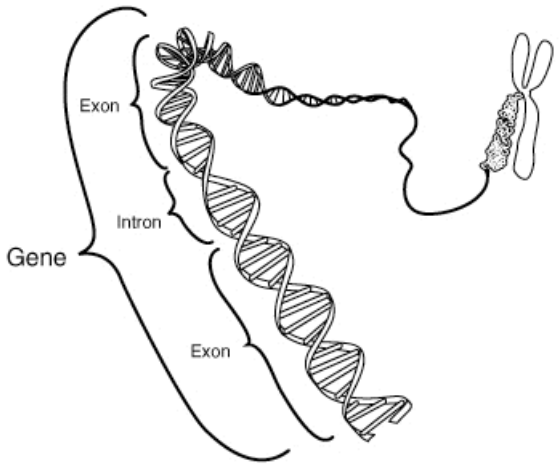



FIGURA 5 - ESQUEMA DE UM GENE

### 2.1.8 Íntrons

Íntrons, são segmentos de DNA de um gene (FIGURA 5) que não codificam qualquer parte da proteína produzida pelo gene e que separa a seqüência constituída pelos exons. Podem ser considerados como parte do DNA-lixo.

### 2.1.9 Exons

Exons são segmento de DNA de um gene (FIGURA 5) que codificam uma seqüência de nucleotídeos(proteina). Geralmente são adjacentes a um segmento de DNA não codificante, um íntron.

### 2.1.10 ORF

ORF(*Open Reading Frames*, ou fase aberta de leitura)(FIGURA 6) é uma região da seqüência nucleotídica (DNA) que pode ser um gene. Uma ORF é caracterizada pela presença de um *start* e um *stop* códon.

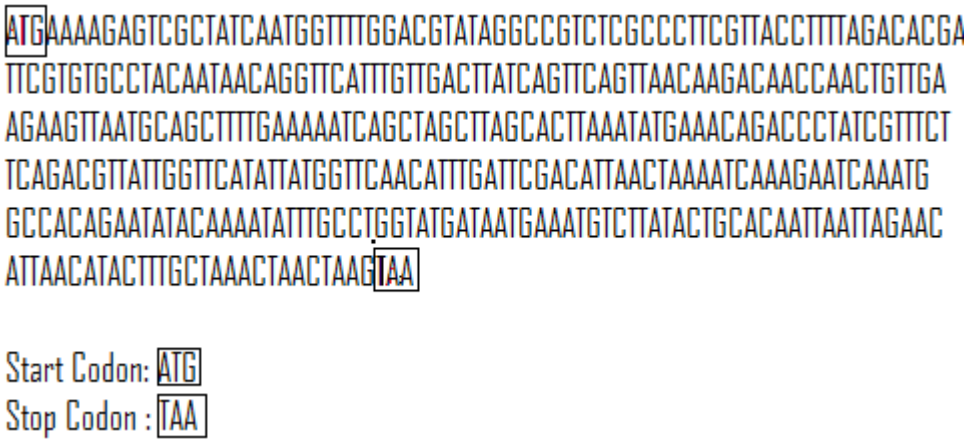



FIGURA 6 – EXEMPLO DE UMA ORF

### 2.1.11*Start* códon

*Start* códon, é códon que indica o possível inicio de uma ORF, o códon mais comum deste tipo é o ATG. Este códon na maioria das proteínas codifica o aminoácido metionina.

### 2.1.12 *Stop* códon

*Stop* códon, é códon que indica o possível final de uma ORF, o *stop* códon pode ser representado por uma das seguintes triplas: TAA, TGA e TAG. O *stop* códon não codifica aminoácidos.

### 2.1.13 Procarionte

Procariontes são seres vivos que possuem uma constituição celular simples onde o material genético fica disperso no citoplasma.

### 2.1.14 Eucarionte

Eucariontes são seres vivos mais complexos e sofisticados, o material genético de suas células é separado do citoplasma por uma membrana.

### 2.1.15 Predição

O seqüenciamento do genoma é processo longo e complexo que envolve os seguintes passos:

- predizer os genes;

• traduzir as regiões codificadoras em proteínas;

• atribuir à função das proteínas baseados na similaridade com seqüências já caracterizadas;

•avaliar a função e termos estruturais usando estruturas conhecidas ou derivadas de modelos.

Dos passos envolvidos no seqüenciamento do genoma, apenas a predição do gene será abordada neste trabalho. Como foi mencionado no inicio, predição significa dar um prognostico, ou seja predizer um gene significa ter a capacidade de identificar se um determinado seguimento do DNA (ORF) é codificador de proteína ou não. Atualmente existem duas abordagens para se fazer predição de gene, a predição por sinal e a predição por conteúdo.

A predição de gene por sinal identifica o gene indiretamente, procurando por um sinal particular associado a este gene. Exemplos de sinais encontrados em seqüências de nucleotídeos (ORF) são:

•Região de início de transcrição;

•Região de término de transcrição;

•Região de *Splice-Junctions;*

•Região de início da tradução (*start* códons);

•Região de término da tradução (*stop* códons).

A predição de gene por conteúdo reconhece o gene de forma direta, identificando segmentos de DNA que possuam características de regiões codificadoras. Por exemplo:

• A presença de aminoácidos comuns na formação de proteínas;

• A existência de “*preference codons*”.

Predizer um gene não é um processo trivial, e pode ser dificultado dependendo da origem do genoma. Basicamente o genoma pode ter duas origens procarionte e eucarionte.

Em um genoma procarionte a predição de gene é facilitada devido a baixa complexidade de seu material genético, uma vez que não há íntrons entre os exons, isto significa dizer que entre o *start* códon e o *stop* códons só haverão códons capazes de produzir aminoácidos que por sua vez constituirão uma proteína. Já em células eucariontes as genes geralmente estão fragmentados isto é há íntrons entre os exons o que dificulta o processo.

Qualquer que seja a abordagem utilizada o uso de recursos computacionais se faz presente devido o grande volume de dados que são manipulados, em especial às técnicas da Inteligência Artificial.

## 2.2 INTELIGENCIA ARTIFICIAL

Segundo Bittencourt (2006, p. 20), Inteligência Artificial (IA) é um ramo da ciência da computação que foi construído a partir de idéias filosóficas, cientificas e tecnológicas herdadas de outras ciências. O objetivo central da Inteligência Artificial (IA) é a criação de modelos para a inteligência e a construção de sistemas computacionais baseados nesses modelos. Dentre as diversas linhas de pesquisa da IA, nos concentraremos em duas que estão presentes neste trabalho, Algoritmos Genéticos e Redes Neurais Artificiais.

### 2.2.1 Algoritmo genético

Os algoritmos genéticos são uma família de modelos computacionais inspirados na evolução, que incorporam uma solução potencial para um problema específico (MIRANDA, 2007 p.1). Eles pertencem a uma área da IA chamada computação evolutiva (CE). Os conceitos de computação evolutiva têm sido empregados em uma variedade de disciplinas, desde ciências naturais e engenharia até biologia e ciência da computação (MIRANDA, 2007 p.1). A idéia básica, surgida nos anos 50, é simular o processo de evolução natural como um paradigma de solução de problemas (SILVA, 2006 p.1), a partir de sua implementação em computador. As ideais que permitem esta simulação são as seguintes:

- A criação de uma população de soluções, possivelmente obtida na sua primeira geração de modo aleatório, e na qual os indivíduos tenham registrados de modo intrínseco os parâmetros que descrevem uma possível solução ao problema proposto.
- A criação de uma entidade chamada função de avaliação capaz de julgar a aptidão de cada um dos indivíduos. Essa entidade não precisa deter o conhecimento sobre como encontrar a solução para o problema, mas apenas atribuir uma “nota” ao desempenho de cada um dos indivíduos da população.
- E, finalmente, a criação de uma série de operadores que serão aplicados à população de uma dada geração para obter os indivíduos da próxima geração. Estes operadores são baseados nos fenômenos que ocorrem na evolução natural. Os principais operadores citados na literatura são: (i)

recombinação (*crossover*): operador que simula a troca de material genético entre os ancestrais que, por sua vez, determina a carga genética dos descendentes; (ii) mutação: operador que realiza mudanças aleatórias no material genético; (iii) seleção (elitismo): permite escolher um individuo ou par deles para gerar descendência.

Para Bittencourt (2006, p. 320), o conceito chave da computação evolutiva (CE) é o de adaptação que unifica a abordagem quanto ao método de solução: uma população inicial de soluções evolui, ao longo de gerações que são simuladas no processo, em direção a soluções mais adaptadas, isto é, com maior valor da função de avaliação por meio de operadores de seleção(elitismo), mutação recombinação(*crossover*).

#### 2.2.1.1 População de soluções

O conjunto de soluções iniciais pode ser aleatório ou pode ser obtido a partir de técnicas convencionais para resolver instâncias mais simples do problema que está sendo tratado. Em termos computacionais as soluções são, geralmente, representadas por seqüências binárias. Por outro lado, usando-se soluções inicialmente aleatórias, pode-se usar sempre o mesmo algoritmo e os mesmos operadores. Por outro lado adaptando o conceito de CE a um problema especifico e empregando soluções iniciais obtidas por métodos convencionais, é necessário adaptar os operadores usuais da CE para o problema específico.

#### 2.2.1.2 Função de avaliação

Para definir a função de avaliação (*fitness*), é necessário encontra uma maneira de codificar as soluções para um problema que se quer resolver. O

resultado dessa codificação corresponde aos cromossomos na evolução natural e é chamado de genótipo. A partir desses cromossomos, a função de avaliação deve ser capaz de determinar a qualidade de uma solução.

### 2.2.1.3 Recombinação

A recombinação (*crossover*) é uma operação que se assemelha a reprodução sexuada na natureza onde há troca de material genético, onde um par de ascendentes da origem a um par de descendentes, e onde cada descendente herda partes aleatoriamente escolhidas de cada ascendente. A figura 6 exemplifica a operação de recombinação, onde o genótipo é representado por uma cadeia de bits.

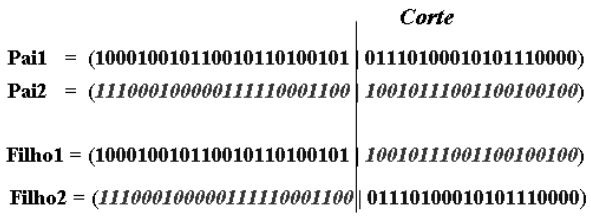



FIGURA 7 - EXEMPLO DE RECOMBINAÇÃO (CROSSOVER)

### 2.2.1.4 Mutação

A mutação é uma operação que leva a mudança aleatória de parte de uma solução. No caso mais simples de cromossomos codificados em binário, a mutação é a simples inversão de um bit (FIGURA 7).


FIGURA 8 - EXEMPLO DE MUTAÇÃO

2.2.1.5 Extração de características.

Os Algoritmos Genéticos têm sido bem sucedidos em diversas aplicações, particularmente em problemas de otimização. Tais problemas são caracterizados pela busca de boas soluções dentre um número elevado de possíveis soluções.

2.2.2 Rede neural artificial

Segundo Cerqueira, Andrade e Poppi(2004, p. 1) rede neural artificial é o nome dado a um conjunto de métodos matemáticos e algoritmos computacionais especialmente projetados para simular o processamento de informações e aquisição de conhecimento do cérebro humano. O estudo das redes neurais artificiais é também conhecido como conexonismo, e é uma das grandes linhas de pesquisa da IA, os primeiros trabalhos na área datam do ano de 1943. Mais formalmente redes neurais artificiais (FIGURA 8) podem ser definidas como sistemas paralelos distribuídos compostos por unidades de processamento simples (nodos ou neurônios artificiais) que calculam determinadas funções matemáticas (normalmente não-lineares).

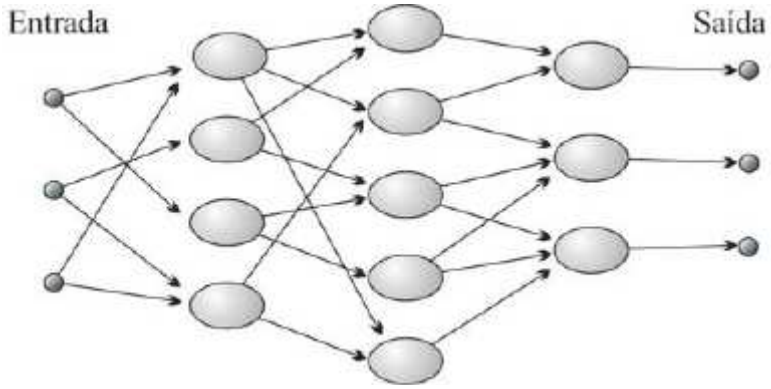



FIGURA 9 - REPRESENTAÇÃO SIMPLIFICADA DE UMA REDE NEURAL ARTIFICIAL

2.2.2.1 Reconhecimento de padrões.

Uma área em que as redes neurais tem grande aplicação é o reconhecimento de padrões, que tem por objetivo a classificação de objetos (padrões) em um número de categorias ou classes. O reconhecimento de padrões abrange as seguintes fases: a representação dos dados de entrada e sua mensuração, a extração das características e a identificação e classificação do objeto em estudo.

2.2.2.2 FAN – Free Associative Neurons

Neste trabalho utilizamos a rede FAN (Free Associative Neurons ) (RAITTZ, 2002) para o reconhecimento de padrões. Segundo Raittz (2002, p. 3) a rede FAN combina conjunto difusos e redes neurais para o reconhecimento de padrões. Ainda segundo Raittz (2002, p. 33), em uma rede FAN cada padrão de entrada é expandido para em uma vizinhança difusa. O grau de similaridade entre a vizinhança difusa e o padrão original de entrada é feito por meio das técnicas

utilizadas na teoria dos conjuntos difusos. Em FAN, dois princípios básicos fundamentam o processo de representação dos dados:

• um padrão representa mais que um simples ponto em termos de informação. Assim, na modelagem será considerada uma região ao redor do padrão;

• cada padrão contém mais informações que vão além dos valores individuais. Fazendo a correlação dos valores das características, a performance pode ser aumentada. O primeiro aspecto está relacionado com a decomposição do padrão; ou seja, a geração de uma vizinhança em torno do padrão de entrada. A vizinhança constitui um conjunto de padrões difusos no espaço de entrada, próximos ao padrão original.

O outro passo no processo de modelagem dos dados é a projeção de cada vizinhança difusa em uma região que denominada espaço FAN, que consiste em um conjunto de pontos do espaço. O algoritmo FAN mede a similaridade entre essas projeções e as unidades FAN.

Em suma, o processo de modelagem é realizado pelos seguintes procedimentos:

• decomposição do padrão (vizinhança difusa);

• projeção da vizinhança difusa.

Como objetivo deste trabalho não é descrever a implementação da rede FAN, e sim utilizá-la, dada a sua portabilidade, para maiores esclarecimentos recomendamos a leitura de (RAITTZ, 2002).

## 2.3 METODOLOGIAS DE SITEMAS

### 2.3.1 Orientação a Objeto

A abordagem orientada a objeto traz vantagens para a concepção e desenvolvimento de softwares, principalmente complexos e de grande porte, como por exemplo, os softwares usados na área de bioinformática. A orientação a objetos modela o mundo real em classes e instâncias, onde cada classe é a estrutura de uma variável. Nela, são declarados atributos e métodos que poderão ser executados ou acessados nas instâncias da mesma classe. As classes possuem uma função muito importante na modelagem orientada a objetos: elas dividem o problema, modularizam a aplicação e reduzem o nível de acoplamento do software.

Devido ao alto nível de abstração que se pode obter, o modelo orientado a objeto pode ser utilizado desde o início da concepção do software até seu desenvolvimento, possibilitando assim facilidade em se relacionar as partes do sistema aos conceitos que ele implementa, e a implementação com as decisões do projeto. Isso possibilita uma boa manutenção do software, durante e após o desenvolvimento, porque os dados e as funções que pertencem ao mesmo conceito ficam agrupados.

A orientação a objeto prioriza a estrutura de um sistema às suas funções, essa característica é uma das suas principais vantagens porque as funções que um sistema realiza podem sempre mudar com o tempo, enquanto raramente ocorrem mudanças na estrutura.

### 2.3.2 UML – *Unified Modeling Language*

A Linguagem de Modelagem Unificada (*Unified Modeling Language* - UML) é uma linguagem de diagramação ou notação para especificar, visualizar e documentar modelos de sistemas de software Orientados à Objeto. Oferece suporte ao desenvolvimento de software de grande porte através de métodos de análise e projeto, que modelam esse sistema de modo a fornecer a todos os envolvidos uma compreensão única do projeto.

Com a escolha da abordagem orientada a objetos, a utilização da UML traz a possibilidade de documentação adequada das informações sobre o sistema, visualização da estrutura do software através de representação gráfica, especificação desde a fase de análise até a fase de implementação e testes do sistema e realização de mapeamento dos modelos gerados.

Por meio de seus diagramas é possível representar graficamente softwares sob diversas perspectivas de visualização, dessa forma facilita a comunicação de todas as pessoas envolvidas no processo de desenvolvimento de um sistema.

### 2.3.3 Arquitetura em Camadas

A utilização da arquitetura de aplicativos em camadas oferece vantagens como a otimização das habilidades de equipes, a redução de custos associados ao desenvolvimento, além de favorecer a estensibilidade e reutilização do código.

A separação do sistema em camadas possibilita maior facilidade e clareza na implementação da persistência de dados, do controle de segurança, da comunicação em rede e do fluxo de visualização. Essa separação permite que os sistemas possam ser mais facilmente modificados e estendidos para atender a novas exigências, bem como possibilita que a interface com o usuário apresente várias visões de um só modelo, sem interferir com a lógica de negócio.

### 2.3.4 *Design Patterns*

Os *Design Patterns* (padrões de projetos) descrevem soluções já testadas para problemas recorrentes no desenvolvimento de sistemas de software orientados a objetos. Um padrão de projeto estabelece um nome e define o problema, a solução, quando aplicar esta solução e suas conseqüências.

Também podem ser definido como uma solução para um problema em um determinado contexto. Em termos de orientação a objetos, eles identificam classes, instâncias, seus papéis, colaborações e distribuição de responsabilidades.

A sua utilização é importante porque esses padrões já foram testados; eles refletem a experiência, o conhecimento adquirido e as idéias de desenvolvedores que tiveram sucesso utilizando-os em seus próprios trabalhos. *Design Patterns* são reutilizáveis, pois provêem uma solução pronta que pode ser adaptada a diferentes problemas, conforme necessário.

Sua utilização força uma forma otimizada e clara de comunicação entre os desenvolvedores, documentação e maiores possibilidades de exploração para alternativas de soluções para o projeto. Melhora a qualidade geral do programa, pois reduz a complexidade do código oferecendo uma abstração das classes e instâncias e proporciona redução no tempo de aprendizado de novas bibliotecas ou funções. Também acarretam um vocabulário comum de desenho, facilitando a comunicação, documentação e aprendizado dos sistemas de software.

#### 2.3.4.1 MVC – *Model-View-Controller*

O *design pattern* MVC (*Model-View-Controller*, ou modelo-visão-controlador) permite a separação do modelo de dados das várias formas que o dado pode ser acessado e manipulado. Um sistema MVC é dividido em modelo de dados, conjunto de visões e conjunto de controladores. MVC é útil principalmente para aplicações grandes e distribuídas, em que dados idênticos são visualizados e manipulados de

formas variadas. A chave para o MVC é a separação de responsabilidades: as visões fornecem a interface do usuário, os controladores são responsáveis por selecionar uma visão apropriada e por fazer alterações no modelo de dados, e o modelo é responsável por representar os dados-base da aplicação.

#### 2.3.4.2 DAO – *Data Access Object*

O *design pattern* DAO (*Data Access Object*, ou objeto de acesso a dados) implementa o mecanismo de acesso que trabalha com a base de dados. Basicamente, age como um "adaptador" entre o negócio e a base de dados, separando a lógica de acesso a dados da lógica de negócio, ele é responsável por recuperar e persistir os dados. Esse *pattern* permite adaptação a diferentes esquemas de armazenagem de dados sem afetar os componentes de negócio.

#### 2.3.4.3 VO – *Value Object*

VO (*Value Object*) é um objeto Java utilizado para armazenar informações. Quando a aplicação precisa trazer uma quantidade de informações significativa de um serviço remoto, fazer essa busca trazendo uma informação por vez acarretaria em alto custo de rede e de desempenho. Então, uma única requisição de rede traz todos os dados, que ficam armazenados no VO, minimizando assim o número de conexões com o servidor remoto. Também pode ser utilizado para enviar informações que serão armazenadas na base de dados, remetendo um conjunto de informações que serão persistidas no banco de dados.

#### 2.3.4.4 *Singleton*

O *design pattern Singleton* se refere a uma classe que tem uma única instância e um único ponto de acesso a essa instância. Ela pode ser estendida e utilizada sem que seu código seja modificado. É utilizado para representar algo único em um sistema.

#### 2.3.4.5 *Intercepting Filter*

Esse *pattern* refere-se a uma classe que recebe todas as requisições *HyperText Transfer Protocol* (http) que são efetuadas para o container web, essas requisições são “interceptadas” antes de chegar ao seu destino e também depois de serem processadas pelas classes correspondentes. Esse *pattern* é usado geralmente para realizar controle de acesso na aplicação, *logging* dos acessos entre outras ações.

#### 2.3.4.6 *Front Controller*

Possibilita que a aplicação tenha um único ponto de acesso, esse ponto recebe as requisições que são enviadas pelas camadas de apresentação e em seguida encaminha essa requisição para o componente de negocio que irá realizar o processamento necessário. Como a aplicação possui um único ponto de acesso, a manutenção do sistema se torna mais fácil de ser realizada. Além disso o sistema pode expandir-se de maneira organizada.

## 2.4 TECNOLOGIA E SOFTWARE

### 2.4.1 Java SDK

Versão: 6

Licença: *Open-source*

O Java SDK (*Software Development Kit*) é um pacote de desenvolvimento Java fornecido pela Sun Microsystems. Constitui um conjunto de programas que engloba compilador, interpretador e utilitários como *debbugers*, fornecendo um pacote de ferramentas básicas para o desenvolvimento de aplicações Java.

### 2.4.2 Netbeans

Versão: 5.5.1

Licença: *Open-source*

Ambiente de desenvolvimento integrado (IDE) para múltiplas linguagens e plataformas. Possibilita o desenvolvimento de diversos programas, aplicativos e ferramentas, de forma otimizada e padronizada. Oferece uma estrutura flexível com várias ferramentas que auxiliam no processo de desenvolvimento, como editores de: *HyperText Markup Language* (html), css, javascript, entre outros recursos que o tornam uma das ferramentas mais produtivas para o desenvolvimento de software.

### 2.4.3 Tomcat

Versão: 5.5

Licença: *Open-source*

O Tomcat é um servidor de aplicações Java para web. É desenvolvido com código aberto dentro do conceituado projeto Apache Jakarta e oficialmente endossado pela Sun, como a Implementação de Referência para as tecnologias Java Servlet e JavaServer Pages (JSP).

O Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo em um ambiente de produção. Tecnicamente o Tomcat é um container Web, cobrindo parte da especificação Java 2 *Enterprise Edition* (J2EE) com tecnologias como Servlet e JSP, tendo também a capacidade de atuar como servidor Web/HTTP, ou pode funcionar integrado a um servidor web dedicado.

### 2.4.4 PostgreSQL

Versão: 8.2.4

Licença: *Open-source*

Sistema gerenciador de bancos de dados (SGBD) relacional e orientado a objetos. Oferece mecanismos eficientes de segurança e integridade de dados, além de suportar quase todas as instruções da *Structured Query Language* (SQL) e ser de distribuição livre. O PostgreSQL é extremamente robusto e confiável, além de ser flexível e rico em recursos. É considerado objeto-relacional por implementar, além das características de um SGBD relacional, algumas características de orientação a objeto, como herança e tipos personalizados.

### 2.4.5 PgAdmin III

Versão: 1.6.3

Licença: *Open-source*

Abrangente ferramenta para gerenciamento de bases de dados PostgreSQL, projetado para atender às mais diversas necessidades, desde escrita de consultas SQL simples até construção de bases de dados complexas, com suporte aos mais recentes recursos do PostgreSQL. Desenvolvido por uma comunidade de especialistas em bancos de dados em todo o mundo, disponível em mais de trinta idiomas.

### 2.4.6 JDBC

Versão: 4

Licença: *Open-source*

Versão do driver: 8.2-505 JDBC 4

Licença do driver: BSD

*Java Database Connectivity* (JDBC) é uma *Application Programming Interface* (API) para execução e manipulação de resultados a consultas SQL através de Java, fazendo o envio de instruções SQL para qualquer banco de dados relacional. Define classes Java que permitem ao programador o processamento dos resultados dos comandos SQL emitidos. Pode ser utilizada com diferentes tipos de bancos de dados.

