Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

CARLOS EUGENIO DOS SANTOS JUNIOR

INTEGRAÇÃO DE UM APLICATIVO PARA RECONHECIMENTO DE PADRÕES NA SEQUÊNCIA DE DNA COM BANCO DE DADOS XML

Assis, SP
2011

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

CARLOS EUGENIO DOS SANTOS JUNIOR

INTEGRAÇÃO DE UM APLICATIVO PARA RECONHECIMENTO DE PADRÕES NA SEQUÊNCIA DE DNA COM BANCO DE DADOS XML

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Bacharelado em Ciência da Computação.

Orientadora: Profa. Dra. Marisa Atsuko Nitto
Área de Concentração: Informática.

Assis, SP
2011

FICHA CATALOGRÁFICA

dos SANTOS, Carlos Eugenio Junior.

Integração de um Aplicativo para reconhecimento de padrões na sequência de DNA com banco de dados XML / Carlos Eugenio dos Santos Junior. Fundação Educacional do Município de Assis – FEMA – Assis, 2011.

85p.

Orientadora: Profa. Dra. Marisa Atsuko Nitto.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

CDU 001.6

Biblioteca FEMA

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

INTEGRAÇÃO DE UM APLICATIVO PARA RECONHECIMENTO DE PADRÕES NA SEQUÊNCIA DE DNA COM BANCO DE DADOS XML

CARLOS EUGENIO DOS SANTOS JUNIOR

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Bacharelado em Ciências da Computação, analisado pela seguinte comissão examinadora:

Orientadora: Profa. Dra. Marisa Atsuko Nitto
Analisador : Prof. Dr. Luiz Carlos Begosso

Assis, SP
2011

DEDICATÓRIA

Dedico este trabalho aos meus pais, Carlos Eugenio e Wanda, que por diversas vezes em suas vidas priorizaram a minha formação acadêmica, para que eu tivesse condições de concluir este trabalho e também por me apoiarem, incentivarem e valorizarem meus esforços nessa luta.

AGRADECIMENTOS

Primeiramente a **Deus**, pois sem ele nada é possível. Muito obrigado pelo fato de conseguir terminar uma faculdade, e realizar um dos meus maiores sonhos. Obrigado meu Deus pelo presente;

A minha orientadora Profa. Dra. Marisa Atsuko Nitto, pela orientação maravilhosa mostrando sempre onde deveria melhorar para a conquista da formação. Muito obrigado pela paciência, apoio, disponibilidade e seu grande conhecimento no tema da pesquisa;

Á minha irmã, Letícia Aparecida Burlin dos Santos, por estar sempre ao lado, apoiando e incentivando nesta jornada;

Aos meus familiares e amigos pela contínua e ilimitada paciência, companheirismo, estímulo, entusiasmo e compreensão nos inúmeros momentos que nos privamos de horas de lazer junto e por me apoiarem nas horas mais complicadas e difíceis;

Ao meu amigo Antonio Rafael Pepece Junior pelo incentivo e crer na esperança de uma vida melhor;

A Fundação Educacional do Município de Assis por ter-me possibilitado o desenvolvimento deste trabalho;

Aos meus professores do Curso de Ciência da Computação da FEMA, pelos ensinamentos durante o curso;

Aos grandes amigos que conheci no ambiente da instituição de ensino, pelo apoio e companheirismo recebido;

A todos que direta e indiretamente, contribuíram para a realização deste trabalho.

RESUMO

Durante os últimos anos tem havido um crescimento substancial na quantidade de dados biológicos gerados em projetos de sequenciamento de genomas e proteomas de diversas espécies. Tais dados (sequências de DNA – nucleotídeos - e de proteínas - aminoácidos) são produzidos em larga escala e armazenados, sendo inviáveis de serem lidos ou analisados por especialistas através de métodos manuais tradicionais. Por outro lado, sabe-se que grandes quantidades de dados equivalem a um maior potencial de informação. Entretanto, as informações contidas nos dados não estão caracterizadas explicitamente, uma vez que, sendo dados operacionais, não interessam quando estudadas individualmente. Diante deste cenário, surge a necessidade de se explorar estes dados para extrair informação/conhecimento e utilizá-los no âmbito do problema.

Neste projeto será desenvolvido um aplicativo para reconhecimento e busca de padrões na sequencia de DNA integrado a um banco de dados XML. O reconhecimento de padrão será modelado através de autômatos finitos e serão utilizados os algoritmos conhecidos na literatura para fazer a busca de padrões.

Palavras Chave: Bioinformática. Reconhecimento de Padrões. XML.

ABSTRACT

During the past year there has been a substantial increase in the amount of biological data generated in genome sequencing projects and proteomes of several species.Such data (DNA sequences - nucleotide - and protein - amino acids) are mass-produced and stored, and impractical to be read or considered by experts using traditional manual methods. On the other hand, it is known that large amounts of data equals a greater potential for information. However, the information contained in the data is not explicitly characterized since, and operational data, no matter when studied individually. In this scenario, there is a need to exploit these data to extract information / knowledge and use them as part of the problem. This project will be developed an application for recognition and look for patterns in DNA sequences integrated into an XML database. The pattern recognition will bemodeled by finite automata and algorithms used are known in the literature to searchfor patterns.

Keywords: Bioinformatics, Pattern Recognition, XML.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A bioinformática é uma nova ciência que envolve a união de diversas linhas de conhecimento: a engenharia de software, a matemática, a estatística, a ciência da computação e a biologia molecular. Ela apareceu na metade da década de 90, com o surgimento dos sequenciadores automáticos de DNA (ácido desoxirribonucleico), onde houve uma explosão na quantidade de sequências a serem armazenadas, exigindo recursos computacionais cada vez mais eficientes. Além do armazenamento ocorria, paralelamente, a necessidade de análise desses dados, o que tornava indispensável a utilização de plataformas computacionais eficientes para a interpretação dos resultados obtidos (PROSDOCINI ET AL., 2002).

O DNA é um composto orgânico cujas moléculas contêm as instruções genéticas que coordenam o desenvolvimento e funcionamento de todos os seres vivos e alguns vírus. O seu principal papel é armazenar as informações necessárias para a construção das proteínas e RNA (ácido ribonucleico). Os segmentos de DNA que contêm a informação genética são denominados genes e o restante da sequência de DNA tem importância estrutural ou está envolvido na regulação do uso da informação genética (MOUNT, 2001).

A estrutura da molécula de DNA foi descoberta conjuntamente pelo norte-americano James Watson e pelo britânico Francis Crick em sete de março de 1953, o que lhes valeu o Prêmio Nobel de Fisiologia e Medicina em 1962, juntamente com Maurice Wilkins. A Figura 1 mostra a estrutura de um DNA.

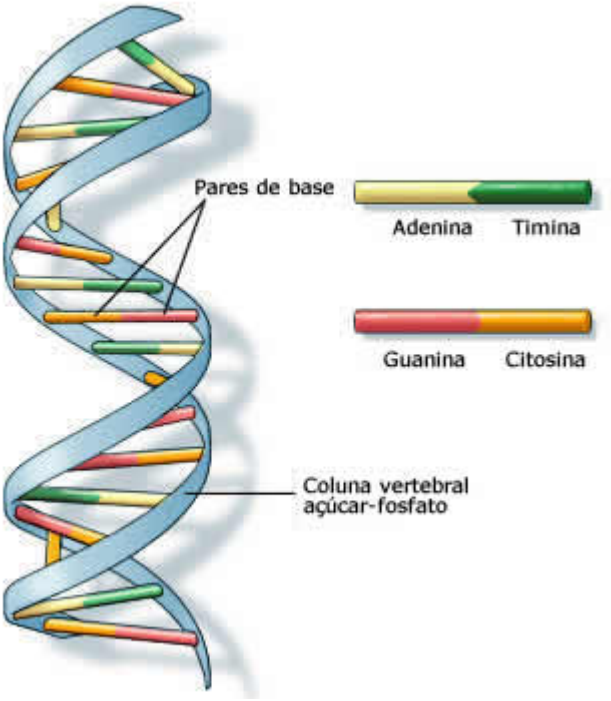

Figura 1 – Estrutura de um DNA. (FREUDENRICH, 2000).

Embora possa parecer complicado, o DNA em uma célula é um padrão feito de quatro partes diferentes, chamadas nucleotídeos. Imagine um conjunto de blocos que possui somente quatro formas, ou um alfabeto com apenas quatro letras. O DNA é uma longa fileira desses blocos ou letras. As quatro bases no alfabeto do DNA são:

- **adenina (A)** - uma purina;
- **citosina (C)** - uma pirimidina;
- **guanina (G)** - uma purina;
- **timina (T)** - uma pirimidina.

A bioinformática é imprescindível para a manipulação dos dados biológicos. Ela pode ser definida como uma modalidade que abrange todos os aspectos de aquisição, processamento, armazenamento, distribuição, análise e interpretação da informação biológica. Através da combinação de procedimentos e técnicas da matemática, estatística e ciência da computação são elaboradas várias ferramentas

que nos auxiliam a compreender o significado biológico representado nos dados gênicos. Além disso, através da criação de bancos de dados com as informações já processadas, acelera a investigação em outras áreas como a medicina, a biotecnologia, a agronomia, entre outras (BORÉM; SANTOS, 2001). Os métodos computacionais utilizados por pesquisadores dessa nova área são: uso de banco de dados públicos e formatos de dados, alinhamento e busca de sequência, predição de genes, alinhamento múltiplo de sequências, análise filogenética, análise da sequência de proteínas, análise das propriedades da estrutura protéica e análise de *microarrays* de DNA, entre outros (GIBAS; JAMBECK, 2002).

O banco de dados em bioinformatica é uma questão muito importante. Alguns dos bancos de dados biológicos utilizam sistemas baseados no modelo relacional, sistemas orientados a objetos ou ainda alguns gerenciadores específicos. O grande desafio é encontrar a melhor forma de armazenamento e de pesquisa para os dados gerados por projetos de pesquisa na área da bioinformática, (WIECZOREK e LEAL, 2003). O Projeto Genoma Humano, por exemplo, engloba três bilhões de pares de bases para cada indivíduo, onde cada base contém centenas de *gigabytes* com crescimento significativo e outros possuem dezenas de *terabytes*, e com o aumento das informações esses números tendem a crescer. Para tanto, surge a necessidade de se possuir formas de armazenamento, acesso e pesquisa sobre tais dados, para que se consiga trazer a informação da melhor maneira possível, devendo existir assim, técnicas diferenciadas para o tratamento destes dados, que são grandes cadeias de DNA (em banco de dados, grandes cadeias de caracteres).

A grande maioria dos bancos de dados é atrelada a um sistema denominado SGBD (Sistema de Gerenciamento de Banco de Dados). Este sistema é responsável por intermediar os processos de definição, manipulação, construção e administração do banco de dados solicitados pelos usuários ou por outras aplicações, facilitando a execução destes processos. Entretanto, um dos problemas atuais desta área ocorre devido à inexistência de um SGBD específico para aplicações de Bioinformática. A maioria das ferramentas criadas acessam dados diretamente de arquivos textos ou binários, sem a utilização de um SGBD, o que os impede de beneficiar-se de mecanismos eficazes de armazenamento, acesso eficiente a disco e gerenciamento inteligente da memória, entre outros (LIFSCHITZ, 2006).

A modelagem em XML (*eXtensible Markup Language*) trouxe novo alento para a área de bioinformática, além de ser adequada à representação de conteúdos e constituir padrão para a troca de informação, permite a manipulação dos dados através de consultas e modificações com linguagens apropriadas (RAMALHO; HENRIQUES, 2002). A modelagem através de *XML Schema* é uma abordagem nova, que sinaliza para a possibilidade de uma padronização da informação genômica. Além disso, ela também otimiza o tratamento por parte de software que manipula esse tipo de dado (WALMSLEY, 2001). Esta linguagem foi utilizada com um grande êxito no armazenamento dos dados biológicos como as sequências de DNA, RNA e de proteínas. Com esta linguagem pode-se gerar um documento contendo todos os dados relacionados à sequência de bases de aminoácidos, e também todo o processo de obtenção e de outros dados de suma importância para os pesquisadores. Outra utilização importante está na troca das informações entre os laboratórios e pesquisadores, que utilizam a Internet como meio de troca pela facilidade e com o grande suporte que ela vem recebendo.

Será desenvolvido, neste projeto de pesquisa, um aplicativo de reconhecimento de padrões na sequência de DNA integrado a um banco de dados XML. O reconhecimento de padrão será modelado através de autômatos finitos (SISPER, 2007; LEWIS e PAPADIMITRIOU, 2004; DIVERIO e MENEZES, 1999; HOPCROFT ET AL., 2002). A escolha em trabalhar com este tipo de banco é que os pesquisadores da área têm apostado no banco de dados XML para resolver problemas de padronização de dados, pois as sequências são oriundas de diferentes bancos públicos ou privados. A implementação do aplicativo de reconhecimento da sequencia de DNA e de reconhecimento de padrão serão feitas em Java (DEITEL, 2003; POTTS e FRIEDEL JR, 2004; LEMAY e PERKINS, 1996).

Um dos pontos relevantes no desenvolvimento deste projeto foi a integração desses aplicativos com um banco de dados XML, pois foi possível estabelecer um padrão nos dados armazenados. Este padrão XML é muito interessante por permitir aos pesquisadores da área uma troca de informação pela *web* mais facilmente.

1.1 - Objetivos

O objetivo principal, neste projeto de pesquisa, é desenvolver métodos de reconhecimento de padrão, utilizando os conceitos de autômatos finitos. Neste projeto serão utilizadas técnicas e métodos amplamente divulgados na literatura, nas quais tem a finalidade de reconhecer caracteres na sequência de DNA. Aquisição de conhecimentos sobre as tecnologias Java e XML para o desenvolvimento do aplicativo integrado ao banco de dados. Será desenvolvido um aplicativo em Java, que seja capaz de reconhecer padrões em uma sequência de DNA e fazer a integração de um arquivo texto com informações relevantes, ou seja informações que caracterizam um DNA, armazenando-as em um Banco de Dados XML. A utilização de XML é importante no sentido da padronização da informação genômica. Esta padronização também facilita a troca de informacão pela *internet* devido ao suporte que ela vem recebendo.

1.2 - Justificativa

O desenvolvimento deste projeto tem grande relevância, tendo em vista, que a bioinformática veio para revolucionar a área biológica, com o desenvolvimento de ferramentas computacionais, para auxiliar os profissionais da área nas pesquisas e as análises das sequências atribuídas a cada indivíduo. Os computadores são agora parte integrante do mundo biológico e sem elas, os avanços na biologia e na medicina seriam, sem dúvida, impossíveis. As ferramentas computacionais aceleram os processos de análise de dados biológicos e as descobertas de informações biológicas desconhecidas. Estas informações podem ser ferramentas úteis em avanços terapêuticos no prolongamento e melhoramento da qualidade de vida. E o desenvolvimento do aplicativo de busca de padrões na sequência de DNA integrado com o banco de dados XML vem atender as necessidades da área, tendo em vista o crescimento exponencial dos dados biológicos.

1.3 – Motivação

A motivação para a escolha do tema foi pelo fato de bioinformática ser uma área nova e multidisciplinar. A área de bioinformática necessita de inúmeras ferramentas computacionais para auxiliar os profissionais na análise de dados, manipulação de dados e um banco de dados eficiente para armazenar os dados. Por isso o interesse em desenvolver aplicativos nesta área. E utilizar os conhecimentos adquiridos para o desenvolvimento de novos aplicativos e atuar na área futuramente.

1.4 – Estruturas do Trabalho

1. Introdução
2. Fundamentação Teórica Básica
3. Desenvolvimento do Projeto
4. Conclusão.

2. FUNDAMENTAÇÃO TEÓRICA BÁSICA

Neste capítulo será feita a fundamentação teórica básicas das tecnologias utilizadas para desenvolver o aplicativo. Será feita uma descrição sobre reconhecimento de padrões e os algoritmos utilizados para a busca de padrões. Serão apresentados os conceitos de autômatos finitos, tecnologias Java e banco de dados.

2.1 – Reconhecimentos de Padrões

Os seres humanos são capazes de reconhecer padrões com grande rapidez, pois faz parte da natureza humana. Ao observar um objeto é possível fazer uma coleta de informações, as quais são comparadas com as propriedades e comportamentos conhecidos e armazenados em sua mente. Através dessa comparação os seres humanos são capazes de reconhecer o alvo de sua observação.

As iniciativas para o reconhecimento de padrões artificialmente são divididas em duas categorias: de reconhecimento de itens concretos e abstratos (GIBSON, 2008). O reconhecimento de itens concretos envolve o reconhecimento de impressões digitais, assinaturas, objetos físicos, formas de ondas, voz, faces, enfim, o reconhecimento de itens que existem concretamente. Os elementos abstratos seriam itens sem forma física como, por exemplo, a solução para um determinado problema.

O reconhecimento de padrões envolve três níveis de processamento: filtragem da entrada, extração de características e classificação (GIBSON, 2008). Assim, geralmente os grandes desafios são encontrados na escolha de técnicas para efetuar esses três aspectos.

A filtragem da entrada de dados tem o objetivo de eliminar dados desnecessários ou distorcidos fazendo com que a entrada apresente apenas dados relevantes para o reconhecimento do objeto em análise. A extração de características consiste da análise dos dados de entrada a fim de extrair e derivar informações úteis para o processo de reconhecimento. O estágio final do reconhecimento de padrões é a

classificação, onde através da análise das características da entrada de dados o objeto em análise é declarado como pertencente a uma determinada categoria.

Em geral, acredita-se que um problema de reconhecimento de padrões bem definido e restrito (com pequenas variações intra-classe e grandes variações inter-classes) permitirá uma representação compacta dos padrões e uma estratégia de decisão simples. Mas nem sempre os padrões a serem reconhecidos possuem essas características. Nesse fato reside a importância de algoritmos de extração e seleção de características. As quatro abordagens mais bem conhecidas de reconhecimento de padrões são: casamento (*template matching*), abordagem estatística, sintática e redes neurais (JAIN ET AL, 2002).

A Tabela 1 mostra algumas aplicações que utilizam Reconhecimento de Padrões.

Tabela 1 - Aplicações do Reconhecimento de Padrões (JAIN ET AL, 2002)

Domínio do Problema	**Aplicação**	**Padrão de Entrada**	**Classes de Padrão**
Bioinformática	Análise de Sequência	DNA / Sequência de Proteínas	Tipos conhecidos de genes/padrões
Mineração de Dados	Busca por Padrões significantes	Pontos em um espaço Multidimensional	Compactar e bem separar grupos
Classificação de Documentos	Busca na Internet	Documento Texto	Categorias semânticas (negócios, esportes e etc.)
Análise de Documentos de Imagem	Máquinas de Leitura para Cego	Intensidade ou alcance de Imagem	Natureza do produto defeituoso ou não
Automação Industrial	Inspeção de Circuito impresso em Placas	Intensidade ou alcance de Imagem	Natureza do produto defeituoso
Recuperação de Base de Dados Multimídia	Busca na Internet	Vídeo Clipe	Gêneros de vídeos
Reconhecimento Biométrico	Identificação Pessoal	Face, íris, impressão digital	Usuários autorizados para controle de acesso
Sensoriamento Remoto	Prognostico da produção de colheita	Imagem multiespectral	Categorias de Aproveitamento de terra, desenvolvimento de padrões de colheita
Reconhecimento de Voz	Inquérito por telefone sem assistência de operador	Voz em forma de onda	Palavras faladas

2.1.1 – Algoritmos de Reconhecimento de Padrões

Nesta seção serão apresentados alguns algoritmos conhecidos na literatura e que são responsáveis para a realização do reconhecimento dos padrões. Esses algoritmos são utilizados justamente para encontrar cadeias de caracteres conhecidas como padrão em um determinado arquivo, sequência ou texto.
A utilização dos algoritmos é mais comum em análise de texto (dados biológicos armazenados em arquivos texto), pois os mesmos são armazenados em forma linear, ou seja, as mesmas são descritas como sequência de nucleotídeos ou aminoácidos, as mesmas são cadeias de caractere muito longas. Serão apresentados cinco algoritmos sendo:

- Algoritmo da Força Bruta ou Ingênuo;
- Algoritmo de Knuth-Morris-Pratt;
- Algoritmo de Boyer e Moore;
- Algoritmo de Ukkonen.

2.1.1.1 – Algoritmo de Força Bruta ou Ingênuo

O Algoritmo de Força Bruta tem característica de percorrer toda a cadeia de caractere lido, buscando o padrão desejado, porém a mesma realiza essa busca através de todas as posições no texto entre 0 e n-m. Este processo é conhecido como deslocamento de “janelas”, isto é, no texto lido a comparação é feita através de sequências de caractere parecendo pequenas janelas contendo as sequências desejáveis. A figura 2 mostra o funcionamento do algoritmo de força bruta.

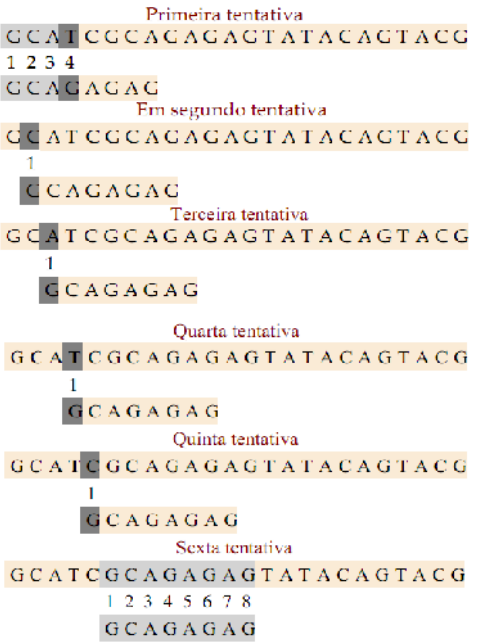

Figura 2 - Funcionamento do algoritmo de força bruta

O que se pode verificar é que a busca é realizado caractere por caractere até localizar o padrão que procura.

Algoritmo da Força Bruta

```
forcaBruta( char *T, char  *P ){
        n = strlen(T);
        m = strlen(P);
        para ( i=0; i < n-m; i++ ) {
                k=i; cont=1;
                para (j = 0; j < m; j++){
                se ( T[k] == P[j] and (cont != m) ){
                        cont++;
                        k++;
                        }
                        se (cont==m)
                                Return (1);
                }
                Return(0);
        }
}
```

2.1.1.2 – Algoritmo de Knuth-Morris-Pratt

A idéia central desse algoritmo é aproveitar os caracteres reconhecidos nas sequencias em que o mesmo passou conseguindo maior deslocamento do padrão lido. Ele utiliza do pré-processamento do padrão lido no que resulta em uma tabela, essa tabela é consultada quando há uma colisão. A função da tabela, na verdade, seria para determinar a maior parte aproveitável do texto genético e com isso aproveitar várias comparações. Este algoritmo foi proposto para resolver o seguinte problema: dada duas strings X e Y encontrar se existir ocorrência entre Y e X. Passando para formula matemática podemos atribuir para que todo *i, 1<= i <= n tem-se* $A_{k+1} = B_i$.

Algoritmo de KMP

```
Kmp( string A,B )
Calcula_next( B,m )
        Enquanto start=0 e i<m faz
                Se B[j]=A[i] então
                        i = i +1;
                        j= j + 1;
                Senão j=next[j]+1
                        Se j=0 então
                                j = 1;
                                i = i + 1;
                se j = m + 1 entao
                        start = i –m;
                retorna start;
```

Algoritmo da criação da Tabela de índices

```
Calcula_next (B,m)
        next[1] = -1;
        next[2]=0;
        para cada i de 3 até m faz
                j = next[i-1]+1;
                enquanto B[i-1] != B[j] e j>0
                        j = next[j] +1;
                next[i] = j;
        retorna next;
```

O algoritmo Knuth-Morris-Pratt representa o algoritmo principal em si, onde o índice do texto está sendo representado pela letra i, o índice da palavra está sendo representado pela letra j, o tamanho da palavra é representado pela letra m e o processo da posição inicial está utilizado com a palavra start. Enquanto o algoritmo da criação da tabela de índices está relacionado ao índice da palavra está sendo representado por i, o tamanho está representado por m, e o índice auxiliar está sendo representado por j.

2.1.1.3 – Algoritmo de Boyer-Moore

Este algoritmo tende fazer comparações da direita para esquerda, ou seja, o algoritmo posiciona o padrão genético mais a esquerda do texto genético fazendo uma verificação da direita para a esquerda, caso a comparação do padrão não tiver nenhuma diferença isto indica que foi localizado o padrão, do contrário ocorrerá uma mudança na posição do padrão genético, porém essa mudança está baseada em duas heurísticas. Essas heurísticas evitam que o algoritmo faça comparações desnecessárias referente a busca de padrões.

As Heurísticas se dão por Heurística-do-Bom-Sufixo e Heurística-do-Mau-caractere. A Heurística-do-Mau-Caractere funciona quando se tem um erro no padrão e ele usa a informação do mau caractere para propor uma nova mudança. Já a Heurística-do-

Bom-Sufixo é quando se encontra um padrão diferente no texto podendo avançar várias posições.

Algoritmo Boyer-Moore

```
c=ultima_ocorrencia(Y);
r= bom_sufixo(Y);
i = 1;
while(i<=n-m+1){
        j=m;
        while(x_{i+j-1} = y_j e j>0){
                j=j-1
        }
        if(j=0){
                S = S U {i}
                i = i + R_0
        }
        else
                i = max{ R_j, j-Cxi+j-1 }
}
```

2.1.1.4 – Algoritmo de Ukkonen

O Algoritmo de Ukkonen é baseado no algoritmo clássico da programação dinâmica. A simples metodologia do algoritmo da programação dinâmica tem o objetivo o cálculo de uma forma matricial que é responsável pelo armazenamento dos custos de transformação de subsequências de uma cadeia de subsequência de outra cadeia (HARADA, 1994). A principal contribuição do algoritmo é a diminuição do número de elementos a serem calculados na matriz de programação dinâmica.

Este algoritmo apresenta três princípios de movimentação de dados na matriz que são:

- Inserção;
- Remoção;
- Substituição.

No processo de inserção na matriz referenciada pode se dizer que o processo gasta um custo de valor 2 não só na inserção mas também na remoção pode se dizer que o custo é o mesmo, em uma substituição o custo é de 3 (isso se o caractere for diferente), caso o caractere seja igual tem se o valor 0. A figura 3 mostra a matriz com grafo de dependência.

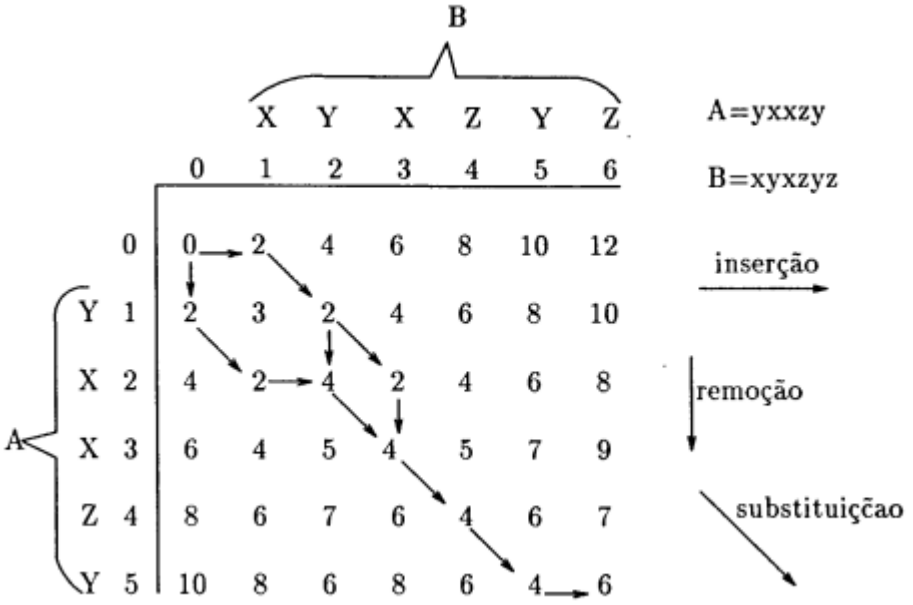

Figura 3 - Matriz D(i,j) com grafo de dependência e legenda de arco (HARRADA,1994).

2.2 – Autômatos

O conceito de autômato é bem simples e fácil de entender, e pode ser definido como um modelo simplificado de um computador, ou seja, cada autômato tem sua configuração e sua própria definição. Os mesmos podem ser descrito como um grafo sendo que cada nó são os estados e as arestas transições entre estados. Os autômatos podem ser divididos em várias partes e elas são:

- Forma de Entrada de Dados;
- Forma de Saída de Dados;
- Estados Internos;
- Ter ou Não forma de armazenamento de Dados;
- Pode adquirir forma rígida e não Programável.

2.2.1 – Autômato Finito Determinístico (AFD)

Um Autômato Finito Determinístico (AFD) pode ser considerado como uma máquina de estado, pois para cada par de estados e símbolo de entrada, existe consecutivamente um próximo estado determinístico. A definição de um Autômato Finito Determinístico, sobre um alfabeto S é um sistema (K,Σ, δ, i, F) onde:

K é um conjunto de estados finito, não vazio;
Σ é um alfabeto de entrada (finito);
δ: K x Σ é a função de transição;
i Є K é o estado inicial;
F ⊆ K é o conjunto de estados finais.

Neste autômato se aceita uma cadeia partindo do estado inicial e mudando de estado conforme a função de transição, atingindo o estado final.

A figura 4 mostra a representação de um AFD.

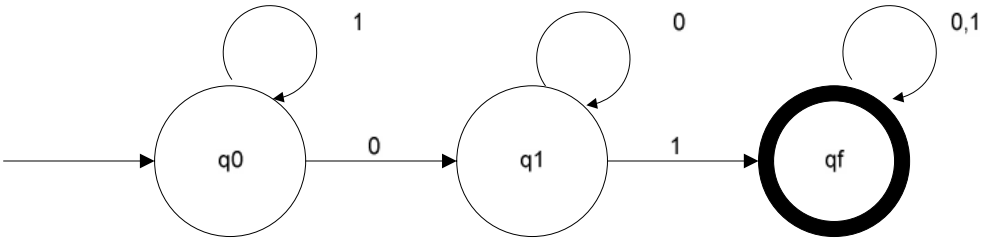

Figura 4 - Representação de um AFD

2.2.2 – Autômato Finito Não Determinístico (AFND)

A diferença entre um Autômato Finito Determinístico (AFD) e um Autômato Finito não Determinístico (AFND) é bem simples, pois no AFD a função de transição determina em qual estado da cadeia iria levar, porém em um AFND isso não acontece, pois a função de transição fornece uma lista de conjunto (estados) possível para a próxima transição, esta lista pode ser vazia ou ter número de transições. Com isso a possibilidade de escolha para uma transição é bem vasta, observando que um AFND escolhe e advinha o caminho de aceitação, com isso a escolha de um caminho errado que não levam ao estado final é bem irrelevante.
A definição de um Autômato Finito não Determinístico M, sobre um alfabeto S é um sistema (K, Σ, δ, i, F), onde:

K é um conjunto (finito, não vazio) de estados,
Σ é um alfabeto de entrada (finito),
δ: K x (Σ U { Є }) → P(K) é a função de transição,
i Є K é o estado inicial,
F⊆ K é o conjunto de estados finais.

A figura 5 mostra a representação de um AFND.

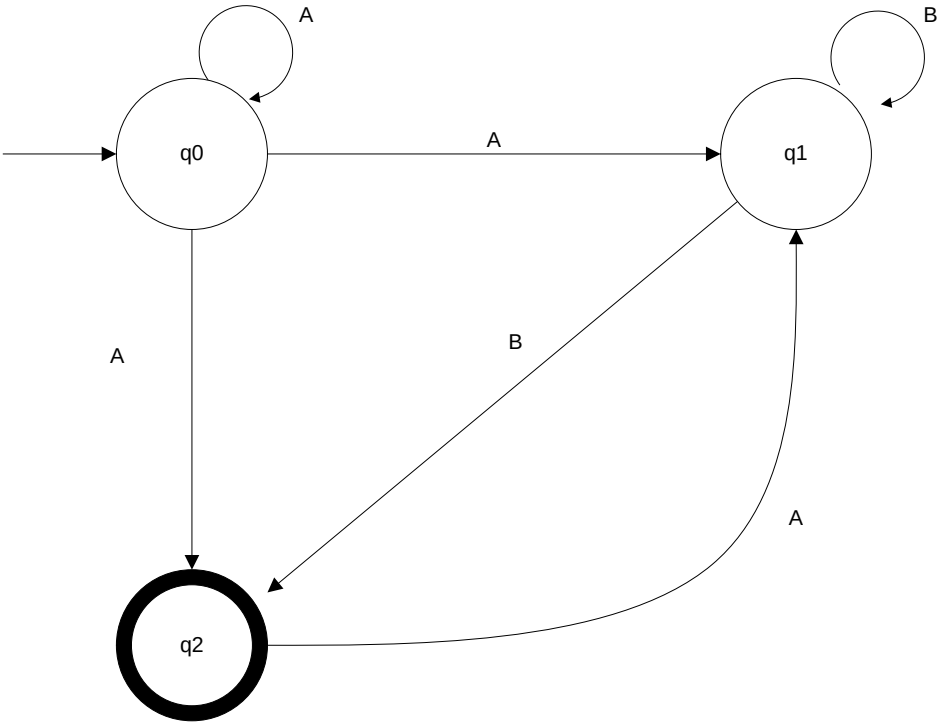

Figura 5 - Representação de um AFND

2.3 – Tecnologias Java

Neste projeto será utilizada a tecnologia Java mais adequada para o desenvolvimento do aplicativo para reconhecimento e busca de padrões.

Java é uma linguagem de programação orientada a objetos, desenvolvida por uma pequena equipe de pessoas na *Sun Microsystems*. Inicialmente elaborada para ser a linguagem-base de projetos de software para produtos eletrônicos, Java teve seu grande *boom* em 1995, devido ao sucesso mundial da *World Wide Web* (WWW).

Com o novo ânimo trazido pelo advento da WWW, a equipe da Sun desenvolveu um *browser* totalmente escrito em Java, tendo-o terminado no início de 1995 e denominado-o *HotJava*. O grande diferencial de *HotJava* para outros *browsers* da época (como o Mosaic, o Netscape Navigator e o Lynx) é que ele permitia a inserção

de programas escritos em Java dentro de páginas HTML comuns. *HotJava* como *browser* foi um fiasco comercial, mas abriu os olhos dos desenvolvedores para um fato muito importante: as páginas HTML estariam fadadas a serem estáticas e sem ações embutidas em si, não houvesse uma linguagem padrão na qual fossem escritos programas que pudessem ser embutidos nas páginas *Web*. *HotJava* demonstrou que isso era possível (ou seja, incluir um programa, no caso escrito em Java, em uma página HTML rodando em um browser preparado para dar suporte à execução do programa, no caso o próprio *HotJava*). A grande sacada de Java veio logo a seguir, quando a Netscape anunciou que sua próxima versão do *browser Navigator*, iria dar suporte a aplicativos Java embutidos em documentos HTML. Em seguida, a Microsoft anunciou o mesmo para o seu Internet Explorer. E Java estourou no mundo, a Sun contabilizava inúmeros *downloads* de seu JDK, diversas empresas desenvolveram IDEs para a programação em Java, e vieram *JavaScript*, *JavaBeans*, a briga deste com *ActiveX*.

De 1998 até a hoje Java desenvolveu-se e tornou-se um dos maiores repositórios de projetos livres do mundo o famoso Java.net. Em 1999 surgiu a plataforma de desenvolvimento e distribuição corporativa e outra distribuição para dispositivos móveis.

2.4 – Banco de Dados Biológicos

O mapeamento do genoma humano e de outros organismos gera diariamente um elevado volume de informações que são sistematicamente armazenadas em bancos de dados computacionais, sendo estas informações fontes de estudo para a biologia e medicina através da bioinformática. Com o avanço das tecnologias a quantidade de dados gerada pelos laboratórios aumentou drasticamente, e as informações coletadas pelos experimentos utilizando genes do DNA não puderam mais ser armazenadas desta maneira “arcaica” (CRITCHLOW; MUSICK; SLEZAK, 2000). A molécula do DNA Humano pode conter três bilhões de caracteres e entre eles, os 100 mil genes estimados para a espécie *Homo Sapiens* (COSTA, 2000).

Para contornar tal dificuldade, devem ser implementados, bancos de dados que disponibilizem, de modo confiável, os dados e ferramentas de análise. Em muitos casos, esses bancos são abertos, o que aumenta ainda mais a aplicabilidade da pesquisa (FÉLIX, 2000).

Um problema a ser superado, quando se fala em banco de dados para bioinformática, é que bancos de dados têm sido utilizados em grande parte para administrar dados empresariais, números simples ou datas. Poucos bancos de dados tiveram uma habilidade nativa para lidar com dados complexos, como dados multimídia, texto, dados espaciais, ou dados genéticos (sucessão de genes). A maioria destes dados é difícil de serem controlados, como questões de achar a semelhança (em grandes cadeias de caracteres), questões sobre sucessões de gene e questões de localização de genes em cadeias de DNA.

A tabela 2 mostra alguns bancos de dados que permitem trabalhar com sucessões genômicas, sendo a maioria de institutos e universidades, que vêm trabalhando na elaboração de bancos de dados específicos para trabalhar com expressões gênicas.

Tabela 2 - Bancos de Dados com capacidade de armazenar e buscar dados genômicos (WIECZOREK e LEAL, 2002)

Banco de Dados	**Instituto/Empresa**
NIH - Banco de dados de expressão gênica	Molecular Pharmacology of Cancer
SMD - Banco de Dados de Microarrays	Stanford University
YMGV - Visão global sobre Microarray de levedura	http://www.transcriptome.ens.fr/ymgv/
Oracle 8*i*/9*i* – Banco de dados comercial	Oracle Corporation

A Oracle (ORACLE CORP., 1999) apresenta uma proposta interessante para a solução dos problemas de banco de dados em bioinformática: devem ser elaborados

bancos de dados que sejam capazes de controlar tipos complexos, de modo a conseguir suprir as necessidades do domínio da aplicação, além de prover apoio a qualquer tipo de dado definido pelo usuário, ou seja, um banco de dados extensível. Este banco de dados extensível dará apoio às necessidades do sistema para definir tipos de dados novos que sejam capazes de criar entidades de domínio como sucessões genotípicas; uso de operadores definidos pelo usuário; indexação de domínio específico, fornecendo apoio para índices específicos de dados genômicos e otimizar a extensibilidade fazendo assim uma ordenação inteligente dos predicados em questão, envolvendo tipos de dados definidos pelo usuário.

Outra abordagem pode ser através de sistemas envolvendo *Data Warehouses* (Armazéns de Dados), pois estes são utilizados pela indústria há muitos anos, e como demonstrado pela figura 6, são constituídos tipicamente de cinco camadas: as fontes de dados, que contém os dados a serem integrados (adicionados) ao *Data Warehouse* através dos Wrapper's (analisadores gramaticais de dados), os mediadores (que traduzem os dados para a representação do *Data Warehouse*), o próprio *Data Warehouse*, que é um grande repositório de dados, geralmente um banco de dados relacional, que apresenta uma visão consistente dos dados provenientes das fontes de dados, e finalmente os usuários, que interagem com o sistema através de uma interface (CRITCHLOW; MUSICK e SLEZAK, 2000). A figura 6 mostra a estrutura de um *Data Warehouse*.

Figura 6 - Estrutura de um Data Warehouse (WIECZOREK e LEAL, 2002)

Segundo (CRITCHLOW; MUSICK; SLEZAK, 2000), o desafio para a criação de um *Data Warehouse* para o ambiente da bioinformática está no fato de que deve-se desenvolver uma infra estrutura flexível o bastante para controlar a natureza dinâmica do domínio, pois fontes de dados para aplicações científicas são extremamente dinâmicas.

Não tem como falar de Banco de Dados Biológicos sem mencionar o Principal Banco Genético do Mundo o *GenBank*. Esse banco iniciou-se com o objetivo de fazer mapeamento das mutações que ocorriam nos seres humanos. O Banco iniciou-se no ano de 1982 contém aproximadamente 126.551.501.141 bases em 135.440.924 registros e divisões tradicionais de 191401393188. Sendo que o banco tem uma coleção anotada de todas as sequências de DNA. Para validar um bom Banco de Dados Biológico tem que ter:

- Qualidade dos dados;
- Fácil acesso as informações;
- Integração;
- Anotações consistentes;
- Mecanismos para extrair do conjunto de dados apenas informações que seja útil e que seja do interesse do pesquisador.

Além de concentrar informações sobre o código genético de milhares de seres vivos, o NCBI (Centro Nacional de Informação Biotecnológica) fornece uma série de ferramentas para trabalhar com essas informações.

Uma terceira abordagem seria a utilização de bases de dados XML, pois recentemente alguns esforços estão sendo dedicados para a construção de documentos de definição XML que permitem conversões entre bancos de dados que se utilizam de diferentes tecnologias de XML (SHUI, 2002).

Existem muitos projetos em andamento que provêem bibliotecas de repositório de dados em muitas linguagens, como Java e C/C++. Porém, muitos destes projetos estão preocupados em como analisar gramaticalmente os dados XML, ao invés de estabelecer um banco de dados XML bem formulado, capaz de integrar bancos de

dados diferentes, criando assim um repositório de informação biológica. A grande preocupação, neste caso, é de como integrar estas diversas bases de dados XML, visto que os dados biológicos não possuem uma estrutura padrão, pois os dados podem variar de tipo de uma base para outra. É importante salientar que neste caso, por se tratar de um tema novo no Brasil, a maioria do material encontrado para a consulta e pesquisa foram artigos científicos escritos em inglês, com poucos materiais específicos do tema em português.

2.4 – Banco Biológico GenBank.

As sequências de DNA estão sempre depositadas em forma de fita única, mesmo que seja um fragmento de DNA. Isso porque quando o DNA é transcrito para a forma de RNA apenas uma das fitas é transcrita, a que chamamos de fita senso. A outra fita complementar a molécula de DNA, chamada de fita anti-senso não é transcrita para a forma de RNA. Deste modo, apenas a fita senso de DNA é depositada no banco de dados. É claro que isso é válido para os casos onde o gene é conhecido. Quando não se conhece o gene, qualquer uma das fitas pode ser a fita depositada. No entanto, serão trabalhados apenas os genes conhecidos. O primeiro ponto que se tem que ter em mente é que a fita de DNA depositada é a fita senso, que é transcrita para a forma de RNA (PROSDOCIMI, 2001).

O segundo ponto que você deve saber é que a molécula de RNA mensageiro, apesar de, na realidade, corresponder à fita complementar do DNA, não está depositada no Genbank desta forma. Ao invés disso, o RNA mensageiro depositado no GenBank é semelhante à fita senso de DNA. A diferença entre a fita senso de DNA e fita de RNA mensageiro depositadas no GenBank constitui basicamente no fato dos *exons* da molécula de DNA, pois os *introns* são removidos durante o processo de edição do RNA. Deste modo o RNA mensageiro é sempre menor que a fita senso de DNA.

A figura 7 mostra este processo pra melhor entendimento de como as informações são armazenadas no Genbank.

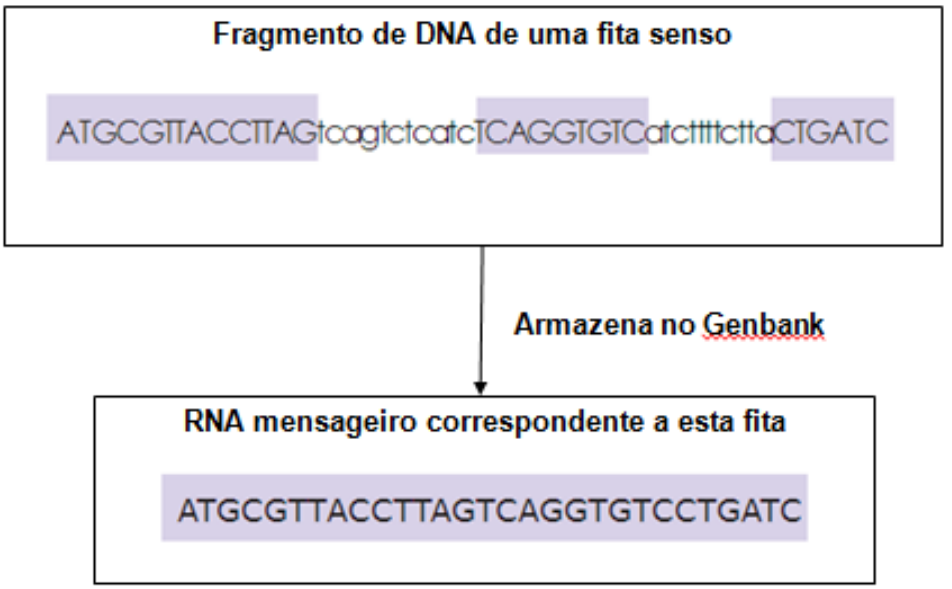

Figura 7 – Processo de transcrição de uma fita senso.