### 2.4.7 JUDE - Java and UML Developer Environment

Versão: 5.0.1

Licença: Open-source

JUDE (*Java and UML Developer Environment*) é uma poderosa ferramenta gráfica para modelagem UML. Suporta modelagem de softwares orientados a objeto desenvolvidos em Java. Possui características como adição de métodos no

diagrama de seqüência e a reflexão dessa alteração no diagrama de classes. Apresenta boa performance.

### 2.4.8 DBDesigner

Versão: 4.0.5.6

Licença: *Open-source*

Ferramenta visual de modelagem de banco de dados. É multi-plataforma, oferece suporte a vários tipos de banco de dados, permite engenharia reversa, possibilita criação de relacionamentos e tabelas de forma visual, além da importação de informações de bancos de dados existentes.

### 2.4.9 JSP

Especificação: 2.0

Licença: *Open-source*,

Sob os termos de uso da Sun Microsystems JSP (*Java Server Pages*) é uma tecnologia utilizada no desenvolvimento de aplicações para Web. Pelo fato de ser baseada na linguagem de programação Java, tem a vantagem da portabilidade de plataforma, que permite a sua execução em diferentes sistemas operacionais. Permite ao desenvolvedor de páginas para internet a produção de aplicações que acessam o banco de dados, manipulam arquivos no formato texto, recuperam informações a partir de formulários e também informações sobre o visitante e o servidor.

### 2.4.10 HTML

Especificação: 4.01

Licença: Sob os termos de utilização da World Wide Web Consortium(W3C) o HTML (*HyperText Markup Language*, ou linguagem de formatação de hipertexto) trata-se de uma linguagem de marcação utilizada para produzir páginas na internet. Documentos HTML são feitos para prover estrutura lógica da informação destinada à apresentação de páginas da rede mundial de computadores.

### 2.4.11 CSS - Cascading Style Sheet

Versão: 2.1

Licença: Sob os termos de utilização da W3C World Wide Web Consortium CSS (*Cascading Style Sheet*, ou folha de estilo em cascata) é um mecanismo simples para adicionar estilos (fontes, cores, espaçamentos) aos documentos Web. O uso de CSS permite separar a marcação HTML da apresentação do site, ou seja, o HTML destina-se unicamente a estruturar e marcar o conteúdo, enquanto a CSS fica responsável pela parte visual do documento.

### 2.4.12 JSTL - JSP Standard Tag Library

Versão: 1.1 para JSP 2.0

Licença: Open-source, sob os termos de uso da Sun Microsystems JSTL (JSP *Standard Tag Library*) consiste em uma coleção de bibliotecas, cada uma com um propósito bem definido. Permitem que páginas JSP sejam escritas sem código Java, aumentando assim a legibilidade do código e a interação entre desenvolvedores e web designers.

### 2.4.13 Log2Java

Versão: 1.2.13

Licença: *Open-source*

*Framework* de *logging* de mensagens utilizado para indicar o comportamento de determinada aplicação. Tem como características a flexibilidade e rapidez de geração de *logging* em tempo de execução, sem inserir grandes custos de desempenho para a aplicação. A importância do *logging* pode ser notada em fases de desenvolvimento, para atividades de rastreamento, passando principalmente por fases de testes e integração. Entretanto, seu maior uso é pós-implantação, para trace de acompanhamento e verificação do funcionamento da aplicação.

### 2.4.14 XML

Versão: 1.0

Codificação: UTF-8

Licença: Sob os termos de utilização da W3C *World Wide Web Consortium* o XML (*eXtensible Markup Language*) é uma linguagem universal que permite a troca de informações de forma estruturada através da Internet. Permite que os desenvolvedores transportem dados de um servidor para outro da rede de forma transparente e organizada.

### 2.4.15 JavaDB

Versão: 10.2.2.0

Licença: *Open-source*

JavaDB é um banco de dados relacional escrito em Java. Possui suporte a *store procedure, triggers* entre outros recursos. Por ser escrito em Java ele é portável e pode ser executado em qualquer plataforma que possua uma Maquina virtual java (JVM). Além da portabilidade outras características que ele possui é a segurança, facilidade de uso e também o seu tamanho reduzido, com apenas 2 Mb de espaço em disco já é possível executá-lo. Por essas características ele tornou-se uma das melhores soluções open-source de banco de dados embarcado.

### 2.4.16 Jfreechart

Versão: 1.0.5

Licença: *Open-source*

É uma opção open-source de biblioteca gráfica escrita 100% em Java. Com o Jfreechart é possível confeccionar gráficos de qualidade profissional de maneira fácil e rápida. Entre as suas vantagens destacam-se a possibilidade de exportação para formatos png (*Portable Network Graphics*), jpeg (*Joint Photographic Experts Group*) e SVG (*Scalable Vectorial Graphics*) , também integração com componentes gráficos swing. Outra característica que deve ser mencionada é a sua documentação que é bem completa e detalhada.

### 2.4.17 Commons-Math

Versão: 1.0

Licença: *Open-source*

Conjunto de bibliotecas matemáticas do conceituado grupo Apache. A *commons math* possui classe e métodos prontos para se trabalhar com operações que envolvam cálculos estatísticos e outros cálculos matemáticos.

### 2.4.18 Commons-FileUpload

Versão: 1.2

Licença: *Open-source*

A API Commons-FileUpload é uma biblioteca do grupo Apache para upload de arquivos para aplicações web. Com essa biblioteca o envio e recepção de arquivos através de servlet se torna uma tarefa fácil de ser realizada. Com um conjunto de classes específicas que acompanham a biblioteca, o processo de recepção do arquivo e gravação do mesmo em uma base de dados pode ser realizado com poucas linhas de código.

### 2.4.19 JFan

Versão: 2.0

Licença: Sob os termos de utilização da UFPR

API escrita em Java que foi desenvolvida por alunos do curso de graduação em Tecnologia em Informática da UFPR. A JFan é uma API que possibilita a consulta a uma rede *de Free Associative Neurons* (FAN) que tenha sido previamente treinada, com isso a integração de redes neurais em projetos de software ocorre de maneira mais rápida e fácil.

### 2.4.20 EasyFan

Versão: 2.0

Licença: Sob os termos de utilização da UFPR

O EasyFAN foi desenvolvido por alunos do curso de graduação de Tecnologia em Informática da UFPR. É uma ferramenta pronta para ser utilizado por iniciantes da área de Inteligência Artificial, mais precisamente reconhecimento de padrões, bem como por pessoas experientes que desejarem conhecer e explorar os benefícios da técnica FAN. Possui modelos de "*wizard*" e também permite refinar e avaliar o treinamento, explorando o monitoramento eficiente da rede apresentado em forma de tabelas e gráficos.

### 2.4.21 JavaMail

Versão: 1.4.1

Licença: *Open-source*

O JavaMail é um framework desenvolvido pela Sun, independente de plataforma, que fornece uma maneira simples de se integrar envio de e-mail a aplicações. Com o JavaMail é possível construir aplicações que forneçam o serviço de envio de e-mail de modo rápido e seguro. Possui suporte a conexões *Simple Mail Transfer Protocol* (smtp) autenticadas entre outras funcionalidades.

### 2.4.22 BioJava

Versão: 1.5

Licença: *Open-source*

É um framework escrito em Java para processar dados biológicos. Inclui uma série de recursos como manipulação de seqüências biológicas, verificação de arquivos com genes, ferramentas para fazer análise de dados biológicos entre outros recursos disponíveis. Atualmente é o principal framework Java para trabalhar

com bioinformática, possui uma comunidade de usuários ativa e a cada dia são acrescentados novos recursos.

### 2.4.23 Subversion

Versão: 1.4

Licença: *Open-source*

Subversion, também conhecido por SVN, é um sistema de controle de versão desenhado especificamente para ser um substituto moderno do CVS. Possui vários recursos como operações de *commit* verdadeiramente atômicas, integração com WebDAV, baixo custo de processamento na mudança de diretórios e tratamento eficiente de arquivos binários.

### 2.4.24 Tortoise

Versão: 1.4.4

Licença: *Open-source*

É um dos clientes que existem para o Subversion, roda exclusivamente em ambiente Windows possuindo grande integração com o Windows Explorer. Com o Tortoise pode-se fazer *commit* e *update* de maneira rápida e fácil. É uma das melhores ferramentas que existe para trabalhar com o Subversion, pois foi desenvolvido pela mesma comunidade de desenvolvedores.

# 3 SOLUÇÃO PROPOSTA

O J-ORFinder é um sistema para ser usado em pesquisas de dados biológicos, mais especificamente na catalogação de genes codificadores de proteínas no genoma (código de DNA) de seres procariontes, possibilitando que pesquisadores identifiquem genes e até mesmo descubram novos genes que ainda não foram catalogados no meio científico.

O projeto é dividido em duas partes interdependentes: Um módulo WEB e um Módulo Desktop. O primeiro é a ferramenta disponível para o usuário final, um software que utiliza redes neurais para reconhecimento de ORFs que pode ser utilizado pelos pesquisadores através de um browser de qualquer computador conectado a internet. O segundo é utilizado localmente apenas pelo(s) administrador(es) do sistema e tem como objetivo exportar redes neurais devidamente treinadas e configuradas para serem usadas no módulo WEB. Para o treinamento destas redes neurais é prevista a utilização do sistema EasyFan e os dados utilizados neste treinamento são gerados pelo J-ORFinder Desktop que implementa técnicas de computação evolutiva para a extração de características.

## 3.1 MÓDULO DESKTOP

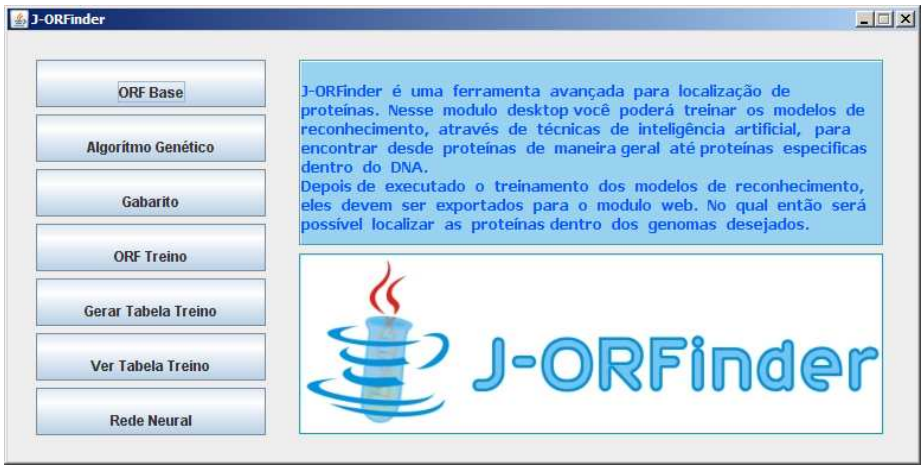



FIGURA 10 - TELA INICIAL DO MÓDULO DESKTOP

O módulo Desktop tem como objetivo gerar um arquivo XML contendo a rede neural e os gabaritos que deram origem às características que foram usadas para o treinamento desta rede. Cada gabarito é um conjunto de números inteiros que combinados com uma ORF (possíveis genes codificadores de proteínas) gerará uma característica que é um número de dupla precisão. Da biologia, cada ORF é formada por um conjunto de códons e cada códon é formado por 3 bases nitrogenadas. Como são 4 bases nitrogenadas possíveis (A, C, T e G), matematicamente existem 64 possíveis códons diferentes. Para que este conjunto de códons seja uma ORF, necessariamente, o primeiro códon deve ser um códon start e o último um códon stop. O códon start mais comum é o ATG e os códons stop conhecidos são os TGA, TAG e TAA. Mas nem toda ORF, ou seja, parte do DNA entre um códon start e stop é uma região codificadora de proteína. É para fazer esta distinção que o sistema foi desenvolvido.

### 3.1.1Algoritmo Genético

Os gabaritos citados anteriormente são o resultado do processamento do Algoritmo genético sobre ORFs previamente cadastradas no sistema. Estas ORFs que servirão de base para o Algoritmo Genético (AG) foram chamadas por nós de ORFs Base. O objetivo do AG implementado no sistema é extrair o padrão existente destas ORFs base. Por isso, a escolha destas ORFs base deve ser feita cuidadosamente. Todo processamento do AG deve ser feito em cima de uma ORF base real, ou seja, uma ORF codificadora de proteína de algum organismo conhecido pelo pesquisador. O AG aceita também que seja feito o processamento baseado em duas ORFs Base, neste caso ele busca diferenciar uma ORF da outra, por este motivo, é recomendável que uma ORF seja codificadora de proteína e a outra não.

#### 3.1.1.1 Método de escolha

O método de escolha dos indivíduos para reprodução nos AGs é parte vital e determinante para se alcançar uma boa solução.

O método da “roleta”, é hoje o método mais comum para seleção de indivíduos em Algoritmos Genéticos e pode ser eficiente para alguns tipos específicos de problemas em comparação com outros métodos como o Deterministic Sampling e o método do Torneio, como pode ser observado em Soares(1997, p. 93) “Algoritmos Genéticos: Estudos Técnicas e Aplicações”. Apesar disso este método dá margem para um conhecido problema enfrentado na implementação de AGs: a Convergência Prematura. Este problema acontece pela falta de diversidade causada pelo método, pois cromossomos menos aptos acabam tendo baixíssimas chances de serem escolhidos para se reproduzir, isso faz com que toda a população acabe convergindo para uma solução chamada de Máximo local, onde pouco se varia de um cromossomo para outro. No intuito de resolver este problema o nosso professor orientador Dr. Roberto Tadeu Raitzz, projetou um novo algoritmo de seleção, ainda não conhecido pela comunidade acadêmica, que tem como diferencial, deixar as chances de cada indivíduo proporcional à sua colocação geral na população e não à sua nota. Dessa forma diminui-se a discrepância nas chances de escolha entre os melhores e os piores indivíduos tornando as próximas gerações mais diversificadas e com maior chance de chegar a uma solução global para o problema.

Inicialmente, deve-se ordenar a população em uma lista em ordem decrescente por nota, por exemplo: caso uma população tenha 100 indivíduos, a de maior nota estará no índice 0, a de segunda maior nota no índice 1 e assim por diante até de menor nota que estará no índice 99. Feito isso, deve-se sortear um índice pseudo-aleatório de 0 a 99. Este índice tende a ser um número mais próximo de 0 como se pode observar no algoritmo abaixo implementado em Java:

```
return (int) (this.tamPopulacao *
Math.pow(Math.random(), expoenteSorteio));
```

FIGURA 11 - MÉTODO EM JAVA PARA GERAÇÃO DE ÍNDICE PSEUDO-ALEATÓRIO PARA O AG (FUNÇÃO DE SORTEIO)

O método pow da classe Math irá retornar o resultado do primeiro argumento elevado ao segundo argumento. O primeiro argumento (Math.random()) no caso é um número aleatório entre 0 e 1, e o segundo (expoenteSorteio) é uma constante pré-estabelecida que deve ser um número maior que 1 e, segundo o professor Raitzz, preferencialmente menor que 2 para que não diminua demais as chances dos menos adaptados, gerando, conseqüentemente, a falta de diversidade da população. Isso pode ser explicado matematicamente da seguinte forma: Sempre que se eleva um número entre 0 e 1 a uma potencia positiva maior que 1, o resultado tenderá a 0, sendo que, esta tendência é diretamente proporcional ao valor da potência, ou seja, quanto maior esta potência maior a tendência do resultado a 0.

Obviamente, que o resultado da função de sorteio não será um número inteiro na maioria dos casos, por isso é utilizada apenas a porção inteira deste resultado( return (int)). Este valor obtido, então, será utilizado como índice para a escolha de um “pai” na população ordenada anteriormente. Desta forma, as soluções com maiores notas têm mais chances de serem selecionadas, sendo que, o número índice sorteado pelo método tende a ser baixo e a população está ordenada de forma decrescente de acordo com a nota.Com uma solução “pai” selecionada, um novo número pseudo-aleatório é gerado (a partir da mesma função) para a seleção da “mãe”.

#### 3.1.1.2 Processando o AG

Durante o processamento, durante o processamento do algoritmo genético o pesquisador pode visualizar a estrutura e a nota do melhor indivíduo gerado pelo AG até o momento. No nosso caso cada individuo é um gabarito, e no final do processamento será escolhido o melhor gabarito gerado de todas as gerações do AG. O pesquisador poderá também acompanhar o andamento do processo através da barra de tarefas, e botões parar, continuar e reiniciar o processamento. Além de visualizar o gráfico S sem que o processamento seja interrompido.

### 3.1.2 O gráfico S

É o alinhamento em ordem crescente da tradução dos códons (64 possíveis) pelos seus respectivos números atribuídos pelo gabarito. Segundo Raittz o gráfico S ocorre com frequência em se tratando de padrões biológicos. No nosso caso, quanto maior a curvatura do gráfico, melhor o gabarito, ou seja, maior a propensão deste gabarito gerar uma característica diferenciável quando combinada com uma ORF do tipo que deu origem a ele.  Por exemplo, supondo que um gabarito seja processado e gerado tendo uma ORF codificadora de proteína do tipo x como base. Se ele for um bom gabarito, ele tenderá a gerar um número mais alto combinado a uma outra ORF proteína de tipo x do que com outras ORFs de outros tipos quaisquer ou ORFs não codificadoras de proteínas.

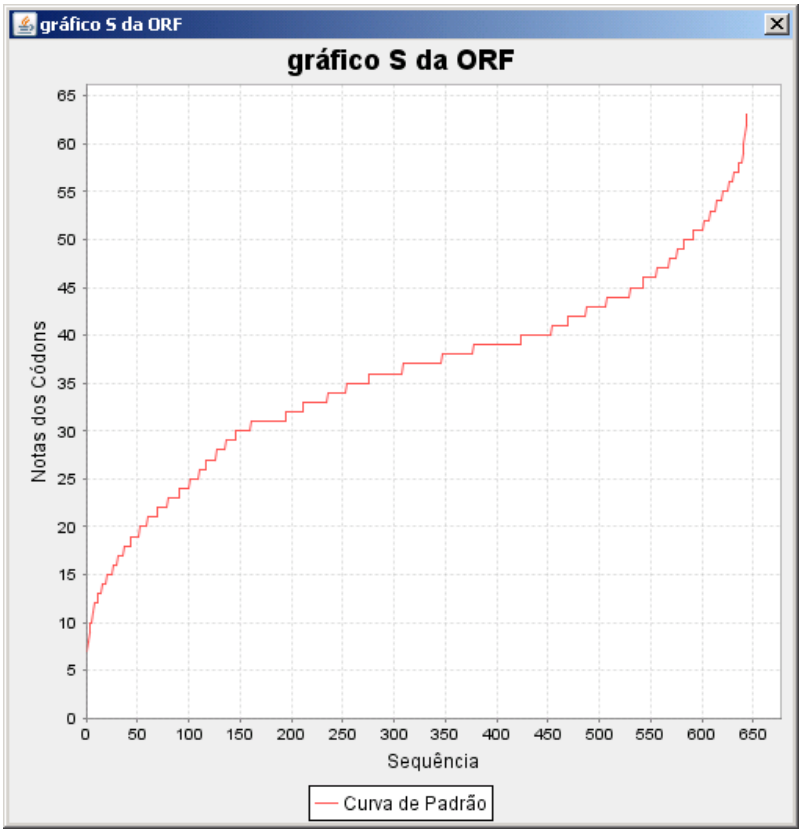



FIGURA 12 - EXEMPLO DE GRÁFICO S

### 3.1.3 Tabela treino

Cada tabela treino é uma relação de todas as características geradas a partir da combinação de gabaritos com ORFs treino. Para que uma rede neural possa ser devidamente treinada no EasyFan, duas tabelas treino devem ser geradas a partir de mesmos gabaritos e ORFs treino distintas. Após serem geradas as tabelas devem ser exportadas para arquivo texto.

### 3.1.4 A rede neural

Após as duas tabelas treino terem sido exportadas pelo J-ORFinder, o EasyFan deve ser iniciado para começar o treinamento da rede neural. Para isto uma das tabelas treino deve ser carregada como conjunto de treinamento e a outra como conjunto de teste no EasyFAN.

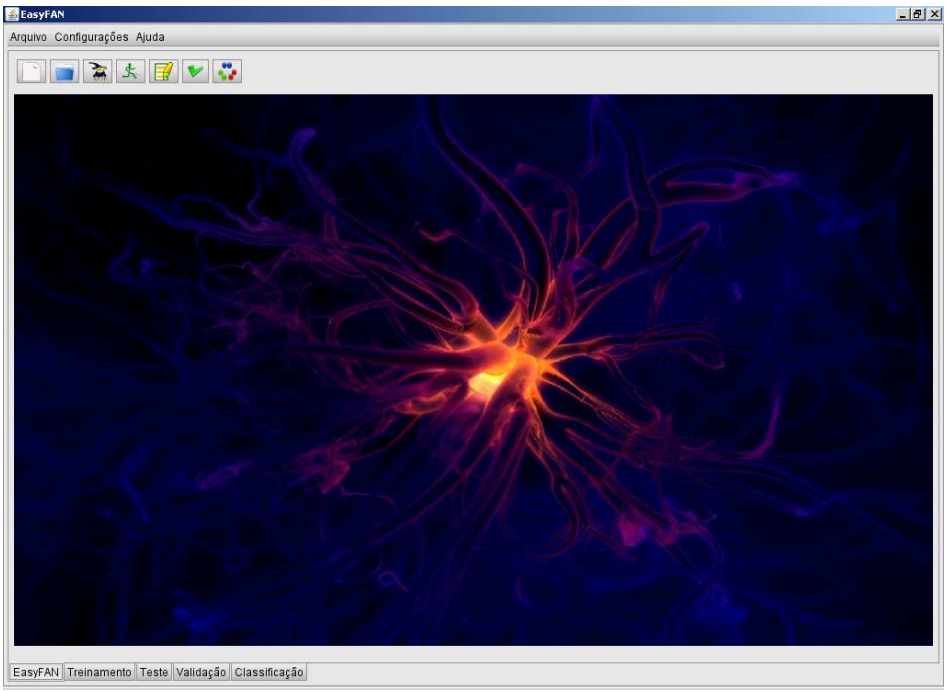



FIGURA 12 - TELA INICIAL DO SISTEMA EASYFAN

Após treinar a rede, durante um determinado período de tempo, o pesquisador deve exporta-la para arquivo. Feito isso à rede exportada deve ser cadastrada no J-ORFinder informando uma das duas tabelas treino que foram utilizadas no seu treinamento. Através da tabela treino informada o J-ORFinder acessa os gabaritos que deram origem a ele e monta um único arquivo XML com a rede neural exportada pelo EasyFan juntamente com os gabaritos que geraram os dados (tabela treino) utilizados no seu treinamento.

## 3.2. MÓDULO WEB

O Módulo WEB é o produto final do projeto. É a ferramenta que ficará disponível aos pesquisadores na internet e que utilizará as redes neurais exportadas pelo módulo desktop no reconhecimento das ORFs.

### 3.2.1Importação das redes neurais

Como será mostrado a seguir, para realizar uma busca por ORFs é necessário que se escolha uma das redes neurais disponíveis no sistema. Estas, por sua vês, são cadastradas e mantidas pelo administrador do sistema, assim sendo, para cadastrar uma nova rede no sistema, basta ter o arquivo xml exportado pelo módulo Desktop e estar logado como administrador no J-ORFinder WEB. Após cadastrada, a rede estará disponível para ser usada por todos os usuários do sistema. Como o sistema não tem razão de existir sem redes neurais cadastradas, ao instalar o sistema uma rede é cadastrada automaticamente e posteriormente o sistema permite que sejam excluídas redes neurais desde que pelo menos uma rede sempre exista em sua base de dados.

### 3.2.2 Funcionamento

O sistema exigirá um cadastro prévio simples para o pesquisador. Após o cadastro, cada usuário poderá utilizar o sistema com o seu login e senha cadastrados. O J-ORFinder WEB foi desenvolvido de modo que os usuários trabalhem com o conceito de projeto e pesquisa, sendo que cada usuário poderá criar vários projetos e cada projeto conter várias pesquisas. A cada nova pesquisa o usuário deverá preencher o campo DNA com o código genético que ele pretende efetuar a busca por ORFs, informar o nome da pesquisa, um tamanho mínimo em códons para as ORFs e escolher dentre as redes neurais disponíveis qual será utilizada na busca.

### 3.2.3 Busca e reconhecimento das ORFs

A partir do momento em que os campos já estão preenchidos o usuário pode clicar no botão Processar. Neste momento, primeiramente o sistema fará a leitura do código DNA 6 vezes: 3 vezes em uma direção e mais 3 vezes na direção inversa que é o código complementar do DNA. Isso ocorre porque os códons possuem 3 bases nitrogenadas e a leitura do código genético pode começar em 3 posições diferentes como mostra a ilustração a baixo.

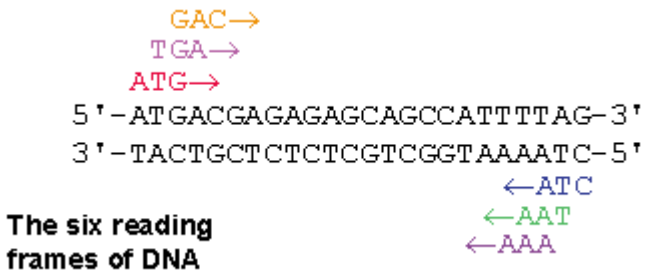



FIGURA 14 - ILUSTRAÇÃO DE LEITURA DO CÓDIGO GENÉTICO (EXTRAÍDA DE HTTP://MEMBERS.COX.NET)

Em cada uma das linhas de leitura, o sistema encontra, através dos códons start e stop, todos os ORFs do DNA submetido. Cada ORF, então, é submetida a um processo de extração de características, onde cada um dos gabaritos relacionados à rede neural escolhida é combinado com a ORF encontrada gerando uma característica (número de dupla precisão). Este conjunto de características, também conhecido por conjunto de classificação, por sua vês, é submetido à rede neural escolhida que irá classificar de acordo com o seu conhecimento a qual classe pertence a ORF. Por exemplo, se a ORF submetida é um gene codificador de proteína ou não, ou se é um gene codificador de um tipo de proteína específico, como é o caso dos ORFs do tipo NIFH. Isto depende do objetivo da rede.

Para a implantação desta parte do projeto, foi utlizada a biblioteca JFAN. Esta biblioteca que é parte essencial do projeto EasyFan, permite que as redes treinadas no EasyFan possam ser utilizadas em outros softwares quaisquer em linguagem Java. A JFAN foi originalmente projetada para ler os conjuntos de dados de classificação a partir de arquivos. Isto foi um problema constatado durante o projeto, já que no J-ORFinder WEB, os dados para classificação extraídos de cada ORF são armazenados em memória e a gravação em arquivo poderia inviabilizar o projeto, devido ao alto custo de processamento gasto nesta tarefa. Por este motivo, nesta parte do projeto foi fundamental a participação do tecnólogo e membro da equipe do EasyFan Filipe Pais Lenfers. Ele desenvolveu e nos disponibilizou uma nova versão da biblioteca JFAN capaz de ler conjuntos de dados de classificação a partir de estruturas Java em memória, o que solucionou o nosso problema.

Após todos ORFs terem sido encontrados e suas características extraídas e submetidas a rede neural, é mostrado ao pesquisador, o resultado de todo o processamento: Todos os ORFs encontrados e a classificação de cada um deles de acordo com a rede neural utilizada. Como pode ser visto na figura a baixo, cada ORF é expresso por uma linha proporcional ao seu tamanho real (Quantidade de códons) e o resultado da rede diferenciado através de cores: Preta para reconhecido e Vermelha para não reconhecido.

A pesquisa fica salva no banco e pode ser acessada quando quiser pelo pesquisador. Ao clicar em cima de qualquer uma das ORFs da pesquisa, o sistema mostrará uma tela de informações sobre a ORF, onde o pesquisador poderá alterar

a classificação da ORF caso não concorde com a classificação da rede neural e também inserir uma observação sobre a ORF. Além destas funcionalidade o pesquisador poderá fazer uma consulta à base de dados do National Center for Biotechnology Information(NCBI), escolhendo alguns parâmetros específicos e clicando no botão procurar.

# 4 ESPECIFICAÇÃO TÉCNICA

## 4.1 DESKTOP

### 4.1.1 CASOS DE USO

Conforme demonstrado no apêndice 1.

## 4.2 WEB

### 4.1.2 CASOS DE USO

Conforme demonstrado no apêndice 2.

## 4.3 BANCO DE DADOS

### 4.3.1Diagrama Entidade-Relacionamento/ Desktop

Conforme demonstrado no apêndice 14.

### 4.3.2 Dicionário de dados/Desktop

Conforme demonstrado no apêndice 16.

### 4.3.3 Diagrama Entidade-Relacionamento/ Web

Conforme demonstrado no apêndice 15.

### 4.3.4 Dicionário de dados/Web

Conforme demonstrado no apêndice 17

## 5 DEPÊNDENCIA DO SISTEMA

Para o seu funcionamento o J-ORFinder Deskop depende do Java 6.0 ou superior e também do EasyFAN. Já o modulo Web depende do container web tomcat 5.5 ou superior, banco de dados postgres versão 8.2 e também de um browser.

# 6 IMPLEMENTAÇÕES FUTURAS

- Buscar uma integração maior com o NCBI e outros bancos de dados mundiais de proteínas;
- Suporte a um número maior de formatos no momento da visualização da consulta;
- Internacionalização da aplicação;
- Modo administrativo para visualizar informações sobre percentuais de acertos das pesquisas em andamento;
- Portar a aplicação para trabalhar em cluster e suportar grandes volumes de dados;
- Integrar o J-ORFinder com outras ferramentas utilizadas pelos biólogos;
- Mostrar informações gerenciais aos usuários do sistema sobre o andamento de um determinado projeto ou de uma pesquisa especifica.

# 7 RESULTADOS E CONCLUSÕES

## 7.1 COMPARAÇÕES DE GABARITOS

Quando terminamos a implementação do nosso algoritmo genético não estávamos certos se ele estava efetivamente gerando bons gabaritos ou não. Geralmente algoritmos genéticos trabalham em determinados problemas para gerar soluções que são facilmente classificáveis a olho nu como boas ou ruins. É o caso de problemas de calculo de rotas, posições geográficas, etc. Este claramente não era o nosso caso. Como poderíamos então avaliar um simples conjunto de 64 números inteiros?

Dois anos antes de orientar o nosso projeto o professor Dr. Roberto Tadeu Raittz havia desenvolvido um algoritmo genético em linguagem C para extração de características de ORFs, ou seja, geração de gabaritos. Raittz havia obtido bons resultados com os gabaritos gerados por este algoritmo e nos passou o código fonte para nos basearmos.

Como o nosso AG foi inteiro escrito do zero em Java, a única forma que tínhamos de ter certeza que ele estava funcionando corretamente era implementar no J-ORFinder o mesmo método de avaliação desenvolvido por Raitzz para que as notas e os gráficos dos melhores gabaritos pudessem ser comparados diretamente. Ao contrário do que se possa imaginar, encontrar a função de avaliação dentre as mais de 700 linhas de código em C do professor não foi tarefa fácil, mas felizmente nós encontramos e implementamos no nosso projeto e pudemos, assim, comprovar que o AG estava funcionando corretamente como pode ser observado no experimento.

### 7.1.1 O experimento

Para que os gabaritos pudessem ser comparados os parâmetros de processamento do algoritmo genético tiveram que ser exatamente os mesmos:

Elitismo: 1%

Mutação: 15%

População: 100

Gerações: 2000

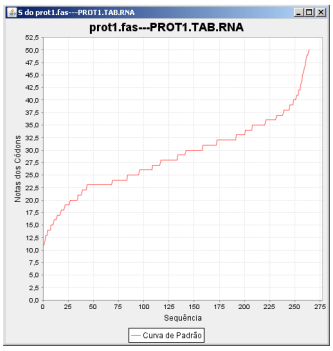
Para a geração dos gabaritos foram utilizadas nove ORFs codificadoras de proteínas. Elas foram disponibilizadas pelo doutorando e mestre em biologia Giovani Pisa que nos ajudou durante o desenvolvimento do projeto disponibilizando dados e nos explicando conceitos específicos de biologia.

Segue abaixo uma relação de comparações entre os gabaritos gerados pelos dois algoritmos que mostra as notas e gráficos S de cada gabarito.

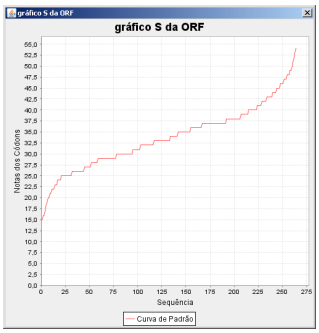
**Algoritmo Raittz** | **J-ORFinder**

**Proteína 1**

5.785592

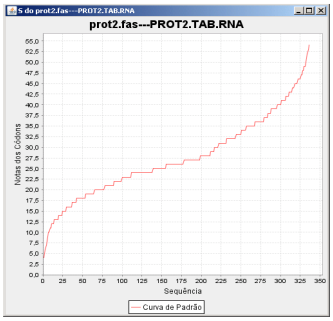



5.68966

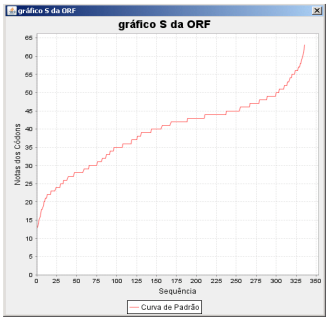



**Proteína 2**

4.873809

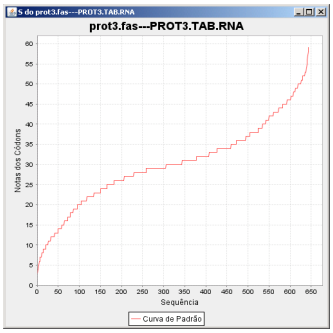



4.711865




**Proteína 3**

4.12501




4.0904

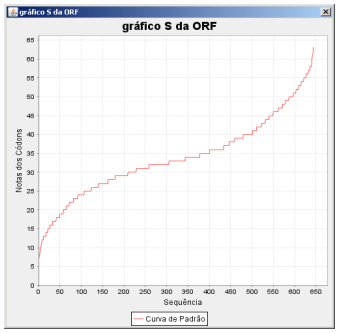



FIGURA 15 – FIGURA COMPARAÇÃO DOS GABARITOS GERADOS A PARTIR DAS PROTEÍNAS 1, 2 E 3

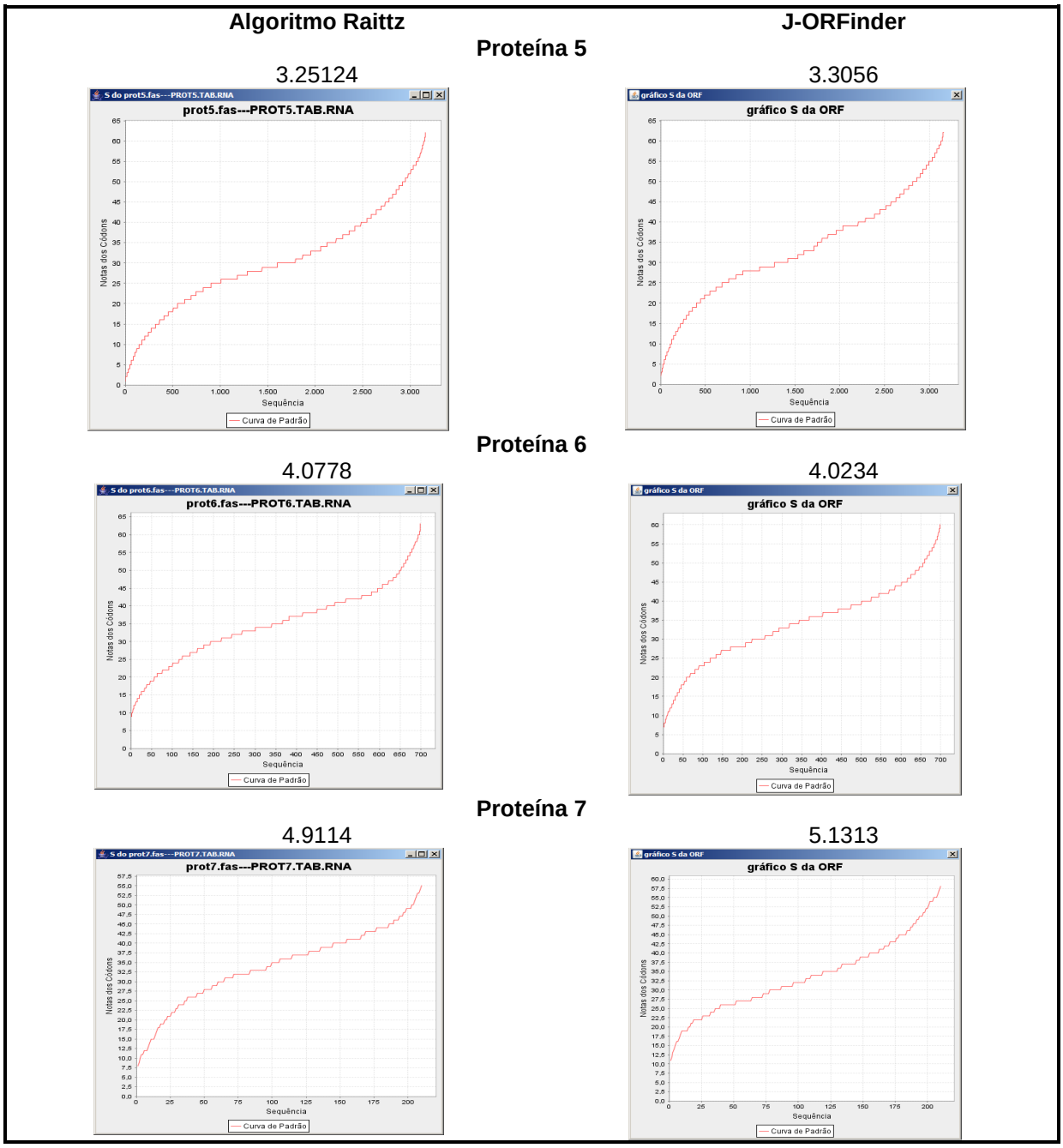



FIGURA 16 – FIGURA COMPARAÇÃO DOS GABARITOS GERADOS A PARTIR DAS PROTEÍNAS 5, 6 E 7

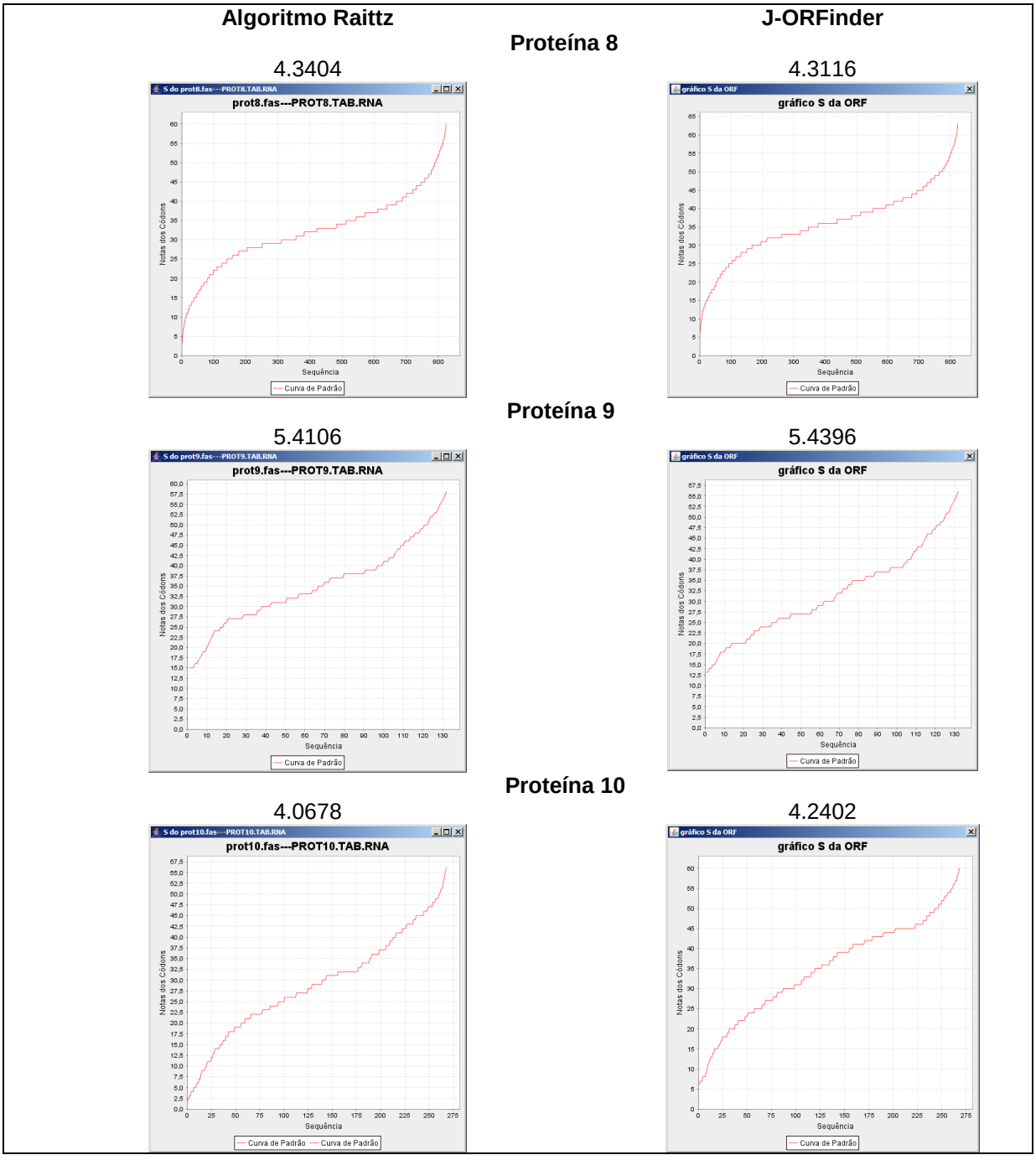



FIGURA 17 – FIGURA COMPARAÇÃO DOS GABARITOS GERADOS A PARTIR DAS PROTEÍNAS 8, 9 E 10

| Proteínas | Algoritmo Raittz | JORFinder |
|---|---|---|
| Proteína 1 | 5,7856 | 5,6897 |
| Proteína 2 | 4,8738 | 4,7119 |
| Proteína 3 | 4,1250 | 4,0904 |
| Proteína 5 | 3,2512 | 3,3056 |
| Proteína 6 | 4,0778 | 4,0234 |
| Proteína 7 | 4,9114 | 5,1313 |
| Proteína 8 | 4,3404 | 4,3116 |
| Proteína 9 | 5,4106 | 5,4396 |
| Proteína 10 | 4,0678 | 4,2402 |
| Total | 40,843 | 40,9436 |

QUADRO 1 - NOTAS DOS GABARITOS GERADOS NO EXPERIMENTO

Algoritmos genéticos, assim como outras técnicas de computação evolutiva, não são exatos. Um mesmo AG processado duas vezes com os mesmos parâmetros de entrada, tempo de processamento e parâmetros de processamento sempre irá gerar resultados diferentes, apesar de muito semelhantes os resultados quase nunca serão iguais. Ao alcançar uma variação quase insignificante de 40,8437 para 40,9436 e ao analisarmos os gráficos gerados a partir dos gabaritos de cada um dos sistemas, pudemos concluir que o nosso AG estava funcionando corretamente e dar sequência ao projeto.

## 7.2 TREINAMENTO DAS PRIMEIRAS REDES

Logo que completamos a parte de IA do módulo Desktop fizemos algumas experiências com o treinamento de redes neurais no EasyFAN a partir de dados (tabelas treino) gerados pelo J-ORFinder. As primeiras experiências foram feitas a partir de algumas ORFs reais codificadoras de proteínas e ORFs não reais, ou seja, criadas aleatoriamente por um programa de computador. O objetivo era saber se a rede neural seria capaz de distinguir entre um código genético extraído de um organismo real e um código aleatório gerado por computador. Os resultados foram animadores: Com aproximadamente 15 ORFs reais de organismos diferentes e mais de 130 ORFs aleatórias, a rede rapidamente convergiu para 100% de acerto. Estando treinada precisávamos testá-la baseada em novos dados. Para isso criamos uma nova tabela treino no J-ORFinder gerada a partir de sete novas proteínas e mais 60 ORFs aleatórias e mais uma vez, o resultado não foi diferente, 100% de acerto. Restava saber se uma rede neural devidamente treinada seria capaz de distinguir entre ORFs reais codificadoras e não codificadoras de proteínas e também entre ORFs reais não codificadoras e ORFs aleatórias.

### 7.2.1 Proteínas x Não proteínas x Aleatórias

Para realizar este experimento, geramos duas tabelas treino (conjunto de dados) no J-ORFinder baseada em ORFs de três tipos diferentes: ORFs reais codificadoras de proteínas, ORFs reais não codificadoras de proteínas e aleatórias.

As tabelas treino foram geradas a partir de ORFs treino distintas, ou seja, as ORFs que foram utilizadas em uma não foram utilizadas na outra. Além disso, procurou-se ter quantidades semelhantes de cada um dos tipos de ORFs em cada uma das tabelas. Lembrando que os mesmo gabaritos devem ser utilizados ao gerar dois conjuntos para o treinamento de uma única rede. Estes gabaritos foram gerados a partir de nove proteínas diferentes (ORFs base).

A rede neural então foi submetida ao treinamento durante 10 minutos tendo os dados abaixo como entrada:

| Classe | Tipo de ORF | Conjunto de treinamento | Conjunto de teste |
|---|---|---|---|
| 1 | proteínas | 1081 | 1080 |
| 2 | não proteínas | 784 | 783 |
| 3 | aleatórias | 1044 | 1043 |
| Total | | 2909 | 2906 |

QUADRO 2- COMPOSIÇÃO DAS TABELAS TREINO UTILIZADAS NO EXPERIMENTO

Após os 10 minutos de treinamento pudemos verificar a Matriz de confusão da rede neural. Onde são mostrados os acertos e erros da rede de acordo com as classes dos dados de entrada.

| Matriz de confusão | | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 1043 | 26 | 12 |
| 2 | 34 | 558 | 190 |
| 3 | 5 | 82 | 954 |
| Acertos | 2555 | | |
| Erros | 349 | | |
| Total (%) | 87,9820 | | |

QUADRO 3- MATRIZ DE CONFUSÃO OBTIDA COM A REDE

Quanto maior o número de classes utilizadas no treinamento de uma rede neural, maior as chances de erro. Por este motivo e tendo em mente que as redes neurais que serão exportadas para o J-ORFinder WEB trabalharão apenas com duas classes, os resultados obtidos foram muito bons e as perspectivas melhores ainda, já que a quantidade de dados utilizada foi limitada, assim como o tempo de

processamento do experimento. Fatores estes que influem diretamente na performance da rede.

### 7.2.2 Proteínas x Não Proteínas

Para testar o discernimento entre estas duas classes, fizemos o treinamento de uma nova rede neural: Basicamente, repetimos o experimento com os mesmos dados e parâmetros de processamento do experimento anterior, tirando apenas das duas tabelas os dados referentes às ORFs aleatórias. Segue a baixo o resultado:

| Matriz de confusão | | |
|---|---|---|
| | 1 | 2 |
| 1 | 1062 | 19 |
| 2 | 78 | 704 |
| Acertos | 1786 | |
| Erros | 77 | |
| Total (%) | 95,8668 | |

QUADRO 4- MATRIZ DE CONFUSÃO OBTIDA COM A REDE

Como pode se observar, a rede teve mais facilidade em identificar proteínas do que não proteínas. Das 1081 proteínas o acerto girou em torno de 98,24% e das não proteínas, em torno de 90%.

# 8 CONCLUSÕES

Projetos em áreas multidisciplinares como a bioinformática demandam de muito mais estudo e dedicação, assim como projetos que implementam técnicas de inteligência artificial, onde a exatidão da informática se transforma em subjetividade.

Juntar estas características num único projeto foi um verdadeiro desafio. Durante o seu desenvolvimento tivemos que pesquisar a fundo diversas técnicas de áreas aparentemente antagônicas: Começando pelos *Design Patterns* da Engenharia de Software, modelagem de Dados utilizando UML, passando pelos métodos de cruzamento da Computação evolutiva até as Bases Nitrogenadas da Biologia.

O software foi desenvolvido de forma a utilizar o conhecimento de redes neurais previamente treinadas pelos pesquisadores. E o conhecimento das redes neurais é proveniente de dados escolhidos por especialistas na área. Por isso o conhecimento do pesquisador administrador do sistema é imprescindível para que o sistema funcione corretamente. Pois será ele o responsável por escolher os genes que irão gerar os dados para o treinamento das redes neurais.

Após ter sido concluída a implementação do software, depois de um ano de projeto, os resultados que obtivemos nos surpreenderam, não esperávamos atingir um índice tão alto de acerto. Por isso, acreditamos que o software foi concluído com sucesso e será de grande valia aos pesquisadores que virão a utilizá-lo.