As letras em maiúsculo correspondem aos *exons* de um determinado gene e as letras em minúsculo correspondem aos *introns*. É importante salientar as seguintes observações:

- O RNA mensageiro constrói apenas os *exons*, a fita original de DNA;
- O RNA mensageiro não está na forma de fita complementar;
- O RNA mensageiro não está depositado com suas bases de uracil (que é o que na realidade ocorre), mas sim com as bases de timina.

A explicação para este fato está na síntese de DNA complementar, ou cDNA. A maioria das seqüências de RNA que estão no GenBank foram obtidas através da síntese de DNA complementar. O cDNA é uma molécula similar ao DNA porém esta é sintetizada no laboratório. A síntese de cDNA é realizada com a transação reversa, uma enzima de origem viral que tem a capacidade de produzir uma molécula de DNA dupla fita a partir da cópia de uma molécula de RNA. Como o nome diz, a transação reversa faz o caminho contrário daquele percorrido pela RNA polimerase, que a partir de uma molécula dupla fita de DNA produz RNA (PROSDOCINI, 2001).

A molécula de DNA produzida pela transação reversa é chamada de DNA

complementar (ou cDNA) e é formada pelos mesmos compostos encontrados na molécula de DNA (açúcar desoxirribose, grupos fosfato, ponte de hidrogênio e as 4 bases nitrogenadas, adenina, timina, guanina e citosina). A descoberta desta enzima viral e de seu mecanismo de ação abriu um importante caminho na biologia molecular. A molécula de RNA é extremamente instável e dependendo do RNA em questão o número de cópias pode estar muito reduzido. Deste modo, a possibilidade em se trabalhar com moléculas de cDNA ao invés de RNA tem facilitado muito o trabalho do cientista no laboratório (PROSDOCIMI, 2001).

A figura 8 mostra a transação reversa de uma molécula de DNA.

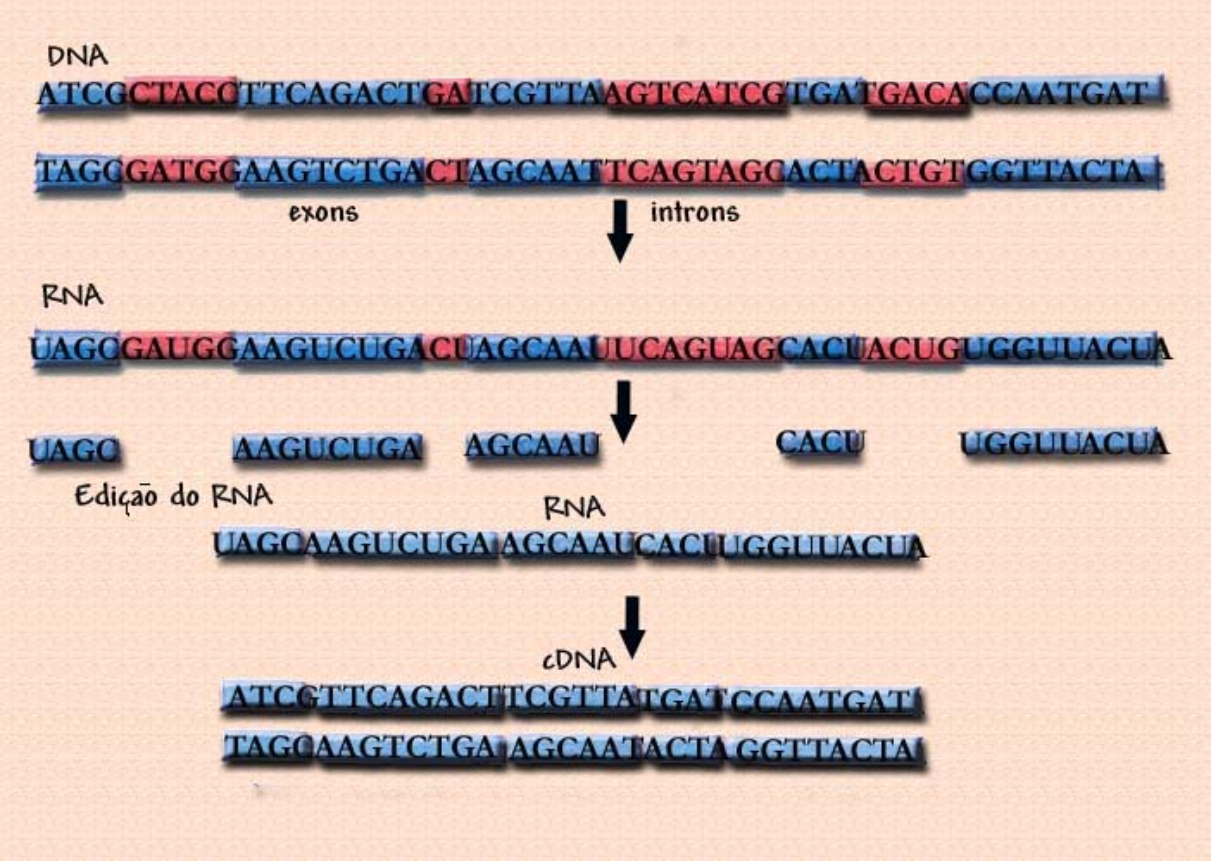

Figura 8 – Transação reversa de uma molécula de DNA (PROSDOCIMI, 2001)

O cDNA é uma molécula estável, de fácil manuseio e sua multiplicação é bastante simples.

2.5 – Oracle Berkeley DB XML.

Oracle Berkeley DB XML é um banco de dados XML escrito em C++ que esta sobre a camada de outro banco de dados o Oracle Berkeley DB. Sua principal função é fornecer suporte a consultas, inserções e manipulação dos arquivos XML armazenados nele, usando recursos como XQuery, XPath, índices, validação dos arquivos, controle das transações e até mesmo replicação.

A grande vantagem do Oracle Berkeley DB XML é que ela pode trabalhar isoladamente da camada principal do banco de dados do Oracle Berkeley DB. Isso é possível porque o Oracle Berkeley DB XML é um banco de dados embutido que pode trabalhar diretamente nas aplicações. Ao mencionar embutido, isto quer dizer que não é necessário nenhum processo de servidor (*daemon*) para iniciar ou parar seus serviços, tudo pode ser acessado através de API (*Application Programming Interface*) própria. A figura 9 mostra a arquitetura do Oracle Berkeley.

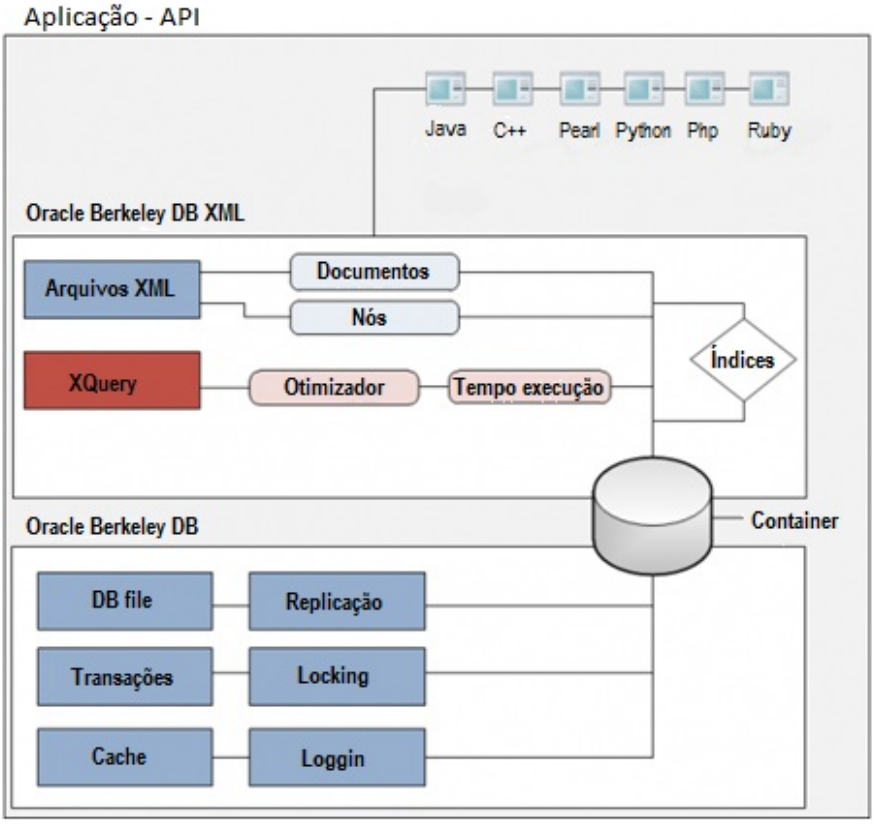

Figura 9 – Arquitetura do Oracle Berkeley (ALMEIDA, 2011)

O importante é não confundir um banco de dados relacional com um banco de dados embutido, que é o caso do Oracle Berkeley DB XML, porque não estão presentes em sua arquitetura, requisitos básicos como:

- Modelo de dados relacional;
- Padrões SQL (ANSI, ISO, SQL-1992, SQL:1999, SQL:2003 e etc);
- Serviços de administração do banco de dados (*daemons*);
- E extensões processuais, tais como PL/SQL, T-SQL, SQL PL e etc.

Desse modo, o Oracle Berkeley DB XML torna-se um banco de dados embutido bem simples, de fácil utilização e que tem como objetivo principal, manipular e gerenciar os arquivos XML.

2.5.1 – Armazenamento de Documento XML

O Berkeley DB XML permite um armazenamento eficiente de documentos XML, indexação de dados flexível e rápido acesso utilizando XQuery ou Xpath. De acordo com (GEOFF, 2005 e FALCÃO, 2006) as principais características do armazenamento de documentos do banco de dados Berkeley DB XML são:

- Construído sobre o Berkeley BD, ele herdou todas as suas características, incluindo transações/recuperação ACID e replicação para uma alta disponibilidade;
- Armazenamento nativo e recuperação de dados XML e não-XML dentro de uma mesma transação, sem haver necessidade de um mapeamento ou uma translação;
- Controle de armazenamento flexível de nós ou documento inteiro;
- Agrupamento lógico de documentos;

- Esquema e método de validação por documento;
- Chave com valor para suporte de meta-dados;
- Suporte a namespace de XML;
- XQuery com depuração no suporte.

2.5.2 – Indexação de Documentos XML

O Sistema Berkeley DB XML tem indexação única e dinâmica, permitindo a recuperação otimizada de conteúdo XML. Sua máquina de consulta foi projetada baseada em estatística e planejamento de custos da recuperação, para entregar resultados rapidamente durante o processamento de instruções complexas XQuery em grandes conjuntos de dados. De acordo com (FALCÃO, 2006), os principais destaques para um grande conjunto de indexação são:

- Indexação flexível de nós XML, elementos, atributos e meta-dados;
- Nós índices de nível que melhoram o desempenho da consulta, especialmente para grandes documentos XML;
- A criação do índice complexo e remoção em tempo de execução;
- Índices específicos orientados para os pontos fortes;
- Índices de tipo e existência específica;
- Planejamento e otimização de consulta interativa índice;
- Parte do documento re-indexação.

2.5.3 – Consulta a Documentos com Suporte XML

A linguagem XQuery oferece para os bancos de dados XML o que o SQL oferece aos bancos de dados relacionais. Com o XQuery é mais fácil expressar relações complexas, agrupamentos, condições e conjuntos de resultados que podem ser otimizadas e executadas de forma mais rápida sob um conjunto grande de dados.

Os padrões do Berkeley DB XML se aproximam bastante dos definidos pelo XQuery, além de oferecer uma das mais complexas implementações. De acordo com (GEOFF, 2005), as características principais para a consulta de acesso ao documento XML que podem ser destacadas são:

- XQuery e XPath;
- Consultas em um único recipiente, ou através de muitos;
- Consultas em recipientes e fontes de rede de dados XML;
- Permanente identificadores de documentos para acesso direto;
- Otimização de consultas, no custo do mecanismo de consulta;
- Avaliação da expressão simplificada para caminho e avaliação do predicado;
- Documento de *streaming* a partir da memória URI, ou arquivo;
- DOM-como a navegação de resultado XML conjuntos.

2.5.4 – Modificação em Documento XML

O banco de dados Berkeley DB XML fornece uma API completa para efetuar modificações que permitem atualizações muito mais eficientes. No entanto, não faz parte ainda do padrão XQuery as alterações de um documento XML. O suporte ao banco de dados Berkeley DB XML a estas modificações só serão oferecidas quando esses padrões forem aceitos. Entretanto, existem alguns recursos que podem auxiliar neste processo como:

- Alteração parcial de documentos;
- Modificação de documentos em pontos específicos com transações;
- Modificação simultânea de diferentes seções do conteúdo.

2.6 – Representação da Informação Genômica

As representações das informações genômicas que são obtidas através dos esforços realizados dos seqüenciamentos de nucleotídeos de DNA são essenciais para o tratamento computacional, onde visa o desempenho da análise dos processos biológicos. Essas informações precisam ser representadas e armazenadas. No armazenamento, a preocupação ocorre com a eficiência do acesso e com a manutenção da consistência da informação depositada. Para isso, é necessário uma representação adequada dos dados (modelo de representação).

A modelagem em XML (*eXtended Markup Language*), além de ser adequada à representação de conteúdos e constituir padrão para a troca de informação, permite a manipulação dos dados através de consultas e modificações com linguagens apropriadas. O paradigma de modelagem através de XML Schema é uma abordagem nova, que sinaliza para a possibilidade de uma padronização da informação genômica. O interesse nos modelos XML Schema reside no fato de se construir uma base de dados XML nativa e fornecer os dados armazenados a outros sistemas, utilizando a infra-estrutura da *web*. Já a utilização de conceitos de orientação a objetos (OO), procura prover propriedades de flexibilidade e reuso aos esquemas, ou seja, o esquema pode ser utilizado por diferentes aplicações, ou os tipos da biblioteca podem ser reutilizados para novos esquemas.

O uso dessa metodologia tem permitido tirar conclusões interessantes pelos pesquisadores sobre a modelagem em XML aplicada à troca de dados biológicos e sobre a utilização de esquemas na troca de dados genômicos.

A figura 10 mostra as etapas de desenvolvimento dos esquemas XML.

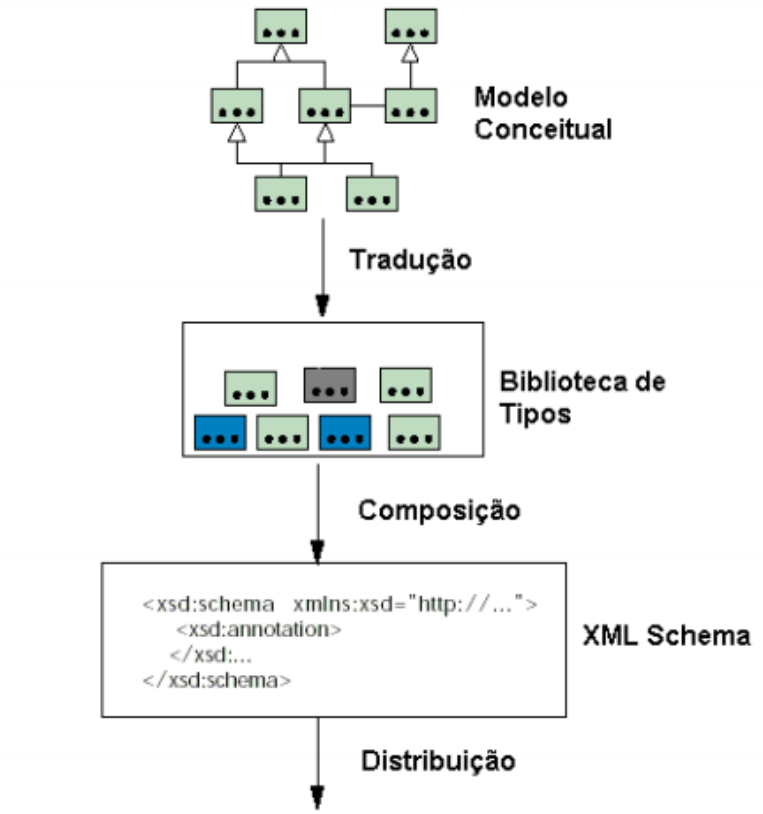

Figura 10 - Etapas de desenvolvimento dos esquemas XML. (VEIGA e PORTO, 2003)

Os processos de desenvolvimento adotado são iniciados pela concepção de um modelo conceitual ou conhecido também na literatura por ontologia. A biblioteca de tipos XML Schema desenvolvida para compor esquemas de elementos genomicos foi baseada no modelo conceitual proposto por (PATON ET AL, 2000), na parte que descreve os mecanismos de transcrição para a formação de RNA mensageiro e de tradução das estruturas primárias de uma proteína funcional ou enzima. Importante salientar que é nesta etapa que são aplicados os padrões de projeto de orientação a objetos relacionados á reusabilidade. A última etapa do processo é a de composição, onde a aplicação usuária escolhe tipos da biblioteca para montar os esquemas XML que irão definir o formato e o conteúdo do documento XML.

O XML Schema é basicamente uma linguagem de definição de tipos, chamados simples e complexos, ou seja, eles são chamados desta forma podem envolver outros tipos e com isto formar seu conteúdo. Para projetar um esquema XML é

necessário definir estes tipos, definir seu conteúdo, denominado *element content*, e utilizar padrões de projeto para combiná-los. A figura 11 mostra como o sistema se comporta diante de informações XML com os diagramas de cada atributo de determinada classe.

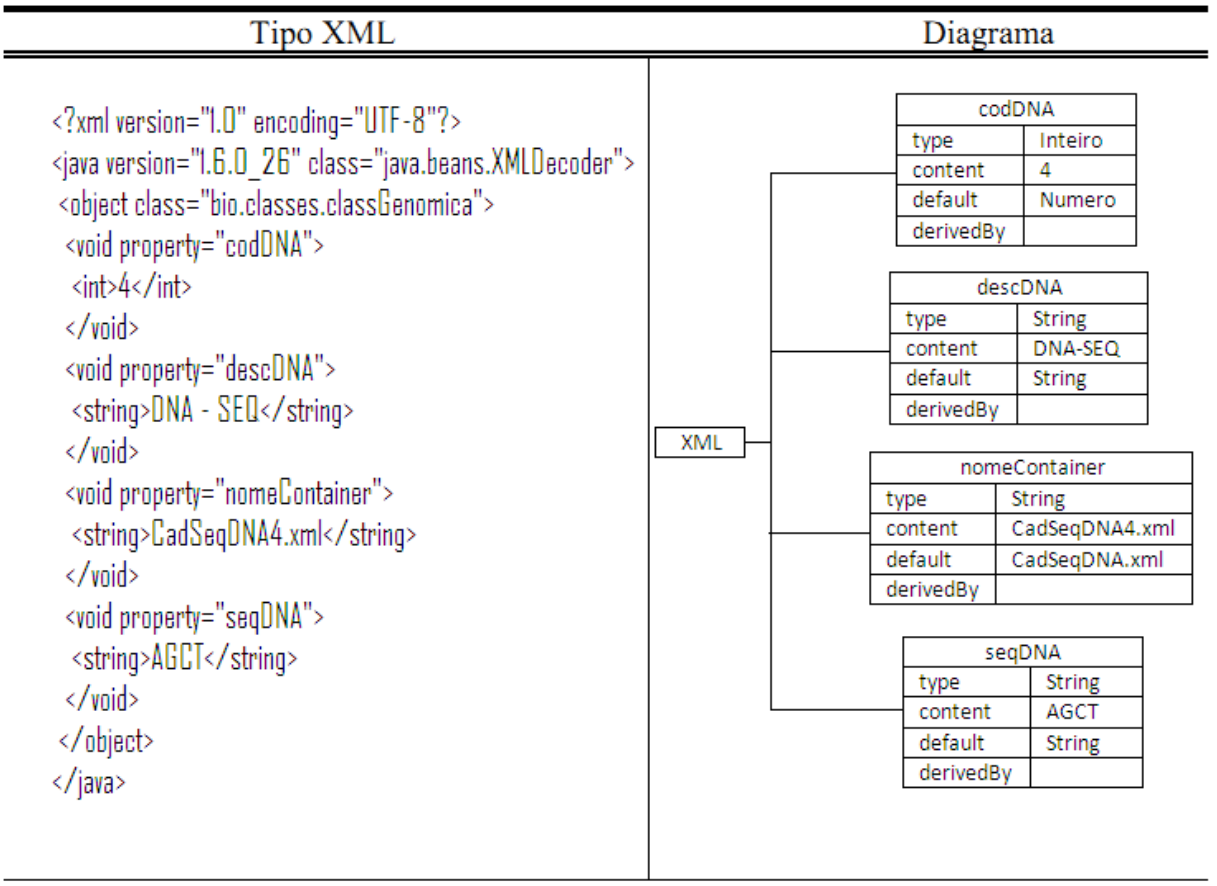

Figura 11 – Representação da Classe em formato XML.

A figura 11 mostra como identificar os atributos gerados na instrução XML, onde todas as TAGs são definidas com sua propriedade e o tipo de informação armazenada. Na implementação do diagrama fica explícito que todos os *containers* definido são definitivamente as TAGs correspondente as informações de XML.

Após as modelagens de como o arquivo XML se comporta com as informações obtidas para o seu funcionamento, será necessário definir um modelo para realizar a comunicação entre o Banco de Dados XML com o aplicativo que fará a identificação das informações XML com a manipulação das informações geradas.

A figura 12 mostra este processo de troca de dados com XML.

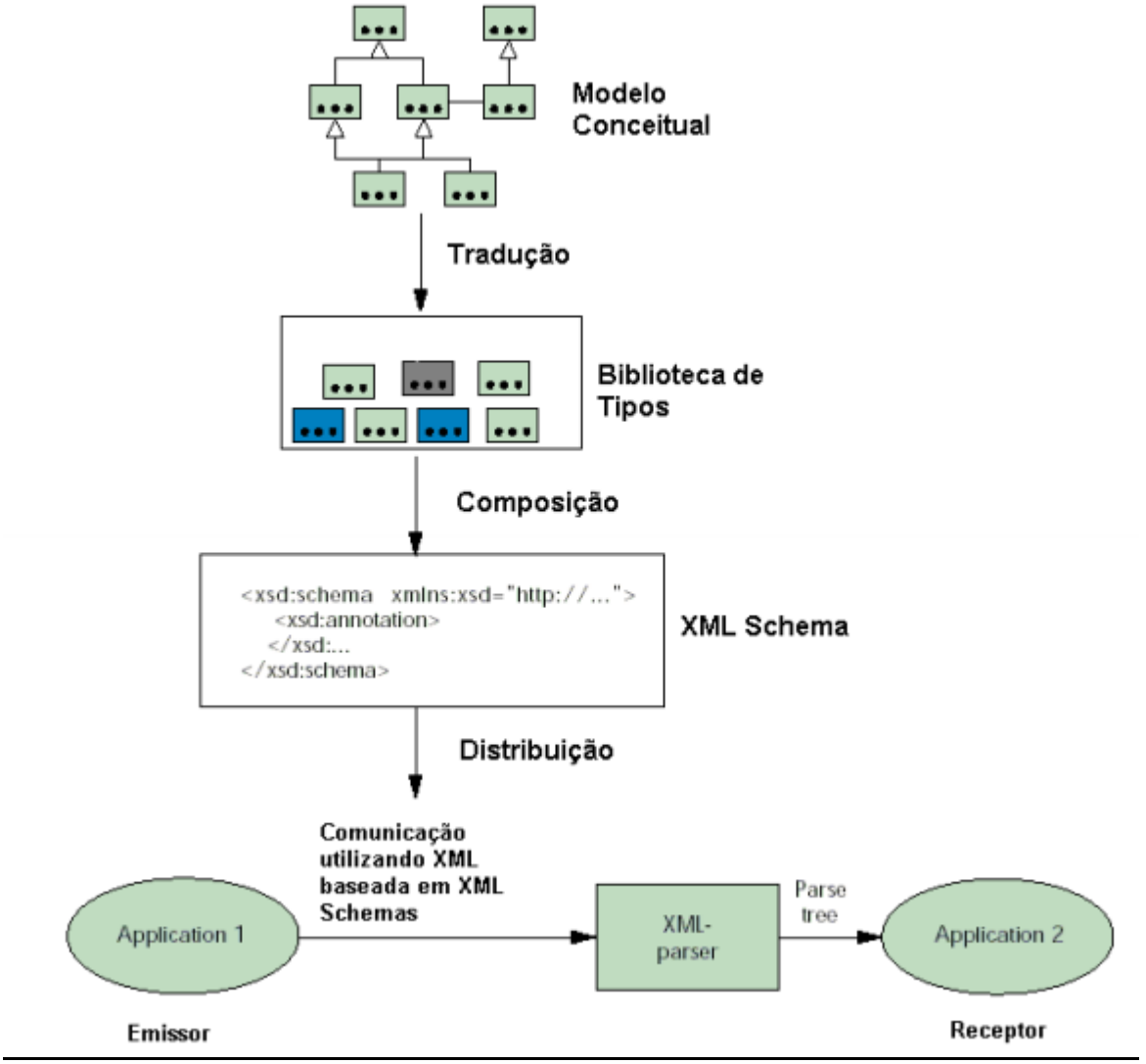

Figura 12 – Processo de troca de dados XML (VEIGA e PORTO, 2003)

Um dos pontos importantes a ser observado é que os dados podem passar por um processo de validação para serem aceitos pelo receptor.

3. DESENVOLVIMENTO DO PROJETO

Neste capítulo, será feita a descrição e modelagem do problema a ser desenvolvido no projeto, além de descrever toda a especificação dos aplicativos e sua implementação. Será mostrado como integrar aplicações de reconhecimento e busca de padrões na sequência de DNA com um banco de dados XML.

3.1 – Descrição do Problema

Neste projeto será desenvolvido um aplicativo para reconhecimento e busca de padrões na sequência de DNA. O reconhecimento da cadeia de caracteres da sequência de DNA será feito utilizando os conceitos de autômatos finitos, enquanto para a busca de padrão será utilizado os algoritmos de busca conhecidos na literatura. Será criado um banco de dados XML para armazenar as informações geradas pelo aplicativo. Esse aplicativo será integrado ao banco de dados XML.
A implementação será feita utilizando as tecnologias Java.

3.2 – Modelagem do Problema

Nesta seção, será apresentada a modelagem do problema. Esta modelagem é uma representação da arquitetura de integração do aplicativo de reconhecimento e busca de padrões na sequência de DNA com o banco de dados XML. O problema em questão é bem complexo e requer cuidados em seu detalhamento. O desenvolvimento do projeto foi dividido em cinco módulos para facilitar na implementação.

- Módulo1 - Criação do Banco de Dados XML;
- Módulo 2 - Desenvolvimento do Sistema de Cadastro;
- Módulo 3 - Desenvolvimento do Aplicativo de Reconhecimento de Padrões;
- Módulo 4 - Desenvolvimento do Aplicativo de Busca de Padrões;
- Módulo 5 - Integração dos Aplicativos com o banco de Dados.

A figura 13 mostra a modelagem do problema.

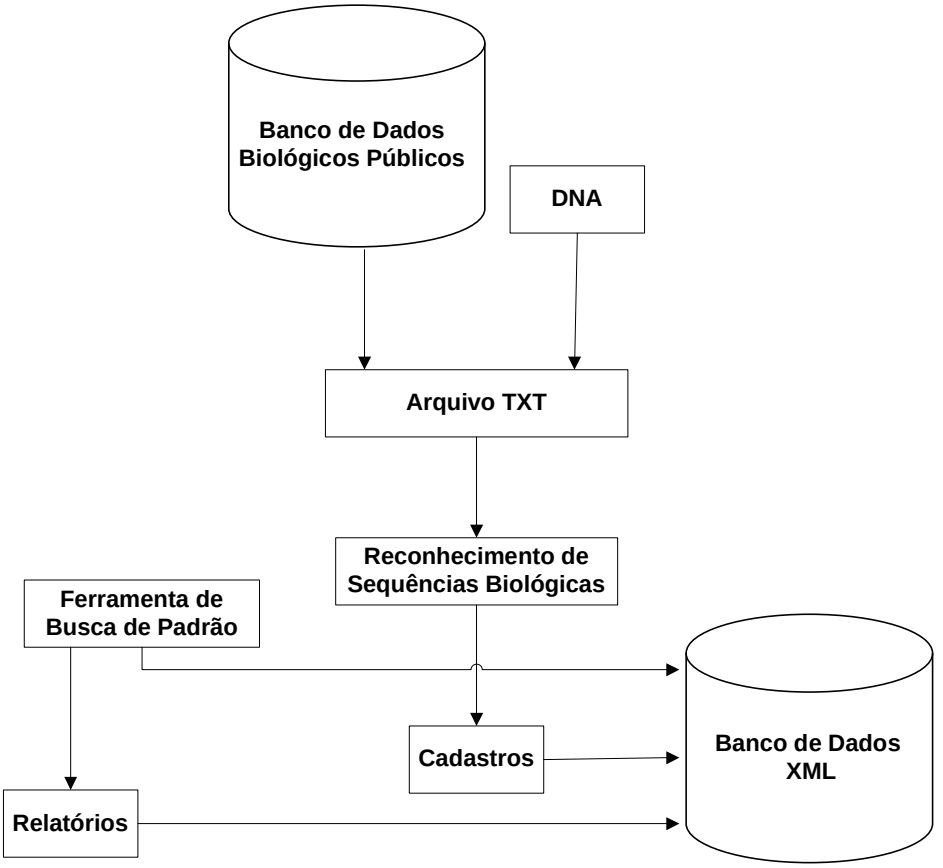

Figura 13 – Modelagem do Problema

O Banco de Dados Biológicos Públicos representa o repositório de informações genômicas com acesso público. Cadastros é um sistema para cadastrar cadeias de caracteres da seqüência de DNA. Essas informações são gravadas em arquivo txt com os padrões caracterizados com o banco de dados XML. O reconhecimento e a busca de padrões nessas seqüências serão feitas após a gravação destes arquivos txt, onde será feito inicialmente o reconhecimento e definido o padrão será feita a busca.

3.3 – Especificação

Nesta seção, será feita a especificação do modelo de integração do aplicativo com o banco de dados XML, onde foi utilizada uma metodologia orientada a objetos para a construção dos diagramas UML (*Unified Modeling Language*), utilizando a ferramenta ArgoUML.

Na especificação serão apresentados os diagramas de casos de uso, diagrama de classes, os diagramas de sequência e o diagrama de atividade.

3.3.1 – Diagrama Entidade-Relacionamento

O Diagrama Entidade-Relacionamento tem o objetivo de mostrar de forma simples os fluxos de Informações que serão realizados pelo sistema de integração. A figura 14 representa o Diagrama de Entidade-Relacionamento.

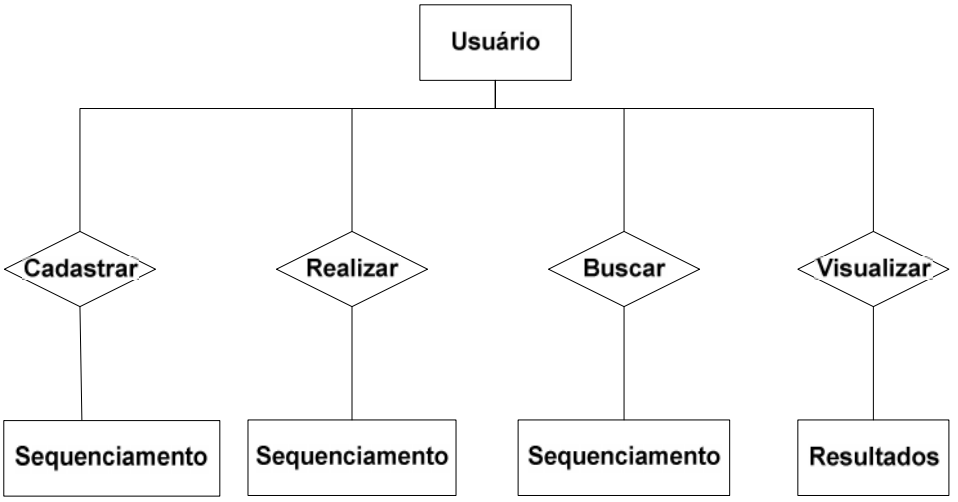

Figura 14 – Diagrama de Entidade Relacionamento

Este diagrama mostra o fluxo de informação do usuário.

- **Cadastrar:** será realizado o cadastramento do indivíduo com o seu DNA e sua respectiva descrição;

- **Realizar:** será disponibilizado para realizar a padronização na seqüência genética, tanto no reconhecimento da cadeia quanto na busca de um padrão genético;
- **Buscar:** será possível realizar as buscas dos processos realizados no reconhecimento de padrões genéticos;
- **Visualizar:** permitirá visualizar os processos realizados.

3.3.2 – Diagrama de Casos de Uso

O Diagrama de Caso de Uso é uma técnica utilizada para realizar as modelagens de um sistema, onde são definidos todos os seus requisitos e suas funcionalidades.
A figura 15 mostra o diagrama de casos de uso geral.

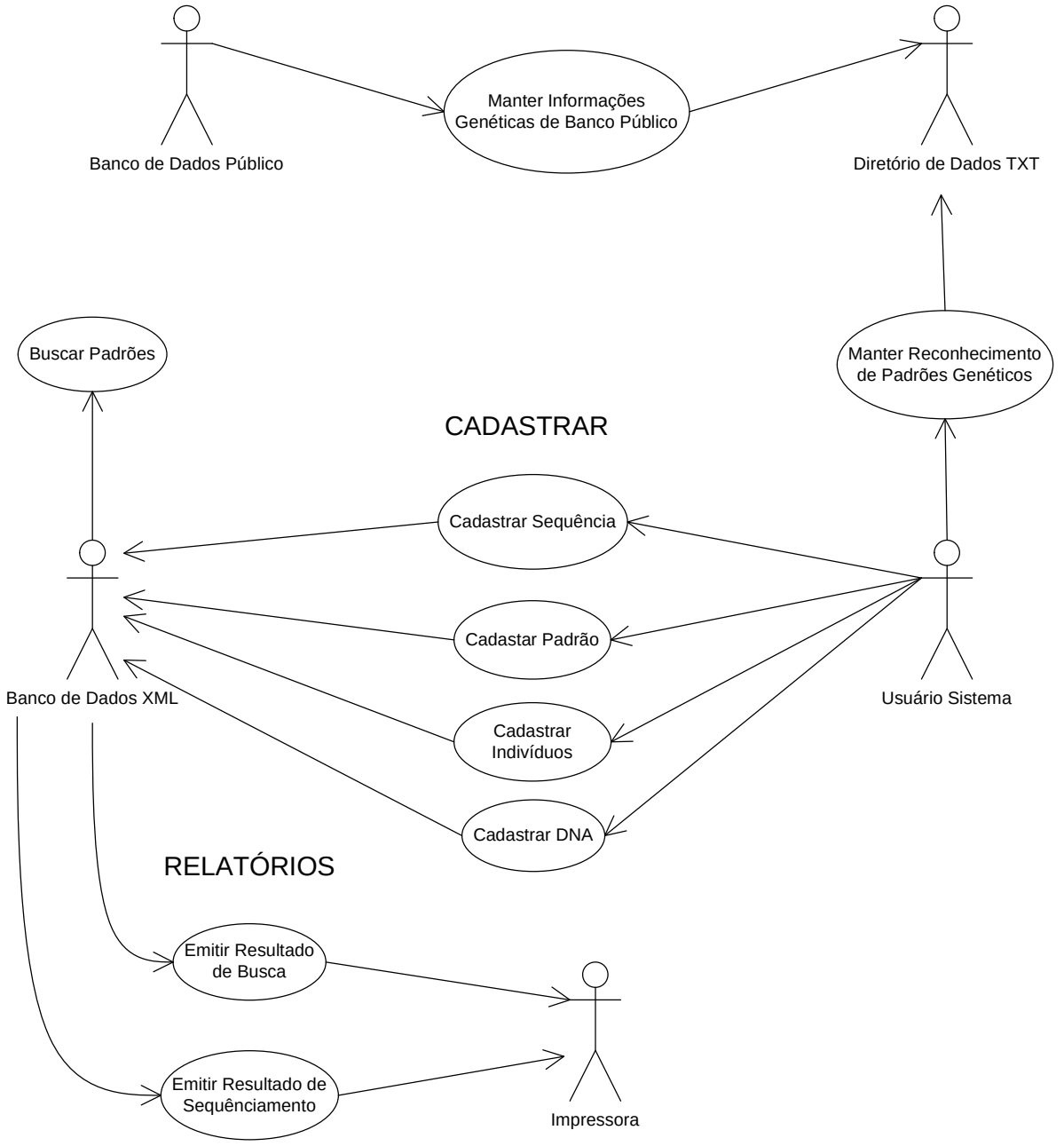

Figura 15 – Diagrama de Caso de Uso Geral

- **Manter Informações Genéticas de Banco de Dado Público:** sendo encarregado de se atribuir e recuperas as informações genéticas de outros lugares para poder ser utilizado no Sistema de Integração de Informação.
- **Manter Reconhecimento de Padrões Genéticos:** neste processo será realizado o processamento da informação genética de acordo com o diretório raiz, para a busca e análise da informação.

- **Cadastrar Sequências:** cadastramento básico criado para ter em Banco de Dados XML a informação genética referente ao indivíduo.
- **Cadastrar Padrão:** cadastramento necessário junto ao Cadastramento de Sequencias.
- **Cadastrar Indivíduo:** cadastramento necessário junto ao Cadastramento de Sequências.
- **Cadastrar DNA:** cadastramento necessário junto ao Cadastramento de Sequências.

3.3.3 – WBS (Estrutura Analítica do Projeto)

Os WBS são formas de representar o assunto como um projeto sendo separado por partes de um projeto, isto significa dizer que é uma estrutura que decompõe partes de um projeto. A figura 16 mostra a WBS do modelo proposto.

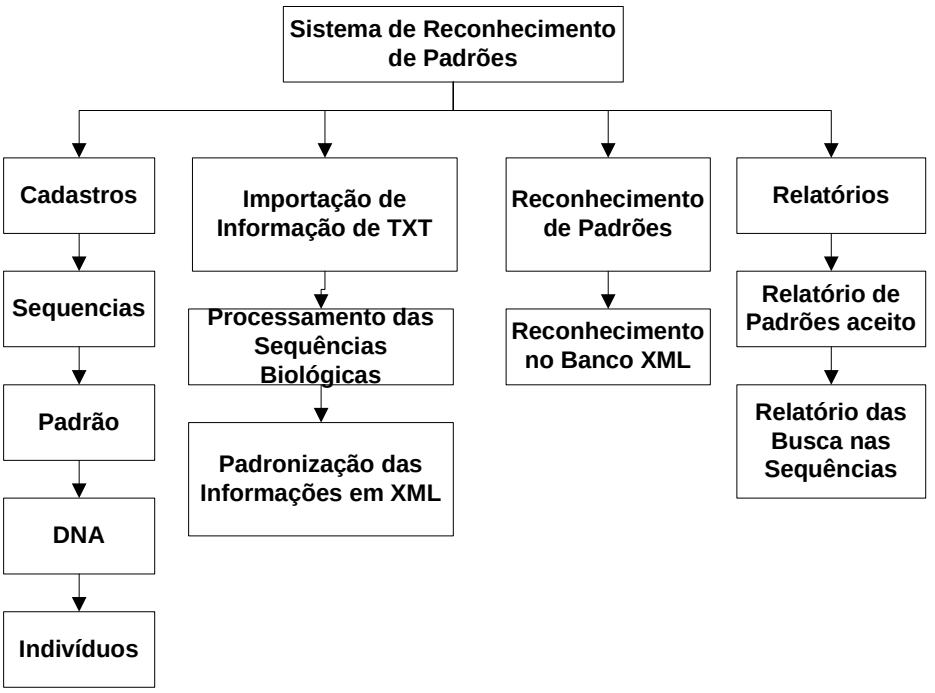

Figura 16 – WBS do Projeto

Na figura 16 pode ser observado que o projeto será realizado em partes por questão de padronização e necessidade das informações primordiais. No aplicativo de reconhecimento de padrões será abordado:

- **Cadastros:** serão realizados os passos de cadastramento de sequencia, padrão, DNA e indivíduos;
- **Importação de Informação de TXT:** processo onde serão recuperadas as informações pertinentes para as etapas do processamento. Processamento das sequências biológicas e padronização das informações em XML;
- **Reconhecimento de Padrões:** processo fundamental para a consistência das informações referentes às sequências genéticas, pois só poderão ser armazenadas em banco as informações que forem com o padrão genético. Após esse processo terá que utilizar o processo de reconhecimento em banco XML;
- **Relatórios:** processos onde podem ser realizadas as conferências feitas pelo sistema, podendo ser observado todas as informações geradas pelo processamento da ferramenta de reconhecimento de padrões. Podendo englobar consultas das mesmas.

3.3.4 – Diagrama de Classe

Os diagramas de classe são formas de representação da estrutura e relações das classes criadas no projeto, sendo fundamentais para a modelagem. Os processos se interligam entre si, mantendo a integridade dos mesmos, pois foi criada uma única classe de cadastramento de DNA. Uma classe de armazenamento da sequência analisada e uma classe de controle de inserções. Na figura 17 será mostrado o diagrama de classe.

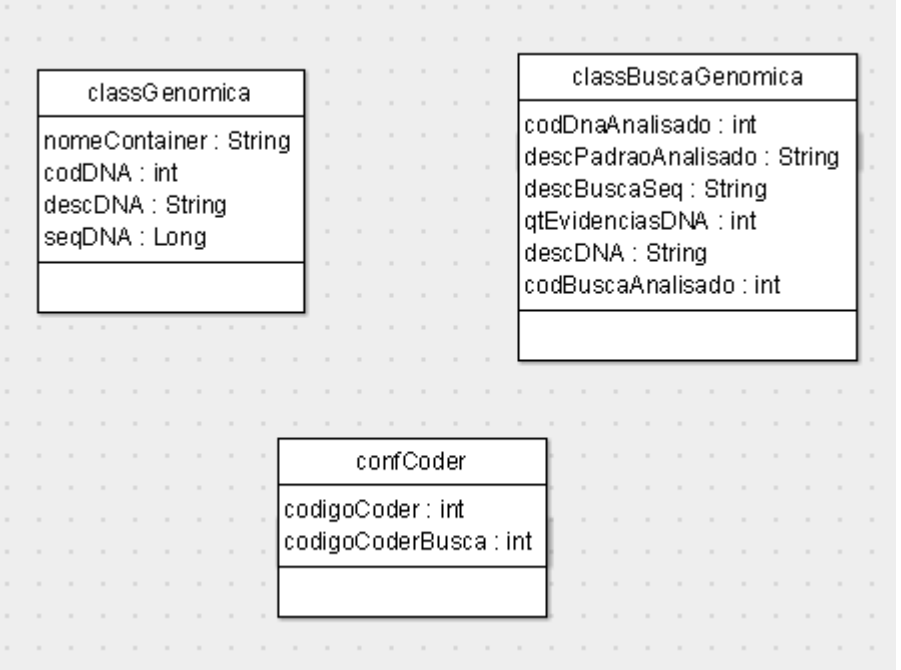

Figura 17 – Representação do Diagrama de Classe

O diagrama de classe apresenta as duas classes criadas. A classe para o cadastramento e a classe para o controle.