# REFERÊNCIAS


ALUR, D. **Core J2ee Patterns.** New York: Elsevier, 2004.

APACHE. **Apache**. Disponível em: <http://commons.apache.org/>Acesso em: 07/11/2007.

APACHE TOMCAT. **Apache tomcat**. Disponível em: <http://tomcat.apache.org/> Acesso em: 07/11/2007.

BIOJAVA. **Framework biojava**. Disponível em: <http://www.biojava.org/wiki/Main_Page > Acesso em: 06/11/2007.

BITTENCOURT, G. **Inteligência artificial:** ferramentas e teorias. 3. ed. Florianópolis: Editora da UFSC, 2006

CERQUEIRA, E. O; ANDRADE ,J; POPPI R. **Redes neurais e suas aplicações em calibração cultivalorada.** disponível em <http://www.scielo.br/scielo.php?pid=S0100 >. Acesso em: 23/09/2007

CORE J2EE PATTERNS. **Data Access object**. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html> Acesso em: 06/11/2007.

CORE J2EE PATTERNS. **Front controller**. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/FrontController.html> Acesso em: 06/11/2007.

CORE J2EE PATTERNS. **Intercepting filter**. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/InterceptingFilter.htmll> Acesso em: 06/11/2007.

CORE J2EE PATTERNS. **Transfer object**. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/TransferObject.html> Acesso em: 06/11/2007.

DARWIN, C. **On the rigin of species by means of natural selection, or the preservation of favoured races in the struggle for life**. London: 1859.

DEITEL, H. ; DEITEL. P. **Java:** como programar**.** 6. ed**.** São Paulo: Pearson, 2005.

EXON. In: WIKIPEDIA a enciclopédia livre. Disponível em < http://pt.wikipedia.org/wiki/Ex%C3%A3o> . Acesso em: 05/12/2007.

FURLAN, J. DAVI. **Modelam de objetos através da UML:**unifield modeling language**.** São Paulo: Makron Books, 1998.

GARRET, L. F. V.; IGNACIO, F. A.; KUSTER, C.W.; LENFERS, F. P.; ZOTTO S. P. **EASYFAN.** 165 f. Trabalho de Conclusão de Curso de Tecnologia em Informática(Disciplina de Projetos) – Curso de Tecnologia em Informática, Setor da Escola Técnica, Universidade Federal do Paraná, Curitiba, 2006.

GENOMA. ***Projeto genoma***. Disponível em: <http://www.comciencia.br/reportagens/genoma/genoma1.htm> Acesso em: 01/12/2007.

INTRON. In: WIKIPEDIA a enciclopédia livre. Disponível em <http://pt.wikipedia.org/wiki/Especial:Search?search=INTRONS&fulltext=Pesquisa> . Acesso em: 05/12/2007.

MIRANDA, M.N. Algoritmos Genéticos: fundamentos e Aplicações**. GTA - UFRJ – Grupo de teleinformática de automação.** Disponível em: <http://www.gta.ufrj.br/~marcio/genetic.html>. Acesso em: 24/09/2007

PREDIÇÃO. In: DICIONARIO universal: língua portuguesa. Disponível em <http://www.priberam.pt/dlpo> . Acesso em: 05/12/2007.

PRESSMAN, R. S. **Software Engineering:** a practitioner's approach**.** 6. ed. New York: Pressman and Associates, 2005.

RAITTZ, R. T. **FAN 2002**:um modelo neuro-fuzzy para reconhecimento de padrões. 85 f. Tese(Doutorado em Engenharia da Produção) – Programa de Pós-graduação em Engenharia da Produção, Universidade Federal de Santa Catarina, 2002. T

REVISTA PESQUISA. **Do câncer ao genoma humano**. Disponível em: < http://www.revistapesquisa.fapesp.br/?art=537&bd=1&pg=2&lg=> Acesso em: 01/12/ 2007.

SILVA, E. S. R. L. **Investigação do comportamento dinâmico e avaliação de estratégias de identificação**, **controle e otimização de um reator FCC**.157 f. Tese (Doutorado em Engenharia de Materiais e Metalúrgica) - ENGENHARIA DE MATERIAIS E METALÚRGICA , Universidade Federal do Paraná, 2006.

SUN. **Java technology**. Disponível em: <http://java.sun.com/>Acesso em: 08/11/2007.

UNIVERSIDADE FEDERAL DO PARANÁ. Sistemas de Bibliotecas. **Citações e notas de rodapé.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos cientifico, 3).

UNIVERSIDADE FEDERAL DO PARANÁ. Sistemas de Bibliotecas. **Redação e editoração.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos cientifico, 9).

UNIVERSIDADE FEDERAL DO PARANÁ. Sistemas de Bibliotecas. **Referências.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos cientifico, 4).

UNIVERSIDADE FEDERAL DO PARANÁ. Sistemas de Bibliotecas. **Teses,**

**dissertações, monografias e outros trabalhos acadêmicos.** Curitiba: Editora UFPR, 2007. (Normas para apresentação de documentos cientifico, 2).

W3C. **Cascading style sheets home page**. Disponível em: <http://www.w3.org/Style/CSS/> Acesso em: 07/11/2007.

W3C. **Extensible markup language (XML)**. Disponível em: <http://www.w3.org/XML/>
Acesso em: 07/11/2007.

W3C. **HTML**. Disponível em: <http://www.w3.org/Style/CSS/> Acesso em: 07/11/2007.

WIKIPEDIA. **Wikipedia a enciclopédia livre**. Disponível em: <http://pt.wikipedia.org/wiki>. Acesso em: 08/11/2007.

# GLOSSÁRIO

**ARQUIVO FASTA**: Formato de arquivo padronizado (.fas) com objetivo de armazenar código genético. Dividido em duas partes: Cabeçalho e conteúdo.

**API**: Application Programming Interface (ou interface de programação de aplicativos) é um conjunto de rotinas e padrões estabelecidos por um software para utilização de suas funcionalidades por programas aplicativos. Programas que não querem envolver-se em detalhes da implementação do software, apenas fazer utilização de seus serviços.

**Biologia Molecular**: Estudo da Biologia em nível molecular, com especial foco no estudo da estrutura e função do material genético e seus produtos de expressão, as proteínas.

**BASE NITROGENADA**: É um composto cíclico contendo nitrogênio.

**BIO-INFORMÁTICA**: Área multidisciplinar que combina conhecimentos de química, física, biologia, engenharia genética  e ciência da computação para processar dados biológicos

***CLUSTER***: Um cluster, ou aglomerado de computadores, é formado por um conjunto de computadores. É construído muitas vezes a partir de computadores convencionais, sendo que estes vários computadores são ligados em rede e comunicam-se através do sistema de forma que trabalham como se fosse uma única máquina de grande porte.

**CÓDON**: Conjunto de 3 bases nitrogenadas.

**CÓDON START E STOP**: Tipos específicos de códons que determinam o início e o fim de cada ORF codificador de proteína.

***COMMIT****:* Ato de confirmar uma transação que foi iniciada, seja ela para um banco de dados ou para um servidor de controle de versão de código.

**COMPILADOR**: Programa que, a partir de um código escrito em uma linguagem, o código fonte, cria um programa semânticamente equivalente porém escrito em outra linguagem, código objeto

***CONTAINER WEB***: É um servidor que implementa as especificações da arquitetura J2EE, possibilitando o controle de segurança, transações, ciclo de vida de uma classe entre outras funções.

***DEBUG*** : Programa ou componente de um programa que auxilia o desenvolvedor a encontrar erros de programação em seu código ou em programas desenvolvidos por terceiros.

***DDBJ***: Centro de Informação Biológica e banco de dados de DNA do Japão.

***DESIGN*** : Esforço criativo relacionado à configuração, concepção, elaboração e definição de algo, como um objeto ou uma imagem, em geral voltado a uma determinada função. De forma ampla, o termo design se refere à concepção de uma solução prévia para um problema, em uma acepção mais específica, à profissão da pessoa que projeta. O profissional que trabalha na área de design é chamado de

designer, visto que a palavra pertence à língua inglesa e normalmente não se traduz.

**DNA**: Material genético das células. Constituído de duas fitas complementares de bases nitrogenadas.

**DNA COMPLEMENTAR**: moléculas de DNA (cDNA) obtidas a partir da transcrição reversa de moléculas de RNA-mensageiro.

**DUPLA PRECISÃO**: tipo de dado que ocupa 8 Bytes em memória e pode ser representado por até 15 dígitos.

**EMBL**: Laboratório Europeu de Biologia Molecular (acrónimo de European Molecular Biology Laboratory).

***FRAMEWORK*** : Estrutura de suporte definida em que um projeto de software pode ser organizado e desenvolvido. Um framework pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e unir diferentes componentes de um projeto de software.

***Genbank***: Banco de dados público de sequencias genéticas.

***Gigabase:*** Unidade de medida para tamanho de bases do NCBI. Cada Gigabase significa um bilhão de bases nitrogenadas

**GENOMA**: Completo código genético de um determinado indivíduo.

**HIPERTEXTO**: Texto que inclui links ou atalhos para outros documentos, permitindo ao leitor pular facilmente de um texto para outro, relacionados, de forma não-linear. O termo foi criado por Ted Nelson, em 1965, para descrever documentos que expressam estruturas de idéias não-lineares, em oposição ao formato linear dos filmes, dos livros e da fala.

**IDE**:Integrated Development Environment é um ambiente integrado para desenvolvimento de software.

**INDIVÍDUOS (AGS)**: Mais conhecidos como cromossomos. Consistem em uma estrutura de dados considerada como uma das possíveis soluções do problema ao qual o AG se propõem a resolver. O nome indivíduo foi adotado para evitar confusões com os demais termos Biológicos existentes no trabalho.

**JAVA**: Linguagem de programação universal criada pela Sun Microsystems para o desenvolvimento de aplicações. As aplicações Java podem ser executadas tanto em uma estação isolada como em estações distribuídas entre servidores e clientes de uma rede. É chamada universal por ser uma linguagem multiplataforma, que pode ser entendida e processada por máquinas que rodam diferentes sistemas operacionais, desde o Windows até os vários tipos de Unix.

***LOGGING**:* Registro de determinadas informações, as quais podem ser definidas, quando ocorre uma ação em alguma parte da aplicação.

**MOLÉCULA**: Uma molécula é um conjunto eletricamente neutro de dois ou mais átomos unidos por pares compartilhados de elétrons (ligações covalentes) que se comportam como uma única partícula.

**NUCLEOTÍDEO**: São compostos ricos em energia e que auxiliam os processos metabólicos, principalmente as biossínteses, na maioria das células. Funcionam ainda como sinais químicos, respondendo assim a hormônios e outros estímulos extracelulares.

***OPEN-SOURCE**:* Código aberto, tipo de software cujo código fonte é visível publicamente. Respeita as quatro liberdades definidas pela Free Software Foundation, é advogado pela Iniciativa do Código Aberto (Open Source Initiative).

**ORF**: Trecho de código considerado passível de ser um gene codificador de proteína. É delimitado por um códon start e um códon stop.

**ORFs NIFH**: Tipo específico de gene. Presente em seres com a capacidade de fixar nitrogênio.

**POLÍMERO**: Os polímeros são compostos químicos de elevada massa molecular relativa, resultantes de reações químicas de polimerização. Estes contêm os mesmos elementos nas mesmas proporções relativas, mas em maior quantidade.

**PROJETO JAKARTA**: Responsável pela criação e manutenção de vários softwares livres para a plataforma Java. É o "guarda-chuva" de vários outros projetos sob o controle da Apache Software Foundation. Todos os "produtos" Jakarta são licenciados através da licença Apache.

**RNA-MENSAGEIRO**: Responsável por carregar informações do DNA para o ribossomo, local onde ocorre a síntese protéica.

**SDK**: Software Development Kit, ou kit de desenvolvimento de software. Normalmente os SDK's são disponibilizados por empresas ou projetos open-source para que programadores externos tenham uma melhor integração com o software proposto.

***SERVLET**:* Tecnologia que insere novos recursos a um servidor

***SOFTWARE**:* É um programa de computador, uma seqüência de instruções a serem seguidas e/ou executadas na manipulação, redirecionamento ou modificação de um dado/informação ou acontecimento. É também o nome dado ao comportamento exibido por essa seqüência de instruções quando executada em um computador ou máquina semelhante. Tecnicamente, é o nome dado ao conjunto de produtos desenvolvidos durante o processo de software, o que inclui não só o programa de computador propriamente dito, mas também manuais, especificações e planos de teste.

***SPLICE JUNCTIONS**:* São pontos da seqüência de DNA aonde o DNA "supérfluo" é removido, durante o processo de síntese da proteína.

***STORE PROCEDURED***: Conjunto de comandos SQL que podem ser armazenados no servidor. Uma vez que isto tenha sido feito, os clientes não precisam reenviar os comandos individuais mas, pode fazer referência às *stored procedures*.

***TRIGGER***: Objeto que fica no banco de dados e está associado com uma tabela, sendo que a sua ativação ocorre no momento em que um determinado evento acontece na tabela.

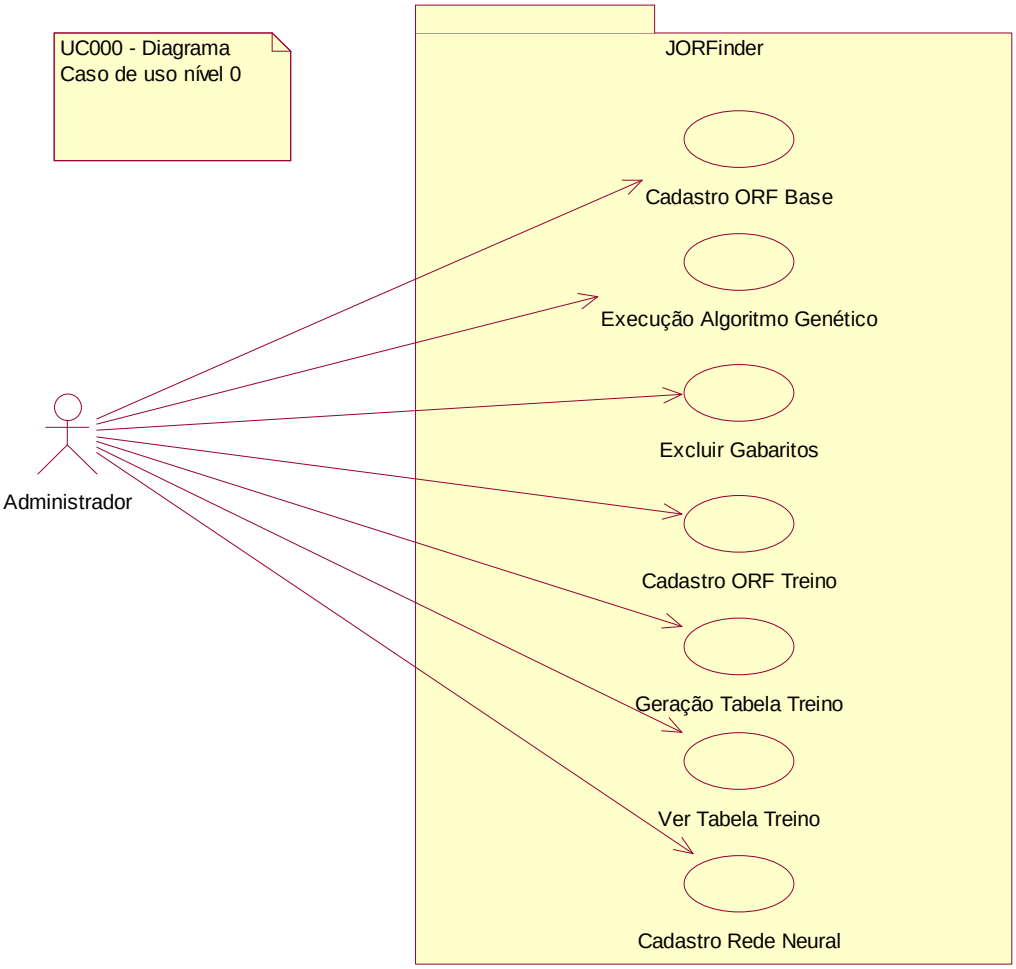
***UPLOAD*** : Envio de arquivos para um servidor.

**WEBDAV**: Acrônimo de Web-based Distributed Authoring and Versioning, ou Criação e Distribuição de Conteúdo pela Web. É um protocolo de transferência de arquivos que suporta bloqueio de recursos. Quando uma pessoa está editando um arquivo, ele fica bloqueado, impedindo que outras pessoas façam alterações ao mesmo tempo.

**WEB DESIGNER**: Profissional competente para a elaboração do projeto estético e funcional de um website. Tende à multidisciplinaridade, uma vez que a construção de páginas Web requer subsídios de diversas áreas técnicas, além do design propriamente dito.

# APÊNDICES

## APÊNDICE 1 – DIAGRAMA DE CASOS DE USO MÓDULO DESKTOP



**ESPECIFICAÇÃO DOS CASOS DE USO**

MÓDULO DESKTOP

UC001 - Cadastro ORF base

Descrição:

Este caso de uso descreve como um arquivo no padrão FASTA que contém uma ORF conhecida e classificada como codificadora de proteína é cadastrada no sistema.

Pré- requisito:

1- Existir um arquivo no padrão FASTA, contendo a ORF classificada e o sistema estar disponível.

Pós-condições:

1- O sistema deve ter cadastrado a ORF e gravado o registro no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica no botão ORF base.

2 - O sistema abre a janela ORF base.

3 - O usuário clica no botão cadastrar.

4 - O sistema abra a janela cadastrar ORF base. (A1)

5 - O usuário clica no botão arquivo.

6 - O sistema abra a janela de carregamento de arquivo.

7 - O usuário escolhe o arquivo a ser carregado e clica em carregar. (E1) (E2)

8 - O usuário edita o campo nome. (E4)

9- O usuário edita o campo observação(E4)

10- O usuário escolhe a classe. (E4)

11- O usuário edita o campo ORF. (E4)

12 - O usuário clica no botão cadastrar. (E3) (E5)

13 - O sistema grava os dados no banco de dados.

14 - O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “O arquivo não pode ser carregado!”

E2 - O sistema apresenta a mensagem: ”O arquivo não está no padrão FASTA!”.

E3 - O sistema apresenta a mensagem: ”O arquivo não pode ser cadastrado!”.

E4 – O campo não informado, o sistema emite a mensagem “O campo está em branco”

E5 – No Banco de Dados um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe uma ORF deste tipo com este nome no banco”

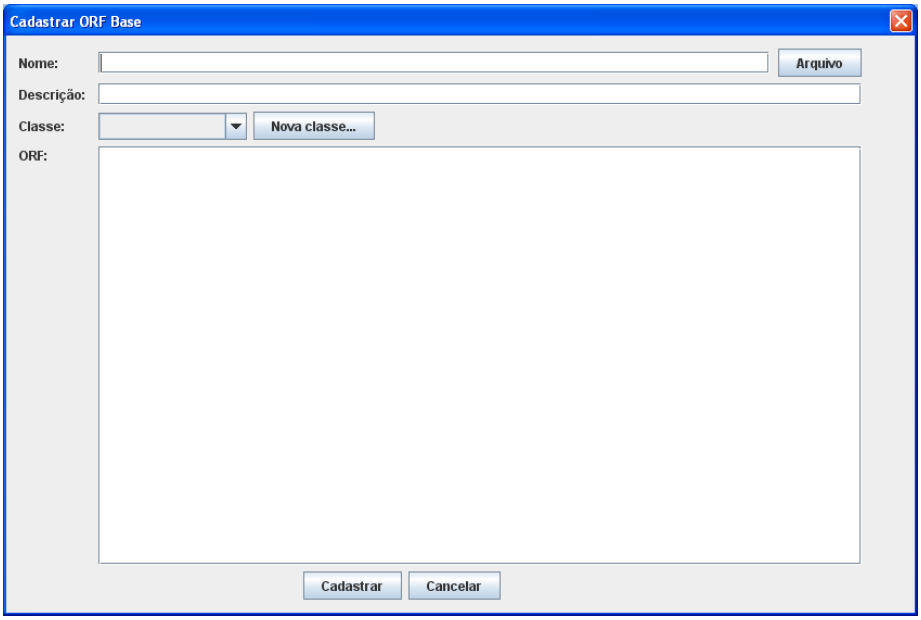



Diagrama de Seqüência: Cadastro ORF Base

Conforme demonstrado no apêndice 8.

UC002 - Editar ORF base

Descrição:

Este caso de uso descreve como um arquivo no padrão FASTA que contém uma ORF conhecida e classificada como codificadora de proteína já cadastrado no sistema é editado.

Pré- requisito:

1- Existir o arquivo no padrão FASTA contendo a ORF classificada no cadastro do sistema.

Pós-condições:

1- O sistema deve ter gravado o arquivo editado  no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica no botão ORF base.

2 - O sistema abre a janela ORF base.

3 - O usuário escolhe a ORF base para edição entre as OFR disponíveis.

4 - O usuário clica no botão editar.

5 - O sistema abre a janela Adicionar ORF base. (A1)

6 - O usuário edita o campo nome. (E2)

7-  O usuário edita o campo observação.  (E2)

8 - O usuário escolhe a classe. (E2)

9 - O usuário edita o campo ORF. (E2)

10- O usuário clica no botão salvar. (A1)(E3)

11- O sistema grava o arquivo editado no banco de dados.

12- O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

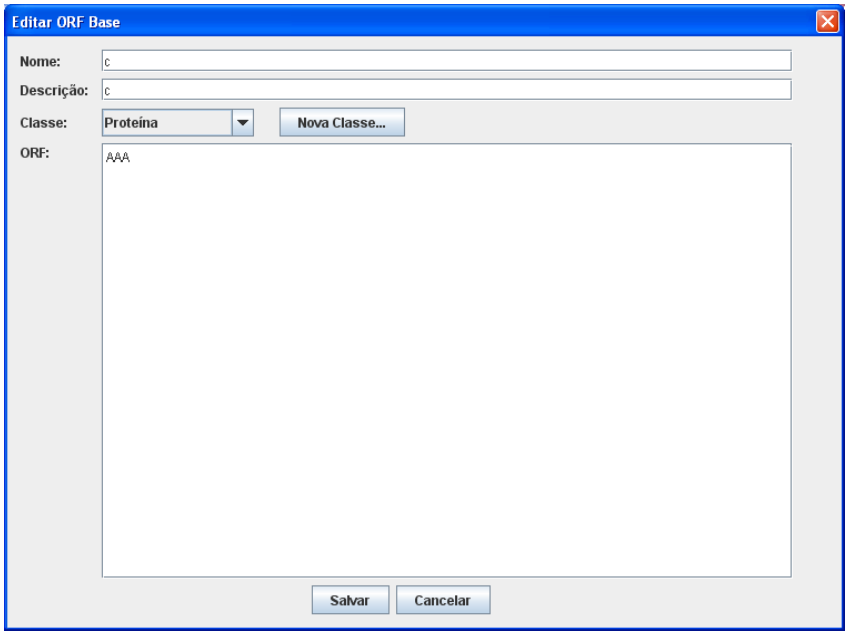
Fluxo de exceção:

E1 - O sistema apresenta a mensagem “O arquivo não pode ser salvo!”

E2 – Campo não foi informado, o sistema emite a mensagem “O campo está em branco”

E3 – No Banco de Dados um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe uma ORF deste tipo com este nome no banco”




UC004 - Excluir ORF base

Descrição:

Este caso de uso descreve como excluir uma ORF Base previamente cadastrada.

Pré- requisito:

1- Existir uma ORF Base cadastrada no sistema

Pós-condições:

1- O sistema deve ter excluído a ORF  no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica no botão ORF base.

2 - O sistema abre a janela ORF base. (A1)

3 - O usuário escolhe a ORF base para Exclusão entre as ORFs disponíveis.

4 - O usuário clica no botão excluir. (E1)

5 - O sistema apresenta a mensagem “Tem certeza que deseja excluir esta ORF Base?”

6- O sistema exclui a ORF no banco de dados.

7- O caso de uso é finalizado.