- **Classe classGenomica**: Classe responsável para o cadastramento da sequência genética, sendo a classe principal para se ter o bom funcionamento do sistema.
- **Classe classBuscaGenomica**: Classe responsável para armazenar os padrões analisados, contendo as informações do DNA analisado e a quantidade de caracteres encontrado na sequência de DNA.
- **confCoder**: responsável para obter informações precisas de inserção em banco, pois junto com o conjunto de serialização fará a consistências das informações, disponibilizando códigos únicos para a inserções em banco.

3.3.5 – Diagrama de Atividade

O diagrama de atividades é um diagrama definido pela UML, e tem por objetivo representar os fluxos conduzindo sempre em um bom processamento das informações. A Figura 18 mostra o diagrama de atividade do processo de cadastramento de DNA.

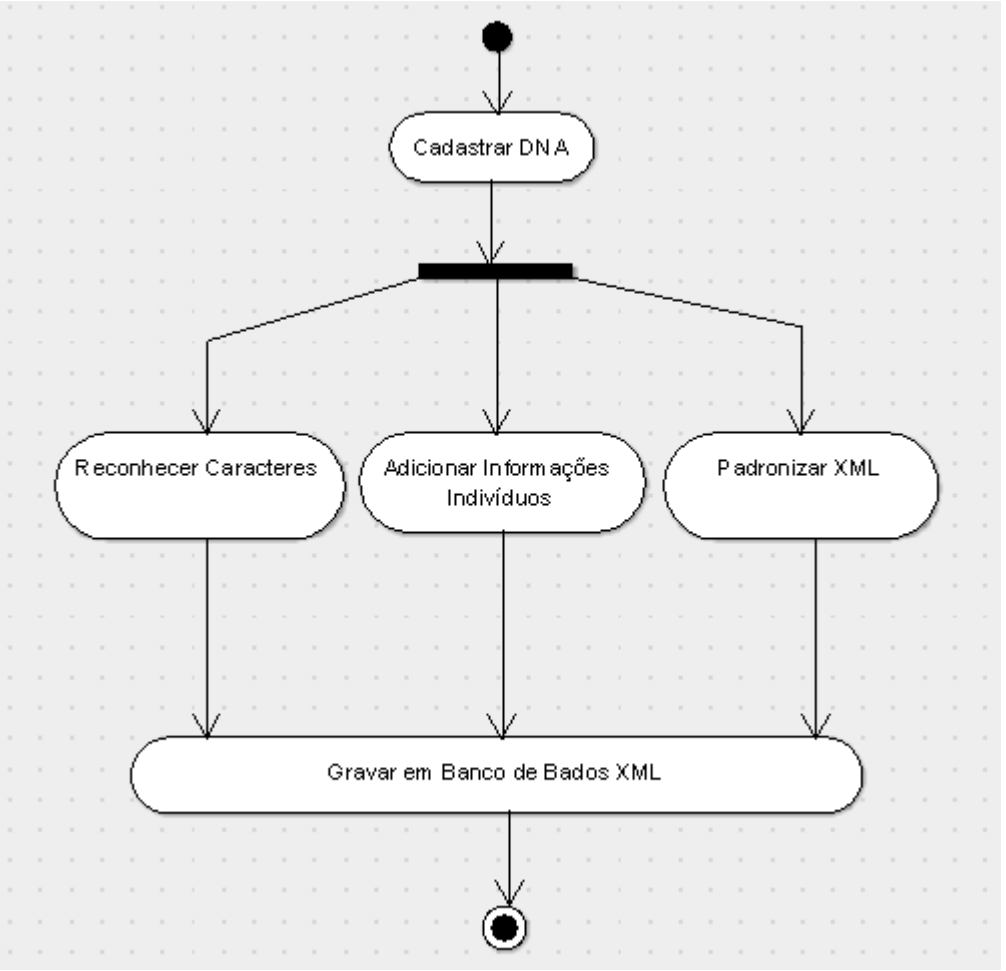

Figura 18 – Diagrama de Atividade do Cadastramento de DNA

A Figura 19 mostra o desenvolvimento do diagrama de atividade para o reconhecimento de padrões genéticos em uma cadeia de DNA.

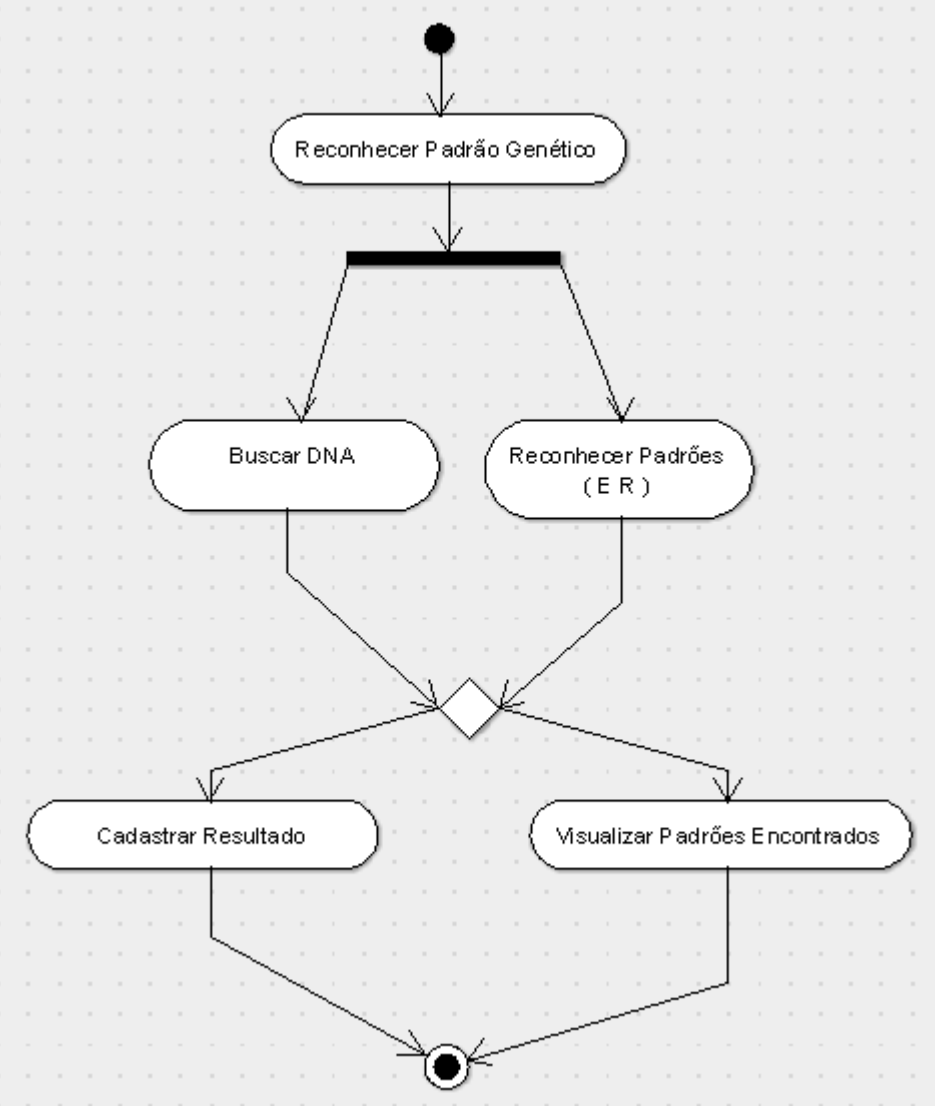

Figura 19 – Diagrama de Atividade do Processo de Busca de Padrões.

O diagrama de atividade no processo de busca de padrões genéticos mostra a visualização do resultado na busca, ou realiza a gravação das informações em banco de dados XML.

3.3.6 – Diagrama de Sequência

O diagrama de sequência é um diagrama para representar a sequência de um processo. Como um aplicativo pode ter uma grande quantidade de fluxo e métodos, o diagrama de sequencia descreve a maneira como o grupo de objetos se comporta ao longo do tempo. Os dois diagramas de sequencia são:

- **Cadastramento da Sequencia Biológica**: definida como ponto principal do aplicativo, pois será necessário para fazer as buscas referentes aos padrões analisados. O diagrama de sequência de cadastramento verifica se a cadeia de caracteres é aceita ou não;
- **Busca de Padrões Genéticos**: neste processo são realizadas as buscas dos padrões na sequência cadastrada.

A figura 20 mostra o diagrama de sequencia para o cadastramento da sequencia biológica.

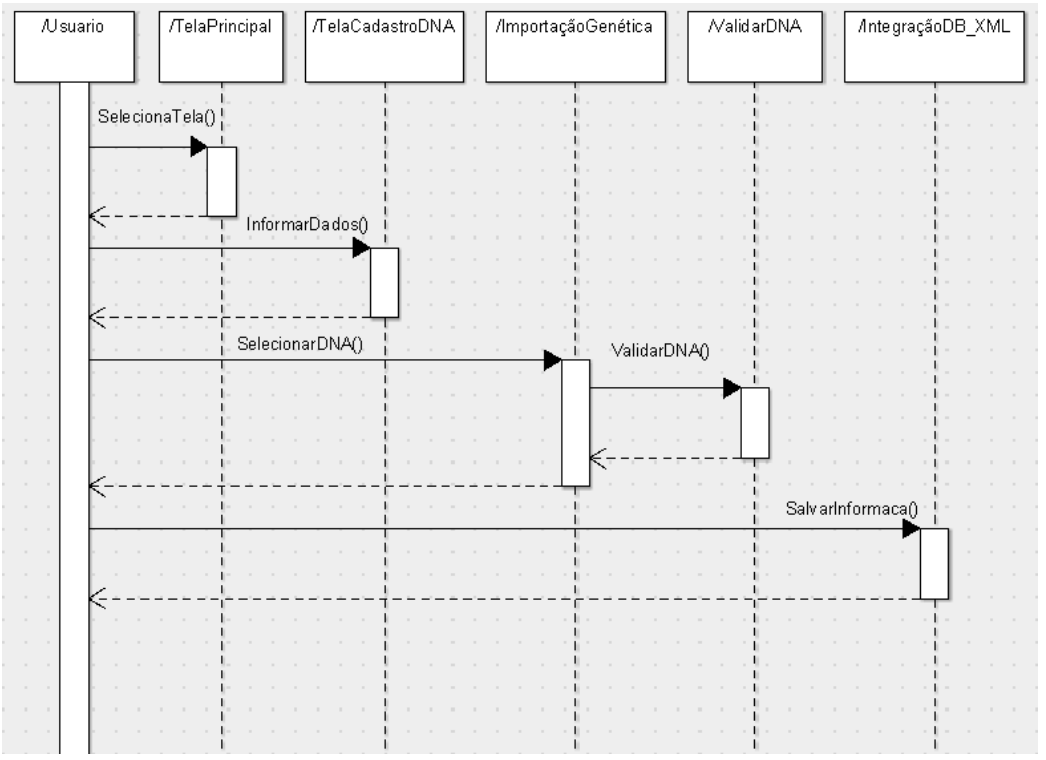

Figura 20 – Diagrama de Sequência do Cadastramento da sequencia de DNA.

A figura 21 mostra o diagrama de sequencia para a busca de padrões genéticos.

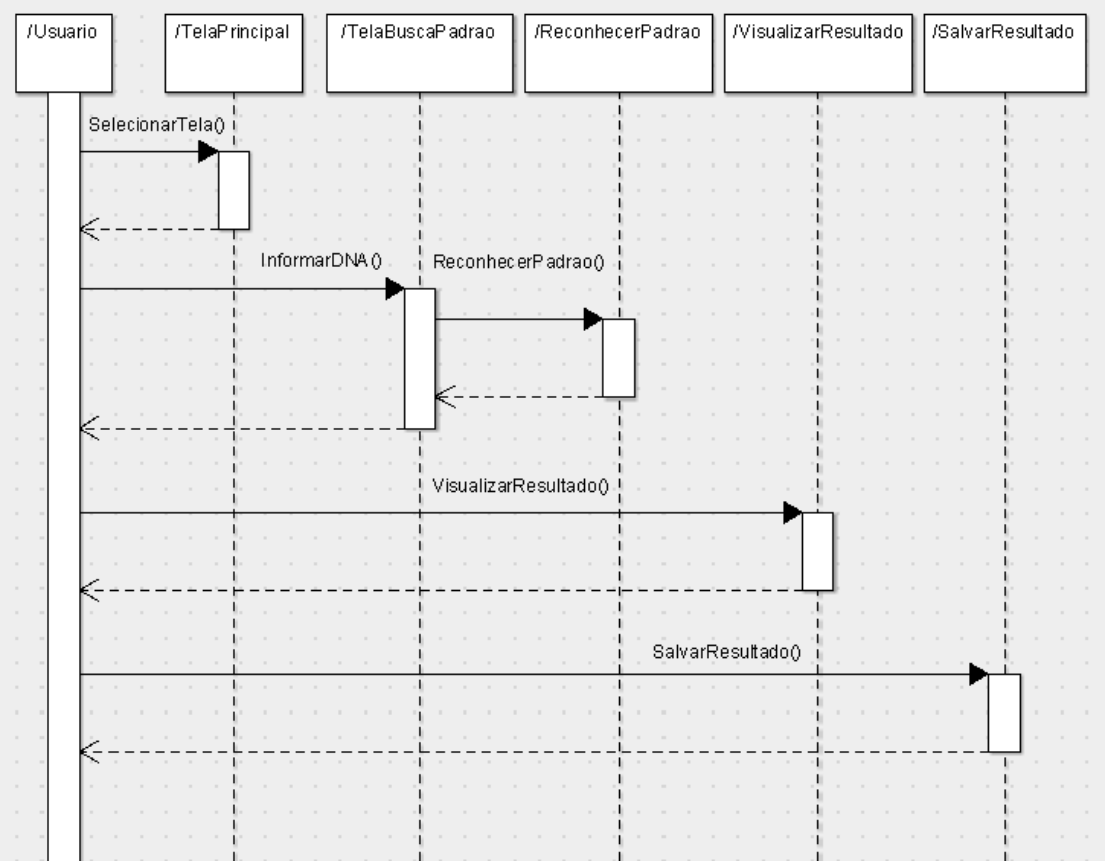

Figura 21 – Diagrama de Sequência da Busca de Padrões.

Neste diagrama de sequencia será analisada uma sequencia de DNA cadastrada em um banco de dados XML e busca os padrões procurados. No processo de busca existem duas opções de fluxo, isto é, pode ser realizada a inserção das informações analisadas, ou apenas mostrar na interface o padrão identificado.

3.4 – Implementações

Nesta seção serão apresentados com mais detalhes todos os módulos que serão implementados. A implementação dos aplicativos e a integração com o banco de dados será feita utilizando as tecnologias Java.

3.4.1 – Criação do Banco de Dados XML

Neste módulo, será feita a criação do banco de dados XML para o armazenamento de informações, busca de informações e atualizações das informações. O banco de dados XML será baseado em um banco de dado relacional (GRAVES, 2003). A figura 22 mostra a modelagem da arquitetura do banco de dados.

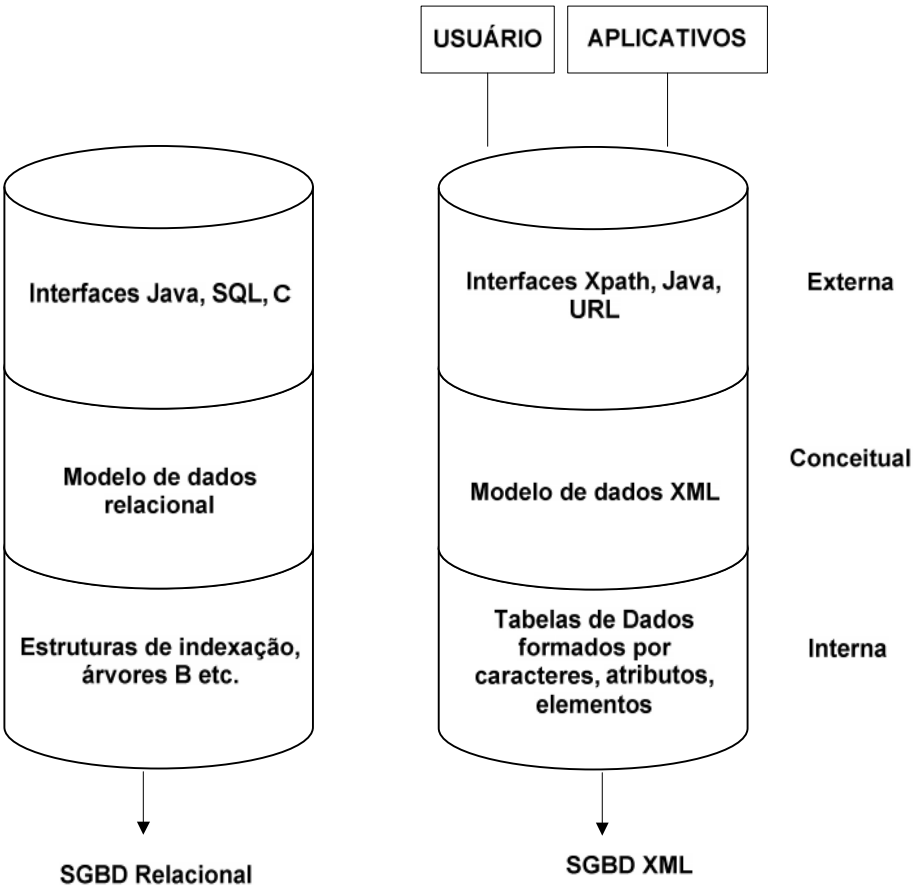

Figura 22 – Representando um SGBD XML usando um SGBD relacional (GRAVES, 2003)

3.4.1.1 - Preparação do Banco de Dados DB Berkeley XML

Após a instalação do Banco de Dados DB Berkeley XML será necessário realizar a configuração e criar um *container*. Este *container* nada mais é do que o espaço útil onde são realizadas todas as transações e armazenadas todas as informações.

Para iniciar o banco de dados é necessário ir em:

- Menu INICIAR – Executar (Digitar: “ dbXml “).

A figura 23 mostra a interface de inicialização do banco.

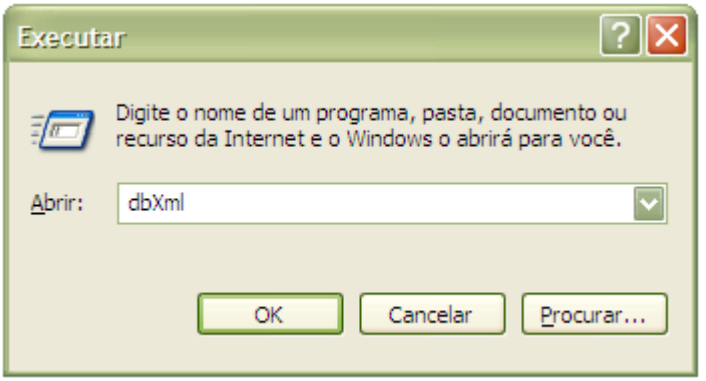

Figura 23 – Interface de Inicialização

Após a execução do menu iniciar, ele carrega a interface do banco de dados Oracle Berkeley DB XML. A figura 24 mostra a interface do banco.

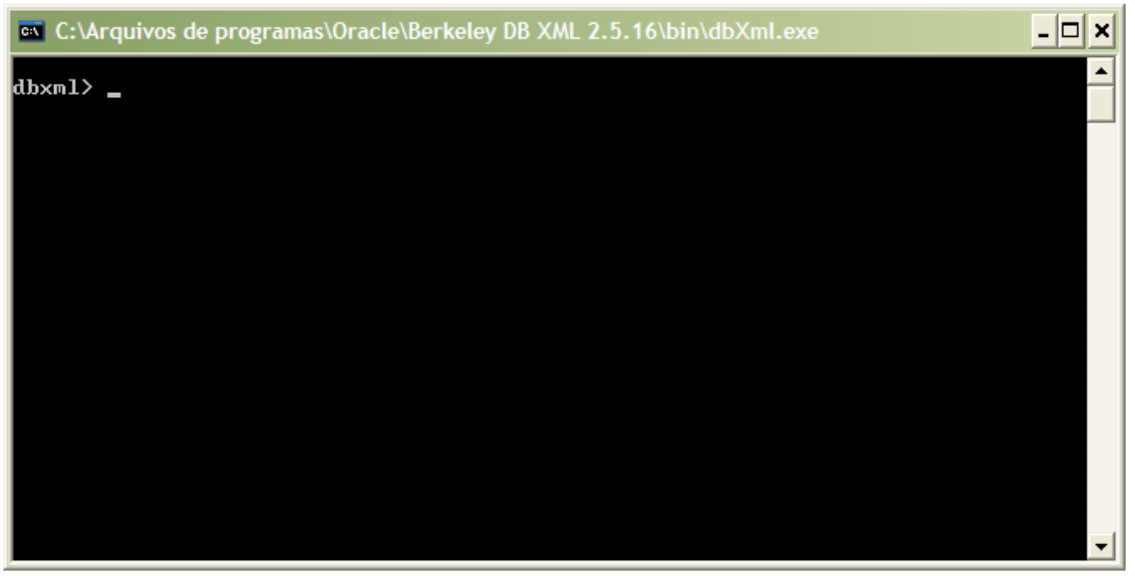

Figura 24 – Interface do Banco de Dados XML.

Com a interface de comunicação do banco de dados XML aberta é necessário realizar a configuração central e a criação do *container* para poder utilizar o Banco de Dados DB XML.