Fluxo alternativo:

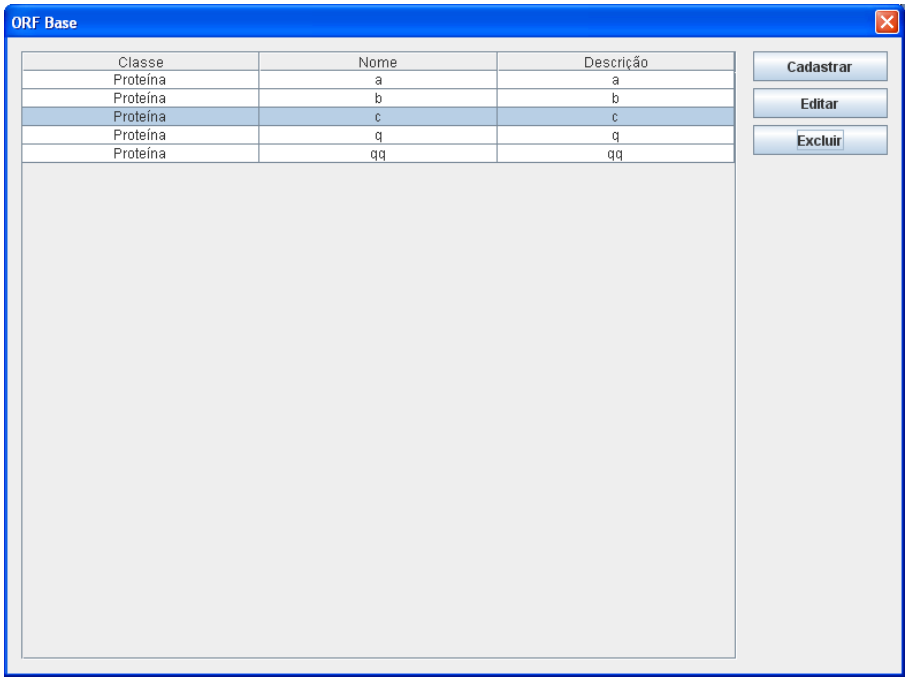
A1 - O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1- Existe um Gabarito que use a ORF que está sendo excluída

1- O sistema apresenta a mensagem : “Não é possível excluir essa ORF existem gabaritos que dependem dela ”




UC005 - Execução Algoritmo Genético

Descrição:

Este caso de uso descreve quais passos são necessários para executar o Algoritmo Genético

Pré- requisito:

1- Existir pelo menos uma  ORF Base  cadastrada (UC001).

Pós-condições:

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica o botão Algoritmo Genético.

2 - O sistema abre a janela  Algoritmo Genético. (A2)

3 - O usuário clica no botão adicionar. (A1)

4 - O sistema abre a tela Carregar ORF Base. (A2)

5 - O usuário escolhe a ORF base que ser processada pelo Algoritmo Genético.

6 - O usuário clica no botão carregar.

7-  O usuário preenche o campo gerações. (E1)

8- O usuário preenche o campo mutação. (E1)

9- O usuário preenche o campo elitismo.  (E1)

10- O usuário preenche o campo tamanho da população. (E1)

11- O usuário clica no  botão processar.

12 - O sistema abre a tela processando. Chama UC006

14 - O caso de uso é finalizado.

Fluxo alternativo:

A1 -  O usuário clica no botão excluir.

1-  O sistema exclui a ORF selecionada

.

A2  -  O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 –Campo não for informado o sistema, emite a mensagem “O campo está em branco”

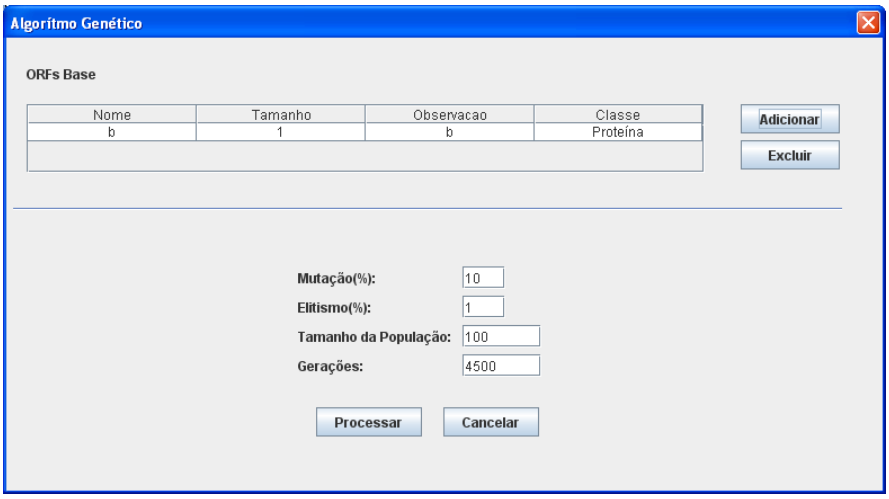



Diagrama de Seqüência: Execução Algoritmo Genético

Conforme demonstrado no apêndice 7.

UC006 - Processar Algoritmo Genético

Descrição:

Este caso de uso descreve quais passos são necessários para processar o Algoritmo Genético

Pré- requisito:

1- O botão processar da tela de execução do Algoritmo Genético ter sido clicado (UC005).

Pós-condições:

1- O sistema deve ter produzido um gabarito com base no processamento do Algoritmo Genético.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica o botão Iniciar Processamento.

2 - O sistema começa o processamento do Algoritmo Genético. (A1) (A2) (A3)

4- O usuário clica em Salvar Processamento(A4)

5- O usuário preenche o nome do gabarito (E1) (E2)

6- O usuário preenche a descrição do gabarito (E1)

3 - O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica no botão sair.

1- O caso de uso é finalizado.

A2 - O usuário clica no botão  iniciar/ parar.

1- Se o Algoritmo Genético já estiver iniciado o processamento, ele é interrompido senão ele é iniciado.

A3 - O usuário clica no botão  visualizar gráfico.

1- O Sistema apresenta o gráfico com a representação gráfica do gabarito gerado.

A4 - O usuário clica no botão  cancelar

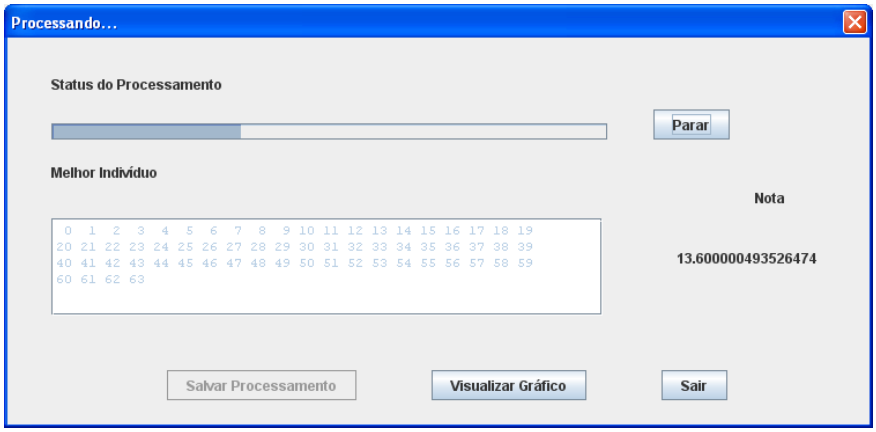
1- O Sistema fecha tela de Salvar Processamento.

Fluxo de exceção:

E1 – Campo não for informado o sistema, emite a mensagem “O campo está em branco”

E2 – No Banco de Dados um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1 -  Emite a mensagem “Erro! Já existe um Gabarito com este nome no banco”




UC007- Excluir  Gabarito.

Descrição:

Este caso de uso descreve os passos necessários para se fazer a exclusão e a consulta aos gabaritos processados.

Pré- requisito:

1- Existir gabaritos processados.

Pós-condições:

1- O usuário ter obtido as informações sobre os gabaritos processados e/ou excluído os gabaritos desejados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1- O usuário clica no botão gabarito.

2- O sistema abre a janela gabarito.

3- O usuário escolhe um gabarito.

4- O usuário clica no botão excluir. (A1) (E1)

5- O sistema apresenta a mensagem “Tem certeza que deseja excluir o gabarito xxxxx?(SIM /NÃO)”

6- O usuário clica no botão fechar.

7- O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica no botão visualizar

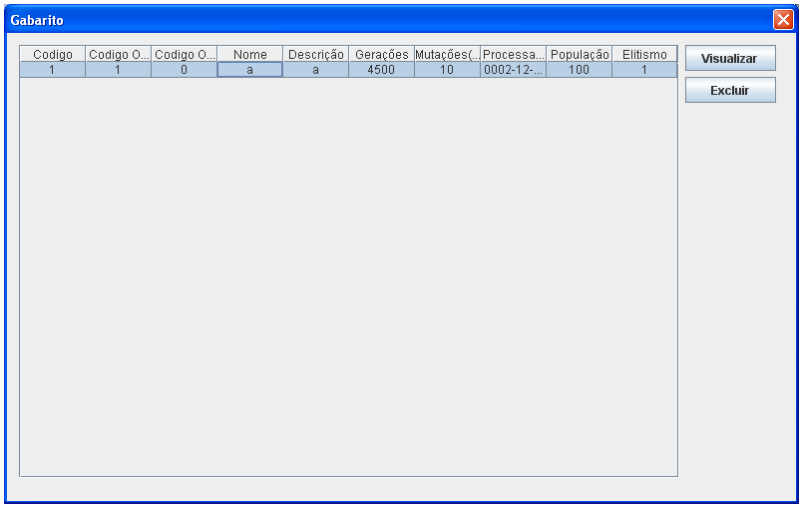
1- O sistema abre a janela visualizar gabarito.

2- O sistema volta ao fluxo principal.

Fluxo de exceção:

E1 – Existem tabelas treinos que usem o gabarito que está tentando-se excluir

1- O sistema apresenta a mensagem: “não é possível excluir esse gabarito, existem tabelas treinos que depende dele.”.




UC008 - Cadastrar ORF treino.

Descrição:

Este caso de uso descreve os passos necessários para se fazer o cadastro de uma ORF que será utilizada pela Rede Neural com fins de treino.

Pré- requisito:

1- Existir uma ORF para ser cadastrada.

Pós-condições:

1- O sistema deve ter cadastrado a ORF no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica no botão ORF treino.

2 - O sistema abre a janela ORF treino.

3 - O usuário clica no botão cadastrar.

4 - O sistema abre a tela cadastrar ORF base. (A1)

5 - O usuário clica o botão arquivo.

6 - O sistema abre a tela abrir.

7 - O usuário escolhe o arquivo que contém a ORF a ser cadastrada.

8 - O usuário clica no botão abrir. (E2)

9 - O usuário edita o campo nome. (E1)

10- O usuário edita o campo  observação.  (E1)

11- O usuário escolhe a classe da ORF. (E1)

12 - O usuário edita o campo ORF. (E1)

13 - O usuário clica em cadastrar.

14 - O sistema registra a ORF no banco de dados.

15 - O caso de uso é finalizado.

Fluxo alternativo:

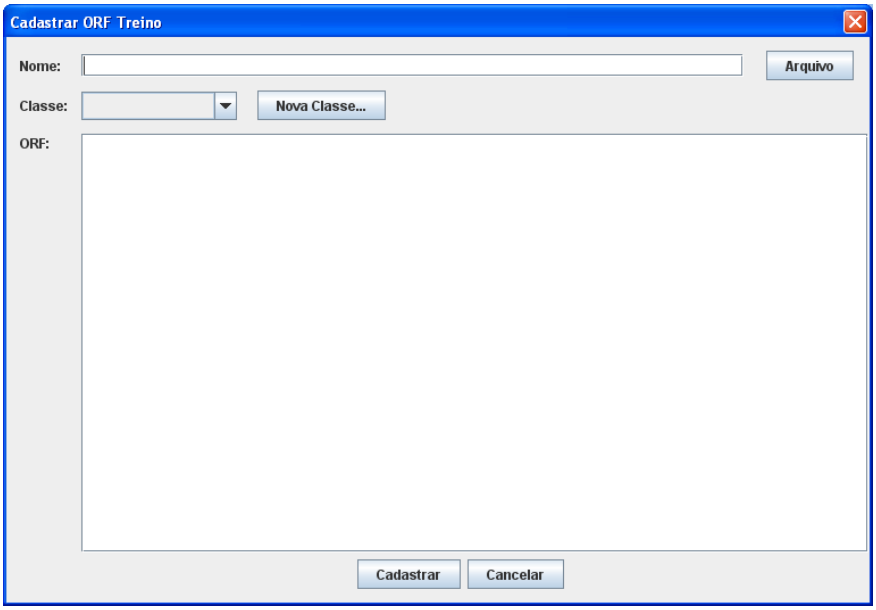
A1 – O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 – Campo não for informado, o sistema emite a mensagem “O campo está em branco”

E2 - O sistema apresenta a mensagem: ”O arquivo não está no padrão FASTA!”.




UC009 - Editar ORF Treino

Descrição:

Este caso de uso descreve como uma ORF Treino já cadastrada no sistema é editada.

Pré- requisito:

1- Existir a ORF Treino no cadastro do sistema.

Pós-condições:

1- O sistema deve ter gravado a ORF editada no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1- O usuário clica no botão  ORF treino.

2 - O sistema abre a tela ORF treino. (A2)

3 - O usuário escolhe a ORF treino a ser editado.

4 - O usuário clica no botão editar. (A1)

5 - O sistema abre a tela adicionar ORF Treino.

6 - O usuário edita o campo nome. (E1)

7 - O usuário edita o campo  observação.  (E1)

8 - O usuário escolhe a classe da ORF. (E1)

9 - O usuário edita o campo ORF. (E1)(E2)

10 - O usuário clica no botão salvar.

11 - O caso de uso é finalizado.

Fluxo alternativo:

A1  - O usuário clica no botão excluir

1- O sistema exclui a ORF selecionada.

2- O sistema volta ao fluxo principal.

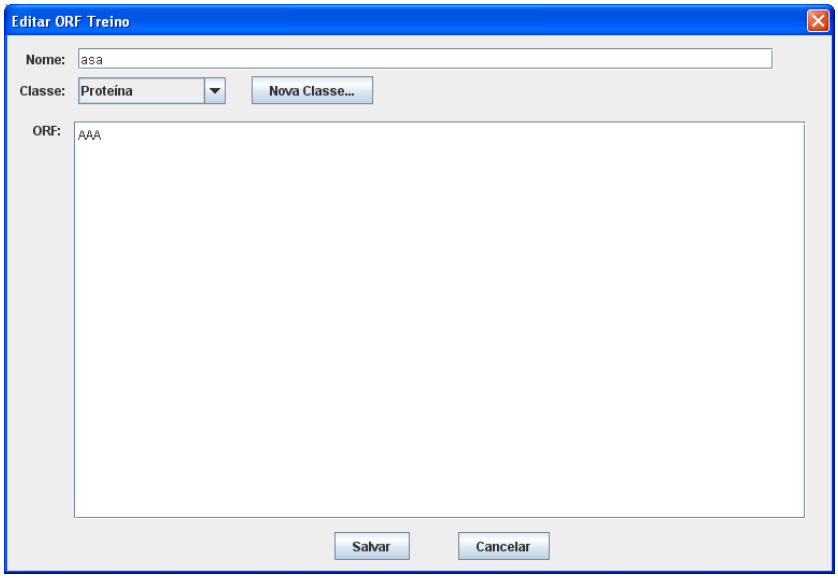
A2 – O usuário clica no botão  cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 – Campo não for informado, o sistema emite a mensagem “O campo está em branco”

E2 - O sistema apresenta a mensagem: ”A ORF não está no padrão FASTA!”.



UC010 - Excluir ORF Treino

Descrição:

Este caso de uso descreve como excluir uma ORF Base previamente cadastrada.

Pré- requisito:

1- Existir uma ORF Base cadastrada no sistema

Pós-condições:

1- O sistema deve ter excluído a ORF no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 - O usuário clica no botão ORF Treino.

2 - O sistema abre a janela ORF Treino. (A1)

3 - O usuário escolhe a ORF Treino para Exclusão entre as OFR disponíveis.

4 - O usuário clica no botão excluir. (E1)

5 - O sistema apresenta a mensagem “Tem certeza que deseja excluir esta ORF Treino?”

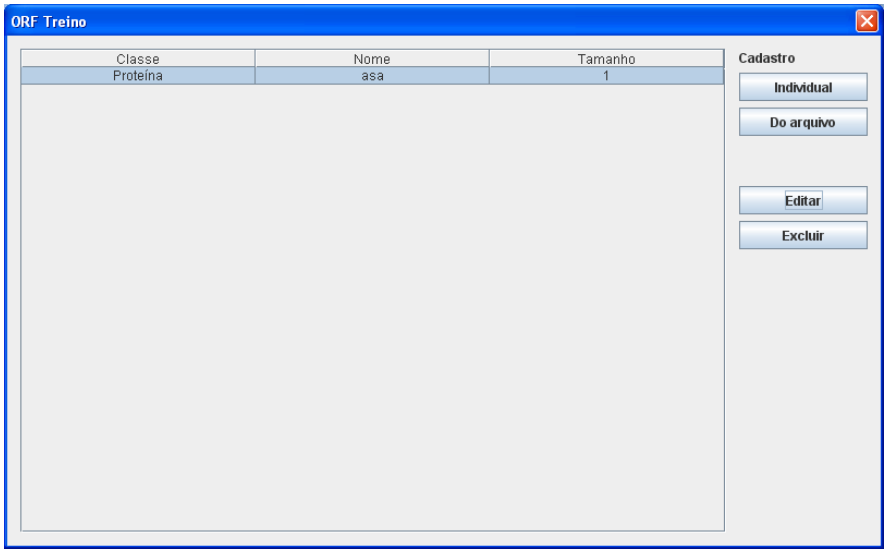
6- O sistema exclui a ORF no banco de dados.

7- O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica no botão cancelar.

1- O caso de uso é finalizado.




UC011 – Geração tabela treino.

Descrição:

Este caso de uso descreve os passos necessários para a geração de uma tabela treino.

Pré- requisito:

1- Existir gabaritos e ORF treinos disponíveis cadastrados no banco de dados.

Pós-condições:

1- O sistema deve ter gerado uma tabela treino.

Autor:

1- Usuário.

Fluxo de eventos principal:

1- O usuário clica no botão gerar tabela treino.

2- O sistema abre a janela gerar tabela treino.(A1)(A2)(A3)(A4)(A5)

3- O usuário clica no botão gerar.

4- O sistema abre a tela gerar tabela treino.

5- O sistema inicia a geração da tabela treino.

6- O sistema apresenta a tabela treino gerada. (A6)

7- O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário clica no botão  cancelar.

1- O caso de uso é finalizado.

A2  - O usuário clica no botão retirar (Gabarito).

1- O sistema exclui o gabarito selecionado.

2- O sistema volta ao fluxo principal.

A3  - O usuário clica no botão retirar (ORF Treino).

1- O sistema exclui a ORF Treino selecionada.

2- O sistema volta ao fluxo principal.

A4 - O usuário clica no botão Incluir Todos (Gabarito).

1- O sistema inclui todos os Gabaritos cadastrados.

2- O sistema volta ao fluxo principal.

A5 - O usuário clica no botão Incluir Todos (ORF Treino).

1- O sistema inclui todas as ORF Treino cadastradas.

2- O sistema volta ao fluxo principal.

A6 – O usuário clica no botão visualizar.

1- O sistema abre a janela visualizar gabarito.

2- O usuário clica no botão fechar.

3- O sistema volta ao fluxo principal.

A4 – O usuário clica no botão retirar.

1- O sistema exclui a ORF selecionada.

2- O sistema volta ao fluxo principal.

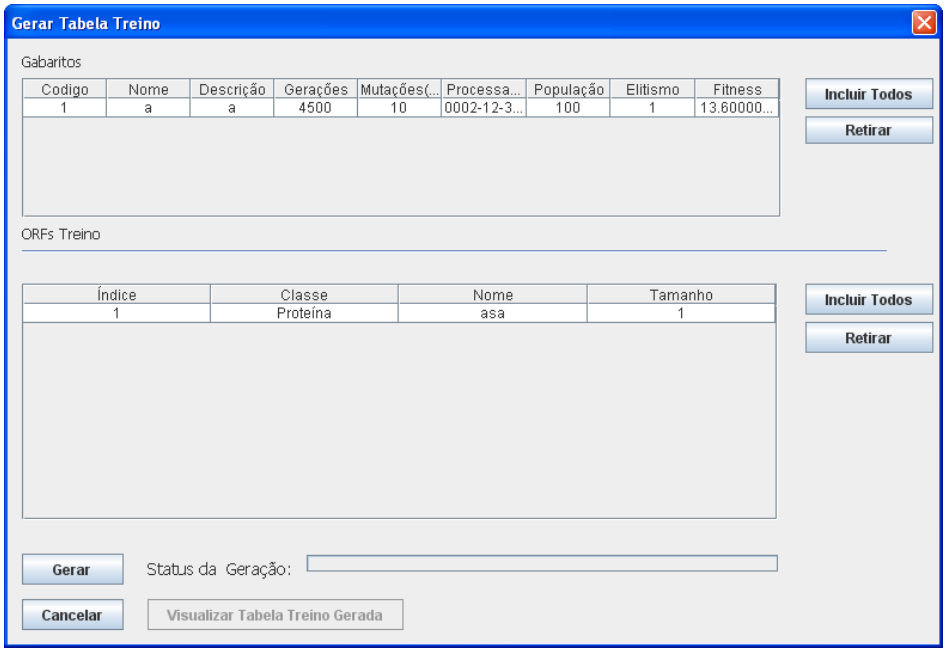



Diagrama de Seqüência: Geração Tabela Treino

Conforme demonstrado no apêndice 10.

UC012 – Ver tabela treino.

Descrição:

Este caso de uso descreve os passos necessários para se fazer consultas a tabelas treino geradas.

Pré- requisito:

1- Existir  tabelas treino geradas.

Pós-condições:

1- O usuário ter obtido as informações desejadas sobre as tabelas treino geradas.

Autor:

1- Usuário.

Fluxo de eventos principal:

1- O usuário clica no botão ver tabela treino.

2- O sistema abre a janela ver tabela treino.

3- O usuário clica na tabela treino e clica em visualizar. (A1)

4- O sistema abre a janela visualizar tabela treino. (A2)

5- O usuário clica no botão visualizar gabarito. (A2)

6- O sistema abre a janela visualizar gabarito.

7- O usuário clica no botão visualizar ORF treino. (A2)

8- O sistema  abre a tela visualizar ORF.

9-  O caso de uso é finalizado.

Fluxo alternativo:

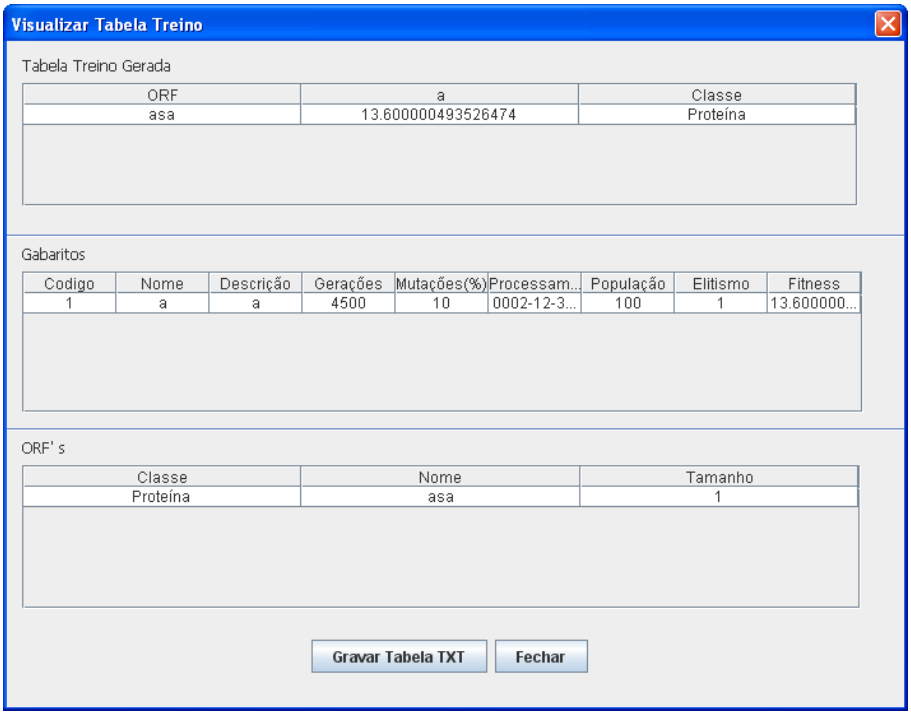
A1 – O usuário clica no botão excluir.

1- O sistema exclui a tabela treino selecionada.

2 – O sistema volta ao fluxo principal.

A2 – O usuário clica no botão fechar.

1- O caso de uso é finalizado.




UC013 – Cadastro Rede Neural.

Descrição:

Este caso de uso descreve os passos necessários para a geração dos dados para exportação da Rede Neural para o módulo web.

Pré- requisito:

1- Existir tabelas treino geradas e uma rede treinada e exportada pelo sistema EasyFAN.

Pós-condições:

1- O sistema deve ter os dados da Rede para exportação.

Autor:

1- Administrador.

Fluxo de eventos principal:

O usuário clica no botão cadastrar. (A1)(A2)(A3)

O sistema abre a tela de Cadastrar Rede Neural.