A figura 25 mostra como criar um *contêiner.* O *contêiner* é criado com o comando:

```
dbxml> CreateContainer c:/DB_BIO/dbBioInfo.dbxml
```

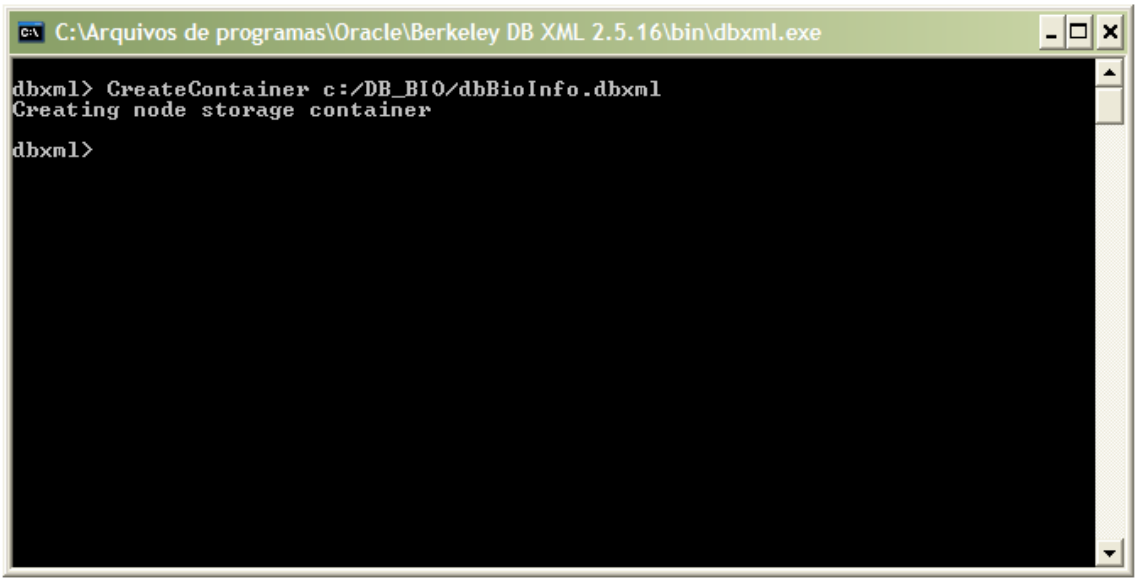

Figura 25– Criação do Container no Banco de Dados XML.

O contêiner foi criado com sucesso. A figura 26 mostra a identificação do *container* criado em disco e o caminho especificado no ato da criação do banco.

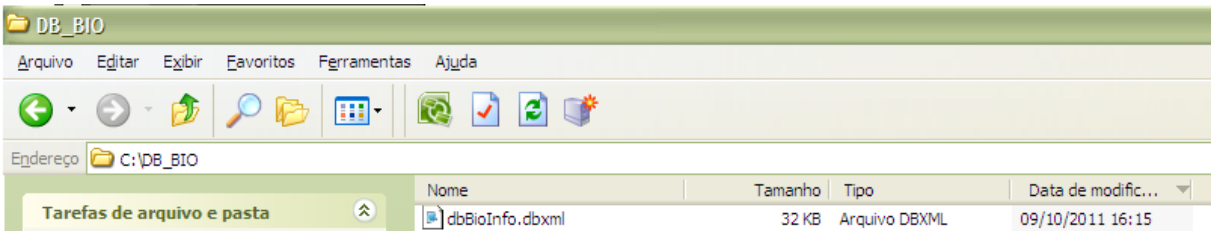

Figura 26 – *Container* Criado no Diretório Especificado.

Após estes procedimentos, será possível iniciar os processos de transações entre o Banco XML e a aplicação desenvolvida.

3.4.2 – Desenvolvimento do Aplicativo de Reconhecimento da Sequência

Neste módulo, será feita a modelagem do aplicativo de reconhecimento da sequência de DNA e para isso foram utilizados autômatos finitos. Os autômatos são apropriados para este tipo de problema. A sequência de DNA é composta pelo alfabeto DNA = { A, C, G, T }. A figura 27 mostra o autômato de reconhecimento da sequência de DNA. A sequência de DNA pode ser interpretada como uma cadeia de caractere.

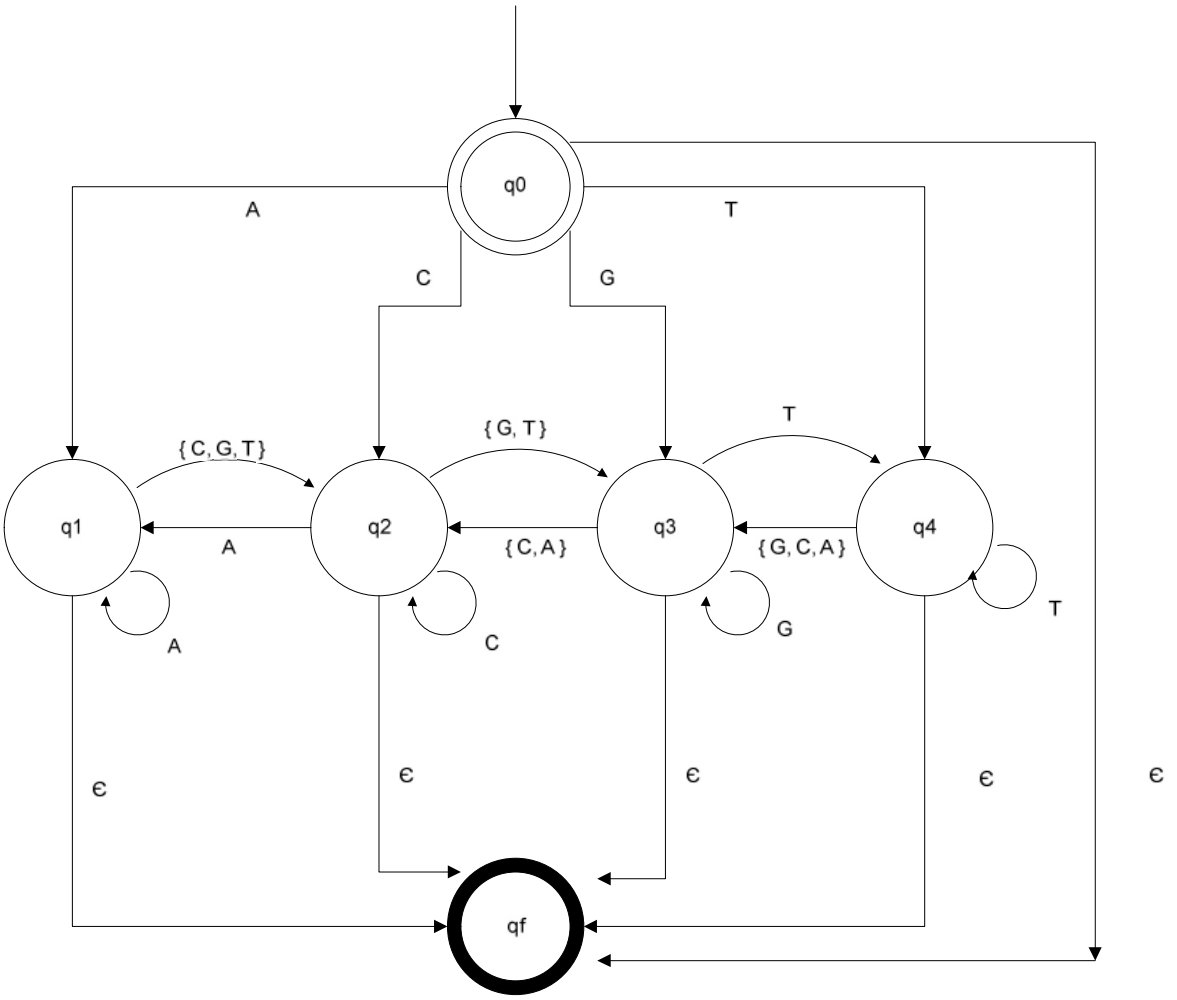

Figura 27 - Autômato de Reconhecimento de Sequência

Este aplicativo será integrado com o sistema de cadastramento da sequência de DNA.

3.4.3 – Desenvolvimento do Sistema de Cadastro

Nesta seção, será apresentado a implementação do sistema de cadastro da sequencia de DNA com as informações necessárias para o processamento computacional.

3.4.3.1 – Visão Geral do Sistema

A interface de inicialização do sistema oferece quatro opções de menu: arquivo, cadastros, processamentos e relatórios. A figura 28 mostra a interface principal do sistema.

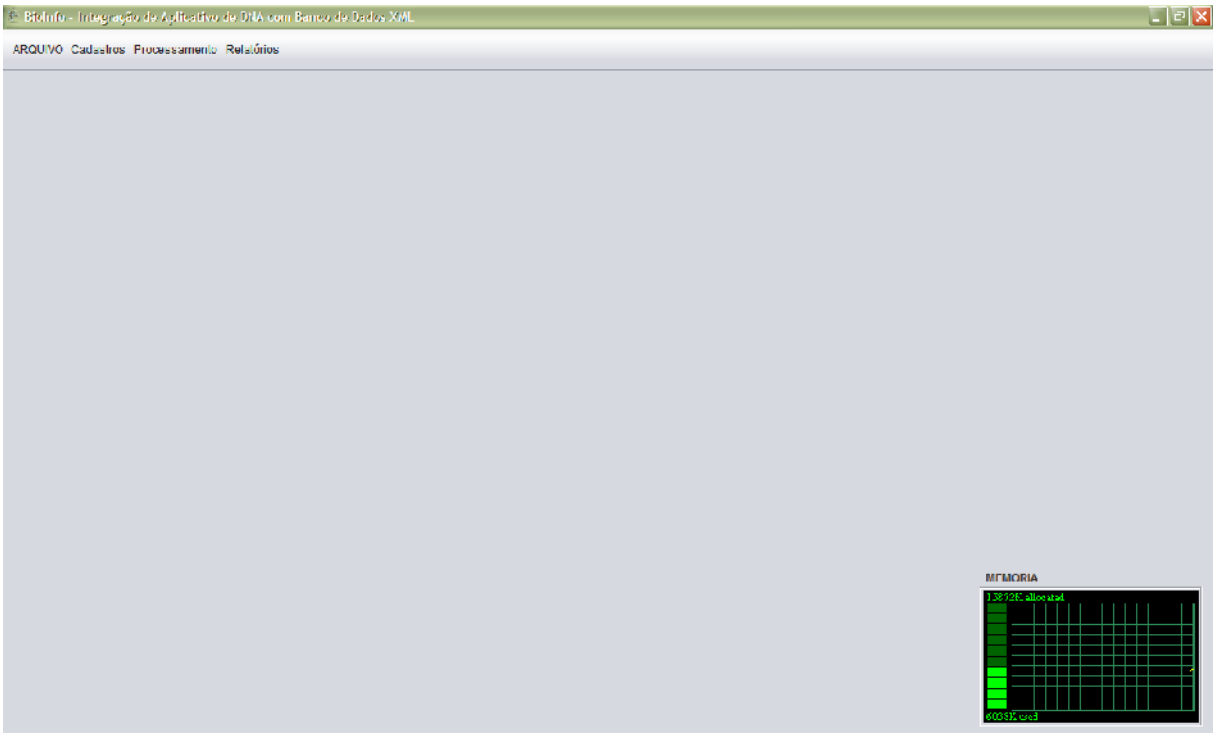

Figura 28 – Tela Inicial do Sistema.

ARQUIVO: é definida pela opção configuração do sistema, opção de sair e opção de realizar manutenção em tela do sistema.

Cadastros: é definida a opção de fazer o cadastramento da sequência genética, onde são utilizados os conceitos de autômatos para fazer o reconhecimento da cadeia de DNA.

Processamento: é definida a opção para analisar uma sequencia genética que já está armazenada no banco de dados XML, utilizando autômatos e expressão regular para realização do processamento.

Relatórios: é definido o relatório final de análise das sequências genéticas, pois após o reconhecimento da cadeia e o processamento da busca de padrões, será mostrado em forma de relatório os processos e as ocorrências obtidas.

Um fato interessante nessa implementação foi à construção de uma classe para a visualização da memória da máquina utilizada. Ela foi construída por uma *theard* que tem por objetivo fazer uma varredura da interface principal do sistema no momento da execução. A figura 29 mostra a interface de monitoramento do processamento das operações nas sequencias de DNA. Este monitoramento mantém o usuário informado das condições em que a máquina está sendo utilizada.

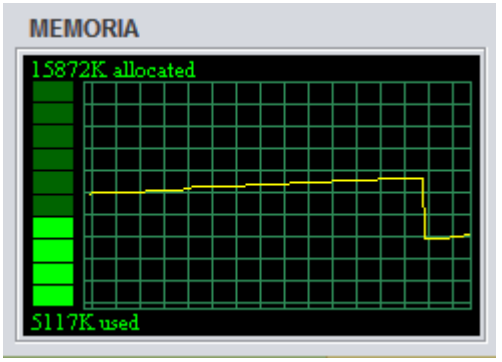

Figura 29 – Monitoramente de Processamento das Operações Genéticas.

3.4.2.2 – Cadastramento do Indivíduo

Para o cadastramento do individuo é necessário descrever todos os passos de um cadastramento de sequências genéticas, podendo ser definidos em imagem para melhor identificação dos processos. A figura 30 mostra a interface de cadastramento da sequencia de cada individuo.

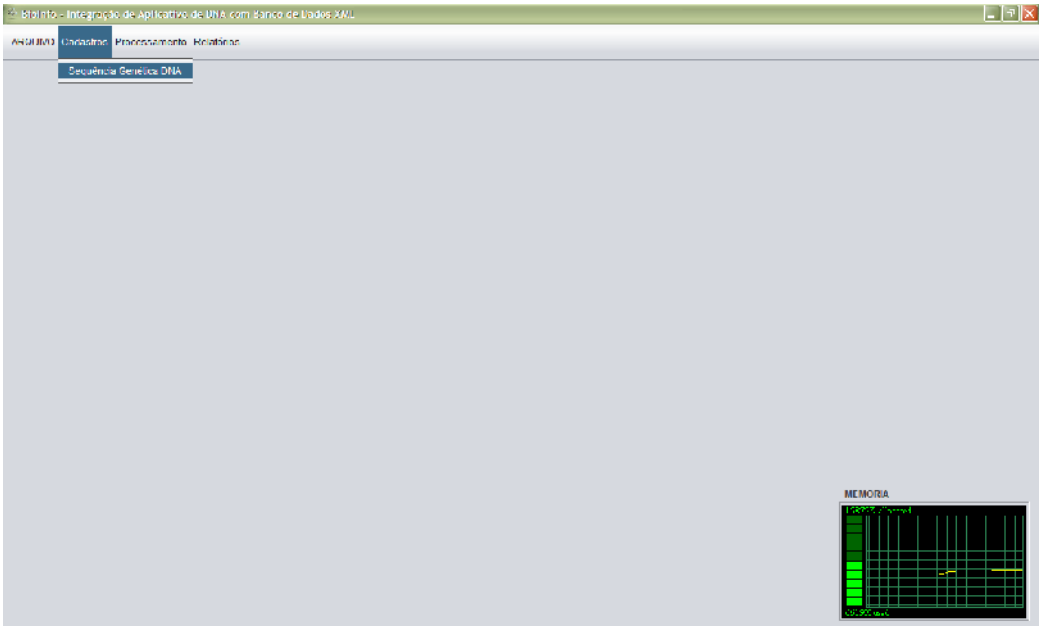

Figura 30 –Cadastramento do DNA.

Para cadastrar uma sequencia de DNA, o usuário deve selecionar a opção Cadastros. A figura 31 mostra a interface encarregada de realizar os cadastramentos das sequências. A execução do cadastramento será mostrado passo a passo para cada indivíduo.

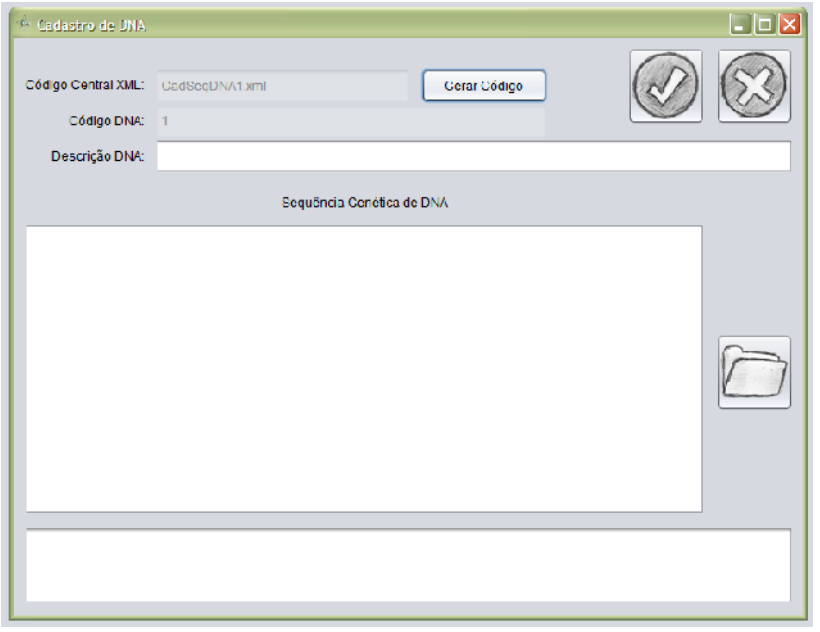

Figura 31 – Interface de Cadastramento da Sequência de DNA.

A interface principal do cadastramento de DNA apresenta quatro campos, onde após o processamento serão armazenados no banco de dados XML. Os quatro campos são:

- **Código Central XML**: este código está associado a um preenchimento automático, pois está configurado com um arquivo *.bin* em Java. Ele tem por objetivo atribuir um código automático sequencial, mas não armazenado em banco de dados XML, mas sim em uma configuração Java. Essa programação foi elaborada pelas funções de serialização que a plataforma Java disponibiliza;
- **Código DNA**: este campo tem o mesmo conceito do primeiro campo, pois armazena apenas o número sequencial, disponibilizando apenas para a visualização e não para a sua alteração. Este código será utilizado para realizar as buscas no sistema;
- **Descrição de DNA**: campo dedicado para descrever o DNA;
- **Sequência Genética de DNA**: campo dedicado para receber os valores de DNA.

Esses campos são de suma importância para o cadastramento correto de uma sequencia de DNA para o sistema. Nesta interface de cadastramento de DNA existe também uma opção para realizar importações de sequências genéticas recuperadas de outros bancos de dados biológicos. Esta opção encontra a direita da interface. A figura 32 mostra a interface de importação de arquivo para o sistema.

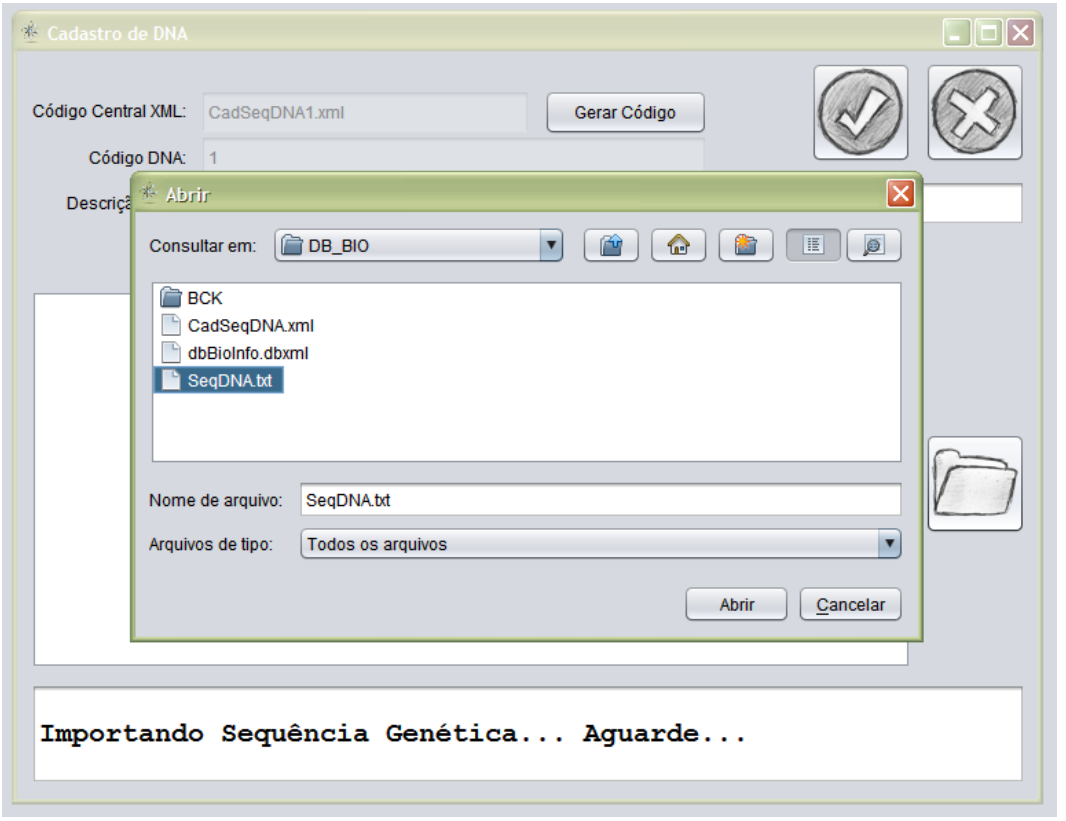

Figura 32 – Importação da Sequência Genética.

Durante o processo de importação, a interface apresenta um campo onde insere uma mensagem “Importando Sequência Genética... Aguarde...”.

Os fluxos de informações se tornaram peças fundamentais no sistema, pois eles são necessários para que as informações estejam em um único lugar. Para realizar o processo de gravação das informações são necessários que os atributos informados sejam válidos.

A figura 33 mostra o processo de importação da sequência genética concluída.

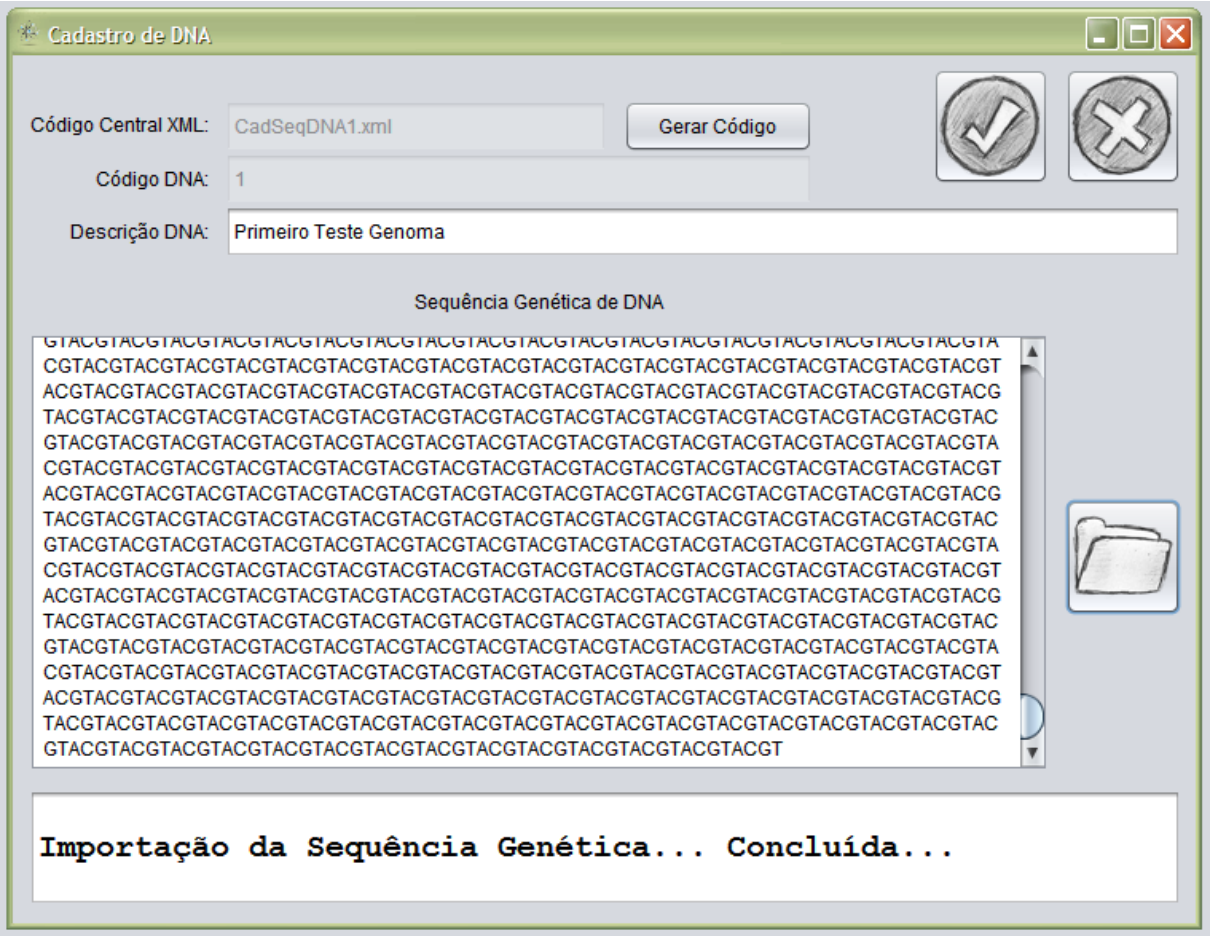

Figura 33 – Interface da Importação da Sequência

Após a conclusão da importação da sequência genética, um passo importante neste processo é validar a sequência importada, pois ela pode vir com um caractere diferente.

A figura 34 mostra o processo de validação da sequência importada. Para isso foi inserido uma cadeia de caractere inválida “xxxxxxxxxx”.

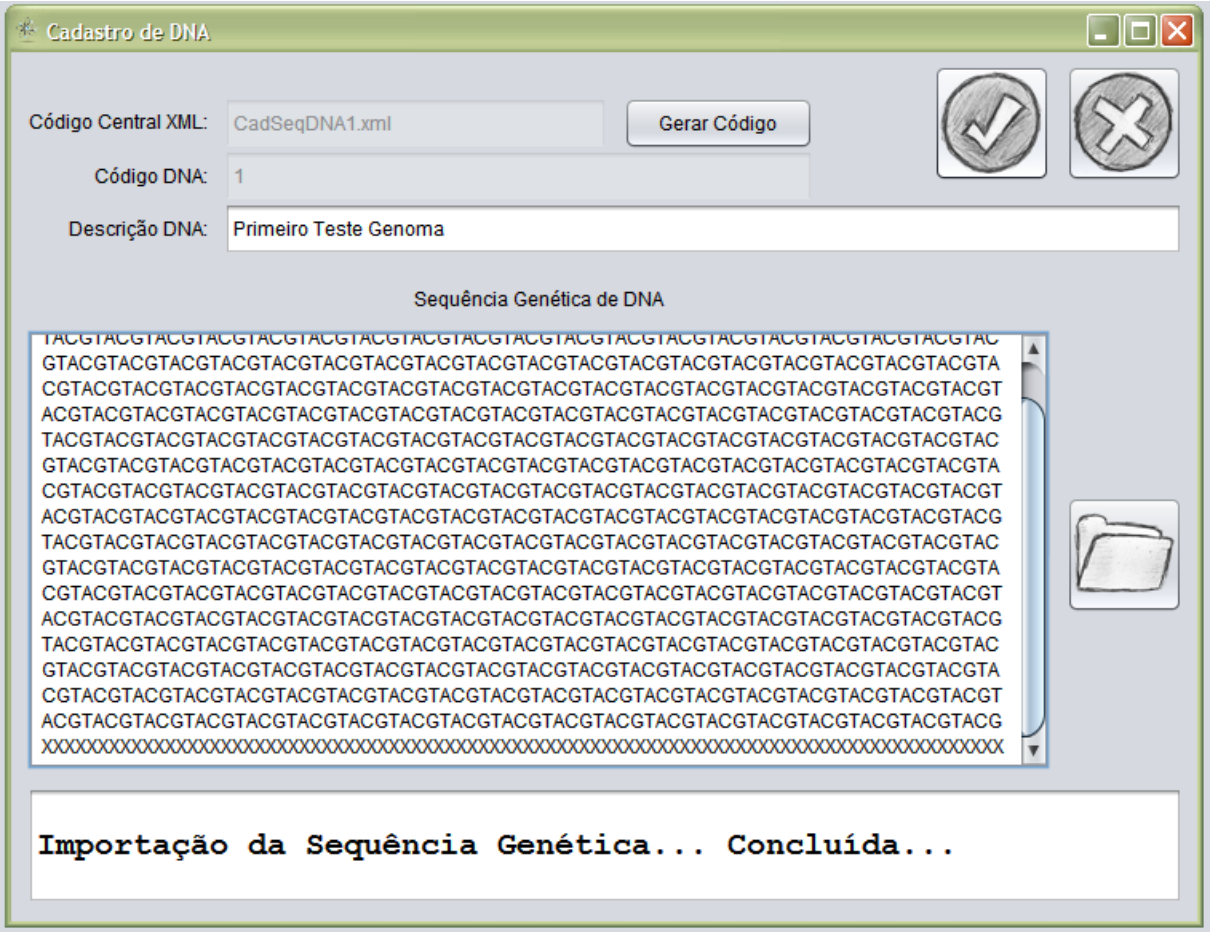

Figura 34 – Importação de Sequência Inválida

Ao incluir uma cadeia de caractere inválida na sequencia de DNA, o sistema envia uma mensagem de erro. Este processo de análise de caractere por caractere é feito pelo autômato desenvolvido para o reconhecimento da cadeia. Se os caracteres pertencem ao conjunto de alfabeto definido no autômato, então aceita cadeia, caso contrário rejeita cadeia.

A figura 35 mostra que a cadeia não foi aceita pelo autômato.

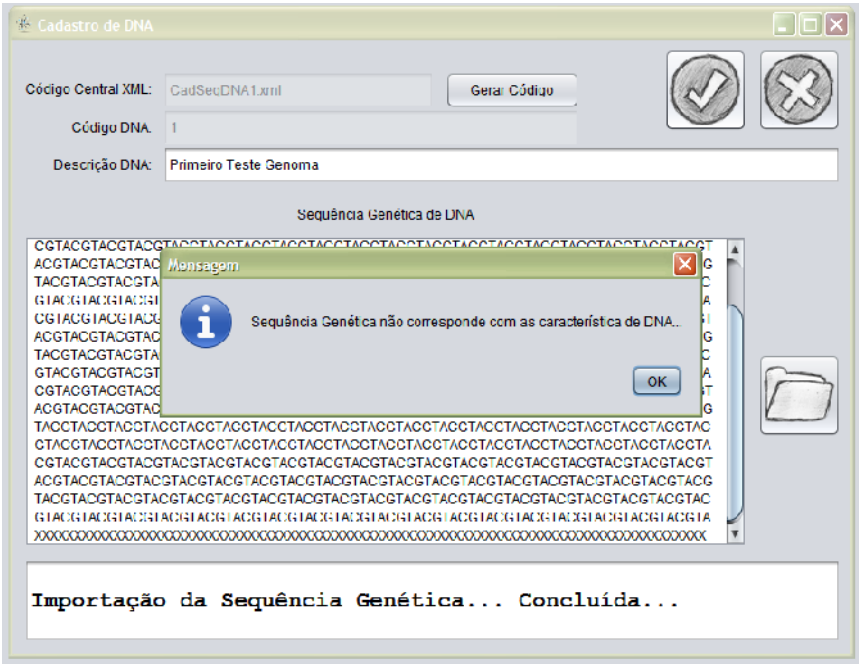

Figura 35 – Validação Não Aceita da Sequência Genética

A figura 36 mostra que a cadeia foi aceita pelo autômato.

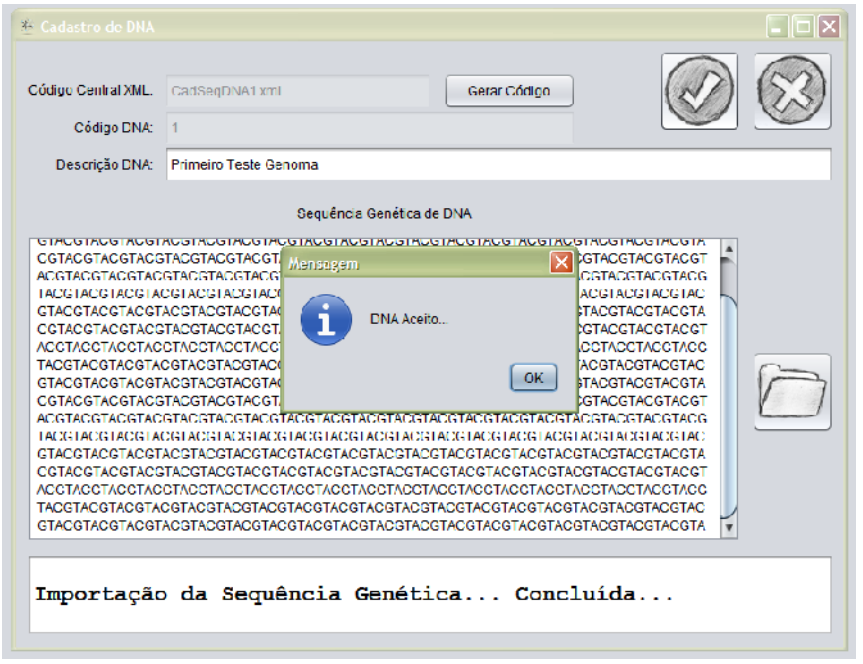

Figura 36 – DNA aceito pelo autômato de reconhecimento.

Após as validações dos processos de importação, a informação genética será armazenada em banco de dados. A figura 37 mostra a interface do banco de dados Berkeley DBXML, com um comando para fazer a abertura do *container* .

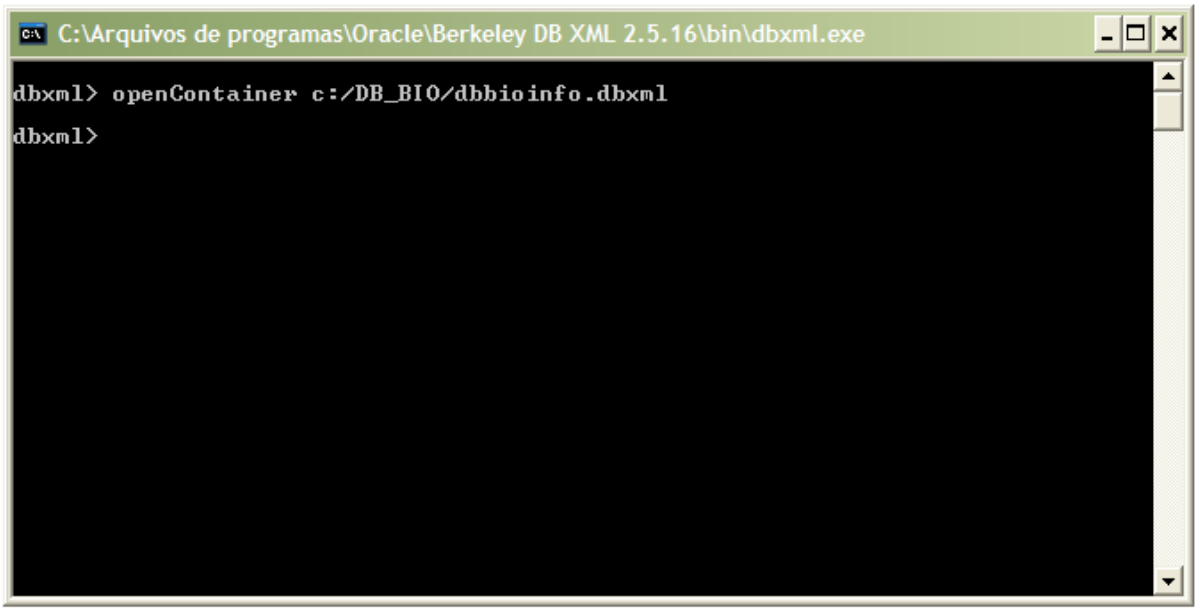

Figura 37 – Abertura de *Container* XML.

Se a validação for aceita, o próximo passo será verificar quantos arquivos XML estão disponível no banco, utilizando o comando:

```
dbxml> getDocument
```

Este comando tem por objetivo fazer uma varredura no banco para encontrar todos os XML armazenados. A figura 38 mostra o número de documentos encontrados no banco.

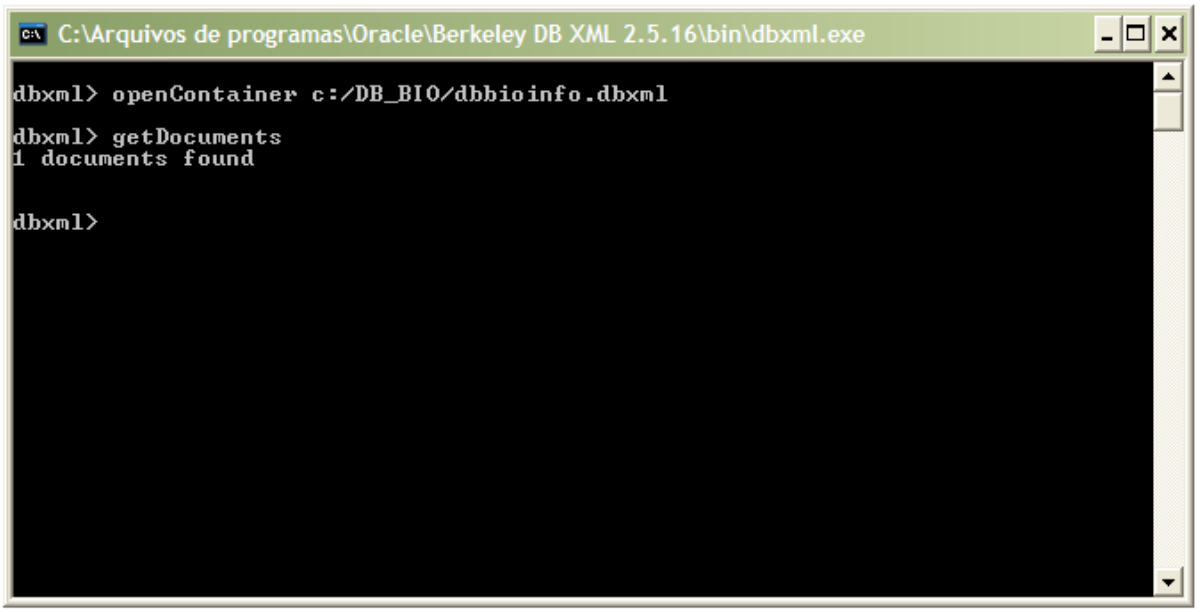

Figura 38 – Verificação do Conteúdo do *Container* XML.

Para visualizar as informações armazenadas no banco é necessário utilizar o comando:

```
dbxml> print
```

Este comando adiciona no processo de cadastramento todas as TAGs definidas. A figura 39 mostra o carregamento da sequência armazenada no banco de dados XML, com todas as padronizações feitas e validadas.

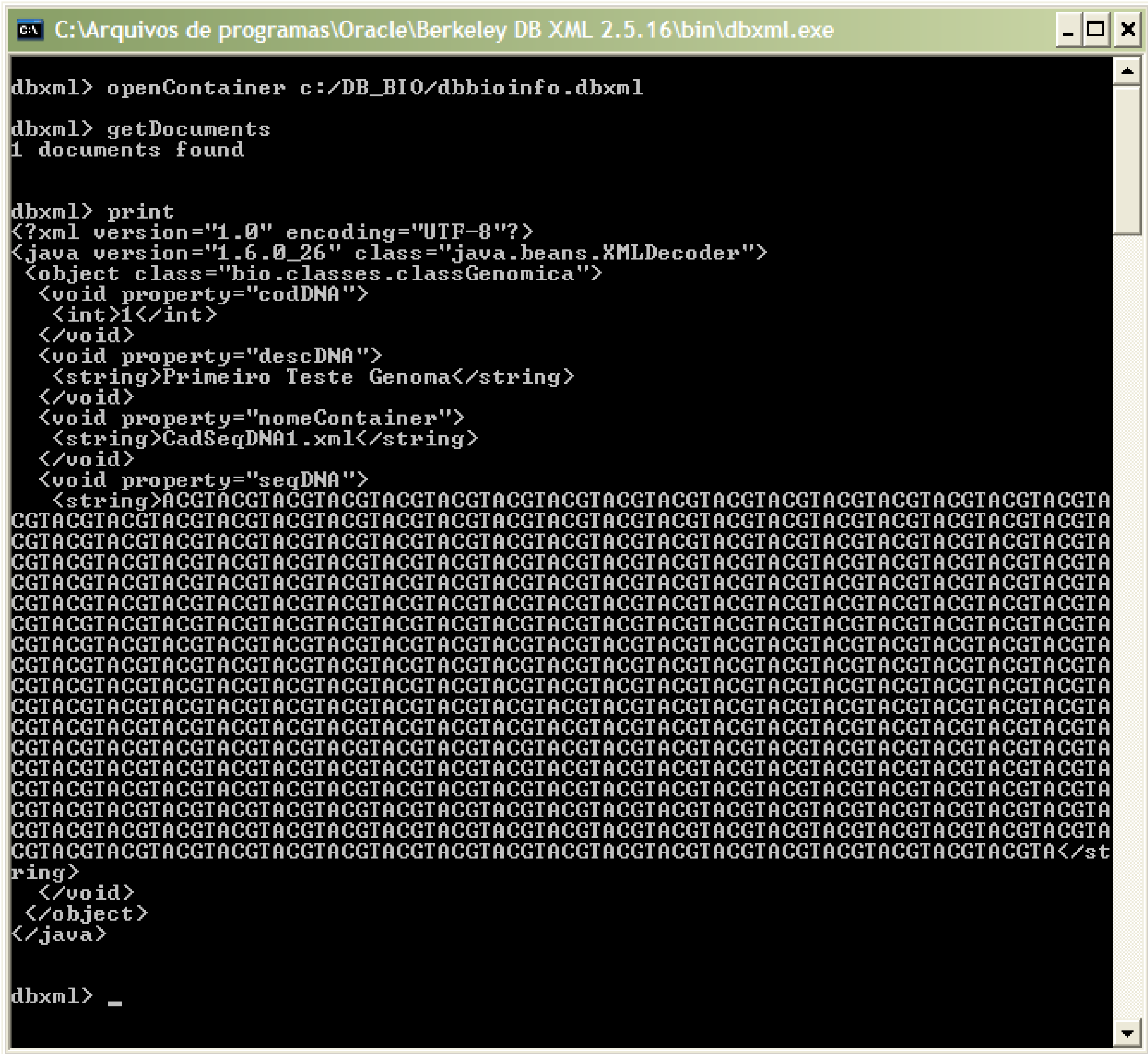

Figura 39 – Execução de Comando para Visualização de conteúdo de um Banco de Dados XML.

3.4.4 – Desenvolvimento do Aplicativo de Busca de Padrões

Nesta seção, será apresentado à implementação do aplicativo de busca de padrão em uma sequência de DNA. O algoritmo utilizado para realizar a busca de padrões em uma sequência de DNA é baseado no do Knuth-Morris-Pratt, pelo fato de ser mais apropriada para estes tipos de problemas.

Para isso é necessário tratar todas as funcionalidades disponibilizadas pela ferramenta Java na construção de uma expressão regular capaz de atender as complexidades que o problema exige.

Será utilizada uma biblioteca chamada *regex* que tem por objetivo realizar as manipulações de expressão regular. Uma expressão regular no formato *regex* (Padrão do Java) representa uma expressão já compilado, sendo já preparada para o uso. No processo de elaboração de uma expressão regular é definitivamente muito custoso, sendo possível ter acesso direto ao construtor da classe padrão. Neste caso, será utilizado o método estático *compile (String regex),* obtendo um pool de expressões já compiladas, reaproveitando o tempo no caso de querer compilar novamente a mesma expressão. As expressões regulares são de dois tipos:

- **Expressão Regular no Padrão Exato**: onde é pesquisado um único padrão em um arquivo de texto;
- **Expressão Regular com Correspondência de Padrões (RE)**: onde são apresentados os padrões múltiplos de uma sequencia, podendo ser manipulado por arquivos de dados genômicos.

A figura 40 mostra como trabalhar expressão regular no ambiente Java. A definição do que vai ser aceito está localizado no *pattern (RE)* definido na primeira linha de código da imagem. No atributo text está sendo atribuído os valores correspondente á sequência genética, para que depois possa ser analisado pela expressão regular.