O usuário informa o nome da Rede. (E1)

O usuário informa a descrição da Rede. (E1)

O usuário escolhe o arquivo exportado pelo sistema EasyFan com a rede treinada. (E1)

O usuário clica em Cadastrar Rede.

O Sistema apresenta a mensagem “Rede cadastrada com sucesso!”

A tela de Cadastrar rede Neural é fechada.

O usuário clica no botão Exportar

O sistema gera um arquivo com a rede para importação pelo sistema JORFinder WEB.

O Caso de Uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha a tela.

1- O caso de uso é finalizado.

A2 – O usuário clica no botão Excluir.

1-O sistema exclui a rede selecionada da base de dados.

A3 – O usuário clica no botão Visualizar.

1-O sistema abre a tela de visualização da Rede.

Fluxo de exceção:

E1 – Campo não for informado, o sistema emite a mensagem “O campo está em branco”

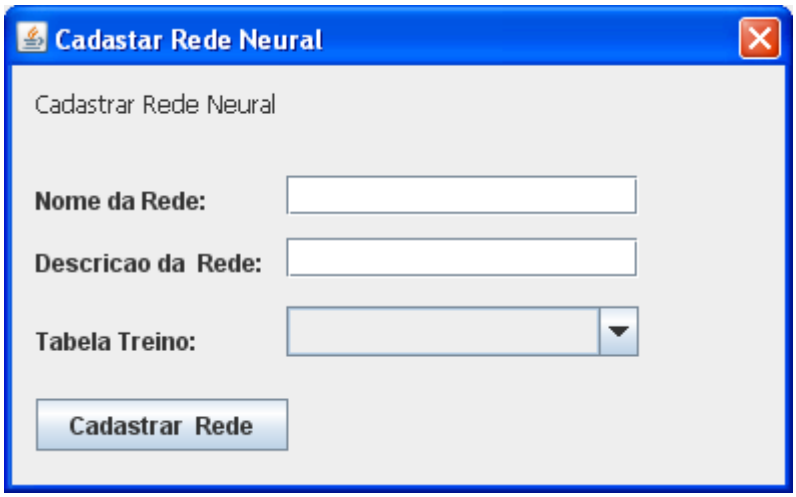
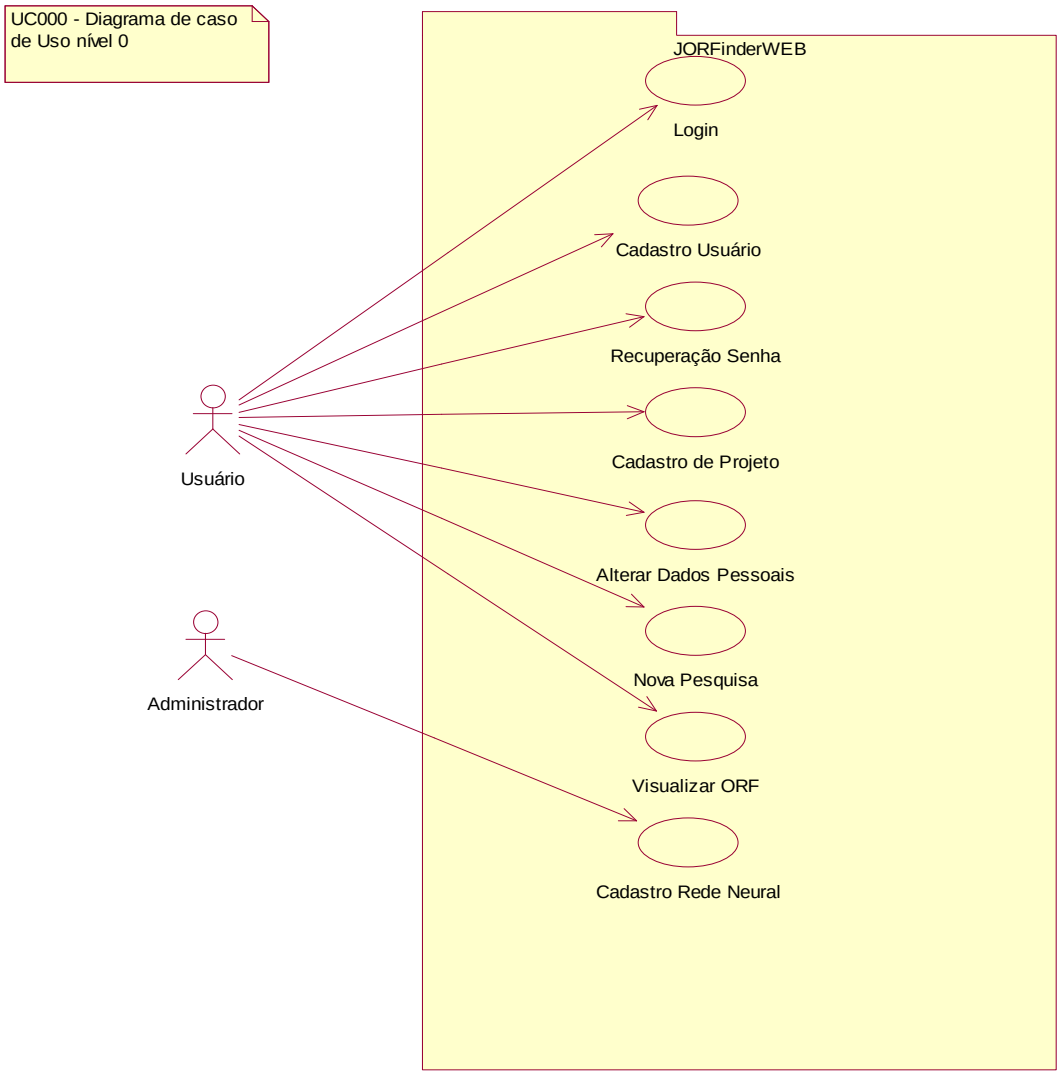



Diagrama de Seqüência: Cadastro Rede Neural

Conforme demonstrado no apêndice 11.

# APÊNDICE 2 – DIAGRAMA DE CASOS DE USO MÓDULO WEB



## ESPECIFICAÇÃO DOS CASOS DE USO

MÓDULO WEB

CASOS DE USO

UC001 - Login

Descrição:

Este caso de uso descreve ocorre o processo de login por parte do usuário no sistema.

Pré- requisito:

1- O usuário ter uma conta cadastrada no sistema.

Pós-condições:

1- O deve ter acesso as funcionalidades do sistema.

Autor:

1- Usuário.

Fluxo de eventos principal:

1 - O usuário digita o endereço do sistema no browser.

2 - O sistema abre a tela de login. (A1)

3 - O usuário digita o login. (E1)

4 - O usuário digita a senha. (E1)

5 - O usuário clica no botão Entrar. (E2)

6 - O sistema abre a tela principal do sistema.

7 - O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 - O sistema apresenta a mensagem: ”Usuário ou senha incorreto”.

Diagrama de Seqüência: Login

Conforme demonstrado no apêndice 12.

UC002 - Cadastro usuário

Descrição:

Este caso de uso descreve como é feito o cadastro de um novo usuário para acesso ao sistema.

Pré- requisito:

Pós-condições:

1- O usuário tem um cadastro válido para acesso ao sistema.

Autor:

1- Usuário.

Fluxo de eventos principal:

1 - O usuário clica no link Cadastrar-se.

2 - O sistema abre a tela de Cadastro de Usuário. (A1)

3 - O usuário informa o nome. (E1)

4 - O usuário informa o email (E1) (E2)( E3)

5 - O usuário informa a instituição (E1)

6 - O usuário informa o login (E1)( E3)

7- O usuário informa a senha (E1)

8 - O usuário redigita a senha (E1)

9- O usuário clica no botão salvar. (E1)

10- O sistema grava a conta nova no sistema.

11- O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 - O sistema apresenta a mensagem “Erro! Campo email não está preenchido corretamente!”

E3 – No Banco de Dados já existe um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe um usuário com este nome/email no banco”


UC003 - Recuperação da senha

Descrição:

Este caso de uso descreve como o usuário recupera a senha esquecida pelo sistema.

Pré- requisito:

1- O usuário precisa ter um login válido e lembrar-se do login e email que foi cadastrado no sistema

Pós-condições:

1- O sistema deve enviar para o email do usuário sua senha.

Autor:

1- Usuário.

Fluxo de eventos principal:

1 - O usuário clica no link Esqueci a senha.

2 - O sistema abre a tela de Esqueceu a Senha.

3 - O usuário informa o email (E1)(E2)

4- O usuário informa o login (E1)

5 - O usuário clica no botão Enviar senha (E1)(E3)

6- O sistema envia a senha para o usuário.

7- O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 - O sistema apresenta a mensagem “Erro! Campo email não está preenchido corretamente!”

E3 - O sistema apresenta a mensagem “Erro! Não existe email cadastrado com esse login!”


UC004 - Cadastro de Projeto

Descrição:

Este caso de uso descreve como o usuário cria um novo projeto dentro do sistema.

Pré- requisito:

1- O usuário deve ter logado no sistema.

Pós-condições:

1- O sistema deve ter criado um projeto de acordo com os dados informados pelo usuário.

Autor:

1- Usuário.

Fluxo de eventos principal:

1 - O usuário clica no botão Novo Projeto.

2 - O sistema abre a Tela de Cadastro de Projeto. (A1)

3- O usuário preenche o campo Nome. (E1)(E2)

4- O usuário preenche o campo Descrição. (E1)

5- O usuário seleciona os usuários que farão parte do projeto

11- O usuário clica no botão Cadastrar.

12 - O sistema salva o projeto.

14 - O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 – No Banco de Dados já existe um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe um projeto com este nome no banco”


UC005- Editar Projeto.

Descrição:

Este caso de uso descreve como usuário altera os dados de um projeto.

Pré- requisito:

1- Existir um projeto cadastrado.

2 – usuário ser o dono do projeto.

Pós-condições:

1- O sistema deve ter atualizado os dados do projeto no Banco de dados.

Autor:

1- Usuário.

Fluxo de eventos principal:

1 - O usuário clica no link editar.

2 - O sistema abre a Tela de Cadastro de Projeto.  (A1)

3- O usuário altera o campo Nome. (E1)(E2)

4- O usuário altera o campo Descrição. (E1)

5- O usuário seleciona ou remove os usuários que farão parte do projeto

11- O usuário clica no botão Alterar.

12 - O sistema salva o projeto.

14 - O caso de uso é finalizado.

Fluxo alternativo:

A1 - O usuário clica fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 – No Banco de Dados já existe um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe um projeto com este nome no banco”

UC007 - Excluir Projeto

Descrição:

Este caso de uso descreve os passos necessários para excluir um projeto

Pré- requisito:

1- Existir um projeto cadastrado e o usuário ser o dono do projeto.

Pós-condições:

1- O sistema deve ter excluído o projeto no banco de dados

Autor:

1- Usuário.

Fluxo de eventos principal:

1 – O usuário abre a tela de projetos.

2 - O usuário clica no link Excluir.

3 – O sistema apresenta a mensagem “Tem certeza que deseja excluir esse registro?”

4 - O sistema exclui o projeto no banco de dados.

5 - O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha o browser.

1- O caso de uso é finalizado.


UC008 - Cadastro Rede Neural

Descrição:

Este caso de uso descreve como o administrador cadastra uma nova rede no sistema.

Pré- requisito:

1- O administrador estar logado no sistema e existir um arquivo de rede gerado pelo módulo J-ORFinder Desktop.

Pós-condições:

1- O sistema deve ter gravado a nova rede no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1- O usuário clica no link Rede Neural.

2 - O sistema abre a tela de Rede Neural. (A1)

3 - O usuário clica em nova rede.

4 - O usuário informa o campo Rede com o caminho para o arquivo a ser importado. (E1)

5 - O usuário informa o campo Nome (E1)(E2)

6 - O usuário informa o campo Objetivo da rede

10 - O usuário clica no botão cadastrar.

11 - O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 – No Banco de Dados já existe um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe uma rede com este nome no banco”


Diagrama de Seqüência: Cadastro Rede Neural

Conforme demonstrado no apêndice 11.

UC009 - Excluir Rede Neural

Descrição:

Este caso de uso descreve como excluir uma Rede previamente cadastrada.

Pré- requisito:

1- Existir uma Rede cadastrada no sistema e essa Rede não pode estar sendo usada em nenhum projeto.

Pós-condições:

1- O sistema deve ter excluído a Rede no banco de dados.

Autor:

1- Administrador.

Fluxo de eventos principal:

1 – O usuário abre a tela de Redes.

2 - O usuário clica no link Excluir.

3 – O sistema apresenta a mensagem “Tem certeza que deseja excluir esse registro?”

4 - O sistema exclui a Rede Neural no banco de dados.

5 - O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha o browser.

1- O caso de uso é finalizado.


UC010 – Alterar Dados Pessoais.

Descrição:

Este caso de uso descreve como usuário altera seus dados pessoais.

Pré- requisito:

1- o usuário estar logado no sistema.

Pós-condições:

1- O sistema deve ter os dados do usuário no banco de dados.

Autor:

1- Usuário.

Fluxo de eventos principal:

1- O usuário clica no link Meus Dados.

2- O sistema abre a tela Meus dados

3- O usuário edita o campo Nome (E1)

4- O usuário edita o campo email (E1) (E2)(E3)

5- O usuário edita o campo Instituição. (E1)

6- O usuário edita o campo senha (E1)

7- O usuário confirma o campo senha (E1)

8- O usuário clica no botão Alterar Dados

7- O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário clica no botão cancelar.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

E2 - O sistema apresenta a mensagem “Erro! Campo email não está preenchido corretamente!”

E3 – No Banco de Dados já existe um registro com o conteúdo do campo com o mesmo valor informado pelo usuário

1- Emite a mensagem “Erro! Já existe um usuário com este email no banco”


UC011 – Nova pesquisa

Descrição:

Este caso de uso descreve como usuário efetua uma nova pesquisa de ORFs no sistema.

Pré- requisito:

1- O usuário estar logado em um projeto.

Pós-condições:

1- O sistema apresenta o resultado da pesquisa para o usuário.

Autor:

1- Usuário.

Fluxo de eventos principal:

1- O usuário clica no botão Nova Pesquisa.

2- O sistema abre a tela de Nova Pesquisa.

3- O usuário escolhe a rede

4- O usuário informa o tamanho mínimo da ORF (E1)

5- O usuário informa o nome da pesquisa (E1)

6- O usuário informa o DNA a ser consultado (E1)

7- O usuário clica em Pesquisar.

8- O sistema apresenta o resultado da pesquisa com base na consulta feita a rede neural.

9- O caso de uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha o browser.

1- O caso de uso é finalizado.

Fluxo de exceção:

E1 - O sistema apresenta a mensagem “Erro! Campo não está preenchido!”

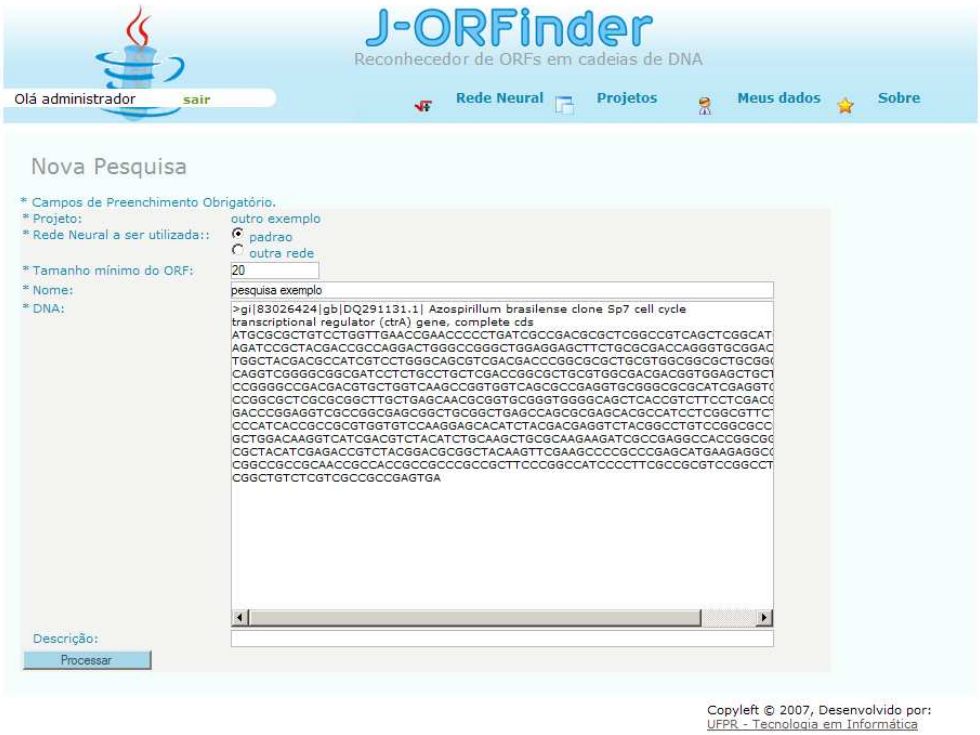



Diagrama de Seqüência: Nova Pesquisa

Conforme demonstrado no apêndice 13.

UC012 – Visualizar ORF

Descrição:

Este caso de uso descreve como o usuário visualiza a ORF encontrada pela pesquisa efetuada em um DNA.

Pré- requisito:

1- Existir uma pesquisa feita e a mesma retornar pelo menos uma ORF.

Pós-condições:

1- O usuário ter acesso aos dados referentes à ORF encontrada.

Autor:

1- Usuário.

Fluxo de eventos principal:

O usuário clica em uma ORF encontrada pela Pesquisa.

O sistema abre a tela de Visualização de ORF.

O sistema apresenta a seqüência de genes da ORF e sua classificação (A2)(A3)(A4)

O Caso de Uso é finalizado.

Fluxo alternativo:

A1 – O usuário fecha a tela.

1- O caso de uso é finalizado.

A2 – O usuário altera a classificação da ORF.

1 - O usuário edita a classificação

2- O usuário clica em Salvar

3- O sistema salva os dados

A3 – O usuário informa uma observação.

1-O usuário informa uma observação.

2- O usuário clica em Salvar

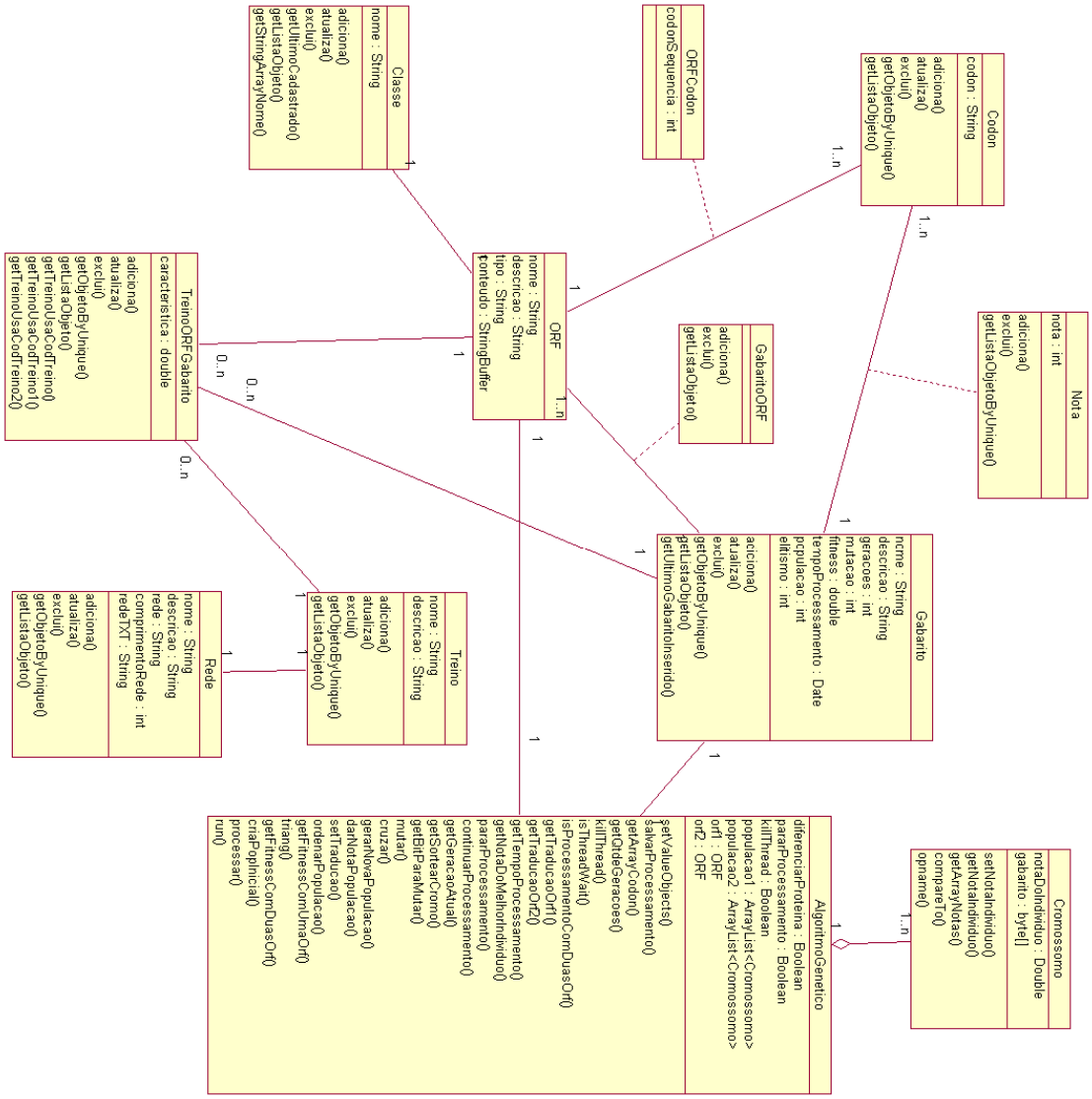
3- O sistema salva os dados

A3 – O usuário consulta a base NCBI.

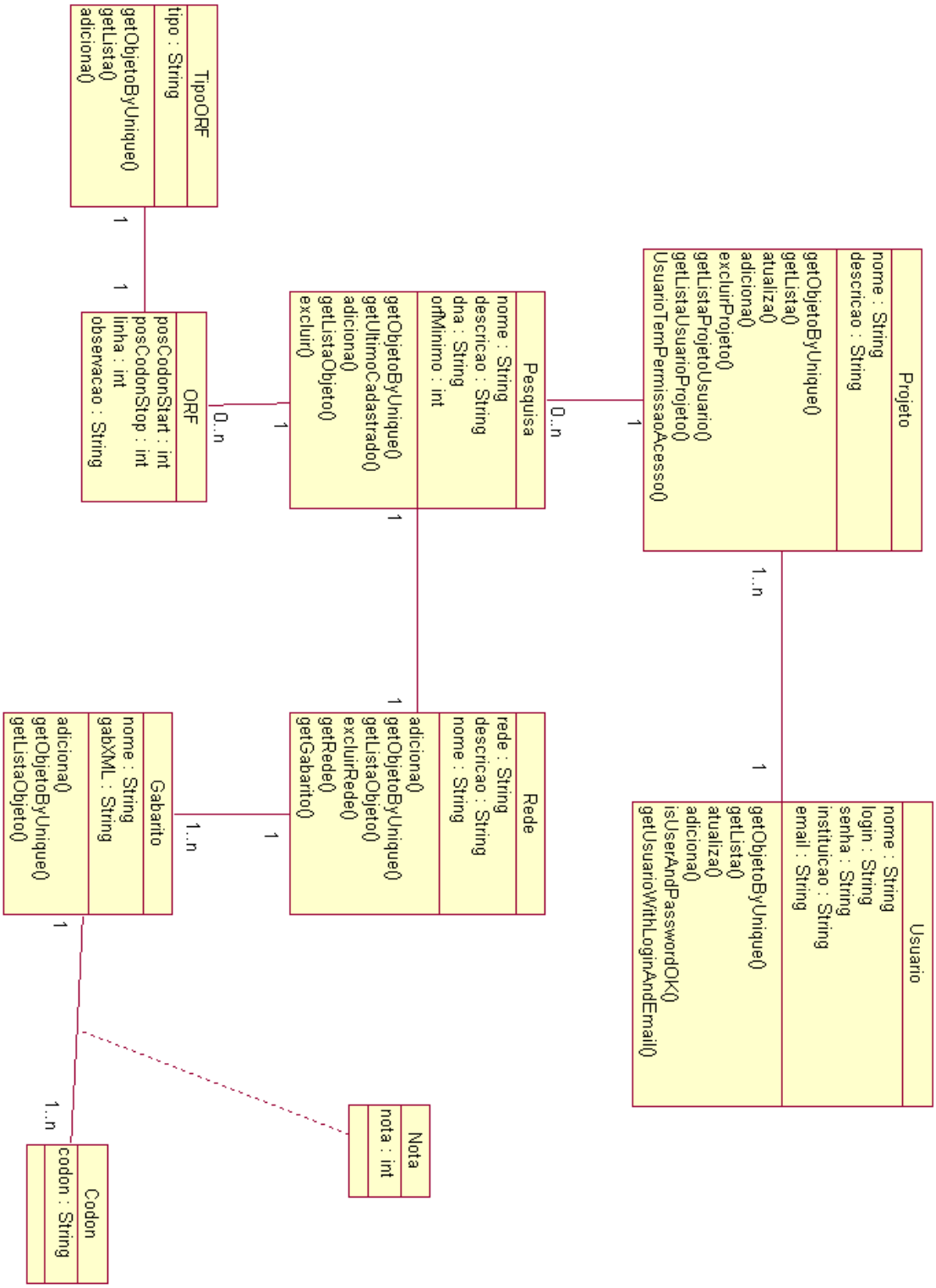
1-O usuário clica em Pesquisar.