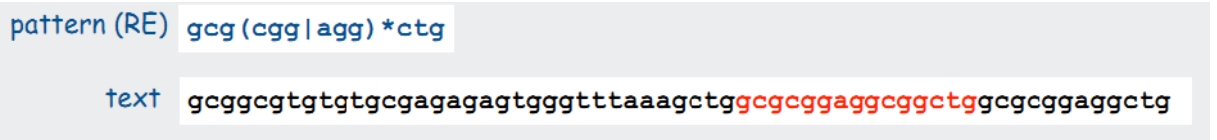

Figura 40 – Representação de uma expressão regular para que possa realizar a busca das Informações (ALGORITHMS, 2011)

Para realizar o teste correspondente a algum padrão de sequência de DNA deve ser definido um marcador genômico onde é feita a identificação das sequências.

A figura 41 mostra uma expressão regular de uma notação para especificar um grupo de conjunto de *strings*.

regular expression	in set	not in set
.*spb.* contains the trigraph spb	raspberry crispbread	subspace subspecies
a* | (a*ba*ba*ba*)* number of b's is a multiple of 3	bbb aaa bbbaababbaa	b bb baabbbaa
.*0.... fifth to last digit is 0	1000234 98701234	111111111 403982772
gcg(cgg|agg)*ctg fragile X syndrome indicator	gcgctg gcgcggctg gcgcggaggctg	gcgcgg cggcggcggctg gcgcaggctg

Figura 41 – Representação das expressões regulares com seus valores de Entradas e os valores que não serão aceitos (ALGORITHMS, 2011)

- **Expressão Regular**: manipulação de determinadas expressões para que possa ser realizada a identificação dos mesmos no ato do processamento.
- **Em conjunto**: coluna destinada para exemplificar as possíveis entradas para que a expressão regular possa processar.
- **Não no Processo**: coluna onde que os valores que não serão aceitas pela expressão definida na coluna expressão regular.

Para a implementação do aplicativo de reconhecimento de padrões foi utilizado o algoritmo de Knuth-Morris-Pratt , onde apresenta duas características básicas:

- Garantias do tempo linear;

- Nenhum *backup* no fluxo de Informações, resultando em busca mais precisa e com maior rendimento.

A figura 42 mostra como que o algoritmo realiza a codificação com o auxílio da biblioteca Java *regex* para busca de padrões.

Figura 42 – Processo de Busca do Padrão Genético pela Expressão Regular (ALGORITHMS, 2011)

O processo de busca de padrões será baseado no reconhecimento de cadeia, porém difere no processo de montagem das instruções. O sistema entrará com informações genômicas e o algoritmo de busca irá reconhecer a cadeia que deseja ser analisada.

A figura 43 mostra o processo de busca nas sequências.

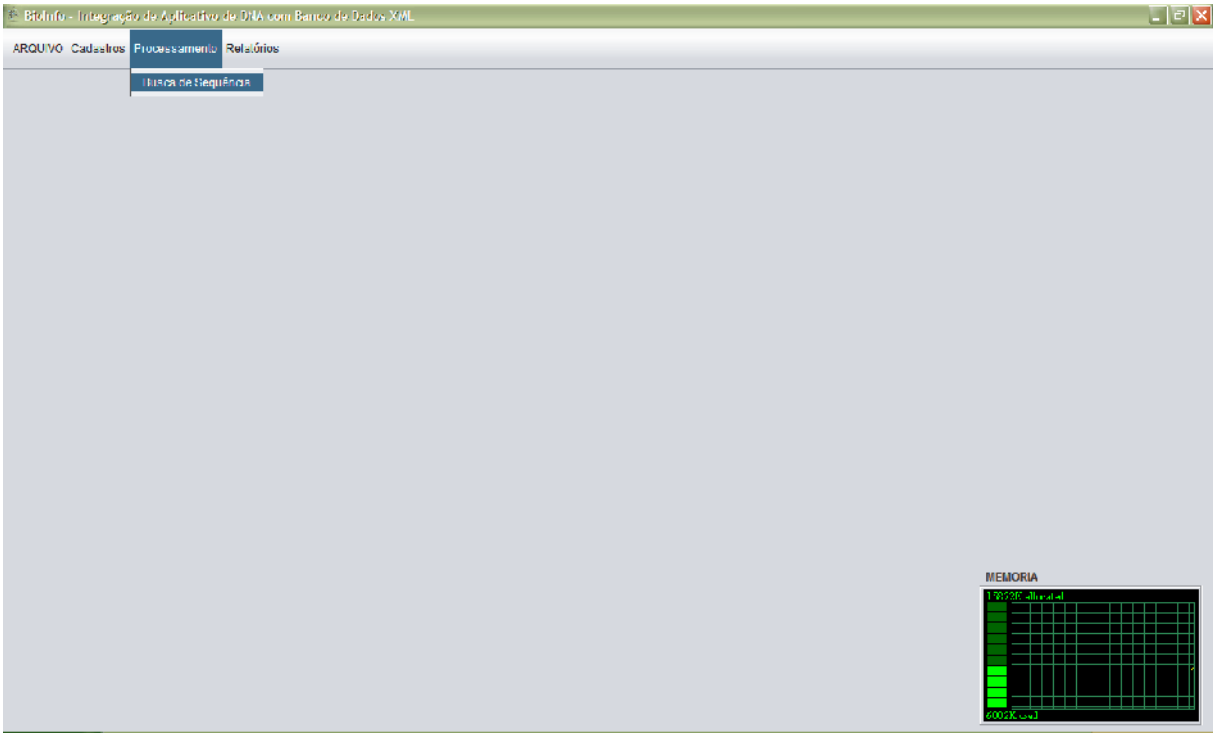

Figura 43 – Interface Para Busca de Padrões.

Para realizar o processo de busca de padrões, selecionar Processamento –> Busca de Sequência. Após este processo será mostrado a interface de comunicação para realizar a busca. A figura 44 mostra a interface para a busca de padrões na sequência de DNA.

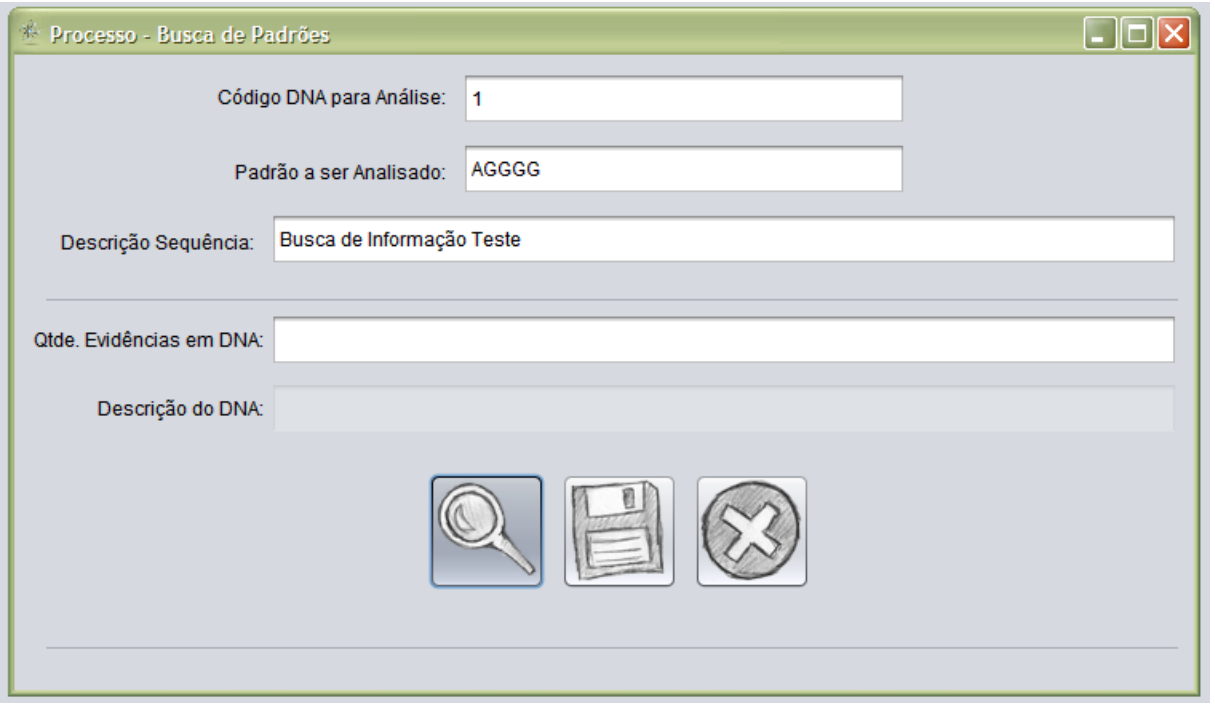

Figura 44 – Interface para Busca de Padrões.

Nesta interface existem dois botões funcionais e um de cancelamento, sendo que o primeiro é o botão responsável para realizar a busca do XML no Banco e analisar o atributo que se refere aos códigos genéticos, e o resultado desse processo será atribuído no campo de Qtde.

A figura 45 mostra o processamento da busca de padrões que poder ser salvo em banco de dados XML.

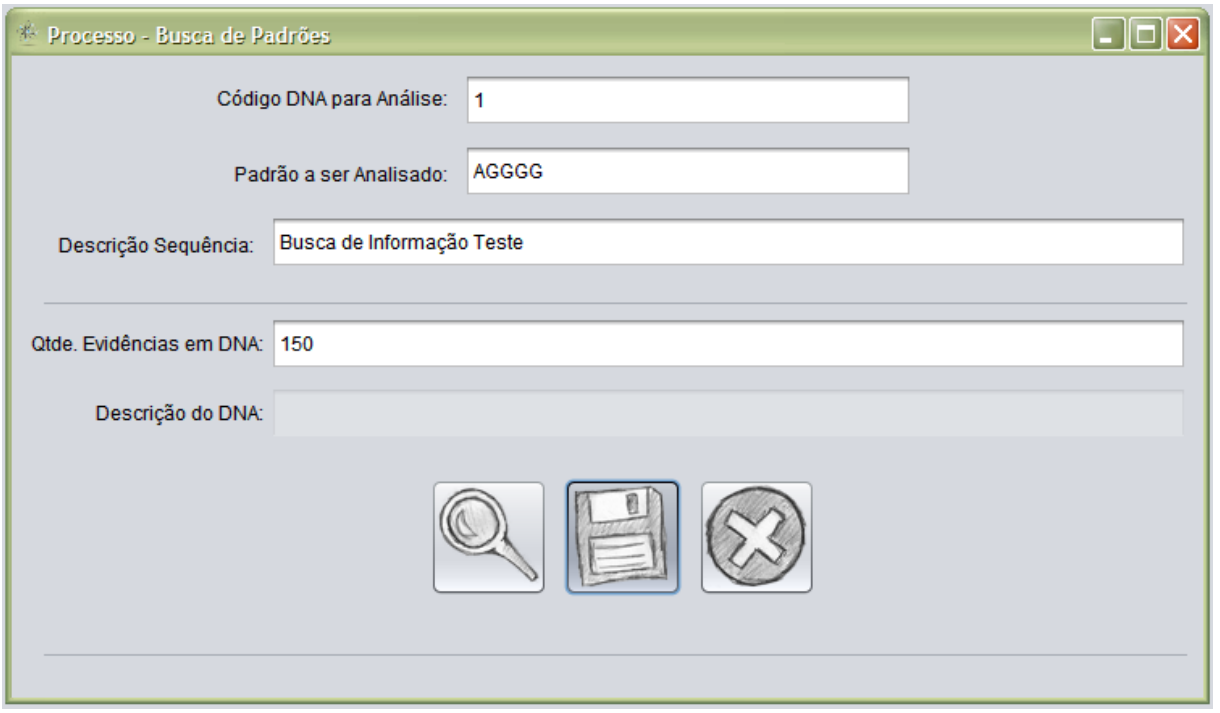

Figura 45 – Interface de Salvamento das Informações Geradas Com o Processo de Busca de Padrões.

3.4.5 – Integração dos Aplicativos com o Banco de Dados

Nesta seção, será feita a integração do aplicativo com o banco XML , onde serão armazenados arquivo de texto, formato .txt. O processo será da seguinte maneira:

- Busca de alguns padrões de sequência de DNA no Banco de Dados Público, ou seja, será feita a importação de arquivo de sequência real para que se possa validar e trabalhar essas informações;
- Após os *downloads* das sequências que se deseja fazer o reconhecimento das cadeias, verificando se a sequência condiz com o sistema. Neste processo será desenvolvido o aplicativo para validação das sequências de DNA através de autômatos;
- Após as validações necessárias, será feita a integração das informações recuperadas de um Banco de Dados Público para o Banco de Dados XML.

Para fazer as inserções das informações já validadas pelo autômato, ela deve ser armazenada no Banco de Dados XML

A figura 46 mostra a integração do aplicativo com o banco de dados.

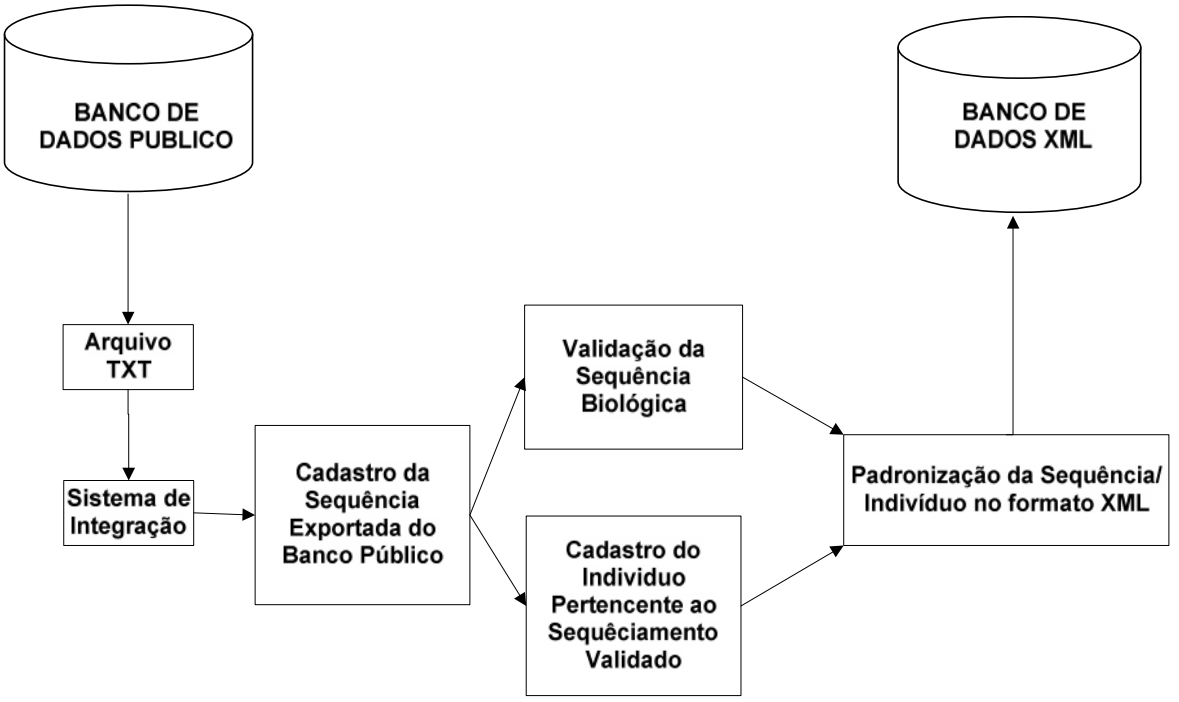

Figura 46 – Integração do Aplicativo com o Banco de Dados

5. CONCLUSÃO

O desenvolvimento deste projeto é de suma importância para o aprimoramento de conhecimentos nesta área, e para projetos futuros devido à demanda de mercado. O projeto é desafiador, devido aos esforços para se tentar adotar um padrão em bancos de dados voltados para a bioinformática. O padrão utilizado é o XML, pois facilita a padronização de informações da sequência de DNA, permitindo autonomia na manipulação de dados. A pouca literatura sobre banco de dados XML para bioinformática dificultou um pouco para modelar o banco de dados e a integração com o aplicativo de reconhecimento e busca de padrões.

Pretende-se dar continuidade neste trabalho de pesquisa e futuramente atuar profissionalmente nesta área. O desenvolvimento deste trabalho foi muito importante, pois trouxe novos conhecimentos.

REFERÊNCIAS BIBLIOGRÁFICAS

BORÉM, A. e SANTOS, F.R., Biotecnologia Simplificada, Editora Suprema. Viçosa, MG, 2001.

DEITEL, H.M. e DEITEL, P.J., **Java, como programar**, trad. Carlos Arthur Lang Lisboa, 4ª ed. , Porto Alegre, Bookman, 2003.

DIVERIO, T.A e MENEZES, P.B., **Teoria da Computação**, Editora Sagra-Luzzatto, 1ª Ed., 1999.

FREUDENRICH, C., **Como Funciona o DNA**, Traduzido por HowStuffWorks Brasil, 2000.

GIBAS, C. e JAMBECK, P., **Desenvolvendo Bioinformática**, Campus Elsevier, 2002.

HOPCROFT, J.E.; ULLMAN, J.D. e MOTWANI, R., **Introdução à Teoria de Autômatos, Linguagens e Computação**, Editora Campus, 2002.

LEMAY, L. e PERKINS, C.L., **Teach yourself – Java in 21 days**, Sams.net Publishing, 1996.

LEWIS, H. R. e PAPADIMITRIOU, C. H. , **Elementos de Teoria da Computação**, BMA Bookman, 2004.

LIFSCHITZ, S., **Algumas Pesquisas em Banco de Dados e Bioinformática**, Anais do XXVI Congresso da SBC, 2006.

MOUNT, D.W., **Bioinformatics: Sequence and Genome Analysis**, Cold Spring Harbor Laboratory Press, 2001.

POTTS, A. e FRIEDEL JR, **D. Java programming language handbook**, CH, Coriolis Group Books, 2004.

PROSDOCINI ET AL., **Bioinformática: Manual do Usuário**, Biotecnologia Ciência e Desenvolvimento, Ano 5, 29, 2002.

RAMALHO, J. C. e HENRIQUES, P., **XML e XSL: da Teoria a Prática**, 1ª Ed., Lisboa, PO, FCA, 2002.

SISPSER, M., **Introdução a Teoria da Computação**, 2ª Ed., Editora Thomson Learning, 2007.

WALMSLEY, P., **Definitive XML Schema**. Prentice Hall PTR, 2001.

WIECZOREK, E.M. e LEAL, E., **Padrões de Tipos e Métodos para Banco de Dados em Bioinformatica**, III Congresso Cientifico do CEULP/ULBRA, 2003.

Robert S., ***Algorithms in C++***, Addison-Wesley 1992, Cap. 19.
W. B. Frakes e R. Baeza-Yates**, Information Retrieval- Data Structures and Algorithms**, Prentice-Hall 1992, Cap. 10.

GRAVES, M. **Projeto de Banco de Dados com XML**. Local de publicação: Pearson Education do Brasil, 2003.

CASTRAVECHI, D. **Estudo das Diferentes Modelagens Empregadas Em Banco de Dados Genômicos**, Londrina, 2004.

GALVÃO, L. R. **Definição de um Modelo Conceitual para Representação de Esquema XML**, Pernambuco, 2003.

COSTA, V. R. **Genoma Decifrado, Trabalho Dobrado**, Ciência Hoje. Vol. 28, nº 166, novembro, 2000.

FÉLIX, J. M. **Genoma Funcional**, Biotecnologia, Ciência & Desenvolvimento, Nº 24, 2002.

TEIXEIRA, Marcus Vinícius Carneiro. **Gerenciamento de Anotações de Biossequências utilizando Associação entre ontologias e esquemas XML**, São Carlos, 2008.

NASCIMENTO, Aldo Monteiro. **Um Modelo para Integração de Documentos XML em nível de Instância**, Curitiba, 2008.

GIBSON, William. **Reconhecimentos de Padrões**, trad. Fabio Fernandes, Editora Aleph, 2008.

JAIN, Anil; BOLLE, Ruud; PANKANTI, Sharath. **Biometrics Personal Identification in Networked Society.** 2002. 422p. Kluwer Academic Publishers, 2002.

CRITCHLOW, T.; MUSICK, R. e SLEZAK, T. **An Overview of Bioinformatics Research at Lawrence Livermore National Laboratory,** Department of Energy by University of California Lawrence Livermore National Laboratory, Califórnia U.S., 2000.

WIECZOREK, E.M. e LEAL, E., **Caminhos e Tendencias do Uso de Banco de Dados em Bioinfomática**, Centro Universitario Luterano de Palmas, 2002.

SHUI, W. M. **Utilizing Multiple Bioinformatic Information Sources: An XML Database pproach 2001 Bioinformatics Honours Thesis**. Sydney: University of New South Wales, 2001.

ORACLE CORP. **Oracle8i Data Cartridge Developer's Guide: Release 8.1.5 (Part No. A68002-01)**, Oracle Corporation, Redwood Shores, 1999.

FELLOW, A. J.; DUIN, R. P. W.; MAO, J. **Statistical Pattern Recognition: A Review**. 2000. 34p. IEEE Transactions on Pattern Analysis and machine Intelligence, Vol.22, 2000.

HARADA, M. M. **Casamento Apropriado de Padrões**, Campinas, 1994.

ALMEIDA, R. **Introdução ao Oracle Berkeley DB XML**, 2011. Disponível em http://imasters.com.br/artigo/20078/banco-de-dados/introducao-ao-oracle-berkeley-db-xml. Acesso em julho de 2011.

PROSDOCIMI, F. **Indrodução a Bioinformática,** 2001.

GREHAN, R., **Berkeley DB Adds XML Smarts**, 2004. Disponível em http://www.infoworld.com/d/data-management/berkeley-db-adds-xml-smarts-077.

GEOFF, L. **Oracle Database 10g Relase 2 XML DB Technical Overview**, Oracle Corporation, 2005.

FALCÃO, N. A. **Armazenamento de Mídias e Objetos Virtuais Utilizando Suporte Nativo a XML do Oracle 10g**, Trabalho de Graduação, Projeto AMADeUs-MM, Universidade federal do Pernambuco, 2006.

VEIGA, D.F. e PORTO, L.M. **Modelos XML Schema Para a Representação da Informação Genômica**, Universidade Federal se Santa Catarina, Publicações, 2003.

SEDGEWICK, R e WAYNE, **ALGORITHMS 4Th Edition**, 2011. Disponível em http://algs4.cs.princeton.edu/home/. Acesso em Setembro de 2011.