2- O sistema abre uma tela com a orf sendo submetida ao sistema NCBI

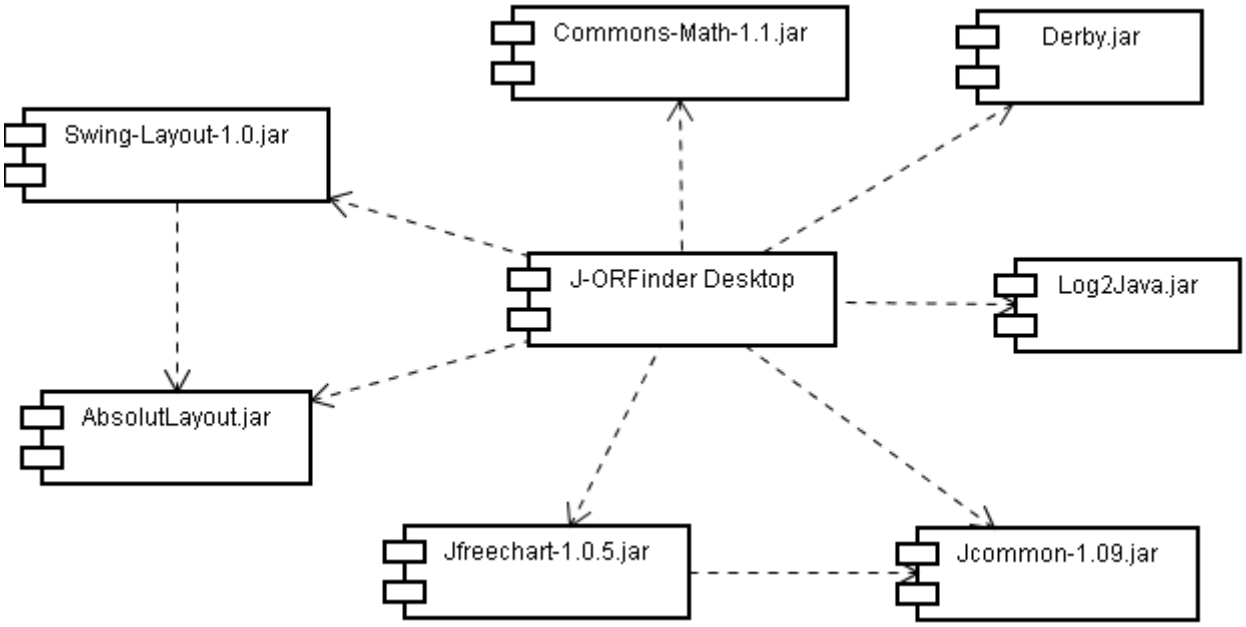
# APÊNDICE 3 – DIAGRAMA DE CLASSES MÓDULO DESKTOP



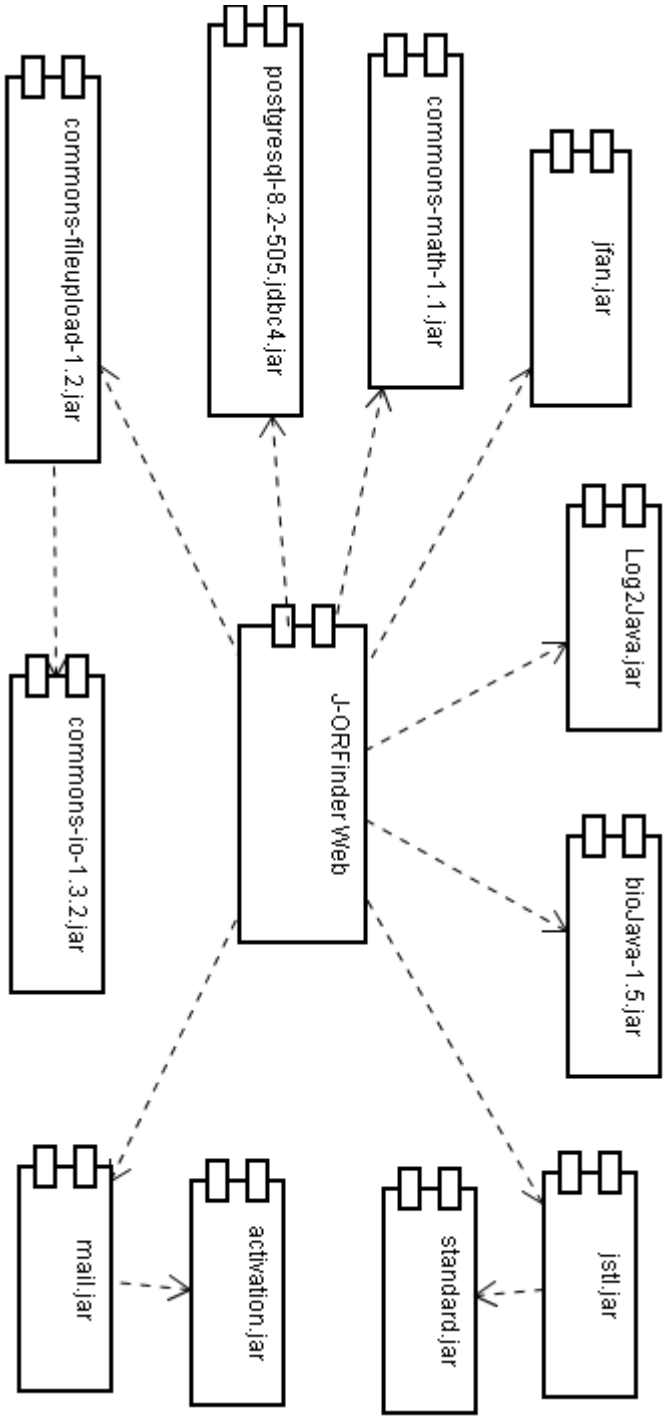
# APÊNDICE 4 – DIAGRAMA DE CLASSES MÓDULO WEB



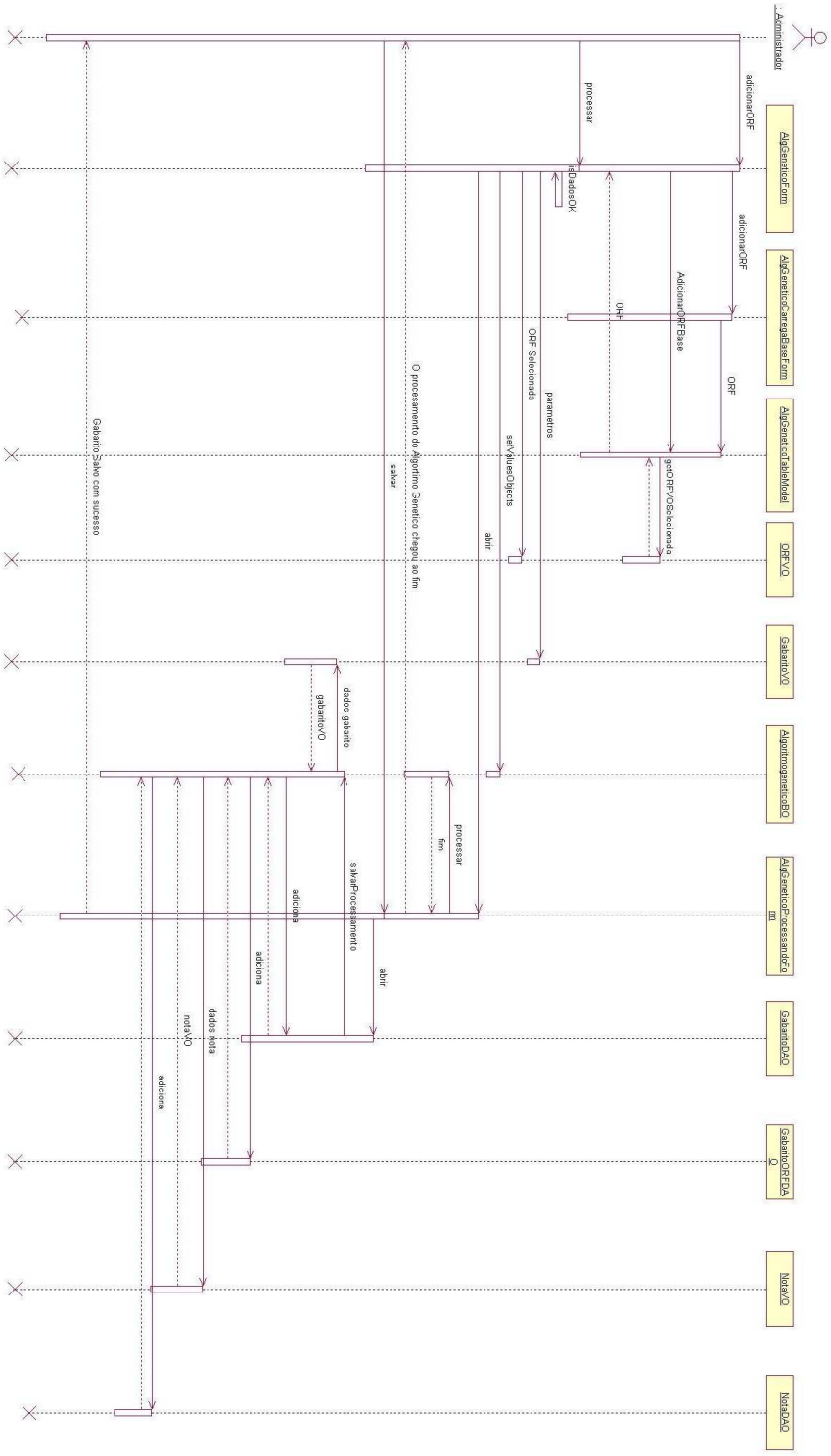
## APÊNDICE 5 – DIAGRAMA DE COMPONENTES MÓDULO DESKTOP



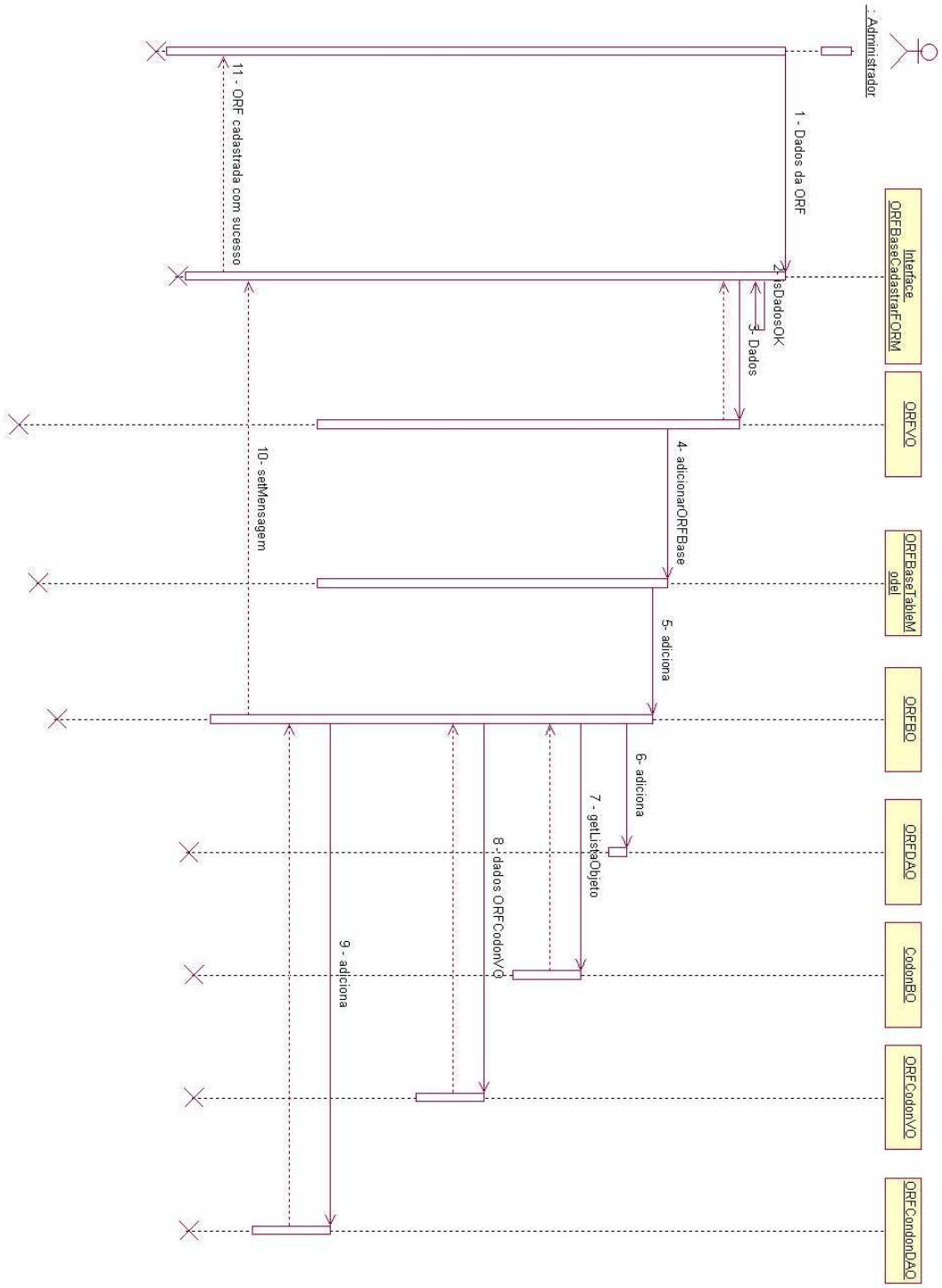
# APÊNDICE 6 – DIAGRAMA DE COMPONENTES MÓDULO WEB



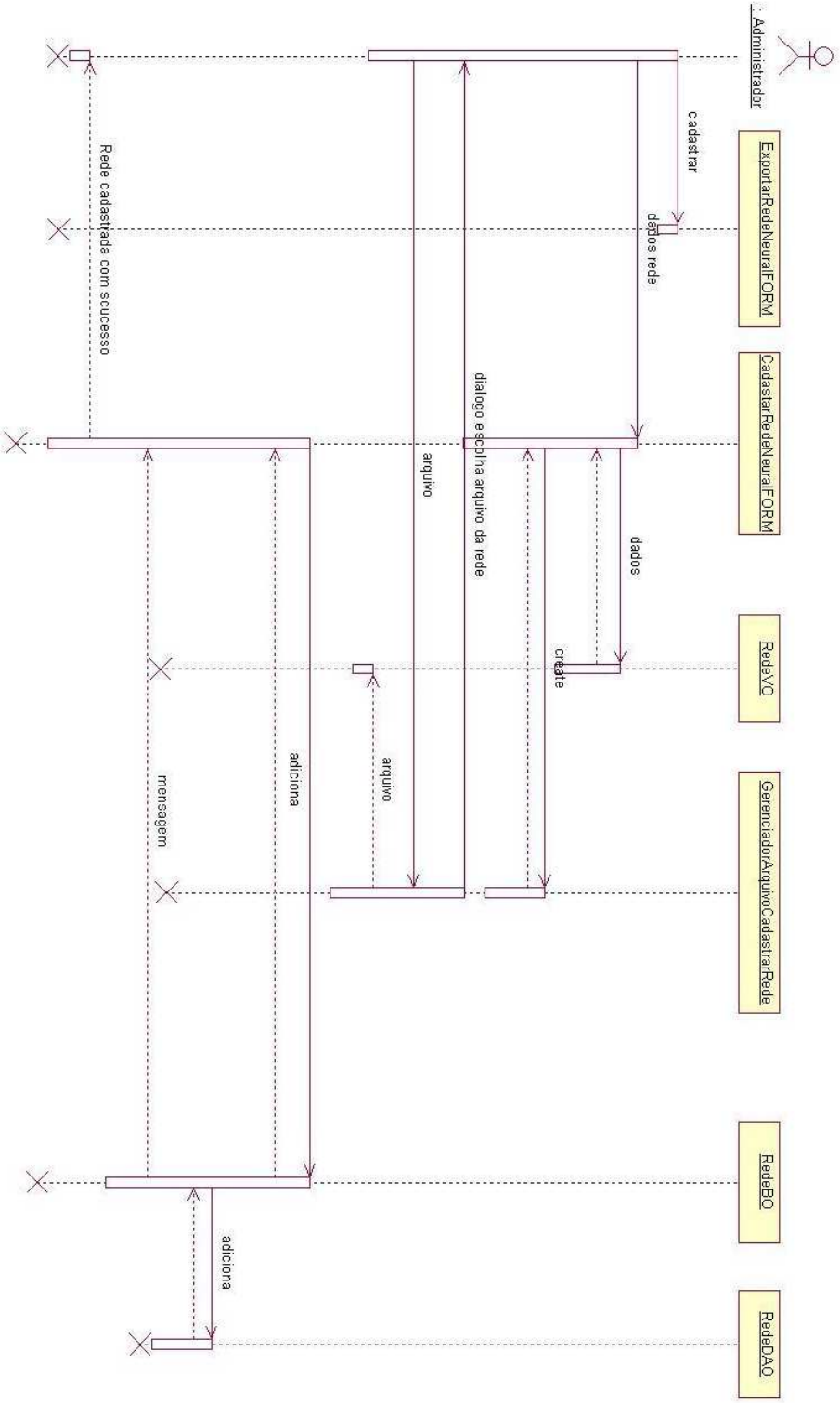
# APÊNDICE 7 – DIAGRAMA DE SEQUENCIA MÓDULO DESKTOP ALGORITMO GENÉTICO



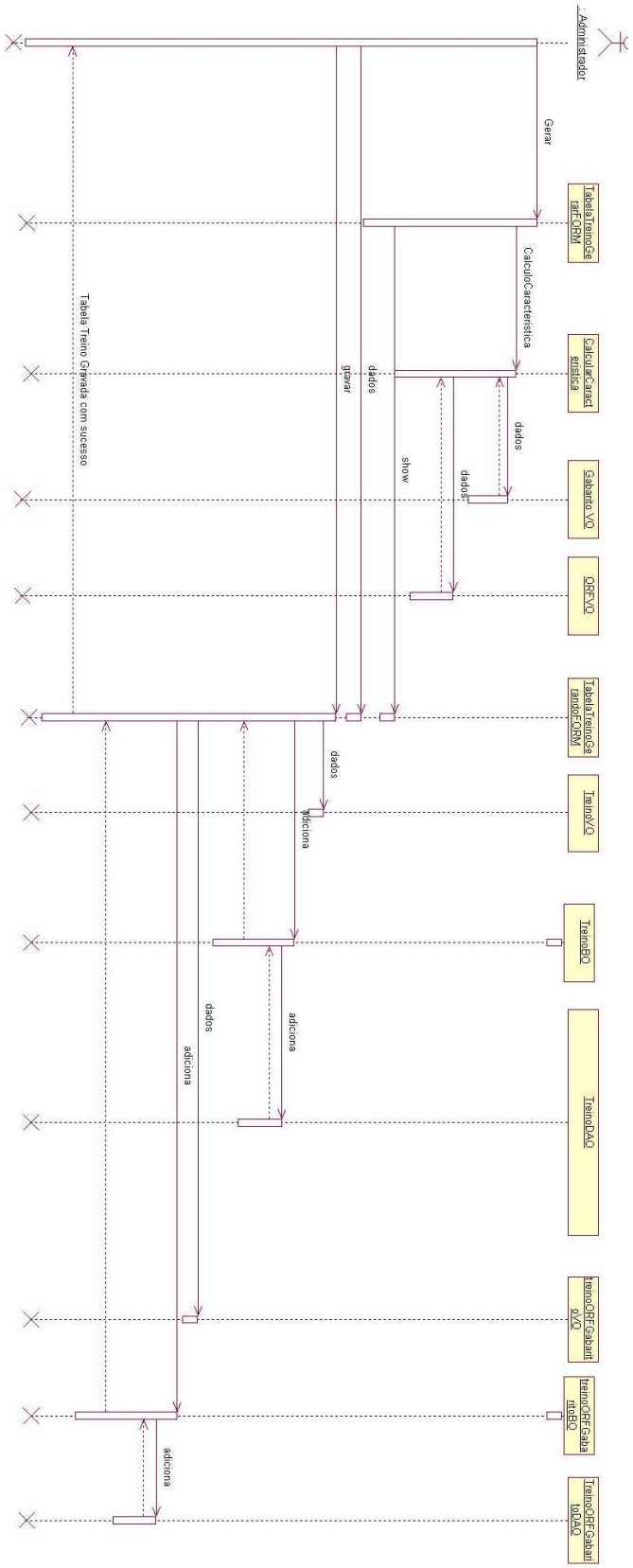
# APÊNDICE 8 – DIAGRAMA DE SEQUENCIA MÓDULO DESKTOP CADASTRO DE ORF BASE



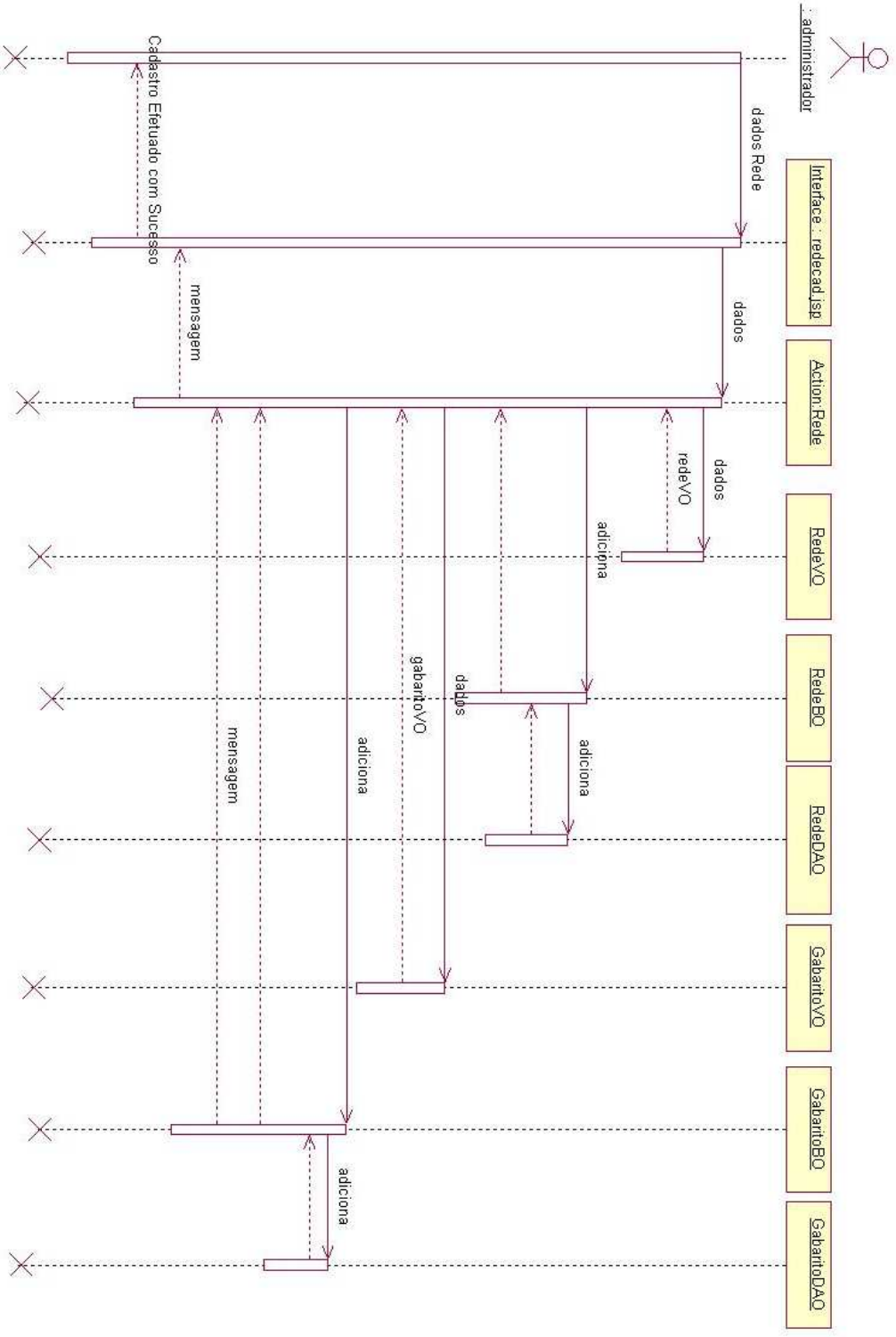
# APÊNDICE 9 – DIAGRAMA DE SEQUENCIA MÓDULO DESKTOP CADASTRO DE REDE NEURAL



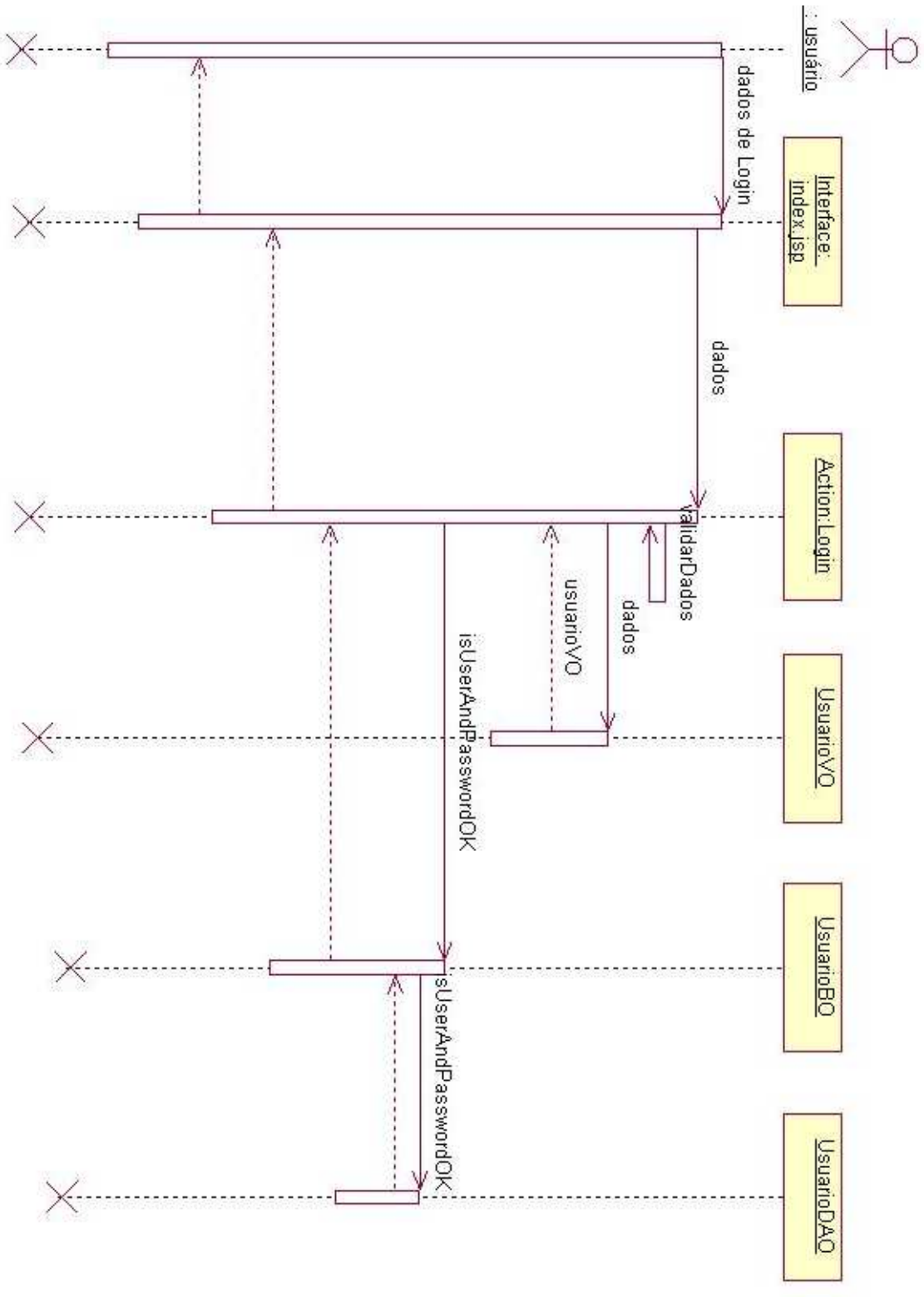
# APÊNDICE 10 – DIAGRAMA DE SEQUENCIA MÓDULO DESKTOP GERAÇÃO DA TABELA TREINO



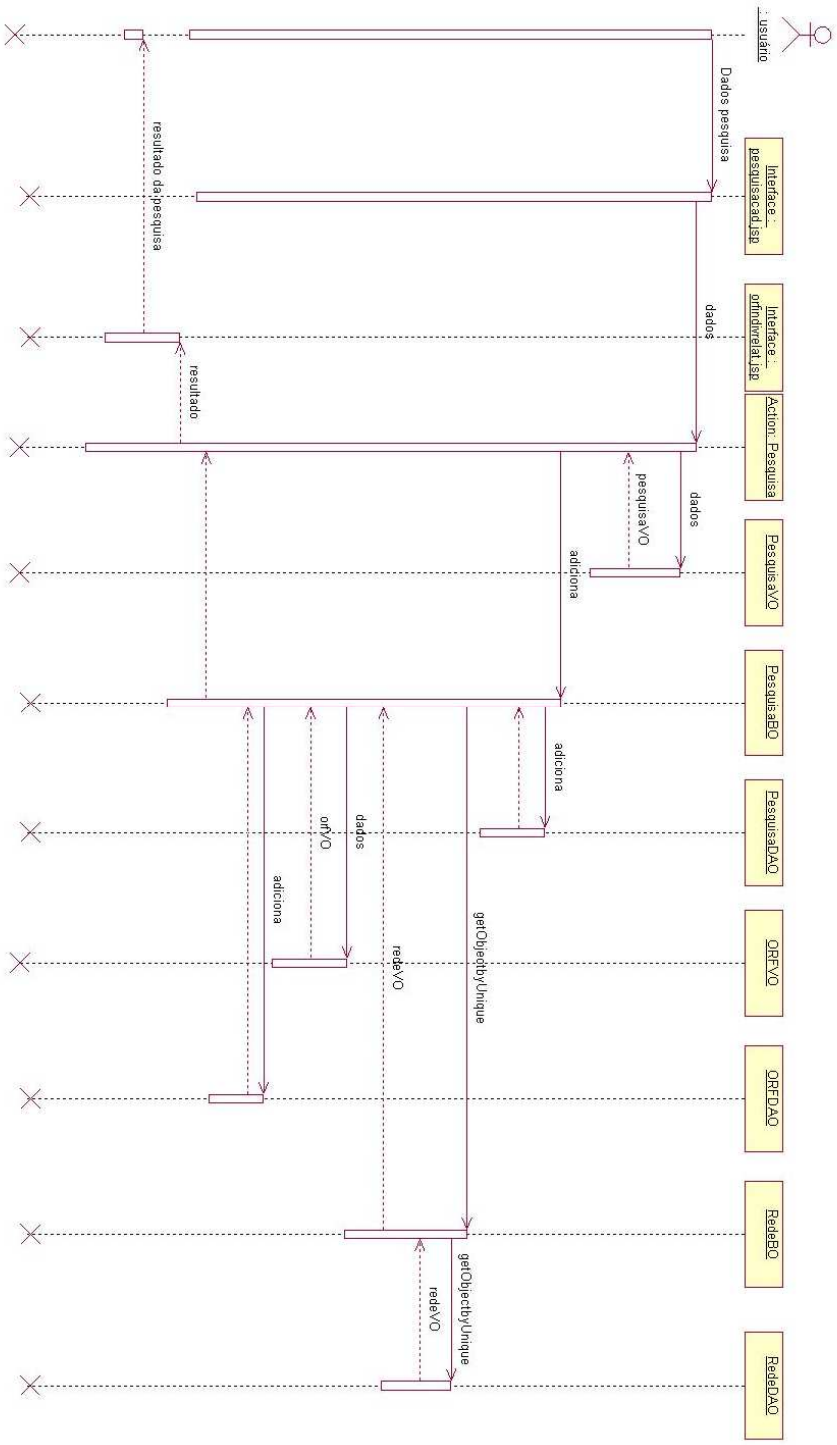
# APÊNDICE 11– DIAGRAMA DE SEQUENCIA MÓDULO WEB CADASTRO DE REDE NEURAL



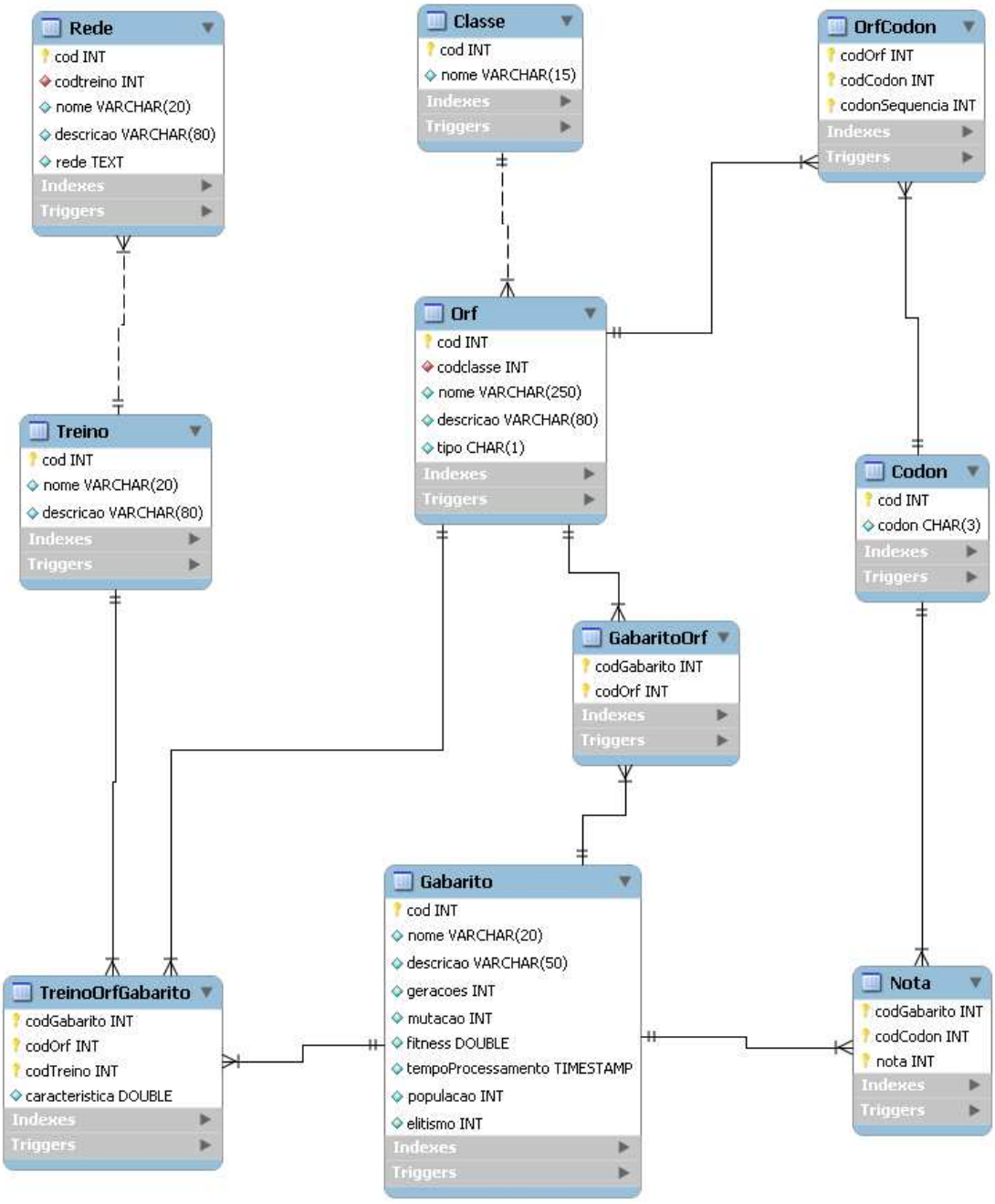
# APÊNDICE 12 – DIAGRAMA DE SEQUENCIA MÓDULO WEB LOGIN



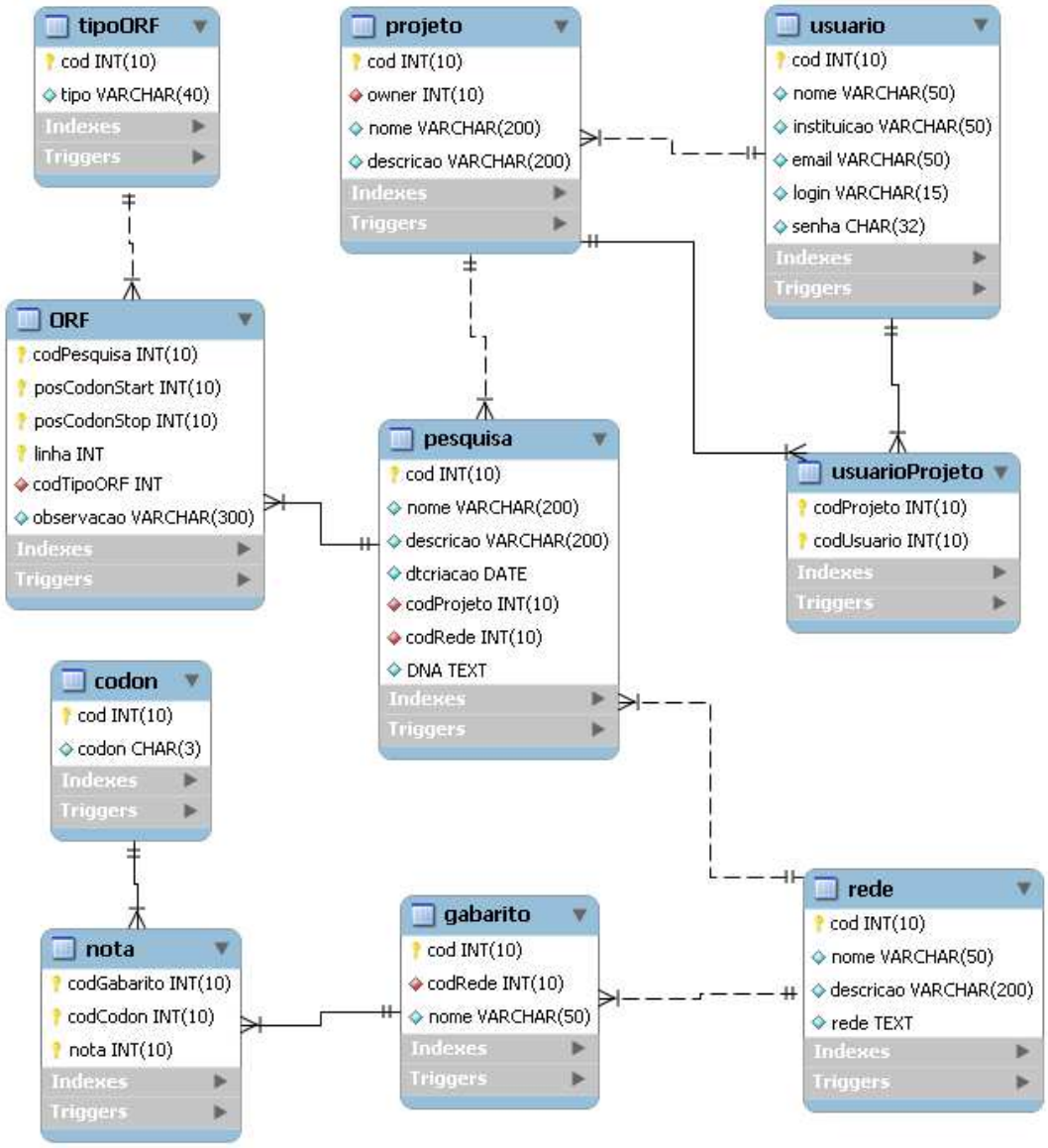
# APÊNDICE 13 – DIAGRAMA DE SEQUENCIA MÓDULO WEB NOVA PESQUISA



# APÊNDICE 14 – DIAGRAMA DE ENTIDADE RELACIONAMENTO MÓDULO DESKTOP



# APÊNDICE 15 – DIAGRAMA DE ENTIDADE RELACIONAMENTO MÓDULO WEB



# APÊNDICE 16 – DICIONÁRIO DE DADOS MÓDULO DESKTOP

Tabela treino

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| znome | varchar(20) | Yes | No | | |
| descricao | varchar(80) | | | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| treino_pkey | Primary key | (cod) |
| treino_nome_key | Unique | (nome) |

Tabela gabarito

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| nome | varchar(20) | Yes | No | | |
| descricao | varchar(50) | No | No | | |
| geracoes | int | No | No | | |
| mutacao | int | No | No | | |
| fitness | double | No | No | | |
| tempoProcessamento | timestamp | | | | |
| populacao | int | No | No | | |
| elitismo | int | No | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| gabarito_pkey | Primary key | (cod) |
| gabarito_nome_key | Unique | (nome) |

Tabela classe

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| nome | varchar(15) | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| classe_pkey | Primary key | (cod) |
| classe_nome_key | Unique | (nome) |

Tabela codon

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| codon | varchar(3) | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| tipoorf_pkey | Primary key | (cod) |

Tabela rede

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| codTreino | int | Yes | No | | |
| nome | varchar(20) | Yes | No | | |
| descricao | varchar(80) | No | No | | |
| rede | clob(1M) | No | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| rede_pkey | Primary key | (cod) |
| rede_codTreino_fkey | Foreign key | (codTreino) REFERENCES treino(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela ORF

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | int | Yes | Yes | identity (start with 1, increment by 1) | |
| codClasse | int | Yes | No | | |
| nome | varchar(250) | Yes | No | | |
| descricao | varchar(80) | No | No | | |
| rede | clob(1M) | No | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| rede_pkey | Primary key | (cod) |
| rede_codTreino_fkey | Foreign key | (codTreino) REFERENCES treino(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela treinoORFGabarito

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| codGabarito | int | Yes | Yes | | |
| codOfr | int | Yes | Yes | | |
| codTreino | int | Yes | Yes | | |
| caracteristica | double | No | No | | |
| | | | | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| treinoOrfGabarito_pkey | Primary key | (codGabarito, codOrf, codTreino) |
| treinoOrfGabarito_codGabarito_fkey | Foreign key | (codGabarito) REFERENCES gabarito(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |
| treinoOrfGabarito_codOrf_fkey | Foreign key | (codOrf) REFERENCES ORF(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |
| treinoOrfGabarito_codTreino_fkey | Foreign key | (codTreino) REFERENCES treino(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela gabaritoORF

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| codGabarito | int | Yes | Yes | | |
| codOrf | int | Yes | Yes | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| gabaritoOrf_pkey | Primary key | (codGabarito, codOrf) |
| treinoOrfGabarito_codGabarito_fkey | Foreign key | (codGabarito) REFERENCES gabarito(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |
| treinoOrfGabarito_codOrf_fkey | Foreign key | (codOrf) REFERENCES ORF(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela ORFCodon

| Name | Data Type | Not Null? | Primaty Key? | Default | Comment |
|---|---|---|---|---|---|
| codOrf | int | Yes | Yes | | |
| codCodon | int | Yes | Yes | | |
| codonSequencia | int | Yes | Yes | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| orfCodon_pkey | Primary key | (codOrf, codCodon, codonSequencia) |
| orfCodon_codCodon_fkey | Foreign key | (codCodon) REFERENCES codon(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |
| orfCodon_codOrf_fkey | Foreign key | (codOrf) REFERENCES orf(cod) ON UPDATE NO ACTION ON DELETE CASCADE; |

Tabela nota

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| codGabarito | int | Yes | Yes | | |
| posCodon | int | Yes | Yes | | |
| nota | int | Yes | Yes | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| nota_pkey | Primary key | (codGabarito, codCodon, nota) |
| nota_codGabarito_fkey | Foreign key | (codNota) REFERENCES nota(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |
| nota_codCodon_fkey | Foreign key | (codCodon) REFERENCES codon(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

# APÊNDICE 17 – DICIONÁRIO DE DADOS MÓDULO WEB

Tabela tipoORF
Constraints

| Name | Type | Definition |
|---|---|---|
| tipoorf_pkey | Primary key | (cod) |

Tabela rede

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('rede_cod_seq') | |
| nome | varchar(50) | Yes | No | | |
| descricao | varchar(200) | No | No | | |
| rede | text | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| rede_pkey | Primary key | (cod) |
| rede_nome_key | Unique | (nome) |

Tabela usuario

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('usuario_cod_seq') | |
| nome | varchar(50) | Yes | No | | |
| instituicao | varchar(50) | Yes | No | | |
| email | varchar(50) | Yes | No | | |
| login | varchar(15) | Yes | No | | |
| senha | char(32) | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| usuario_pkey | Primary key | (cod) |
| usuario_nome_key | Unique | (nome) |
| usuario_email_key | Unique | (email) |

Tabela projeto

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('projeto_cod_seq') | |
| owner | Integer | Yes | No | | |
| nome | varchar(200) | Yes | No | | |
| descricao | varchar(200) | No | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| projeto_pkey | Primary key | (cod) |
| projeto_nome_key | Unique | (nome) |
| projeto_owner_fkey | Foreign key | (owner) REFERENCES usuario (cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela codon

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('codon_cod_seq') | |
| codon | varchar(3) | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| codon_pkey | Primary key | (cod) |

Tabela gabarito

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('gabarito_cod_seq') | |
| codRede | integer | Yes | No | | |
| nome | varchar(50) | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| gabarito_pkey | Primary key | (cod) |
| gabarito_codRede_fkey | Foreign key | (codRede) REFERENCES rede (cod) ON UPDATE CASCADE ON DELETE CASCADE; |

Tabela pesquisa

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| cod | serial | Yes | Yes | nextval('pesquisa_cod_seq') | |
| nome | varchar(200) | Yes | No | | |
| descricao | varchar(200) | No | No | | |
| dtcriacao | Date | No | No | | |
| codProjeto | integer | Yes | No | | |
| codRede | integer | Yes | No | | |
| DNA | text | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| pesquisa_pkey | Primary key | (cod) |
| pesquisa_codProjeto_fkey | Foreign key | (codProjeto) REFERENCES projeto (cod) ON UPDATE CASCADE ON DELETE CASCADE; |
| pesquisa_codRede_fkey | Foreign key | (codRede) REFERENCES rede (cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela usuarioProjeto

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| codProjeto | integer | Yes | No | | |
| codUsuario | integer | Yes | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| usuarioProjeto_codProjeto_fkey | Foreign key | (codProjeto) REFERENCES projeto (cod) ON UPDATE CASCADE ON DELETE CASCADE; |
| usuarioProjeto_codUsuario_fkey | Foreign key | (codUsuario) REFERENCES usuario (cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela ORF

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| codPesquisa | integer | Yes | Yes | | |
| posCodonStart | integer | Yes | Yes | | |
| posCodonStop | integer | Yes | Yes | | |
| linha | integer | Yes | Yes | | |
| codTipoORF | integer | Yes | No | | |
| observacao | varchar(300) | No | No | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| ORF_pkey | Primary key | (codPesquisa, posCodonStart, posCodonStop, linha) |
| ORF_codPesquisa_fkey | Foreign key | (codPesquisa) REFERENCES pesquisa (cod) ON UPDATE CASCADE ON DELETE CASCADE; |
| ORF_codTipoORF_fkey | Foreign key | (codTipoORF) REFERENCES tipoORF(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

Tabela nota

| Name | Data Type | Not Null? | Primary Key? | Default | Comment |
|---|---|---|---|---|---|
| codGabarito | integer | Yes | Yes | | |
| posCodon | integer | Yes | Yes | | |
| nota | integer | Yes | Yes | | |

Constraints

| Name | Type | Definition |
|---|---|---|
| nota_pkey | Primary key | (codGabarito, codCodon, nota) |
| nota_codGabarito_fkey | Foreign key | (codNota) REFERENCES nota(cod) ON UPDATE CASCADE ON DELETE CASCADE; |
| nota_codCodon_fkey | Foreign key | (codCodon) REFERENCES codon(cod) ON UPDATE NO ACTION ON DELETE NO ACTION; |

# ANEXOS

## ANEXO 1- PLANO DE PROJETO